République Algérienne Démocratique et populaire

Ministère del'Enseignement Supérieur et de la Recherche Scientifique

Université Larbi Ben M'hidi Oum El Bouaghi

Faculté des Sciences Exactes et Science de la Nature et de la Vie

Département de Mathématique et Informatique

Filière : Informatique

Spécialité : Architectures distribuées

Mémoire Présenté en vue de l'obtention du Diplôme de Master en Informatique

Etude du problème des datasets déséquilibrés Dans les algorithmes D 'apprentissage automatique

Présenter par : Soualem Maamoune

Devant le jury :

Président	**Dr.Boussaha Karima**	**Université d'Oum Bouaghi**
Examinateur	**Dr.Khantoul Bilel**	**Université d'Oum Bouaghi**
Encadrant	**Dr.Hdouci Adnane**	**Université d'Oum Bouaghi**

Année Universitaire : **2022 / 2023**

Résumé :

Ce travail se concentre sur l'étude du problème de Dataset déséquilibrés dans les algorithmes d'apprentissage automatique. Les dataset déséquilibrés sont courants dans de nombreux domaines, où les exemples de la classe minoritaire sont rares par rapport à ceux de la classe majoritaire. Cela entraîne des défis pour les modèles
d'apprentissage automatique, car ils ont tendance à être biaisés en faveur de la classe majoritaire, ce qui conduit à des performances médiocres pour la classe minoritaire.

Dans cette étude, nous proposons d'utiliser trois techniques de pré-traitement des données pour équilibrer les dataset :

- Le Sous-échantillonnage (RUS - Radom UnderSampling)

- Le Sur-échantillonnage (ROS - Radom Over-Sampling)

- La technique SMOTE (Synthetic Minority Over-sampling Technique)

Le sous-échantillonnage consiste à supprimer aléatoirement des exemples de la classe majoritaire afin d'équilibrer les classes, tandis que le sur-échantillonnage duplique aléatoirement des exemples de la classe minoritaire. SMOTE génère quant à lui de nouveaux exemples synthétiques de la classe minoritaire en interpolant les caractéristiques des exemples existants.

L'objectif de cette étude est de comparer l'efficacité de ces techniques de pré-traitement sur des dataset déséquilibrés. Nous évaluerons les performances des modèles d'apprentissage automatique, tels que la régression logistique, Les réseaux de neurones et les forêts aléatoires, sur ces dataset pré-traités. Les mesures de performance, telles que la précision, le rappel, le score AUC et le F1-score, seront utilisées pour évaluer la capacité des modèles à classer correctement les exemples des classes minoritaires.

Les résultats de cette étude permettront de mieux comprendre l'impact des dataset déséquilibrés sur les performances des modèles d'apprentissage automatique, ainsi que l'efficacité des techniques de pré-traitement pour résoudre ce problème.

Mots clés : L'apprentissage automatique, Dataset déséquilibrés,

Les techniques de pré-traitement des données

Summary :

This work focus on the study of the problem of unbalanced datasets in machine learning algorithms. Unbalanced datasets are common in many fields, where examples of the minority class are sparse compared to those of the majority class. This leads to challenges for models

machine learning because they tend to be biased towards the majority class, leading to poor performance for the minority class.

In this study, we propose to use three data pre-processing techniques to balance the dataset:

- Under-sampling (RUS - Radom UnderSampling)

- Over-sampling (ROS - Radom Over-Sampling)

- The SMOTE technique (Synthetic Minority Over-sampling Technique)

Undersampling consists of randomly removing examples from the majority class in order to balance the classes, while oversampling randomly duplicates examples from the minority class. SMOTE generates new synthetic examples of the minority class by interpolating the characteristics of the existing examples.

The objective of this study is to compare the effectiveness of these pre-processing techniques on unbalanced datasets. We will evaluate the performance of machine learning models, such as logistic regression, neural networks, and random forests, on these pre-processed datasets. Performance measures, such as precision, recall, AUC-score and F1-score, will be used to assess the ability of the models to correctly classify examples of minority classes.

The results of this study will provide insight into the impact of unbalanced datasets on the performance of machine learning models, as well as the effectiveness of pre-processing techniques to address this issue.

Keywords: Machine learning, Unbalanced dataset, Data pre-processing techniques

ملخص

يركزهدا العمل على دراسة مشكلة مجموعات البيانات غير المتوازنة في خوارزميات التعلم الآلي. مجموعات البيانات غير المتوازنة ش ائعة في العديد من المجالات ، حيث تكون أمثلة فئة الأقلية متفرقة مقارنة بتلك الخاصة بفئة الأغلبية. هذا يؤدي إلى تحديات لنماذج التعلم الآلي ، لأنها تميل إلى أن تكون منحازة نحو طبقة الأغلبية ، مما يؤدي إلى أداء ضعيف لفئة الأقلية.

في هذه الدراسة ، نقترح استخدام ثلاث تقنيات للمعالجة المسبقة للبيانات لموازنة مجموعة البيانات:

- تقنية التخفيف (RUS – Radom UnderSampling)
- تقنية التحميل (ROS – Radom Over–Sampling)
- تقنية SMOTE (تقنية الإفراط في أخذ العينات الاصطناعية للأقليات الاصطناعية)

يتكون RUS من إزالة أمثلة عشوائية من فئة الأغلبية من أجل تحقيق التوازن بين الفئات ، في حين أن ROS يؤدي إلى تكرار لأمثلة من فئة الأقلية. يولد SMOTE أمثلة تركيبية جديدة لفئة الأقلية من خلال استيفاء خصائص الأمثلة الموجودة.

الهدف من هذه الدراسة هو مقارنة فعالية تقنيات ما قبل المعالجة هذه على مجموعات البيانات غير المتوازنة. سنقوم بتقييم أداء نماذج التعلم الآلي ، مثل الانحدار اللوجستي ، والشبكات العصبية ، والغابات العشوائية ، على مجموعات البيانات التي تمت معالجتها مسبقًا. سيتم استخدام مقاييس الأداء ، مثل الدقة والاستدعاء ودرجة AUC ودرجة F1 ، لتقييم قدرة النماذج على تصنيف أمثلة فئات الأقليات بشكل صحيح.

ستوفر نتائج هذه الدراسة نظرة ثاقبة حول تأثير مجموعات البيانات غير المتوازنة على أداء نماذج التعلم الآلي ، فضلاً عن فعالية تقنيات المعالجة المسبقة لمعالجة هذه المشكلة.

الكلمات المفتاح: التعلم الآلي ، مجموعات البيانات غير المتوازنة ، تقنيات المعالجة المسبقة للبيانات

Remerciements

Je remercie Dieu tout puissant pour la santé, la volonté, le courage et la patience qu'il m'a donnés durant ces années d'étude.

Je souhaite exprimer toute ma reconnaissance à mes parents qui m'ont soutenue tout au long de ces années et m'ont toujours encouragée à faire ce que je souhaitais et à donner le meilleur de moi-même.

Ce travail a été réalisé dans le cadre d'un mémoire de Master de L'université Larbi Ben Mhidi d'Oum El Bouaghi .

Je tiens tout d'abord à remercier et à témoigner notre reconnaissance au Dr HIDOUCI Adnane pour avoir encadré ce travail.

Son dynamisme, sa patience et ses qualités humaines sont une source permanente durant ce travail.

Nous tenons à exprimer notre profonde gratitude et nos sincères remerciements aux: Membres de jury d'avoir accepté de juger notre travail et de l'avoir enrichi.

Je remercie l'ensemble des enseignants et étudiants du département de l'Informatique.

Dédicaces

A mon cher père, ma chère mère, ma source de patience et de force

À mon honorable cheikh Zahi Khudair, qui m'a guidé pour atteindre cette étape

A tous ceux qui m'ont soutenu financièrement et moralement

A mes frères, sœurs, amis, la source de mon bonheur

A tous ceux qui m'ont encouragé et aidé même d'un mot

À mes camarades de classe et frères en résidence, vous avez tout l'amour et l'appréciation

Ce message est le fruit de 16 ans d'étude, de veille, de fatigue et de pression, et si Dieu le veut, ce sera le début de notre succès et de notre éclat dans la vie

Table de matières

Table de matières

Liste des Figures

Liste des tableaux

Introduction Générale

1. Context de recherche

L'étude du problème des datasets déséquilibrés dans les algorithmes d'apprentissage automatique est un domaine de recherche en plein essor.

Avec l'avènement de l'intelligence artificielle et de l'apprentissage automatique, les algorithmes basés sur des modèles prédictifs sont de plus en plus utilisés pour résoudre une grande variété de problèmes dans de nombreux domaines, tels que la santé, la finance, la sécurité et bien d'autres encore.

Cependant, l'un des défis majeurs auxquels sont confrontés les chercheurs et les praticiens de l'apprentissage automatique est le déséquilibre des datasets utilisés pour l'entraînement de ces modèles.

Dans de nombreux cas, les datasets réels sont caractérisés par une répartition inégale des classes, c'est-à-dire que certaines classes sont sous-représentées tandis que d'autres sont sur-représentées. Par exemple, dans un problème de détection de fraudes, les transactions frauduleuses représentent généralement une petite proportion de l'ensemble des données disponibles.

Le déséquilibre des datasets peut entraîner des problèmes majeurs lors de l'entraînement des modèles d'apprentissage automatique. Les algorithmes de base ont tendance à être biaisés en faveur des classes majoritaires, ce qui peut conduire à une mauvaise performance de prédiction pour les classes minoritaires.

Par conséquent, il est essentiel de développer des méthodes et des techniques spécifiques pour aborder ce problème et améliorer les performances des modèles sur les classes sousreprésentées.

Le contexte de recherche dans l'étude des datasets déséquilibrés consiste à identifier les causes du déséquilibre, à évaluer les impacts sur les performances des modèles d'apprentissage automatique et à proposer des approches pour remédier à ce problème.

Les chercheurs s'intéressent à la fois aux aspects théoriques et pratiques de ce problème, en explorant des méthodes telles que le sur-échantillonnage, le souséchantillonnage, la génération synthétique de données et les algorithmes d'apprentissage en ligne.

L'objectif final de cette recherche est de permettre aux algorithmes d'apprentissage automatique de traiter de manière efficace et équitable les datasets déséquilibrés, en améliorant la performance de prédiction pour toutes les classes et en garantissant une prise de décision équilibrée. Cela aura un impact significatif sur de nombreux domaines d'application et contribuera à l'avancement de l'apprentissage automatique dans des scénarios réels.

2. Problématique, Motivation et Positionnement

Problématique :

La problématique de l'étude des datasets déséquilibrés dans les algorithmes d'apprentissage automatique réside dans le fait que les modèles traditionnels ont tendance à être biaisés en faveur des classes majoritaires, ce qui conduit à une mauvaise performance de prédiction pour les classes minoritaires. Cela pose un défi majeur pour les chercheurs et les praticiens qui cherchent à développer des modèles d'apprentissage automatique performants et équitables dans des scénarios réels où les datasets déséquilibrés sont courants. La problématique de cette étude consiste donc à trouver des solutions efficaces pour améliorer la performance des modèles sur les classes sous-représentées tout en maintenant une équité dans la prise de décision.

Motivation :

La motivation derrière cette étude est multiple. Tout d'abord, l'utilisation croissante de l'apprentissage automatique dans des domaines sensibles tels que la santé et la finance nécessite des modèles fiables et équitables, capables de traiter des datasets déséquilibrés. Ensuite, l'amélioration de la performance de prédiction pour les classes minoritaires est essentielle dans de nombreux scénarios réels, où la détection de cas rares ou d'événements critiques peut avoir des conséquences majeures. De plus, le développement de méthodes efficaces pour traiter les datasets déséquilibrés peut contribuer à réduire les biais et les inégalités dans les systèmes d'intelligence artificielle, favorisant ainsi une utilisation plus éthique et responsable de ces technologies.

Positionnement :

Cette étude se positionne à la croisée de l'apprentissage automatique, de la statistique et de la recherche opérationnelle. Elle vise à explorer les différentes méthodes et techniques existantes pour aborder le problème des datasets déséquilibrés, en les évaluant de manière critique et en

proposant des améliorations. L'approche adoptée peut être à la fois théorique, en proposant de nouvelles méthodes d'échantillonnage ou de génération de données, et pratique, en développant des outils et des frameworks pour faciliter l'application de ces méthodes dans des scénarios réels. L'objectif ultime est de contribuer à l'avancement des connaissances et des pratiques dans le domaine de l'apprentissage automatique, en permettant aux modèles de faire face efficacement aux datasets déséquilibrés et de prendre des décisions équitables et précises dans des situations réelles.

3. Organisation du mémoire

Le présent mémoire est organisé en trois principaux chapitres en plus de l'introduction générale et la conclusion générale.

➢ Chapitre 1 : « **Généralités sur l'Apprentissage automatique** » est un domaine de l'intelligence artificielle.

Notre présentation s'appuie sur définition de l'apprentissage automatique,

Ses types, les algorithmes de classification, les étapes pour crées un modèle

d'apprentissage automatique ...

Chapitre 2 : « **Dataset non équilibré** » est consacré à une présentation générale du problème des données déséquilibrées qui est devenu un problème majeur pour l'apprentissage automatique, nous présenterons les solutions à ce problème à travers la méthode d'équilibrages de dataset RUS, ROS , SMOTE

➢ Chapitre 3 : présente nos différentes contributions et toutes les expérimentations que nous avons réalisé au cours de notre étude, ce chapitre présente aussi les discussions des résultats obtenus et une comparaison avec l'état de l'art. Une conclusion générale qui synthétise toutes les idées extraites de nos différentes expérimentations et présente aussi les pistes de recherche ouvertes à partir de ce travail de thèse de master.

Chapitre 01 : Généralités sur l'apprentissage automatique.

Introduction

L'apprentissage automatique, également connu sous le nom de machine learning en anglais, est un sous-domaine de l'intelligence artificielle (IA) qui se concentre sur le développement de méthodes et d'algorithmes permettant aux ordinateurs d'apprendre à partir de données et d'améliorer leur performance sans être explicitement programmés. En d'autres termes, l'apprentissage automatique permet aux machines d'acquérir des connaissances et des compétences à partir de l'expérience.

Dans ce chapitre nous nous intéressons au domaine de l'intelligence artificielle en expliquant certains des principaux concepts clés qui le composent, en particulier l'apprentissage automatique nécessaire pour la compréhension de ce mémoire. Nous commençons par définir l'intelligence artificielle et l'apprentissage automatique. Nous présentons ensuite les étapes de l'apprentissage automatique et ses différents types, les modèles d'apprentissage automatique.

Enfin, nous terminons par une conclusion.

1.1. Définition de l'intelligence artificielle

L'intelligence artificielle (IA) (en anglais artificiel intelligence (AI)) consiste à rendre un être artificiel (une machine) intelligent à travers différentes théories et techniques, visant à permettre aux machines d'être capable de penser, raisonner et même d'apprendre comme un humain.

1.2. Le domaine des Intelligence artificielle

L'apprentissage automatique (machine Learning en anglais) qui lui-même comprend l'apprentissage profond (Deep earning en anglais), ces trois concepts sont étroitement liés. (Voir la figure 1).

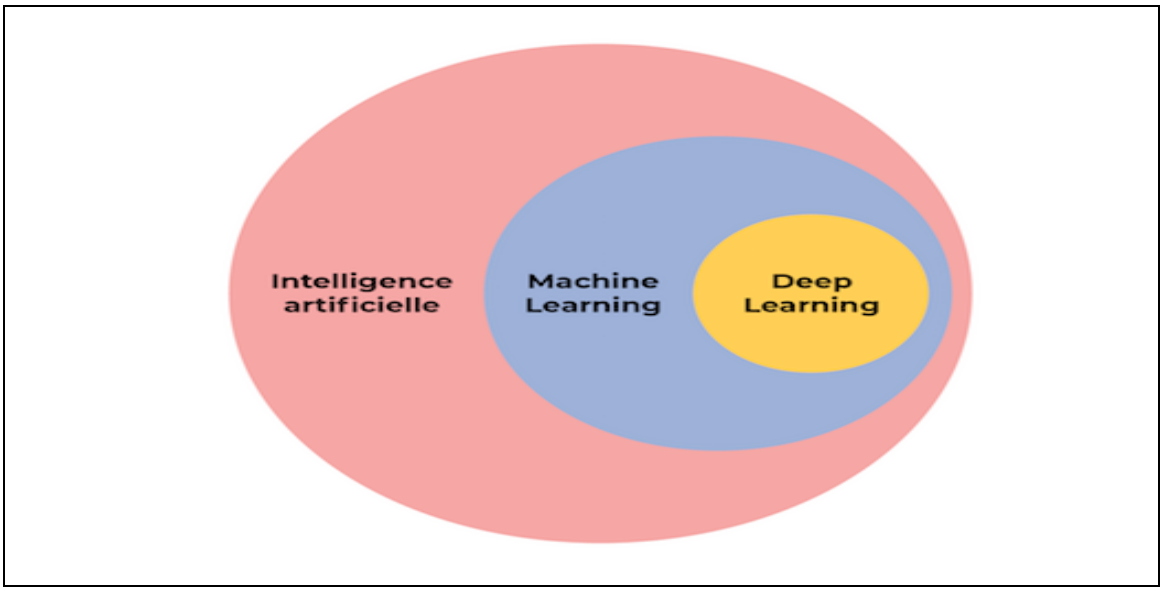

Figure 1 : Les domaines de l'intelligence artificielle [19]

1.3. Définition de l'apprentissage automatique (AA)

La première définition de L'apprentissage automatique a été introduite par l'informaticien chercheur Arthur Samual en 1958, Il l'a défini comme étant le domaine d'études qui donne aux ordinateurs la possibilité d'apprendre sans être explicitement programmé. [1]

Puis elle a été enrichie par le professeur Tom Mitchell en 1998: « Un programme informatique apprendrait de l'expérience E en ce qui concerne une tâche T et une mesure de performance P, si sa performance sur T, telle que mesurée par P, s'améliore avec l'expérience E ».[2]

On déduit alors que l'apprentissage automatique correspond au fait d'apprendre à une machine à accumuler de la connaissance à partir d'expériences, afin de résoudre au mieux un problème considéré.

1.4. Historique

Les premières études sur l'AA remontent à des travaux de statistique dans les années 1920. Mais ce n'est qu'après la seconde guerre mondiale que les premières expériences deviennent possibles.

Le premier cas d'étude était sur les réseaux neuronaux en 1943, lorsque le neurophysiologiste Warren McCulloch et le mathématicien Walter Pitts ont écrit un article sur les neurones et leur fonctionnement. Ils ont décidé de créer un modèle de cela en utilisant un circuit électrique, et donc le réseau neuronal est né.

En 1950, Alan Turing a créé le test de Turing. Pour qu'un ordinateur passe, il doit être capable de convaincre un humain qu'il est un humain et non un ordinateur.

1952 a vu le premier programme informatique qui pouvait apprendre en cours d'exécution. C'était un jeu qui jouait aux dames, créé par Arthur Samuel.

Frank Rosenblatt a conçu le premier réseau de neurones artificiels en 1958, appelé Perceptron.

1967 - L'algorithme du « plus proche voisin » est écrit, permettant aux ordinateurs de commencer à utiliser la reconnaissance de formes très basique.

1981 - Gerald Dejong introduit le concept d'apprentissage basé sur l'explication (EBL), dans lequel un ordinateur analyse les données d'entraînement et crée une règle générale qu'il peut suivre en rejetant les données sans importance.

1982 -John Hopfield a suggéré de créer un réseau qui avait des lignes bidirectionnelles, semblable à la façon dont les neurones fonctionnent réellement.

En 1986, trois chercheurs du département de psychologie de Stanford ont décidé d'étendre un algorithme créé par Widrow et Hoff en 1962. Cela a donc permis de multiples couches à utiliser dans un réseau de neurones, créant ce que l'on appelle des « apprenants lents », qui apprendront sur une longue période de temps.

La fin des années 80 et les années 90 n'ont pas apporté grand-chose sur le terrain. Depuis lors, il y a eu beaucoup plus de progrès dans le domaine, comme en 1998, lorsque la recherche aux laboratoires AT&T Bell sur la reconnaissance des chiffres a permis une bonne précision dans la détection des codes postaux manuscrits du US Postal Service. [3]

Depuis le début du 21e siècle, de nombreuses entreprises ont réalisé que l'apprentissage automatique augmentera le potentiel de calcul. C'est pourquoi ils y font des recherches plus poussées, afin de garder une longueur d'avance sur la concurrence.

Certains grands projets comprennent :

Google Brain (2012) : Il s'agissait d'un réseau neuronal profond créé par Jeff Dean de Google, qui se concentrait sur la détection de motifs dans les images et les vidéos. Il a pu utiliser les ressources de Google, ce qui le rendait incomparable à des réseaux de neurones beaucoup plus petits. Il a ensuite été utilisé pour détecter des objets dans des vidéos YouTube.[4]

OpenAI (2015) - Il s'agit d'une organisation à but non lucratif créée par Elon Musk et d'autres, pour créer une intelligence artificielle sûre qui peut bénéficier à l'humanité.[5]

Amazon Machine Learning Platform (2015) - Cela fait partie d'Amazon Web Services et démontre comment la plupart des grandes entreprises veulent s'impliquer dans la machine learning. Cette plateforme est utilisée dans divers outils de leurs systèmes internes, des services régulièrement utilisés tels que les recommandations de recherche et Alexa, à des services plus expérimentaux comme Prime Air et Amazon Go.[6]

1.5. Les étapes de l'apprentissage automatique

Afin de développer et gérer un modèle d'apprentissage automatique adapté à une mise en production, il faut passer par les étapes suivantes :

- Collecte des données
- Prétraitement des données
- Selection du modèle
- Entrainement et test du modèle
- Évaluation.

1.5.1. Collecter des données

La première étape consiste à la collecte de données, celle-ci est très importante car la qualité et la quantité des données collectées déterminent la qualité du modèle à venir.

L'ensemble de données (Data set en anglais) peut être collecté à partir de diverses sources telles qu'un fichier, une base de données, un capteur et de nombreuses autres sources similaires.

Cependant les données collectées ne peuvent pas être utilisées directement, elles peuvent avoir beaucoup de champs manquants, des valeurs extrêmement grandes ou non organisées, ou bien être bruyantes. Par conséquent, la préparation des données est effectuée.

1.5.2. Prétraitement des données

Le prétraitement des données est un processus de nettoyage des données brutes en données propres.

Voici quelques-unes des techniques de prétraitement de base:

- Conversion des données : Comme les modèles d'apprentissage automatique ne peuvent gérer que des fonctionnalités numériques, les données catégorielles et ordinales doivent donc être converties en fonctionnalités numériques.

- Ignorer les valeurs manquantes : A la détection des données manquantes, la ligne ou la colonne les contenants peuvent être supprimées selon les besoins. Cette méthode est connue pour être efficace, mais elle ne devrait pas être appliquée excessivement.

- Remplissage des valeurs manquantes : Les données manquantes dans l'ensemble de données peuvent être remplacées manuellement par la valeur moyenne, médiane ou la plus haute fréquence utilisée.

- Apprentissage automatique : La prédiction des données manquantes en utilisant l'apprentissage automatique, en effet nous pouvons prédire quelles données seront présentes à la position vide en utilisant les données existantes.

- Détection des valeurs aberrantes : Certaines données d'erreurs qui pourraient être présentes dans l'ensemble de données s'écartent considérablement des autres observations de l'ensemble de données. Par exemple : Poids humain = 800

24 kg ; en raison d'une faute de frappe sur l'extra 0. Celles-ci doivent être supprimées ou remplacées.

1.5.3. Selection du modèle d'apprentissage

Les modèles d'AA sont homogènes aux fonctions qui prédisent une sortie pour une entrée donnée. Il existe plusieurs algorithmes selon le type d'apprentissage.

1.5.4. Entrainement et test du modèle

Pour l'entrainement d'un modèle, il faut initialement diviser le modèle en 3 trois sections qui sont : Données de formation « Training Seten anglais », Données de validation « Validation Set en anglais » et Données de test « Testing Set en anglais ».

On entraîne le classificateur à l'aide d'un ensemble de données d'apprentissage, ajuste les paramètres à l'aide d'un ensemble de validation, puis teste les performances du classificateur sur un « ensemble de données de test ». Un point important à noter est que pendant la formation du classificateur, seul l'ensemble d'entrainement et / ou de validation est disponible. L'ensemble de données de test ne doit pas être utilisé pendant l'entrainement du classificateur. L'ensemble de test ne sera disponible que lors du test du classificateur.

1.5.5. L'évaluation

Elle est une partie intégrante du processus d'élaboration du modèle. Elle aide à trouver le meilleur modèle qui représente les données et à quel point le modèle choisi fonctionnera à l'avenir.

1.6. Types d'apprentissage automatique

Pour donner à la machine la capacité d'apprendre, on utilise différentes méthodes d'apprentissage automatique :

1.6.1. Apprentissage supervisé

Cette approche consiste à faire apprendre une machine à travers des exemples d'entrées qui sont labellisés avec la sortie souhaitée afin que l'algorithme puisse prédire les valeurs des sorties en fonction des entrées.

Plus formellement, étant donné un ensemble de données D, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver une fonction f en entrée x et la variable à prédire y. La fonction décrivant la relation entre x et ys'appelle un modèle de prédiction. D'une manière générale, la machine peut apprendre une relation $f: x \rightarrow y$ qui relie x à y en ayant analysé des millions d'exemples d'associations $x \rightarrow y$et l'utiliser pour catégoriser les données non triées par la suite.

On distingue deux problèmes d'apprentissage supervisé la classification et la régression que nous décrivons dans ce qui suit :

1.6.1.1. Classification

La classification consiste à prédire la valeur ou bien la classe d'une variable d'entrée discrète.

Par exemple, classification peut servir àprédire si un email est un spam ou non selon le nombre de liens présent dans l'email.

1.6.1.2. Régression

La régression consiste à prédire la valeur ou bien la classe d'une variable d'entrée continue.

Par exemple, en utilisant la régression on peut prédirele prix d'un appartement selon sa surface habitable.

1.6.2. Apprentissage non supervisé

On parle d'apprentissage non supervisé quand on ne dispose pas d'étiquettes, l'algorithme doit apprendre tout seul à trouver le point commun entre les données d'entrée et les regrouper dans des clusters ou classes. On prend comme exemple la reconnaissance d'images. L'algorithme va pouvoir regrouper des images de voiture dans un cluster et des immeubles dans un autre cluster ainsi de suite, sans la structure ou le concept de l'objet ne soit prédéfinitqui est dans notre cas une voiture et un immeuble ; il saura seul que le point en commun entre les images de voitures est la roue, et qu'un bâtiment a des forme plus droite.

1.6.3. Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une combinaison entre l'apprentissage supervisé (donnée labellisé) et l'apprentissage non supervisé (donnée non labellisé). Cette approche a pour but de prédire les labels des données non labellisé et améliorer les performances.

1.6.4. Apprentissage par renforcement

L'algorithme va apprendre à se comporter à partir d'un environnement réel ou simulateur, en interagissant avec ce dernier afin de construire son propre data set (données).

1.6.5. Apprentissage par transfert

L'apprentissage automatique permet de sauvegarder toutes les données et connaissances déjà traité ; de ce fait, on peut résoudre une tâche à partir d'une autre tâche similaire déjà résolu.

1.7. Les modèles de l'apprentissage automatique adoptés

Il existe plusieurs algorithmes d'apprentissage automatique selon la complexité du problème.

1.7.1. La régression logistique

La régression logistique (**LR**) est un algorithme d'apprentissage automatique utilisé pour résoudre des problèmes de classification binaire ou multinomiale. Contrairement à la régression linéaire qui prédit des valeurs continues, la régression logistique prédit la probabilité qu'une observation appartienne à une classe particulière.

Le principe de base de la régression logistique repose sur la fonction logistique (ou sigmoïde) qui transforme une valeur linéaire en une probabilité comprise entre 0 et 1. L'algorithme utilise des coefficients de régression pour pondérer les caractéristiques (variables indépendantes) et calcule une valeur linéaire, qui est ensuite transformée en probabilité à l'aide de la fonction logistique.

Lors de l'entraînement du modèle de régression logistique, les coefficients de régression sont estimés à l'aide de méthodes d'optimisation telles que la méthode du maximum de vraisemblance ou la descente de gradient. Une fois que les coefficients sont estimés, le modèle peut être utilisé pour prédire la probabilité d'appartenance à une classe spécifique.

La régression logistique est largement utilisée en raison de ses avantages, tels que :

1. Interprétabilité : Les coefficients de régression peuvent être interprétés pour comprendre l'impact relatif des caractéristiques sur la probabilité de classification.

2. Flexibilité : La régression logistique peut être étendue pour gérer des problèmes de classification multinomiale (plus de deux classes).

3. Robustesse : La régression logistique est relativement robuste aux données manquantes et bruitées.

4. Efficacité : L'algorithme de régression logistique est relativement rapide et peut être utilisé pour de grands dataset.

Il existe également des variantes de la régression logistique, telles que la régression logistique pénalisée (L1 ou L2) pour la sélection automatique des caractéristiques et pour la régularisation afin de prévenir le surajustement. [7]

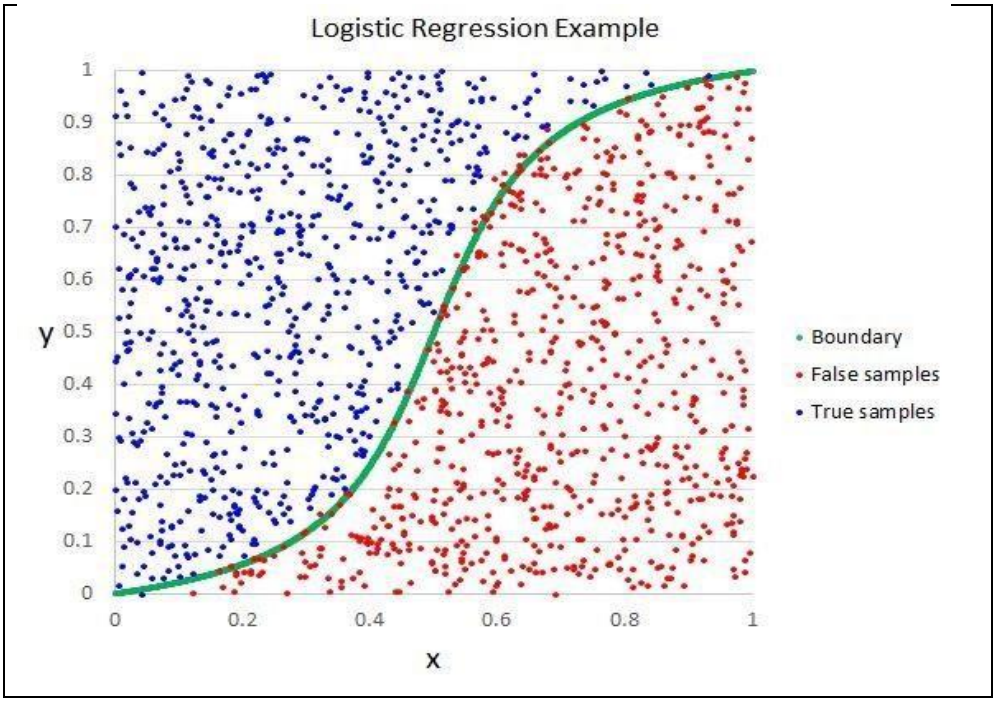

Figure 2 : Schéma du fonctionnement de l'algorithme RL [20]

Les avantages de la régression logistique incluent sa simplicité, son interprétabilité et sa rapidité d'entraînement et de prédiction.

1.7.2. Foret Aléatoire

RF (Random Forest) est un algorithme d'apprentissage automatique utilisé pour la classification et la régression. Il appartient à la famille des méthodes d'ensemble, qui combinent les prédictions de plusieurs modèles individuels pour améliorer la performance globale du modèle.

Le fonctionnement de l'algorithme Random Forest repose sur la construction d'un ensemble d'arbres de décision. Chaque arbre de décision est construit en utilisant un sous-ensemble aléatoire des exemples d'entraînement et un sous-ensemble aléatoire des caractéristiques. Ces arbres sont construits de manière indépendante et chacun effectue des prédictions individuelles. [8]

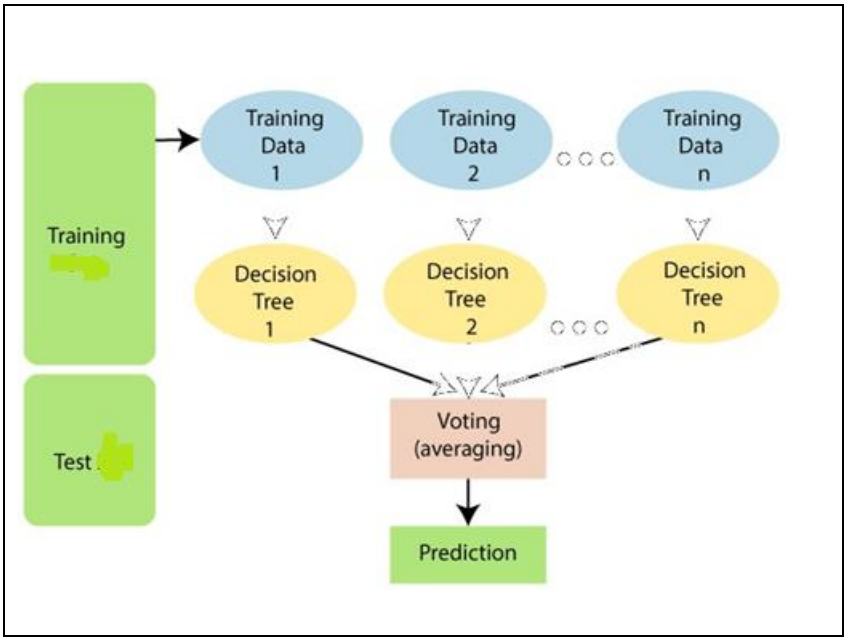

Figure 3 : Schéma du fonctionnement de l'algorithme FA [21]

Lorsqu'il s'agit de classer un nouvel exemple, chaque arbre de décision dans le Random Forest émet une prédiction, et la classe prédite est déterminée par un vote majoritaire parmi tous les arbres. Pour la régression, la prédiction finale est généralement calculée en prenant la moyenne des prédictions de tous les arbres.

Random Forest possède plusieurs avantages. Il est efficace pour les dataset de grande dimension, il peut gérer des données manquantes ou bruitées, et il fournit une estimation de l'importance des caractéristiques utilisées dans la classification ou la régression. De plus, il est moins susceptible au surajustement par rapport à un arbre de décision unique.

1.7.3. Les réseaux de neurones

Les réseaux de neurones (**ANN**) sont des systèmes de traitement de l'information dont la structure s'inspire de celle du système nerveux. [9] C'est des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Ainsi toute structure hiérarchique de réseaux est évidemment un réseau.

Fondamentalement, chaque neurone d'un réseau de neurones n'est qu'une fonction mathématique. Chaque neurone calcule une somme pondérée de ses entrées (plus le poids d'une entrée est important, plus cette entrée affecte la sortie du neurone). Cette somme pondérée est ensuite injectée dans une fonction non linéaire appelée fonction d'activation (une étape qui permet aux réseaux de neurones de modéliser des phénomènes non linéaires complexes).

Comportement

Un réseau neuronal typique comprend quelques dizaines voire des millions de neurones artificiels appelés unités disposées en une série de couches, chacune se connectant aux couches de chaque côté.

Certains d'entre eux, appelés unités d'entrée (Input layers en anglais), sont conçus pour recevoir diverses formes d'informations du monde extérieur que le réseau tentera de découvrir, de reconnaître ou de traiter d'une autre manière.

D'autres unités sont installées de l'autre côté du réseau et signalent comment il réagit aux informations qu'il a apprises ; ce sont les unités de sortie (Output layers an anglais). Entre les unités d'entrées et les unités de sortie se trouvent une ou plusieurs couches d'unités cachées (HIDDEN LAYERS), qui, ensemble, forment la majorité du cerveau artificiel. Automatique La plupart des réseaux de neurones sont entièrement connectés, ce qui signifie que chaque unité cachée et chaque unité de sortie est connectée à chaque unité des couches de chaque côté.

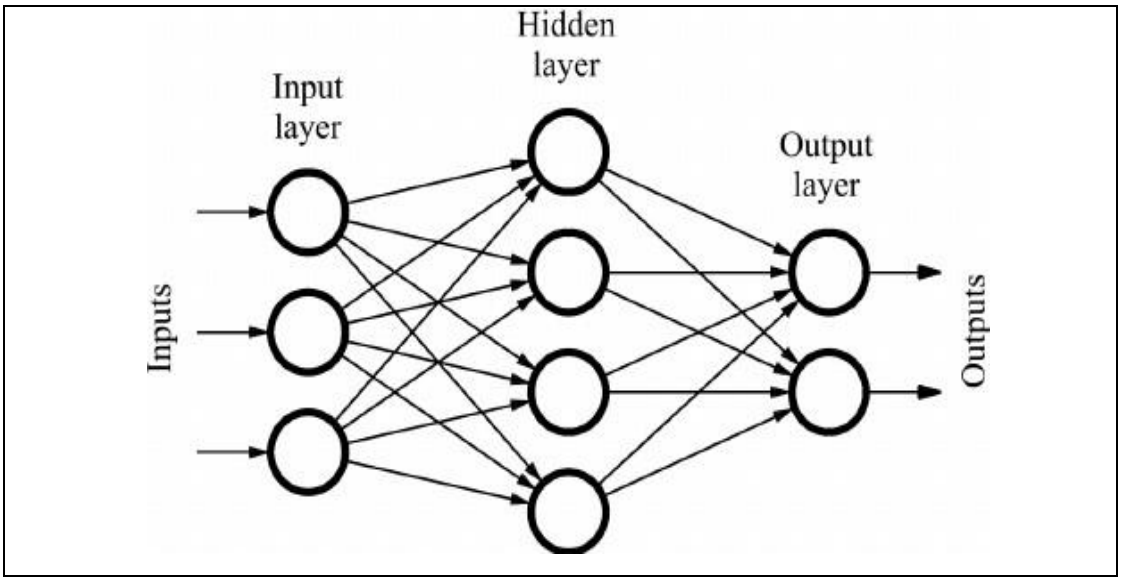

Figure 4 : Schéma du fonctionnement de l'algorithme ANN [22]

Les connexions entre une unité et une autre est représenté par un nombre appelé poids, qui peut être positif (si une unité en active une autre) ou négatif (si une unité en supprime ou en inhibe une autre). Plus le poids est élevé, plus une unité a d'influence sur une autre. (Cela correspond à la façon dont les cellules cérébrales se déclenchent à travers de minuscules lacunes appelées synapses.

REMARQUE :

Cependant, pour les dataset déséquilibrés, Logistic Regression, Random Forest, Les réseaux de neurones **peuvent encore être influencé par la classe majoritaire**.

1.8. Les applications de l'apprentissage automatique

Les cas d'usages de l'AA sont nombreux, afin d'illustrer l'utilité de l'AA nous citons :

- La Fouilles de données (data mining en anglais) : utilisé pour l'extraction d'informations. Elle permet d'utiliser toutes les données qui se trouve des bases de données (BDD) hétérogènes, afin de générer des programmes pour l'apprentissage. Par exemple : trouver une prescription pour un patient à travers des fichiers médicaux antérieurs.

- La robotique : Les robots sont maintenant omniprésents, surtout dans les usines. Ils aident, par exemple, dans la production de masse à automatiser des étapes de travail cohérente

- Transcription automatique de la parole (Chat bot) : désigne un agent automatisé avec lequel il est possible d'interagir par texte via les principales plateformes de messagerie telles que Facebook, Messenger ou Skype.

- La reconnaissance d'objets ou de personnes dans des images, reconnaissance du langage parlé. Exemple : Environnement de développement biométrique de reconnaissance de visages et d'empreintes digitales.

1.9. Conclusion

Nous avons présenté les grandes lignes de l'apprentissage automatique. Nous avons exploré son fonctionnement ainsi que les algorithmes utilisés. Enfin, nous avons exposé d'une part quelques limitations rencontrées en utilisant ces algorithmes et les différents domaines d'applications de l'AA.

Chapiter2: Dataset non équilibré (déséquilibré)

• Introduction

Le déséquilibre des datasets peut avoir un impact significatif sur les performances des modèles d'apprentissage automatique. Les algorithmes traditionnels, lorsqu'ils sont entraînés sur des datasets déséquilibrés, ont tendance à favoriser les classes majoritaires et à ignorer les classes minoritaires. Cela entraîne une mauvaise généralisation et une performance médiocre pour la prédiction des classes sousreprésentées.

Dans ce chapitre, nous allons explorer les défis associés aux datasets non équilibrés et les stratégies couramment utilisées pour y remédier, et les conséquences de ce déséquilibre sur les performances des modèles et les techniques permettant de les atténuer. En comprenant les problématiques spécifiques liées aux datasets non équilibrés, nous pourrons développer des approches plus efficaces pour résoudre ces problèmes et obtenir des résultats plus précis et équilibrés.

2.1. Définition:

Dans un problème de classification, il arrive souvent d'avoir des datasets très déséquilibrés. On parle d'un **dataset déséquilibré** lorsque le ratio des observations d'une classe par rapport à l'ensemble des observations est très faible.

Cette notion de déséquilibre de classes est relativement fréquente dans plusieurs secteurs comme le secteur médical ou le secteur bancaire et elle est problématique lorsqu'elle n'est pas traitée.

Un déséquilibre signifie que le nombre de points de données disponibles pour différentes classes est différent :

S'il y a deux classes, alors des données équilibrées signifieraient 50 % de points pour chacune des classes. Pour la plupart des techniques d'apprentissage automatique, peu de déséquilibre n'est pas un problème. Ainsi, s'il y a 60% de points pour une classe et 40% pour l'autre classe, cela ne devrait pas entraîner de dégradation significative des performances. Ce n'est que lorsque le déséquilibre de classe est élevé, comme

70% de points pour une classe et 30 % pour l'autre ou plus, que les critères d'optimisation standard ou les mesures de performance peuvent ne pas être aussi efficaces et nécessiter des modifications. [10]

Pour les données que nous utiliserons dans notre étude, il contient toutes les conditions mentionnées ci-dessus, ce qui, entré dans la Figure (5) :

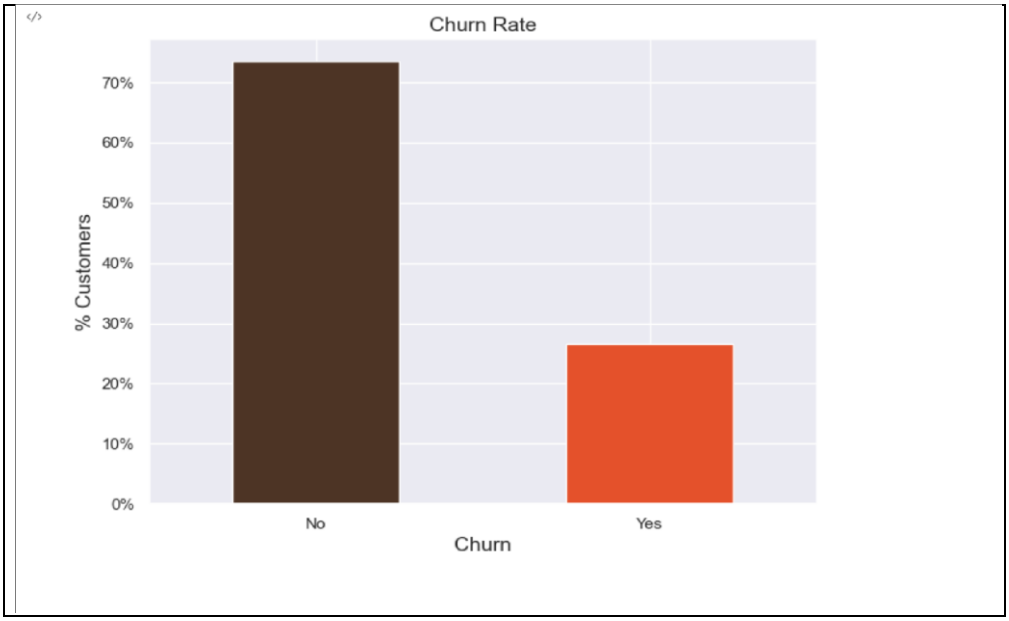

Figure 5 : Schéma création présente Taux de désabonnement

Des clients des télécommunications

La classe (NO) : 5174 avec un pourcentage 73,42 %

La classe (YES) : 1869 avec un pourcentage 26,58 %

2.2. Qu'est-ce que les données déséquilibrées ?

Lorsque la classification des données n'est pas égale, on peut parler de données déséquilibrées. Il s'agit d'une tâche de classification qui entraîne divers problèmes dans la sortie du modèle.

Le problème du déséquilibre des classes est très courant, qu'il s'agisse de jeux de données de test réels ou de la compétition Kaggle. Les problèmes de classification du monde réel comportent un certain niveau de déséquilibre de classification. Cela se produit généralement lorsqu'il n'y a pas d'instances de données appropriées qui correspondent à une classe. Par conséquent, il est essentiel de choisir la bonne métrique d'évaluation du modèle. Si le modèle comporte un ensemble de données déséquilibré, votre résultat sera inutile. En revanche, si vous résolvez un problème de la vie réelle avec ce modèle, le résultat sera un gaspillage.

Dans diverses situations, le déséquilibre des classes se produira toujours. Un bon exemple est lorsque vous considérez l'ensemble de données des transactions frauduleuses et non frauduleuses. Vous trouverez moins de transactions frauduleuses que de transactions non frauduleuses. C'est là que vous trouverez des problèmes.

Le problème des dataset déséquilibrés est très courant et il est inévitable que cela se produise. Ce problème se pose lorsqu'une classe domine sur une autre classe. Cela rend le modèle d'apprentissage automatique plus biaisé en faveur de la classe majoritaire. Elle provoque une mauvaise classification des classes minoritaires. Il s'agit d'un problème très courant en apprentissage automatique où nous avons des dataset avec un rapport disproportionné d'observations dans chaque classe.

Maintenant que nous connaissons l'impact des dataset déséquilibrés, c'est un soulagement de savoir qu'il existe des méthodes pour corriger ledit déséquilibre. [11]

2.3. Les problèmes liés aux les datasets déséquilibrées

Les données déséquilibrées posent plusieurs problèmes en apprentissage automatique :

1. **Biais de modèle** : Lorsque les dataset sont déséquilibrés, les modèles d'apprentissage automatique ont tendance à être biaisés en faveur de la classe majoritaire. Cela signifie que le modèle aura du mal à bien généraliser et à prédire avec précision les exemples de la classe minoritaire.

2. **Faible performance pour la classe minoritaire** : Étant donné que les données de la classe minoritaire sont sous-représentées, les modèles peuvent avoir des difficultés à apprendre les motifs spécifiques de cette classe. Par conséquent, la performance de prédiction pour la classe minoritaire est souvent médiocre, ce qui peut être préjudiciable dans des domaines où la détection des exemples de cette classe est cruciale.

3. **Prédiction biaisée** : Les modèles entraînés sur des dataset déséquilibrés ont tendance à prédire davantage la classe majoritaire, car cela minimise l'erreur globale. Cela peut conduire à des prédictions biaisées et à des décisions injustes, en ne prenant pas suffisamment en compte les exemples de la classe minoritaire.

4. **Évaluation trompeuse** : Lorsque les dataset sont déséquilibrés, les mesures de performance traditionnelles peuvent être trompeuses. Par exemple, un modèle peut avoir un taux de précision élevé simplement en prédisant constamment la classe majoritaire. Cela peut donner une impression faussement positive de la performance du modèle, en masquant sa mauvaise performance pour la classe minoritaire.

5. **Surapprentissage (Overfitting)** : Lorsque les données de la classe minoritaire sont rares, les modèles peuvent surapprendre les exemples de cette classe, en les mémorisant plutôt qu'en généralisant les motifs sous-jacents. Cela peut entraîner une performance médiocre lors de la prédiction de nouveaux exemples de la classe minoritaire. [12]

Il est donc crucial de traiter les problèmes posés par les dataset déséquilibrés afin de garantir des modèles d'apprentissage automatique équitables, précis et capables de bien représenter toutes les classes présenter dans les données

2.4. Méthodes d'équilibrage

La correction des déséquilibres a généré une littérature abondante depuis le début des années 2000, et de nombreuses méthodes sont implémentées et disponibles, sur Python comme sur R. Ces méthodes peuvent agir à 3 niveaux :

- au niveau des données (data-level solutions)

- niveau des algorithmes (algorithm-level solutions)

- ou au niveau de l'erreur de classification.

Dans notre cas, nous nous concentrerons **au niveau des données (**Data-level solutions)

Ces solutions consistent à opérer un rééchantillonnage des données utilisées pour l'entraînement des algorithmes de Machine Learning. Ceci vise un rééquilibrage des classes, pour faciliter l'apprentissage. Trois méthodes principales existent :

2.4.1. Le sous-échantillonnage aléatoire (random undersampling, RUS)

La technologie d'équilibrage RUS (Random Under-Sampling) est une technique utilisée dans le domaine de l'apprentissage automatique pour résoudre le problème de déséquilibre de classes dans un ensemble de données. Le déséquilibre de classes se produit lorsque les classes d'intérêt dans les données d'apprentissage sont représentées de manière disproportionnée, c'est-à-dire qu'une classe est beaucoup plus fréquente que l'autre.

L'équilibrage RUS est une approche qui consiste à réduire aléatoirement la taille de la classe majoritaire (la classe qui est plus fréquente) pour qu'elle corresponde à la taille de la classe minoritaire (la classe qui est moins fréquente). Cela permet de créer un nouvel ensemble de données équilibré où les deux classes sont représentées de manière similaire. [13]

Voici comment fonctionne l'équilibrage RUS :

1. Identifiez la classe majoritaire et la classe minoritaire dans votre ensemble de données.

2. Déterminez la taille de la classe minoritaire. C'est la taille que vous souhaitez utiliser pour l'échantillonnage aléatoire.

3. Sélectionnez aléatoirement un sous-ensemble d'échantillons de la classe majoritaire avec la taille de la classe minoritaire.

4. Combiner les échantillons sélectionnés de la classe majoritaire avec l'ensemble de la classe minoritaire pour créer un nouvel ensemble de données équilibré.

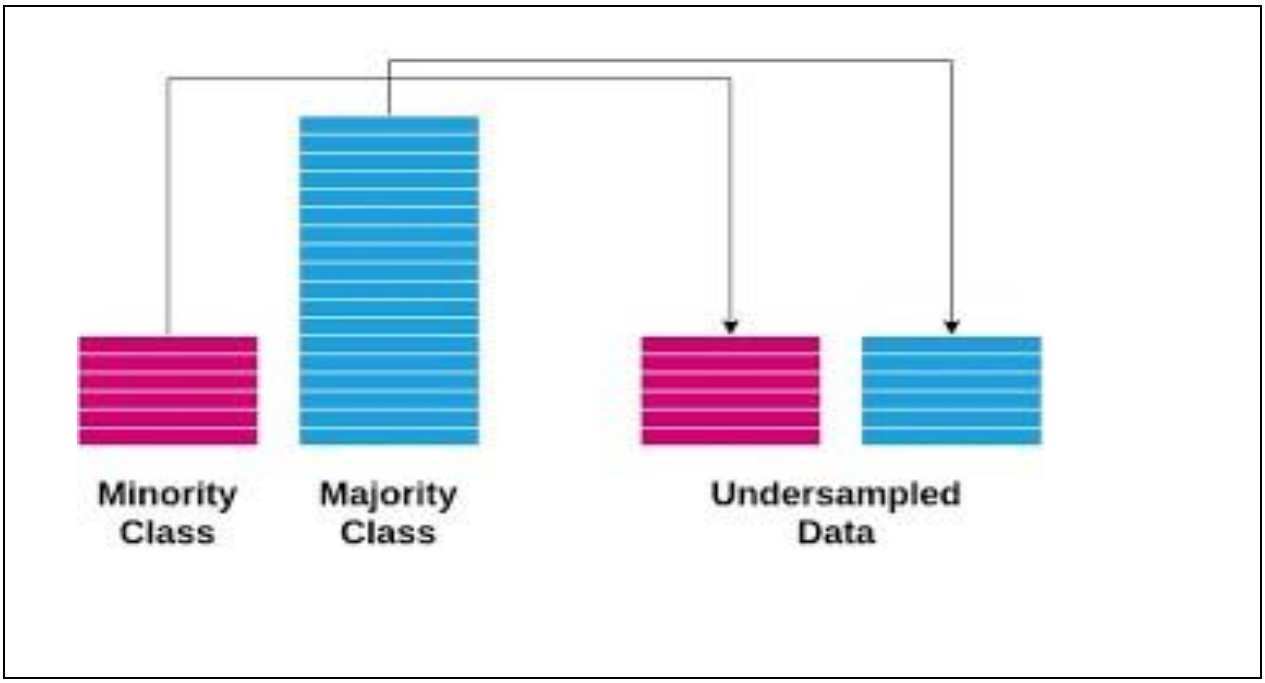

Figure 6. Schéma création d'une donnée synthétique dans RUS [23]

L'équilibrage RUS est généralement utilisé lorsque le déséquilibre de classes est significatif et que vous disposez d'un grand nombre d'échantillons dans la classe majoritaire. Cela peut aider à améliorer les performances du modèle en réduisant le biais introduit par le déséquilibre.

Il convient de noter que l'équilibrage RUS peut entraîner une perte d'informations car il élimine aléatoirement des échantillons de la classe majoritaire. Il est important de prendre en compte cette perte d'informations potentielle lors de l'utilisation de cette technique.

2.4.2. Le sur-échantillonnage aléatoire (random oversampling, ROS)

La technologie d'équilibrage ROS (Random Over-Sampling) est une technique utilisée dans le domaine de l'apprentissage automatique pour traiter le problème de déséquilibre de classes dans un ensemble de données. Le déséquilibre de classes se produit lorsqu'une classe est sous-représentée par rapport à une autre dans les données d'apprentissage.

L'équilibrage ROS vise à résoudre ce problème en augmentant aléatoirement la taille de la classe minoritaire (la classe qui est moins fréquente) pour équilibrer le nombre d'échantillons entre les classes. [14]

Voici comment fonctionne l'équilibrage ROS :

1- Identifiez la classe majoritaire et la classe minoritaire dans votre ensemble de données.

2- Déterminez la taille de la classe majoritaire. C'est la taille que vous souhaitez atteindre pour équilibrer les classes.

3- Sélectionnez aléatoirement des échantillons de la classe minoritaire pour atteindre la taille de la classe majoritaire. Vous pouvez utiliser des techniques telles que le tirage aléatoire avec remplacement pour sélectionner les échantillons.

4- Répétez l'étape précédente jusqu'à ce que la taille de la classe minoritaire atteigne la taille de la classe majoritaire.

5- Combiner les échantillons de la classe minoritaire sélectionnés avec l'ensemble de la classe majoritaire pour créer un nouvel ensemble de données équilibré.

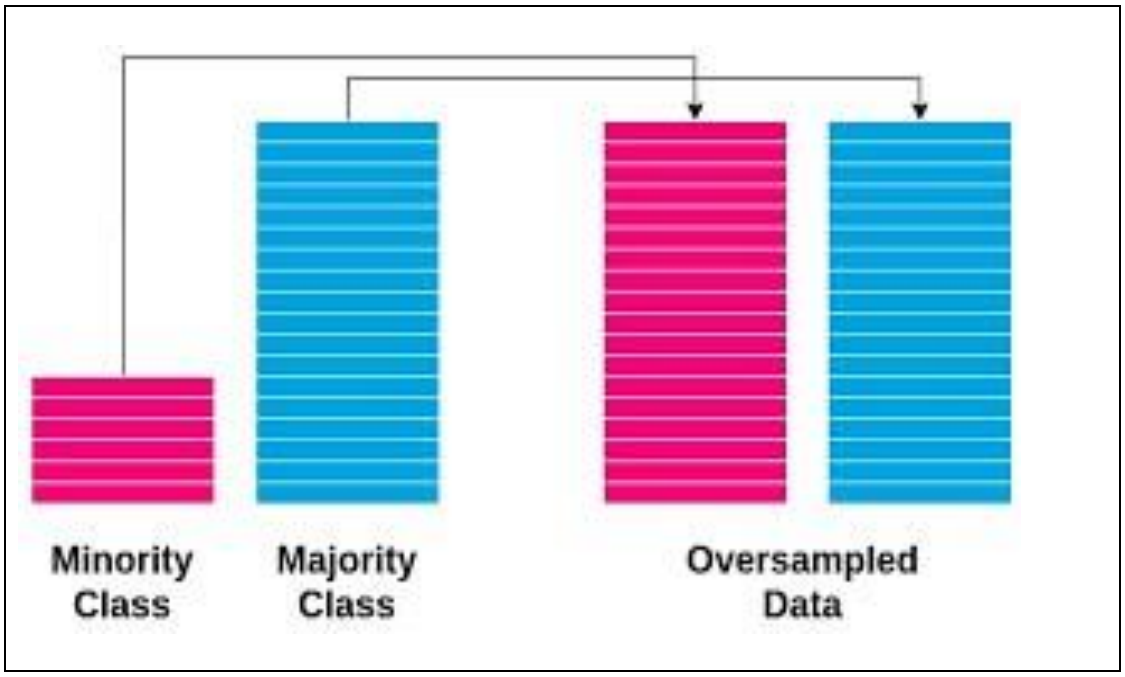

Figure 7 : Schéma création d'une donnée synthétique dans ROS [24]

L'équilibrage ROS peut aider à améliorer les performances du modèle en fournissant plus d'exemples de la classe minoritaire, ce qui permet au modèle d'apprendre des motifs plus précis pour cette classe.

2.4.3. Le sur-échantillonnage synthétique (ROS pour Synthetic Minority Oversampling Technique)

SMOTE est une technique que vous pouvez utiliser pour le suréchantillonnage des données. Cette technique crée de nouveaux exemples synthétiques au lieu de suréchantillonner par remplacement. SMOTE introduit des exemples synthétiques dans les segments de ligne pour suréchantillonner les échantillons de la classe minoritaire. Il joint tous les k classes minoritaires qui sont proches des voisins. Le choix des voisins parmi les k voisins les plus proches est aléatoire. Le nombre dépend de la quantité de suréchantillonnage dont le modèle a besoin.

La fonction principale de SMOTE est de construire des classes minoritaires. Il existe un algorithme simple pour créer ces classes.

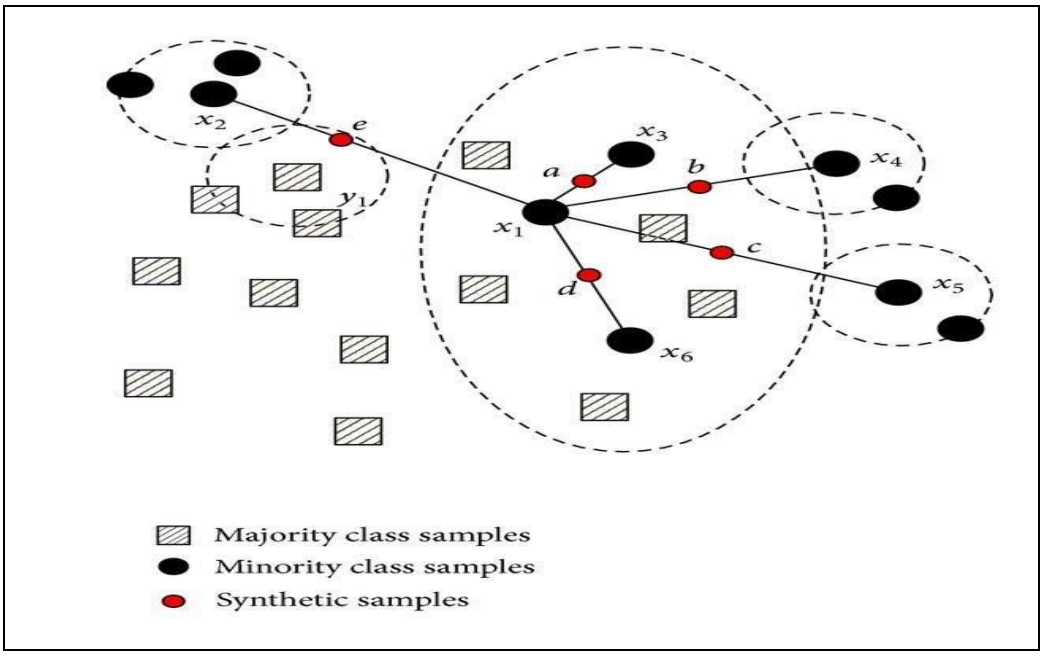

Figure 8 : Schéma création d'une donnée synthétique dans SMOTE [25]

Comme vous le savez peut-être, le développement d'instances répétitives ou le suréchantillonnage peuvent entraîner un sur ajustement. De plus, la frontière de décision devient encore plus étroite. Vous pouvez résoudre ce problème en générant des échantillons similaires plutôt que de les répéter tous. SMOTE génère des échantillons nouvellement construits qui ont des caractéristiques différentes des échantillons précédents. Par conséquent, la limite de décision devient plus souple. Cela aidera l'algorithme à estimer l'hypothèse exacte. Vous trouverez ci-dessous quelques avantages de SMOTE :

– L'information ne sera pas perdue.

– Cette technique est simple, et vous pouvez facilement l'interpréter et l'implémenter dans le modèle.

– Elle améliore l'overfitting en tant qu'exemples synthétiques. Cela aidera à générer de nouvelles instances au lieu de les répliquer.

Ainsi, SMOTE génère de nouvelles observations par interpolation entre les observations existantes dans l'ensemble de données. Cette technique améliore l'approche où les exemples dans une classe minoritaire sont dupliqués pour créer un équilibre. SMOTE synthétise de nouveaux exemples plutôt que de dupliquer des exemples.

SMOTE sélectionne des exemples qui se trouvent à proximité dans un espace de fonctionnalités. Il prend ensuite un nouvel exemple à un point le long du segment, joignant les exemples adjacents. Pour élaborer davantage, il trouve les k-plus proches voisins (k-NN) dans la classe minoritaire. Le k-NN fait référence à une approche de classification où la probabilité qu'un point de données appartienne à un groupe ou à un autre en fonction des points de données les plus proches du point de données.

SMOTE choisit une instance de la classe minoritaire au hasard et calcule son k-NN. Un voisin est alors choisi au hasard.

Après cela, un exemple synthétique est créé à un point choisi au hasard entre les deux exemples. Ce processus peut générer autant d'exemples synthétiques nécessaires à une classe minoritaire pour créer un équilibre.

Le pseudo algorithme de SMOTE est donnée ci-dessous :

1. Sélectionnez les données à suréchantillonner (données générales avec étiquettes de classe minoritaire).

2. Choisissez une instance des données.

3. Trouvez ses k voisins les plus proches de ce point de données.

4. Choisissez un point de données aléatoire qui se trouve dans k voisins les plus proches du point de données sélectionné et créez un point de données synthétique n'importe où sur la ligne joignant ces deux points.

5. Répétez le processus jusqu'à ce que les données soient équilibrées.

```
Input: Nombre d'individus de la classe minoritaire T,taux de SMOTE N(<1 ou entier non-nul)
       Nombre de plus proches voisins k.
Output: N·T individus synthétiques de la classe minoritaire.
Si N < 100%, on choisit un échantillon aléatoire de N·T individus, dont chacun servira à générer un individu
synthétique ;
if N < 1 then
  Randomiser le vecteur des T individus;
  T = N.T;
  N = 1 ;
End
Nvariables = Nombre de variables;
ORIGINAUX (T × Nvariables) : matrice des individus originaux de la classe minoritaire; SYNTHETIQUES (N · T ×
Nvariables) : matrice des individus synthétiques générés ; NouvIndex ← 1 : variable de comptage des individus
synthétiques générés;
for i ← 1 to T do Calculer les k plus proches voisins de l'individu i, dont les index correspondants(dans la matrice
ORIGINAUX) sont stockés dans un vecteur V oisins(k × 1);
 for j ← 1 to N do
  Choisir aléatoirement un nombre nn entre 1 et k, qui indexe l'un des k voisins dans le vecteur Voisins ;
  for var ← 1 to Nvariables do
    Distance = ORIGINAUX[V oisins[j], var] − ORIGINAUX[i, var] ;
     Ecart = Nombre aléatoire entre 0 et 1;
     SYNTHETIQUES [NouvIndex, var] = ORIGINAUX[i, var] + Ecart · Distance ;
  end
   NouvIndex = NouvIndex + 1;
 end
end
```

Conclusion

En conclusion, le problème des données déséquilibrées est une problématique courante et importante dans le domaine de l'apprentissage automatique et de l'analyse de données. Ce problème a un impact négatif sur les performances des modèles qui se basent sur ces données, en privilégiant souvent la classe majoritaire au détriment des classes minoritaires. Cela entraîne un biais de classification et une diminution de la précision globale du modèle.

Trois solutions courantes ont été discutées dans ce chapitre pour traiter le problème des données déséquilibrées : Rus, Ros et SMOTE. Ces techniques offrent des méthodes efficaces pour corriger le déséquilibre des données en ajustant la taille des classes minoritaires ou en générant des données synthétiques.

Chapitre 3: Expérimentations et résultats

• Introduction

Dans ce chapitre, les résultats de différents tests établis en utilisant les techniques présentées dans les chapitres précédents sont présentés.

3.1. Jeu de données (Datasets)

Taux de désabonnement des clients des télécommunications :

Programmes ciblés de fidélisation de la clientèle

Le taux de désabonnement, également appelé taux d'attrition ou taux de désabonnement des clients, est le taux auquel les clients cessent de faire affaire avec une entité. Il est le plus souvent exprimé en pourcentage d'abonnés au service qui interrompent leur abonnement au cours d'une période donnée [16]

Contexte :

"Prédire le comportement pour fidéliser les clients. Notre objectif est d'analyser toutes les données clients pertinents et de développer des programmes de fidélisation ciblés."

Contenu :

L'ensemble de données comprend des informations sures :

- Clients qui sont partis au cours du dernier mois - la colonne s'appelle Churn

- Services auxquels chaque client s'est inscrit - téléphone, lignes multiples, Internet, sécurité en ligne, sauvegarde en ligne, protection de l'appareil, support technique et streaming TV et films

- Informations sur le compte client

- depuis combien de temps il est client, contrat, mode de paiement, facturation sans papier, frais mensuels et frais totaux.

- Informations démographiques sur les clients

- sexe, tranche d'âge et s'ils ont des partenaires et des personnes à charge

https://www.kaggle.com/code/nirajpoudel/customer-churn-prediction-for-atelephone-company/input

3.2. Protocol Expérimental

3.2.1 Environnement de développement

Les algorithmes sont implémentés en utilisant Python 3.10.10

• **Python** est un langage de programmation le plus utilisé dans le domaine du Machine Learning, du Big Data et de la Data Science. Créé en 1991, le langage de programmation

Python apparu à l'époque comme une façon d'automatiser les éléments les plus ennuyeux de l'écriture de scripts ou de réaliser rapidement des prototypes d'applications.

[17]

Depuis quelques années, toutefois, ce langage de programmation s'est hissé parmi les plus utilisés dans le domaine du développement de Environnement de développements, de gestion d'infrastructure et d'analyse de données. Il s'agit d'un élément moteur de l'explosion du Big Data.

Python est un langage de programmation open source créé par le programmeur **Guido van Rossum en 1991**.

Il tire son nom de l'émission Monty Python's Flying Circus. Le langage Python doit sa popularité à **plusieurs avantages** qui profitent aussi bien aux débutants qu'aux experts. Tout d'abord, il est **facile à apprendre et à utiliser**. Ses caractéristiques sont peu nombreuses, ce qui

permet de créer des programmes rapidement et avec peu d'efforts. De plus, sa syntaxe est conçue pour être lisible et directe.

Un autre avantage du Python est sa popularité. Ce langage **fonctionne sur tous les principaux systèmes** d'exploitation et plateformes informatiques. De plus, même s'il ne s'agit clairement pas du langage le plus rapide, il compense sa lenteur par sa versatilité.

Enfin, même s'il est principalement utilisé pour le scripting et l'automatisation, ce langage est aussi utilisé pour créer des Environnement de développements de qualité professionnelle. Qu'il s'agisse d'applications ou de services Web, le Python est utilisé par un grand nombre de développeurs pour créer des Environnement de développements.

Nous utiliserons Python sur une plateforme Visual Studio Code

•**Visual Studio Code (VS Code)** est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un support intégré pour JavaScript, TypeScript et Node.js et dispose d'un riche écosystème de plugins pour d'autres langages (tels que C++, C#, Java, Python, PHP et Go) et des runtimes (tels que. NET et Unity)

Nous utiliserons plusieur outils, notamment Jupyter NoteBook

•**Jupyter** est une application Web open source qui permet la création et le partage de documents contenant du code en direct, des équations mathématiques, des illustrations et des textes explicatifs. Les composants précédents sont utilisés dans un but précis dans les opérations d'analyse de données, par exemple, les opérations statistiques, l'apprentissage automatique et même les exercices de programmation de base, entre autres.

3.2.2 Bibliothèques Environnement de développementles

1. **Pandas Pandas**

Est un outil d'analyse et de manipulation de données open source rapide, puissant, flexible et facile à utiliser, construit sur le langage de programmation Python.

2. **Numpy NumPy**

A été créé en 2005 par Travis Oliphant. C'est un projet open source et vous pouvez l'utiliser librement. NumPy est une bibliothèque python utilisée pour travailler avec des tableaux. Il a également des fonctions pour travailler dans le domaine de l'algèbre linéaire, de la transformée de Fourier et des matrices.

3. **Matplotlib :**

Matplotlib est une bibliothèque de langage de programmation Python utilisée pour tracer des graphiques et des graphiques 2D et 3D. Matplotlib fournit une API puissante qui vous permet de créer facilement divers tracés et graphiques, tels que des lignes, des perpendiculaires, des cercles, des formes géométriques et des tracés statistiques. Les hhbpdéveloppeurs et les chercheurs de différents domaines l'utilisent pour visualiser et analyser des données.

4. Tensorflow TensorFlow

Est une plateforme de bout en bout qui facilite la création et le déploiement de modèles de machine learning. Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie.

5. Keras Keras : Est une API de haut niveau de TensorFlow permettant de créer et d'entraîner des modèles de Deep learning. Elle est utilisée dans le cadre du prototypage rapide, de la recherche de pointe et du passage en production. Elle présente trois avantages majeurs : convivialité, Modularité et facilité de composition, Facilité d'extension [27]

3.2.3 Configuration matérielle utilisée

La configuration du matériel utilisé dans notre implémentation est comme suivie :

- Un PC portable ThinkPad T460p ;
- Intel(R) Core (TM) i5-6440HQ CPU @ 2.60GHz 2.60 GHz;
- Carte graphique Intel UHD Graphics ;
- RAM de taille 8 GO SSD ;

3.2.4. Prétraitement

Afin de nettoyer les datasets pour ne laisser que les informations jugées importantes, nous utilisons les techniques suivantes :

Analyse Exploratoire les datasets

✓ importer des bibliothèques

✓ Lisons le fichier de données dans le python notebook (librairie Pandas)

✓ Vérification des types de données de toutes les colonnes

✓Supprimer les ID client de l'ensemble de données (librairie Pandas)

✓ Explorons les données pour voir s'il y a des valeurs manquantes

- Après avoir examiné la sortie ci-dessus, nous pouvons dire qu'il y a 11 valeurs manquantes pour les charges totales. Remplaçons supprimons ces 11 lignes de notre ensemble de données
- Suppression des valeurs manquantes (librairie Pandas)

✓ Conversion des frais totaux en un type de données numérique (librairie Pandas)

✓ Convertir la variable prédictive en une variable numérique binaire

✓ Certaines des colonnes n'ont pas de service Internet ou pas de service téléphonique, qui peuvent être remplacés par un simple Non

✓ Convertir Oui et Non en 1 ou 0

✓ Convertir Male et Female en 1 ou 0

✓ One hot encoding pour les colonnes catégorielles

Visualisation de datasets

- Regardons d'abord le taux de désabonnement dans nos données

- Représentation graphique de la colonne "Monthly Charges" décalage 'Churn=Yes','Churn=No'

- Représentation graphique de la colonne "tenure" décalage 'Churn=Yes','Churn=No'

- Représentation graphique de la colonne "contract" décalage 'Churn=Yes','Churn=No'

Développement des modèles prédictifs

La régression logistique (Logistic Regression)

✓ Train-Test Split : commence par diviser les données en ensembles d'entraînement et de test à l'aide de la fonction train_test_split () de la bibliothèque sklearn. Model_selection.

✓ Cross-validation avec Logistic Regression :

Ensuite, une validation croisée à 5 plis est effectuée sur un modèle de classification de la forêt aléatoire. La fonction cross_val_score () de sklearn. Model_selection est utilisée pour effectuer la validation croisée. La moyenne et l'écart type des scores obtenus sont imprimés à l'écran.

✓ Performance du meilleur modèle de régression logistique :

En utilisant la validation croisée, le meilleur modèle de régression logistique est sélectionné à l'aide de la classe GridSearchCV de sklearn. Model_selection.

✓ Entraînement du modèle de régression logistique :

Un modèle de régression logistique est initialisé avec les hyperparamètres spécifiés. Le modèle est entraîné sur l'ensemble d'entraînement à l'aide de la méthode fit ().

✓ Standardisation des données :

Les fonctionnalités d'entraînement et de test sont standardisées à l'aide de la classe StandardScaler de sklearn. preprocessing. Cela est fait pour mettre les variables à la même échelle afin d'améliorer les performances du modèle.

✓ Prédiction et évaluation du modèle :

Le modèle formé est utilisé pour prédire les classes de l'ensemble de test à l'aide de la méthode predict(). Les prédictions sont ensuite évaluées à l'aide de différentes mesures de performance telles que la précision, le score F1, etc. La matrice de confusion est également calculée et affichée sous forme de heatmap.

✓ Courbe ROC :

Une courbe ROC (Receiver Operating Characteristic) est tracé pour évaluer les performances du modèle. Les scores de probabilité de la classe positive sont utilisés pour tracer la courbe ROC à l'aide de la fonction roc_curve() de sklearn.metrics. La zone sous la courbe (AUC) est également calculée à l'aide de la fonction roc_auc_score().

Forêt aléatoire (Random Forest)

✓ Train-Test Split : commence par diviser les données en ensembles d'entraînement et de test à l'aide de la fonction train_test_split() de la bibliothèque sklearn.model_selection.

✓ Cross-validation avec Random Forest :

Ensuite, une validation croisée à 5 plis est effectuée sur un modèle de classification de la forêt aléatoire. La fonction cross_val_score() de sklearn.model_selection est utilisée pour effectuer la validation croisée. La moyenne et l'écart type des scores obtenus sont imprimés à l'écran.

✓ Performance du meilleur modèle de forêt aléatoire :

En utilisant la validation croisée, le meilleur modèle de forêt aléatoire est sélectionné à l'aide de la classe GridSearchCV de sklearn.model_selection.

✓ Entraînement du modèle de forêt aléatoire :

Un modèle de forêt aléatoire est initialisé avec les hyperparamètres spécifiés. Le modèle est entraîné sur l'ensemble d'entraînement à l'aide de la méthode fit().

✓ Standardisation des données :

Les fonctionnalités d'entraînement et de test sont standardisées à l'aide de la classe StandardScaler de sklearn.preprocessing. Cela est fait pour mettre les variables à la même échelle afin d'améliorer les performances du modèle.

✓ Prédiction et évaluation du modèle :

Le modèle formé est utilisé pour prédire les classes de l'ensemble de test à l'aide de la méthode predict(). Les prédictions sont ensuite évaluées à l'aide de différentes mesures de performance telles que la précision, le score F1, etc. La matrice de confusion est également calculée et affichée sous forme de heatmap.

✓ Courbe ROC :

Une courbe ROC (Receiver Operating Characteristic) est tracé pour évaluer les performances du modèle. Les scores de probabilité de la classe positive sont utilisés pour tracer la courbe

ROC à l'aide de la fonction roc_curve() de sklearn.metrics. La zone sous la courbe (AUC) est également calculée à l'aide de la fonction roc_auc_score().

Les réseaux de neurones ANN

✓ Importez les bibliothèques nécessaires :

```python from tensorflow_addons import losses import tensorflow as tf from tensorflow import keras from sklearn.metrics import confusion_matrix, classification_report

```

✓ Définissez la fonction `ANN` avec les paramètres `X_train`, `y_train`,

✓ Créez un modèle séquentiel à l'aide de Keras :

Ce modèle est composé de trois couches : une couche d'entrée avec 16 neurones et une fonction d'activation ReLU, une couche intermédiaire avec 15 neurones et une fonction d'activation ReLU, et une couche de sortie avec 1 neurone et une fonction d'activation sigmoïde.

✓ Compilez le modèle en spécifiant l'optimiseur et la fonction de perte :

L'optimiseur utilisé est Adam, et la fonction de perte est spécifiée par le paramètre `loss` passé à la fonction `ANN`.

✓ Entraînez le modèle avec les données d'entraînement :

Si la valeur de `weights` est -1, le modèle est entraîné sans utiliser de poids de classe. Sinon, les poids de classe sont spécifiés à l'aide du paramètre `class_weight` lors de l'entraînement.

✓ Évaluez les performances du modèle sur les données de test :

Cela affiche les métriques de perte et d'exactitude (accuracy) du modèle sur les données de test.

✓ Effectuez des prédictions sur les données de test

Le modèle effectue des prédictions sur les données de test, puis arrondit les valeurs prédites à la valeur entière la plus proche (0 ou 1).

✓ Affichez le rapport de classification

✓ Retournez les prédictions

3.2.5. Conception

Différents tests ont été effectués avec les trois classifieur (**LR, RF, ANN**) et ceci pour :

- Les datasets original (déséquilibré).

- Les datasets après l'application des techniques (**RUS, ROS, SMOTE)**

Afin d'évaluer ces approches, **la précision, le recall, ainsi que le F1-score, Aire sous la courbe ROC (AUC-ROC) sont** utilisés.

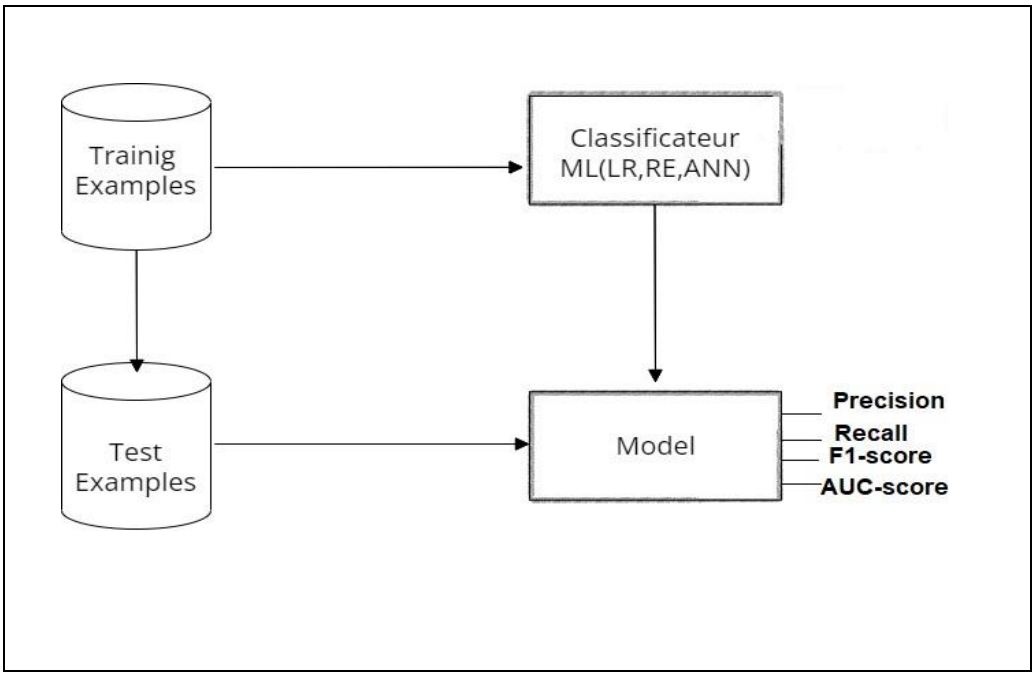

Figure 09. Conception de Dataset non équilibré et équilibré

3.2.5.2. Mesures des performances

Pour les datasets déséquilibrés, les métriques couramment utilisées pour mesurer la performance des Modèles sont :

• La combinaison de la **Precision**, du **Recall**, **Aire sous la courbe ROC (AUC-ROC) et** du **F1-score** :

❖ **La Precision** : est le ratio des observations prédites positives correctement sur le total des observations prédites positives tout court, soit : P = TP / (TP + FP). Cette métrique permet de voir à quel point nos prédictions positives tombent "juste".

❖ **Le Recall** (ou Sensitivity): est le ratio des observations prédites positive correctement sur l'ensemble des observations réellement positives,

soit: R = TP / (TP + FN). Cette métrique permet de mesurer à quel point l'on capture tous les vrais positifs dans nos prédictions.

❖ **Le F1-score** : est une moyenne pondérée de la Precision et du Recall. Son expression est : F1 = 2* RP / (R + P). Cette métrique est en générale plus utile que l'accuracy, car elle prend en compte à la fois les faux positifs et les faux négatifs.

• Les **matrices de confusions** sont des tables qui permettent de visualiser la performance d'un modèle en affichant les mesures des TP, TN, FP, et FN. Toutes les observations qui se situent sur la diagonale de la matrice ont été correctement prédites par le modèle, tandis que les observations qui ne se situent pas sur la diagonale correspondent à des erreurs du modèle. Un modèle parfait aurait donc l'ensemble de ses prédictions sur la diagonale d'une matrice de confusion. [18]

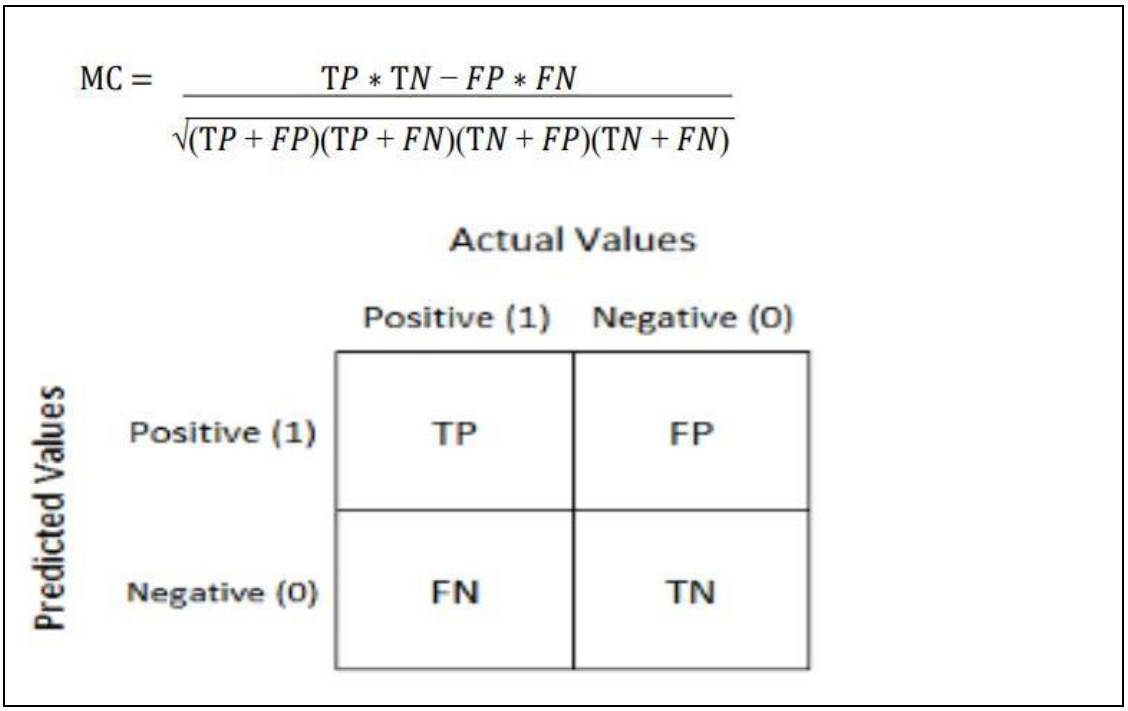

Figure 11 : Schéma création matrices de confusions [26]

❖ **Aire sous la courbe ROC (AUC-ROC)** : L'AUC-ROC est une mesure couramment utilisée pour évaluer la performance d'un modèle de classification binaire. Elle représente la probabilité que le modèle classe un exemple positif aléatoire de manière plus correcte qu'un exemple négatif aléatoire. L'AUC-ROC est moins sensible aux déséquilibres de classe et fournit une évaluation globale de la performance du modèle.

3.2.5.3. Evaluation de notre approche pour l'augmentation des données

Nous avons testé les techniques d'équilibrage (**RUS,ROS,SMOTE**) avec les classifiers (**LR,RF,ANN**)

Chaque technique utilise une méthode pour équilibrer les datasets

RUS : Cela fonctionne en supprimant au hasard certains échantillons de la catégorie la plus représentative des données pour équilibrer les catégories.

ROS : Il ajoute au hasard des copies répétées d'échantillons dans la catégorie la moins représentée dans les données pour améliorer sa représentation et atteindre l'équilibre.

SMOTE : crée de nouveaux échantillons synthétiques dans la catégorie la moins représentée en générant de nouveaux points parmi les échantillons existants à l'aide de la technique de diffusion linéaire.

3.3. Résultats

Nous exposons dans ce qui suit les résultats de l'étude comparative de techniques d'équilibrage par rapport au classifieur et datasets. Par la suite nous exposons les résultats de l'approche proposée.

3.3.1. Résultats avant l'équilibrage

Nous exposons d'abord une récapitulation de résultats des tests avant équilibrage avec les classifiers (**LR, RF, ANN**) été en général non satisfaisante

Résultats avant l'équilibrage				
		LR	RF	ANN
Precision	00 01	0.84 0.67	0.85 0.52	0.82 0.66
Recall	00 01	0.90 0.53	0.90 0.53	0.91 0.46
F1-score	00 01	0.87 0.59	0.82 0.53	0.87 0.54
AUC- score		0.71897696	0.69781954	0.68553121

Tableau 1- Récapitulation les datasets avec les Classifiers (LR ,RF, ANN)

3.3.2. Résultats après l'équilibrage

3.3.2.1. Résultats avec la technique RUS

```
    print(df_test_under.Churn.value_counts())
[42] ✓ 0.2s

... Random under-sampling:
    Churn
    0    1869
    1    1869
    Name: count, dtype: int64
```

Tableau 2– dataset Equilibré avec RUS

Résultats après l'équilibrage				
RUS				
		LR	RF	ANN
Precision	00	0.77	0.65	0.73
	01	0.75	0.77	0.73
Recall	00	0.74	0.65	0.73
	01	0.78	0.77	0.73
F1-score	00	0.75	0.69	0.73
	01	0.76	0.73	0.73
AUC- score		0.75534759	0.70989704	0.73128342

Tableau 3– Récapitulation les datasets avec le Classifier (LR ,RF , A NN) par RUS

3.3.2.2. Résultats avec la technique ROS

```
66]  ✓ 0.1s

..  Random over-sampling:
    Churn
    0    5163
    1    5163
    Name: count, dtype: int64
```

Tableau 4– dataset Equilibré avec ROS

Résultats après l'équilibrage				
ROS				
		LR	RF	ANN
Precision	00	0.78	0.71	0.80
	01	0.75	0.76	0.74
Recall	00	0.74	0.78	0.70
	01	0.79	0.69	0.82
F1-score	00	0.76	0.75	0.75
	01	0.77	0.72	0.78
AUC- score		0.76766698	0.69781954	0.76282671

Tableau 5– Récapitulation les datasets avec le Classifier (LR ,RF , A NN) par ROS

3.3.2.3. Résultats avec la technique SMOTE

Method3: SMOTE

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(sampling_strategy='minority')
X_sm, y_sm = smote.fit_resample(X, y)

y_sm.value_counts()
```

[79] ✓ 2.7s

```
Churn
0    5163
1    5163
Name: count, dtype: int64
```

Tableau 6 - dataset Equilibré avec SMOTE

Résultats après l'équilibrage				
SMOTE				
		LR	RF	ANN
Precision	00	0.78	0.71	0.80
	01	0.75	0.76	0.74
Recall	00	0.74	0.78	0.70
	01	0.79	0.69	0.82
F1-score	00	0.76	0.75	0.75
	01	0.77	0.72	0.78
AUC- score		0.76766698	0.69781954	0.76282671

Tableau 7– Récapitulation les datasets avec le Classifier (LR ,RF , A NN) par SMOTE

3.4. Analyse des résultats

Nous analyserons les résultats des trois classifieurs avant et après balancement des données, et nous ferons une comparaison à trois niveaux entre les résultats , les classifieurs , les techniques d'équilibrage

3.4.1. Le premier niveau : Comparer les résultats avant et après l'équilibrage

D'après les résultats donnés, nous pouvons conclure plusieurs choses :

• Avant l'équilibrage : Les trois modèles (Logistic Regression, Random Forest Classifier et ANN) ont montré des performances différentes. LogisticRegression a obtenu le niveau le plus

élevé de précision, de rappel et de score F1 pour la classe 0, tandis que ANN a obtenu le niveau le plus élevé de précision, de rappel et de score F1 pour la classe 1. RandomForestClassifier a obtenu des performances moyennes par rapport aux deux autres modèles.

• Après l'équilibrage : Après l'application de la technique Undersampling (RUS), une amélioration des performances des modèles a été observée. LogisticRegression et RandomForestClassifier ont atteint des niveaux similaires de précision, de rappel et de score F1 pour les deux classes 0 et 1. ANN a également montré une amélioration de la précision, du rappel et du score F1 pour les deux classes 0 et 1.

• Nous pouvons conclure que l'application de techniques d'équilibrage

(**RUS,ROS,SMOTE**) peut contribuer à améliorer les performances des modèles sur des dataset déséquilibrés, réduisant ainsi le biais du modèle envers la classe majoritaire.

• Avant équilibrage, nous avons observé de bons résultats pour le score auc des trois classificateurs indiquant que le modèle est capable de discriminer modérément entre les deux classes 1,0 Après équilibrage, nous avons remarqué une amélioration significative du score AUC, ce qui augmente la capacité du modèle à faire la distinction entre les deux catégories

• Cependant, il y a encore de la place pour améliorer les performances des modèles, en particulier pour RandomForestClassifier, qui continue à avoir des performances faibles en termes de rappel pour la classe 1.

En général, il convient de prendre en compte ces résultats lors de la prise de décisions concernant le choix du modèle approprié ou l'application de techniques d'équilibrage dans un projet de classification de données.

3.4.2. Le deuxième niveau : Comparer les performances des classifieurs

Sur la base des résultats donnés, nous pouvons tirer les conclusions suivantes concernant les classificateurs LR, RF et ANN :

1. Régression Logistique (LR) :

- Avant l'équilibrage : La régression logistique a montré de bonnes performances avec une précision élevée pour la classe 0 et une précision moyenne pour la classe 1. Cependant, le rappel pour la classe 1 et f1 étaient relativement faible, Néanmoins le modèle a montré un bon score AUC

- Après l'équilibrage : Après l'application de la technique d'équilibrage, on a observé une amélioration des performances de la régression logistique, avec des valeurs de précision et de rappel qui se sont rapprochées pour les deux classes 0 et 1, En plus d'une légère amélioration du score AUC 2. Forêt Aléatoire (RF) :

- Avant l'équilibrage : La forêt aléatoire a obtenu des performances moyennes, avec une bonne précision pour la classe 0 et une précision moyenne pour la classe 1. Cependant, le rappel et f1 pour la classe 1 étaient faible, et un score AUC moyenne

- Après l'équilibrage : Après l'application de la technique d'équilibrage, on a constaté une légère amélioration des performances de la forêt aléatoire en termes de précision et de rappel pour les deux classes 0 et 1, et il y a une amélioration du score AUC 3. Réseau de Neurones Artificiels (ANN) :

- Avant l'équilibrage : L'ANN a montré de bonnes performances, avec une précision moyenne pour la classe 0 et une précision élevée pour la classe 1. Cependant, le rappel et f1 pour la classe 1 étaient faible.

Après l'équilibrage : Après l'application de la technique d'équilibrage, on a observé une amélioration des performances de l'ANN, avec des valeurs de précision, rappel et de AUC qui se sont rapprochées pour les deux classes 0 et 1.

Le score AUC avant était bon, mais s'est amélioré après l'équilibrage

En résumé, avant l'équilibrage, à la fois la régression logistique, la forêt aléatoire et l'ANN ont obtenu des résultats moyens à bons, mais ont souffert d'un faible rappel et f1 pour la classe 1.

Après l'équilibrage, les trois classificateurs ont montré une amélioration, avec des valeurs de précision et de rappel plus proches pour les deux classes

3.4.3. Le troisième niveau : Comparer les performances des techniques d'équilibrage

Sur la base des résultats présentés et des performances des trois techniques (RUS, ROS, SMOTE) dans les trois classificateurs (LR, RF, ANN), nous pouvons dire que toutes les techniques atteignent des performances similaires, bien qu'il y ait une légère supériorité de SMOTE et ROS car elles ont le même principe, ce qui est excessif en prise de données, contrairement à la technique RUS, qui repose sur la réduction des données, mais cette légère supériorité ne cache pas que les techniques ont fourni de bons résultats, et que toutes les techniques ont fourni des résultats très similaires.

Il peut être intéressant, en fonction d'autres critères tels que le coût, le temps et la facilité de mise en œuvre, de choisir la méthode la plus appropriée. De ce point de vue je choisis la technologie SMOTE

3.5. Aussi, pourquoi dois choisir SMOTE ?

Effectivement, l'utilisation des techniques aléatoires comme RUS (Random Undersampling) et ROS (Random Over-sampling) peut avoir un impact négatif sur les performances du modèle, car elles peuvent supprimer des données importantes ou ajouter des données inutiles au modèle.

Lors de l'utilisation de **RUS**, des échantillons sont supprimés de manière aléatoire de la classe majoritaire, ce qui peut entraîner la perte de données importantes ou d'informations précieuses présentes dans la classe majoritaire. Cela peut réduire la capacité du modèle à reconnaître des motifs importants ou à classer correctement les données.

En ce qui concerne **ROS**, l'ajout d'échantillons aléatoires à la classe minoritaire peut complexifier les données et entraîner une augmentation du bruit dans le modèle. Certains échantillons supplémentaires peuvent ne pas contenir d'informations utiles ou ne pas représenter

correctement la classe minoritaire, ce qui peut introduire des biais inutiles ou augmenter la similitude entre les échantillons.

L'équilibrage des données est important pour traiter le problème du déséquilibre, mais ces facteurs doivent être pris en compte lors de l'utilisation de techniques d'échantillonnage aléatoire. Il peut être préférable d'utiliser une technique comme

SMOTE, qui génère de nouveaux échantillons basés sur les données d'origine et préserve les informations importantes, afin d'éviter la perte de données importantes ou l'ajout de bruit inutile.

Cependant, il est important de prendre en considération la nature des données et le contexte spécifique de l'application lors du choix de la meilleure approche

3.6. Conclusions

Dans ce chapiter Nos expérimentations ont montré que l'utilisation de ces techniques peut améliorer significativement les performances des modèles prédictifs sur des ensembles de données déséquilibrés.

Cependant, il convient de noter que le choix de la technique dépend des caractéristiques spécifiques de l'ensemble de données et qu'il est important de prendre en compte les implications de chaque méthode

Conclusions générales

Les données déséquilibrées sont l'un des problèmes potentiels dans le domaine de l'exploration de données et de l'apprentissage automatique.

Ce problème peut être abordé en analysant correctement les données.

Ce projet nous a permis d'explorer des solutions à ce problème grâce aux techniques suivantes : sous-échantillonnage aléatoire (RUS), suréchantillonnage (ROS) et suréchantillonnage synthétique (SMOTE).

Ces techniques ont été appliquées à des données déséquilibrées et ont pour but d'équilibrer les classes.

Notre approche consiste d'abord à trouver des solutions au problème de déséquilibre des données qui fait l'objet de notre projet, puis nous choisissons un jeu de données qui s'applique à notre problème et qui devrait être des classes déséquilibrées.

Dans l'étape suivante, nous avons créé un modèle de formation pour classer les dataset déséquilibrés à l'aide de classificateurs LR, RF, ANN, puis nous avons équilibré les dataset grâce à des techniques soumises à des algorithmes d'apprentissage automatique, à savoir RUS, ROS, puis SMOTE.

Après avoir entraîné les données équilibrées et non équilibrées, nous avons obtenu des résultats que nous avons comparés. Nous avons obtenu de bons résultats car nous avons remarqué que les performances des trois techniques sont très proches et ont considérablement amélioré les performances du modèle en termes de valeur f1, de rappel, de précision et de valeur AUC, qui sont les mêmes métriques que nous avons précédemment sélectionnées pour évaluer notre travail...

À la fin ….

Ce projet a fait l'objet d'une expérience intéressante qui nous a permis d'améliorer nos connaissances et nos compétences en programmation.

Nous proposons dans le futur d'implémenter ces techniques sur d'autres types de données et de filtres, et pourquoi pas d'utiliser de meilleures techniques

Bibliographies

1. Samuel, A. L. Some studies in machine learning using the game of checkers. IBM Journal of research and development, 3(3), 210-229, 1959. [En ligne]

2. Tom M Mitchell et al. Machine learning. McGraw-Hill Education, 1st Edition 01 Mars 1997. [En ligne]

3. Gouvernance de l'intelligence artificielle dans les entreprises. CIGREF, Septembre2016. [En ligne]

4. "Brain Team", Google Research. [En ligne]

5. "OpenAI". Openai. [En ligne]

6. "Introducing machine learning". Amazon,09 Avril 2015. [En ligne]

7. Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. John Wiley & Sons

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

9. Claude Touzet, LES RÉSEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME, Collection de l'EERIE, 1992. *En ligne+

10. https://blog.octo.com/donnees-desequilibrees-que-faire/

11 . https://datascience.eu/fr/programmation-informatique/smote/

12. Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. Intelligent Data Analysis, 6(5), 429-449.

13. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284

14. 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.

15. Hu, F., & Li, H. (2013). A Novel Boundary Oversampling Algorithm Based on Neighborhood Rough Set Model: NRSBoundary-SMOTE. Mathematical Problems in Engineering, 2013, 1–10. DOI : 10.1155/2013/694809

16. https://www.kaggle.com/code/nirajpoudel/customer-churn-prediction-for-atelephone-company/input

17. https://www.lebigdata.fr/python-langage

18. https://www.aquiladata.fr/insights/comment-gerer-le-desequilibre-des-classesdans-un-jeu-de-donnees/

19. https://vie-etudiante.cegepjonquiere.ca/qu-est-ce-que-l-intelligence-artificielle.html

20. https://fr.linkedin.com/pulse/r%C3%A9gression-logistique-st%C3%A9phanejaubert?trk=pulse-article_more-articles_related-content-card

21. https://www.javatpoint.com/machine-learning-random-forest-algorithm

22. https://kobia.fr/que-sont-les-reseaux-de-neurones/

23. https://www.mdpi.com/2076-3417/13/6/4006

24. https://www.mdpi.com/2076-3417/13/6/4006

25. https://www.researchgate.net/figure/llustration-of-synthetic-minority-oversamplingtechnique_fig2_343326638

26. https://towardsdatascience.com/understanding-confusion-matrixa9ad42dcfd62?gi=b2bf89934e9b

27. https://www.tensorflow.org/, https://keras.io/ https://pandas.pydata.org/

28. https://www.w3schools.com/