Ministère de l'enseignement Supérieur et de la recherche Scientifique
وزارة التعليم العالي والبحث العلمي

Badji Mokhtar Annaba University
Université Badji Mokhtar – Annaba

جامعة باجي مختار – عنابـــــــــة

Faculté de Technologie

كلية التكنولوجيا

Département d'Informatique

قســـــــم الاعلام الالي

Thèse

Présentée pour obtenir le diplôme de

Doctorat

Spécialité : Systemes Informatique

Filière : Informatique

Par :

Yasmine Chachoui

Thème :

Guidage de l'Apprenant dans un EIAH : Optimisation des processus d'évaluation automatique de la programmation par l'intelligence artificielle

Thèse soutenue le 15/10/2025 devant le jury composé de :

N°	Nom et prénom	Grade	Etablissement	Qualité
01	Farah Nadir	Prof.	Université Badji Mokhtar -Annaba	Président
02	Bensebaa Tahar	Prof.	Université Badji Mokhtar -Annaba	Directeur de thèse
03	Hotte Richard	Prof.	Université TÉLUQ	Co-Directeur de thèse
04	Merouani Hayet Farida	Prof.	Université Badji Mokhtar -Annaba	Examinatrice
05	Zemmal Nawal	MCA	Université Moahmed Chérif Messaadia Souk-Ahras	Examinatrice
06	Farou Brahim	Prof.	Université 8 Mai 1945 Guelma	Examinateur
07	Azizi Nabiha	Prof.	Université Badji Mokhtar -Annaba	Invitée

الملخص

تندرج هذه الأطروحة ضمن مجال البيئات المعلوماتية للتعلم البشري (EIAH) وتركز على تحسين تعلم الخوارزميات في التعليم العالي. لا يزال معدل الفشل المرتفع للطلاب في دورات البرمجة تحديًا رئيسيًا، ويرجع ذلك إلى عوامل متعددة مثل عدم التوازن في المهارات السابقة، وعدم كفاية الأساليب التعليمية التقليدية، وعدم التوازن في مجموعات البيانات التعليمية، ونقص الوسائل الدقيقة لتقييم فهم المفاهيم المُدرّسة.

ولمعالجة هذه التحديات، تقدم هذه الأطروحة مساهمتين رئيسيتين. تتناول المساهمة الأولى مشكلة عدم التوازن في الفئات داخل البيانات التعليمية، وهي مشكلة تعقّد تطوير نماذج تنبؤية موثوقة. تم إجراء الدراسة على عينة مكونة من 2176 طالبًا في السنة الأولى من البرمجة، حيث بلغ معدل الفشل 76%. وللتخفيف من هذا التفاوت، تم تطبيق تقنيات الإفراط في أخذ العينات مثل SMOTE وثلاثة من نسخه. ثم تم تنفيذ طريقة تُسمى Equi-Fused-Data-based SMOTE لدمج مجموعات البيانات المُعاد أخذ عيناتها بطريقة غير متكررة. كانت النتائج واعدة، حيث وصلت الدقة إلى 93.85%، و-F1 score إلى 92.86%، وAUC إلى 98.08%، مما يدل على فعالية هذا النهج في تحسين أداء النماذج التنبؤية بشكل ملحوظ.

تركز المساهمة الثانية على تقييم فهم مفاهيم البرمجة في إطار تصنيف متعدد الملصقات. تتفوق النماذج المسبقة التدريب المحددة، مثل CodeBERT وUniXcoder، على النماذج العامة مثل BERT وRoBERTa من حيث الدقة، و-F1 score، وترميز هامنج في مهام التصنيف متعددة الملصقات المتعلقة بالبرمجة. من بين هذه النماذج، يبرز UniXcoder كأفضل نموذج، متفوقًا حتى على CodeBERT الذي يظهر أداءً مرضيًا. تقترح هذه الأطروحة نهجًا للدمج المرجح لتحسين دقة التقييمات. تظهر النتائج أن نهج الدمج المرجح يسمح للنماذج بمعالجة أنواع مختلفة من الأخطاء، مما يؤدي إلى تحسين شامل في الأداء، مع تفوق UniXcoder.

تُظهر النتائج التجريبية مدى أهمية الأساليب المستخدمة وتفتح آفاقًا لأبحاث مستقبلية تهدف إلى تعميق دمج الذكاء الاصطناعي في البيئات المعلوماتية للتعلم البشري (EIAH).

كلمات مفتاحية: البيئات المعلوماتية للتعلم البشري، التقييم الآلي، عدم التوازن في الفئات، الإفراط في أخذ العينات، النماذج اللغوية المسبقة التدريب، الذكاء الاصطناعي في التعليم.

Résumé

Cette thèse s'inscrit dans le domaine des Environnements Informatiques pour l'Apprentissage Humain et se concentre sur l'amélioration de l'apprentissage de l'algorithmique dans l'enseignement supérieur. Le taux élevé d'échec des étudiants dans les cours de programmation reste un défi majeur, attribuable à un ensemble de facteurs tels que le déséquilibre des compétences préalables, l'insuffisance des méthodes pédagogiques traditionnelles, le déséquilibre des ensembles de données éducatives, et le manque de moyens pour évaluer de manière nuancée la compréhension des concepts enseignés.

Pour répondre à ces défis, cette thèse propose deux contributions principales. La première aborde le problème du déséquilibre des classes dans les données éducatives, une problématique qui complique la formation de modèles prédictifs fiables. L'étude a été menée sur un échantillon de 2176 étudiants de première année en programmation, présentant un taux d'échec de 76%. Afin d'atténuer ce déséquilibre, des techniques de suréchantillonnage, telles que SMOTE et trois de ses variantes, ont été appliquées. Une méthode nommée Equi-Fused-Data-based SMOTE a ensuite été implémentée pour combiner les jeux de données suréchantillonnés de manière non répétitive. Les résultats obtenus sont prometteurs, avec une précision de 93,85%, un F1-score de 92,86%, et un AUC de 98,08%, démontrant ainsi l'efficacité de cette approche pour améliorer significativement la performance des modèles prédictifs.

La seconde contribution concerne l'évaluation de la compréhension des concepts de programmation dans le cadre d'une classification multi-label. Les modèles spécifiques pré-entraînés, tels que CodeBERT et UniXcoder, surpassent les modèles généraux comme BERT et RoBERTa en termes de précision, score F1 et la perte de Hamming pour les tâches de classification multi-label liées à la programmation. Parmi ces modèles, UniXcoder se distingue comme le meilleur, surpassant même CodeBERT, qui affiche pourtant des performances satisfaisantes. Cette thèse propose une approche de fusion pondérée pour améliorer encore la précision des évaluations. Les résultats montrent que l'approche de fusion pondérée permet aux modèles de traiter différents types d'erreurs, ce qui conduit à une amélioration globale des performances, avec UniXcoder en tête.

Les résultats expérimentaux démontrent la pertinence des méthodes employées et ouvrent des perspectives pour des recherches futures visant à approfondir l'intégration de l'intelligence artificielle dans les EIAH.

Mots clés : EIAH, Evaluation automatique, Déséquilibre des classes, Suréchantillonnage, Modèles de langage pré-entrainés, Intelligence artificielle en éducation.

Abstract

This thesis falls within the field of Computer-Based Environments for Human Learning (EIAH) and focuses on improving algorithmic learning in higher education. The high failure rate of students in programming courses remains a major challenge, attributable to various factors such as the imbalance of prior skills, the insufficiency of traditional teaching methods, the imbalance in educational datasets, and the lack of nuanced means to assess the understanding of taught concepts.

To address these challenges, this thesis proposes two main contributions. The first tackles the issue of class imbalance in educational data, a problem that complicates the development of reliable predictive models. The study was conducted on a sample of 2,176 first-year programming students, with a failure rate of 76%. To mitigate this imbalance, oversampling techniques such as SMOTE and three of its variants were applied. A method named Equi-Fused-Data-based SMOTE was then implemented to combine the oversampled datasets in a non-repetitive manner. The results are promising, with an accuracy of 93.85%, an F1-score of 92.86%, and an AUC of 98.08%, demonstrating the effectiveness of this approach in significantly improving the performance of predictive models.

The second contribution focuses on the evaluation of programming concept comprehension within the framework of multi-label classification. Specific pre-trained models, such as CodeBERT and UniXcoder, outperform general models like BERT and RoBERTa in terms of accuracy, F1-score, and Hamming loss for multi-label classification tasks related to programming. Among these models, UniXcoder stands out as the best, even surpassing CodeBERT, which still shows satisfactory performance. This thesis proposes a weighted fusion approach to further enhance the accuracy of assessments. The results show that the weighted fusion approach allows the models to handle different types of errors, leading to an overall improvement in performance, with UniXcoder leading the way.

The experimental results demonstrate the relevance of the methods employed and open up avenues for future research aimed at deepening the integration of artificial intelligence into EIAH.

Key words: EIAH, Automatic assessment, Class imbalance, Oversampling, Pre-trained language models, Artificial intelligence in education.

Remerciements

Je tiens tout d'abord à remercier chaleureusement le Professeur Tahar Benseeba, qui a été mon encadrant tout au long de cette aventure.

Je suis également profondément reconnaissante envers le Professeur Nabiha Azizi. Son assistance et le temps précieux qu'elle a consacré à m'accompagner m'ont été d'une aide inestimable. Sa générosité d'esprit fait d'elle une personne d'une rare qualité, dont l'influence a profondément marqué mon parcours.

Mes remerciements les plus empreints de reconnaissance vont également au Professeur Richard Hotte. Sa bienveillance, les opportunités exceptionnelles qu'il m'a offertes, et son engagement ont été d'une valeur inestimable.

Je tiens à exprimer toute ma reconnaissance au Président du jury, qui a généreusement accepté de présider cette soutenance. Je remercie également les examinateurs pour l'honneur qu'ils m'ont fait en acceptant de participer au jury. Leur expertise a enrichi ce travail, et je leur suis profondément reconnaissante pour leur implication.

Je ne saurais omettre de mentionner l'ensemble des membres de l'équipe du centre de recherche LICEF à l'Université TELUQ de Montréal. Travailler à leurs côtés a été une expérience des plus enrichissantes, et leur contribution a été essentielle à la réussite de ce projet.

Je suis infiniment reconnaissante envers mes parents, dont le soutien indéfectible et les encouragements constants ont été le pilier de ma réussite. Leur présence rassurante, tant dans les moments de doute que dans ceux de succès, m’a insufflé la force de persévérer avec confiance et détermination, malgré les épreuves. Leurs innombrables sacrifices, leur générosité incommensurable, ainsi que leur foi inébranlable en moi, ont été pour moi des sources d'inspiration inestimables. Sans eux, ce projet n'aurait jamais vu le jour.

Enfin, je souhaite également remercier mes frères Lyes et Yanis. Lyes ne m’a jamais refusé son aide, m'accompagnant avec une bienveillance et une disponibilité sans faille tout au long de ce chemin. Mounia, Yanis, ainsi que son épouse Rym, m'ont accompagnée avec une bienveillance et une générosité infinie.

Table des Matières

Liste des Figures

Liste des Tableaux

Liste des Abréviations

ADASYN	Adaptative Synthetic Sampling (Échantillonnage Synthétique Adaptatif)
AUC	Area Under the Curve (Surface sous la Courbe)
BBN	Bayesian Belief Network (Réseau de Croyance Bayésien)
BERT	Bidirectional Encoder Representations from Transformers (Représentations Bidirectionnelles d'Encoders à partir de Transformers)
Bi-LSTM	Réseaux de Neurones Récurrents Bidirectionnels à Longue Mémoire à Court Terme (Bidirectional Long Short-Term Memory Recurrent Neural Networks)
BR	Binary Relevance (Pertinence Binaire)
CC	Chain Classifier (Chaîne de Classificateurs)
DNN	Deep Neural Network (Réseaux de Neurones Profonds)
EDM	Educational Data Mining (Fouille de Données Éducatives)
EIAH	Environnements Informatiques pour l'Apprentissage Humain (Computer-based Environments for Human Learning)
GBM	Gradient Boosting Machines (Machines de Gradient Boosting)
GPT	General-Purpose Technology (Technologie à Usage Général)
IA	Intelligence Artificielle (Artificial Intelligence)
IHM	Interface Homme Machine (Human-Computer Interaction)
IQR	Intervalle Interquartile (Interquartile Range)
kNN	k-Nearest Neighbor (k-Plus Proches Voisins)
LLM	Grand Modèles de Langage (Large Language Models)
LP	Label Powerset (Ensemble de Labels)
MCC	Coefficient de Corrélation de Matthews (Matthews Correlation Coefficient)
MLM	Masked Language Modeling (Modélisation de Langage Masquée)
MLP	Multi Layer Perceptron (Perceptron Multi-Couche)
MOOC	Massive Open Online Courses (Cours en Ligne Ouverts et Massifs)
NLP	Natural Language Processing (Traitement du Langage Naturel)
NSP	Next Sentence Prediction (Prédiction de la Phrase Suivante)
PLM	Pre-trained Language Models (Modèles de Langage Pré-entraînés)
QCM	Question à Choix Multiples (Multiple Choice Question)
ROBERTA	Robustly Optimized BERT Approach (Approche BERT Robuste et Optimisée)
ROC	Receiver Operating Characteristic (Caractéristique de Fonctionnement du Récepteur)
ROS	Random Over-Sampling (Suréchantillonnage Aléatoire)
RUS	Random Under-Sampling (Sous-échantillonnage Aléatoire)
SMOTE	Synthetic Minority Over-sampling Technique (Technique de Suréchantillonnage Synthétique des Minorités)
SMOTE-ENN	Edited Nearest Neighbors (Voisins les Plus Proches Édités)
SMOTE-NC	SMOTE for Nominal and Continuous data (SMOTE pour Données Nominales et Continues)
STEM	Science, Technology, Engineering, and Mathematics (Sciences, Technologies, Ingénierie et Mathématiques)
SVM	Support Vector Machines (Machines à Vecteurs de Support)
TIC	Technologie de l'Information et de la Communication (Information and Communication Technology)
UML	Unified Modeling Language (Langage de Modélisation Unifié

Introduction Générale

L'échec académique des étudiants dans les cours d'algorithmique demeure une problématique majeure dans l'enseignement supérieur. Cet échec persistant soulève de nombreuses questions quant à la capacité des méthodes pédagogiques traditionnelles à répondre aux besoins diversifiés des apprenants. En effet, le décalage entre les attentes académiques et les compétences initiales des étudiants est souvent pointé du doigt comme l'une des causes principales des difficultés rencontrées. Les étudiants sont confrontés à des exigences qu'ils peinent à satisfaire, faute d'un accompagnement pédagogique adapté. Cela conduit à un taux élevé d'échecs, révélateur des failles des approches actuelles.

Les environnements informatiques pour l'apprentissage humain (EIAH) se présentent comme une réponse potentielle à ce défi. Ces environnements, en intégrant des stratégies pédagogiques innovantes et des outils technologiques avancés, offrent une nouvelle perspective sur l'apprentissage. Toutefois, l'efficacité des EIAH dépend largement de leur capacité à s'adapter aux spécificités de l'enseignement de la programmation. La question de l'évaluation, en particulier, reste un enjeu crucial. Une évaluation mal conçue peut engendrer un impact négatif, renforçant les difficultés des étudiants plutôt que de les atténuer. Il est donc essentiel de repenser les méthodes d'évaluation pour qu'elles reflètent véritablement le niveau de compétence des apprenants et non pas uniquement leur capacité à réussir un examen ponctuel.

Cette thèse s'inscrit dans ce contexte de remise en question des approches éducatives traditionnelles. Elle propose d'explorer des solutions fondées sur l'intelligence artificielle (IA) pour améliorer la réussite des étudiants en programmation. Plus précisément, deux contributions majeures sont développées dans ce travail. La première concerne le problème du déséquilibre des classes dans les données éducatives, un problème qui complique la création de modèles prédictifs fiables. Les étudiants en réussite sont souvent sous-représentés, ce qui biaise les prédictions et limite la capacité d'identifier en amont les apprenants en difficulté. Pour remédier à cette situation, des techniques avancées de rééquilibrage des données, telles que le SMOTE et ses variantes, ainsi que la méthode proposée nommée Equi-Fused-Data-based SMOTE, sont explorées pour améliorer la précision des prédictions.

La seconde contribution se concentre sur l'évaluation automatique de la compréhension des concepts de programmation. L'évaluation est un processus complexe qui ne se limite pas à la vérification de la syntaxe des codes soumis par les étudiants. Comprendre un énoncé algorithmique et être capable de traduire cette compréhension en un programme fonctionnel exige une maîtrise des concepts profonds du domaine. C'est pourquoi, des modèles de langage pré-entraînés, comme BERT, RoBERTa, CodeBERT, et UniXcoder, sont utilisés pour évaluer de manière fine et précise la compréhension des étudiants. Ces modèles ont permis de mesurer dans quelle mesure les étudiants ont compris les concepts en comparant leurs réponses avec les solutions jugées correctes par l'expert, en s'intéressant aux nuances logiques et sémantiques du code.

En intégrant l'IA dans les EIAH, cette recherche ambitionne non seulement d'affiner l'évaluation des compétences en programmation, mais également de proposer un accompagnement pédagogique plus personnalisé, en phase avec les besoins spécifiques de chaque apprenant. L'objectif fondamental est de réduire les taux d'échec tout en facilitant une

appropriation approfondie des concepts algorithmiques par les étudiants. En optimisant la précision des modèles prédictifs et en développant des outils d'évaluation plus nuancés, cette thèse aspire à offrir aux apprenants un cadre d'apprentissage sur mesure et plus performant, où chacun peut bénéficier d'un soutien adapté à ses lacunes et potentiels spécifiques.

Problématique et Justification de la Recherche

L'enseignement de l'algorithmique, constitue un enjeu central dans les cursus de sciences et technologies. Toutefois, les taux d'échec élevés dans ce domaine restent un problème persistant. Ces échecs sont le résultat d'une combinaison complexe de facteurs pédagogiques, cognitifs et technologiques qui nécessitent une attention particulière.

Tout d'abord, il est essentiel de reconnaître que l'algorithmique, cœur de l'apprentissage de la programmation, pose des défis spécifiques. L'acquisition des compétences en algorithmique demande non seulement une compréhension théorique approfondie des concepts, mais aussi une capacité à les appliquer dans des situations pratiques complexes. Beaucoup d'étudiants rencontrent des difficultés à franchir cette étape, ce qui les mène à l'échec, notamment dans les premières années de leur parcours universitaire. Ces difficultés sont amplifiées par un déséquilibre entre les compétences préalables des étudiants et les exigences académiques. Les approches traditionnelles peinent à répondre aux besoins variés des apprenants.

Un autre facteur majeur qui exacerbe ce problème est le déséquilibre des classes dans les données éducatives. Ce déséquilibre se manifeste par une surreprésentation des étudiants en échec par rapport à ceux qui réussissent (ou l'inverse), complique la tâche des modèles prédictifs. Par conséquent, les résultats des modèles sont biaisés, les rendant moins aptes à identifier les étudiants à risque ou ceux qui pourraient bénéficier d'un accompagnement spécifique. La gestion de ce déséquilibre est donc nécessaire pour améliorer la qualité des prédictions et, par conséquent, pour mettre en place des stratégies pédagogiques nettement plus ciblées.

Dans ce contexte, les EIAH se présentent comme une solution potentielle pour atténuer ces difficultés. Ces environnements permettent de concevoir des systèmes d'apprentissage adaptatifs, capables de s'ajuster aux besoins spécifiques de chaque apprenant. Cependant, malgré leur potentiel, les méthodes traditionnelles d'évaluation qui y sont intégrées se révèlent souvent insuffisantes pour détecter et diagnostiquer les difficultés spécifiques des apprenants, surtout lorsque ces difficultés sont liées à la compréhension des concepts algorithmiques fondamentaux. Par ailleurs, l'évaluation de la compréhension en programmation ne peut se limiter à la correction d'exercices ou à la vérification de la syntaxe des codes soumis par les étudiants. Comprendre un langage de programmation, c'est avant tout saisir sa logique interne, maîtriser la pensée algorithmique, et être capable de transposer cette compréhension dans des solutions pratiques. Or, les méthodes d'évaluation traditionnelles ne permettent pas de capturer cette dimension profonde de la compréhension. Elles sont souvent axées sur des critères superficiels et ne parviennent pas à rendre compte de la complexité des processus cognitifs sous-jacents à l'apprentissage de la programmation.

Pour pallier ce manque, l'intégration de l'IA dans les EIAH apparaît comme une avenue prometteuse. Les modèles de langage pré-entraînés (PLM), tels que BERT, RoBERTa, CodeBERT, et UniXcoder, offrent des outils puissants pour analyser finement la compréhension des concepts algorithmiques. Ces modèles peuvent être ajustés pour permettre une évaluation de la similarité entre les solutions de code des étudiants et celles jugées correctes par l'expert, en tenant compte non seulement de la syntaxe, mais aussi de la sémantique et de la structure logique des programmes. Cependant, malgré leur potentiel, ces outils doivent être

intégrés de manière cohérente dans des environnements d'apprentissage qui prennent en compte la diversité des profils étudiants et la complexité des compétences à acquérir.

Ainsi, la problématique centrale de cette recherche repose sur la nécessité de repenser les approches d'évaluation dans l'enseignement de l'algorithmique dans le cadre des EIAH. Il s'agit de développer des solutions qui, d'une part, corrigent les biais liés au déséquilibre des classes et, d'autre part, offrent des moyens plus adaptés pour évaluer la compréhension des apprenants. Cette thèse justifie son importance par l'urgence de répondre à ces défis, en proposant des solutions concrètes basées sur l'IA et les EIAH pour améliorer les processus d'apprentissage et réduire les taux d'échec. L'objectif est de créer un cadre d'apprentissage où chaque apprenant, quel que soit son niveau de départ, puisse bénéficier d'un soutien adapté à ses besoins, et où l'évaluation de la compréhension devienne un outil puissant pour guider les parcours éducatifs.

Objectifs de la Recherche

L'objectif principal de cette thèse est de concevoir des solutions visant à réduire le taux d'échec en programmation en s'appuyant sur des techniques d'intelligence artificielle et des méthodes de guidage de l'apprenant dans un EIAH. Deux axes de recherche principaux sont explorés :

1. Équilibrage des classes et modélisation prédictive: un des défis dans l'analyse des performances académiques réside dans le déséquilibre des classes. Pour y remédier, cette thèse explore l'utilisation de diverses techniques de suréchantillonnage, notamment SMOTE et ses variantes, ainsi qu'une méthode nommée Equi-Fused-Data-based SMOTE, afin d'améliorer la performance des modèles de classification prédictive.
2. Évaluation automatique de la compréhension des concepts de programmation via l'IA: en explorant l'utilisation de modèles de langage pré-entraînés comme BERT, RoBERTa, CodeBERT, et UniXcoder pour évaluer la compréhension des étudiants en programmation. L'objectif est de quantifier la similarité entre les solutions de code soumises par les étudiants et les solutions correctes, et ainsi mesurer la maîtrise des concepts fondamentaux de programmation tels que les déclarations de variables, les opérations arithmétiques, et les instructions conditionnelles. Une nouvelle technique de fusion pondérée est également introduite pour améliorer la précision de l'évaluation.

Méthodologie

La méthodologie adoptée dans cette recherche s'articule autour de plusieurs axes complémentaires, intégrant des approches de collecte de données, d'analyse approfondie, de modélisation prédictive, et d'évaluation automatique des performances des étudiants en programmation.

La première étape de cette recherche a consisté en la collecte des données auprès d'un large échantillon d'étudiants inscrits en première année dans des filières de sciences et technologies. L'échantillon, composé de 2176 étudiants, a permis d'obtenir une base de données représentative, riche en informations sur les performances académiques et les compétences en programmation des apprenants. Ensuite, pour garantir la qualité des données, des techniques de nettoyage et de prétraitement ont été mises en œuvre. L'analyse exploratoire des données a ensuite été réalisée pour identifier les tendances générales et les disparités au sein de l'échantillon. Cette étape a permis de mettre en lumière les facteurs qui contribuent aux échecs des étudiants, en particulier les difficultés liées à la compréhension des concepts de

programmation. Ces analyses ont également révélé le déséquilibre des classes dans les données, un problème qui a guidé les choix méthodologiques ultérieurs en matière de modélisation prédictive.

Pour répondre aux défis posés par le déséquilibre des classes, la recherche a adopté des techniques avancées de modélisation prédictive, intégrant des méthodes d'échantillonnage et d'équilibrage des classes. De plus, une méthode innovante, nommée Equi-Fused-Data-based SMOTE, a été développée dans le cadre de cette recherche pour améliorer l'équilibre des données et renforcer la précision des modèles de classification. Par ailleurs, les modèles prédictifs utilisés ont été soigneusement sélectionnés et ajustés pour maximiser leur capacité à identifier les étudiants à risque, tout en minimisant les erreurs de classification. Les performances des modèles ont été évaluées à l'aide de métriques, telles que la précision, le rappel, le F1-score, et l'AUC-ROC (Area Under the Receiver Operating Characteristic Curve).

L'évaluation de la compréhension des concepts de programmation a constitué un axe central de cette recherche. Pour cela, des modèles de langage pré-entrainés ont été employés. Ces modèles ont été adaptés pour analyser la similarité entre les solutions de code soumises par les étudiants et les solutions jugées correctes par l'expert. Cette évaluation automatisée a permis de dépasser les limitations des approches traditionnelles, en offrant une analyse plus fine des compétences algorithmiques des étudiants.

L'une des avancées les plus significatives de cette méthodologie réside dans l'utilisation d'une technique de fusion pondérée, qui combine les forces de plusieurs modèles pour améliorer la précision des évaluations. Cette approche permet de compenser les faiblesses des modèles individuels, en exploitant leurs complémentarités pour fournir une évaluation plus nuancée de la compréhension des concepts par les étudiants. La classification multi-étiquettes a également été appliquée pour prendre en compte la diversité des compétences évaluées, offrant ainsi une vision plus complète des performances des étudiants. Tout au long du processus, une attention particulière a été portée à la fiabilité des modèles et des résultats obtenus. Les modèles ont été testés sur plusieurs jeux de données, incluant des données de validation et de test, afin de s'assurer de leur performance dans des contextes variés.

Contributions de la Recherche

Cette thèse apporte deux contributions majeures au domaine des EIAH :

1) Rééquilibrage des classes dans les données éducatives

La première contribution de cette recherche relève de l'ordre numérique et s'attaque à un défi récurrent dans les analyses de données éducatives: le déséquilibre des classes. Ce déséquilibre, qui se manifeste par une surreprésentation des étudiants en situation d'échec par rapport à ceux qui réussissent, complique la formation de modèles de classification efficaces pour prédire les performances académiques. La qualité des prédictions étant altérée par ce biais, il est donc essentiel de rééquilibrer les données afin d'améliorer la précision des modèles.

Pour répondre à cette problématique, plusieurs techniques d'échantillonnage avancées ont été mises en œuvre. Parmi celles-ci, les méthodes SMOTE (Synthetic Minority Over-sampling Technique), SMOTE Borderline, SMOTE-ENN, et ADASYN ont été utilisées pour augmenter artificiellement la proportion d'exemples de la classe minoritaire (les étudiants réussissant), afin de rétablir un équilibre entre les classes. De plus, une méthode nommée Equi-Fused-Data-based SMOTE est proposée dans le cadre de cette recherche. Cette technique fusionne les données de manière équilibrée et optimise la performance des modèles prédictifs, en tenant compte des spécificités des données éducatives.

L’étude a été conduite auprès de 2176 étudiants de première année en sciences et technologies, permettant ainsi une validation empirique de ces approches. Les résultats obtenus montrent que les techniques d’échantillonnage proposées améliorent significativement la précision des modèles de classification. Ces méthodes offrent la possibilité de mieux identifier les étudiants en difficulté et de cibler plus efficacement les interventions pédagogiques. À terme, cette contribution pourrait participer à la réduction des taux d'échec en fournissant un soutien personnalisé aux apprenants, favorisant ainsi une amélioration globale de la réussite académique.

2) Evaluation précise de la compréhension en programmation

La seconde contribution de cette thèse porte sur l’évaluation de la compréhension des concepts de programmation, un aspect fondamental de l’apprentissage de l’algorithmique. La compréhension, en tant que processus mental complexe, engage plusieurs dimensions cognitives, allant de l’assimilation des connaissances à l’application pratique de ces dernières. Dans le cadre de l’enseignement de la programmation, elle implique non seulement la maîtrise des aspects syntaxiques des langages de programmation, mais aussi une immersion dans la logique algorithmique et la capacité à résoudre des problèmes de manière efficace.

Pour évaluer la compréhension des étudiants de manière précise et automatisée, cette recherche s’appuie sur des modèles de langage pré-entraînés, tels que BERT, RoBERTa, CodeBERT, et UniXcoder. Ces modèles, initialement conçus pour le traitement du langage naturel, ont été adaptés pour analyser la similarité entre les solutions de code soumises par les étudiants et celles jugées correctes par d’expert. En capturant les nuances sémantiques et les structures algorithmiques des codes, ces modèles permettent d’évaluer finement les compétences des apprenants en matière de programmation.

Une contribution majeure de cette thèse réside dans l’introduction d’une technique de fusion pondérée, qui combine les forces de plusieurs modèles de langage afin d’améliorer la précision des évaluations. Cette approche permet de compenser les faiblesses des modèles individuels en exploitant leurs complémentarités, offrant ainsi une évaluation plus nuancée de la compréhension des concepts par les étudiants. Les tests réalisés sur un ensemble de données comprenant 83 solutions d’étudiants à divers problèmes algorithmiques ont démontré que les modèles spécifiques au domaine, tels que CodeBERT et UniXcoder, surpassent les modèles généralistes, notamment dans l'évaluation de la logique et des structures de code.

Structure de la Thèse

La thèse est structurée en cinq chapitres principaux, chacun d'eux abordant un aspect de la problématique centrale, à savoir l'amélioration de l'apprentissage dans les EIAH grâce à des techniques avancées en intelligence artificielle et en évaluation automatique.

Le premier chapitre explore les EIAH. Ce chapitre se penche sur les fondements théoriques et les concepts clés qui définissent ces environnements. Il examine l’évolution des EIAH dans le contexte éducatif, en mettant en lumière leur rôle fondamental dans la personnalisation des parcours d'apprentissage et dans l'amélioration de l'efficacité pédagogique. L'analyse approfondie porte sur les différents types d'EIAH, les défis qu'ils posent en termes de conception, d'implémentation, et d'évaluation. Ce chapitre établit un cadre théorique qui servira de référence tout au long de la thèse, soulignant l'importance d'un soutien pédagogique adapté et personnalisé offert par ces environnements numériques.

Le deuxième chapitre aborde les défis spécifiques liés à l'enseignement de la programmation, en particulier les obstacles que rencontrent les étudiants dans la maîtrise des

concepts fondamentaux et la logique algorithmique. Il explore les différentes approches pédagogiques pour enseigner la programmation, en mettant en évidence les limites des méthodes traditionnelles face aux besoins des apprenants.

Le troisième chapitre traite de l'intégration de l'IA dans l'éducation, avec un focus particulier sur le déséquilibre des données et l'utilisation de modèles pré-entraînés. Ce chapitre examine comment les techniques d'IA peuvent être appliquées pour améliorer la qualité des prévisions dans un contexte éducatif où les données sont souvent déséquilibrées. Il explore en profondeur les concepts de déséquilibre des classes, les défis qu'il pose pour la modélisation prédictive, et les solutions apportées par l'IA, notamment par le biais de modèles pré-entraînés. L'étude porte également sur l'efficacité des modèles de langage pré-entraînés pour analyser et évaluer la compréhension des étudiants, offrant ainsi des perspectives nouvelles pour personnaliser l'apprentissage et fournir un retour plus précis et immédiat.

Le quatrième chapitre se concentre sur la méthodologie adoptée pour traiter le déséquilibre des données, ainsi que sur les résultats et les discussions qui en découlent. Ce chapitre présente en détail les techniques spécifiques utilisées pour rééquilibrer les données, telles que le suréchantillonnage, la méthode introduite Equi-Fused-Data et l'utilisation de modèles de classification adaptés aux contextes de déséquilibre. Il analyse les performances de ces modèles en termes de précision, de rappel, de score F1, et d'AUC-ROC, en comparant les différentes approches pour identifier celles qui offrent les meilleures performances. Les résultats obtenus sont discutés de manière approfondie, en mettant en lumière les implications pour l'évaluation automatique et la personnalisation de l'apprentissage dans les EIAH.

Le cinquième chapitre poursuit l'analyse en se concentrant sur la méthodologie liée à l'utilisation des modèles pré-entraînés pour l'évaluation de la compréhension des étudiants. Ce chapitre décrit les étapes méthodologiques suivies pour adapter ces modèles aux spécificités des données éducatives, en détaillant les processus de prétraitement, de modélisation, et d'évaluation. Il présente les résultats obtenus grâce à l'application de ces modèles, en les comparant à d'autres approches pour démontrer leur efficacité et leur précision dans l'évaluation des compétences en programmation. La discussion qui suit met en perspective ces résultats dans le contexte des EIAH, en proposant des recommandations pour l'amélioration des pratiques d'évaluation et en soulignant l'importance d'une approche holistique qui combine plusieurs techniques d'IA pour optimiser les résultats pédagogiques. Ce chapitre conclut la thèse en offrant des pistes pour de futures recherches et applications dans le domaine de l'éducation assistée par l'IA.

SECTION Aspect Théorique

Ces dernières années, la programmation a connu un essor considérable grâce au développement rapide d'outils fonctionnels et fiables. L'intégration des nouvelles technologies de l'information et de la communication, associée à l'intelligence artificielle, a transformé de nombreux aspects de notre quotidien, de l'éducation au divertissement. Ce contexte a renforcé l'importance de la programmation, désormais considérée comme une compétence essentielle du XXIe siècle (Ling Chean et al., 2018). En conséquence, la profession d'informaticien a acquis une nouvelle envergure, rendant nécessaire une adaptation des méthodes d'apprentissage de la programmation pour suivre ce rythme.

Malgré cette évolution, l'enseignement de la programmation reste perçu comme un défi majeur, notamment pour les apprenants novices (Abdessemed et al., 2018 ; Futschek et Moschitz, 2010 ; Gross et Powers, 2005 ; Lahtinen et al.). Plusieurs études, telles que celles de (Futschek et Moschitz, 2010 ; Guibert et Girard, 2003 ; Kaasbøll, 1998 ; Pillay et Jugoo, 2005), soulignent des taux d'échec élevés dans les cours d'introduction à la programmation, oscillant entre 25 et 80 % selon les contextes. Ce taux d'échec est souvent attribué à des explications complexes, des méthodes pédagogiques inadaptées, ou encore à des attentes qui ne correspondent pas aux compétences initiales des apprenants (Ling Chean et al., 2018).

Outre les méthodes d'enseignement, plusieurs facteurs influencent la réussite en programmation, notamment la capacité intellectuelle, la motivation, et les stratégies d'apprentissage (Pillay et Jugoo, 2005 ; Ramalingam et Wiedenbeck, 1998). Si des études antérieures ont mis en avant certains facteurs comme les styles d'apprentissage (Kolb, 1984), des recherches plus récentes ont nuancé cet impact, affirmant que les préférences des apprenants ne sont pas directement corrélées à leurs performances (Nancekivell et al., 2020).

Face à ces défis, il devient pertinent de repenser l'évaluation des apprenants en programmation, car de nombreux étudiants sont confrontés à des exigences pour lesquelles ils ne sont pas préparés. C'est ici qu'interviennent les Environnements Informatiques pour l'Apprentissage Humain (EIAH), conçus pour offrir des solutions pédagogiques adaptées aux besoins individuels des apprenants.

Dans cette première section théorique, nous explorerons d'abord en détail les EIAH, en mettant en lumière leurs composantes essentielles et leur impact sur l'éducation. Nous analyserons les avantages qu'ils offrent en termes de personnalisation des apprentissages, mais aussi leurs limitations, notamment en matière d'évaluation. Une attention particulière sera portée sur l'importance de la compréhension dans ces environnements, clé pour réduire les taux d'échec et favoriser une meilleure rétention des connaissances.

Nous nous pencherons ensuite sur les défis spécifiques à l'apprentissage de la programmation, en nous concentrant sur les obstacles auxquels font face les novices. Des travaux connexes seront également passés en revue pour offrir une vue d'ensemble des approches pédagogiques existantes et des solutions proposées par la recherche. Sur le plan technique, cette section abordera des concepts avancés comme l'intelligence artificielle et l'apprentissage automatique, avec un focus sur le déséquilibre des classes dans les données éducatives. Nous étudierons les méthodes proposées pour atténuer ce déséquilibre, notamment via des techniques algorithmiques adaptées. Enfin, nous analyserons l'utilisation des modèles de langage pré-entraînés (BERT, RoBERTa, CodeBERT, UniXcoder) et leur apport dans l'évaluation automatique des compétences en programmation, en soulignant leur potentiel dans les EIAH.

Cette section se veut une synthèse exhaustive des enjeux théoriques, pratiques et techniques liés à l'enseignement de la programmation, offrant ainsi une vue d'ensemble cohérente pour appréhender les défis actuels et les solutions émergentes dans les environnements d'apprentissage informatisés.

CHAPITRE I : Les Environnements Informatiques pour l'Apprentissage Humain (EIAH)

1 Introduction aux EIAH

L'avènement de l'Internet participatif et collaboratif a ouvert de nouvelles perspectives pour réinventer le domaine de l'éducation (Dodero et al., 2015; Johanes & Lagerstrom, 2017; Kurilovas et al., 2015). Cette évolution a transformé le rôle des établissements d'enseignement en les poussant à intégrer l'utilisation de diverses technologies afin d'améliorer les expériences d'enseignement et d'apprentissage. Depuis plusieurs décennies, accentuée mais non initiée par la pandémie de COVID-19, l'apprentissage en ligne a été en constante évolution, répondant à des besoins croissants de flexibilité et d'accessibilité dans l'éducation (Qiao et al., 2021; UNESCO, 2020). Les institutions éducatives ont progressivement adopté des environnements virtuels, des plateformes d'apprentissage à distance, et des outils numériques pour compléter et enrichir l'enseignement traditionnel.

L'apprentissage en ligne, qui se définit comme tout apprentissage facilité par l'utilisation d'appareils ou d'applications électroniques, se décline sous plusieurs formes: des cours en ligne, des manuels électroniques, des webinaires ou des conférences en direct (.C. Clark and R.E. Mayer, 2016). Sa popularité a atteint son apogée en raison de la pandémie de COVID-19, entraînant ainsi le passage en ligne de nombreuses pratiques éducatives. Dès 2012, des plateformes comme Coursera faisaient leur apparition, proposant des cours en ligne gratuits issus de grandes universités. Cette initiative a rapidement rencontré un vif succès, attirant des millions d'utilisateurs à travers le monde (Al-Mouh et al., 2014). Dans la même lancée, edX a été fondée en 2013 par le MIT et Harvard, avec pour mission de démocratiser l'accès à des formations en ligne de qualité dans des domaines variés, allant des sciences informatiques à la gestion en passant par les sciences humaines. Khan Academy, créée en 2006 par Salman Khan, s'est quant à elle imposée comme une ressource précieuse pour les élèves de tous âges. Ses tutoriels vidéo éducatifs gratuits ont contribué à lever les barrières à l'éducation et à la rendre accessible à un public plus large. Cette modalité pédagogique répondait à des besoins et des aspirations de plus en plus marqués en matière d'éducation, misant sur la flexibilité et l'accessibilité pour répondre aux attentes d'un public en quête de nouvelles connaissances et compétences. Si la crise sanitaire a indéniablement accéléré l'adoption de ce type d'apprentissage, il est important de reconnaître qu'elle n'en est pas à l'origine. L'essor de cette modalité pédagogique s'inscrit dans une tendance de fond, portée par l'évolution des technologies et une demande croissante pour une éducation plus flexible et plus accessible.

Parallèlement, un autre type d'apprentissage, l'apprentissage mobile, a vu le jour. Celui-ci se fait par le biais d'appareils mobiles tels que les smartphones, les tablettes ou les ordinateurs portables, offrant une accessibilité et un coût abordable pour les apprenants situés dans des lieux géographiques différents (Asif & Krogstie, 2012; Junxiang & Yu, 2019; Yallihep & Kutlu, 2019; Zare, 2011). L'apprentissage mobile, plus connu sous le terme m-learning, permet non seulement une flexibilité temporelle et spatiale, mais aussi une personnalisation accrue des parcours d'apprentissage grâce à des applications et des contenus adaptés aux besoins individuels.

L'intégration des technologies dans l'éducation ne se limite pas à la simple numérisation des contenus. Elle inclut également le développement de nouveaux environnements informatiques pour l'apprentissage humain, qui offrent des opportunités uniques pour personnaliser l'apprentissage, favoriser l'interactivité, et fournir un retour d'information immédiat aux apprenants. Les EIAH utilisent des technologies avancées comme l'intelligence artificielle, la réalité augmentée, et les simulations interactives pour créer des expériences d'apprentissage immersives et engageantes (Samaniego Erazo et al., 2015) (figure I.1). Ces environnements

permettent de surmonter certaines des limitations des méthodes traditionnelles en offrant des approches adaptatives et centrées sur l'apprenant, tout en soutenant les enseignants dans leur mission pédagogique. Ainsi, l'apprentissage en ligne et mobile ont ouvert de nouvelles perspectives pour l'éducation en offrant des solutions innovantes pour surmonter les barrières géographiques et économiques. L'intégration des EIAH dans ce contexte amplifie ces opportunités en offrant des expériences immersives et personnalisées.

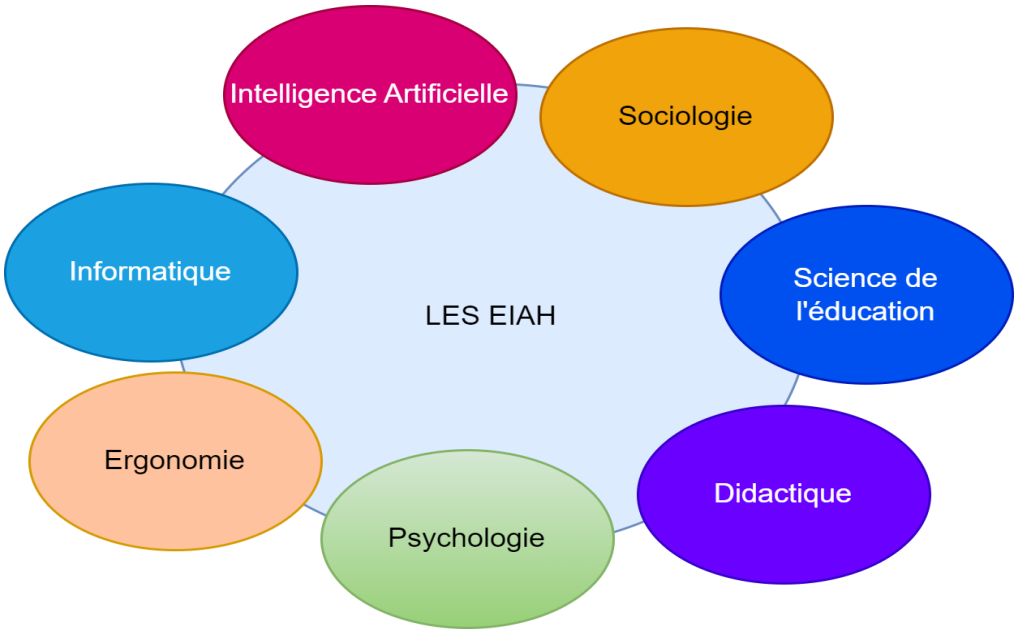

Figure I.1 Croisement Pluridisciplinaire EIAH

1.1 Définition des EIAH

Les Environnements Informatiques pour l'Apprentissage Humain représentent une catégorie de systèmes éducatifs qui ont connu un essor considérable ces dernières décennies. Ces technologies, au cœur des pratiques pédagogiques modernes, ont pour vocation de personnaliser l'apprentissage en répondant aux besoins spécifiques de chaque apprenant, quelle que soit sa culture, son sexe, ses origines ou son éducation (Al-Othman et al., 2017; Bonnat et al., 2018; Gamra & Khlifa, 2019; Gilliot et al., 2019). En effet, ces systèmes sont conçus pour faciliter l'adaptation des instructions pédagogiques et permettent ainsi à chaque apprenant de s'approprier les contenus à son propre rythme.

Plus précisément, les EIAH sont des systèmes informatiques sophistiqués, conçus pour soutenir les activités d'apprentissage en offrant des retours d'information dynamiques en fonction des progrès de l'apprenant (Nkambou et al., 2010). Ces systèmes intègrent des algorithmes avancés d'intelligence artificielle, combinés avec des théories pédagogiques éprouvées, afin de personnaliser et d'adapter le processus d'apprentissage (Bonnat et al., 2018; Gamra & Khlifa, 2019; Korchi & Oughdir, 2015). Par exemple, les techniques d'apprentissage machine sont couramment utilisées pour analyser les comportements des apprenants, identifier leurs lacunes, et proposer des interventions pédagogiques sur mesure.

Les EIAH offrent ainsi une approche centrée sur l'apprenant, caractérisée par un tutorat intelligent, des évaluations adaptatives, et une rétroaction en temps réel. En tirant parti de l'interaction entre l'IA et l'utilisateur, ces environnements créent des expériences d'apprentissage hautement interactives et engageantes, ce qui favorise la rétention des informations et une meilleure compréhension des concepts complexes (Alkhatlan & Kalita, 2018; Baidada et al., 2019; Nkambou et al., 2010; Pezzino, 2018).

Cependant, l'évolution vers une éducation de plus en plus individualisée a mis en lumière les divergences terminologiques qui existent entre les disciplines concernées. Dans la communauté EIAH, la personnalisation des contenus et des apprentissages repose sur la modélisation des profils d'apprenants en fonction de leurs besoins, capacités, intérêts et préférences. À l'inverse, dans le domaine de l'Interface Homme-Machine (IHM), le concept d'adaptation inclut également la prise en compte du contexte d'utilisation, rendant ainsi la conception des EIAH encore plus complexe (Lee & Ko, 2011). Cette diversité de perspectives souligne l'importance d'améliorer en permanence ces environnements afin d'assurer une adaptation plus fine et une personnalisation optimale de l'expérience d'apprentissage pour chaque apprenant.

1.2 Objectifs Pédagogiques des EIAH

La mise en œuvre des EIAH consiste à concevoir, organiser, et gérer des dispositifs pédagogiques intégrant des acteurs humains et des artefacts informatiques sophistiqués (Bonnat et al., 2018). Ces environnements, incluant également les plateformes d'apprentissage en ligne, visent à atteindre divers objectifs pédagogiques pour les étudiants et les établissements d'enseignement. Ils sont conçus pour susciter et accompagner l'apprentissage humain en proposant des activités interactives et personnalisées, adaptées aux profils individuels des apprenants (Grandbastien & Nowakowski, 2014; Khribi et al., 2008). Ainsi, chaque EIAH doit être développé en réponse à une intention didactique ou pédagogique claire, et être utilisé par les apprenants dans le cadre de situations d'enseignement spécifiques (Bonnat et al., 2018). De plus, ces environnements doivent être évalués de manière approfondie pour s'assurer qu'ils répondent aux attentes et pour identifier les facteurs influençant leur efficacité.

Dans un contexte où la diversité des dispositifs pédagogiques, des profils d'apprenants et des environnements d'apprentissage est en constante évolution, cette variabilité nécessite une réflexion approfondie sur la mise en œuvre de différents niveaux de personnalisation, d'adaptation et d'individualisation. En éducation, la personnalisation est souvent perçue comme un ensemble de possibilités permettant de répondre aux besoins spécifiques de chaque utilisateur. L'objectif est d'offrir des services permettant aux apprenants de vivre une expérience d'apprentissage unique, avec des ressources adaptées à leurs besoins, objectifs, désirs et caractéristiques individuelles (Asif & Krogstie, 2012; Xiong et al., 2024). Les EIAH poursuivent donc plusieurs objectifs pédagogiques spécifiques :

- Améliorer les expériences d'apprentissage des étudiants: les EIAH visent à rendre l'apprentissage plus engageant, interactif et adapté aux besoins individuels des apprenants, favorisant ainsi une meilleure compréhension et rétention des informations. Par exemple, les environnements virtuels immersifs peuvent enrichir l'expérience d'apprentissage en permettant aux étudiants d'interagir avec des simulations réalistes (Dalgarno & Lee, 2010). Ces environnements utilisent souvent des interfaces utilisateur avancées et des technologies immersives pour créer des expériences multisensorielles qui augmentent l'engagement et la motivation.
- Promouvoir l'apprentissage actif: les EIAH encouragent les apprenants à participer activement à leur processus d'apprentissage, ce qui renforce la compréhension des sujets étudiés. Les outils tels que les quiz interactifs, les forums de discussion et les projets collaboratifs en ligne stimulent l'interaction et la réflexion critique, permettant aux apprenants de devenir des acteurs de leur propre apprentissage (Persico et al., 2010). Ces dispositifs exploitent les théories de l'apprentissage constructiviste, qui mettent l'accent sur l'apprentissage par la pratique et la résolution de problèmes (Hooshyar et al., 2015).
- Rendre les étudiants proactifs dans leurs études: les EIAH fournissent des outils et des ressources qui permettent aux apprenants de mieux gérer leur temps d'étude et de

devenir plus autonomes. Des tableaux de bord personnalisés, par exemple, aident les étudiants à suivre leurs progrès, à planifier leurs activités d'apprentissage, et à identifier les domaines nécessitant une attention particulière. Cette approche favorise l'autorégulation, un facteur clé de la réussite académique (Xiao & Yang, 2019).

- Réduire le temps nécessaire à l'apprentissage de nouveaux concepts: grâce à des méthodes d'enseignement interactives et adaptatives, les EIAH peuvent accélérer l'apprentissage en s'ajustant au rythme de chaque apprenant. Les systèmes de tutorat intelligents, par exemple, utilisent des algorithmes d'apprentissage machine pour identifier les lacunes dans la compréhension et fournir des exercices ciblés, permettant aux étudiants de combler rapidement ces lacunes et de progresser plus efficacement (Yallihep & Kutlu, 2019).
- Fournir un retour d'information personnalisé: les EIAH utilisent des algorithmes d'intelligence artificielle pour analyser les performances des apprenants et fournir un retour d'information spécifique à chaque individu (Hattie & Timperley, 2007). Ce retour personnalisé peut inclure des recommandations de ressources supplémentaires, des suggestions d'amélioration, et des encouragements pour aider les apprenants à surmonter les obstacles et à atteindre leurs objectifs éducatifs (Kourgiantakis et al., 2019).
- Réduire le coût de l'accès à l'éducation: en intégrant la technologie, les EIAH permettent aux établissements d'enseignement de toucher un public plus large tout en réduisant les coûts associés à l'éducation traditionnelle. Les MOOCs (Massive Open Online Courses), par exemple, offrent un accès à des cours de haute qualité à un coût réduit, permettant ainsi de démocratiser l'accès à l'éducation (Al-Mouh et al., 2014; Sunar et al., 2018). Cette approche permet également d'optimiser l'utilisation des ressources et du personnel enseignant, rendant l'éducation plus accessible et plus durable.

Les EIAH représentent une avancée majeure dans la manière dont l'éducation peut être dispensée, offrant des solutions flexibles, efficaces et personnalisées pour répondre aux besoins variés des apprenants. Leur développement et leur mise en œuvre requièrent une collaboration multidisciplinaire, ainsi qu'une évaluation continue pour garantir leur efficacité et leur adaptabilité aux besoins éducatifs en constante évolution.

1.3 Avantages des EIAH

Les EIAH présentent une multitude d'avantages qui révolutionnent l'éducation moderne. Parmi ces avantages, l'approche personnalisée de l'apprentissage est souvent citée comme étant la plus transformative. Les EIAH sont conçus pour s'adapter aux divers styles d'apprentissage, capacités cognitives, et besoins spécifiques des étudiants, en exploitant des algorithmes d'intelligence artificielle capables de modéliser le profil de chaque apprenant. Ces technologies permettent une personnalisation du parcours éducatif en ajustant dynamiquement les contenus et les méthodes pédagogiques. Des études montrent que cette approche centrée sur l'individu améliore significativement la rétention de l'information et l'engagement des apprenants (Håklev et al., 2017; Nkambou et al., 2010).

La flexibilité offerte par les EIAH constitue également un atout majeur. Contrairement aux méthodes d'apprentissage traditionnelles, les étudiants peuvent accéder aux ressources pédagogiques à tout moment et en tout lieu, s'adaptant ainsi à leur rythme de vie. Cette flexibilité est particulièrement bénéfique pour les apprenants non traditionnels, tels que les adultes qui travaillent ou les étudiants vivant dans des régions éloignées. Des recherches montrent que la possibilité de réviser des contenus selon les besoins individuels conduit à une meilleure compréhension et à une réduction des taux d'abandon (T. Anderson & Elloumi, 2004).

L’apprentissage à distance et via des dispositifs mobiles est un autre avantage clé des EIAH. Ces environnements permettent de surmonter les contraintes géographiques et financières en rendant l’éducation accessible à un public mondial. Cette accessibilité est capitale dans un contexte où l’éducation de qualité est souvent centralisée dans les grandes villes et inaccessible pour ceux vivant dans des régions rurales ou en développement (MAIGA & HOTTE, 2021). Les EIAH, en réduisant les coûts liés aux déplacements et à l'hébergement, participent à la démocratisation de l’éducation.

En définitive, l’utilisation des EIAH favorise le développement des compétences numériques chez les étudiants, un aspect essentiel dans une société de plus en plus axée sur la technologie. L’exposition régulière aux outils numériques dans un contexte éducatif permet aux apprenants de développer une maîtrise des technologies qui va bien au-delà de la simple utilisation. Ces compétences sont nécessaires pour la réussite académique (Veerasamy et al., 2019).

1.4 Limites et les Défis des EIAH

Malgré les nombreux avantages offerts par les EIAH, il est essentiel de reconnaître les limites et les défis inhérents à ces technologies. En effet, bien que les EIAH soient conçus pour démocratiser l'accès à l'éducation, ils soulèvent d'importantes préoccupations en matière d'inégalité d'accès. La fracture numérique, qui désigne la disparité dans l'accès aux technologies de l'information et de la communication (TIC), est l'une des principales limites des EIAH. Dans de nombreuses régions du monde, notamment dans les zones rurales ou les pays en développement, l'accès à Internet et aux dispositifs technologiques reste limité, compromettant ainsi l'équité dans l'éducation (MAIGA & HOTTE, 2021). Cette fracture numérique peut aggraver les inégalités sociales, en excluant certains apprenants des opportunités offertes par les EIAH (Selwyn, 2009).

Par ailleurs, bien que les technologies éducatives puissent potentiellement enrichir l'apprentissage, elles présentent également des risques de distraction pour les apprenants. Les multiples sollicitations des réseaux sociaux, des jeux en ligne et d'autres formes de divertissement numérique peuvent détourner l'attention des étudiants et nuire à leur concentration. Une étude menée par (Junco, 2012) a révélé que l'utilisation intensive des médias sociaux est associée à une baisse des performances académiques. Ce constat soulève des questions sur la capacité des EIAH à maintenir l'engagement des apprenants dans un environnement saturé de distractions numériques.

Par ailleurs, l'apprentissage autodirigé, souvent encouragé par les EIAH, ne convient pas à tous les profils d'apprenants. Bien que certains étudiants puissent prospérer dans un cadre d'apprentissage autonome, d'autres ont besoin d'une structure plus rigide et d'un encadrement direct pour progresser efficacement (P. Kirschner & Van Merrienboer, 2013). L'absence de guidance peut entraîner une dérive de l'apprentissage, où les étudiants se sentent perdus ou démotivés. La flexibilité des EIAH, bien que bénéfique pour certains, peut se révéler contre-productive pour ceux qui requièrent un encadrement régulier et des interactions humaines pour maintenir leur motivation et leur progression.

En outre, l'interaction limitée entre les étudiants et les enseignants dans les environnements EIAH constitue une autre limite majeure. La qualité des interactions pédagogiques, essentielle pour une compréhension profonde des concepts, peut être compromise dans un cadre virtuel. Les rétroactions, souvent automatisées dans les EIAH, peuvent manquer de la nuance et de l'approfondissement nécessaires pour guider les étudiants vers une réflexion critique (Kourgiantakis et al., 2019). De plus, le manque de contact direct avec les enseignants et les pairs peut conduire à un sentiment d'isolement, réduisant ainsi la motivation et l'engagement des apprenants (Litalien et al., 2019).

Enfin, il convient de souligner que la traduction de l'expérience pédagogique traditionnelle dans un environnement numérique n'est pas toujours optimale. La dynamique des salles de classe, où l'interaction en temps réel et les échanges spontanés jouent un rôle important dans l'apprentissage, est difficilement reproductible dans un cadre EIAH. L'absence de cet environnement social peut affecter la construction des connaissances, qui repose en grande partie sur l'interaction avec les autres et l'apprentissage collaboratif.

Bien que les EIAH représentent une avancée significative dans l'intégration des technologies dans l'éducation, ils ne sont pas exempts de défis. Les questions d'inégalité d'accès, de distraction, d'absence de structure pour certains apprenants, de limitation des interactions humaines et de difficulté à reproduire l'environnement pédagogique traditionnel sont autant de facteurs qui doivent être pris en compte lors de la conception et de l'implémentation de ces systèmes. Pour surmonter ces défis, une réflexion continue et une adaptation des pratiques pédagogiques sont nécessaires afin de maximiser le potentiel des EIAH tout en minimisant leurs effets négatifs.

Cependant, pour que ces environnements informatiques atteignent leur plein potentiel, il faut comprendre les modules fondamentaux qui les composent. Ces éléments clés, tels que les modules de profilage des apprenants, les systèmes de tutorat intelligent et les plateformes de gestion de contenu, jouent un rôle central dans la personnalisation de l'apprentissage et l'adaptation des ressources pédagogiques aux besoins individuels des étudiants. Une exploration approfondie de ces composants permet de mieux appréhender comment les EIAH peuvent être structurés pour répondre aux attentes variées des apprenants.

1.5 Les modules Composants un EIAH

Les EIAH se distinguent par leur architecture modulaire, composée de divers éléments interdépendants, chacun visant à optimiser l'expérience d'apprentissage des utilisateurs. Ces modules sont conçus pour offrir un soutien pédagogique adapté, tout en répondant aux besoins spécifiques de chaque apprenant. Voici une présentation plus détaillée des composantes clés des systèmes EIAH (figure I.2):

1) **Modèle du domaine**: ce module représente l'ensemble des connaissances relatives au domaine d'enseignement. Il englobe la structure conceptuelle du sujet, les relations entre les concepts et les prérequis nécessaires pour comprendre chaque notion. Ce modèle constitue la base sur laquelle sont élaborées les activités d'apprentissage et les évaluations. L'organisation du contenu selon le modèle du domaine permet de structurer l'enseignement de manière logique et progressive (Ulferts et al., 2021).
2) **Modèle de l'apprenant**: central à la personnalisation des EIAH, ce modèle capture les caractéristiques individuelles de l'apprenant. Il comprend des informations sur les connaissances préalables, les préférences d'apprentissage, les styles cognitifs, et les performances passées. Le modèle de l'apprenant permet au système d'adapter dynamiquement le contenu et la méthode d'enseignement pour correspondre au profil unique de chaque utilisateur (Elghibari & Elouahbi, 2015; Premlatha et al., 2016a).
3) **Objectifs pédagogiques**: les objectifs pédagogiques définissent les compétences et connaissances que l'apprenant est censé acquérir à l'issue d'une leçon ou d'une activité. Ces objectifs guident la conception des contenus et des évaluations, assurant que chaque module d'enseignement répond aux standards pédagogiques attendus (L. Anderson et al., 2001).
4) **Contenu pédagogique** : ce composant inclut les leçons, exercices, évaluations, et autres ressources utilisées dans le processus d'apprentissage. Le contenu pédagogique est élaboré pour être en adéquation avec le modèle du domaine et les objectifs pédagogiques. Il peut prendre diverses formes, incluant des textes, des vidéos, des

simulations interactives, et des quiz, chacun étant choisi en fonction de son efficacité pour atteindre les objectifs fixés (El-Sabagh, 2021).

5) **Interface utilisateur** : l'interface utilisateur est l'aspect visuel et interactif du système, par lequel les apprenants interagissent avec les EIAH. Une interface bien conçue améliore l'engagement et l'accessibilité, permettant aux utilisateurs de naviguer facilement entre les modules et de bénéficier d'une expérience fluide et intuitive (Alsumait & Al-Osaimi, 2009).
6) **Adaptabilité et personnalisation**: grâce à des algorithmes sophistiqués, les EIAH peuvent ajuster en temps réel le contenu, la difficulté, et la progression des leçons en fonction des besoins, préférences, et performances de l'apprenant. Cette adaptabilité assure un apprentissage plus efficace et individualisé, en maintenant un équilibre optimal entre défi et soutien (Shemshack & Spector, 2020; Xiong et al., 2024).
7) **Évaluation et rétroaction** : les outils d'évaluation permettent de mesurer les performances de l'apprenant à divers stades du processus d'apprentissage. Le retour d'information personnalisé, qui peut être immédiat ou différé, joue un rôle clé dans la consolidation des connaissances et l'amélioration des compétences. Il aide également les apprenants à identifier leurs points faibles et à orienter leur apprentissage (Pezzino, 2018).
8) **Intelligence Artificielle**: l'IA est utilisée pour analyser les comportements des apprenants, prédire leurs besoins futurs, et recommander du contenu adapté. Elle joue également un rôle dans l'automatisation de certaines tâches, telles que l'évaluation ou la personnalisation du parcours d'apprentissage, améliorant ainsi l'efficacité et la précision du système (Nkambou et al., 2010).
9) **Suivi et analyse des données**: ce composant assure la collecte et l'analyse des données générées par les interactions des apprenants avec le système. Ces données sont utilisées pour affiner les modèles du domaine et de l'apprenant, améliorer l'efficacité des EIAH, et fournir des connaissances aux enseignants sur les progrès et les besoins des étudiants (D'mello & Kory, 2015).
10) **Support aux enseignants**: les EIAH ne se limitent pas à l'apprenant; ils intègrent également des outils dédiés aux enseignants. Ces outils permettent aux éducateurs de suivre les progrès des apprenants, de personnaliser les parcours pédagogiques, et de fournir un soutien supplémentaire si nécessaire. Ils facilitent également la gestion des cours et l'analyse des performances à un niveau macro (Farhan et al., 2018).
11) **Module de scénarisation**: la scénarisation pédagogique est un élément clé des EIAH, permettant de structurer les séquences d'apprentissage de manière cohérente et dynamique. Ce module aide à définir les parcours pédagogiques en fonction des objectifs et des profils des apprenants, en intégrant divers types d'activités et de ressources, tout en respectant une logique pédagogique progressive (Paquette, 2002).

Ces composantes, en interaction constante, permettent de créer des environnements d'apprentissage informatisés à la fois complets, adaptatifs et centrés sur l'utilisateur, répondant ainsi aux exigences des pédagogies modernes tout en offrant une expérience d'apprentissage enrichie et personnalisée.

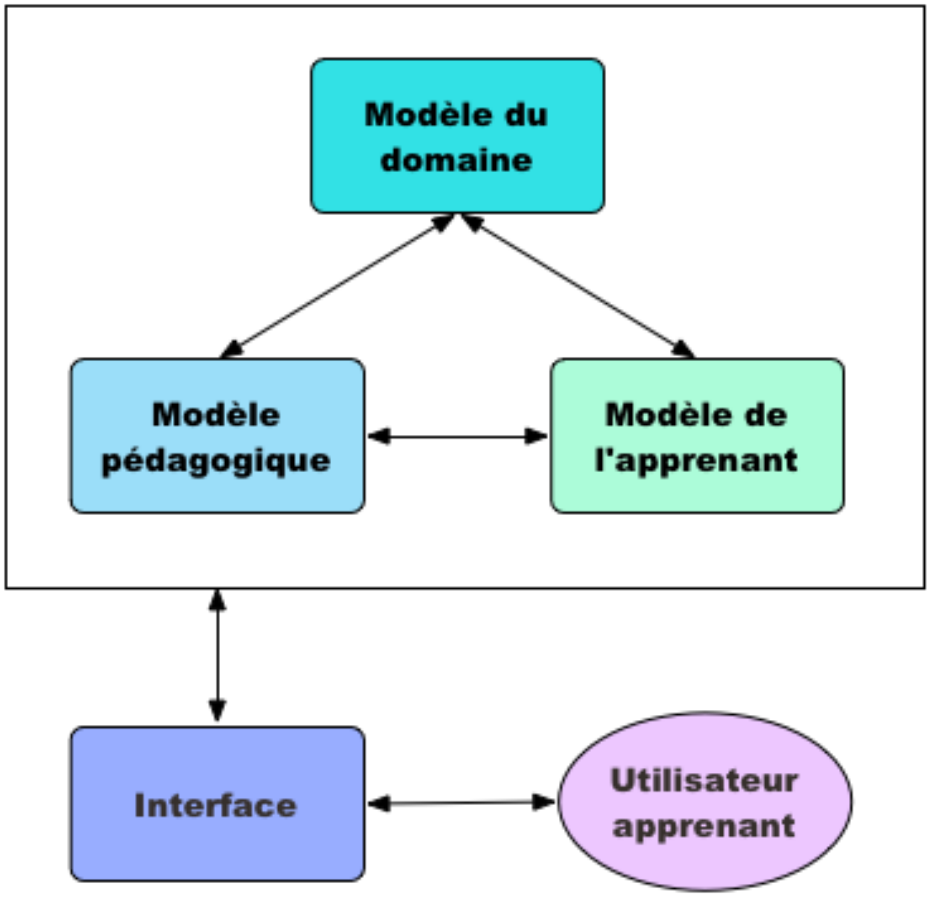

Figure I.2 Architecture Simplifiée des Modules Composant un EIAH (Mayers & Lefebvre, 1992)

1.5.1 Modèle Apprenant (Profil Apprenant)

Ces modèles jouent un rôle dans la personnalisation des parcours éducatifs, offrant aux étudiants des expériences d'apprentissage sur mesure et dynamiques. Les EIAH alimentés par l'intelligence artificielle permettent de proposer des parcours d'apprentissage personnalisés en prenant en compte les particularités intrinsèques des apprenants, telles que leurs capacités cognitives, leurs styles d'apprentissage, leurs motivations et leurs objectifs éducatifs (Shute & Zapata-Rivera, 2012). Le modèle apprenant, ou profil apprenant, constitue une représentation numérique des données collectées sur chaque étudiant. Ces données incluent une variété d'informations indispensables pour la personnalisation des parcours éducatifs (Korchi & Oughdir, 2015; Premlatha et al., 2016a) :

- **Profil cognitif**
  - Compétences de l'apprenant dans différentes disciplines.
  - Niveau de compréhension de concepts spécifiques.
  - Styles d'apprentissage préférés (visuel, auditif, kinesthésique).
  - Capacités de résolution de problèmes et de pensée critique.
- **Profil motivationnel**
  - Objectifs d'apprentissage à court et long terme.
  - Niveau d'engagement et de motivation.
  - Intérêts personnels et professionnels.
- **Profil comportemental**
  - Données d'interaction avec les plateformes d'apprentissage (temps passé, participation aux activités, achèvement des modules).
  - Réactions aux différentes méthodes d'enseignement.

- Préférences pour les types de supports éducatifs (vidéos, textes, exercices pratiques).

❖ **Profil émotionnel**

- Réactions émotionnelles face à des tâches d'apprentissage difficiles.
- Niveau de stress et d'anxiété lors des évaluations.
- Satisfaction globale par rapport à l'expérience d'apprentissage.

Ces éléments constituent un modèle apprenant au cœur des processus de personnalisation, permettant aux systèmes éducatifs de s'adapter en temps réel aux besoins spécifiques de chaque apprenant. D'ailleurs, le principal objectif du profilage de l'apprenant dans les EIAH est de personnaliser l'expérience éducative pour optimiser les acquis pédagogiques. En comprenant les capacités de l'apprenant et son niveau de compréhension d'un sujet particulier, il est possible de créer des programmes d'apprentissage plus pertinents et mieux adaptés aux besoins individuels (Al-Othman et al., 2017). Cela va au-delà de la simple transmission de connaissances, car il s'agit également de motiver et d'encourager l'apprenant en lui offrant des contenus qui correspondent à ses centres d'intérêt et à ses modes de fonctionnement préférés (Drachsler & Greller, 2011).

Pour créer un profil apprenant efficace, il est nécessaire de collecter des données provenant de diverses sources. Parmi celles-ci, on trouve les tests cognitifs, les auto-évaluations, les analyses d'apprentissage basées sur les interactions avec les systèmes éducatifs, ainsi que les entretiens personnels (Akçapınar et al., 2020). La nature des données recueillies et la méthode de collecte varient en fonction de la plateforme d'apprentissage en ligne utilisée. Par exemple, dans un système de gestion de l'apprentissage, les données sur l'engagement de l'apprenant, le temps passé sur les modules, et les performances aux évaluations sont souvent utilisées pour enrichir le profil (Baker & Siemens, 2014). De plus, les auto-évaluations permettent de capter des informations subjectives mais cruciales, comme les styles d'apprentissage privilégiés ou les intérêts personnels, ce qui complète la vision holistique de l'apprenant.

Une fois le profil de l'apprenant établi, il sert de fondement à la conception de programmes d'apprentissage personnalisés. Une composante clé de ces programmes est l'utilisation d'algorithmes d'apprentissage adaptatif. Ces algorithmes ajustent continuellement le contenu éducatif en fonction des progrès de l'apprenant, garantissant ainsi que celui-ci reçoit un matériel ni trop difficile ni trop facile (Abdessemed et al., 2018). Cela rend l'expérience d'apprentissage non seulement plus engageante, mais aussi plus efficace, car elle respecte le rythme et les besoins spécifiques de chaque individu (Desmarais & Baker, 2012).

1.5.2. Avantages du Modèle Apprenant

Le modèle apprenant, étant un pilier fondamental des EIAH, vise à offrir une représentation holistique et détaillée de chaque apprenant. Cette approche présente divers bénéfices notables, notamment :

❖ Amélioration de la qualité globale de l'expérience d'apprentissage

- Personnalisation des parcours éducatifs : le modèle apprenant permet d'adapter les contenus pédagogiques en fonction des besoins spécifiques de chaque étudiant, favorisant ainsi une meilleure compréhension des matières étudiées (Grandbastien & Nowakowski, 2014).
- Adaptation des contenus pédagogiques aux styles d'apprentissage: en tenant compte des styles d'apprentissage variés, tels que les approches visuelles, auditives ou kinesthésiques, le modèle apprenant aide à fournir du matériel pédagogique qui résonne mieux avec l'apprenant (Felder, 1988).

- Augmentation de l'engagement des apprenants: la personnalisation des contenus favorise un engagement accru des apprenants, réduisant ainsi le taux d'abandon dans les programmes en ligne (Chi & Wylie, 2014).

❖ Soutien, retour d'information et conseils personnalisés
- Accès à des recommandations éducatives spécifiques en fonction des performances: en analysant les données sur les performances des étudiants, les EIAH peuvent suggérer des activités de révision ciblées ou des ressources complémentaires (Abdessemed et al., 2018).
- Renforcement des chances de réussite et d'assimilation des concepts: un retour d'information immédiat et personnalisé permet aux étudiants de corriger leurs erreurs en temps réel, ce qui améliore la rétention des connaissances (Hattie & Timperley, 2007).
- Encouragement continu par un retour d'information ciblé, ajusté aux besoins de l'apprenant: l'accompagnement individualisé, fondé sur le profil de l'apprenant, augmente la motivation intrinsèque des étudiants (Litalien et al., 2019).

❖ Adaptabilité et dynamique intrinsèque du modèle apprenant
- Mise à jour en temps réel en fonction du progrès de l'étudiant: le modèle apprenant évolue continuellement en fonction des interactions de l'étudiant, garantissant ainsi une pertinence continue du contenu pédagogique (Premlatha et al., 2016a).
- Suivi des avancées de l'apprenant, permettant de répondre à ses évolutions: le modèle apprenant offre une vue d'ensemble des progrès et des difficultés de l'étudiant, facilitant des interventions pédagogiques précises et opportunes (Desmarais & Baker, 2012).

❖ Précision du modèle apprenant
- Optimisation du retour d'information et du soutien pédagogique: un modèle apprenant précis permet d'ajuster le niveau de difficulté du contenu et les stratégies pédagogiques en fonction des besoins individuels (Kourgiantakis et al., 2019).
- Évitement des biais dans les recommandations éducatives, minimisant ainsi les risques de frustration ou d'échec de l'apprenant: une mauvaise calibration du modèle apprenant peut induire des recommandations inappropriées, ce qui nuit à l'efficacité de l'apprentissage (Koedinger et al., 2012).

❖ Collecte et utilisation des données diverses
- Utilisation de sources variées telles que les entrées du profil étudiant, les résultats des évaluations, et les ressources d'apprentissage interactives: la diversification des sources de données permet de créer un profil plus complet et nuancé de l'apprenant (Pardo & Siemens, 2014).

L'importance du modèle apprenant réside également dans sa capacité à favoriser la rétention et la mémorisation, qui sont souvent des défis dans les environnements d'apprentissage traditionnels. Les systèmes de tutorat intelligents, en utilisant les données du modèle apprenant, peuvent réduire de manière significative les interférences causées par l'oubli et établir des associations pertinentes plus tôt que dans les méthodes traditionnelles, en donnant la priorité aux révisions et aux retours en arrière stratégiques (Hattie & Timperley, 2007).

❖ Optimisation de la rétention et de la mémorisation
- Réduction des interférences dues à l'oubli grâce à des révisions ciblées: les rappels réguliers et les révisions adaptatives ont prouvé leur efficacité pour renforcer la consolidation des connaissances (Karpicke & Roediger, 2008; Rawson & Dunlosky, 2011).

- Établissement précoce de connexions et d'associations pertinentes: un modèle apprenant précis est essentiel pour faciliter la formation de connexions durables entre nouvelles informations et connaissances antérieures (Schwartz, 2021).
- Amélioration de la rétention à long terme par des retours en arrière stratégiques: les stratégies de retour en arrière, lorsqu'elles sont mises en œuvre de manière systématique, renforcent les liens cognitifs et favorisent la rétention à long terme (Bertilsson et al., 2020).

1.5.3 Limites du Modèle Apprenant

Malgré les nombreux avantages du profilage de l'apprenant, certaines limites doivent être reconnues. Bien que cet outil soit essentiel pour la personnalisation de l'expérience d'apprentissage, il présente plusieurs défis qui peuvent limiter son efficacité.

- ❖ Risque de biais lors de la collecte et de l'analyse des données
  - Biais algorithmique: les algorithmes utilisés pour analyser les données des apprenants peuvent introduire des biais, en particulier s'ils sont entraînés sur des ensembles de données non représentatifs ou biaisés (Noble, 2018). Cela peut entraîner des recommandations pédagogiques inéquitables.
  - Discrimination involontaire basée sur les caractéristiques démographiques: il existe un risque que les caractéristiques démographiques (telles que l'âge, le sexe, l'origine ethnique) influencent la manière dont les profils sont interprétés, ce qui peut conduire à un traitement inégal des apprenants (Binns, 2017).
  - Biais culturel et préjugés inconscients: les différences culturelles peuvent également biaiser la manière dont les apprenants interagissent avec le contenu pédagogique, ce qui n'est pas toujours pris en compte dans la conception des modèles apprenants (Henrich et al., 2010).
- ❖ Réticence à partager des informations personnelles
  - Préoccupations liées à la confidentialité: de nombreux apprenants peuvent être réticents à partager des informations personnelles, notamment en raison de préoccupations concernant la confidentialité et la sécurité des données (Leiva et al., 2021).
  - Impact sur la précision du profil: l'absence de données personnelles pertinentes peut limiter la capacité du modèle apprenant à fournir des recommandations précises et adaptées, compromettant ainsi l'efficacité de la personnalisation (Pardo & Siemens, 2014).
- ❖ Évolution des profils d'apprenants au fil du temps
  - Dynamique des compétences et des préférences: les compétences, préférences et styles d'apprentissage des apprenants évoluent au fur et à mesure qu'ils progressent dans leur parcours éducatif (Kolb, 1984). Un profil initialement pertinent peut devenir obsolète, nécessitant des mises à jour régulières pour maintenir l'efficacité de la personnalisation.
  - Nécessité d'une réévaluation continue: pour que le modèle apprenant reste pertinent, il doit être réévalué et mis à jour en continu, ce qui peut être complexe et chronophage (Bull & Kay, 2007).
- ❖ Sur-personnalisation et perte d'exposition à la diversité
  - Risques de sur-personnalisation: un profil apprenant trop spécifique peut limiter l'exposition des apprenants à des perspectives et à des contenus variés, restreignant ainsi leur développement intellectuel (Pariser, 2012).
  - Réduction de la flexibilité cognitive: la sur-personnalisation peut également réduire la capacité des apprenants à s'adapter à de nouvelles situations ou à explorer des domaines au-delà de leur zone de confort (Poudel & Shrestha, 2023).

❖ Limites dans la prise en compte des styles d'apprentissage
    - Hétérogénéité des styles d'apprentissage: les styles d'apprentissage varient considérablement d'un individu à l'autre, et un profil standardisé peut ne pas capturer toute la complexité des préférences individuelles (P. A. Kirschner, 2017a).
    - Réduction des styles d'apprentissage à des catégories simplifiées: la tendance à réduire les styles d'apprentissage à des catégories simplifiées peut conduire à une personnalisation inefficace, qui ne répond pas aux besoins réels des apprenants (Thongchotchat et al., 2023).

2 Les Styles d'Apprentissage

Depuis que l'idée des styles d'apprentissage a été introduite dans le monde de l'éducation, elle a fait l'objet d'un débat parmi les éducateurs. Le concept de style d'apprentissage suggère que chaque personne a sa propre façon de traiter l'information et d'apprendre de nouvelles idées. Ces styles d'apprentissage dépendraient de facteurs tels que la personnalité, la culture et même le sexe. Nous proposons de définir et de décrire les styles d'apprentissage, de discuter des limites de ces styles et, enfin, d'explorer des études récentes qui suggèrent que la notion de styles d'apprentissage semble être un mythe.

2.1 Définition des Styles d'Apprentissage

La notion de style d'apprentissage repose sur la prémisse que chaque apprenant possède une manière unique de comprendre et de mémoriser les connaissances (Pashler et al., 2008; Subagja & Rubini, 2023; Thongchotchat et al., 2023). Dans la littérature pédagogique, quatre styles d'apprentissage principaux sont fréquemment identifiés : auditif, visuel, kinesthésique et lecture/écriture.

- Style auditif: les apprenants auditifs préfèrent écouter des conférences ou des discussions. Ils retiennent mieux l'information lorsqu'elle est présentée verbalement et sont souvent sensibles aux nuances tonales et rythmiques du discours (Felder, 1988).
- Style visuel: les apprenants visuels nécessitent des aides visuelles pour assimiler l'information. Ils tirent profit de la visualisation des concepts à travers des graphiques, des images, des schémas et des vidéos
- Style kinesthésique: les apprenants kinesthésiques apprennent mieux par le biais d'activités pratiques et d'expériences tactiles. Ils préfèrent manipuler des objets et participent activement à des simulations ou des projets concrets. L'auteur (Kolb, 1984) propose que l'apprentissage expérientiel, qui engage les sens et le mouvement, est particulièrement bénéfique pour ces apprenants.
- Style lecture/écriture : les apprenants en lecture/écriture privilégient la lecture de textes et la rédaction de notes pour assimiler l'information. Ils trouvent bénéfique de réécrire les concepts et de se référer à des manuels ou des articles.

Pour optimiser l'efficacité des EIAH et de l'apprentissage en ligne, il peut être utile de fournir du matériel pédagogique adapté aux divers styles d'apprentissage. Cela permet de répondre aux besoins d'un large éventail d'apprenants, améliorant ainsi l'accessibilité et l'inclusivité des programmes éducatifs (Morris, 2019).

- Apprenants visuels: utilisation de présentations visuelles comprenant des diagrammes, des graphiques et des animations pour illustrer les concepts clés.
- Apprenants auditifs: intégration de cours magistraux enregistrés, de podcasts, et de discussions en ligne pour faciliter la compréhension verbale.

- Apprenants kinesthésiques: développement de simulations interactives et de tâches pratiques qui permettent une immersion tactile et expérientielle.
- Apprenants lecture/écriture: fourniture de documents textuels, de livres électroniques et d'opportunités pour la prise de notes et la rédaction.

L'adaptation des EIAH aux différents styles d'apprentissage n'est pas seulement une question de préférence individuelle, mais une stratégie pédagogique fondée sur des preuves visant à maximiser l'engagement et la réussite des apprenants. Toutefois, cette approche présente également certaines limites.

2.2 Limites des Styles d'Apprentissage

Malgré le débat actuel sur les styles d'apprentissage, plusieurs études récentes montrent qu'il pourrait s'agir en fait d'un mythe. Les chercheurs affirment que de nombreux modèles de styles d'apprentissage populaires ne sont pas étayés par des preuves scientifiques. Ils affirment également que le fait d'enseigner à un apprenant selon son style d'apprentissage préféré n'améliore pas nécessairement ses résultats scolaires (P. A. Kirschner, 2017a; Morris, 2019; Nancekivell et al., 2020). Par exemple, une étude (Pashler et al., 2008) a comparé l'enseignement dispensé à des étudiants utilisant leur style d'apprentissage préféré à celui dispensé à des étudiants utilisant un style d'apprentissage non préféré. Les résultats n'ont pas montré de différences significatives dans les résultats scolaires entre les deux groupes.

De même, une analyse (Coffield et al., 2004) affirmait qu'il y avait peu de preuves en faveur des styles d'apprentissage et que le concept reposait sur des hypothèses erronées. En 2017, un groupe interdisciplinaire de chercheurs a publié une critique des croyances actuelles concernant les styles d'apprentissage. Les études de (P. A. Kirschner, 2017a; P. Kirschner & Van Merrienboer, 2013) ont analysé un ensemble complet de recherches sur les styles d'apprentissage. Les chercheurs affirment que le concept de style d'apprentissage ne reflète pas la complexité de l'apprentissage et découragent l'utilisation de cette idée. En outre, ils ont suggéré qu'il pourrait être préjudiciable de s'accrocher à cette notion, en particulier si les éducateurs s'appuient sur elle pour élaborer des stratégies d'enseignement. Par conséquence, bien que le concept des styles d'apprentissage puisse sembler raisonnable, certains éducateurs et chercheurs ont émis des doutes quant à la validité et à l'exactitude de la théorie. Pour résumer, voici quelques-unes des limites des styles d'apprentissage:

- **Manque de preuves**: malgré la croyance largement répandue que les styles d'apprentissage existent, il y a peu de preuves empiriques à l'appui. Certaines études suggèrent qu'il n'y a pas de relation significative entre le style d'apprentissage préféré d'un élève et ses résultats scolaires (P. A. Kirschner, 2017b; Nancekivell et al., 2020). De surcroît, le fait de s'appuyer uniquement sur les styles d'apprentissage préférés peut entraver le processus d'apprentissage en limitant l'exposition des étudiants à de nouvelles méthodes d'apprentissage.
- **La complexité du cerveau**: le cerveau est un organe incroyablement complexe, dont les différentes régions travaillent ensemble pour stocker et traiter l'information. Il est donc très peu probable que les styles d'apprentissage puissent être divisés en quatre catégories seulement (Almeida & Winfield, 2012).
- **Manque de ressources**: les enseignants risquent de ne pas disposer des ressources nécessaires pour personnaliser l'enseignement : compte tenu de la multiplicité des styles d'apprentissage, les enseignants n'auraient probablement pas les ressources nécessaires pour créer des plans de cours sur mesure pour chaque élève. Enseigner de cette manière serait un défi majeur et, dans la pratique, pourrait conduire à de nombreuses méthodologies d'enseignement inutilement compliquées et inefficaces (Tomlinson, 2001).

- Une autre limite est que l'évaluation des styles d'apprentissage est souvent subjective. La plupart des évaluations des styles d'apprentissage sont autodéclarées, ce qui rend difficile la vérification de l'exactitude des préférences des apprenants. De plus, certains étudiants peuvent avoir des préférences qui ne sont pas incluses dans les catégories communes utilisées dans la plupart des évaluations (Nancekivell et al., 2020).

Les styles d'apprentissage sont donc un sujet de débat dans le monde de l'éducation depuis de nombreuses années. La théorie suggère que chaque personne a sa propre façon de traiter et d'intégrer l'information. Cependant, plusieurs études récentes ont discrédité cette notion, arguant du fait qu'il existe peu de preuves scientifiques à l'appui de cette théorie. Il est important de comprendre que, même si les styles d'apprentissage semblent efficaces, il n'en demeure pas moins important d'adopter une approche nuancée et de ne pas se fier à des catégories trop simplistes. D'ailleurs, les cours d'apprentissage en ligne devraient donc être conçus en tenant compte des diverses limites et des preuves scientifiques récentes afin de garantir des parcours d'apprentissage personnalisés et éventuellement des résultats d'apprentissage optimaux.

3 L'évaluation dans le Cadre des EIAH

Dans le domaine de l'éducation, l'évaluation est définie comme un processus de collecte et d'analyse de données, visant à déterminer l'efficacité de l'apprentissage, de l'enseignement et du programme d'études. Il s'agit d'un élément essentiel de tout système éducatif, y compris des EIAH. Elle est réalisée depuis les étapes initiales de l'apprentissage jusqu'aux étapes finales, marquant l'achèvement du processus éducatif.

Au sein d'un système EIAH, l'évaluation implique l'appréciation des performances de l'étudiant afin de fournir un retour d'information et de promouvoir son apprentissage (Chachoua, 2019; Durfee et al., 2007; Gross & Powers, 2005). L'évaluation assistée par ordinateur désigne l'utilisation d'ordinateurs et de programmes informatiques pour évaluer les performances des étudiants. D'une part, elle est largement utilisée en raison de sa rapidité et de sa précision permettant ainsi aux éducateurs d'économiser du temps, des efforts et des ressources. D'autre part, dans les EIAH, ce type d'évaluation permet aux étudiants de recevoir un retour d'information immédiat sur leurs performances, ce qui facilite l'apprentissage et minimise le risque d'erreurs. Ces méthodes d'évaluation peuvent également reconnaître avec précision les capacités des élèves et générer des rapports qui peuvent aider les enseignants à comprendre les progrès réalisés par les apprenants dans le cadre du programme d'études (Hattie & Timperley, 2007; Labat, 2002; Pezzino, 2018).

Techniquement, l'évaluation dans un processus d'apprentissage implique la collecte d'informations sur les connaissances, les compétences et les attitudes des élèves à l'égard d'un sujet spécifique (Cubero-Ibáñez et al., 2017; Xiao & Yang, 2019). Ces informations sont une composante essentielle de la conception des cours et des programmes d'enseignement. L'évaluation doit donc être un processus continu dans le cycle d'apprentissage, et le retour d'information doit être fréquent (Labat, 2002). Dans le contexte des EIAH, l'évaluation vise à fournir aux étudiants un retour d'information sur les progrès qu'ils réalisent dans l'apprentissage de certains sujets et sur la manière dont ils peuvent s'améliorer (Dahalan & Hussain, 2010; Xiao & Yang, 2019). L'évaluation permet donc, une certaine remise en question pour les deux parties (étudiants et enseignants).

Différents aspects de la performance de l'élève peuvent être évalués dans le cadre des EIAH, notamment les connaissances, les compétences, l'attitude et le comportement (Dahalan & Hussain, 2010; Durfee et al., 2007; Kay & Knaack, 2009). Pour commencer, l'évaluation des connaissances est généralement utilisée pour déterminer si l'élève comprend les concepts enseignés dans le matériel pédagogique. Ensuite, l'évaluation des compétences vérifie la

capacité de l'élève à appliquer les connaissances acquises dans le contexte approprié. Quant à l'évaluation de l'attitude, elle consiste à évaluer l'attitude de l'étudiant à l'égard de la matière, du cours et du processus d'apprentissage. D'ailleurs, l'évaluation du comportement se concentre sur les actions de l'étudiant, telles que la participation et l'engagement dans les activités.

Néanmoins, il est impératif de réaliser des évaluations en amont dans le cadre des EIAH afin de fournir un retour d'information ciblé dès le début du processus d'apprentissage (Pezzino, 2018). En effet, une évaluation en amont permet aux étudiants d'ajuster leurs stratégies d'apprentissage et de se concentrer sur les domaines à améliorer. En outre, elle permet également d'identifier les domaines dans lesquels l'enseignement doit être révisé pour répondre aux besoins d'apprentissage des étudiants.

Les outils d'évaluation utilisés dans le cadre des EIAH sont multiples et visent à mesurer diverses dimensions de l'apprentissage. Ces outils comprennent notamment:

- Questions à choix multiples (QCM): les QCM sont largement utilisés en raison de leur capacité à évaluer rapidement les connaissances factuelles et la compréhension conceptuelle des apprenants. Leur conception doit cependant être bien réfléchie pour éviter les biais et assurer la validité des évaluations (Haladyna et al., 2002).
- Questions à réponse courte: ces questions permettent aux apprenants de formuler des réponses concises, favorisant l'expression de leur compréhension sans les contraintes des formats de réponse prédéfinis. Elles sont particulièrement utiles pour évaluer la capacité à synthétiser des informations et à formuler des concepts avec précision (Kramer et al., 2023).
- Essais: les essais sont des outils d'évaluation approfondis qui encouragent les étudiants à démontrer leur capacité à organiser et articuler leurs idées de manière cohérente. Ils sont efficaces pour évaluer la pensée critique, les compétences analytiques et la capacité de rédaction (L. Anderson et al., 2001).
- Évaluations basées sur les performances: celles-ci incluent des simulations et des jeux éducatifs qui permettent d'observer et d'évaluer les compétences pratiques et les comportements des apprenants dans des contextes quasi-réels.
- Portfolios numériques: les portfolios permettent aux apprenants de rassembler et de présenter leurs travaux au fil du temps. Ils offrent une vue d'ensemble des compétences et des progrès de l'apprenant, favorisant la réflexion et l'auto-évaluation. Les portfolios peuvent inclure des projets, des essais, des réflexions personnelles et des enregistrements multimédias (Sadik, 2008).
- Questionnaires d'auto-évaluation: ces questionnaires permettent aux apprenants de réfléchir sur leur propre apprentissage, d'identifier leurs forces et leurs faiblesses, et de définir des objectifs d'amélioration. Ils encouragent la métacognition et l'apprentissage autonome (Nicol & Macfarlane, 2006).
- Jeux sérieux: ils sont conçus pour combiner des objectifs éducatifs avec des éléments de jeu. Ils peuvent améliorer la motivation et l'engagement des apprenants tout en offrant des opportunités d'apprentissage expérientiel (Brisson et al., 2012; Yallihep & Kutlu, 2019).
- Simulation et modélisation: ces techniques permettent aux apprenants de travailler dans des environnements virtuels qui imitent des situations réelles. Elles sont particulièrement utiles dans les domaines où la pratique réelle est coûteuse ou risquée, comme l'aviation (Alfred et al., 2019; Sargent, 2011).

Ces outils sont particulièrement pertinents pour les domaines nécessitant des compétences techniques et interpersonnelles complexes (Alfred et al., 2019). Ils permettent une évaluation holistique des connaissances, des compétences, des attitudes et des comportements des apprenants. Chaque méthode d'évaluation a ses propres avantages et inconvénients, et le choix

des méthodes est majoritairement guidé par les objectifs d'apprentissage, le contexte éducatif et les besoins des apprenants pour assurer la validité et la fiabilité des résultats obtenus (Biggs, 2003).

3.1 Méthodes d'Evaluation des Apprenants dans les EIAH

L'évaluation est un processus complexe et multifacette destiné à mesurer les connaissances, les compétences, les aptitudes et les performances des apprenants. Elle se décline sous différentes formes et méthodes, adaptées aux objectifs spécifiques et aux contextes d'apprentissage. Dans le cadre des EIAH, on distingue plusieurs types d'évaluation:

- **Des évaluations formatives:** ce sont des évaluations continues réalisées tout au long de l'apprentissage pour fournir des retours d'information aux élèves et aux enseignants. Ces retours permettent aux apprenants d'ajuster leur apprentissage, de développer leurs compétences et de progresser de manière ciblée (Staff, 1998). Les évaluations formatives peuvent prendre la forme de questions orales, de devoirs, de projets, de présentations ou de discussions (Garrison, C., & Ehringhaus, M., 2007).
- **Des évaluations sommatives:** ce sont des évaluations réalisées à la fin d'une unité d'enseignement, d'un cours ou d'une année scolaire pour mesurer le niveau de compétence atteint par les élèves (Garrison, C., & Ehringhaus, M., 2007). Ces évaluations peuvent prendre la forme de tests standardisés, de travaux pratiques ou de projets finaux (Brown, 2017).
- **Des évaluations diagnostiques:** elles sont utilisées pour évaluer les connaissances préalables des apprenants avant le début d'un cours ou d'une formation. Elles permettent d'identifier les besoins individuels et de personnaliser l'enseignement en conséquence (Brown, 2017).
- **Des évaluations par les pairs:** les apprenants peuvent être invités à évaluer le travail de leurs pairs. Cela favorise l'apprentissage collaboratif et permet aux apprenants de développer des compétences d'analyse et de critique constructives (Coombe, 2007).
- **Des évaluations automatiques:** elles sont souvent utilisées pour évaluer rapidement les réponses aux questions à choix multiples, les exercices de codage ou d'autres types de tâches. Les résultats sont généralement générés automatiquement sans intervention humaine (R. Bennett, 1998).
- **Des évaluations ipsatives:** cette méthode compare les performances actuelles d'un apprenant avec ses performances antérieures, mettant l'accent sur le progrès individuel plutôt que sur la comparaison avec les autres étudiants (Gipps & Farajnezhad, 2022).
- **Des évaluations adaptatives:** les technologies des EIAH permettent de créer des évaluations adaptatives qui ajustent la difficulté des questions en fonction des réponses de l'apprenant, offrant ainsi une mesure plus précise de ses compétences (VanLehn, 2011).

Toutefois, il faut noter que l'évaluation en éducation ne se limite pas à la mesure des connaissances académiques des apprenants. Elle englobe également l'évaluation des compétences non académiques, telles que les compétences sociales, émotionnelles et créatives. L'évaluation vise à mesurer la capacité des élèves à résoudre des problèmes, à travailler en équipe, à communiquer efficacement, à penser de manière critique et à faire preuve de créativité. Dans ce sens, l'évaluation dans le cadre des EIAH prend en compte la diversité des aptitudes des apprenants et cherche à promouvoir un apprentissage holistique et intégré.

Toutefois, pour être véritablement efficace, celle-ci doit répondre à des critères de validité, de fiabilité et d'équité. La validité fait référence à la capacité d'un outil d'évaluation à mesurer exactement ce qu'il est censé mesurer. Cela nécessite un alignement entre les évaluations et les objectifs d'apprentissage, afin que les résultats obtenus reflètent fidèlement les compétences et

les connaissances visées par le programme éducatif (Weideman, 2013). La fiabilité, quant à elle, se réfère à la cohérence et à la précision des résultats d'une évaluation. Un test est considéré comme fiable si les résultats sont similaires lorsqu'il est administré dans des conditions identiques ou très similaires à des moments différents (Crocker & Algina, 1986). Pour assurer cette notion de fiabilité, il est nécessaire de minimiser les sources d'erreur qui pourraient affecter les scores des étudiants, telles que les variations dans l'administration du test ou les biais de notation. Par exemple, l'utilisation de critères de notation bien définis peut aider à augmenter la fiabilité des évaluations (Shavelson & Webb, 1991). Dans le contexte de l'évaluation par des algorithmes d'intelligence artificielle, la question de la fiabilité devient encore plus complexe. Les modèles d'IA peuvent introduire des biais qui affectent la cohérence des résultats. Par exemple, un déséquilibre des classes dans les ensembles de données utilisés pour entraîner les algorithmes peut entraîner une évaluation inégale des apprenants (Akter et al., 2022). Si certaines classes de réponses sont sous-représentées, l'algorithme pourrait ne pas bien les reconnaître, ce qui affecterait la fiabilité de l'évaluation (H. He & E. A. Garcia, 2009). De plus, les biais algorithmiques peuvent découler des données d'entraînement qui reflètent des préjugés humains ou des inégalités systémiques. Par conséquent, la nécessité de corriger ces biais afin d'assurer la fiabilité des évaluations automatisées est plus qu'essentielle (Buolamwini & Gebru, 2018). L'intégration de techniques comme le rééquilibrage des classes dont nous discuterons plus en détail dans le chapitre suivant (chapitre III), peut également contribuer à améliorer la fiabilité des évaluations par IA. L'équité est une autre dimension essentielle de l'évaluation en éducation. Elle implique de prendre en compte les différences individuelles, les contextes culturels et les besoins spécifiques des apprenants, afin d'assurer des opportunités égales pour tous (Linn, 1993). Une évaluation équitable doit être conçue de manière à ne pas désavantager certains groupes d'apprenants.

3.2 Les Enjeux de l'Évaluation dans les EIAH : Limites et Solutions

Le choix des méthodes, des instruments et des techniques d'évaluation dépend en grande partie des objectifs assignés par le processus éducatif. Toutefois, les limites de l'évaluation dans les EIAH représentent un défi significatif à relever. Les évaluations dans ces contextes souffrent de multiples biais dont:

- Peu de moyens et de méthodes d'évaluations sont offerts par les EIAH. Ces moyens ont des limites notables:
    - **L'unicité du modèle d'évaluation**: les EIAH utilisent souvent un seul modèle d'évaluation, ne tenant pas compte de la diversité des approches pédagogiques et des profils d'apprenants (Håklev et al., 2017).
    - **Rigidité de la méthode employée**: les méthodes d'évaluation dans les EIAH sont souvent statiques et prédéfinies, avec des critères spécifiques qui ne permettent pas une adaptation dynamique aux besoins individuels des apprenants (Kay & Knaack, 2009).
    - **Format des résultats**: le format des résultats d'évaluation dans les EIAH nécessite une considération non seulement de la valeur associée à une compétence, mais aussi de l'ensemble du processus ayant conduit à ce résultat (Smith et al., 2023).
- **Méthodes d'évaluation souvent implicites**: les résultats sont parfois difficiles à interpréter en raison d'un manque de clarté et de transparence dans les méthodes d'évaluation (Durand, 2007). Cette ambiguïté peut être due à :
    - **Manque de clarté**: les méthodes d'évaluation ne sont pas toujours explicitement décrites, rendant difficile pour les enseignants et les apprenants de comprendre les critères et les processus utilisés pour générer les résultats.

- **Incapacité à englober le cheminement complet**: les évaluations ne capturent souvent pas l'ensemble du cheminement intellectuel de l'apprenant, se concentrant plutôt sur des résultats finaux sans prendre en compte les étapes intermédiaires et les efforts déployés (Baker & Siemens, 2014).

- **Absence de prise en compte des erreurs habituelles des apprenants**: les EIAH ne considèrent pas toujours les erreurs récurrentes des apprenants ou celles induites par l'exercice lui-même. Par conséquent, les apprenants peuvent interpréter leurs erreurs comme un échec personnel, générant de l'anxiété dans les phases initiales de l'apprentissage (Bouhineau, 2013).

En outre, l'évaluation dans les EIAH présente une autre série de limites, comprenant :

- **Problématique de la subjectivité**: l'évaluation implique une interprétation des performances de l'étudiant par l'évaluateur, ce qui peut être influencé par des préjugés personnels. Cette subjectivité peut affecter la justice et l'équité des résultats (R. E. Bennett, 2011).
- **Fiabilité**: la cohérence des mesures d'évaluation est une préoccupation majeure. Une évaluation fiable doit produire des résultats cohérents à travers différentes administrations et contextes, ce qui n'est pas toujours le cas dans les EIAH (Crocker & Algina, 1986).
- **Validité**: la précision et la justesse des résultats obtenus lors de l'évaluation sont importantes. Une évaluation valide doit mesurer ce qu'elle est censée mesurer, mais les EIAH peuvent parfois échouer à capturer les compétences et les connaissances visées (Weideman, 2013).

Le défi des EIAH réside dans l'assurance de la fiabilité et de la validité des évaluations assistées par ordinateur, tout en préservant l'intégrité de la capacité d'apprentissage de l'apprenant. Les stratégies envisagées pour surmonter les limitations de l'évaluation dans les EIAH incluent :

- **Diversification des méthodes d'évaluation utilisées**: en intégrant différentes approches, telles que les évaluations formatives et sommatives, les évaluations adaptatives, ainsi que les évaluations par pairs et auto-évaluations, il est possible de réduire les biais et d'obtenir une image plus complète des compétences et des connaissances des apprenants (R. E. Bennett, 2011).
- **Adoption de grilles d'évaluation pour garantir une évaluation cohérente**: les grilles d'évaluation standardisées permettent de rendre les critères d'évaluation explicites et transparents, ce qui augmente la cohérence et la fiabilité des évaluations (Jonsson & Svingby, 2007).

Ces approches visent à améliorer l'évaluation dans les EIAH en réduisant la subjectivité, en garantissant la fiabilité et la validité des évaluations, tout en favorisant un environnement d'apprentissage optimal pour les apprenants. D'ailleurs, d'autres chercheurs ont opté pour la scénarisation, qui désigne la création et la structuration de scénarios d'apprentissage détaillant les activités pédagogiques et évaluatives pour favoriser l'acquisition de connaissances (Koper & Tattersall, 2005). Cependant, cette approche présente également quelques inconvénients :

- **Élaboration de scénarios très complexes en taille et en entités**: la complexité des scénarios peut rendre leur mise en œuvre difficile et chronophage (Davinia et al., 2005).

- **Difficultés de réutilisation de scénarios très spécialisés**: les scénarios conçus pour des contextes spécifiques peuvent ne pas être facilement adaptables à d'autres situations ou apprenants (Vega-Gorgojo et al., 2006).
- **Spécification de l'évaluation non modifiable**: une fois l'activité opérationnalisée et démarrée, il est souvent impossible de modifier les critères d'évaluation, ce qui peut limiter la flexibilité nécessaire pour répondre aux besoins changeants des apprenants (Persico et al., 2010).

Pour remédier à cela, les chercheurs se sont intéressés à des scénarios distincts, jouant des scénarios pour isoler les problèmes et les corriger. Cette approche offre plusieurs possibilités mais ne permet pas toujours d'exprimer tous les scénarios possibles, et il reste difficile de définir le meilleur scénario (Shemshack & Spector, 2020; Wang & Hannafin, 2005). Dans ce contexte, la personnalisation de l'apprentissage émerge comme une stratégie prometteuse pour adresser ces limitations.

4 Personnalisation de l'Apprentissage

Il demeure une absence de consensus absolu, accompagnée de débats quant à la définition de la personnalisation. En effet, les nombreuses définitions proposées dans la littérature ne parviennent pas à englober tous les aspects qui caractérisent une personne ou l'évolution de sa personnalité. Ceci s'explique par le fait que les préférences, les besoins et les désirs individuels ou communautaires sont sujets à une légitime évolution. La notion même de "personnalité" est fluide et multidimensionnelle. Elle est influencée par une multitude de facteurs (génétiques, environnementaux, culturels, etc.) et évolue au cours de la vie. De plus, les besoins et les attentes des individus sont hétérogènes et dynamiques. Ils varient selon le contexte, les expériences et les objectifs de chacun.

Dans le domaine de l'éducation, la personnalisation vise à offrir des services afin de permettre à l'apprenant de vivre une expérience unique en lui adaptant des ressources qui correspondent à ses besoins, ses objectifs et son profil (Asif & Krogstie, 2012). Il est donc important de s'interroger sur comment définir la personnalisation? Quels sont les paramètres à considérer pour élaborer et mettre en pratique une personnalisation appropriée ? En outre, avant toute chose, comment évaluer la pertinence des informations proposées à l'apprenant ?

4.1 Définition de la Personnalisation

La personnalisation de l'apprentissage est couramment définie par l'adaptation des interactions, des cours, la découverte du contenu, l'ajustement du support de collaboration ou encore l'adaptation des évaluations (Baidada et al., 2019). Cette approche vise à répondre de manière ciblée aux besoins spécifiques des apprenants en tenant compte de leurs caractéristiques individuelles et contextuelles. Deux formes d'adaptation sont particulièrement mises en avant dans ce contexte :

- L'adaptation de la présentation, qui consiste à moduler la manière dont les informations sont affichées et transmises à l'apprenant,
- L'adaptation de la navigation, qui vise à orienter l'apprenant de manière fluide et personnalisée à travers les contenus pédagogiques (Khribi et al., 2008).

Cependant, il convient de noter que les concepts de personnalisation et d'individualisation restent souvent confus et sont fréquemment utilisés de manière interchangeable. Cette confusion découle de leur proximité sémantique, car les deux termes renvoient respectivement aux notions de personne et d'individu, lesquelles partagent des significations connexes mais

distinctes (Grandbastien & Nowakowski, 2014). La personnalisation, dans son acception la plus large, tend à se concentrer sur l'adaptation des processus éducatifs en fonction des préférences et besoins généraux des apprenants, souvent sur la base de données groupées ou de modèles prédictifs. En revanche, l'individualisation va plus loin en visant une approche encore plus fine et centrée sur chaque apprenant pris isolément, en prenant en compte ses trajectoires d'apprentissage uniques, ses interactions spécifiques avec le contenu, ainsi que ses progrès singuliers.

Ainsi, bien que la personnalisation et l'individualisation de l'apprentissage partagent des objectifs similaires, à savoir l'optimisation des parcours d'apprentissage pour maximiser l'efficacité pédagogique, leur application et leur portée peuvent varier considérablement. L'ambiguïté persistante entre ces deux concepts souligne la nécessité de clarifier leurs implications respectives dans le développement de systèmes éducatifs intelligents (Chachoua, 2019).

4.2 La Personnalisation dans le Domaine de L'Education EIAH

L'éducation est un domaine riche et diversifié, englobant une variété d'éléments pouvant être personnalisés ou recommandés, ainsi que différents types d'actions sur ces éléments dans le but de parvenir à un objectif éducatif (Santos & Boticario, 2010). En effet, la finalité de la personnalisation dans le domaine de l'éducation réside dans l'amélioration de l'efficacité de l'apprentissage, offrant aux apprenants des parcours correspondant à leurs besoins spécifiques et répondant à leurs exigences (Elghibari & Elouahbi, 2015; Kamsa et al., 2018; Kurilovas et al., 2015; Limongelli et al., 2008; Lin et al., 2013). En d'autres termes, cette approche doit être guidée par des objectifs pédagogiques plutôt que par les préférences de l'apprenant, contrairement à ce qui prévaut dans le domaine du marketing à titre d'exemple (Santos & Boticario, 2010).

Les auteurs (Almeida & Winfield, 2012), figures de proue en sciences cognitives, affirment que « les enseignants doivent devenir des experts de la dynamique cérébrale de leurs élèves ». Les outils EIAH offrent aux apprenants la possibilité d'acquérir des connaissances à tout moment et en tout lieu, en leur fournissant des informations adaptées à leurs besoins (Premlatha et al., 2016a). Toutefois, pour répondre efficacement aux exigences individuelles, les systèmes éducatifs personnalisés doivent intégrer deux composantes essentielles. Premièrement, il faut reconnaître que tous les apprenants ne peuvent être traités de manière uniforme en raison de leurs divergences comportementales et de leurs caractéristiques intellectuelles singulières. En effet, chaque apprenant se distingue par ses attitudes, aptitudes et son niveau de motivation, qui influencent de manière significative son processus d'apprentissage (Chen, 2008). Deuxièmement, ce processus d'apprentissage ne saurait être prédit de manière statique (Baylari & Montazer, 2009; Premlatha et al., 2016a). À titre d'exemple, les systèmes éducatifs en ligne, qui présentent fréquemment des structures de domaine et de contenu sous une forme statique, omettent souvent de prendre en compte les objectifs individuels, les expériences préalables, les connaissances acquises et les compétences des apprenants (Dahalan & Hussain, 2010). Par conséquent, il est indispensable de déployer des efforts visant à intégrer des fonctionnalités dynamiques au sein de ces systèmes personnalisés. En effet, certains systèmes négligent la corrélation entre les aptitudes de l'apprenant et le niveau de difficulté du contenu qui lui est proposé (Chen, 2008). De plus, une fois les évaluations terminées, les apprenants sont rarement réévalués sur les mêmes compétences afin de vérifier s'ils ont tiré des enseignements de leurs erreurs ou s'ils conservent les acquis précédemment obtenus (Kramer et al., 2023; Stiggins & Chappuis, 2005).

4.3 Personnalisation des Fonctionnalités des EIAH

De nombreux travaux se sont concentrés sur la personnalisation des fonctionnalités des EIAH, mettant en lumière non seulement l'importance de l'engagement de l'apprenant (Premlatha et al., 2016b), mais également la manière d'observer et d'exploiter les données générées par les apprenants (Clerc et al., 2015). Plus précisément, ces recherches (Baylari & Montazer, 2009) s'intéressent aux caractéristiques, aux préférences et aux besoins spécifiques des apprenants.

4.3.1 Méthodes de Collecte et D'Analyse des Données Apprenants

Un système éducatif personnalisé doit être capable de collecter, d'analyser et d'exploiter les données relatives aux actions de l'apprenant (traces), ainsi qu'à ses préférences et ses besoins. Ces traces servent à améliorer dynamiquement le processus d'apprentissage (Johanes & Lagerstrom, 2017; Premlatha et al., 2016b). Traditionnellement, deux méthodes principales sont utilisées pour recueillir des informations sur l'apprenant : la première consiste à recourir à des tests directs, tels que des questionnaires ou des évaluations spécifiques, permettant d'obtenir des informations explicites directement auprès de l'apprenant. La seconde méthode repose sur l'analyse des interactions de l'apprenant avec le système éducatif, c'est-à-dire l'exploitation des traces d'apprentissage laissées au fil de son parcours dans l'EIAH (Klašnja-Milićević et al., 2011). Cette dernière approche implique souvent l'utilisation de techniques avancées telles que la fouille de données (data mining) pour interpréter ces données (Kostadinov, 2003), ou l'application de modèles multidimensionnels pour affiner le modèle de l'apprenant, comme l'ont fait (Baylari & Montazer, 2009) en intégrant ces éléments dans le développement de systèmes éducatifs personnalisés.

Néanmoins, il serait réducteur de limiter la collecte d'informations sur l'apprenant à ces seules méthodes. Des recherches récentes explorent d'autres voies, notamment en intégrant des capteurs biométriques, des analyses émotionnelles, ou encore des algorithmes d'apprentissage automatique pour mieux comprendre les états cognitifs et émotionnels des apprenants en temps réel (Baker & Siemens, 2014). Par ailleurs, les travaux de (Vassileva, 2009) soulignent l'importance de prendre en compte le contexte social de l'apprentissage, arguant que la présence sociale est un facteur clé de la réussite en e-formation (Grandbastien & Nowakowski, 2014). Cela ouvre la porte à l'exploration de nouvelles dimensions, telles que l'analyse des interactions sociales de l'apprenant au sein de communautés d'apprentissage, et la prise en compte de la dynamique collective pour enrichir l'expérience éducative.

4.3.2 Méthodologies De Personnalisation Utilisées dans le Domaine de l'Education

L'apprentissage personnalisé suppose une amélioration qualitative de l'enseignement, capable de s'adapter aux divers besoins des utilisateurs afin de compléter leur processus d'apprentissage (Paneva & Zhelev, 2007). L'idée maîtresse réside dans l'atteinte d'un objectif commun : offrir aux utilisateurs les ressources dont ils ont besoin sans qu'ils aient à les solliciter explicitement (Clerc et al., 2015; Kurilovas et al., 2015). Cette approche peut être envisagée sous deux perspectives : celle de l'enseignant, qui repose sur son expérience et son intuition, et celle de l'apprenant, qui pourrait parfois ressentir une certaine frustration face à ce processus (Kurilovas et al., 2015).

Lorsque la personnalisation est perçue comme un moyen de présenter des informations et des services spécifiquement adaptés à chaque individu ou communauté, l'enjeu fondamental réside dans la capacité à identifier, filtrer et modéliser ces informations de manière à ce qu'elles correspondent parfaitement aux préférences de l'utilisateur. Ainsi, l'application des systèmes personnalisés peut être analysée sous quatre angles distincts:

1) Quelle composante du système doit être personnalisée ?
2) Quelles informations le système utilise-t-il pour la personnalisation ?

3) Comment le système recueille-t-il les données nécessaires à la personnalisation?
4) Comment modéliser un ensemble de données variées pour une personnalisation optimale ?

Nous avons tenté de brosser un tableau (Tableau 1.1) des différentes méthodes de personnalisation utilisées dans la littérature de l'éducation. Les techniques de fouilles de données sont classées selon quatre catégories : classification, association, regroupement, et l'extraction de modèles séquentiels. Par ailleurs, certains de ces travaux ont centré leur attention sur l'approche de (Felder and Silverman) qui classifie les styles d'apprentissage comme étant actif ou réfléchi, sensoriel ou intuitif, visuel ou verbal, et séquentiel ou global.

Tableau 1.1 Méthodologies de Personnalisation Utilisées dans le Domaine de L'Education

Recherche	Technologie	Méthode utilisée
(Premlatha et al., 2016b)	Réseaux de croyances bayésiens, Arbres de décision, Bayesien Belief Network (BBN)	- Utilisation des traces collectées du comportement de l'apprenant en se basant sur ses caractéristiques et ses besoins à travers les navigations logs. - Classification des apprenants via les réseaux de croyances bayésiens et techniques d'arbres de décision pour construire un profil basé sur la performance et le niveau de savoir de l'apprenant. - Création d'un BBN pour la classification des apprenants.
(Kamsa et al., 2018)	Algorithme ACO, Agent optimisateur, ECOD	- Optimisation du chemin d'apprentissage en tenant compte des besoins et capacités cognitives des apprenants ainsi que de l'expérience collective de leurs pairs. - Utilisation d'un agent optimisateur pour fournir un parcours personnalisé et optimal basé sur l'algorithme ACO (Ant Colony Optimization). • Développement d'une approche ECOD (e-Learning Custom Optimized and Dynamic) pour améliorer l'acquisition des apprenants.
(Baidada et al., 2019)	Filtrage collaboratif, Filtrage du contenu	- Approche hybride de recommandation prenant en compte les préférences des apprenants ainsi que celles de leurs camarades dans un groupe social. - Combinaison de filtrage du contenu et filtrage collaboratif pour améliorer la pertinence des recommandations.
(Abed et al., 2020)	Apprentissage en profondeur	- Étude du data mining dans la personnalisation web utilisant un modèle d'apprentissage profond hybride basé sur le modèle de réseau neuronal à propagation inverse.
(Patel & Sajja, 2021)	Apprentissage par renforcement, système multi-agents	- Système d'apprentissage personnalisé utilisant l'approche du reinforcement learning pour ajuster le niveau de difficulté en fonction de la compréhension des étudiants.
(Jin & Kim, 2023)	GPT	- Système d'eLearning personnalisé pour les langages de programmation, intégrant GPT pour la

		génération dynamique de contenu et la personnalisation des parcours d'apprentissage.

En définitive, la réelle personnalisation requiert un profil de l'utilisateur contenant des informations sur ses préférences, ses buts, et son historique afin d'être stockées et utilisées par le système (Paneva & Zhelev, 2007). Elle se traduit par des services intelligents qui se souviennent de qui est l'apprenant, de ce qu'il a fait, de ce qu'il est entrain de faire et dans quel contexte il se situe.

4.3.3 Évolution Pédagogique Digitale: Concilier Personnalisation et Respect de la Vie Privée

Les technologies numériques portent un potentiel considérable pour le développement pédagogique, mais pour en tirer les valeurs pédagogiques escomptées, elles nécessitent d'être encadrées par des dispositifs pédagogiques basés sur des méthodes incitatives et interactives. Ces méthodes impliquent de nouveaux rôles pour les acteurs, enseignants et étudiants, et visent au développement de leurs compétences (Pardo & Siemens, 2014). De manière générale, la personnalisation dans n'importe quel domaine d'application implique des choix, de la flexibilité et du contrôle (Woolf, 2008). Toutefois, il est essentiel de considérer quelles informations les utilisateurs sont prêts à partager avec les systèmes informatiques et les différentes plateformes. Les solutions proposées soulèvent la question de la vie privée des utilisateurs, car les données traitées côté serveur peuvent enfreindre les législations en vigueur (Asif & Krogstie, 2012).

La mise en œuvre de la personnalisation doit être équilibrée avec le respect de la vie privée des utilisateurs. Par exemple, des recherches montrent que les apprenants sont souvent réticents à partager des informations personnelles, ce qui peut limiter l'efficacité des systèmes personnalisés (Pardo & Siemens, 2014). De plus, la personnalisation peut être enrichie par des techniques de contextualisation, qui tiennent compte des circonstances spécifiques et des préférences de l'utilisateur au moment de l'interaction (Dabbagh & Kitsantas, 2012). Cependant, il demeure important d'établir des mécanismes de protection des données (Voigt & Bussche, 2017).

5 La Compréhension

La compréhension est souvent définie comme un processus cognitif complexe qui permet à l'individu de donner un sens à un concept ou une situation. Ce processus a été largement étudié par des chercheurs en éducation et en psychologie cognitive. Selon le modèle de la compréhension développé par (Kintsch, 1998), la compréhension repose sur la construction de représentations mentales qui intègrent les nouvelles informations aux connaissances préexistantes. Cette théorie met en avant le rôle crucial des schémas cognitifs, qui facilitent l'organisation et l'intégration de nouvelles informations dans un cadre mental cohérent (Gernsbacher, 1999). Plus particulièrement, (Bransford, 2000) soulignent que la compréhension véritable implique la capacité à transférer les connaissances à de nouveaux contextes, un processus que les chercheurs appellent le transfert de l'apprentissage. Ce transfert n'est possible que lorsque les apprenants ont développé une compréhension profonde des concepts, leur permettant de les appliquer de manière flexible et innovante dans des situations variées.

De ce fait, atteindre une compréhension approfondie exige le développement de compétences cognitives telles que la pensée critique, la concentration, et la capacité à interroger ses propres idées (Akiguet-Bakong, 2008). En 1956, Bloom propose une taxonomie des objectifs éducatifs, il place la compréhension juste après la mémorisation, mais avant l'application, l'analyse, la synthèse et l'évaluation. Cela illustre la nécessité de la compréhension comme base pour les niveaux supérieurs de la cognition. Les recherches de (Mirzajani et al., 2016) vont plus loin en

démontrant que la compréhension n'est pas simplement un produit passif de l'exposition à l'information, mais un processus actif qui nécessite la réflexion critique et l'engagement intellectuel. De même, (Perkins, 1992) a développé le concept de "compréhension en profondeur" qui implique non seulement la connaissance des faits, mais aussi la capacité à expliquer, interpréter, et appliquer les connaissances de manière créative et innovante.

La compréhension est également influencée par les expériences antérieures, les croyances, les valeurs, et le contexte culturel de l'apprenant. (VYGOTSKY, 1978b) a fortement contribué à cette idée avec sa théorie socio-culturelle de l'apprentissage, où il explique que le développement cognitif de l'individu est profondément enraciné dans les interactions sociales et culturelles. Cette théorie soutient que la compréhension est façonnée par le contexte social et les interactions avec les autres, ce qui souligne l'importance de la diversité et de l'inclusion dans le processus d'apprentissage. Ainsi, ces études montrent clairement que la compréhension est un processus dynamique et central dans l'apprentissage. Pour garantir une évaluation juste, il est nécessaire de vérifier que les apprenants ont bien assimilé ce qui leur est demandé avant toute évaluation.

5.1 Résolution de Problèmes et Compréhension

La résolution de problèmes est un processus cognitif complexe qui implique la manipulation de l'information pour atteindre un objectif spécifique. Selon (Jonassen, 2010), ce processus peut être défini comme l'engagement cognitif visant à surmonter des obstacles conceptuels pour atteindre un but qui ne peut être atteint par des procédures routinières. Ce processus inclut l'identification du problème, la génération de solutions possibles, la sélection de la meilleure solution, et l'évaluation de son efficacité. Pour réussir ces étapes, une compréhension profonde et précise des éléments du problème est essentielle. Des recherches, telles que celles de (Kalyuga, 2007; Lishinski et al., 2016), montrent que les experts en résolution de problèmes se distinguent des novices par leur capacité à structurer et à comprendre profondément le problème avant d'envisager une solution. Ils construisent des représentations mentales complexes et bien organisées qui leur permettent de reconnaître les modèles pertinents et de relier les nouvelles informations à leurs connaissances antérieures. Cette compétence est élémentaire, car elle permet aux experts de filtrer les informations non pertinentes et de se concentrer sur les aspects essentiels du problème.

Une résolution efficiente des problèmes requiert un ensemble d'étapes méthodiques, incluant l'analyse circonstanciée, la détermination précise du problème, l'élaboration de solutions envisageables, une évaluation précise de ces solutions, et enfin, la mise en œuvre de la solution optimale. Cependant, cette capacité à résoudre efficacement des problèmes repose fondamentalement sur la faculté de compréhension, car elle exige une appréhension profonde du problème en question. Les auteurs (Sweller et al., 2019) ont démontré que les individus dotés de compétences avancées en compréhension sont capables d'identifier les aspects essentiels d'un problème, de formuler des déductions appropriées, et de concevoir des solutions efficaces. Par conséquent, les compétences en compréhension jouent un rôle clé dans le processus de résolution de problèmes.

Plus particulièrement, un modèle théorique largement accepté qui relie la compréhension à la résolution de problèmes est la théorie des schémas, telle que développée par (Sweller, 1988). Selon cette théorie, la résolution de problèmes repose sur la capacité à accéder et à appliquer des schémas cognitifs : des structures mentales qui organisent les informations et guident la reconnaissance des modèles. Ces schémas sont construits à travers l'apprentissage et sont essentiels pour naviguer efficacement dans des situations complexes. (Sweller, 1988) a montré que les novices en résolution de problèmes, qui manquent de schémas bien développés, sont plus susceptibles de se concentrer sur des détails superficiels plutôt que sur les relations

structurelles profondes qui définissent le problème. En revanche, les experts utilisent leurs schémas pour simplifier la complexité du problème et identifier rapidement des solutions efficaces. Ce phénomène a été confirmé par plusieurs études, notamment celle de (Akiguet-Bakong, 2008). Cette étude a impliqué 40 participants, sélectionnés parmi les élèves les plus performants et les moins performants en arithmétique, répartis équitablement en deux groupes équilibrés en termes de sexe et de niveau en arithmétique. Quatre problèmes arithmétiques différents ont été présentés, et des questionnaires ont été utilisés pour guider les sujets dans leur raisonnement avant la résolution de chaque problème. Les résultats de cette étude indiquent que la qualité du traitement de l'énoncé est un facteur déterminant dans la résolution des problèmes arithmétiques. L'utilisation d'un support mémoire a démontré que des capacités de mémoire de travail renforcées améliorent la compréhension et la résolution des problèmes. De plus, cette étude souligne la différence entre les connaissances déclaratives (utilisées pour la compréhension) et les connaissances procédurales (impliquées dans la résolution), mettant en lumière que les sujets ne possèdent pas le même niveau de compétence dans ces domaines.

L'enseignement basé sur la résolution de problèmes, souvent appelé apprentissage par problèmes (Problem-Based Learning), a été largement étudié pour ses effets positifs sur la compréhension des étudiants. (Hmelo-Silver, 2004) a montré que l'apprentissage par problèmes non seulement améliore la capacité des étudiants à résoudre des problèmes, mais renforce également leur compréhension des concepts sous-jacents en les obligeant à réfléchir de manière critique et à intégrer les connaissances de manière cohérente. Ce type d'apprentissage favorise le développement de compétences métacognitives, telles que la capacité à évaluer et à ajuster sa propre compréhension au cours du processus de résolution de problèmes. Une étude menée par (Anthonysamy, 2021) a révélé que les étudiants ayant des compétences métacognitives plus élevées sont non seulement meilleurs pour résoudre des problèmes complexes, mais aussi plus capables de transférer ces compétences à de nouveaux domaines. Cela renforce l'idée que la compréhension et la métacognition sont inextricablement liées dans le processus de résolution de problèmes.

5.2 L'Impact de la Compréhension sur les Performances des Apprenants

Plusieurs études confirment l'importance de la compréhension dans l'amélioration des performances académiques. Par exemple, (Hattie, 2009) a démontré que les stratégies d'enseignement favorisant une compréhension profonde, telles que l'apprentissage par la découverte et l'enseignement explicite de la métacognition, ont un impact significatif sur les résultats scolaires des élèves. Une compréhension solide permet aux apprenants de saisir le sens et les implications des concepts, facilitant ainsi l'application des connaissances dans des situations nouvelles. Une étude menée par (Van Der Stel & Veenman, 2014) a montré que les étudiants ayant des compétences de compréhension avancées réussissent mieux dans les tâches de résolution de problèmes complexes que ceux ayant une compréhension limitée.

La compréhension joue également un rôle clé dans l'engagement et la motivation des apprenants. Les étudiants qui comprennent bien les objectifs d'apprentissage sont plus susceptibles de trouver du sens dans ce qu'ils apprennent, ce qui augmente leur motivation intrinsèque. (Schunk & Zimmerman, 2007) ont démontré que les élèves qui relient les objectifs d'apprentissage à leurs propres intérêts sont plus engagés et persévèrent davantage face aux défis académiques. Cette motivation accrue se traduit par une amélioration des performances scolaires.

Un des effets les plus durables d'une compréhension profonde est la facilitation de l'apprentissage à long terme. Selon (Fiorella & Mayer, 2016), un apprentissage durable se produit lorsque les apprenants sont capables de transférer et d'appliquer les connaissances acquises à de nouveaux contextes, même longtemps après l'instruction initiale. Les stratégies

d'enseignement qui encouragent la métacognition et l'autorégulation sont particulièrement efficaces pour promouvoir ce type d'apprentissage. Par exemple, (Panadero et al., 2017) ont montré que les apprenants utilisant des stratégies métacognitives pour surveiller et réguler leur compréhension sont plus susceptibles de retenir et de transférer leurs connaissances, ce qui améliore leurs performances à long terme.

Cependant, la compréhension peut être entravée par divers obstacles, tels que la surcharge cognitive, les malentendus conceptuels et les lacunes dans les connaissances préalables (Chi & Wylie, 2014).en effet, (Sweller et al., 2019) a démontré que la surcharge cognitive, causée par un excès d'informations à traiter simultanément, peut nuire à la compréhension en empêchant les apprenants de former des représentations mentales cohérentes. Cela peut affecter négativement les résultats académiques, d'où l'importance de renforcer les connaissances de base des apprenants avant d'introduire des concepts plus complexes (Wright, 2015).

6 Conclusion

Les EIAH marquent un progrès notable dans le domaine de l'apprentissage en offrant une réponse aux défis actuels de l'éducation grâce à leur adaptabilité et leur personnalisation remarquables qui favorisent une expérience d'apprentissage plus interactive et sur mesure pour chaque apprenant. Grâce aux progrès technologiques comme l'intelligence artificielle, ces systèmes ouvrent des portails pour améliorer la qualité de l'instruction tout en la rendant disponible à un plus large public. Cependant, il est également important de prendre en compte les défis qu'ils impliquent, notamment en ce qui concerne l'accès inégal aux technologies et le maintien de l'échange humain dans les processus d'apprentissage. Pour tirer le meilleur parti de leurs capacités, il est préférable de continuer à explorer les moyens d'améliorer ces systèmes afin d'offrir une personnalisation plus poussée et une expérience d'apprentissage authentiquement inclusive.

Les EIAH se révèlent être un outil efficace pour accompagner les évolutions de l'éducation en mettant en avant la nécessité d'une approche pluridisciplinaire pour surmonter les défis liés à leur mise en place. Dans le chapitre suivant, nous aborderons l'un des domaines d'utilisation clé de ces environnements: l'algorithmique qui revêt une importance capitale dans le développement des compétences en programmation et pose des défis bien spécifiques pour l'apprentissage.

CHAPITRE II : DOMAINE D'APPLICATION LA PROGRAMMATION

1 Introduction à la Programmation

Au XXIe siècle, la programmation est devenue une forme d'alphabétisation indispensable, reconnue pour son rôle déterminant dans l'amélioration des capacités de réflexion et de résolution de problèmes (Ling Chean, Ban Ho, et Chai 2018). En effet, la programmation peut significativement renforcer la cognition, incluant des aspects tels que les connaissances, l'attention, l'analyse, l'évaluation, la création, la prise de décision et le raisonnement logique (Cooke et Schvaneveldt 1988; Guibert et Girard 2003; Malliarakis, Satratzemi, et Xinogalos 2013).

En tant qu'activité de résolution de problèmes, la programmation nécessite une série d'étapes méthodiques, dont la formulation claire du problème pour en comprendre pleinement les exigences. Il est également essentiel d'identifier les données nécessaires (entrées) et les résultats attendus (sorties) (Corritore et Wiedenbeck 1991; Eckerdal 2009; Hooshyar et al. 2015). Alors que la programmation favorise le développement des compétences cognitives, la pensée algorithmique apporte une approche structurée et précise à la résolution de problèmes. En effet, elle implique l'acquisition de compétences variées telles que la décomposition fonctionnelle, la conception descendante, la répétition, la manipulation des données et la différenciation entre algorithme et programme (Parmentier 2018).

Ces compétences constituent un fondement stable pour réussir dans le domaine de la programmation, bien que les novices rencontrent de nombreux défis. Si certains apprennent rapidement à sélectionner des structures ou à manipuler des variables, d'autres éprouvent des difficultés avec des concepts plus avancés comme la récursion ou les pointeurs. Cette variabilité des compétences met en lumière l'importance d'une compréhension équilibrée et approfondie des concepts clés pour bâtir une base solide en programmation (Corritore et Wiedenbeck 1991; Estupiñán et Jesús & Martín Estévez 2019).

Cette section se consacrera donc à l'exploration détaillée des mécanismes inhérents à la pensée algorithmique et à la compréhension des programmes informatiques. Nous examinerons les fondements de la programmation, les compétences nécessaires, les enjeux et impacts de l'apprentissage de la programmation, ainsi que les défis spécifiques auxquels les novices sont confrontés, y compris leurs parcours d'apprentissage, les défis pédagogiques et les solutions envisageables. De plus, nous analyserons les objectifs pédagogiques pour la pensée algorithmique, les stratégies de compréhension, la variabilité des compétences en programmation et les approches pédagogiques pour faciliter cette compréhension. Enfin, nous discuterons des modèles cognitifs utilisés pour l'évaluation en programmation, afin d'éclairer les voies de l'apprentissage dans ce domaine.

1.1 La Programmation

La programmation est une compétence fondamentale dans notre société contemporaine, caractérisée par une dépendance croissante aux technologies numériques. Elle englobe non seulement l'écriture de code, mais aussi la conception de solutions logicielles pour des problèmes complexes. La programmation peut être définie comme un processus de construction de correspondances entre le domaine du problème et le domaine de la programmation, en passant par plusieurs domaines intermédiaires. Ce processus nécessite une compréhension profonde des abstractions et des transformations conceptuelles nécessaires pour traduire un problème en une solution algorithmique et code exécutable (Pennington 1987; Winslow 1996).

D'un point de vue technique, la programmation peut être définie comme l'art de formuler des instructions destinées à un ordinateur afin qu'il exécute des tâches spécifiques. Ce processus consiste à traduire des concepts, des problèmes ou des procédures issus du monde réel en un langage compréhensible par la machine (Van-Roy et Haridi 2004). La programmation s'appuie sur des langages de programmation tels que Python, Java, et C++, qui fournissent des structures syntaxiques et sémantiques contribuant à l'expression logique des instructions.

Ces langages de programmation sont dotés de règles précises qui permettent aux programmeurs de développer des algorithmes adaptés à une grande variété de problématiques. Par exemple, les structures de contrôle de flux, telles que les boucles et les conditions, permettent de définir le comportement d'un programme en fonction de diverses situations (Aho et Aho 2007). De plus, les abstractions offertes par ces langages, telles que les fonctions, les classes et les modules, facilitent la gestion et permettent la réutilisation et la modularité du code. Ainsi, grâce à ces outils, les apprenants peuvent concevoir et implémenter des algorithmes pour résoudre des problèmes allant du traitement de données à la création d'applications interactives. Par exemple, Python est souvent utilisé pour l'analyse de données et l'apprentissage automatique, tandis que Java est privilégié pour les applications d'entreprise et Android, et C++ pour les systèmes nécessitant des performances optimisées comme les jeux vidéo et les systèmes embarqués (Stroustrup 2013).

Cette section explore les bases de la programmation, les compétences nécessaires, les enjeux et l'impact de cette discipline dans l'éducation des apprentis programmeurs.

1.2 Processus de Conception d'un Programme

Le processus de conception d'un programme est une démarche structurée qui commence par une analyse et une modélisation du problème, suivies de l'écriture et de la mise en œuvre du code tout en tenant compte des contraintes et des spécificités de l'environnement d'exécution. Cette approche séquentielle permet de développer des solutions efficaces et se déroule généralement selon les étapes suivantes :

1. Analyse approfondie et modélisation précise de la tâche à accomplir :
    a. Définition des objectifs et des exigences: la première étape consiste à définir clairement les objectifs du programme. Cela implique de comprendre ce que le programme doit accomplir, quelles sont les fonctionnalités requises, et quelles sont les contraintes éventuelles (temps, ressources, etc.). Les apprenants doivent poser des questions fondamentales telles que "Quoi faire ?" et "Comment le faire?" (Pressman et Maxim 2014).
    b. Décomposition de la tâche: une fois les objectifs définis, il est essentiel de décomposer la tâche complexe en sous-tâches plus gérables. Cette approche permet de traiter chaque partie du problème indépendamment, facilitant ainsi la conception et la mise en œuvre du programme.
    c. Modélisation et abstraction: la modélisation implique la création de représentations abstraites du problème à résoudre. Cela peut inclure des diagrammes de flux, des schémas de base de données, ou des diagrammes UML (Unified Modeling Language) pour visualiser les différents composants du programme et leurs interactions. L'abstraction permet de simplifier les aspects complexes du problème en se concentrant sur les éléments essentiels (Booch, Rumbaugh, et Jacobson 1999).
    d. Compétence en résolution de problèmes: les apprenants doivent développer une compétence forte en résolution de problèmes pour identifier les solutions possibles et choisir la plus appropriée. Cela nécessite une pensée critique et

analytique pour évaluer les différentes approches et anticiper les défis potentiels (Jonassen 2000).

2. Réalisation du programme en mettant en œuvre le code conçu :
    a. Écriture du code: une fois que le plan et la modélisation sont en place, la prochaine étape est l'écriture du code. Cela implique de traduire les solutions conçues en instructions de programmation.
    b. Compréhension des contraintes de l'environnement d'exécution: il est essentiel de comprendre les contraintes spécifiques de l'environnement dans lequel le programme sera exécuté. Cela inclut la connaissance des ressources matérielles disponibles, comme la mémoire et le processeur, ainsi que des contraintes de temps d'exécution (Hennessy et Patterson 2011).
    c. Respect des règles de syntaxe et de sémantique: chaque langage de programmation a ses propres règles de syntaxe (la structure du code) et de sémantique (le sens du code). Il est important de maîtriser ces règles pour éviter les erreurs de compilation et les bogues. Les apprenants doivent également être familiers avec les bibliothèques et les frameworks pertinents qui peuvent simplifier le développement (Stroustrup 2013).
    d. Gestion des erreurs potentielles: lors de l'exécution d'un programme, diverses erreurs peuvent survenir, telles que des erreurs de syntaxe, des erreurs d'exécution, et des erreurs logiques. Les apprenants doivent apprendre à détecter, diagnostiquer et corriger ces erreurs (Zeller 2005).

1.3 De la Compréhension du Domaine à la Syntaxe du Code : Les Fondamentaux de la Programmation

Comme susmentionné, la programmation informatique est en effet un processus complexe qui nécessite la maîtrise de plusieurs sous-tâches, chacune requérant l'application de connaissances spécialisées (Guibert et Girard 2003). Avant tout, il faut posséder une compréhension approfondie du domaine réel du problème à résoudre. Cette connaissance permet de planifier une solution sous forme de code informatique, en utilisant des méthodes spécifiques au domaine en question (Hu 2011; Wing 2008). Une compréhension détaillée de la syntaxe et de la sémantique d'un langage de programmation est également indispensable. Enfin, la maîtrise des stratégies de conception de programmes est essentielle pour résoudre des problèmes, même de complexité modérée. Ces compétences sont fondamentales pour toute tâche, qu'il s'agisse de composer, de déboguer, de modifier ou de documenter un programme. L'élément clé pour chaque tâche est la compréhension du programme lui-même (Selby et al. 2010; Wing 2006).

Comprendre un programme implique de reconstruire des correspondances, ce qui permet d'accomplir des tâches telles que la modification ou la documentation du programme (Soloway et Ehrlich 1984). La réussite de cette compréhension repose en partie sur la capacité à extraire et interpréter les diverses informations contenues dans le programme. Ces informations incluent, mais ne se limitent pas à, la syntaxe, la sémantique, la structure du code et les intentions du programmeur (B. Adelson et E. Soloway 1985; Brooks 1983). Chaque type d'information joue un rôle décisif dans la compréhension efficace d'un programme. Par exemple, la structure syntaxique fournit des indices sur la logique du programme, tandis que la sémantique révèle l'intention fonctionnelle derrière le code. Il est donc essentiel de saisir les relations entre ces différents types d'informations pour obtenir une vision cohérente du programme (A. Von Mayrhauser et A. M. Vans 1995).

En d’autres termes, la compréhension des programmes informatiques consiste à identifier les parties essentielles d’un programme et à déduire les relations qui les unissent. Cela implique non seulement une analyse détaillée des éléments constitutifs du programme, mais aussi une appréciation de la manière dont ces éléments interagissent pour produire le comportement souhaité (Storey 2006). Cependant, cette capacité à comprendre les programmes informatiques n'est pas innée et représente un véritable défi pour les novices en programmation.

2 Les Défis du Novice en Programmation

Un novice en programmation est une personne qui débute dans l'apprentissage de la programmation informatique. Il peut s'agir d'un étudiant en informatique, d'un autodidacte ou de toute personne cherchant à acquérir des compétences en programmation pour diverses raisons professionnelles ou personnelles. Ce débutant est souvent confronté à de nouveaux langages de programmation, à des concepts abstraits tels que les variables, les boucles et les conditions, ainsi qu'à la logique et à la structure des algorithmes. En outre, il doit faire face aux défis liés à l'environnement de développement, à la gestion des erreurs et au débogage des programmes (Chiu et Huang 2015; Eckerdal 2009; Hannebauer, Hesenius, et Gruhn 2018; Lahtinen, Kirsti, et Hannu-Matti 2005).

2.1 Parcours d’Apprentissage du Novice en Programmation

Le parcours d'un novice en programmation se caractérise généralement par une courbe d'apprentissage progressive, comme le soulignent les travaux de (Sim et Lau 2018). Ce cheminement se divise fréquemment en plusieurs étapes distinctes.

Dans un premier temps, les novices entreprennent des exercices élémentaires et des exemples pratiques afin de se familiariser avec les concepts fondamentaux, tels que les variables, les structures de contrôle (incluant les boucles et les conditions), et les types de données. Cette étape est capitale pour établir une base solide en programmation. Les outils pédagogiques interactifs, tels que les environnements de développement intégrés (IDEs) simplifiés, les EIAH dédiés à la programmation, les jeux sérieux et les tutoriels en ligne, jouent un rôle déterminant à ce stade. Les recherches (Jolivet 2018; Lahtinen, Kirsti, et al. 2005) ont démontré qu’une approche centrée sur les problèmes concrets et les exercices pratiques peut améliorer de manière significative la compréhension et la rétention des concepts de base.

À mesure que le novice progresse, il est confronté à des projets d’étude plus complexes et à des problèmes de programmation réels. Cette exposition permet de renforcer ses compétences et sa compréhension des concepts plus avancés. Les projets pratiques, tels que la création de petites applications ou la résolution de problèmes algorithmiques moyennement complexes, s'avèrent particulièrement bénéfiques. Plusieurs chercheurs tels que (Cooke et Schvaneveldt 1988; Corritore et Wiedenbeck 1991; Futschek et Moschitz 2010; Kanaki et al. 2022) ont mis en évidence l'importance de la pratique régulière et de l'application concrète des connaissances acquises pour consolider les compétences en programmation.

Cependant, la phase initiale d'apprentissage peut être marquée par une certaine frustration et des difficultés, dues notamment à la complexité croissante des concepts et aux erreurs fréquentes de syntaxe ou de logique. Il est donc essentiel de développer des outils pédagogiques appropriés et de fournir un soutien adéquat. Des études ont montré que le soutien par les pairs, les forums de discussion et les sessions de mentorat peuvent aider à surmonter ces obstacles (Robins, Rountree, et Rountree 2003). Par ailleurs, maintenir la motivation des apprenants constitue un défi constant. Des stratégies pédagogiques efficaces incluent la gamification, où les concepts de jeu sont appliqués à l'apprentissage tels que (Anon 2013; Fournier et Wirz 2009;

Kazandzhy 2017), et la mise en place de défis de programmation progressifs (Abdessemed et al. 2018). D'autres recherches indiquent que la reconnaissance des réalisations et les retours d'informations positifs jouent également un rôle crucial dans le maintien de la motivation (Kourgiantakis, Sewell, et Bogo 2019; Lee et Ko 2011; Ryan et Deci 2000; Tsai, Tsai, et Lin 2015). Par ailleurs, il est également important d'appliquer les connaissances acquises dans des projets concrets afin de permettre aux apprenants non seulement de renforcer leurs compétences techniques, mais aussi de développer des compétences non techniques telles que le travail en équipe et la gestion de projet (Denny, Hanks, et Simon 2010).

2.2 Les Difficultés Rencontrées par les Novices en Programmation

Des chercheurs de divers horizons académiques, incluant l'éducation, la psychologie de l'éducation et l'informatique, ont conçu différents outils pour approfondir la compréhension de la programmation chez les débutants. Ces initiatives visent à maximiser les chances de réussite des étudiants (Gross et Powers 2005; Guzdial 2004; Linn 2000). D'ailleurs, ces outils pédagogiques ne sont pas de simples aides: ils sont des éléments indispensables pour surmonter les défis colossaux que représente l'apprentissage de la programmation (Soloway, Guzdial, et Hay 1994). Cependant, deux défis majeurs subsistent. Le premier concerne la nature de la relation entre les apprenants et les environnements de programmation, qui privilégient souvent une approche de commande et de contrôle plutôt qu'une dynamique de collaboration ou de cocréation des connaissances. Cette dynamique peut constituer un obstacle significatif dans le processus d'apprentissage des novices (Kelleher et Pausch 2005; Lewis 2010). Le second défi réside dans le manque d'analyses rétrospectives permettant d'évaluer l'efficacité de ces environnements (Cooper, Dann, et Pausch 2003; Proulx 2000). La documentation sur l'impact réel de ces outils reste assez limitée

Les études convergent généralement vers la difficulté rencontrée par les apprenants dans l'élaboration de stratégies pour résoudre des problèmes complexes (Winslow 1996). Pour commencer, les étudiants novices en programmation éprouvent des difficultés à construire un modèle mental solide de l'ordinateur, élément clé pour construire des modèles plus sophistiqués (Lister 2011). Il est communément admis que les programmeurs débutants ont souvent du mal à appréhender les concepts fondamentaux de la programmation dès les premières étapes, soulignant ainsi l'importance d'une bonne compréhension des concepts abstraits, de la pensée logique et des compétences en résolution de problèmes. D'autre part, ces obstacles, qu'ils soient d'ordre sémantique, conceptuel ou pragmatique, peuvent engendrer frustration et abandon. En effet, les apprenants montrent souvent une connaissance superficielle des programmes, se concentrant sur une approche séquentielle (ligne par ligne) plutôt que d'utiliser des structures de programme plus significatives. Par conséquent, leurs connaissances tendent à être spécifiques au contexte, ce qui les entrave souvent dans l'application adéquate de leurs connaissances acquises. Ainsi, bien qu'ils puissent maîtriser la syntaxe et la sémantique des énoncés individuels, ils éprouvent des difficultés à les assembler pour créer des programmes cohérents.

Une première piste à explorer est de comprendre comment aider les débutants, souvent dépourvus des compétences et des connaissances nécessaires, à réussir dans les cours d'algorithmique. En effet, plusieurs études ont montré que les premiers cours de programmation présentent un taux d'abandon alarmant, variant entre 25 % et 80 % à travers le monde (Bennedsen et Caspersen 2007; Watson et Li 2014). Cette situation interpelle et incite à s'interroger sur les facteurs déterminants qui sous-tendent la réussite ou l'échec dans l'apprentissage algorithmique. Des recherches antérieures ont identifié deux types de problèmes majeurs. Premièrement, les problèmes liés à l'utilisabilité des outils de programmation, qui incluent la complexité syntaxique des langages eux-mêmes (Robins et al. 2003).

Deuxièmement, les défis liés à l'abstraction des comportements dynamiques dans des méthodes statiques, une compétence essentielle mais difficile à maîtriser pour les novices (Soloway et Ehrlich 1984).

Par ailleurs, il est important de prendre en compte plusieurs facteurs individuels qui peuvent influencer la réussite des apprenants. Parmi ceux-ci figurent les capacités intellectuelles, les antécédents académiques, l'expérience en mathématiques et en informatique (Jacobs 2005), l'auto-efficacité, l'orientation vers un but, la motivation (Litalien et al. 2019; Yilmaz et Karaoglan Yilmaz 2023), l'utilisation de stratégies d'apprentissage, et la nature du modèle conceptuel du domaine (Bergin et Reilly 2005; Kinnunen et Malmi 2006). Les étudiants sont souvent exacerbés par des explications compliquées et des méthodes d'enseignement peu engageantes (Jenkins 2001). Ils voient fréquemment les outils de programmation comme des juges sévères de la qualité de leur code, ce qui peut transformer les erreurs de compilation ou d'exécution en sources de stress et d'anxiété, parfois conduisant à l'abandon des cours (Lahtinen, Ala-Mutka, et Järvinen 2005). Toutefois, il est essentiel de reconnaître qu'une rétroaction constructive et bien conçue peut transformer les erreurs en opportunités d'apprentissage, réduisant ainsi le sentiment d'échec et d'anxiété (Hattie et Timperley 2007).Ces éléments soulignent l'importance d'une approche holistique et personnalisée dans l'enseignement de la programmation, visant à surmonter les obstacles techniques et à renforcer les compétences cognitives des apprenants (Beaubouef et Mason 2005).

Ainsi, pour résumer, les recherches sur l'algorithmique s'accordent sur plusieurs domaines problématiques pour les débutants, tels que les tableaux, les types de données structurées, la récursion, les pointeurs, les références et la manipulation de la mémoire (Lahtinen, Kirsti, et al. 2005; Piteira et Costa 2012). Ces difficultés varient en nature : erreurs de syntaxe, difficultés sémantiques ou pragmatiques, c'est-à-dire la difficulté à appliquer les connaissances en programmation à des problèmes concrets (Guibert et Girard 2003; Lahtinen, Kirsti, et al. 2005; McCall et Kölling 2019). Nous synthétisons ci-après les difficultés les plus fréquemment rencontrés par les novices en matière de programmation :

- Les novices ont des difficultés à développer des stratégies cognitives pour résoudre de gros problèmes (Bonar et Soloway 1985; Guibert et Girard 2003),
- Les novices se concentrent beaucoup sur les détails du texte du programme et ne comprennent pas les informations implicites du code (Mayer 1981),
- Les novices sont souvent limités à une connaissance superficielle de la programmation (Lahtinen, Kirsti, et al. 2005; Piteira et Costa 2012),
- Les novices ne disposent pas d'un modèle simple et viable de l'ordinateur qui pourrait servir de base pour construire des modèles plus sophistiqués (Guibert et Girard 2003),
- Les novices n'ont pas de connaissances suffisantes en mathématiques (Futschek et Moschitz 2010),
- Les novices considèrent les outils de programmation comme des autorités qui jugent de ce qui est bon ou mauvais dans leur code (Lee et Ko 2011),
- Les novices ont du mal à comprendre que chaque état d'instruction exécuté est créé par l'état des instructions précédentes (Lahtinen, Kirsti, et al. 2005),
- Les novices ne perçoivent pas d'où viennent les données "lisibles" ou "futures" et comment elles sont stockées en mémoire (Bayman et Mayer 1983),
- Pour la plupart des débutants, l'écriture d'un programme et les erreurs de syntaxe, d'exécution ou de jugement du compilateur peuvent être considérées comme un signe ou une preuve d'échec personnel (Hattie et Timperley 2007; Jenkins 2001). Si toutes ces formes de retour d'information sont importantes pour aider un débutant à comprendre ce que sont les programmes et comment les ordinateurs les interprètent, l'expérience peut néanmoins être intimidante,

- ❖ Les langages de programmation actuels ne reflètent pas exactement les stratégies cognitives utilisées par les programmeurs débutants. De nombreux bugs de programmation pour débutants peuvent être directement attribués à une utilisation inappropriée du style ou de la stratégie de spécification du langage naturel (Eckerdal et Thuné 2005),
- ❖ Les déclarations conditionnelles et l'utilisation du « goto » simple posent une difficulté majeure, les apprenants ne comprennent pas ce qui se produira une fois que l'exécution du programme sera déplacée vers la ligne souhaitée. De plus, avec le « goto » conditionnel, les apprenants ont du mal à comprendre quelle action entreprendre si la condition n'est pas remplie (Soloway, Bonar, et Ehrlich 1983),
- ❖ Les instructions de l'affectation posent problème. Les apprenants semblent confondre la résolution ou le stockage d'une équation avec le symbole égal, avec le concept d'égalité (Du Boulay 1986).

De manière un peu plus générale, les novices en programmation font face à diverses lacunes:

- ❖ Un manque de connaissances préalables sur la nature même des ordinateurs.
- ❖ Un manque de connaissances préalables sur le fonctionnement des ordinateurs.
- ❖ Un manque de connaissances spécifiques au domaine.
- ❖ Un manque de connaissances préalables sur les algorithmes.
- ❖ Un manque de connaissances préalables en matière de programmation.
- ❖ Un manque de compétences mathématiques suffisantes.

2.3 Surmonter les Défis de l'Apprentissage de la Programmation

Plusieurs études ont cherché à identifier et à reconnaître les difficultés susmentionnées afin de soutenir de manière efficace le processus d'apprentissage (Bennedsen et Caspersen 2007; Jenkins et Davy 2002; Watson et Li 2014). Les chercheurs ont identifié deux types de problèmes majeurs: les difficultés liées à l'accessibilité des outils utilisés pour maîtriser la syntaxe des langages de programmation, et la nécessité d'abstraire des comportements dynamiques pour les convertir en une méthode statique unique.

- Accessibilité des outils de programmation: le premier problème, celui de l'utilisabilité des outils, réside dans la complexité des logiciels et des environnements de programmation utilisés pour enseigner l'algorithmique. Les apprenants peuvent être découragés par des interfaces complexes, des commandes difficiles à comprendre et des erreurs fréquentes, ce qui peut rapidement démotiver les étudiants (Guzdial 2015). Des recherches ont montré que simplifier les interfaces et fournir des tutoriels clairs peut considérablement améliorer l'engagement des étudiants (Farhan et al. 2018; Halverson et Graham 2019). Par exemple, des environnements de développement intégrés (IDE) simplifiés comme Scratch ont montré des résultats positifs en rendant la programmation plus accessible et ludique (Resnick et al. 2009).
- Abstraction et compréhension des concepts: le second problème est lié à la nécessité d'abstraction dans la programmation. Comprendre des concepts abstraits et les appliquer de manière efficace représente un défi majeur pour de nombreux apprenants. La programmation nécessite souvent de penser de manière logique et de traduire des comportements dynamiques en un code statique. Cette transition peut être difficile à appréhender, entraînant confusion et frustration (Ben-Ari 2001). Des approches pédagogiques comme l'utilisation de métaphores peuvent aider à rendre ces concepts plus accessibles (Sorva 2013). De plus, l'utilisation de visualisations interactives pour illustrer le fonctionnement interne des programmes peut faciliter la compréhension (Shaffer et al. 2010).

La transition des novices vers des projets complexes pose divers défis pédagogiques qui nécessitent des approches innovantes pour être surmontés. Parmi celles-ci, l'intégration de méthodes de tutorat et l'utilisation de plateformes éducatives interactives se révèlent particulièrement prometteuses (Drini 2018). L'une des approches suggère de présenter un modèle concret et familier d'ordinateur, tandis qu'une autre recommande d'encourager les apprenants à reformuler les données techniques dans leurs propres termes Ceci favorise, d'une part, l'activation des connaissances existantes qui sont pertinentes pour la compréhension du matériel nouvellement présenté (Falkner et Vivian 2015; Lister 2011). Son efficacité a été démontrée par les travaux ultérieurs d'Allogène (Fournier et Wirz 2009), un environnement de programmation informatisé d'assistance destiné aux débutants. À travers des étapes successives, l'apprenant sera amené à passer de l'énoncé du problème, formulé en langage naturel, à la modélisation de sa solution exprimée d'abord dans un langage de description, puis dans un langage de modélisation directement assimilé par la machine. D'autre part, l'activité d'élaboration peut influencer la performance du transfert puisque le but de l'élaboration est d'aider l'apprenant à décrire les concepts clés dans ses propres mots, en utilisant ses connaissances existantes (Chi et Wylie 2014). Cette démarche méthodologique, fondée sur l'activation et l'intégration des connaissances préexistantes, crée une transition naturelle vers l'exploration approfondie de la pensée algorithmique, que nous aborderons dans la section suivante.

3 La Pensée Algorithmique

La pensée algorithmique est un processus cognitif complexe qui implique la capacité à formuler clairement un problème, à le décomposer en plusieurs sous-problèmes bien définis, et à concevoir une solution étape par étape pour résoudre chaque sous-tâche de manière systématique (Futschek 2006; Sarı et al. 2022). Cette pensée englobe un ensemble de compétences et de concepts fondamentaux, tels que la décomposition fonctionnelle, la répétition (par itération et/ou récursion), l'organisation des données de base (comme les enregistrements, les tableaux et les listes), la généralisation et le paramétrage, la distinction entre l'algorithme et le programme, ainsi que les concepts de conception descendante et de raffinement (Ostrowska-Wawryniuk, Strzała, et Słyk 2022; Sungkaew, Lungban, et Lamhya 2022). Selon (Futschek 2006; Futschek et Moschitz 2010; Parmentier 2018) , la pensée algorithmique peut être considérée comme une compétence spécifique de résolution de problèmes, qui se compose de plusieurs aptitudes interdépendantes telles que:

- Analyser un problème donné: cela implique d'examiner le problème, d'en comprendre les aspects clés, et de formuler une compréhension claire de ce qui doit être résolu.
- Préciser les problèmes: il s'agit de définir précisément les problèmes identifiés dans l'analyse, en les décomposant en sous-problèmes plus petits et mieux définis.
- Trouver les actions de base appropriées pour des problèmes donnés: cette aptitude consiste à identifier les actions de base nécessaires pour résoudre les problèmes spécifiques identifiés précédemment, en utilisant des instructions ou des opérations simples et compréhensibles.
- Construire des algorithmes corrects pour des problèmes donnés en utilisant les actions de base, ce qui implique la conception de séquences logiques d'instructions en utilisant les actions de base identifiées, de manière à résoudre efficacement les sous-problèmes et à atteindre l'objectif global de résolution du problème initial.
- Prendre en compte tous les cas spéciaux et normaux possibles d'un problème: cette aptitude nécessite d'envisager toutes les situations et les conditions possibles dans lesquelles le problème peut se présenter, afin de garantir que l'algorithme fonctionne correctement et de manière fiable dans tous les scénarios envisageables.

- Évaluer l'exactitude, l'efficacité et la fin des algorithmes: il s'agit d'évaluer et de vérifier si les algorithmes conçus répondent correctement aux exigences du problème, s'ils sont efficaces en termes de temps d'exécution et de ressources requises, et s'ils aboutissent à une solution satisfaisante.
- Améliorer l'efficacité des algorithmes: cette aptitude consiste à optimiser les algorithmes existants en les rendant plus rapides, plus efficaces ou en améliorant leur qualité globale, tout en maintenant leur précision et leur fiabilité.

Par ailleurs, la pensée algorithmique est influencée par divers facteurs cognitifs humains, notamment la pensée abstraite et logique, la pensée structurelle, la créativité et la capacité à résoudre des problèmes (Parmentier 2018). Ces facteurs jouent un rôle déterminant dans la capacité d'un apprenant à adopter une approche algorithmique efficace lors de la résolution de problèmes informatiques. Par conséquent, il est impératif d'adopter une approche didactique appropriée pour guider les novices dans le développement de leurs compétences en pensée algorithmique (Dadhich et Bhaumik 2023; Vuji\vcić, Jan\vcec, et Mezak 2021). Dans la littérature, les approches suivantes peuvent être préconisées :

- Approche par pseudocode: introduire les étudiants à l'utilisation de pseudocode avant de les exposer à des langages de programmation spécifiques. Le pseudocode permet aux apprenants de se concentrer sur la logique et la structure de l'algorithme sans se préoccuper de la syntaxe du langage de programmation. Par exemple, donner un problème simple comme trier une liste de nombres et demander aux étudiants d'écrire un pseudocode pour le tri par insertion.
- Apprentissage par projets: impliquer les étudiants dans des projets pratiques qui nécessitent la conception et l'implémentation d'algorithmes.
- Utilisation de jeux et de simulations: utiliser des jeux et des simulations interactives pour enseigner les concepts algorithmiques de manière ludique et engageante. Par exemple, utiliser un jeu comme "Lightbot" où les étudiants programment un robot pour qu'il allume des cases sur un plateau de jeu, apprenant ainsi les concepts de séquences, boucles et conditions (Kazandzhy 2017).
- Pair programming (programmation en binôme): encourager les étudiants à travailler par paires pour résoudre des problèmes de programmation. Cette méthode permet aux apprenants de discuter de leurs idées et de collaborer pour trouver des solutions (Denny et al. 2010).
- Analyse de cas concrets: présenter aux étudiants des études de cas réels où des algorithmes ont été utilisés pour résoudre des problèmes complexes. Analyser ces cas en détail pour comprendre les choix algorithmiques effectués.
- Problèmes décomposés en sous-problèmes: enseigner aux étudiants à décomposer des problèmes complexes en sous-problèmes plus simples et gérables, puis à résoudre chaque sous-problème individuellement (Bey et Bensebaa 2011).
- Exercices progressifs: proposer une série d'exercices qui augmentent progressivement en complexité, permettant aux étudiants de renforcer leurs compétences et leur compréhension à chaque étape (Abdessemed et al. 2018). Par exemple, commencer par des exercices simples de manipulation de tableaux, puis progresser vers des algorithmes de tri et enfin vers des algorithmes de recherche et d'optimisation.

3.1 Taxonomie de la Pensée Algorithmique: Compétences Fondamentales pour les Etudiants en Programmation

En référence aux recherches menées par (Futschek 2006; Futschek et Moschitz 2010; Hu 2011; Kanaki et al. 2022; Parmentier 2018; Wing 2008), nous avons identifié les compétences fondamentales nécessaires à la réussite des étudiants dans un cours initial de

programmation. Ces compétences ont été catégorisées en utilisant une taxonomie inspirée des travaux de (Paquette 2002) et spécifiquement adaptée au domaine de l'algorithmique. Il convient de noter que cette taxonomie peut être élargie à d'autres domaines impliquant la résolution de problèmes. Structurée en trois niveaux hiérarchiques, allant du plus général au plus spécifique, cette classification caractérise les actions humaines et l'apprentissage qui en découle. Les objectifs pédagogiques de chaque niveau sont synthétisés dans le tableau ci-dessous:

Tableau 2.1 Hiérarchie des Compétences en Résolution de Problèmes Algorithmiques

Niveau	Objectif	Aptitudes vérifiées
Création	Guider le novice pour transformer un problème en un autre pouvant être résolu plus facilement (la capacité à résoudre des problèmes)	- Capacité à décomposer les problèmes par fonctionnalité en plusieurs sous-problèmes, o Capacité à trouver les actions primaires adaptées à un problème donné, - Capacité à concevoir des solutions, o Capacité à élaborer un algorithme correct pour un problème donné en utilisant les actions nécessaires, o Capacité à produire une solution étape par étape pour résoudre chaque sous-tâche.
Reproduction	Guider le novice pour utiliser les processus intellectuels supérieurs de synthèse, d'instanciation ou de transposition afin de développer une solution efficace	- Capacité à généraliser, o De l'applicabilité spécifique à l'applicabilité plus large, o Reconnaitre les parties des solutions réutilisables, - Capacité à affiner, - Capacité à envisager tous les cas possibles d'un problème : les cas spéciaux et normaux.
Autogestion	Guider le novice pour penser en termes d'évaluation et d'autocritique	- Capacité à évaluer l'exactitude des algorithmes, o Capacité à évaluer l'efficacité des algorithmes, o Capacité à évaluer la terminaison des algorithmes, - Capacité à améliorer l'efficacité des algorithmes.

En synthèse, cette taxonomie de la pensée algorithmique comprend trois niveaux d'objectifs pédagogiques. Ces paliers sont conçus pour guider les étudiants dans l'acquisition progressive des compétences nécessaires à la résolution efficace de problèmes dans le domaine de la programmation. L'application de cette taxonomie dans un cadre pédagogique peut fournir une structure claire pour l'enseignement et l'évaluation de la pensée algorithmique, favorisant ainsi la réussite des étudiants.

Niveau 1 : Pensée Algorithmique Générale

Au premier niveau, la pensée algorithmique générale implique :

- La compréhension des concepts fondamentaux de l'algorithmique et leurs applications dans la résolution de problèmes.
- La reconnaissance de l'importance de la pensée algorithmique dans le domaine de la programmation et des sciences informatiques.
- Le développement d'une compréhension approfondie des principes de base de la décomposition fonctionnelle, de la répétition, de l'organisation des données et de la conception descendante.

Exemples concrets:

- **Décomposition fonctionnelle**: enseigner aux étudiants à diviser un problème complexe en parties plus simples, par exemple, en créant des fonctions séparées pour chaque opération dans un programme de calculatrice.
- **Répétition**: introduire les boucles for et while pour résoudre des problèmes de répétition, comme l'affichage des nombres de 1 à 1000.

Niveau 2 : Compétences en Analyse Algorithmique

Au deuxième niveau, les compétences en analyse algorithmique exigent :

- Une analyse critique du problème donné afin d'identifier ses caractéristiques clés.
- Une décomposition du problème en sous-problèmes plus petits et mieux définis.
- Une identification des actions de base appropriées pour résoudre les sous-problèmes identifiés.
- Une construction des algorithmes corrects en utilisant les actions de base sélectionnées pour résoudre les sous-problèmes.
- Une évaluation de la justesse, l'efficacité et les limites des algorithmes développés.

Exemples concrets :

- **Analyse critique**: analyser un problème de tri pour identifier le meilleur algorithme à utiliser (tri à bulles, tri par insertion, etc.).
- **Décomposition**: décomposer un programme de gestion de stocks en modules pour l'ajout, la suppression et la recherche d'articles.

Niveau 3 : Compétences en Optimisation Algorithmique

Finalement, au troisième niveau, les compétences en optimisation algorithmique impliquent :

- Une exploration approfondie des algorithmes développés pour identifier les possibilités d'amélioration.
- Une proposition d'amélioration visant à optimiser les performances des algorithmes en termes d'efficacité, de temps d'exécution et de consommation de ressources.
- Une évaluation de l'impact des améliorations proposées sur les performances globales des algorithmes.

Exemples concrets:

- **Optimisation**: réduire la complexité temporelle d'un algorithme de tri en passant du tri à bulles ($O(n^2)$) au tri rapide ($O(n \log n)$).
- **Évaluation des améliorations**: comparer les performances d'un algorithme avant et après optimisation en termes de temps d'exécution et d'utilisation de la mémoire.

Cette taxonomie, structurée en trois niveaux, propose un cadre exhaustif pour l'enseignement de la pensée algorithmique, accompagnant les étudiants des concepts de base à l'optimisation avancée des algorithmes. Cette hiérarchisation permet aux enseignants d'équiper les étudiants de compétences solides en résolution de problèmes, indispensables à leur réussite dans le domaine de la programmation.

4 Différences de Compréhension entre Programmeurs Novices et Experts

Diverses études telles que (Robins et al. 2003; Soloway et Ehrlich 1984) se sont penchées sur l'évaluation de la compréhension des programmes chez les apprenants novices. Ces recherches visent à identifier les types d'informations que ces apprenants recueillent et les liens qu'ils établissent entre ces celles-ci. L'objectif est de déterminer si les novices les plus performants dans leur compréhension diffèrent de ceux considérés comme moins compétents, tant en termes de connaissances acquises qu'en termes de connexions mentales établies. Les chercheurs ont constaté que les programmeurs experts possèdent la capacité de construire plusieurs représentations d'un programme, ce qui leur permet d'en avoir une meilleure compréhension globale. Ils développent également une représentation mentale abstraite qui met l'accent sur la fonction générale du programme. D'ailleurs, pour comprendre un programme, les étudiants « experts » identifient des caractéristiques qui correspondent à des modèles familiers et, une fois le modèle reconnu, ils peuvent souvent omettre de vérifier les détails. En revanche, les apprenants novices ont tendance à se concentrer davantage sur une représentation concrète du fonctionnement du programme. Étant encore en phase d'apprentissage, ils se focalisent principalement sur les détails textuels du programme, souvent sans saisir les informations implicites qu'il contient. Leur attention est surtout portée vers la création d'une représentation mentale détaillée et procédurale des textes du programme afin de faciliter leur compréhension. De plus, la capacité des novices à comprendre les informations relatives aux fonctions n'est généralement observée que dans le cadre de programmes courts et est presque inexistante lorsqu'il s'agit de programmes plus longs et plus complexes.

Les implications de ces observations sont décisives pour le développement de stratégies pédagogiques en programmation. Des techniques telles que l'enseignement explicite des structures de code communes, la pratique de la décomposition de problèmes et l'exposition graduelle à des programmes de complexité croissante peuvent être bénéfiques (Abdessemed et al. 2018; Bey et Bensebaa 2011).

4.1 Variabilité des Compétences en Programmation

Une compréhension approfondie de la programmation nécessite de posséder à la fois des connaissances déclaratives (savoir quoi faire) et des connaissances procédurales (savoir comment faire). En outre, des compétences en mémoire, compréhension, résolution de problèmes, abstraction et pensée logique sont indispensables (Elghibari et Elouahbi 2015). Des chercheurs (Bonar et Soloway 1985; Estupiñán et Jesús & Martín Estévez 2019) ont entrepris d'identifier les domaines spécifiques de compréhension de la programmation dans lesquels les apprenants peuvent présenter des niveaux de compétence variables. Les résultats de ces études ont révélé que certains sujets démontrent un niveau élevé de compréhension dans des domaines tels que la sélection de structures, les structures d'itération, les variables, les opérateurs et les priorités. En revanche, d'autres sujets manifestent un niveau de compréhension plus faible pour des concepts plus avancés tels que les tableaux, les types de données structurées, la récursion, les pointeurs et les références. Cette variabilité peut être attribuée à plusieurs facteurs, tels que

les différences individuelles en termes de capacités cognitives, d'expériences antérieures et de styles d'apprentissage (Keinänen, Ursin, et Nissinen 2018).

Cette variabilité des compétences en programmation parmi les apprenants souligne l'importance d'une approche pédagogique flexible et inclusive. En effet, adopter des méthodes d'enseignement adaptées aux besoins individuels des étudiants permet de favoriser une compréhension approfondie et une maîtrise plus uniforme des concepts avancés de la programmation (Biggs 2003). Notamment, l'utilisation de techniques pédagogiques variées, telles que l'apprentissage actif, les travaux pratiques et les projets collaboratifs, est essentielle pour renforcer la compréhension des concepts complexes. L'apprentissage actif, par exemple, encourage les étudiants à s'engager activement avec le matériel d'apprentissage, ce qui peut améliorer la rétention et la compréhension (Prince 2004). De même, les projets collaboratifs favorisent le développement des compétences en résolution de problèmes et la coopération entre pairs, deux compétences cruciales dans le domaine de la programmation. En outre, le recours à des outils de visualisation et à des environnements de programmation interactifs peut grandement faciliter l'apprentissage de concepts abstraits et dynamiques. Les visualisations permettent de représenter graphiquement des concepts complexes, rendant ainsi plus accessibles des idées qui peuvent être difficiles à appréhender par des moyens textuels seuls (Sorva 2013). Les environnements de programmation interactifs offrent un cadre dans lequel les étudiants peuvent expérimenter et apprendre de manière autonome et engageante (Boyce et Barnes 2010; Resnick et al. 2009). Ainsi, en intégrant ces diverses approches et outils, il devient possible de créer un environnement d'apprentissage plus inclusif et efficace, répondant aux besoins diversifiés des étudiants.

4.2 Compréhension Conceptuelle et Erreurs Fréquentes chez les Apprenants en Programmation

Dans le domaine de la psychologie de l'éducation, il est essentiel de distinguer entre la simple obtention de la bonne réponse et la véritable compréhension des concepts enseignés. Cette nuance prend une importance particulière lorsqu'il s'agit de la compréhension des programmes informatiques. En effet, des études montrent qu'il existe des différences significatives entre l'apprentissage par cœur, où l'apprenant se contente de mémoriser une procédure, et l'apprentissage basé sur la compréhension structurelle, où l'apprenant explore les concepts sous-jacents (F. P. Deek, M. Turoff, et J. A. McHugh 1999; Winslow 1996). Par conséquent, il est important d'utiliser des méthodes pédagogiques qui favorisent une réelle compréhension (Ben-Ari 2001).

4.2.1 Identification et Classification des Erreurs

Les erreurs les plus courantes sont souvent de nature syntaxique, telles que des fautes d'orthographe dans les noms de variables ou des déclarations incorrectes de variables (Qian et Lehman 2017). Par ailleurs, l’étude de (Robins et al. 2003) a révélé que les concepts de pointeurs et de manipulation de la mémoire posent généralement des défis considérables aux apprenants. La maîtrise approfondie de ces concepts constitue un prérequis essentiel pour une compréhension globale et efficace de la programmation (Lister et al. 2004).

Pour relever ces défis, une analyse comparative des études réalisées en didactique informatique révèle l'existence de différentes classes d'erreurs, telles que les erreurs temporelles et anthropomorphiques. Les erreurs temporelles se réfèrent aux erreurs qui surviennent en raison d'une mauvaise gestion ou compréhension de la séquence des opérations dans un programme. Ces erreurs sont courantes chez les débutants qui n'ont pas encore acquis une intuition solide

du flux de contrôle dans un programme (Ahmadzadeh, Elliman, et Higgins 2005). Par exemple, un apprenant peut essayer d'utiliser une variable avant de l'initialiser, ou peut ne pas comprendre l'ordre d'exécution des instructions dans une boucle ou une condition. Une mauvaise gestion du timing dans les interactions entre différentes parties d'un programme peut également conduire à des erreurs, comme les erreurs de concurrence où plusieurs processus tentent de modifier une ressource partagée simultanément sans synchronisation adéquate.

Les erreurs anthropomorphiques sont celles où les apprenants attribuent des caractéristiques humaines aux composants du programme. Ces erreurs se produisent lorsque les novices conçoivent ou expliquent le comportement d'un programme en termes de motivations humaines ou d'actions intentionnelles des éléments du code (PANE, RATANAMAHATANA, et MYERS 2001). Par exemple, un apprenant pourrait penser qu'un objet "sait" ce qu'il doit faire sans comprendre que c'est le code écrit par le programmeur qui détermine son comportement. Ce type de pensée peut mener à des malentendus fondamentaux sur la manière dont les programmes fonctionnent, en particulier concernant la logique des algorithmes et les interactions entre les différentes parties d'un programme (Guzdial 2004). C'est pourquoi, l'apprentissage de la programmation nécessite la mise en place d'activités pédagogiques adaptées à chaque étape du processus, ainsi qu'un environnement de soutien adéquat.

4.2.2 Stratégies Pédagogiques pour Aborder les Erreurs

Pour aborder ces erreurs, il faut mettre en œuvre des stratégies pédagogiques spécifiques:

- Visualisation et animation du code: utiliser des outils qui montrent visuellement l'exécution du code, étape par étape, peut aider les apprenants à mieux comprendre la séquence des opérations et à corriger les erreurs temporelles. Des environnements de développement interactifs comme ceux qui permettent de déboguer visuellement peuvent être très utiles.
- Décomposition et modélisation: enseigner aux étudiants à décomposer les problèmes en étapes logiques et à modéliser les comportements du programme sans recourir à des analogies anthropomorphiques. Encourager une approche méthodique où chaque partie du programme est comprise en termes de règles strictes et de flux de contrôle.
- Exercices de révision et de correction: proposer des exercices où les étudiants doivent identifier et corriger des erreurs dans des exemples de code. Ces exercices peuvent être conçus pour cibler spécifiquement les erreurs temporelles et anthropomorphiques, aidant les apprenants à reconnaître et éviter ces pièges.
- Retour d'information immédiat et explicatif: fournir une rétroaction détaillée et immédiate lors des exercices de programmation. Par exemple, lorsqu'une erreur temporelle est commise, expliquer pourquoi l'ordre des opérations est incorrect et comment cela affecte le résultat final.

4.3 Évaluation des Compétences en Programmation

La conception des évaluations repose sur trois piliers fondamentaux, à savoir, un cadre conceptuel pour appréhender la cognition et l'acquisition de compétences par l'apprenant dans le domaine, un ensemble de croyances relatives aux modalités d'observations destinées à démontrer les compétences de l'apprenant, et un processus d'interprétation visant à attribuer un sens à ces démonstrations (Durfee, Schneberger, et Amoroso 2007; Rahaman et Hoque 2022; Yilmaz et Karaoglan Yilmaz 2023). L'absence d'un modèle cognitif spécifique en informatique risque de restreindre la production d'évaluations particulièrement efficaces et persuasives (Chaouachi et al. 2019; Elghibari et Elouahbi 2015).

Nous possédons désormais une connaissance approfondie des défis auxquels sont confrontés les apprenants lorsqu'ils s'initient à la programmation, mais les causes de ces difficultés restent encore à élucider. L'absence d'un modèle de développement des connaissances pour les débutants en programmation constitue un obstacle majeur à l'évaluation efficace de ces apprenants (Bouhineau, Luengo, et Mandran 2018). Il devient donc essentiel de concevoir un modèle cognitif robuste permettant de saisir les dynamiques de l'apprentissage en programmation. Ce que nous apprenons lors d'une évaluation ne se limite pas à un instantané des compétences actuelles des étudiants; cela façonne également la structure et l'orientation des évaluations futures. En perfectionnant notre approche d'évaluation, nous augmentons notre capacité à observer et à interpréter les acquis des apprenants de manière plus précise et nuancée. Ce processus d'amélioration continue garantit que les outils pédagogiques et les méthodes d'enseignement restent pertinents et efficaces, favorisant ainsi un apprentissage plus profond et durable chez les étudiants.

4.3.1 Méthodes Empiriques d'Evaluation des Performances

Une démarche empirique peut mobiliser diverses mesures pour évaluer les performances des apprenants, telles que :

- Notes: résultats obtenus dans les différents travaux de cours et/ou notes finales.
- Enquêtes: attitudes, préférences et évaluations des cours par les étudiants.
- Statistiques sur l'utilisation de l'outil: fréquence et modalités d'usage des ressources pédagogiques mises à disposition.
- Taux de déclaration de spécialisation: proportion d'étudiants choisissant de se spécialiser dans le domaine étudié.
- Taux de rétention des enseignements: pourcentage d'étudiants qui poursuivent leur parcours académique après l'enseignement reçu.
- Groupes de discussion et entretiens: recueil des perceptions et des expériences des étudiants à travers des échanges approfondis.
- Observation directe des interactions: analyse des comportements et des interactions des étudiants au sein de l'environnement d'apprentissage.
- Artefacts produits par les apprenants: analyse des travaux réalisés par les étudiants, comme les examens et les devoirs, indépendamment des notes attribuées.

4.3.2 Méthodes d'Evaluation des Compétences en Programmation

Pour une évaluation plus fine des compétences en programmation, il est pertinent d'incorporer diverses méthodes techniques telles que :

- Tests unitaires: les tests unitaires vérifient individuellement les composants du code, tels que les fonctions et les méthodes. Ils présentent plusieurs avantages, notamment la détection rapide des erreurs locales et la facilité d'automatisation. (Daka et Fraser 2014) soulignent que les tests unitaires sont cruciaux pour identifier les erreurs de manière isolée, ce qui facilite la correction rapide des bogues. Cependant, ces tests peuvent manquer les erreurs d'interaction entre composants et nécessitent une bonne couverture de code pour être efficaces. Une faible couverture de code peut limiter leur efficacité, comme l'a noté (Madeyski 2010).
- Revue de code: la revue de code est une évaluation par les pairs ou par un expert du code écrit par les développeurs. Elle offre un retour détaillé sur la qualité du code et les meilleures pratiques, tout en encourageant la collaboration et l'apprentissage mutuel. (Mcintosh et al. 2016) affirment que les revues de code améliorent la qualité du logiciel et renforcent les compétences des développeurs. Cependant, cette pratique peut être subjective et nécessite du temps et de l'expérience pour être efficace. La subjectivité et

le temps requis pour une revue approfondie sont des défis notables, comme le soulignent (Bacchelli et Bird 2013).

- Analyse statique : l'analyse statique consiste en un examen du code sans exécution pour détecter les erreurs potentielles et les failles de sécurité. Elle identifie les problèmes avant l'exécution et couvre un large éventail de problèmes potentiels. L'analyse statique est efficace pour détecter les erreurs et les failles de sécurité. Cependant, elle peut générer des faux positifs et ne détecte pas les erreurs d'exécution, comme l'ont noté (Johnson et al. 2013).
- Simulation et modélisation: la simulation et la modélisation utilisent des modèles pour simuler le comportement du code dans divers scénarios. Elles permettent de tester des situations variées et complexes, et peuvent être utilisées pour l'enseignement des concepts avancés. Selon (Sargent 2011), la simulation et la modélisation sont essentielles pour comprendre les dynamiques complexes des systèmes logiciels. Cependant, leur mise en place peut être complexe et nécessite une bonne compréhension des modèles utilisés. La mise en œuvre et la compréhension des simulations requièrent des compétences avancées et une préparation approfondie.

En intégrant ces méthodes techniques à l'évaluation empirique des compétences en programmation, nous obtenons une vue plus complète et précise des capacités des apprenants. Cette approche combinée permet de surmonter les limites inhérentes à chaque méthode prise isolément.

Travaux Connexes

5 Travaux Connexes

Notre démarche consiste à segmenter les environnements de programmation conçus pour les débutants en trois catégories principales. La première catégorie englobe les environnements de programmation visuels, basés sur des blocs, tels que Alice (Cooper & Dann, 2000), Scratch (Maloney et al., 2008), Tiled Grace (Homer & Noble, 2013), la seconde catégorie concerne les jeux sérieux, tandis que la troisième traite des systèmes de tuteurs intelligents.

5.1 Environnements Visuels

Les environnements de programmation basés sur des blocs visuels tels qu'Alice (Cooper et Dann, 2000) ou Scratch (Maloney et al., 2008) ont pour objectif commun de limiter les facteurs qui peuvent décourager les apprenants en supprimant la nécessité de mémoriser les noms des procédures ou en simplifiant la syntaxe et d'éviter ainsi les erreurs qui peuvent être causées (Price & Barnes, 2015). La principale caractéristique de ces outils est l'utilisation du glisser-déposer de blocs de code qui, une fois assemblés, formeront un programme. Ces blocs peuvent représenter soit des structures de haut niveau comme les traitements ou des boucles, soit des opérateurs de bas niveau comme la comparaison de division ou d'égalité.

Ces outils pédagogiques ont indubitablement démontré leur efficacité tant en milieu scolaire qu'en contexte d'apprentissage informel (Price & Barnes, 2015). De même, l'étude de (Futschek & Moschitz, 2010) présente une approche où les étudiants se consacrent à l'élaboration d'algorithmes visant à résoudre des problèmes spécifiques. En implémentant ces algorithmes, les étudiants ressentent le besoin d'y jouer, découvrant ainsi les points forts et les lacunes de leurs codes. Dans les scénarios d'apprentissage proposés, les auteurs ont observé que les étudiants se muent en acteurs et scénaristes de leur processus d'apprentissage, favorisant une

efficacité accrue lorsqu'ils travaillent en groupe. Néanmoins, subsiste une contrainte majeure : la communication entre l'apprenant et l'ordinateur est à sens unique, où la machine exécute le code sans donner d'interprétation ou d'explication, constituant ainsi une limitation significative (Lee & Ko, 2011). Par ailleurs, cette situation peut engendrer frustration et isolement chez l'apprenant, le laissant dans l'incompréhension de ce qui se passe. Ci-dessous, quelques exemples d'environnements conçus pour la programmation :

- Alice (Cooper & Dann, 2000) est un environnement qui permet aux élèves de mettre en œuvre des algorithmes en manipulant des objets par glisser-déposer, et de visualiser leurs algorithmes sous forme d'animations. La particularité de cet environnement est que les programmes fonctionnent sans erreur de compilation (Gross & Powers, 2005).
- Karel (Becker, 2001) est un outil conçu pour préparer les élèves à résoudre des problèmes et assurer la réussite de leur apprentissage du langage Pascal.
- Raptor (Kuen, 2013) est un environnement de développement dans lequel les étudiants ont la possibilité de construire des programmes sous forme d'organigrammes.
- Allogène (Fournier & Wirz, 2009) est un environnement de programmation informatique qui aide les apprenants débutants. Par étapes successives, l'apprenant passe de l'énoncé du problème formulé en langage naturel, à la modélisation de sa solution exprimée d'abord dans un langage de description, puis dans un langage de modélisation directement assimilé par la machine.
- Codewitz (Matthíasdóttir, 2004) est un projet qui vise à développer des solutions pour l'apprentissage pratique de la programmation en introduisant des concepts de programmation et des structures de langage telles que les variables, les boucles et les déclarations conditionnelles.

5.2 Jeux Sérieux

L'emploi des jeux sérieux (Serious Games) dédiés à l'apprentissage de la programmation sur des dispositifs mobiles exerce une influence tangible sur les performances des apprenants, tel que démontré par l'étude de (Yallihep & Kutlu, 2019). Parallèlement, les travaux de recherches de (Cubero-Ibáñez et al., 2017) et une étude empirique approfondie des jeux informatiques à vocation pédagogique menée par (Boyle et al., 2016), ont mis en évidence l'impact positif de ces outils quant à l'acquisition de connaissances nouvelles, à l'encouragement de l'engagement des apprenants et au soutien apporté à leur processus d'apprentissage. Ci-dessous une liste non exhaustive des jeux sérieux spécifiquement développés pour l'enseignement de la programmation.

- **Gidget**, jeu sérieux élaboré par (Lee & Ko, 2011), présente un protagoniste robotique éponyme confronté au défi de composer le code adéquat pour mener à bien ses missions. La recherche menée par (Lee & Ko, 2011) s'articule autour du principe de la co-création, impliquant une collaboration entre l'apprenant et le personnage virtuel du jeu informatique dans l'élaboration du code visant à atteindre des objectifs ludiques. Les auteurs appuient leur choix méthodologique sur plusieurs diverses études dans les domaines de l'éducation et de la psychologie pédagogique, révélant que les individus attendent des ordinateurs des réactions sociales similaires à celles des êtres humains. Deux versions du jeu ont été créées pour explorer l'influence du retour d'information sur la motivation des apprenants. Les auteurs soulèvent un point important qui doit être pris en compte : le rôle essentiel du retour d'information dans le processus d'apprentissage. Par exemple, pour certains apprenants, un retour négatif est perçu comme une indication de la capacité générale à effectuer une tâche, plutôt que de la capacité à effectuer une tâche particulière. Cependant, une telle évaluation qualitative peut être un obstacle majeur et décourager les apprenants.

- Lightbot (Kazandzhy, 2017) est un jeu visant à initier les joueurs novices ou peu expérimentés en programmation aux concepts fondamentaux de ce domaine. Chacun des niveaux de ce jeu requiert l'utilisation de logique de programmation pour être résolu. Les apprenants utilisent des blocs de code déjà préexistants pour avancer à travers vingt (20) niveaux divisés en trois (3) étapes, sachant que seul le premier niveau de chaque étape est débloqué (accessible dès le départ).
- Le jeu Bead Loom (Boyce & Barnes, 2010) est un jeu qui incite les élèves à découvrir les fonctions de bouclage pour créer des motifs de perles. Plutôt que de placer chaque perle individuellement, les joueurs utilisent des fonctions de bouclage pour créer des motifs complexes et répétitifs. Ce jeu éducatif vise à renforcer les compétences en logique et en algorithmes des élèves, en leur permettant de visualiser et d'expérimenter avec des structures de contrôle de flux telles que les boucles donnant lieu ainsi à une compréhension plus profonde et plus intuitive des principes fondamentaux de la programmation.

Diverses initiatives ont été entreprises pour inciter les étudiants à concevoir et tester leurs propres jeux à travers des projets tels que Georgia Computes! (Guzdial, 2004), Game2Learn (Barnes et al., 2007), Lightbot: Code Hour, ScratchJr (Resnick et al., 2009), Games for Change, Robozzle, Cargo-Bot, et CodeSpark Academy. Ces programmes visent à rendre l'apprentissage de la programmation plus engageant et accessible en intégrant des éléments ludiques et interactifs (Sweller, 1988). D'autres recherches ont examiné l'impact des rétroactions interactives et des tutoriels adaptatifs pour soutenir l'apprentissage en temps réel, contribuant ainsi à un apprentissage plus personnalisé et efficace (Aleven et al., 2006).

5.3 Tuteurs Intelligents

Les tuteurs intelligents ont été principalement élaborés dans l'optique d'aider les apprenants à déboguer leurs programmes et à diagnostiquer leurs erreurs. Le rôle de ces tuteurs est de surveiller les interactions des apprenants et de les orienter vers la résolution adéquate en cas de difficultés (Pillay, 2003). Des études récentes, dont celle de (du Boulay, 2016) ont démontré la fiabilité et l'efficacité des tuteurs intelligents. Les nouveaux systèmes de tuteurs intelligents visent à renforcer la motivation et l'engagement des apprenants en ayant recours à la gammification comme levier de motivation (Dermeval & Bittencourt, 2020). Dans le domaine de l'algorithmique, ces tuteurs intelligents se manifestent à travers diverses initiatives, parmi lesquelles :

- Proust (Johnson & Soloway, 1985) est un système qui utilise les programmes de l'apprenant en entrée afin de générer une liste de bogues détaillant chaque erreur en sortie.
- Intellitutor (H. Ueno & A. Nakajima, 1993) est un tuteur intelligent qui débogue les programmes écrits en Pascal et C en présentant à l'apprenant une liste de bogues, accompagnée d'une explication pour chaque erreur, son emplacement, sa nature et des indications sur la manière de la corriger.
- SIPLesII (Pillay, 2003) diagnostique les erreurs sémantiques dans les programmes pour débutants. Ce système conserve la solution de chaque problème et demande au développeur de soumettre un nouvel algorithme pour chaque déviation dans la structure de l'algorithme de l'étudiant et des algorithmes de solution préalablement enregistrés.
- CodeTutor (Ahmadzadeh, Elliman, & Higgins, 2019) est un tuteur intelligent récent qui utilise des techniques de machine learning pour analyser les erreurs de code des étudiants et fournir des conseils personnalisés. Il identifie non seulement les erreurs, mais propose également des solutions alternatives et des explications détaillées pour aider les apprenants à comprendre les concepts sous-jacents et à améliorer leur code.

- Pythia (Biderman et al., 2023) développé par des chercheurs de l'Université de Stanford, utilise l'apprentissage profond pour fournir des explications détaillées et personnalisées aux erreurs de code en Python. Pythia est capable d'identifier non seulement les erreurs syntaxiques, mais aussi les erreurs logiques et les problèmes de conception.
- CodeWhisperer est un outil d'IA générative développé par Amazon, qui aide les développeurs à écrire du code plus rapidement et de manière plus efficace. Il peut suggérer des blocs de code entiers, voire des fonctions complètes, en fonction du contexte et des commentaires de l'utilisateur (Yetiştiren et al., 2023).

Les tuteurs intelligents de nouvelle génération se distinguent par plusieurs caractéristiques avancées et communes, illustrant des progrès significatifs dans le domaine de l'éducation informatique.

- ❖ Utilisation de l'apprentissage profond: l'intégration de modèles d'apprentissage profond permet à ces systèmes d'analyser de vastes quantités de code et de discerner des motifs complexes. Cette capacité leur confère une grande adaptabilité et une amélioration continue au fil du temps, optimisant ainsi l'efficacité de l'enseignement et du débogage des programmes (LeCun et al., 2015).
- ❖ Explications personnalisées: les tuteurs intelligents contemporains sont conçus pour fournir des explications détaillées et adaptées aux erreurs de code, prenant en compte le niveau de compétence de l'apprenant ainsi que ses connaissances antérieures. Cette personnalisation aide à clarifier les concepts et à renforcer la compréhension de l'étudiant, améliorant ainsi l'expérience d'apprentissage (Koedinger et al., 2012).
- ❖ Suggestions de corrections: au-delà de la simple détection des erreurs, ces systèmes avancés offrent souvent des suggestions de corrections concrètes. Cette fonctionnalité facilite grandement le processus de débogage en guidant l'apprenant vers la solution, tout en expliquant les raisons des erreurs et les moyens de les éviter à l'avenir (Vanlehn, 2006).

D'autres outils ont tenté de simplifier le processus de débogage des programmes en permettant aux apprenants de sélectionner des questions "pourquoi" sur la sortie du programme. Certaines approches fournissent aux tâches un scénario pour l'apprentissage des principes des algorithmes sans recourir à des ordinateurs. Dans ces approches, les apprenants mettent en œuvre des algorithmes, ce qui renforce leur compréhension de la conceptualisation algorithmique (Futschek, 2006). À titre d'exemple :

- Le tuteur de Lisp a été développé pour fournir un enseignement individualisé aux programmeurs de Lisp débutants. La tâche du tuteur est de surveiller les apports de l'élève et de guider l'apprenant sur la bonne voie s'il éprouve des difficultés (Farrell et al., 1984).
- Bridge est un système de tutorat conçu pour aider les élèves à assimiler les concepts de bouclage en Pascal, tout en fournissant une stratégie d'enseignement adaptée à l'historique de l'élève conservée par le système (Bonar & Cunningham, 1988).
- PyTutor est un tuteur intelligent qui aide les étudiants à apprendre la programmation en Python. Il fournit des visualisations étape par étape de l'exécution du code, ce qui permet aux étudiants de comprendre plus facilement les erreurs et les concepts complexes (Guo, 2013).

6 Conclusion

L'apprentissage de la programmation représente un défi complexe qui nécessite une compréhension approfondie des aspects conceptuels et pratiques. L'étude détaillée de la pensée

algorithmique et de la compréhension des programmes informatiques révèle un paysage d'une grande complexité, où se juxtaposent les défis inhérents à l'assimilation de concepts fondamentaux et les opportunités offertes par des approches pédagogiques variées. Cette exploration met en lumière la notion essentielle selon laquelle la maîtrise de la programmation dépasse largement la simple écriture de code. Elle requiert une appréhension précise des principes sous-jacents et une capacité à décomposer les problèmes complexes en entités plus gérables.

Les stratégies de compréhension en programmation, telles que décrites par (Estupiñán et Jesús & Martín Estévez 2019; F. P. Deek et al. 1999; Futschek 2006), mettent en évidence l'importance de développer une représentation mentale abstraite des programmes. De même, les erreurs fréquentes et les difficultés cognitives rencontrées par les apprenants nécessitent une attention particulière. Les erreurs temporelles et anthropomorphiques sont courantes chez les débutants, et des stratégies pédagogiques spécifiques, telles que la visualisation et l'animation du code, la décomposition et la modélisation, ainsi que les exercices de révision et de correction, peuvent aider à les surmonter. Par ailleurs, l'importance des modèles cognitifs pour l'évaluation en programmation ne peut être sous-estimée. Un cadre conceptuel solide est nécessaire pour évaluer efficacement les compétences des apprenants et pour guider l'amélioration continue des méthodes d'enseignement.

Une compréhension approfondie des défis et des stratégies pédagogiques en programmation est nécessaire pour concevoir des cursus de formation mieux adaptés. Les connaissances et les modèles développés à partir de recherches empiriques peuvent contribuer à améliorer l'apprentissage et la maîtrise de la programmation. Dans cette perspective, le prochain chapitre explorera comment l'intelligence artificielle, notamment les techniques de suréchantillonnage peuvent être utilisées pour une meilleure représentativité des profils apprenants. De plus, les modèles de langage pré-entraînés seront abordés dans l’optique d’améliorer la compréhension des solutions des apprenants, ouvrant ainsi de nouvelles voies pour une éducation toujours plus personnalisée.

CHAPITRE III : L'APPRENTISSAGE AUTOMATIQUE AU SERVICE DE LA PEDAGOGIE

1 Introduction à L'Intelligence Artificielle

Dans un monde de plus en plus façonné par la technologie, l'intelligence artificielle (IA) s'impose comme l'une des forces motrices de la transformation des secteurs clés de la société, notamment l'éducation. L'IA est aujourd'hui une réalité tangible qui redéfinit non seulement la manière dont les machines interagissent avec les êtres humains, mais aussi comment nous apprenons, enseignons et évaluons les processus éducatifs. Les origines de l'intelligence artificielle remontent aux années 1950, lorsque des pionniers tels qu'Alan Turing et John McCarthy ont commencé à explorer la possibilité de créer des machines capables de penser et de résoudre des problèmes complexes. Turing, avec son célèbre test de Turing (1950), a posé la question fondamentale de savoir si les machines pouvaient imiter l'intelligence humaine (Saygin et al., 2000). Au fil des décennies, l'intelligence artificielle a évolué, passant d'algorithmes simples à des systèmes complexes capables de traiter de vastes quantités de données, de reconnaître des motifs, et même de prendre des décisions autonomes (Hwang et al., 2020). Les avancées technologiques, telles que l'apprentissage automatique (machine learning) et l'apprentissage profond (deep learning), ont permis à l'IA de s'infiltrer dans divers domaines, de la médecine à la finance, en passant par l'éducation.

1.1 Définition et Portée de l'Intelligence Artificielle

L'intelligence artificielle est un domaine multidisciplinaire qui vise à créer des systèmes capables de simuler ou d'imiter les processus cognitifs humains. Plus précisément, l'IA englobe un ensemble de théories, de méthodes, et de technologies permettant aux machines d'exécuter des tâches qui nécessitent, dans leur essence, une forme d'intelligence, comme l'apprentissage, la perception, le raisonnement, la reconnaissance de modèles, et la prise de décision autonome (Hwang et al., 2020). l'IA peut être vue à travers quatre paradigmes: systèmes qui pensent comme des humains, systèmes qui agissent comme des humains, systèmes qui pensent de manière rationnelle, et systèmes qui agissent de manière rationnelle (Alzoubi et al., 2024).

L'une des approches techniques couramment utilisées dans l'IA est l'apprentissage automatique. Il est divisé en plusieurs types, notamment l'apprentissage supervisé, non supervisé, et par renforcement, chacun d'eux ayant des applications spécifiques (Alzoubi et al., 2024). L'apprentissage profond, une extension de l'apprentissage automatique, utilise des réseaux de neurones artificiels multicouches pour modéliser des relations complexes dans les données, atteignant des niveaux de performance remarquables dans des domaines tels que la reconnaissance vocale ou la traduction automatique (A. Rana et al., 2018).

Dans le contexte éducatif, l'IA a eu un impact considérable en transformant la manière dont les enseignants gèrent les processus d'enseignement et d'apprentissage. L'une des applications majeures de l'IA en éducation est l'exploration de données éducatives (Educational Data Mining : EDM), un domaine qui se concentre sur l'analyse des vastes ensembles de données générés par les systèmes éducatifs pour extraire des connaissances utiles (Amrieh et al., 2016). Grâce à l'EDM, il est possible d'identifier des motifs cachés, des tendances et des relations dans les données scolaires, ce qui permet de personnaliser l'enseignement, de prédire les résultats académiques, et d'optimiser les interventions pédagogiques (M. Li et al., 2018; Premlatha et al., 2016). Par exemple, des modèles prédictifs basés sur l'EDM peuvent être utilisés pour anticiper les élèves à risque de décrochage scolaire et intervenir de manière proactive pour améliorer leur rétention (Barros et al., 2019).

En outre, l'IA est également employée dans la création de systèmes d'apprentissage adaptatif qui ajustent le contenu et le rythme d'apprentissage en fonction des besoins individuels des élèves. Ces systèmes utilisent des algorithmes pour analyser les données de performance en temps réel et fournir des recommandations personnalisées, offrant ainsi une approche plus individualisée de l'éducation (Brisson et al., 2012; Klašnja-Milićević et al., 2011; Premlatha et al., 2016). Cependant, malgré les avantages potentiels de l'IA en éducation, il est essentiel de considérer les défis associés, tels que les questions de biais dans les données, la confidentialité des informations des apprenants, et les implications éthiques de l'utilisation de systèmes automatisés pour prendre des décisions éducatives (Ashman et al., 2014).

1.2 Les Avantages de l'Intelligence Artificielle en Education

L'un des principaux atouts de l'IA dans l'éducation réside dans sa capacité à traiter et à analyser des quantités massives de données en temps réel. Traditionnellement, l'analyse des données dans le secteur éducatif était un processus laborieux et chronophage, nécessitant souvent des semaines, voire des mois, pour extraire des informations utiles. Avec l'IA, ce processus est considérablement accéléré. Les algorithmes d'apprentissage automatique peuvent analyser des données en quelques secondes, permettant aux enseignants de prendre des décisions éclairées et d'adapter leurs méthodes pédagogiques en temps réel (M. Li et al., 2018).

De plus, l'IA offre la possibilité de créer des parcours d'apprentissage personnalisés pour chaque élève. Grâce à l'analyse des données sur les performances, les styles d'apprentissage, et les besoins individuels des apprenants, les systèmes d'IA peuvent recommander des contenus et des activités adaptés à chaque étudiant. Par exemple, les plateformes d'apprentissage adaptatif utilisent des algorithmes pour ajuster le niveau de difficulté des exercices en fonction des progrès de l'apprenant, optimisant ainsi le processus d'apprentissage (Limongelli et al., 2008; Premlatha et al., 2016).

L'IA peut également jouer un rôle crucial dans l'identification précoce des élèves à risque. En analysant des données telles que les notes, les absences, et les interactions en classe, les systèmes d'IA peuvent détecter des motifs de sous-performance et alerter les enseignants afin qu'ils interviennent rapidement. Cette approche proactive peut contribuer à réduire les taux de décrochage scolaire et à améliorer les résultats des élèves (Barros et al., 2019).

1.3 Les Défis et les Limites de l'Intelligence Artificielle en Education

Malgré ses nombreux avantages, l'introduction de l'IA dans le domaine de l'éducation n'est pas sans défis. L'un des principaux obstacles est la complexité de la mise en œuvre des systèmes d'IA. La préparation des données, la gestion des données manquantes, le choix des algorithmes appropriés, et leur configuration correcte sont autant d'étapes cruciales qui nécessitent des compétences techniques spécialisées (Hwang et al., 2020). De plus, les ensembles de données éducatifs sont souvent déséquilibrés et les algorithmes d'IA ne sont pas infaillibles. Ils dépendent fortement de la qualité des données qui leur sont fournies. Si les données sont biaisées ou incomplètes, les résultats obtenus par ces algorithmes d'IA peuvent être inexacts, donnant lieu à des décisions erronées (Radwan & Cataltepe, 2017). Par exemple, un système d'IA basé sur des données historiques peut perpétuer des inégalités en reproduisant des biais existants dans les données, comme les disparités raciales ou socio-économiques (Ashman et al., 2014).

La question de la sécurité des données individuelles se pose avec acuité dans l'utilisation de l'IA en éducation. Les systèmes d'IA collectent et analysent de grandes quantités d'informations sur les élèves, y compris leurs performances académiques, leurs comportements, et parfois même leurs informations personnelles (Ashman et al., 2014). De ce fait, il est indispensable de mettre

en place des mesures de protection des données pour s'assurer que ces informations soient utilisées de la manière la plus éthique et sécurisée possible.

L'Apprentissage Automatique Machine Learning

2 Introduction à L'Apprentissage Automatique

L'apprentissage est à un sous-ensemble de l'IA qui consiste à construire des modèles d'apprentissage capables d'apprendre à partir de l'expérience. Cela implique la création d'algorithmes et de modèles statistiques qui permettent aux machines d'apprendre à partir de données plutôt que d'instructions explicitement programmées. L'apprentissage automatique est particulièrement utile dans le domaine de l'éducation, où les problèmes d'exploration de données sont très fréquents (Khalaf Hamoud et al., 2022). Ces données peuvent inclure des données numériques discrètes ou continues telles que les résultats des apprenants, leurs moyennes, et les interactions d'apprentissage en ligne etc. ou alors des données catégoriques telles que leur niveau de compréhension, leurs types de support pédagogique préféré, leur niveau de satisfaction (Premlatha et al., 2016). De plus, il y a des données textuelles telles que, leurs réponses et les commentaires des enseignants. Ainsi, en utilisant des algorithmes d'apprentissage automatique, l'exploration de données éducatives peut être utilisée afin de :

- En savoir plus sur les profils des apprenants,
- Améliorer leurs résultats,
- Faire des analyses prédictives pour identifier les étudiants à risque,
- Proposer des interventions préventives à chaque profil d'apprenant,
- Adapter l'apprentissage en temps réel en fonction des performances et des besoins de chaque apprenant,
- Améliorer l'expérience d'apprentissage dans son ensemble.

Par ailleurs, il existe deux types principaux d'algorithmes d'apprentissage automatique : l'apprentissage supervisé et l'apprentissage non supervisé. Tout d'abord, l'apprentissage supervisé se repose sur la formation d'un modèle d'apprentissage automatique à l'aide de données étiquetées (Bishop, 2006). Ainsi, l'algorithme d'apprentissage automatique reçoit un ensemble d'entrées et de sorties correspondantes et est entraîné à prédire la sortie pour de nouvelles entrées. Par exemple, dans une application d'apprentissage des langues, un algorithme d'apprentissage automatique supervisé peut être entraîné à prédire la traduction correcte d'un mot en fonction de son contexte. En revanche, l'apprentissage non supervisé implique la formation d'un modèle d'apprentissage automatique à l'aide de données non étiquetées (Conneau et al., 2020). Dans ce cas, l'algorithme d'apprentissage automatique reçoit un ensemble de données sans aucune information de classification ou d'étiquetage et est chargé d'identifier des modèles et des relations dans les données. Ainsi, l'apprentissage non supervisé peut être utilisé dans le domaine de l'éducation pour identifier et grouper des clusters (groupes) d'étudiants ayant des styles d'apprentissage similaires ou pour regrouper des ressources éducatives similaires (Bouchet et al., 2013).

En termes d'avantages, l'utilisation de l'apprentissage automatique dans l'éducation permet de personnaliser l'expérience d'apprentissage pour chaque apprenant, notamment dans le cadre des EIAH (Quan, 2020). En effet, les algorithmes d'apprentissage automatique peuvent être

employés de sorte à analyser les données de l'expérience d'apprentissage d'un apprenant, identifier ses forces et ses faiblesses et adapter les ressources d'apprentissage (cours, exercices, etc.) en conséquence. Ces informations peuvent être utilisées pour éclairer les stratégies pédagogiques et les ajustements de programme (Klašnja-Milićević et al., 2011).

3 Algorithmes d'Apprentissage Automatique

Les algorithmes d'apprentissage automatique représentent un ensemble de méthodes informatiques qui permettent aux systèmes informatiques d'apprendre à partir de données et de prendre des décisions réfléchies (Bengio & Lecun, 2007; Z. Zhang, 2016). Ces algorithmes sont fondamentaux dans divers domaines, de la reconnaissance d'images à la prédiction des tendances financières. Il existe principalement deux types d'algorithmes d'apprentissage automatique : l'apprentissage supervisé et l'apprentissage non supervisé (figure III.1). Les algorithmes supervisés, tels que les réseaux de neurones (Hornik et al., 1990) et les machines à vecteurs de support (Cortes & Vapnik, 1995), apprennent à partir de données étiquetées, tandis que les algorithmes non supervisés, comme la classification ascendante hiérarchique et la réduction de dimensionnalité, identifient des structures et des modèles dans des ensembles de données non étiquetés (Rui Xu & D. Wunsch, 2005). Ces algorithmes présentent toutefois des défis, tels que la nécessité de grandes quantités de données de haute qualité et la complexité de leur interprétabilité (Ribeiro et al., 2016). Nous examinerons dans ce qui suit ces deux algorithmes en détail.

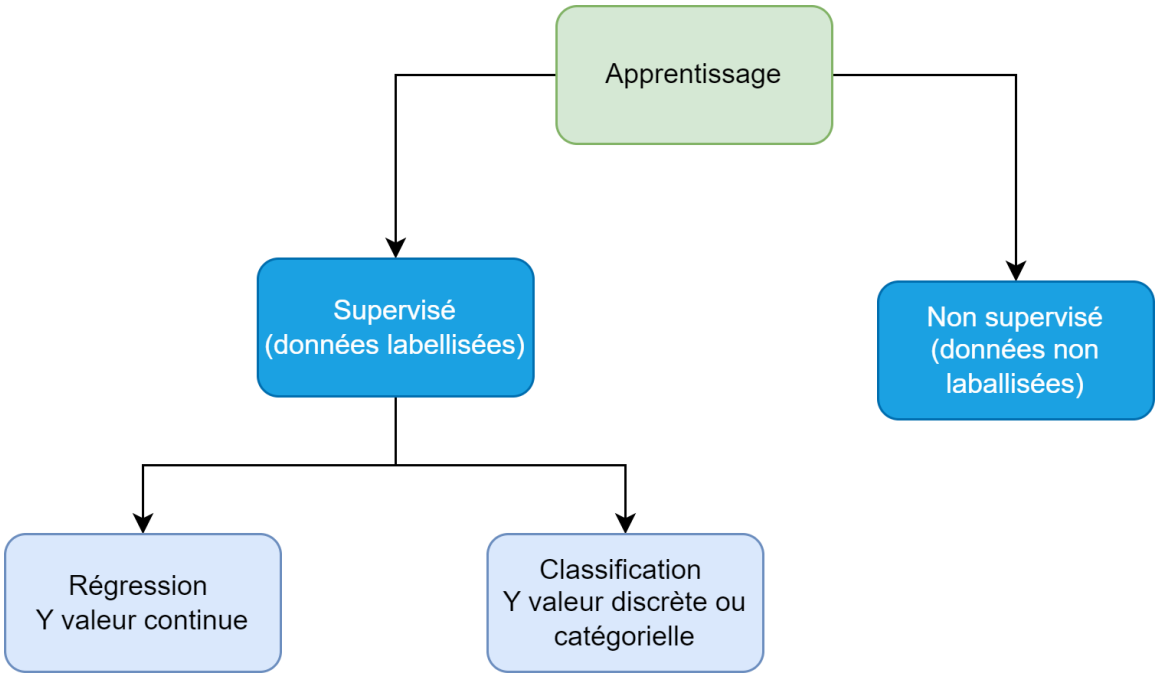

Figure III.1 Les Types d'Apprentissage Automatique Supervisé vs Non Supervisé

3.1 Les Algorithmes d'Apprentissage Supervisé

L'apprentissage supervisé est une technique d'apprentissage automatique qui consiste à entraîner un modèle à partir de données étiquetées. Les données étiquetées sont des données pour lesquelles la sortie souhaitée est connue. En d'autres termes, les algorithmes d'apprentissage supervisé prennent des données étiquetées en entrée et trouvent des relations entre les variables d'entrée et de sortie (Bishop, 2006). Cette relation est ensuite utilisée pour prédire la sortie du nouvel ensemble de données (ensemble de test) sur la base de l'apprentissage

précédent. Ce processus est une sorte de régression où le modèle apprend à prédire la variable de sortie en fonction des variables d'entrée. Par conséquent, les algorithmes d'apprentissage supervisé peuvent être utilisés pour résoudre une variété de problèmes, tels que la classification et la régression. D'ailleurs, dans les tâches de classification, le modèle apprend à prédire la classe (ou la catégorie) des données d'entrée, tandis que dans les tâches de régression, le modèle apprend à prédire une variable continue spécifique à l'ensemble de données (J. Han et al., 2012; James et al., 2021). Il existe plusieurs algorithmes d'apprentissage supervisé qui fonctionnent selon des principes différents. Les algorithmes utilisés dans nos recherches sont détaillés ci-dessous.

3.1.1 La Régression Logistique

La régression logistique est une méthode statistique largement utilisée pour la classification binaire, permettant de prédire la probabilité qu'une observation appartienne à l'une des deux catégories possibles. Contrairement à la régression linéaire qui modélise la relation entre des variables continues, la régression logistique modélise la probabilité qu'un événement se produise, en utilisant la fonction logistique, aussi appelée fonction sigmoïde. La régression logistique s'appuie sur la transformation logistique appliquée à une combinaison linéaire des variables indépendantes (Wilson & Lorenz, 2015). Mathématiquement, cela s'exprime ainsi :

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i X_i)}}$$

où :

- $(p(y = 1|X))$ est la probabilité que l'événement (y) (classe 1) se produise, étant donné le vecteur de variables indépendantes $(X = (X_1, X_2, \dots, X_n))$.
- (β_0)est l'ordonnée à l'origine (intercept) du modèle.
- (β_i) sont les coefficients des variables indépendantes (X_i), qui sont estimés à partir des données.
- (e) est la base du logarithme naturel, approximativement égale à 2,718.

Le résultat de cette fonction logistique se situe entre 0 et 1, représentant ainsi une probabilité. Une valeur seuil (typiquement 0,5) est ensuite appliquée pour classifier l'observation en une des deux catégories.

Dans le domaine de l'éducation, la régression logistique est fréquemment utilisée pour prédire des résultats binaires tels que le succès ou l'échec académique d'un élève en fonction de divers facteurs prédictifs, tels que les notes antérieures, la fréquentation des cours, le statut socio-économique, etc.

Les Avantages de la Régression Logistique

- Prédiction des performances des élèves: la régression logistique peut aider à identifier les élèves à risque d'échec scolaire. Par exemple, en utilisant des données historiques sur les performances académiques, un modèle peut prédire la probabilité qu'un élève obtienne une mauvaise note dans une future évaluation (Dekker et al., 2009).
- Identification des facteurs déterminants: cet algorithme permet également d'identifier les variables les plus influentes sur la performance des élèves. Cela peut aider les éducateurs à se concentrer sur les facteurs clés qui affectent les résultats académiques, tels que le temps d'étude ou la participation en classe (David W. Hosmer Jr. et al., 2013).
- Evaluation de l'efficacité des programmes: en comparant les résultats avant et après la mise en œuvre d'un programme éducatif, la régression logistique peut être utilisée pour évaluer l'impact de ce programme. Par exemple, un modèle peut évaluer l'efficacité

d'une nouvelle méthode d'enseignement en prédisant la probabilité d'amélioration des notes des élèves après son introduction (David W. Hosmer Jr. et al., 2013).

Les Limites de la Régression Logistique

- Incapacité à modéliser les facteurs qualitatifs complexes: la régression logistique se concentre sur les variables quantitatives et n'intègre pas facilement les facteurs qualitatifs tels que la motivation, l'engagement ou l'attitude des élèves (Alyahyan & Düştegör, 2020).
- Absence de modélisation des interactions complexes: bien que certaines interactions entre variables puissent être intégrées dans le modèle, la régression logistique classique ne capture pas facilement les interactions complexes ou non linéaires entre les différents facteurs prédictifs. Cela peut conduire à des prédictions erronées si ces interactions sont significatives (James et al., 2021).
- Limites des facteurs externes: la régression logistique ne prend pas non plus en compte les variables externes telles que les événements sociaux ou économiques qui peuvent affecter la performance des élèves de manière significative. Ces facteurs contextuels nécessitent souvent des approches plus complexes pour être modélisés avec précision (Gelman & Hill, 2006).

Bien que la régression logistique offre des outils puissants pour la prédiction et l'évaluation dans le domaine éducatif, elle présente certaines limitations qui doivent être prises en compte lors de son utilisation.

3.1.2 Arbres de Décision

Les arbres de décision constituent une méthode d'apprentissage supervisé largement employée tant pour la classification que pour la régression. Ils se manifestent sous la forme d'une structure arborescente, où chaque nœud interne représente un test appliqué à une caractéristique (ou attribut) spécifique de l'ensemble de données, chaque branche incarne le résultat de ce test, et chaque feuille correspond à une classe cible ou une valeur prédite (Yohannes & Hoddinott, 1999). Cette architecture hiérarchique permet de modéliser de manière intuitive et visuelle les relations complexes entre une variable dépendante et plusieurs variables indépendantes (figure III.2).

L'algorithme fondamental pour la construction d'un arbre de décision est le Recursive Binary Splitting. Cet algorithme procède à une division récursive de l'ensemble de données en sous-ensembles, fondée sur les valeurs des attributs, dans le but d'optimiser la séparation des classes cibles. Pour les arbres de classification, les critères les plus couramment utilisés pour évaluer la qualité de cette division sont l'impureté de Gini et l'entropie (Breslow & Aha, 1996). En revanche, pour les arbres de régression, c'est souvent la somme des erreurs quadratiques qui est privilégiée.

La mesure de Gini évalue la pureté des nœuds après division. Elle est calculée comme suit :

$$\text{Gini}(t) = 1 - \sum (i = 1)^{C} p_i^2$$

où (p_i) est la proportion d'observations de la classe (i) dans le nœud (t), et (C) est le nombre total de classes. Un Gini de 0 correspond à un nœud pur, c'est-à-dire où toutes les observations appartiennent à une seule classe.

L'entropie mesure le degré de désordre ou d'incertitude dans les données :

$$\text{Entropy}(t) = -\sum (i = 1)^{C} p_i \log_2(p_i)$$

où (p_i) est, comme dans le cas du Gini, la proportion d'observations de la classe (i).

Par ailleurs, pour un arbre de décision régressif, l'objectif est de minimiser l'erreur quadratique moyenne, définie par :

$$\text{MSE} = \frac{1}{n}\sum(i = 1)^n(y_i - \hat{y}_i)^2$$

où (y_i) est la valeur réelle et ($\hat{y}_i$) est la valeur prédite.

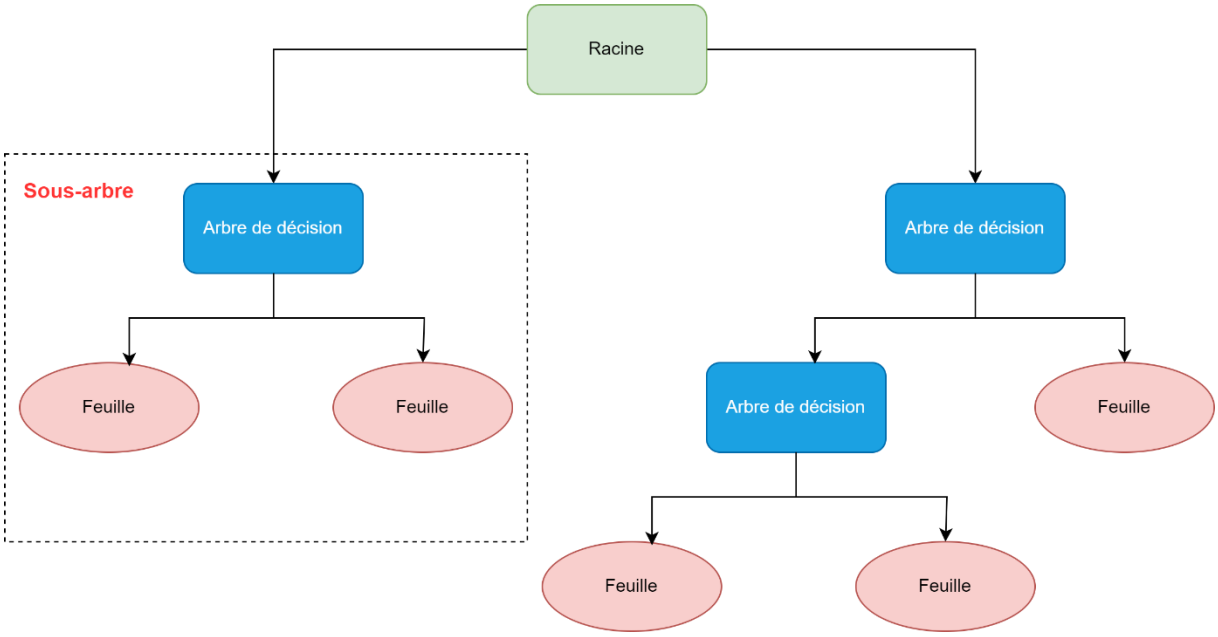

Figure III.2 Algorithme Arbre de Décision

Les Avantages des Arbres de Décision

- Facilité d'interprétation et de visualisation: l'une des forces majeures des arbres de décision réside dans leur nature visuelle et intuitive. Les règles de décision sont représentées de manière hiérarchique, ce qui permet de suivre aisément le cheminement logique menant à une prédiction donnée. Cette transparence est particulièrement utile dans les contextes où l'explicabilité des modèles est nécessaire.
- Robustesse face à des données peu préparées: contrairement à d'autres algorithmes d'apprentissage automatique, les arbres de décision ne nécessitent pas de normalisation ou de transformation préalable des données. Ils peuvent gérer à la fois des variables numériques et catégorielles sans nécessiter de transformations complexes (Breslow & Aha, 1996).
- Polyvalence: les arbres de décision peuvent être utilisés aussi bien pour des tâches de classification que de régression. De plus, ils sont capables de modéliser des relations non linéaires sans nécessiter de transformations explicites des variables d'entrée (Mienye & Jere, 2024).
- Performance globale: bien qu'ils ne soient pas toujours les plus performants dans des contextes très complexes, les arbres de décision offrent une bonne performance générale, en particulier sur des ensembles de données de taille moyenne, où ils peuvent souvent surpasser des modèles plus complexes en termes de rapidité et de simplicité.

Les Limites des Arbres de Décision

- Biais vers la classe dominante: les arbres de décision peuvent être biaisés envers la classe majoritaire dans les ensembles de données déséquilibrés. Il faut rééquilibrer les données avant l'entraînement, par des techniques telles que le sur-échantillonnage ou le sous-échantillonnage (Siers & Islam, 2020).
- Complexité accrue avec des hypothèses complexes: à mesure que l'arbre se développe et que les hypothèses se multiplient, l'arbre peut devenir excessivement complexe et difficile à interpréter. Cela peut également entraîner des problèmes de surajustement (overfitting), où l'arbre s'ajuste trop étroitement aux particularités de l'ensemble d'entraînement, au détriment de sa capacité à généraliser sur des données nouvelles (Breslow & Aha, 1996).
- Surapprentissage et manque de généralisation: le surapprentissage est un problème majeur avec les arbres de décision. Cela se produit lorsque l'arbre s'ajuste trop aux données d'entraînement, capturant des bruits aléatoires plutôt que des motifs sous-jacents. Cela conduit à une mauvaise performance sur les ensembles de données de test. Des techniques comme la taille de l'arbre (pruning) ou l'utilisation d'ensembles (e.g., forêt aléatoire) sont souvent utilisées pour atténuer ce problème (Siers & Islam, 2020).

3.1.3 Machines à Vecteurs de Support (SVM)

Les machines à vecteurs de support (SVM) sont des algorithmes d'apprentissage supervisé utilisés principalement pour les tâches de classification, bien qu'ils puissent également être appliqués à des problèmes de régression. Introduites par (Cortes & Vapnik, 1995), les SVM reposent sur l'idée de trouver la meilleure limite de séparation (ou hyperplan) entre différentes classes de données, de manière à maximiser la marge de séparation entre les points de données des différentes classes (figure III.3).

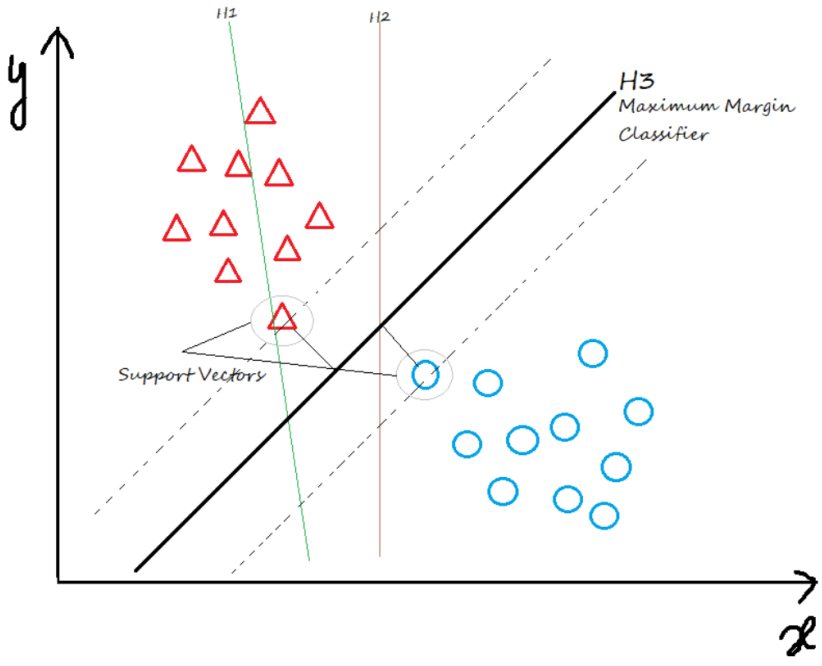

Figure III.3 Algorithme SVM (Bonthu, 2021)

Les SVM sont des généralisations des classifieurs linéaires, mais elles peuvent être étendues à des problèmes non linéaires à l'aide de techniques telles que les noyaux (kernels). L'objectif principal d'un SVM est de trouver l'hyperplan qui maximise la marge, c'est-à-dire la distance entre l'hyperplan et les points de données les plus proches de chaque classe, appelés vecteurs de support (Bonthu, 2021).

Pour un problème de classification binaire, étant donné un ensemble de données d'entraînement $(\{(x_i, y_i)\}_{i=1}^{n})$où$(x_i)$ représente les vecteurs d'entrée et $(y_i \in \{-1, +1\})$ les étiquettes correspondantes, l'objectif est de trouver un hyperplan défini par le vecteur de poids (w) et le biais (b) tel que :

$$f(x) = w^T x + b$$

Le SVM cherche à minimiser la fonction objectif suivante :

$$\min_{w,b}(1/2) * ||w||^2$$

sous les contraintes de classification correcte pour tous les points de données :

$$y_i(w^T x_i + b) \geq 1 \quad \forall i$$

Dans les cas où les données ne sont pas linéairement séparables, on peut recourir à la formulation duale du problème et à l'utilisation de fonctions noyaux (kernels) pour projeter les données dans un espace de dimension plus élevée où elles deviennent linéairement séparables. La fonction objectif duale est exprimée en termes de multiplicateurs de Lagrange (α_i), mathématiquement exprimée comme suit:

$$\max_{\alpha}\left(\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(x_i, x_j)\right)$$

sous les contraintes:

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \quad \text{et} \quad 0 \leq \alpha_i \leq C$$

où $(K(x_i, x_j))$est la fonction noyau (par exemple, linéaire, polynomiale, ou Gaussienne RBF) et (C) est un paramètre de régularisation qui contrôle le compromis entre la maximisation de la marge et la minimisation de l'erreur de classification.

Parmi les Avantages des SVM, on cite :

- Efficacité pour les ensembles de données de grande taille: les SVM sont particulièrement efficaces pour les ensembles de données avec un grand nombre de caractéristiques (variables d'entrée). Leur capacité à gérer des espaces de grande dimension est due à l'utilisation de noyaux, qui permettent de traiter des problèmes non linéaires sans avoir à transformer explicitement les données (Chauhan et al., 2019).
- Robustesse pour les caractéristiques non linéaires: les SVM peuvent gérer des ensembles de données comportant des relations non linéaires entre les caractéristiques grâce à l'utilisation de fonctions noyaux. Ces noyaux transforment les données dans un espace de caractéristiques où un hyperplan linéaire peut séparer les classes (Chandra & Bedi, 2018).

En revanche, certaines limites sont à considérer notamment :

- Sensibilité aux paramètres de réglage: les performances des SVM dépendent fortement des paramètres de réglage, tels que le paramètre de régularisation (C) et les paramètres de la fonction noyau. Un mauvais choix de ces paramètres peut entraîner un surajustement ou un sous-ajustement du modèle, ce qui affecte sa performance générale. L'optimisation de ces paramètres requiert souvent des méthodes de recherche par validation croisée (Bonthu, 2021).

- Sensibilité aux données aberrantes: les SVM sont sensibles aux valeurs aberrantes, car ces dernières peuvent influencer la position de l'hyperplan de séparation. Les outliers, en étant mal classés, peuvent réduire la marge de séparation et dégrader les performances du modèle (Chandra & Bedi, 2018).
- Lenteur pour les grands ensembles de données: bien que les SVM soient performantes pour les ensembles de données de grande dimension, leur temps d'entraînement peut devenir prohibitif lorsque le nombre d'échantillons est très élevé. Cela est dû à la nécessité de résoudre un problème d'optimisation quadratique, qui devient plus complexe et plus long à mesure que la taille des données augmente (Chauhan et al., 2019).

Les machines à vecteurs de support sont une méthode puissante et flexible pour résoudre des problèmes de classification et de régression, en particulier dans des contextes où les données présentent des caractéristiques complexes et non linéaires. La sélection des noyaux, l'ajustement des paramètres et l'utilisation de techniques complémentaires peuvent maximiser l'efficacité des SVM dans des situations variées.

3.1.4 Les k plus Proches Voisins (kNN)

L'algorithme des k plus proches voisins (k-Nearest Neighbors, kNN) est une méthode d'apprentissage supervisé simple mais efficace, utilisée pour les tâches de classification et de régression. Proposé par (Silverman & Jones, 1989), le kNN se base sur l'idée que les observations similaires devraient avoir des résultats similaires. Lorsqu'un nouvel exemple est présenté, l'algorithme identifie les (k) exemples les plus proches dans l'ensemble de données d'entraînement et prédit la sortie en fonction des étiquettes ou des valeurs de ces (k) voisins.

L'algorithme kNN ne repose pas sur une phase d'entraînement explicite, mais fonctionne en stockant simplement tous les exemples d'entraînement et en calculant la similarité entre le nouvel exemple et les exemples d'entraînement lors de la prédiction.

La mesure de la similarité entre deux exemples est souvent basée sur la distance euclidienne. Pour deux vecteurs d'entrée $(x_i) et (x_j)$, la distance euclidienne est définie comme:

$$d(x_i, x_j) = sqrt\left(\sum (m = 1)^M \left(x_i^m - x_j^m\right)^2\right)$$

où (M) est le nombre de caractéristiques. D'autres mesures de distance, comme la distance de Manhattan ou la distance de Minkowski, peuvent également être utilisées en fonction des caractéristiques des données.

Pour une tâche de classification, l'étiquette prédite pour un nouvel exemple est déterminée par un vote majoritaire parmi les (k) voisins les plus proches. Si $(k = 1)$, l'étiquette de l'exemple le plus proche est attribuée. Pour une tâche de régression, la prédiction est souvent la moyenne des valeurs des (k) voisins les plus proches. Le choix de (k), qui représente le nombre de voisins à considérer, est un paramètre déterminant dans cet algorithme. Un choix trop petit rend le modèle sensible au bruit (overfitting), tandis qu'un (k) trop grand peut diluer la pertinence des voisins les plus proches.

Les Avantages du kNN

- Simplicité d'implémentation: l'un des principaux avantages du kNN est sa simplicité. Il ne nécessite aucun modèle complexe ou phase d'entraînement explicite. L'algorithme repose simplement sur le calcul des distances entre les points, ce qui le rend facile à comprendre et à mettre en œuvre (Z. Zhang, 2016).
- Efficacité pour les petits ensembles de données: le kNN est particulièrement efficace pour les petits ensembles de données. Comme l'algorithme ne nécessite pas

d'entraînement préalable et effectue simplement des calculs de distance, il est capable de fonctionner rapidement sur des ensembles de données de petite taille (Z. Zhang, 2016).

- Adaptabilité aux caractéristiques non linéaires: le kNN fonctionne bien dans des situations où les relations entre les variables d'entrée et la sortie ne sont pas linéaires. En utilisant la similarité locale entre les exemples, il peut capturer des tendances complexes dans les données sans avoir besoin de modéliser explicitement la non-linéarité (Duda et al., 2001).

Les limites du kNN

- Sensibilité aux données aberrantes: comme le kNN se base sur les points les plus proches, il peut être fortement influencé par les outliers (valeurs aberrantes). Un outlier peut fausser la prédiction en attirant l'étiquette vers une classe incorrecte ou en affectant la valeur prédite lors d'une tâche de régression (i & Herrera, 2008).
- Dépendance aux paramètres de réglage: le choix du paramètre (k) et de la mesure de distance utilisée influence grandement la performance du modèle. Un mauvais choix de ces paramètres peut entraîner une dégradation des performances. La validation croisée est souvent utilisée pour sélectionner le meilleur paramètre (k), mais cela peut devenir coûteux en termes de calcul (Z. Zhang, 2016).
- Lenteur pour les grands ensembles de données: l'algorithme kNN devient inefficace pour les grands ensembles de données. Le besoin de calculer les distances entre un nouvel exemple et tous les exemples d'entraînement entraîne un temps de calcul important, ce qui rend l'algorithme peu adapté aux contextes où la rapidité est essentielle. Des structures de données telles que les arbres k-d peuvent être utilisées pour accélérer les recherches, mais elles ne résolvent pas toujours le problème (Beyer et al., 1999).

Le kNN est un algorithme de classification et de régression puissant et simple, adapté à des tâches nécessitant une faible complexité computationnelle sur des ensembles de données de petite taille ou présentant des non-linéarités.

3.1.5 Naive Bayes

Le Naive Bayes est une famille de classifieurs probabilistes basée sur l'application du théorème de Bayes, avec une hypothèse forte d'indépendance entre les caractéristiques. Les classifieurs Naive Bayes sont largement utilisés en raison de leur efficacité, de leur simplicité, et de leur performance surprenante dans de nombreuses applications, notamment le filtrage de spam, la classification de texte, et la détection d'anomalies (Duda et al., 2001).

Le classifieur Naive Bayes repose sur l'application du théorème de Bayes, qui exprime la probabilité conditionnelle d'une classe (C) donnée un ensemble de caractéristiques $(X = (x_1, x_2, \dots, x_n))$ comme suit :

$$P(C \mid X) = \frac{P(X \mid C) \cdot P(C)}{P(X)}$$

Où :

- $(P(C|X))$ est la probabilité a posteriori de la classe (C) étant donnée les caractéristiques (X),
- $(P(X|C))$est la probabilité de l'ensemble des caractéristiques (X) étant donnée la classe (C) ,
- $(P(C))$ est la probabilité a priori de la classe (C) ,
- $(P(X))$ est la probabilité des caractéristiques (X).

Le terme "naive" dans Naive Bayes vient de l'hypothèse simplificatrice que les caractéristiques (x_i) sont conditionnellement indépendantes les unes des autres étant donné la classe (C). Cela permet de factoriser la probabilité ($P(X|C)$) comme suit :

$$P(X \mid C) = \prod_{i=1}^{n} P(x_i \mid C)$$

Ainsi, l'expression de la probabilité a posteriori devient :

$$P(C \mid X) \propto P(C) \cdot \prod_{i=1}^{n} P(x_i \mid C)$$

Ce classifieur attribue alors l'étiquette de classe ($\hat{C}$) à l'exemple pour lequel la probabilité a posteriori est maximale :

$$\hat{C} = \arg\max_{C} \left[P(C) \cdot \prod_{i=1}^{n} P(x_i \mid C) \right]$$

Il existe plusieurs variantes du classifieur Naive Bayes, adaptées à différents types de données:

- Naive Bayes Gaussien: utilisé pour les données continues, où la distribution conditionnelle des caractéristiques est supposée suivre une distribution normale (Gaussienne). Dans ce cas, les probabilités conditionnelles ($P(x_i|C)$) sont modélisées par une fonction de densité gaussienne:

  $$P(x_i \mid C) = \frac{1}{\sqrt{2\pi\sigma_C^2}} \exp\left(- \frac{(x_i - \mu_C)^2}{2\sigma_C^2} \right)$$

  Où (μ_C) et (σ_C^2) sont la moyenne et la variance de la caractéristique (x_i)pour la classe (C).
- Naive Bayes Multinomial: principalement utilisé pour la classification de texte, où les caractéristiques représentent des comptes discrets (par exemple, le nombre de fois qu'un mot apparaît dans un document). Le modèle suit une distribution multinomiale:

  $$P(x_i|C) = \frac{n_{C,x_i} + \alpha}{n_C + \alpha N}$$

  Où (n_{C,x_i}) est le nombre de fois que le mot (x_i) apparaît dans les documents de la classe(C), (n_C) est le nombre total de mots dans la classe (C), (N) est le nombre total de mots dans le vocabulaire, et (α) est un paramètre de lissage de Laplace.
- Naive Bayes Bernoulli: algorithme approprié pour des données binaires, où chaque caractéristique est modélisée comme une variable de Bernoulli (présence ou absence d'une caractéristique).

Les Avantages de Naive Bayes

- Simplicité: le Naive Bayes est très simple à implémenter et rapide à entraîner, même sur de grands ensembles de données. En raison de sa structure simple, il est également très efficace en termes de calculs (H. Zhang, 2004).
- Robustesse à l'overfitting: le classifieur Naive Bayes est souvent robuste à l'overfitting, même avec l'hypothèse d'indépendance souvent irréaliste. Cela en fait un excellent choix comme modèle de référence pour de nombreux problèmes de classification (Rish, 2001).

- Efficacité pour les grandes dimensions: Naive Bayes gère bien les ensembles de données avec un grand nombre de caractéristiques (par exemple, les modèles de texte). Sa simplicité structurelle permet de traiter efficacement des ensembles de données de haute dimension.

Les Limites de Naive Bayes

- Hypothèse d'indépendance irréaliste: l'hypothèse d'indépendance conditionnelle entre les caractéristiques est rarement vérifiée dans la pratique. Cette hypothèse simplifie considérablement les calculs, mais peut réduire la précision du modèle lorsqu'il existe des corrélations fortes entre les caractéristiques (Domingos & Pazzani, 1997).
- Problèmes avec les données rares: dans le cas de données catégorielles, Naive Bayes peut attribuer une probabilité nulle à une classe si une combinaison de caractéristiques n'a jamais été observée dans les données d'entraînement. Cela peut être atténué par des techniques de lissage, mais reste un problème potentiel (J. Han et al., 2012).
- Sensibilité aux mauvaises estimations de probabilités: en présence de données biaisées ou de petites tailles d'échantillons, les estimations des probabilités a priori et conditionnelles peuvent être peu fiables, affectant ainsi la performance du modèle (N. Friedman et al., 1997).

Le classifieur Naive Bayes reste un choix pratique pour de nombreux problèmes de classification, notamment lorsque la simplicité et la rapidité sont des critères importants.

3.1.6 Perceptron Multicouches (MLP)

Le Perceptron Multicouches (MLP) est l'un des types les plus courants de réseaux de neurones artificiels utilisés en apprentissage automatique pour les tâches de classification et de régression. Il s'agit d'une extension du perceptron simple, introduit par (Rosenblatt, 1958), qui permet de modéliser des relations non linéaires complexes grâce à l'ajout de plusieurs couches de neurones. Le MLP est une architecture de réseau de neurones feedforward entièrement connecté, où les neurones sont organisés en couches et chaque neurone d'une couche est connecté à tous les neurones de la couche suivante (Rumelhart et al., 1986).

Architecture d'un MLP

Un MLP est composé de trois types de couches principales (voir figure III.4) :

1) Couche d'entrée (Input Layer): cette couche reçoit les données d'entrée, où chaque neurone représente une caractéristique du jeu de données.
2) Couches cachées (Hidden Layers): ces couches, situées entre l'entrée et la sortie, permettent au réseau d'apprendre des représentations abstraites des données. Le nombre de couches cachées et de neurones dans chaque couche détermine la profondeur et la capacité d'expression du modèle.
3) Couche de sortie (Output Layer): cette couche produit la sortie finale du réseau. Dans une tâche de classification, cette couche utilise généralement une fonction d'activation softmax pour fournir des probabilités de classe. Pour les tâches de régression, elle peut utiliser une fonction d'activation linéaire.

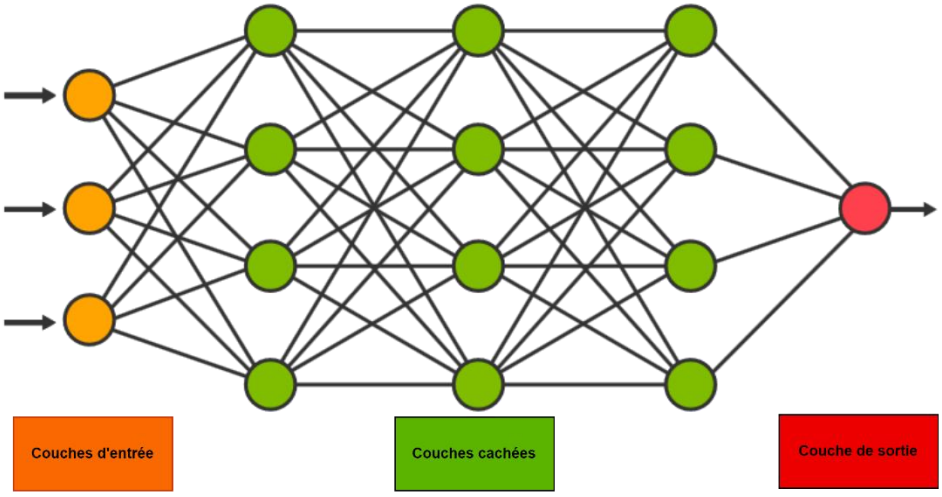

Figure III.4 Architecture d'un Perceptron Multicouches

Le MLP utilise des fonctions d'activation non linéaires pour introduire de la non-linéarité dans le modèle, ce qui permet d'apprendre des relations complexes. Les fonctions d'activation courantes incluent la fonction sigmoïde, la tangente hyperbolique (tanh), et la fonction Rectified Linear Unit (ReLU).

1) Propagation avant (Forward Propagation)

Lors de la phase de propagation avant, les données d'entrée traversent le réseau, couche par couche, en étant multipliées par les poids associés aux connexions entre les neurones. À chaque neurone, une fonction d'activation est appliquée à la somme pondérée des entrées pour produire une sortie. La sortie de chaque couche cachée devient l'entrée de la couche suivante, jusqu'à atteindre la couche de sortie.

2) Fonction de coût (Cost Function)

Pour évaluer la performance du réseau, une fonction de coût est utilisée. Dans une tâche de classification, la fonction de coût couramment utilisée est l'entropie croisée :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\widehat{y^{(i)}}\right) + \left(1-y^{(i)}\right)\log\left(1-\widehat{y^{(i)}}\right)\right]$$

Où $(\widehat{y^{(i)}})$ est la prédiction du modèle pour l'exemple (i), $(y^{(i)})$ est la vraie étiquette, et (m) est le nombre d'exemples dans le jeu de données.

3) Rétropropagation (Backpropagation)

Le MLP est entraîné en ajustant les poids du réseau afin de minimiser la fonction de coût. Cela est réalisé à l'aide de l'algorithme de rétropropagation, qui repose sur la descente de gradient stochastique. La rétropropagation calcule les gradients des erreurs par rapport aux poids en utilisant la règle de la chaîne, et les poids sont mis à jour dans la direction opposée au gradient:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Où (η) est le taux d'apprentissage, et $(\nabla_\theta J(\theta))$ est le gradient de la fonction de coût par rapport aux poids (θ).

Les Avantages du Perceptron Multicouches

- Capacité à modéliser des relations non linéaires: grâce à l'utilisation de fonctions d'activation non linéaires et de multiples couches cachées, le MLP peut apprendre des représentations complexes et capturer des tendances non linéaires dans les données, ce qui le rend puissant pour une variété de tâches d'apprentissage.
- Flexibilité et adaptabilité: Le MLP est capable de résoudre une large gamme de problèmes, de la classification binaire aux problèmes de régression multivariée. Il est également adaptable à différentes tailles et types de données, y compris les données continues et catégorielles (A. Rana et al., 2018).
- Capacité d'apprentissage universelle: théoriquement, un MLP avec une seule couche cachée contenant un nombre suffisant de neurones peut approximer n'importe quelle fonction continue, ce qui fait de lui un approximateur universel (théorème d'approximation universelle de (Hornik et al., 1990).

Les Limites du Perceptron Multicouches

- Complexité computationnelle: le MLP peut être gourmand en calcul, notamment pour les grands réseaux avec de nombreuses couches cachées et de nombreux neurones. L'entraînement du réseau peut être long et nécessite souvent des ressources matérielles puissantes, comme des GPU (Bengio & Lecun, 2007).
- Problème de surapprentissage: avec une capacité d'expression importante, le MLP peut facilement surapprendre les données d'entraînement, en particulier lorsque ces dernières sont limitées. Des techniques de régularisation telles que la dropout ou la régularisation L2 sont souvent nécessaires pour atténuer ce problème (Srivastava et al., 2014).
- Nécessité de réglage des hyperparamètres: le MLP comporte de nombreux hyperparamètres à régler (par exemple, le nombre de couches, le nombre de neurones, le taux d'apprentissage), ce qui peut rendre son utilisation difficile. Une recherche systématique des meilleurs hyperparamètres, souvent à travers des techniques comme la validation croisée, est indispensable pour obtenir des performances optimales (Bergstra & Bengio, 2012).

Bien que le Perceptron Multi-Couches soit un modèle d'apprentissage puissant et flexible, capable de capturer des relations non linéaires complexes, il n'est pas exempt de certaines limitations. En effet, le MLP peut parfois manquer de robustesse, notamment face à la variance des données ou au risque de surapprentissage. C'est ici que les approches ensemblistes entrent en jeu. Ces méthodes exploitent la diversité et la complémentarité de plusieurs modèles pour améliorer la précision et la généralisation des prédictions.

3.2 Approche Ensembliste

Les approches ensemblistes en apprentissage automatique représentent une avancée notable dans l'optimisation des performances des modèles de prédiction. Elles reposent sur le principe fondamental que la combinaison de plusieurs modèles peut engendrer des performances globales supérieures à celles obtenues par chaque modèle pris isolément (Hastie et al., 2009). Cette approche s'appuie sur l'idée que différents modèles ont la capacité de capturer diverses caractéristiques des données, et leur agrégation peut ainsi atténuer les biais et les variabilités propres à chaque modèle individuel (Batista et al., 2004). L'émergence des techniques ensemblistes se révèle particulièrement significative dans les domaines de la classification et de la régression, où elles permettent de surmonter les limitations des modèles de base en réduisant le risque de surapprentissage et en améliorant la capacité de généralisation. Parmi les techniques d'ensemble les plus utilisées, le bagging, le boosting et le stacking se distinguent par

leur capacité à combiner de manière astucieuse les résultats de multiples modèles afin d'obtenir des prédictions précises.

Le bagging, ou Bootstrap Aggregating, consiste à entraîner plusieurs modèles de manière indépendante sur des sous-ensembles aléatoires des données d'apprentissage, générés par rééchantillonnage avec remplacement (M. Galar et al., 2012). Le modèle final résulte de la combinaison des prédictions des différents modèles, généralement par vote majoritaire pour la classification ou par moyenne pour la régression. L'algorithme de forêt aléatoire est un exemple emblématique de cette approche. Le modèle combine des arbres de décision pour améliorer la robustesse et la précision des prédictions (Breiman, 2001). En revanche, le boosting se caractérise par la construction séquentielle des modèles, chaque nouveau modèle étant conçu pour corriger les erreurs commises par les modèles précédents. Ce processus ajuste les poids des instances mal classées afin de corriger les prédictions erronées (Freund & Schapire, 1997). Des algorithmes tels que AdaBoost et Gradient Boosting illustrent parfaitement cette méthode, offrant des variations spécifiques permettant d'optimiser les performances en fonction des caractéristiques des données. Enfin, le stacking implique la combinaison de plusieurs modèles de base à l'aide d'un modèle méta, souvent appelé classifieur de niveau supérieur. Ce modèle méta apprend à partir des prédictions des modèles de base pour améliorer la décision finale (Dey & Mathur, 2023). Ci-dessous, nous allons détailler les algorithmes utilisés dans le cadre de nos recherches.

3.2.1 Forêt Aléatoire

Les forêts aléatoires sont un algorithme d'apprentissage supervisé basé sur le principe de l'ensemble, qui combine les prédictions de plusieurs arbres de décision pour améliorer la précision des prédictions (figure III.5). Introduite par (Breiman, 2001), la forêt aléatoire est une extension de l'approche des arbres de décision, qui vise à atténuer les limitations de ces derniers, notamment le surapprentissage et la sensibilité aux variations dans les données. L'idée clé derrière les forêts aléatoires est de construire plusieurs arbres de décision indépendants lors de l'entraînement, en introduisant de la variabilité à la fois dans les sous-ensembles de données et dans les caractéristiques utilisées pour construire chaque arbre. Cela permet d'obtenir des prédictions plus robustes en agrégeant les résultats de ces arbres, ce qui réduit la variance globale du modèle.

1) **Bootstrap Aggregating (Bagging) : l**e processus de construction d'une forêt aléatoire commence par l'application du **bagging** (bootstrap aggregating). Cette technique consiste à générer plusieurs sous-échantillons de l'ensemble de données d'entraînement en échantillonnant avec remise. Chaque sous-échantillon est ensuite utilisé pour entraîner un arbre de décision distinct. Ce processus permet de réduire la variance des prédictions, car chaque arbre est exposé à une version légèrement différente des données d'entraînement.
2) **Sélection aléatoire des caractéristiques:** une autre caractéristique clé des forêts aléatoires est la sélection aléatoire d'un sous-ensemble de caractéristiques à chaque nœud lors de la construction de chaque arbre. Contrairement aux arbres de décision classiques qui considèrent toutes les caractéristiques disponibles à chaque division, les forêts aléatoires limitent cette sélection, ce qui introduit de la diversité entre les arbres et empêche qu'un petit nombre de caractéristiques dominent les décisions à travers tous les arbres.
3) **Vote majoritaire et moyenne:** pour les problèmes de classification, les forêts aléatoires utilisent le principe du **vote majoritaire** pour agréger les prédictions des arbres individuels. Chaque arbre vote pour une classe, et la classe avec le plus grand nombre

de votes est choisie comme prédiction finale. Pour les problèmes de régression, la prédiction finale est la **moyenne** des prédictions de tous les arbres.

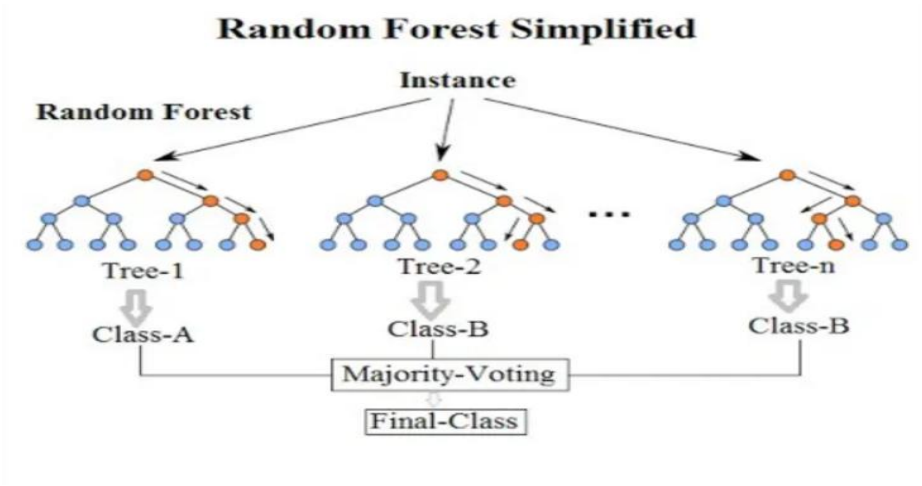

Figure III.5 La Forêt Aléatoire Simplifiée (Koehrsen, 2020)

Les Avantages des Forêts Aléatoires

- **Précision accrue**: en combinant les résultats de plusieurs arbres de décision, les forêts aléatoires réduisent la variance des prédictions et, par conséquent, améliorent la précision globale du modèle. Cette approche d'ensemble permet de compenser les erreurs commises par des arbres individuels, résultant en des prédictions plus fiables (Breiman, 2001).
- **Robustesse face aux données manquantes et aux valeurs aberrantes**: contrairement aux arbres de décision individuels, les forêts aléatoires sont moins sensibles aux valeurs manquantes et aux valeurs aberrantes. Les arbres individuels dans une forêt peuvent être construits avec des sous-échantillons de données, ce qui réduit l'impact des données manquantes sur le modèle global (Cutler et al., 2007).
- **Polyvalence pour la classification et la régression**: les forêts aléatoires sont efficaces à la fois pour les tâches de classification et de régression. Elles sont capables de gérer des problèmes complexes et non linéaires, avec des ensembles de données comportant des variables numériques et catégorielles (Breiman, 2001).

Les Limites des Forêts Aléatoires

- **Lenteur lors de l'entraînement**: l'une des principales limitations des forêts aléatoires est le temps d'entraînement. La nécessité de construire un grand nombre d'arbres de décision peut rendre l'entraînement long et coûteux en termes de ressources informatiques, en particulier pour de grands ensembles de données. Cette complexité algorithmique peut être un frein dans des contextes où la rapidité est essentielle (Oshiro et al., 2012).
- **Difficulté d'interprétation**: contrairement aux arbres de décision individuels, qui sont facilement interprétables en raison de leur structure simple, les forêts aléatoires sont beaucoup plus complexes. La combinaison des résultats de plusieurs arbres rend difficile l'interprétation des relations entre les variables d'entrée et la variable de sortie. Cette boîte noire représente souvent un inconvénient (Biau & Scornet, 2016).

3.2.2 AdaBoost

L'AdaBoost (Adaptive Boosting) est un algorithme d'ensemble introduit par (Freund & Schapire, 1997). Il appartient à la famille des techniques de boosting, qui visent à améliorer la précision des prédictions en combinant plusieurs modèles faibles (des modèles légèrement meilleurs que le hasard) pour former un modèle fort. Contrairement aux méthodes d'ensemble comme les forêts aléatoires, AdaBoost pondère les exemples mal classés pour les rendre plus importants dans l'entraînement des modèles suivants. AdaBoost fonctionne en entraînant plusieurs classifieurs faibles (typiquement des arbres de décision de faible profondeur, aussi appelés souches d'arbres). Chaque classifieur est entraîné séquentiellement, et les erreurs des modèles précédents sont utilisées pour ajuster les pondérations des données d'entraînement, de manière à ce que les classifieurs suivants se concentrent davantage sur les exemples difficiles.

Au début, AdaBoost assigne des poids égaux à tous les exemples de données. Si l'ensemble de données contient (m) exemples, chaque exemple est initialement pondéré par $(w_i = \frac{1}{m})$, où (w_i) est le poids du $(i) - \text{ème}$ exemple. AdaBoost entraîne ensuite un classifieur faible $(h_t(x))$ sur les données pondérées. Ce classifieur peut être n'importe quel modèle de base.

L'erreur pondérée (ϵ_t) du classifieur est calculée en prenant en compte les poids des exemples mal classés :

$$\epsilon_t = \sum_{i=1}^{m} w_i \cdot I\big(y_i \neq h_t(x_i)\big)$$

où (I) est une fonction indicatrice qui vaut 1 si l'exemple est mal classé et 0 sinon, et (y_i) est la véritable étiquette de l'exemple (x_i).

Le poids du classifieur $(h_t(x))$ dans le modèle final est déterminé par son erreur pondérée. Si le classifieur est plus précis, il reçoit un poids plus important :

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$$

Les pondérations des exemples de données sont ensuite mises à jour pour le classifieur suivant, de manière à augmenter les poids des exemples mal classés :

$$w_i \leftarrow w_i \cdot \exp\left(\alpha_t \cdot I\big(y_i \neq h_t(x_i)\big)\right)$$

Ces poids sont ensuite normalisés pour s'assurer qu'ils forment une distribution de probabilité. Enfin, le modèle final est une combinaison pondérée de tous les classifieurs faibles :

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\right)$$

où $(H(x))$est la prédiction finale du modèle, obtenue en sommant les prédictions pondérées des classifieurs faibles.

Les Avantages d'AdaBoost

- Efficacité avec des modèles simples: AdaBoost peut transformer des modèles faibles, comme des arbres de décision de faible profondeur, en un modèle fort, ce qui lui permet de bien performer même avec des classifieurs de base très simples (Freund & Schapire, 1997).

- Flexibilité: AdaBoost est un algorithme très flexible qui peut être utilisé pour différents types de données et de modèles de base, y compris les classifieurs non paramétriques comme les arbres de décision (Schapire & Singer, 1999).
- Robustesse contre le surapprentissage: contrairement à d'autres modèles d'ensemble comme les forêts aléatoires, AdaBoost ne souffre généralement pas de surapprentissage tant que le nombre de classifieurs faibles reste raisonnable. Cela est dû à son mécanisme de pondération adaptative qui se concentre sur les erreurs difficiles à corriger (Dietterich, 2000).
- Réduction des biais: AdaBoost réduit efficacement le biais des classifieurs faibles en se concentrant sur les erreurs des modèles précédents. Cela conduit à une amélioration progressive de la précision du modèle final (Dietterich, 2000).

Les Limites d'AdaBoost

- Sensibilité aux données aberrantes: AdaBoost peut être sensible aux outliers, car il essaie de corriger toutes les erreurs, même celles qui sont dues à des anomalies dans les données. Cela peut entraîner un surajustement si les outliers sont nombreux (Mease et al., 2007).
- Complexité computationnelle: bien qu'AdaBoost soit généralement rapide à entraîner avec des classifieurs simples, son processus itératif et adaptatif peut devenir coûteux en temps de calcul lorsque les ensembles de données sont volumineux ou lorsque des classifieurs plus complexes sont utilisés (Y. Sun et al., 2007).
- Dépendance aux données d'entraînement: AdaBoost peut être très dépendant de la qualité des données d'entraînement. Si ces dernières sont bruitées ou biaisées, le modèle final en souffrira également(Schapire & Singer, 1999).
- Problèmes de performance avec des données non équilibrées: AdaBoost peut avoir des difficultés avec des ensembles de données non équilibrés, car il tente de corriger les erreurs sans distinction, ce qui peut conduire à une sur-représentation de la classe dominante (Y. Sun et al., 2007).

3.2.3 Balanced Bagging

Le Balanced Bagging est une méthode d'ensemble développée pour améliorer la performance des modèles de classification sur des ensembles de données déséquilibrés, où une classe est largement dominante par rapport aux autres. Il s'inspire de la technique de bagging (Bootstrap Aggregating), mais en introduisant une pondération ou un échantillonnage équilibré pour traiter les déséquilibres des classes (Breiman, 1996) (voir figure III.6). Cette méthode est particulièrement utile pour des tâches où les classes minoritaires ont une importance critique, comme la détection de fraudes ou le diagnostic médical.

Comme le bagging traditionnel, le Balanced Bagging fonctionne en générant plusieurs échantillons bootstrap (échantillons aléatoires avec remise) à partir des données d'entraînement, puis en entraînant un classifieur sur chaque échantillon. Toutefois, dans le Balanced Bagging, les échantillons sont construits de manière à équilibrer les classes. Cela peut être fait en sous-échantillonnant la classe majoritaire ou en sur-échantillonnant la classe minoritaire, voire en combinant les deux approches (X. -Y. Liu et al., 2009).

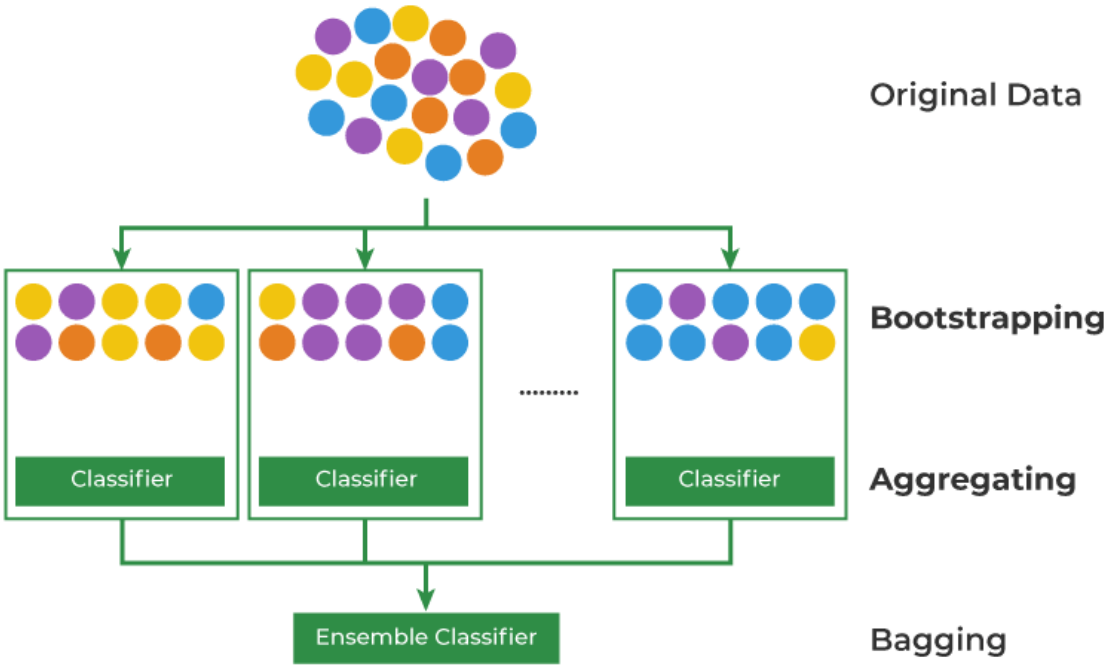

Figure III.6 Algorithme Balanced Bagging (ML | Bagging Classifier, 2019)

1) Échantillonnage équilibré

Pour chaque itération du bagging, un échantillon équilibré est créé à partir de l'ensemble de données d'entraînement. Cela implique généralement de réduire le nombre d'exemples de la classe majoritaire ou d'augmenter le nombre d'exemples de la classe minoritaire, de sorte que les deux classes soient représentées de manière égale dans l'échantillon. Soit (D) l'ensemble de données d'entraînement, contenant (n_1) exemples de la classe majoritaire et (n_2) exemples de la classe minoritaire, avec $(n_1 \gg n_2)$. Le Balanced Bagging crée un échantillon (D_t) pour chaque itération (t) tel que :

$$\left|D_t^{\text{majoritaire}}\right| = \left|D_t^{\text{minoritaire}}\right| = \min(n_1, n_2)$$

où $(D_t^{\text{majoritaire}})et(D_t^{\text{minoritaire}})$ représentent respectivement les sous-ensembles des classes majoritaire et minoritaire.

2) Entraînement des classifieurs

Une fois que l'échantillon équilibré est généré, un classifieur faible est entraîné sur cet échantillon. Typiquement, des arbres de décision sont utilisés comme classifieurs faibles en raison de leur capacité à gérer à la fois les variables numériques et catégoriques.

3) Combinaison des classifieurs

Après l'entraînement de plusieurs classifieurs faibles sur différents échantillons équilibrés, leurs prédictions sont combinées pour former la prédiction finale du modèle. Comme pour le bagging traditionnel, cette combinaison est réalisée par vote majoritaire dans le cas de la classification ou par moyenne dans le cas de la régression. Soit $(H_t(x))$ la prédiction du (t) −ème classifieur pour une instance (x), la prédiction finale du modèle est donnée par :

$$H(x) = \text{mode}\left(H_1(x), H_2(x), \dots, H_T(x)\right)$$

où (T) est le nombre total de classifieurs faibles.

Les Avantages du Balanced Bagging

- Gestion des déséquilibres de classes: le principal avantage du Balanced Bagging réside dans sa capacité à gérer les déséquilibres de classes, ce qui améliore la performance globale du modèle sur les classes minoritaires. Cela est particulièrement utile dans des contextes où les classes minoritaires sont d'un intérêt majeur (X. -Y. Liu et al., 2009).
- Réduction du biais: en équilibrant les classes, le Balanced Bagging réduit le biais introduit par une classe majoritaire dominante. Cela permet de produire des modèles mieux adaptés aux données déséquilibrées (C. Chen & Breiman, 2004).
- Simplicité et flexibilité: le Balanced Bagging est relativement simple à implémenter et peut être utilisé avec une variété de classifieurs faibles, ce qui le rend flexible pour différents types de problèmes de classification (X. -Y. Liu et al., 2009).
- Amélioration de la généralisation: en utilisant plusieurs échantillons équilibrés, le Balanced Bagging améliore la généralisation du modèle final, réduisant ainsi le risque de surajustement (overfitting) aux données d'entraînement (Biau et al., 2008).

Les Limites du Balanced Bagging

- Complexité computationnelle accrue: le Balanced Bagging nécessite la génération de plusieurs échantillons équilibrés et l'entraînement de plusieurs classifieurs. Cela peut être coûteux en termes de temps de calcul, surtout pour les grands ensembles de données (M. Galar et al., 2012).
- Dépendance à l'échantillonnage: la performance du Balanced Bagging peut être fortement influencée par la méthode d'échantillonnage utilisée. Un mauvais choix d'échantillonnage peut nuire à l'efficacité du modèle (Batista et al., 2004).
- Perte d'information potentielle: dans le cas du sous-échantillonnage de la classe majoritaire, il est possible de perdre des informations importantes, ce qui peut affecter la performance du modèle. De plus, le sur-échantillonnage de la classe minoritaire peut introduire de la redondance dans les données, rendant le modèle plus susceptible au surajustement (H. He & E. A. Garcia, 2009).
- Sensibilité aux données aberrantes: le Balanced Bagging peut être sensible aux données aberrantes, qui peuvent affecter la performance des classifieurs faibles si elles sont incluses dans les échantillons équilibrés (Y. Sun et al., 2007).

3.2.4 Gradient Boosting Machines (GBM)

Les Gradient Boosting Machines (GBM) sont des algorithmes d'ensemble qui utilisent le boosting pour créer un modèle fort à partir de plusieurs modèles faibles. Contrairement à des méthodes comme le bagging, qui combinent des prédictions en parallèle, le boosting construit les modèles séquentiellement, en corrigeant les erreurs des modèles précédents. Le GBM utilise une approche de descente de gradient pour minimiser une fonction de coût (J. H. Friedman, 2002; Jerome H. Friedman, 2001).

Le GBM fonctionne en entraînant un ensemble de classifieurs faibles, souvent des arbres de décision de faible profondeur, et en les combinant pour réduire progressivement l'erreur de prédiction. À chaque étape, un nouvel arbre est ajouté pour corriger les résidus (différence entre les prédictions et les valeurs réelles) des modèles précédents. Le processus commence par l'initialisation du modèle avec une constante, généralement la moyenne des valeurs cibles pour une tâche de régression, ou la classe la plus fréquente pour une tâche de classification.

$$F_0(x) = \arg\min_c \sum_{i=1}^{m} L(y_i, c)$$

où $(F_0(x))$ est le modèle initial, (L) est la fonction de perte et (y_i) sont les valeurs cibles.

Ensuite, le GBM minimise une fonction de coût en utilisant la méthode de la descente de gradient. À chaque itération (t), un nouvel arbre est entraîné pour approximer les résidus de la fonction de coût de l'étape précédente. La fonction de coût pour l'étape (t) est donnée par:

$$\text{résidus}_i = -\frac{\partial L\left(y_i, F_{t-1}(x_i)\right)}{\partial F_{t-1}(x_i)}$$

Cet arbre est ensuite ajouté au modèle précédent avec un facteur d'ajustement, appelé taux d'apprentissage ((η)):

$$F_t(x) = F_{t-1}(x) + \eta \cdot h_t(x)$$

où $(h_t(x))$est l'arbre de décision entraîné sur les résidus et (η) est un hyperparamètre qui contrôle la contribution de chaque arbre.

Ce processus est répété pour un certain nombre d'itérations (T), chaque nouvel arbre étant ajouté pour corriger les erreurs des arbres précédents. Le modèle final est la somme des prédictions de tous les arbres:

$$F_T(x) = F_0(x) + \eta \sum_{t=1}^{T} h_t(x)$$

Les Avantages des GBM

- Précision élevée: le GBM est capable de produire des modèles avec une haute précision en minimisant les erreurs de prédiction de manière séquentielle. Il est souvent utilisé pour résoudre des problèmes complexes (Jerome H. Friedman, 2001).
- Flexibilité: le GBM peut être utilisé pour diverses tâches, y compris la régression, la classification binaire et multi-classes, ainsi que la prédiction de probabilités. Il supporte également différentes fonctions de perte, ce qui le rend adaptable à de nombreux contextes (Natekin & Knoll, 2013).
- Réduction des biais et des variances: en combinant plusieurs modèles faibles et en ajustant les erreurs des prédictions précédentes, le GBM parvient à réduire à la fois le biais et la variance du modèle final, conduisant à une meilleure généralisation (Hastie et al., 2009).
- Gestion des données déséquilibrées: le GBM peut être adapté pour traiter les ensembles de données déséquilibrés en modifiant les fonctions de perte ou en ajustant les pondérations des classes (SUN et al., 2009).

Les Limites des GBM

- Complexité computationnelle: le GBM est plus coûteux en termes de temps de calcul que les méthodes d'ensemble comme le bagging ou les forêts aléatoires. Chaque itération nécessite l'entraînement d'un nouvel arbre, ce qui peut devenir prohibitif pour les grands ensembles de données (T. Chen & Guestrin, 2016).
- Sensibilité aux hyperparamètres: le GBM dépend fortement de l'ajustement des hyperparamètres, tels que le taux d'apprentissage ((η)), le nombre d'arbres ((T)), et la profondeur des arbres. Un mauvais choix d'hyperparamètres peut nuire à la performance du modèle (Prokhorenkova et al., 2018).
- Surapprentissage: le GBM peut facilement surapprendre les données d'entraînement, surtout si le nombre d'arbres est trop élevé ou si le taux d'apprentissage est trop faible. Des techniques comme la validation croisée et la régularisation sont nécessaires pour contrôler ce risque (Ribeiro et al., 2016).

- Difficulté d'interprétation: comme pour la plupart des modèles d'ensemble, le modèle final produit par le GBM est difficile à interpréter, surtout lorsqu'un grand nombre d'arbres est impliqué limitant son utilisation dans des domaines où l'interprétabilité est nécessaire (Ribeiro et al., 2016).

Le GBM a plusieurs variantes et améliorations, telles que XGBoost, LightGBM, et CatBoost, qui optimisent l'entraînement des arbres et réduisent la complexité computationnelle tout en améliorant la performance. Ces variantes introduisent des techniques comme le shrinkage (paramètre de rétrécissement), la régularisation (L1 et L2), et l'optimisation des arbres pour rendre le GBM plus efficace.

Comme mentionné plus haut, Les algorithmes d'apprentissage supervisé sont divers. Ils sont largement utilisés dans de nombreux domaines. Cependant, l'apprentissage supervisé a également des limites. Il nécessite des données étiquetées pour l'entraînement, ce qui peut être coûteux et fastidieux. De plus, les modèles d'apprentissage supervisé peuvent être surajustés aux données d'entraînement, ce qui peut entraîner une mauvaise performance sur de nouvelles données.

3.3 Les Algorithmes d'Apprentissage non supervisé

Contrairement aux algorithmes d'apprentissage supervisé, l'apprentissage non supervisé ne nécessite pas de données étiquetées. Le système d'apprentissage automatique analyse l'ensemble des données par lui-même et en extrait des modèles et des relations significatifs (Bishop, 2006). Ce type d'apprentissage est principalement utilisé pour les tâches de regroupement et d'association. Les principales caractéristiques des algorithmes non supervisés sont leur capacité à effectuer des regroupements non supervisés et à extraire des règles d'association (Rui Xu & D. Wunsch, 2005). Dans les tâches de mise en grappes, le modèle regroupe les points similaires, tandis que dans l'extraction de règles d'association, le modèle identifie les éléments cooccurrents dans l'ensemble de données.

4 Métriques d'Evaluation des Algorithmes d'Apprentissage Automatique

La réussite des algorithmes d'apprentissage dépend en grande partie de l'efficacité des mesures d'évaluation qui permettent de déterminer leurs performances. Ces mesures évaluent la qualité des prédictions faites par le modèle en les comparant aux résultats réels. Le choix des métriques d'évaluation varie en fonction du type d'algorithmes et du problème qu'ils visent à résoudre. Nous allons examiner de manière détaillée les différentes mesures utilisées pour évaluer les algorithmes d'apprentissage automatique, notamment l'exactitude, la précision, le rappel, le F1-score, et la courbe ROC, ainsi que leur pertinence dans l'évaluation des performances. Selon (Boser et al., 1992) :

Exactitude (Accuracy): il s'agit de la mesure la plus couramment utilisée pour évaluer les algorithmes d'apprentissage automatique. Elle mesure le rapport entre les observations correctement prédites et le nombre total d'observations dans un ensemble de données. En termes plus simples, il s'agit du pourcentage de prédictions correctes faites par le modèle. Par exemple, si un modèle prédit correctement 90 cas sur 100, l'exactitude est de 90 %. Bien que l'exactitude soit une mesure fiable pour les problèmes de classification binaire, elle présente des limites lorsqu'elle est appliquée à des ensembles de données déséquilibrés, où une classe est fortement représentée par rapport à l'autre. Dans ce cas, un modèle qui prédit toujours la classe majoritaire aura une exactitude élevée, mais ne sera pas utile dans un contexte pratique. Elle se calcule en

divisant le nombre d'observations correctement prédites par le nombre total d'observations, multiplié par 100.

$$\text{Exactitude} = \frac{\text{Nombre d'observations prédites correcteemnt}}{\text{Nombre total d'observations}} * 100\%$$

La précision (precision): cette mesure évalue la proportion de prédictions vraies positives par rapport au nombre total de prédictions positives. En d'autres termes, elle détermine la fréquence à laquelle le modèle fait des prédictions positives correctes. La précision est essentielle lorsque le coût des faux positifs est élevé. Par exemple, dans le cadre de la détection de tumeurs, il est préférable d'avoir un modèle très précis, car un diagnostic erroné de cancer peut avoir de graves conséquences. En revanche, lorsque le coût des faux négatifs est élevé, le rappel devient la mesure la plus appropriée.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Le rappel (Recall): également connu sous le nom de sensibilité, le rappel mesure la proportion de prédictions réellement positives par rapport au nombre total de cas positifs. Il évalue la capacité du modèle à identifier les cas positifs dans l'ensemble de données. Les faux négatifs sont les cas où le modèle prédit un résultat négatif pour un cas positif. Par conséquent, le rappel est plus approprié lorsque le coût des faux négatifs est élevé. Par exemple, dans la détection des fraudes, il est crucial d'identifier le plus grand nombre possible de cas frauduleux, car en manquer peut entraîner des pertes financières considérables.

$$\text{Rappel} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Le F1-score : Il s'agit d'une mesure qui combine la précision et le rappel. C'est la moyenne harmonique de la précision et du rappel, utilisée lorsque ces deux mesures sont d'importance égale. Le F1-score permet d'équilibrer la précision et le rappel et fournit une mesure globale de la performance d'un modèle. Il est particulièrement utile lorsqu'on travaille avec des ensembles de données déséquilibrés, car il prend en compte à la fois les faux positifs et les faux négatifs.

$$\text{F1} - \text{score} = 2 * \frac{\text{Précision} * \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

La courbe ROC: (Receiver Operating Characteristic) et son équivalent numérique, l'Aire sous la courbe ROC (AUC) sont des outils pour évaluer les modèles de classification. La courbe ROC illustre graphiquement le compromis entre le taux de vrais positifs et le taux de faux positifs à différents seuils de classification (Bradley, 1997). Un modèle idéal aurait une courbe ROC se rapprochant du coin supérieur gauche, indiquant une capacité maximale à discriminer entre les classes. L'AUC quantifie cette performance en mesurant la surface sous la courbe ROC. Plus précisément, une AUC égale à 1 signifie une capacité parfaite de discrimination, tandis qu'une AUC de 0,5 suggère une performance équivalente à une classification aléatoire. L'AUC-ROC est particulièrement utile dans les cas de déséquilibre de classe, offrant une mesure indépendante du seuil de classification pour la sélection et l'optimisation des modèles de classification.

Perte de Hamming (Hamming Loss): c'est une mesure d'évaluation utilisée principalement dans les problèmes de classification multi-label ou multi-classe, où plusieurs étiquettes peuvent être attribuées à une seule instance. Cette métrique quantifie la fraction des étiquettes incorrectement prédites, c'est-à-dire celles qui sont soit faussement positives (prédites comme étant présentes alors qu'elles ne le sont pas), soit faussement négatives (non prédites alors qu'elles devraient l'être). Contrairement aux mesures traditionnelles telles que l'exactitude ou la

précision, qui se concentrent sur des prédictions correctes ou des erreurs globales, la perte de Hamming tient compte des erreurs sur chaque étiquette individuellement, ce qui en fait une mesure particulièrement adaptée aux contextes où chaque classe a un poids égal.

La perte de Hamming est définie comme suit : pour un ensemble de données avec (n) échantillons et (L) étiquettes possibles, la perte de Hamming est calculée comme la moyenne du nombre d'étiquettes incorrectement prédites par échantillon divisé par le nombre total d'étiquettes. Le Hamming Loss prend des valeurs entre 0 et 1, où 0 indique une prédiction parfaite (toutes les étiquettes sont correctement prédites pour tous les échantillons), et 1 représente un cas où toutes les étiquettes sont incorrectement prédites.

$$[\text{Hamming Loss} = \frac{1}{n \times L}\sum_{i=1}^{n}\sum_{j=1}^{L} 1(y_{ij} \neq \widehat{y_{ij}})]$$

Où :

- $(y_{ij\backslash})$ est la véritable valeur de l'étiquette (j) pour l'échantillon (i) ,
- $(\widehat{y_{ij}})$est la valeur prédite pour l'étiquette (j) pour l'échantillon (i),
- $(1(\cdot))$ est la fonction indicatrice qui vaut 1 si l'argument est vrai, et 0 sinon.

Outre les mesures susmentionnées, plusieurs autres mesures sont utilisées pour évaluer les algorithmes d'apprentissage automatique, telles que l'erreur absolue moyenne (MAE), l'erreur quadratique moyenne (MSE) et l'erreur quadratique moyenne racine (RMSE). La MAE, la MSE et la RMSE sont des mesures de régression qui évaluent la différence entre les valeurs réelles et prédites d'une variable continue. Elles sont utiles pour déterminer la précision du modèle dans la prédiction des valeurs numériques. Le choix des bonnes mesures pour l'évaluation dépend du problème traité, du type d'ensemble de données et des objectifs spécifiques de l'étude.

5 Classification à l'Aide de l'Apprentissage Automatique

La classification est un processus d'organisation ou de catégorisation de points de données en groupes similaires sur la base de leurs propriétés ou caractéristiques. Il s'agit d'un problème fondamental dans divers domaines tels que l'exploration de données, la reconnaissance des formes et l'apprentissage automatique.

Dans le contexte de l'apprentissage automatique, la classification fait référence au processus de formation d'un modèle capable de prédire la classe ou l'étiquette d'un nouveau point de données inédit sur la base d'un ensemble donné de caractéristiques ou d'attributs. Les modèles d'apprentissage automatique peuvent effectuer des tâches de classification de plusieurs manières, telles que la classification binaire et la classification multi-classe (Honeine et al., 2013; Riston et al., 2023).

5.1 Classification Binaire

La classification binaire est l'une des pierres angulaires de l'apprentissage supervisé en intelligence artificielle. La classification binaire repose sur la notion de décision binaire, c'est-à-dire la capacité de séparer les données en deux catégories distinctes, souvent appelées classe positive et classe négative (Riston et al., 2023).

Formellement, un problème de classification binaire peut être défini comme la recherche d'une fonction $(f: X \rightarrow \{0, 1\})$, où (X) représente l'espace des caractéristiques (features) des données, et 0 et 1 représentent les deux classes. Le but est de trouver une fonction (f) qui

minimise l'erreur de classification, c'est-à-dire le nombre de cas où la prédiction (f(x)) diffère de la vraie classe de (x).

La nature binaire de ce problème conduit à l'utilisation de diverses mesures d'évaluation, telles que l'exactitude (accuracy), la précision (precision), le rappel (recall), et le score F1, chacune offrant un aperçu différent de la performance du modèle.

La classification binaire, malgré sa simplicité apparente, pose plusieurs défis, notamment en ce qui concerne la gestion des classes déséquilibrées. Lorsque l'une des classes est beaucoup moins représentée que l'autre, les modèles peuvent être biaisés en faveur de la classe majoritaire (SUN et al., 2009). Des techniques telles que le sur-échantillonnage de la classe minoritaire, le sous-échantillonnage de la classe majoritaire, sont souvent nécessaires pour pallier ce problème (Chawla et al., 2002; Fernández et al., 2018; H. He & E. A. Garcia, 2009). Un autre défi réside dans le choix du bon modèle. Chaque algorithme a ses forces et ses faiblesses, et la performance peut varier en fonction des caractéristiques des données. Par exemple, les SVM sont robustes face aux données de haute dimension, mais peuvent être inefficaces sur de grands volumes de données (Chauhan et al., 2019). Les réseaux de neurones, en revanche, peuvent gérer des volumes massifs de données, mais au prix d'une complexité accrue et d'un besoin important de ressources computationnelles (Srivastava et al., 2014). Par ailleurs, les approches ensemblistes, telles que les forêts aléatoires et les techniques de boosting (e.g., AdaBoost, Gradient Boosting Machines), constituent une autre avancée majeure dans la classification binaire. En combinant plusieurs modèles pour réduire le biais et la variance, ces méthodes ont permis des améliorations significatives dans la précision des prédictions (Y. Sun et al., 2007).

5.2 Classification Multi-classe

La classification multi-classe, aussi appelée classification multinomiale vise à catégoriser des données en plus de deux classes distinctes. Elle consiste à assigner une étiquette $(y \in \{1, 2, \ldots, K\})$ à une instance (x) de l'espace des caractéristiques (X), où (K) représente le nombre de classes. Cette généralisation du problème binaire introduit des défis supplémentaires, notamment en ce qui concerne la définition des frontières de décision entre plusieurs classes. Dans un espace de caractéristiques de haute dimension, ces frontières peuvent devenir extrêmement complexes et non linéaires. Ce qui nécessite des méthodes d'apprentissage sophistiquées pour être correctement modélisées (Bishop, 2006).

Une des premières étapes dans la classification multi-classe est le choix du cadre d'apprentissage. Les approches traditionnelles incluent le One-vs-Rest (OvR), où un classificeur binaire est entraîné pour chaque classe en la distinguant de toutes les autres, et le One-vs-One (OvO), où un classifieur est entraîné pour chaque paire de classes. Ces méthodes permettent de transformer un problème multi-classe en plusieurs sous-problèmes binaires, facilitant ainsi l'application d'algorithmes de classification binaire classiques (Honeine et al., 2013; Silva-Palacios et al., 2017).

L'un des principaux défis de la classification multi-classe réside dans la complexité des frontières de décision. Contrairement à la classification binaire, où il n'y a qu'une seule frontière à modéliser, la classification multi-classe implique plusieurs frontières entre différentes paires de classes. Cela peut rendre l'optimisation du modèle plus difficile, en particulier lorsque les classes sont linéairement non séparables. De plus, comme dans la classification binaire, certaines classes peuvent être sur-représentées ou sous-représentées, ce qui peut biaiser le modèle en faveur des classes majoritaires. Des techniques telles que la pondération des classes, le sur-échantillonnage, et le sous-échantillonnage sont souvent nécessaires pour traiter ce problème (H. He & E. A. Garcia, 2009). Enfin, les coûts computationnels représentent

également un défi significatif, notamment avec des algorithmes tels que les réseaux de neurones profonds et les méthodes d'ensemble. L'entraînement de ces modèles sur des ensembles de données volumineux et à haute dimension peut nécessiter des ressources informatiques importantes, ce qui limite leur accessibilité pour certaines applications (Whalen et al., 2013).

5.3 Classification Multi-étiquettes

La classification multi-étiquettes (multi-label), une sous-catégorie de l'apprentissage supervisé, représente un problème où chaque instance peut être associée à plusieurs étiquettes simultanément (Doquire & Verleysen, 2013). Contrairement à la classification binaire et à la classification multi-classe, qui attribuent une seule étiquette par instance, la classification multi-label reflète la complexité des données du monde réel où des objets ou événements peuvent appartenir à plusieurs catégories à la fois (figure III.7). Ce paradigme est largement utilisé dans des domaines variés tels que l'extraction d'informations, la bio-informatique et la recommandation de contenu.

Dans la classification multi-labels, chaque instance $(x \in X)$ est associée à un ensemble d'étiquettes ($Y \subseteq \{y_1, y_2, \dots, y_k\}$), où (k) représente le nombre total d'étiquettes possibles. Cela signifie qu'au lieu de prédire une seule étiquette pour chaque instance, le modèle doit prédire un sous-ensemble d'étiquettes pertinent pour chaque instance donnée (Alzanin et al., 2023). L'objectif de la classification multi-labls est de minimiser l'erreur de prédiction sur l'ensemble des étiquettes, ce qui nécessite souvent des compromis entre précision, rappel et d'autres mesures d'évaluation spécifiques à cette tâche. Les métriques d'évaluation les plus couramment utilisées incluent le Hamming Loss, qui mesure la fraction des étiquettes mal prédites, et le F1-micro/macro, qui évalue l'équilibre entre précision et rappel au niveau des étiquettes (X. Chen et al., 2022).

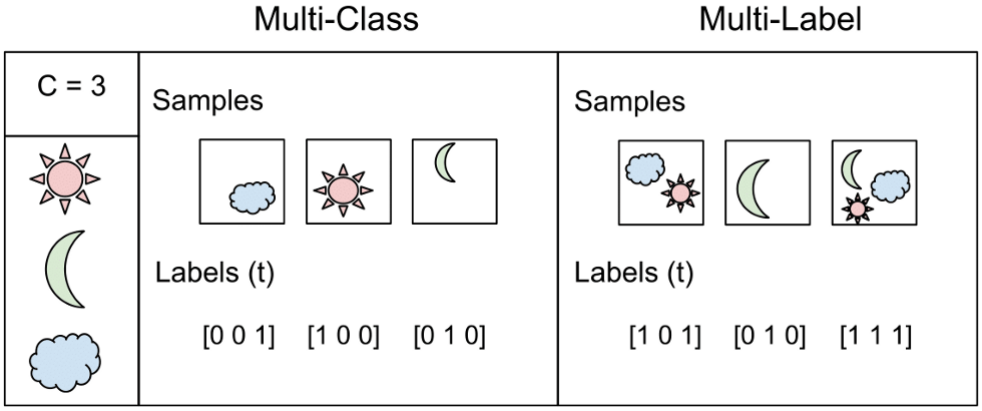

Figure III.7 Classification Multi-class vs Multi-label (Gautam Sharma, 2021)

La classification multi-labels peut être abordée de deux manières principales: les méthodes de transformation de problème et les méthodes d'adaptation d'algorithme.

1) Méthodes de transformation de problème: ces méthodes convertissent un problème multi-labels en plusieurs sous-problèmes plus simples, tels que des problèmes de classification binaire ou multi-classe. Les algorithmes utilisés sont :
   - Binary Relevance (BR): la méthode BR transforme le problème multi-labels en un ensemble de problèmes de classification binaire distincts, un pour chaque étiquette. Bien que simple et efficace, cette approche ignore les corrélations potentielles entre les étiquettes (Montañes et al., 2014).
   - Classifier Chains (CC): pour prendre en compte les corrélations entre les étiquettes, la méthode CC ordonne les étiquettes et utilise les prédictions des

étiquettes précédentes comme caractéristiques pour les suivantes. Cela améliore souvent les performances par rapport à BR, mais augmente la complexité (Read et al., 2011).

- Label Powerset (LP): LP traite chaque combinaison unique d'étiquettes comme une classe dans un problème de classification multi-classe. Bien que cette approche puisse capturer toutes les corrélations entre les étiquettes, elle souffre souvent de la rareté des données, surtout lorsque le nombre d'étiquettes est élevé (Tsoumakas et al., 2011).

2) Méthodes d'adaptation d'algorithme: ces méthodes adaptent directement des algorithmes existants pour traiter les problèmes multi-labels sans les transformer.
   - Réseaux de neurones multi-labels: les réseaux de neurones peuvent être adaptés pour la classification multi-labels en utilisant des sorties multiples avec des fonctions de perte adaptées telles que la sigmoïde et la binary cross-entropy. Ces réseaux permettent de capturer des corrélations complexes entre les étiquettes, mais nécessitent une grande quantité de données pour un entraînement efficace (Sapozhnikova, 2009).
   - Random k-Labelsets (RAkEL): cette méthode est une extension de Label Powerset, où plusieurs ensembles aléatoires d'étiquettes sont utilisés pour créer des sous-problèmes de classification multi-classe, dont les prédictions sont ensuite combinées. Cette approche permet de réduire le problème de rareté des données rencontré avec LP (Tsoumakas et al., 2011).
   - Arbres de décision multi-labels: les arbres de décision peuvent être modifiés pour traiter plusieurs étiquettes simultanément, en effectuant des séparations basées sur les gains d'information pour l'ensemble des étiquettes. De même, la forêt aléatoire et d'autres méthodes d'ensemble peuvent également être adaptées pour la classification multi-label (Kocev et al., 2007).

Cependant, la classification multi-label présente plusieurs défis notamment :

- Corrélations entre étiquettes: les étiquettes peuvent être corrélées entre elles, ce qui signifie que la présence d'une étiquette peut influencer la probabilité d'une autre. Ignorer ces corrélations, comme c'est le cas pour le BR, peut conduire à des performances sous-optimales (Read et al., 2011). Les méthodes comme Classifier Chains et les réseaux de neurones multi-labels visent à capturer ces relations, mais au prix d'une complexité accrue.
- Complexité computationnelle: le nombre d'étiquettes possible dans un problème multi-labels peut être exponentiel, ce qui pose des défis en termes de stockage et de temps de calcul. Les méthodes qui traitent chaque combinaison d'étiquettes séparément, comme Label Powerset, souffrent particulièrement de ce problème (Tsoumakas et al., 2011).
- Déséquilibre des étiquettes: dans de nombreux scénarios, certaines étiquettes sont beaucoup plus fréquentes que d'autres, ce qui peut biaiser les modèles.
- Evaluation et interprétation: évaluer la performance des modèles multi-labels est plus complexe que pour les modèles classiques. Les métriques doivent tenir compte de la précision des prédictions sur plusieurs étiquettes simultanément, ce qui complique l'interprétation des résultats (X. Chen et al., 2022).

6 Les Ensembles de Données Déséquilibrés

Les ensembles de données déséquilibrés représentent un défi omniprésent dans divers domaines, y compris celui de l'éducation, où l'analyse des données joue un rôle pour éclairer la prise de décisions (Krawczyk, 2016). L'analyse des performances des apprenants à travers des indicateurs multiples peut offrir des informations précieuses, facilitant ainsi l'évaluation de l'efficacité des programmes éducatifs, l'identification des apprenants à risque et l'amélioration continue de l'environnement pédagogique. Cependant, l'un des obstacles majeurs rencontré dans l'analyse des données éducatives est la prévalence des ensembles de données déséquilibrés (Radwan & Cataltepe, 2017). Ce déséquilibre se caractérise par une représentation inégale des catégories au sein des données, ce qui peut introduire des biais dans les analyses (figure III.8). Par exemple, dans le cadre de l'évaluation des performances académiques, il n'est pas rare que le nombre d'étudiants réussissant un examen soit nettement supérieur à celui des étudiants échouant, entraînant ainsi une distribution asymétrique des classes.

Cette disparité a des répercussions significatives sur l'efficacité des algorithmes d'apprentissage automatique. En effet, lorsque les modèles sont exposés à des données où une classe domine, ils tendent à privilégier cette classe majoritaire (Barros et al., 2019). Cela peut compromettre la capacité des modèles à identifier avec précision les cas minoritaires, tels que les apprenants à risque, et à prédire de manière fiable les résultats scolaires.

Des approches méthodologiques ont été développées pour atténuer les effets du déséquilibre dans les ensembles de données. Parmi ces stratégies, on retrouve la rééchantillonnage, qui peut inclure le sous-échantillonnage de la classe majoritaire ou le suréchantillonnage de la classe minoritaire (Chawla et al., 2002) (voir figure III.9). Ces techniques permettent de rééquilibrer les classes pour améliorer les performances des modèles prédictifs (Lemaitre et al., 2016).

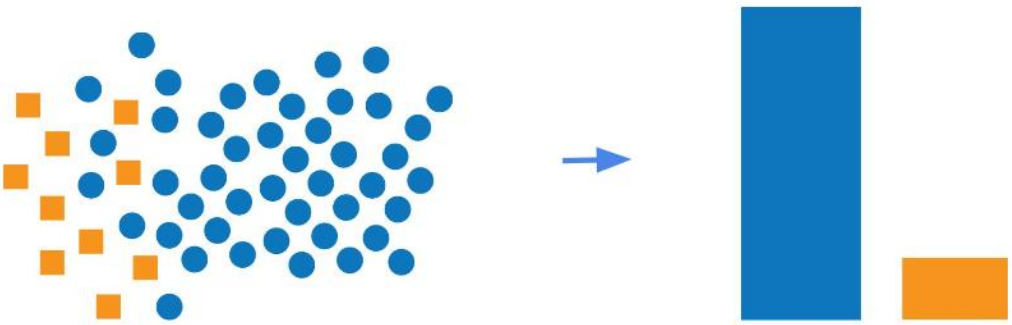

Figure III.8 Démonstration du déséquilibre des classes

Le sous-échantillonnage consiste à réduire la taille de la classe majoritaire en supprimant un certain nombre d'exemples, ce qui permet de rééquilibrer les classes au sein des données (Z. Sun et al., 2024). Cette approche est conçue pour atténuer le biais qui peut survenir lorsqu'une classe est surreprésentée, permettant ainsi aux modèles d'apprentissage automatique de traiter équitablement les différentes catégories. On trouve différentes approches notamment :

- ✓ **Sous-échantillonnage aléatoire** : le sous-échantillonnage aléatoire est l'une des techniques les plus simples. Il consiste à éliminer de manière aléatoire des exemples de la classe majoritaire pour équilibrer les classes. Bien que cette méthode soit facile à mettre en œuvre, elle peut entraîner une perte d'informations précieuses si des échantillons représentatifs de la classe majoritaire sont supprimés. Cela peut affaiblir la performance globale du modèle en réduisant sa capacité à généraliser (Tarekegn et al., 2021).
- ✓ **Approches sensibles aux coûts** : les approches sensibles aux coûts ajustent le processus de sous-échantillonnage en fonction du coût associé à l'erreur de classification. Ces méthodes attribuent des coûts différents aux erreurs commises sur les classes minoritaires et majoritaires, en cherchant à minimiser les erreurs sur les classes sous-représentées (Tarekegn et al., 2021). Cette approche permet de prendre en compte l'importance relative des différentes classes, tout en réduisant le déséquilibre global des données

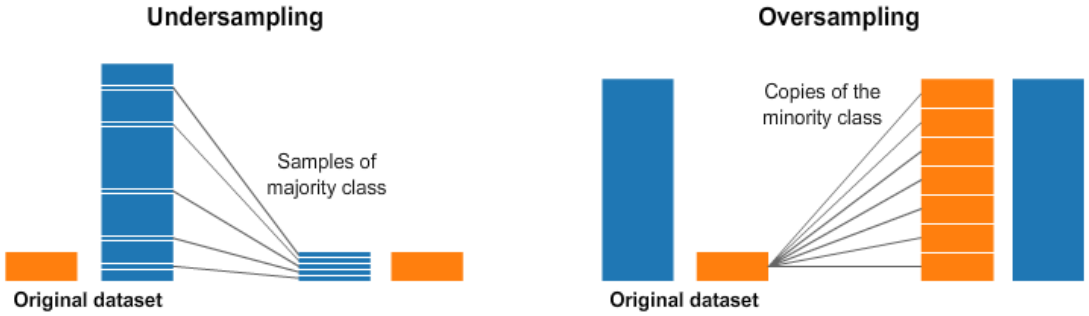

Figure III.9 Différence entre le sous-échantillonnage et le suréchantillonnage

Bien que le sous-échantillonnage puisse atténuer le déséquilibre des classes, il présente des inconvénients notables. Le principal problème réside dans la perte d'informations, car la suppression d'exemples, en particulier dans le sous-échantillonnage aléatoire, peut entraîner la perte de données pertinentes pour la classe majoritaire. Par conséquent, le choix de la technique de sous-échantillonnage doit être soigneusement évalué en fonction de la nature spécifique des données et du problème à résoudre (Tarekegn et al., 2021).

Le suréchantillonnage de la classe minoritaire vise à rétablir un équilibre entre les classes en dupliquant ou en générant artificiellement des exemples de la classe sous-représentée. De cette manière, les algorithmes d'apprentissage automatique peuvent mieux saisir les caractéristiques des classes minoritaires, réduisant ainsi le biais potentiel introduit par des distributions asymétriques des données. L'une des méthodes les plus utilisées dans ce domaine est la technique de suréchantillonnage appelée Synthetic Minority Over-sampling Technique (SMOTE), introduite par (Chawla et al., 2002).

SMOTE crée de nouveaux exemples synthétiques en interpolant entre les échantillons existants de la classe minoritaire, plutôt qu'en les dupliquant simplement. Cette approche réduit les problèmes de surapprentissage qui peuvent survenir lorsque des échantillons minoritaires sont simplement répliqués plusieurs fois (Chawla et al., 2002). Des variantes plus récentes de SMOTE ont été développées pour améliorer l'efficacité de cette technique dans des contextes spécifiques. Par exemple, SMOTE-ENN combine SMOTE avec Edited Nearest Neighbors (ENN) pour éliminer les bruits dans les données suréchantillonnées (Batista et al., 2004). Une autre approche, Borderline-SMOTE, se concentre sur les exemples minoritaires proches des frontières de décision, où les risques de mauvaise classification sont plus élevés (Majzoub et

al., 2020). Ces techniques permettent de mieux équilibrer les classes tout en préservant l'intégrité des données.

6.1 Le Ratio de Déséquilibre

Le ratio de déséquilibre est une métrique utilisée dans le domaine de l'apprentissage automatique pour évaluer la répartition des classes dans un ensemble de données, particulièrement ceux qui impliquent des problèmes de classification binaire (Dong et al., 2024). Le ratio de déséquilibre mesure la proportion relative entre les classes minoritaires et majoritaires. Un déséquilibre marqué entre les classes peut conduire à une performance biaisée des modèles d'apprentissage automatique, car les modèles ont tendance à favoriser la classe majoritaire.

Soit un ensemble de données avec deux classes, A et B, où A est la classe majoritaire et B est la classe minoritaire. Le ratio de déséquilibre $((R_{imbalance}))$ est défini par la formule :

$$R_{imbalance} = \frac{n_A}{n_B}$$

où :

- (n_A) est le nombre d'exemples de la classe A.

- (n_B)est le nombre d'exemples de la classe B.

Dans cette recherche, nous nous concentrons sur l'évaluation approfondie des techniques de suréchantillonnage proposées par la communauté scientifique, en examinant leur efficacité spécifique dans l'analyse des données éducatives.

7 Les Techniques de Suréchantillonnage

Ces dernières années, les techniques de suréchantillonnage ont connu un essor significatif dans les ensembles de données éducatifs, permettant la collecte et l'analyse d'ensembles de données importants et diversifiés. Leur utilisation s'est avérée prometteuse en raison de leur potentiel d'amélioration de la précision des prédictions pour les classes sous-représentées. Il existe plusieurs techniques de suréchantillonnage, dont SMOTE (Chawla et al., 2002) est l'une des plus populaires. Dans ce qui suit, nous examinerons chacune des techniques de suréchantillonnage suivantes : SMOTE, SMOTE Borderline, SMOTE-ENN et ADASYN (Adaptative Synthetic Sampling) et comment elles ont été utilisées dans le domaine de l'éducation.

7.1 SMOTE : Une Approche Synthétique pour les Problèmes de Classification Déséquilibrée

SMOTE est une technique de suréchantillonnage synthétique qui consiste à créer de nouveaux exemples de la classe minoritaire en interpolant les exemples existants. Cette technique a été proposée pour la première fois par (Chawla et al., 2002).

7.1.1 Principe de Fonctionnement de SMOTE

Le fonctionnement de SMOTE repose sur un processus de génération d'exemples synthétiques. Plus précisément, pour chaque exemple appartenant à la classe minoritaire, SMOTE sélectionne k exemples similaires de la même classe, où k est un paramètre déterminé par l'utilisateur. Ces exemples similaires sont choisis en utilisant des méthodes de voisinage,

généralement basées sur la distance euclidienne dans l'espace des caractéristiques. Une fois les exemples similaires sélectionnés, SMOTE crée de nouveaux exemples en interpolant les caractéristiques de l'exemple d'origine avec celles des exemples voisins (Chawla et al., 2002; Fernández et al., 2018). Concrètement, le processus fonctionne comme suit :

1) SMOTE sélectionne aléatoirement l'un des k voisins les plus proches de l'exemple d'origine.
2) La différence entre les caractéristiques de l'exemple d'origine et celles du voisin sélectionné est calculée.
3) Cette différence est multipliée par un facteur aléatoire compris entre 0 et 1, introduisant ainsi une variation contrôlée.
4) Le résultat de cette multiplication est ensuite ajouté aux caractéristiques de l'exemple d'origine, produisant ainsi un nouvel exemple synthétique.

Ce processus est répété pour générer le nombre d'exemples synthétiques souhaité, augmentant ainsi la représentation de la classe minoritaire tout en préservant la diversité des données.

Cette technique s'est avérée très pertinente dans de nombreux domaines. Toutefois, il est essentiel de disposer d'un modèle capable de distinguer avec précision les différentes classes, sur la base des données disponibles. En effet, dans de nombreux cas, les données disponibles sont biaisées, ce qui signifie qu'une classe peut avoir beaucoup plus de points de données que l'autre. Cela peut souvent conduire à la formation d'un modèle biaisé, qui n'est pas en mesure d'identifier correctement la classe minoritaire.

7.1.2 Les Avantages de la Technique SMOTE

La technique SMOTE présente plusieurs avantages notables, qui en font une méthode privilégiée pour aborder les défis posés par les classes minoritaires sous-représentées.

- ✓ Amélioration des performances du modèle : l'un des principaux avantages de SMOTE réside dans sa capacité à améliorer les performances des modèles de classification en garantissant que la classe minoritaire ne soit pas ignorée (S. Wang et al., 2021). Dans de nombreux contextes, la sous-représentation de la classe minoritaire conduit à un biais des modèles en faveur de la classe majoritaire, compromettant ainsi la qualité des prédictions. SMOTE atténue ce biais en générant de nouveaux exemples synthétiques pour la classe minoritaire, augmentant ainsi sa présence dans l'ensemble de données. Cela est particulièrement critique dans des domaines où les erreurs de classification de la classe minoritaire peuvent avoir des conséquences graves. Par exemple, dans le diagnostic médical, la mauvaise classification de patients atteints de maladies rares peut entraîner des retards dans le traitement ou des décisions cliniques erronées. Une étude de (Fernández et al., 2018) montre que l'application de SMOTE dans des contextes médicaux améliore la précision des modèles en matière de détection des maladies rares, en renforçant l'apprentissage des caractéristiques spécifiques à ces conditions.
- ✓ Simplicité d'implémentation : un autre avantage de SMOTE réside dans sa simplicité d'implémentation. La technique est relativement facile à intégrer dans un pipeline d'apprentissage automatique existant sans nécessiter de modifications. Cette simplicité découle de la nature du processus de suréchantillonnage, qui repose sur des opérations basiques d'interpolation entre les exemples existants de la classe minoritaire. De plus, la technique peut être appliquée à une variété de types de données, y compris les données numériques, catégorielles, et mixtes, sans nécessiter de prétraitement complexe (S. Feng et al., 2021).
- ✓ Applicabilité: SMOTE se distingue également par sa large applicabilité. En effet, cette technique peut être utilisée en combinaison avec un large éventail de classifieurs, tels que les forêts aléatoires, les machines à vecteurs de support (SVM), et les réseaux

neuronaux. Cette flexibilité rend la technique SMOTE pertinente pour une vaste gamme de problèmes, allant de la détection de fraudes à la classification d'images, en passant par le domaine médical.

- ✓ Réduction du biais en faveur de la classe majoritaire : le principal avantage de SMOTE réside dans sa capacité à réduire le biais en faveur de la classe majoritaire. Ce biais est souvent un problème récurrent dans les ensembles de données déséquilibrés, où les modèles ont tendance à sur-apprendre les caractéristiques de la classe majoritaire au détriment de la classe minoritaire. SMOTE atténue ce biais en augmentant la représentation de la classe minoritaire, ce qui permet aux modèles de mieux comprendre les motifs sous-jacents de cette classe (G. A. Pradipta et al., 2021). Dans des applications critiques, telles que la détection des fraudes ou le diagnostic médical, cette réduction du biais est particulièrement importante (Batista et al., 2004).

La technique SMOTE offre des avantages significatifs pour le traitement des ensembles de données déséquilibrés, en améliorant les performances des modèles et en réduisant le biais en faveur de la classe majoritaire. Dans des contextes critiques, elle se révèle être un outil indispensable pour assurer des prédictions précises et fiables.

7.1.3 Limites de la Techniques SMOTE

Bien que la technique SMOTE ait démontré son efficacité pour traiter les ensembles de données déséquilibrés, elle n'est pas exempte de limitations. Celles-ci peuvent affecter la performance globale des algorithmes d'apprentissage automatique, en particulier dans des contextes sensibles tel que l'éducation. Parmi les limites, on note :

- ✓ **Génération d'échantillons synthétiques irréalistes:** l'une des principales critiques de SMOTE est qu'elle peut générer des échantillons synthétiques qui ne reflètent pas fidèlement la véritable distribution de la classe minoritaire. En interpolant des exemples existants, SMOTE risque de produire des points de données artificiels qui ne capturent pas la complexité des motifs réels présents dans la classe minoritaire. Cette création d'échantillons irréalistes peut entraîner des erreurs de classification, en particulier lorsque les nouvelles données ne correspondent pas aux schémas synthétiques générés par l'algorithme. Ce problème est particulièrement pertinent dans les domaines où la précision des données est importante, comme dans le diagnostic médical. Par exemple, une mauvaise représentation des caractéristiques cliniques rares pourrait conduire à des diagnostics incorrects, compromettant ainsi la prise en charge des patients (Fernández et al., 2018).
- ✓ **Surapprentissage et réduction de la généralisation:** un autre inconvénient majeur de SMOTE réside dans sa propension à entraîner un surajustement des modèles. En générant des échantillons synthétiques, SMOTE augmente artificiellement la taille de la classe minoritaire, ce qui peut amener le modèle à devenir trop sensible aux données d'apprentissage (le surapprentissage se produit lorsque le modèle apprend les moindres détails des exemples synthétiques au point de ne plus pouvoir généraliser correctement sur des données inédites) (Chawla et al., 2002). Le surapprentissage est particulièrement problématique dans les scénarios où les modèles doivent être appliqués à des données variées et imprévisibles, comme dans les systèmes éducatifs. Par exemple, un modèle surajusté pourrait identifier à tort des élèves comme étant à risque en se basant sur des motifs artificiels plutôt que sur des indicateurs fiables.
- ✓ **Sensibilité aux données bruyantes ou mal étiquetées:** SMOTE est également sensible aux données bruyantes ou mal étiquetées. En effet, l'algorithme repose sur les données existantes pour générer de nouveaux exemples, ce qui signifie que si les données de départ contiennent des erreurs ou du bruit, ces imperfections seront amplifiées dans les

échantillons synthétiques (Batista et al., 2004). Cela peut entraîner une détérioration de la performance des modèles, notamment en augmentant le taux de faux positifs ou de faux négatifs. Dans des contextes où la qualité des données est inégale ou où les erreurs d'étiquetage sont courantes, comme c'est souvent le cas dans les bases de données éducatives, l'application de SMOTE peut aggraver les problèmes de classification au lieu de les résoudre.

- ✓ **Augmentation de la complexité computationnelle**: un autre inconvénient de SMOTE est l'augmentation de la complexité computationnelle. En augmentant la taille de l'ensemble de données en générant de nouveaux exemples, SMOTE accroît également les exigences en matière de calcul et de mémoire. Cette augmentation de la charge computationnelle peut être un obstacle important, surtout dans les situations où les ressources sont limitées ou lorsque des algorithmes complexes doivent être utilisés. L'allongement du temps d'exécution et la consommation accrue de ressources peuvent rendre la technique SMOTE moins attrayante pour les applications en temps réel ou pour les analyses à grande échelle nécessitant une efficacité optimale (Tarekegn et al., 2021).
- ✓ **Inefficacité dans les cas d'extrême déséquilibre des classes** : SMOTE peut s'avérer inefficace lorsque le déséquilibre des classes est trop important. Dans les cas où la classe minoritaire représente une fraction minuscule de l'ensemble de données, la simple génération de points de données synthétiques peut ne pas suffire à résoudre le problème. Au contraire, cela peut exacerber les difficultés de classification en introduisant davantage de bruit sans améliorer substantiellement la détection des cas rares. Cette inefficacité est particulièrement pertinente dans des domaines tels que la détection de fraudes ou les applications de cybersécurité, où les attaques ou les anomalies sont souvent extrêmement rares. Dans de tels cas, d'autres approches, telles que les méthodes d'échantillonnage adaptatif ou les techniques de coût-sensible, peuvent être plus appropriées (H. Han et al., 2005).

En dépit de ses nombreux avantages, SMOTE présente des limitations significatives qui doivent être soigneusement prises en compte lors de son utilisation. La génération d'échantillons synthétiques irréalistes, le risque de surapprentissage, la sensibilité aux données bruitées, l'augmentation de la complexité computationnelle, et l'inefficacité face à des déséquilibres extrêmes sont autant de facteurs qui peuvent nuire à la performance des modèles.

7.2 Utilisation de SMOTE sur les Ensembles de Données Educatifs

Dans le contexte éducatif, où les données sont souvent utilisées pour évaluer l'efficacité des programmes et identifier les élèves à risque, une gestion efficace des ensembles de données déséquilibrés est essentielle. En effet, des analyses biaisées peuvent conduire à des interventions inadéquates ou à des conclusions erronées, compromettant ainsi les efforts pour améliorer l'environnement d'apprentissage et fournir un soutien ciblé aux apprenants (Tarekegn et al., 2021). Dans ce cadre, l'équilibrage des données permet non seulement d'obtenir des prédictions plus fiables, mais aussi de s'assurer que les ressources éducatives sont distribuées de manière plus équitable. Cependant, au-delà des défis superficiels posés par le déséquilibre des classes, il est important de reconnaître les complexités plus profondes qui émergent lors de l'application de ces techniques d'apprentissage automatique à des ensembles de données éducatifs. Ces complexités incluent non seulement l'asymétrie inhérente aux distributions des classes, mais aussi les biais potentiels qui peuvent être introduits lors du suréchantillonnage, particulièrement si les exemples générés ne reflètent pas fidèlement la variabilité naturelle des données (Fernández et al., 2018). De plus, l'utilisation imprudente de ces techniques peut entraîner une performance biaisée des modèles, avec des prédictions inexactes qui, en fin de compte, peuvent nuire à l'efficacité des programmes éducatifs et des interventions conçues pour

soutenir les apprenants. Par conséquent, il est impératif de choisir les techniques de suréchantillonnage avec soin, en tenant compte des caractéristiques spécifiques des ensembles de données et des objectifs éducatifs à atteindre.

(Wongvorachan et al., 2023) ont approfondi l'analyse des techniques d'échantillonnage dans le contexte éducatif, en examinant diverses méthodes pour gérer le déséquilibre des classes. Leur étude, utilisant les données de la High School Longitudinal Study de 2009, a comparé l'efficacité du suréchantillonnage aléatoire (ROS), du sous-échantillonnage aléatoire (RUS) et d'une approche hybride combinant SMOTE-NC (pour les variables nominales et continues) avec RUS. L'analyse des performances des modèles a été réalisée à l'aide de la forêt aléatoire, un algorithme de classification robuste et adapté aux grandes bases de données. Les résultats ont montré que si ROS est efficace pour les ensembles de données modérément déséquilibrés, l'approche hybride ROS-RUS surpasse les autres techniques dans les cas d'extrême déséquilibre des données. Ces conclusions ont des implications pratiques pour la mise en place de systèmes de prévision du rendement académique et des systèmes d'alerte précoce dans les établissements éducatifs. En outre, cette étude souligne l'importance d'adapter les méthodes de traitement des données en fonction du degré de déséquilibre, contribuant ainsi à l'amélioration des pratiques d'exploration de données dans le secteur éducatif.

Par ailleurs, l'étude menée par (Khalaf Hamoud et al., 2022) s'est concentrée sur l'optimisation de SMOTE en combinaison avec des techniques de sélection des caractéristiques. L'objectif était d'améliorer la prédiction des performances des étudiants en identifiant les attributs les plus corrélés avec les résultats scolaires. Quatre approches de sélection des caractéristiques ont été appliquées, avant et après l'utilisation de SMOTE, afin de mesurer leur impact sur la précision des modèles prédictifs. Les résultats ont démontré que l'application de SMOTE sans sélection de caractéristiques permettait aux algorithmes supervisés, tels que les modèles logistiques arborescents, d'atteindre une précision élevée. Toutefois, pour les algorithmes non supervisés, l'application combinée de la sélection des caractéristiques et de SMOTE a conduit à une amélioration significative de la précision des prédictions. Ces résultats mettent en lumière l'importance de la sélection de caractéristiques dans les processus de modélisation, en particulier dans un contexte éducatif où les données sont souvent déséquilibrées.

L'importance du prétraitement des données éducatives a été explorée dans l'étude (Intayoad et al., 2019), qui a évalué l'efficacité de SMOTE et de ses variantes (Borderline-SMOTE1, Borderline-SMOTE2, SVM-SMOTE) pour équilibrer les ensembles de données. Ces techniques ont été appliquées à des données complexes provenant de l'utilisation web dans le domaine éducatif, avec l'objectif de distinguer plus efficacement les étudiants ayant échoué aux examens de ceux ayant réussi. L'étude a comparé les performances de diverses méthodes de classification, notamment Naive Bayes, l'arbre de décision et les k plus proches voisins, après l'application des techniques de suréchantillonnage. Les résultats ont montré que les variantes de SMOTE sont particulièrement efficaces pour améliorer la détection des classes minoritaires, augmentant ainsi la précision, le rappel et le score F1 des modèles de classification. Ces résultats renforcent l'idée que le prétraitement des données, incluant l'utilisation de techniques de suréchantillonnage, est essentiel pour une analyse fiable des données éducatives.

Malgré ses nombreux avantages, SMOTE n'est pas exempt de limitations. Comme l'ont souligné (S. Feng et al., 2021), l'une des principales critiques est que SMOTE peut introduire du bruit ou provoquer un chevauchement des classes, conduisant à une mauvaise classification. Ce problème est souvent dû à la génération d'échantillons synthétiques qui ne reflètent pas fidèlement la distribution des données d'origine, notamment dans les contextes où la sélection des plus proches voisins et le choix du ratio de suréchantillonnage sont critiques. Pour remédier à ces problèmes, (S. Feng et al., 2021) ont proposé des variantes stables de SMOTE, visant à

réduire l'instabilité des méthodes de suréchantillonnage. Ces variantes utilisent des méthodes de sélection d'instances défectueuses basées sur la distance et une interpolation plus uniforme pour créer des échantillons synthétiques. Leur étude, qui a inclus une analyse mathématique et empirique à travers 26 ensembles de données et quatre classifieurs courants, a démontré que ces variantes stables offrent une meilleure performance et une plus grande fiabilité par rapport aux méthodes SMOTE traditionnelles.

Les résultats ont montré que les variantes stables de SMOTE surpassent largement les techniques classiques en termes de stabilité, avec des écarts de performance réduits et une meilleure constance dans les mesures d'AUC et de MCC (Coefficient de Corrélation de Matthews). Ces conclusions renforcent l'idée que, pour certaines applications sensibles, telles que la prédiction de défauts logiciels ou d'autres domaines où la fiabilité est importante, l'adoption de ces techniques stables est fortement recommandée.

L'utilisation de SMOTE dans le contexte des données éducatives constitue une approche puissante pour gérer le déséquilibre des classes et améliorer la précision des modèles prédictifs. Bien que SMOTE présente certaines limites, notamment en ce qui concerne l'introduction de bruit dans les données, les recherches récentes ont mis en lumière des variantes plus stables et efficaces. L'intégration de SMOTE, en combinaison avec des techniques de sélection des caractéristiques et d'autres approches d'échantillonnage, offre des perspectives prometteuses pour l'amélioration des analyses et des interventions dans le domaine de l'éducation. En définitive, l'importance de choisir des techniques adaptées aux spécificités des données ne saurait être surestimée, car elle conditionne la qualité des résultats obtenus et leur utilité dans un contexte éducatif.

7.3 Les Variantes de SMOTE

Diverses variantes de SMOTE ont été développées ces dernières années. Parmi celles-ci, Borderline-SMOTE, SMOTE-NC (Nominal and Continuous feautures), SVM-SMOTE, Safe-Level SMOTE, MSMOTE (Minority SMOTE), SMOTEBoost, SMOTE-Tomel Links.

7.3.1 SMOTE-Borderline

SMOTE- Borderline améliore le SMOTE original en se concentrant sur les échantillons qui sont à la limite entre les classes, ce qui le rend plus efficace pour traiter les ensembles de données déséquilibrés avec des classes qui se chevauchent (H. Han et al., 2005). Elle peut générer des échantillons synthétiques dans des régions de l'espace des caractéristiques qui sont plus difficiles à apprendre pour les classifieurs améliorant ainsi leurs performances. Néanmoins, SMOTE-Borderline peut générer des échantillons bruyants si la limite de décision n'est pas bien définie (Nabus et al., 2022). Elle peut également s'avérer coûteuse en termes de calcul lorsque l'ensemble de données comporte un grand nombre d'échantillons limites. Son efficacité dépend fortement de la qualité de l'algorithme d'identification de la région limite.

7.3.1.1 Utilisation de SMOTE Borderline sur les ensembles de données éducatifs

Dans le domaine de l'éducation, SMOTE Borderline a été utilisé pour traiter les ensembles de données déséquilibrés, tels que ceux relatifs aux performances académiques des étudiants. L'objectif est de prédire avec précision les performances des étudiants, en particulier ceux à risque d'échec, afin de pouvoir intervenir de manière proactive. Par exemple, l'étude menée par (Rachburee & Punlumjeak, 2021) s'est concentrée sur l'amélioration des modèles prédictifs dans le cadre éducatif en utilisant le SMOTE Borderline.

L'ensemble de données concernait les performances académiques des étudiants, avec une forte disproportion entre les étudiants performants (classe majoritaire) et ceux en difficulté (classe minoritaire). Pour équilibrer les classes, les chercheurs ont appliqué le SMOTE Borderline, en

se concentrant sur les échantillons situés à la frontière de décision. Cela a permis de générer des exemples synthétiques pour les étudiants en difficulté, tout en évitant les risques de sur-génération de données dans les régions moins critiques de l'espace de caractéristiques. Les modèles prédictifs développés après l'application du SMOTE Borderline ont montré une amélioration significative en termes de précision, rappel et F1-score par rapport aux autres méthodes de rééchantillonnage. Les instructeurs ont ainsi pu mieux identifier les étudiants en difficulté et mettre en place des interventions ciblées pour améliorer leurs performances académiques. De ce fait, l'utilisation du SMOTE Borderline a permis d'affiner la capacité des modèles à repérer les cas critiques dans des environnements d'apprentissage, aidant ainsi à améliorer l'expérience et les résultats éducatifs des apprenants.

7.3.2 SMOTE-ENN

SMOTE-ENN est une méthode populaire utilisée pour les problèmes de classification dans l'apprentissage automatique. SMOTE-ENN a été proposé pour améliorer les performances des classifieurs sur des données déséquilibrées en tirant parti des atouts de SMOTE et d'ENN. Cette technique a été introduite pour la première fois par (Batista et al. en 2004). SMOTE-ENN vise à améliorer la qualité de la classe minoritaire en créant des exemples synthétiques à l'aide de SMOTE et en réduisant simultanément le nombre d'exemples bruyants à l'aide d'ENN. Cela permet d'améliorer les performances du classifieur en équilibrant la distribution des classes et en réduisant l'impact du bruit. En fait, SMOTE et ENN ont tous deux leurs propres forces et faiblesses. SMOTE est efficace pour générer de nouvelles instances synthétiques afin d'améliorer la classe minoritaire, mais elle peut générer des exemples bruyants. ENN peut filtrer ces exemples bruyants, mais elle peut également supprimer des instances utiles de la classe minoritaire (Batista et al., 2004). La technique combinée SMOTE-ENN a donc le potentiel d'améliorer la robustesse du classifieur en améliorant la classe minoritaire tout en réduisant le bruit.

Plus précisément, cette méthode permet de résoudre le problème de surajustement qui se pose souvent lorsque SMOTE est appliqué à de grands ensembles de données (S. Wang et al., 2021). Elle consiste à créer de nouvelles instances synthétiques de la classe minoritaire, qui sont ensuite utilisées pour équilibrer la distribution des classes.

L'algorithme SMOTE-ENN est mis en œuvre comme suit :

- Générer des exemples synthétiques de la classe minoritaire à l'aide de SMOTE.
- Supprimer les exemples identifiés comme bruyants par ENN.
- Former un classifieur sur l'ensemble de données équilibré.
- Utiliser le classifieur formé pour classer les nouveaux exemples.
- Si nécessaire, utiliser le classifieur formé pour générer de nouvelles instances synthétiques.

Les avantages de l'utilisation de SMOTE-ENN sont les suivants (Al-Ashoor & Abdullah, 2022):

- Amélioration des performances des classifieurs sur des ensembles de données déséquilibrés.
- Réduction de l'impact du bruit.
- Génération efficace des exemples synthétiques de la classe minoritaire tout en supprimant les exemples bruyants à l'aide d'ENN.
- Applicabilité : l'algorithme est facile à mettre en œuvre et peut être utilisé avec différents classifieurs.

Cependant, SMOTE-ENN présente certaines limites :

- Elle peut conduire à un surajustement et réduire les performances de généralisation du classifieur (Krawczyk, 2016).
- Elle peut ne pas fonctionner correctement avec des ensembles de données plus petits.
- Le choix des paramètres peut avoir un impact significatif sur les performances du classifieur.
- Elle est coûteuse en termes de calcul et peut nécessiter des ressources supplémentaires.

7.3.2.1 Utilisation de SMOTE-ENN sur les Ensembles de Données Educatifs

La technique de suréchantillonnage SMOTE-ENN combine les points forts des techniques de sous-échantillonnage et de suréchantillonnage en utilisant la règle Edited Nearest Neighbors. Elle s'est avérée être une approche efficace pour améliorer la précision globale des modèles d'apprentissage. En effet, de nombreuses études ont montré que SMOTE-ENN peut améliorer la précision de prédiction des modèles lorsqu'elle est appliquée à des ensembles de données éducatifs.

Dans l'étude (Nabil et al., 2021), l'objectif principal était d'évaluer l'efficacité des techniques d'apprentissage profond (deep learning) dans le domaine de l'exploitation des données éducatives pour prédire les performances académiques des étudiants, en particulier pour identifier les étudiants susceptibles d'échouer dans des cours tels que "Programmation" et "Structures de données" au cours de leur première année universitaire. Pour atteindre cet objectif, les chercheurs ont utilisé un ensemble de données collectées dans le cadre d'une université publique sur une période de quatre ans. Ce jeu de données a été exploité pour développer des modèles prédictifs visant à estimer les performances des étudiants dans des cours à venir en se basant sur leurs notes dans les cours précédents de la première année académique. Plusieurs techniques d'apprentissage machine ont été utilisées, notamment un réseau neuronal profond (DNN), un arbre de décision, une régression logistique, un classifieur à vecteurs de support (SVC) et la méthode des k plus proches voisins. De plus, afin de résoudre le problème de déséquilibre dans le jeu de données, des techniques de rééchantillonnage ont été comparées, telles que SMOTE, ADASYN, ROS et SMOTE-ENN. Ces méthodes visent à équilibrer les classes minoritaires et majoritaires dans les données, améliorant ainsi la capacité des modèles à prédire précisément les performances académiques des étudiants.

Les résultats expérimentaux ont révélé que le modèle DNN proposé surpassait les autres modèles testés, offrant une précision de prédiction de 89 % pour les performances des étudiants dans le cours de structures de données. De plus, le modèle DNN s'est avéré plus performant pour identifier les étudiants à risque d'échec à un stade précoce du semestre, dépassant ainsi en précision les autres modèles tels que l'arbre de décision, la régression logistique, le classifieur à vecteurs de support et la méthode des k plus proches voisins. Ces résultats suggèrent que les techniques d'apprentissage profond, en particulier le réseau neuronal profond, peuvent jouer un rôle déterminant dans la prédiction des performances académiques des étudiants et dans l'identification précoce des risques d'échec. De plus, l'utilisation de techniques de rééchantillonnage comme SMOTE-ENN a contribué à améliorer la qualité des prédictions en résolvant le problème de déséquilibre des données.

De même, l'étude de (Ghorbani & Ghousi, 2020) a visé l'utilisation de différentes techniques de rééchantillonnage, dont SMOTE-ENN, pour résoudre le problème des ensembles de données déséquilibrés dans la prédiction des performances des étudiants. Dans le contexte actuel, la prédiction des performances des étudiants est devenue une préoccupation majeure, favorisée par les avancées technologiques. L'exploitation des données en utilisant le data mining s'avère nécessaire dans le domaine de l'éducation, notamment pour l'analyse des performances étudiantes. L'étude entreprend une comparaison entre plusieurs techniques de rééchantillonnage, telles que Borderline SMOTE, Random Over Sampler, SMOTE, SMOTE-ENN, SVM-SMOTE et SMOTE-Tomek, pour gérer cette problématique de déséquilibre. Deux

ensembles de données distincts sont utilisés pour évaluer ces méthodes. Pour évaluer ces techniques, différents classifieurs d'apprentissage machine sont utilisés, parmi lesquels la forêt aléatoire, K plus proches voisins, XG-boost, SVM (Radial Basis Function), l'arbre de décision et la régression logistique. En outre, pour la validation des modèles, des méthodes telles que Random hold-out et la validation croisée sont appliquées.

Les résultats obtenus grâce à diverses métriques d'évaluation indiquent que des ensembles de données avec moins de classes et des caractéristiques nominales conduisent à de meilleures performances des modèles. De plus, il est souligné que les classifieurs ne se comportent pas bien avec des données déséquilibrées, soulignant ainsi la nécessité de résoudre ce problème. L'étude démontre que l'utilisation de jeux de données équilibrés améliore les performances des classifieurs. En utilisant le test de Friedman, un test de signification statistique, les résultats confirment que la technique de rééchantillonnage SVM-SMOTE et SMOTE-ENN sont plus efficaces que les autres méthodes.

7.3.3 ADASYN

La technique ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning) a été proposée pour traiter les ensembles de données déséquilibrés et surmonter les limites des classifieurs traditionnels (He et al., 2008). C'est une extension de SMOTE, ADASYN génère des échantillons synthétiques pour équilibrer la distribution des classes déséquilibrées en estimant la distribution de la densité des échantillons de la classe minoritaire. ADASYN fonctionne de manière adaptative, dans le but d'augmenter le niveau de difficulté de l'apprentissage à partir des instances minoritaires à chaque itération. Le niveau de difficulté des points de données générés est déterminé en fonction de la proximité de l'échantillon minoritaire par rapport à la limite de décision. Plus l'échantillon est proche de la limite, plus le niveau de difficulté est élevé et plus le nombre d'échantillons synthétiques générés est important. ADASYN met à jour la distribution et génère des échantillons synthétiques jusqu'à ce que l'équilibre de la distribution des classes soit atteint.

La technique ADASYN a été proposée pour résoudre le problème du déséquilibre de la distribution des classes qui prévaut dans de nombreuses applications du monde réel. Ainsi, ADASYN est utilisé avec plusieurs classifieurs d'apprentissage automatique, tels que les arbres de décision, les forêts aléatoires, KNN et Naive Bayes, pour générer ses données synthétiques. Les performances des classifieurs sont considérablement améliorées avec un minimum d'itérations lorsqu'ils sont appliqués avec le suréchantillonnage ADASYN aux ensembles de données déséquilibrés. Par ailleurs, ADASYN a démontré des avantages significatifs dans la performance de son application dans le traitement des ensembles de données déséquilibrés. Les principaux avantages de l'algorithme sont:

- La caractéristique adaptative de cet algorithme lui permet de s'ajuster à la distribution de la densité de la classe minoritaire en fonction des exigences de l'application et des niveaux de déséquilibre (He et al., 2008).
- L'introduction d'ADASYN offre une solution pour traiter manuellement de manière ciblée les niveaux importants de déséquilibre entre les classes.
- L'utilisation des données synthétiques d'ADASYN améliore considérablement les performances des classifieurs dans la détection de la classe minoritaire, tout en réduisant le taux de faux négatifs et en améliorant la précision et le rappel globaux du modèle (Fernández et al., 2018).

Malgré ses avantages, l'application de l'algorithme ADASYN présente certaines limites notamment (Al-Ashoor & Abdullah, 2022):

- La validation de la méthode reste assez restreinte sur des ensembles de données de grande dimension impliquant des relations complexes et non linéaires.

- Une autre limite est que la méthode est souvent surajoutée aux données d'apprentissage, ce qui entraîne une mauvaise classification sur de nouveaux ensembles de données.
- ADASYN peut également générer des données synthétiques susceptibles d'entraîner une sursaturation de l'ensemble de données, ce qui peut affecter la précision de généralisation du classifieur.

Par conséquent, il faut évaluer les performances des classifieurs sur de nouveaux ensembles de données. De plus, un ajustement fin des paramètres peut s'avérer nécessaire pour atténuer les limites de performance associées à l'algorithme de suréchantillonnage synthétique d'ADASYN.

7.3.3.1 Utilisation d'ADASYN sur les Ensembles de Données Educatifs

L'étude de (Rozi et al., 2023) se concentre sur le domaine émergent du Data Mining éducatif (EDM), qui s'avère important pour évaluer les performances des étudiants dans les cours, voire pour anticiper leurs accomplissements futurs. Le EDM offre un potentiel considérable en générant des modèles basés sur la connaissance, utiles tant pour les enseignants que pour les étudiants. Toutefois, dans ce domaine, on est confronté à un problème récurrent : le déséquilibre des classes. Dans cette recherche (Rozi et al., 2023), l'étude met en évidence un déséquilibre marqué dans la distribution des classes, en particulier dans un ensemble de données à classes multiples comprenant plus de deux étiquettes de classe. Pour aborder cette problématique, l'étude se focalise sur la gestion des classes déséquilibrées et sur la classification, utilisant différentes méthodes telles que la régression linéaire, la forêt aléatoire et le stacking pour la classification, ainsi que SMOTE, ADASYN et SMOTE-ENN pour les algorithmes de rééchantillonnage. Ces méthodes sont évaluées à l'aide d'une validation croisée à 10 volets et d'un ratio de division de 80%-20%.

Les résultats obtenus indiquent que la meilleure performance est observée avec la classification via la méthode du stacking appliquée à un jeu de données rééchantillonné par ADASYN, avec un score F1 de 97%. L'étude démontre également que les techniques de rééchantillonnage améliorent les performances de classification. Cependant, même sans rééchantillonnage, la classification produit des résultats satisfaisants, probablement car les schémas généraux des données pour chaque classe étaient déjà de bonne qualité initialement. Ainsi, l'étude suggère que le traitement des données originales n'engendre pas nécessairement de désavantages, du moins dans ce contexte spécifique.

Les techniques de suréchantillonnage telles que SMOTE et SMOTE-ENN et ADASYN offrent des stratégies efficaces pour traiter les ensembles de données éducatifs déséquilibrées. Ces techniques ont le potentiel d'améliorer la précision de prédiction des modèles et de fournir une compréhension plus approfondie des résultats des apprenants quand elles sont appliquées à des ensembles de données éducatifs. Cependant, il est essentiel de prendre en compte les forces et les limites de ces techniques lors du développement de modèles. Toutefois, SMOTE est généralement préférée lorsqu'il s'agit d'ensembles de données fortement déséquilibrés, car cette méthode génère davantage d'échantillons synthétiques et peut gérer plusieurs classes minoritaires. En revanche, SMOTE-ENN est plus efficace lorsqu'il s'agit de données bruitées ou de classes qui se chevauchent.

Alors que les techniques de suréchantillonnage jouent un rôle critique dans l'amélioration de la qualité des modèles prédictifs pour les ensembles de données déséquilibrés, les avancées en intelligence artificielle vont bien au-delà du traitement des déséquilibres de données. Une autre révolution dans le domaine de l'éducation est apportée par les modèles de langage pré-entrainés. Ces modèles utilisent des techniques de traitement du langage naturel pour évaluer de manière plus nuancée la compréhension des étudiants et personnaliser les expériences d'apprentissage.

8 Introduction aux Modèles de Langage Pré-entrainé

Dans le domaine de l'éducation, en particulier dans les environnements d'apprentissage et les plateformes d'E-Learning, l'IA a joué un rôle clé dans le développement de méthodes permettant d'évaluer la compréhension des étudiants de manière plus précise (González-Calatayud et al., 2021; Rahaman & Hoque, 2022). Les méthodes d'évaluation traditionnelles reposent souvent sur des tests standardisés ou des questionnaires, qui peuvent manquer de profondeur pour évaluer la compréhension conceptuelle d'un étudiant. Cependant, avec les avancées de l'IA, telles que le traitement du langage naturel (NLP), les modèles de langages pré-entrainés, et l'analyse de données, de nouvelles approches pour évaluer la compréhension des apprenants ont émergé. Ces méthodes évaluent non seulement l'exactitude des réponses des étudiants, mais aussi la qualité et la profondeur de leur compréhension des concepts enseignés. Donc, ces analyses vont au-delà d'une simple vérification de la réponse correcte et examinent la structure, le vocabulaire utilisé, la cohérence et la profondeur de la réponse. De plus, ces systèmes intègrent souvent des outils d'évaluation adaptatifs qui ajustent dynamiquement le niveau de difficulté ou le contenu des cours en fonction des réponses des apprenants en temps réel (Gardner et al., 2021).

Le Traitement du langage naturel est un domaine de l'intelligence artificielle qui se concentre sur l'interaction entre les ordinateurs et les humains via le langage naturel. Il vise à permettre aux machines de comprendre, interpréter et générer du langage humain de manière pertinente. Les techniques de NLP englobent la reconnaissance de la parole, l'analyse sémantique, la traduction automatique, et plus encore (Sawicki et al., 2023). Ce domaine combine des concepts de linguistique computationnelle, d'apprentissage automatique et de modélisation statistique pour traiter et analyser de grandes quantités de données textuelles (Kurni et al., 2023).

8.1 Les Techniques de Traitement du Langage Naturel dans l'Éducation

Les techniques de NLP sont nécessaires pour préparer les données textuelles aux applications éducatives. Parmi ces méthodes, on cite :

- ✓ La tokenisation: c'est une technique qui divise le texte en unités minimales appelées tokens, qui peuvent être des mots, des phrases ou même des sous-parties de mots. Ce processus permet la réalisation d'une analyse syntaxique et sémantique, facilitant la recherche d'expressions spécifiques et l'extraction d'informations (S. F. Chaerul Haviana et al., 2023).
- ✓ Stemming: cette technique représente la réduction des mots à leur forme de base (stem), en supprimant les suffixes dérivés (running → run). Cette méthode permet de regrouper les variantes morphologiques d'un mot, mais elle peut parfois entraîner une perte de précision sémantique (S. F. Chaerul Haviana et al., 2023).
- ✓ Lemmatisation: c'est le processus de réduction des mots à leur lemme, ou forme canonique, en tenant compte du contexte grammatical (exemple : better est le lemme de good dans certains contextes). Contrairement au stemming, la lemmatisation préserve la précision sémantique en tenant compte de la partie du discours (verbe, nom, etc.) (S. F. Chaerul Haviana et al., 2023).
- ✓ Sémantique et désambiguïsation du sens des mots: ce sont des techniques visant à déterminer le sens précis d'un mot en fonction de son contexte. La sémantique computationnelle permet d'attribuer un sens aux mots et aux phrases dans un texte

donné. Cette méthode est utilisée pour résoudre les ambiguïtés lexicales et améliorer la compréhension automatique des textes (Sawicki et al., 2023).

L'application du Traitement du Langage Naturel dans l'éducation illustre de manière convaincante comment les outils basés sur l'intelligence artificielle peuvent transformer l'efficacité des évaluations (Lan et al., 2024; Litman, 2016). Par exemple, (Kasneci et al., 2023) discutent de l'utilisation des grands modèles de langage (LLMs) dans le domaine de l'éducation. Ils soulignent que ces modèles peuvent non seulement générer du contenu éducatif, mais aussi analyser les réponses des étudiants pour offrir des retours personnalisés qui tiennent compte des besoins individuels d'apprentissage. Ces recherches montrent que le NLP permet d'apporter des évaluations plus nuancées et adaptatives, en transformant les pratiques éducatives traditionnelles (Hwang et al., 2020).

8.2 Modèles de Langage Pré-entraînés

Les modèles de langage pré-entraînés tels que BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) et RoBERTa (Robustly Optimized BERT Approach) (Liu et al., 2019) ont été largement utilisés dans l'éducation pour améliorer divers aspects de l'enseignement et de l'apprentissage, y compris l'analyse de texte pour évaluer la compréhension des étudiants, la génération automatique de résumés, la notation des réponses des étudiants et la personnalisation de l'instruction. Par exemple, la génération de questions pour la compréhension de lecture a été appliquée dans des contextes éducatifs pour créer des tuteurs de lecture qui fournissent des questions de compréhension adaptées aux capacités de lecture des apprenants (Uto et al., 2023). En outre, BERT et RoBERTa ont été développés davantage pour évaluer les Cours en Ligne Ouverts et Massifs (MOOC) en utilisant l'apprentissage adversarial, contribuant à améliorer la qualité de l'éducation en ligne. Par exemple, une variante novatrice nommée MOOC-BERT a été conçue pour identifier la présence cognitive dans les environnements MOOC. Cette variante a montré des performances améliorées par rapport aux modèles d'apprentissage profond en exploitant des données de discussion non étiquetées à grande échelle (Z. Liu et al., 2023). Par ailleurs, dans un contexte d'évaluation automatisée, (Fernandez et al., 2022) ont proposé une nouvelle méthode de notation automatique pour l'évaluation de la compréhension de lecture, s'appuyant sur un modèle BERT ajusté contextuellement. Cette approche innovante a non seulement amélioré la précision de la notation, mais a aussi optimisé l'efficacité du processus, réduisant ainsi la charge de travail des correcteurs humains.

8.3 BERT

BERT est un modèle de traitement du langage naturel développé par (Devlin et al., 2019), qui a révolutionné la manière dont les modèles de NLP sont conçus et utilisés. BERT repose sur une architecture de transformateur bidirectionnel, ce qui lui permet de comprendre le contexte d'un mot en analysant les mots qui le précèdent et le suivent dans une phrase. Cette approche marque une avancée significative par rapport aux modèles traditionnels de NLP, qui étaient généralement unidirectionnels et donc moins efficaces pour comprendre le contexte global d'une phrase (Rogers et al., 2020).

Le modèle BERT est pré-entraîné sur un vaste corpus de texte, incluant des données de Wikipédia et des livres, ce qui lui permet de généraliser efficacement à de nombreuses tâches de NLP après un ajustement (fine-tuning) spécifique à la tâche en question. Par exemple, pour une tâche de classification de texte, le modèle BERT pré-entraîné peut être ajusté avec un ensemble de données spécifiques à cette tâche pour améliorer les performances.

8.3.1 Architecture de BERT

L'architecture de BERT (voir Figure III.10) repose sur le Transformateur (Vaswani et al., 2017), un modèle de deep learning qui a révolutionné le traitement du langage naturel grâce à son efficacité et sa capacité à capturer les relations complexes entre les mots dans une séquence. Contrairement aux modèles séquentiels traditionnels comme les RNN (Réseaux de Neurones Récurrents), BERT utilise une architecture basée uniquement sur des encoders. L'encoder est une composante d'un modèle de transformateur qui transforme une séquence d'entrée en une représentation contextuelle plus abstraite, capturant les relations entre les éléments de la séquence via des mécanismes d'attention. Il fonctionne en empilant plusieurs couches de self-attention et de réseaux de neurones feed-forward pour produire des représentations de plus en plus complexes (Rogers et al., 2020).

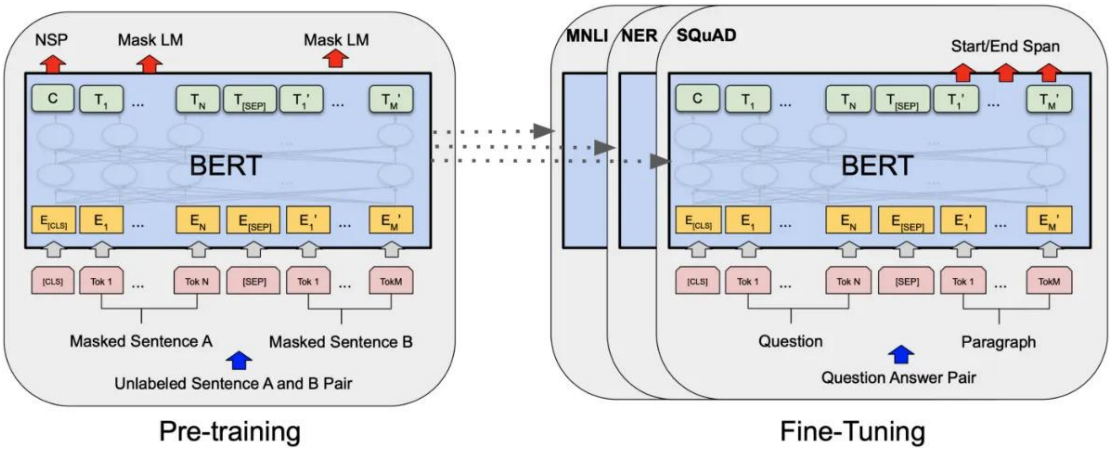

Figure III.10 Architecture de Bert (CodeSerra, 2022)

1) **Mécanisme d'Attention**
   Le cœur de l'architecture de BERT est le mécanisme d'attention. Dans le contexte des Transformateurs, l'attention permet au modèle de pondérer différemment chaque mot dans une séquence en fonction de sa pertinence par rapport à la tâche (Sharma et al., 2022). Par exemple, pour une tâche de compréhension de texte, certains mots peuvent être plus informatifs que d'autres, et l'attention permet au modèle de se concentrer sur ces mots clés tout en ignorant ceux moins importants. Le mécanisme d'attention est défini par une matrice qui calcule un score de similarité entre chaque mot de la séquence d'entrée et les autres mots, permettant ainsi au modèle de déterminer l'importance relative de chaque mot par rapport à tous les autres (Devlin et al., 2019).
2) **Self-Attention**
   La self-attention est une version spécifique du mécanisme d'attention dans laquelle chaque mot d'une séquence d'entrée est mis en relation non seulement avec les autres mots de la séquence, mais également avec lui-même. Cela permet au modèle de capturer des dépendances à longue distance dans la séquence permettant ainsi de comprendre le contexte global d'une phrase ou d'un texte. Dans BERT, chaque couche de l'encoder utilise un mécanisme de multi-head self-attention, où plusieurs mécanismes d'attention sont appliqués en parallèle. Cela permet de capturer différents types de relations entre les mots (Devlin et al., 2019).
3) **Architecture Encoder-Only**
   Contrairement aux architectures encoder-decoder utilisées dans d'autres modèles comme GPT (Generative Pretrained Transformer), BERT est basé uniquement sur des encoders (Devlin et al., 2019). Une approche adaptée aux tâches de compréhension de texte, car elle permet de traiter simultanément toutes les positions d'une séquence

d'entrée sans avoir besoin de décoder les sorties une par une. Chaque encoder dans BERT est composé de deux sous-couches principales :

- Multi-Head Self-Attention: comme mentionné précédemment, cette sous-couche permet de capturer les relations contextuelles entre les mots d'une séquence.
- Feed-Forward Neural Network (FFNN): après le mécanisme d'attention, les représentations obtenues sont passées à travers un réseau de neurones classique pour générer les sorties finales permettant de modéliser des transformations non linéaires complexes des représentations textuelles. Afin de capturer des représentations de plus en plus abstraites du texte à chaque couche, ces encoders sont empilés en plusieurs couches.

4) **Apprentissage et Pré-entrainement**

Le modèle BERT est pré-entraîné sur deux tâches principales: le masquage de mots (MLM) et la prédiction de la phrase suivante (NSP). Ces deux tâches tirent parti de l'architecture bidirectionnelle de BERT, qui utilise l'information contextuelle à la fois des mots précédents et des mots suivants dans une phrase.

- Masked Language Modeling (MLM): cette tâche consiste à masquer aléatoirement certains mots dans une séquence d'entrée, puis à entraîner le modèle à prédire ces mots masqués en se basant sur le contexte des mots environnants (Li et al., 2020).
- Next Sentence Prediction (NSP): cette tâche permet au modèle de comprendre les relations entre les phrases en l'entraînant à prédire si une phrase donnée est susceptible de suivre une autre phrase (Li et al., 2020).

Les mécanismes avancés de BERT lui permettent de saisir les subtilités du langage, ce qui en fait un outil précieux pour diverses applications, y compris celles liées à l'éducation.

8.3.2 Utilisation de BERT dans le Domaine de l'Éducation

L'application de BERT dans le domaine de l'éducation a montré un potentiel considérable pour améliorer l'apprentissage automatique et l'analyse du texte, notamment dans des tâches telles que la détection de plagiat, la génération de résumés, l'analyse des réponses aux questions, et la génération de textes éducatifs. Par exemple, l'approche de (Viji & Revathy, 2022) repose sur une combinaison de BERT et de modèles profonds pour améliorer la détection de similarité sémantique, un défi majeur dans l'analyse de textes. Leur méthode utilise BERT pour extraire des représentations contextuelles riches, capturant ainsi la signification profonde des phrases. Ensuite, ils intègrent ces représentations dans un réseau de neurones récurrents bidirectionnel Siamese Bi-LSTM, qui est conçu pour comparer efficacement les paires de textes. Cette architecture Siamese permet de mesurer la similarité en calculant la distance entre les vecteurs des phrases traitées. Leur modèle a montré une précision remarquable de 91 % dans la détection des similarités sémantiques, surpassant les méthodes traditionnelles.

Ainsi, Dans un contexte éducatif, cette approche présente des avantages significatifs, notamment dans la détection de plagiat ou l'identification de similitudes entre les travaux des étudiants. Les méthodes traditionnelles de détection de plagiat reposent souvent sur des comparaisons lexicales superficielles, qui échouent à reconnaître les similarités lorsque les étudiants reformulent les phrases. En revanche, l'utilisation de BERT permet de comprendre le sens des textes au-delà des mots exacts, rendant la détection de plagiat plus précise et moins vulnérable aux reformulations.

Cependant, des recherches telles que celles de Li et al. (2020) ont révélé que les plongements lexicaux (embedding) de phrases générées par BERT peuvent être limités dans leur capacité à capturer pleinement le sens sémantique, ce qui peut nuire aux performances dans les tâches de similarité sémantique. Pour pallier ces limitations, ils ont proposé BERT-flow, une technique qui transforme les embeddings de phrases de BERT en représentations plus fluides et efficaces,

ce qui a montré une amélioration significative des performances sur diverses tâches de similarité sémantique.

D'autres études, comme celle de (Peinelt et al., 2020), ont exploré l'intégration d'informations contextuelles supplémentaires, telles que des informations de sujet, pour améliorer la compréhension du langage naturel. Leur approche a démontré des gains significatifs dans les cas spécifiques aux domaines, ce qui est particulièrement pertinent pour l'éducation, où les contenus sont souvent spécialisés.

Plusieurs variantes de BERT ont été développées pour améliorer ses performances dans des contextes spécifiques. L'une des plus notables est RoBERTa (Liu et al., 2019), qui modifie les méthodes de pré-entraînement en utilisant un plus grand corpus de données et en supprimant certaines contraintes, telles que la tâche de prédiction de la phrase suivante. Ces modifications permettent à RoBERTa de surpasser BERT sur plusieurs tâches de NLP, notamment celles impliquant la modélisation du contexte dans un document.

8.4 RoBERTa

RoBERTa est une version améliorée du modèle BERT, introduite par (Liu et al., 2019). Ce modèle a été conçu pour surmonter certaines limitations observées dans BERT, notamment en termes de performance et d'efficacité sur diverses tâches de traitement du langage naturel. En ajustant les techniques de pré-entraînement et en optimisant l'utilisation des données, RoBERTa est un modèle incontournable pour les tâches de NLP, surpassant souvent BERT. Ce modèle a prouvé ses performances dans des applications variées, allant de la classification de texte à la génération de réponses, et a également montré un potentiel important dans le domaine de l'éducation.

8.4.1 Architecture de RoBERTa

RoBERTa est basé sur l'architecture Transformer, tout comme BERT, mais avec des ajustements significatifs dans le processus de pré-entraînement. Contrairement à BERT, qui utilise le masquage des tokens et la prédiction de la phrase suivante comme principales tâches de pré-entraînement (Devlin et al., 2019), RoBERTa se concentre uniquement sur le masquage des tokens mettant de côté la tâche de prédiction de la phrase suivante. Ce changement permet de mieux exploiter les données de pré-entraînement et d'améliorer la performance sur les tâches en aval.

Par ailleurs, l'une des principales caractéristiques de RoBERTa est l'augmentation de la taille des mini-lots (mini-batches) et la formation sur de plus grands ensembles de données. En effet, RoBERTa a été formé sur un corpus de 160 Go de texte provenant de diverses sources, ce qui est nettement supérieur au corpus utilisé pour BERT. De plus, le modèle utilise un taux d'apprentissage adaptatif et un nombre d'étapes de formation beaucoup plus élevé, ce qui lui permet de mieux généraliser sur des tâches variées. Cependant, le modèle conserve l'architecture Transformer à 12 couches avec des mécanismes d'attention, permettant au modèle de capturer efficacement les relations contextuelles entre les mots dans une séquence (Liu et al., 2019). Cette architecture améliore la compréhension du contexte dans les textes, ce qui est primordial pour des tâches telles que la classification de texte, la traduction, et la génération de réponses.

L'architecture de RoBERTa est essentiellement similaire à celle de BERT, utilisant des couches d'encoder avec des mécanismes de self-attention et de multi-head attention, mais avec des optimisations dans l'entraînement. Les différences entre l’architecture de BERT et RoBERTa sont:

- Données d'entraînement étendues: RoBERTa est pré-entraîné sur un corpus de données plus vaste.
- Pas de prédiction de phrase suivante: contrairement à BERT, RoBERTa ne utilise pas la tâche NSP lors de l'entraînement, ce qui simplifie l'architecture et améliore les performances sur certaines tâches.
- Hyperparamètres ajustés: RoBERTa utilise des tailles de batch plus grandes et des périodes d'entraînement plus longues, ce qui contribue à une amélioration des performances par rapport à BERT.

8.4.2 Utilisation de RoBERTa dans le Domaine de l'Éducation

Dans le domaine de l'éducation, RoBERTa peut être utilisé pour diverses applications, allant de l'automatisation de la correction des devoirs à l'analyse des performances des étudiants. L'un des domaines où RoBERTa est particulièrement utile est l'évaluation automatique des textes écrits par les étudiants. Grâce à ses capacités de compréhension contextuelle, RoBERTa peut analyser les rédactions des étudiants, évaluer leur pertinence, leur cohérence, et leur structure, et fournir un retour d'information précis (M. A. Tayal et al., 2023). Par exemple, dans les systèmes d'évaluation automatisée des essais, RoBERTa peut être utilisé pour évaluer la grammaire, le style, et la logique des arguments des étudiants, tout en fournissant des recommandations pour améliorer leur écriture.

Par ailleurs, RoBERTa excelle à identifier des sentiments, des intentions dans de larges corpus de données textuelles (МЕНЦИЕВ А et al., 2023). Le modèle a également été utilisé pour la génération de réponses et la compréhension des questions. Il a montré des performances supérieures par rapport à BERT et à d'autres modèles. Par exemple, dans les systèmes de questions-réponses, RoBERTa est capable de comprendre les nuances des questions posées et de générer des réponses précises en s'appuyant sur les connaissances contextuelles capturées lors de son pré-entraînement (H. Sun et al., 2018).

8.5 CodeBERT

CodeBERT, développé par (Z. Feng et al., 2020), est un modèle pré-entraîné conçu pour traiter à la fois les langages de programmation et les langages naturels. Basé sur l'architecture Transformer de (Vaswani et al., 2017), ce modèle est une extension des avancées réalisées avec BERT, mais spécifiquement adapté pour les tâches liées à la compréhension, la recherche, la documentation et à la génération de code.

CodeBERT est un modèle bimodal, ce qui signifie qu'il peut traiter à la fois des données en langage naturel et en langage de programmation. Il a été pré-entraîné sur un vaste corpus de données comprenant 2,1 millions de paires de données bimodales (liens entre le code et les descriptions en langage naturel) et 6,4 millions de données de code monomodales. L'architecture Transformer de CodeBERT lui permet de capturer efficacement le contexte des langages de programmation, tout en maintenant une compréhension fine du langage naturel (Z. Feng et al., 2020).

8.5.1 Architecture de CodeBERT

L'architecture de CodeBERT est basée sur l'encoder de BERT, avec des ajustements pour mieux gérer le code source et les langages de programmation (Figure III.11).

Le processus de pré-entraînement de CodeBERT comprend deux tâches principales : le masquage des tokens et la prédiction de la séquence suivante, similaires aux méthodes utilisées pour BERT, mais adaptées aux spécificités des langages de programmation, ce qui lui permet de capturer des relations spécifiques aux langages de programmation. Par exemple, dans la tâche de masquage des tokens, certains éléments du code ou du texte sont masqués et le modèle

doit les prédire en utilisant le contexte environnant, ce qui améliore sa capacité à comprendre et à générer du code syntaxiquement correct (E. Mashhadi & H. Hemmati, 2021). Par ailleurs, CodeBERT est pré-entraîné sur un corpus combinant du texte et du code source provenant de diverses langues de programmation. Il est donc spécialement conçu pour comprendre les aspects syntaxiques et sémantiques du code.

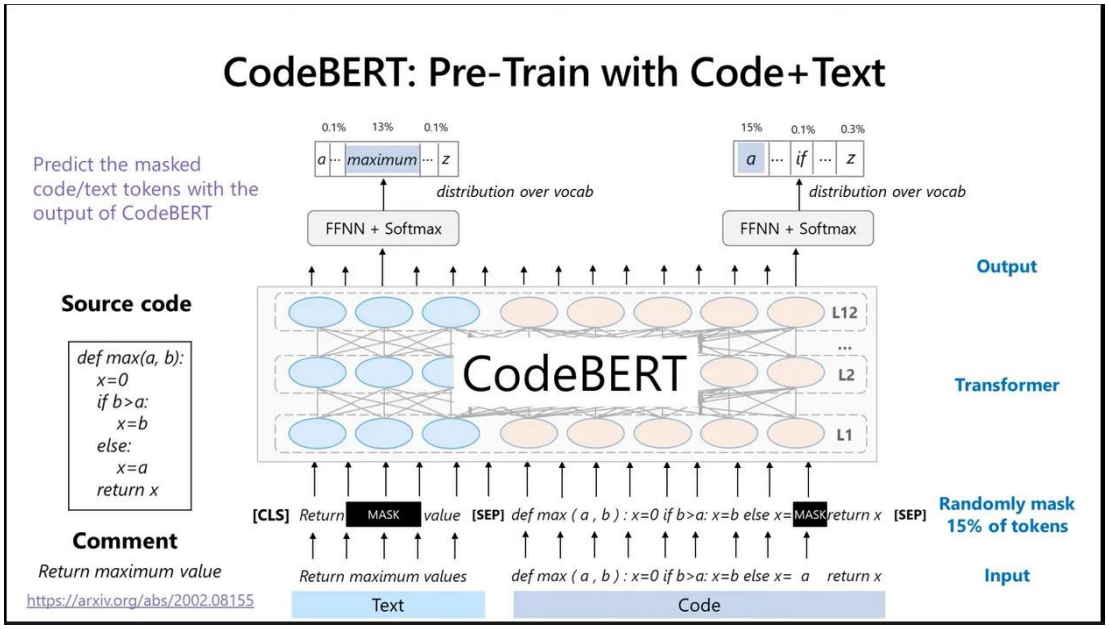

Figure III.11 Architecture de CodeBert (CodeSerra, 2022)

8.5.2 Utilisation de CodeBERT dans le Domaine de l'Education

Dans le domaine de l'éducation, CodeBERT a été utilisé pour automatiser l'évaluation du code. Par exemple, (Muddaluru et al., 2023) ont utilisé CodeBERT pour l'évaluation automatique de code en langage C. Les résultats obtenus de l'étude ont montré que CodeBERT a permis d'améliorer la précision des notations automatiques en capturant non seulement la syntaxe du code, mais aussi sa sémantique. Cependant, bien que les modèles d'apprentissage profond, tels que les LSTM, aient montré des performances impressionnantes dans la notation automatique, ils souffrent d'un manque d'interprétabilité. Pour pallier cela, des modèles plus simples, tels que les forêts aléatoires, bien que moins précis, se sont avérés plus adaptés pour fournir des explications claires et des aperçus pédagogiques, particulièrement utiles dans des contextes où les données sont limitées. En outre, l'utilisation de CodeBERT pour la détection des clones de code dans les soumissions des étudiants permet de détecter le plagiat. La capacité de CodeBERT à identifier des similitudes non triviales dans le code facilite également l'identification des erreurs courantes et des motifs récurrents dans les travaux des étudiants, ce qui permet aux enseignants de mieux cibler leurs efforts pédagogiques.

Les capacités bimodales de CodeBERT et son architecture Transformer en font un outil puissant pour diverses tâches liées à la programmation, tant dans l'industrie que dans le domaine de l'éducation. Par exemple, l'étude de (Yang et al., 2023), CodeBERT a été utilisé pour générer du code d'exploitation de haute qualité en combinant les connaissances extraites des bases de données de vulnérabilités avec les exigences syntaxiques et sémantiques du code. Cette approche permet d'améliorer les évaluations de sécurité en générant du code qui respecte à la fois les règles syntaxiques et les contraintes de sécurité. De plus, CodeBERT a été largement utilisé pour la recherche de code et la génération de documentation. Grâce à ses capacités bimodales, il peut identifier et extraire des informations pertinentes à partir de fichiers de code, facilitant ainsi la création automatique de documentation technique. Cette capacité est particulièrement précieuse dans les projets de développement à grande échelle où la

documentation manuelle peut être fastidieuse et sujette aux erreurs (E. Mashhadi & H. Hemmati, 2021).

8.6 UniXCoder

UniXcoder est un modèle pré-entraîné développé pour unifier le traitement du langage naturel (NLP) et la programmation (Guo et al., 2022). Ce modèle, qui s'inscrit dans la lignée des modèles comme CodeBERT (Z. Feng et al., 2020), CodeT5 (Y. Wang et al., 2021), GraphCodeBert (Guo et al., 2021), vise à combler l'écart entre les langages naturels et les langages de programmation. En se basant sur l'architecture Transformer (Vaswani et al., 2017), UniXcoder est conçu pour comprendre et générer du code tout en prenant en compte les aspects contextuels des descriptions en langage naturel associées. UnixCoder a été appliqué avec succès dans des tâches telles que la génération automatique de code, la traduction entre différents langages de programmation, et l'amélioration de la qualité des logiciels à travers la détection des bugs.

Le modèle est pré-entraîné sur des ensembles de données massifs qui incluent à la fois du texte et du code, ce qui lui permet d'apprendre à générer du code en fonction des instructions en langage naturel et à annoter du code de manière cohérente. Ce pré-entraînement est complété par un ajustement (fine-tuning) sur des tâches spécifiques. De plus, UniXCoder se distingue par sa capacité à traiter simultanément des données en langage naturel et en langage de programmation, ce qui en fait un modèle véritablement multimodal.

8.6.1 Architecture de UniXCoder

UniXcoder est un modèle pré-entraîné qui étend les concepts de BERT et CodeBERT à une large gamme de langages de programmation et d'outils de développement. Les particularités de son architecture incluent:

1) Modèle multimodal: UniXcoder est conçu pour traiter non seulement le code source mais aussi les commentaires et la documentation associés.
2) Pré-entrainement avec un corpus multilingue: il est pré-entraîné sur un corpus qui inclut une variété de langages de programmation et des ressources supplémentaires comme des commentaires et des descriptions de code.
3) L'architecture de UniXcoder est également basée sur l'encoder de BERT, avec des modifications pour traiter les aspects multimodaux du code source et des commentaires associés. En effet, UnixCoder est basé sur l'architecture Transformer, qui repose sur des mécanismes d'attention pour capturer les relations contextuelles entre les tokens d'entrée. Cependant, ce modèle va au-delà de la simple intégration des langages naturels et de programmation en intégrant des mécanismes d'attention croisée qui permettent au modèle de mieux comprendre les interactions entre le code et les descriptions associées (Guo et al., 2022).

8.6.2 Utilisation de UniXcoder dans le Domaine de l'Education

À ce jour, et à notre connaissance, aucune étude spécifique n'a encore appliqué UniXcoder directement à des tâches éducatives traditionnelles, telles que l'évaluation automatique, l'analyse des performances des étudiants ou la conception de systèmes de tutorat intelligents. Initialement conçu pour la compréhension et la génération de code source, UniXcoder se révèle particulièrement prometteur dans le domaine de l'enseignement de la programmation et des disciplines connexes. Néanmoins, la majeure partie des applications d'UniXcoder se concentrent sur des tâches d'ingénierie logicielle comme l'autocomplétion de code, la détection de bugs et la synthèse de code.

9 Conclusion

L'intégration des modèles de langage pré-entraînés tels que BERT, RoBERTa, CodeBERT et UniXcoder dans le domaine de l'éducation représente une avancée significative dans la manière dont l'enseignement et l'évaluation des compétences sont abordés. Ces modèles, reposant sur des architectures de réseaux de neurones complexes et des techniques de traitement du langage naturel, ont montré leur capacité à non seulement améliorer la précision des évaluations, mais également à fournir des retours pédagogiques plus pertinents et adaptés aux besoins des apprenants.

L'utilisation de ces technologies offre un potentiel immense pour révolutionner les environnements informatiques pour l'apprentissage humain, notamment par l'automatisation des tâches d'évaluation et la personnalisation des expériences éducatives. Par exemple, les capacités de BERT et RoBERTa à analyser des réponses complexes, ainsi que celles de CodeBERT et UniXcoder à automatiser l'évaluation de code, permettent d'enrichir considérablement les pratiques pédagogiques dans les disciplines STEM et au-delà. De plus, la précision et la flexibilité offertes par ces modèles facilitent l'identification des lacunes conceptuelles des étudiants. Cependant, malgré ces avancées, il demeure nécessaire de continuer à évaluer et à ajuster ces modèles pour répondre aux défis spécifiques de l'éducation.

SECTION Aspect Pratique

CHAPITRE IV : AMELIORER L'EVALUATION DES APPRENITSSAGES GRACE AU SURECHANTILLONNAGE

1 Introduction

Ce chapitre présente en détail la méthodologie mise en œuvre pour pallier au problème du déséquilibre des classes dans un contexte éducatif, ainsi que les résultats obtenus et leur interprétation. Cette étude s'inscrit dans le cadre d'un programme universitaire ambitieux, visant à optimiser l'apprentissage de l'algorithmique dans un EIAH. Plus précisément, elle vise à classer les apprenants et la réalisation de travaux préliminaires afin de garantir que ceux-ci disposent des compétences requises avant de se soumettre à des évaluations sommatives futures, ce qui pourrait potentiellement diminuer le taux d'échec. En s'appuyant sur des approches innovantes, cette première contribution vise non seulement à apporter des solutions concrètes, mais également à enrichir le corpus académique existant. De ce fait, elle participe à l'élaboration de stratégies éducatives mieux adaptées, s'inscrivant résolument dans une démarche d'amélioration continue.

Tout d'abord, l'importance du consentement éclairé des étudiants pour le partage de leurs notes dans le cadre de cette recherche ne peut être sous-estimée. Toutes les données utilisées ont été collectées avec le plein consentement des participants, en stricte conformité avec les directives éthiques de l'université concernant l'utilisation des données relatives aux étudiants. Cette précaution garantit non seulement la légitimité de notre recherche, mais aussi le respect de la vie privée des étudiants.

La motivation de cette étude découle d'une analyse approfondie des données collectées sur deux semestres auprès de 2176 étudiants de première année inscrits en sciences et technologie. Les résultats ont montré que seuls 518 étudiants (23,80%) ont réussi le module algorithmique, tandis que 1658 étudiants (76,2%) ont échoué. Ce taux d'échec alarmant souligne la nécessité d'interventions pédagogiques efficaces pour améliorer les performances des étudiants.

Pour aborder ce problème, nous avons utilisé quatre techniques d'échantillonnage différentes: SMOTE (Chawla et al. 2002), SMOTE Borderline, SMOTE-EN, ADASYN et une méthode nouvellement introduite nommée Equi-Fused-Data-based SMOTE (Chachoui et al. 2024). Ces techniques ont été choisies pour leur capacité à gérer les ensembles de données déséquilibrés, en augmentant artificiellement la proportion d'exemples de la classe minoritaire (les étudiants ayant réussi) afin de former des modèles de classification plus robustes.

Le schéma suivant (Figure IV.1) illustre notre approche de recherche, qui se déroule en plusieurs étapes clés: collecte des données, prétraitement des données, partitionnement des données, application des variantes de SMOTE et la méthode proposée Equi-Fused-Data pour équilibrer les données, et utilisation de modèles supervisés pour classifier les résultats.

Dans les sections qui suivent, nous décrirons en détail chacune de ces étapes, en expliquant les choix méthodologiques effectués, les résultats obtenus à chaque phase, et les implications de ces résultats pour l'amélioration de l'apprentissage de l'algorithmique. Notre discussion se concentrera également sur les défis rencontrés, les solutions adoptées et les perspectives futures de recherche dans ce domaine.

2 Méthodologie

La figure présentée ci-dessous (Figure IV.1) illustre la méthodologie utilisée pour aborder la problématique du faible taux de réussite dans le module algorithmique. Le processus commence par la collecte des données des étudiants, suivi d'un prétraitement minutieux pour nettoyer et préparer ces données en vue de leur analyse. Une fois les données prétraitées, elles sont scindées en deux ensembles distincts: un ensemble d'entraînement représentant 80 % des données et un ensemble de test représentant les 20 % restants. Cette étape est nécessaire pour

garantir que le modèle formé sur l'ensemble d'entraînement peut être validé de manière indépendante sur l'ensemble de test.

Ensuite, pour aborder le problème de déséquilibre des classes (où la majorité des étudiants ont échoué), nous avons appliqué quatre techniques d'échantillonnage différentes: SMOTE, SMOTE Borderline, SMOTE-ENN et ADASYN. Ces techniques visent à augmenter artificiellement le nombre d'exemples dans la classe minoritaire (les étudiants ayant réussi) afin d'améliorer la performance des modèles de classification. Les données équilibrées sont alors utilisées pour entraîner plusieurs modèles de classification supervisée, notamment SVM, la forêt aléatoire, kNN, l'arbre de décision, le naïve bayes, la régression logistique multinomiale, gradient boosting machines et adaBoost, en utilisant une validation croisée à k-fold (k=10) pour évaluer la performance des modèles. Les valeurs prédites par ces modèles sont ensuite comparées aux valeurs réelles pour évaluer l'efficacité de chaque méthode de suréchantillonnage séparément.

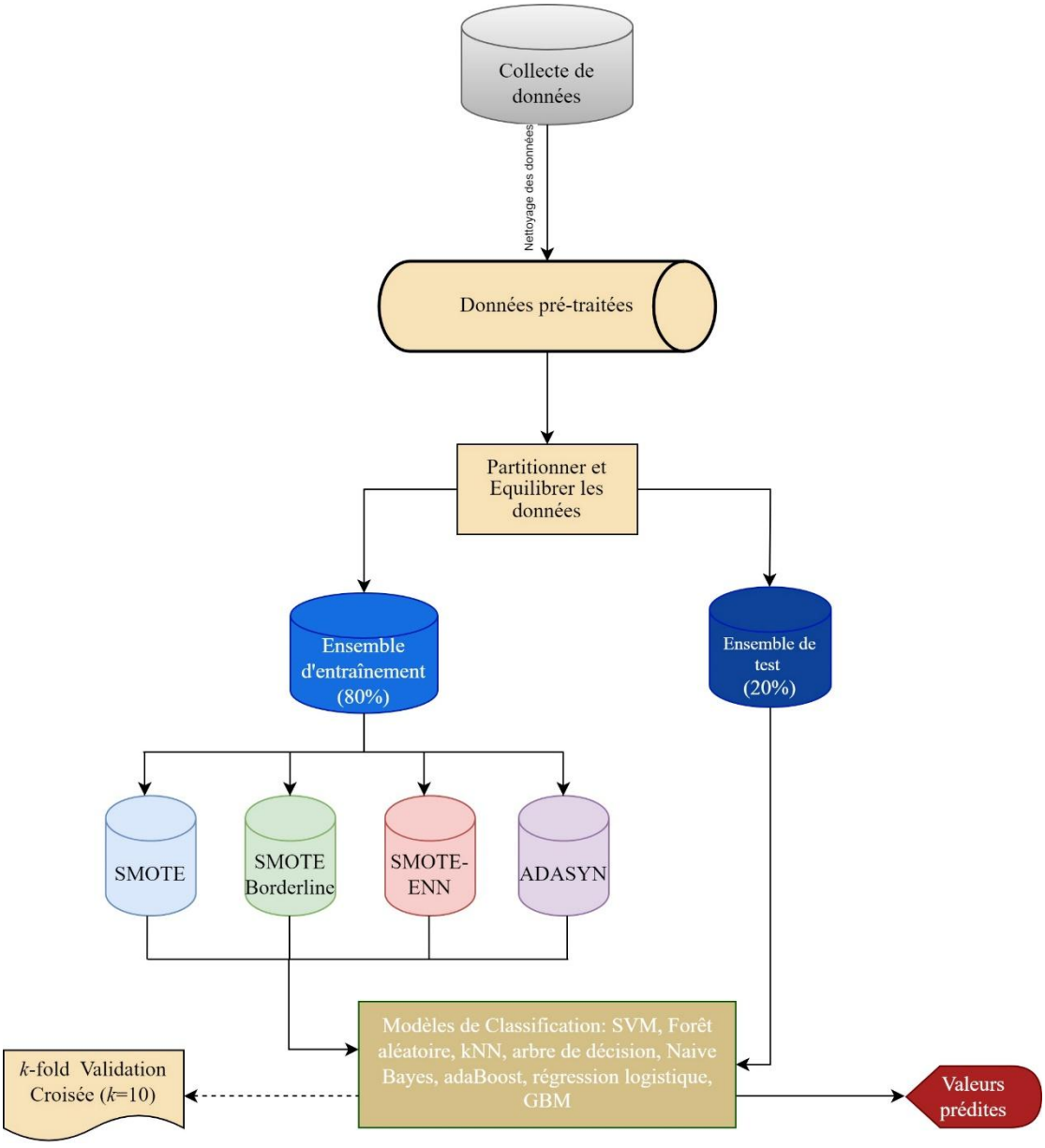

Figure IV.1 Les étapes de la méthodologie suivie

2.1 Collecte de données

Le jeu de données (dataset) comprend des évaluations, tant formatives que sommatives, des étudiants en sciences et technologies inscrits entre 2020 et 2022. L'élaboration de ce jeu de données a été réalisée dans le respect des directives relatives aux données des étudiants de l'Université d'Annaba.

Les étudiants ont été répartis aléatoirement en groupes pour les évaluations. Chaque semestre comprenait deux évaluations formatives et une évaluation sommative. Chaque évaluation a été soigneusement conçue pour répondre aux objectifs d'apprentissage spécifiques et aux prérequis. Les enseignants des cours d'Informatique 1 (INF1) et Informatique 2 (INF2) ont évalué les performances des étudiants de manière anonyme. Afin d'assurer la confidentialité et le respect de la vie privée des étudiants, toutes les évaluations ont été anonymisées avant la notation. Cela signifie que toutes les informations permettant d'identifier les étudiants, telles que les noms et les numéros d'étudiant, ont été retirées des copies. Chaque étudiant a reçu un code unique utilisé pour identifier son travail lors de la notation. Après anonymisation, les copies des étudiants ont été attribués aléatoirement à différents professeurs pour la notation, garantissant ainsi une évaluation équitable et objective du travail de chaque étudiant. En outre, le jeu de données contient des informations provenant de quatre principales caractéristiques collectées lors de diverses évaluations dans les cours (INF1) et (INF2).

Cette étude a utilisé un total de 2176 instances, avec les attributs listés dans le Tableau (Tableau 4.1). Etant donné que l'objectif est de prédire les performances des étudiants, nous avons été confrontés à un problème de classification multi-classes. Plutôt que d'opter pour une classification binaire restrictive (réussite/échec), nous avons choisi un système à trois catégories. Cette option reconnaît la gamme nuancée des performances des étudiants (échec, modéré, excellent) présente dans le jeu de données. Avec cette granularité plus fine, nous espérons représenter plus précisément les performances individuelles et fournir une base d'analyse plus approfondie. Ainsi, 36,12% des instances (786) sont étiquetées comme "Faible" (score de 0 à 6), représentant les étudiants en difficulté académique. 63,10 % des instances (1373) sont étiquetées "Modérée" (score de 7 à 13), indiquant que les étudiants répondent aux attentes de leur niveau scolaire, tandis que les 0,78 % restants des instances (17) sont étiquetées "Élevée" (score de 14 à 20), indiquant que les étudiants excellent dans les cours d'introduction à la programmation. Par conséquent, ce jeu de données est déséquilibré.

Tableau 4.1 Description du jeu de données

Attribut	Description	Valeur	Min	Max	Moy	Ecart-type
Grade1	Score moyen des travaux formatifs INF1 (de 0 à 20)	Numérique	0	19.50	10.72	5.39
Grade2	Score moyen des travaux formatifs INF2 (de 0 à 20)	Numérique	0	20	9.88	5.82
Exam1	Score de l'évaluation sommative (INF1)	Numérique	0	18	4.70	3.46
Exam2	Score de l'évaluation sommative (INF2)	Numérique	0	18.50	2.57	2.66

Performance	Rendement scolaire	Ordinale: faible, moyen, bon	-	-	-	-

Les attributs Grade1 et Grade2 représentent la moyenne (intervalle de 0 à 20) des différents éléments du cours, tels que les devoirs formatifs et les projets dans (INF1) et (INF2), respectivement.

De plus, les caractéristiques comportementales liées à l'engagement des étudiants ont été évaluées en utilisant une combinaison de participation en classe, qui fait référence à la fréquence et à la qualité de la participation aux activités et aux projets de groupe. Les registres de présence servent de mesure d'engagement. De plus, le travail de projet se réfère à la qualité et à l'achèvement des projets individuels et de groupe. Ces caractéristiques ont également été évaluées sur une échelle de 0 à 20 et intégrées dans leur score de devoir formatif. Exam 1 et Exam 2 sont les scores (intervalle de 0 à 20) des deux évaluations sommatives qui couvrent des sujets tels que le codage binaire et les bases de la programmation pour Exam 1, et des concepts avancés tels que les chaînes de caractères, les boucles et les tableaux pour Exam 2. Ainsi, ce jeu de données représente une source riche d'informations permettant d'analyser les performances académiques des étudiants, en tenant compte de divers facteurs formatifs et sommatives ainsi que des caractéristiques comportementales liées à leur engagement en classe.

2.2 Prétraitement des données

Afin d'assurer l'intégrité de l'analyse proposée, une procédure exhaustive de nettoyage des données et de sélection des caractéristiques a été effectuée. Cette procédure comprenait les étapes suivantes, chacune étant soigneusement mise en œuvre pour garantir la qualité et la fiabilité des données, conditions préalables essentielles pour une analyse rigoureuse.

Formatage des données

La variable cible a été convertie de son format initial, qui contenait des valeurs catégorielles, en une représentation numérique à l'aide de l'encodage des étiquettes (label encoding). Cette transformation est essentielle pour permettre aux algorithmes de traitement des données et de modélisation de manipuler efficacement les informations, car beaucoup de ces algorithmes exigent des entrées numériques.

Identification et suppression des doublons

Les entrées dupliquées ont été identifiées et supprimées. Cette étape a abouti à la suppression de 286 instances, réduisant ainsi le jeu de données à 1890 instances. La suppression des doublons est essentielle pour éviter que les résultats de l'analyse ne soient biaisés par la présence de données redondantes, ce qui pourrait fausser les conclusions tirées de l'étude.

Correction des fautes de frappe et des inexactitudes

Toutes les fautes de frappe ou inexactitudes, telles que les erreurs d'orthographe dans le jeu de données, ont été corrigées afin de maintenir l'intégrité des données. Ces corrections sont indispensables pour garantir que les analyses subséquentes ne soient pas affectées par des erreurs de données qui pourraient induire en erreur les modèles prédictifs et les résultats analytiques.

Traitement des valeurs aberrantes

La méthode de l'intervalle interquartile (IQR) a été utilisée pour identifier les valeurs aberrantes potentielles. Les premier (Q1) et troisième (Q3) quartiles ont été calculés pour chaque colonne, et les points de données situés en dehors de l'intervalle Q1 - 1,5 * IQR et Q3 + 1,5 * IQR ont été étiquetés comme valeurs aberrantes potentielles. Cependant, aucune valeur aberrante n'a été identifiée dans cette analyse. L'identification et le traitement des valeurs aberrantes sont cruciaux pour prévenir les effets disproportionnés de ces valeurs extrêmes sur les analyses statistiques et les modèles prédictifs.

Traitement des valeurs manquantes

Il n'y avait aucune valeur manquante dans le jeu de données. L'absence de valeurs manquantes simplifie le processus de nettoyage des données et garantit que chaque instance contient des informations complètes pour une analyse précise et exhaustive.

Randomisation des données

Afin de réduire le biais d'échantillonnage, le jeu de données a été généré de manière aléatoire. Cette étape vise à garantir que l'analyse subséquente soit réalisée sur un échantillon représentatif, exempt de biais systémique. La randomisation est essentielle pour améliorer la généralisation des résultats de l'analyse à l'ensemble de la population étudiée. En appliquant méticuleusement ces étapes, nous avons pu préparer un jeu de données propre et fiable, indispensable pour une analyse rigoureuse et des conclusions valides.

2.3 Sélection des Caractéristiques

Pour la sélection des caractéristiques, nous avons utilisé trois techniques distinctes et comparé leurs résultats.

Premièrement, nous avons utilisé l'algorithme Information Gain Attribute Evaluator en conjonction avec une méthode de classement. Cet algorithme évalue la valeur d'un attribut en mesurant son gain d'information par rapport à la classe. La méthode de classement nous permet d'ordonner les attributs sur la base de leurs évaluations individuelles. Les attributs significatifs identifiés et leurs notes d'évaluation correspondantes sont les suivants : Grade1 (0,57) en premier, Exam1 (0,50) en deuxième, Grade2 (0,49) en troisième et enfin Exam2 (0,35). Les notes 1 et 2 représentent la moyenne des caractéristiques comportementales des élèves sur deux semestres, intégrant des facteurs tels que la motivation, l'engagement, les évaluations formatives et le travail d'équipe. Chaque caractéristique a été évaluée individuellement. Exam1 et Exam2 reflètent les notes finales des étudiants lors des évaluations sommatives.

Deuxièmement, nous avons utilisé la méthode Classifier Subset Evaluator avec l'algorithme Bagging et la technique Best First. Cette méthode évalue les sous-ensembles à l'aide de données d'entraînement et utilise un classificateur pour estimer le "mérite" d'un ensemble d'attributs. L'algorithme BestFirst explore les sous-ensembles d'attributs à l'aide d'une combinaison d'escalade de colline et de retour en arrière. Dans notre étude, il a été utilisé avec un ensemble vide et a effectué une recherche vers l'avant. Les mêmes attributs (Grade1, Exam1, Grade2 et Exam2) ont été sélectionnés dans ce meilleur sous-ensemble, qui a atteint une valeur de 0,94.

Troisièmement, nous avons utilisé l'évaluateur d'attributs par corrélation avec une approche par ratissage. Cet algorithme évalue la valeur des attributs en mesurant la corrélation (Pearson) entre l'attribut et la classe. Les résultats de corrélation suivants ont été obtenus : Exam2 (0,25), Exam1 (0,25), Grade1 (0,22) et Grade2 (0,19). Sur la base de ces résultats, nous avons sélectionné ces quatre attributs pour notre étude.

2.4 Partitionner et Equilibrer les Données

Cette section examine la méthodologie employée pour diviser et équilibrer les données pour les expériences.

Séparation Entraînement/Test

Pour éviter les biais et le bruit provenant d'instances chevauchantes, l'ensemble de données a été rééchantillonné en deux ensembles : l'entraînement (80 %, soit 1512 instances) et le test (20 %, soit 378 instances). Cette répartition courante de 80-20 nous a permis de recueillir suffisamment de données pour l'entraînement tout en laissant une portion significative pour une évaluation impartiale des performances du modèle. L'ensemble de test est resté constant tout au long des essais expérimentaux afin de garantir la cohérence et la fiabilité des résultats obtenus.

En utilisant cette approche, nous avons assuré que le modèle pourrait être évalué de manière rigoureuse sans être influencé par les mêmes données utilisées pour son entraînement. Cela permet de tester la capacité de généralisation du modèle sur des données inédites afin d'évaluer sa performance réelle dans des situations du monde réel.

2.5 Rééchantillonnage

Des variations de la technique SMOTE (Chawla et al., 2002) ont été utilisées pour équilibrer les données. Le principe de fonctionnement détaillé de SMOTE est présenté dans le chapitre 3 section 7.1. La figure IV.2 illustre les différentes étapes de l'algorithme SMOTE appliqué dans notre méthodologie.

Pour note application, nous avons défini les paramètres suivants : k = 5 plus proche voisins et un montant de suréchantillonnage N visant à atteindre une distribution de classes approximativement 1:1.

Pour aborder le déséquilibre de classe du jeu de données, nous avons appliqué une variation de la technique SMOTE. Bien que SMOTE soit efficace pour suréchantillonner la classe minoritaire, cette technique peut également générer du bruit. Par conséquent, nous avons étudié SMOTE Borderline (voir chapitre 3 section 7.3.1), SMOTE-ENN (voir chapitre 3 section 7.3.2) et ADASYN (voir chapitre 3 section 7.3.3). Premièrement, SMOTE Borderline priorise le suréchantillonnage près de la frontière de décision, ce qui peut améliorer les performances de classification. Deuxièmement, SMOTE-ENN combine le suréchantillonnage et le nettoyage par le plus proche voisin édité pour éliminer les instances de la classe majoritaire bruyantes proches de la classe minoritaire. Enfin, ADASYN effectue un suréchantillonnage adaptatif des instances minoritaires en fonction de leur difficulté d'apprentissage, améliorant potentiellement l'efficacité du suréchantillonnage.

Les résultats obtenus sont les suivants :

- SMOTE a atteint une distribution presque équilibrée (3273 instances, 1091 par classe).
- SMOTE Borderline a donné des résultats identiques à ceux de SMOTE.
- SMOTE-ENN a abouti à 3061 instances (1091 "élevé", 997 "faible", 973 "modéré").
- ADASYN a généré 3258 instances (1091 pour les classes "modéré" et "élevé", 1076 pour "faible").

En appliquant ces différentes méthodes de rééchantillonnage, nous avons pu analyser leur impact sur la qualité de l'équilibre des données et la performance des modèles de classification.

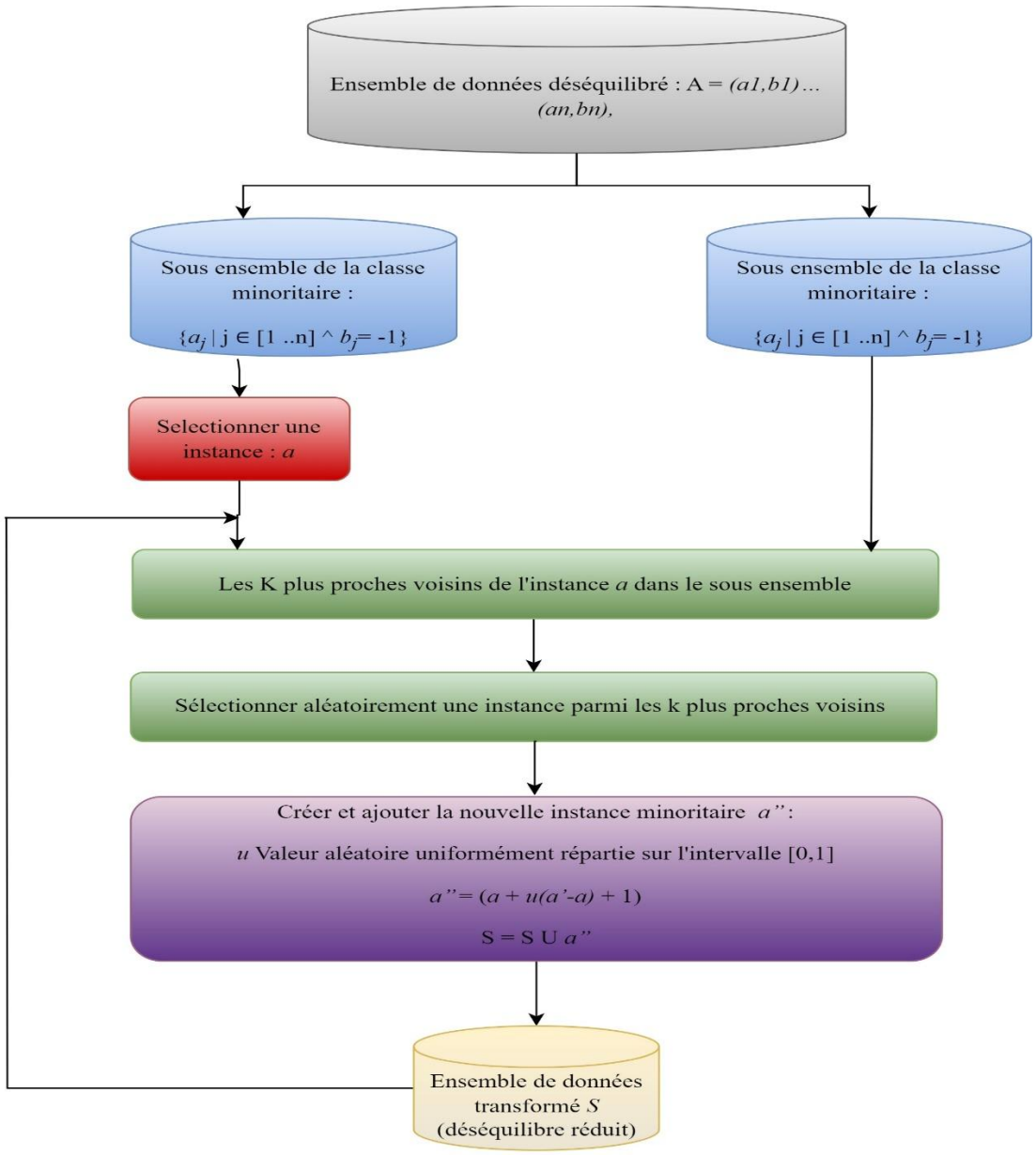

Figure IV.2 Déroulement de la technique SMOTE

3 Modèle de Coopération de Données Synthétiques pour l'Amélioration de l'Apprentissage Multi-classes

Afin de remédier au défi critique des ensembles de données déséquilibrés dans l'apprentissage multi-classe, nous introduisons la méthode Equi-Fused-Data-based SMOTE (Chachoui et al. 2024). Cette technique fusionne les avantages du suréchantillonnage et de l'apprentissage ensembliste pour offrir une solution efficace. Les ensembles de données déséquilibrés posent en effet un problème majeur pour les algorithmes de classification traditionnels, rendant les métriques d'évaluation usuelles, telles que l'exactitude, trompeuses et inadéquates. Equi-Fused-Data-based SMOTE propose une approche basée sur la priorisation des métriques alternatives telles que l'AUC (Area Under the Curve) et le F1-score, garantissant ainsi une évaluation plus précise et équilibrée des modèles. Cette méthode vise à améliorer de manière significative les performances des algorithmes de classification.

3.1 Composants du Modèle

Le modèle SMOTE basé sur des données équilibrées et fusionnées se distingue par une structure en deux principaux composants, comme illustré dans la figure suivante (Figure IV.3).

Tout d'abord, ce modèle intègre quatre méthodes de suréchantillonnage renommées: SMOTE, SMOTE Borderline, SMOTE-ENN et ADASYN, chacune apportant ses avantages spécifiques. SMOTE est particulièrement efficace pour traiter le déséquilibre général des données en créant des échantillons synthétiques équilibrés. La méthode SMOTE Borderline, quant à elle, cible spécifiquement les échantillons situés à la frontière entre les classes majoritaire et minoritaire, renforçant ainsi la robustesse du modèle face aux cas difficiles. SMOTE-ENN, qui combine SMOTE avec l'élimination des plus proches voisins, affine davantage la qualité des données générées en éliminant les bruits et les anomalies. Enfin, ADASYN adapte son approche en privilégiant l'apprentissage des échantillons minoritaires les plus complexes à classifier, augmentant ainsi la diversité et la représentativité des données suréchantillonnées. Par ailleurs, l'application conjointe de ces techniques, sans redondances, permet de capturer une variété étendue d'échantillons potentiels de la classe minoritaire, enrichissant considérablement le jeu de données d'entraînement.

La seconde étape de notre modèle repose sur l'utilisation du modèle balanced bagging, une approche d'ensemble spécialement conçue pour les jeux de données déséquilibrés. Balanced Bagging traite le déséquilibre des classes en générant plusieurs sous-modèles à partir de sous-ensembles équilibrés de données suréchantillonnées, comme le souligne l'étude de (Barros et al. 2019). D'ailleurs, l'apprentissage ensembliste, qui combine les prédictions de plusieurs modèles, surpasse souvent les performances des modèles individuels en termes de précision et de robustesse. Cette stratégie favorise une diversité accrue au sein de l'ensemble tout en minimisant l'influence prédominante de la classe majoritaire. La Figure IV.3 illustre de manière détaillée le fonctionnement de ce système, en exposant la séquence des différentes phases de traitement et d'apprentissage. Ainsi, le modèle SMOTE basé sur des données équilibrées et fusionnées, grâce à l'utilisation synergique de multiples techniques de suréchantillonnage et du modèle balanced bagging, offre une solution efficace pour adresser les défis posés par les jeux de données déséquilibrés. Cette approche améliore non seulement la qualité des données d'entraînement mais aussi la performance générale des modèles prédictifs, établissant ainsi une nouvelle norme dans le domaine de la modélisation des données déséquilibrées.

3.2 Fondement Théorique

En tirant parti de la synergie entre le suréchantillonnage et l'apprentissage ensembliste, la méthode Equi-Fused-Data-based SMOTE exploite l'intégration de divers ensembles de données suréchantillonnés au sein du modèle d'ensemble balanced bagging. Cette approche présente plusieurs avantages notables:

- **Représentation améliorée** : Le suréchantillonnage permet d'optimiser la représentation des classes minoritaires, favorisant ainsi un apprentissage plus efficace de leurs caractéristiques par l'ensemble.
- **Réduction des biais** : L'utilisation de multiples ensembles de données suréchantillonnés minimise le risque de biais induits par les techniques individuelles.
- **Meilleure généralisabilité** : La diversité au sein de l'ensemble favorise l'élaboration de modèles plus robustes et généralisables, réduisant ainsi le risque de surapprentissage à un seul ensemble de données suréchantillonnées.

La présente approche s'appuie sur un fondement théorique solide, étayé par des études ayant démontré l'efficacité du suréchantillonnage et de l'apprentissage ensembliste dans le cadre de l'apprentissage déséquilibré.

En effet, les travaux de (Chawla et al. 2002; Fernandez et al. 2018) mettent en lumière les avantages de l'algorithme SMOTE pour traiter les ensembles de données déséquilibrés. Ces études montrent comment SMOTE améliore la distribution des classes minoritaires, permettant ainsi une classification plus équilibrée et précise. De même, (Barros et al. 2019) soulignent les bénéfices des méthodes ensemblistes telles que le balanced bagging. Leur recherche démontre que le balanced bagging surpasse des techniques telles que les perceptrons multicouches (MLP) et les arbres de décision en termes de performance sur les métriques G-mean et Unbalanced Accuracy Ratio (UAR). Cela permet d'éviter le paradoxe de la précision souvent rencontré dans le contexte des ensembles de données éducatifs, où une métrique globale de précision peut masquer de faibles performances sur les classes minoritaires. Enfin, (Pristyanto et al. 2021) ont démontré que la combinaison de techniques de suréchantillonnage telles qu'ADASYN avec l'apprentissage ensembliste peut efficacement résoudre les problèmes de déséquilibre, en améliorant à la fois la précision et le rappel des modèles. Ainsi, la méthode Equi-Fused-Data-based SMOTE s'inspire de ces avancées pour proposer une solution intégrée qui combine suréchantillonnage et apprentissage ensembliste, visant à améliorer les performances des algorithmes de classification dans des contextes de données déséquilibrées.

Le modèle SMOTE basé sur des données équilibrées et fusionnées a été évalué empiriquement sur des ensembles de données multi-classes déséquilibrés en comparant ses performances aux techniques individuelles de suréchantillonnage et à d'autres méthodes ensemblistes. L'efficacité du modèle a été évaluée à l'aide de métriques telles que l'AUC-ROC, le F1-score et la précision globale. En intégrant ces bases théoriques, le modèle SMOTE basé sur des données équilibrées et fusionnées démontre qu'il s'agit d'une approche théoriquement solide pour surmonter les défis de l'apprentissage multi-classes déséquilibré.

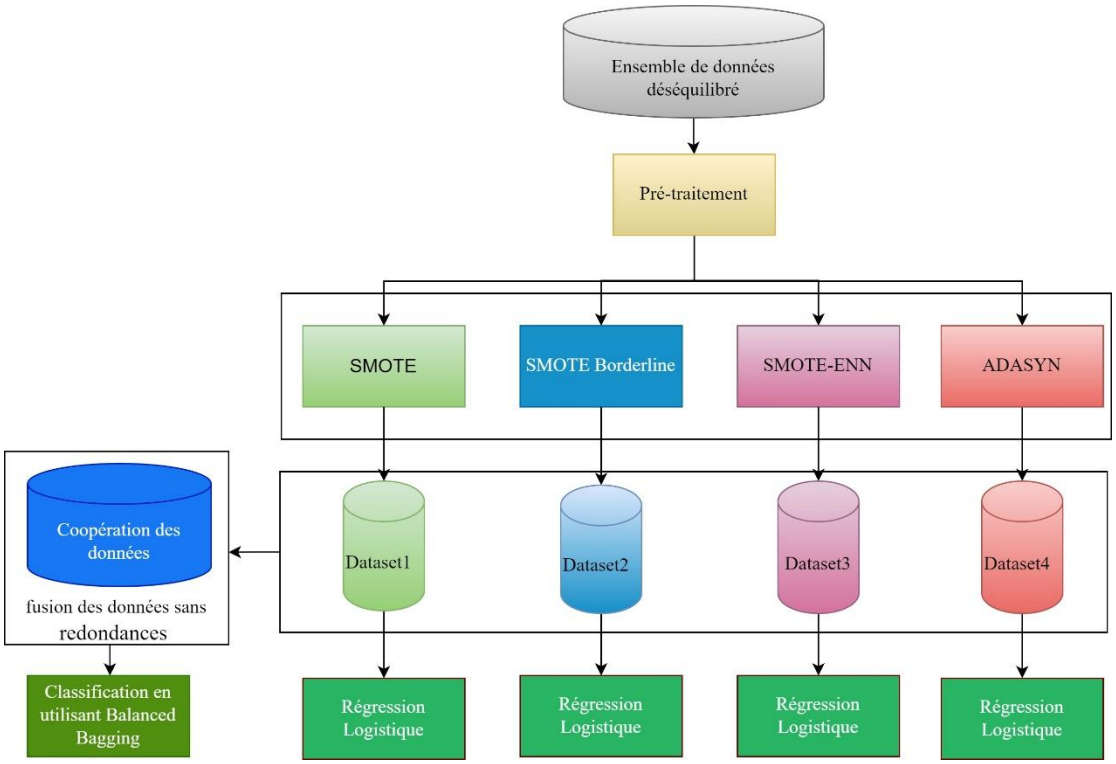

Figure IV.3 Schéma du déroulement de la méthode Equi-fusion des données

3.3 Formulation du Problème

Dans le contexte de l'apprentissage multi-classe déséquilibré, le modèle SMOTE basé sur des données équilibrées et fusionnées aborde le déséquilibre des classes en combinant les

avantages de la suréchantillonnage et de l'apprentissage ensembliste. Pour clarifier le fondement théorique de cette approche, le problème doit être défini mathématiquement, en considérant la combinaison des ensembles de données et l'exclusion des doublons. Soit Di (for i = 1, 2, …, n) l'ensemble de données obtenu à partir de la ième technique de suréchantillonnage (par exemple, SMOTE, SMOTE Borderline, SMOTE-ENN et ADASYN). Chaque D_i contient des instances appartenant à plusieurs classes (c classes) avec une distribution déséquilibrée. Ici, c désigne le nombre de classes, et n est le nombre total de techniques de suréchantillonnage utilisées. Chaque instance x_j^i dans D_i est un vecteur de caractéristiques de dimension d associé à une étiquette de classe $y_j^i \in \{1, 2, \ldots, c\}$. Chaque technique de suréchantillonnage génère un ensemble de données D_i. Soit fi (x, y) la fonction de suréchantillonnage utilisée par la ième technique, où x est un point de données et y est son étiquette de classe.

$$D_i = f_i(x_j, y_j) | (x_j, y_j) \in D_{original}, j = 1,2, \ldots, |D_{original}|$$

où $D_{original}$ est l'ensemble de données déséquilibré original.

La combinaison de ces ensembles de données implique leur fusion en un ensemble unifié, noté $D_{combiné}$. Cette fusion vise à combiner les instances de toutes les classes pour assurer une représentation plus équilibrée de l'ensemble des classes.

$$D_{combiné} = \bigcup_{i=1}^{m} D_i$$

Cependant, comme il peut y avoir des chevauchements entre les ensembles de données, l'ensemble de données combiné résultant peut contenir des instances dupliquées. Pour éliminer les redondances et maintenir l'intégrité des données, les instances redondantes (doublons) présentes dans $D_{combiné}$ sont supprimées. L'ensemble de données unique résultant est noté D_{unique} . Ce processus garantit que chaque instance dans l'ensemble de données est unique et conserve la diversité nécessaire pour un apprentissage efficace.

$$D_{\text{unique}} = D_{\text{combiné}} \setminus \text{Redondance}$$

De plus, pour tirer parti de l'apprentissage ensembliste, le modèle balanced bagging est également utilisé avec un classificateur de base tel que la régression logistique. Dans cette technique d'ensemble, plusieurs sous-modèles sont créés, chacun étant entraîné sur un sous-ensemble équilibré des données obtenues par suréchantillonnage. Soit (B={B1, B2,...,Bm}) l'ensemble des m classificateurs de base, où chaque B_i est entraîné sur un sous-ensemble équilibré de D_{unique}. Ce processus d'équilibrage assure une représentation équitable de toutes les classes pendant l'entraînement.

$$B_i = EntraînerClassifieur(D_{sousensemble}, paramètres)$$

où $D_{sousensemble}$ est un sous-ensemble équilibré de D_{unique} et paramètres désignent les hyperparamètres du classificateur (la régression logistique dans ce cas).

3.4 Paramètres d'Implémentation

Pour cet ensemble de données, quatre ensembles de données équilibrés ont été fusionnés, produisant 8155 instances. Pour éliminer les doublons, l'ensemble de données combiné a été traité, aboutissant à une taille finale de 8136 instances.

Ensuite, chacun de ces ensembles de données a été classifié en utilisant la régression logistique. Le modèle ensembliste balanced bagging (Barros et al., 2019) a ensuite été implémenté comme suit :

- Un classifieur de forêt aléatoire : l'ensemble comprenait 20 estimateurs de base (arbres). Cette décision a été prise pour atteindre un équilibre entre la complexité du modèle et l'efficacité computationnelle. La stratégie d'échantillonnage a été définie sur "not majority", ce qui signifie que la distribution des classes a été équilibrée avec la technique d'échantillonnage pour garantir l'égalité pendant l'entraînement. De plus, l'échantillonnage a été effectué sans remplacement pour assurer l'intégrité de l'ensemble de données. La graine aléatoire pour la reproductibilité a été fixée à 42.
- Un classifieur de régression logistique multinomiale : le modèle de base est configuré avec « multi_class=multinomial » pour gérer la régression multinomiale avec un max d'itération équivaut à 1000 pour garantir la convergence. Le classifieur utilise 20 estimateurs,ce qui offre un bon compromis entre performance et temps de calcul. La stratégie de rééchantillonnage est définie par « sampling_strategy='not majority », rééchantillonnant ainsi toutes les classes sauf la majorité pour traiter le déséquilibre des classes. De plus, le rééchantillonnage est effectué avec remplacement (`replacement=True`) pour assurer la diversité des ensembles d'entraînement, et une graine aléatoire (`random_state=42`) est fixée pour la reproductibilité des résultats.
- Un classifieur Stacking avec la forêt aléatoire et la régression logistique multinomiale : le modèle de forêt aléatoire est configuré avec nombre d'estimateurs=100 et random_state=42, tandis que la régression logistique est configurée avec multi_class='multinomial', solver='lbfgs', et max_iter=1000. Les classifieurs de bagging équilibrés sont configurés avec un nombre d'estimateurs=20, la stratégie de rééchantillonnage est fixée à 'not majority', et replacement=True pour assurer une diversité d'ensembles d'entraînement, avec une graine aléatoire (random_state=42) pour la reproductibilité. Un classifieur Stacking combine ces classifieurs avec la régression logistique comme méta-modèle, évalué par une validation croisée stratifiée à 10 plis pour garantir la robustesse des résultats.

3.4.1 Entraînement et Ajustement des Hyperparamètres

Les données ont été classifiées en utilisant une variété d'algorithmes d'apprentissage supervisé, notamment la SVM, la Forêt Aléatoire, l'Arbre de Décision, le kNN, le naive bayes, adaBoost, régression logistique et le modèle gradient boosting machines. Le Tableau 4.2 fournit des informations sur les algorithmes et le réglage des paramètres.

Tableau 4.2 La configuration des algorithmes d'apprentissage

Modèle	Paramètres
SVM	Noyau = RBF (Fonction de base radiale), Probabilité= True
Forêt aléatoire	Nombre d'estimateurs: 100, Critère: Gini, Profondeur maximum: Aucune, échantillon minimum par feuille: 1, Echantillons Bootstrap: True
Arbre de décision	Critère: Gini, Séparateur: meilleur, Profondeur maximum: Aucune, échantillon minimum par split: 2, échantillon minimum par feuille: 1
kNN	K = 5, distance métrique : Distance euclidienne
Naive Bayes	Naive Bayes gaussien

adaBoost	N_estimateurs = 50, Learning rate = 1.0, Estimateur de base = arbre de décision (max_profondeur = 1)
Gradient Boosting Machines	N_estimateurs = 100, Learning rate = 1.0, max_profondeur = 1, random state = 42
Régression logistique	Multinomiale, Solveur: lbfgs, Max Itérations: 1000, Random State: 42

3.4.2 Validation Croisée

Pour garantir des résultats cohérents et réduire le risque de surapprentissage, une validation croisée en 10 volets a été appliquée à tous les algorithmes utilisés dans cette étude. La validation croisée en 10 volets est une méthode bien établie dans le domaine de l'apprentissage automatique pour évaluer la performance des modèles de manière impartiale et fiable.

Concrètement, l'ensemble de données initial a été divisé en dix sous-ensembles de taille approximativement égale. Lors de chaque itération de la validation croisée, neuf de ces sous-ensembles ont été utilisés pour entraîner le modèle, tandis que le sous-ensemble restant a servi de jeu de test. Ce processus a été répété dix fois, chaque sous-ensemble étant utilisé une fois comme jeu de test. Cette approche permet de s'assurer que chaque point de données contribue à la fois à l'entraînement et à la validation, ce qui réduit le risque que les résultats soient biaisés par une partition particulière des données.

De même, pour le modèle SMOTE basé sur des données équilibrées et fusionnées, une validation croisée en 10 volets a été mise en œuvre sur l'ensemble de données synthétique combiné. Cet ensemble de données synthétique a été généré en appliquant des techniques de suréchantillonnage pour équilibrer les classes avant la fusion des données. La même approche de division en dix sous-ensembles a été utilisée, avec neuf sous-ensembles dédiés à l'entraînement et un sous-ensemble utilisé pour les tests à chaque itération. Ainsi, chaque itération du processus a permis d'évaluer la capacité du modèle à généraliser sur des données non vues, tout en minimisant les risques de surapprentissage.

L'utilisation cohérente du même ensemble de test pour l'évaluation finale a permis de comparer directement les performances des différents algorithmes et modèles dans des conditions équivalentes. Cette méthodologie a non seulement renforcé la validité des conclusions mais a également offert une vue d'ensemble précise des forces et des limites de chaque approche. En appliquant une validation croisée en 10 volets, nous avons pu obtenir des estimations fiables de la performance des modèles, assurant ainsi que les résultats obtenus sont représentatifs des véritables capacités des algorithmes en contexte réel.

3.5 Évaluation du Modèle

Les performances de chaque algorithme ont été évaluées en utilisant une gamme de métriques d'évaluation couramment employées dans les tâches de classification : taux de faux positifs (FPR), Exactitude (Accuracy), Précision, Rappel, F1-score et AUC-ROC (les définitions sont présentées dans le chapitre 3 section 4). Pour rappel,

- Taux de faux positifs (FPR) : proportion d'instances positives incorrectement prédites par rapport à toutes les instances négatives dans toutes les classes :

$$FPR = \frac{FP}{(FP + TN)}$$

où FP est le nombre de faux positifs (instances incorrectement prédites comme positives) et TN sont les vrais négatifs (instances correctement prédites comme négatives).

- AUC-ROC : évalue la capacité du modèle à distinguer entre différentes classes. Nous avons utilisé une combinaison de micro-AUC et de pondération pour un problème de classification multi-classes. L'approche micro-AUC traite toutes les classes de manière égale, indépendamment de leur distribution dans les données. Elle considère tous les vrais positifs et négatifs de toutes les classes pour générer une seule courbe ROC et son AUC correspondante. La méthode de pondération AUC calcule d'abord les valeurs AUC individuelles pour chaque classe en utilisant leurs courbes ROC respectives. Ces AUC individuelles sont ensuite combinées en une moyenne pondérée, avec des poids assignés en fonction de la fréquence relative de chaque classe dans les données d'entraînement. Le résultat prend en compte la prévalence de chaque classe dans les données pour déterminer son importance. Nous avons combiné les performances du modèle sur toutes les classes, en tenant compte à la fois de leur pouvoir discriminant individuel (via la micro-moyenne) et de leur importance relative dans les données (via la moyenne pondérée), pour obtenir une compréhension plus nuancée des performances du modèle dans un contexte de classification multi-classe.

Dans la prochaine section, nous allons détailler de manière approfondie les résultats obtenus et examiner les contributions théoriques ainsi que les apports pratiques dans le domaine de l'éducation. Nous allons d'abord présenter une analyse des performances des différents algorithmes d'apprentissage supervisé utilisés dans cette étude, en mettant en lumière les métriques clés telles que la précision, le rappel, et la F-mesure. Ensuite, nous discuterons des implications théoriques des résultats obtenus, notamment en ce qui concerne les approches de traitement des données déséquilibrées et l'optimisation des modèles prédictifs. Par ailleurs, nous nous pencherons sur les limites de la méthode proposée. Enfin, nous comparerons les différents résultats obtenus à partir des diverses méthodes de suréchantillonnage employées. Nous explorerons en détail les avantages et les inconvénients de chaque technique, en mettant en évidence laquelle s'avère la plus efficace dans des contextes spécifiques.

4 Les résultats

4.1 Ensemble de données déséquilibrées

Nous avons évalué la performance de divers modèles d'apprentissage automatique sur un ensemble de données déséquilibrées. Les résultats présentés dans le tableau 4.3 montrent que certains algorithmes se démarquent par leur capacité à discriminer les classes malgré le déséquilibre.

La machine à vecteurs de support (SVM) a atteint une exactitude de 91,79 %, avec une précision de 91,85 %, un rappel de 91,80 %, un score F1 de 91,55 % et une aire sous la courbe (AUC) de 98,19%. Ces résultats indiquent une excellente performance globale. En effet, l'AUC particulièrement élevée suggère que le SVM est très efficace pour distinguer entre les classes, même dans des conditions de déséquilibre de données, ce qui en fait un modèle robuste pour ce type de tâche. Par ailleurs, le modèle de régression logistique multinomiale a surpassé légèrement le SVM avec une exactitude de 92,59%, une précision de 92,60 %, un rappel de 92,59 %, un score F1 de 92,50 % et une AUC de 98,42 %. Ces chiffres démontrent une performance légèrement meilleure que le SVM, surtout en termes d'AUC, ce qui renforce la

capacité du modèle à gérer les classes déséquilibrées en offrant une séparation nette entre les classes positives et négatives.

En revanche, le modèle de forêt aléatoire a présenté une exactitude de 91,70 %, avec une précision de 91,75 %, un rappel de 91,79 %, un score F1 de 91,74% et une AUC de 97,24 %. Bien que ces résultats soient très bons, ils restent légèrement inférieurs à ceux d'autres modèles tels que : SVM et régression logistique. Cependant, le modèle de forêt aléatoire conserve un avantage en termes d'interprétabilité et de capacité à gérer les variabilités des données grâce à son ensemble d'arbres de décision. L'algorithme adaBoost, quant à lui, a montré des performances modestes avec une exactitude de 77,24 %, une précision de 82,02 %, un rappel de 77,20 %, un score F1 de 77,49 % et une AUC de 93,10 %. Ces résultats montrent que, même si adaBoost améliore la précision, il a du mal à maintenir un rappel élevé et un bon F1 score sur cet ensemble de données déséquilibrées, ce qui limite son utilité dans ce contexte.

Le modèle d'arbre de décision a atteint une exactitude de 89,15 %, avec une précision de 89,12 %, un rappel de 89,15 %, un score F1 de 89,13 % et une AUC de 87,06 %. Malgré ces performances respectables, elles sont toutefois inférieures à celles des modèles précédemment mentionnés, indiquant que l'arbre de décision seul peut avoir des difficultés à capturer les complexités des données déséquilibrées sans un cadre d'ensemble comme la forêt aléatoire.

Par ailleurs, le modèle des k-plus proches voisins (kNN) a démontré des performances élevées avec une exactitude de 92,85 %, une précision de 92,94 %, un rappel de 92,85 %, un score F1 de 92,84 % et une AUC de 96,38 %. Ces résultats démontrent une très bonne performance globale, indiquant que kNN est capable de gérer efficacement les données déséquilibrées, probablement grâce à sa nature non paramétrique et sa capacité à bien capturer les distributions locales des données. De même, le modèle Naive Bayes a obtenu une exactitude de 92,06 %, avec une précision de 92,08 %, un rappel de 92,05 %, un score F1 de 92,02 % et une AUC de 90,17 %. Malgré une AUC plus faible que certains des meilleurs modèles, le modèle reste compétitif grâce à sa simplicité et sa rapidité, même s'il peut faire des hypothèses simplificatrices sur l'indépendance des caractéristiques qui ne sont pas toujours valides. Dans la même lignée, le modèle Gradient Boosting Machine (GBM) a montré une exactitude de 90,47%, avec une précision de 89,00 %, un rappel de 90,47 %, un score F1 de 89,62 % et une AUC de 93,66 %. Bien que performant, GBM n'atteint pas les sommets des SVM ou de la régression logistique. Cependant, cet algorithme reste un choix solide grâce à sa capacité à améliorer progressivement ses performances par le biais de combinaisons d'apprentissage faibles. Le tableau 4.3 ci-dessous fournit plus de détails sur les résultats.

Tableau 4.3 Comparaison des Performances des modèles d'apprentissage automatique sur l'ensemble de données déséquilibrées

Modèle	Exactitude	FPR	Précision	Rappel	F1-score	AUC
SVM	91.79%	6.32%	91.85%	91.80%	91.55%	98.19%
Régression Logistique	92.59%	6.54%	92.60%	92.59%	92.50%	98.42%
Forêt aléatoire	91.70%	6.91%	91.75%	91.79%	91.74%	97.24%
adaBoost	77.24%	12.81%	82.02%	77.20%	77.49%	93.10%
Arbre de decision	89.15%	8.62%	89.12%	89.15%	89.13%	87.06%
kNN	92.85%	5.56%	92.94%	92.85%	92.84%	96.38%
Naive Bayes	92.06%	6.55%	92.08%	92.05%	92.02%	90.17%

GBM	90.47%	8.48%	89.00%	90.47%	89.62%	93.66%

D’après les figures IV.4 et IV.5, On peut observer que les algorithmes SVM, régression logistique et la forêt aléatoire émergent comme les meilleurs choix dans ce comparatif, grâce à leur robustesse et à leurs excellentes performances sur l'ensemble des métriques évaluées. À l'inverse, l'algorithme AdaBoost et l'arbre de décision se distinguent par des performances un peu plus modestes, notamment en termes de rappel et d'exactitude, ce qui peut limiter leur utilité dans certaines applications. En définitive, ces résultats soulignent l'importance de choisir le bon algorithme en fonction de la nature des données et des exigences spécifiques de la tâche de classification.

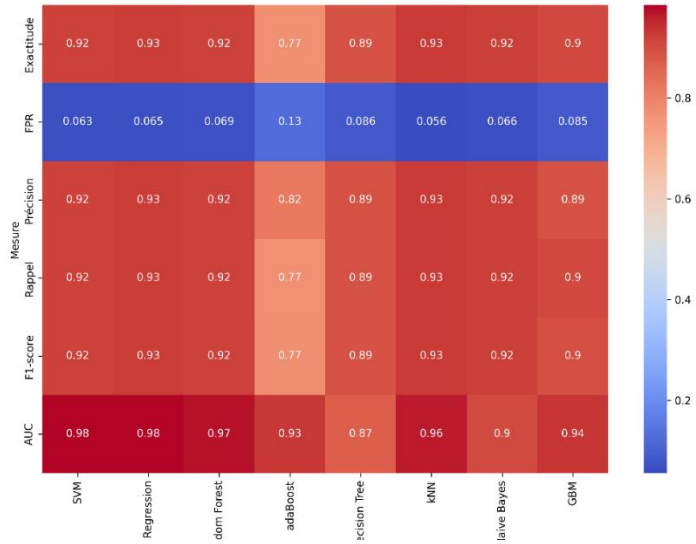

Figure IV.4 Carte thermique de la comparaison des performances des modèles de classification sur l’ensemble de données déséquilibré

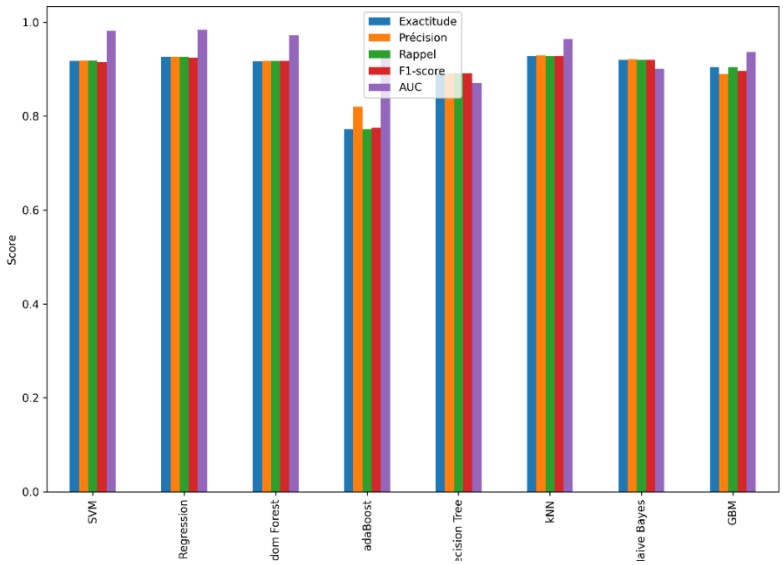

Figure IV.5 Analyse comparative des performances des modèles de classification sur l’ensemble de données déséquilibré

4.2 Ensemble de données modifiées avec SMOTE

Pour cette deuxième expérimentation, nous avons appliqué la technique de rééchantillonnage SMOTE afin de balancer l'ensemble de données, et ainsi mieux gérer le déséquilibre initial des classes. Les résultats obtenus, présentés dans le tableau 4.4, révèlent des améliorations notables dans les performances des modèles d'apprentissage automatique.

La machine à vecteurs de support (SVM) a démontré une excellente performance avec une exactitude de 94,17 %, une précision de 94,86 %, un rappel de 94,17 %, un score F1 de 94,29 % et une aire sous la courbe (AUC) de 98,21 %. L'utilisation de SMOTE semble avoir permis au modèle SVM de mieux gérer la classification des données déséquilibrées, comme en témoigne la réduction du taux de faux positifs (FPR) à 4,07 %. Ces résultats montrent que le SVM est non seulement capable de discriminer efficacement entre les classes, mais également de maintenir une haute précision et un bon rappel. Par ailleurs, le modèle de régression logistique multinomiale a surpassé légèrement le SVM, avec une exactitude de 94,44 %, une précision de 94,68 %, un rappel de 94,40 %, un score F1 de 94,50 % et une AUC de 98,50 %. Le FPR de 3,58 % indique une très bonne capacité à minimiser les erreurs de classification des instances négatives. Ces résultats soulignent que la régression logistique multinomiale, lorsqu'elle est combinée avec SMOTE, est extrêmement efficace pour gérer les ensembles de données déséquilibrées.

Le modèle de forêt aléatoire a également montré une amélioration notable avec une exactitude de 92,06 %, une précision de 92,09 %, un rappel de 92,06 %, un score F1 de 92,06 % et une AUC de 97,06 %. Bien que légèrement inférieur aux SVM et à la régression logistique en termes de précision globale, le modèle de forêt aléatoire reste un modèle solide, offrant une bonne balance entre les différentes métriques de performance. De même, le modèle k-plus proches voisins (kNN) a présenté des performances élevées avec une exactitude de 92,32 %, une précision de 92,92 %, un rappel de 92,30 %, un score F1 de 92,45 % et une AUC de 96,59 %. Le FPR de 4,70 % est relativement bas, ce qui confirme l'efficacité du kNN dans la classification précise après l'application de SMOTE.

En revanche, le modèle Naive Bayes a obtenu une exactitude de 90,47 %, avec une précision de 92,55 %, un rappel de 90,40 %, un score F1 de 91 % et une AUC de 92,51 %. Ces résultats, bien que légèrement inférieurs à ceux des autres modèles, montrent une amélioration grâce à SMOTE, en particulier au niveau de la précision. De même, le modèle d'arbre de décision a affiché une exactitude de 90,74 %, avec une précision de 90,66 %, un rappel de 90,74 %, un score F1 de 90,65 % et une AUC de 88,12 %. En dépit des performances du modèle, ses résultats restent inférieurs à ceux des autres modèles plus sophistiqués, soulignant peut-être une sensibilité plus élevée à la variance des données. Sur la même lancée, le modèle Gradient Boosting Machine (GBM) a obtenu une exactitude de 92,06 %, avec une précision de 92,14 %, un rappel de 92,06%, un score F1 de 92,09 % et une AUC de 97,73 %. Ces résultats indiquent une performance robuste, proche de celle du modèle de forêt aléatoire, et démontrent l'efficacité du GBM après l'application de SMOTE.

A l'inverse, l'algorithme adaBoost a montré une exactitude de 84,65 %, une précision de 85,82%, un rappel de 84,65 %, un score F1 de 84,98 % et une AUC de 92,28 %. Malgré l'amélioration par SMOTE, adaBoost reste moins performant que les autres modèles dans ce contexte. Le tableau 4.4 ci-dessous fournit plus de détails sur les résultats.

Tableau 4.4 Comparaison des Performances des modèles d'apprentissage automatique sur l'ensemble de données équilibré par SMOTE

Modèle	Exactitude	FPR	Précision	Rappel	F1-score	AUC

SVM	94.17%	4.07%	94.86%	94.17%	94.29%	98.21%
Forêt aléatoire	92.06%	6.06%	92.09%	92.06%	92.06%	97.06%
Arbre de decision	90.74%	7.92%	90.66%	90.74%	90.65%	88.12%
kNN	92.32%	4.70%	92.92%	92.30%	92.45%	96.59%
Naive Bayes	90.47%	4.99%	92.55%	90.40%	91%	92.51%
Régression Logistique	94.44%	3.58%	94.68%	94.40%	94.50%	98.50%
adaBoost	84.65%	10.37%	85.82%	84.65%	84.98%	92.28%
GBM	92.06%	6.03%	92.14%	92.06%	92.09%	97.73%

L'application de SMOTE afin de balancer les données a amélioré les performances de tous les modèles, en particulier pour les SVM et la régression logistique, qui ont montré des résultats exceptionnels. Les modèles de forêt aléatoire, kNN et GBM ont également bien performé, offrant des solutions efficaces pour les ensembles de données déséquilibrées. Ces résultats illustrent l'importance de techniques de rééchantillonnage comme SMOTE pour améliorer la classification dans des contextes de déséquilibre des classes. Dans l'ensemble, les résultats obtenus sont prometteurs. En effet, la méthode SMOTE permet d'améliorer la précision et l'adéquation du modèle entre les différents algorithmes, comme le montre les figures IV.6 et IV.7.

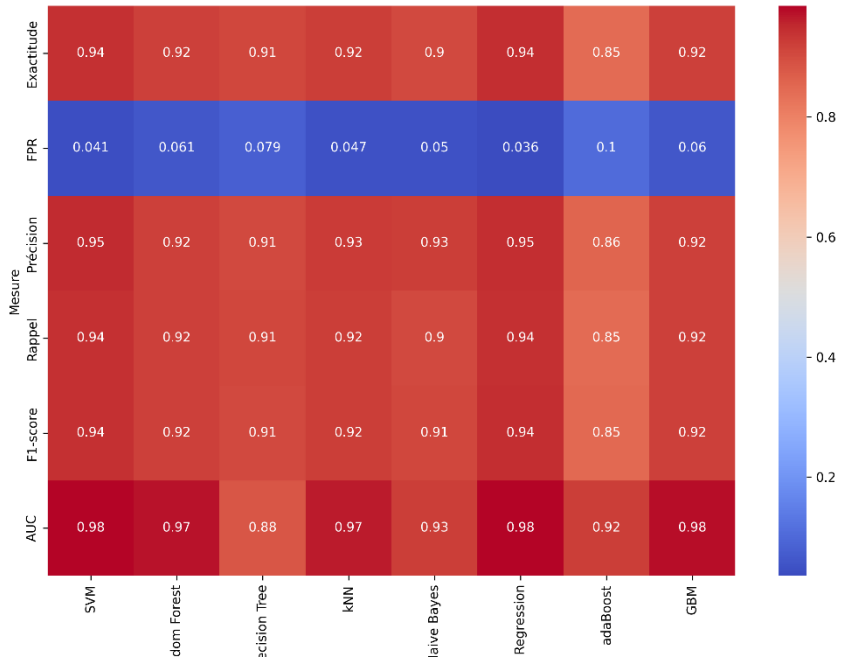

Figure IV.6 Carte thermique de la comparaison des performances des modèles de classification sur l'ensemble de données équilibré par SMOTE

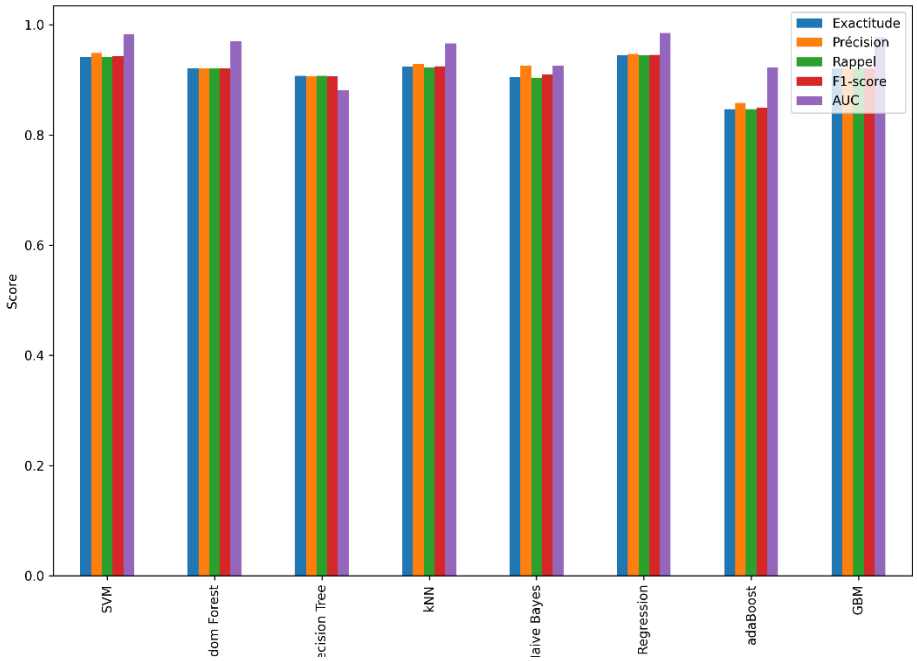

Figure IV.7 Analyse comparative des performances des modèles de classification sur l'ensemble de données équilibré par SMOTE

4.3 Ensemble de données modifiées avec SMOTE Borderline

Dans cette expérimentation, nous avons appliqué la technique de suréchantillonnage SMOTE Borderline pour balancer l'ensemble de données, cherchant à améliorer la gestion du déséquilibre initial des classes. Les résultats obtenus, présentés dans le tableau 4.5, montrent des performances variées parmi les différents modèles d'apprentissage automatique, avec des améliorations qu'on peut qualifier de notables pour certains.

Le modèle SVM a démontré une excellente performance avec une exactitude de 94,17 %, une précision de 95,12 %, un rappel de 94,18 %, un score F1 de 94,31 % et une aire sous la courbe (AUC) de 97,98 %. Le faible taux de faux positifs (FPR) de 3,31 % souligne la capacité du SVM à minimiser les erreurs de classification des instances négatives. Ces résultats démontrent une fois de plus que SVM, avec l'aide de SMOTE Borderline, est particulièrement efficace pour traiter les données déséquilibrées, maintenant un bon équilibre entre précision et rappel. Toutefois, le modèle de régression logistique a surpassé le SVM avec une exactitude de 94,97 %, une précision de 95,69 %, un rappel de 94,90 %, un score F1 de 95,07 % et une AUC de 98,40 %. Le FPR de seulement 2,30 % indique une capacité exceptionnelle à discriminer entre les classes, faisant de la régression logistique un choix de premier plan lorsqu'il est associé à SMOTE Borderline.

De même, le modèle de forêt aléatoire a montré une performance robuste avec une exactitude de 91,79 %, une précision de 91,91 %, un rappel de 91,70 %, un score F1 de 91,82 % et une AUC de 97,24 %. Même si légèrement inférieur aux SVM et à la régression logistique, le modèle de forêt aléatoire offre une performance stable et fiable, grâce à sa capacité à réduire la variance. Le modèle Gradient Boosting Machine (GBM) a obtenu une exactitude de 91,53 %, avec une précision de 91,61 %, un rappel de 91,50 %, un score F1 de 91,56 % et une AUC de 97,44 %. Ces résultats montrent une performance solide, proche de celle du modèle de la forêt aléatoire, et démontrent l'efficacité de GBM après l'application de SMOTE Borderline. Dans le même ordre d'idées, le modèle k-plus proches voisins (kNN) a obtenu une exactitude

de 91,53 %, une précision de 92,32 %, un rappel de 91,50 %, un score F1 de 91,69 % et une AUC de 96,24 %. Ces résultats montrent que kNN, après l'application de SMOTE Borderline, est capable de gérer efficacement les données déséquilibrées, en offrant une bonne balance entre toutes les métriques de performance.

A l'inverse, le modèle Naive Bayes a obtenu une exactitude de 87,03 %, avec une précision de 90,88 %, un rappel de 87 %, un score F1 de 87,70 % et une AUC de 90,89%. Malgré la précision relativement élevée de ce modèle, l'exactitude globale et le score F1 sont plus faibles comparativement aux autres modèles, suggérant que Naive Bayes, même si efficace, peut être limité dans certains contextes de données déséquilibrées. Pareillement, l'algorithme d'arbre de décision a affiché une exactitude de 89,41 %, avec une précision de 89,40 %, un rappel de 89,41 %, un score F1 de 89,39 % et une AUC de 87,30 %. Ces résultats sont inférieurs à ceux des autres modèles. Ce qui indique une performance plus modérée et une sensibilité potentiellement accrue à la variance des données.

Enfin, le modèle AdaBoost a montré des performances moins significatives avec une exactitude de 75,92 %, une précision de 77,77 %, un rappel de 75,93 %, un score F1 de 71,42 % et une AUC de 74,94 %. Ces résultats indiquent que, malgré l'application de SMOTE Borderline, AdaBoost a du mal à gérer efficacement le déséquilibre des données, avec un FPR particulièrement élevé de 25,16 %. Les résultats obtenus sont détaillés dans le tableau 4.5.

Tableau 4.5 Comparaison des Performances des modèles d'apprentissage automatique sur l'ensemble de données équilibré par SMOTE Borderline.

Modèle	Exactitude	FPR	Précision	Rappel	F1score	AUC
SVM	94.17%	3.31%	95.12%	94.18%	94.31%	97.98%
Forêt aléatoire	91.79%	5.89%	91.91%	91.70%	91.82%	97.24%
Arbre de decision	89.41%	8.46%	89.40%	89.41%	89.39%	87.30%
kNN	91.53%	5.07%	92.32%	91.50%	91.69%	96.24%
Naive Bayes	87.03%	6.07%	90.88%	87%	87.70%	90.89%
Régression Logisitique	94.97%	2.30%	95.69%	94.90%	95.07%	98.40%
AdaBoost	75.92%	5.16%	77.77%	75.93%	71.42%	74.94%
GBM	91.53%	6.45%	91.61%	91.50%	91.56%	97.44%

SMOTE Borderline a amélioré les performances de la plupart des modèles, particulièrement la régression logistique et SVM, qui ont montré des résultats exceptionnels. Les modèles forêt aléatoire, kNN et GBM ont également bien performé, offrant des solutions fiables pour les ensembles de données déséquilibrées. Les résultats de cette expérimentation soulignent l'importance de techniques de rééchantillonnage avancées comme SMOTE Borderline pour améliorer la classification dans des contextes de déséquilibre des classes.

Par rapport à la technique SMOTE, la technique SMOTE Borderline a permis d'obtenir un ensemble de données bien équilibré présentant des distributions de classes similaires. Les résultats indiquent une amélioration significative en termes d'exactitude, de précision, de rappel et de score F1 pour plusieurs modèles. La courbe AUC-ROC a également montré un pouvoir discriminatoire plus élevé pour ces modèles, ce qui signifie une meilleure efficacité de la classification. Les figures IV.8 et IV.9 ci-dessous présente les performances de la technique SMOTE Borderline.

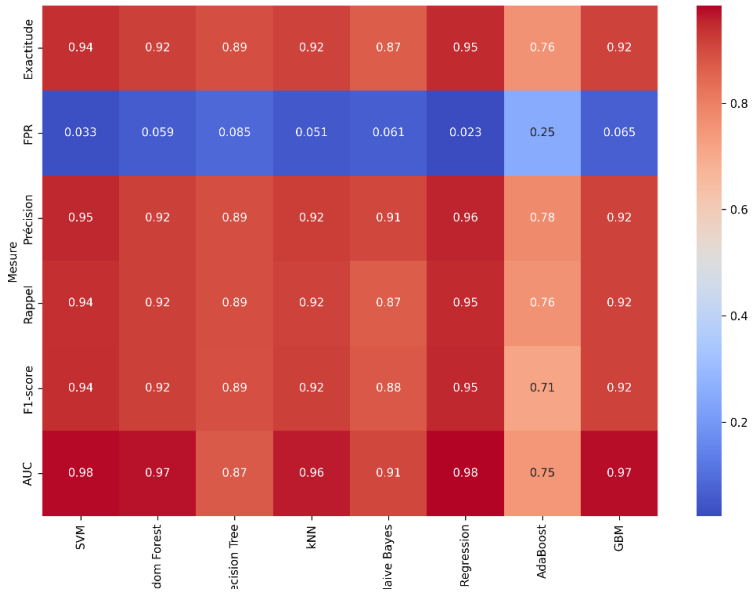

Figure IV.8 Carte thermique de la comparaison des performances des modèles de classification sur l'ensemble de données équilibré par SMOTE Borderline

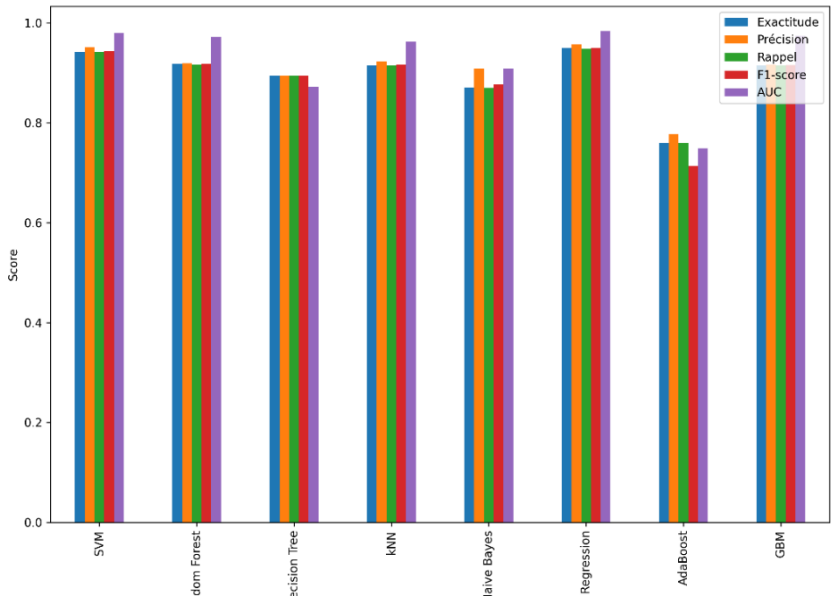

Figure IV.9 Analyse comparative des performances des modèles de classification sur l'ensemble de données équilibré par SMOTE Borderline

4.4 Ensemble de données modifiées avec SMOTE-ENN

Dans cette expérimentation, nous avons utilisé la technique de rééchantillonnage SMOTE-ENN pour équilibrer l'ensemble de données. Les résultats obtenus, présentés dans le tableau 4.6, montrent des variations significatives parmi les différents modèles d'apprentissage automatique, mettant en évidence les effets positifs de SMOTE-ENN sur certaines métriques de performance.

Le modèle SVM a démontré une performance significative avec une exactitude de 93,65 %, une précision de 94,7 %, un rappel de 93,65 %, un score F1 de 93,80 % et une aire sous la courbe (AUC) de 98,20 %. Le faible taux de faux positifs (FPR) de 3,11 % souligne la capacité du SVM à discriminer efficacement les classes, minimisant les erreurs de classification des instances négatives. Ces résultats démontrent que SVM, associé avec la technique SMOTE-ENN, est particulièrement performant pour traiter les données déséquilibrées, offrant un bon équilibre entre précision et rappel. Toutefois, le modèle de régression logistique a surpassé le SVM avec une exactitude de 94,97 %, une précision de 95,50 %, un rappel de 94,98 %, un score F1 de 95,06 % et une AUC de 98,41 %. Le FPR de seulement 2,61 % indique une capacité exceptionnelle à minimiser les erreurs, mettant en avant la régression logistique comme un choix optimal lorsqu'il est associé à SMOTE-ENN.

Le modèle de forêt aléatoire a montré une performance solide avec une exactitude de 90,74 %, une précision de 91,24 %, un rappel de 90,70 %, un score F1 de 90,84 % et une AUC de 97,19 %. Bien que légèrement inférieur aux SVM et à la régression logistique, le modèle de forêt aléatoire offre une performance stable. De même, le modèle k-plus proches voisins (kNN) a obtenu une exactitude de 92,06 %, une précision de 92,83 %, un rappel de 92 %, un score F1 de 92,20 % et une AUC de 95,43 %. Ces résultats montrent que kNN, après l'application de SMOTE-ENN, est capable de gérer efficacement les données déséquilibrées, en offrant une bonne balance entre toutes les métriques de performance. Pareillement, le modèle Naive Bayes a obtenu une exactitude de 90,21 %, avec une précision de 92,99 %, un rappel de 92,20 %, un score F1 de 90,77 % et une AUC de 93,36 %. Bien que ce modèle montre une précision relativement élevée, l'exactitude globale et le score F1 sont légèrement inférieurs comparativement aux autres modèles, suggérant que Naive Bayes peut être limité dans certains contextes de données déséquilibrées malgré l'application de SMOTE-ENN. Ces résultats sont alignés avec ceux de SMOTE et SMOTE Borderline.

L'algorithme d'arbre de décision a affiché une exactitude de 90,74 %, avec une précision de 91,25 %, un rappel de 90,70 %, un score F1 de 90,87 % et une AUC de 90,80 %. Ces résultats, bien que respectables, sont inférieurs à ceux des modèles plus sophistiqués, indiquant une performance modérée et une sensibilité potentiellement accrue à la variance des données. En revanche, le modèle AdaBoost a montré des performances moins significatives avec une exactitude de 84,65 %, une précision de 84,42 %, un rappel de 84,65 %, un score F1 de 84,32 % et une AUC de 89,48 %. Malgré l'application de SMOTE-ENN, le modèle AdaBoost a du mal à gérer efficacement le déséquilibre des données, avec un FPR particulièrement élevé de 13,40 %. Tout de même, il est important de souligner que ces résultats sont relativement supérieurs à ceux obtenus avec SMOTE Borderline. Enfin, le modèle Gradient Boosting Machine (GBM) a atteint une exactitude de 92,32 %, avec une précision de 92,70 %, un rappel de 92,30 %, un score F1 de 92,43 % et une AUC de 98,05 %. Ces résultats montrent une bonne performance, proche de celle de la forêt aléatoire, et démontrent l'efficacité de GBM après l'application de SMOTE-ENN. Les résultats suivants sont plus explicites dans le tableau 4.6.

Tableau 4.6 Comparaison des Performances des modèles d'apprentissage automatique sur l'ensemble de données équilibré par SMOTE-ENN

Modèle	Exactitude	FPR	Précision	Rappel	F1-score	AUC
SVM	93.65%	3.11%	94.7%	93.65%	93.80%	98.20%
Forêt aléatoire	90.74%	5.92%	91.24%	90.70%	90.84%	97.19%
Arbre de decision	90.74%	6.13%	91.25%	90.70%	90.87%	90.80%
kNN	92.06%	4.66%	92.83%	92%	92.20%	95.43%

Naive Bayes	90.21%	4.42%	92.99%	92.20%	90.77%	93.36%
Régression Logistique	94.97%	2.61%	95.50%	94.98%	95.06%	98.41%
AdaBoost	84.65%	13.40%	84.42%	84.65%	84.32%	89.48%
GBM	92.32%	5.05%	92.70%	92.30%	92.43%	98.05%

L'application de la technique de SMOTE-ENN sur l'ensemble des données déséquilibrés a amélioré les performances de la plupart des modèles, particulièrement la régression logistique et SVM, qui ont montré des résultats significatifs. Les modèles de la forêt aléatoire, kNN et GBM ont également bien performé, offrant des solutions efficaces pour les ensembles de données déséquilibrées. Ces résultats mettent en lumière les avantages d'utiliser la méthode SMOTE-ENN pour améliorer la classification dans des contextes de déséquilibre des classes, comme le montre les figures IV.10 et IV.11 ci-dessous.

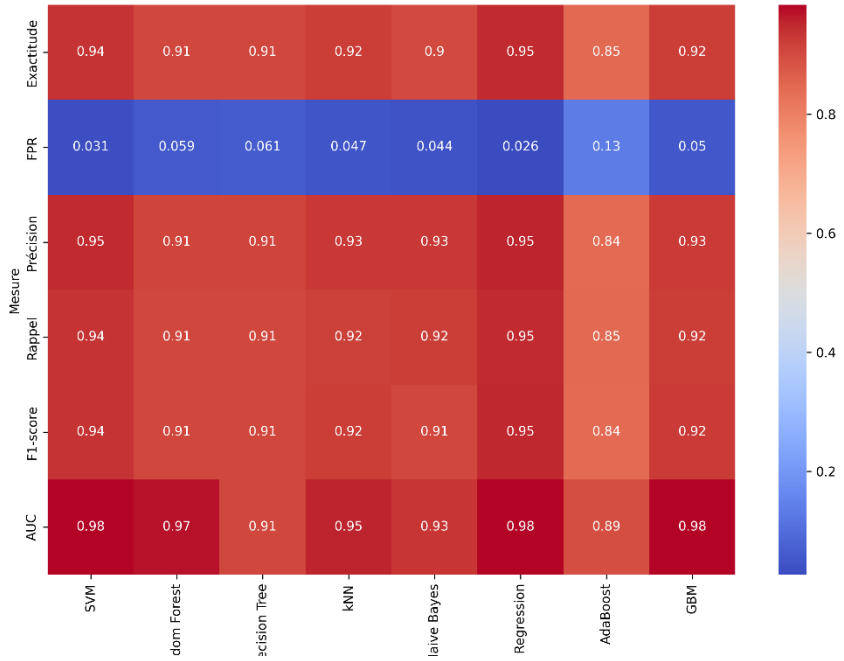

Figure IV.10 Carte thermique de la comparaison des performances des modèles de classification sur l'ensemble de données équilibré par SMOTE-ENN

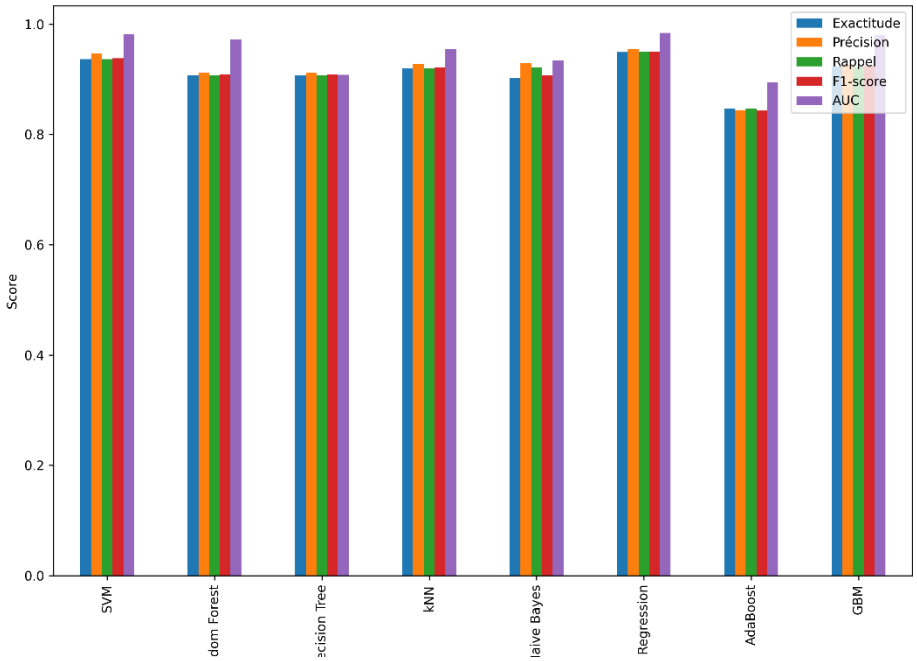

Figure IV.11 Analyse comparative des performances des modèles de classification sur l'ensemble de données équilibré par SMOTE-ENN

4.5 Ensemble de données modifiées avec ADASYN

Dans cette dernière expérimentation, nous avons utilisé la technique ADASYN pour balancer l'ensemble de données. Les résultats obtenus, présentés dans le tableau 4.7, montrent des performances variées parmi les différents modèles d'apprentissage automatique. Comparons ces résultats avec ceux obtenus précédemment à l'aide de SMOTE, SMOTE Borderline, et SMOTE-ENN pour mettre en évidence les différences et les améliorations.

Le modèle de régression logistique a surpassé les autres modèles avec une exactitude de 95,50 %, une précision de 96,09 %, un rappel de 95,52 %, un score F1 de 95,58 % et une AUC de 98,43 %. Le FPR de seulement 2,06 % est le plus bas parmi toutes les techniques de rééchantillonnage testées, indiquant une capacité exceptionnelle à minimiser les erreurs. En effet, le modèle de régression logistique montre une amélioration continue avec chaque technique, soulignant son adaptabilité et sa robustesse. Pareillement, le modèle SVM a démontré une performance significative avec une exactitude de 93,91 %, une précision de 94,94 %, un rappel de 93,91 %, un score F1 de 94,05 % et une aire sous la courbe (AUC) de 98,14 %. Le faible taux de faux positifs (FPR) de 2,84 % est légèrement inférieur à celui obtenu avec SMOTE Borderline (3,11 %) et SMOTE-ENN (3,31 %), et avec SMOTE (4,07 %). Ces résultats mettent en évidence qu'implémenter le modèle SVM, avec la méthode ADASYN, reste très performant pour traiter les données déséquilibrées, maintenant un bon équilibre entre précision et rappel.

Le modèle de forêt aléatoire a montré une bonne performance avec une exactitude de 92,32 %, une précision de 92,43 %, un rappel de 92,32 %, un score F1 de 92,34 % et une AUC de 97,15 %. Comparé aux résultats de SMOTE (92,06 %) et SMOTE-ENN (90,74 %) et SMOTE Borderline (91,79 %), ADASYN permet au modèle de forêt aléatoire d'atteindre une meilleure performance globale. Le modèle k-plus proches voisins (kNN) a obtenu une exactitude de 92,06 %, une précision de 92,71 %, un rappel de 92,06 %, un score F1 de 92,20 % et une AUC de 96,67 %. Ces résultats sont comparables à ceux obtenus avec SMOTE (92,32 %) et SMOTE-

ENN (92,06 %), mais légèrement inférieurs à ceux de SMOTE Borderline (91,53 %), attestant que kNN reste une option fiable pour la gestion des données déséquilibrées.

En revanche, le modèle Naive Bayes a atteint une exactitude de 89,41 %, avec une précision de 92,41 %, un rappel de 89,42 %, un score F1 de 90 % et une AUC de 92,60%. Bien que ce modèle montre une précision relativement élevée, l'exactitude globale est inférieure à celle obtenue avec SMOTE-ENN (90,21 %), SMOTE (90,47 %), et SMOTE Borderline (87,03 %). ADASYN n'apporte pas d'amélioration significative par rapport aux autres techniques pour Naive Bayes. De même, l'algorithme d'arbre de décision a affiché une exactitude de 91 %, avec une précision, un rappel, et un score F1 de 91 %, et une AUC de 89,39 %. Ces résultats sont tout de même légèrement inférieurs à ceux obtenus avec SMOTE-ENN (90,74 %), SMOTE (90,74 %), et SMOTE Borderline (89,41 %), indiquant que ADASYN apporte une amélioration « modérée » pour l'arbre de décision.

Le modèle AdaBoost a montré des performances faibles avec une exactitude de 85,71 %, une précision de 86,21 %, un rappel de 85,70 %, un score F1 de 85,88 % et une AUC de 91,85 %. Bien que meilleur que les résultats obtenus avec SMOTE (84,65 %) et SMOTE Borderline (75,92 %), ADASYN reste moins efficace pour AdaBoost comparé aux autres modèles testés. Enfin, le modèle Gradient Boosting Machine (GBM) a obtenu une exactitude de 91 %, avec une précision de 91,08 %, un rappel de 91,05 %, un score F1 de 91,03 % et une AUC de 97,08 %. Ces résultats sont comparables à ceux obtenus avec SMOTE (92,06 %) et SMOTE Borderline (91,53 %), montrant que GBM reste performant indépendamment de la technique de rééchantillonnage utilisée. Le tableau 4.7 fournit plus de détails sur les résultats.

Tableau 4.7 Comparaison des Performances des modèles d'apprentissage automatique sur l'ensemble de données équilibré par ADASYN

Modèle	Exactitude	FPR	Précision	Rappel	F1-score	AUC
SVM	93.91%	2.84%	94.94%	93.91%	94.05%	98.14%
Forêt aléatoire	92.32%	5.60%	92.43%	92.32%	92.34%	97.15%
Arbre de Décision	91%	7.07%	91%	91%	91%	89.39%
kNN	92.06%	4.83%	92.71%	92.06%	92.20%	96.67%
Naive Bayes	89.41%	4.93%	92.41%	89.42%	90%	92.60%
Régression Logisitque	95.50%	2.06%	96.09%	95.52%	95.58%	98.43%
AdaBoost	85.71%	10.60%	86.21%	85.70%	85.88%	91.85%
GBM	91%	6.87%	91.08%	91.05%	91.03%	97.08%

L'utilisation de la technique ADASYN a amélioré les performances de la plupart des modèles, particulièrement la régression logistique et SVM, qui ont montré des résultats significatifs. Les modèles Forêt aléatoire, kNN et GBM ont également bien performé. Comparé aux autres techniques de rééchantillonnage (SMOTE, SMOTE Borderline, et SMOTE-ENN), ADASYN montre une amélioration notable pour certains modèles comme la régression logistique et SVM, tout en offrant des performances comparables ou légèrement meilleures pour d'autres modèles comme La Forêt aléatoire et kNN, comme le montre les figures IV.12 et IV.13 ci-dessous. Ces résultats comme ceux des précédentes expérimentations illustrent l'importance de choisir la technique de rééchantillonnage appropriée pour optimiser la

performance des modèles d'apprentissage automatique dans des contextes de déséquilibre des classes.

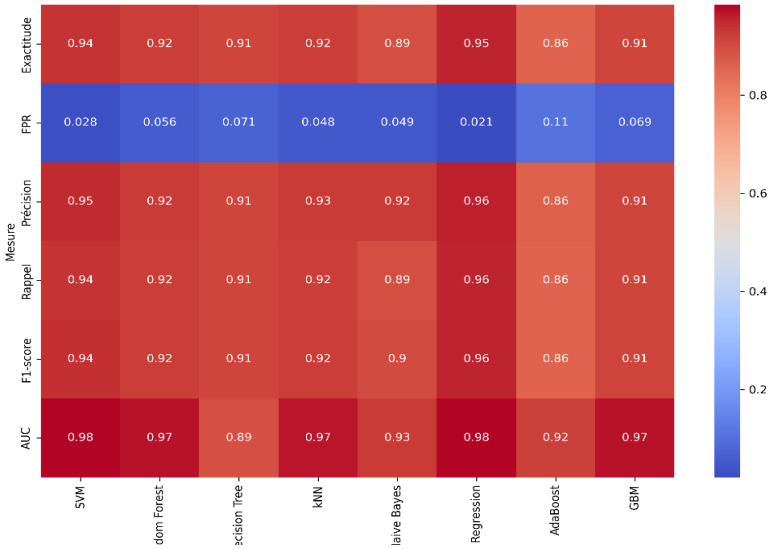

Figure IV.12 Carte thermique de la comparaison des performances des modèles de classification sur l'ensemble de données équilibré par ADASYN

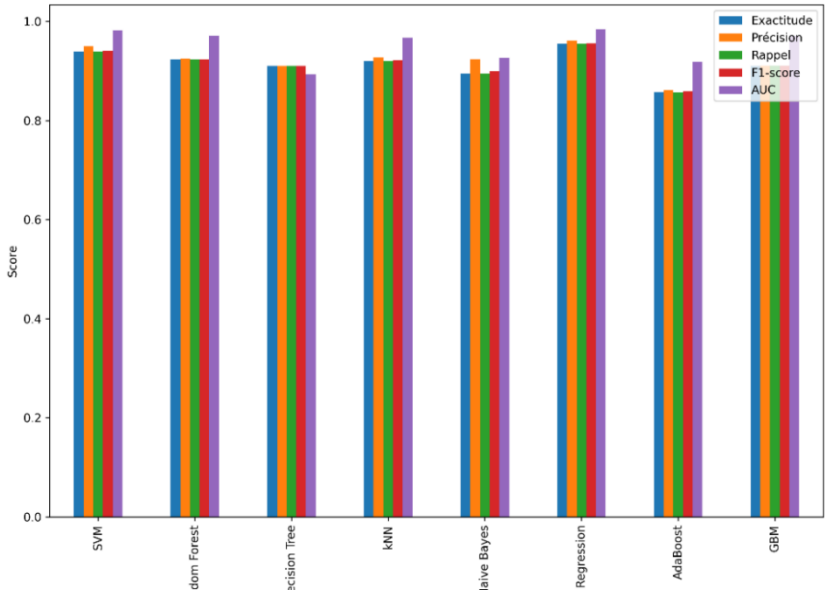

Figure IV.13 Analyse comparative des performances des modèles de classification sur l'ensemble de données équilibré par ADASYN

4.6 Les Résultats de Equi-Fused-Data-based SMOTE

Les résultats expérimentaux présentés dans le tableau 4.8 mettent en évidence les performances significatives des modèles à travers diverses mesures d'évaluation, indiquant leur efficacité dans le traitement de la tâche de classification. Ainsi, en examinant les différentes étapes de l'approche, nous pouvons mieux comprendre l'impact de chaque configuration de classifieur sur la performance globale de la méthode proposée à savoir Equi-Fused-Data.

Tout d'abord, les résultats de l'algorithme Balanced Bagging avec une forêt aléatoire comme classifieur montrent une forte efficacité en termes de performance. Cette configuration affiche une exactitude de 93.85%, un F1-score de 92.86% et une AUC de 98.08%. Ces résultats s'alignent sur les performances de la technique SMOTE-ENN, qui s'est révélée être la technique la plus efficace pour cet ensemble de données. La cohérence de ces résultats indique que l'utilisation de données synthétiques pour les problèmes multi-classes peut être une stratégie viable. En effet, en combinant les différentes versions suréchantillonnées de l'ensemble de données, nous incorporons un ensemble diversifié d'échantillons synthétiques, équilibrant ainsi la distribution des classes et capturant un éventail plus large de modèles dans toutes les classes. La stabilité du Balanced Bagging démontre que cette technique d'ensemble reste efficace pour traiter le déséquilibre des classes dans les problèmes multi-classes, fournissant des prédictions stables comme le confirment nos résultats expérimentaux.

Par la suite, nous avons évalué l'algorithme Balanced Bagging en employant la régression logistique multinomiale comme classifieur, étant donné que ce modèle se révèle particulièrement performant lorsqu'il est associé à diverses techniques de suréchantillonnage, comme le suggèrent nos observations. Les résultats montrent une amélioration considérable par rapport à l'approche précédente, avec une exactitude de 95.76%, un F1-score de 95.85%, et une AUC de 98.94%. Cette amélioration suggère que la régression logistique multinomiale est particulièrement efficace pour ce type de problème, probablement en raison de sa capacité à modéliser des relations linéaires entre les caractéristiques et les classes. La précision accrue de 96.32% et la réduction du FPR à 1.97% indiquent une meilleure capacité à discriminer entre les classes, ce qui est capital pour les ensembles de données déséquilibrés.

Enfin, nous avons combiné les classifieurs forêt aléatoire et régression logistique multinomiale dans une approche d'ensemble pour le Balanced Bagging. Cette méthode ensembliste a légèrement réduit les performances globales par rapport à l'utilisation exclusive de la régression logistique multinomiale, avec une exactitude de 92.85%, un F1-score de 92.79%, et une AUC de 98.27%. Bien que cette configuration n'ait pas surpassé les performances de la régression logistique multinomiale seule, elle démontre que la combinaison de différents modèles peut offrir des avantages en termes de robustesse et de généralisation. Cependant, cette approche nécessite un ajustement minutieux pour maximiser les gains potentiels.

Tableau 4.8 La performance du Balanced Bagging sur l'ensemble de données équi-fusionné

Métrique	Forêt aléatoire	Régression Logistique	Ensemble
Exactitude	93.85%	95.76%	92.85%
FPR	5.35%	1.97%	6.07 %
Précision	92.86%	96.32%	92.85%
Rappel	92.80%	95.77%	92.86%
F1-score	92.86%	95.85%	92.79%
AUC	98.08%	98.94%	98.27%
MCC	0.83	0.91	0.83

La méthode proposée, utilisant la coopération de données synthétiques, montre un potentiel significatif pour mieux se généraliser à travers différents ensembles de données. En évitant la nécessité de sélectionner une technique de suréchantillonnage unique ou de régler les

hyperparamètres pour une méthode spécifique, cette approche offre une flexibilité et une robustesse accrues. Les résultats expérimentaux soulignent l'importance d'explorer différentes configurations de classifieurs et de combinaisons pour optimiser les performances des modèles dans des contextes variés, comme soulignée par la figure IV.14.

Pour résumer, l'implémentation du modèle Balanced Bagging avec la régression logistique multinomiale comme classifieur de base nous donne les résultats suivants (tableau 4.9):

Tableau 4.9 Analyse comparative des différentes techniques de suréchantillonnage sur l'ensemble déséquilibré, suréchantilloné, et équi-fusion des données

Dataset	Exactitude	FPR	Précision	Rappel	F1-score	AUC
Déséquilibré	92.59%	6.54%	92.60%	92.59%	92.50%	98.42%
SMOTE	94.44%	3.58%	94.68%	94.40%	94.50%	98.50%
SMOTE Borderline	94.97%	2.30%	95.69%	94.90%	95.07%	98.40%
SMOTE-ENN	94.97%	2.61%	95.50%	94.98%	95.06%	98.41%
ADASYN	95.50%	2.06%	96.09%	95.52%	95.58%	98.43%
Equi-Fused-Data	**95.76%**	**1.97%**	**96.32%**	**95.77%**	**95.85%**	**98.94%**

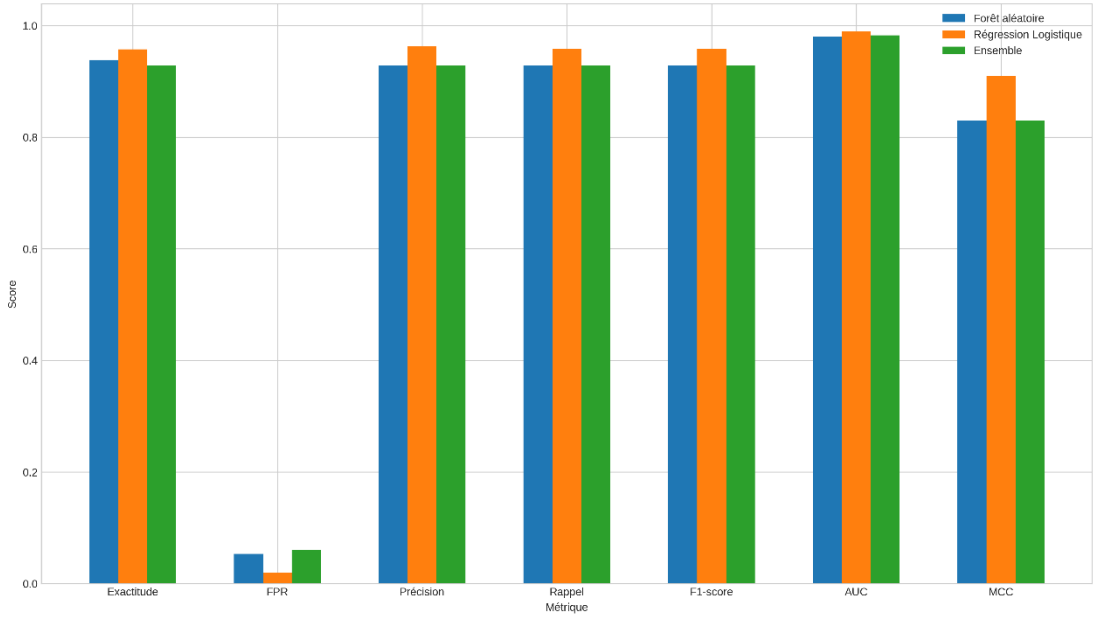

Figure IV.14 Analyse comparative des performances du balanced bagging avec différents modèles de classification sur l'ensemble de données équilibré par la fusion des données

On observe une amélioration générale des performances avec chaque méthode de rééchantillonnage par rapport à l'ensemble de données déséquilibré d'origine. Les meilleures performances sont obtenues avec Equi-Fused-Data, affichant une exactitude de 95.76%, une précision de 96.32%, un rappel de 95.77%, un F1-score de 95.85%, une AUC de 98.94% et le plus faible taux de faux positifs à 1.97%.

Le modèle proposé semble surmonter les limitations des techniques simples de suréchantillonnage en incorporant des échantillons synthétiques divers provenant de multiples

versions des données. Cette représentation exhaustive des données améliore vraisemblablement l'apprentissage du modèle. De plus, la stratégie Equi-Fused-Data est spécifiquement conçue pour les problèmes à classes multiples, offrant ainsi des avantages par rapport aux techniques principalement basées sur la classification binaire.

La figure IV.15 ci-dessous illustre la différence entre l'application de la régression logistique multinomiale sur l'ensemble de données déséquilibré, les ensembles de données SMOTE modifié, SMOTE Borderline, SMOTE-ENN, ADASYN, et l'ensemble de données Equi-Fused-Data.

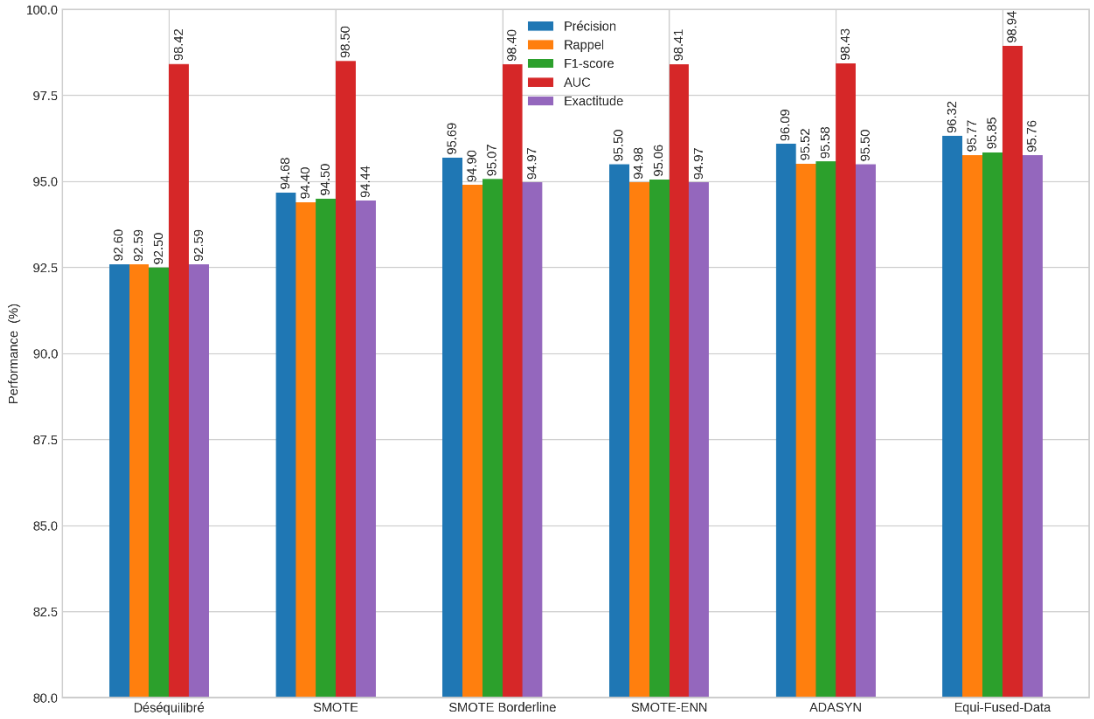

Figure IV.15 Analyse comparative des différentes techniques de suréchantillonnage sur l'ensemble déséquilibré, suréchantillonné, et équi-fusion des données

5 Complexité des Algorithmes

5.1 Complexité des Méthodes de Suréchantillonnage

Les techniques de suréchantillonnage présentées ici sont employées pour remédier au problème du déséquilibre des classes dans les jeux de données, où une classe minoritaire est représentée par un nombre d'instances bien inférieur aux classes majoritaires.

Tout d'abord, en termes de complexité temporelle, la méthode SMOTE requiert un temps proportionnel à $(O(N \log N))$ pour la recherche des (k) plus proches voisins. Cette recherche est essentielle pour identifier les instances à partir desquelles les échantillons synthétiques seront créés. Ensuite, la génération de ces instances synthétiques elle-même a une complexité de$(O(N \cdot k))$, où (N) représente le nombre d'échantillons de la classe minoritaire et (k) le nombre de voisins pris en compte pour la génération. De même, pour la complexité spatiale, SMOTE nécessite un espace de stockage proportionnel à $(O(N \cdot k))$ pour conserver les instances synthétiques créées.

Ensuite, pour SMOTE Borderline, la visée est d'améliorer SMOTE en concentrant la génération d'échantillons synthétiques sur les frontières entre les classes, ce qui permet de

mieux définir les zones de séparation entre les classes majoritaire et minoritaire. Par conséquence, la complexité temporelle de cette méthode inclut celle de SMOTE, avec une étape supplémentaire consistant à identifier les instances proches des frontières décisionnelles. Cette étape supplémentaire élève la complexité temporelle à $(O(N^2))$ pour la recherche des voisins proches. En termes de complexité spatiale, elle est similaire à celle de SMOTE mais inclut un facteur additionnel pour les instances générées à proximité des frontières, nécessitant ainsi un espace de stockage proportionnel à $(O(N \cdot k))$.

En revanche, la complexité temporelle de SMOTE-ENN est une combinaison des complexités de SMOTE $((O(N\,log\,N + N \cdot k)))$ et de ENN$((O(N^2)))$, ce qui en fait une méthode assez gourmande en temps de calcul. Même chose pour la complexité spatiale, elle inclut l'espace nécessaire pour les instances synthétiques créées par SMOTE ainsi que celles conservées après le nettoyage par ENN, rendant ainsi la méthode plus exigeante en termes de stockage.

Enfin, la complexité temporelle de la méthode ADASYN est de $((O(N^2)))$ pour le calcul de la difficulté de classification des instances, suivie d'une complexité de $(O(N \cdot k))$pour la génération des instances synthétiques. Cette méthode vise à générer plus d'instances dans les régions de l'espace de données où la classification est plus difficile. En termes de complexité spatiale, ADASYN nécessite un espace proportionnel à $(O(N))$ pour le stockage des instances synthétiques générées de manière adaptative, ce qui peut être moins gourmand que certaines autres méthodes de suréchantillonnage.

5.2 Complexité de la Méthode Equi-Fused-Data

Pour analyser la complexité temporelle et spatiale de l'algorithme Equi-Fused-Data, il est nécessaire d'examiner chaque étape de manière détaillée, en tenant compte des opérations effectuées par les méthodes de suréchantillonnage sous-jacentes (SMOTE, SMOTE Borderline, SMOTE-ENN, ADASYN) ainsi que des opérations de concaténation et de suppression des doublons.

5.2.1 Suréchantillonnage et Suppression des doublons

1. Complexité des Méthodes de Suréchantillonnage : Comme susmentionné pour chaque méthode de suréchantillonnage.
2. Concaténation des Instances Synthétiques : une complexité temporelle estimée à $(O(n))$, où (n) est le nombre total d'instances synthétiques générées. Cette opération est linéaire par rapport au nombre d'instances.
3. Suppression des Doublons : une complexité temporelle de $(O(n\,log\,n))$, en utilisant des structures de données efficaces comme les ensembles pour détecter et supprimer les doublons.

Par conséquent, la complexité temporelle totale de l'algorithme Equi-Fused-Data est dominée par la complexité des méthodes de suréchantillonnage utilisées, qui sont généralement linéaires ou légèrement supérieures à linéaires par rapport au nombre d'échantillons minoritaires. La concaténation et la suppression des doublons ajoutent une complexité linéaire et logarithmique respectivement, mais ces opérations sont souvent négligeables par rapport aux méthodes de suréchantillonnage elles-mêmes.

La complexité spatiale fait référence à la quantité de mémoire requise par l'algorithme pour stocker et manipuler les données :

1. Stockage des Données Originales : une complexité spatiale de l'ordre de $(O(m)), o$ù(m) est le nombre d'échantillons dans l'ensemble de données original.

2. Stockage des Instances Synthétiques : une complexité spatiale $(O(n)), o\grave{u}\ (n)$ est le nombre total d'instances synthétiques générées par toutes les méthodes combinées.

Ainsi, la complexité spatiale de la première phase de la méthode Equi-Fused-Data est principalement dictée par la quantité de données générées par les méthodes de suréchantillonnage et le besoin de stockage temporaire pour les opérations de concaténation et de suppression des doublons. Bien que l'algorithme puisse nécessiter une mémoire supplémentaire pour gérer les données synthétiques, cela reste généralement gérable pour des ensembles de données de taille modérée à grande.

5.2.2 Classification en utilisant le modèle Balanced Bagging

1. Prétraitement et Suréchantillonnage
   - Le chargement des fichiers CSV a une complexité temporelle de $(O(m)), o\grave{u}(m)$est le nombre de lignes dans le fichier, et une complexité spatiale de $(O(m \cdot d)), o\grave{u}(d)$ est le nombre de caractéristiques (colonnes) dans le fichier.
   - Pour l'encodage des caractéristiques catégorielles, la complexité temporelle est de $(O(m \cdot d))$ et la complexité spatiale est de $(O(m \cdot d_{encodé})), o\grave{u}\ (d_{encodé})$ est le nombre total de colonnes après l'encodage. L'encodage des étiquettes présente une complexité temporelle et spatiale de$(O(m))$, car chaque étiquette est encodée individuellement et nous n'avons qu'une seule colonne d'étiquettes.
   - L'alignement du jeu de test avec les colonnes encodées du jeu d'entraînement a une complexité temporelle de $(O(m_{test} \cdot d))$ et une complexité spatiale de$(O(m_{test} \cdot d_{encodé}))$.
   - Le suréchantillonnage avec SMOTE implique une recherche des (k) plus proches voisins avec une complexité temporelle de $(O(N\ log\ N))$ et une génération d'instances synthétiques avec une complexité temporelle de$(O(N \cdot k))$. La complexité spatiale pour stocker ces nouvelles instances synthétiques est de $(O(N \cdot d)), o\grave{u}(N)$ est le nombre d'instances minoritaires.
2. Classification
   a. Balanced Bagging avec Régression Logistique

L'initialisation et l'ajustement du classifieur Balanced Bagging impliquent plusieurs régressions logistiques. La complexité temporelle pour ajuster un seul classifieur est $(O(n \cdot d \cdot k)), o\grave{u}(n)$ est le nombre d'instances après suréchantillonnage, (d) est le nombre de caractéristiques, et (k) est le nombre d'itérations pour la convergence. Avec (T) estimations, la complexité devient$(O(T \cdot n \cdot d \cdot k))$. La complexité spatiale pour stocker les multiples modèles et instances échantillonnées est$(O(T \cdot m \cdot d))$.

La validation croisée a une complexité temporelle de $(O(T \cdot n \cdot d \cdot k \cdot v)), o\grave{u}(v)$ est le nombre de plis (ici$(v\ =\ 10)$). La mémoire nécessaire pour stocker les modèles pendant cette validation croisée est également prise en compte dans la complexité spatiale.

Les prédictions sur le jeu de test présentent une complexité temporelle de $(O(T \cdot m_{test} \cdot d))$ et une complexité spatiale de $(O(T \cdot m_{test}))$ pour stocker les probabilités prédictives.

Ainsi pour résumer,

- ✓ La complexité temporelle totale pour la phase de prétraitement et de suréchantillonnage est$(O(m \cdot d + N\ log\ N + N \cdot k))$. Pour le Balanced Bagging et la validation croisée, elle est$(O(T \cdot n \cdot d \cdot k \cdot v))$, et pour les prédictions et évaluations, elle est$(O(T \cdot m_{test} \cdot d + m_{test} \cdot n_{classes}))$.
- ✓ La complexité spatiale totale pour la phase de prétraitement et de suréchantillonnage est $(O(m \cdot d + N \cdot d))$. Pour le Balanced Bagging et la validation croisée, elle est

$(O(T \cdot m \cdot d))$, et pour les prédictions et évaluations, elle est$(O(T \cdot m_{test} + n_{classes} \cdot m_{test}))$.

b. Balanced Bagging avec la Forêt Aléatoire

La complexité temporelle pour ajuster un algorithme de forêt aléatoire est \(O(T $\cdot n \cdot d \cdot log\ n \cdot k)), o$ù (T) est le nombre d'arbres, (n) le nombre d'instances, (d) le nombre de caractéristiques et (k) la profondeur maximale des arbres. Avec Balanced Bagging, cette complexité devient $(O(E \cdot T \cdot n \cdot d \cdot log\, n \cdot k)), (E)$ étant le nombre d'estimations. La complexité spatiale est $(O(E \cdot T \cdot n \cdot d))$. La validation croisée et les prédictions sur le jeu de test sont similaires à ceux de la régression logistique.

Ainsi, l'utilisation d'une forêt aléatoire comme algorithme de base pour le Balanced Bagging augmente la complexité temporelle et spatiale, en raison de la construction et de l'agrégation des arbres, nécessitant plus de mémoire pour stocker les arbres et les données associées. Comparativement, Balanced Bagging avec une forêt aléatoire est généralement plus complexe que celui avec une régression logistique, en termes de temps et de mémoire.

Le choix de la technique de suréchantillonnage la plus appropriée dépend de divers facteurs tels que la taille du jeu de données, la distribution des classes et les contraintes de temps et de mémoire. SMOTE et ADASYN sont populaires pour leur simplicité et efficacité. SMOTE Borderline est particulièrement utile lorsque les instances minoritaires sont situées près des frontières des classes majoritaires. SMOTE-ENN peut améliorer les performances en éliminant les instances mal classifiées, bien que sa complexité temporelle soit plus élevée.

La méthode Equi-Fused Data avec Balanced Bagging est une approche prometteuse pour traiter les jeux de données déséquilibrés. Bien qu'elle présente une complexité temporelle et spatiale relativement élevée en raison des multiples étapes de prétraitement, de suréchantillonnage, de validation croisée et d'évaluation. Toutefois, cette complexité est gérable et la méthode offre des performances significatives et équilibrées.

6 Discussion

6.1 Rééquilibrage des Classes: Vers une Évaluation Équitable des Performances Étudiantes avec l'Apprentissage Automatique

Comme l'a démontré la première expérience de cette étude, le déséquilibre des classes constitue un défi majeur dans le domaine de l'éducation. Les ensembles de données éducatifs présentent fréquemment une distribution inégale des catégories de performance des étudiants, créant des obstacles significatifs à l'analyse précise et équitable des résultats. Ce déséquilibre biaise les modèles d'apprentissage automatique en faveur de la classe majoritaire, ce qui conduit à une précision globale trompeusement élevée. Cependant, cette apparente exactitude (accuracy) masque souvent une performance médiocre dans l'identification des étudiants appartenant aux classes minoritaires, ceux qui ont des performances différentes de la norme.

Les résultats de cette étude soulignent l'importance de prendre en compte le déséquilibre des classes lors de l'utilisation de l'apprentissage automatique pour l'évaluation éducative. En ignorant cette disparité, on risque de développer des modèles qui ne sont pas véritablement représentatifs de la diversité des performances des étudiants, ce qui peut avoir des conséquences graves. Par exemple, les étudiants en difficulté pourraient ne pas être correctement identifiés, manquant ainsi l'opportunité de bénéficier des interventions nécessaires pour améliorer leurs résultats académiques.

L'application de techniques telles que le suréchantillonnage des classes minoritaires, comme le modèle SMOTE (Chawla et al. 2002) peut significativement atténuer ce problème. En rééquilibrant les ensembles de données, ces techniques permettent aux modèles d'apprentissage automatique de mieux reconnaître et prédire les performances des étudiants des classes minoritaires, assurant ainsi une évaluation plus juste et plus précise. Cela est particulièrement fondamental dans le contexte éducatif, où chaque étudiant mérite une évaluation honnête de ses capacités et de ses besoins.

Ces conclusions appellent à une intégration plus consciente et absolue des considérations liées au déséquilibre des classes dans le développement et l'application des modèles d'apprentissage automatique en éducation. En adoptant ces approches, les éducateurs et les chercheurs peuvent améliorer la fiabilité et l'équité des outils d'évaluation, contribuant à une meilleure compréhension et à un soutien plus efficace des performances des étudiants. Par conséquent, l'attention accrue portée au déséquilibre des classes dans l'apprentissage automatique éducatif est non seulement justifiée mais essentielle pour promouvoir l'équité dans le système éducatif.

6.2 Efficacité des techniques de suréchantillonnage

Le premier objectif de cette étude est de démontrer l'efficacité de diverses techniques de suréchantillonnage pour traiter les problèmes de déséquilibre des classes dans l'ensemble de données éducatives. Les résultats montrent que les techniques SMOTE, SMOTE Borderline, SMOTE-ENN et ADASYN sont efficaces pour traiter le problème du déséquilibre des classes, assurant des performances cohérentes entre les ensembles d'entraînement et de test et prévenant le surapprentissage.

L'analyse proposée démontre que SMOTE produit des résultats prometteurs, en particulier pour les modèles de régression logistique et SVM, qui affichent des valeurs AUC significatives (par exemple, 98% pour SVM). Cette constatation est cohérente avec celles d'autres études, telles que celles de (Rozi, Wibowo, et Warsito 2023; Tariq et al. 2023; Wongvorachan, He, et Bulut 2023), qui ont rapporté une performance raisonnable de SMOTE tout en soulignant que son efficacité dépend de facteurs tels que la distribution des données, le niveau de bruit et le classifieur utilisé. Par exemple, dans cet ensemble de données, la combinaison de SMOTE et la régression logistique multinomiale a obtenu la meilleure performance, alors que dans l'étude de (Tariq et al. 2023), le meilleur classifieur était kNN, ce qui implique que différents classifieurs peuvent répondre différemment au suréchantillonnage. Certains algorithmes, tels que kNN, qui reposent sur des distances locales, peuvent bénéficier davantage de SMOTE car il offre une frontière de décision plus claire. En outre, dans les travaux de (Khalaf Hamoud et al. 2022), La forêt aléatoire a surpassé les autres algorithmes en termes de précision, rappel et F1-score (81%). Le modèle a produit des résultats cohérents (92%), indiquant que l'utilisation de SMOTE a amélioré les performances du modèle dans de nombreux cas, mais son efficacité reste dépendante de divers facteurs.

De même, SMOTE Borderline a amélioré l'efficacité des modèles SVM en augmentant la précision, le rappel, le F1-score et l'AUC. Cependant, son efficacité varie selon l'algorithme, ce qui implique qu'il n'est pas intrinsèquement supérieur à SMOTE. Des recherches supplémentaires sur son application à divers ensembles de données éducatifs sont nécessaires. SMOTE-ENN a surpassé d'autres techniques, obtenant des valeurs AUC impressionnantes pour les modèles SVM et Random Forest (98,2% et 97,19%, respectivement). De plus, il a produit un taux de faux positifs (FPR) plus faible, démontrant son efficacité dans l'amélioration de la classification à travers plusieurs modèles. Enfin, ADASYN a fourni des résultats de performance satisfaisants. Bien que la plupart des modèles aient atteint des valeurs AUC allant de 92,6% à 98,14%, les classifieurs AdaBoost et l’arbre de décision avaient une valeur AUC

basse (environ 90%). Cela implique qu'ADASYN peut être moins efficace pour certains algorithmes par rapport à d'autres méthodes. Similaire aux résultats de (Rozi et al. 2023), ADASYN a amélioré les performances de classification, mais son efficacité, comme celle de SMOTE, dépend des caractéristiques spécifiques des données et des modèles utilisés. Par exemple, dans cet ensemble de données, ADASYN a obtenu les meilleurs résultats avec SVM, la forêt aléatoire et kNN, alors que dans le travail de (Rozi et al. 2023), l'algorithme de stacking a obtenu les meilleurs résultats globaux. Les figures IV.16 et IV.17 présentent une comparaison complète de l'exactitude et de l'AUC des différents modèles. La plupart des modèles, à l'exception de Naive Bayes, affichent une exactitude globale plus élevée lorsqu'ils utilisent des techniques de suréchantillonnage. En termes d'AUC, ces techniques produisent des résultats cohérents à travers tous les modèles.

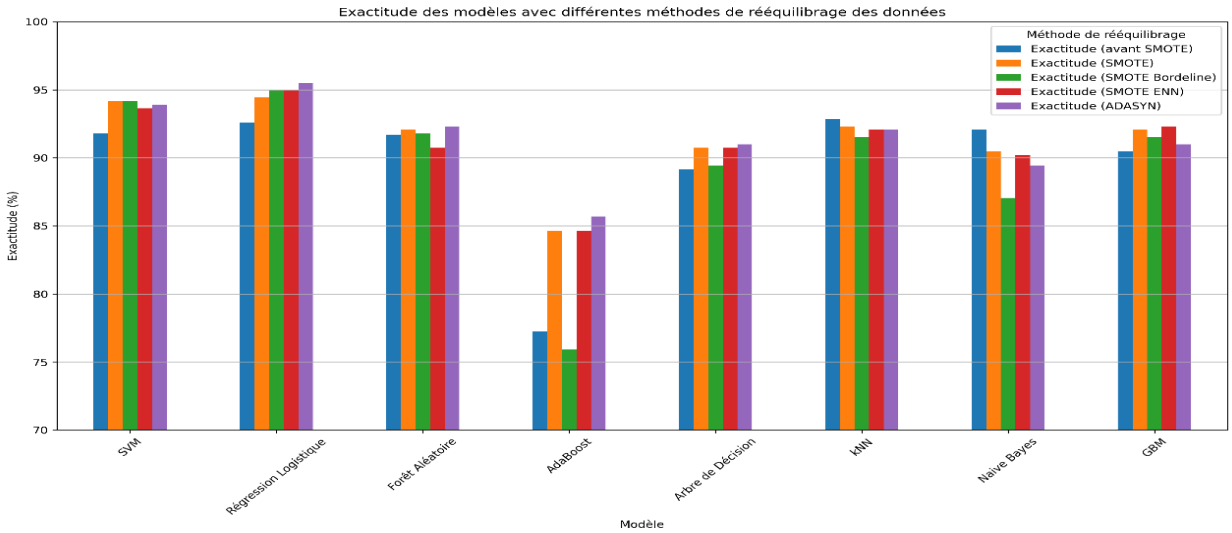

Figure IV.16 L'exactitude des modèles avec différentes méthodes de rééquilibrage des données

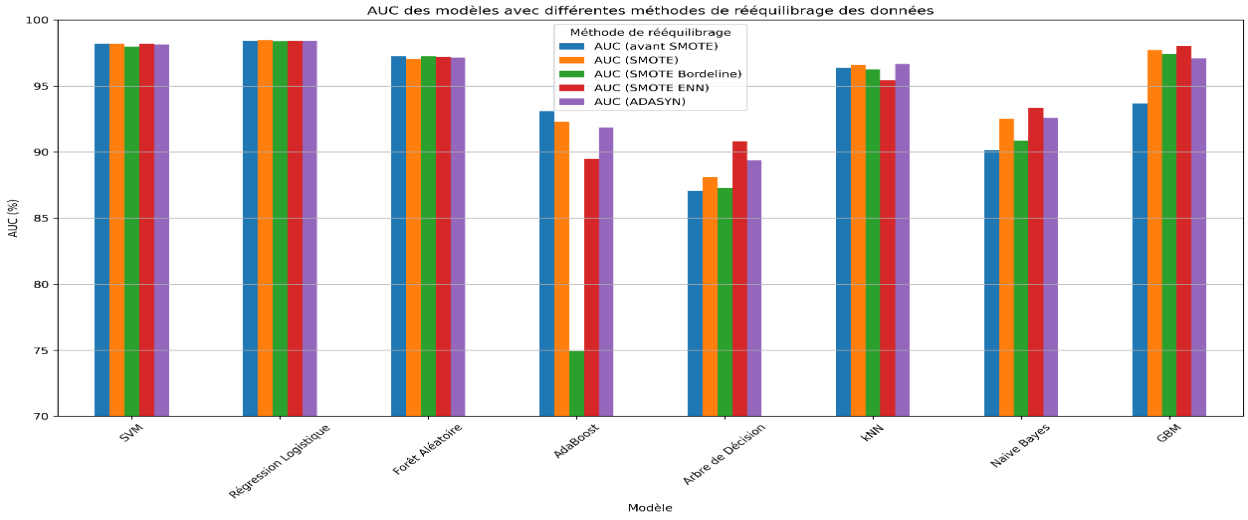

Figure IV.17 L'AUC des modèles avec différentes méthodes de rééquilibrage des données

6.3 Modèle SMOTE basé sur des données équi-fusionnées

Un des objectifs de ce travail de recherche est d'examiner l'efficacité d'un modèle de coopération de données non redondantes pour améliorer les performances globales. Le modèle SMOTE proposé, basé sur des données équi-fusionnées, aborde efficacement le problème du déséquilibre des classes, comme le montre le Tableau 4.9. En termes de performances, ce

modèle surpasse les autres techniques, affichant des résultats impressionnants avec une Précision, un Rappel, et un Score F1 supérieurs à 95%, une AUC de 98,94 %, et un Taux de Faux Positifs de 1.97% (Figure IV.18).

Ces résultats mettent en évidence les avantages potentiels de la combinaison d'éléments provenant de différentes techniques de suréchantillonnage pour créer une solution plus complète et efficace pour identifier les classes minoritaires dans les ensembles de données éducatifs. En intégrant diverses approches, le modèle SMOTE équi-fusionné améliore la capacité à détecter les performances atypiques des étudiants, assurant ainsi une évaluation plus juste et nuancée. De plus, la stabilité du modèle balanced bagging observée dans les expériences souligne l'efficacité continue des techniques d'ensemble pour traiter les déséquilibres de classes dans les problèmes multi-classes. Cette approche, soutenue par les études de (Barros et al. 2019; Pristyanto et al. 2021), offre des prédictions satisfaisantes. Le balanced bagging, en combinant plusieurs modèles pour réduire la variance et augmenter la précision, s'avère particulièrement pertinent pour les contextes éducatifs où la fiabilité des prédictions est indéniablement fondamentale.

Ainsi, cette découverte souligne le potentiel de cette méthode pour l'évaluation éducative. En utilisant des modèles comme SMOTE équi-fusionné et des techniques d'ensemble, les éducateurs et les chercheurs peuvent développer des outils d'évaluation plus précis et équitables. Ces outils permettent non seulement d'identifier de manière plus fiable les étudiants appartenant aux classes minoritaires, mais aussi de fournir des interventions ciblées pour ceux qui en ont le plus besoin.

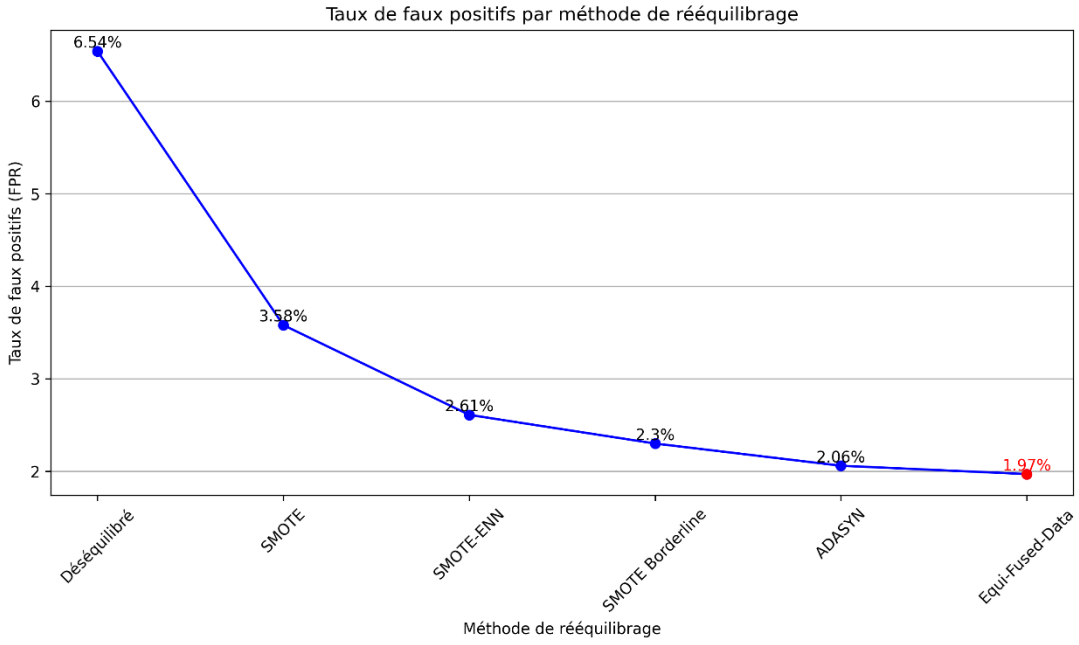

Figure IV.18 Analyse du taux de faux positifs par méthode de rééquilibrage

6.4 Implications Éducatives

Les résultats de cette étude mettent en lumière l'efficacité du modèle SMOTE basé sur des données équi-fusionnées, ainsi que l'importance des techniques de suréchantillonnage et des modèles de classification pour traiter le déséquilibre des classes dans les évaluations éducatives. Cela est particulièrement pertinent dans le contexte éducatif, où le déséquilibre des données peut entraîner une sous-représentation des groupes minoritaires, influençant ainsi les décisions pédagogiques et les résultats des étudiants.

En outre, les résultats soulignent la nécessité de ne pas se limiter à la seule mesure de l'exactitude pour évaluer les performances d'un modèle. En effet, d'autres métriques telles que le Rappel, le Taux de Faux Positifs (FPR), et l'AUC sont tout aussi critiques. Le Rappel, par exemple, mesure la capacité du modèle à identifier correctement les instances positives, ce qui est essentiel pour détecter les étudiants qui pourraient être en difficulté. De même, un faible Taux de Faux Positifs est capital pour éviter de classer à tort des étudiants comme nécessitant une intervention, ce qui pourrait gaspiller des ressources précieuses et générer du stress inutile.

Ces conclusions revêtent une importance considérable pour les EIAH et les plateformes d'e-learning. En adoptant des techniques de suréchantillonnage telles que Equi-Fused-Data-based SMOTE, ces systèmes peuvent améliorer la précision des outils d'évaluation et de prédiction des performances des étudiants. Cela peut mener à l'identification précoce des étudiants à risque, permettant ainsi des interventions ciblées. Par exemple, des programmes de soutien peuvent être mis en place pour les étudiants identifiés comme ayant besoin d'une aide supplémentaire, augmentant ainsi leurs chances de réussite.

De plus, l'application de ces techniques peut contribuer à une plus grande équité dans l'éducation en s'assurant que tous les étudiants, indépendamment de leurs antécédents ou de leurs performances antérieures, bénéficient des mêmes opportunités d'apprentissage et de soutien. En fin de compte, l'intégration de modèles de classification avancés et de techniques de suréchantillonnage dans le processus éducatif peut non seulement améliorer les résultats individuels des étudiants, mais également contribuer à une amélioration générale de la qualité de l'éducation. Ainsi, les résultats de cette étude ouvrent la voie à des pratiques éducatives plus informées et basées sur des données solides, permettant aux éducateurs de mieux répondre aux besoins diversifiés de leurs étudiants.

6.5 Limitations

Bien que le modèle SMOTE basé sur des données équi-fusionnées et la méthode de bagging équilibré aient montré des résultats prometteurs pour traiter le déséquilibre des classes dans les évaluations éducatives, certaines limitations importantes doivent être prises en compte. L'étude actuelle s'est concentrée sur un ensemble de données spécifique, centré sur un contexte éducatif particulier, notamment la performance algorithmique, et une population étudiante donnée. Cette spécificité géographique et contextuelle soulève des questions quant à la généralisation des résultats à d'autres contextes éducatifs, caractérisés par des populations étudiantes variées, des objectifs d'apprentissage multiples et des types d'évaluation diversifiés. Il est donc impératif de mener des investigations supplémentaires pour évaluer l'applicabilité de ces méthodes dans des environnements éducatifs différents.

La taille relativement limitée de l'ensemble de données utilisé (2176 instances) représente une contrainte significative, restreignant la portée des conclusions statistiques et appelant à une validation sur des ensembles de données plus vastes et diversifiés. Cette contrainte pourrait impacter la robustesse des résultats et limiter la capacité à généraliser les conclusions à des scénarios éducatifs nettement plus larges.

Les techniques de suréchantillonnage employées dans cette étude ont démontré leur efficacité, mais elles présentent également des limitations potentielles. Parmi celles-ci, l'introduction de biais dans les données. Cela peut affecter négativement la généralisation du modèle. En outre, l'efficacité de ces techniques varie considérablement en fonction de la distribution spécifique des données et des caractéristiques du déséquilibre des classes, rendant essentielle l'adaptation des méthodes de suréchantillonnage aux particularités de chaque ensemble de données. Dans cette optique, l'exploration d'approches alternatives, telles que l'apprentissage sensible au coût, pourrait offrir des perspectives intéressantes et enrichir la compréhension de leur efficacité dans

le contexte de l'évaluation éducative. Ces méthodes, en tenant compte des coûts associés aux erreurs de classification, pourraient fournir des solutions plus adaptées et équilibrées, améliorant ainsi la précision des évaluations. Par ailleurs, bien que le balanced bagging se soit révélé performant, il se distingue par son coût computationnel élevé par rapport aux techniques de suréchantillonnage plus simples, surtout lorsqu'il s'agit de traiter de grands ensembles de données. L'exploration de stratégies d'optimisation pour améliorer l'évolutivité de ces méthodes est donc décisive.

En définitive, il est important de noter que cette étude a principalement abordé les déséquilibres de classes par des techniques de suréchantillonnage. L'intégration de connaissances spécifiques au domaine, telles que les données démographiques des étudiants ou les profils cognitifs, pourrait significativement améliorer l'exactitude des modèles. En incorporant ces informations, il serait possible de développer des modèles d'évaluation plus adaptés aux besoins individuels des étudiants.

7 Conclusion

De nombreuses études ont révélé un taux d'échec alarmant dans le premier cours d'introduction à la programmation. Ce problème est exacerbé par le déséquilibre des classes dans les ensembles de données, rendant les modèles de classification conventionnels inadaptés. Cette étude aborde un problème réel de classification multi-classes en explorant diverses techniques d'échantillonnage pour améliorer les performances des modèles d'apprentissage automatique. Pour relever ce défi, plusieurs techniques d'échantillonnage ont été utilisées. Les résultats montrent que SMOTE-ENN et ADASYN offrent des performances particulièrement favorables en augmentant significativement la capacité des modèles à apprendre et à se généraliser. En particulier, SMOTE-ENN s'est distingué en préservant l'intégrité globale des données tout en améliorant les mesures de performance, telles que la précision, le rappel, le score F1, et l'AUC. En outre, l'utilisation de modèles tels que la régression logistique multinomiale, le SVM et la forêt aléatoire s'est avérée adéquate pour ce type de problème de classification multi-classes. Toutefois, il est capital de prendre en compte le temps de calcul et d'autres paramètres opérationnels pour optimiser l'application de ces modèles dans des contextes réels.

Une avancée notable de cette étude réside dans la proposition de combiner diverses données synthétiques non redondantes pour créer un ensemble de données fusionné. Cette coopération de données a démontré une amélioration significative de l'efficacité des modèles d'apprentissage dans le traitement des ensembles de données multi-classes. Les résultats expérimentaux ont révélé une augmentation de l'exactitude, de la précision, du rappel, du score F1 et de l'AUC, attestant du potentiel de cette technique pour améliorer les performances de classification.

Pour les travaux futurs, nous suggérons d'employer la technique de coopération des données pour recommander des ressources personnalisées ciblant chaque apprenant. En intégrant cette approche, il serait possible de fournir des interventions éducatives plus ciblées et efficaces, améliorant ainsi les résultats d'apprentissage des étudiants. Que ce soit dans un environnement présentiel ou informatisé, cette personnalisation pourrait répondre plus précisément aux besoins individuels des apprenants. En conclusion, les techniques d'échantillonnage avancées et la coopération des données présentent des perspectives prometteuses pour l'amélioration des modèles de classification dans les cours d'introduction à la programmation. Ces méthodes offrent non seulement une solution au problème du déséquilibre des classes, mais elles ouvrent également la voie à des applications éducatives plus personnalisées.

CHAPITRE V : EVALUER LA COMPREHENSION EN PROGRAMMATION : UNE APPROCHE BASEE SUR LES MODELES DE LANGAGES PRE-ENTRAINES

1 Introduction

Dans le domaine de l'éducation, la compréhension est un objet d'étude majeur, sujet à des recherches approfondies qui examinent ses multiples facettes et implications. La compréhension est fondamentale pour la résolution de problèmes, un aspect central de la réussite académique. En effet, la capacité à résoudre des problèmes complexes nécessite de mobiliser et d'appliquer des connaissances de manière flexible et créative. Plusieurs études empiriques (Akiguet-Bakong 2008; Lishinski et al. 2016; Sim et Lau 2018; Van Der Stel et Veenman 2014; Veerasamy et al. 2019) ont mis en évidence que les apprenants disposant de compétences solides en compréhension sont plus aptes à résoudre des problèmes et à appliquer leurs connaissances de manière efficace dans diverses disciplines. Ces compétences comprennent l'analyse critique, la synthèse d'informations et la capacité à établir des connexions entre des concepts disparates.

Les travaux de recherche susmentionnés ont démontré que les apprenants ayant une bonne compréhension tendent à obtenir de meilleures performances académiques. Par exemple, (Akiguet-Bakong 2008) souligne l'importance de la compréhension dans l'apprentissage des sciences, tandis que (Lishinski et al. 2016) mettent en lumière son rôle dans l'apprentissage des mathématiques. Les chercheurs (Sim et Lau 2018; Veerasamy et al. 2019) montrent que la compréhension favorise non seulement la résolution de problèmes, mais aussi l'innovation et la créativité, compétences essentielles dans le monde académique. Enfin, les auteurs (Van Der Stel et Veenman 2014) insistent sur l'importance de la métacognition dans le développement de la compréhension, suggérant que les élèves doivent non seulement apprendre à comprendre, mais aussi à réfléchir sur leur propre processus de compréhension.

Dans le cadre de l'enseignement de la programmation, la compréhension se définit comme la capacité à appréhender la logique intrinsèque de divers langages de programmation, tels que Python, R, Java, C/C++, entre autres. Cette compréhension va au-delà de la simple connaissance syntaxique; elle implique une immersion approfondie dans les principes fondamentaux des langages de programmation. Lors de la conception d'un programme, les apprenants doivent d'abord formuler une vision claire et exhaustive des objectifs et des exigences du problème à résoudre. Cette étape initiale est cruciale, car une mauvaise interprétation des besoins peut mener à des solutions inadéquates ou inefficaces. Ensuite, les apprenants doivent démontrer une capacité à décomposer des tâches complexes en sous-tâches gérables, un processus connu sous le nom de décomposition modulaire. Cette méthode permet de simplifier des problèmes complexes en unités plus petites et plus faciles à gérer, facilitant ainsi l'élaboration de solutions robustes. Cette étape exige une maîtrise des concepts fondamentaux de la programmation et une aptitude à penser de manière logique et structurée (Futschek 2006; Kanaki et al. 2022).

L'apprentissage de la programmation ne se limite pas à la maîtrise de la syntaxe d'un langage particulier. Il implique également le développement d'une pensée analytique pour concevoir des solutions logicielles robustes. Il est donc essentiel de mettre en place des activités pédagogiques adaptées à chaque étape du processus d'apprentissage du codage et de disposer d'un environnement de soutien adéquat pour faciliter ce processus. Ceci est particulièrement vrai dans le cadre de plateformes telles que l'apprentissage en ligne (Raunigr et Vajgl 2018), les jeux sérieux (Chiu et Huang 2015; Pietrikova, Juhar, et Stastna 2015) ou les systèmes de tutorat intelligents (Dermeval et Bittencourt 2020; Hooshyar et al. 2015; Trakosas, Tikva, et Tambouris 2023).

Ce qui précède corrobore la nécessité de systèmes éducatifs innovants dans les environnements d'apprentissage numériques, comme le souligne l'étude de (Tayoub et Chroqui 2023). D’ailleurs, l'IA joue un rôle prépondérant dans le développement de ces environnements

d'apprentissage. Elle permet de personnaliser l'expérience d'apprentissage en fonction des besoins spécifiques de chaque apprenant, en analysant les données relatives aux performances antérieures, aux préférences d'apprentissage et aux lacunes pour recommander un contenu adapté à chacun (Baylari et Montazer 2009; Haoran et al. 2019; Lin et al. 2013). Pour atteindre cet objectif, une évaluation précise s'avère être un levier crucial pour optimiser la réussite de l'apprentissage, tant sur les plateformes traditionnelles que sur les plateformes d'apprentissage en ligne (Rahaman et Hoque 2022).

Cette nécessité est particulièrement prégnante dans le domaine de la programmation, où les étudiants sont confrontés à des concepts complexes tout en développant leurs compétences pratiques. Malgré l'importance de ce rôle, l'évaluation précise de la compréhension du code par les étudiants demeure un défi. Des études telles que celles de (Abdessemed et al. 2018; Pillay et Vikash R. 2005; Watson et Li 2014) montrent que les apprenants novices éprouvent des difficultés à comprendre les instructions algorithmiques élémentaires. Ces difficultés peuvent être partiellement attribuées aux limites des méthodes d'évaluation traditionnelles, qui reposent souvent uniquement sur la précision de la syntaxe ou la fonctionnalité du code. Une telle approche ne parvient pas à saisir les nuances des constructions de programmation, en particulier lorsqu'il s'agit de distinguer entre différents concepts essentiels pour une évaluation complète de la compréhension du code. Cette insuffisance engendre des conséquences notables, freinant le développement des compétences et menant inévitablement à une diminution des performances académiques.

Pour combler cette lacune, nous avons utilisé des modèles de langage pré-entraînés (PLM) tels que BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), CodeBERT (Feng et al. 2020) et UniXcoder (Guo et al. 2022), en combinaison avec un classifieur Chains utilisant différents modèles tels que la forêt aléatoire, Gradient Boosting Machines, le perceptron multi-couches et la régression logistique etc. Ces PLM ont été spécifiquement entraînés sur de grands corpus de texte afin qu'ils puissent comprendre les subtilités de la structure du texte, de la syntaxe et de la sémantique. CodeBERT (Feng et al., 2020) et UniXcoder (Guo et al. 2022), quant à eux, sont des modèles de langage développés pour comprendre et générer du code dans différents langages de programmation. Ils sont pré-entraînés sur un large corpus de code pour améliorer les tâches liées au code, les rendant ainsi adaptés à l'analyse de similarité de code.

L'objectif de cette étude est d'évaluer comment les étudiants comprennent divers concepts de programmation, définis ici comme des étiquettes. Il est crucial d'identifier ces concepts de base pour mieux appréhender la compréhension du code, en particulier chez les apprenants novices, car ils sont les fondements des tâches de programmation plus complexes. Les étiquettes incluent: la déclaration de variables (Étiquette 1), la lecture de données (Étiquette 2), l'écriture de données (Étiquette 3), les opérations arithmétiques (Étiquette 4), et les instructions conditionnelles telles que if-else et switch (Étiquette 5).

Une nouvelle approche, appelée technique de fusion pondérée, est introduite pour améliorer la précision dans l'identification des constructions de programmation dans le code soumis. Cette méthode exploite les forces de différents modèles d'IA, pondérant la performance globale du modèle avec celle de chaque étiquette spécifique. Pour valider cette technique, un ensemble de données composé de 83 solutions d'étudiants à quatre problèmes algorithmiques distincts a été collecté. 64 étudiants de première année en science et technologie de l'université d'Annaba ont participé à l'étude. Un expert a ensuite évalué les solutions de codage produites par les étudiants pour déterminer leur compréhension de chaque concept algorithmique. Des techniques de traitement du langage naturel ont été utilisées pour prétraiter les données, en conjonction avec les PLM tels que BERT, RoBERTa, CodeBERT et UniXcoder, afin d'analyser et de quantifier la similarité entre les solutions des étudiants et celles des experts.

Cette recherche compare et contraste les performances de ces modèles de classification multi-étiquettes mettant en lumière les défis inhérents à la compréhension des différents concepts algorithmiques dans le code des étudiants. La technique de fusion pondérée introduite vise à surmonter les limites des modèles d'IA uniques en fusionnant leur précision pour différentes étiquettes, offrant une méthode d'évaluation plus précise. Nos objectifs de recherche sont les suivants :

- ✓ Analyser les performances de différents PLM dans des concepts spécifiques de compréhension du code.
- ✓ Évaluer l'efficacité de la technique de fusion pondérée pour améliorer la précision globale de l'évaluation.

Compte tenu de ces objectifs, la deuxième contribution de cette thèse est de démontrer l'efficacité des modèles spécifiques au domaine. Il est montré que UniXcoder et CodeBERT, des PLM spécifiquement conçus pour l'analyse de code, surpassent les modèles à usage général comme BERT et RoBERTa dans l'évaluation de la compréhension du code des étudiants. Ce constat souligne l'importance de prendre en compte les connaissances spécifiques au domaine lors de la construction de modèles d'apprentissage automatique pour les tâches liées au code.

Dans ce qui suit, nous discuterons de la méthodologie adoptée, des résultats obtenus, de la complexité des approches utilisées, des limites rencontrées et nous conclurons sur les implications des résultats obtenus.

2 Méthodologie

La figure suivante V.1 illustre la méthodologie adoptée pour prédire la compréhension nuancée des apprenants en matière de concepts algorithmiques. En premier lieu, les soumissions des étudiants aux exercices algorithmiques, ainsi que les solutions correctes fournies par l'expert, ont été recueillies et prétraitées grâce à des techniques avancées de NLP. À cette fin, plusieurs modèles de langage pré-entraînés, notamment BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), CodeBERT (Feng et al., 2020) et UniXcoder (Guo et al. 2022), ont été employés pour évaluer la similarité sémantique entre les soumissions des étudiants et les solutions expertes.

BERT (Devlin et al. 2019) a été sélectionné pour sa capacité à comprendre le contexte bidirectionnel des phrases, ce qui permet une analyse plus fine et nuancée des soumissions. RoBERTa (Liu et al. 2019), qui améliore BERT par des ajustements dans les paramètres de pré-entraînement, a été utilisé pour affiner encore la précision de la similarité sémantique. CodeBERT (Feng et al. 2020), spécialement conçu pour les langages de programmation, a permis de traiter les spécificités syntaxiques et sémantiques des codes soumis. UnixCoder (Guo et al. 2022), avec son modèle optimisé pour la compréhension du code informatique, a enrichi l'analyse en capturant les nuances et les structures complexes des soumissions de code. Son utilisation a permis d'augmenter la fiabilité des mesures de similarité sémantique.

Enfin, plusieurs classifieurs (Chains) ont été mis en œuvre, combinés à une approche de fusion pondérée pour classer les soumissions avec une grande précision. Cette approche de fusion pondérée intègre les résultats des différents modèles de langage et des classifieurs, en attribuant à chacun un poids proportionnel à sa pertinence et à sa performance. Ce processus permet de déterminer de manière plus exacte la compréhension des apprenants, en tenant compte des nuances subtiles dans leurs soumissions et en offrant une évaluation plus détaillée de leur maîtrise des concepts algorithmiques.

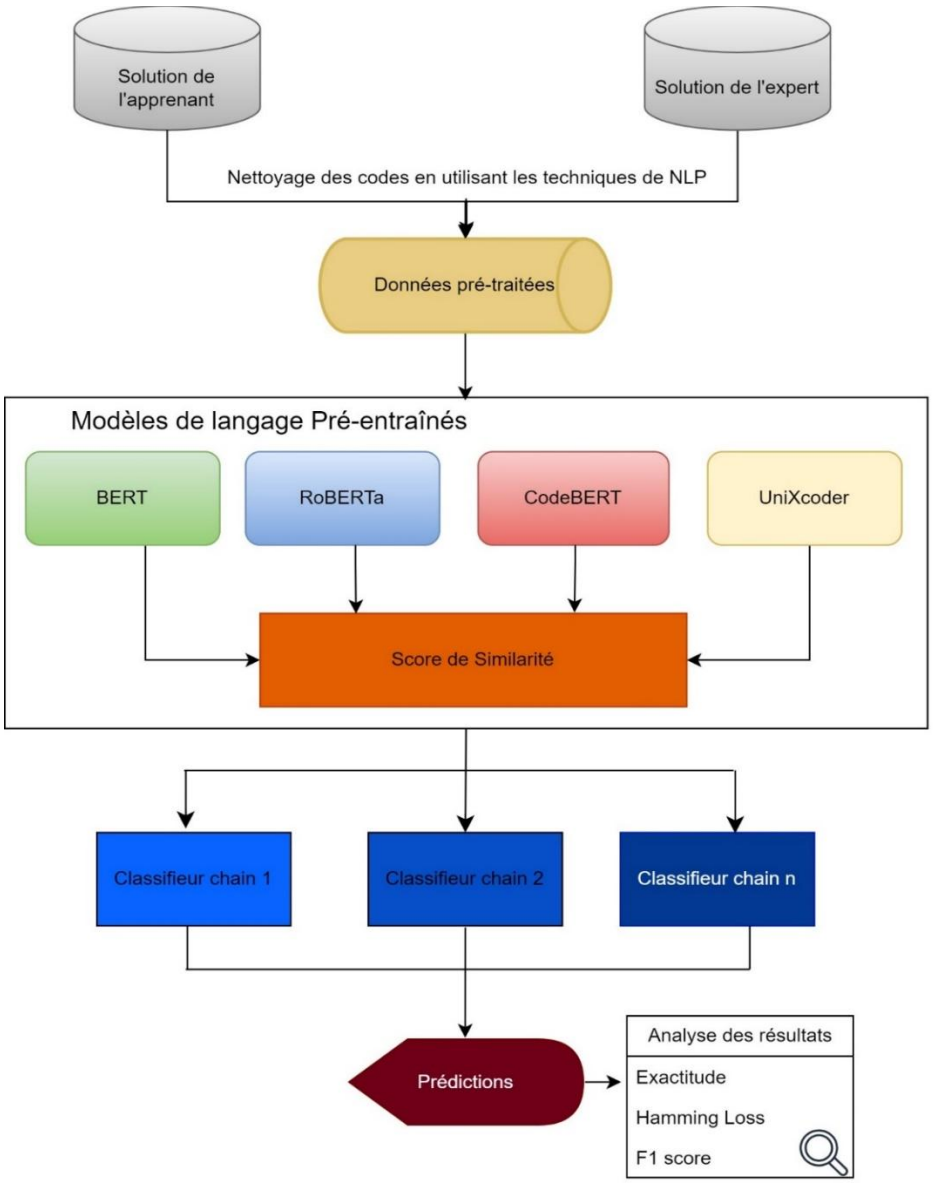

Figure V.1 Les Etapes de la Méthodologie Suivie

2.1 Collecte des données

Les données employées dans cette recherche proviennent des solutions aux exercices algorithmiques produits par les étudiants de première année en Sciences et Technologie (n=64) de l'Université d'Annaba, inscrits au cours introductif de programmation en C durant le premier semestre. Le groupe d'étudiants était composé de 31 hommes et de 33 femmes. Tous les participants étaient des débutants sans prérequis en informatique. Il est important de noter que les styles d'apprentissage individuels n'ont pas été pris en compte, en raison des débats en cours concernant leur réel impact sur l'apprentissage (Kirschner 2017; Nancekivell, Shah, et Gelman 2020; Wilkinson, Boohan, et Stevenson 2014).

Afin de prévenir toute forme de triche, les étudiants ont été répartis en deux groupes distincts, chacun recevant deux des quatre exercices proposés. Par conséquent, chaque étudiant n'était tenu de résoudre que deux des quatre exercices, ce qui a conduit à une disparité dans le nombre d'exercices tentés. De plus, un petit nombre d'étudiants n'ont tenté aucun exercice en raison de leurs compétences limitées en programmation, soulignant ainsi des niveaux variés d'engagement et de compétence parmi les participants.

Par ailleurs, les solutions soumises par les étudiants ont été collectées de manière anonyme et évaluées à l'aide d'une grille d'évaluation prédéfinie. Cette grille, notée de 0 (le plus bas) à 10 (le plus haut), évaluait non seulement la fonctionnalité et le respect des principes fondamentaux de la programmation, mais aussi le niveau global de compréhension (très faible, faible, moyen, bon). Afin d'atténuer les biais potentiels, un expert a anonymisé les évaluations en supprimant les noms et identifiants avant de procéder à la notation. L'évaluation s'est concentrée sur les compétences de base du premier semestre, telles que la déclaration et les types de variables, les opérations d'entrée/sortie, les opérateurs arithmétiques et logiques, ainsi que les instructions conditionnelles (if-else, switch). Pour simplifier l'évaluation, un système binaire a été utilisé pour enregistrer la maîtrise de chaque compétence par les étudiants en fonction de leurs solutions.

Le jeu de données résultant comprend 83 instances (lignes) et 11 caractéristiques, énumérées dans le Tableau 5.1, couvrant les quatre exercices conçus pour évaluer la compréhension des concepts if-else. L'objectif de cette recherche étant de comparer les solutions des étudiants à celles de l'expert à l'aide de modèles d'intelligence artificielle, et de prédire la compréhension de chaque concept algorithmique par les apprenants, cette étude se présente comme un problème de classification multi-étiquette.

Tableau 5.1 Description du jeu de données

Attribut	Description	Type	Valeur	Min	Max
Exercice	L'énoncé de l'exercice	Texte	Texte		
Compétence visée	Competence ciblée par l'exercice	Texte	Texte		
Solution Expert	Solution correcte proposée par l'expert	Texte	Texte		
Solution étudiant	Solution proposée par l'étudiant	Texte	Texte		
Score	Score obtenu par l'étudiant pour la solution	Numérique	0 to 10	0	10
Niveau de Compréhension	Compréhension globale	Ordinale	Très faible, faible, moyen, bon	-	-
Variable_Acquired	Maîtrise de la déclaration de variables par l'apprenant	Binaire	0,1	0	1
Read_Acquired	Maîtrise des opérations d'entrée par l' apprenant	Binaire	0,1	0	1
Write_Acquired	Maîtrise des opérations de sortie par l' apprenant	Binaire	0,1	0	1

Operators_Acquired	Maîtrise des opérateurs arithmétiques et logiques par l'apprenant	Binaire	0,1	0	1
If-else_Acquired	Maîtrise des instructions conditionnelles (if-else, switch) par l'apprenant	Binaire	0,1	0	1

2.2 Exploration des données

Le jeu de données utilisé présente un déséquilibre notable dans la distribution des étiquettes, certaines étiquettes ayant un nombre d'échantillons nettement supérieur à d'autres (par exemple, "variable_acquired" et "write_acquired"), comme le démontre le Tableau 5.2 et la Figure V.2 suivante.

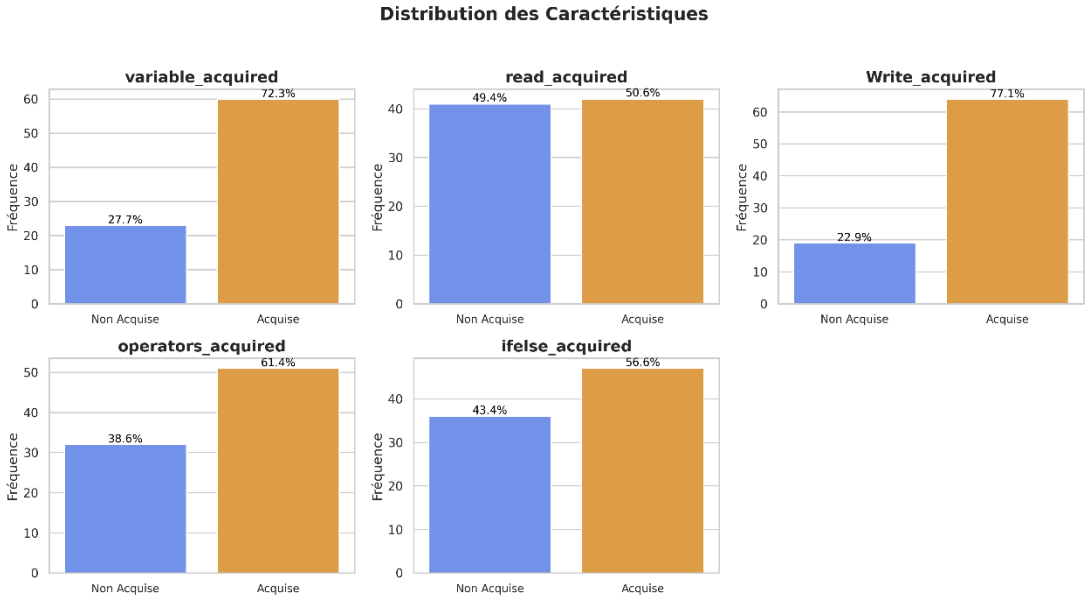

Figure V.2 Distribution des caractéristiques

Dans le cadre de la classification multi-étiquette, les options pour rééquilibrer les jeux de données sont limitées. Les méthodes existantes sont sujettes à controverse car elles risquent d'altérer la structure inhérente du jeu de données, ce qui pourrait entraîner la perte d'informations cruciales. En raison de ces préoccupations, cette étude a opté pour une évaluation des modèles sur le jeu de données original, et ce, malgré le déséquilibre. Cette approche vise à préserver l'intégrité et les relations naturelles au sein des données. En évitant les manipulations susceptibles de biaiser les résultats, l'étude assure que les modèles apprennent et opèrent sur des données reflétant fidèlement la réalité observée. Ainsi, bien que le déséquilibre des étiquettes pose des défis pour la modélisation et l'évaluation, il est important de maintenir les données dans leur état naturel. Cette démarche garantit que les modèles développés peuvent généraliser et fonctionner efficacement sur des données réelles, où des déséquilibres similaires sont souvent présents. De plus, cette approche permet de tester la robustesse et la capacité des modèles à gérer des distributions d'étiquettes variées, fournissant ainsi des résultats plus représentatifs et applicables à des situations réelles.

Tableau 5.2 Distribution des étiquettes dans un jeu de données multi-étiquette déséquilibré

Label	variable_acquired	read_acquired	write_acquired	operators_acquired	ifelse_acquired
1	60	42	64	51	47
0	23	41	19	32	36

2.3 Prétraitement du code

Diverses techniques de prétraitement du texte en NLP ont été utilisées pour nettoyer les codes (solutions de l'expert et celles des étudiants). Ce processus comprenait les étapes suivantes :

- Suppression des commentaires : Les commentaires simples et multi-lignes ont été supprimés du code à l'aide d'expressions régulières.
- Tokenisation et suppression des mots vides : La bibliothèque NLTK (nltk.tokenize) a été utilisée pour tokeniser les codes. Le code est découpé en tokens individuels à l'aide d'une expression régulière (r'\S+|\s'). Cette expression sépare le texte en correspondant soit à un ou plusieurs caractères non blancs, soit à des caractères blancs, assurant ainsi que les éléments de code et les espaces individuels soient capturés comme des tokens distincts, tout en prenant en compte des caractères de ponctuation importants tels que (, ; !), et les mots vides courants ont été supprimés durant ce processus.
- Conversion en minuscules : Tous les tokens sont convertis en minuscules pour assurer la cohérence.
- Préservation de la ponctuation essentielle : Seuls certains caractères de ponctuation et alphanumériques sont préservés, tels que (= + - * /), et les caractères non-mots sont supprimés du texte.
- Lemmatisation ou racinisation : En NLP, la racinisation est une technique utilisée pour réduire les mots à leur forme racine, impliquant la suppression des suffixes et des préfixes pour obtenir la forme racine. La lemmatisation, quant à elle, réduit les mots à leur forme de dictionnaire "lemme" en considérant le contexte du mot, assurant ainsi une précision linguistique. Dans cette étude, un lemmatiseur a été utilisé pour convertir les tokens en leurs formes de base pour le contenu des fonctions d'écriture uniquement. Cette amélioration a permis de réduire les variations de tokens tout en préservant la sémantique.
- Fusion des tokens : Les mots nettoyés et traités ont été fusionnés pour reconstruire le texte dans sa forme finale, prêt à être analysé.

2.4 Évaluation de la similarité du code

Pour évaluer la similarité sémantique entre les solutions de référence et celles produites par les étudiants, quatre modèles de langage pré-entraînés ont été utilisés: BERT, RoBERTa, CodeBERT et UniXcoder dont les caractéristiques et architectures respectives sont détaillées au chapitre 3 (sections 8.3 à 8.6).

En combinant ces quatre modèles, ce travail de recherche bénéficie d'une analyse riche et nuancée de la similarité sémantique, intégrant à la fois les aspects linguistiques et les spécificités des langages de programmation. Cela permet une évaluation plus précise et complète des solutions des étudiants par rapport aux solutions correctes, en tenant compte des subtilités contextuelles et syntaxiques de chaque soumission.

2.5 Mesure de la similarité sémantique basée sur BERT

Le modèle pré-entraîné BERT ("bert-base-uncased") est utilisé pour quantifier la similarité sémantique entre les soumissions des étudiants et les solutions correctes. Nous avons utilisé le BertTokenizer pour convertir les solutions prétraitées en représentations numériques et avons assuré des longueurs de comparaison cohérentes en remplissant les séquences plus courtes jusqu'à un maximum de 128 tokens. Pour tenir compte des solutions potentiellement plus longues, nous avons mis en œuvre une approche de segmentation. Nous avons défini une fonction, `get_bert_embedding`, qui génère des plongements sémantiques (embeddings) au niveau des phrases :

- Tokenisation : Utilisation du tokenizer pour convertir le texte en tokens numériques.
- Remplissage : Ajout de tokens spéciaux et remplissage des segments plus courts à la longueur souhaitée.
- Extraction des plongements sémantiques : Les gradients sont désactivés pour une inférence efficace, et le modèle BERT est utilisé pour extraire l'état caché final qui capture le sens encodé de chaque phrase.
- Moyenne des plongements sémantiques: Pour obtenir un seul plongement sémantique représentatif, la moyenne est calculée sur tous les plongements sémantiques de tokens dans la phrase.

Ensuite, une fonction est définie pour calculer la similarité, en itérant à travers les segments de l'étudiant et de la solution correcte et créant des comparaisons par paires. Pour chaque segment, la fonction suit ces étapes :

`Calcul de similarité (segments_apprenant, segments_expert)`

`Entrées :`

- `` `segments_apprenant` : Une liste contenant des segments de code étudiant. ``
- `` `segments_expert` : Une liste contenant les segments correspondants de la solution correcte. ``

`Sortie :`

- `Une liste contenant le score de similarité pour chaque paire segment segment_apprenant correct.`

`Algorithme`

`1. Initialiser une liste vide de similarités.`

`2. Pour chaque segment étudiant` `` `s` `` `dans` `` `segments_apprenant` ``
`:`

`- Extraire le plongement sémantique de` `` `s` `` `en utilisant` `` `get_bert_embedding(s)` ``, `noté` `` `s_emb` ``.

- `Pour chaque segment de solution correcte` `` `c` `` `dans` `` `segments_expert` ``
- `Obtenir le plongement sémantique de` `` `c` `` `en utilisant` `` `get_bert_embedding(c)` ``, `noté` `` `c_emb` ``.
- `Calculer la similarité cosinus entre` `` `s_emb` `` `et` `` `c_emb` ``
  `en utilisant la formule suivante :`

$$Cosine\ Similarity(s_{emb}, c_{emb}) = \frac{s_{emb} * c_{emb}}{(||s_{emb}|| * ||c_{emb}||)}$$

où `s_emb` et `c_emb` représentent des vecteurs n-dimensionnels, qui sont les sorties du modèle BERT pour le segment de code étudiant (`s_emb`) et le segment de solution correcte correspondant (`c_emb`). Cette formule calcule la similarité cosinus entre ces plongements sémantiques, capturant leur relation sémantique.

- Ajouter la similarité cosinus calculée à la liste `similarités`

3. Retourner la liste des similarités

Enfin, la similarité moyenne de toutes les comparaisons est calculée pour obtenir une valeur unique reflétant la proximité sémantique globale avec la solution correcte. La valeur de similarité cosinus varie entre -1 (dissemblable) et 1 (similaire).

De cette manière, grâce à l'architecture de BERT qui capture intrinsèquement la séquence et les relations contextuelles dans le texte, il est possible d'évaluer la similarité entre les solutions non seulement au niveau des mots, mais aussi de saisir des relations et des nuances sémantiques plus profondes. En incluant des segments qui se chevauchent, nous avons pu tenir compte des solutions potentiellement plus longues tout en assurant un traitement efficace. Pour gérer l'importance de la séquence en programmation, le code a été segmenté en morceaux significatifs. Le plongement sémantique de chaque segment conserve son contexte séquentiel, permettant à BERT de comprendre le flux et la logique du code. Cette approche assure que chaque segment est analysé avec une compréhension approfondie de son rôle dans l'ensemble du programme, facilitant ainsi une évaluation plus précise et contextuellement riche des solutions proposées par les étudiants.

Contrairement aux plongements sémantiques traditionnels, BERT génère des plongements sémantiques basés sur le contexte fourni par les tokens environnants. Cela signifie que l'ordre et les relations entre les tokens sont intégrés dans les représentations. De plus, le mécanisme d'auto-attention de BERT considère la position et la pertinence de chaque token par rapport à tous les autres tokens de la séquence. Cela garantit que la nature séquentielle et les dépendances dans la logique de programmation sont préservées.

2.6 Mesure de la similarité sémantique basée sur RoBERTa

Nous avons employé RobertaTokenizer et RobertaModel pour traiter et encoder les solutions prétraitées. À l'instar de BERT, nous avons rempli les séquences plus courtes jusqu'à un maximum de 128 tokens et adopté une approche de segmentation chevauchante pour gérer les solutions potentiellement plus longues. La fonction `get_roberta_embedding` a reproduit l'implémentation de BERT et a généré des plongements sémantiques au niveau des phrases en tokenisant, remplissant, extrayant l'état caché final avec les gradients désactivés, et calculant la moyenne des plongements sémantiques des tokens.

Pour comparer les segments des étudiants et des solutions correctes, nous avons défini une fonction qui itérait à travers les segments, obtenait les plongements sémantiques RoBERTa pour chaque paire, et calculait leur similarité cosinus. Les similarités individuelles étaient collectées et un score de similarité moyen était calculé pour chaque solution étudiante, reflétant leur proximité sémantique globale avec l'ensemble des solutions correctes. En combinant les scores de similarité basés sur RoBERTa avec ceux basés sur BERT, nous avons comparé les

performances de différents modèles et avons acquis des informations sur les relations sémantiques entre les solutions. Cela nous a permis d'explorer les forces et les faiblesses de chaque modèle dans la capture des nuances de sens et d'évaluer leur pertinence pour différentes tâches liées à la compréhension et à l'évaluation du code.

2.7 Mesure de la similarité sémantique basée sur CodeBERT

CodeBERT, un modèle de réseau neuronal pré-entraîné spécialement conçu pour la compréhension du code, a été utilisé pour évaluer la similarité sémantique entre les soumissions des étudiants et les solutions correctes. Nous avons exploité le modèle `microsoft/codebert-base` et son tokenizer associé pour convertir les solutions prétraitées en représentations numériques.

Pour garantir des longueurs de comparaison cohérentes, nous avons rempli les séquences plus courtes jusqu'à une longueur fixe de 246 tokens. Avec les gradients désactivés pour une inférence efficace, nous avons extrait les plongements sémantiques de l'état caché final du modèle, capturant le sens encodé de chaque solution. La similarité cosinus, une mesure de la distance angulaire entre les vecteurs, a ensuite été calculée entre les plongements sémantiques des solutions des étudiants et des solutions correctes correspondantes. Cela a donné un score de similarité unique pour chaque paire de solutions, quantifiant leur proximité sémantique. Enfin, les scores calculés ont été intégrés dans le cadre de données original avec les solutions prétraitées pour une analyse ultérieure.

Pour adapter le modèle CodeBERT pré-entraîné à notre tâche de prédiction de la similarité sémantique entre les solutions des étudiants et des experts, nous avons affiné le modèle en utilisant l'optimiseur AdamW et la fonction de perte d'erreur quadratique moyenne (Mean Squared Error). Le taux d'apprentissage a été fixé à 1e-5, la taille du lot à 16, et le modèle a été entraîné pendant 5 époques. Ces hyperparamètres ont été sélectionnés sur la base d'expériences préliminaires et d'observations empiriques pour optimiser les performances. L'affinage sur cet ensemble de données spécifique a permis au modèle d'apprendre des schémas et des nuances spécifiques à la tâche, améliorant ainsi sa performance dans la capture des similarités sémantiques de manière efficace.

2.8 Mesures de la similarité sémantique basée sur UniXCoder

Pour évaluer la similarité sémantique entre les soumissions des étudiants et les solutions correctes, nous avons utilisé UnixCoder. Le modèle `microsoft/unixcoder-base` et son tokenizer associé ont été utilisés pour convertir les solutions prétraitées en représentations numériques. Nous avons commencé par charger le modèle UnixCoder et le tokenizer associé. Les soumissions des étudiants et les solutions correctes, toutes deux prétraitées, ont été tokenisées en utilisant le tokenizer d'UnixCoder. Les séquences de tokens ont été remplies ou tronquées à une longueur fixe de 246 tokens pour garantir des comparaisons cohérentes. Avec les gradients désactivés pour une inférence efficace, nous avons extrait les plongements sémantiques de l'état caché final du modèle pour les solutions des étudiants et les solutions correctes. Cette étape a permis de capturer le sens encodé de chaque solution. La similarité cosinus a ensuite été calculée entre les plongements sémantiques des solutions des étudiants et les solutions correctes correspondantes. Les scores de similarité obtenus, représentant la proximité sémantique entre chaque paire de solutions, ont été ensuite intégrés dans le cadre de données original pour une analyse ultérieure.

2.9 Vérifications de la qualité des données

Trois vérifications de la qualité des données ont été effectuées :

- ❖ Valeurs manquantes : Aucune valeur manquante n'a été trouvée dans l'ensemble de données.
- ❖ Doublons : Les doublons ont été identifiés puis supprimés de l'ensemble de données (83 lignes réduites à 82 lignes).
- ❖ Randomisation : Les données ont été mélangées pour éviter les biais d'échantillonnage.

2.10 Chains Classifier pour la Classification Multi-label

Le classifieur Chains, connu pour son efficacité et son interprétabilité, pour aborder le défi de la classification multi-label, a été utilisé dans cette étude. Tout d'abord, l'ensemble de données a été divisé en deux ensembles : 80% pour l'entraînement et 20% pour le test. De plus, une validation croisée à 10 plis a été utilisée pour fournir une estimation fiable des performances du modèle. Les caractéristiques sélectionnées pour la classification multi-label représentent des caractéristiques extraites des codes prétraités des étudiants et des solutions d'experts, telles que le score moyen de similarité cosinus entre les segments de code des étudiants et les segments de solutions correctes, calculé en utilisant BERT, RoBERTa, CodeBERT et UniXcoder. En outre, le niveau de compréhension et le score obtenu sont également utilisés. Ces caractéristiques sont standardisées pour assurer une échelle uniforme et faciliter le processus de modélisation. La standardisation est effectuée en utilisant le StandardScaler de scikit-learn.

2.10.1 Implémentation et Réglage des Paramètres

L'implémentation s'est déroulée comme suit :

1. Représentation des caractéristiques
   Les caractéristiques ont été converties en matrices creuses pour un traitement efficace.
2. Préparation des variables cibles
   Les variables cibles ont été converties de séries Pandas en tableaux Numpy. Pandas est une bibliothèque Python puissante pour l'analyse de données, et une série est un objet unidimensionnel étiqueté similaire à un tableau. NumPy, une autre bibliothèque essentielle, fournit des tableaux multidimensionnels efficaces et des opérations mathématiques. La conversion des variables cibles en tableaux Numpy permet de tirer parti de ces opérations optimisées pour une analyse ultérieure.
3. Construction de la Chaîne de Classifieurs (CC) :
   - ❖ Un objet CC avec un classifieur de base choisi a été instancié pour chaque variable cible.
   - ❖ Divers classifieurs de base ont été mis en œuvre, tirant parti de leurs forces :
     - La forêt aléatoire pour sa robustesse et son interprétabilité, le nombre d'estimateurs étant fixé à 100 et le critère à Gini.
     - SVC pour les relations non linéaires, les paramètres par défaut étant utilisés.
     - Balanced bagging pour la gestion des données déséquilibrées, une forêt aléatoire étant implémentée avec le nombre d'estimateurs fixé à 100 et l'état aléatoire à 42.
     - kNN pour son efficacité, le nombre de voisins étant fixé à 5.
     - Gradient boosting machines pour une haute précision, une recherche par grille utilisée avec :
       - nombre d'estimateurs [50, 100, 150],
       - le taux d'apprentissage [0.01, 0.1, 0.5],
       - la profondeur maximale [3, 5, 7].
     - La régression logistique pour son interprétabilité, les hyperparamètres par défaut étant utilisés.

- MLP pour capturer les relations non linéaires et son adaptabilité. Les paramètres pour la recherche par grille étaient :
    - tailles des couches cachées [(100,), (50, 50), (50, 30, 20)],
    - les fonctions d'activation ['relu', 'tanh']
    - l'alpha (régularisation L2) à [0.0001, 0.001, 0.01]
    - état aléatoire fixé à 42 pour la reproductibilité.

4. Entraînement de la Chaîne : Chaque objet CC a été entraîné.
5. Prédiction et Conversion : Les prédictions sur le jeu de test ont été obtenues et converties en tableaux denses pour évaluation.

2.11 Évaluation des Modèles

Pour évaluer les modèles, nous avons utilisé trois métriques :

1. Précision globale basée sur les étiquettes: elle mesure la précision pour chaque étiquette indépendamment puis moyenne ces précisions. C'est une mesure plus douce qui donne un aperçu des performances du modèle sur une base individuelle pour chaque étiquette. Elle est calculée en prenant la précision moyenne de chaque étiquette sur l'ensemble des instances.
2. Perte de Hamming: cette métrique présentée en détail au chapitre 3 (Section 4), quantifie le nombre moyen de prédictions d'étiquettes incorrectes par exemple.
3. F1 Score: c'est la moyenne harmonique de la précision et du rappel, moyennée sur toutes les étiquettes. Elle fournit une évaluation équilibrée en considérant à la fois la précision (le ratio des prédictions positives correctes sur le nombre total de prédictions positives) et le rappel (le ratio des prédictions positives correctes sur toutes les vraies instances positives) pour chaque étiquette, pondérée par le nombre d'instances vraies pour chaque étiquette. Pour rappel, elle est calculée comme suit :

$$F1\ Score = 2 * \frac{(Précision * Rappel)}{(Précision + Rappel)}$$

3 Fusion Pondérée des Précisions Spécifiques aux Étiquettes

La démarche consistant à combiner divers modèles en fonction de leur précision spécifique aux étiquettes repose sur la compréhension que chaque algorithme possède ses propres forces et faiblesses en matière d'attribution précise des catégories pertinentes à chaque type de point de données. En classification multi-étiquettes, un point de données $x \in X$ peut appartenir simultanément à plusieurs catégories $Y \subseteq L$. En combinant différentes approches, nous visons à créer un système robuste avec des performances globales supérieures à toute méthode individuelle.

Mathématiquement, nous sélectionnons les k meilleurs modèles $Mk = \{m_1,m_2,\ldots,m_k\}$ à partir d'un ensemble de n modèles candidats $M = \{m_1,m_2,\ldots,m_n\}$ en fonction de leur précision globale A_i. L'analyse de la capacité de chaque algorithme à prédire des étiquettes spécifiques nous permet d'identifier les catégories qui sont plus difficiles à classer et de déterminer quels algorithmes excellent dans la reconnaissance de combinaisons particulières d'étiquettes. Cette perception aide à prioriser les prédictions des algorithmes qui sont efficaces pour attribuer des étiquettes pertinentes à chaque point de données. La fusion pondérée des précisions spécifiques aux étiquettes implique l'intégration des métriques de performance associées à différentes étiquettes à travers plusieurs modèles, comme illustré dans la Figure V.3.

Tout d'abord, les précisions spécifiques aux étiquettes sont collectées pour chaque modèle, où les étiquettes individuelles (Étiquette$_1$, Étiquette$_2$, …, Étiquete$_m$) sont associées aux précisions correspondantes obtenues à partir de différents modèles tels que la forêt aléatoire, la régression logistique, machines à gradient boosting et le balanced bagging. Ensuite, des précisions globales sont établies pour chaque modèle, signifiant la performance globale de ces modèles sur l'ensemble de l'ensemble d'étiquettes. Cette méthode est mathématiquement exprimée comme suit :

$$Précision\ Pondérée\ par\ Labely_i = \frac{\sum_{j=1}^{N} (A_{ij} * w_{j)}}{\sum_{j=1}^{N} w_j}$$

Où :

- Précision Pondérée de l'Étiquette$_i$ représente la précision pondérée pour l'Étiquette i.

- A_{ij} est la précision spécifique à l'étiquette pour l'Étiquette i obtenue à partir du modèle j.

- w_j représente la précision globale pour le modèle j.

- N est le nombre total de modèles.

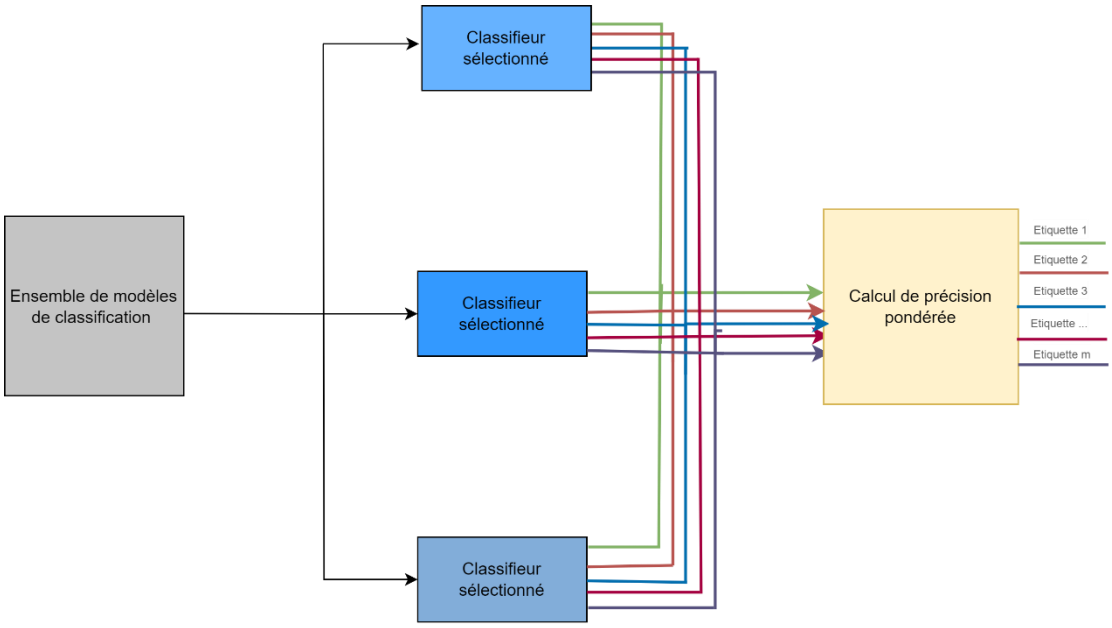

Figure V.3 Schéma Illustratif de la Fusion Pondérée des Précisions Spécifiques aux Etiquettes

- La précision spécifique à l'étiquette A_{ij} fait référence à la précision avec laquelle un modèle particulier j prédit une étiquette spécifique i. Elle est mesurée comme la proportion de prédictions correctes pour cette étiquette parmi toutes les instances où l'étiquette est présente. Dans ce contexte, "prédire" signifie la capacité d'un modèle à attribuer correctement une ou plusieurs étiquettes à un point de données donné en fonction des motifs appris à partir des données d'entraînement.
- La précision globale w_j représente la précision globale d'un modèle sur toutes les étiquettes et tous les points de données, reflétant sa performance générale.

Théoriquement, l'information mutuelle peut soutenir cette approche en mesurant les relations entre différentes étiquettes, aidant à comprendre quelles étiquettes sont souvent prédites ensemble. L'information mutuelle quantifie dans quelle mesure la connaissance d'un label réduit l'incertitude sur une autre, identifiant ainsi les dépendances entre les étiquettes. Cette compréhension permet la conception de stratégies qui se concentrent sur des groupes spécifiques d'étiquettes (Wang, Zhao, et Xu 2018). De plus, l'information mutuelle permet de décomposer les erreurs de prédiction, révélant quelles étiquettes posent le plus de problèmes et quels algorithmes ont des difficultés avec des étiquettes spécifiques. Cette perception aide à améliorer l'approche combinée en se concentrant sur les étiquettes difficiles et en affinant les algorithmes pour de meilleures performances.

3.1 Implémentation et Hypothèses

Le calcul de la fusion pondérée commence par prendre en considération les précisions spécifiques aux étiquettes pour chaque modèle. Chaque précision spécifique à l'étiquette est pondérée proportionnellement à la précision globale de son modèle respectif pour cette étiquette spécifique. Ces poids sont déterminés en multipliant chaque précision spécifique à l'étiquette par la précision globale correspondante du modèle. Le point culminant implique la sommation des résultats pondérés de tous les modèles, formant ainsi une somme pondérée. Enfin, la somme pondérée est divisée par l'agrégat des précisions globales de tous les modèles, donnant ainsi la précision pondérée résultante pour chaque étiquette.

Une préoccupation est la dépendance à l'égard des précisions des échantillons précédents, qui peuvent ne pas nécessairement impliquer la même précision sur l'échantillon actuel. Pour atténuer cela, notre méthodologie suppose que la performance passée est un proxy raisonnable pour la performance future, nous permettant d'utiliser les précisions historiques pour informer les prédictions actuelles. Cependant, les modèles doivent être continuellement mis à jour avec de nouvelles données pour garantir que les précisions pondérées restent pertinentes et précises. Ainsi, cette méthodologie nous permet de bénéficier des performances individuelles des modèles concernant des étiquettes spécifiques, en utilisant les précisions globales des modèles comme facteurs de pondération pour dériver une estimation de précision complète et pondérée pour chaque étiquette.

Dans la section suivante, nous discuterons des résultats obtenus, en examinant en détail l'efficacité de notre approche de fusion pondérée et son impact sur la performance globale du modèle.

4 Résultats

Nous avons utilisé un ensemble de classifieurs pour le modèle Chains dans le cadre de la tâche de classification multi-étiquettes visant à évaluer la compréhension du code des élèves. Les modèles sélectionnés incluent la forêt aléatoire, le balanced bagging, le classifieur de vecteurs de support (SVC), kNN, GBM, la régression logistique et le MLP.

Chaque étiquette représente un concept spécifique : déclaration de variable acquise (Label 1), lecture acquise (Label 2), écriture acquise (Label 3), opérations acquises (Label 4) et déclarations conditionnelles acquises (Label 5), telles que les structures if-else et switch. Les performances des modèles de langage pré-entraînés sont détaillées ci-dessous.

4.1 Résultats basés sur les similarités de BERT

Les modèles sélectionnés ont donné des résultats significatifs lors de l'utilisation des similarités de BERT. Le classifieur Chains utilisant la forêt aléatoire et les machines à gradient

comme modèles de base a montré des résultats prometteurs dans diverses acquisitions de concepts. Ces modèles ont atteint une précision globale et un score F1 de 0,84 et un hamming loss de 0.15, respectivement. En particulier, le modèle de forêt aléatoire a présenté une performance significative pour les étiquettes d'écriture et de déclaration conditionnelle avec des précisions de 0,94 et 0,88, respectivement. En revanche, bien que le modèle avec les machines à gradient ait également montré une performance significative dans la prédiction des étiquettes de déclaration de variable et de déclaration conditionnelle avec des précisions de 0,88 chacune, il a montré une précision comparativement plus faible dans l'identification de l'étiquette d'opérations à 0,70, accompagnée d'un hamming loss élevée de 0,29.

De plus, le modèle kNN en tant que base pour le classifieur Chains, ainsi que le balanced bagging et le SVM ont montré des performances compétitives avec des précisions globales de 0,82 et 0,83, respectivement. En particulier, la performance des kNN dans l'identification de l'étiquette d'écriture avec une précision de 0,94, indiquant sa capacité à reconnaître plusieurs étiquettes simultanément avec des erreurs de prédiction minimales, comme en témoigne le faible taux de hamming loss. De plus, le modèle SVM a prédit les étiquettes de lecture, d'écriture et d'opérations avec des précisions de 0,82, 0,94 et 0,88, respectivement, bien qu'il ait présenté une précision plus faible dans la prédiction des déclarations conditionnelles par rapport aux autres modèles.

Par ailleurs, la régression logistique en tant que base pour le classifieur Chains a atteint une précision globale modérée de 0,81. Bien que le modèle montre une bonne performance dans la prédiction des étiquettes d'écriture et d'opérations avec des précisions de 0,94 et 0,82, respectivement, il rencontre des défis dans la prédiction des déclarations conditionnelles. En revanche, le MLP a moins bien performé par rapport aux autres modèles, atteignant une précision globale de 0,80. Malgré sa capacité à identifier les déclarations d'écriture et conditionnelles avec des précisions de 0,94 et 0,82, respectivement, il a eu du mal à prédire l'étiquette de lecture, avec une précision et un score F1 approchant de 0,65. Le Tableau 5.3 ci-dessous fournit une analyse comparative détaillée des performances des différents modèles basés sur les similarités de BERT.

Tableau 5.3 Analyse Comparative des différents modèles de classification basée sur les similarités BERT

Modèle	Métrique	Label1	Label2	Label3	Label4	Label5	Global
Forêt aléatoire	Précision	0.88	0.76	0.94	0.76	0.88	0.84
	Hamming loss	0.11	0.23	0.05	0.23	0.11	0.15
	F1 score	0.88	0.76	0.93	0.76	0.88	0.84
Balanced Bagging	Précision	0.70	0.76	0.94	0.76	0.94	0.82
	Hamming loss	0.29	0.23	0.05	0.23	0.05	0.17
	F1 score	0.71	0.75	0.94	0.76	0.94	0.81
SVM	Précision	0.76	0.82	0.94	0.88	0.76	0.83
	Hamming loss	0.23	0.17	0.05	0.11	0.23	0.16
	F1 score	0.76	0.80	0.94	0.88	0.75	0.83

kNN	Précision	0.76	0.82	0.94	0.82	0.76	0.82
	Hamming loss	0.23	0.17	0.05	0.17	0.23	0.17
	F1 score	0.76	0.80	0.93	0.81	0.76	0.81
Gradient Boosting Machines	Précision	0.88	0.82	0.94	0.70	0.88	0.84
	Hamming loss	0.11	0.17	0.05	0.29	0.11	0.15
	F1 score	0.88	0.82	0.93	0.70	0.88	0.84
Régression Logistique	Précision	0.76	0.76	0.94	0.82	0.76	0.81
	Hamming loss	0.23	0.23	0.05	0.17	0.23	0.18
	F1 score	0.76	0.75	0.93	0.81	0.75	0.80
MLP	Précision	0.88	0.64	0.94	0.70	0.82	0.80
	Hamming loss	0.11	0.35	0.05	0.29	0.17	0.20
	F1 score	0.88	0.65	0.93	0.70	0.82	0.79

Pour les différents classifieurs utilisés, les résultats suggèrent une précision significative dans l'identification de différents concepts de programmation. Tant la forêt aléatoire que les machines à gradient ont atteint une précision globale de 0,84, démontrant leur efficacité dans l'identification de plusieurs étiquettes. Cependant, certains classificateurs ont rencontré des difficultés avec certaines étiquettes, tandis que d'autres ont montré une performance constante. Par exemple, le MLP et la régression logistique ont eu des difficultés avec l'étiquette de lecture. Il existe également une variabilité dans l'identification de l'étiquette des déclarations conditionnelles, suggérant que prédire les déclarations conditionnelles dans le code d'un étudiant est difficile, quel que soit le classificateur utilisé.

De manière intéressante, l'invariance de la précision pour l'étiquette d'écriture à travers différentes méthodes peut être due à plusieurs facteurs (Figure V.4). Cette cohérence pourrait découler du déséquilibre du jeu de données, où l'étiquette d'écriture pourrait être surreprésentée, ce qui faciliterait l'apprentissage et la prédiction précise des modèles. De plus, le biais du modèle et l'utilisation de caractéristiques similaires dérivées de BERT pourraient conduire à une précision uniformément élevée, car tous les modèles pourraient exploiter un espace de caractéristiques partagé qui est particulièrement efficace pour identifier les opérations d'écriture.

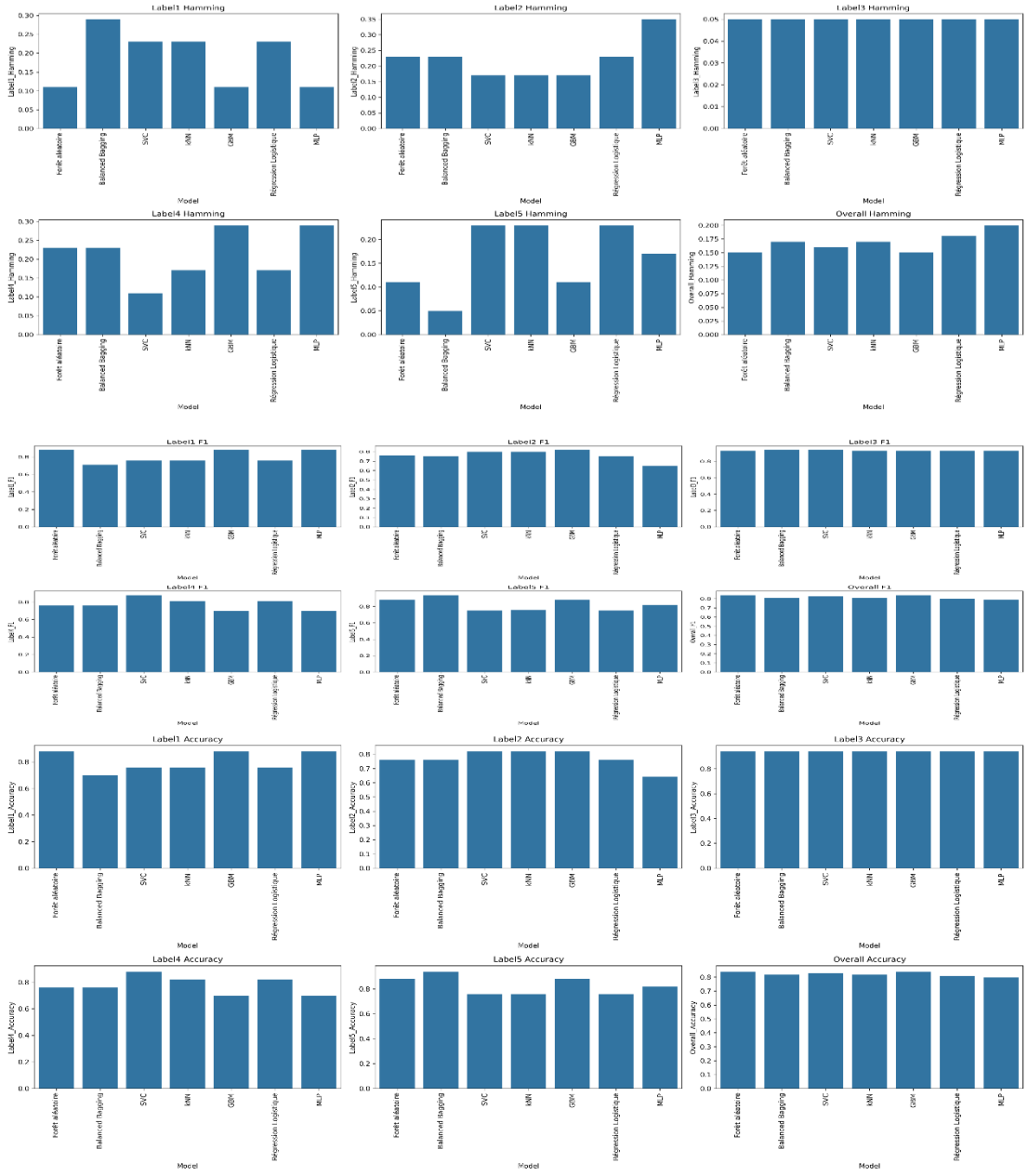

Figure V.4 Analyse comparative de différents modèles basée sur les similarités BERT

Les boîtes à moustaches (Figure V.5) illustrent la performance des différents modèles sur différentes étiquettes en termes de scores F1 et de précision. Pour les scores F1, l'étiquette d'écriture (Label3) montre la moins grande variabilité, avec une plage stable de 0,83 à 0,90, tandis que les étiquettes de variable et de déclarations conditionnelles (Labels 1 et 5) présentent des scores médians plus élevés autour de 0,80. La lecture (Label2) a la plage la plus large, indiquant une plus grande variabilité de la performance. Les scores F1 globaux varient de 0,76 à 0,85, avec une médiane autour de 0,80. En termes de précision, des schémas similaires sont observés : l'écriture (Label3) reste la plus cohérente avec une plage étroite, tandis que les étiquettes de variable et de déclarations conditionnelles (Labels 1 et 5) présentent des précisions médianes plus élevées. La lecture (Label2) inclut une valeur aberrante (outlier) près de 0,65, indiquant une performance occasionnellement moins bonne (modèle MLP). Les valeurs aberrantes sont des points de données qui tombent en dehors de 1,5 IQRs du quartile supérieur ou inférieur. La précision globale varie également de 0,76 à 0,85, reflétant les schémas des scores F1.

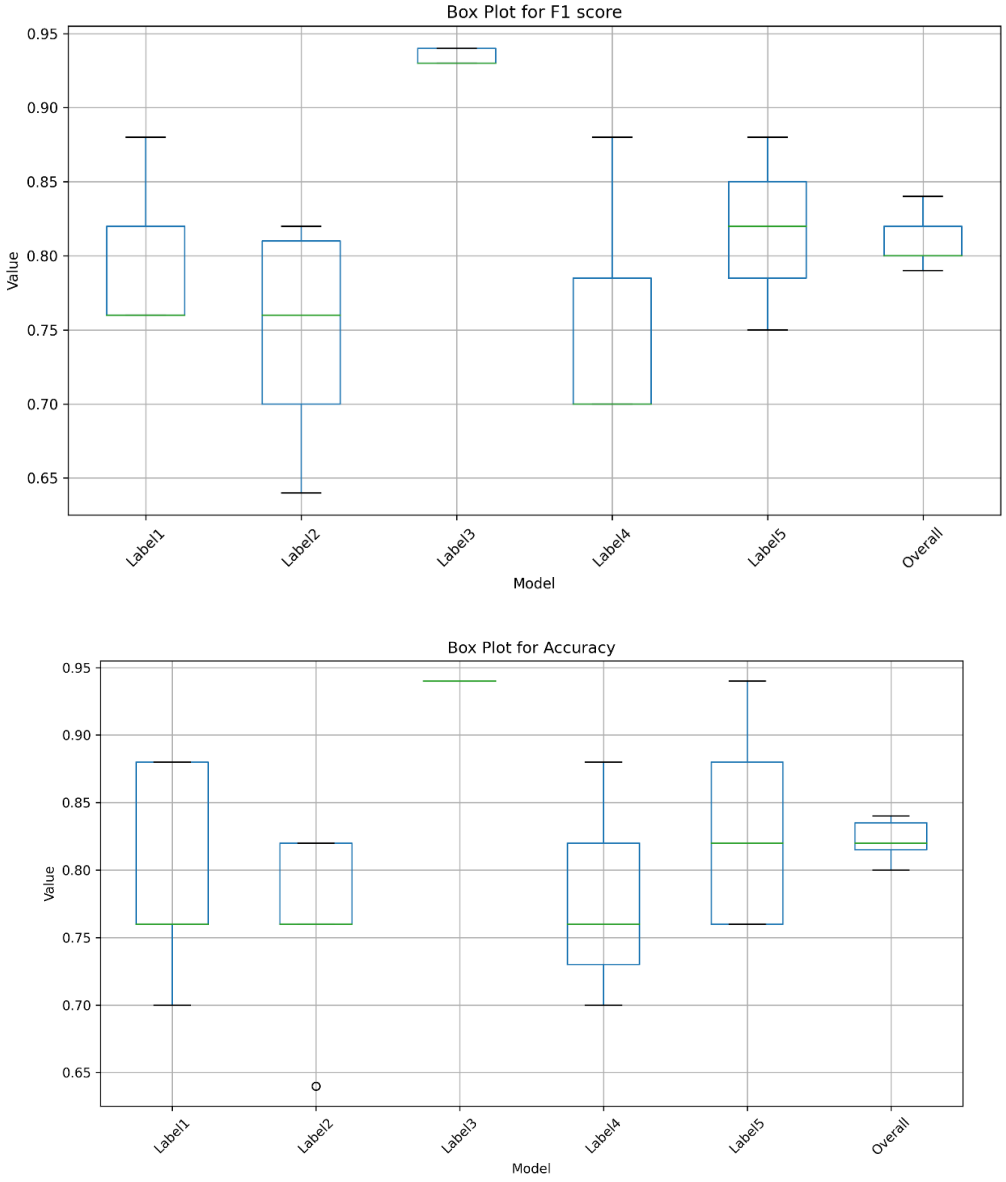

Figure V.5 Boites à moustaches de la précision et le f1 score des différents modèles basées sur les similarités BERT

4.2 Résultats basés sur les similarités de RoBERTa

En se basant sur les similarités issues du PLM RoBERTa, le modèle des machines à gradient boosting s'est révélé être le plus performant, avec une précision globale de 0,89 et la plus faible perte de Hamming de 0,1. Le modèle a montré une performance significative dans l'identification des étiquettes de déclaration de variable et des déclarations conditionnelles, avec une précision de 0,94. Cependant, il a rencontré des difficultés avec l'étiquette de lecture. En revanche, la forêt aléatoire a atteint une précision globale de 0,84. Bien que le modèle ait efficacement prédit les déclarations conditionnelles avec une précision de 0,88 et les opérations avec une précision de 0,87, sa performance pour la déclaration de variable était légèrement inférieure à 0,76. Cette différence pourrait être due au processus d'apprentissage itératif des machines de gradient, leur permettant potentiellement de capturer des relations plus complexes entre les caractéristiques par rapport à l'approche de moyenne de la forêt aléatoire.

Tant le SVM que le kNN ont atteint des précisions globales similaires autour de 0,83. Le SVM a montré de bonnes performances pour l'étiquette de lecture avec une précision de 0,82 et l'étiquette d'opération avec un score F1 de 0,88, mais a eu du mal avec les déclarations

conditionnelles similaires à d'autres modèles. En revanche, le kNN a affiché une performance significative dans la prédiction des déclarations conditionnelles et des étiquettes de lecture. Cela suggère que la dépendance du kNN à trouver les voisins les plus proches dans l'espace de similarité de RoBERTa pourrait être particulièrement efficace pour ces concepts spécifiques. Alors que le modèle SVM peut identifier des frontières claires entre les classes, le kNN pourrait être mieux adapté aux concepts avec des caractéristiques chevauchantes ou des représentations plus nuancées dans l'espace de représentation RoBERTa.

Tout comme les résultats de BERT, la régression logistique et le MLP ont sous-performé pour l'étiquette de déclaration conditionnelle. La régression logistique a atteint une précision globale modérée de 0,82 mais a eu du mal avec la déclaration conditionnelle. Le MLP, tout en prédisant bien l'étiquette d'écriture et la déclaration conditionnelle avec une précision de 0,82, a eu des difficultés avec l'étiquette de lecture. Cela peut être dû à la nature déséquilibrée de l'ensemble de données. Le balanced bagging a également montré des limites, en particulier pour les étiquettes d'écriture et d'opérations avec une précision de 0,74 et 0,81, respectivement. Cela renforce l'idée que les modèles linéaires comme la régression logistique pourraient ne pas être idéaux pour capturer des relations complexes entre les concepts de code, en particulier pour des tâches comme la prédiction des déclarations conditionnelles. L'architecture ou le processus d'entraînement du MLP pourrait nécessiter une optimisation supplémentaire pour de meilleures performances sur l'étiquette de lecture. D'autre part, les limitations du balanced bagging pourraient éventuellement être liées à la méthode d'ensemble spécifique utilisée et aux modèles sous-jacents qu'elle combine.

Tableau 5.4 Analyse Comparative des différents modèles de classification basée sur les similarités RoBERTa

Modèle	Métrique	Label1	Label2	Label3	Label4	Label5	Global
Forêt aléatoire	Précision	0.76	0.76	0.94	0.88	0.88	0.84
	Hamming loss	0.23	0.23	0.05	0.11	0.11	0.15
	F1 score	0.76	0.76	0.93	0.87	0.87	0.84
Balanced Bagging	Précision	0.76	0.76	0.88	0.76	0.82	0.80
	Hamming loss	0.23	0.23	0.11	0.23	0.17	0.20
	F1 score	0.76	0.75	0.88	0.74	0.81	0.79
SVM	Précision	0.76	0.82	0.94	0.88	0.76	0.83
	Hamming loss	0.23	0.17	0.05	0.11	0.23	0.16
	F1 score	0.76	0.80	0.94	0.88	0.75	0.83
kNN	Précision	0.76	0.82	0.94	0.82	0.82	0.83
	Hamming loss	0.23	0.17	0.05	0.17	0.17	0.16
	F1 score	0.76	0.80	0.93	0.81	0.83	0.82
	Précision	0.82	0.88	0.94	0.88	0.94	0.89

Gradient Boosting Machines	Hamming loss	0.17	0.11	0.05	0.11	0.05	0.10
	F1 score	0.82	0.88	0.93	0.87	0.94	0.89
Régression Logistique	Précision	0.76	0.82	0.94	0.82	0.76	0.82
	Hamming loss	0.23	0.17	0.05	0.17	0.23	0.17
	F1 score	0.76	0.81	0.94	0.81	0.75	0.82
MLP	Précision	0.88	0.65	0.94	0.70	0.82	0.8
	Hamming loss	0.11	0.35	0.05	0.29	0.17	0.2
	F1 score	0.88	0.64	0.93	0.70	0.82	0.79

Les performances des différents modèles utilisant les similarités de RoBERTa se sont avérées comparables à celles de BERT, permettant une bonne compréhension de certaines étiquettes. Cependant, des difficultés subsistent dans la prédiction précise des déclarations conditionnelles et des étiquettes de lecture. Par ailleurs, certains classifieurs ont montré des performances stables, alors que d'autres, tels que la régression logistique et le MLP, ont eu des difficultés à prédire certaines étiquettes.

Notamment, la Figure V.6 montre que la précision de l'étiquette d'écriture reste élevée à travers différentes méthodes, bien qu'elle ne soit pas toujours constante à 100%. Cette performance cohérente pourrait s'expliquer par un déséquilibre dans l'ensemble de données, où l'étiquette d'écriture serait surreprésentée, facilitant ainsi les prédictions du modèle.

La Figure V.7 illustre les scores F1 et les métriques de précision pour différents modèles utilisant les similarités RoBERTa à travers différentes étiquettes. Pour les scores F1, la variable (Label 1) varie d'environ 0,75 à 0,85, avec une médiane autour de 0,80 et une valeur aberrante proche de 0,90. La lecture (Label 2) montre une plage plus large de 0,65 à 0,85, avec une médiane autour de 0,75 et une valeur aberrante proche de 0,65. En revanche, l'écriture (Label 3) démontre une grande cohérence, avec des scores F1 variant de 0,83 à 0,90 et une médiane d'environ 0,86. Les scores F1 de l'opération (Label 4) varient de 0,70 à 0,85, avec une médiane autour de 0,77, indiquant une variabilité modérée. Enfin, les déclarations conditionnelles (Label 5) montrent des scores allant de 0,75 à 0,85, avec une médiane autour de 0,80 et une valeur aberrante proche de 0,65. Pour la précision, des tendances similaires sont observées. La précision globale varie de 0,80 à 0,84, reflétant les motifs des scores F1.

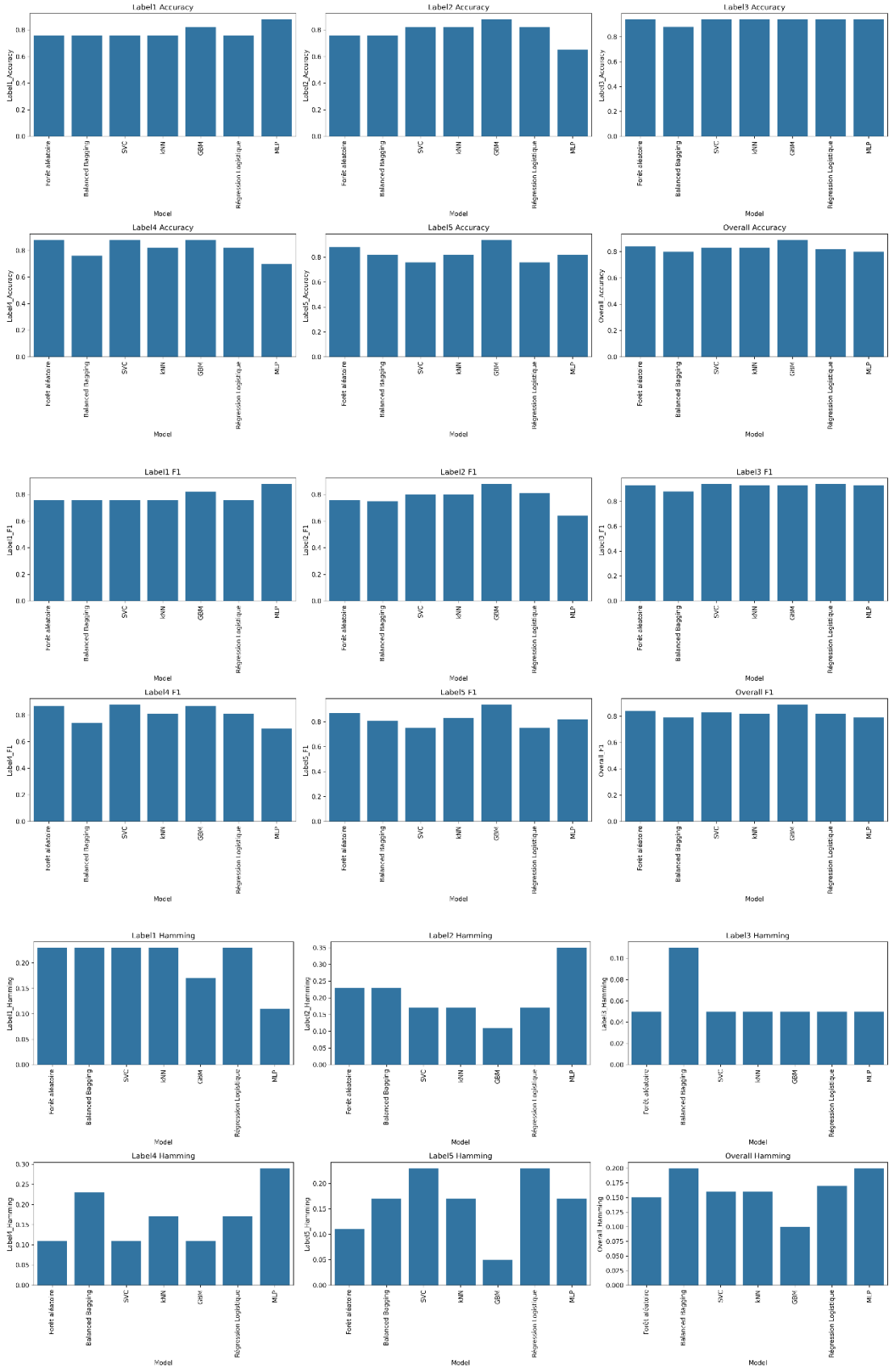

Figure V.6 Analyse comparative de différents modèles basée sur les similarités RoBERTa

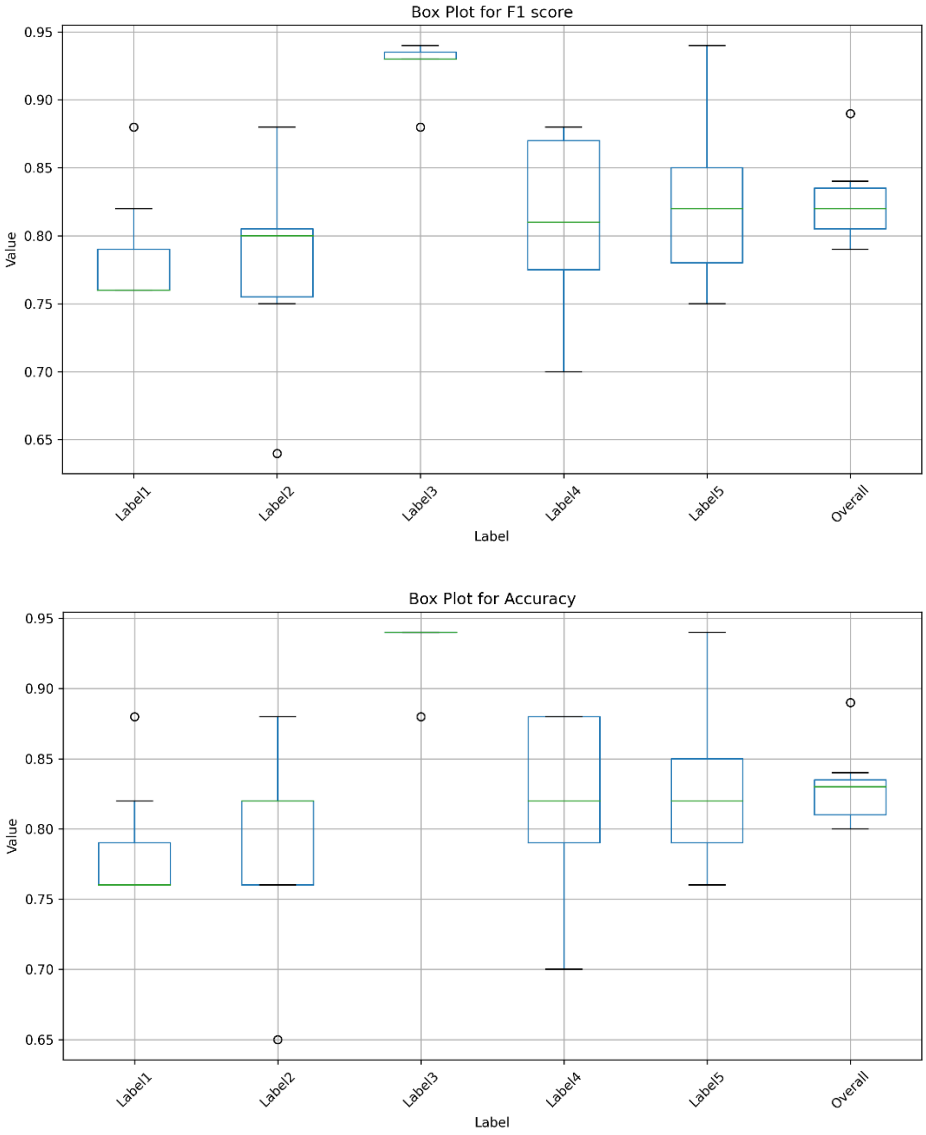

Figure V.7 Boites à moustaches de la précision et le f1 score des différents modèles basées sur les similarités RoBERTa

4.3 Résultats basés sur les similarités de CodeBERT

L'utilisation des similarités issues de CodeBERT a donné des résultats prometteurs. À la fois la forêt aléatoire et le balanced bagging ont obtenu des résultats impressionnants. La forêt aléatoire s'est imposée comme le meilleur modèle avec une précision globale et un score F1 de 0,88. Le modèle a montré une performance efficace dans la prédiction de la plupart des étiquettes, en particulier la déclaration de variables et les déclarations conditionnelles avec une précision de 0,94 pour les deux. Le balanced bagging a suivi de près avec une précision globale de 0,87 et un score F1 de 0,86. Bien que les deux aient eu des difficultés légères avec l'étiquette de lecture (sous-représentée dans l'ensemble de données), le modèle balanced bagging avait un avantage significatif pour la déclaration de variables avec une précision significative de 0,99, potentiellement en raison de la capacité de la méthode d'ensemble à exploiter les forces de plusieurs modèles sous-jacents.

En revanche, à la fois kNN et les machines à gradient ont montré une précision globale modérée, autour de 0,83 mais avec des forces et des faiblesses intéressantes. kNN a atteint une précision exceptionnelle pour l'étiquette d'écriture de 0,99, suggérant son efficacité dans la recherche des voisins les plus proches dans l'espace de similarité CodeBERT pour ce concept

spécifique. Cependant, le modèle a eu des difficultés avec la déclaration conditionnelle. A l'inverse, le modèle des machines à gradient a montré des résultats bons pour la déclaration conditionnelle avec une précision de 0,88, mais a eu des difficultés avec l'étiquette de lecture.

Comme pour les expériences précédentes, SVM, la régression logistique et MLP ont présenté une performance globale modérée autour de 0,82 avec des limitations pour des étiquettes spécifiques. Cela renforce l'idée que les modèles linéaires comme la régression logistique pourraient ne pas être idéaux pour les relations complexes dans le code, en particulier pour des tâches comme la prédiction de déclarations conditionnelles. Les résultats sont détaillés dans le tableau 5.5.

Tableau 5.5 Analyse Comparative des différents modèles de classification basée sur les similarités CodeBERT

Modèle	Métrique	Label1	Label2	Label3	Label4	Label5	Global
Forêt aléatoire	Précision	0.82	0.82	0.94	0.88	0.94	0.88
	Hamming loss	0.17	0.17	0.05	0.11	0.05	0.11
	F1 score	0.82	0.81	0.93	0.88	0.94	0.88
Balanced Bagging	Précision	0.82	0.82	0.99	0.82	0.88	0.87
	Hamming loss	0.17	0.17	0.01	0.17	0.11	0.12
	F1 score	0.83	0.80	0.99	0.81	0.87	0.86
SVM	Précision	0.76	0.82	0.94	0.82	0.76	0.82
	Hamming loss	0.23	0.17	0.05	0.17	0.23	0.17
	F1 score	0.76	0.80	0.94	0.81	0.75	0.81
kNN	Précision	0.82	0.70	0.99	0.76	0.88	0.83
	Hamming loss	0.17	0.29	0.01	0.23	0.11	0.16
	F1 score	0.81	0.67	0.99	0.75	0.88	0.82
Gradient Boosting Machines	Précision	0.76	0.88	0.94	0.70	0.88	0.83
	Hamming loss	0.23	0.11	0.05	0.29	0.11	0.16
	F1 score	0.76	0.88	0.93	0.70	0.88	0.83
Régression Logistique	Précision	0.76	0.82	0.94	0.82	0.76	0.82
	Hamming loss	0.23	0.17	0.05	0.17	0.23	0.17
	F1 score	0.76	0.81	0.94	0.81	0.75	0.81
MLP	Précision	0.76	0.82	0.94	0.82	0.82	0.83

	Hamming loss	0.23	0.17	0.05	0.17	0.17	0.16
	F1 score	0.76	0.81	0.93	0.81	0.82	0.83

Les similarités pré-entraînées de CodeBERT semblent particulièrement efficaces pour identifier des concepts tels que la déclaration de variables, l'écriture et les déclarations conditionnelles. Les modèles tels que la forêt aléatoire et le balanced bagging se sont révélés être les plus performants, tandis que d'autres modèles ont des forces et des faiblesses spécifiques en fonction du concept, comme le montre la Figure V.8.

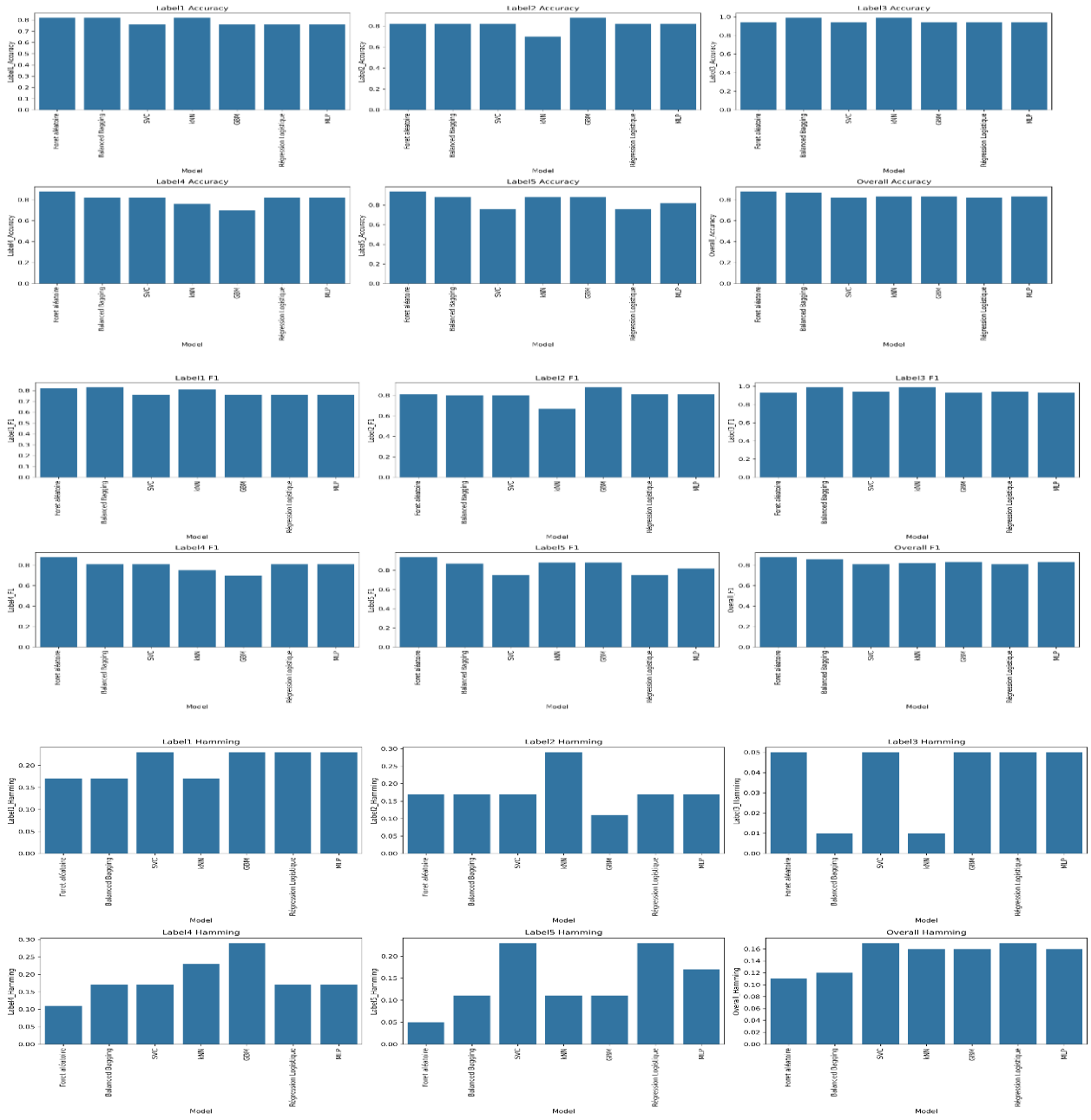

Figure V.8 Analyse comparative de différents modèles basée sur les similarités CodeBERT

La figure V.9 illustre la performance de différents modèles sur diverses étiquettes en termes de scores F1 et de précision basés sur les similarités CodeBERT. Pour les scores F1, l'étiquette d'écriture (Label 3) se distingue avec un score significativement supérieur, sa boîte

étant nettement plus élevée que les autres. L'étiquette de lecture (Label 2) montre le moins de variabilité. Les étiquettes de variables, opérations arithmétiques et déclarations conditionnelles (Labels 1, 4 et 5) présentent de bonnes performances en termes de score F1 médian et de dispersion des données (IQR). Cependant, nous observons des valeurs aberrantes possibles pour les étiquettes de lecture et de déclarations conditionnelles (Labels 2 et 5).

En termes de précision, la boîte de l'écriture (Label 3) est plus élevée que celles de toutes les autres étiquettes. La moustache inférieure pour l'écriture est aussi plus élevée que les moustaches supérieures des variables, de la lecture, des opérations et des déclarations conditionnelles (Labels 1, 2, 4 et 5). De plus, le schéma montre des valeurs aberrantes potentielles pour la lecture (Label 2) au-dessus de la moustache supérieure et pour les déclarations conditionnelles (Label 5) en dessous de la moustache inférieure.

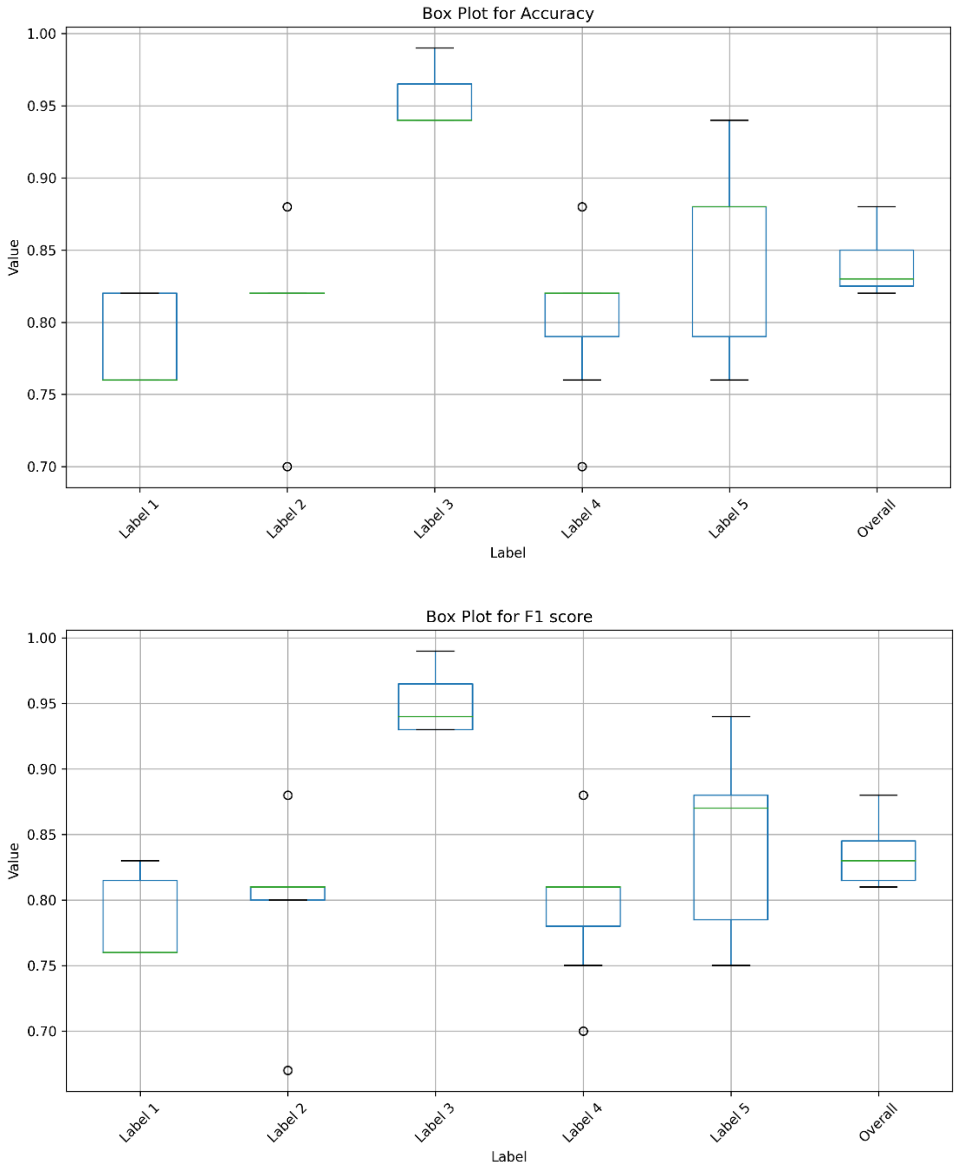

Figure V.9 Boites à moustaches de la précision et le f1 score des différents modèles basées sur les similarités CodeBERT

4.4 Résultats basés sur les similarités de UniXcoder

Les modèles de forêt aléatoire et Balanced Bagging affichent les meilleures performances globales avec une précision et un score F1 très élevés et constants. Ces deux modèles ont une précision globale de 0.92, indiquant leur efficacité pour cette tâche de classification multi-

étiquettes. Les modèles SVM, kNN et MLP suivent de près avec une précision globale de 0.91, montrant également de très bonnes performances, bien qu'elles soient légèrement inférieures à celles des deux premiers modèles.

En revanche, la régression logistique offre des performances globales légèrement inférieures avec une précision globale de 0.89, mais elle reste compétitive par rapport aux autres modèles. En revanche, les machines à gradient boosting montrent les performances les plus faibles avec une précision globale de 0.83, une perte de Hamming de 0.16 et un score F1 de 0.83. Cela pourrait être dû à la complexité du modèle ou à la nature des données utilisées.

Tableau 5.6 Analyse Comparative des différents modèles de classification basée sur les similarités UniXcoder

Modèle	**Métrique**	**Label1**	**Label2**	**Label3**	**Label4**	**Label5**	**Global**
Forêt aléatoire	Précision	0.93	0.93	1.0	0.89	0.86	0.92
	Hamming loss	0.06	0.06	0.0	0.10	0.10	0.07
	F1 score	0.93	0.93	1.0	0.89	0.86	0.92
Balanced Bagging	Précision	0.93	0.93	1.0	0.89	0.86	0.92
	Hamming loss	0.06	0.06	0.0	0.10	0.13	0.07
	F1 score	0.93	0.93	1.0	0.89	0.86	0.92
SVM	Précision	0.89	0.93	1.0	0.93	0.82	0.91
	Hamming loss	0.10	0.06	0.0	0.06	0.17	0.08
	F1 score	0.89	0.93	1.0	0.93	0.82	0.91
kNN	Précision	0.89	0.96	1.0	0.86	0.86	0.91
	Hamming loss	0.10	0.03	0.0	0.13	0.13	0.08
	F1 score	0.89	0.96	1.0	0.86	0.86	0.91
Gradient Boosting Machines	Précision	0.76	0.88	0.94	0.70	0.88	0.83
	Hamming loss	0.23	0.11	0.05	0.29	0.11	0.16
	F1 score	0.76	0.88	0.93	0.70	0.88	0.83
Régression Logistique	Précision	0.89	0.86	0.93	0.93	0.86	0.89
	Hamming loss	0.10	0.13	0.06	0.06	0.13	0.10
	F1 score	0.89	0.86	0.93	0.93	0.86	0.89
MLP	Précision	0.89	0.93	1.0	0.89	0.82	0.91
	Hamming loss	0.10	0.06	0.0	0.10	0.17	0.08

	F1 score	0.89	0.93	1.0	0.89	0.83	0.91

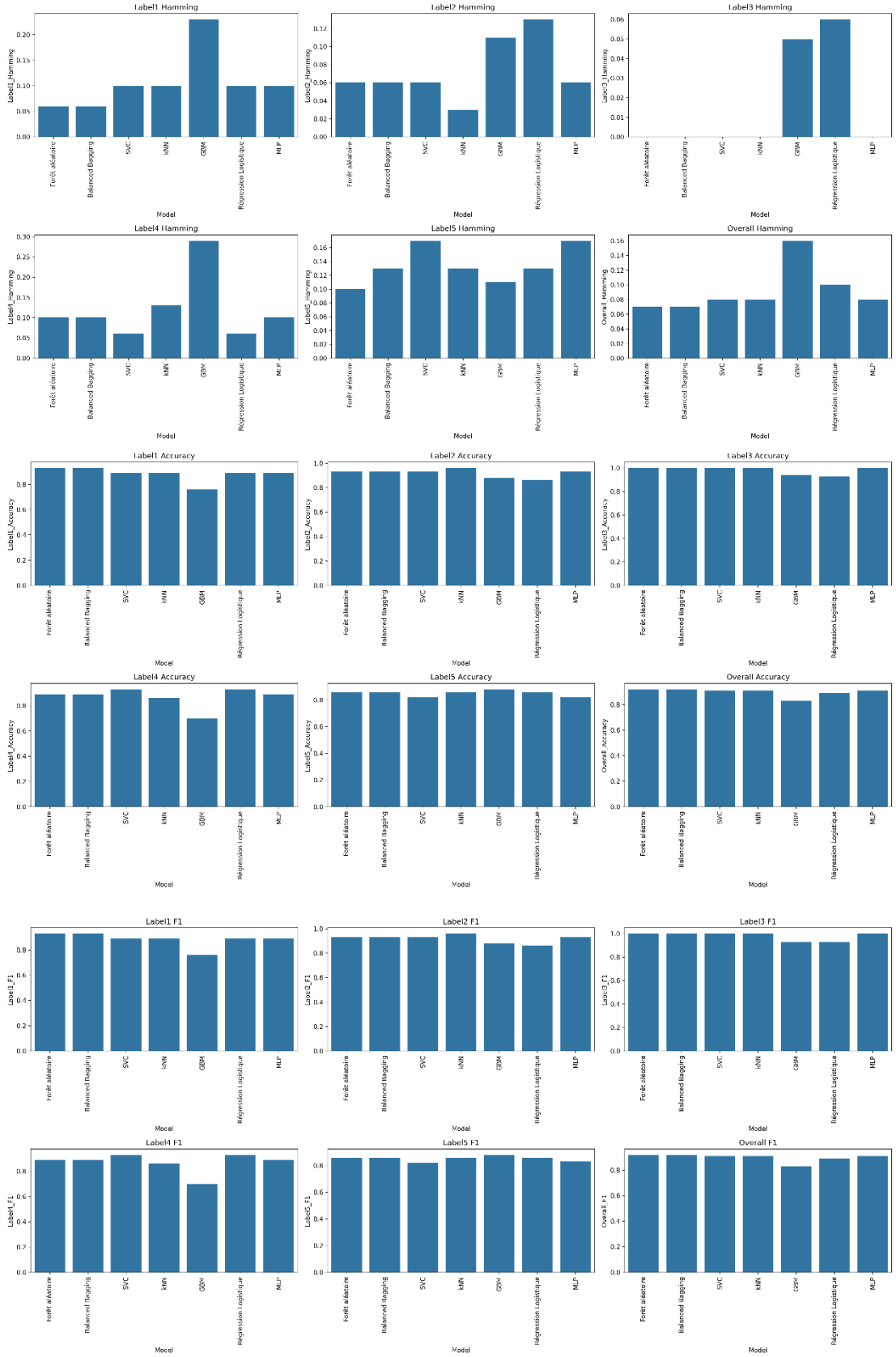

Figure V.10 Analyse comparative de différents modèles basée sur les similarités UniXcoder

En examinant les performances par label, on observe que tous les modèles réussissent à prédire parfaitement le label d'écriture acquise avec une précision et un score F1 de 1.0 et un Hamming

loss de 0.0 (figure V.10). Cependant, les performances varient davantage pour les labels de déclaration conditionnelles et opérations arithmétiques, indiquant que ces labels sont peut-être plus difficiles à prédire correctement. On constate qu'en utilisant les similarités de UniXcoder, de meilleures performances sont atteintes avec les modèles de foret aléatoire et Balanced Bagging, suivis de près par SVM, kNN et MLP. Les résultats montrent également que certains labels sont plus difficiles à prédire correctement, ce qui pourrait nécessiter une attention particulière dans l'amélioration des modèles ou le traitement des données.

La figure V.11 présente les performances des classifieurs, basées sur les similarités de UniXcoder. On constate que celles-ci sont globalement bonnes avec des F1 scores et des précisions élevées. Cependant, il existe une variabilité pour certains labels, en particulier pour le label 5 (if/else), ce qui suggère que les classifieurs rencontrent davantage de difficultés avec certaines structures de code. Que ce soit pour la précision ou le f1 score, les tendances sont similaires. Pour le label 1 (déclaration de variable), la médiane est proche de 0,90, avec une plage interquartile (IQR) étroite, ce qui indique une performance stable malgré quelques valeurs aberrantes autour de 0,75. Le label 2 (lecture) affiche une médiane similaire de 0,90, une IQR légèrement plus large et une valeur aberrante notable sous 0,85. Le label 3 (écriture) a une médiane proche de 0,98, une IQR étroite et un outlier autour de 0,90. Pour le label 4 (opérations arithmétiques), la médiane est de 0,90 avec une IQR plus large, suggérant une variabilité accrue. Le label 5 (if/else) montre une médiane de 0,85, une IQR large et plusieurs outliers en dessous de 0,80, indiquant une performance plus variable.

4.5 Résultats de la Fusion Pondérée

Les résultats de la fusion pondérée des précisions spécifiques aux étiquettes sont prometteurs. Pour chaque modèle pré-entraîné, nous avons identifié les meilleurs modèles en fonction de leur précision combinée pour toutes les étiquettes. Pour illuster, les modèles de la forêt aléatoire, Gradient Boosting Machines et balanced bagging ont été sélectionnés pour BERT et CodeBERT. Pour RoBERTa, les modèles étaient la forêt aléatoire, Gradient Boosting Machines et kNN. Enfin, pour UniXcoder, les modèles sélectionnés sont la forêt aléatoire, balanced bagging et SVM.

Tout d'abord, en ce qui concerne la déclaration de variable (Label 1), le modèle UniXcoder se distingue avec une précision exceptionnelle de 0.93, surpassant BERT (0.82), RoBERTa (0.78) et CodeBERT (0.80). En termes de lecture (Label 2), UniXcoder atteint également une précision de 0.93, démontrant une amélioration notable par rapport aux autres modèles : BERT (0.78), RoBERTa (0.82) et CodeBERT (0.84). Cette tendance montre la robustesse de UniXcoder dans l'identification des opérations de déclaration de variable et de lecture. Concernant l'écriture (Label 3), UniXcoder atteint une précision parfaite de 1.0, ce qui le positionne au-dessus des autres modèles pré-entraînés. BERT, RoBERTa et CodeBERT affichent des performances similaires avec des précisions respectives de 0.94, 0.94 et 0.96. Pour les opérations arithmétiques (Label 4), UniXcoder et RoBERTa montrent des précisions élevées de 0.93 et 0.86 respectivement. CodeBERT et BERT obtiennent des résultats inférieurs de 0.80 et 0.74, respectivement. Enfin, concernant les déclarations conditionnelles (Label 5), Bien que BERT et CodeBERT obtiennent une précision de 0.90, légèrement supérieure à celle de RoBERTa (0.88) et de UniXcoder (0.86), UniXcoder reste compétitif, démontrant ainsi sa polyvalence.

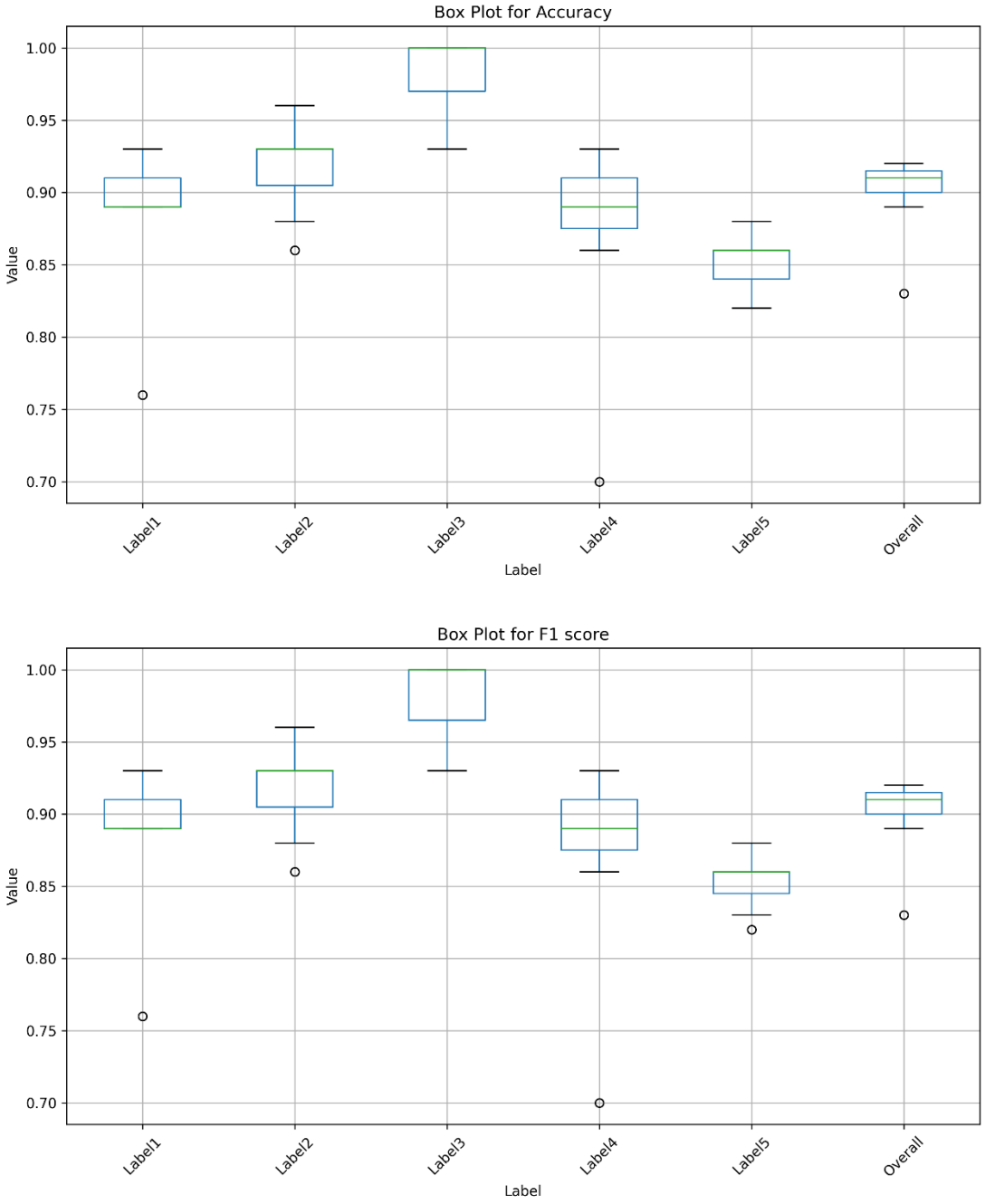

Figure V.11 Boites à moustaches de la précision et le f1 score des différents modèles basées sur les similarités UniXcoder

Même si chaque modèle présente des forces distinctes selon les étiquettes, UniXcoder se distingue généralement comme le modèle le plus performant, atteignant des précisions exceptionnelles, particulièrement pour les déclarations de variable, de lecture et d'écriture. Ces résultats soulignent l'efficacité des similarités basées sur UniXcoder, surpassant les autres modèles évalués. Cess observations obtenues par UniXcoder peuvent être attribués en grande partie à la structure inhérente du modèle, spécifiquement conçue pour comprendre et traiter du code. Cette structure confère à UniXcoder une capacité accrue à saisir les nuances syntaxiques et sémantiques du code, ce qui se reflète dans ses performances élevées sur plusieurs étiquettes.

Tableau 5.7 Résultats de la Fusion Pondérée

Modèle	Label 1	Label 2	Label 3	Label 4	Label 5
BERT	0.82	0.78	0.94	0.74	**0.90**
RoBERTa	0.78	0.82	0.94	0.86	0.88
CodeBERT	0.80	0.84	0.96	0.80	**0.90**
UniXcoder	**0.93**	**0.93**	**1.0**	**0.93**	0.86

Les résultats démontrent que l'approche de fusion pondérée entraîne des améliorations prometteuses en termes de performance comme le montre la Figure V.12. En particulier, pour l'étiquette de déclarations conditionnelles avec des valeurs allant de 0,86 à 0,90. UniXcoder se distingue avec des performances remarquables, atteignant un score parfait de 1.0 pour le Label 3 et des scores de 0.93 pour les Labels 1, 2 et 4. En comparaison, BERT et RoBERTa montrent des performances similaires avec des scores variés, mais généralement inférieurs à ceux d'UniXcoder. CodeBERT, bien que performant, est légèrement inférieur à UniXcoder mais supérieur à BERT et RoBERTa sur plusieurs labels. Cela suggère que les modèles d'ensemble choisis pour chaque modèle de base abordent différents types d'erreurs, conduisant à de meilleures performances globales.

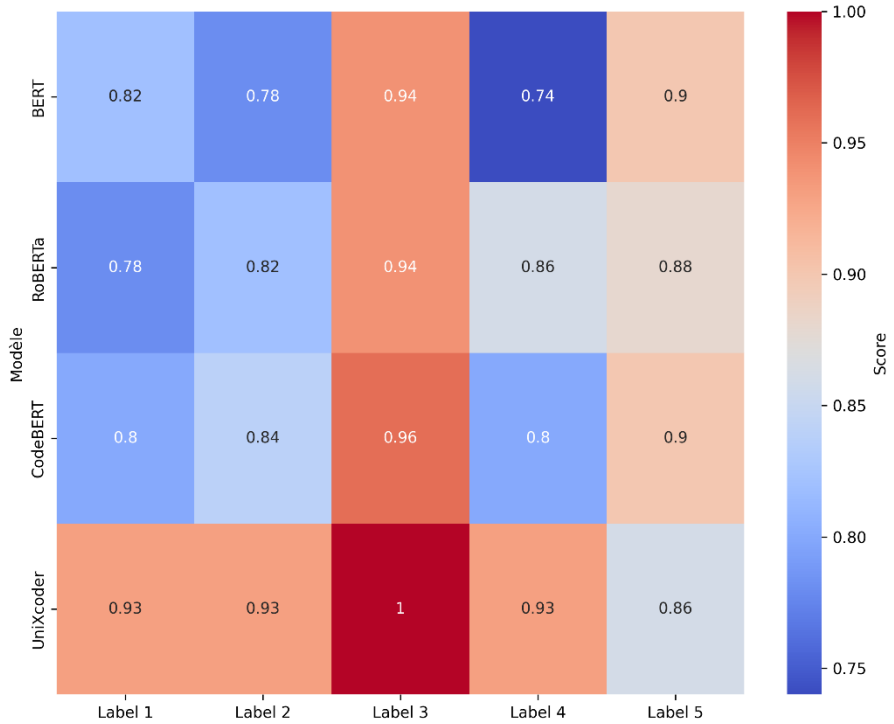

Figure V.12 Carte thermique de la comparaison des performances des modèles de classification

5 Complexité des Algorithmes

5.1 Complexité du Calcul de Similarité utilisant les PLMs

La complexité temporelle et spatiale d'un algorithme constitue une considération essentielle dans l'évaluation de ses performances et de son efficacité. Étant donné que le modèle pré-entraîné UnixCoder s'est révélé le plus performant dans cette étude, nous allons à présent examiner en détail sa complexité temporelle et spatiale.

5.1.1 Complexité Temporelle

La complexité temporelle se réfère à la quantité de temps qu'un algorithme prend pour s'exécuter en fonction de la taille de l'entrée. Dans ce contexte, l'algorithme utilise le modèle UnixCoder pour évaluer la similarité entre les solutions des étudiants et les solutions jugées correctes par l'expert. L'analyse commence par le chargement du modèle et du tokenizer pré-entraîné de `transformers`, une opération qui est relativement fixe en termes de temps $(O(1))$ puisque le modèle et le tokenizer sont simplement chargés en mémoire. Ensuite, les solutions des étudiants et les solutions correctes sont tokenisées. Cette opération a une complexité temporelle linéaire par rapport à la taille des données d'entrée, $O(n)$, où n est le nombre de solutions à tokeniser. Le temps de tokenisation est également influencé par la longueur maximale fixée des séquences (ici, 246). Le passage à travers le modèle pour obtenir les plongements sémantiques, effectué sans gradient, implique une complexité temporelle qui dépend de la profondeur du modèle et du nombre de paramètres, généralement représentée comme $O(m * n)$, où m est le nombre de couches et n est la longueur de la séquence. Le calcul de la similarité cosinus entre les embeddings des solutions des étudiants et celles de l'expert a une complexité temporelle quadratique, $O(n^2)$, par rapport au nombre de solutions, puisque chaque paire de solutions doit être comparée.

5.1.2 Complexité Spatiale

La complexité spatiale, quant à elle, se réfère à la quantité de mémoire nécessaire pour exécuter l'algorithme. Lors de la tokenisation, des objets tensors sont créés pour stocker les séquences tokenisées, ce qui requiert de l'espace proportionnel à la taille des entrées, soit $O(n * k)$, où n est le nombre de séquences et k la longueur maximale des séquences. Le modèle pré-entraîné, lorsqu'il est chargé, occupe une quantité fixe de mémoire dépendant de son architecture et de ses paramètres (souvent plusieurs centaines de mégaoctets). Les plongements sémantiques générés pour chaque solution sont stockés en mémoire, nécessitant de l'espace $O(n * d)$, où d est la dimension des embeddings. Le calcul de la similarité cosinus implique la création d'une matrice de similarité de taille $n*n$, contribuant à une complexité spatiale quadratique,$O(n^2)$. Enfin, l'ajout des scores de similarité au DataFrame original nécessite une mémoire additionnelle proportionnelle au nombre d'entrées,$O(n)$.

Le calcul de la similarité en utilisant le modèle pré-entraîné UniXcoder présente une complexité temporelle et spatiale principalement influencée par le nombre de solutions étudiantes et correctes. La tokenisation et l'extraction des embeddings ont une complexité linéaire, tandis que le calcul de la similarité cosinus apporte une complexité quadratique significative. Pour des ensembles de données volumineux, ces caractéristiques peuvent devenir un facteur limitant en termes de temps d'exécution et de consommation mémoire, nécessitant des optimisations pour maintenir des performances acceptables.

5.2 Complexité de la Classification Multi-Etiquettes

La complexité temporelle et spatiale du modèle Chains avec comme base la forêt aléatoire et le balanced bagging sont présentés ci-dessous :

5.2.1 Complexité Temporelle de la Forêt aléatoire

Le processus débute par le chargement des données et le nettoyage des valeurs manquantes, une opération dont la complexité temporelle est $O(n)$, où n est le nombre de lignes du DataFrame. L'encodage de la variable catégorielle est une opération linéaire, $O(n)$, puisqu'il parcourt toutes les lignes pour effectuer le mapping des valeurs. La normalisation des caractéristiques est également une opération linéaire, $O(n * m)$, où n est le nombre de lignes et m est le nombre de caractéristiques. La normalisation implique le calcul de la moyenne et de l'écart type pour chaque caractéristique, suivi par la transformation de chaque valeur. La partie essentielle du code consiste à entraîner et évaluer des chaînes de classification pour chaque variable cible. L'initialisation du `ClassifierChain` avec `RandomForestClassifier` et l'entraînement du modèle ont une complexité temporelle plus élevée.

1) RandomForestClassifier: La complexité de l'entraînement d'un RandomForestClassifier dépend du nombre d'arbres (t) et de la profondeur de chaque arbre (d). Pour un ensemble de données de taille n, la complexité temporelle approximative est$O(t * n * log(n) * d)$. Chaque arbre dans la forêt est construit indépendamment en utilisant un sous-ensemble aléatoire des données, et le processus de décision basé sur les caractéristiques est relativement coûteux.
2) ClassifierChain: La complexité d'entraînement d'une chaîne de classification dépend de la longueur de la chaîne, c'est-à-dire du nombre de variables cibles. Si l'on a k variables cibles, chaque modèle est entraîné séquentiellement, ce qui ajoute une complexité linéaire, rendant la complexité totale $O(k * t * n * log(n) * d)$.

Le calcul des métriques de performance comme l'accuracy, le hamming loss et le F1 score se fait pour chaque cible, ce qui implique une complexité $O(k * n)$ pour l'évaluation.

5.2.2 Complexité Spatiale de la Forêt Aléatoire

Le DataFrame d'origine et les matrices résultantes après normalisation sont stockés en mémoire. Si le DataFrame initial a n lignes et m colonnes de caractéristiques, la complexité spatiale est $O(n * m)$. Par ailleurs, les modèles de forêt aléatoire sont particulièrement gourmands en mémoire. Chaque arbre dans la forêt stocke les informations sur les décisions à chaque nœud. La complexité spatiale pour un seul arbre est$O(n * d)$, où d est la profondeur de l'arbre. Pour une forêt de t arbres, cela devient$O(t * n * d)$. De plus, les chaînes de classification ajoutent une dimension supplémentaire à la complexité spatiale. Chaque modèle dans la chaîne doit stocker ses propres paramètres, ce qui conduit à une complexité $O(k * t * n * d)$ pour k variables cibles.

La mise en œuvre des chaînes de classification avec RandomForestClassifier est influencée par une complexité temporelle et spatiale qui croît linéairement avec le nombre de variables cibles et les données d'entrée, mais de manière exponentielle avec la profondeur et le nombre d'arbres de la forêt. Pour maintenir des performances acceptables, des optimisations sont souvent nécessaires. En effet, pour des ensembles de données volumineux, la complexité temporelle et spatiale élevée peut devenir un goulot d'étranglement. Par conséquent, il est important d'envisager des techniques d'optimisation telles que la réduction de la taille des données, la sélection de caractéristiques, ou l'utilisation de méthodes d'approximation pour réduire le temps d'entraînement et la consommation mémoire.

5.2.3 Complexité Temporelle du Balanced Bagging

La complexité temporelle et spatiale de l'implémentation du Balanced Bagging avec des chaînes de classification est influencée par plusieurs facteurs clés. Le Balanced Bagging, via la bibliothèque `imblearn`, est conçu pour améliorer la performance sur des ensembles de données déséquilibrés en ré-échantillonnant les classes minoritaires. Dans ce contexte, chaque modèle de `ClassifierChain` utilise le balanced bagging avec une forêt aléatoire comme estimateur de

base. La complexité temporelle de cet ensemble repose sur la complexité de la forêt aléatoire, déjà notablement élevée avec $O(t * n * log(n) * d)$ pour t arbres, n échantillons et d profondeur. Le processus de ré-échantillonnage ajouté par le balanced bagging augmente cette complexité, puisque des sous-ensembles équilibrés sont créés et chaque échantillon est utilisé plusieurs fois, augmentant ainsi le temps de traitement de chaque itération. La complexité temporelle de la chaîne entière est donc $O(k * t * n * log(n) * d)$ multiplié par le facteur de ré-échantillonnage, ce qui peut devenir prohibitif pour de grands ensembles de données.

5.2.4 Complexité Spatiale du Balanced Bagging

L'utilisation du balanced bagging dans des chaînes de classification entraîne également une augmentation significative de la consommation mémoire. Chaque modèle de la chaîne doit non seulement stocker ses propres paramètres mais aussi gérer les multiples échantillons générés par le ré-échantillonnage. La complexité spatiale de la forêt aléatoire, $O(t * n * d)$, est multipliée par le nombre de ré-échantillonnages effectués par le bagging équilibré. De plus, la chaîne de classification, qui entraîne successivement plusieurs modèles pour chaque variable cible, amplifie encore cette consommation. Ainsi, pour k variables cibles, la complexité spatiale totale devient $O(k * t * n * d)$ multiplié par le facteur de ré-échantillonnage, ce qui peut rapidement consommer une grande quantité de mémoire, rendant nécessaire la gestion prudente des ressources pour des applications pratiques.

5.3 Complexité de la Fusion Pondérée

5.3.1 Complexité Temporelle de la Fusion Pondérée

La complexité temporelle de la fusion pondérée des précisions spécifiques aux étiquettes dépend de plusieurs étapes clés dans le processus :

1) Calcul des Précisions Spécifiques aux Étiquettes: Pour chaque modèle (j) et chaque étiquette(i), nous devons calculer la précision spécifique à l'étiquette(A_{ij}). Si nous avons N modèles et M étiquettes, cela nécessite $(O(N \times M))$ opérations, car nous devons évaluer la précision pour chaque combinaison modèle-étiquette.
2) Calcul des Précisions Globales: La précision globale pour chaque modèle (w_j) est calculée sur toutes les étiquettes et tous les points de données. Si nous avons (n) points de données et M étiquettes, chaque modèle nécessite $(O(n \times M))$ opérations pour évaluer sa précision globale. Pour N modèles, cela devient$(O(N \times n \times M))$.
3) Pondération des Précisions Spécifiques: Pour chaque étiquette (i), nous devons effectuer la pondération en multipliant (A_{ij}) par (w_j) pour chaque modèle j, ce qui nécessite $(O(N \times M))$opérations.
4) Somme et Normalisation des Précisions Pondérées: La somme des résultats pondérés et la normalisation nécessitent des opérations supplémentaires de$(O(N \times M))$.

La complexité temporelle totale peut donc être approximée par la somme des différentes étapes :

$$O(N \times M) + O(N \times n \times M) + O(N \times M) + O(N \times M) = O(N \times n \times M)$$

5.3.2 Complexité Spatiale de la Fusion Pondérée

La complexité spatiale de la fusion pondérée dépend principalement de la mémoire nécessaire pour stocker les précisions spécifiques aux étiquettes, les précisions globales des modèles, et les résultats intermédiaires.

1) Stockage des Précisions Spécifiques aux Étiquettes: Nous devons stocker (A_{ij}) pour chaque modèle (j) et chaque étiquette(i). Avec N modèles et M étiquettes, cela nécessite $(O(N \times M))$ unités de mémoire.

2) Stockage des Précisions Globales: Chaque modèle nécessite le stockage de sa précision globale(w_j), ce qui nécessite ($O(N)$) unités de mémoire.
3) Stockage des Résultats Pondérés: Les résultats pondérés pour chaque étiquette nécessitent également ($\mathrm{O(N \times M)}$) unités de mémoire.

La complexité spatiale totale est donc :

$$\mathrm{O(N \times M) + O(N) + O(N \times M) = O(N \times M)}$$

La complexité temporelle de la fusion pondérée des précisions spécifiques aux étiquettes est($\mathrm{O(N \times n \times M)}$), où N est le nombre de modèles, n est le nombre de points de données, et M est le nombre d'étiquettes. La complexité spatiale est($\mathrm{O(N \times M)}$). Ces complexités reflètent le coût computationnel et la mémoire nécessaires pour implémenter efficacement la fusion pondérée dans un système de classification multi-étiquettes.

6 Discussion

Cette section discute de l'efficacité des PLMs pour évaluer la compréhension du code par les apprenants. Une comparaison de quatre modèles : BERT et RoBERTa, des PLMs généralistes établis, et CodeBERT, et UniXcoder, des modèles spécifiquement entraînés sur des tâches liées au code, a été réalisée. Par la suite, une nouvelle approche de fusion pondérée a été proposée pour exploiter les forces complémentaires de différents modèles d'apprentissage automatique dans la tâche de la classification multi-étiquettes. Cette analyse vise à valider les hypothèses de recherche suivantes :

- **Hypothèse 1 (H1)** : les PLMs spécifiquement conçus pour les tâches centrées sur le code, comme CodeBERT et UniXcoder, atteignent une performance supérieure par rapport aux PLMs généralistes comme BERT et RoBERTa pour évaluer la compréhension du code par les apprenants.
- **Hypothèse 2 (H2)** : l'approche de fusion pondérée des modèles d'apprentissage automatique permet d'améliorer la précision et la robustesse de l'évaluation de la compréhension du code par les apprenants.
- **Hypothèse 3 (H3)**: l'architecture de UniXcoder, étant spécifiquement optimisée pour les tâches liées au code, offre de meilleurs résultats en termes de performance et de précision par rapport à CodeBERT.

6.1 Les modèles spécifiques pré-entraînés donnent-ils de meilleurs résultats ?

Dans cette étude, nous avons évalué la performance de quatre modèles de langage pré-entraînés pour la classification multi-étiquettes de la compréhension du code par des apprenants novices. Les modèles analysés sont BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), CodeBERT (Feng et al., 2020) et UniXcoder (Guo et al., 2022). Ces modèles ont été utilisés pour générer des scores basés sur les similarités sémantiques. Chacun d'eux présente des forces et des faiblesses distinctes.

Tout d'abord, en utilisant les similarités issues du modèle BERT avec comme base pour le classifieur Chains les modèles de forêt aléatoire et de gradient boosting machines, une précision globale de 0,84 est obtenue. Ces modèles ont montré leur efficacité dans la prédiction de certains concepts. Bien que BERT ait montré des résultats prometteurs dans la reconnaissance de certains concepts tels que la déclaration de variables et les opérations de lecture/écriture, il a eu des difficultés avec les déclarations conditionnelles et a produit des résultats variables d'une étiquette à l'autre. Ces résultats suggèrent que si BERT capture des relations sémantiques

générales, il peut nécessiter un affinement supplémentaire pour mieux saisir les structures de programmation spécifiques.

Ensuite, le modèle basé sur RoBERTa avec comme base pour le classifieur Chains le modèle gradient boosting machines s'est révélé être le plus performant avec une précision globale de 0,89. Cette combinaison a démontré son efficacité dans la prédiction des étiquettes de déclaration de variables et de déclarations conditionnelles, indiquant une meilleure compréhension de ces concepts par rapport aux modèles basés sur BERT. Cependant, des difficultés similaires avec d'autres étiquettes suggèrent que bien que RoBERTa soit capable de traiter des structures syntaxiques complexes, il n'est peut-être pas entièrement optimisé pour capturer les subtilités spécifiques au code.

En revanche, les modèles basés sur CodeBERT, en particulier le classifieur Chains avec comme base la forêt aléatoire et le balanced bagging, se sont révélés être les modèles les plus performants avec une précision élevée pour la plupart des étiquettes. Plus précisément, CodeBERT a montré une amélioration dans la prédiction des déclarations conditionnelles par rapport à BERT et RoBERTa. Étant donné que CodeBERT a été développé spécifiquement pour les tâches liées au code, il a montré la précision globale la plus élevée et une performance stable pour la plupart des étiquettes. Alors que les modèles généraux tels que BERT et RoBERTa fournissent des informations pertinentes, les modèles spécifiques au domaine tels que CodeBERT fournissent des performances supérieures pour des tâches spécialisées telles que la compréhension du code.

Enfin, les résultats obtenus à partir des similarités de UniXcoder démontrent que ce modèle spécifique au domaine surpasse de manière significative les performances des modèles généralistes, tels que BERT et RoBERTa, et même de CodeBERT, dans certaines mesures clés. Cette constatation souligne l'importance de la formation spécifique au domaine préalable pour des tâches telles que l'analyse du code, ce qui permet au modèle de capturer les subtilités des langages de programmation et de leurs fonctionnalités. Par conséquent, l'hypothèse (H1) est confirmée. Cependant, le choix du modèle approprié pour la classification multi-étiquettes de la compréhension du code des étudiants nécessite une évaluation minutieuse de leurs forces et faiblesses.

Les modèles de forêt aléatoire et Balanced Bagging comme base pour le classifieur Chains utilisant les similarités UniXcoder affichent une précision globale de 0.92, surpassant légèrement la performance globale de CodeBERT. L'analyse par label révèle que UniXcoder permet d'obtenir une précision parfaite (1.0) pour le label d'écriture, tout en maintenant des performances élevées pour les autres labels. Ces résultats corroborent l'hypothèse (H3) selon laquelle l'architecture de UniXcoder, spécifiquement optimisée pour les tâches liées au code, offre des performances supérieures en termes de précision par rapport à CodeBERT. Ces résultats soulignent également la nécessité d'une optimisation spécifique pour des tâches complexes et spécialisée telles que l'évaluation de la compréhension du code. L'utilisation de modèles comme UniXcoder, qui intègrent des connaissances spécialisées du domaine de la programmation, se révèle essentielle pour améliorer la précision des évaluations.

6.2 Méthode de Fusion Pondérée

Pour tirer pleinement parti des forces intrinsèques de chaque modèle de langage pré-entraîné (PLM), nous avons introduit une nouvelle approche nommée fusion pondérée des précisions spécifiques aux étiquettes. Cette méthode repose sur l'idée fondamentale que chaque modèle possède des compétences et des forces particulières pour certaines tâches spécifiques, et que la combinaison de ces compétences peut améliorer de manière significative les performances globales du modèle.

La fusion pondérée est une technique qui permet de combiner les prédictions de plusieurs modèles en attribuant des poids différents à chacune des prédictions en fonction de leur précision relative pour des étiquettes spécifiques. Par exemple, si un modèle excelle dans la prédiction des déclarations conditionnelles mais est moins performant dans la prédiction des opérations arithmétiques, la fusion pondérée ajustera les contributions de ce modèle de manière à maximiser les performances globales. Cette approche tire parti des points forts de chaque modèle tout en minimisant leurs faiblesses individuelles.

Les résultats obtenus par cette méthode ont été particulièrement prometteurs. En appliquant la fusion pondérée, nous avons pu résoudre des problèmes persistants et complexes tels que la prédiction des déclarations conditionnelles. Ces déclarations sont souvent difficiles à évaluer en raison de leur nature complexe et de la nécessité de comprendre le flux logique du code. La fusion pondérée a permis une évaluation plus précise de cette étiquette. En particulier, les modèles UniXcoder et CodeBERT ont démontré des améliorations significatives par rapport aux modèles généralistes comme BERT et RoBERTa. Ils ont montré une précision plus élevée sur plusieurs étiquettes, confirmant leur supériorité dans les tâches liées à la compréhension du code. Cette méthode a également permis d'identifier et de corriger des erreurs spécifiques que les modèles généralistes avaient tendance à commettre, améliorant ainsi la performance globale.

L'analyse des résultats obtenus grâce à la fusion pondérée des précisions spécifiques aux étiquettes a révélé une compréhension plus approfondie de la capacité de chaque modèle à reconnaître et à classer divers concepts de programmation. Cette approche nous a permis de mieux comprendre les compétences et les limitations de chaque modèle, et de combiner leurs forces de manière à maximiser les performances globales tout en atténuant leurs limitations. La fusion pondérée a révélé que chaque modèle excelle dans des domaines spécifiques. Par exemple, UniXcoder a montré une performance exceptionnelle dans la reconnaissance des variables, lecture/ écriture et des opérations arithmétiques complexes. En revanche, CodeBERT a démontré une meilleure performance dans les déclarations conditionnelles. En combinant ces modèles à l'aide de la fusion pondérée, nous avons pu obtenir une évaluation plus précise de la compréhension du code par les apprenants.

Les résultats de notre analyse ont confirmé que UniXcoder est le modèle le plus performant parmi ceux étudiés, validant ainsi notre hypothèse (H3). L'efficacité de la fusion pondérée dans le traitement de concepts difficiles, tels que les déclarations conditionnelles, a également confirmé l'hypothèse (H2). Ces résultats montrent que la méthode de fusion pondérée est capable d'améliorer la robustesse et la précision des évaluations de la compréhension du code. Par ailleurs, cette étude a également démontré que les PLMs spécialisés dans les tâches de programmation, tels que CodeBERT et UniXcoder, surpassent les modèles généralistes dans l'évaluation de la compréhension du code par les apprenants. Les résultats de notre approche de fusion pondérée montrent clairement que cette technique est capable d'améliorer la précision des évaluations, confirmant ainsi la validité des hypothèses H2 et H3.

Les résultats de notre étude soulignent l'importance des approches d'ensemble et des techniques de fusion pondérée dans l'amélioration des outils d'évaluation de la compréhension du code. Ils montrent également le potentiel de ces techniques pour contribuer à l'amélioration de l'éducation de la programmation, en fournissant des évaluations plus justes et plus précises des compétences des apprenants. Ces résultats encouragent à poursuivre les recherches sur l'utilisation des approches d'ensemble pour combiner les forces de différents modèles d'apprentissage automatique, afin de développer des outils d'évaluation encore plus stratégiques, ouvrant ainsi la voie à de nouvelles recherches et développements dans le domaine de l'éducation à la programmation.

6.3 Implications Théoriques

Les résultats de cette étude enrichissent considérablement la compréhension théorique des modèles de langages pré-entraînés (PLMs) dans des domaines spécialisés, notamment les contextes éducatifs axés sur la programmation. La validation des hypothèses (H1) et (H2) offre des éclairages précieux sur l'efficacité de la formation spécifique au domaine et des approches d'ensemble. D'un point de vue théorique, cela corrobore l'idée que les PLMs peuvent être adaptés et perfectionnés pour des tâches spécialisées grâce à des techniques de formation et de fusion sur mesure.

Tout d'abord, la performance supérieure de UniXcoder et CodeBERT met en lumière l'importance cruciale de la formation spécifique au domaine pour les PLMs. Cette constatation s'aligne parfaitement avec les perspectives théoriques qui soulignent la nécessité d'intégrer des connaissances contextuelles et spécifiques au domaine pour optimiser la performance des modèles dans des tâches spécialisées. En d'autres termes, la capacité d'un modèle à comprendre et à traiter des données spécifiques à un domaine particulier, comme la programmation, est essentielle pour améliorer sa précision et son efficacité. Cette approche contextualisée permet non seulement d'affiner les capacités du modèle mais aussi de rendre ses prédictions plus pertinentes et fiables.

Ensuite, la mise en œuvre réussie de la fusion pondérée vient renforcer les théories soutenant les approches d'apprentissage en ensemble. En tirant parti des forces combinées de plusieurs modèles, les méthodes d'ensemble peuvent compenser les limitations inhérentes à chaque modèle individuel. Cette stratégie conduit à une amélioration notable de la précision des prédictions. Ce résultat est en phase avec des recherches récentes démontrant l'efficacité des méthodes d'apprentissage en ensemble pour améliorer les performances de classification dans divers domaines (Alzanin et al. 2023; M. Li, S. Wang, et L. Guo 2018). En effet, les méthodes d'ensemble exploitent la diversité et la complémentarité des modèles individuels pour produire des résultats plus stables et fiables, ce qui est particulièrement bénéfique dans des contextes éducatifs où les erreurs peuvent avoir des conséquences importantes.

Les implications théoriques de ces résultats s'étendent bien au-delà du contexte immédiat de l'éducation à la programmation. Elles suggèrent que les PLMs peuvent être adaptés et combinés de manière similaire dans d'autres domaines spécialisés pour améliorer les performances. Par exemple, dans le domaine médical, les PLMs pourraient être formés sur des ensembles de données spécifiques pour diagnostiquer des maladies avec une précision accrue. De même, dans le domaine financier, ces modèles pourraient être utilisés pour analyser et prédire les tendances du marché en se basant sur des données économiques et contextuelles spécifiques. Ainsi, les applications potentielles des PLMs formés et fusionnés sont vastes et variées, ouvrant la voie à de nouvelles avancées dans de nombreux domaines spécialisés.

Ce travail de recherche non seulement valide des hypothèses clés sur l'efficacité des PLMs dans des contextes spécialisés, mais il ouvre également de nouvelles perspectives pour leur application et leur développement futur. En démontrant l'importance de la formation spécifique au domaine et des approches d'ensemble, la présente étude offre des bases solides pour des recherches ultérieures visant à exploiter pleinement le potentiel des PLMs dans divers domaines.

6.4 Implications dans le Contexte Éducatif : Au-delà des Scores de Précision

Dans un contexte d'EIAH, l'application de ces modèles offre une promesse significative pour automatiser l'évaluation de la compréhension du code des apprenants, améliorant ainsi les méthodes d'enseignement et soutenant l'apprentissage des étudiants. Cette automatisation permet de créer des plateformes d'apprentissage plus interactives et adaptatives, facilitant une

personnalisation accrue des parcours éducatifs et une réactivité plus grande aux besoins individuels des apprenants. De plus, les implications pratiques suivantes mettent en évidence le potentiel des modèles de langages pré-entraînés dans l'éducation, facilitant un environnement d'apprentissage plus personnalisé et réactif.

Identification précise des concepts de programmation

En identifiant avec précision différents concepts de programmation dans le code soumis, ces modèles peuvent fournir aux enseignants des informations précieuses qui permettent des retours personnalisés et des interventions ciblées pour les apprenants. Par exemple, si un étudiant rencontre des difficultés récurrentes avec les boucles ou les structures conditionnelles, le modèle peut alerter l'enseignant et suggérer des exercices spécifiques pour renforcer ces compétences. Cette granularité dans la détection des compétences permet de personnaliser l'apprentissage en fonction des besoins individuels de chaque élève, ce qui peut améliorer l'efficacité de l'enseignement.

Intégration dans des plateformes d'apprentissage adaptatif

Le modèle pourrait être intégré dans des plateformes d'apprentissage adaptatif pour ajuster dynamiquement le contenu d'apprentissage et le niveau de difficulté en fonction de la progression des apprenants dans la compréhension des différents concepts de programmation. Par exemple, une plateforme pourrait commencer par des exercices de base sur les variables et les structures de contrôle, puis progressivement introduire des concepts plus avancés comme les algorithmes de tri et les structures de données, en s'adaptant en temps réel aux performances de l'étudiant. Cette adaptabilité peut aider à maintenir l'engagement des étudiants et à s'assurer qu'ils ne sont pas dépassés ou sous-stimulés par le matériel d'apprentissage.

Détection précoce des difficultés

La détection précoce des apprenants en difficulté avec certains concepts pourrait permettre une intervention rapide et empêcher l'aggravation des lacunes d'apprentissage. Par exemple, si un étudiant montre des signes de confusion persistante avec les concepts de récursivité, le modèle peut identifier ce problème tôt et recommander des sessions de tutorat ou des ressources supplémentaires pour aider l'étudiant à surmonter cet obstacle. Une intervention rapide peut prévenir l'accumulation de lacunes, garantissant que les étudiants restent sur la bonne voie dans leur apprentissage.

Surveillance continue et soutien en temps réel

La surveillance continue du code des étudiants à l'aide de modèles Chains peut détecter rapidement les difficultés émergentes, permettant aux enseignants de fournir un soutien opportun et d'empêcher l'aggravation des lacunes de compétences. Par exemple, si un étudiant soumet plusieurs solutions incorrectes à un problème de programmation, le modèle peut automatiquement générer des indices ou des conseils pour guider l'étudiant vers la solution correcte, réduisant ainsi la frustration et augmentant la motivation à persévérer.

Évaluation de la qualité et de l'efficacité du code

Enfin, les modèles Chains peuvent être formés pour évaluer des aspects englobant du code tels que la qualité et l'efficacité du code, fournissant des informations précieuses sur la compréhension des élèves. Par exemple, le modèle peut analyser le code soumis pour vérifier s'il suit les meilleures pratiques de programmation, comme l'utilisation appropriée des fonctions et des structures de données, ou si le code est optimisé pour la performance. Ces évaluations peuvent aider les étudiants à développer de bonnes habitudes de codage en amont et à comprendre l'importance de l'efficacité et de la lisibilité du code.

L’approche de la fusion pondérée proposée, bien qu'elle engendre un coût computationnel légèrement supérieur comme le démontrent les analyses temporelles (entre 1,46 et 1,81 secondes de traitement supplémentaire), présente des avantages significatifs par rapport aux méthodes traditionnelles basées sur les cas de test. En effet, là où les approches conventionnelles par test unitaires nécessitent une définition exhaustive et fastidieuse de tous les cas de test possibles, l’approche présentée offre une évaluation plus flexible et plus complète des solutions proposées par les apprenants. Cette supériorité se manifeste particulièrement dans la capacité des modèles pré-entrainés à contextualiser le code et ainsi appréhender des cas limites qui auraient pu échapper à une batterie de tests prédéfinis. L'utilisation combinée de différents modèles, orchestrée par le mécanisme de fusion pondérée, permet une évaluation holistique qui transcende les limitations intrinsèques des cas de test, lesquels sont par nature restreints aux scénarios explicitement programmés. Ainsi, malgré un temps de traitement plus long, cette approche se révèle plus efficiente en termes de couverture d'évaluation et de capacité d'adaptation, éliminant la nécessité de maintenir et de mettre à jour constamment une base exhaustive de cas de test.

Dans le cadre des EIAH, l'intégration des modèles Chains, conjuguée aux modèles de langages pré-entraînés tels qu'Unix (Guo et al. 2022), CodeT5 (Wang et al. 2021), et CodeBERT (Feng et al. 2020), peut transformer l'enseignement et l'évaluation des compétences en programmation. Cette synergie offre une approche nettement plus personnalisée, réactive et efficiente, enrichissant ainsi l'apprentissage des étudiants. Non seulement cette transformation améliore-t-elle l'expérience éducative des apprenants, mais elle dote également les enseignants d'outils puissants pour appréhender et répondre avec précision aux besoins spécifiques de chaque apprenant.

6.5 Limitations

Bien que cette contribution fournisse des résultats prometteurs, certaines limitations méritent une investigation plus approfondie. Ces aspects incluent plusieurs points importants qui, s'ils sont pris en compte, pourraient améliorer davantage la performance et l'adaptabilité du modèle dans divers contextes.

1. Biais potentiels dans l'ensemble de données

L'ensemble de données actuel peut présenter des biais envers des langages de programmation spécifiques ou des populations d'étudiants. Par exemple, si l'ensemble de données est dominé par des codes écrits en C/C++ et Java, le modèle pourrait ne pas bien généraliser à d'autres langages de programmation comme Python ou JavaScript. De plus, des biais démographiques peuvent exister si l'ensemble de données est majoritairement constitué de travaux d'étudiants d'une région géographique spécifique ou d'un niveau d'expérience particulier. Ces biais peuvent limiter la généralisabilité et l'équité du modèle dans différents environnements de programmation et démographies d'étudiants.

2. Qualité et représentativité des données

L'efficacité de ce modèle est intrinsèquement liée à la qualité et à la représentativité de l'ensemble de données utilisé pour l'entraînement et l'évaluation. Malgré nos efforts pour élaborer des données diversifiées, il peut exister des biais ou des limitations inhérents à l'ensemble de données qui pourraient influencer la généralisabilité et l'équité de notre modèle. Pour cela, élargir la portée et la diversité des ensembles de données de collecte de code pourrait améliorer la généralisabilité du modèle et garantir l'équité dans différents paradigmes de programmation et contextes étudiants. Cela pourrait impliquer l'inclusion d'une gamme plus large de langages de programmation, de niveaux de compétence et de types de problèmes. Par exemple, intégrer des codes provenant de concours de programmation, de projets open-source,

ou de différentes institutions académiques pourrait offrir un ensemble de données plus représentatif.

3. **Pertinence des métriques d'évaluation**

Bien que les métriques choisies, comme la précision, fournissent des informations précieuses, elles pourraient ne pas capturer pleinement les résultats d'apprentissage souhaités. La sélection de métriques d'évaluation appropriées est cruciale pour évaluer précisément la performance de notre modèle. Bien que nous ayons utilisé des métriques établies, un examen plus approfondi de leur pertinence et des biais potentiels est justifié pour assurer une évaluation complète de l'efficacité des différents modèles. Par exemple, des métriques comme la recall, la F1-score, ou des mesures spécifiques aux tâches de classification multi-étiquettes pourraient offrir une vision plus complète de la performance du modèle. De plus, l'analyse qualitative des erreurs du modèle pourrait révéler des perspectives importantes pour améliorer les métriques de performance.

4. **Incorporation des connaissances spécifiques au domaine**

La performance du modèle pourrait être améliorée en incorporant des connaissances spécifiques au domaine ou en explorant des paradigmes d'apprentissage alternatifs tels que l'apprentissage actif. L'intégration de règles spécifiques à la syntaxe et aux structures des langages de programmation utilisés, ou l'amélioration des techniques d'apprentissage semi-supervisé et supervisé déjà adoptées, pourrait renforcer davantage les capacités du modèle. De plus, l'incorporation de rétroactions en temps réel pourrait aider à adapter le modèle aux besoins individuels des étudiants, améliorant ainsi les résultats d'apprentissage.

5. **Scalabilité et mise en œuvre pratique**

L'intégration des modèles Chains dans les plates-formes éducatives existantes peut faciliter la collecte de données, la fourniture de rétroaction en temps réel et la création de parcours d'apprentissage personnalisés, comblant ainsi le fossé entre la recherche et la mise en œuvre pratique. Cependant, il existe des limitations liées à l'infrastructure et aux ressources informatiques qui pourraient affecter la mise en œuvre et la scalabilité de notre approche. Par exemple, des contraintes sur les serveurs, la disponibilité des GPU pour l'inférence en temps réel, et les défis de déploiement dans des environnements éducatifs variés doivent être soigneusement étudiés et surmontés. En abordant ces limitations et en explorant ces pistes de recherche, nous visons à affiner le modèle Chains pour obtenir des expériences et des résultats d'apprentissage encore meilleurs pour les étudiants.

Malgré les avancées prometteuses de notre modèle, une exploration et une optimisation continues sont nécessaires pour surmonter les limitations actuelles et maximiser l'impact éducatif. Par une amélioration constante de la qualité des données, la diversification des contextes d'apprentissage, et l'adoption de nouvelles méthodologies d'évaluation, nous espérons développer un modèle robuste, équitable et largement applicable.

7 Conclusion

Cette étude a examiné l'efficacité des modèles de langage pré-entraînés et a introduit une méthode de fusion pondérée pour améliorer la classification multi-étiquettes de la compréhension du code chez les apprenants novices. Les résultats révèlent que les modèles de langage spécifiquement entraînés pour des tâches de programmation, tels que CodeBERT et UniXcoder, surpassent significativement leurs homologues généraux comme BERT et RoBERTa dans l'évaluation de la compréhension du code, confirmant ainsi notre hypothèse initiale (H1). Ces découvertes soulignent l'importance d'adapter les modèles de langage aux

domaines spécifiques pour obtenir des performances optimales et fournir des évaluations précises et fiables des compétences en programmation.

L'introduction de la méthode de fusion pondérée a démontré son efficacité en combinant les forces des différents modèles, ce qui a conduit à une amélioration significative de la précision sur toutes les étiquettes, en particulier pour des concepts complexes tels que les instructions conditionnelles et les opérations arithmétiques, validant ainsi notre seconde hypothèse (H2).

Les résultats obtenus avec UniXcoder ont montré que ce modèle surpasse même CodeBERT dans certaines mesures clés, ce qui confirme notre troisième hypothèse (H3) selon laquelle l'architecture optimisée pour les tâches de programmation de UniXcoder offre des performances supérieures en termes de précision. Cette constatation met en lumière l'importance de la formation spécifique au domaine pour capturer les subtilités des langages de programmation et des structures de code.

Les implications théoriques de cette étude sont vastes, enrichissant notre compréhension de la manière dont les PLM peuvent être adaptés et perfectionnés pour des tâches spécialisées. Cette recherche corrobore l'idée que la formation spécifique au domaine et les approches d'ensemble, telles que la fusion pondérée, sont essentielles pour optimiser les performances des modèles dans des contextes spécialisés. Dans un contexte éducatif, les implications pratiques sont tout aussi significatives. L'application de ces modèles peut transformer l'évaluation de la compréhension du code des étudiants, permettant une personnalisation accrue des parcours d'apprentissage et une réactivité plus grande aux besoins individuels des apprenants. L'intégration de ces modèles dans des plateformes d'apprentissage adaptatif peut ajuster dynamiquement le contenu d'apprentissage et le niveau de difficulté en fonction de la progression des apprenants, assurant un engagement continu et une amélioration des résultats d'apprentissage.

Malgré les résultats prometteurs de cette étude, certaines limitations méritent une investigation plus approfondie. Les biais potentiels dans l'ensemble de données, la qualité et la représentativité des données, et la pertinence des métriques d'évaluation sont des aspects cruciaux à considérer pour améliorer la généralisabilité du modèle. Les futures recherches devraient se concentrer sur l'impact de cette approche sur les résultats d'apprentissage des étudiants et sur l'intégration avec les technologies éducatives existantes.

CONCLUSION GENERALE

Conclusion Générale

La thèse présentée ici s'inscrit dans un contexte académique où les défis de l'apprentissage en algorithmique, particulièrement dans un environnement informatisé pour l'apprentissage humain, nécessitent des solutions efficaces et adaptées. Ce travail a exploré les moyens de pallier les problèmes liés au déséquilibre des classes dans les jeux de données éducatifs, en se concentrant sur l'amélioration des processus d'apprentissage à travers l'intégration de techniques d'intelligence artificielle, notamment le rééquilibrage des données et l'utilisation de modèles pré-entraînés.

1 Applicabilité des Contributions

Les contributions de cette thèse, notamment les approches développées pour traiter le déséquilibre des classes et évaluer la compréhension des concepts de programmation, ont démontré leur efficacité dans les contextes expérimentaux. Les techniques telles que SMOTE, SMOTE Borderline, SMOTE-ENN, et ADASYN ont permis de rééquilibrer les données, améliorant ainsi les performances des modèles de classification. Plus spécifiquement, l'introduction du modèle Equi-Fused-Data-based SMOTE, qui combine des techniques de suréchantillonnage avec l'apprentissage ensembliste, s'est avérée particulièrement efficace pour les données déséquilibrées dans des contextes éducatifs multi-classes.

1.1 Résultats obtenus

Les résultats obtenus dans cette thèse montrent une amélioration significative des performances des modèles de classification grâce aux différentes techniques développées et testées, notamment en ce qui concerne la précision, la réduction des erreurs, et l'efficacité des méthodes ensemblistes et des modèles pré-entraînés.

- Amélioration de la précision des modèles: l'application des techniques de rééquilibrage des données a conduit à des taux d'exactitude remarquables. Par exemple, la régression logistique a atteint une exactitude de 95,50% avec la méthode ADASYN, tandis que le SVM a obtenu une précision de 95,12% après l'application de SMOTE Borderline, démontrant une amélioration considérable par rapport à l'ensemble de données non équilibré où ces modèles présentaient des taux d'exactitude inférieurs, autour de 91,79%.
- L'approche Equi-Fused-Data-based SMOTE, qui combine plusieurs techniques de rééchantillonnage avec des méthodes ensemblistes comme le Balanced Bagging, a également permis d'améliorer la précision globale des modèles. Par exemple, cette méthode a permis d'atteindre une AUC de 98,20% pour le modèle SVM, illustrant sa capacité à mieux discriminer entre les classes.
- Réduction des erreurs de classification: les méthodes de suréchantillonnage ont significativement réduit le taux de faux positifs (FPR) dans les modèles de classification.
    - Avec SMOTE Borderline, le FPR du SVM est descendu à **3,31%**, et celui de la régression logistique a atteint un **minimum** de **2,30%,** comparé à 6,32% et 6,54% respectivement pour les données non équilibrées.
    - De plus, la méthode Equi-Fused-Data-based SMOTE a démontré une efficacité particulière en réduisant les erreurs grâce à l'application conjointe de SMOTE, SMOTE Borderline, SMOTE-ENN, et ADASYN, ce qui a permis de stabiliser

les performances et de minimiser les erreurs de classification dans les scénarios les plus complexes (un taux de faux positif égal à 1.97%).

- Efficacité des méthodes ensemblistes
  - L'approche Balanced Bagging, intégrée dans la méthode Equi-Fused-Data-based SMOTE, a montré une augmentation de la performance des modèles. Par exemple, l'utilisation de Balanced Bagging a permis de stabiliser les performances du modèle, qui a atteint une AUC de 98,94%.
  - L'approche fusion pondérée a été particulièrement efficace pour l'évaluation de la compréhension des étudiants en classification multi-label, en se concentrant sur des concepts spécifiques de programmation. Contrairement à l'utilisation de techniques comme SMOTE, cette approche a permis de pondérer les performances des modèles selon les étiquettes spécifiques, ce qui a conduit à une évaluation plus fine et adaptée de la maîtrise des concepts algorithmiques par les étudiants.
- Amélioration grâce aux modèles pré-entraînés : l'intégration de modèles de langage pré-entraînés comme BERT, RoBERTa, CodeBERT, et UniXcoder a joué un rôle déterminant dans l'évaluation de la compréhension du code par les étudiants. Ces modèles, spécifiquement conçus pour saisir les subtilités du code et du langage naturel, ont permis de mieux capturer les nuances et la sémantique des soumissions des étudiants. CodeBERT et UniXcoder, en particulier, ont montré des performances supérieures pour l'évaluation des tâches de programmation, surpassant les modèles génériques comme BERT et RoBERTa. Par exemple, UniXcoder a atteint une précision globale de 92% dans la tâche d'évaluation multi-label, avec une réduction significative des erreurs de classification pour les concepts algorithmiques complexes.
- Optimisation des ressources éducatives : les améliorations apportées aux modèles ont des implications directes pour l'optimisation des ressources éducatives. Grâce à une précision accrue, ces modèles permettent de mieux cibler les étudiants nécessitant un soutien pédagogique. Par exemple, les gains obtenus avec Equi-Fused-Data permettent d'affiner les prédictions et de réduire les interventions inutiles, tout en améliorant l'efficacité des actions correctives pour les étudiants en difficulté.

1.2 Applicabilité Pratique

L'applicabilité de ces résultats va au-delà du cadre académique. Les techniques développées et testées dans cette thèse peuvent être intégrées dans des systèmes éducatifs informatisés pour fournir un retour d'information personnalisé aux étudiants en temps réel. Par exemple:

- Implémentation dans les plateformes EIAH: les EIAH peuvent intégrer ces modèles pour évaluer automatiquement et de manière précise les compétences des étudiants, permettant ainsi d'adapter les parcours pédagogiques en fonction des besoins individuels.
- Soutien pédagogique ciblé: les prédictions fournies par les modèles permettent aux enseignants de mieux cibler leur soutien pédagogique, en identifiant rapidement les étudiants à risque de décrochage ou de faible performance.
- Amélioration continue: les systèmes basés sur ces modèles peuvent être utilisés pour améliorer en continu les stratégies d'enseignement, en fournissant des données précieuses sur l'efficacité des méthodes pédagogiques appliquées.

Ces résultats montrent que les techniques développées dans le cadre de cette thèse offrent des solutions efficaces pour gérer les défis posés par les jeux de données déséquilibrés, en particulier dans le contexte de l'éducation algorithmique. L'amélioration des performances des modèles de classification, associée à une réduction des erreurs, permet de proposer des outils

pédagogiques plus adaptés et plus performants, répondant ainsi aux besoins spécifiques des apprenants. N'est-ce pas précisément pour eux, les apprenants, que les environnements informatiques pour l'apprentissage humain sont au cœur de ces développements?

2 Perspectives

Les contributions de cette thèse ouvrent la voie à des avancées notables dans plusieurs domaines liés à l'éducation assistée par l'intelligence artificielle. En s'appuyant sur les résultats obtenus, plusieurs pistes de recherche et d'application peuvent être envisagées pour approfondir et étendre l'impact des techniques développées.

Amélioration des techniques de rééquilibrage des données dans des contextes complexes

Bien que les techniques actuelles de rééquilibrage des données, comme SMOTE et ADASYN, aient prouvé leur efficacité, il reste encore des défis à relever, notamment dans les contextes de données hétérogènes et multi-étiquettes. Le développement de nouvelles méthodes de rééquilibrage qui tiennent compte des interactions complexes entre les étiquettes et les caractéristiques des données pourrait améliorer encore la précision des modèles. Ainsi, la conception d'algorithmes de rééquilibrage adaptatifs qui ajustent dynamiquement les stratégies de rééchantillonnage en fonction de la structure des données, afin d'améliorer les performances dans des environnements éducatifs variés, est une piste de recherche à explorer.

Exploration des modèles de langage de grande taille (LLMs)

Avec l'émergence des **modèles de langage de grande taille (LLMs) tels que GPT-3 et GPT-4, une opportunité se présente pour améliorer encore davantage les systèmes d'évaluation automatisée de la compréhension en programmation. Ces modèles, entraînés sur des corpus massifs et dotés de capacités de génération textuelle plus fines que les modèles pré-entraînés traditionnels, pourraient offrir une analyse encore plus précise et nuancée des réponses des étudiants. Ainsi, il serait possible d'explorer l'efficacité des LLMs pour capter des nuances plus complexes dans les réponses des étudiants, et examiner leur potentiel pour identifier des concepts algorithmiques à des niveaux plus avancés.

Personnalisation de l'apprentissage à grande échelle

L'une des grandes promesses de l'IA dans l'éducation est la personnalisation des parcours d'apprentissage. En s'appuyant sur les techniques développées dans cette thèse, il serait possible d'améliorer encore cette personnalisation, en intégrant des modèles prédictifs capables d'identifier les besoins spécifiques des étudiants de manière anticipative, en tenant compte non seulement de leurs performances passées, mais aussi de leur profil cognitif et comportemental. A titre illustratif, développer des systèmes d'apprentissage adaptatif qui exploitent les données multi-sources pour créer des profils étudiants plus complets, permettant ainsi une personnalisation des contenus pédagogiques en temps réel, adaptée à chaque étudiant.

Intégration de l'IA pour la remédiation pédagogique

Les systèmes d'IA peuvent non seulement évaluer les performances des étudiants, mais aussi proposer des stratégies de remédiation adaptées. En s'appuyant sur les résultats des modèles développés dans cette thèse, une piste de recherche consiste à utiliser ces modèles pour diagnostiquer les difficultés spécifiques des étudiants et leur proposer des solutions ciblées, sous forme d'exercices adaptés ou de recommandations pédagogiques. Par exemple, concevoir des environnements d'apprentissage intelligents où les modèles de classification multi-label ne se contentent pas de détecter les lacunes des étudiants, mais proposent également des

interventions pédagogiques personnalisées pour combler ces lacunes tout en gardant une analyse fine de leur compréhension générale.

Applications des modèles multi-label dans d'autres disciplines

Si les résultats de cette thèse se concentrent sur l'enseignement de la programmation, les techniques développées peuvent être étendues à d'autres domaines éducatifs. Par exemple, les systèmes de classification multi-label peuvent être utilisés dans des domaines, où la compréhension des concepts est essentielle et peut être difficile à évaluer de manière automatisée.

Développement de systèmes d'évaluation en temps réel

Enfin, une autre perspective concerne le développement de systèmes capables d'évaluer la compréhension des étudiants en temps réel, tout au long de leur apprentissage. En utilisant des modèles de langage avancés et des techniques de rééquilibrage des données, il serait possible de créer des systèmes qui suivent les progrès des étudiants de manière continue et adaptent les évaluations et les exercices en conséquence.

Ces perspectives illustrent le potentiel considérable des techniques développées dans cette thèse pour améliorer l'éducation assistée par l'intelligence artificielle, en ouvrant la voie à des environnements d'apprentissage plus personnalisés, adaptatifs et efficaces.

Références

.C. Clark and R.E. Mayer. (2016). e-Learning : Promise and Pitfalls. In e-Learning and the Science of Instruction (p. 7-28). https://doi.org/10.1002/9781119239086.ch1

Rana, A. Singh Rawat, A. Bijalwan, & H. Bahuguna. (2018). Application of Multi Layer (Perceptron) Artificial Neural Network in the Diagnosis System : A Systematic Review. 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE), 1-6. https://doi.org/10.1109/RICE.2018.8509069

Von Mayrhauser et A. M. Vans. 1995. « Program comprehension during software maintenance and evolution ». Computer 28(8):44-55. doi: 10.1109/2.402076.

Abdessemed, M., Bensebaa, T., Belhaoues, T. E., & Bey, A. (2018). Automatic exercise sequencing-based algorithmic skills. International Journal of Innovation and Learning, 23:1, 104-121. https://doi.org/10.1504/ijil.2018.088788

Abed, Q. A., Fadhil, O. M., & Al-Yaseen, W. L. (2020). Data mining in web personalization using the blended deep learning model. Indonesian Journal of Electrical Engineering and Computer Science, 20, 1507-1512.

Ahmadzadeh, Marzieh, Dave Elliman, et Colin Higgins. 2005. « An analysis of patterns of debugging among novice computer science students ». SIGCSE Bull. 37(3):84-88. doi: 10.1145/1151954.1067472.

Aho, Alfred V., et Alfred V. Aho, éd. 2007. Compilers: Principles, Techniques, & Tools. 2nd ed. Boston: Pearson/Addison Wesley.

Akçapınar, G., Chen, M.-R., Majumdar, R., Flanagan, B., & Ogata, H. (2020). Exploring Student Approaches to Learning through Sequence Analysis of Reading Logs. https://doi.org/10.1145/3375462.3375492

Akiguet-Bakong, S. (2008). Effet de la compréhension de l'énoncé sur la résolution des problèmes arithmétiques. Corela, 6-2. https://doi.org/10.4000/corela.300

Akter, S., Dwivedi, Y. K., Sajib, S., Biswas, K., Bandara, R. J., & Michael, K. (2022). Algorithmic bias in machine learning-based marketing models. Journal of Business Research, 144, 201-216. https://doi.org/10.1016/j.jbusres.2022.01.083

Al-Ashoor, A., & Abdullah, S. (2022). Examining Techniques to Solving Imbalanced Datasets in Educational Data Mining Systems. International Journal of Computing, 21(2), 205-213. https://doi.org/10.47839/ijc.21.2.2589

Al-Mouh, N. A., Al-Khalifa, A. S., & Al-Khalifa, H. S. (2014). A First Look into MOOCs Accessibility. In K. Miesenberger, D. Fels, D. Archambault, P. Peňáz, & W. Zagler (Éds.), Computers Helping People with Special Needs (p. 145-152). Springer International Publishing.

Al-Othman, M. A., Cole, J. H., Zoltowski, C. B., & Peroulis, D. (2017). An Adaptive Educational Web Application for Engineering Students. IEEE Access, 5, 359-365. https://doi.org/10.1109/ACCESS.2016.2643164

Aleven, V., Mclaren, B., Roll, I., & Koedinger, K. (2006). Toward Meta-cognitive Tutoring : A Model of Help Seeking with a Cognitive Tutor. I. J. Artificial Intelligence in Education, 16, 101-128.

Alfred, M. C., Neyens, D. M., & Gramopadhye, A. K. (2019). Learning in simulated environments : An assessment of 4-week retention outcomes. Applied Ergonomics, 74, 107-117. https://doi.org/10.1016/j.apergo.2018.08.002

Alkhatlan, A., & Kalita, J. (2018). Intelligent Tutoring Systems : A Comprehensive Historical Survey with Recent Developments (arXiv:1812.09628). arXiv. http://arxiv.org/abs/1812.09628

Almeida, F., & Winfield, C. (2012). Reading in the brain by Stanislas Dehaene. Ilha do Desterro A Journal of English Language, Literatures in English and Cultural Studies, 0. https://doi.org/10.5007/2175-8026.2012n63p214

Alsumait, A., & Al-Osaimi, A. (2009). Usability Heuristics Evaluation for Child E-learning Applications. In Journal of Software (Vol. 5, p. 430). https://doi.org/10.4304/jsw.5.6.654-661

Alyahyan, E., & Düştegör, D. (2020). Predicting academic success in higher education : Literature review and best practices. International Journal of Educational Technology in Higher Education, 17(1), 3. https://doi.org/10.1186/s41239-020-0177-7

Alzanin, S. M., Gumaei, A., Haque, M. A., & Muaad, A. Y. (2023). An Optimized Arabic Multilabel Text Classification Approach Using Genetic Algorithm and Ensemble Learning. Applied Sciences, 13(18), 10264. https://doi.org/10.3390/app131810264

Alzoubi, A., Aqeel, I., & Alzoubi, H. (2024). Review of Artificial Intelligence and Machine Learning Recent Advancements (p. 223-236). https://doi.org/10.1007/978-3-031-55221-2_14

Amrieh, E. A., Hamtini, T., & Aljarah, I. (2016). Mining Educational Data to Predict Student's academic Performance using Ensemble Methods. International Journal of Database Theory and Application, 9(8), 119-136. https://doi.org/10.14257/ijdta.2016.9.8.13

Anderson, L., DR, K., PW, A., KA, C., Mayer, R., PR, P., Raths, J., & MC, W. (2001). A Taxonomy for Learning, Teaching, and Assessing : A Revision of Bloom's Taxonomy of Educational Objectives.

Anderson, T., & Elloumi, F. (2004). Theory and Practice of Online Learning. http://lst-iiep.iiep-unesco.org/cgi-bin/wwwi32.exe/[in=epidoc1.in]/?t2000=020568/(100).

Anon. 2013. « RAPTOR - A Vehicle to Enhance Logical Thinking ». 12.

Anthonysamy, L. (2021). The use of metacognitive strategies for undisrupted online learning : Preparing university students in the age of pandemic. Education and Information Technologies, 26. https://doi.org/10.1007/s10639-021-10518-y

Ashman, H., Brailsford, T., Cristea, A. I., Sheng, Q. Z., Steward, C., Toms, E. G., & Wade, V. (2014). The ethical and social implications of personalization technologies for e-learning. Information & Management, 51(6), 819-832. https://doi.org/10.1016/j.im.2014.04.003

Asif, M., & Krogstie, J. (2012). Research Issues in Personalization of Mobile Services. International Journal of Information Engineering and Electronic Business, 4(4), 1-8. https://doi.org/10.5815/ijieeb.2012.04.01

Adelson et E. Soloway. 1985. « The Role of Domain Expenence in Software Design ». IEEE Transactions on Software Engineering SE-11(11):1351-60. doi: 10.1109/TSE.1985.231883.

Bacchelli, Alberto, et Christian Bird. 2013. « Expectations, outcomes, and challenges of modern code review ». P. 712-21 in 2013 35th International Conference on Software Engineering (ICSE).

Baidada, M., Mansouri, K., & Poirier, F. (2019). Hybrid Recommendation Approach in Online Learning Environments. In Á. Rocha & M. Serrhini (Éds.), Information Systems and Technologies to Support Learning (Vol. 111, p. 39-43). Springer International Publishing. https://doi.org/10.1007/978-3-030-03577-8_5

Baker, R., & Siemens, G. (2014). Educational data mining and learning analytics (p. 253-272). https://doi.org/10.1017/CBO9781139519526.016

Barnes, T., Richter, H., Powell, E., Chaffin, A., & Godwin, A. (2007). Game2Learn : Building CS1 learning games for retention. ACM SIGCSE Bulletin, 39(3), 121. https://doi.org/10.1145/1269900.1268821

Barros, T. M., SouzaNeto, P. A., Silva, I., & Guedes, L. A. (2019). Predictive Models for Imbalanced Data : A School Dropout Perspective. Education Sciences, 9(4), 275. https://doi.org/10.3390/educsci9040275

Batista, G., Prati, R., & Monard, M.-C. (2004). A Study of the Behavior of Several Methods for Balancing machine Learning Training Data. SIGKDD Explorations, 6, 20-29. https://doi.org/10.1145/1007730.1007735

Baylari, A., & Montazer, Gh. A. (2009). Design a personalized e-learning system based on item response theory and artificial neural network approach. Expert Systems with Applications, 36(4), 8013-8021. https://doi.org/10.1016/j.eswa.2008.10.080

Bayman, Piraye, et Richard E. Mayer. 1983. « A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements ». Communications of the ACM 26(9):677-79. doi: 10.1145/358172.358408.

Beaubouef, Theresa, et John Mason. 2005. « Why the high attrition rate for computer science students: some thoughts and observations ». SIGCSE Bull. 37(2):103-6. doi: 10.1145/1083431.1083474.

Becker, B. W. (2001). Teaching CS1 with karel the robot in Java. ACM SIGCSE Bulletin, 33(1), 50-54. https://doi.org/10.1145/366413.364536

Ben-Ari, Mordechai. 2001. « Constructivism in computer science education. » Journal of Computers in Mathematics and Science Teaching 20(1):45-73.

Bengio, Y., & Lecun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Éds.), Large-scale kernel machines. MIT Press.

Bennedsen, Jens, et Michael E. Caspersen. 2007. « Failure rates in introductory programming ». SIGCSE Bull. 39(2):32-36. doi: 10.1145/1272848.1272879.

Bennett, R. (1998). Reinventing Assessment. Speculations on the Future of Large-Scale Educational Testing. A Policy Information Perspective.

Bennett, R. E. (2011). Formative assessment : A critical review. Assessment in Education: Principles, Policy & Practice, 18(1), 5-25. https://doi.org/10.1080/0969594X.2010.513678

Bergin, Susan, et Ronan Reilly. 2005. « Programming: factors that influence success ». SIGCSE Bull. 37(1):411-15. doi: 10.1145/1047124.1047480.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. J. Mach. Learn. Res., 13(null), 281-305.

Bertilsson, F., Stenlund, T., Wiklund-Hörnqvist, C., & Jonsson, B. (2020). Retrieval Practice : Beneficial for All Students or Moderated by Individual Differences? Psychology Learning & Teaching, 20, 147572572097349. https://doi.org/10.1177/1475725720973494

Bey, Anis, et Tahar Bensebaa. 2011. « An eAssessment Approach of Algorithmic ProblemSolving Skills ». International Journal of Computer Applications 25(10):15-21. doi: 10.5120/3150-4354.

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When Is "Nearest Neighbor" Meaningful? In C. Beeri & P. Buneman (Éds.), Database Theory—ICDT'99 (p. 217-235). Springer Berlin Heidelberg.

Biau, G., & Scornet, E. (2016). A random forest guided tour. TEST, 25(2), 197-227. https://doi.org/10.1007/s11749-016-0481-7

Biau, G., Devroye, L., & Lugosi, G. (2008). Consistency of Random Forests and Other Averaging Classifiers. Journal of Machine Learning Research, 9, 2015-2033. https://doi.org/10.1145/1390681.1442799

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., & Van Der Wal, O. (2023). Pythia : A Suite for Analyzing Large Language Models Across Training and Scaling. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Éds.), Proceedings of the 40th International Conference on Machine Learning (Vol. 202, p. 2397-2430). PMLR. https://proceedings.mlr.press/v202/biderman23a.html

Biggs, John. 2003. Teaching for Quality Learning at University.

Binns, R. (2017). Fairness in Machine Learning : Lessons from Political Philosophy.

Bishop, C. (2006). Pattern Recognition and Machine Learning. In Journal of Electronic Imaging (Vol. 16, p. 140-155). https://doi.org/10.1117/1.2819119

Bonar, J., & Cunningham, R. (1988). Bridge : Intelligent Tutoring with Intermediate Representations. 20.

Bonar, Jeffrey, et Elliot Soloway. 1985. « Preprogramming Knowledge: A Major Source of Misconceptions in Novice Programmers ». Human–Computer Interaction 1(2):133-61. doi: 10.1207/s15327051hci0102_3.

Bonnat, C., Marzin, P., Girault, I., & d'Ham, C. (2018). Modélisation didactique pour la conception d'étayages dans un EIAH : Exemple d'une activité de conception expérimentale en biologie. Sciences et Technologies de l'Information et de la Communication pour l'Éducation et la Formation, 25(2), 31-61. https://doi.org/10.3406/stice.2018.1766

Bonthu, H. (2021, juillet 6). Start Learning SVM (Support Vector Machine) Algorithm Here! Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/07/svm-support-vector-machine-algorithm/

Booch, Grady, James Rumbaugh, et Ivar Jacobson. 1999. « Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series) ». J. Database Manag. 10.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 144-152. https://doi.org/10.1145/130385.130401

Bouchet, F., Harley, J. M., Trevors, G. J., & Azevedo, R. (2013). Clustering and Profiling Students According to their Interactions with an Intelligent Tutoring System Fostering Self-Regulated Learning. 5(1), 43.

Bouhineau, D. (2013). Utilisation de traits sémantiques pour une méthodologie de construction d'un système d'aide dans un EIAH de l'algorithmique. 13. https://doi.org/hal-00862437f

Bouhineau, Denis, Vanda Luengo, et Nadine Mandran. 2018. « Concevoir, produire, décrire, évaluer, améliorer, partager et conserver des données, opérateurs, processus d'analyse et résultats d'études sur les logs d'activités d'apprentissages humains avec ordinateur ». 26.

Boyce, A., & Barnes, T. (2010). BeadLoom Game : Using game elements to increase motivation and learning. Proceedings of the Fifth International Conference on the Foundations of Digital Games - FDG '10, 25-31. https://doi.org/10.1145/1822348.1822352

Boyle, E. A., Hainey, T., Connolly, T. M., Gray, G., Earp, J., Ott, M., Lim, T., Ninaus, M., Ribeiro, C., & Pereira, J. (2016). An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games. Computers & Education, 94, 178-192. https://doi.org/10.1016/j.compedu.2015.11.003

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, 30(7), 1145-1159. https://doi.org/10.1016/S0031-3203(96)00142-2

Bransford, J. (2000). How People Learn : Brain, Mind, Experience, and School : Expanded Edition (2000).

Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), 123-140. https://doi.org/10.1007/BF00058655

Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

Breslow, L., & Aha, D. (1996). Simplifying Decision Trees : A Survey. Knowl. Eng. Rev., 12(1). https://doi.org/10.1017/S0269888997000015

Brisson, A., Pereira, G., Prada, R., Paiva, A., Louchart, S., Suttie, N., Lim, T., Lopes, R., Bidarra, R., & Bellotti, F. (2012). Artificial intelligence and personalization opportunities for serious games. Proc. of 8th AIIDE Conf, 51-57.

Brooks, Ruven. 1983. « Towards a theory of the comprehension of computer programs ». International Journal of Man-Machine Studies 18(6):543-54. doi: 10.1016/S0020-7373(83)80031-5.

Brown, G. (2017). Assessment of Student Achievement. In Assessment of Student Achievement (p. 154). https://doi.org/10.4324/9781315162058

Bull, S., & Kay, J. (2007). STUDENT MODELS THAT INVITE THE LEARNER IN: THE SMILI ☺ OPEN LEARNER MODELLING FRAMEWORK TECHNICAL REPORT 580. I. J. Artificial Intelligence in Education, 17.

Buolamwini, J., & Gebru, T. (2018). Gender Shades : Intersectional Accuracy Disparities in Commercial Gender Classification. In S. A. Friedler & C. Wilson (Éds.), Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Vol. 81, p. 77-91). PMLR. https://proceedings.mlr.press/v81/buolamwini18a.html

Chachoua, S. (2019). Contribution à l'évaluation de l'apprenant et l'adaptation pédagogique dans les plateformes d'apprentissage : Une approche fondée sur les traces. 179.

Chachoui, Yasmine, Nabiha Azizi, Richard Hotte, et Tahar Bensebaa. 2024. « Enhancing algorithmic assessment in education: Equi-fused-data-based SMOTE for balanced learning ». Computers and Education: Artificial Intelligence 6:100222. doi: 10.1016/j.caeai.2024.100222.

Chandra, M., & Bedi, S. (2018). Survey on SVM and their application in image classification. International Journal of Information Technology, 13. https://doi.org/10.1007/s41870-017-0080-1

Chaouachi, Maher, Imène Jraidi, Susanne P. Lajoie, et Claude Frasson. 2019. « Enhancing the Learning Experience Using Real-Time Cognitive Evaluation ». International Journal of Information and Education Technology 9(10):678-88. doi: 10.18178/ijiet.2019.9.10.1287.

Chauhan, V. K., Dahiya, K., & Sharma, A. (2019). Problem formulations and solvers in linear SVM: a review. Artificial Intelligence Review, 52(2), 803-855. https://doi.org/10.1007/s10462-018-9614-6

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE : Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321-357. https://doi.org/10.1613/jair.953

Chen, C.-M. (2008). Intelligent web-based learning system with personalized learning path guidance. Computers & Education, 51(2), 787-814. https://doi.org/10.1016/j.compedu.2007.08.004

Chen, C., & Breiman, L. (2004). Using Random Forest to Learn Imbalanced Data. University of California, Berkeley.

Chen, T., & Guestrin, C. (2016). XGBoost : A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. https://doi.org/10.1145/2939672.2939785

Chen, X., Cheng, J., Liu, J., Xu, W., Hua, S., Tang, Z., & Sheng, V. S. (2022). A Survey of Multi-label Text Classification Based on Deep Learning. In X. Sun, X. Zhang, Z. Xia, & E. Bertino (Éds.), Artificial Intelligence and Security (p. 443-456). Springer International Publishing.

Chi, M. T. H., & Wylie, R. (2014). The ICAP Framework : Linking Cognitive Engagement to Active Learning Outcomes. Educational Psychologist, 49(4), 219-243. https://doi.org/10.1080/00461520.2014.965823

Chi, Michelene T. H., et Ruth Wylie. 2014. « The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes ». Educational Psychologist 49(4):219-43. doi: 10.1080/00461520.2014.965823.

Chiu, Chiung-Fang, et Hsing-Yi Huang. 2015. « Guided Debugging Practices of Game Based Programming for Novice Programmers ». International Journal of Information and Education Technology 5(5):343-47. doi: 10.7763/IJIET.2015.V5.527.

Clerc, F., Lefevre, M., Guin, N., & Marty, J.-C. (2015). Mise en place de la personnalisation dans le cadre des MOOCs. 13.

CodeSerra. (2022, juillet 23). CodeBERT. Medium. https://codeserra.medium.com/codebert-83171b23c33c

Coffield, Ecclestone, K., Moseley, & Hall, E. (2004). Learning styles and pedagogy in post 16 education : A critical and systematic review.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., & Stoyanov, V. (2020). Unsupervised Cross-lingual Representation Learning at Scale. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Éds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (p. 8440-8451). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.747

Cooke, Nancy J., et Roger W. Schvaneveldt. 1988. « Effects of Computer Programming Experience on Network Representations of Abstract Programming Concepts ». International Journal of Man-Machine Studies 29(4):407-27. doi: 10.1016/S0020-7373(88)80003-8.

Coombe, C. (2007). Improving Assessment through Student Involvement. Learning and Teaching in Higher Education: Gulf Perspectives, 4, 47-48. https://doi.org/10.18538/lthe.v4.n1.07

Cooper, S., & Dann, W. (2000). Developing Algorithmic Thinking With Alice. 5.

Cooper, Stephen, Wanda Dann, et Randy Pausch. 2003. « Teaching objects-first in introductory computer science ». P. 191-95 in Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03. New York, NY, USA: Association for Computing Machinery.

Corritore, Cynthia L., et Susan Wiedenbeck. 1991. « What Do Novices Learn during Program Comprehension? » International Journal of Human-Computer Interaction 3(2):199-222. doi: 10.1080/10447319109526004.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. 20, 273-297. https://doi.org/10.1007/BF00994018

Crocker, L., & Algina, J. (1986). Introduction to Classical and Modern Test Theory. Holt, Rinehart and Winston, 6277 Sea Harbor Drive, Orlando, FL 32887 ($44.

Cubero-Ibáñez, J., Ibarra-Sáiz, M. S., & Rodríguez-Gómez, G. (2017). Assessment literacy through serious games in virtual learning environments. Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM 2017, 1-5. https://doi.org/10.1145/3144826.3145375

Cutler, D. R., Edwards Jr., T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). RANDOM FORESTS FOR CLASSIFICATION IN ECOLOGY. Ecology, 88(11), 2783-2792. https://doi.org/10.1890/07-0539.1

D'mello, S. K., & Kory, J. (2015). A Review and Meta-Analysis of Multimodal Affect Detection Systems. ACM Comput. Surv., 47(3). https://doi.org/10.1145/2682899

Dabbagh, N., & Kitsantas, A. (2012). Personal Learning Environments, social media, and self-regulated learning : A natural formula for connecting formal and informal learning. Social Media in Higher Education, 15(1), 3-8. https://doi.org/10.1016/j.iheduc.2011.06.002

Dadhich, Manish, et Amiya Bhaumik. 2023. « Demystification of Generative Artificial Intelligence (AI) Literacy, Algorithmic Thinking, Cognitive Divide, Pedagogical knowledge: A Comprehensive Model ». 2023 IEEE International Conference on ICT in Business Industry & Government (ICTBIG) 1-5.

Dahalan, H. M., & Hussain, R. M. R. (2010). Development of Web-Based Assessment in Teaching and Learning Management System (e-ATLMS). Procedia - Social and Behavioral Sciences, 9, 244-248. https://doi.org/10.1016/j.sbspro.2010.12.144

Daka, Ermira, et Gordon Fraser. 2014. « A Survey on Unit Testing Practices and Problems ». Proceedings - International Symposium on Software Reliability Engineering, ISSRE 201-11. doi: 10.1109/ISSRE.2014.11.

Dalgarno, B., & Lee, M. (2010). What are the learning affordances of 3-D Virtual environments? British Journal of Educational Technology, 41, 10-32. https://doi.org/10.1111/j.1467-8535.2009.01038.x

David W. Hosmer Jr., Stanley Lemeshow, & Rodney X. Sturdivant. (2013). Application of Logistic Regression with Different Sampling Models. In Applied Logistic Regression (p. 227-242). https://doi.org/10.1002/9781118548387.ch6

Davinia, H.-L., Dimitriadis, Y., & Asensio-Pérez, J. (2005). Computational Representation of Collaborative Learning Flow Patterns Using IMS Learning Desing. Educ Technol Soc, 8.

Dekker, G., Pechenizkiy, M., & Vleeshouwers, J. (2009). Predicting Students Drop Out : A Case Study. In Computers, Environment and Urban Systems (p. 50).

Denny, Paul, Brian Hanks, et Beth Simon. 2010. PeerWise: Replication study of a student-collaborative self-testing web service in a U.S. setting.

Dermeval, D., & Bittencourt, I. I. (2020). Co-designing Gamified Intelligent Tutoring Systems with Teachers. 28, 73-91. https://doi.org/10.5753/rbie.2020.28.0.73

Desmarais, M. C., & Baker, R. S. J. d. (2012). A review of recent advances in learner and skill modeling in intelligent learning environments. User Modeling and User-Adapted Interaction, 22(1), 9-38. https://doi.org/10.1007/s11257-011-9106-8

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. North American Chapter of the Association for Computational Linguistics. https://api.semanticscholar.org/CorpusID:52967399

Dey, R., & Mathur, R. (2023). Ensemble Learning Method Using Stacking with Base Learner, A Comparison. In N. Chaki, N. D. Roy, P. Debnath, & K. Saeed (Éds.), Proceedings of International Conference on Data Analytics and Insights, ICDAI 2023 (p. 159-169). Springer Nature Singapore.

Dietterich, T. G. (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees : Bagging, Boosting, and Randomization. Machine Learning, 40(2), 139-157. https://doi.org/10.1023/A:1007607513941

Dodero, J. M., Palomo-Duarte, M., Ruiz-Rube, I., Traverso, I., Mota, J. M., & Balderas, A. (2015). Learning Technologies and Semantic Integration of Learning Resources. IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 10(1), 11-16. https://doi.org/10.1109/RITA.2015.2391312

Domingos, P., & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning, 29(2), 103-130. https://doi.org/10.1023/A:1007413511361

Dong, J., Jiang, Z., Pan, D., Chen, Z., Guan, Q., Zhang, H., Gui, G., & Gui, W. (2024). A survey on confidence calibration of deep learning under class imbalance data. https://doi.org/10.36227/techrxiv.171623912.22862601/v1

Doquire, G., & Verleysen, M. (2013). Mutual information-based feature selection for multilabel classification. Advances in cognitive and ubiquitous computing, 122, 148-155. https://doi.org/10.1016/j.neucom.2013.06.035

Drachsler, H., & Greller, W. (2011). The Pulse of Learning Analytics Understandings and Expectations from the Stakeholders. In ACM International Conference Proceeding Series. https://doi.org/10.1145/2330601.2330634

Drini, Merlinda. 2018. « Using new methodologies in teaching computer programming ». P. 120-24 in 2018 IEEE Integrated STEM Education Conference (ISEC).

du Boulay, B. (2016). Recent Meta-reviews and Meta–analyses of AIED Systems. International Journal of Artificial Intelligence in Education, 26(1), 536-537. https://doi.org/10.1007/s40593-015-0060-1

Du Boulay, Benedict. 1986. « Some Difficulties of Learning to Program ». Journal of Educational Computing Research 2(1):57-73. doi: 10.2190/3LFX-9RRF-67T8-UVK9.

Duda, R., Hart, P., & G.Stork, D. (2001). Pattern Classification. In Wiley Interscience (Vol. xx).

Durand, G. (2007). Pour une scénarisation des activités d'évaluation des apprentissages dans les EIAH. 8.

Durfee, A., Schneberger, S., & Amoroso, D. L. (2007). EVALUATING STUDENTS COMPUTER-BASED LEARNING USING A VISUAL DATA MINING APPROACH. 9(1), 28.

E. Mashhadi & H. Hemmati. (2021). Applying CodeBERT for Automated Program Repair of Java Simple Bugs. 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR), 505-509. https://doi.org/10.1109/MSR52588.2021.00063

Eckerdal, Anna, et Michael Thuné. 2005. « Novice Java programmers' conceptions of "object" and "class", and variation theory ». SIGCSE Bull. 37(3):89-93. doi: 10.1145/1151954.1067473.

Eckerdal, Anna. 2009. « Novice Programming Students' Learning of Concepts and Practise ». Acta Universitatis Upsaliensis, Uppsala.

El-Sabagh, H. A. (2021). Adaptive e-learning environment based on learning styles and its impact on development students' engagement. International Journal of Educational Technology in Higher Education, 18(1), 53. https://doi.org/10.1186/s41239-021-00289-4

Elghibari, F., & Elouahbi, R. (2015). Modèle d'un processus d'apprentissage adapté aux évolutions cognitives du profil de l'apprenant. 1ère Edition du Workshop International sur les Approches Pédagogiques & E-Learning, 5.

Estupiñán, Ricardo, et Lizette Jesús & Martín Estévez. 2019. « Comprehension Higher Education 10.5281/Zenodo.3244297. » Consulté 15 avril 2020 (https://www.researchgate.net/profile/Lizette_Martin_Estevez/publication/334068006_Comprehension_Higher_Education/links/5d155c3892851cf4405174ee/Comprehension-Higher-Education.pdf).

F. P. Deek, M. Turoff, et J. A. McHugh. 1999. « A common model for problem solving and program development ». IEEE Transactions on Education 42(4):331-36. doi: 10.1109/13.804541.

Falkner, Katrina, et Rebecca Vivian. 2015. Coding Across the Curriculum: Resource Review.

Farhan, M., Aslam, M., Jabbar, S., & Khalid, S. (2018). Multimedia based qualitative assessment methodology in eLearning : Student teacher engagement analysis. Multimedia Tools and Applications, 77(4), 4909-4923. https://doi.org/10.1007/s11042-016-4212-6

Farrell, R., Anderson, J., & Reiser, B. (1984). An interactive computer-based tutor for LISP. In Proceedings of the National Conference on Artificial Intelligence (p. 109).

Felder, R. (1988). Learning and Teaching Styles in Engineering Education. Journal of Engineering Education -Washington-, 78, 674-681.

Feng, S., Keung, J., Yu, X., Xiao, Y., & Zhang, M. (2021). Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction. Information and Software Technology, 139. https://doi.org/10.1016/j.infsof.2021.106662

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In T. Cohn, Y. He, & Y. Liu (Éds.), Findings of the Association for Computational Linguistics : EMNLP 2020 (p. 1536-1547). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.139

Fernández, A., García, S., Herrera, F., & Chawla, N. V. (2018). SMOTE for learning from imbalanced data : Progress and challenges, marking the 15-year anniversary. J. Artif. Int. Res., 61(1), 863-905.

Fernandez, N., Ghosh, A., Liu, N., Wang, Z., Choffin, B., Baraniuk, R., & Lan, A. (2022). Automated Scoring for Reading Comprehension via In-context BERT Tuning. In M. M. Rodrigo, N. Matsuda, A. I. Cristea, & V. Dimitrova (Éds.), Artificial Intelligence in Education (p. 691-697). Springer International Publishing.

Fiorella, L., & Mayer, R. (2016). Learning as a Generative Activity : Eight Learning Strategies that Promote Understanding. https://doi.org/10.1017/CBO9781107707085

Fournier, J.-P., & Wirz, J. (2009). Allogène : Un environnement d'apprentissage de l'algorithmique. 14.

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1), 119-139. https://doi.org/10.1006/jcss.1997.1504

Friedman, J. H. (2002). Stochastic gradient boosting. Nonlinear Methods and Data Mining, 38(4), 367-378. https://doi.org/10.1016/S0167-9473(01)00065-2

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian Network Classifiers. Machine Learning, 29(2), 131-163. https://doi.org/10.1023/A:1007465528199

Futschek, G. (2006). Algorithmic Thinking : The Key for Understanding Computer Science. In R. T. Mittermeir (Éd.), Informatics Education – The Bridge between Using and Understanding Computers (Vol. 4226, p. 159-168). Springer Berlin Heidelberg. https://doi.org/10.1007/11915355_15

Futschek, G., & Moschitz, J. (2010). Developing Algorithmic Thinking by Inventing and Playing Algorithms. 10.

Futschek, Gerald, et Julia Moschitz. 2010. « Developing Algorithmic Thinking by Inventing and Playing Algorithms ». 10.

Futschek, Gerald. 2006. « Algorithmic Thinking: The Key for Understanding Computer Science ». P. 159-68 in Informatics Education – The Bridge between Using and Understanding Computers. Vol. 4226, édité par R. T. Mittermeir. Berlin, Heidelberg: Springer Berlin Heidelberg.

G. A. Pradipta, R. Wardoyo, A. Musdholifah, I. N. H. Sanjaya, & M. Ismail. (2021). SMOTE for Handling Imbalanced Data Problem : A Review. 2021 Sixth International Conference on Informatics and Computing (ICIC), 1-8. https://doi.org/10.1109/ICIC54025.2021.9632912

Gamra, S., & Khlifa, Z. B. (2019). EIAH l'Apprentissage Autrement. 15.

Gardner, J., O'Leary, M., & Yuan, L. (2021). Artificial intelligence in educational assessment : 'Breakthrough ? Or buncombe and ballyhoo?'. Journal of Computer Assisted Learning, 37(5), 1207-1216. https://doi.org/10.1111/jcal.12577

Garrison, C., & Ehringhaus, M. (2007). Formative and Summative Assessment in the Classroom. Association for Middle Level Education.

Gautam Sharma. (2021). Multi-Label Classification. Multi-label classification sounds Analytics Vidhya | Medium. https://medium.com/analytics-vidhya/multi-label-classification-a9643d221954

Gelman, A., & Hill, J. (2006). Data Analysis Using Regression And Multilevel/Hierarchical Models. In Cambridge Universty Press (Vol. 3). https://doi.org/10.1017/CBO9780511790942

Gernsbacher, M. (1999). Comprehension : A Paradigm for Cognition. American Scientist, 87.

Ghorbani, R., & Ghousi, R. (2020). Comparing Different Resampling Methods in Predicting Students' Performance Using Machine Learning Techniques. IEEE Access, 8, 67899-67911. https://doi.org/10.1109/ACCESS.2020.2986809

Gilliot, J.-M., Mawas, N., & Garlatti, S. (2019). Intégrer le temps long dans les EIAH pour développer le pouvoir d'agir des apprenants. 10.

Gipps, C., & Farajnezhad, Z. (2022). Beyond Testing ~ Towards a Theory of Educational Assessment by Caroline V. Gipps. https://doi.org/10.13140/RG.2.2.10759.47526

González-Calatayud, V., Prendes-Espinosa, P., & Roig-Vila, R. (2021). Artificial Intelligence for Student Assessment : A Systematic Review. Applied Sciences, 11(12). https://doi.org/10.3390/app11125467

Grandbastien, M., & Nowakowski, S. (2014). Connaissances embarquées pour personnaliser les environnements d'apprentissage : Application à la plate-forme OP4L. 21. http://sticef.univ-lemans.fr/num/vol2014/15-grandbastien-epa/sticef_2014_NS_grandbastien_15.htm

Gross, P., & Powers, K. (2005). Evaluating assessments of novice programming environments. Proceedings of the 2005 International Workshop on Computing Education Research - ICER '05, 99-110. https://doi.org/10.1145/1089786.1089796

Gross, Paul, et Kris Powers. 2005. « Evaluating Assessments of Novice Programming Environments ». P. 99-110 in Proceedings of the 2005 international workshop on Computing education research - ICER '05. Seattle, WA, USA: ACM Press.

Guibert, Nicolas, et Patrick Girard. 2003. « Programmation sur Exemple et Enseignement assisté par ordinateur de l'algorithmique : le projet MELBA ». 4.

Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., & Yin, J. (2022). UniXcoder : Unified Cross-Modal Pre-training for Code Representation. In S. Muresan, P. Nakov, & A. Villavicencio (Éds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers) (p. 7212-7225). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.499

Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Svyatkovskiy, A., Fu, S., Tufano, M., Deng, S. K., Clement, C., Drain, D., Sundaresan, N., Yin, J., Jiang, D., & Zhou, M. (2021). GRAPHCODEBERT: PRE-TRAINING CODE REPRESEN- TATIONS WITH DATA FLOW. https://arxiv.org/abs/2009.08366

Guo, P. J. (2013). Online python tutor : Embeddable web-based program visualization for cs education. Proceeding of the 44th ACM Technical Symposium on Computer Science Education, 579-584. https://doi.org/10.1145/2445196.2445368

Guzdial, M. (2004). Programming Environments for Novices.

Guzdial, Mark. 2004. « Programming Environments for Novices ».

Guzdial, Mark. 2015. « Learner-Centered Design of Computing Education: Research on Computing for Everyone ». Synthesis Lectures on Human-Centered Informatics 8:1-165. doi: 10.2200/S00684ED1V01Y201511HCI033.

H. He & E. A. Garcia. (2009). Learning from Imbalanced Data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284. https://doi.org/10.1109/TKDE.2008.239

H. Ueno, & A. Nakajima. (1993). INTELLITUTOR: integrated intelligent programming environment for novices. Transactions on Information and Communications Technologies vol 3, © 1993 WIT Press, www.witpress.com, ISSN 1743-3517.

Håklev, S., Faucon, L., Hadzilacos, T., & Dillenbourg, P. (2017). Orchestration Graphs : Enabling Rich Social Pedagogical Scenarios in MOOCs. https://doi.org/10.1145/3051457.3054000

Haladyna, T. M., Downing, S. M., & Rodriguez, M. C. (2002). A review of multiple-choice item-writing guidelines for classroom assessment. Applied Measurement in Education, 15(3), 309-334. https://doi.org/10.1207/S15324818AME1503_5

Halverson, Lisa R., et Charles R. Graham. 2019. « Learner Engagement in Blended Learning Environments: A Conceptual Framework ». Online Learning 23(2). doi: 10.24059/olj.v23i2.1481.

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In Adv Intell Comput (Vol. 3644, p. 887). https://doi.org/10.1007/11538059_91

Han, J., Kamber, M., & Pei, J. (2012). 9—Classification : Advanced Methods. In J. Han, M. Kamber, & J. Pei (Éds.), Data Mining (Third Edition) (p. 393-442). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-381479-1.00009-5

Hannebauer, Christoph, Marc Hesenius, et Volker Gruhn. 2018. « Does Syntax Highlighting Help Programming Novices? » Empirical Software Engineering 23(5):2795-2828. doi: 10.1007/s10664-017-9579-0.

Haoran, Xie, Chu Hui-Chun, Hwang Gwo-Jen, et Wang Chun-Chieh. 2019. « Trends and evelopment in technology-enhanced adaptive/personalized learning: A systématic review of journl publications from 2007 to 2017. » Computers & Education 140:16.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Ensemble Learning. In T. Hastie, R. Tibshirani, & J. Friedman (Éds.), The Elements of Statistical Learning : Data Mining, Inference, and Prediction (p. 605-624). Springer New York. https://doi.org/10.1007/978-0-387-84858-7_16

Hattie, J. (2009). Visible Learning : A Synthesis of Over 800 Meta-Analyses Relating to Achievement. In Visible Learning : A Synthesis of Over 800 Meta-Analyses Relating to Achievement. https://doi.org/10.4324/9780203887332

Hattie, J., & Timperley, H. (2007). The Power of Feedback. Review of Educational Research, 77(1), 81-112. https://doi.org/10.3102/003465430298487

Hattie, John, et Helen Timperley. 2007. « The Power of Feedback ». Review of Educational Research 77(1):81-112. doi: 10.3102/003465430298487.

He, H., Bai, Y., Garcia, E., & Li, S. (2008). ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In Proceedings of the International Joint Conference on Neural Networks (p. 1328). https://doi.org/10.1109/IJCNN.2008.4633969

Hennessy, John L., et David A. Patterson. 2011. Computer Architecture, Fifth Edition: A Quantitative Approach. 5th éd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Henrich, J., Heine, S. J., & Norenzayan, A. (2010). The weirdest people in the world? Behavioral and Brain Sciences, 33(2-3), 61-83. Cambridge Core. https://doi.org/10.1017/S0140525X0999152X

Hmelo-Silver, C. (2004). Problem-Based Learning : What and How Do Students Learn? Educational Psychology Review, 16, 235-266. https://doi.org/10.1023/B:EDPR.0000034022.16470.f3

Homer, M., & Noble, J. (2013). A tile-based editor for a textual programming language. 2013 First IEEE Working Conference on Software Visualization (VISSOFT), 1-4. https://doi.org/10.1109/VISSOFT.2013.6650546

Honeine, P., Noumir, Z., & Richard, C. (2013). Multiclass classification machines with the complexity of a single binary classifier. Signal Processing, 93(5), 1013-1026. https://doi.org/10.1016/j.sigpro.2012.11.009

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S. -J. (2015). A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. Journal of Computer Assisted Learning, 31(4), 345-361. https://doi.org/10.1111/jcal.12099

Hornik, K., Stinchcombe, M., & White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Networks, 3(5), 551-560. https://doi.org/10.1016/0893-6080(90)90005-6

Hosmer, et Lemeshow. 2000. « Model-Building Strategies and Methods for Logistic Regression ». P. 91-142 in Applied Logistic Regression.

Hu, Chenglie. 2011. « Computational Thinking: What It Might Mean and What We Might Do about It ». P. 223 in Proceedings of the 16th annual joint conference on Innovation and technology in computer science education - ITiCSE '11. Darmstadt, Germany: ACM Press.

Hwang, G.-J., Xie, H., Wah, B. W., & Gašević, D. (2020). Vision, challenges, roles and research issues of Artificial Intelligence in Education. Computers and Education: Artificial Intelligence, 1, 100001. https://doi.org/10.1016/j.caeai.2020.100001

i, S., & Herrera, F. (2008). An Extension on « Statistical Comparisons of Classifiers over Multiple Data Sets » for all Pairwise Comparisons. Journal of Machine Learning Research - JMLR, 9.

Intayoad, W., Kamyod, C., & Temdee, P. (2019). Synthetic Minority Over-Sampling for Improving Imbalanced Data in Educational Web Usage Mining. ECTI Transactions on Computer and Information Technology, 12(2), 118-129. https://doi.org/10.37936/ecti-cit.2018122.133280

Jacobs, K. L. 2005. « Investigation of Interactive Online Visual Tools for the Learning of Mathematics ». International Journal of Mathematical Education in Science and Technology 36(7):761-68. doi: 10.1080/00207390500271149.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). Classification. In G. James, D. Witten, T. Hastie, & R. Tibshirani (Éds.), An Introduction to Statistical Learning : With Applications in R (p. 129-195). Springer US. https://doi.org/10.1007/978-1-0716-1418-1_4

Jenkins, Tony, et John Davy. 2002. « Diversity and Motivation in Introductory Programming ». Innovation in Teaching and Learning in Information and Computer Sciences 1(1):1-9. doi: 10.11120/ital.2002.01010003.

Jenkins, Tony. 2001. The motivation of students of programming. Vol. 33.

Jerome H. Friedman. (2001). Greedy function approximation : A gradient boosting machine. The Annals of Statistics, 29(5), 1189-1232. https://doi.org/10.1214/aos/1013203451

Jin, J., & Kim, M. (2023). GPT-Empowered Personalized eLearning System for Programming Languages. Applied Sciences. https://api.semanticscholar.org/CorpusID:265591341

Johanes, M. P., & Lagerstrom, L. (2017). Adaptive Learning : The Premise, Promise, and Pitfalls.

Johnson, Brittany, Yoonki Song, Emerson Murphy-Hill, et Robert Bowdidge. 2013. « Why don't software developers use static analysis tools to find bugs? » P. 672-81 in 2013 35th International Conference on Software Engineering (ICSE).

Johnson, W. L., & Soloway, E. (1985). PROUST : Knowledge-Based Program Understanding. IEEE Transactions on Software Engineering, SE-11(3), 267-275. https://doi.org/10.1109/TSE.1985.232210

Jolivet, Sébastien. 2018. « Descripteurs didactiques d'exercices de mathématiques: catégorisation; utilité et utilisabilité d'un modèle ». 10.

Jonassen, D. (2010). Learning to solve problems : A handbook for designing problem-solving learning environments. Learning to Solve Problems: A Handbook for Designing Problem-Solving Learning Environments, 1-437. https://doi.org/10.4324/9780203847527

Jonassen, David. 2000. « Toward a Design Theory of Problem Solving ». Educational Technology Research and Development 48:63-85. doi: 10.1007/BF02300500.

Jonsson, A., & Svingby, G. (2007). The use of scoring rubrics : Reliability, validity and educational consequences. Educational Research Review, 2(2), 130-144. https://doi.org/10.1016/j.edurev.2007.05.002

Junco, R. (2012). Too much face and not enough books : The relationship between multiple indices of Facebook use and academic performance. Computers in Human Behavior, 28(1), 187-198. https://doi.org/10.1016/j.chb.2011.08.026

Junxiang, G., & Yu, H. (2019). Design of Mobile Learning System for Courses of Computer Science and Technology. Journal of Physics: Conference Series, 1237, 052002. https://doi.org/10.1088/1742-6596/1237/5/052002

Kalyuga, S. (2007). Expertise Reversal Effect and Its Implications for Learner-Tailored Instruction. Ed Psychol Rev, 19, 509-539. https://doi.org/10.1007/s10648-007-9054-3

Kamsa, I., Elouahbi, R., & El khoukhi, F. (2018). The Combination between the Individual Factors and the Collective Experience for Ultimate Optimization Learning Path using Ant Colony Algorithm. International Journal on Advanced Science, Engineering and Information Technology, 8(4), 1198. https://doi.org/10.18517/ijaseit.8.4.2787

Kanaki, Kalliopi, Michail Kalogiannakis, Emmanouil Poulakis, et Panagiotis Politis. 2022. « Investigating the Association between Algorithmic Thinking and Performance in Environmental Study ». Sustainability 14(17):10672. doi: 10.3390/su141710672.

Karpicke, J., & Roediger, H. (2008). The Critical Importance of Retrieval for Learning. Science (New York, N.Y.), 319, 966-968. https://doi.org/10.1126/science.1152408

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). ChatGPT for good ? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274

Kay, R., & Knaack, L. (2009). Assessing Learning, Quality and Engagement in Learning Objects : The Learning Object Evaluation Scale for Students (LOES-S). Educational Technology Research and Development, 57, 147-168. https://doi.org/10.1007/s11423-008-9094-5

Kazandzhy, L. (2017). Lightbot : Code Hour. 2.

Keinänen, Meiju, Jani Ursin, et Kari Nissinen. 2018. « How to Measure Students' Innovation Competences in Higher Education: Evaluation of an Assessment Tool in Authentic Learning Environments ». Studies in Educational Evaluation 58:30-36. doi: 10.1016/j.stueduc.2018.05.007.

Kelleher, Caitlin, et Randy Pausch. 2005. « Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers ». ACM Comput. Surv. 37(2):83-137. doi: 10.1145/1089733.1089734.

Khalaf Hamoud, A., Baqr Mohammed Kamel, M., Sahl Gaafar, A., Salah Alasady, A., Majeed Humadi, A., Akeel Awadh, W., & Mohammed Dahr, J. (2022). A prediction model based machine learning algorithms with feature selection approaches over imbalanced dataset. Indonesian Journal of Electrical Engineering and Computer Science, 28(2), 1105-1116. https://doi.org/10.11591/ijeecs.v28.i2.pp1105-1116

Khribi, M. K., Jemni, M., & Nasraoui, O. (2008). Automatic Recommendations for E-Learning Personalization Based on Web Usage Mining Techniques and Information Retrieval. 2008 Eighth IEEE International Conference on Advanced Learning Technologies, 241-245. https://doi.org/10.1109/ICALT.2008.198

Kinnunen, Päivi, et Lauri Malmi. 2006. « Why students drop out CS1 course? » P. 97-108 in Proceedings of the Second International Workshop on Computing Education Research, ICER '06. New York, NY, USA: Association for Computing Machinery.

Kintsch, W. (1998). Comprehension : A paradigm for cognition. (p. xvi, 461). Cambridge University Press.

Kirschner, P. A. (2017a). Stop propagating the learning styles myth. Computers & Education, 106, 166-171. https://doi.org/10.1016/j.compedu.2016.12.006

Kirschner, P., & Van Merrienboer, J. J. G. (2013). Do Learners Really Know Best ? Urban Legends in Education. Educational Psychologist, 48, 169-183. https://doi.org/10.1080/00461520.2013.804395

Klašnja-Milićević, A., Vesin, B., Ivanović, M., & Budimac, Z. (2011). E-Learning personalization based on hybrid recommendation strategy and learning style identification. Computers & Education, 56(3), 885-899. https://doi.org/10.1016/j.compedu.2010.11.001

Kocev, D., Vens, C., Struyf, J., & Džeroski, S. (2007). Ensembles of Multi-Objective Decision Trees. In Proceedings of the 18th European Conference on Machine Learning (Vol. 4701, p. 631). https://doi.org/10.1007/978-3-540-74958-5_61

Koedinger, K. R., Corbett, A. T., & Perfetti, C. (2012). The Knowledge-Learning-Instruction Framework : Bridging the Science-Practice Chasm to Enhance Robust Student Learning. Cognitive Science, 36(5), 757-798. https://doi.org/10.1111/j.1551-6709.2012.01245.x

Koehrsen, W. (2020, août 18). Random Forest Simple Explanation. Medium. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

Kolb, D. (1984). Experiential Learning : Experience As The Source Of Learning And Development. In Journal of Business Ethics (Vol. 1).

Koper, R., & Tattersall, C. (2005). Preface to Learning Design : A Handbook on Modelling and Delivering Networked Education and Training. Journal of Interactive Media in Education, 2005. https://doi.org/10.5334/2005-18

Korchi, A., & Oughdir, L. (2015). Modélisation de l'apprenant dans un EIAH à base d'ontologie (cas de l'apprenant et son profil). 5.

Kostadinov, D. (2003). Personnalisation de l'information et gestion des profils utilisateurs. Mémoire de DEA PRiSM, Versailles.

Kourgiantakis, T., Sewell, K. M., & Bogo, M. (2019). The Importance of Feedback in Preparing Social Work Students for Field Education. Clinical Social Work Journal, 47(1), 124-133. https://doi.org/10.1007/s10615-018-0671-8

Kramer, S. L., Posner, M. A., Browman, A. S., Lawrence, N. R., Roem, J., & Krier, K. (2023). The Impacts of a Standards-Based Grading System Emphasizing Formative Assessment, Feedback, and Re-Assessment : A Mixed Methods, Cluster Randomized Control Trial in Ninth Grade Mathematics Classrooms. Journal of Research on Educational Effectiveness, 1-32. https://doi.org/10.1080/19345747.2023.2287594

Krawczyk, B. (2016). Learning from imbalanced data : Open challenges and future directions. Progress in Artificial Intelligence, 5(4), 221-232. https://doi.org/10.1007/s13748-016-0094-0

Kuen, K. C. (2013). Learning Programming Concepts Using Flowcharting Software. 4.

Kumar, Mukesh, Karan Bajaj, Bhisham Sharma, et Sushil Narang. 2022. « A Comparative Performance Assessment of Optimized Multilevel Ensemble Learning Model with Existing Classifier Models ». Big Data 10(5):371-87. doi: 10.1089/big.2021.0257.

Kurilovas, E., Zilinskiene, I., & Dagiene, V. (2015). Recommending suitable learning paths according to learners' preferences : Experimental research results. Computers in Human Behavior, 51, 945-951. https://doi.org/10.1016/j.chb.2014.10.027

Kurni, M., Mohammed, M. S., & Srinivasa, K. G. (2023). Natural Language Processing for Education. In M. Kurni, M. S. Mohammed, & S. K G (Éds.), A Beginner's Guide to Introduce Artificial Intelligence in Teaching and Learning (p. 45-54). Springer International Publishing. https://doi.org/10.1007/978-3-031-32653-0_3

Labat, J.-M. (2002). EIAH: Quel retour d'informations pour le tuteur? 81-88. https://doi.org/edutice-00000644

Lahtinen, Essi, Ala-Mutka Kirsti, et Järvinen Hannu-Matti. 2005. « A Study of the Difficulties of Novice Programmers ». In Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '05). Association for Computing Machinery, New York, NY, USA 14-18. doi: https://doi.org/10.1145/1067445.1067453.

Lan, Y., Li, X., Du, H., Lu, X., Gao, M., Qian, W., & Zhou, A. (2024). Survey of Natural Language Processing for Education : Taxonomy, Systematic Review, and Future Trends. https://arxiv.org/abs/2401.07518

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521, 436-444. https://doi.org/10.1038/nature14539

Lee, M. J., & Ko, A. J. (2011). Personifying Programming Tool Feedback Improves Novice Programmers' Learning. In Proceedings of the Seventh International Workshop on Computing Education Research (ICER '11). Association for Computing Machinery, New York, NY, USA, 109-116. https://doi.org/10.1145/2016911.2016934

Leiva, L. A., Arapakis, I., & Iordanou, C. (2021). My Mouse, My Rules : Privacy Issues of Behavioral User Profiling via Mouse Tracking. Proceedings of the 2021 Conference on Human Information Interaction and Retrieval. https://api.semanticscholar.org/CorpusID:231693112

Lemaitre, G., Nogueira, F., & Aridas, C. K. (2016). Imbalanced-learn : A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. https://arxiv.org/abs/1609.06570

Lewis, Colleen M. 2010. « How programming environment shapes perception, learning and goals: logo vs. scratch ». P. 346-50 in Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE '10. New York, NY, USA: Association for Computing Machinery.

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., & Li, L. (2020). On the Sentence Embeddings from Pre-trained Language Models. In B. Webber, T. Cohn, Y. He, & Y. Liu (Éds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (p. 9119-9130). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.733

Limongelli, C., Sciarrone, F., & Vaste, G. (2008). LS-p lan : An effective combination of dynamic courseware generation and learning styles in web-based education. Adaptive Hypermedia and Adaptive Web-Based Systems, 133-142.

Lin, C. F., Yeh, Y., Hung, Y. H., & Chang, R. I. (2013). Data mining for providing a personalized learning path in creativity : An application of decision trees. Computers & Education, 68, 199-210. https://doi.org/10.1016/j.compedu.2013.05.009

Ling Chean, Swee, Sin Ban Ho, et Ian Chai. 2018. « A Conceptual Framework on Constructing Effective Learning Content for Programming Novices ». International Journal of Engineering & Technology 7(2.15):150. doi: 10.14419/ijet.v7i2.15.11374.

Linn, Marcia C. 2000. « Designing the Knowledge Integration Environment ». International Journal of Science Education 22(8):781-96. doi: 10.1080/095006900412275.

Linn, R. L. (1993). Educational Assessment : Expanded Expectations and Challenges. Educational Evaluation and Policy Analysis, 15(1), 1-16. https://doi.org/10.3102/01623737015001001

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in CS1. Proceedings of the 47th ACM Technical Symposium on Computing Science Education, 329-334. https://doi.org/10.1145/2839509.2844596

Lister, Raymond, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, Beth Simon, et Lynda Thomas. 2004. « A multi-national study of reading and tracing skills in novice programmers ». P. 119-50 in Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education, ITiCSE-WGR '04. New York, NY, USA: Association for Computing Machinery.

Lister, Raymond. 2011. « Concrete and other neo-Piagetian forms of reasoning in the novice programmer ». P. 9-18 in Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114, ACE '11. AUS: Australian Computer Society, Inc.

Litalien, D., Gillet, N., Gagné, M., Ratelle, C. F., & Morin, A. J. S. (2019). Self-determined motivation profiles among undergraduate students : A robust test of profile similarity as a function of gender and age. Learning and Individual Differences, 70, 39-52. https://doi.org/10.1016/j.lindif.2019.01.005

Litman, D. (2016). Natural Language Processing for Enhancing Teaching and Learning. Proceedings of the AAAI Conference on Artificial Intelligence, 30. https://doi.org/10.1609/aaai.v30i1.9879

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa : A Robustly Optimized BERT Pretraining Approach. ArXiv, abs/1907.11692. https://api.semanticscholar.org/CorpusID:198953378

M. A. Tayal, R. Joshi, M. Darvekar, M. Malghade, & C. Sonboir. (2023). Automated Exam Paper Checking Using Semantic Analysis. 2023 OITS International Conference on Information Technology (OCIT), 957-962. https://doi.org/10.1109/OCIT59427.2023.10431267

M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, & F. Herrera. (2012). A Review on Ensembles for the Class Imbalance Problem : Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(4), 463-484. https://doi.org/10.1109/TSMCC.2011.2161285

M. Li, S. Wang, & L. Guo. (2018). The effectiveness of ensemble learning based on four different classifiers for predicting membrane protein types. 2018 Chinese Control And Decision Conference (CCDC), 5829-5834. https://doi.org/10.1109/CCDC.2018.8408150

Madeyski, Lech. 2010. « The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment ». Information and Software Technology 52(2):169-84. doi: 10.1016/j.infsof.2009.08.007.

MAIGA, A. A., & HOTTE, R. (2021). Directeur de publication adjoint. Monographie de l'enfant de Gao : l'école en Afrique subsaharienne.

Majzoub, H., Elgedawy, I., Akaydın, Ö., & Ulukök, M. (2020). HCAB-SMOTE: A Hybrid Clustered Affinitive Borderline SMOTE Approach for Imbalanced Data Binary Classification. Arabian Journal for Science and Engineering, 45. https://doi.org/10.1007/s13369-019-04336-1

Malliarakis, Christos, Maya Satratzemi, et Stelios Xinogalos. 2013. « Towards a New Massive Multiplayer Online Role Playing Game for Introductory Programming ». P. 156 in Proceedings of the 6th Balkan Conference in Informatics on - BCI '13. Thessaloniki, Greece: ACM Press.

Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by Choice : Urban Youth Learning Programming with Scratch. 5.

Matthíasdóttir, A. (2004). Learning objects in a multimedia interactive environment : The codewitz project. Proceedings of the 5th International Conference on Computer Systems and Technologies - CompSysTech '04, 1. https://doi.org/10.1145/1050330.1050415

Mayer, Richard E. 1981. « The Psychology of How Novices Learn Computer Programming ». ACM Omput. Surv. 13(1):121-41. doi: 10.1145/356835.356841.

Mayers, A., & Lefebvre, B. (1992). Une modélisation de l'architecture cognitive d'un étudiant pour un système tutoriel intelligent. In C. Frasson, G. Gauthier, & G. I. McCalla (Éds.), Intelligent Tutoring Systems (Vol. 608, p. 277-285). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-55606-0_35

McCall, Davin, et Michael Kölling. 2019. « A New Look at Novice Programmer Errors ». ACM Transactions on Computing Education 19(4):1-30. doi: 10.1145/3335814.

Mcintosh, Shane, Yasutaka Kamei, Bram Adams, et Ahmed E. Hassan. 2016. « An empirical study of the impact of modern code review practices on software quality ». Empirical Softw. Engg. 21(5):2146-89. doi: 10.1007/s10664-015-9381-9.

Mease, D., Wyner, A. J., & Buja, A. (2007). Boosted Classification Trees and Class Probability/Quantile Estimation. Journal of Machine Learning Research, 8(16), 409-439.

Mienye, D., & Jere, N. (2024). A Survey of Decision Trees : Concepts, Algorithms, and Applications. IEEE Access, PP, 1-1. https://doi.org/10.1109/ACCESS.2024.3416838

Mirzajani, H., Delaviz, M., Kookandeh, M., Rezaee, S., Kamalifar, A., & Shani, H. (2016). Smart Schools an Innovation in Education : Malaysian's Experience. Asian Journal of Education and Training, 2, 11-15. https://doi.org/10.20448/journal.522/2016.2.1/522.1.11.15

ML | Bagging classifier. (2019, mai 16). GeeksforGeeks. https://www.geeksforgeeks.org/ml-bagging-classifier/

Montañes, E., Senge, R., Barranquero, J., Ramón Quevedo, J., José del Coz, J., & Hüllermeier, E. (2014). Dependent binary relevance models for multi-label classification. Handwriting Recognition and other PR Applications, 47(3), 1494-1508. https://doi.org/10.1016/j.patcog.2013.09.029

Morris, T. H. (2019). Experiential learning – a systematic review and revision of Kolb's model. Interactive Learning Environments, 1-14. https://doi.org/10.1080/10494820.2019.1570279

Muddaluru, R., Thoguluva, S., Prabha, S., Pati, P. B., & Balakrishnan, R. (2023). Auto-grading C programming assignments with CodeBERT and Random Forest Regressor (p. 6). https://doi.org/10.1109/ICCCNT56998.2023.10308341

Nabil, A., Seyam, M., & Abou-Elfetouh, A. (2021). Prediction of Students' Academic Performance Based on Courses' Grades Using Deep Neural Networks. IEEE Access, 9, 140731-140746. https://doi.org/10.1109/ACCESS.2021.3119596

Nabus, H., Ali, A., Hasan, S., Shamsuddin, S. M., Mustapha, I., & Saeed, F. (2022). Adaptive Generation-based Approaches of Oversampling using Different Sets of Base and Nearest Neighbor's Instances.

International Journal of Advanced Computer Science and Applications, 13, 527-534. https://doi.org/10.14569/IJACSA.2022.0130461

Nancekivell, S. E., Shah, P., & Gelman, S. A. (2020). Maybe they're born with it, or maybe it's experience : Toward a deeper understanding of the learning style myth. Journal of Educational Psychology, 112(2), 221-235. https://doi.org/10.1037/edu0000366

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in Neurorobotics, 7. https://doi.org/10.3389/fnbot.2013.00021

Nicol, D., & Macfarlane, D. (2006). Formative Assessment and Self-Regulated Learning : A Model and Seven Principles of Good Feedback Practice. Studies in Higher Education, 31, 199-218. https://doi.org/10.1080/03075070600572090

Nkambou, R., Bourdeau, J., & Mizoguchi, R. (Éds.). (2010). Advances in Intelligent Tutoring Systems (Vol. 308). Springer. https://doi.org/10.1007/978-3-642-14363-2

Noble, S. (2018). Algorithms of Oppression : How Search Engines Reinforce Racism. https://doi.org/10.2307/j.ctt1pwt9w5

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How Many Trees in a Random Forest? In P. Perner (Éd.), Machine Learning and Data Mining in Pattern Recognition (p. 154-168). Springer Berlin Heidelberg.

Ostrowska-Wawryniuk, Karolina, Marcin Strzała, et Jan Słyk. 2022. « Form Follows Parameter: Algorithmic-Thinking-Oriented Course for Early-stage Architectural Education ». Nexus Network Journal 24:503-22.

Panadero, E., Jönsson, A., & Botella, J. (2017). Effects of self-assessment on self-regulated learning and self-efficacy : Four meta-analyses. Educational Research Review, 22, 74-98. https://doi.org/10.1016/j.edurev.2017.08.004

PANE, JOHN F., CHOTIRAT "ANN" RATANAMAHATANA, et BRAD A. MYERS. 2001. « Studying the language and structure in non-programmers' solutions to programming problems ». International Journal of Human-Computer Studies 54(2):237-64. doi: https://doi.org/10.1006/ijhc.2000.0410.

Paneva, D., & Zhelev, Y. (2007). Models, techniques and applications of e-learning personalization.

Paquette, P. G. (2002). Modélisation des connaissances et des compétences—Pour concevoir et apprendre. Presses de l'Université du Québec.

Pardo, A., & Siemens, G. (2014). Ethical and Privacy Principles for Learning Analytics. British Journal of Educational Technology, 45. https://doi.org/10.1111/bjet.12152

Pariser, E. (2012). The Filter Bubble : How the New Personalized Web Is Changing What We Read and How We Think. Penguin Books.

Parmentier, Yannick. 2018. « Enseigner la pensée informatique à l'école primaire: formation initiale et continue des professeurs ». 11.

Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R. (2008). Learning Styles : Concepts and Evidence. Psychological Science in the Public Interest, 9(3), 105-119. https://doi.org/10.1111/j.1539-6053.2009.01038.x

Patel, M., & Sajja, P. S. (2021). Application for Multi-Agent System : A Case of Customised eLearning. 2021 International Conference on Computing, Communication and Green Engineering (CCGE), 1-6.

Peinelt, N., Nguyen, D., & Liakata, M. (2020). tBERT: Topic Models and BERT Joining Forces for Semantic Similarity Detection. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Éds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (p. 7047-7055). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.630

Pennington, Nancy. 1987. « Stimulus structures and mental representations in expert comprehension of computer programs ». Cognitive Psychology 19(3):295-341. doi: 10.1016/0010-0285(87)90007-7.

Perkins, D. N. (1992). Smart schools : From training memories to educating minds. (p. ix, 262). Free Press.

Persico, D., Pozzi, F., & Sarti, L. (2010). Monitoring collaborative activities in computer supported collaborative learning. Distance Education, 31, 5-22. https://doi.org/10.1080/01587911003724603

Pezzino, M. (2018). Online assessment, adaptive feedback and the importance of visual learning for students. The advantages, with a few caveats, of using MapleTA. International Review of Economics Education, 28, 11-28. https://doi.org/10.1016/j.iree.2018.03.002

Pietrikova, Emilia, Jan Juhar, et Jana Stastna. 2015. « Towards Automated Assessment in Game-Creative Programming Courses ». P. 1-6 in 2015 13th International Conference on Emerging eLearning Technologies and Applications (ICETA). Starý Smokovec, High Tatras, Slovakia: IEEE.

Pillay, N. (2003). Developing intelligent programming tutors for novice programmers. ACM SIGCSE Bulletin, 35(2), 78-82. https://doi.org/10.1145/782941.782986

Pillay, Nelishia, et Jugoo Vikash R. 2005. « An Investigation into Student Characteristics Affecting Novice Programming Performance ». ACM SIGCSE Bulletin 37(4):107-10. doi: 10.1145/1113847.1113888.

Piteira, Martinha, et Carlos J. Costa. 2012. « Computer programming and novice programmers ». in International Conference on Information Systems and Design of Communication.

Poudel, P., & Shrestha, S. (2023). Exploring the user's preferences of different adaptation policies in adaptive menu design. Trends in Computer Science and Information Technology, 8, 005-011. https://doi.org/10.17352/tcsit.000062

Premlatha, K. R., Dharani, B., & Geetha, T. V. (2016). Dynamic learner profiling and automatic learner classification for adaptive e-learning environment. Interactive Learning Environments, 24(6), 1054-1075. https://doi.org/10.1080/10494820.2014.948459

Premlatha, K. R., Dharani, B., & Geetha, T. V. (2016a). Dynamic learner profiling and automatic learner classification for adaptive e-learning environment. Interactive Learning Environments, 24(6), 1054-1075. https://doi.org/10.1080/10494820.2014.948459

Premlatha, K. R., Dharani, B., & Geetha, T. V. (2016b). Dynamic learner profiling and automatic learner classification for adaptive e-learning environment. Interactive Learning Environments, 24(6), 1054-1075. https://doi.org/10.1080/10494820.2014.948459

Pressman, Roger, et Bruce Maxim. 2014. Software Engineering: A Practitioner's Approach, 8th Ed.

Price, T. W., & Barnes, T. (2015). Comparing Textual and Block Interfaces in a Novice Programming Environment. Proceedings of the Eleventh Annual International Conference on International Computing Education Research - ICER '15, 91-99. https://doi.org/10.1145/2787622.2787712

Prince, Michael. 2004. « Does Active Learning Work? A Review of the Research ». Journal of Engineering Education 93(3):223-31. doi: 10.1002/j.2168-9830.2004.tb00809.x.

Pristyanto, Yoga, Anggit Ferdita Nugraha, Irfan Pratama, Akhmad Dahlan, et Lucky Adhikrisna Wirasakti. 2021. « Dual Approach to Handling Imbalanced Class in Datasets Using Oversampling and Ensemble Learning Techniques ». P. 1-7 in 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM). Seoul, Korea (South): IEEE.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost : Unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Éds.), Advances in Neural Information Processing Systems (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf

Proulx, Viera K. 2000. « Programming patterns and design patterns in the introductory computer science course ». SIGCSE Bull. 32(1):80-84. doi: 10.1145/331795.331819.

Qian, Yizhou, et James Lehman. 2017. « Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review ». ACM Transactions on Computing Education 18:1-24. doi: 10.1145/3077618.

Qiao, P., Zhu, X., Guo, Y., Sun, Y., & Qin, C. (2021). The Development and Adoption of Online Learning in Pre- and Post-COVID-19 : Combination of Technological System Evolution Theory and Unified Theory of Acceptance and Use of Technology. Journal of Risk and Financial Management, 14, 162. https://doi.org/10.3390/jrfm14040162

Quan, Y. (2020). Development of computer aided classroom teaching system based on machine learning prediction and artificial intelligence KNN algorithm. Journal of Intelligent & Fuzzy Systems, 39(2), 1879-1890. https://doi.org/10.3233/JIFS-179959

Rachburee, N., & Punlumjeak, W. (2021). Oversampling technique in student performance classification from engineering course. International Journal of Electrical and Computer Engineering (IJECE), 11(4), 3567-3574. https://doi.org/10.11591/ijece.v11i4.pp3567-3574

Radwan, A. M., & Cataltepe, Z. (2017). Improving Performance Prediction on Education Data with Noise and Class Imbalance. Intelligent Automation & Soft Computing, 1-8. https://doi.org/10.1080/10798587.2017.1337673

Rahaman, Md. A., & Hoque, A. S. Md. L. (2022). An effective evaluation system to grade programming assignments automatically. International Journal of Learning Technology, 17(3), 267-290. https://doi.org/10.1504/IJLT.2022.127194

Raunigr, Petr, et Marek Vajgl. 2018. « Possibilities of an Automated Assessment of Java Programming Homework in Elearning ». P. 060012 in. Thessaloniki, Greece.

Rawson, K., & Dunlosky, J. (2011). Optimizing Schedules of Retrieval Practice for Durable and Efficient Learning : How Much Is Enough? Journal of experimental psychology. General, 140, 283-302. https://doi.org/10.1037/a0023956

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. Machine Learning, 85(3), 333-359. https://doi.org/10.1007/s10994-011-5256-5

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch : Programming for all. Commun. ACM, 52(11), 60-67. https://doi.org/10.1145/1592761.1592779

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). « Why Should I Trust You? » : Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135-1144. https://doi.org/10.1145/2939672.2939778

Rish, I. (2001). An Empirical Study of the Naïve Bayes Classifier. IJCAI 2001 Work Empir Methods Artif Intell, 3.

Riston, T., Suherman, S. N., Yonnatan, Y., Indrayatna, F., Pravitasari, A. A., Sari, E. N., & Herawan, T. (2023). Oversampling Methods for Handling Imbalance Data in Binary Classification. In O. Gervasi, B. Murgante, A. M. A. C. Rocha, C. Garau, F. Scorza, Y. Karaca, & C. M. Torre (Éds.), Computational Science and Its Applications – ICCSA 2023 Workshops (p. 3-23). Springer Nature Switzerland.

Robins, Anthony, Janet Rountree, et Nathan Rountree. 2003. « Learning and Teaching Programming: A Review and Discussion ». Computer Science Education 13(2):137-72. doi: 10.1076/csed.13.2.137.14200.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A Primer in BERTology : What We Know About How BERT Works. Transactions of the Association for Computational Linguistics, 8, 842-866. https://doi.org/10.1162/tacl_a_00349

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408. https://doi.org/10.1037/h0042519

Rozi, A. F., Wibowo, A., & Warsito, B. (2023). Resampling Technique for Imbalanced Class Handling on Educational Dataset. JUITA : Jurnal Informatika, 11(1), 77. https://doi.org/10.30595/juita.v11i1.15498

Rui Xu & D. Wunsch. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3), 645-678. https://doi.org/10.1109/TNN.2005.845141

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536. https://doi.org/10.1038/323533a0

Ryan, Richard M., et Edward L. Deci. 2000. « Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. » American Psychologist 55(1):68-78. doi: 10.1037/0003-066X.55.1.68.

S. F. Chaerul Haviana, S. Mulyono, & Badie'Ah. (2023). The Effects of Stopwords, Stemming, and Lemmatization on Pre-trained Language Models for Text Classification : A Technical Study. 2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), 521-527. https://doi.org/10.1109/EECSI59885.2023.10295797

Sadik, A. (2008). Digital storytelling : A meaningful technology-integrated approach for engaged student learning. Educational Technology Research and Development, 56, 487-506. https://doi.org/10.1007/s11423-008-9091-8

Samaniego Erazo, G. N., Esteve-González, V., & Vaca, B. (2015). Teaching and Learning in digital worlds : Strategies and issues in higher education (p. 129-136).

Santos, O. C., & Boticario, J. G. (2010). Modeling recommendations for the educational domain. Procedia Computer Science, 1(2), 2793-2800. https://doi.org/10.1016/j.procs.2010.08.004

Sapozhnikova, E. P. (2009). ART-Based Neural Networks for Multi-label Classification. In N. M. Adams, C. Robardet, A. Siebes, & J.-F. Boulicaut (Éds.), Advances in Intelligent Data Analysis VIII (p. 167-177). Springer Berlin Heidelberg.

Sargent, R. (2011). Verification and validation of simulation models. In Engineering Management Review, IEEE (Vol. 37, p. 183). https://doi.org/10.1109/WSC.2010.5679166

Saritas, Mücahid Mustafa, et Ali Yasar. 2019. « Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification ». International Journal of Intelligent Systems and Applications in Engineering (7(2)):88-91. doi: https://doi.org/10.18201/ijisae.2019252786.

Sarı, Uğur, Hüseyin Miraç Pektaş, Ömer Faruk Şen, et Harun Çelik. 2022. « Algorithmic thinking development through physical computing activities with Arduino in STEM education ». Education and Information Technologies 27:6669-89.

Sawicki, J., Ganzha, M., & Paprzycki, M. (2023). The State of the Art of Natural Language Processing—A Systematic Automated Review of NLP Literature Using NLP Techniques. Data Intelligence, 5(3), 707-749. https://doi.org/10.1162/dint_a_00213

Saygin, A. P., Cicekli, I., & Akman, V. (2000). Turing Test : 50 Years Later. Minds and Machines, 10. https://doi.org/10.1023/A:1011288000451

Schapire, R. E., & Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning, 37(3), 297-336. https://doi.org/10.1023/A:1007614523901

Schunk, D., & Zimmerman, B. (2007). Influencing Children's Self-Efficacy and Self-Regulation of Reading and Writing Through Modeling. Reading & Writing Quarterly, 23, 7-25. https://doi.org/10.1080/10573560600837578

Schwartz, N. H. (2021). Kirschner, P. A., & Hendrick, C. (2020). How learning happens : Seminal works in educational psychology and what they mean in practice. Routledge. ISBN 9780367184575. TechTrends, 65(1), 120-121. https://doi.org/10.1007/s11528-020-00565-6

Selby, Cynthia C., C. Selby, John Woollard, et J Woollard. 2010. « Computational Thinking: The Developing Definition ». 6.

Selwyn, N. (2009). The digital native – myth and reality. Aslib Proceedings, 61, 364-379. https://doi.org/10.1108/00012530910973776

Shaffer, Clifford, Matthew Cooper, Alexander Alon, Monika Akbar, Michael Stewart, Sean Ponce, et Stephen Edwards. 2010. « Algorithm Visualization: The State of the Field ». TOCE 10. doi: 10.1145/1821996.1821997.

Sharma, R., Chen, F., Fard, F., & Lo, D. (2022). An exploratory study on code attention in BERT. Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, 437-448. https://doi.org/10.1145/3524610.3527921

Shavelson, R., & Webb, N. (1991). Generalizability Theory : A Primer. https://doi.org/10.1002/9781118445112.stat00068

Shemshack, A., & Spector, J. M. (2020). A systematic literature review of personalized learning terms. Smart Learning Environments, 7(1), 33. https://doi.org/10.1186/s40561-020-00140-9

Shute, V., & Zapata-Rivera, D. (2012). Adaptive Educational Systems. Adaptive Technologies for Training and Education, 7-27. https://doi.org/10.1017/CBO9781139049580.004

Siers, M., & Islam, M. Z. (2020). Class Imbalance and Cost-Sensitive Decision Trees : A Unified Survey Based on a Core Similarity. ACM Transactions on Knowledge Discovery from Data, 15, 1-31. https://doi.org/10.1145/3415156

Silva-Palacios, D., Ferri, C., & Ramírez-Quintana, M. J. (2017). Improving Performance of Multiclass Classification by Inducing Class Hierarchies. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland, 108, 1692-1701. https://doi.org/10.1016/j.procs.2017.05.218

Silverman, B. W., & Jones, M. C. (1989). E. Fix and J.L. Hodges (1951) : An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation : Commentary on Fix and Hodges (1951). International Statistical Review / Revue Internationale de Statistique, 57(3), 233-238. JSTOR. https://doi.org/10.2307/1403796

Sim, Tze Ying, et Sian Lun Lau. 2018. « Online Tools to Support Novice Programming: A Systematic Review ». P. 91-96 in 2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e). Langkawi Island, Malaysia: IEEE.

Smith, G., Shute, V., Rahimi, S., Dai, C.-P., & Kuba, R. (2023). Stealth Assessment and Digital Learning Game Design (p. 81-100). https://doi.org/10.4018/979-8-3693-0568-3.ch004

Soloway, Elliot, et Kate Ehrlich. 1984. « Empirical Studies of Programming Knowledge ». IEEE Transactions on Software Engineering SE-10(5):595-609. doi: 10.1109/TSE.1984.5010283.

Soloway, Elliot, Jeffrey Bonar, et Kate Ehrlich. 1983. « Cognitive strategies and looping constructs: an empirical study ». Commun. ACM 26(11):853-60. doi: 10.1145/182.358436.

Soloway, Elliot, Mark Guzdial, et Kenneth E. Hay. 1994. « Learner-centered design: the challenge for HCI in the 21st century ». Interactions 1:36-48.

Sorva, Juha. 2013. « Notional machines and introductory programming education ». ACM Trans. Comput. Educ. 13(2). doi: 10.1145/2483710.2483713.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1), 1929-1958.

Staff, K. (1998, octobre 2). Inside the black box : Raising standards through classroom assessment. Kappan Online. https://kappanonline.org/inside-the-black-box-raising-standards-through-classroom-assessment/

Stiggins, R., & Chappuis, J. (2005). Using Student-Involved Classroom Assessment to Close Achievement Gaps. Theory Into Practice - THEORY PRACT, 44, 11-18. https://doi.org/10.1207/s15430421tip4401_3

Storey, Margaret-Anne. 2006. « Theories, tools and research methods in program comprehension: past, present and future ». Software Quality Journal 14(3):187-208. doi: 10.1007/s11219-006-9216-4.

Stroustrup, Bjarne. 2013. The C++ Programming Language. Fourth edition. Upper Saddle River, NJ: Addison-Wesley.

Subagja, S., & Rubini, B. (2023). Analysis of Student Learning Styles Using Fleming's VARK Model in Science Subject. JURNAL PEMBELAJARAN DAN BIOLOGI NUKLEUS, 9, 31-39. https://doi.org/10.36987/jpbn.v9i1.3752

Sun, H., Dhingra, B., Zaheer, M., Mazaitis, K., Salakhutdinov, R., & Cohen, W. W. (2018). Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text. https://arxiv.org/abs/1809.00782

Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, 40(12), 3358-3378. https://doi.org/10.1016/j.patcog.2007.04.009

SUN, Y., WONG, A. K. C., & KAMEL, M. S. (2009). CLASSIFICATION OF IMBALANCED DATA: A REVIEW. International Journal of Pattern Recognition and Artificial Intelligence, 23(04), 687-719. https://doi.org/10.1142/S0218001409007326

Sun, Z., Ying, W., Zhang, W., & Gong, S. (2024). Undersampling method based on minority class density for imbalanced data. Expert Systems with Applications, 249, 123328. https://doi.org/10.1016/j.eswa.2024.123328

Sunar, A. S., Abbasi, R. A., Davis, H. C., White, S., & Aljohani, N. R. (2018). Modelling MOOC learners' social behaviours. Computers in Human Behavior, S0747563218305995. https://doi.org/10.1016/j.chb.2018.12.013

Sungkaew, Kornchulee, Piyamas Lungban, et Sirinya Lamhya. 2022. « Game development software engineering: digital educational game promoting algorithmic thinking ». International Journal of Electrical and Computer Engineering (IJECE).

Sweller, J. (1988). Cognitive load during problem solving : Effects on learning. Cognitive Science, 12(2), 257-285. https://doi.org/10.1016/0364-0213(88)90023-7

Sweller, J., Van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive Architecture and Instructional Design : 20 Years Later. Educational Psychology Review, 31(2), 261-292. https://doi.org/10.1007/s10648-019-09465-5

Tarekegn, A. N., Giacobini, M., & Michalak, K. (2021). A review of methods for imbalanced multi-label classification. Pattern Recognition, 118, 107965. https://doi.org/10.1016/j.patcog.2021.107965

Tariq, Muhammad Arham, Allah Bux Sargano, Muhammad Aksam Iftikhar, et Zulfiqar Habib. 2023. « Comparing Different Oversampling Methods in Predicting Multi-Class Educational Datasets Using Machine Learning Techniques ». Cybernetics and Information Technologies 23(4):199-212. doi: 10.2478/cait-2023-0044.

Tayoub, Saida, et Razane Chroqui. 2023. « Massive open online courses: a systematic literature review focusing on the achievements and challenges ». International Journal of Learning Technology 18(3):326-43. doi: 10.1504/IJLT.2023.134584.

Thongchotchat, V., Kudo, Y., Okada, Y., & Sato, K. (2023). Educational Recommendation System Utilizing Learning Styles : A Systematic Literature Review. IEEE Access, 11, 8988-8999. https://doi.org/10.1109/ACCESS.2023.3238417

Tomlinson, C. (2001). How to Differentiate Instruction in Mixed Ability Classrooms.

Trakosas, Dimitrios, Christina Tikva, et Efthimios Tambouris. 2023. « Visual Programming and Computational Thinking Environments for K-9 Education: A Systematic Literature Review ». International Journal of Learning Technology 18(1):94-121. doi: 10.1504/IJLT.2023.131313.

Tsai, Fu-Hsing, Chin-Chung Tsai, et Kuen-Yi Lin. 2015. « The Evaluation of Different Gaming Modes and Feedback Types on Game-Based Formative Assessment in an Online Learning Environment ». Computers & Education 81:259-69. doi: 10.1016/j.compedu.2014.10.013.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2011). Random k-Labelsets for Multi-Label Classification. IEEE Trans. Knowl. Data Eng., 23, 1079-1089. https://doi.org/10.1109/TKDE.2010.164

Ulferts, H., Willermark, S., Cooc, N., Kim, G., Brühwiler, C., Hollenstein, L., Tatto, M., Frey, A., & Fink, A. (2021). Teaching as a Knowledge Profession : Studying Pedagogical Knowledge across Education Systems. https://doi.org/10.1787/20769679

UNESCO. (2020). Education in a post-COVID world : Nine ideas for public action | UNESCO. https://www.unesco.org/en/articles/education-post-covid-world-nine-ideas-public-action

Uto, M., Tomikawa, Y., & Suzuki, A. (2023). Difficulty-Controllable Neural Question Generation for Reading Comprehension using Item Response Theory. In E. Kochmar, J. Burstein, A. Horbach, R. Laarmann-Quante, N. Madnani, A. Tack, V. Yaneva, Z. Yuan, & T. Zesch (Éds.), Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023) (p. 119-129). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.bea-1.10

Van Der Stel, M., & Veenman, M. V. J. (2014). Metacognitive skills and intellectual ability of young adolescents : A longitudinal study from a developmental perspective. European Journal of Psychology of Education, 29(1), 117-137. https://doi.org/10.1007/s10212-013-0190-5

Van-Roy, Peter, et Seif Haridi. 2004. Concepts, Techniques, and Models of Computer Programming. Cambridge, Mass: MIT Press.

Vanlehn, K. (2006). The Behavior of Tutoring Systems. International Journal of Artificial Intelligence in Education, 16, 227-265.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist, 46(4), 197-221. https://doi.org/10.1080/00461520.2011.611369

Vassileva, J. (2009). Toward Social Learning Environments. Learning Technologies, IEEE Transactions on, 1, 199-214. https://doi.org/10.1109/TLT.2009.4

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. https://arxiv.org/abs/1706.03762

Veerasamy, A. K., D'Souza, D., Lindén, R., & Laakso, M. (2019). Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses. Journal of Computer Assisted Learning, 35(2), 246-255. https://doi.org/10.1111/jcal.12326

Vega-Gorgojo, G., Bote-Lorenzo, M. L., Gómez-Sánchez, E., Dimitriadis, Y. A., & Asensio-Pérez, J. I. (2006). A semantic approach to discovering learning services in grid-based collaborative systems. Future Generation Computer Systems, 22(6), 709-719. https://doi.org/10.1016/j.future.2006.02.012

Viji, D., & Revathy, S. (2022). A hybrid approach of Weighted Fine-Tuned BERT extraction with deep Siamese Bi – LSTM model for semantic text similarity identification. Multimedia Tools and Applications, 81(5), 6131-6157. https://doi.org/10.1007/s11042-021-11771-6

Voigt, P., & Bussche, A. (2017). The EU General Data Protection Regulation (GDPR) : A Practical Guide. https://doi.org/10.1007/978-3-319-57959-7

Vuji\vcić, Lidija, Lucija Jan\vcec, et Jasminka Mezak. 2021. « DEVELOPMENT OF ALGORITHMIC THINKING SKILLS IN EARLY AND PRESCHOOL EDUCATION ». EDULEARN21 Proceedings.

VYGOTSKY, L. S. (1978a). Internalization of Higher Psychological Functions. In M. Cole, V. Jolm-Steiner, S. Scribner, & E. Souberman (Éds.), Mind in Society (p. 52-57). Harvard University Press; JSTOR. https://doi.org/10.2307/j.ctvjf9vz4.9

VYGOTSKY, L. S. (1978b). Mastery of Memory and Thinking. In M. Cole, V. Jolm-Steiner, S. Scribner, & E. Souberman (Éds.), Mind in Society (p. 38-51). Harvard University Press; JSTOR. https://doi.org/10.2307/j.ctvjf9vz4.8

Wang, F., & Hannafin, M. (2005). Design-based research and technology-enhanced learning environments. Educational Technology Research and Development, 53(4), 5-23. Educational Technology Research and Development, 53, 5-23. https://doi.org/10.1007/BF02504682

Wang, S., Dai, Y., Shen, J., & Xuan, J. (2021). Research on expansion and classification of imbalanced data based on SMOTE algorithm. Sci Rep, 11. https://doi.org/10.1038/s41598-021-03430-5

Wang, Xidong, Lei Zhao, et Jianhua Xu. 2018. « Multi-label Feature Selection Method Based on Multivariate Mutual Information and Particle Swarm Optimization ». P. 84-95 in Neural Information Processing, édité par L. Cheng, A. C. S. Leung, et S. Ozawa. Cham: Springer International Publishing.

Wang, Y., Wang, W., Joty, S., & Hoi, S. C. H. (2021). CodeT5 : Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In M.-F. Moens, X. Huang, L. Specia, & S. W. Yih (Éds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (p. 8696-8708). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.685

Wang, Yue, Weishi Wang, Shafiq Joty, et Steven C. H. Hoi. 2021. « CodeT5: Identifier-Aware Unified Pre-Trained Encoder-Decoder Models for Code Understanding and Generation ».

Watson, Christopher, et Frederick W. B. Li. 2014. « Failure Rates in Introductory Programming Revisited ». P. 39-44 in Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14. Uppsala, Sweden: ACM Press.

Weideman, A. (2013). Validation and validity beyond Messick. Per Linguam, 28. https://doi.org/10.5785/28-2-526

Whalen, S., Peisert, S., & Bishop, M. (2013). Multiclass classification of distributed memory parallel computations. Pattern Recognition Letters, 34(3), 322-329. https://doi.org/10.1016/j.patrec.2012.10.007

Wilkinson, Tracey, Mairead Boohan, et Michael Stevenson. 2014. « Does Learning Style Influence Academic Performance in Different Forms of Assessment? » J Anat 224(3):304-8. doi: 10.1111/joa.12126.

Wilson, J. R., & Lorenz, K. A. (2015). Short History of the Logistic Regression Model. In J. R. Wilson & K. A. Lorenz (Éds.), Modeling Binary Correlated Responses using SAS, SPSS and R (p. 17-23). Springer International Publishing. https://doi.org/10.1007/978-3-319-23805-0_2

Wing, Jeannette M. 2008. « Computational Thinking and Thinking about Computing ». Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366(1881):3717-25. doi: 10.1098/rsta.2008.0118.

Wing, Jeannette. 2006. « Computational Thinking ». Communications of the ACM 49:33-35. doi: 10.1145/1118178.1118215.

Winslow, Leon E. 1996. « Programming pedagogy—a psychological overview ». SIGCSE Bull. 28(3):17-22. doi: 10.1145/234867.234872.

Wongvorachan, T., He, S., & Bulut, O. (2023). A Comparison of Undersampling, Oversampling, and SMOTE Methods for Dealing with Imbalanced Classification in Educational Data Mining. Information, 14(1). https://doi.org/10.3390/info14010054

Woolf, B. (2008). Building Intelligent Interactive Tutors, Student-Centered Strategies for Revolutionizing E-Learning.

Wright, J. D. (2015). International Encyclopedia of the Social & Behavioral Sciences : Second Edition. In International Encyclopedia of the Social & Behavioral Sciences : Second Edition (p. 23185).

X. -Y. Liu, J. Wu, & Z. -H. Zhou. (2009). Exploratory Undersampling for Class-Imbalance Learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2), 539-550. https://doi.org/10.1109/TSMCB.2008.2007853

Xiao, Y., & Yang, M. (2019). Formative assessment and self-regulated learning : How formative assessment supports students' self-regulation in English language learning. System, 81, 39-49. https://doi.org/10.1016/j.system.2019.01.004

Xiong, Z., Li, H., Liu, Z., Chen, Z., Zhou, H., Rong, W., & Ouyang, Y. (2024). A Review of Data Mining in Personalized Education : Current Trends and Future Prospects. https://doi.org/10.3868/s110-009-024-0004-9

Yallihep, M., & Kutlu, B. (2019). Mobile serious games : Effects on students' understanding of programming concepts and attitudes towards information technology. Education and Information Technologies, 25(2), 1237-1254. https://doi.org/10.1007/s10639-019-10008-2

Yang, G., Zhou, Y., Chen, X., Zhang, X., Han, T., & Chen, T. (2023). ExploitGen : Template-augmented exploit code generation based on CodeBERT. Journal of Systems and Software, 197, 111577. https://doi.org/10.1016/j.jss.2022.111577

Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the Code Quality of AI-Assisted Code Generation Tools : An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT (arXiv:2304.10778). arXiv. http://arxiv.org/abs/2304.10778

Yilmaz, Ramazan, et Fatma Gizem Karaoglan Yilmaz. 2023. « The Effect of Generative Artificial Intelligence (AI)-Based Tool Use on Students' Computational Thinking Skills, Programming Self-Efficacy and Motivation ». Computers and Education: Artificial Intelligence 4:100147. doi: 10.1016/j.caeai.2023.100147.

Yohannes, Y., & Hoddinott, J. (1999). Classification and Regression Trees : An Introduction.

Z. Liu, X. Kong, H. Chen, S. Liu, & Z. Yang. (2023). MOOC-BERT: Automatically Identifying Learner Cognitive Presence From MOOC Discussion Data. IEEE Transactions on Learning Technologies, 16(4), 528-542. https://doi.org/10.1109/TLT.2023.3240715

Zare, S. (2011). Personalization in Mobile Learning for People with Special Needs. Universal Access in Human-Computer Interaction. Users Diversity. Stephanidis C. (eds) Universal Access in Human-Computer Interaction. Applications and Services., 6768, 662-669. https://doi.org/10.1007/978-3-642-21657-2_71

Zeller, Andreas. 2005. Why Programs Fail: A Guide to Systematic Debugging. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Zhang, H. (2004). The Optimality of Naive Bayes. The Florida AI Research Society. https://api.semanticscholar.org/CorpusID:8891634

Zhang, Z. (2016). Introduction to machine learning : K-nearest neighbors. Annals of Translational Medicine, 4(11). https://atm.amegroups.org/article/view/10170

МЕНЦИЕВ А, У., НГУЕН Х, Ф., & ЗАРИПОВА Р, С. (2023). COMPARATIVE ANALYSIS OF THE EFFECTIVENESS OF VADER NLTK AND ROBERTA IN THE CONTEXT OF SENTIMENT ANALYSIS. Приборы и системы. Управление, контроль, диагностика. https://doi.org/10.25791/pribor.11.2023.1451