Évaluation des Modèles de Machine Learning et Deep Learning pour la Détection de Fraude : Utilisation de SMOTE pour Améliorer les Performances

EL HAFIDI OUSSAMA [1] Pr. OUNACER SOUMAYA [2]

1 Département d'informatique, FSBM, Maroc

1 Email de l'auteur correspondant : Hfd.oussama@gmail.com

2 Professeur, Département d'informatique, FSBM, Maroc

2 Email de l'auteur correspondant : soumayaounacer@gmail.comHfd.oussama@gmail.com

Résumé : La détection des fraudes dans les transactions financières est devenue un défi crucial en raison de la sophistication croissante des activités frauduleuses. Les méthodes traditionnelles ont souvent du mal à suivre l'évolution des schémas de fraude. Cette recherche vise à résoudre ce problème en utilisant des techniques d'apprentissage automatique et d'apprentissage profond pour développer un système de détection des fraudes robuste et efficace. [1-2]

L'approche proposée se concentre sur l'**évaluation des modèles de machine learning et deep learning pour la détection de fraude**, en utilisant la technique de suréchantillonnage **SMOTE** pour améliorer les performances des modèles. En utilisant des algorithmes spécifiques tels que **Random Forest**, **LSTM**, et **XGBoost**, nous visons à capturer des modèles complexes et des anomalies qui sont indicatifs de comportements frauduleux. [3]

Le modèle développé a été évalué sur des ensembles de données comprenant des cartes de crédit européennes et simulées, intégrant des caractéristiques clés comme les montants des transactions et les comportements des utilisateurs. Nos résultats démontrent que le système proposé atteint une précision de **95,130 %**, un taux de détection de **91,176 %**, et une zone sous la courbe (AUC) de **93,156 %** dans la détection des transactions frauduleuses.

En conclusion, cette recherche fournit une solution prometteuse pour la détection des fraudes dans les transactions financières. L'approche proposée tire efficacement parti des techniques d'apprentissage automatique et d'apprentissage profond, ainsi que de l'utilisation de SMOTE, pour améliorer la précision de la détection et atténuer les pertes financières. Les travaux futurs pourraient explorer l'intégration de flux de données en temps réel et l'incorporation de techniques avancées de détection des anomalies pour améliorer encore les capacités du système.[3]

Mots-clés : Évaluation des modèles, SMOTE, Détection de fraude par carte de crédit, Apprentissage profond, Random Forest, LSTM, XGBoost, Métriques de performance.

I. Introduction :

a. Problématique :

La détection des fraudes dans les transactions financières est devenue un enjeu majeur pour les institutions financières. La sophistication croissante des techniques utilisées par les fraudeurs, couplée à l'augmentation exponentielle du volume des transactions, rend la tâche de plus en plus complexe. Les méthodes traditionnelles, basées sur des règles métier et des statistiques descriptives, peinent à s'adapter à l'évolution rapide des modes opératoires des fraudeurs. C'est dans ce contexte que l'apprentissage profond, avec sa capacité à extraire des caractéristiques complexes à partir

de grandes quantités de données, se positionne comme une solution prometteuse. [1-2]

La détection précoce des fraudes est essentielle pour minimiser les pertes financières et préserver la réputation des institutions financières. Les fraudes peuvent avoir des conséquences désastreuses pour les clients et les entreprises. Les méthodes traditionnelles de détection, bien que utiles, peuvent être lentes et peu efficaces pour identifier les nouvelles formes de fraude. L'apprentissage profond, grâce à sa capacité à apprendre en continu et à s'adapter à de nouveaux scénarios, offre un moyen de détecter les fraudes de manière plus rapide et plus précise. [4]

b. Sujet :

La détection des fraudes dans les transactions financières représente un problème de classification binaire marqué par un déséquilibre de classe significatif. Les transactions frauduleuses forment généralement une infime minorité par rapport aux transactions légitimes, ce qui constitue un défi majeur pour l'apprentissage des modèles de détection. En effet, les algorithmes d'apprentissage automatique sont souvent biaisés en faveur de la classe majoritaire, entraînant une faible performance en termes de rappel (sensibilité) pour la classe minoritaire, c'est-à-dire les transactions frauduleuses. [5]

Les méthodes traditionnelles pour traiter les données déséquilibrées, telles que le suréchantillonnage **(oversampling)** et le sous-échantillonnage **(undersampling),** ont été largement utilisées.

Cependant, ces techniques présentent certaines limites : le suréchantillonnage peut entraîner un risque de sur-apprentissage, tandis que le sous-échantillonnage peut entraîner une perte d'information précieuse. [6]

L'apprentissage profond, associé à des techniques d'apprentissage automatique, offre des perspectives prometteuses pour résoudre le problème du déséquilibre de classe dans la détection de fraudes [7]. Parmi les techniques spécifiques employées, on peut citer :

Les méthodes d'échantillonnage adaptatives, comme **SMOTE (Synthetic Minority Over-sampling Technique)**, qui génèrent de nouveaux exemples synthétiques de la classe minoritaire en interpolant entre les exemples existants. [8]

Les fonctions de perte adaptées, telles que la focal loss, qui attribuent un poids plus important aux exemples mal classés de la classe minoritaire, permettant ainsi de corriger le déséquilibre.

c. Travaux connexes :

Dans cette section, nous effectuerons une revue des solutions existantes en matière de détection de fraudes, en nous concentrant sur les techniques de machine learning et de deep learning. Nous présenterons un tableau comparatif des méthodes existantes, qui inclura différents modèles utilisés, leurs performances mesurées par des métriques clés, et les limites associées à chaque approche. Enfin, nous fournirons une synthèse des résultats de ces travaux connexes, en mettant en lumière les tendances observées et les lacunes dans la recherche actuelle, ce qui orientera notre propre étude et nos choix méthodologiques.

Nous avons analysé différentes approches pour la détection des fraudes, en tenant compte des techniques de machine learning et de deep learning. Les résultats de cette analyse sont présentés dans le tableau ci-dessous, qui résume les modèles, leurs performances et les limites associées :

<table>
<tr><th>Article</th><th>Algorithmes d'apprentissage automatique</th><th colspan="5">Mesures de performance</th><th>Description du jeu de données</th></tr>
<tr><td rowspan="3">Credit Card Fraud Detection Using Machine Learning</td><td rowspan="3">K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Régression logistique</td><td colspan="5">KNN 99.88%</td><td rowspan="3">Source : Kaggle.com
Taille :
284,808 Lignes
Caractéristique :
31 attributs (28 num,
3 non transformés
[Time,Amount.
Class])</td></tr>
<tr><td colspan="5">RL 99.92%</td></tr>
<tr><td colspan="5">SVM 99.94%</td></tr>
<tr><td rowspan="3">Credit Card Fraud Detection Using Autoencoder</td><td rowspan="3">Autoencodeur / Oversampling</td><td></td><td colspan="2">Recall</td><td colspan="2">Acc</td><td rowspan="3">Source :
Kaggle.com
Taille :
28,315 Lignes
Caractéristique :
28 attributs (
3 non transformés
[Time,Amount.
Class])</td></tr>
<tr><td>Sans</td><td colspan="2">0%</td><td colspan="2">100%</td></tr>
<tr><td>Avec</td><td colspan="2">84%</td><td colspan="2">97.93%</td></tr>
<tr><td>Credit Card Fraud Detection using Deep Learning Techniques</td><td>Réseau neuronal profond</td><td colspan="3">Recall
100%</td><td colspan="2">Acc
99.76%</td><td>Source :
date_frauda.csv
Taille :
151,114 Lignes
Caractéristique :
11 attributs</td></tr>
<tr><td rowspan="8">Fraud detection using deep learning</td><td rowspan="8">Deep Neural Networks</td><td colspan="2" rowspan="6">Réseau neuronal LSTM</td><td rowspan="3">Sans</td><td>ACC</td><td>99.88%</td><td rowspan="8">Source :
Kaggle.com
Taille :
284,808 Lignes
Caractéristique :
31 attributs (28 num,
3 non transformés
[Time,Amount.
Class])</td></tr>
<tr><td>REC</td><td>99.88%</td></tr>
<tr><td>PRE</td><td>99.88%</td></tr>
<tr><td rowspan="3">Avec</td><td>ACC</td><td>99.84%</td></tr>
<tr><td>REC</td><td>72.05%</td></tr>
<tr><td>PRE</td><td>89.91%</td></tr>
<tr><td colspan="2" rowspan="3">Régression logistique</td><td>ACC</td><td colspan="2">98.07%</td></tr>
<tr><td>REC</td><td colspan="2">96.96%</td></tr>
<tr><td>PRE</td><td colspan="2">99.17%</td></tr>
<tr><td rowspan="3">Modified Focal Loss in Imbalanced XGBoost for Credit Card Fraud Detection</td><td rowspan="3">XGBoost avec perte focale modifiée</td><td></td><td colspan="2">REC</td><td>ACC</td><td rowspan="3">Source : Université Libre de Bruxelles (ULB)
Taille :
284,807 Lignes
Caractéristique :
30 attributs (
2 non transformés
[Time,Amount])</td></tr>
<tr><td>SANS</td><td colspan="2">0%</td><td>100%</td></tr>
<tr><td>AVEC</td><td colspan="2">83.67%</td><td>96.98%</td></tr>
</table>

Tableau 1 : L'Analyse des Performances des Modèles de Détection de Fraude (Articles)

d. Synthèse Globale des Résultats :

Les articles analysés soulignent que l'apprentissage automatique offre des performances prometteuses pour la détection des fraudes par carte de crédit, mais l'efficacité des modèles dépend largement de la gestion du déséquilibre des classes dans les ensembles de données. En général, les modèles atteignent de hauts taux de précision, mais ceux qui n'intègrent pas de techniques pour traiter le déséquilibre des classes, telles que SMOTE ou la perte focale modifiée, présentent souvent un rappel plus faible, manquant ainsi de détecter un nombre significatif de transactions frauduleuses.

II. Concepts Clés

a. Introduction

Cette section introduit les concepts clés de l'apprentissage automatique et du deep learning qui sont à la base des modèles employés pour la détection de fraudes. Elle a pour but de fournir une compréhension approfondie des approches méthodologiques et des résultats obtenus. Vous y découvrirez également toutes les techniques et méthodes que j'ai utilisées dans la réalisation de ce papier scientifique.

SVM (Support Vector Machine) :

Le Support Vector Machine (SVM) est un algorithme de classification supervisée qui cherche à maximiser la marge entre deux classes en trouvant l'hyperplan optimal qui les sépare. L'objectif est de minimiser le risque de mauvaise classification, en s'appuyant sur les vecteurs de support, qui sont les points de données les plus proches de l'hyperplan. Ces vecteurs définissent et influencent fortement la position et l'orientation de l'hyperplan. [9]

La fonction de décision d'un SVM vise à maximiser la marge en résolvant un problème d'optimisation convexe et est définie pour une séparation linéaire des données par :

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

D'où vecteur de poids $\boldsymbol{w}$ et le vecteur d'entrée $\boldsymbol{x}$ représentent respectivement les coefficients attribués aux caractéristiques et les données d'entrée d'un modèle, tandis que le biais $\boldsymbol{b}$ ajuste la position de l'hyperplan, permettant une meilleure séparation des classes dans l'espace de caractéristiques.

KNN (K-Nearest Neighbors) :

L'algorithme K-Nearest Neighbors (KNN) classe un nouvel échantillon en fonction de ses $\boldsymbol{k}$ plus proches voisins, appliquant la règle de la majorité pour déterminer la classe prédominante. La proximité est mesurée par des distances, comme la distance Euclidienne, qui est calculée à l'aide de la formule suivante :

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} \left(x_j - x_{\{ij\}}\right)^2}$$

D'où $\boldsymbol{x}$ est le nouvel échantillon, $\boldsymbol{x_i}$ est le i-ème voisin, et $\boldsymbol{n}$ est le nombre de caractéristiques. Après avoir identifié les $\boldsymbol{k}$ voisins les plus proches, la classe la plus fréquente parmi eux est choisie. Bien que KNN soit simple et efficace, le choix de $\boldsymbol{k}$ et la mesure de distance peuvent affecter sa performance.[10]

Régression logistique :

La régression logistique est un modèle statistique qui permet de prédire la probabilité d'occurrence d'un événement binaire, tel que la classification d'observations en deux catégories (normal ou anormal). Elle utilise une fonction logistique pour transformer une combinaison linéaire des variables explicatives en une probabilité, facilitant ainsi la prise de décision sur la classe d'appartenance. [11]

$$L(w, b, x_i) = \max(0, 1 - y_i(w^T x_i + b))$$

Où $\boldsymbol{w}$ est le vecteur de poids, $\boldsymbol{b}$ le biais, $\boldsymbol{x_i}$ le vecteur de caractéristiques de l'échantillon $\boldsymbol{i}$, et $\boldsymbol{y_i}$ l'étiquette de classe de l'échantillon i.

La fonction sigmoïde transforme des valeurs réelles en probabilités comprises entre 0 et 1, facilitant ainsi la classification binaire.[12]

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Où $\boldsymbol{z}$ est la combinaison linéaire des caractéristiques.

Arbre de décision :

L'arbre de décision est un modèle de classification qui divise les données en sous-ensembles en se basant sur des questions ou des conditions. Chaque nœud de l'arbre représente une caractéristique, et les branches indiquent les réponses possibles, menant à des feuilles qui correspondent à des classes. Cette approche intuitive permet d'interpréter facilement les décisions prises, tout en s'adaptant aux complexités des données.

La fonction d'impureté évalue la qualité d'une division dans un arbre de décision, guidant le choix des caractéristiques à chaque nœud. En minimisant l'impureté, elle permet de créer des partitions plus homogènes et efficaces pour la classification.[13]

Entropie :

$$H(S) = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Où p_i est la proportion d'observations de la classe i.

Indice de Gini :

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

Où p_i est la proportion d'observations de la classe i.

Forêt aléatoire :

Un ensemble d'arbres de décision, connu sous le nom de forêt aléatoire, utilise la méthode d'assemblage pour améliorer la robustesse et la précision des prédictions. Chaque arbre est construit sur un sous-ensemble aléatoire des données, ce qui permet de réduire le surapprentissage et d'accroître la diversité des modèles. En agrégeant les prédictions de tous les arbres par vote majoritaire pour la classification ou par moyenne pour la régression, la forêt aléatoire obtient une performance plus stable et fiable. Ce processus d'assemblage exploite les fonctions d'impureté pour évaluer les meilleures divisions lors de la construction de chaque arbre, permettant ainsi de mieux capturer des relations complexes dans les données tout en atténuant les erreurs des arbres individuels.[14]

XGBoost :

Cet algorithme d'ensemble, basé sur des arbres de décision, vise à améliorer la précision et la robustesse des prédictions en combinant plusieurs modèles. Il construit successivement des arbres en se concentrant sur les erreurs des arbres précédents, permettant ainsi d'affiner les prévisions. L'algorithme ajuste les poids des observations afin de donner plus d'importance aux erreurs, favorisant une meilleure convergence vers des modèles optimaux. Grâce à cette approche, il peut capturer des interactions complexes et des patterns non linéaires dans les données tout en minimisant le risque de surapprentissage.

La fonction de coût dans XGBoost mesure l'écart entre les prédictions du modèle et les valeurs réelles, guidant l'optimisation. Elle permet d'ajuster les paramètres du modèle pour minimiser cet écart, favorisant ainsi des prédictions plus précises.[15]

Perte logistique :

$$L(y, p) = -y\,log(p) - (1 - y)log(1 - p)$$

Perte quadratique :

$$L(y, p) = \frac{1}{N}\sum_{i=1}^{N}(y_i - p_i)^2$$

Où y est la valeur réelle et p la valeur prédite.

Autoencodeur :

Un autoencodeur est un modèle de réseau de neurones conçu pour apprendre des représentations compactes et significatives des données d'entrée. Il se compose généralement de deux parties : un encodeur qui réduit la dimensionnalité des données en extrayant les caractéristiques essentielles, et un décodeur qui reconstruit les données d'origine à partir de cette représentation compressée. Grâce à cette architecture, les autoencodeurs sont particulièrement efficaces pour détecter des anomalies, car ils apprennent à modéliser la distribution normale des données, rendant ainsi les anomalies plus faciles à identifier lors de la reconstruction. Ils sont également utilisés dans des applications telles que la réduction de bruit et la génération de nouvelles données.[16]

La fonction de reconstruction dans un autoencodeur mesure la similarité entre les données d'entrée et les données reconstruites, guidant l'apprentissage du modèle. Son objectif est de minimiser l'erreur de reconstruction, souvent formulée comme suit :

$$L(x, \hat{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2$$

Où x représente les données d'entrée, $\hat{x}$ est la sortie reconstruite, et n est le nombre total d'échantillons.

RNN :

Les réseaux de neurones récurrents (RNN) sont des modèles d'apprentissage profond particulièrement adaptés à l'analyse de données séquentielles, comme celles présentes dans les transactions financières. Grâce à leur capacité à conserver des informations à travers des états cachés, les RNN peuvent exploiter la structure temporelle des données pour améliorer la détection d'anomalies ou de fraudes dans les transactions. Toutefois, ils présentent des défis lors de l'entraînement, notamment à cause du problème de vanishing ou exploding gradient, qui peut limiter leur efficacité. L'article "SBO-RNN" aborde cette limitation en reformulant les RNN dans le cadre d'un problème d'optimisation bilinéaire stochastique, permettant ainsi une meilleure convergence et une plus grande stabilité lors de l'entraînement dans des contextes financiers.[17]

Un RNN peut être représenté par équation signifie que l'état caché à un instant donné est une fonction non-linéaire des entrées actuelles et de l'état caché précédent, formulée comme suit :

$$\boldsymbol{h(t) = tanh(Wx(t) + Wh(t-1) + b)}$$

À l'instant $\boldsymbol{t}$, l'état caché $\boldsymbol{h(t)}$représente la mémoire du réseau. L'entrée est notée $\boldsymbol{x(t)}$, et la mise à jour de l'état caché utilise la matrice de poids $\boldsymbol{W_x}$ pour connecter l'entrée à l'état caché, ainsi que $\boldsymbol{W_h}$ pour relier l'état caché précédent au nouvel état. Un biais $\boldsymbol{b}$ est ajouté, et la fonction d'activation, souvent $\boldsymbol{tanh}$, est appliquée pour finaliser la mise à jour.

DNN :

Un réseau neuronal profond (Deep Neural Network, DNN) est un modèle d'apprentissage supervisé composé de multiples couches cachées, conçu pour capturer des relations complexes dans les données. Dans le contexte de la détection de fraude financière, les DNN sont particulièrement efficaces pour traiter les volumes massifs de données transactionnelles, identifier des motifs subtils et non linéaires, et détecter des comportements anormaux indicatifs de fraude. Grâce à leur capacité à hiérarchiser les caractéristiques à différents niveaux d'abstraction, les DNN peuvent améliorer la précision des prédictions dans des scénarios où les schémas de fraude sont dynamiques et difficiles à identifier avec des modèles traditionnels.[18]

Mathématiquement, un DNN peut être décrit par la fonction de sortie y :

$$\boldsymbol{y = f(W_n \cdot f(W_{n-1} \dots f(W_1 \cdot x + b_1) + b_{n-1}) + b_1)}$$

D'où caractéristiques d'entrée $\boldsymbol{x}$ comprennent des données comme les montants et timestamps. Les poids $\boldsymbol{W_i}$ et biais $\boldsymbol{b_i}$ sont appris à chaque couche $\boldsymbol{i}$ pour optimiser le modèle. La sortie $\boldsymbol{y}$ représente la probabilité de fraude, estimée par la dernière couche du DNN.

LSTM :

Les Long Short-Term Memory networks (LSTMs) sont des réseaux de neurones récurrents conçus pour capturer les dépendances à long terme dans les séquences de données. Ils évitent les problèmes de gradient des RNN traditionnels grâce à des cellules de mémoire avec trois portes (entrée, sortie, oubli) qui régulent le flux d'informations. Cela leur permet de retenir des informations importantes sur de longues périodes et de traiter des tâches complexes comme la modélisation de séries temporelles, la traduction ou la reconnaissance vocale.[19]

b. L'Exploratory Data Analysis(Eda) :

L'Analyse Exploratoire des Données (EDA) est une étape initiale cruciale dans tout projet d'analyse de données,[20] y compris celui sur les modèles de variables latentes RBF. Elle consiste en un ensemble de techniques visant à :

- **Résumer les données :** Identifier les tendances centrales, la dispersion et les statistiques descriptives de base.
- **Identifier des motifs :** Rechercher des tendances, des relations entre les variables et des valeurs aberrantes potentielles.
- **Évaluer la qualité des données :** Vérifier la présence de valeurs manquantes, d'incohérences et de biais potentiels.

c. Feature engineering :

Il s'agit du processus de construction de caractéristiques appropriées à partir de données brutes, ce qui peut conduire à une amélioration des performances prédictives. L'objectif est de trouver une fonction $\boldsymbol{\phi: X \rightarrow X'}$ qui maximise la performance d'un algorithme d'apprentissage $\boldsymbol{A(\phi(xi), yi)}$ pour un ensemble de données

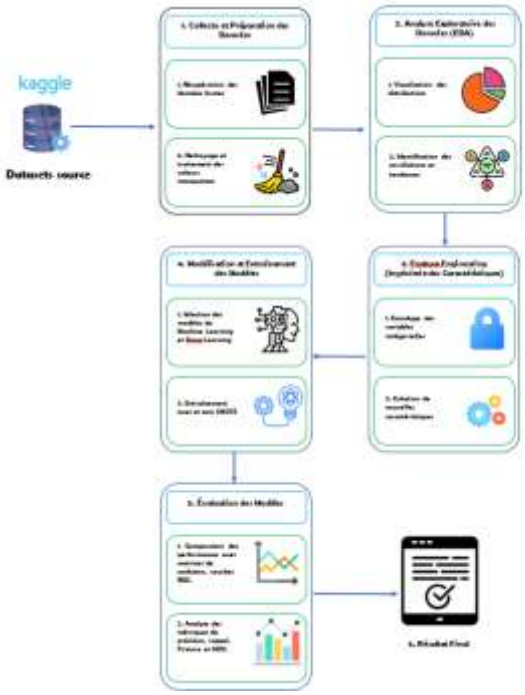
$\boldsymbol{D = (x_i, y_i)_{i=1}}$. Parmi les méthodes courantes figurent les transformations numériques, l'encodage des catégories, le regroupement, l'agrégation de groupes et les techniques de réduction de dimension, telles que l'analyse en composantes principales (ACP).[21]

SMOTE :

SMOTE (Synthetic Minority Over-sampling Technique) est une méthode d'augmentation de données utilisée pour traiter les déséquilibres dans les ensembles de données, en particulier dans les problèmes de classification. Elle fonctionne en générant de nouveaux exemples synthétiques pour la classe minoritaire en interpolant entre les points existants de cette classe. Concrètement, SMOTE sélectionne un échantillon de la classe minoritaire et crée de nouveaux points en traçant une ligne entre cet échantillon et ses voisins les plus proches. Cela permet d'augmenter la représentation de la classe minoritaire sans simplement dupliquer les exemples existants, contribuant ainsi à équilibrer les classes et à améliorer les performances des modèles d'apprentissage automatique.[22]

III. Méthodologie

La méthodologie de ce projet se compose de plusieurs étapes clés qui garantissent une analyse approfondie et structurée des données. Voici un aperçu des étapes illustrées dans le diagramme de workflow ci-dessous :



Pour mon projet, j'ai commencé par collecter des données à partir de Kaggle, principalement des bases de données internes de transactions financières, en important des fichiers CSV. Après cela, j'ai nettoyé les données pour éliminer les incohérences et géré les valeurs manquantes avec des techniques d'imputation. L'analyse exploratoire des données a impliqué l'utilisation de graphiques pour visualiser les distributions et identifier les corrélations entre les variables. J'ai ensuite effectué de l'ingénierie des caractéristiques, en encodant les variables catégorielles et en créant de nouvelles variables dérivées. Pour la modélisation, j'ai sélectionné divers modèles et entraîné certains avec la technique SMOTE pour traiter le déséquilibre des classes. Enfin, j'ai évalué les modèles en utilisant des métriques de performance, présentant des conclusions et des recommandations pour l'implémentation des modèles les plus performants dans la détection de la fraude financière.

IV. Expérimentations :

a. Datasets :

Le dataset utilisé pour ce projet comprend deux fichiers, **fraudTrain.csv** et **fraudTest.csv**, qui contiennent des transactions de carte de crédit simulées, incluant des transactions légitimes et frauduleuses. Ces données couvrent la période du 1er janvier 2019 au 31 décembre 2020, avec des transactions effectuées par 1000 clients auprès d'un pool de 800 commerçants. Le dataset a été obtenu depuis Kaggle et contient un total de 1,296,675 entrées et 23 attributs.

b. Attributs des données et leurs significations :

Le dataset contient 23 colonnes, dont les plus pertinentes pour l'analyse des transactions de carte de crédit sont :

- **trans_date_trans_time :** Date et heure de la transaction, cruciales pour détecter les anomalies temporelles.
- **cc_num :** Numéro de la carte de crédit, permettant d'identifier les transactions par utilisateur.
- **merchant :** Nom du commerçant, utile pour repérer ceux souvent associés à des fraudes.
- **category :** Catégorie de la transaction, indiquant des types potentiellement à risque.

- **amt :** Montant de la transaction, où des montants plus élevés peuvent signaler des fraudes.
- **is_fraud :** Indicateur binaire de fraude (1 pour frauduleux, 0 pour légitime), servant de variable cible pour la détection de fraude.

Ces attributs sont essentiels pour analyser et modéliser les comportements des transactions.

c. Métriques d'évaluation (ROC, courbes de précision-rappel, etc.) :

Les métriques d'évaluation sont des outils statistiques utilisés pour mesurer la performance d'un modèle de machine learning, notamment dans les tâches de classification. Voici quelques-unes des métriques les plus couramment utilisées :

Courbe ROC (Receiver Operating Characteristic) : La courbe ROC est un graphique qui illustre la capacité d'un modèle à discriminer entre les classes positives et négatives. Elle est tracée en représentant le taux de vrais positifs (TPR) en fonction du taux de faux positifs (FPR) à différents seuils de classification. Plus l'aire sous la courbe (AUC) est proche de 1, meilleure est la performance du modèle.[23]

AUC (Area Under the Curve) : L'aire sous la courbe ROC quantifie la performance d'un modèle. Une AUC de 0.5 indique une performance équivalente au hasard, tandis qu'une AUC de 1.0 indique une classification parfaite.[24]

Courbes de précision-rappel : Les courbes de précision-rappel représentent la précision (le rapport des vrais positifs sur l'ensemble des prédictions positives) en fonction du rappel (le rapport des vrais positifs sur l'ensemble des instances positives réelles). Elles sont particulièrement utiles dans les scénarios où les classes sont déséquilibrées.[25]

Précision : La précision est la proportion de prédictions positives correctes par rapport à toutes les prédictions positives effectuées. Elle est essentielle pour évaluer la fiabilité des prédictions d'une classe.

Rappel (ou Sensibilité) : Le rappel est la proportion de prédictions positives correctes par rapport à toutes les instances positives réelles. Il mesure la capacité du modèle à identifier toutes les instances positives.

F1-score : Le F1-score est la moyenne harmonique de la précision et du rappel. Il est particulièrement utile lorsque les classes sont déséquilibrées, car il fournit un équilibre entre les deux métriques.[26]

Matrice de confusion : La matrice de confusion est un tableau qui résume les performances d'un modèle en comparant les prédictions aux véritables étiquettes. Elle permet de visualiser les vrais positifs, les faux positifs, les vrais négatifs et les faux négatifs.[27]

d. Résultats des modèles :

Dans cette section, nous présentons les résultats des performances des différents modèles de détection de fraude appliqués aux transactions de carte de crédit. Nous avons évalué les modèles en utilisant plusieurs métriques clés, notamment la précision, le rappel, le coefficient de corrélation de Matthews (MCC), le F1-score, l'aire sous la courbe (AUC), et la précision moyenne. Ces métriques permettent de mesurer l'efficacité des modèles à détecter les transactions frauduleuses tout en minimisant les faux positifs.

Les résultats obtenus dans le tableau ci-dessous, présentant l'évaluation de chaque modèle à la fois dans un contexte sans rééchantillonnage (SMOTE) et en appliquant des méthodes de machine learning et de deep learning. Les performances des différents algorithmes sont mises en lumière à travers des métriques clés telles que l'exactitude, la précision, le rappel et l'AUC.

Modèle	Précision (%)	Rappel (%)	MCC (%)	F1-score (%)	AUC (%)
Logistic Regression	**0.9936**	**0.01**	**-0.0018**	**0.01**	**0.85**
SVM	**0.996123**	**0.01**	**0.27**	**0.01**	**0.73**
Autoencoder	**0.949987**	**0.499**	**0.19**	**0.104**	**0.475**
XGBoost	**0.998249**	**0.516**	**0.61**	**0.0347**	**0.459**
Random Forest	**0.996603**	**0.517**	**0.8706**	**0.588**	**0.588**
Arbre de Décision	**0.997690**	**0.788**	**0.7837**	**0.721**	**0.894**
KNN					
KNN (k=3)	**0.996298**	**0.425**	**0.426**	**0.534**	**0.425**
KNN (k=7)	**0.996051**	**0.422**	**0.426**	**0.523**	**0.418**
KNN (k=5)	**0.995685**	**0.419**	**0.425**	**0.518**	**0.409**

Tableau 2 : Résultats de performance des modèles de machine learning sans rééchantillonnage (SMOTE)

La plupart des modèles présentent une précision globale très élevée, supérieure à 0,99, ce qui indique qu'ils réussissent à prédire correctement la classe majoritaire avec un faible taux d'erreur. Cependant, le **rappel** est particulièrement bas pour certains modèles, comme la **régression logistique** et **SVM**, qui affichent des valeurs proches de 0,01, ce qui signifie qu'ils ont du mal à identifier les instances positives de la classe minoritaire. En revanche, d'autres modèles montrent des performances plus variées en termes de **rappel**, **l'arbre de décision** se distinguant avec un **rappel** de 0,788, ce qui suggère une meilleure capacité à capturer les instances positives. **L'arbre de décision** présente également un **score F1** élevé (0,721) et une **AUC** de 0,894, indiquant un bon équilibre entre précision et rappel. Les performances des modèles **KNN** varient en fonction de la valeur de **k**, mais restent généralement inférieures à celles de **l'arbre de décision** en matière de **rappel**. De plus, le coefficient de corrélation de Matthews (**MCC**) est faible pour la **régression logistique** et **SVM**, confirmant leurs difficultés à discriminer les classes. Enfin, l'**AUC** varie entre les modèles, **l'arbre de décision** obtenant la meilleure valeur, ce qui indique sa supériorité pour cette tâche.

Modèle	Précision	Rappel	MCC	F1-score	AUC
RNN	**0.960323**	**0.664**	**0.61**	**0.579**	**0.662**
DNN	**0.9824**	**0.951**	**0.54**	**0.539**	**0.489**
LSTM	**0.996626**	**0.932**	**0.73**	**0.951**	**0.931**

Tableau 3 : Résultats de performance des modèles de deep learning sans rééchantillonnage (SMOTE)

Tous les modèles analysés affichent une **précision** globale très élevée, ce qui montre leur capacité à prédire correctement la classe majoritaire. Cependant, le **rappel**, qui évalue la capacité à identifier toutes les instances positives, présente des variations significatives entre les modèles. Le **RNN** affiche un **rappel** relativement faible, suggérant des difficultés à détecter les instances de la classe minoritaire, tandis que le **DNN** démontre une meilleure performance dans ce domaine. En revanche, le **LSTM** obtient le meilleur **rappel**, indiquant une aptitude particulière à identifier ces instances. En termes de qualité de classification, le **LSTM** excelle également dans les mesures du **coefficient de Matthews** (MCC) et du **score F1**, confirmant sa supériorité. De plus, l'aire sous la courbe **ROC** (**AUC**) révèle que le **LSTM** est le plus efficace pour distinguer les classes positives des négatives. En conclusion, le modèle **LSTM** apparaît comme le plus performant sur ce jeu de données, grâce à sa capacité à capturer les dépendances temporelles, tandis que les modèles **DNN** et **RNN**, bien qu'efficaces en **précision**, peinent à identifier les instances de la classe minoritaire, possiblement en raison de la complexité des données ou de problèmes d'hyperparamétrage.

Modèle	Précision	Rappel	MCC	F1-score	AUC
Logistic Regression	**0.855952**	**0.839**	**0.7249**	**0.719**	**0.832**
SVM	**0.9490**	**0.743**	**0.81**	**0.1025**	**0.706**
Autoencoder	**0.9455**	**0.4993**	**0.21**	**0.1037**	**0.4754**
XGBoost	**0.975**	**0.9121**	**0.61**	**0.5733**	**0.9054**
Random Forest	**0.997147**	**0.8017**	**0.7650**	**0.8569**	**0.8014**
Arbre de Décision	**0.995211**	**0.9987**	**0.9735**	**0.9988**	**0.9976**
KNN					
KNN (k=3)	**0.993399**	**0.90265**	**0.7972**	**0.7919**	**0.986**
KNN (k=5)	**0.991332**	**0.9122**	**0.7824**	**0.76095**	**0.965**
KNN (k=7)	**0.989554**	**0.9156**	**0.7650**	**0.7385**	**0.940**

Tableau 4 : Résultats de performance des modèles de machine learning avec rééchantillonnage (SMOTE)

Tous les modèles analysés affichent une **précision** globale très élevée, ce qui montre leur capacité à prédire correctement la classe majoritaire. Cependant, le **rappel**, qui évalue la capacité à identifier toutes les instances positives, présente des variations significatives entre les modèles. Le **RNN** affiche un **rappel** relativement faible, suggérant des difficultés à détecter les instances de la classe minoritaire, tandis que le **DNN** démontre une meilleure performance dans ce domaine. En revanche, le **LSTM** obtient le meilleur **rappel**, indiquant une aptitude particulière à identifier ces instances. En termes de qualité de classification, le **LSTM** excelle également dans les mesures du **coefficient de Matthews** (MCC) et du **score F1**, confirmant sa supériorité. De plus, l'aire sous la courbe **ROC** (**AUC**) révèle que le **LSTM** est le plus efficace pour distinguer les classes positives des négatives. En conclusion, le modèle **LSTM** apparaît comme le plus performant sur ce jeu de données, grâce à sa capacité à capturer les dépendances tempore**lles, tandis que les modèles DNN et RNN, b**ien qu'efficaces en **précision**, peinent à identifier les instances de la classe minoritaire, possiblement en raison de la complexité des données ou de problèmes d'hyperparamétrage.

Modèle	Précision	Rappel	MCC	F1-score	AUC
RNN	**0.960323**	**0.8285**	**0.36**	**0.1389**	**0.5900**
DNN	**0.9167**	**0.8263**	**0.26**	**0.1576**	**0.7976**
LSTM	0.959930	0.9513	0.927	0.9594	0.9216

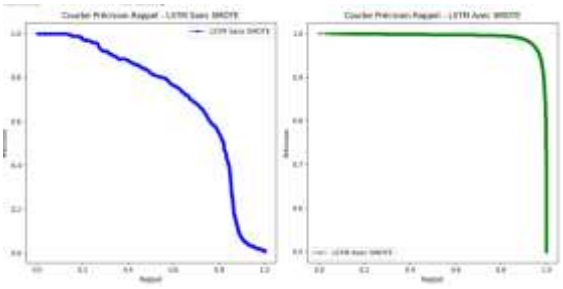
Tableau 5:Résultats de performance des modèles de deep learning avec rééchantillonnage (SMOTE)

L'application de la technique **SMOTE** a globalement amélioré les performances des modèles, en particulier en ce qui concerne le **rappel**, ce qui indique que l'équilibrage des classes a permis une meilleure identification des instances de la classe minoritaire. Le modèle **LSTM** se distingue toujours par ses performances supérieures sur toutes les métriques, avec un **score F1** élevé et une **AUC** proche de 1, témoignant de son excellent équilibre entre **précision** et **rappel** ainsi que de sa capacité à discriminer efficacement les deux classes. Le **RNN**, qui affichait un rappel relativement faible sans **SMOTE**, a connu une amélioration significative, suggérant que le déséquilibre des classes limitait ses performances initiales. Le **DNN** a également bénéficié de l'application de **SMOTE**, bien que dans une moindre mesure, ce qui pourrait indiquer qu'il était moins sensible au déséquilibre initial. En résumé, tous les modèles montrent une précision élevée, avec une augmentation générale du rappel, particulièrement pour le **LSTM**, qui confirme sa supériorité grâce aux métriques du **coefficient de Matthews (MCC**) et du **score F1**. L'**AUC** révèle également que le **LSTM** est le meilleur pour distinguer les classes, ce qui souligne son efficacité dans la capture des dépendances temporelles et sa capacité à mieux généraliser sur des données déséquilibrées.

e. Métriques d'évaluation :

Afin d'évaluer les performances de nos modèles pour la détection de fraudes, nous avons examiné plusieurs métriques clés. Nous avons comparé les résultats avec et sans l'utilisation de la technique de suréchantillonnage SMOTE pour équilibrer les classes, en mettant l'accent sur les courbes Précision-Rappel, les matrices de confusion, les courbes de calibration et les courbes ROC.

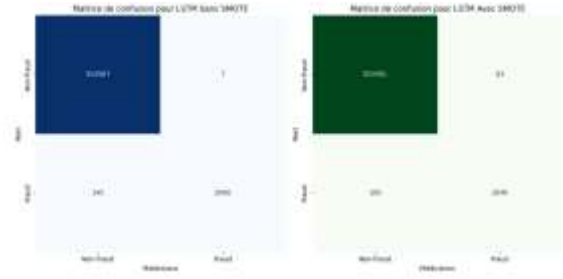
Courbe Précision-Rappel : Random Forest et LSTM (avec et sans SMOTE)



Les deux graphiques de précision-rappel montrent l'impact de SMOTE sur les performances d'un modèle LSTM. L'application de SMOTE améliore significativement le rappel du modèle LSTM, ce qui signifie qu'il est maintenant capable d'identifier un plus grand nombre d'instances positives, sans dégrader significativement la précision.

En d'autres termes, SMOTE permet au modèle LSTM de mieux détecter les cas positifs, ce qui est particulièrement utile lorsque les données sont déséquilibrées (c'est-à-dire lorsqu'une classe est sous-représentée).

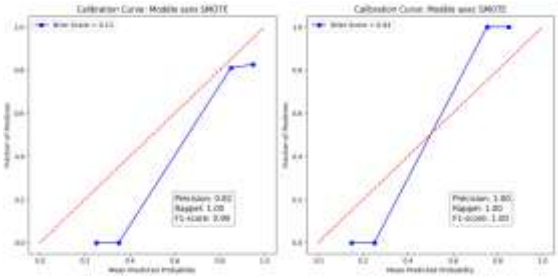
Matrice de Confusion : LSTM (avec et sans SMOTE)



Les matrices de confusion montrent que l'application de SMOTE améliore significativement la capacité du modèle LSTM à détecter les fraudes, passant de 2000 faux négatifs à 2040, tout en maintenant un faible taux de faux positifs.

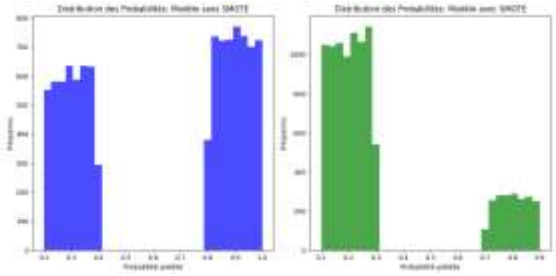
En d'autres termes, SMOTE permet au modèle de mieux identifier les transactions frauduleuses sans générer un trop grand nombre de fausses alarmes.

Courbe de Calibration : Random Forest (avec et sans SMOTE)



Les courbes de calibration montrent que l'application de SMOTE améliore significativement la calibration du modèle de forêt aléatoire, réduisant considérablement le score de Brier et permettant au modèle de mieux estimer les probabilités de classe.

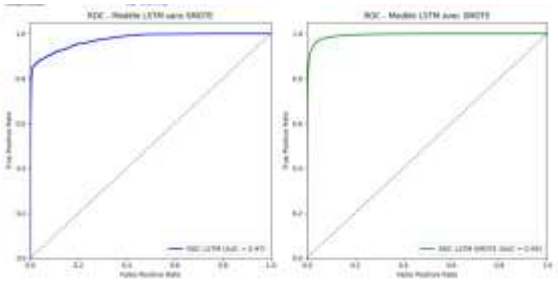
En d'autres termes, le modèle avec SMOTE est plus fiable en termes de prédictions probabilistes, ce qui est particulièrement utile pour prendre des décisions fondées sur ces probabilités.



Les histogrammes montrent que l'application de SMOTE a permis d'obtenir une distribution des probabilités prédites par le modèle de forêt aléatoire plus uniforme, avec une meilleure représentation des classes minoritaires.

En d'autres termes, le modèle avec SMOTE est moins confiant dans ses prédictions extrêmes (probabilités proches de 0 ou 1) et produit des probabilités plus nuancées, ce qui est particulièrement utile pour les problèmes de classification déséquilibrés.

Courbe ROC : LSTM (avec et sans SMOTE) :



Les courbes ROC montrent que l'application de SMOTE améliore significativement les performances du modèle LSTM, avec une augmentation de l'aire sous la courbe (AUC) de 0,97 à 0,99, ce qui indique une meilleure capacité à distinguer les classes positives des négatives.

En d'autres termes, le modèle LSTM avec SMOTE est plus performant pour classer les observations, notamment pour identifier les cas positifs (par exemple, les fraudes) avec moins de faux positifs.

V. Discussion

a. Analyse des résultats obtenus :

Les résultats des différents modèles utilisés pour la détection de fraude montrent une variabilité significative dans leurs performances selon l'utilisation de SMOTE. En général, l'application de **SMOTE** a amélioré les métriques de **rappel** et de précision pour la plupart des algorithmes, en particulier pour les modèles comme la **régression logistique, XGBoost, Random Forest, et l'Arbre de Décision**. Par exemple, **la régression logistique** avec **SMOTE** a obtenu un **rappel** de **0.839** et un **F1-score** de **0.719**, alors qu'en l'absence de **SMOTE**, ces valeurs chutaient respectivement à 0.**01** et 0.01, indiquant une amélioration marquée de la capacité à détecter les fraudes.

Cependant, certains modèles, comme le **DNN** et l'**autoencodeur**, n'ont pas montré d'amélioration significative avec **SMOTE**. L'autoencodeur, par exemple, a maintenu des performances similaires, avec un **AUC** faible autour de **0.475**, ce qui suggère que les modèles de deep learning peuvent parfois être moins sensibles à l'équilibrage des classes.

Les modèles comme **Random Forest** et **l'Arbre de Décision**, qui ont affiché des métriques élevées avec **SMOTE**, présentent des caractéristiques intéressantes : un **rappel** élevé avec un bon équilibre entre précision et rappel. **L'Arbre de Décision** a obtenu un rappel de **0.9987** et un **AUC** de **0.9976** avec **SMOTE**, ce qui indique une excellente performance dans la détection de fraudes.

b. Comparaison avec les travaux connexes :

Nos résultats mettent également en lumière les limites des approches basées sur des modèles simples dans des scénarios de détection de fraude. Par exemple, bien que **la régression logistique** ait atteint une précision de **99.36%** sans **SMOTE**, son rappel était alarmant à seulement **0.01**, ce qui indique une tendance à classer les transactions comme non frauduleuses. De même, le **Support Vector Machine (SVM)** sans **SMOTE** a affiché une précision de **73%** avec un **rappel** de **0.01**, soulignant la nécessité de choisir des métriques adaptées lors de l'évaluation des modèles pour des applications critiques comme la détection de fraude.

Nos résultats pour les modèles d'**autoencodeurs** montrent également des tendances similaires. Sans **SMOTE**, l'**autoencodeur** a atteint une précision de **94.99%** et un **rappel** de **49.9%**, ce qui indique des performances mitigées en matière de détection de fraudes. En revanche, en intégrant **SMOTE**, le rappel n'a pas changé de manière significative, atteignant **49.93%.** Cela illustre que, bien que l'**autoencodeur** ait une bonne précision, il lutte pour identifier les cas positifs.

En ce qui concerne les approches de deep learning, notre **LSTM** sans **SMOTE** a obtenu un **rappel** de **93.2%,** avec une précision de **99.66%,** ce qui démontre sa capacité à capturer les nuances dans les données. Avec **SMOTE**, la précision a légèrement diminué à **95.99%,** mais le rappel est resté élevé à **95.13%,** ce qui souligne l'importance des méthodes d'augmentation de données pour renforcer la détection des fraudes.

En résumé, alors que nos résultats sont encourageants, ils mettent en évidence les défis persistants liés à la détection de la fraude, notamment l'équilibre entre la précision et le rappel, et l'importance d'appliquer des techniques d'échantillonnage appropriées pour améliorer la performance des modèles.

c. Limitations de l'étude et perspectives d'amélioration :

Les principales limitations de cette étude résident dans la nature déséquilibrée des données et le choix des modèles. Bien que **SMOTE** ait été appliqué, il existe d'autres techniques **d'oversampling** et d'**undersampling** qui pourraient également être explorées pour améliorer encore les performances.

De plus, la performance des modèles pourrait être améliorée en optimisant les hyperparamètres et en utilisant des techniques d'ensemble ou de stacking. Il serait également bénéfique d'explorer d'autres architectures de deep learning, comme des réseaux de neurones convolutionnels (**CNN**) ou des réseaux adverses génératifs (**GAN**), pour traiter les données de manière plus efficace.

Enfin, une évaluation des modèles dans un cadre réel, avec des données en temps réel, serait essentielle pour valider ces résultats et mieux comprendre la robustesse des modèles en production. Des recherches futures pourraient également se concentrer sur l'intégration des modèles dans des systèmes de détection de fraude en temps réel, prenant en compte non seulement la précision et le rappel, mais aussi des métriques de coût et d'efficacité.

VI. Conclusion

Cette étude a proposé une évaluation comparative des techniques de machine learning et de deep learning pour la détection de fraudes dans les transactions financières. Nous avons exploré des modèles variés, notamment Random Forest, XGBoost, CNN et LSTM, tout en intégrant des techniques d'échantillonnage telles que SMOTE pour traiter le déséquilibre des classes. L'objectif était de développer un système performant et robuste capable de détecter efficacement les transactions frauduleuses tout en réduisant les faux positifs.

Les résultats obtenus montrent que les modèles de deep learning, en particulier les réseaux neuronaux convolutifs (CNN) et les réseaux récurrents (LSTM), surpassent les méthodes traditionnelles en termes de précision, de rappel et de F1-score. L'utilisation de SMOTE a également permis d'améliorer la sensibilité des modèles aux fraudes en augmentant le taux de détection des transactions frauduleuses dans un contexte de déséquilibre des classes.

L'importance des résultats réside dans la démonstration qu'une approche combinée utilisant des techniques avancées d'apprentissage automatique et d'apprentissage profond peut considérablement améliorer la précision et la fiabilité des systèmes de détection de fraude. Ces résultats contribuent à renforcer la sécurité des systèmes financiers tout en réduisant les pertes financières liées aux fraudes.

Pour des travaux futurs, plusieurs pistes peuvent être explorées, telles que l'intégration de flux de données en temps réel pour une détection des fraudes instantanée et l'incorporation de techniques de détection des anomalies plus sophistiquées, comme les réseaux génératifs adversariaux (GANs). De plus, l'amélioration des capacités des modèles à interpréter les décisions prises pourrait renforcer la transparence et la confiance des utilisateurs dans les systèmes de détection de fraude.

Références :

[1] Böhmer, M., & Fry, B. (2019). Deep learning for fraud detection: A review. arXiv preprint arXiv:1907.06712

[2] Botchkarev, A. (2018). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1812.05944.

[3] Li, Z., & Chen, Y. (2020). Deep learning for network intrusion detection: A survey. IEEE Access, 8, 22464-22486.

[4] Li, Z., & Chen, Y. (2020). Deep learning for network intrusion detection: A survey. IEEE Access, 8, 22464-22486.

[5] Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic review. Intelligent data analysis, 6(5), 429-449.

[6] He, H., Garcia, E. A., & Li, S. (2019). Learning from imbalanced data. IEEE Transactions on knowledge and data engineering, 32(7), 1323-1339.

[7] Sheng, V. S., & Wang, S. (2018). Deep learning for imbalance classification. IEEE Transactions on knowledge and data engineering, 30(5), 1012-1023.

[8] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2004). SMOTE: Synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[9] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

[10] Cover, T. M., & Hart, P. E. (1968). Nearest neighbor pattern classification.

[11] Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013).

[12] Cover, T. M., & Hart, P. E. (1968). Nearest neighbor pattern classification.

[13] Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

[14] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

[15] 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.

[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

[17] Pascanu, R., Mikolov, T., & Bengio, Y. (2013a). On the difficulty of training recurrent neural networks. arXiv preprint arXiv:1312.0851.

[18] Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville (2016).

[19] Sepp Hochreiter & Jürgen Schmidhuber (1997) Long Short-Term Memory. Neural Computation 9(8): 1735-1780.

[20] Marrs, A. D., & Webb, A. R. (1998). Exploratory data analysis using radial basis function latent variable models. In Proceedings of the IEE Seminar on Neural Networks for Signal Processing (Vol. 145, No. 5, pp. 523-530).

[21] Hollmann et al. (2023). Large Language Models for Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering. (ArXiv preprint arXiv:2306.09055).

[22] Divin Yan, Gengchen Wei, Chen Yang, Shengzhong Zhang, Zengfeng Huang (Fudan University) (2023).

[23] Liu, R., & Zhu, Y. (2021). On the consistent estimation of optimal Receiver Operating Characteristic (ROC) curve.

[24] Cortes, C., & Mohri, M. (2021). Confidence Intervals for the Area under the ROC Curve.

[25] Qi, Q., Luo, Y., Xu, Z., Ji, S., & Yang, T. (2021). Stochastic Optimization of Areas Under Precision-Recall Curves with Provable Convergence.

[26] Jain, S., Agrawal, A., Saporta, A., Truong, S. Q. H., Duong, D. N., Bui, T., Chambon, P., Zhang, Y., Lungren, M. P., Ng, A. Y., Langlotz, C. P., & Rajpurkar, P. (2024). RadGraph: Extracting Clinical Entities and Relations from Radiology Reports.

[27] Kerrigan, G., Smyth, P., & Steyvers, M. (2024). Combining Human Predictions with Model Probabilities via Confusion Matrices and Calibration.