DataCamp Kaggle Compétition IEEE-Fraud-Detection

Auteurs:

Fernanda TCHOUACHEU
Mohamed NIANG
Hypolite CHOKKI

Responsables:

Pr. Agathe GUILLOUX
Dr. Simon BUSSY

Sommaire

1 Présentation du challenge

1.1 Sciences des données et Kaggle

Le sujet de ce challenge tutoré datacamp ieee-fraud-detection s'inscrit dans le domaine de la science des données. Cette discipline s'appuie sur des outils mathématiques,statistiques et informatiques afin de traiter, d'exploiter et d'en extraire des informations utiles ou potentiellement utiles pour la prise de décision. Plus précisément, nous sommes confrontés à des problèmes d'analyse de données, de machine learning et de prédiction. Dans ce rapport, nous détaillerons comment traiter ces différents problèmes et les méthodes utilisées.

Kaggle est une plateforme web organisant des compétitions en data science. Sur cette plateforme, les entreprises proposent des problèmes en science des données et offrent un prix aux participants obtenant les meilleures performances. Elle a été fondée en 2010 par Anthony Goldbloom.
Les participants choisissent donc les challenges auxquels ils veulent participer et tentent de résoudre le problème soumis de la meilleure façon possible, tout cela en compétition avec les autres participants. Pour cela, il est possible de soumettre un fichier afin d'obtenir un score qui nous positionne dans un classement publique (qui regroupe tous les participants). Cela va permettre à tout moment, de savoir qui sont les premiers, et de comparer son propre score à celui des autres.
A la fin de la compétition, les classements sont bloqués et les récompenses sont distribuées aux personnes détenant les meilleurs scores.

1.2 Sujet

Les chercheurs de IEEE Computational Intelligence Society (IEEE-CIS) souhaitant améliorer l'expérience client avec une détection de fraude plus précise, ils se sont donc associés à Vesta corporation la plus grande société de paiement au monde, à la recherche de meilleures solutions pour l'industrie de la prévention de fraude. Pour ce fait ils ont lancé sur Kaggle un challenge.
Dans ce dernier, l'objectif est de comparer des modèles d'apprentissage automatique sur un ensemble de données à grande échelle qui prédit au mieux la probabilité qu'une transaction soit frauduleuse.
Les données proviennent des transactions de commerce électronique de Vesta dans le monde réel et contiennent un large éventail de fonctionnalités, du type d'appareil aux fonctionnalités du produit. Ces données sont de deux types, Un jeu de données d'apprentissage est présent afin de permettre aux algorithmes d'apprendre et un jeu de donnée de test (validation et test) permettant aux participants d'évaluer

leurs modèles. La différence entre ces deux jeux de données réside dans le fait que le second (test) ne contient pas les niveaux de pertinence, seul Kaggle en à la connaissance.
Le rôle des compétiteurs est de prédire la valeur de ce champ manquant et de fournir un fichier de soumission (sample-submission) afin d'être noté. Cette évaluation reflète alors la performance du programme pour l'attribution de la pertinence des associations "TransactionID-isFraud". La note est calculée sur le serveur du site en utilisant une formule.
Le RMSE (Root Mean Squared Error) est donné par la formule suivante :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- y est la valeur observée et cachée par kaggle
- $\hat{y}$ est le valeur prédite par le compétiteurs

1.3 Description des données

Les données sont divisées en deux fichiers identity et transaction, qu'il est est possible de joindre par la variable clé TransactionID. Toutes les transactions n'ont pas les informations d'identité qui correspondent.
Les données d'entraînement contiennent des informations sur 590540 clients: le numéro de transaction,le type de carte,les adresses,etc... Aussi, elles contiennent 433 variables et il est également indiqué si chaque client a été victime d'une fraude ou non. L'objectif est alors de prévoir la probabilité de fraude de 506691 client des données test contenant les mêmes variables.

Pour la réalisation de ce projet, la première chose à faire est de comprendre les données. Nous allons ainsi évaluer quelles variables vont être importantes dans la réalisation de nos modèles de prédiction.

1.4 Modélisation des données

Une fois le problème bien posé et la structure de données clairement définie nous avons pu commencer la pratique. Dans cette étape de modélisation, l'objectif est de coder un ensemble d'outils qui devait être à la fois suffisamment générique pour permettre d'exploiter les données avec différentes méthodes d'analyse des données et suffisamment performants pour obtenir des résultats en un temps raisonnablement court. Pour ce faire, nous avons donc utilisé python (grâce à la multitude

de modules présentes pour le data science). Les plus notables de ces modules sont les suivantes :

- pandas : fournit des structures de données haute performance et des outils d'analyse. Ce qui nous a été utile pour charger les données depuis les fichiers csv;
- numPy : propose des outils semblables à Pandas mais est nécessaire pour certaines dépendances;
- seaborn : est une bibliothèque de visualisation de données basée sur matplotlib. Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs;
- sklearn : est une bibliothèque libre destinée à l'apprentissage automatique.

2 Travail réalisé

2.1 Analyse exploratoire des données

Après avoir importé et fusionné les données, nous avons d'abord examiné les valeurs manquantes. Le jeu de données contient une grande partie de valeur manquante et cela s'est produit dans 411 variables sur un total de 433 variables. On a environ 47% des variables qui ont plus de 70% de valeur manquante. Nous avons donc procédé par supprimé certaines variables qui présentaient 99% de valeurs manquantes.

2.1.1 isFraud - Variable cible

Comme le montre le graphique ci-dessous, la distribution de la variable cible est extrêmement déséquilibrée. Seulement 3.5% des transactions sont frauduleuses dans l'ensemble de données d'apprentissage.
Si nous utilisons ces données déséquilibrées comme base pour nos modèles prédictifs, nous pourrions obtenir beaucoup d'erreurs et nos algorithmes seront probablement sur-dimensionnés puisqu'ils "supposeront" que la plupart des transactions ne sont pas frauduleuse.

Ainsi, ce sera un problème, car nous nous intéressons davantage à la classe minoritaire, autrement dit, nous voulons apprendre à nos modèles de bien reconnaître les transactions frauduleuses. Nous avons donc effectué des techniques d'équilibrage sur l'ensemble des données pour donner plus de poids à la classe minoritaire et des métriques d'évaluation soigneusement sélectionnées. Elles seront discutées dans les sections suivantes.

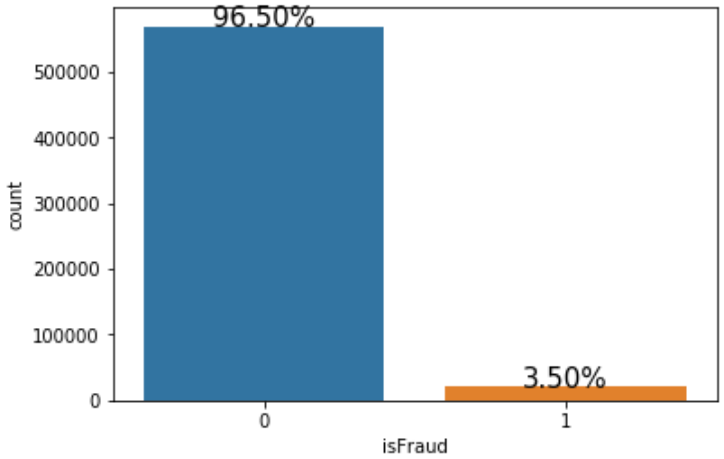


Figure 1 – Distribution de la variable isFraud

2.1.2 TransactionDT - Variable temporelle

La variable 'TransactionDT' est une caractéristique temporelle importante qui a été masquée par de grands nombres séquentiels.

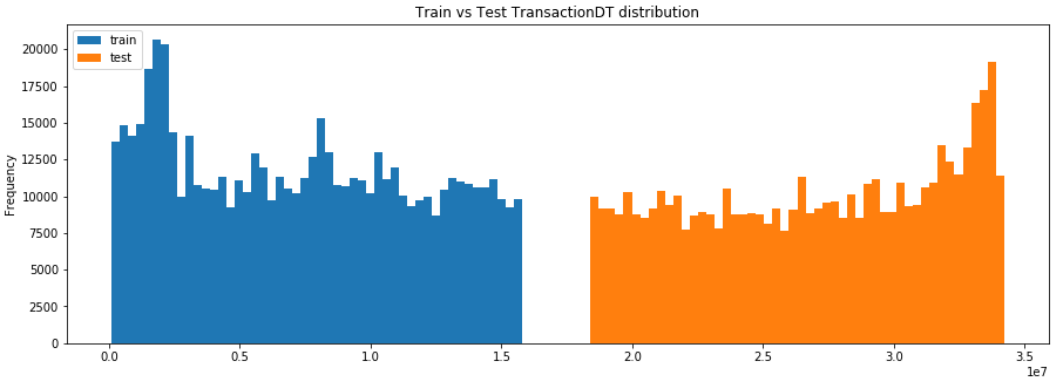


Figure 2 – Distribution de la variable TransactionDT : train vs test

Comme sa valeur varie de 86 400 à 15811131, il est raisonnable de déduire que la valeur réelle est en secondes. En le calculant en secondes, nous pouvons voir que les données d'apprentissage et de test sont dans un intervalle de temps différent.

La période des données train et de test est respectivement de 182 et 183 jours. La période totale est de 395 jours, avec un écart de 30 jours entre le train et le test, comme le montre le graphique de distribution ci-dessous. Par conséquent, nous pouvons créer de nouvelles fonctionnalités telles que les heures dans la journée et les jours dans la semaine pour fournir des informations plus explicites à nos modèles.

Maintenant, visualisons la distribution de la variable 'heure' à partir du graphique de la fréquence des transactions par rapport à la variable cible 'isFraud' à différentes heures:

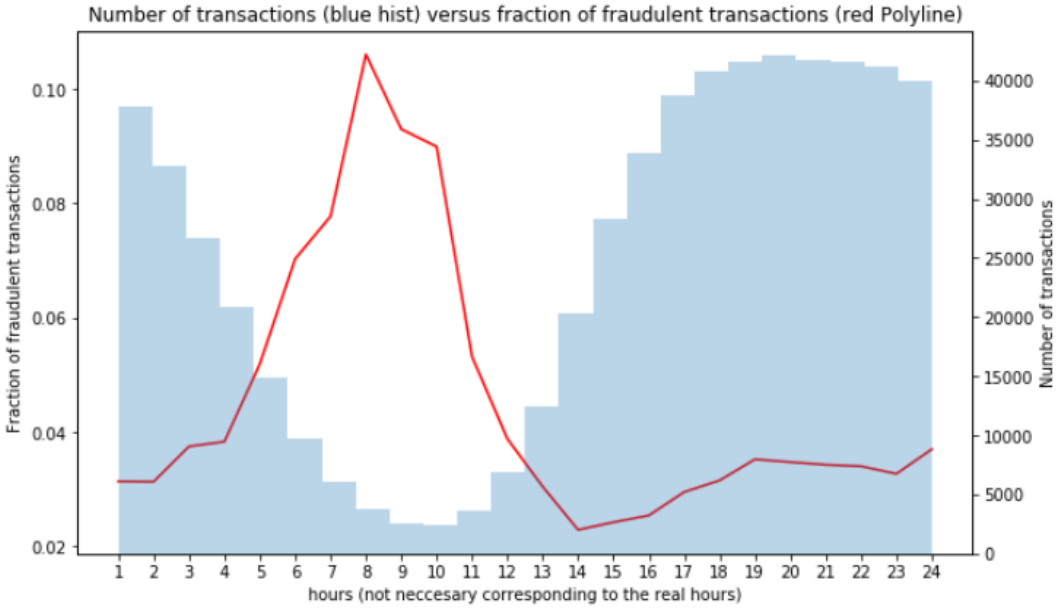


Figure 3 – Distribution du temps en fonction de la variable isFraud

Nous pouvons voir que la fraction de fraude de 4 h à 12 h est nettement plus élevée que pendant les autres heures. En revanche, la fraction de fraude la plus faible se produit de 14 h à 16 h. Et entre 16 h et 19 h, on observe une augmentation progressive de la fraude. Cela nous indique qu'il y a une tendance inversée entre la fréquence des transactions et la fraction de fraude, ce qui est logique car les escrocs peuvent payer des frais non autorisés lorsque les gens dorment et ne risquent pas de se faire remarquer.

2.1.3 TransactionAmt - Montant des transactions

Le montant de la transaction est une autre caractéristique vitale qui a gagné en importance dans cette étude. Elle n'a aucune valeur manquante. Par la suite, nous allons visualisé cette variable dans les ensembles de données de train et test par nuages de points.

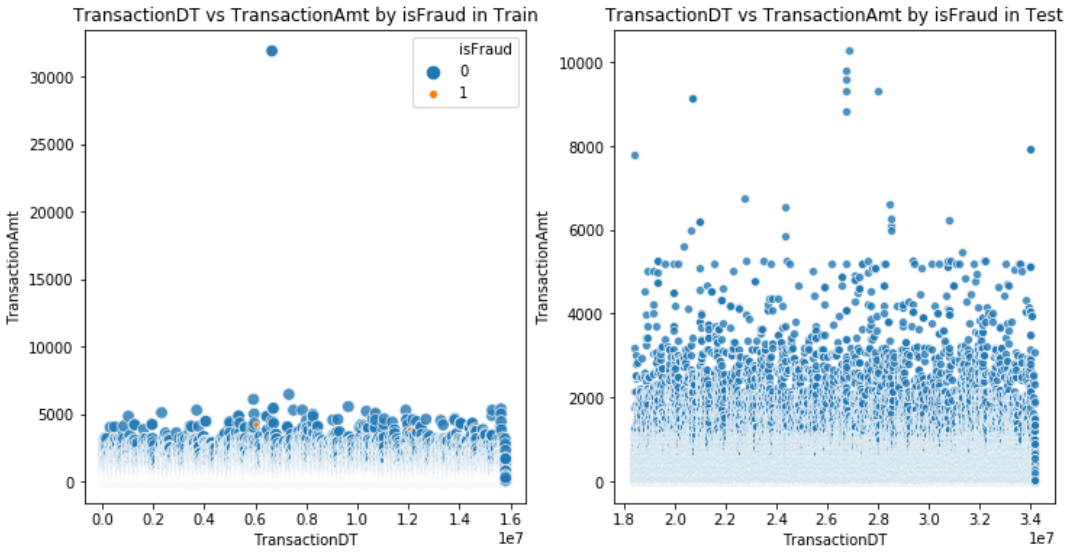


Figure 4 – Distribution des montants de transaction en fonction de la fraude

Comme nous pouvons le voir sur la figure de gauche ci-dessus, deux valeurs extrêmes se retrouvent en double avec une valeur de 31 937$ dans les données d'apprentissage. Des valeurs aberrantes comme celle-ci peuvent causer un problème de sur-ajustement. Par exemple, les modèles arborescents peuvent placer ces valeurs aberrantes dans des nœuds feuilles, qui sont du bruit et ne font pas partie d'un modèle général.
Par conséquent, nous avons décidé de supprimer les valeurs supérieures à 10 000 dans l'ensemble des données d'entraînement pour les modèles classiques. Cependant, nous ne pouvons pas supprimer de lignes dans les données test car Kaggle exige de soumettre un résultat de prédiction qui correspond à la longueur d'origine de l'ensemble de test.

2.1.4 Card6 - Information sur les cartes de paiement

Cette variable ne présente que 15 observations de «carte de paiement» et 30 observations de «carte de débit ou de crédit» dans les données train, mais aucun dans

les données test. Comme la taille de l'échantillon n'est pas suffisante pour trouver un modèle général et que le type de carte majoritaire est le débit, nous pouvons remplacer «carte de paiement» et «carte de débit ou de crédit» par «débiter».

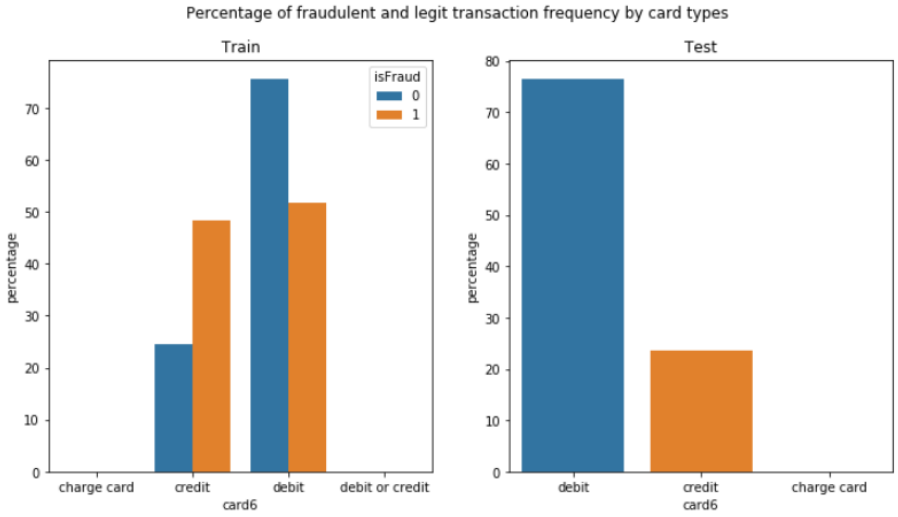


Figure 5 – Pourcentage de fraude par type de cartes

Après transformation, on obtient la représentation suivante :

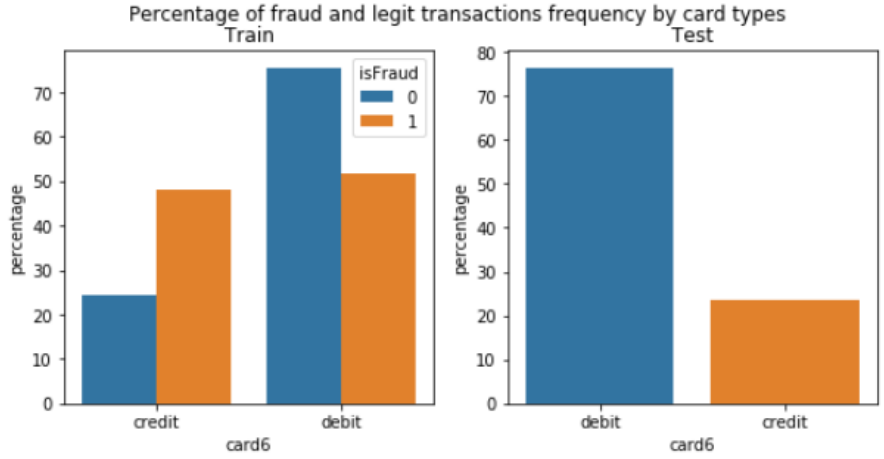


Figure 6 – Pourcentage de Fraude par type de cartes

2.1.5 «Addr1» et «Addr2» - Région et pays de facturation

La caractéristique «addr1» représente la région de facturation des acheteurs, nous comptons environ 300 régions distinctes. Comme nous pouvons le voir sur la figure

ci-dessous, la région 330 a une fréquence plus élevée de transactions frauduleuses que les autres régions, ainsi que la fréquence de transactions la plus élevée.

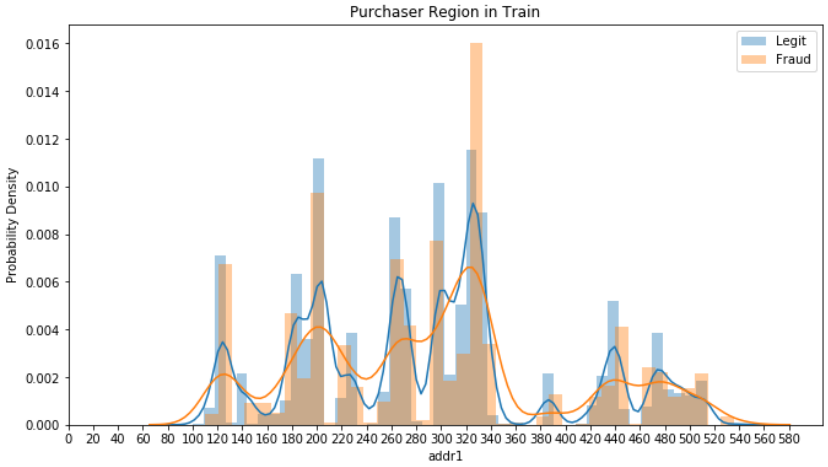


Figure 7 – Distribution de addr1

Pour la caractéristique «addr2», il y a environ 70 pays de facturation et le pays 87 représente environ 88% des transactions. Étant donné que le fournisseur de données Vesta est principalement basé aux États-Unis, 87 représente probablement des comptes américains.

Figure 8 – Distribution de addr2

2.1.6 Dist - Distances des adresses

Pour la variable «dist1», nous avons trouvé que sa valeur moyenne des transactions frauduleuses est 1,5 fois plus élevée que des transactions légitimes. On suppose

que c'est parce que les fraudeurs ont tendance à utiliser la carte sur une distance plus longue que l'adresse du titulaire de la carte.

2.1.7 V1 - V339 - Vesta engineered rich features

Nous avons vérifié la corrélation entre les variables du groupe V1-V339. Comme la plupart des caractéristiques sont corrélées les unes aux autres, nous avons effectué une PCA sur les varibles V qui peuvent non seulement atténuer le problème de la dimensionnalité mais également économiser le temps d'exécution, ainsi que décorréler les fonctionnalités pour améliorer les performances du modèle. L'inconvénient du PCA est que nous perdrons l'interprétation des variables. Mais ce n'est pas grave car la signification de ces variables V est inconnue de toute façon.

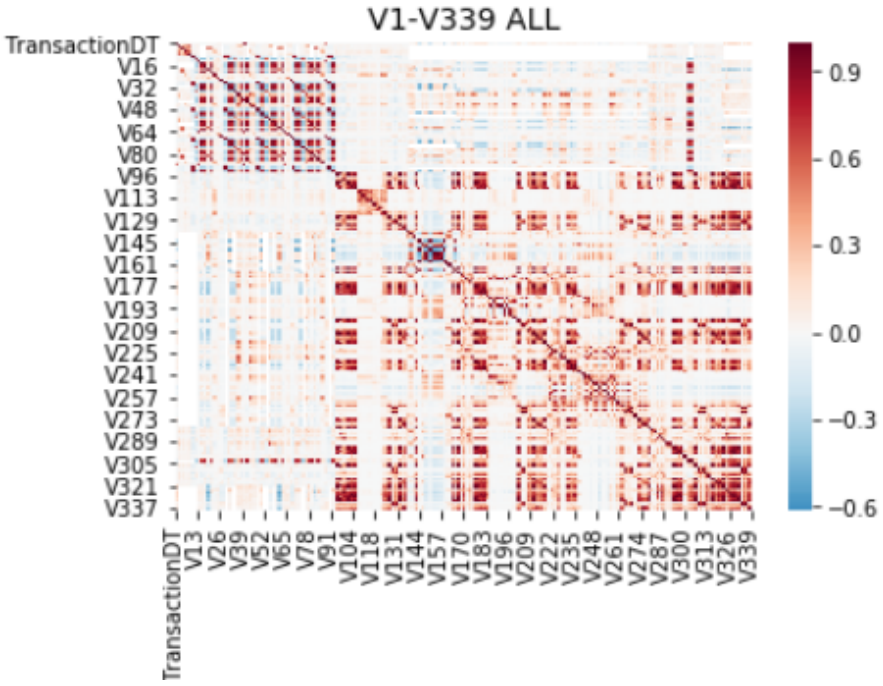


Figure 9 – Corrélations entre les variables V1-V339

2.1.8 P_emaildomain et 'R_emaildomain' - Domaine de messagerie de l'acheteur et du destinataire

P_emaildomain représente le domaine de messagerie de l'acheteur, tandis que le R_emaildoamin représente le domaine de messagerie du destinataire. Nous avons transformé des entités avec des observations / chaînes similaires en les regroupant

c'est-à-dire en faisant du mapping des emails.
Par exemple, les observations «yahoo.com» et «yahoo.com.mx» sont regroupées sous «Yahoo». En outre, toutes les valeurs avec moins de 500 entrées sont définies comme «Autres» et toutes les valeurs manquantes sont remplies par «NoInf». Il ressort de l'exploration des ces variables que les criminels utilisent plus le gmail.

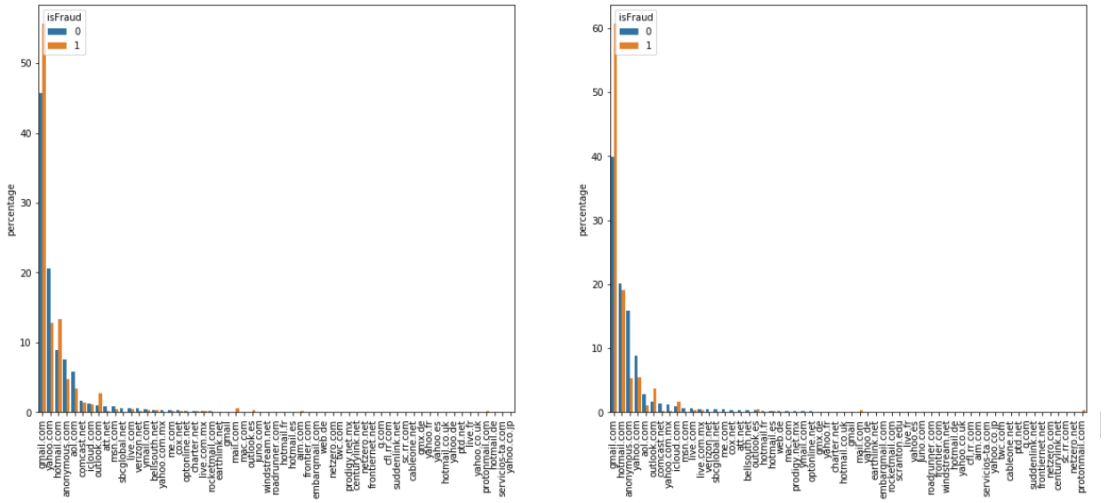


Figure 10 – Pourcentage de fraude par domaine de messagerie du destinataire

2.1.9 Problèmes rencontrés

Durant cette analyse nous avons constaté diverses problèmes :

- Problème des valeurs manquantes qui représente environ 45% des données;
- Imbalanced class (La variable cible a un problème de déséquilibre de classe où l'instance de fraude est beaucoup plus faible que la non-fraude);
- Plusieurs colonnes sont homogènes, par conséquent, ne fournissent aucune information utile pour prédire la variable cible;
- Des transactions de train et de test ne se chevauchent pas;
- problème de mémoire, certains programmes d'apprentissages prennent beaucoup plus de temps lors de l'exécution.

2.2 Préparation des données

Un élément clé de la conduite de la science des données consiste au nettoyage des données. C'est pourquoi nous mettrons en place des procédures de nettoyage

strictes et complètes basées sur les analyses précédentes.
À côté de l'EDA ci-dessus que nous avons effectué, nous pouvons maintenant effectuer une ingénierie des caractéristiques supplémentaire en utilisant certaines statistiques de groupe, la mise à l'échelle et la normalisation fournissant ainsi à nos modèles plus d'informations et révèle davantage la relation entre les principales caractéristiques. Nous avons procédé comme suite.

2.2.1 Encodage d'étiquettes, imputation des valeurs manquante et réduction de dimension

Comme nous utiliserons des modèles arborescents capables de gérer des variables catégorielles, nous avons fait le codage d'étiquettes pour toutes les variables catégorielles. Après cela, nous avons effectué PCA pour compresser les fonctionnalités V en 30 composants principaux, car ils ont expliqué environ 98,6% de la variance totale.
Ensuite, nous avons imputée certaines valeurs manquantes par -999, ce qui est un nombre négatif inférieur à toutes les valeurs non-NAN, car les modèles arborescents sont robustes dans la gestion des valeurs manquantes et traiteront ces nombres -999 comme une catégorie spéciale, avec la même attention comme d'autres nombres. De plus, nous convertissons également des nombres infinis Inf en NaN.

2.2.2 Validation croisée

Pour le fractionnement des données, nous avons procédé de différente manière selon la méthodologie que nous avons employé pour un modèle.

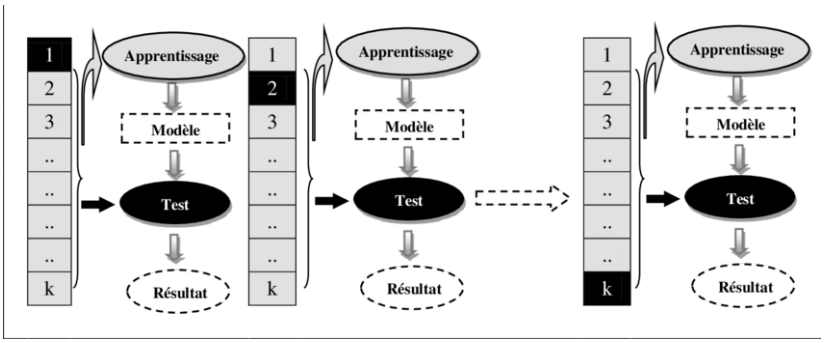


Figure 11 – Procédure de validation croisée

Par exemple, pour le modèle XGBOOST, nous avons procédé par local validation, c'est-à-dire que le découpage se fait de manière locale et pas aléatoire. Pour le

modèle LGBM, nous avons procédé par k-fold cross validation.
Pour les modèles d'apprentissages profond et classique, nous avons appliqué le « testset validation » ou « holdout method » qui consiste à diviser l'échantillon de taille n en deux sous-échantillons, le premier dit d'apprentissage (pratiquement à 70 % de l'échantillon) et le second dit de validation. Le modèle est bâti sur l'échantillon d'apprentissage et validé sur l'échantillon de validation.

2.2.3 Lutte contre les classes déséquilibrée - Resampling

Comme indiqué dans l'EDA - distribution de la variable cible, nous traitons un ensemble de données à déséquilibre extrême, ce qui nous oblige à mettre en œuvre des techniques d'équilibrage. Il existe plusieurs techniques d'équilibrage à la fois sur-échantillonnage et sous-échantillonnage. Puisque le sous-échantillonnage (undersampling) nous ferait perdre des tonnes de lignes et d'informations pour la classe minoritaire (3.50%), nous allons procédé à un sur-échantillonnage.
Le sur-échantillonnage (oversampling) consiste à ajouter des exemples sur la classe minoritaire pour équilibrer les classes. Cet ajout d'exemple s'appuie juste sur un tirage avec remise de l'échantillon constituant la classe minoritaire, donc non robuste pour entraîner les modèles.

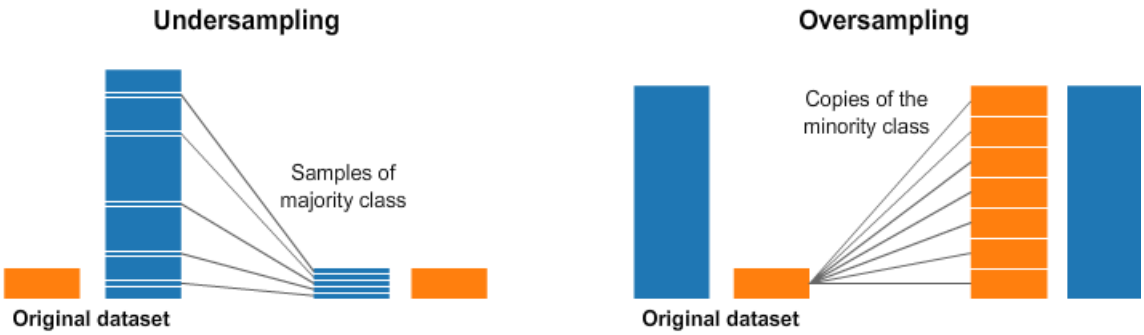


Figure 12 – sous-échantillonnage vs sur-échantillonnage

Cette dernière méthode a été testée mais n'a pas donné de bon résultat car elle s'appuie juste sur une sélection avec remise de l'échantillon constituant la classe minoritaire. Autrement dit, il n y pas de variations dans ces copies. Du coup, en faisant un peu de recherche plus pousée, on s'est aperçu qu'il existe d'autres techniques de ré-échantillonnage (resampling) plus sophistiquées en utlisant le module imbalanced-learn de Python.

Le SMOTE (Synthetic Minority Oversampling Technique) consiste à synthétiser des éléments pour la classe minoritaire, en se basant sur ceux qui existent déjà. Elle consiste à choisir au hasard un point de la classe minoritaire et à calculer les voisins les plus proches pour ce point. Les points de synthèse sont ajoutés entre le point choisi et ses voisins.

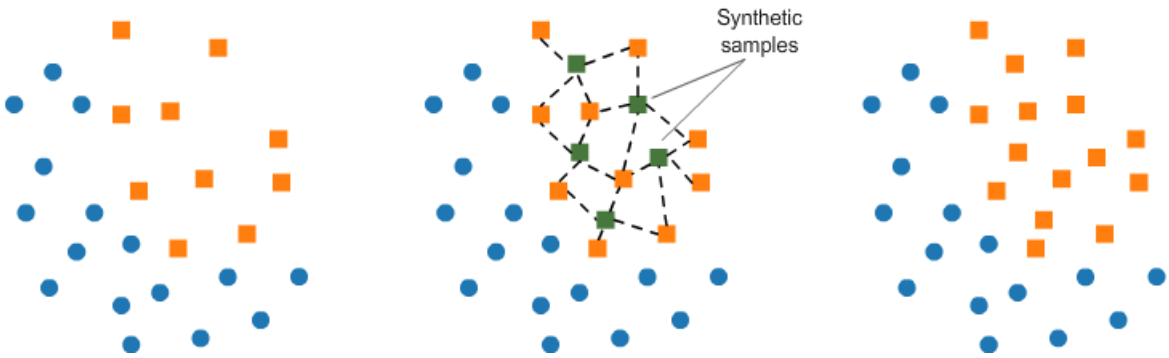


Figure 13 – SMOTE procedure

Cependant, même le SMOTE n'arrivait pas à nous donner de bon résultat c'est-à-dire les modèles n'apprennent pas bien sur le jeux de donnée d'apprentissage, les scores obtenus sur le train est relativement faible ne dépassant pas les 70%. Ainsi pour contourner le probléme, nous avons procédé à une simple stratification lors du split du jeux de données initiales.

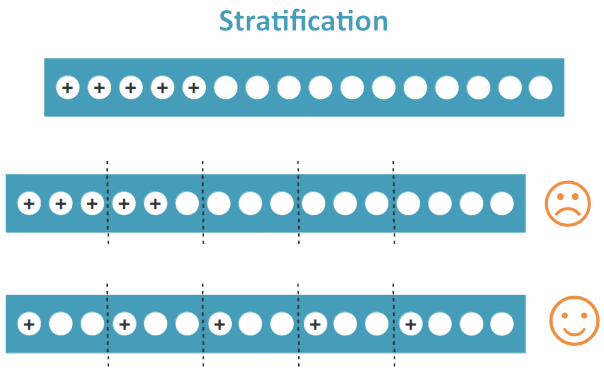


Figure 14 – Stratification procedure

Dans le cas d'un problème de classification, on s'efforce généralement de créer

les k folds de sorte à ce qu'elles contiennent à peu près les mêmes proportions d'exemples de chaque classe que le jeu de données complet.
On cherche à éviter qu'un jeu d'entraînement ne contienne que des exemples positifs et que le jeu de validation correspondant ne contienne que des exemples négatifs, ce qui va affecter négativement la performance du modèle, c'est ce que l'on appelle la stratification.

2.2.4 Bilan de la procédure de travaille

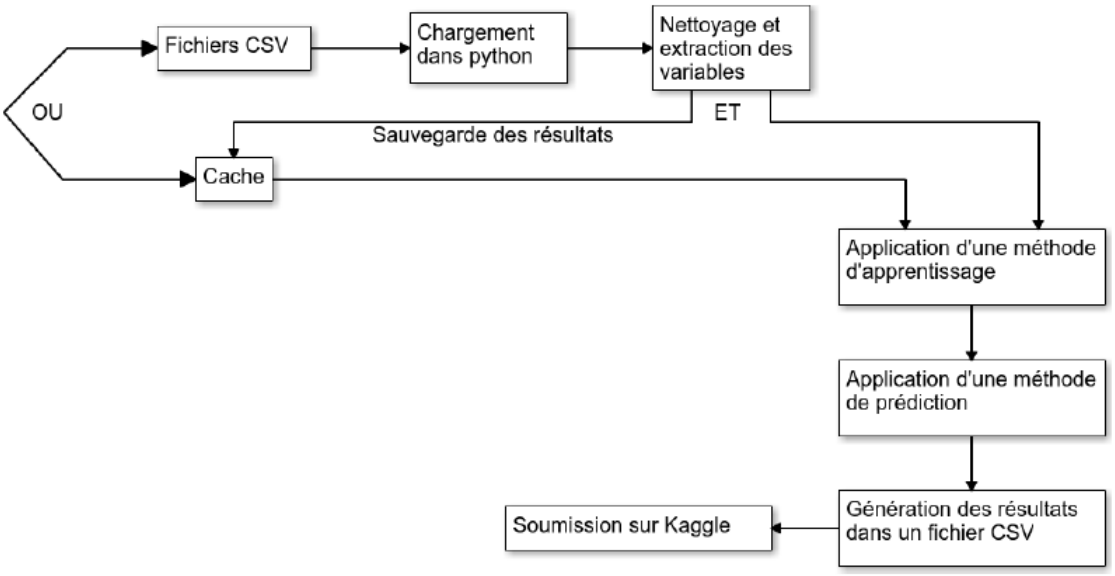


Figure 15 – Procédure d'analyse des données

Cette figure résume la procédure de travaille que nous avons mis en place, en partant des données brutes, pour arriver a l'obtention de résultats (soumission sur Kaggle, etc). Et ce, quelles que soient les méthodes d'apprentissage et de prédiction choisies.

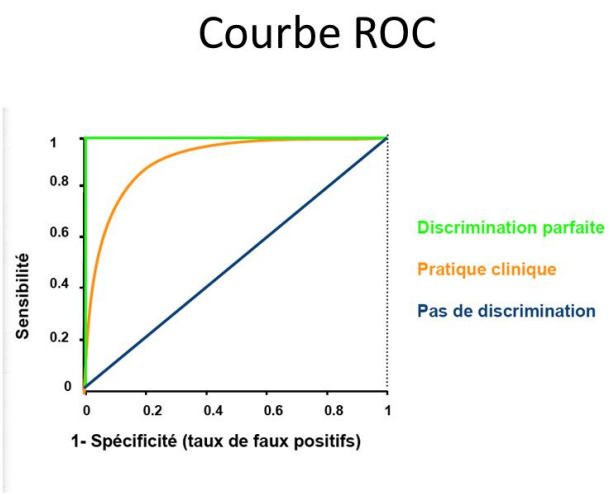
2.3 Critère d'évaluation des modèles

Étant donné que nous sommes confrontés à un problème de classification binaire sur un ensemble de données extrêmement déséquilibré, la matrice de confusion et une série de mesures dérivées sont des outils essentiels pour vérifier les performances de nos modèles. Mais nous devons choisir très soigneusement le critère de

sélection du modèle. La précision de mesure couramment utilisée (accuracy) ne peut pas être utilisée dans ce cas, car elle serait facilement masquée comme un score satisfaisant en classant tous les cas dans la classe majoritaire.

Bien que ce concours Kaggle utilise le score AUC pour évaluer les modèles, nous devons choisir les métriques en fonction de chaque problématique.
Ce qui nous intéresse principalement dans la matrice de confusion sont TP (le nombre de vrais positifs: le modèle prédit correctement la fraude), FP (le nombre de faux positifs: le modèle prédit incorrectement la fraude) et FN (le modèle prédit incorrectement la transaction légitime).

Nous visons à minimiser à la fois FN et FP et à maximiser TP parce que FN causerait une perte pécuniaire directe de fraude non détectée ainsi qu'une expérience client négative, et FP causerait l'insatisfaction des clients en les faisant se sentir gênés et inquiets. Cela pourrait induire un désabonnement, qui est une perte indirecte.
Nous voulons également maximiser le TP, car il détecte correctement la fraude dans la classe minoritaire et stoppe ainsi la perte. Par conséquent, les mesures qu'il convient d'utiliser comme critère de sélection du modèle concernant notre problématique sont la précision, le rappel, f-beta et l'AUC-ROC score.

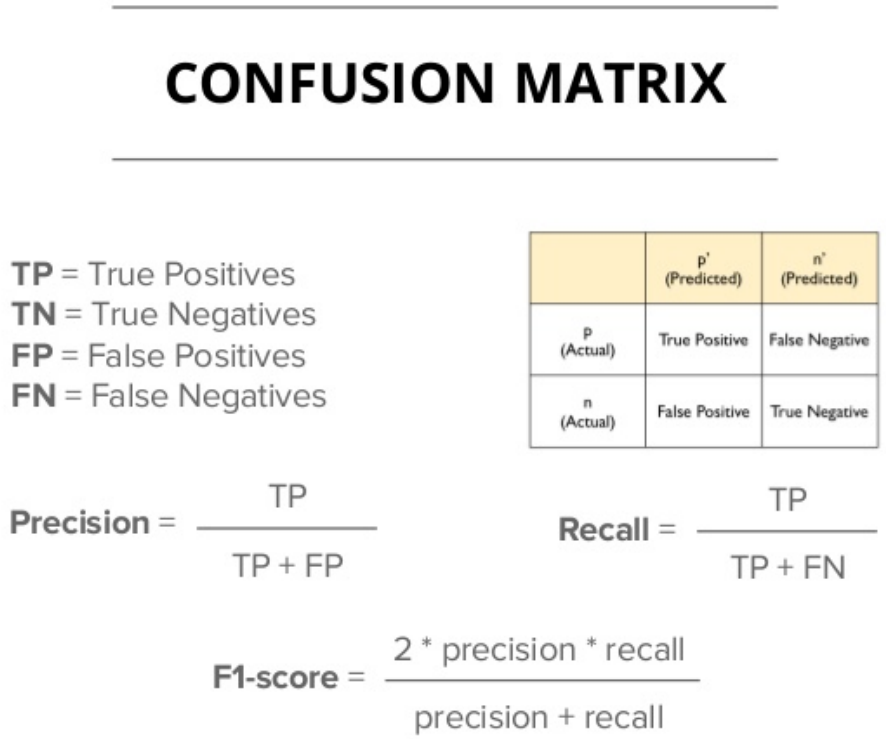


La courbe ROC trace le vrai taux positif sur l'axe des y par rapport au taux de faux positifs sur l'axe des x en fonction du seuil du modèle pour classer un positif. L'AUC représente l'aire sous la courbe ROC. Plus le score AUC est élevé,

meilleures sont les performances globales de notre modèle. L'AUC-ROC prend en considération tous les quadrants dans la matrice de confusion. Cependant, nous nous soucions beaucoup plus du TN.

Pour mettre l'accent sur FP, FN et TP, le rappel et la précision entrent généralement en jeu. Cependant, comme la précision, l'utilisation du rappel et de la précision uniquement peut être dangereuse. Le rappel identifie le taux auquel les observations de la classe positive sont correctement prédites; la précision indique le taux auquel les prédictions positives sont correctes. Mais si nous qualifions toutes les transactions de fraude, notre rappel passe à 1.0.

Et si nous n'identifions qu'une seule transaction correctement, la précision sera de 1.0 (pas de faux positifs). Cependant, notre rappel sera très faible car nous aurons encore de nombreux faux négatifs, ce qui est un compromis entre les mesures que nous choisissons de maximiser, comme lorsque nous augmentons le rappel, nous diminuons la précision.



3 Modélisation des données

Dans cette partie, nous allons tenter de construire des modèles permettant d'obtenir les meilleures prédictions possible. Pour ce faire, nous allons faire varier les paramètres de ces différents algorithmes que nous allons utiliser de façon à trouver quelles peuvent être les valeurs des paramètres les plus optimales possibles sur notre jeu de données.

3.1 Choix des modèles et variation des paramètres

Cependant pour effectuer l'apprentissage sur nos données nous allons en premier lieu utiliser les algorithmes classiques suivant :

- K plus proches voisins
- Arbre de décision
- Forêt aléatoire

Pourquoi avoir pensé à utiliser en premier l'algorithme KNN ? car il s'agit d'un algorithme simple mais très efficace pour classifier des données labellisées.

3.1.1 Les K plus proches voisins

Le K-Nearest Neighbor est une méthode non paramétrique utilisée pour la classification et la régression.

KNN

```
from sklearn import neighbors

knn = neighbors.KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

```
36]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                          weights='uniform')
```

```
pred_train = knn.predict_proba(X_train)
```

```
print("score auc train :",roc_auc_score(y_train, pred_train[:, 1]))
```

```
score auc train : 0.9890928498077949
```

Prediction_test_Knn

```
%%time
pred_test=knn.predict_proba(test_df[useful_cols])
```

```
Wall time: 1h 54min 23s
```

Dans les deux cas, il s'agit de classer l'entrée dans la catégorie à laquelle appartient les k plus proches voisins dans l'espace des caractéristiques identifiées. Dans notre cas, le résultat est une classe d'appartenance donc l' objet d'entrée est classifié selon le résultat majoritaire des statistiques de classes d'appartenance de ses k plus proches voisins.

D'où l'importance d'avoir équilibré les classes avant l'implémentation du modèle. Pour ce faire nous avons pris K=3 comme nombre de voisin. Les prédictions résultantes de l'application de cette méthode ne sont pas satisfaisantes. Nous avons obtenus un score public de 0.681724 lors de notre soumission sur kaggle.

Au vue de ce faible score et du temps de calcul trop élevé, nous avons décidé de ne pas faire une cross-validation pour trouver le K optimal car d'une part, pour son coût en puissance de calcul et d'autre part le fait de devoir conserver toutes les données d'entraînement en mémoire (k-NN convient donc plutôt aux problèmes d'assez petite taille).

Il est également important de noter que cet algorithme est vulnérable à la « curse of dimensionality » : le nombre de données nécessaires pour avoir un bon estimateur croît potentiellement de manière exponentielle avec la dimension, autrement dit avec la complexité de la représentation des données.

3.1.2 Arbre de décision

La popularité de la méthode repose en grande partie sur sa simplicité et de sa forte interprétabilité. Il s'agit de trouver un partitionnement des individus que l'on représente sous la forme d'un arbre de décision. L'objectif est de produire des groupes d'individus les plus homogènes possibles du point de vue de la variable à prédire. Il est d'usage de représenter la distribution empirique de l'attribut à prédire sur chaque sommet (nœud) de l'arbre.

Dans un premier temps l'arbre de décision a été construit avec des hyperparamètres par défaut. Nous avons obtenu un modèle purement sur-ajusté avec un AUC score de 1 sur les données d'apprentissage et de validation avec un score public kaggle de 0.72.
Nous avons donc réglé à l'aide d'une recherche de grille aléatoire avec validation croisée stratifiée en utilisant 'AUC-ROC' comme notation et des hyperparamètres utilisés pour restreindre la taille de l'arbre et tailler l'arbre avec une complexité de coût minimal.
Nous obtenons ainsi un arbre avec le moins de sur-ajustement. Le score AUC résultant de l'apprentissage est de 0,819 et est de 0,818 pour la validation.

3.1.3 Forêt aléatoire

Pour un algorithme de forêts aléatoires que nous allons définir comme “basique”, les besoins en puissance de calculs sont faibles si l'on compare à l'efficacité et à la qualité des résultats fournis. Dans cette configuration, le temps d'apprentissage n'excède pas 2 minutes.

De plus, comme toute méthode d'apprentissage, plus le nombre ou la complexité des paramètres est élevé, plus le temps d'exécution ou d'apprentissage est long (on peut dire des jours voire des semaines).

Grâce à Python et sa richesse en terme de bibliothèques, nous avons pu facilement mettre en oeuvre les forêts aléatoires. En effet, dans la bibliothèque sklearn (scikit-learn), il y a un package: ensemble contenant tout le nécessaire (réglage des hyperparamètres,etc) afin d'utiliser les forêts aléatoires tant pour la classification que pour la régression (nous avons bien évidemment utilisé la classification pour tenter de résoudre notre problème).

Les prédictions résultantes de l'application de cette méthode sont plutôt satisfaisantes. Nous avons obtenus un score public de 0.84 lors de notre soumission sur Kaggle. Ce qui est nettement beaucoup plus mieux que les précédent algorithmes.

Pour approfondir la question des arbres nous avons pris connaissance d'une méthode nommée XGBoost qui utilise les arbres décisionnels et qui est une implémentation de l'algorithme de boosting du gradient. Nos recherches ont débouchés sur deux modèles puissants qui sont très utilisés aux concours d'apprentissage automatique, comme ceux de Kaggle :

- XGboost classifier
- LGBM classifier

Ces deux modèles ont été largement vantées de mérites tant pour le peu de puissance de calcul nécessaire, tant pour les résultats vraiment satisfaisants que ces méthodes fournissent,et étant inconnue de nous, nous avons été séduit par leur résultat à développer ces méthodes afin d'en approfondir la connaissance.

Ces modèles sont non seulement efficace pour les données présentant des problèmes de classe non balancée mais aussi pour des données présentant des outliers, des features non standardisées, des features collinéaires et des missing values.

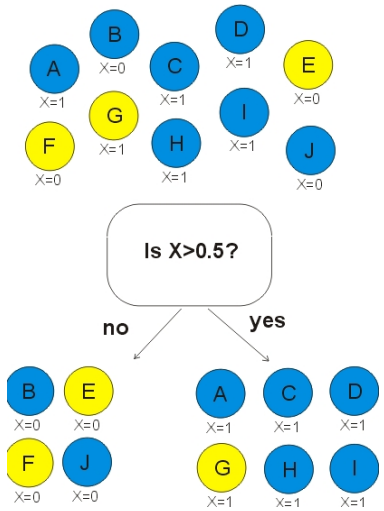
3.1.4 Variable magique - Agrégation de variable

En vu de rendre meilleure les résultats de nos modèles, nous avons opté de crée ce que l'on appelle des variables magiques qui se fait en trois étapes. D'abord, nous avons besoin d'une variable UID pour identifier les clients (cartes de crédit). Ensuite, nous devons créer des caractéristiques de groupe agrégées. Et enfin, nous retirons l'UID.

Supposons que nous ayons 10 transactions A, B, C, D, E, F, G, H, I, J comme le montre la figure ci-dessous.

TransactionID	UID	FeatureX	GroupX	IsFraud
A	1	1	0.75	1
B	1	0	0.75	1
C	1	1	0.75	1
D	1	1	0.75	1
E	2	0	0.33	0
F	2	0	0.33	0
G	2	1	0.33	0
H	3	1	0.66	1
I	3	1	0.66	1
J	3	0	0.66	1

Si nous n'utilisons que les FeatureX, nous pouvons classer correctement 70 % des transactions. Ci-dessous, les cercles jaunes correspondent aux transactions "isFraud=1" et les cercles bleus aux transactions "isFraud=0". Après que le modèle d'arbre ci-dessous ait divisé les données deux groupes gauche et droite, nous prédisons "isFraud=1" pour le groupe de gauche et "isFraud=0" pour le groupe de droite. Ainsi, 7 prédictions sur 10 sont correctes.



Supposons maintenant que nous ayons un UID qui définit les groupes et que nous fassions une caractéristique agrégée en prenant la moyenne de FeatureX dans chaque groupe. Nous pouvons maintenant classer correctement 100 % des transactions. Notons que nous n'utilisons jamais l'UID de la caractéristique dans notre arbre de décision.

3.1.5 XGboost and LGBM classifier

Pour faire simple XGBoost (comme extreme Gradient Boosting) est une implémentation open source optimisée de l'algorithme d'arbres de gradient boosting.

Mais qu'est-ce que le Gradient Boosting ?

Le Gradient Boosting 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 meilleure prédiction.

On parle d'ailleurs de méthode ensembliste. L'idée est donc simple : au lieu d'utiliser un seul modèle, l'algorithme va en utiliser plusieurs qui seront ensuite combinés pour obtenir un seul résultat.

C'est avant tout une approche pragmatique qui permet donc de gérer des problèmes de régression comme de classification. Pour décrire succinctement le principe, l'algorithme travaille de manière séquentielle, contrairement au Random Forest. Cette façon de faire va le rendre plus lent bien sur 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 peut-être surévaluer. A partir de cette première évaluation, chaque individu va être alors pondérée en fonction de la performance de la prédiction précédente et ainsi de suite.

XGBoost se comporte donc remarquablement dans les compétitions d'apprentissage automatique (Machine Learning), mais pas seulement grâce à son principe d'auto-amélioration séquentielle. Il inclut en effet un grand nombre d'hyperparamètres qui peuvent être modifiés et réglés à des fins d'amélioration.

Avec quelque hyperparamètres choisie selon la littérature, le modèle XGBoost combinées avec les variables magiques nous a donné un accuracy de 98% sur

l'échantillon de validation. Après soumission sur Kaggle, ce modèle a pu amélioré excessivement le score public et privé par rapport aux autres modèles classiques que nous avons fait plus haut.

```
print(confusion_matrix(y_train, oof.round()))
```

```
[[568778   1099]
 [ 11797   8866]]
```

```
print(classification_report(y_train, oof.round()))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99    569877
           1       0.89      0.43      0.58     20663

    accuracy                           0.98    590540
   macro avg       0.93      0.71      0.78    590540
weighted avg       0.98      0.98      0.97    590540
```

La figure suivante nous donne les variables les plus importantes utilisées lors de la construction d'un tel modèle.

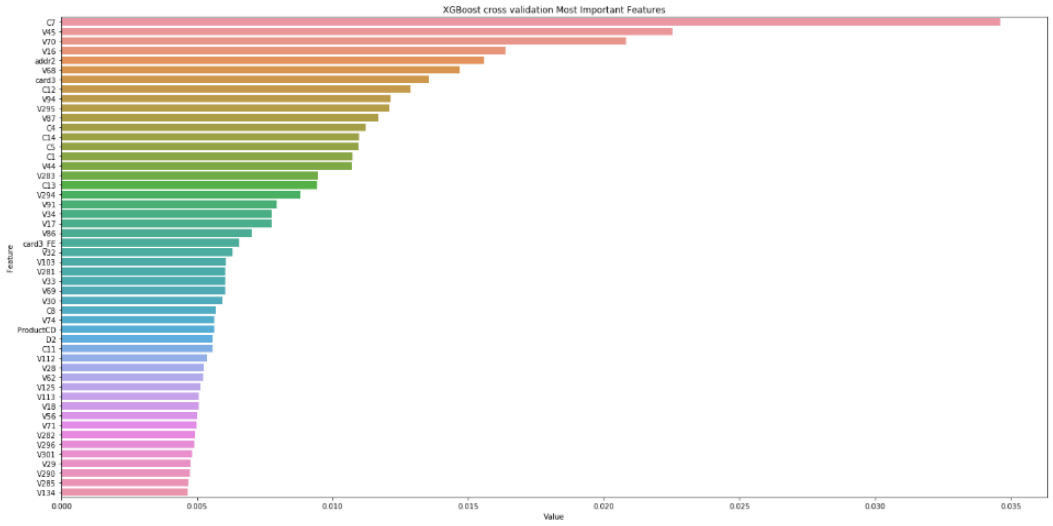


Figure 16 – Importance des variables du XGboost

Avec ce modèle XGBOOST utilisant de nouvelles features obtenues par aggrégation d'autres features, nous avons obtenu un score public de 0.950164 et un score privé de 0.921794 après soumission sur Kaggle.

Avec le modèle LGBM, nous avons obtenu un accuracy de 99% et un AUC de 97.68%. Ce modèle a pu amélioré le score privé sur Kaggle qui est ainsi devenu

0.921948 mais n'a pas amélioré le score public obtenu par le XGboost.

```
print('OOF AUC:', metrics.roc_auc_score(y, oof))
```

```
OOF AUC: 0.9768803636902861
```

```
print(metrics.confusion_matrix(y, oof.round()))
```

```
[[569493    384]
 [  6162  14501]]
```

```
print(metrics.classification_report(y, oof.round()))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99    569877
           1       0.97      0.70      0.82     20663

    accuracy                           0.99    590540
   macro avg       0.98      0.85      0.91    590540
weighted avg       0.99      0.99      0.99    590540
```

La figure suivante nous donne les variables les plus importantes utilisées lors de la construction du LGBM.

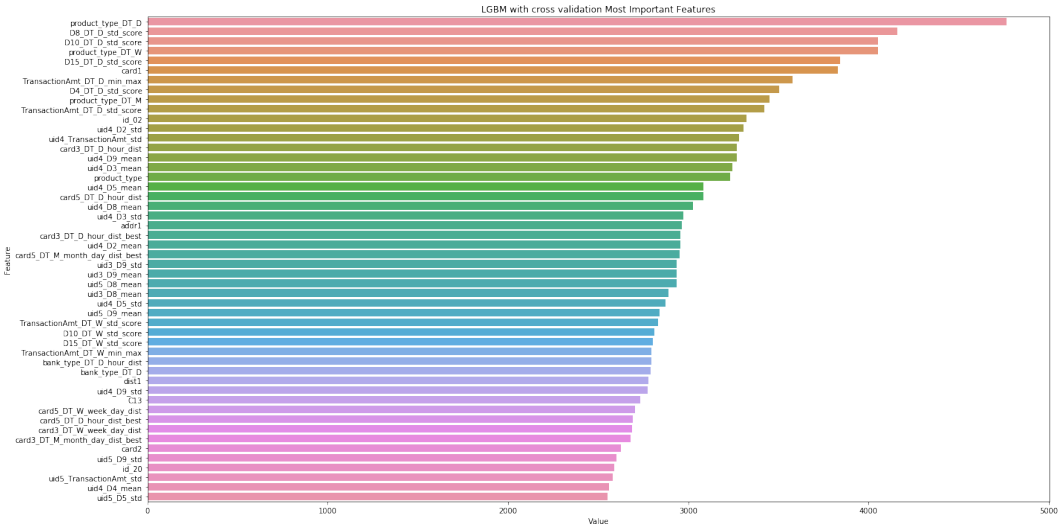


Figure 17 – Importance des variables du LGBM

3.1.6 Autre modélisation du LGBM

Le Light GBM est un modèle complexe de haut de gamme qui surpasse de nombreux autres modèles en ce qui concerne ses performances et sa vitesse. Ce modèle avec les paramètres par défaut se traduit par un score AUC de 0,946 sur le train et de 0,926 sur le test (validation). L' importance des caractéristiques de LGBM

par défaut a été utilisée comme référence pour effectuer l' ingénierie des caractéristiques.

Après réglage avec 200 itérations dans la recherche aléatoire, nous avons obtenu un peu de sur-ajustement mais un modèle globalement amélioré avec un score AUC de 0,9999 sur le train et de 0,9625 sur le test, ce qui est assez bon. La figure suivante nous montre les AUC scores sur le train et sur le test.

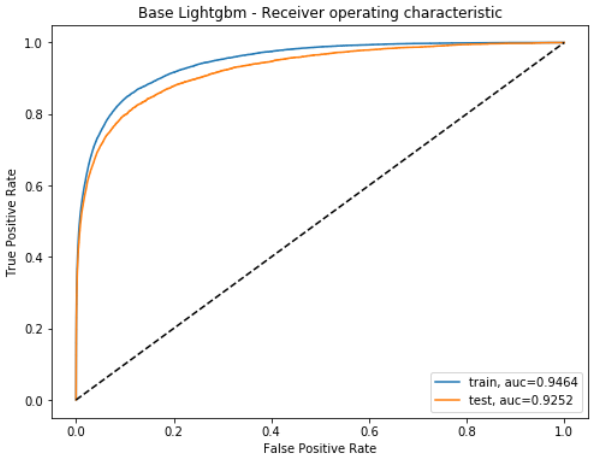
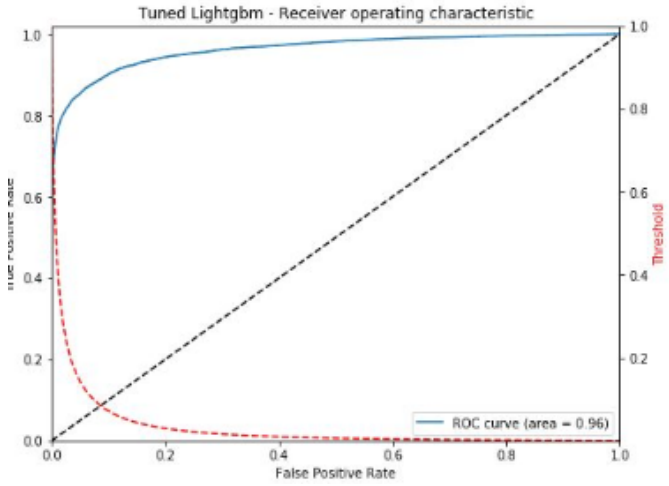


Figure 18 – Courbe ROC

Afin de sélectionner un seuil appliqué sur des données invisibles, nous avons extraire le score de validation. La figure suivante montre le compromis entre True Positive Rate / Recall et False Positive Rate avec des seuils changeant sur l'axe y.



Comme indiqué ci-dessus, nous pouvons sélectionner un seuil plus élevé pour obtenir un meilleur taux de vrai positif / rappel, ou un seuil plus bas pour obtenir un meilleur taux de faux positifs. La stratégie de sélection est basée sur l'objectif recherché.

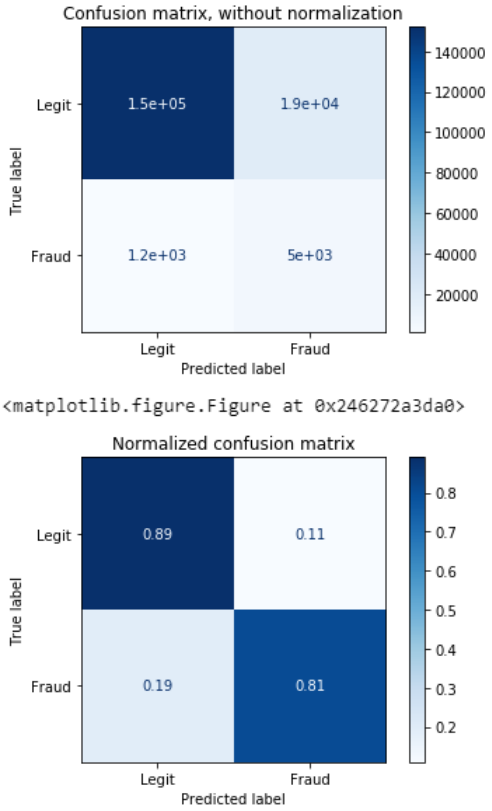


Figure 19 – Matrice de confusion LGMB

Remarque : On observe une petite différence de score Kaggle entre les deux soumissions LGBM (0.94 et 0.92) ceci peut s'expliquer par la différence de réglage des paramètres ou de l'ingénierie des caractéristiques.

Étant donné l'importance des caractéristiques du modèle, nous pouvons voir que les caractéristiques concernant les informations sur la carte de paiement, l'adresse, le temps de transaction, la distance à l'adresse, le domaine de messagerie de l'acheteur et les caractéristiques conçues par des agrégations de caractéristiques principales sont plus robustes pour détecter la fraude.

3.1.7 Neural Network

L'architecture de notre réseau de neurone est assez standard. Nous avons utilisé une couche d'intégration pour que les données catégorielles et numériques passent au travers des couches denses à propagation directe.

Nous avons aussi crée des couches d'intégration de manière à avoir autant de lignes que de catégories et la dimension de l'intégration est le (log1p + 1) du nombre de catégories. Cela signifie donc que les variables catégorielles à très haute cardinalité auront plus de dimensions, mais pas de manière significative, de sorte que les informations seront toujours comprimées à environ 18 dimensions seulement et que le nombre de catégories ne sera que de 2 à 3.

Nous avons ensuite passer les encastrements à travers une couche d'exclusion spatiale qui réduira les dimensions à l'intérieur de l'encastrement par lots, puis les aplatira et les concaténera. Nous avons concatené ensuite ces éléments aux caractéristiques numériques et avons appliqué la norme des lots (batch normalization), puis nous avons ajouté des couches plus denses par la suite. Notre réseau de neurone a donnée les performances suivantes.

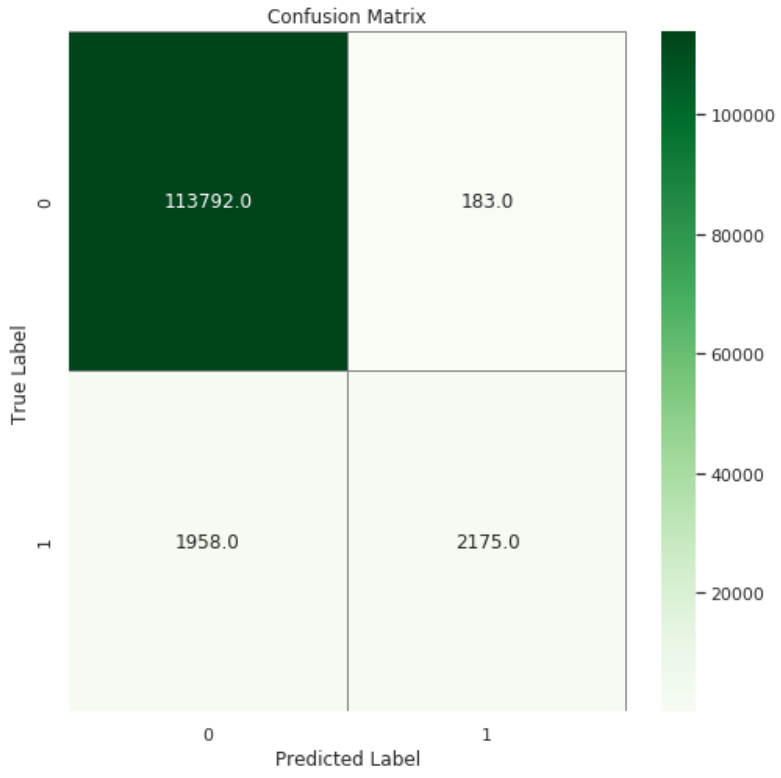


Figure 20 – Matrice de confusion réseau de neurone

Il ressort de l'analyse de la matrice de confusion, d'une part, qu'il y a 1958 FP, c'est-à-dire une transaction non frauduleuse prédite comme étant une transaction frauduleuse et d'autre part, qu'il y a 183 FN, c'est-à-dire une transaction frauduleuse prédite comme étant une transaction non frauduleuse.

A cela, s'ajoute le fait qu'on observe que le modèle LGBM donne un nombre plus signifiant de TP (c'est-dire les tansactions qui sont correctement classées comme frauduleuse).

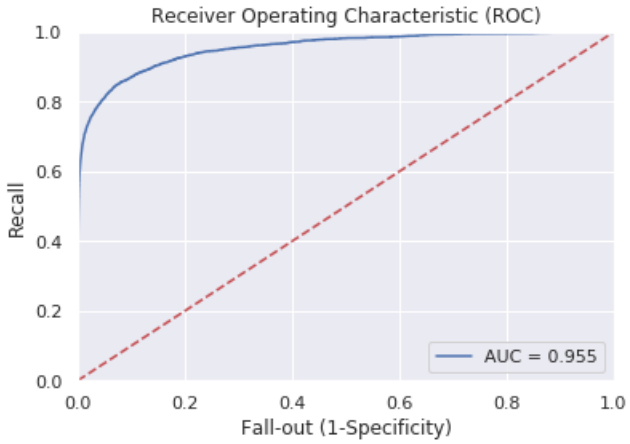


Figure 21 – Courbe ROC réseau de neurone

A la lumière du graphique ci-dessus, il ressort que l'aire en dessous de la courbe est de 0.955, c'est-à-dire que nous avons un AUC de 0.955. Ce qui prouve souligne la robustesse de notre réseau de neurone malgré les nombreux problèmes que nous avons rencontré et un score public kaggle de 0.88.

En conclusion d'un point de vu kaggle, c'est l'algorithme XGboost qui se révèle être la plus efficace pour la prédiction des données.

4 Conclusion

Le contexte de la compétition nous a plongé dans un environnement aussi rude que passionnant où l'on a été amenés à concrétiser et approfondir des domaines que nous n'avions exploré que d'un point de vue théorique et peu pratique. Le fait de pouvoir comparer notre travail avec d'autres professionnels du domaine fut très stimulant et nous a fait réaliser que nous avions encore beaucoup de travail à fournir.

Cela nous a également donné un aperçu de ce qu'il était possible d'accomplir pour des personnes qui travaillent dans le domaine du data science au quotidien. Nous avons mis a profit les connaissances acquises durant les cours de "Introduction au Machine Learning" pour la partie d'analyse ainsi que celui de "Machine Learning & Deep Learning" pour la partie apprentissage.

Nos compétences de programmations ont été mises à l'épreuve durant l'utilisation de Python et de ses nombreuses bibliothèques. Ce challenge étant complet, dans le sens où nous partions de rien et avions pour objectif de terminer avec des résultats, nous avons dû passer par toutes les étapes nécessaires au bon déroulement de celui-ci. Nous avons du réaliser l'analyse du problème et des données, se documenter sur les méthodes à employer, appliquer ces méthodes et finalement commenter les résultats.

Nous nous sommes donc organisés pour répartir toutes ces étapes sur toute la durée du projet en prenant en compte les contraintes de la compétition et celles du projet.

Nous avons appris à exploiter des méthodes de traitements de données à grande dimension (big data) mais aussi des méthodes de prédiction robuste sur des données présentant beaucoup d'anomalies : XGboost et LGBM.

5 Annexes

Annexe 1 : Information sur les variables

Noms des variables	Description des variables
IsFraud	Variable cible fraude=1
TransactionDT	Temps en seconde
TransactionAMT	Montant de paiement de la transaction
ProductCD	Code produit pour chaque transaction
Card1-Card6	Informations sur la carte de paiement
Addr	Adresse
Dist	Distance
P/R _emaildomain	Domaine de messagerie acheteur et destinataire
C1-C14	Comptage,par ex: nombre d'adresse associé à la carte
M1-M9	correspondance, comme le nom et l'adresse sur la carte
DeviceType	Information sur le type de device
DeviceInfo	Information sur la device
id_1-id_38	Informations d'identité
V1-V339	Riches fonctionnalités de Vesta

Figure 22 – Description des variables

Annexe 2 : Soumissions Kaggle 1

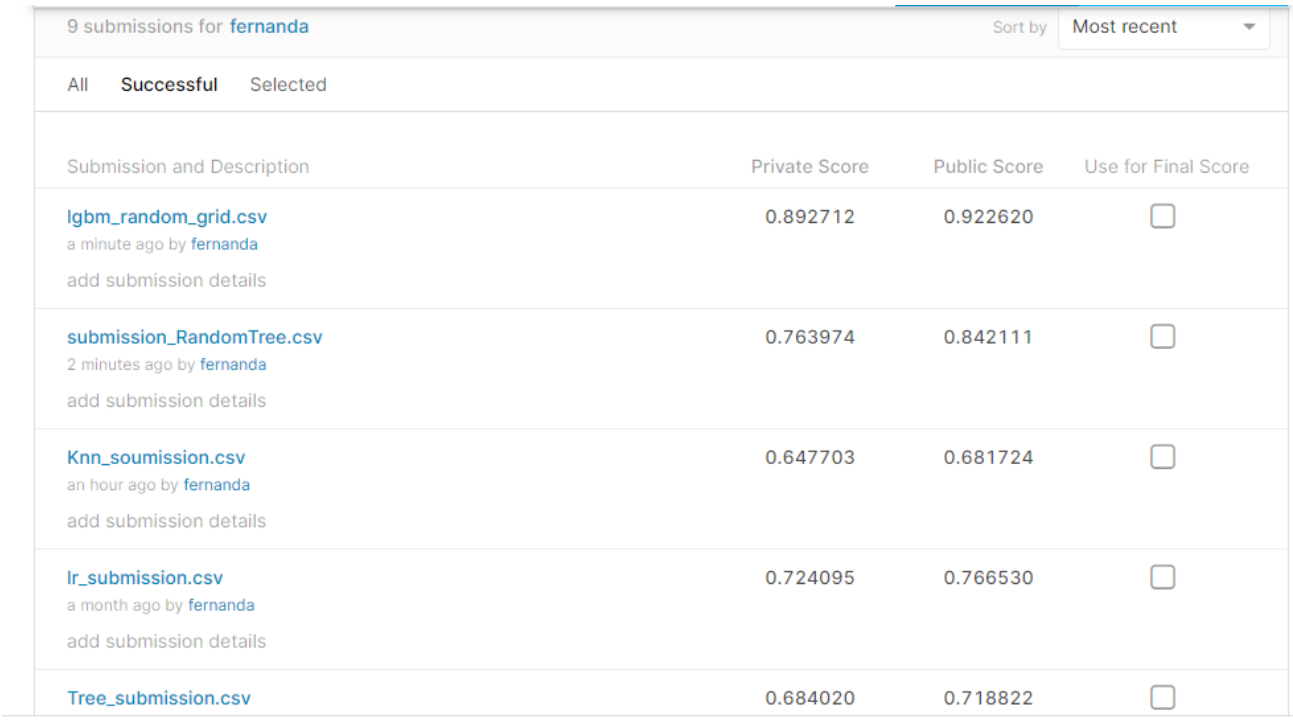


Figure 23 – Score 1 - soumissions kaggle

Annexe 3 : Soumissions Kaggle 2

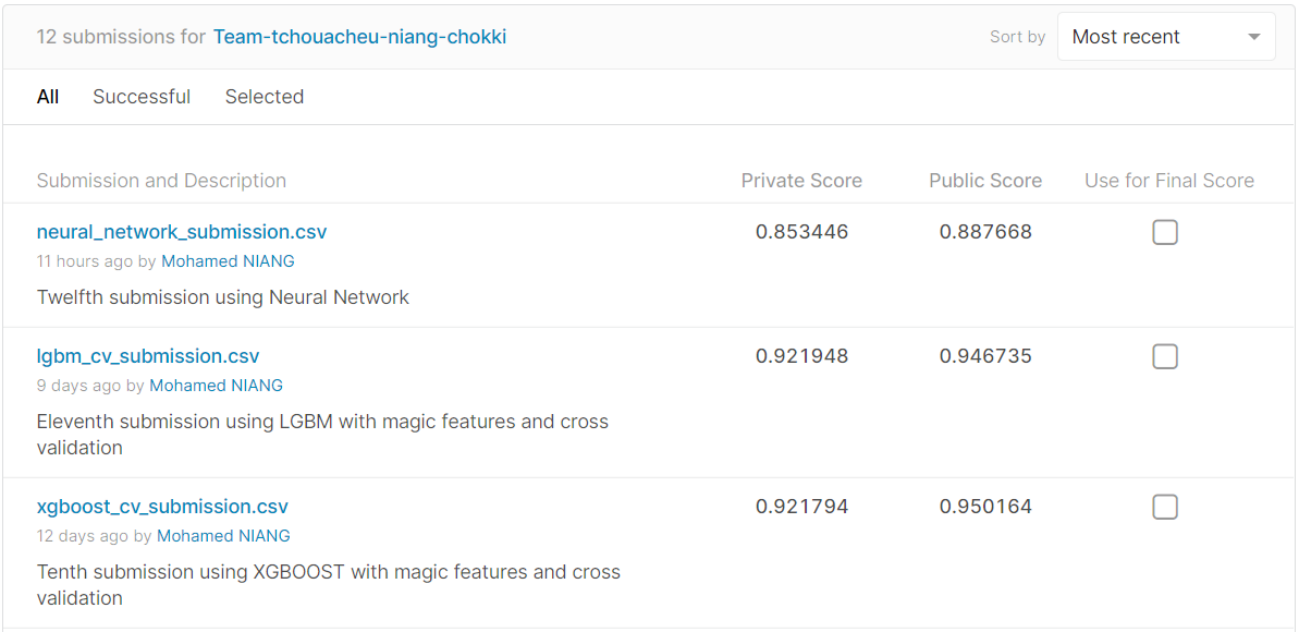


Figure 24 – Score 2 - soumissions kaggle

References

[1] O'REILLY, *Python Data Science Handbook.*

[2] Python, *La documentation et exemples de scikit-learn*

[3] Precision as a Model Selection Criterion
https://www.datascienceblog.net/post/machine-learning/specificity-vs-precision/

[4] Source Kaggle
https://www.kaggle.com/artgor/eda-and-models#Feature-engineering

[5] Source Kaggle
https://www.kaggle.com/artgor/eda-and-models

[6] Source Kaggle
https://www.kaggle.com/shahules/tackling-class-imbalance

[7] Source Kaggle
https://www.kaggle.com/kyakovlev/ieee-fe-with-some-eda

[8] Source Kaggle
https://www.kaggle.com/kyakovlev/ieee-lgbm-with-groupkfold-cv

[9] Gradient Boosting with Scikit-Learn, XGBoost, LightGBM, and CatBoost
https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-

[10] Which algorithm takes the crown: Light GBM vs XGBOOST?
https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/

[11] Complete Guide to Parameter Tuning in XGBoost with codes in Python
https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[12] G. Biau, A. Ficher, B. Guedj et J. D. Malley, COBRA : *A Nonlinear Aggregation Strategy, Journal of Multivariate Analysis p146 (2016), 18–28*

[13] Manuel Fernandez-Delgado, Eva Cernadas, Senen Barro et Dinani Amorim, Do we Need Hundreds of Classiers to Solve Real World Classication Problems ?, *IEEE Trans. Pattern Anal. Mach. Intell. 15 (2014), 3133–31.*