الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

المركز الجامعي بلحاج بوشعيب - عين تموشنت
Centre Universitaire BELHADJ Bouchaib - AïnTémouchent
معهد العلوم
Institut des Sciences

قسم الرياضيات والإعلام الآلي
Département des Mathématiques et Informatique

Rapport de projet de fin d'études

Présenté en vue de l'obtention du diplôme de Master académique en Informatique
Spécialité : Réseaux et ingénierie des données

Extraction des connaissances à partir d'une base de Données (*Application à la Détection de Fraude dans la Consommation d'Electricité et du Gaz*)

Réalisé par :
Mr BENFODDA Mohamed
Mr SAID Djillali

Supervisé par :
Mr. BENOMAR Mohammed Lamine (MCB, Département MI, Institut des sciences, CUAT)

Jury :
Président : Mme BENOSMAN Amina (MCB, Département MI, Institut des sciences, CUAT)
Examinateur : Mme ABDERAHIM (MAA, Département MI, Institut des sciences, CUAT)

Présenté le 23/09/2020

Promotion 2019 - 2020
Réf …-RPFELI-1920CUBBAT-ST-MI

REMERCIEMENTS

Avant toute chose, nous tenons à prosterner devant Allah pour le remercier de nous avoir donné le courage, la patience et la volonté de faire ce travail et de l'achever en temps opportun.

Nous tenons à remercier notre encadreur ***Dr. Mohammed Lamine BENOMAR*** *pour ses précieux conseils, sa patience et ses nobles valeurs humaines.*

Nos remerciements vont également aux membres du jury pour nous avoir honorés par l'évaluation de notre travail.
Merci aussi pour tous ceux que nous avons omis de citer ici et qui de près ou de loin ont contribué au bon déroulement de ce travail.

Merci à vous tous !

Sommaire

INTRODUCTION GENERALE

L'extraction de connaissances à partir des données (ECD) est un processus non trivial d'identification de structures inconnues, valides et potentiellement utiles dans les bases de données**[1]**,l'objectif est d'aider l'être humain à prendre des décision et extraire des informations nouvelles et utiles (Connaissances) à partir des données dont le volume croît très rapidement.

Parmi ces connaissances on trouve la prise de décision critique qui est importante dans de nombreux domaines et secteurs. Comme la détection des fraudes qui est un défi de taille dans différents domaines (réseaux, commerce, paiement,…). Cependant, les transactions frauduleuses représentent une très petite fraction de l'activité globale d'une organisation/entreprise. Néanmoins, un faible pourcentage de cette activité frauduleuse peut rapidement se transformer en des pertes financières importantes dans l'absence des outils efficaces et systèmes intelligents pour y faire face et mettre ces activités à l'écart avec des sanctions.

Les techniques d'Intelligence Artificielle (IA) peuvent apporter une aide précieuse dans ce cas, car ils peuvent traiter une masse importante de données et prendre en compte un grand nombre de cas déjà traités ensuite proposer pour un nouveau cas une décision fondée sur une analyse de tous les cas précédents traiter par apprentissage.

Dans ce travail de fin d'étude intitulé « Extraction des connaissances à partir d'une base de données »en s'intéressent aux méthodes d'Apprentissage Automatique (Machine Learning) qui peuvent apprendre, s'adapter et découvrir de nouvelles façons de prévenir les fraudes. Les spécialistes des données ont réussi à résoudre ce problème grâce au ML et à l'analyse prédictive.

Notre objectif de travail est la détection de fraude dans la consommation d'électricité et de gaz à l'aide d'algorithmes prédictifs agissant sur des données issues des compteurs et reconnaître les clients impliqués dans des activités frauduleuses.

CHAPITRE 1

Introduction à l'Intelligence Artificielle

I. Intelligence Artificielle

I.1. Définition de l'IA (Intelligence Artificielle):

L'Intelligence Artificielle, selon **Larousse**, se définirait comme étant «l'ensemble de théories et de techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence.»

De ce fait, il s'agit d'ordinateurs ou des machines dotées de programmes de performances similaires à l'intelligence humaine, ou même, amplifiées par la technologie. Ces machines sont en mesure de :

- Raisonner
- Traiter de grandes quantités de données
- Discerner des modèles indétectables par l'œil d'un humain
- Comprendre et analyser ces modèles
- Interagir avec l'Homme
- Apprendre progressivement
- Améliorer continuellement ses performances

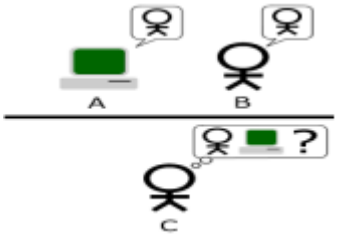
Selon Harry Shum, Président Exécutif de Microsoft, l'IA fonctionne seulement s'il y a présence « d'une vaste quantité de data; d'une puissance informatique extraordinaire, notamment grâce au Cloud; et des algorithmes révolutionnaires, basés sur le Deep Learning ». L'IA s'applique aujourd'hui dans des domaines variés tels que: Les jeux de réflexion, La recherche mathématique, La finance, La médecine, La reconnaissance faciale et la compréhension des langues, La robotique (voir Figure 1).**[2]**

Figure 1 : Domaines d'application de l'IA

I.2 Historique de l'intelligence artificielle

Les premières traces de l'IA remontent à 1950 dans un article d'Alan Turing intitulé « *Computing Machinery and Intelligence* » qui a fait une proposition de test d'intelligence artificielle fondée sur la faculté d'une machine à imiter la conversation humaine (Figure 2)**[3]**.

Figure 2 : Schéma du test de Turing



Dès le milieu des années 60, la recherche autour de l'IA aux U.S.A était principalement financée par le Département de la Défense. Certains experts prédisaient à l'époque que « des machines seront capables, d'ici 20 ans, de faire le travail que toute personne peut faire ».

Figure 3 : Vision de l'IA dans les années 60

L'IA est exploitée sur des terrains jusqu'alors peu communs. On retrouve à cette époque le Data Mining, ou encore les diagnostics médicaux. Il faudra attendre 1997 pour une véritable sortie médiatique lorsque le fameux Deep Blue créé par IBM a battu Garry Kasparov, alors champion du monde d'échec.

Figure 4 : Deep Blue en face Garry Kasparov

Dès le début de notre décennie certaines sociétés vont prendre les devants. En effet, la problématique de l'IA n'est plus d'avoir les cerveaux pour élaborer des systèmes, mais d'avoir de la donnée à traiter. C'est pour cela que Google devient rapidement un pionnier. En 2013, Facebook ouvre le Facebook Artificial Intelligence Research (FAIR). Un tournant qui éloigne le géant de sa mission sociale pour se tourner vers les sciences. Amazon, Microsoft, Apple, Netflix, Tesla ne sont pas en reste non plus, de même de nombreuses sociétés chinoises.

Figure 5 : Facebook Artificial Intelligence Research

Figure 6 : Google lance *Google AI*

I.3. Le Big Data et l'intelligence Artificielle

Le phénomène du Big Data, ou volumes massifs de données, se traduit par une explosion quantitative de données numériques. Il s'agit de stocker un nombre infini d'informations, ou méga données, sur une base numérique. Cela concerne environ 2,5 trillions d'octets de données à traiter tous les jours.

Ces informations peuvent provenir de différents horizons, telles que les messages que nous envoyons, les vidéos ou photos que nous publions, les informations sur le climat ou les évènements météorologiques, les signaux GPS retraçant les trajectoires effectuées, les enregistrements transactionnels d'achats en ligne, les capteurs etc… (Le marché du Big Data atteindrait 67 milliards de dollars en 2021).

Figure 6 : Sources Gratuites de Big Data

Le concept du Big Data doit répondre à une règle dite la règle des 3V. Le Big Data doit contenir:

- ✓ Un grand Volume de données à traiter.
- ✓ Une grande Variété d'informations provenant de diverses sources.
- ✓ Un certain niveau de Vélocité, ou Vitesse, à atteindre, c'est à dire la rapidité à laquelle les données sont générées et traitées

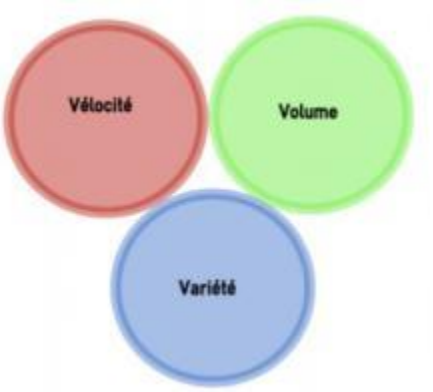


Figure 7 : Les 3 V du Big Data

Le Big Data et l'Intelligence Artificielle sont fortement liées, de par la collecte de données massives. L'intelligence Artificielle nécessite de très grands volumes de données pour être efficace et procéder au machine Learning, ou aussi appelé algorithme d'apprentissage.

L'IA rend le Big Data intelligent en accomplissant des tâches d'analyses complexes beaucoup plus rapidement qu'un humain. Le Big Data permet à l'Intelligence Artificielle de stocker un volume conséquent de données, structurées et non structurées, avec une vitesse de traitement satisfaisante. Ainsi, ses quantités de données a permis d'obtenir des avancées considérables au sein de l'apprentissage de la machine intelligente. **[4]**.

Figure 8 : Logo des logiciels de solutions analytique

I.4. Les technologies de l'IA

I.4.1. Briques technologiques de l'IA

Même si l'intelligence artificielle est principalement associée à une discipline mathématique et des techniques algorithmiques, elle inclut également d'autres briques pour répondre à un usage complet (voir Figure 9). Les principaux éléments constitutifs d'un système d'IA sont les suivants :

1. Des plateformes digitales (données utilisateurs) ou des infrastructures de capteurs (données machines ou environnementales) pour générer des flux réguliers de données/évènements.

2. Un réseau de communications pour permettre la collecte des données/évènements utilisés par l'IA. Ces données doivent être suffisamment représentatives du cas d'usage que l'on cherche à adresser.

3. Une infrastructure de calcul hyperscale de traitement pour stocker et exploiter les flux de données dans un délai raisonnable.

4. Des technologies algorithmiques d'intelligence artificielle (machine Learning, Deep Learning, réseau de neurones, etc…), une mesure de la performance, une mesure de l'erreur et un ensemble d'évènements de référence pour l'apprentissage.

5. Une interface homme / machine simple de type plateforme accessible via mobile ou ordinateur pour les usages de type « aide à la décision » ou une interface évoluée (drone, robot, véhicule autonome) pour les usages de type « décision autonome ». La plateforme est l'élément qui associe une technologie algorithmique à un usage sectoriel. **[5]**

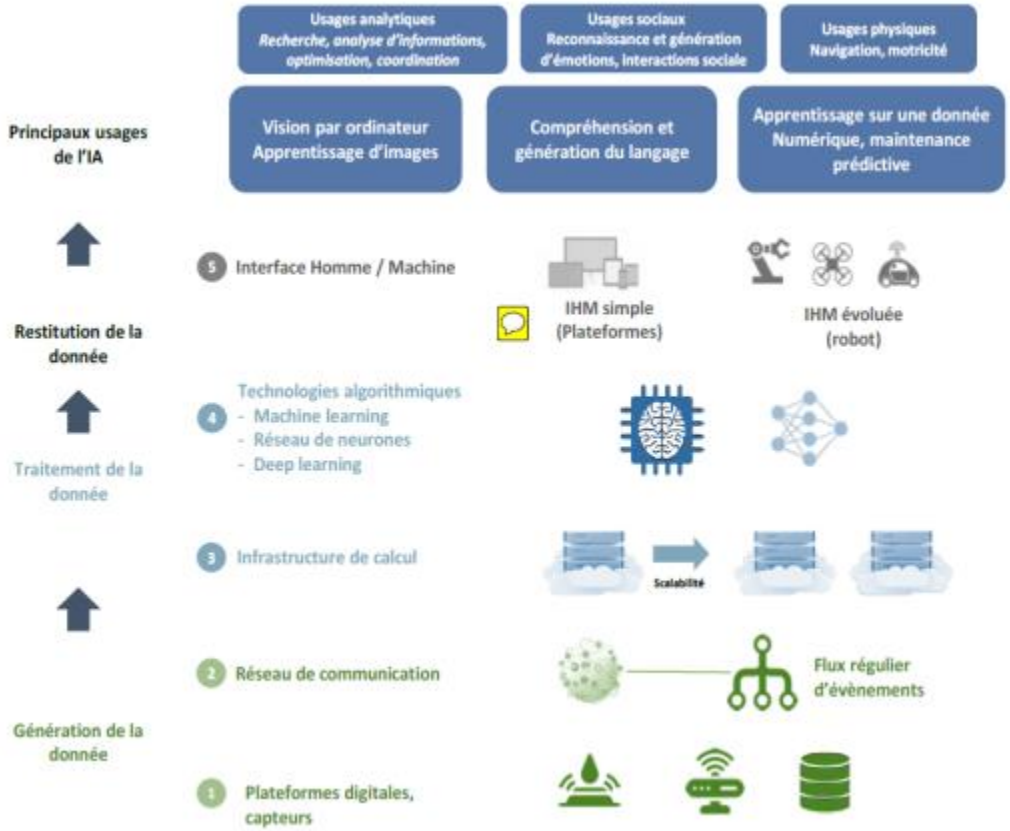


Figure 9 : Briques technologiques de l'IA

I.4.2.Principaux usages de l'IA

L'intelligence artificielle est une technologie révolutionnaire, de plus en plus utilisée par les entreprises de tous les secteurs. Toutefois, en réalité, l'IA est un terme qui englobe une large variété de technologies. À connaître absolument pour se préparer à l'avenir :

➢ Machine Learning

Le Machine Learning permet d'automatiser le processus de création d'algorithmes, en utilisant des données pour les entrainer plutôt que de laisser les développeurs humains rédiger du code. À l'aide des données, il est possible de montrer l'exemple aux algorithmes pour leur enseigner comment effectuer une tâche.

Le Machine Learning est aujourd'hui utilisé pour une très large variété de cas d'usage différents. Il est notamment utilisé dans les domaines de la sécurité des données, de la sécurité personnelle, de la santé, du marketing, de la détection de fraude, de la recherche en ligne, des voitures autonomes et des moteurs de recommandation…etc.

➢ Deep Learning

Le Deep Learning reprend le principe du Machine Learning, mais va plus loin en créant plusieurs couches de Machine Learning après le premier point de décision. C'est ce qu'on appelle un réseau de neurones artificiels.

Ce réseau a pour but de simuler la façon dont le cerveau humain opère. Ainsi, les décisions prises par la première couche de Machine Learning servent d'input pour les décisions suivantes.

➢ Le traitement naturel du langage

Le traitement naturel du langage a pour but de permettre aux machines de comprendre le langage humain. C'est cette technologie d'intelligence artificielle qui a permis de donner naissance à des assistants numériques comme Amazon Alexa, Microsoft Cortana, Google Assistant ou Apple Siri.

Il s'agit d'une technologie essentielle pour l'analyse de données non structurées telles que les enregistrements de santé électroniques, les emails, les messages textuels, les transcriptions ou encore les publications sur les réseaux sociaux. En bref, le Naturel Langage Processing permet d'analyser n'importe quel contenu comprenant du langage humain.

➢ La vision artificielle

Permet aux ordinateurs d'analyser, de traiter et de comprendre une ou plusieurs images. Elle capture et analyse des informations visuelles à l'aide d'une caméra, de la conversion du signal analogique en numérique et du traitement des signaux numériques. Elle sert dans de nombreuses applications, de l'identification des signatures à l'analyse d'imagerie médicale. La vision par ordinateur, dédiée au traitement informatique de l'image est souvent assimilée à la vision artificielle.

➢ La robotique

Est un domaine de l'ingénierie consacré à la conception et à la fabrication de robots. Ces robots servent souvent à exécuter des tâches que les humains ont du mal à réaliser ou à réaliser à la pareille. On en trouve sur les chaînes de

montage de l'industrie automobile ou à la NASA, pour déplacer de gros objets dans l'espace. Depuis peu, les chercheurs utilisent l'apprentissage automatique pour construire des robots capables d'interactions sociales.**[4]**

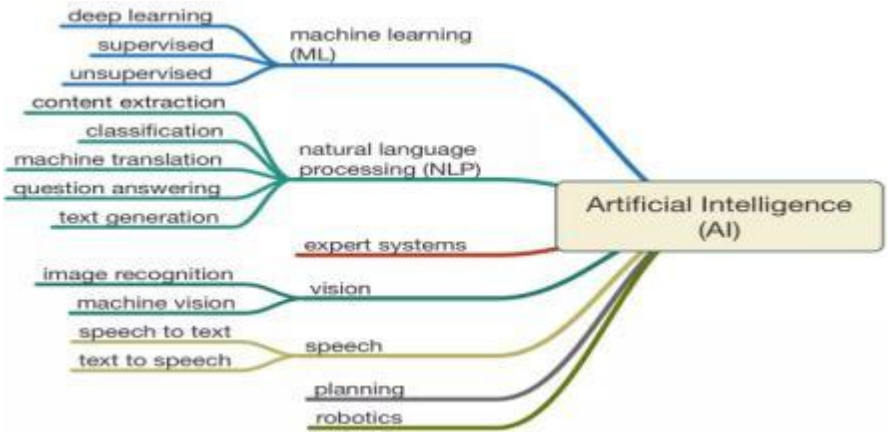


Figure 10 : Les usages de l'IA

➢ Distinction entre Intelligence Artificielle, Machine Learning et Deep Learning

Il y a une confusion fréquente dans le débat public entre « intelligence artificielle », apprentissage automatique (machine Learning) et apprentissage profond (Deep Learning). Pourtant, ces notions ne sont pas équivalentes, mais sont imbriquées :

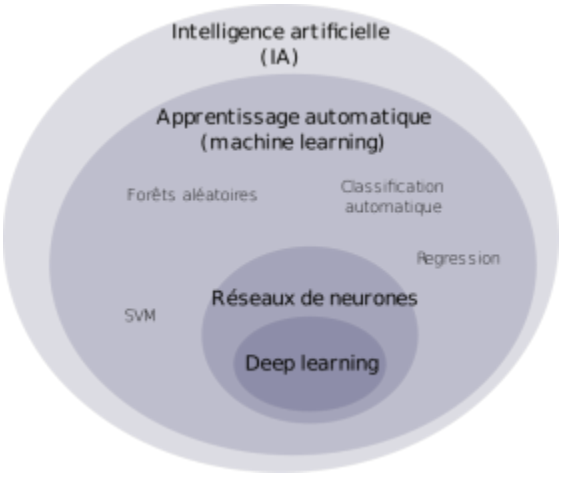


Figure 11 : Schéma montrant le positionnement des notions d'IA, Machine Learning et Deep Learning imbriquées les unes aux autres.

I.5. Apprentissage automatique (Machine Learning)

I.5.1. Introduction

Le Machine Learning est une technologie d'intelligence artificielle permettant aux ordinateurs d'apprendre sans avoir été programmés explicitement à cet effet. Pour apprendre et se développer, les ordinateurs ont toutefois besoin de données à analyser et sur lesquelles s'entraîner (voir Figure 12).

Il s'agit d'une science moderne permettant de découvrir des patterns (Modèle) et d'effectuer des prédictions à partir de données en se basant sur des statistiques, sur du forage de données, sur la reconnaissance de patterns et sur les analyses prédictives.

Par exemple, en se basant sur les informations associées à un client, et sur ses données historiques, le Machine Learning permet de détecter une fraude potentielle en une milliseconde. Ainsi, cette méthode est nettement plus efficace que les méthodes traditionnelles pour l'analyse de données transactionnelles.

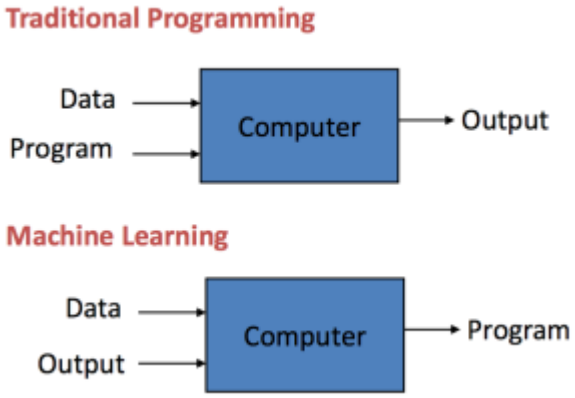


Figure 12 : Programmation Classique vs Machine Learning

I.5.2. Type d'apprentissages

Pour pouvoir extraire les données pertinentes à une entreprise, différentes méthodes sont mises en œuvre. On distingue l'apprentissage supervisé qui sert essentiellement à la classification des données et l'apprentissage non supervisé qui est utilisé dans la recherche d'associations ou de groupes d'individus, le Tableau 1 présente une comparaison entre les deux approches.

Dans la partie suivante nous allons détailler l'apprentissage supervisé. Cependant, l'Apprentissage non-supervisé est utilisé dans :

- **Analyse des grappes (clusters)** : identifie des grappes de similitudes et forme ensuite des groupes d'objets qui sont plus semblables sur certains aspects que d'autres groupes: contrairement à la classification, les groupes (ou grappes) ne sont pas prédéfinis et peuvent prendre des formes différentes selon les données analysées.
- **Analyse d'association**: révèle les corrélations entre deux ou plusieurs éléments indépendants qui ne sont pas directement liés, mais qui se produisent plus souvent ensemble.

Apprentissage Supervisé	Apprentissage Non-supervisé
Méthode Prédictive	Méthode Descriptive
Les données d'entraînement contiennent à la fois les attributs et les résultats désirés (catégories, classes)	Le modèle n'est pas alimenté avec des résultats connus durant la phase d'entraînement
Pour certains exemples les résultats sont connus et sont donnés comme paramètres au modèle pendant le processus d'apprentissage	Peut être utilisé pour regrouper les données d'entrée en classes basées uniquement sur leurs propriétés statistiques
La construction d'un jeu d'entraînement, de validation et de test est cruciale	Donner une signification des clusters et labellisation
Les méthodes sont généralement rapides et précises	La labellisation peut être effectuée même s'il n'y a qu'un faible échantillon d'objets appartenant à la classe
Possibilité de généralisation : donner un résultat correct lorsqu'une nouvelle entrée dont on ne connaît pas la classe est présentée	Possibilité de généralisation dans un domaine donné (selon les données)
Exemple d'algorithme: K-NN, SVM, Arbre de Décision, régression logistique…	Exemple d'algorithme: K-means, Mean-shift…

Tableau 1 - Comparaison des méthodes d'apprentissages supervisés et non supervisés

I.5.3. Apprentissage supervisé

➢ Les notions de bases

Avec l'apprentissage supervisé on peut développer des modèles pour résoudre 2 types de problèmes **[6]**:

• Les problèmes de Régression

• Les problèmes de Classification

Dans les problèmes de régression, on cherche à prédire la valeur d'une variable continue, c'est-à-dire une variable qui peut prendre une infinité de valeurs. Par exemple :

- Prédire le prix d'un appartement (y) selon sa surface habitable (x).
- Prédire la quantité d'essence consommée (y) selon la distance parcourue (x).

Dans un problème de classification, on cherche à classer un objet dans différentes classes, c'est-à-dire que l'on cherche à prédire la valeur d'une variable discrète (qui ne prend qu'un nombre fini de valeurs). Par exemple :

• Prédire si un email est un spam ($classe\ y$ = 1) ou non ($classe\ y$ = 0) selon le nombre de liens présent dans l'email (x).

• Prédire si une tumeur est maligne (y = 1) ou bénigne (y =0) selon la taille de la tumeur ($\boldsymbol{x1}$) et l'âge du patient ($\boldsymbol{x2}$).

• La détection des fraudes dans une entreprise, client avec fraude (y=1) client sans fraude (y=0).

Dans le cas d'un problème de classification, on représente souvent les classes par des symboles, plutôt que par leur valeur numérique (0, 1, …).

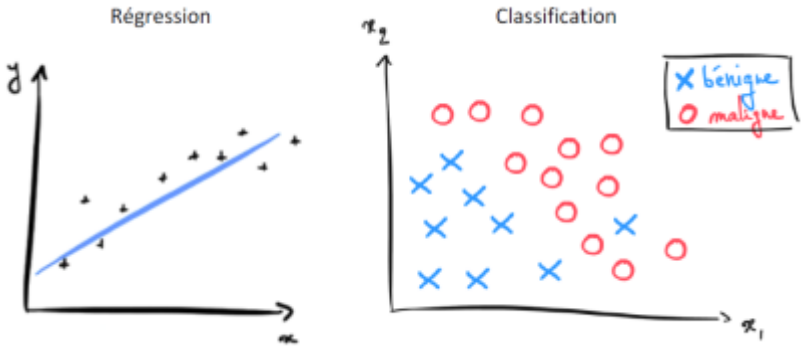


Figure 13 : les types de problèmes d'apprentissage supervisé

Pour maîtriser l'apprentissage supervisé, il faut absolument comprendre et connaitre les 4 notions suivantes :

- Le Dataset
- Le Modèle et ses paramètres
- La Fonction Coût
- L'Algorithme d'apprentissage

➢ **Notion 1 : Apprendre à partir d'exemples (Dataset)**

On parle d'apprentissage supervisé lorsque l'on fournit à une machine beaucoup d'exemples ($\boldsymbol{x}$, $\boldsymbol{y}$) dans le but de lui faire apprendre la relation qui relie $\boldsymbol{x}$ (entrée) à $\boldsymbol{y}$ (sortie)**[6]**.

En Machine Learning, on empile ces exemples ($\boldsymbol{x}$, $\boldsymbol{y}$) dans un tableau que l'on appelle Dataset :

- La variable $\boldsymbol{y}$ porte le nom de Target (la cible). C'est la valeur que l'on cherche à prédire.
- La variable $\boldsymbol{x}$ porte le nom de Feature(facteur, attributs). Un facteur influence la valeur de $\boldsymbol{y}$, et on a en général beaucoup de Features(Ex : âge, sexe, taille, prix…) dans notre Dataset que l'on regroupe dans une matrice $\boldsymbol{X}$.

Ci-dessous, un Dataset qui regroupe des exemples d'appartements avec leur prix $\boldsymbol{y}$ ainsi que certaines de leurs caractéristiques (Features).

Target y	Features x_1	x_2	x_3
Prix	Surface m2	N chambres	Qualité
€ 313,000.00	124	3	1.5
€ 2,384,000.00	339	5	2.5
€ 342,000.00	179	3	2
€ 420,000.00	186	3	2.25
€ 550,000.00	180	4	2.5
€ 490,000.00	82	2	1
€ 335,000.00	125	2	2

Tableau 2 : Dataset labellisé

➢ **Notion 2 : Développer un modèle à partir du Dataset**

En Machine Learning, on développe un modèle à partir de ce Dataset. Il peut s'agir d'un modèle linéaire comme vous pouvez le voir à gauche, ou bien un modèle non-linéaire comme vous pouvez le voir à droite.

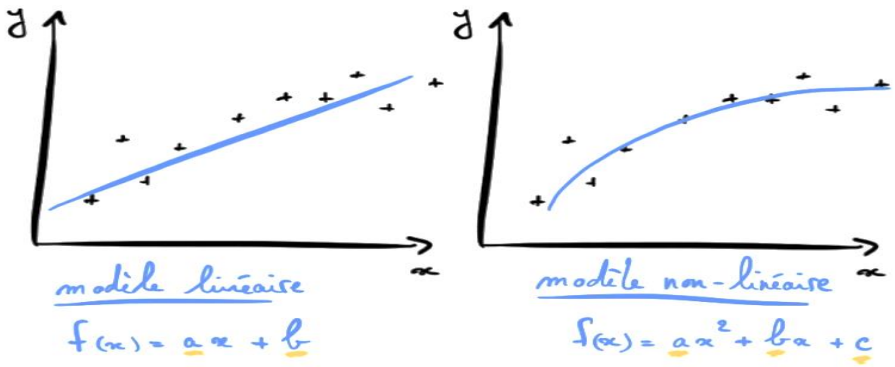


Figure 14 : Les modèles d'apprentissage (linéaire / non linéaire)

On définit a, b, c, etc. comme étant les paramètres d'un modèle.

- **Notion 3 : Les erreurs de notre modèle - la Fonction Coût**

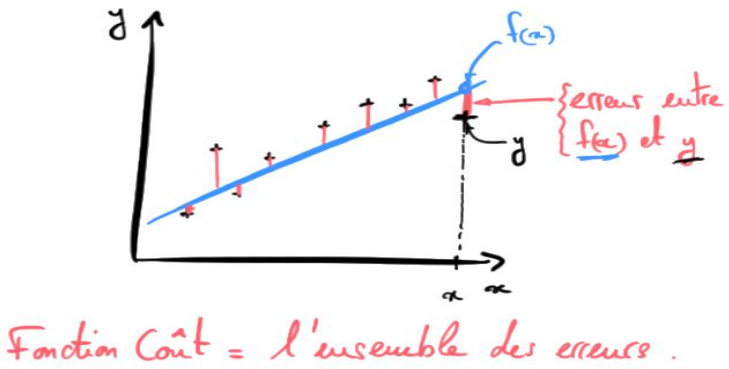


Figure 15 : La fonction coût

Autre chose à noter est qu'un modèle nous retourne des erreurs par rapport à notre Dataset. On appelle **Fonction Coût** l'ensemble de ces erreurs (le plus souvent on prend la moyenne quadratique des erreurs) (Figure 15).

Avoir un bon modèle, c'est avoir un modèle qui nous donne de petites erreurs, donc une petite Fonction Coût.

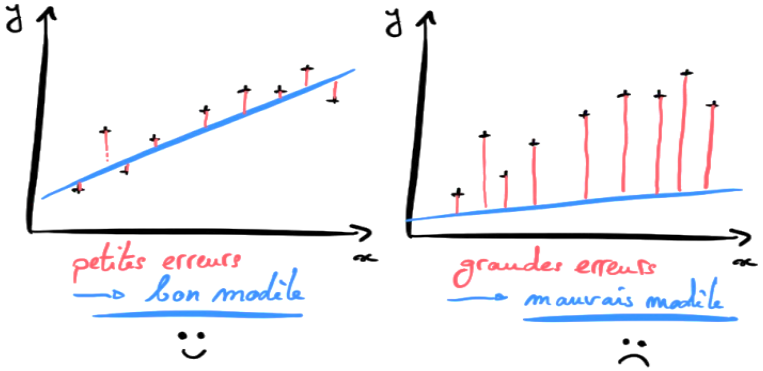


Figure 16 : minimiser l'erreur du modèle (fonction coût)

➢ **Notion 4 : Apprendre, c'est minimiser la Fonction Coût**

Ainsi l'objectif central en apprentissage supervisé, c'est de trouver les paramètres du modèle qui minimisent la **Fonction Coût.** Pour cela, on utilise un algorithme d'apprentissage **[6].**

➢ Entrainer et Tester un modèle

Dans l'étape d'apprentissage (**Figure 17**), un classifieur (une fonction, un ensemble de règles, ...) est construit en analysant (ou en apprenant de) une base de données d'exemples d'entrainement avec leurs classes respectives. Un exemple X = (X1, X2, .., Xm) est représenté par un vecteur d'attributs de dimension m. Chaque exemple est supposé appartenir à une classe prédéfinie représentée dans un attribut particulier de la base de donnée appelé attribut de classe. Puisque la classe de chaque exemple est donnée, cette étape est aussi connue par l'apprentissage supervisé.

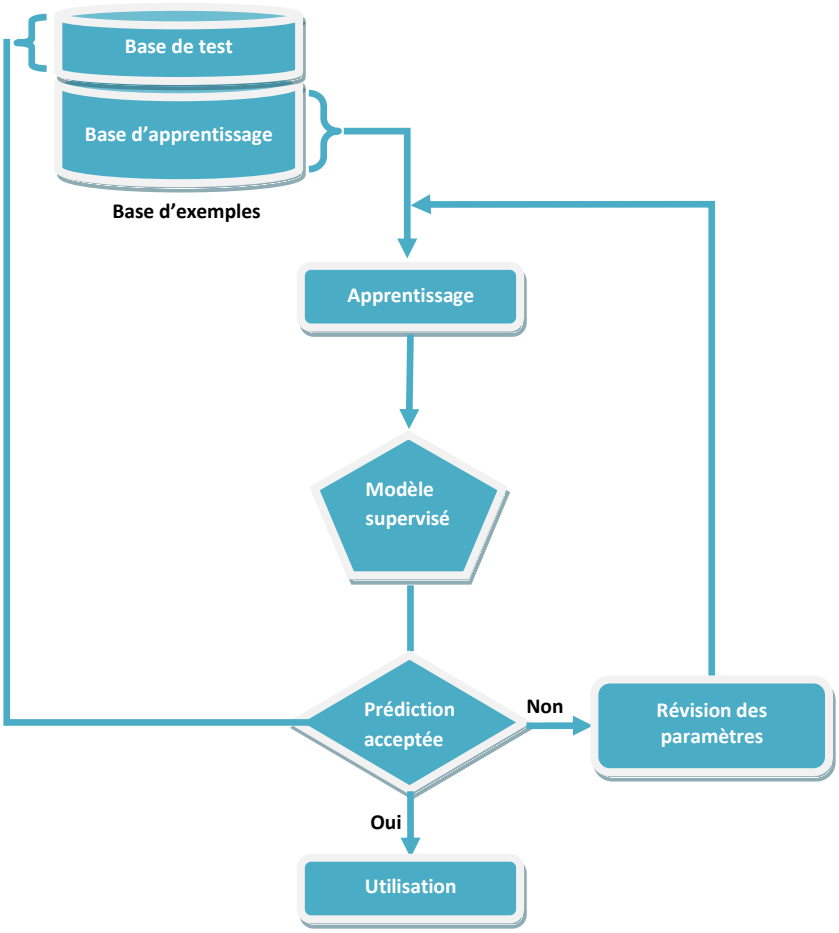


Figure 17 : cycle d'apprentissage supervisé

Dans l'étape de classification (**Figure 18**), le modèle construit dans la première étape est utilisé pour classer les nouvelles données. Mais avant de passer à l'utilisation, le modèle doit être testé pour s'assurer de sa capacité de généralisation sur les données non utilisées dans la phase d'entrainement. Le modèle obtenu peut être testé sur les données d'entrainement elles-mêmes, la précision (le taux de reconnaissance) est généralement élevée mais ne garantit pas automatiquement une bonne précision sur les nouvelles données. En effet, les

données d'entrainement peuvent contenir des données bruitées ou erronées (outliers) qui ne représentent pas le cas général et qui tire le modèle vers leurs caractéristiques. Ce cas est appelée le sur-apprentissage ou en anglais "over fitting" et qui peut être évité en testant le modèle sur une base de données différente appelée base de test. La base de test est un ensemble d'exemples ayant les mêmes caractéristiques que ceux de la base d'entrainement et qui sont écartés au départ de l'entrainement pour effectuer les tests.**[7]**

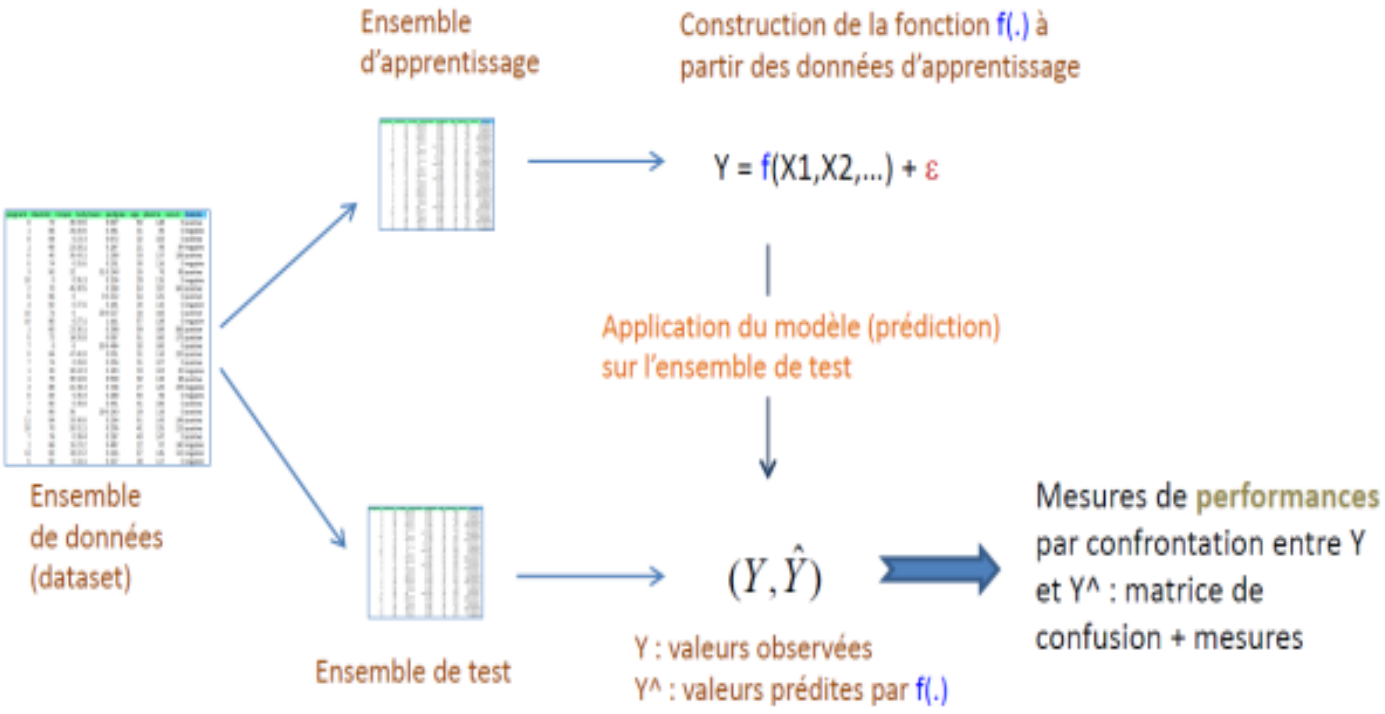


Figure 18 : Analyse Prédictive (Etape de classification)

➢ Evaluation du fractionnement Train-Test

Le fractionnement train-test est une technique permettant d'évaluer les performances d'un algorithme d'apprentissage automatique. Il peut être utilisé pour des problèmes de classification ou de régression et peut être utilisé pour tout algorithme d'apprentissage supervisé.

La procédure consiste à prendre un ensemble de données et à le diviser en deux sous-ensembles. Le premier sous-ensemble est utilisé pour ajuster le modèle et est appelé ensemble de données d'apprentissage. Le deuxième sous-ensemble n'est pas utilisé pour entraîner le modèle ; à la place, l'élément d'entrée de l'ensemble de données est fourni au modèle, puis des prédictions sont faites et comparées aux valeurs attendues. Ce deuxième ensemble de données est appelé ensemble de données de test.

L'objectif est d'estimer les performances du modèle d'apprentissage automatique sur de nouvelles données : données non utilisées pour entraîner le modèle.

C'est ainsi que nous comptons utiliser le modèle dans la pratique. À savoir, pour l'ajuster sur des données disponibles avec des entrées et des sorties connues, puis faites des prédictions sur de nouveaux exemples dans le futur où nous n'avons pas la sortie attendue ou les valeurs cibles. La procédure de test de train est appropriée lorsqu'un ensemble de données suffisamment volumineux est disponible.

Le partage Train-test est principalement basé sur la taille du train Dataset et des jeux de test. Ceci est le plus souvent exprimé sous forme de pourcentage entre 0 et 1 pour le train ou les jeux de données de test. Par exemple, un ensemble de formation d'une taille de 0,67 (67%) signifie que le pourcentage restant de 0,33 (33%) est attribué à l'ensemble de test.

Il n'y a pas de pourcentage de partage optimal. Vous devez choisir un pourcentage partagé qui répond aux objectifs de votre projet avec des considérations qui incluent:

- Coût de calcul de la formation du modèle.
- Coût de calcul lors de l'évaluation du modèle.
- Représentativité de l'ensemble de formation.
- Représentativité de l'ensemble de test.

Néanmoins, les pourcentages fractionnés courants comprennent (voir Figure 19) :

Train: 80%, Test: 20%

Train: 67%, Test: 33%

Train: 50%, Test: 50%

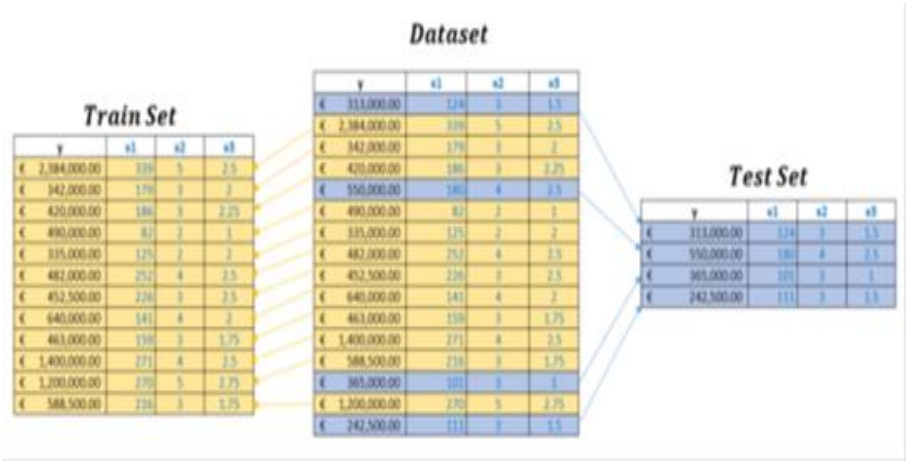


Figure 19 : Exemple de Séparation Dataset en Train Set (80%) et Test Set (20%)

➢ Evaluation du Modèle

A partir du modèle construit dans la première étape (entrainement) l'étape de prédiction est réalisée afin de classer les nouvelles données (échantillon Test). Dans la partie évaluation on trouve la matrice de confusion représentant la confrontation entre Y observation sur l'échantillon Test et la prédiction f(x). Ainsi, la matrice de confusion est une matrice qui rassemble en lignes les observations (y) et en colonnes les prédictions f(x). Les éléments de la matrice représentent le nombre d'exemples correspondants à chaque cas, selon le tableau 3 d'une matrice de confusion pour la classification binaire :

Prédictions f(x) / Observations (y)	Non-Fraudeur (0)	Fraudeur (1)
Non-Fraudeur (0)	**VN** (**Vrai** Négatif)	**FP** (Faux Positif)
Fraudeur (1)	**FN** (**Faux** Négatif)	**VP** (**Vrai** positif)

Vrai : la prédiction est vraie (label prédit = vrai label)
Faux : la prédiction est fausse (label prédit ≠ vrai label)
Positif : la prédiction est Fraudée.
Négatif : la prédiction est Non-Fraudé.

Tableau 3 : Matrice de confusion

Un modèle sans erreurs aura ses résultats rassemblés sur la diagonale de sa matrice de confusion (VP et VN). Dans le cas multi classes la matrice sera plus large avec les classes possibles au lieu des deux classes 1 et 0.

À partir de la matrice de confusion il est possible de calculer la précision « P » du modèle calculée comme suit :

$$P = \frac{VP + VN}{VP + FN + FP + VN}$$

Deux autre mesures sont utilisées : la sensitivité « Sv » et la spécificité « Sp ». La première représente le rapport entre les observations positives correctement prédites et le nombre des observations positives, et la deuxième représente le rapport entre les observations négatives correctement prédites et le nombre total des observations négatives.

$$\begin{cases} Sv = \dfrac{VP}{VP + FN} \\ Sp = \dfrac{VN}{VN + FP} \end{cases}$$

L'outil d'évaluation graphique le plus utilisé dans la littérature est la Courbe ROC (receiver operating characteristic) et le calcul de l'AUC (Area Under Curve). Une courbe ROC est un graphique représentant les performances d'un modèle de classification pour tous les seuils de classification. Cette courbe trace le taux de vrais positifs (TVP) en fonction du taux de faux positifs (TFP). Où :

- Le **TVP** est l'équivalent au Rappel. Il est donc défini comme suit :

$$TVP = \frac{VP}{VP + FN}$$

- Le **TFP** est défini comme suit :

$$TFP = \frac{FP}{FP + VN}$$

Une courbe ROC trace les valeurs TVP et TFP pour différents seuils de classification. Diminuer la valeur du seuil de classification permet de classer plus d'éléments comme positifs, ce qui augmente le nombre de faux positifs et de vrais positifs. La figure 20 représente une courbe ROC classique.

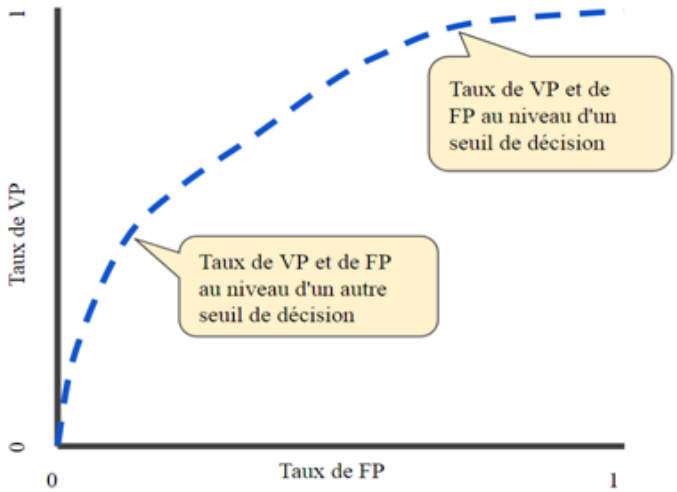


Figure 20 : la courbe ROC

L'AUC ou « aire sous la courbe ROC » mesure l'intégralité de l'aire à deux dimensions située sous l'ensemble de la courbe ROC (par calculs d'intégrales) de (0,0) à (1,1), selon la figure suivante

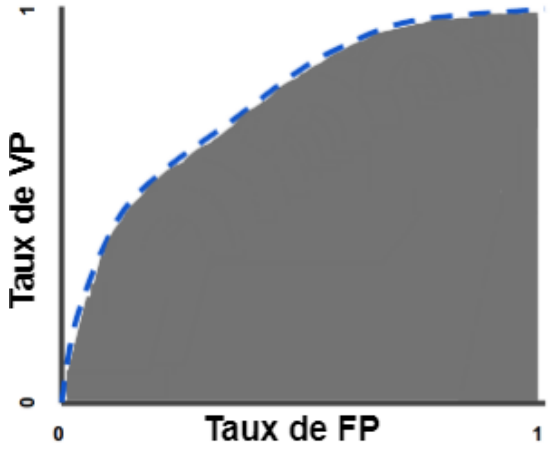


Figure 21 : l'air sous la courbe ROC (AUC)

I.5.4. Exemple d'Algorithme de Machine Learning :

- **XGBoost (eXtreme Gradient Boosting)**

XGBoost Est une implémentation open source optimisée de l'algorithme d'arbres de Boosting de gradient. Le Boosting de Gradient est un algorithme d'apprentissage supervisé dont le principe et de combiner les résultats d'un ensemble de modèles plus simple et plus faibles afin de fournir une meilleur prédiction. On parle d'ailleurs de méthode d'agrégation de modèles. L'idée est donc simple : au lieu d'utiliser un seul modèle, l'algorithme va en utiliser plusieurs qui serons ensuite combinés pour obtenir un seul résultat.

Pour décrire succinctement le principe, l'algorithme travaille de manière séquentielle. Contrairement par exemple au Random Forest. Cette façon de faire va le rendre plus lent bien sûr mais il va surtout permettre à l'algorithme de s'améliorer par capitalisation par rapport aux exécutions précédentes. Il commence donc par construire un premier modèle qu'il va bien sur évaluer (on est bien sûr de l'apprentissage supervisé). A partir de cette première évaluation, chaque individu va être alors pondéré en fonction de la performance de la prédiction. Etc. XGBoost se comporte donc remarquablement dans les compétitions d'apprentissage automatique (Machine Learning), mais pas seulement grâce à son principe d'auto-amélioration séquentielle.

XGBoost est l'algorithme d'apprentissage automatique le plus populaire de nos jours. Quel que soit le type de données (régression ou classification), il est bien connu de fournir de meilleures solutions que les autres algorithmes ML. En fait, depuis sa création (début 2014), il est devenu le meilleur chez les utilisateurs de Kaggle pour traiter des données structurées.

Le fonctionnement est basé sur le Boost qui est une technique séquentielle qui fonctionne sur le principe d'un ensemble (voir Figure 16). Il combine un ensemble d'apprenants faibles et offre une précision de prédiction améliorée. À tout instant t, les résultats du modèle sont pondérés en fonction des résultats de l'instant t-1 précédent. Les résultats prédits correctement reçoivent un poids inférieur et ceux mal classés sont pondérés plus haut.

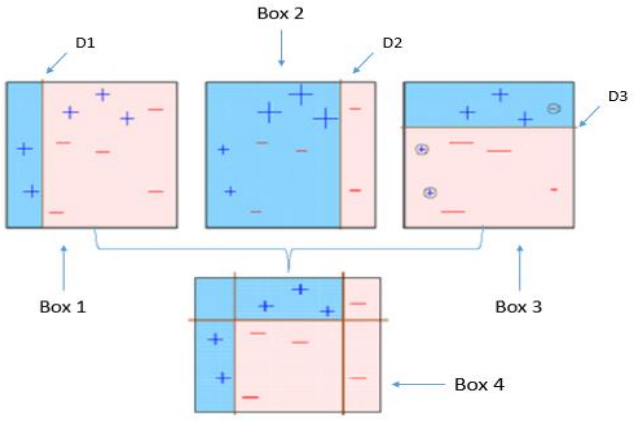


Figure 22 : Principe de Fonctionnement du Boost

Quatre classificateurs (dans 4 cases), illustrés ci-dessus, tentent de classer les classes + et - de manière aussi homogène que possible (Figure 22).

- Box 1 : Le premier classificateur (généralement une souche de décision) crée une ligne verticale (divisée) à D1. Il indique que tout ce qui se trouve à gauche de D1 est + et tout ce qui se trouve à droite de D1 est - . Cependant, ce classificateur classe mal trois points +. Notez qu'un tronçon de décision est un modèle d'arbre de décision qui se divise uniquement à un niveau, par conséquent, la prédiction finale est basée sur une seule entité.

- Box 2 : Le deuxième classificateur donne plus de poids aux trois + points mal classés (voir la plus grande taille de +) et crée une ligne verticale à D2. Encore une fois, il est dit que tout ce qui se trouve à droite de D2 est - et à gauche est + . Pourtant, il fait des erreurs en classant incorrectement trois - points.

- Box 3 : Encore une fois, le troisième classificateur donne plus de poids aux trois - points de mal classés et crée une ligne horizontale à D3. Pourtant, ce classificateur ne parvient pas à classer correctement les points (dans les cercles).

- Box 4 : Il s'agit d'une combinaison pondérée des classificateurs faibles (encadrés 1,2 et 3). Comme vous pouvez le voir, cela permet de classer correctement tous les points.

C'est l'idée de base derrière le renforcement des algorithmes: construire un modèle faible, tirer des conclusions sur l'importance et les paramètres des différentes fonctionnalités, puis utiliser ces conclusions pour construire un nouveau modèle plus fort et tirer parti de l'erreur de classification erronée du modèle précédent et essayer de le réduire .

➢ k-NN (k Plus Proches Voisins)

En Anglais k-Nearest Neighbour (KNN), est l'un des algorithmes les plus appliqués en machine Learning. Il consiste à assigner à un échantillon non classifié, la classification du plus proche des éléments précédemment classifiés. Pour classifier une nouvelle instance, la distance euclidienne (éventuellement avec un facteur de poids) est calculée entre l'instance et chaque instance d'entraînement. La classe de la nouvelle instance est assignée par rapport à celle présentant la plus courte distance. De façon plus générale, les k plus proches voisins sont calculés et la nouvelle instance est assignée à la classe qui est la plus présente parmi les k plus proches voisins, k étant un paramètre à choisir.

Figure 23 : Principe de l'algorithme KNN

CHAPITRE 2

Analyse et Fouille de Données

II. Analyse et Fouille de Données

II.1. Introduction

Les concepts de fouille de données et d'extraction de connaissances à partir de données sont parfois confondus et considérés comme synonymes. Mais, formellement on considère la fouille de données comme une étape centrale du processus d'extraction de connaissances des bases de données (ECBD ou KDD pour Knowledge Discovery in Databases en anglais) **[8]**

L'exploration de données est un sous-domaine de l'informatique qui combine de nombreuses techniques issues des statistiques, de la science des données, des bases de données et de l'apprentissage par machine. Voici les principales évolutions de l'histoire de l'exploration de données, ainsi que sa fusion avec la science des données et les Big data (**voir Figure 24**).

II.2. Définition FDD :

" Le datamining, ou fouille de données, est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de bases de données informatiques (souvent grandes), de façon automatique ou semi-automatique, en vue de détecter dans ces données des règles, des associations, des tendances inconnues ou cachées, des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données" **[9].**

On dénombre les processus suivants :

- Le choix de la base de données
- Le prétraitement, dans le but d'amorcer un nettoyage des données
- Leur transformation dans la forme adéquate à leur traitement
- Le processus d'analyse mathématique (data mining)
- L'interprétation des résultats de l'analyse

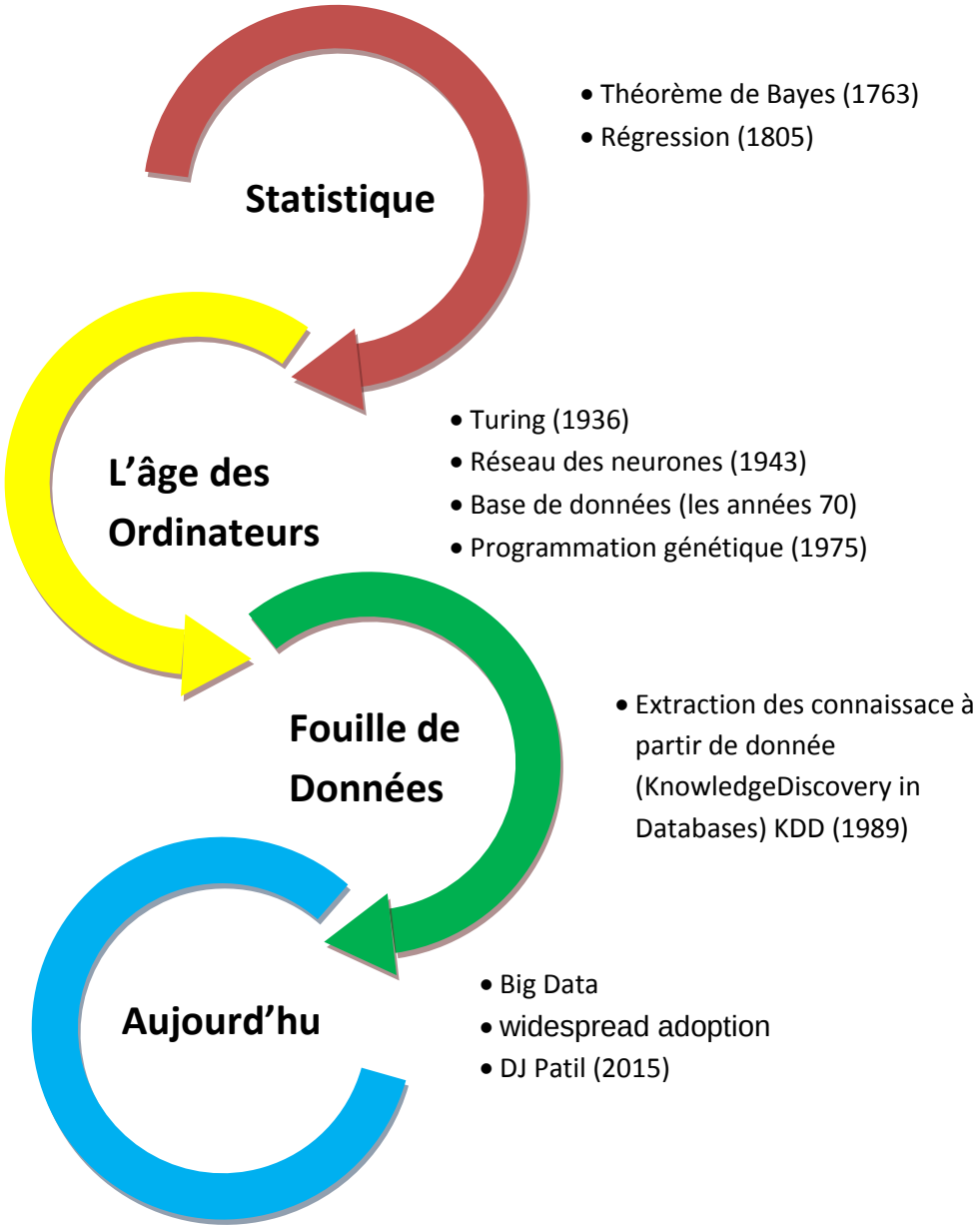


Figure 24 : Historique de la FDD

II.3. Les méthodes de Data Mining

On peut dégager deux grandes catégories de méthodes d'analyse consacrées à la fouille de données **[8]**. La frontière entre les deux peut être définie par la spécificité des techniques, et marque l'aire proprement dite du "Data Mining". On distingue donc :

A. **Les méthodes classiques** : On y retrouve des outils généralistes de l'informatique ou des mathématiques :
– Les requêtes dans les bases de données, simples ou multicritères, dont la représentation est une vue,
– les requêtes d'analyse croisée, représentées par des tableaux croisés,
– les différents graphes, graphiques et représentations,
– les statistiques descriptives,

– l'analyse de données : analyse en composantes principales,
– etc.

B. **Les méthodes sophistiquées** : Elles ont été élaborées pour résoudre des tâches bien définies. Ce sont :
– Les algorithmes de segmentation,
– les règles d'association,
– les algorithmes de recherche du plus proche voisin,
– les arbres de décision,
– les réseaux de neurones,
– les algorithmes génétiques,
– etc.

II.4. Outils de Fouille de Données :

Désignation	Caractéristiques	Système d'exploitation	Coûts/Licence
RapidMiner	Puissant et polyvalent, fournit un environnement intégré pour la préparation des données, l'apprentissage automatique, l'apprentissage en profondeur, l'exploration de texte avec un avantage particulièrement dans l'analyse prédictive	Windows, macOS, Linux	Freeware, différentes versions payantes
WEKA	Une suite populaire d'algorithmes d'apprentissage automatique. Écrite en Java, développée à l'université de Waikato, Nouvelle-Zélande. permet de faire l'analyse statistique, la classification, le clustering et l'analyse prédictive.	Windows, macOS, Linux	Software libre (GPL)
Orange	Crée des visualisations de données particulièrement attrayantes et intéressantes sans connaissances préalables approfondies	Windows, macOS, Linux	Software libre (GPL)
KNIME	est un logiciel libre de l'université de Constance, Le principal outil de data mining ouvert que l'analyse prédictive a rendu accessible au grand public	Windows, macOS, Linux	Software libre (GPL) (à partir de la version 2.1)
SAS	Logiciel d'exploration de données puissant et coûteux pour les grandes entreprises	Windows, macOS, Linux	Freeware limité disponible dans les établissements d'enseignement, prix sur demande seulement, différents modèles extensifs possibles

Le langage R	R est un langage de programmation et un logiciel libre destiné aux statistiques et à la science des données soutenu par la R Foundation for Statistical Computing. Il permet de faire l'analyse statistique, la classification, le clustering et l'analyse prédictive	Windows, macOS, Linux	Software libre (GPL)
Python	Python est un langage de programmation très puissant utilisé en Data Mining pour faire de l'analyse statistique, la classification, le clustering et l'analyse prédictive.	Windows, macOS, Linux	La Python Software Foundation License (PSFL) est une licence libre semblable à la licence BSD.

Tableau 4 : Outils de Fouille de données.

II.5. Le processus ECD

L'extraction de Connaissances dans les Bases de Données (E.C.B.D.) est une discipline récente, à l'intersection des domaines des bases de données, de l'intelligence artificielle, de la statistique, des interfaces homme / machine et de la visualisation. A partir de données collectées par des experts, il s'agit de proposer des connaissances nouvelles qui enrichissent les interprétations du champ d'application, tout en fournissant des méthodes automatiques qui exploitent cette information.

Dans cette partie, nous allons donner un aperçu général sur le processus ECD (définition, étapes. . .) notamment sur l'étape fouille de données (Data Mining).., les techniques utilisées (motif, règle d'association, classification . . .)

✓ Les étapes d'un processus d'ECD

Ce processus comporte quatre étapes principales (**Voir Figure 25**) :

– Nettoyage et intégration des données ;

– la préparation des données ;

– la fouille de données (Data Mining) ;

– l'interprétation.

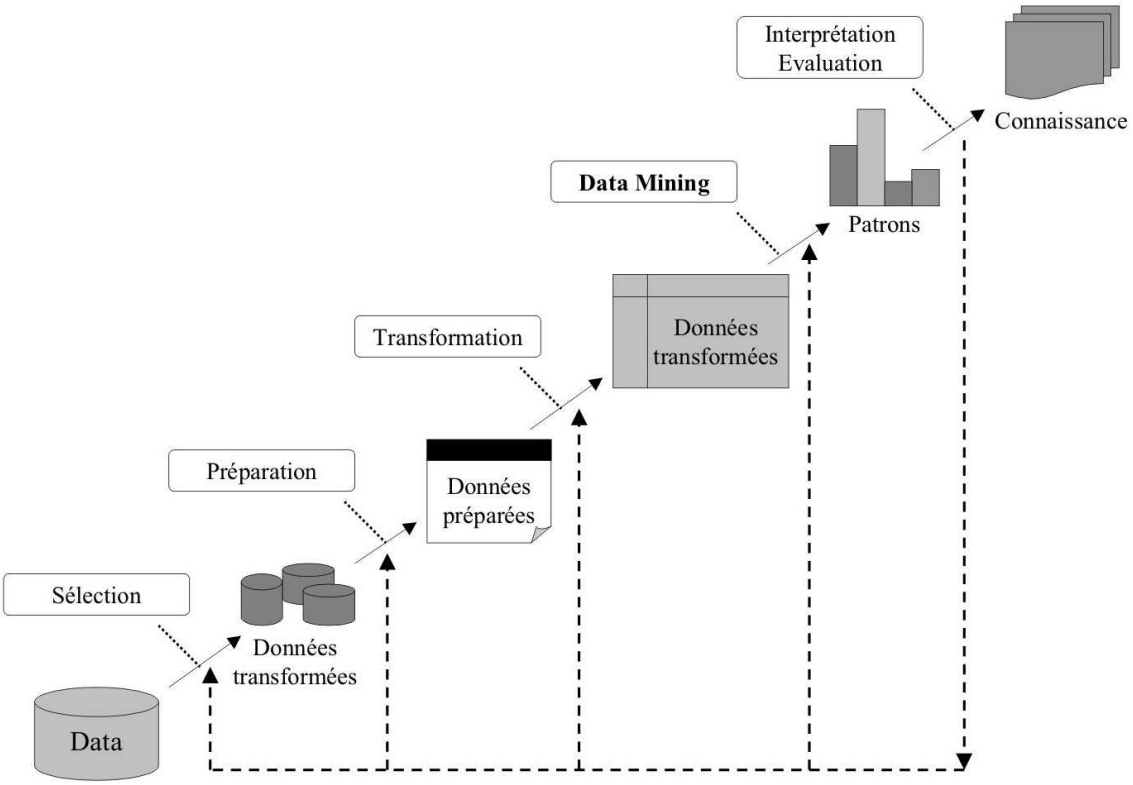


Figure 25 : Un aperçu des étapes de l'ECD

➢ Nettoyage et intégration des données

Le nettoyage des données consiste à traiter ces données bruitées, soit en les supprimant, soit en les modifiant de manière à tirer le meilleur profit. L'intégration est la combinaison des données provenant de plusieurs sources (base de données, sources externes, etc.). Le but de ces deux opérations est de générer des entrepôts de données et/ou des magasins de données spécialisés contenant les données traitées pour faciliter leurs exploitations futures.

➢ Prétraitement des données

Il peut arriver parfois que les bases de données contiennent à ce niveau un certain nombre de données incomplètes et/ou bruitées. Ces données erronées, manquantes ou inconsistantes doivent être traitées si cela n'a pas été fait précédemment. Dans le cas contraire, durant l'étape précédente, les données sont stockées dans un entrepôt. Cette étape permet de sélectionner et transformer des données de manière à les rendre exploitables par un outil de fouille de données. Cette seconde étape du processus d'ECD permet d'affiner les données. Si l'entrepôt de données est bien construit, le prétraitement de données peut permettre d'améliorer les résultats lors de l'interrogation dans la phase de fouille de données.

➢ Fouille de données

La fouille de données (Data Mining en anglais), est le cœur du processus d'ECD. Il s'agit à ce niveau de trouver des pépites de connaissances à partir des données. Tout le travail consiste à appliquer des méthodes intelligentes dans le but d'extraire cette connaissance. Il est possible de définir la qualité d'un modèle en fonction de critères comme les performances obtenus, la fiabilité, la compréhensibilité, la rapidité de construction et d'utilisation et enfin l'évolutivité. Tout le problème de la fouille de données réside dans le choix de la méthode adéquate à un problème donné. Il est possible de combiner plusieurs méthodes pour essayer d'obtenir une solution optimale globale. Nous ne détaillerons pas d'avantage la fouille de données dans ce paragraphe car elle fera l'objet d'une section complète.

➢ Interprétation et Evaluation

Cette phase est constituée de l'évaluation, qui mesure l'intérêt des motifs extraits, et de la présentation des résultats à l'utilisateur grâce à différentes techniques de visualisation. Cette étape est dépendante de la tâche de fouille de données employée.

CHAPITRE 3

PYTHON

Pour le Machine Learning

III. Python

➢ Présentation du Langage :

Créé originellement par le programmeur *Guido van Rossum (Pays-Bas)* depuis 1989, la nature open source du langage Python lui permet d'évoluer grâce à la communauté des développeurs Python « *Python Software Foundation* ». Une mise à jour importante a été le passage de la version 2 à la version 3 publié en décembre 2008.

Le python est un langage de programmation interprété libre (open source), est placé sous une licence proche de la licence *BSD* le *PSFL*, multiplateformes (il fonctionne sous un grand nombre de plateforme telle que Windows, Linux, MAC OS, OS/2, UNIX, Android, IOS …), et multi-paradigmes (programmation impérative structurée, fonctionnelle, orientée objet, et peut être utilisé comme un langage de script).

Python offre une multitude de modules et bibliothèques spécialisées qui apportent des fonctionnalités divers (gestion courrier électronique, application web, manipulation des structures de données, calcules numériques...).

Python est un langage de programmation à usage général qui devient de plus en plus populaire pour le Machine Learning. Les entreprises du monde entier utilisent Python pour récolter des informations à partir de leurs données et gagner un avantage concurrentiel.

Installer un environnement Python pour Machine Learning avec Anaconda

L'installation de *Python* peut-être compliqué pour les novices, tout d'abord il faut se décider entre les versions 2.X et 3.X du langage. Et par la suite, choisir les librairies nécessaires (ainsi que les versions compatibles) pour faire du Machine Learning. Sans oublier les subtilités liées aux différents OS (Windows, Linux, Mac…) qui peuvent rendre l'installation encore plus douloureuse.

Ce document vous guidera pour installer pas à pas un environnement de développement *Python* en utilisant *Anaconda*, fonctionnel avec les packages nécessaires pour faire du Machine Learning.

Anaconda est un gestionnaire de packages, un gestionnaire d'environnement et une distribution Python qui contient une collection de nombreux packages open source (numpy, scikit-learn, scipy, pandas…).

Les packages/Librairies basiques nécessaire pour faire du ML sont :

- Jupyter Notebook : En tant qu'application serveur-client, Jupyter Notebook vous permet de modifier et d'exécuter vos blocs de code via un navigateur Web. L'application peut être exécutée sur un PC sans accès à Internet, ou elle peut être installée sur un serveur distant, où vous pouvez y accéder via Internet. C'est l'éditeur préféré pour les data-scientiste.

- NumPy : est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

- Pandas : est un outil d'analyse et de manipulation de données rapide, puissant, flexible et facile à utiliser.

- Scikit-learn : (également connu sous le nom de sklearn) est une bibliothèque d'apprentissage de logiciels libres pour le langage de programmation Python.il propose divers algorithmes de classification, de régression et de clustering, y compris dessupports vector machines, random forests, gradient boosting, k-means and DBSCAN, et est conçu pour interagir avec les bibliothèques numériques et scientifiques Python NumPy et SciPy.

- Matplotlib : est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques.

Installation Anaconda :

1. Accédez au site Web d'Anaconda en suivant
   L'URL: https://www.anaconda.com/download/
2. Choisissez le système d'exploitation cible (Windows, Mac, Linux…)
3. Sélectionnez la version 3.X (à l'heure de l'écriture de ces lignes, c'est la version 3.6 qui est proposée)
4. Choisissez la version 32 ou 64 bits selon la configuration de votre machine.

Aperçu d'Interface anaconda

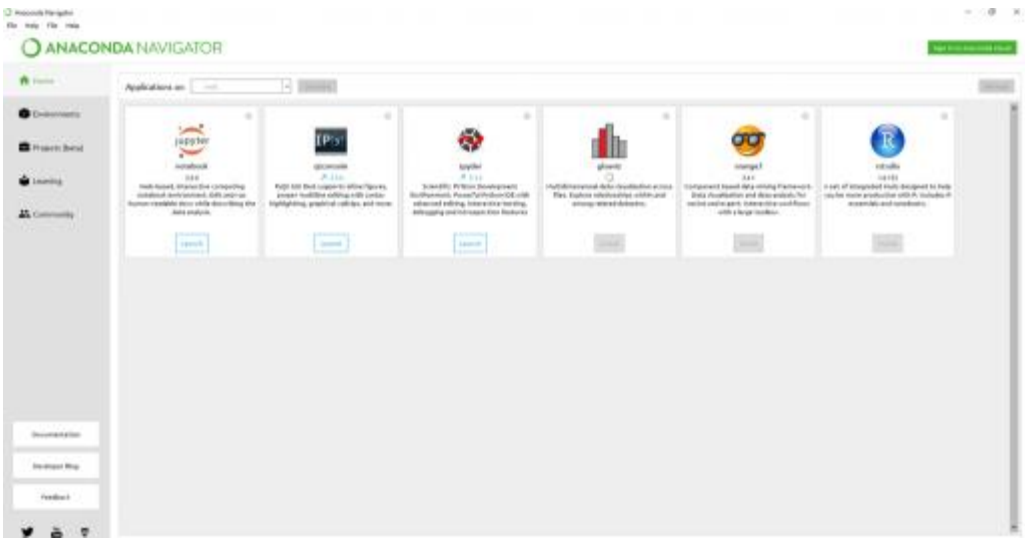


Figure 26 : interface d'Anaconda

Jupyter sur Anaconda 34
Aperçu d'Interface anaconda

➢ Jupyter sur Anaconda :

Dans notre travail nous nous sommes intéressés à la programmation Python sous Jupyter vu la faciliter d'utilisation et l'interface simplifié avec la possibilité de programmer directement dans un navigateur Web :

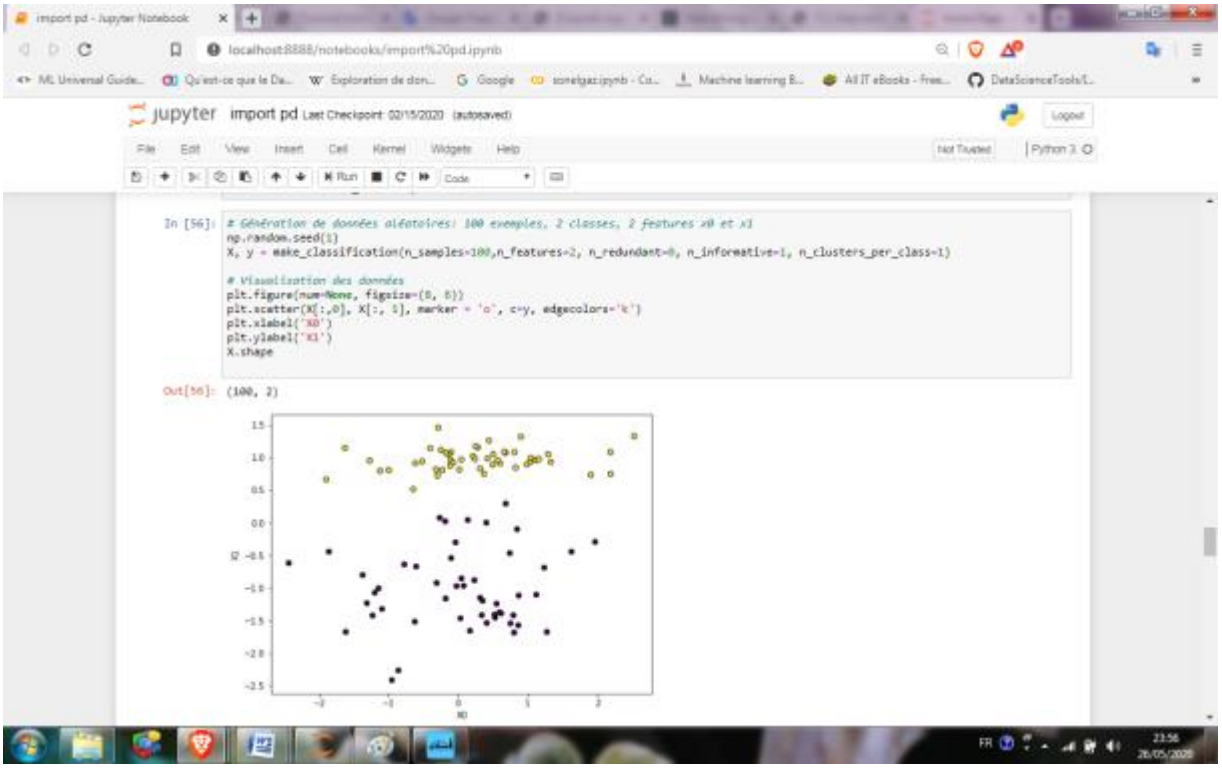


Figure 27 : interface de jupyter

CHAPITRE 4

Implémentation et Résultats

IV. Implémentation et résultats

IV.1. Introduction

Pour implémenter une solution machine Learning, nous avons opté pour une compétition **ZINDI** (https://zindi.africa/) qui est la première plateforme de compétition de science des données en Afrique. **Zindi** héberge tout un écosystème de science des données composé de scientifiques, d'ingénieurs, d'universitaires, d'entreprises, d'ONG, de gouvernements et d'institutions axés sur la résolution des problèmes les plus urgents de l'Afrique.

Zindi travaille avec des entreprises, des organisations à but non lucratif et des institutions gouvernementales pour développer, gérer et préparer des défis basés sur les données. Les solutions sont classées automatiquement en fonction de la précision obtenue. Que vous testiez les données scientifiques pour la première fois ou que vous tentiez de résoudre un problème commercial persistant avec les données, **Zindi** aide les organisations à repousser leurs limites créatives à un coût abordable.

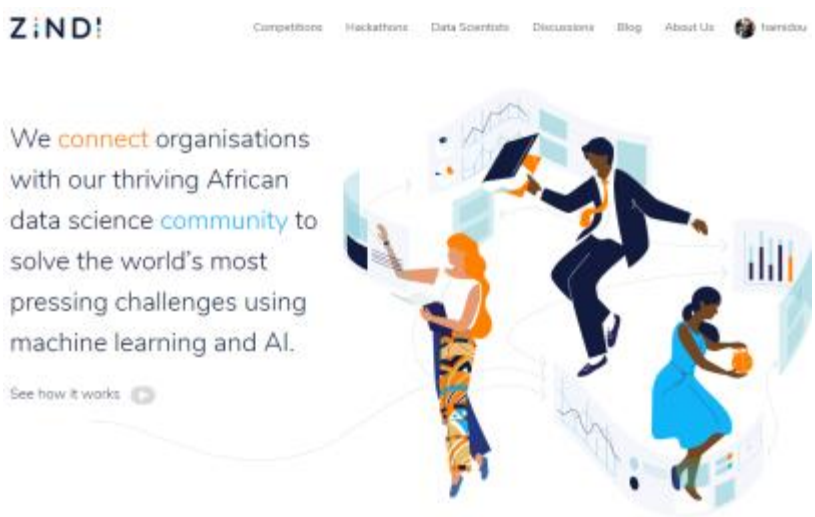


Figure 28 : Page d'accueil ZINDI

IV.2. Expérimentations et résultats

Nous avons opté pour la compétition lancée par **ZINDI** en collaboration avec la **STEG** (Société Tunisienne d'Electricité et de Gaz) intitulé : « Détection de fraude dans la consommation d'électricité et de gaz ».

La **STEG** est une entreprise publique et non administrative, elle est chargée de livrer l'électricité et le gaz à travers la **Tunisie**. L'entreprise a subi d'énormes pertes de l'ordre de 200 millions de dinars tunisiens en raison de manipulations frauduleuses de compteurs par les consommateurs.

En utilisant l'historique de facturation du client, le défi a pour objectif de détecter et de reconnaître les clients impliqués dans des activités frauduleuses. La solution améliorera les revenus de l'entreprise et réduira les pertes causées par de telles activités frauduleuses.

L'événement principal de la compétition est « AI Hack Tunisia 2019 » (du 20 au 22 septembre 2019) qui est le plus grand hackathon ML/AI jamais organisé en Tunisie, en Afrique et dans la région du Moyen-Orient et de l'Afrique du Nord (MENA). Après la fin de la compétition en Tunisie, ZINDI a décidé de mettre el ligne et à titre gratuit tous les Dataset de l'événement afin de permettre aux étudiants et chercheurs du monde entier d'appliquer leurs propres méthodes et techniques de l'Intelligence Artificielle et du Machine Learning et par la suite comparer les résultats en ligne (classement). Dans notre cas, pour le challenge « Détection de fraude dans la consommation d'électricité et de gaz » le site affiche un total de 512 Data-Scientists inscrits et 114 au classement jusqu'au **29/08/2020** (voir la Figure 29)

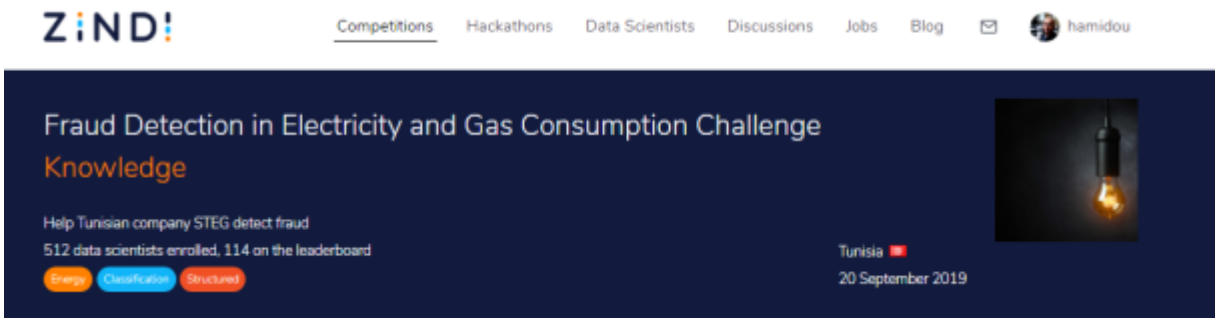


Figure 29 : Compétitions ZINDI avec le nombre de candidats.

IV.2.1. Définition des données (Dataset)

Les données fournies par la **STEG** sont composées de deux fichiers d'apprentissage (**Train**). Le premier est composé de données clients et le second contient l'historique de facturation depuis 2005 :

- Client_train.csv - Informations client dans la partie apprentissage contenant un total de **135.493,00** clients.
- Invoice_train.csv –Les différentes Factures des clients appartenant à la base d'apprentissage avec un total de **4.476.749,00** lignes de données.

Un autre dossier (**Test**) contenant aussi deux Fichiers :

- Client_test.csv - Informations client pour la population test avec un total de **58.069,00** clients.
- Invoice_test.csv - Factures client dans l'ensemble de test (**1.939.730,00**).

Et un fichier de soumission des résultats Test dans le chalenge :

✓ SampleSubmission.csv : est un exemple de ce à quoi devrait ressembler votre fichier de soumission. L'ordre des lignes n'a pas d'importance, mais les noms des ID doivent être corrects. La colonne "cible" est votre prédiction (classe résultat).

Définitions des variables (Attributs) :

❖ **Client**:

- Client_id: identifiant unique du client.
- District: District où se trouve le client.
- Client_catg: catégorie du client.
- Région: zone où se trouve le client.
- Date de création: date de création de l'abonné.
- Target: fraude: 1, pas fraude: 0

❖ **Données de facturation :**

- Client_id: identifiant unique du client.
- Invoice_date: date de la facture.
- Tarif_type: Type de taxe.
- Counter_number: Le numéro de compteur.
- Counter_statue: prend jusqu'à 5 valeurs telles que travailler correctement, ne pas fonctionner, en attente statue…etc.
- Counter_code : modèle du compteur.
- Reading_remarque: note que l'agent STEG prend lors de sa visite chez le client (par exemple: si le compteur montre quelque chose de mal, l'agent donne un mauvais score).
- Counter_coefficient: Un coefficient supplémentaire à ajouter lorsque la consommation standard est dépassée.
- Consommation_level_1: consommation première tranche.
- Consommation_level_2: consommation deuxième tranche.
- Consommation_level_3: consommation troisième tranche.
- Consommation_level_4: consommation quatrième tranche.
- Old_index: Ancien index.
- New_index: nouvel index.
- Month_number: Numéro du mois.
- Counter_type: Type de compteur (ELEC/GAZ).

IV.2.2. Réalisation (programme source)

L'objectif est de prédire si un client est un fraudeur (variable à prédire) à partir de l'historique de ses factures (historique de consommation, compteur, tarif etc...) (Variables explicatives). Notre méthode de classification est implémentée avec le langage de programmation **Python** sous **Jupyter** dans **ANACONDA** version 3.7, ci-dessous les différentes étapes de notre implémentation :

1. Importation des bibliothèques :

```
import numpy as np
import pandas as pd
import datetime
import gc
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
import warnings
warnings.filterwarnings('ignore')
np.random.seed(4590)
```

- ✓ import numpy as np : NumPy est une extension destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.
- ✓ import pandas as pd : **Pandas** est une bibliothèque Python permettant la manipulation et l'analyse des données.
- ✓ import matplotlib.pyplot : **Pyplot** est une collection de fonctions dans le package de visualisation populaire **Matplotlib**. Ses fonctions manipulent des éléments d'une figure, tels que la création d'une figure, la création d'une zone de traçage, le traçage de lignes, l'ajout d'étiquettes de tracé, etc.
- ✓ fromsklearn.metrics import accuracy_score: fonctions pour le calcul de corrélation entre les labels prévisions et cibles, utile pour l'interprétation car elle nous indique à quel point les prédictions de votre modèle sont bonnes.

2. Chargement des Datasets :

```
train_client=pd.read_csv('r:/train/client_train.csv')
train_invoice=pd.read_csv('r:/train/invoice_train.csv')
test_client=pd.read_csv('r:/train/client_test.csv')
test_invoice=pd.read_csv('r:/train/invoice_test.csv')
sub=pd.read_csv('r:/train/SampleSubmission.csv')
```

Il s'agit d'une lecture des données à partir des fichiers CSV :

- ✓ **train_client=pd.read_csv('r:/train/client_train.csv')** : ce fichier contient les informations d'apprentissage (Train) sur les clients de STEG tel que la localisation(disrict,region), la date de création de l'abonné et le target (0 :non frauduleux ,1 :frauduleux).

train_invoice=pd.read_csv('r:/train/invoice_train.csv') :ce fichier contient l'historique de consommation de chaque client à partir de 2005.

test_client=pd.read_csv('r:/train/client_test.csv') :ce fichier contient les données Test des client sans la classe résultat (label ou bien target).

test_invoice=pd.read_csv('r:/train/invoice_test.csv') :ce fichier contient l'historique de consommation des client Test.

3. Afficher les Dataframe importés

Nous allons présenter les principales commandes pour avoir les détails et informations sur les données chargés (sur les Figures ci-dessous « **Entrée** » représente la commande et « **Out** » le résultat sur **Jupyter**), à savoir :

- ✓ **train_client:**Afficher les cinq premières lignes et les cinq dernières lignes, le nombre de lignes est (**135.492,00**) et le nombre de colonne (6 attributs) du fichier client **:**

Entrée [5]:

```
#Afficher le contenu du fichier importé "train_client"
train_client
```

Out[5]:

	disrict	client_id	client_catg	region	creation_date	target
0	60	train_Client_0	11	101	31/12/1994	0.0
1	69	train_Client_1	11	107	29/05/2002	0.0
2	62	train_Client_10	11	301	13/03/1986	0.0
3	69	train_Client_100	11	105	11/07/1996	0.0
4	62	train_Client_1000	11	303	14/10/2014	0.0
...	...	...	...	...	...	...
135488	62	train_Client_99995	11	304	26/07/2004	0.0
135489	63	train_Client_99996	11	311	25/10/2012	0.0
135490	63	train_Client_99997	11	311	22/11/2011	0.0
135491	60	train_Client_99998	11	101	22/12/1993	0.0
135492	60	train_Client_99999	11	101	18/02/1986	0.0

135493 rows × 6 columns

- ✓ **train_client.info() :**Afficher le détail du fichier importé des clients « train_client » comme le nombre d'enregistrement, le nombre des colonnes, type des champs, la tailles du mémoire utilisé :

Entrée [6]:

```
#Informations sur les colonnes du fichier "train_client"
train_client.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135493 entries, 0 to 135492
Data columns (total 6 columns):
disrict          135493 non-null int64
client_id        135493 non-null object
client_catg      135493 non-null int64
region           135493 non-null int64
creation_date    135493 non-null object
target           135493 non-null float64
dtypes: float64(1), int64(3), object(2)
memory usage: 6.2+ MB
```

- ✓ **train_invoice**: Afficher les cinq premières lignes et les cinq dernières lignes du fichier des factures, le nombre des lignes (**4.476.749,00**) et le nombre des colonnes(16) :

Entrée [8]:
```
#Afficher le contenu du fichier importé "train_invoice"
train_invoice
```

Out[8]:

	client_id	invoice_date	tarif_type	counter_number	counter_statue	counter_code	reading_remarque	counter_coefficient	consommati
0	train_Client_0	2014-03-24	11	1335667	0	203	8	1	
1	train_Client_0	2013-03-29	11	1335667	0	203	6	1	
2	train_Client_0	2015-03-23	11	1335667	0	203	8	1	
3	train_Client_0	2015-07-13	11	1335667	0	207	8	1	
4	train_Client_0	2016-11-17	11	1335667	0	207	9	1	
...	...	...	...	...	...	...	...	...	
4476744	train_Client_99998	2005-08-19	10	1253571	0	202	9	1	
4476745	train_Client_99998	2005-12-19	10	1253571	0	202	6	1	
4476746	train_Client_99999	1996-09-25	11	560948	0	203	6	1	
4476747	train_Client_99999	1996-05-28	11	560948	0	203	6	1	
4476748	train_Client_99999	1996-01-25	11	560948	0	203	6	1	

4476749 rows × 16 columns

- ✓ **test_client.head() et test_client.info() :**Afficher les 5 premières lignes du fichier « test_client » et le détails du fichier tel que(le nombre de ligne et de colonne ,type des colonnes, mémoire utilisé) :

Entrée [9]:
```
#Afficher l'entete du fichier importé "test_client"
test_client.head()

```

Out[9]:

	disrict	client_id	client_catg	region	creation_date
0	62	test_Client_0	11	307	28/05/2002
1	69	test_Client_1	11	103	06/08/2009
2	62	test_Client_10	11	310	07/04/2004
3	60	test_Client_100	11	101	08/10/1992
4	62	test_Client_1000	11	301	21/07/1977

Entrée [11]:
```
#Afficher le détails du fichier importé "test_client"
test_client.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58069 entries, 0 to 58068
Data columns (total 5 columns):
disrict          58069 non-null int64
client_id        58069 non-null object
client_catg      58069 non-null int64
region           58069 non-null int64
creation_date    58069 non-null object
dtypes: int64(3), object(2)
memory usage: 2.2+ MB
```

✓ **test_invoice.head()** et **test_invoice.info() :** Afficher le fichier importé « test_invoice » et des informations sur ce fichier :

```
Entrée [12]:
1 #Afficher l'entete du fichier importé "test_invoice"
2 test_invoice.head()
```

Out[12]:

	client_id	invoice_date	tarif_type	counter_number	counter_statue	counter_code	reading_remarque
0	test_Client_0	2018-03-16	11	651208	0	203	8
1	test_Client_0	2014-03-21	11	651208	0	203	8
2	test_Client_0	2014-07-17	11	651208	0	203	8
3	test_Client_0	2015-07-13	11	651208	0	203	9
4	test_Client_0	2016-07-19	11	651208	0	203	9

```
Entrée [15]:
1 #Afficher le détails du fichier importé "test_invoice"
2 test_invoice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1939730 entries, 0 to 1939729
Data columns (total 16 columns):
client_id               object
invoice_date            object
tarif_type              int64
counter_number          int64
counter_statue          int64
counter_code            int64
reading_remarque        int64
counter_coefficient     int64
consommation_level_1    int64
consommation_level_2    int64
consommation_level_3    int64
consommation_level_4    int64
old_index               int64
new_index               int64
months_number           int64
counter_type            object
dtypes: int64(13), object(3)
memory usage: 236.8+ MB
```

✓ **sub.head() :** Afficher le modèle de fichier de soumission sur Zindi et qui doit contenir les clients et le taux de fraude de chaque client selon l'exemple :

```
Entrée [13]:
1 sub.head()
```

Out[13]:

	client_id	target
0	test_Client_0	0.957281
1	test_Client_1	0.996425
2	test_Client_10	0.612359
3	test_Client_100	0.776933
4	test_Client_1000	0.571046

4. Jointure fichiers Clients avec Factures

Dans cette partie on doit lier les deux fichiers « client_train » et « invoice_train » afin d'avoir les 21 attributs (=5 client_train + 16 invoice_train avec un attribut en commun) dans une seule ligne et pour chaque client dans le but de représenter graphiquement les données :

Entrée [17]:
```
#joiture des données client_train et invoice_train
data = pd.merge(train_client,train_invoice, on='client_id', how='left')
```

Entrée [18]:
```
#Afficher le résultat de la jointure
data
```

Out[18]:

	disrict	client_id	client_catg	region	creation_date	target	invoice_date	tarif_type	counter_number	counter_statue	.
0	60	train_Client_0	11	101	31/12/1994	0.0	2014-03-24	11	1335667	0	.
1	60	train_Client_0	11	101	31/12/1994	0.0	2013-03-29	11	1335667	0	.
2	60	train_Client_0	11	101	31/12/1994	0.0	2015-03-23	11	1335667	0	.
3	60	train_Client_0	11	101	31/12/1994	0.0	2015-07-13	11	1335667	0	.
4	60	train_Client_0	11	101	31/12/1994	0.0	2016-11-17	11	1335667	0	.
...	...	...	...	...	...	...	...	...	...	...	.
4476744	60	train_Client_99998	11	101	22/12/1993	0.0	2005-08-19	10	1253571	0	.
4476745	60	train_Client_99998	11	101	22/12/1993	0.0	2005-12-19	10	1253571	0	.
4476746	60	train_Client_99999	11	101	18/02/1986	0.0	1996-09-25	11	560948	0	.
4476747	60	train_Client_99999	11	101	18/02/1986	0.0	1996-05-28	11	560948	0	.
4476748	60	train_Client_99999	11	101	18/02/1986	0.0	1996-01-25	11	560948	0	.

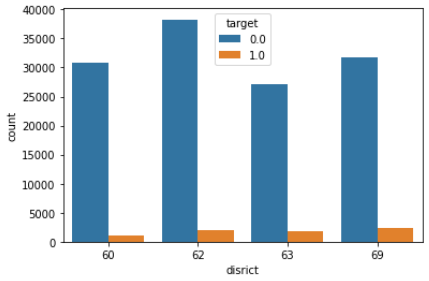
4476749 rows × 21 columns

5. Représentation graphique des données

Dans cette partie nous avons étudié visuellement la dispersion des clients par rapport aux différents attributs (variables) dans le but de comprendre le rôle de chaque variable :

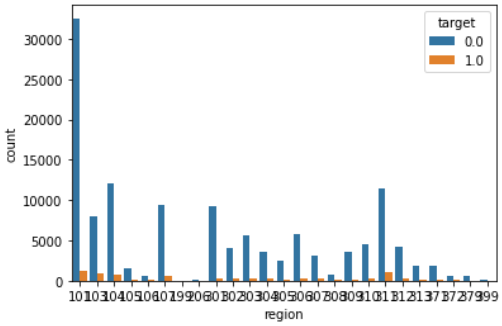
- ✓ **Le nombre des clients (frauduleux (=1), non-frauduleux(=0)) par « District » :**

Entrée [0]:
```
#Nombre  des clients frauduleux et non frauduleux par disrict.
ax = sns.countplot(x="disrict", hue="target",data=train_client)
```



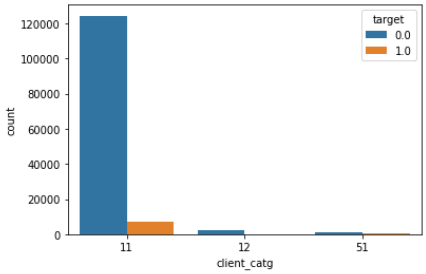
- ✓ **Le nombre des clients (frauduleux(=1), non-frauduleux(=0)) par « Région » :**

Entrée [0]:
```
#Nombre  des clients frauduleux et non frauduleux par région.
ax = sns.countplot(x="region", hue="target",data=train_client)
```



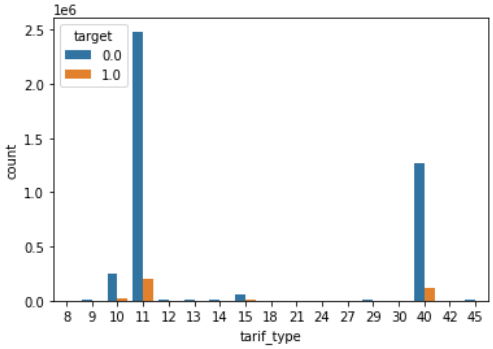
✓ **Le nombre des clients (frauduleux(=1), non-frauduleux(=0)) par « catégorie du client » :**

```
Entrée [0]:
1 #Nombre des clients frauduleux et non frauduleux par catégorie du client.
2 ax = sns.countplot(x="client_catg", hue="target",data=train_client)
```



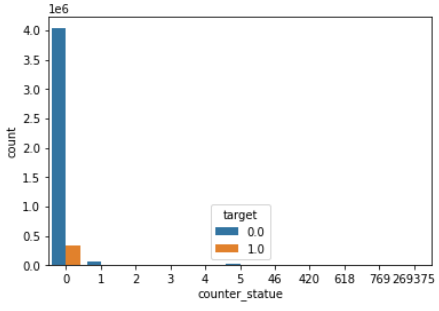
✓ **Le nombre des clients (frauduleux(=1) , non-frauduleux(=0)) par « tarif » :**

```
Entrée [0]:
1 #Nombre des clients frauduleux et non frauduleux par tarif.
2 ax = sns.countplot(x="tarif_type", hue="target",data=data)
```



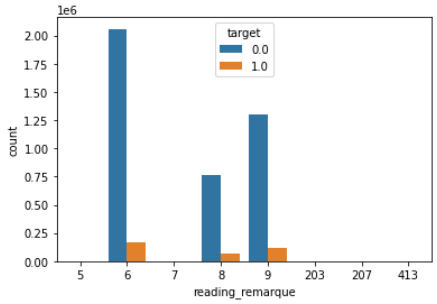
✓ **Le nombre des clients (frauduleux (=1), non-frauduleux(=0)) par « Statutdu Compteur » :**

```
Entrée [0]:
1 #Nombre des clients frauduleux et non frauduleux par statu du compteur.
2 ax = sns.countplot(x="counter_statue", hue="target",data=data)
```



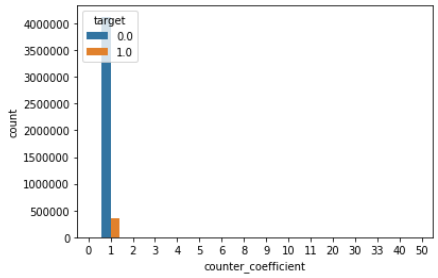
✓ **Le nombre des clients (frauduleux(=1) , non-frauduleux(=0)) par « Remarque sur le Compteur » :**

```
Entrée [0]:
1 #Nombre des clients frauduleux et non frauduleux par champ reading remarque.
2 ax = sns.countplot(x="reading_remarque", hue="target",data=data)
```



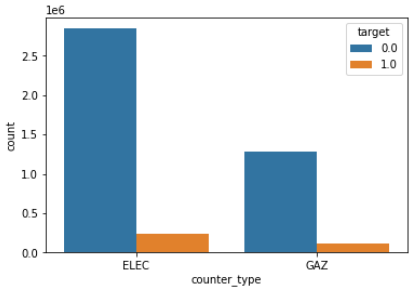
✓ **Le nombre des clients (frauduleux(=1), non-frauduleux(=0)) par « coefficient » :**

```
Entrée [19]:
1 #Nombre des clients frauduleux et non frauduleux par coef compteur.
2 ax = sns.countplot(x="counter_coefficient", hue="target",data=data)
```



✓ **Le nombre des clients (frauduleux(=1), non-frauduleux(=0)) par « type de compteur » :**

```
Entrée [0]:
1 #Nombre des clients frauduleux et non frauduleux par type du compteur(E/G).
2 ax = sns.countplot(x="counter_type", hue="target",data=data)
```



6. Préparation des Données

Cette étape est connue sous le nom de Codage des données, il s'agit de convertir les attributs qualitatifs en valeur quantitatifs afin d'utiliser les méthodes et les algorithmes de calculs.

- ✓ Formater la date de création de l'abonné des fichiers importés (train_invoice, test_invoice, train_client, test_client) en deux champs **year** et **month**de type entier :

Entrée [3]:
```
#Convertir la date de création en entier de l'abonné (year,month)
for df in [train_invoice,test_invoice]:
    df['invoice_date'] = pd.to_datetime(df['invoice_date'])
    df['year'] = df['invoice_date'].dt.year
    df['month'] = df['invoice_date'].dt.month
```

Entrée [4]:
```
#Convertir la date de facture en entier (year,month)
for df in [train_client,test_client]:
    df['creation_date'] = pd.to_datetime(df['creation_date'])
    df['year'] = df['creation_date'].dt.year
    df['month'] = df['creation_date'].dt.month
```

- ✓ Convertir le type des compteurs (**ELEC**,**GAZ**) en entier (0,1) :

Entrée [5]:
```
#Convertion le type du compteur en entier (0,1)
d={"ELEC":0,"GAZ":1}
train_invoice['counter_type']=train_invoice['counter_type'].map(d)
d={"ELEC":0,"GAZ":1}
test_invoice['counter_type']=test_invoice['counter_type'].map(d)
```

- ✓ Définir la fonction d'agrégation« aggs » afin de regrouper les factures par client et calculer les variables (somme, maximum, minimum, et moyenne, nunique) pour les champs concernés en effectuant un changement d'échelle (augmentation du nombre des attributs / dimension) dans le but d'accroitre le degré de discrimination des attributs :

Entrée [6]:
```
#Définir la fonction d'aggrégation avec les paramètres (sum,max,min,mean,nunique) sur les colonnes ci-dessous
aggs = {}
aggs['consommation_level_1'] = ['sum','max','min','mean']
aggs['consommation_level_2'] = ['sum','max','min','mean']
aggs['consommation_level_3'] = ['sum','max','min','mean']
aggs['consommation_level_4'] = ['sum','max','min','mean']

aggs['month'] = ['mean', 'max', 'min','sum']
aggs['year'] = ['nunique','max','min','mean','sum']

aggs['months_number'] = ['max','min','mean','sum']
aggs['reading_remarque'] = ['max','min','mean','sum']
aggs['counter_coefficient'] = ['max', 'min','mean','sum']
aggs['counter_number'] = ['nunique','max','min']
aggs['counter_type'] = ['nunique','mean','sum']
aggs['counter_statue'] = ['nunique','max','min','sum']
aggs['tarif_type'] = ['nunique','max','min','sum']
aggs['counter_code'] = ['nunique','max','mean','min']


aggs['old_index'] = ['mean','max','min','sum']
aggs['new_index'] = ['mean','max','min','sum']
```

✓ Appliquer la fonction d'agrégation avec groupby sur les fichiers importés « invoice_train » et « invoice_test » :

```
#Appliquer l'aggrégation
agg_train = train_invoice.groupby(['client_id']).agg(aggs)
agg_test = test_invoice.groupby(['client_id']).agg(aggs)

```

✓ Afficher résultat d'agrégation appliqué sur le fichier **train_invoice**

Entrée [54]:
```
#afficher résultat d'aggrégation
agg_train
```

Out[54]:

	consommation_level_1				consommation_level_2				consommation_level_3		...
	sum	max	min	mean	sum	max	min	mean	sum	max	...
client_id											
train_Client_0	12334	1200	38	352.400000	370	186	0	10.571429	0	0	...
train_Client_1	20629	1207	190	557.540541	0	0	0	0.000000	0	0	...
train_Client_10	14375	2400	188	798.611111	682	682	0	37.888889	0	0	...
train_Client_100	24	15	0	1.200000	0	0	0	0.000000	0	0	...
train_Client_1000	9292	800	124	663.714286	1468	400	0	104.857143	1643	800	...
...	...	...	...	...	...	...	...	...	...	...	...
train_Client_99995	139	139	0	1.957746	0	0	0	0.000000	0	0	...
train_Client_99996	7620	800	0	185.853659	31	31	0	0.756098	0	0	...
train_Client_99997	9831	1075	33	273.083333	0	0	0	0.000000	0	0	...
train_Client_99998	600	400	200	300.000000	141	135	6	70.500000	0	0	...
train_Client_99999	1378	603	259	459.333333	0	0	0	0.000000	0	0	...

135493 rows × 63 columns

✓ Afficher les champs calculés

Entrée [53]:
```
#afficher les champs calculés
agg_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 135493 entries, train_Client_0 to train_Client_99999
Data columns (total 63 columns):
(consommation_level_1, sum)     135493 non-null int64
(consommation_level_1, max)     135493 non-null int64
(consommation_level_1, min)     135493 non-null int64
(consommation_level_1, mean)    135493 non-null float64
(consommation_level_2, sum)     135493 non-null int64
(consommation_level_2, max)     135493 non-null int64
(consommation_level_2, min)     135493 non-null int64
(consommation_level_2, mean)    135493 non-null float64
(consommation_level_3, sum)     135493 non-null int64
(consommation_level_3, max)     135493 non-null int64
(consommation_level_3, min)     135493 non-null int64
(consommation_level_3, mean)    135493 non-null float64
```

✓ Enlever les parenthèses des champs calculés

```
#Enlever les parenthèses sur les variables calculés
agg_train.columns = ['_'.join(col).strip() for col in agg_train.columns.values]
agg_test.columns = ['_'.join(col).strip() for col in agg_test.columns.values]
```

✓ Faire la jointure à gauche entre **train_client** et **agg_train** avec la colonne en commun **client_id** et **test_client** et **agg_test** avec la colonne en commun **client_id**

```
##Faire la joiture à gauche entre client_train et invoice_train et client_test et invoice_test
train = pd.merge(train_client,agg_train, on='client_id', how='left')
test = pd.merge(test_client,agg_test, on='client_id', how='left')

```

✓ Récupérer la colonne **target** de la base Train sur une matrice à une dimension afin de l'utiliser directement lors de la classification :

```
#Recupere la colonne target sur une matrice à une dimention
target=train['target']

```

✓ Convertir la colonne**client_id**sur train et test en entier :

```
#conversion le champ client_id sur train et test en entier
from sklearn import preprocessing
lbl = preprocessing.LabelEncoder()
lbl.fit(list(train['client_id'].values))
train['client_id'] = lbl.transform(list(train['client_id'].values))
#test
lbl.fit(list(test['client_id'].values))
test['client_id'] = lbl.transform(list(test['client_id'].values))
```

7. Le Modèle d'Apprentissage (Classification)

L'objectif de cette partie est d'implémenter un algorithme de classification pour la séparation des clients en deux classes (frauduleux / non-frauduleux). Notre choix c'est porté sur l'approche des arbres de décision. Un modèle idéal est un modèle qui est bon à la fois en termes de précision, de généralisation et de prédiction. Cela nous amène à BoostingAlgorithms développée en 1989, la famille d'algorithmes de Boosting s'est améliorée au fil des années et parmi eux l'algorithme **XGBoost**.

✓ **Fromxgboost import XGBClassifier :**

```
Entrée [20]:  1  import xgboost as xgb
```

8. Séparation des données :

La bonne manière de mesurer la performance de votre modèle de Machine Learning est de tester celui-ci sur des données qui n'ont pas servi à l'entraînement. On divise ainsi notre Train Dataset aléatoirement en deux parties avec un rapport 80/20 :

• Train set (80%), qui permet à la machine d'entraîner un modèle. utilisé pour s'adapter au modèle d'apprentissage automatique.

• Test set (20%), qui permet d'évaluer la performance du modèle. utilisé pour évaluer le modèle d'apprentissage automatique d'ajustement.

Pour créer un Train set et Test set à partir de notre Dataset, on utilise la fonction **train_test_split** de **Sklearn** :

```
#Sépparer train en X_train et y_train et  X_test et y_test avec
##un pourcentage de 70% pour train et 30 % pour test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train, target, test_size=0.2)
```

La bibliothèque d'apprentissage automatique scikit-learn Python fournit une implémentation de la procédure d'évaluation de fractionnement train-test via la fonction train_test_split ().

La fonction prend un ensemble de données chargé en entrée et renvoie l'ensemble de données divisé en deux sous-ensembles.

Vous diviser votre ensemble de données d'origine en colonnes d' entrée (X) et de sortie (y) puis appeler la fonction en passant les deux tableaux et les diviser de manière appropriée en sous-ensembles de train et de test.

La taille du fractionnement peut être spécifiée via l'argument « test_size » qui prend un nombre de lignes (entier) ou un pourcentage (flottant) de la taille de l'ensemble de données entre 0 et 1.

9. Paramétrage du modèle :

La recherche de grille est le processus consistant à effectuer un réglage hyper paramétrique afin de déterminer les valeurs optimales pour un modèle de donné. Ceci est important car les performances de l'ensemble du modèle sont basées sur les valeurs d'hyper paramètre spécifiées.

Pour cela Il y a des bibliothèques à implémenter, comme celle **GridSearchCV** de la **sklearn** bibliothèque, afin d'automatiser ce processus :

```python
# Recherche la meilleur configuration pour le modèle XGBoost
from pandas import read_csv
from xgboost import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold


# split data into X and y
X = X_train
y = y_train
#grid search
model = XGBClassifier()
n_estimators = [50, 100, 150, 200]
max_depth = [2, 4, 6, 8]
learning_rate=[0.001,0.01,0.1]
param_grid = dict(max_depth=max_depth, n_estimators=n_estimators,learning_rate=learning_rate)
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=7)
grid_search = GridSearchCV(model, param_grid, scoring="roc_auc", n_jobs=-1, cv=kfold, verbose=1)
grid_result = grid_search.fit(X, y)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  6.7min
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed: 44.3min
[Parallel(n_jobs=-1)]: Done 240 out of 240 | elapsed: 60.5min finished
```

```
Best: 0.885715 using {'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 150}
0.703149 (0.035032) with: {'learning_rate': 0.001, 'max_depth': 2, 'n_estimators': 50}
0.726979 (0.011322) with: {'learning_rate': 0.001, 'max_depth': 2, 'n_estimators': 100}
0.726981 (0.011324) with: {'learning_rate': 0.001, 'max_depth': 2, 'n_estimators': 150}
0.727228 (0.011165) with: {'learning_rate': 0.001, 'max_depth': 2, 'n_estimators': 200}
0.765935 (0.012763) with: {'learning_rate': 0.001, 'max_depth': 4, 'n_estimators': 50}
0.774563 (0.009322) with: {'learning_rate': 0.001, 'max_depth': 4, 'n_estimators': 100}
```

Nous pouvons définir une grille du nombre d'arbres (n_estimators) et des tailles d'arbres (max_depth) et le taux d'apprentissage (learning_rate) à évaluer en définissant une grille comme :

n_estimators = [50,100,150, 200]

max_depth = [2,4, 6, 8]

learning_rate=[0.001,0.01,0.1]

param_grid = dict(max_depth = max_depth, n_estimators
=n_estimators,learning_rate=learning_rate)

Et puis évaluez chaque combinaison de paramètres en utilisant la validation croisée 10 fois comme:

kfold = StratifiedKFold (n_splits = 10, shuffle = True, random_state = 7)

grid_search = GridSearchCV (modèle, param_grid, scoring = "neg_log_loss", n_jobs = -1, cv = kfold, verbose = 1)

result = grid_search.fit (X, Y).

Nous pouvons ensuite revoir les résultats pour déterminer la meilleure combinaison et les tendances générales dans la variation des combinaisons de paramètres.

Il s'agit de la meilleure pratique lors de l'application de XGBoost à vos propres problèmes. Les paramètres à prendre en compte pour le réglage sont:

-Le taux d'apprentissage ou le rétrécissement (learning_rate dans XGBoost) doit être défini sur 0.1 ou moins, et des valeurs plus petites nécessiteront l'ajout de plus d'arbres.

-La profondeur des arbres (max_depth dans XGBoost) doit être configurée dans la plage de 2 à 8, où peu d'avantages sont observés avec les arbres plus profonds.

-L'échantillonnage en ligne (sous-échantillon dans XGBoost) doit être configuré dans la plage de 30% à 80% de l'ensemble de données d'apprentissage, et comparé à une valeur de 100% pour aucun échantillonnage.

10. Redéfinir le modèle XGBClassifier() avec les meilleurs paramètres trouvés dans la phase précédente

Entrée [19]:
```
#Redéfinir le modèle avec les meilleur paramètres (nombre d'arbre,profondeur d'arbre et taux d'apprentissage)
clf=XGBClassifier(learning_rate=0.1,max_depth=6,n_estimators=200)
```

11. **Entrainer le modèle** :

La classification est un processus à deux étapes : une étape d'apprentissage (entrainement) et une étape de classification (utilisation). Ainsi nous utilisons la fonction « fit » sur l'ensemble d'apprentissage X_train et Y_train :

```
#Faire entrainer avec le modèle clf
clf.fit(X_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.1, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=200, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
              validate_parameters=False, verbosity=None)
```

12. Prédiction et Evaluation sur l'échantillon test :

A partir du modèle construit dans la première étape (entrainement) l'étape de prédiction est réalisée afin de classer les nouvelles données. Prédiction sur l'échantillon Test :

Entrée [44]:
```
#Prédiction sur l'échantillon X_test
y_pred = clf.predict(X_test)
print(y_pred)
```

```
[0. 0. 0. ... 0. 0. 0.]
```

Afin de mesurer les performances du modèle il est nécessaire d'importer« **metrics » :**

```
#Importation metrics utilisés pour les mésures de performances
from sklearn import metrics
```

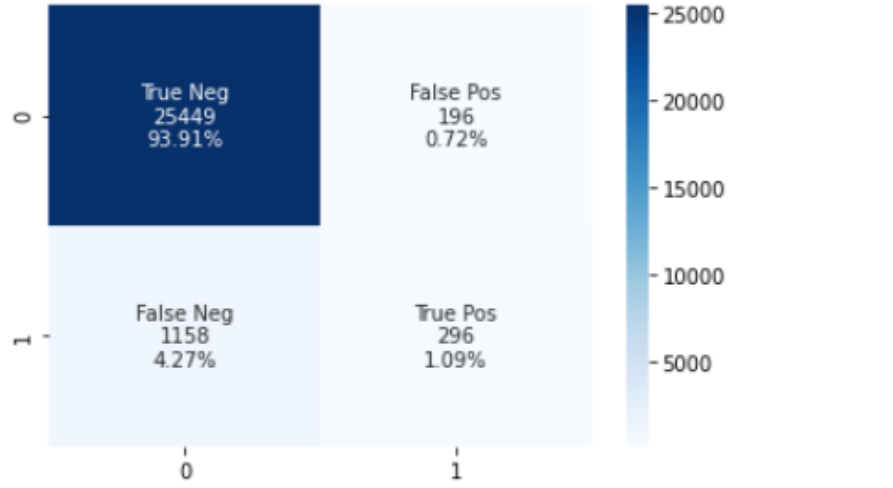
Ensuite on va calculer la Matrice de confusion :

```
[52] from sklearn.metrics import confusion_matrix
     cf_matrix = confusion_matrix(y_test, y_pred)
     print(cf_matrix)

     [[25449   196]
      [ 1158   296]]
```

Représentation graphique de la matrice de confusion :

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff031ff01d0>
```



Calculer la précision (**P=0.95**) du modèle à partir de la matrice de confusion :

```
[51] #Taux de précision
     from sklearn.metrics import accuracy_score
     print(accuracy_score(y_test, y_pred))

     0.9500350566441567
```

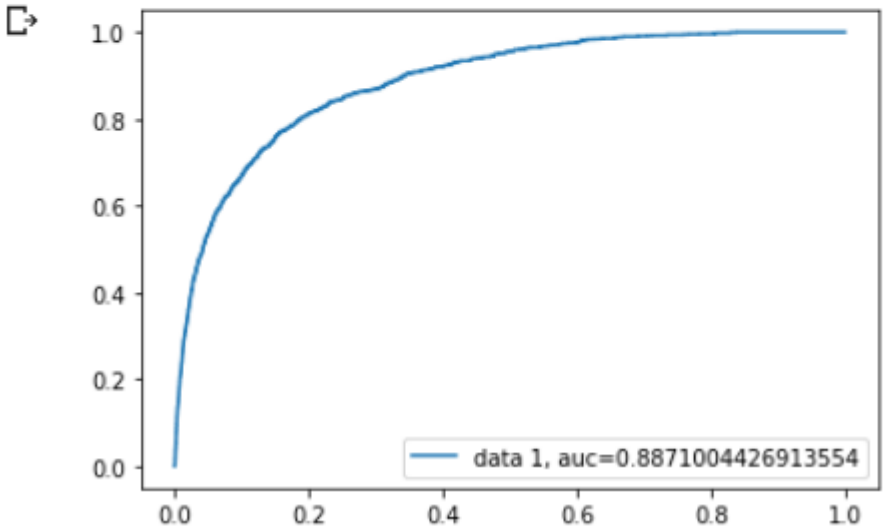
Taux d'erreur (1-P) : **Err**=1-0.95=0,049

La sensitivité **Sv** et la spécificité **Sp** :

$$\begin{cases} Sv = \dfrac{VP}{VP+FN} = \dfrac{296}{296+1158} = \mathbf{0.2035} \\ Sp = \dfrac{VN}{VN+FP} = \dfrac{25449}{25449+196} = \mathbf{0.9923} \end{cases}$$

Représentation ROC et calcul de métrique (AUC=0.88) :

```
[61] from sklearn import metrics
     y_pred_proba = clf.predict_proba(X_test)[::,1]
     fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
     auc = metrics.roc_auc_score(y_test, y_pred_proba)
     plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
     plt.legend(loc=4)
     plt.show()
```



13. Généralisation du modèle :

Une fois le modèle est testé et mesuré avec la partie Train des données, en passe à l'utilisation du classifieur avec des nouveaux exemples (Test) après avoir entrainé le modèle avec la totalité des données (Train, Target).

Entrée [17]:
```
#Faire entrainer le modèle clf avec la totalité des données
clf.fit(train,target)
```

Out[17]:
```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.1, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1,
              objective='binary:logistic', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method=None,
              validate_parameters=False, verbosity=None)
```

Entrée [19]:
```
#Tester le modèle sur des nouveaux exemples de zindi
pred = clf.predict_proba(test)
print(pred)
```

```
[[0.97148526 0.02851473]
 [0.8627589  0.13724114]
 [0.9716921  0.0283079 ]
 ...
 [0.34167135 0.65832865]
 [0.99509656 0.00490347]
 [0.9301362  0.06986379]]
```

Pour la prédiction on a utilisé la fonction predict_proba() pour avoir la probabilité pour chaque abonné d'avoir fraudé. Dans la figure ci-dessous, la première colonne représente l'identifiant du client, la deuxième colonne la probabilité d'appartenance à la classe « 0 » (non-frauduleux) et la troisième colonne représente la probabilité d'appartenance à la classe « 1 » (frauduleux)

```
Entrée [20]:
1  pred = pd.DataFrame(pred)
2  print(pred)
```

```
              0         1
0      0.971485  0.028515
1      0.862759  0.137241
2      0.971692  0.028308
3      0.994575  0.005425
4      0.906532  0.093468
...         ...       ...
58064  0.998313  0.001687
58065  0.972537  0.027463
58066  0.341671  0.658329
58067  0.995097  0.004903
58068  0.930136  0.069864

[58069 rows x 2 columns]
```

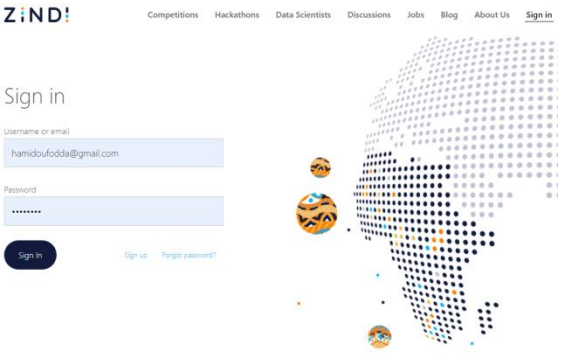
14. Générer le fichier de soumission pour la plateforme ZINDI :

Dans cette étape en crée le fichier de soumission pour l'envoi sur Zindi en convertissant le résultat de prédiction en fichier csv.

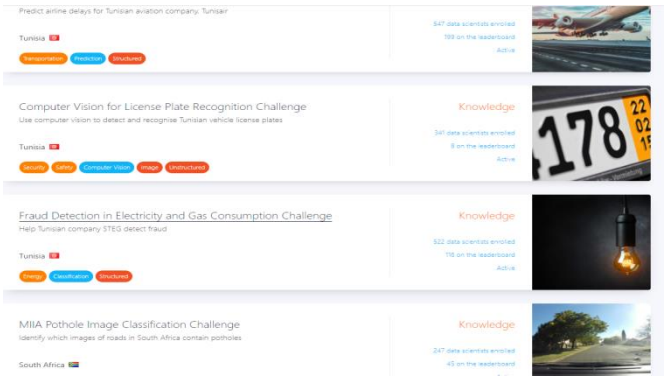
```
submission = pd.DataFrame({
        "client_id": sub["client_id"],
        "target": pred[1]
    })
submission.to_csv('r:steg520.csv', index=False)
```

15. Envoi résultat sur le site Zindi :

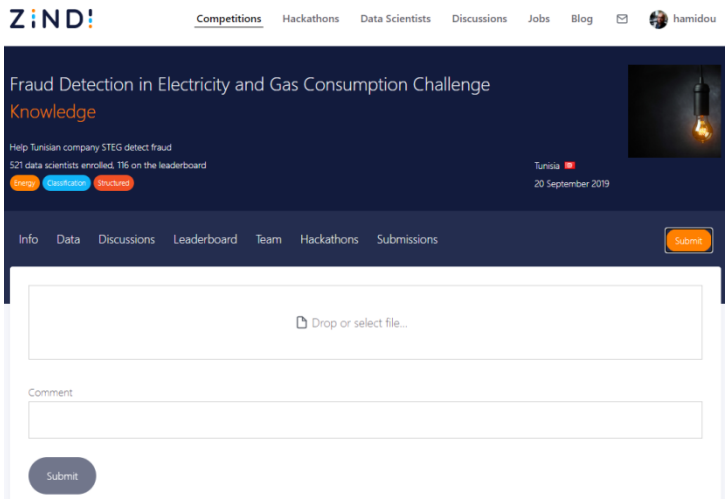
- ✓ Connecter avec mon compte sur Zindi



- ✓ Aller sur la liste des compétitions et sélectionner le challenge dont vous voulez participer (Fraud Detection in Electricity and Gas Consumption challenge):



✓ Cliquer sur **Submit** pour sélectionner votre fichier de soumission :



✓ Voir votre score une fois le fichier est soumis (dans ce cas est à **88,79%**):

	ID	SUBMITTED	FILE	COMMENT	SCORE
✓	tWGjUCLZ	5 months ago	steg69.csv	—	0.885034575452451
✓	fYjwjafa	5 months ago	steg68.csv	—	0.887996908945407
✓	xVgDLkMd	5 months ago	steg67.csv	—	0.886213099409558
✓	zpYJFQpG	5 months ago	steg67.csv	—	0.887141874286803
✓	Y56jU1fP	5 months ago	steg66.csv	—	0.886384446452314
✓	Pqgu8YLM	5 months ago	steg65.csv	—	0.883841264789899
✓	fAdDDNZb	5 months ago	steg64.csv	—	0.883841264789899

✓ Voir votre classement en cliquant sur Leader board : https://zindi.africa/competitions/ai-hack-tunisia-4-predictive-analytics-challenge-1/leaderboard

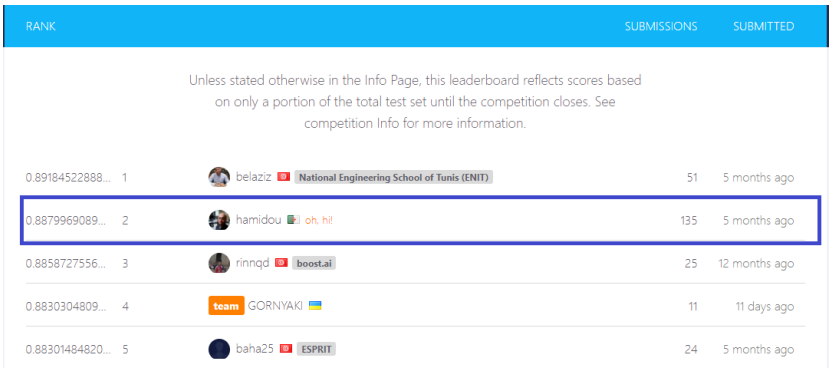


Figure 30 : le classement finale sur Zindi

IV.2.3. Conclusion

Notre approche et méthode de traitement de l'ensemble de données (1 millions d'abonnés dans notre cas) à l'aide des techniques de l'intelligence artificielle et machine Learning, nous a permis d'atteindre des taux de classification très important dans l'étape d'apprentissage(95.00%) et un score de 88.79% dans l'étape test en ligne (ZINDI) sur des nouvelles données avec la 2^{ième}position au classement du challenge international sur un total de 115 participant (au 8/09/2020). Ces résultats nous permettent de participer dans d'autres challenges afin d'améliorer notre approche.

CONCLUSION GENERALE

Nous rappelons que l'objectif de notre travail était d'extraire des connaissances à partir d'une base de données et l'appliquer pour la détection de fraude dans la consommation d'électricité et du gaz, pour cela nous avons utilisés une base de données de la Société Tunisienne d'Electricité et du Gaz disponible en ligne sur la plate-forme **ZINDI.**

Les techniques d'intelligence artificielle et les méthodes de ML peuvent apportés des résultats bénéfique pour ce genre de problème de classification en utilisant un algorithme d'apprentissage supervisé basé sur les arbres de décisions (XGBoost) implémenté via des solutions libres et ouvertes tel que le langage python, la plate-forme anaconda et les bibliothèques du ML telle que scikit-learn…etc.

Notre approche implémentée à l'aide et les méthodes de préparation et classification de l'ensemble de données, environ 1 millions d'abonnés (clients), nous a permis de réaliser des taux de classification élevé dans la partie apprentissage de **95.00%** et un score de **88.79%** dans la partie test avec la deuxième position dans le challenge sur un total de 115 participants classés (au 8/09/2020).

Ces résultats motivant nous encouragent à améliorer encore plus notre méthode et à participer dans d'autre challenge et espérons d'appliquer cette solution au niveau de nos organismes de travail en Algérie.

IV. Bibliographie

IV.1. Les livres

[6] Guillaume Saint-Cirgue (2019), Apprendre le Machine Learning en une semaine, machinelearnia.com, 100 p.

[7] Abdelhamid DJEFFAL 2014, Cours Fouille de données avancée, Université Mohamed Khider – Biskra, 96 p.

IV.2. Les articles

[1] Smyth.P Fayyad U., Piatetsky-shapiro G. From data mining to knowledge discovery in databases. AI Magazine, 17:37–57, 1996.

[9] Kantardzic M. Data mining - concepts, models, methods, and algorithms. IEEE Press, Piscataway, NJ, USA, 2003

IV.3. Mémoires et/ou rapports

[5] DGE (Direction Générale des entreprise) (2019), « Intelligence artificielle - État de l'art et perspectives pour la France », Rapport finale, 333 p.

[8] Brahimi Belgacem (2011), Extraction de connaissances à partir de données incomplètes et imprécises, mémoire de Magister, université de Msila, 123 p.

IV.4. Les sites web.

[2] Microsoft, Tout savoir sur l'intelligence artificielle, 09 février 2018 [consulté le 15/05/2020] Disponible sur : https://experiences.microsoft.fr/business/intelligenceartificielle-ia-business/comprendre-utiliser-intelligence-artificielle/

[3] Valentin - @vblanchot, Des dates et des moments importants dans l'évolution de l'IA, 12 mars 2020 [consulté le 16/05/2020], disponible sur : https://siecledigital.fr/2018/08/20/histoire-intelligence-artificielle/

[4] Loïc Bremme, Le Big Data Magasin IA Cloud et Big data, définition du Big Data, [consulté le 05/07/2020] Disponible sur https://www.lebigdata.fr/definition-big-data

الملخص

تمثل عملنا في استخراج المعرفة من قاعدة بيانات STEG على الأنترنت من أجل الكشف عن الاحتيال في استهلاك الكهرباء والغاز . من أجل ذلك استخدمنا مصنف XGBoost وهو من بين أفضل خوارزميات التصنيفات علم الآلة لهذا النوع من المشاكل وشاهدنا النتائج التي تم الحصول عليها بمعدل دقة و صل إلى 95.00٪ وبدرجة 88.79٪ حيث سمحت لنا بإحتلال المركز الثاني في تحدي زندي (ZINDI) .تدفعنا هذه النتائج المحفزة إلى تعلم المزيد في مجال الذكاء الاصطناعي وتعلم الآلة والمشاركة في تحديات أخرى في المستقبل.

Abstract

Our studywas to extract knowledge from STEG's online data base to detect fraud in electricity and gas consumption. To do this we used the **XGBoost** classifier, which is one of the best classification algorithms for machine learning for this type of problem, and we obtain the results with an accuracy rate of **95.00%** and a score of **88.79%**, which allowed us to take second place in the **Zindi** Challenge. These important results motivate us to learn more about AI and machine learning, and to participate in other challenges in the future.

Résumé

Notre travail consistait à extraire des connaissances à partir d’une base de données en ligne de STEG, afin de détecter les fraudes dans la consommation d'électricité et du gaz. Pour cela nous avons utilisé le classifieur **XGBoost**, qui fait partie des meilleurs algorithmes de classification pour l'apprentissage automatique pour ce type de problème, et nous avons vu les résultats obtenus avec un taux de précision de **95,00%** et un score de **88,79%**, ce qui nous a permis d'occuper la deuxième place du Challenge **Zindi**. Ces résultats stimulants nous motivent à en apprendre davantage dans le domaine de l'IA et de l'apprentissage automatique, et à participer à d'autres défis à l'avenir.