Paris Descartes University

Apprentissage Supervisé

Projet Apprentissage Supervisé

Étudiants :
Ilyes ZEMALACHE
Amine FERDJAOUI
Tanguy HERSERANT

Professeur :
LAZHAR LABIOD

10 Janvier 2020

Table des matières

Résumé

Le projet est décomposé en 3 sous parties, la première analyse concerne des datas synthétiques, la 2eme partie le dataset "VisaPremier" et la 3eme le dataSet "creditCard". Le but de ce projet est d'appliquer différents algorithmes de machine learning supervisés et de les comparer grâce à la courbe ROC AUC.

Pour les parties 2 et 3, deux notesbooks sont disponibles et apporte plus de précision sur certains choix. Comme, par exemple, toutes les courbes ROC-AUC du tuning d'hyperparamètres et l'explications des choix.

1 Introduction

Nous disposons de 5 datasets et chacun servira à une partie :

— Partie 1 : "Aggregation.txt", "flame.txt", "spiral.txt"
— Partie 2 : "VisaPremier.txt"
— Partie 3 : "creditcard.csv"

Les 3 premiers sont des données généré artificiellement et sont donc "propres" et sans bruit. Tandis que les autres sont des données collectés donc comportes des erreurs, des oublis, etc... Le but est d'en tirer le meilleur profit possible.
Pour cela, nous allons devoir les analyser et tester sur de multiples méthodes. Ces méthodes, sont des algorithmes de ML. On retrouvera :

— Decision Tree
— Random Forest
— SVM
— Naive Bayes
— Regression Logistique
— KNN
— LDA / QDA
— Gradient Boosting

Ainsi, nous allons essayer de montrer les limites de ces algorithmes avec la frontière du théorique et de la pratique.

2 Première partie

Nous travaillons sur les datasets suivants : "Aggregation.txt", "flame.txt" et "spiral.txt". Après analyse de ces données, nous avons pu remarquer que les labels sont bien distribués mais la taille des données est relativement petite.
Ainsi, nous allons pouvoir utiliser l'accuracy comme moyen de comparaison. Cependant, pour améliorer les résultats nous allons utiliser la méthode du k-fold afin d'augmenter le nombre d'entrainement et de tests pour les algorithmes. L'utilisation du k-fold risque d'augmenter le temps d'exécution mais cela restera rapide à la vue de la taille des données.

Nom	Type	Taille	Valeur
aggregation	DataFrame	(788, 3)	Column names: 0, 1, 2
flame	DataFrame	(240, 3)	Column names: 0, 1, 2
spiral	DataFrame	(312, 3)	Column names: 0, 1, 2

FIGURE 1 – Descriptions des datasets

2.1 Dataset : Aggregation

Pour cette première partie, nous avons une base de presque 800 induvidus divisés en 7 classes.

On a pu obtenir les 100% d'accuracy.
Ce résultat a été obtenu grâce au Random Forest. Le Decision Tree l'a aussi obtenu. L'entraînement c'est effectué sur 80 folds. Rappelons que ces données sont artificielles et non bruités, donc normal d'obtenir des scores aussi élevés. Il est compliqué de parler d'overfitting ici.

Les figures suivantes nous montre les résultats obtenu sur cette base et la prédiction faite par le Random Forest.

```
LR: Accuracy : 0.989583 écart-type de : 0.039219
LDA: Accuracy : 0.988750 écart-type de : 0.049984
KNN: Accuracy : 0.997500 écart-type de : 0.015612
DTC: Accuracy : 1.000000 écart-type de : 0.000000
NB: Accuracy : 0.994861 écart-type de : 0.027439
RF: Accuracy : 1.000000 écart-type de : 0.000000
SVM: Accuracy : 0.997500 écart-type de : 0.015612
```

FIGURE 2 – Résultats des algorithmes sur le dataset Aggregation

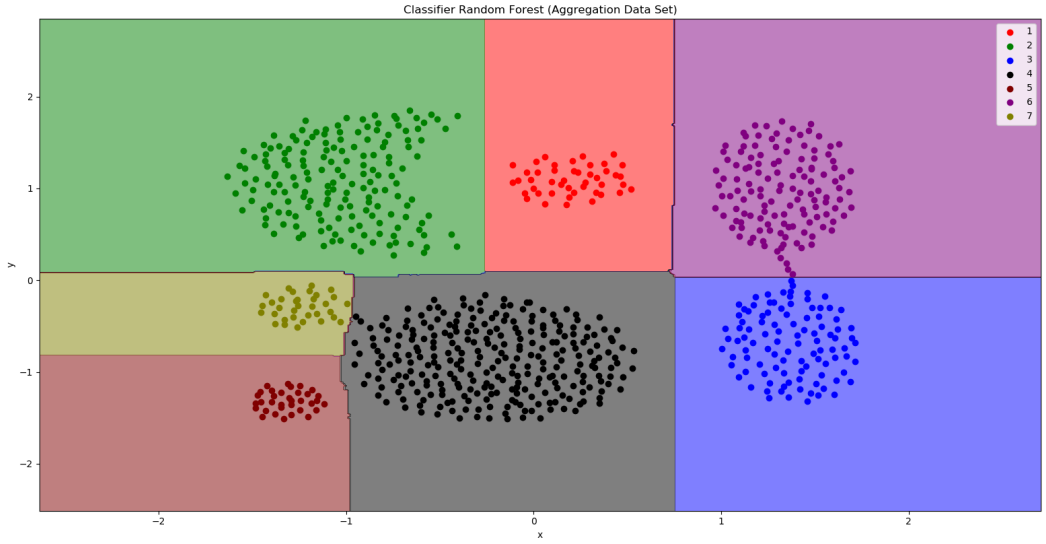

FIGURE 3 – Visualisation des classes prédites par le Random Forest

2.2 Dataset : Flame

La variable cible de cette base de données est binaire. À la vue des données, celles-ci ne sont pas linéaires. Donc les algorithmes linéaires ne devraient pas être utiles.

Nous avons pu obtenir les 100% d'accuracy avec le KNN avec le nombre de folds égal à 150 et n_neighbors=1. Nous savons que le nombre de voisin à 1 représente de l'overfitting, cependant, comme les données sont artificielles, cela ne pose pas de problème.

```
LR: Accuracy : 0.855556 (0.347566)
LDA: Accuracy : 0.863889 (0.340875)
KNN: Accuracy : 1.000000 (0.000000)
DTC: Accuracy : 0.966667 (0.179505)
NB: Accuracy : 0.950000 (0.217945)
SVM: Accuracy : 0.977778 (0.147406)
```

FIGURE 4 – Résultats des algorithmes sur le dataset Flame

Étonnamment, les algorithmes linéaires ne donnent pas de résultats si mauvais que prédit. Cependant il semble impossibles qu'ils atteingnent les 100% de réussite.

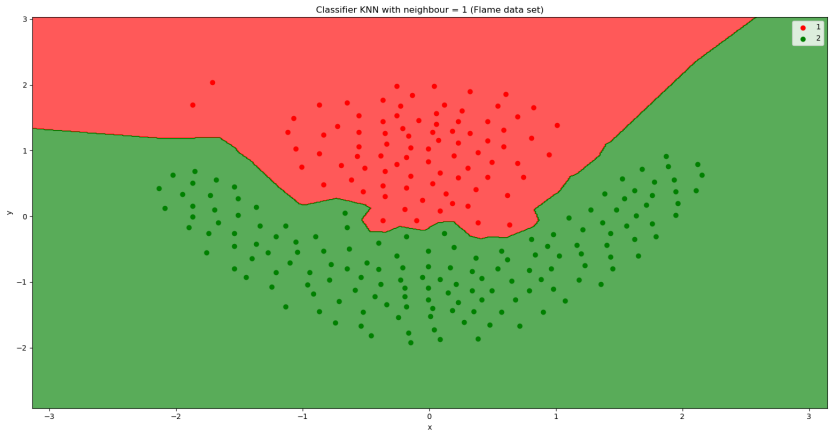

FIGURE 5 – Visualisation des classes prédites par KNN

2.3 Dataset : Spiral

La base de données spiral, est une base à 3 classes. Elle représente une spiral et les algorithmes linéaires ne seront pas utiles.

Voici le résultat avec k-fold = 10

```
LR: Accuracy : 0.000000 (0.000000)
LDA: Accuracy : 0.000000 (0.000000)
KNN: Accuracy : 0.372782 (0.144764)
DTC: Accuracy : 0.216028 (0.166514)
NB: Accuracy : 0.000000 (0.000000)
RF: Accuracy : 0.157863 (0.132716)
SVM: Accuracy : 0.154839 (0.200312)
```

FIGURE 6 – Spiral data set

Et voici le résultat avec k-fold = 110 et k-neighbours = 3

```
LR: Accuracy : 0.263636 (0.431337)
LDA: Accuracy : 0.260606 (0.432007)
KNN: Accuracy : 1.000000 (0.000000)
DTC: Accuracy : 0.960606 (0.166556)
NB: Accuracy : 0.257576 (0.430314)
RF: Accuracy : 0.969697 (0.123464)
SVM: Accuracy : 0.940909 (0.222005)
```

FIGURE 7 – Spiral data set

Le résultat est bien celui que l'on attendait, contrairement à la base Flame. En effet tous les algorithmes linéaires donnent des résultats très mauvais.

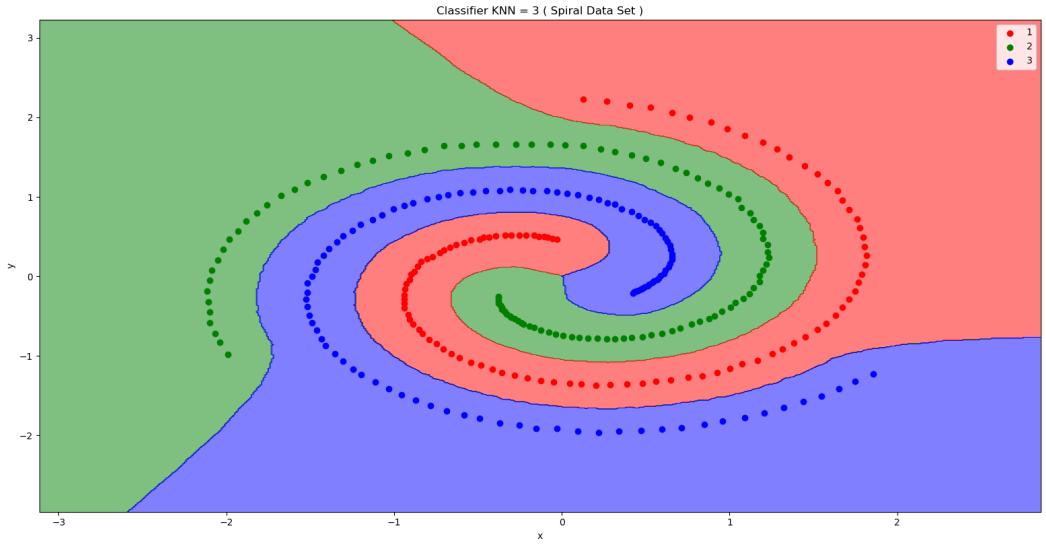

FIGURE 8 – Visualisation des classes prédites par KNN

3 Deuxième partie

Nous utilisons la base "VisaPremier" pour cette partie. Le but est de trouver le meilleur algorithme permettant de donner un score. Ce score représente l'appétence à la carte VISA Premier.

3.1 Analyse de la base

Premièrement nous avons regardé la taille de la base. Celle-ci compte 1073 individus pour 48 variables. Ce qui est une petite base.
Pour savoir si nous allons avoir besoin de "grossir" la base artificiellement ou bien utiliser le k-fold pour améliorer les résultats, nous avons regardé le taux de répartition de la variables target. Pour ce jeu de données, cette variable est "cartevpr".
Son taux de répartition est le suivant :

Label	Nombre de données
0	714
1	359

Ce qui représente environ 33% de labels positifs, ce qui est loin de l'équilibre 5050. Pour résoudre ce problème nous pouvons utiliser le Random over-sampling. Mais nous reviendrons dessus plus tard.

Puis nous avons analysé la base plus en profondeur. Nous avons analysé les différentes variables.

- Nous avons remarqué quelques incohérences, tel que le nombre de livret max égal à 4 et le nombre de carte max égal à 5. Mais sans plus de détails sur le nombre de carte on ne peut rien dire. En effet, ça peut-être dû à des pertes de cartes.
- Les colonnes nbop et nbvie ont des valeurs assez élevées en max, mais pas de signe d'abération.
- Nous avons aussi remarqué que nbimpaye était toujours nulle.

Puis nous avons mis les variables qualitative sous forme quantitative afin de pouvoir les utiliser.

Cela nous permet d'étudier la corrélation entre les variables. L'affichage de cette matrice est impossible du fait de la grosseur de celle-ci. [1] Dans un premier temps, nous avons éliminé des variables qui seraient trop corrélées entre elles et donc être inutile.

1. Vous trouverez joint au rapport le code sous forme d'un notebook qui retrace tout ce qui à été dit dans cette partie. Vous pouvez donc seulement regarder le notebook et sauter cette partie si vous le souhaitez.

Voici nos remarques :

— moycred3 est fortement corrélé avec 5 autres variables donc n'apportera rien en plus au modele. On peut donc la supprimer.
— nbbon et mtbon sont aussi fortement corrélés entre elles ainsi qu'avec d'autres variables, il est donc inutile de garder les deux.
— engagemt et engageml sont fortement corrélés, il semble que les engagements au long termes soit suffisament dominant parmis les engagements (court, moyen et long) pour se confondre avec les engagements totaux. On peut donc supprimer engageml.
— aveparfi et mtvie ainsi que mteparmo et moycredi sont fortement corrélés. Cependant nous n'avons pas forcement d'explications. C'est pourquoi nous decidons de garder les deux variables.

Puis nous avons choisi de ne garder que les variables qui sont corrélés à + ou - 0.2. Nous réduisons donc notre dataset à seulement à 15 variables.

3.2 Modèles

Nous allons utiliser les techniques suivantes :

— Arbre de decision simple avec méthode CART
— Random Forest
— SVM
— Gradient Boosting
— KNN
— Logistic Regression
— Naive Bayes
— LDA / QDA

Nous avons utilisé comme séparation des données 80% pour la base train et le reste pour le test.

3.3 Résultats

Modèle	Paramètres	Resultat AUC	Remarques
Arbre de decision	max_depth=8 min_samples_split=0.3 min_samples_leaf=0.3	0.76	
Random Forest	n_estimators=200 max_depth=10 min_samples_split=0.3 min_samples_leaf=0.1	0.88	
SVM	None	0.77	Test de tuning de paramètres trop long
Gradient Boosting	n_estimators=200 max_depth=10 min_samples_split=0.3 min_samples_leaf=0.1 learning_rate = 0.3	0.96	
KNN	None	0.83	AUC faible. On ne l'améliorera pas.
Logistic Regression	None	0.75	AUC faible. On ne l'améliorera pas.
Naive Bayes	None	0.78	AUC faible. On ne l'améliorera pas.
LDA / QDA	None	0.90	

Gradient Boosting nous apporte les meilleurs résultats [2] suivit de LDA et Random Forest.

Enfin, il suffira d'utiliser la fonction .predict_proba() pour obtenir des scores. De base la limite entre un score qui prédit un + ou un - est 0.50.

On pourrait tester, pour chaque algorithme, chaque seuil, pour le score, afin de trouver le meilleur et peut-être améliorer les résultats.

2. Selon la courbe ROC-AUC, disponibles sur les notebooks.

4 Troisième partie

Nous allons utiliser la base "CreditCard" pour cette partie. Le but est de trouver le meilleur algorithme permettant de trouver si une transaction est frauduleuse ou non.

4.1 Analyse de la base

Nous avons une base assez grosse. Elle comporte près de 300 000 données. Ce qui est plutôt convenable et permettra le bon entrainement des algorithmes. Cependant, nous observons une disparité des données flagrante ! Nous pouvons essayer l'under-sampling pour remédier à ce problème mais nous allons nous retrouver moins de 1000 données.

Valeur	Nombre de données
0	284315
1	492

Table 1 – Nombre de données pour chaque classe

Nous allons donc utiliser Smote pour l'over-sampling et éviter de perdre énormément de données. Cependant, nous allons énormément en rajouter. Smote va permettre de créer de nouvelles données très proches de celles existantes. Ce qui créer artificiellement de nouvelles données non identiques. Peut-être qu'une solution optimale serait de mixer l'over-sampling et l'undersampling ?

Par ailleurs, pour une analyse plus générale, toutes les données ont l'air normales. Pas de valeur aberrante ou manquante détectée. Cependant nous ne connaissons pas la signification des données. Donc il nous est impossible de détecter des aberrations. La variable cible est la dernière : "Class" En regardant les variables corrélés à 'Class', nous n'allons garder que V1, V3, V4, V7, V10, V11, V12, V14, V16, V17 et V18.

4.2 Test des algorithmes

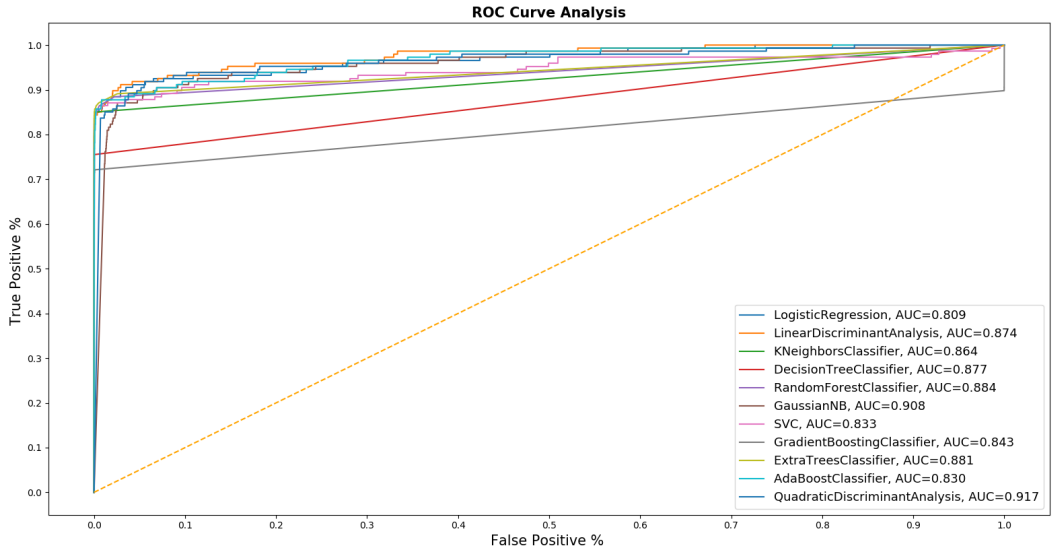

FIGURE 9 – Courbes ROC AUC sans enlever des variables et sans SMOTE

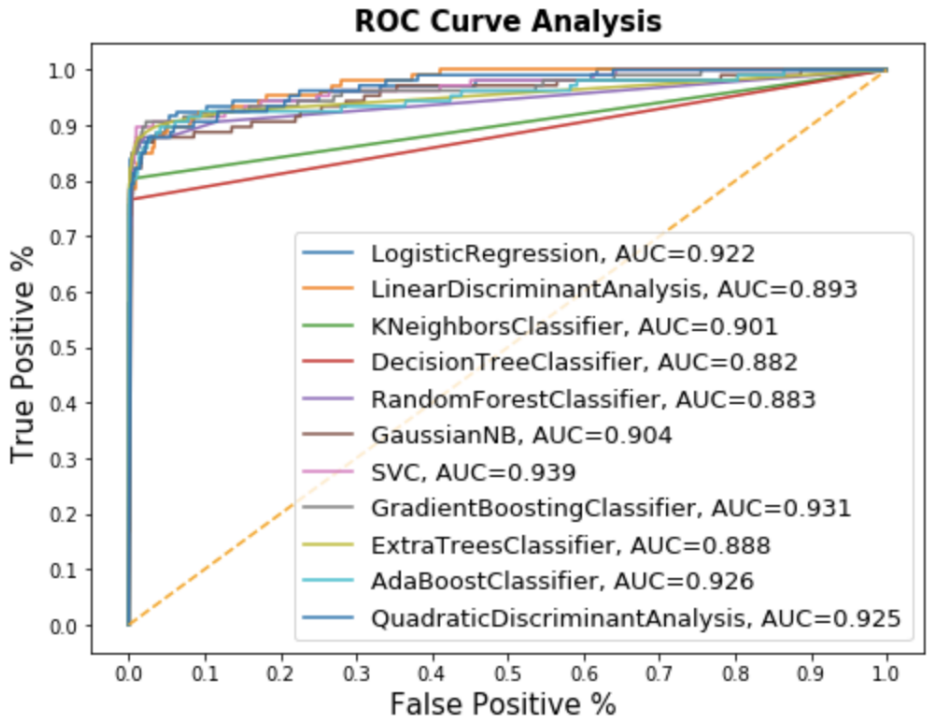

FIGURE 10 – Courbes ROC AUC en enlevant des variables

Nous pouvons remarquer que le fait d'enlever des variables, nous a permis d'améliorer les résultats de façon globale. Le fait d'avoir presque multiplié les individus par 2 doit aussi améliorer les résultats.

Voici les meilleurs algorithmes : SVM (rbf), Gradient Boosting et AdaBoost. Pour le gradient boosting et adaboost, nous allons essayer de les améliorer en choisissant les meilleurs paramètres. Nous ne le ferons pas sur SVM car le temps d'exécution est trop long pour faire du fine-tuning.

— learning_rates et n_estimators pour les 2
— max_depths et min_samples_splits pour Gradient Boosting

4.3 Améliorations

Nous avons utiliser les paramètres suivants :

Gradient Boosting	n_estimators=64 max_depth=4 min_samples_split=0.6 learning_rate=0.05
AdaBoost	n_estimators=64 learning_rate=0.2

Voici les résultats :

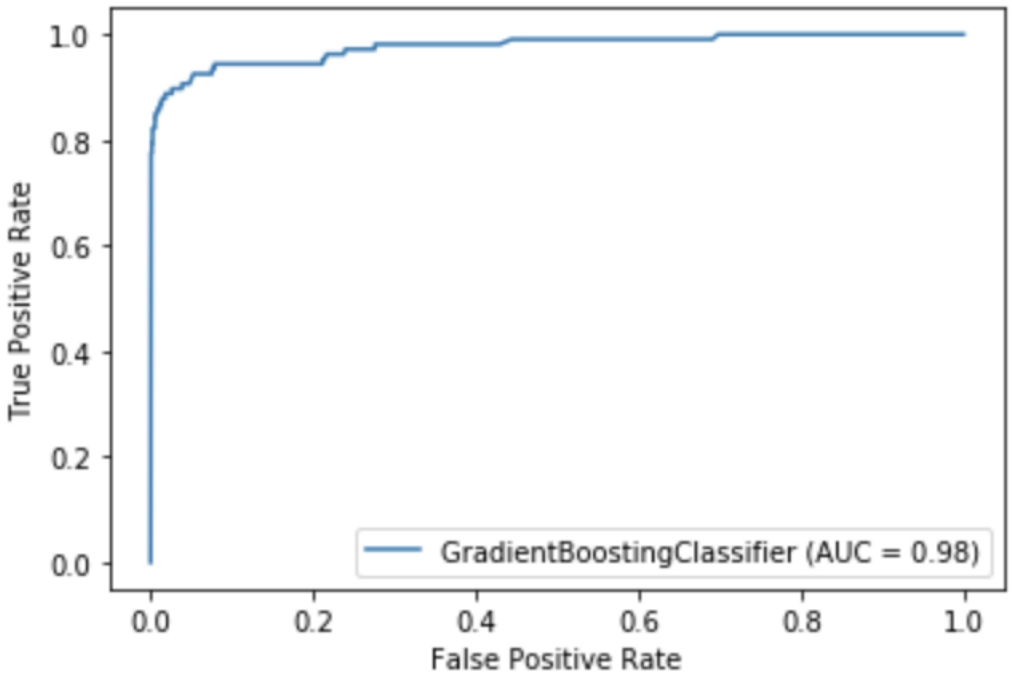

FIGURE 11 – Courbe ROC AUC GradientBoosting

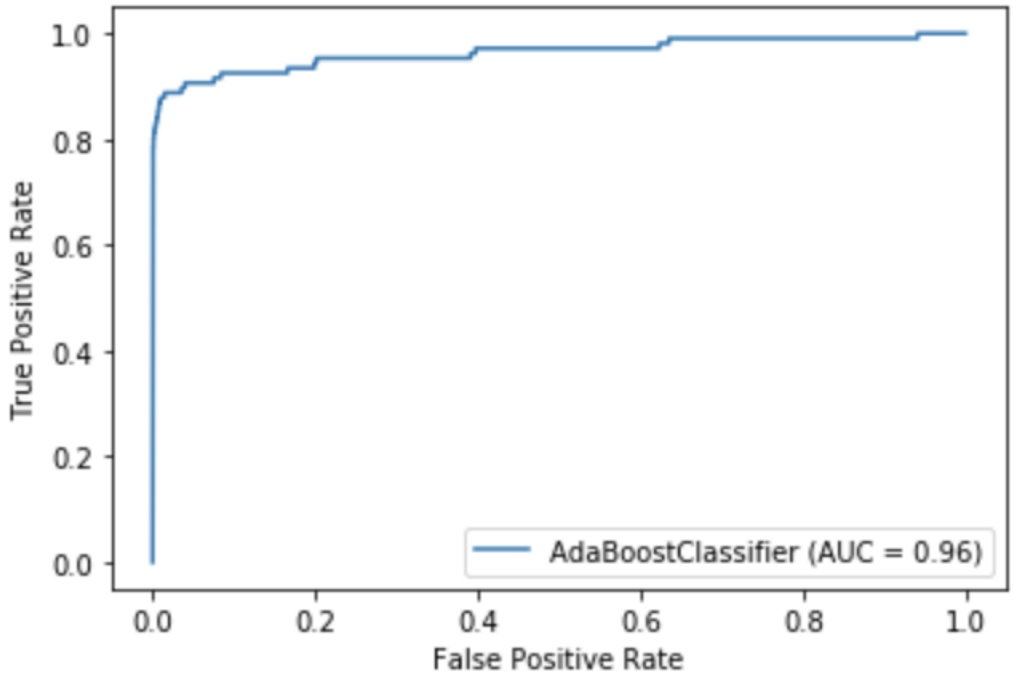

FIGURE 12 – Courbe ROC AUC AdaBoost

Notre choix de variables a amélioré les résultats.

Cependant un AUC à presque 1 est assez étonnant. Soit le problème etait trivial et donc facile à résoudre, soit le modèle overfit.

5 Conclusion

Dans ce projet nous avons pu manipuler plusieurs algorithmes de Machine Learning Supervisé. Nous les avons utiliser pour de la classification.
Nous avons utilisé des données artificielles et réelles.

Vous trouverez joint au projet les fichiers de code retraçant ce rapport avec plus de détails.

La principale chose que nous avons remarqué entre les 2 types de données, c'est que pour les données synthétiques nous n'avons pas besoin de penser à l'overfitting. On obtient donc des résultats excellent avec plusieurs algorithmes alors que dans la réalité il faut chercher et tuner les algorithmes pour pouvoir en tirer le meilleur de chaque.

Ainsi en théorie, les algorithmes sont faciles à utiliser et trouve la solution optimale très rapidement. Tandis que dans la réalité nous avons dû sélectionner des variables parmi toutes celles proposées. Mais aussi nettoyer les données afin de les rendre utilisables. Sans cette partie de 'cleaning' des données, nous ne pourrions utiliser les algorithmes.

A cause de la non parité des données, nous avons dû utiliser une méthode pour obtenir des données à bonne parité pour la variable cible. Pour la dernière base de données nous avons utilisé SMOTE qui permet de créer de nouvelles données différentes des existantes. C'est un concept qui intuitivement devrait être meilleur que le précédant. Nous avons cependant voulu utiliser les deux méthodes. Nous avons essayer de traiter de deux façon différentes les deux dernières parties. D'une part nous avons testé chaque algorithme et essayer de le tuner, tandis que pour la dernière base, nous avons tout essayer avec les valeurs par défauts pour tuner les 2 meilleurs. Nous pensons que la première manière est peut-être plus fiable mais nécessite beaucoup de temps tandis que la seconde est moins fiable mais beaucoup plus rapide.

Si nous devrions améliorer ce projet, nous pourrions essayer d'ajouter des réseaux de neurones pour essayer de prédire de façon plus précise. Même s'il faut garder en tête qu'un réseau de neurone n'est pas forcément meilleur que ce que nous avons utilisé ici.

Pour finir, ce projet nous à permis de pouvoir mettre en pratique les notions vu pendant le cours et nous a permis de nous rendre compte que le travail d'un Data Scientist ne se résume pas au simple fait d'appliquer des algorithmes.