
# Machine Learning

Tia ZOUEIN
Maria José Domenzain Acevedo

Master 2 Data Sciences

Novembre 2022

# Table des matières

# 1 Introduction

Le but de ce projet est d'évaluer la quantité de la glace trouvée dans un endroit donné en Groenlande à travers d'enregistrements d'infrasons. Le déplacement de larges volumes d'aire aboutit aux ondes acoustiques de basse fréquences dans l'atmosphère appelées infrasons. Ces ondes sont inaudibles pour les humains, car ces fréquences sont d'une mésure en dessous de 20Hz. Dans les données brutes, les 4 variables de sortie sont des variables quantitatives d'enregistrememnts d'infrasons. Dans ce projet on se centrera sur la variable cible Y4.

# 2 Présentation de la base de données

On s'intéresse premièrement à étudier la base de données sur laquelle on appliquera des algorithmes de Machine Learning. Elles proviennent de l'ECMWF (de l'acronyme de l'anglais European Centre for MEtium-Range Weather Forecasts).
Elle est consituée de 2556 observations et 14 variables. Les variables explicatives représentent le temps, la température de la mer, la vitesse du vent, la concentration de la glace dans la mer et le débit d'eau liquide du Groenland simulé par des modèles climatiques régionaux pour 5 régions. La variable cible représente les signaux infrasonores, nommée Y4 dans la base de données.

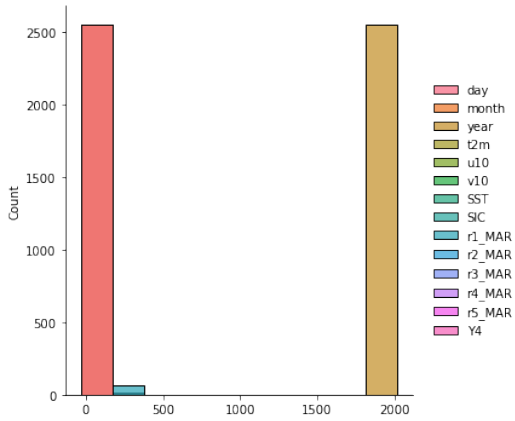


Figure 1 – Visualisation des données

Dans le but de tranformer la variable Y4 en variable binaire, on a choisit le seuil 0 : les valeurs qui sont donc déjà nulles restent nulles, et celles qui sont différentes de 0 seront égales à 1. Ceci est pour maximiser le nombre des observations qui sont dans la classe 1 et réduire le déséquilibre entre les classes qui peut causer des graves problèmes en appliquant des algorithmes de prédiction. Ainsi, 502 des observations sont de la classe 1 et 2054 de classe 0. De plus, on a décomposé la variable 'time' en trois colonnes : l'une pour le jour, la deuxième pour le mois et une troisième colonne pour l'année. Ceci dans le but d'avoir des variables quantitatives, puis on a supprimé la colone 'time'.

D'autre part, il n'y a pas de valeurs manquantes dans la base de données. En outre, comme certains algorithmes exigent la disctribution normale des variables, on a centré et réduit les données avant d'appliquer les algorithmes de machine learning.

# 3 Application des algorithmes

On a appliqué 12 algorithmes de Machine Learning dont on présentera les résultats et on comparera leur performance en se basant sur différents critères. Dans ce but, on a divisé les données en deux un échantillons d'apprentissage (80 % des données) pour entrainer le modèle et un échantillon test (20 %) pour tester la performance du modèle en faisant des predictions sur ce dernier qui contient des observations non vues par l'algorithmes.

## 3.1 Naive Bayes

Étant donné une base de données, on peut calculer la probabilité que les données appartiennet a une classe donnée en utilisant le théorème de Bayes [4] :

$$P(class|data) = \frac{P(data|class)P(class)}{P(data)}$$

Il exige l'indépendance et la normalité des variables. Mais d'après la matrice de corrélation, on a remarqué qu'il existe des variables fortement corréllées et qui correspondent aux variables représentantes les régions. Pour cela, on a enlevé les variables fortement corréllées et sélectionné les variables indépendantes avant d'appliquer l'algorithme.
La matrice de confusion permettant d'évaluer la prerformance du modèle est la suivante :

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 392 | FP= 19 |
| Valeur Actuelle = 1 | FN = 88 | TP= 13 |

Table 1 – Matrice de confusion - Naive Bayes

De cette matrice, on peut calculer la sensabilité du modèle, la spécificité, le f1-score, la précision et l'accuracy.
La sensibilité mesure la capacité du modèle à détecter correctement l'ensemble des signaux de classe 1 parmi ceux qui sont qui sont actuellement de classe 1.
La précision mesure la capacité du modèle à détecter correctement l'ensemble des signaux de classe 1 parmi ceux qui sont prédits de classe 1.
La spécificité mesure la capacité du modèle à détecter correctement l'ensemble des signaux de classe 0.
Le F1-score combine la précision et la sensitivity. Il est privilegié sur l'accuracy pour des données déséquilibrées et où la classe 0 est majoritaire. En fait, l'accuracy prend en considération TN alors que le F1-score non, et la majorité de vrai négatifs faussent la perception de la performance du modèle dans le cas de déséquilibre.

| | |
|---|---|
| Sensitivity = T P / T P + F N | 12.87 |
| Specificity = TN / TN + FP | 95.37 |
| Precision TP / TP + FP | 40.62 |
| F1-score = 2*(Precision*sensitivity)/(Precision + Sensitivity) | 19.54 |
| Accuracy (TP + TN)/(TP + TN + FP + FN) | 79.10 |

TABLE 2 – Scores de performance - Naive Bayes

On remarque que l'accuracy du modèle est élevée. La majorité des prédictions correctes sont pour la classe 0 (TN = 392), alors que 13 parmi 101 ont été correctement prédits de la classe 1.

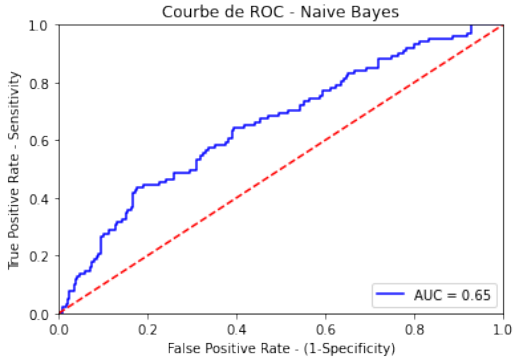


FIGURE 2

La courbe ROC représente le comportement d'un algorithme de classification à deux classes pour tous les seuils de détection possibles. Elle permet de décrire la performance du modèle en traçant le couple (1-spécificité, sensibilité) selon un seuil de classification. Lorsque le seuil augmente, la sensibilité diminue et la spécificité augmente et vise-versa. On peut donc optimiser soit la sensibilité soit la spécificité, et non pas les deux ensemble.

La ligne droite représente ce que donnerait un classifieur qui classe au hasard, et donc, il obtient autant de faux positifs que de vrais positifs quel que soit le seuil choisi. En dessous, nous ferions moins bien que le hasard. Au dessus, le classifieur est meilleur. Plus la courbe a des valeurs élevées, moins le modèle fait d'erreur.
Pour quantifier globalement les performances du classifieur quel que soit le seuil, on calcule l'AUC qui correspond à l'aire de la surface sous la courbe de ROC. Il est égale à 100 % pour un modèle parfait et 50 % pour un modèle non-informatif (aléatoire).
Pour le classifieur Naive Bayes, l'AUC vaut 65.41 %, il fait donc mieux que l'aléatoire.

## 3.2 LDA

De l'anglais *Linear Discriminant Analysis*, LDA est un algorithme de classification linéaire. En utilisant l'algèbre lineaire, cet algorithme calcule la moyenne et l'écart des variables d'entrée à travers une décomposition matricielle. Puis, les prédictions sont

faites en estimant la probabilité qu'un nouveau exemple apartient a une classe donnée étant donnée leurs valeurs calculées.
Cet algorithme suppose que les variables d'entrée suivent une loi normal et qu'elles ne sont pas correlées. Pour cela, on a fait une sélection des variables comme dans le cas de Naive Bayes.[3]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 411 | FP= 0 |
| Valeur Actuelle = 1 | FN = 101 | TP= 0 |

TABLE 3 – Matrice de confusion - Linear Discriminant Analysis

Les résultats de ce modèle ne sont pas du tout satisfaisants. Il n'arrive à prédire aucune fois la classe 1. Ceci peut être à cause du déséquilibre entre les deux classes, en fait, la classe 0 est majoritaire.
Donc, malgré que l'accuracy du modèle est élevée (80 %), le modèle n'a pas une bonne performance, car dans ce cas, le f1-score, ainsi que la sensitivité, sont nulles.

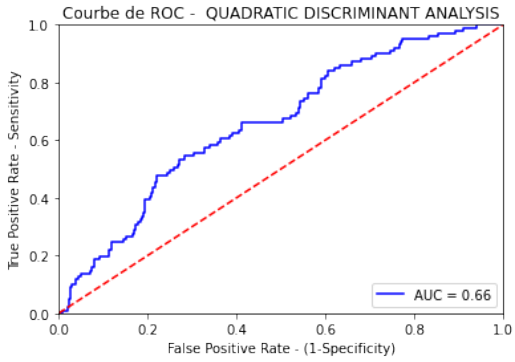


FIGURE 3

L'AUC vaut 64.32 %. Il est un peu élevé car le modèle prédit correctement la classe 0 (Specificité vaut 93 %).

### 3.3 QDA

De l'anglais *Quadratic Discriminant Analysis*, le QDA est très simmilaire au LDA sauf qu'on relache l'hypothèse que la moyenne et la covariance de toutes les classes est égale. Ainsi, il faut calculer ces dernières séparemment. [1]
La matrice de confusion permettant d'évaluer la prerformance du modèle est la suivante :

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 389 | FP= 22 |
| Valeur Actuelle = 1 | FN = 87 | TP= 14 |

Table 4 – Matrice de confusion - QDA

| | |
|---|---|
| Sensitivity | 13.86 |
| Specificity | 94.64 |
| Precision | 38.88 |
| F1-score | 20.43 |
| Accuracy | 78.71 |

Table 5 – Scores de performance - Quadratic Discriminant Analysis

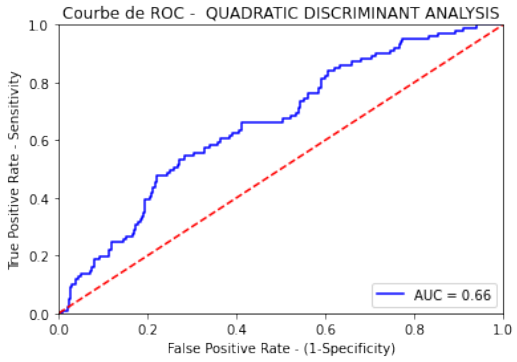


Figure 4

l'AUC vaut 66 %.

### 3.4 KNN

De l'anglais, K-Nearest Neighbours, KNN est un algorithme non-parametrique qui se sert de la proximité pour faire des classifications ou predictions à propos d'une donnée individuel. Il peut être utilisé pour classification ainsi que pour regression. Pour classification, la variable est élue par majority vote, où le label qui est le plus representé aux alentours du donnée est celui qui est utilisé. Pour l'implementation, on prend k voisins du point que l'on cherche. Ensuite on définit la distance que sera utilisé (Euclidienne, Manhattan, Minkowski, Hamming, etc.) La dificulté se trouve dans le choix de k, car de cela dépend si le modèle sera surapprenti ou sousapprenti. [7]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 385 | FP= 26 |
| Valeur Actuelle = 1 | FN = 88 | TP= 13 |

TABLE 6 – Matrice de confusion - KNN

| | |
|---|---|
| Sensitivity | 12.87 |
| Specificity | 93.6 |
| Precision | 33.3 |
| F1-score | 18.57 |
| Accuracy | 77.73 |

TABLE 7 – Scores de performance - KNN

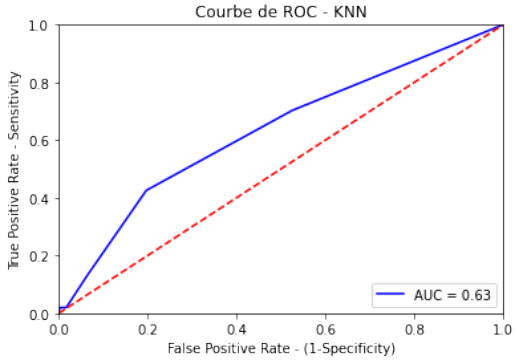


FIGURE 5

l'AUC vaut 63%.

### 3.5 Logistique

La régression logistique permet de prédire la probabilité qu'un événement arrive (valeur de 1) ou non (valeur de 0) à partir de l'optimisation des coefficients de régression. Ce n'est donc pas la réponse binaire qui est directement modélisée, mais la probabilité de réalisation d'une des deux modalités. Cette probabilité, est modélisée par une courbe sigmoïde, bornée par 0 et 1, et elle est définie par la fonction logistique suivante :

$$P = \frac{\exp(\beta_0 + \beta_1.x1 + \beta_2.x2 + .... + \beta_k.x_k)}{1 + \exp(\beta_0 + \beta_1.x1 + \beta_2.x2 + .... + \beta_k.x_k)} \quad (1)$$

avec $\beta_i$ les coefficients du modèle à estimer. Ce modèle n'est pas linéaire puisque la probabilité ne s'exprime pas comme une addition des effets des différentes variables explicatives.

Pour cela, on utilise la fonction logit qui permet de mettre en relation la probabilité de réalisation (bornée entre 0 et 1), et la combinaison linéaire de variable explicatives :

$$logit(P) = \log(\frac{P}{1-P}) = \sum_{i=1}^{k} \beta_i.Xi \tag{2}$$

Cette valeur est extrêmement élevée pour une probabilité proche de 1. Par contre, pour des probabilités au-dessous de 50%, les valeurs sont diminués pour être très faible pour des probabilités proches de 0.
En appliquant l'algorithme de la régression logistique, un problème est apparu : il ne prédit aucune fois la classe 1. Pour pallier à ça, on a codé une fonction qui nous donne le seuil optimale (0.18) qui maximise la sensibilité et la spécificité, puis on a appliqué l'algorithme. Les résultats sont beaucoup plus satisfaisants que dans le cas où le seuil est celui par défaut 50 %.

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 169 | FP= 242 |
| Valeur Actuelle = 1 | FN = 26 | TP= 75 |

Table 8 – Matrice de confusion - Régression logistique

| | |
|---|---|
| Sensitivity | 74.25 |
| Specificity | 41 |
| Precision | 23.65 |
| F1-score | 35.88 |
| Accuracy | 47.65 |

Table 9 – Scores de performance - Régression Logistique

Malgré que l'accuracy est trop faible, le F1-score a augmenté et le modèle a prédit mieux la classe 1. Par contre, la spécificité a diminué. Donc en diminuant le seuil, on maximise la sensibilité et on minimise la spécificité. Selon l'objective de la prédiction, on choisit le seuil.

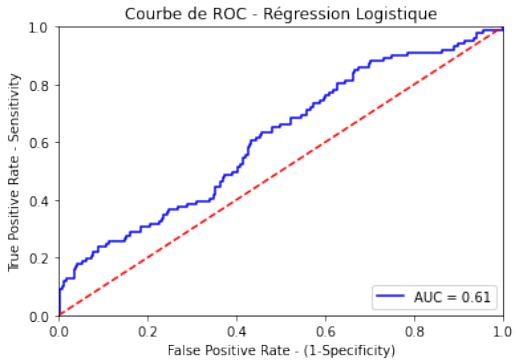


FIGURE 6

l'AUC vaut 61.28 %.

### 3.6 Decision Tree

Il s'agit d'une méthode non-parametrique utilisée pour classification et prediction. Un decision tree est une structure d'arbre où chaque noeud interne représente un test d'une variable, chaque branche represente le résultat du test et chaque noeud terminal represente les valeurs des variables cibles. L'arbre utilise les noeuds et les feuilles pour arriver à une décision. Tout en haut de l'arbre se trouve l'ensemble de la base de données et au fur et à mesure qu'on avance, cela se divise en ensembles de plus en plus petits.[9]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 336 | FP= 75 |
| Valeur Actuelle = 1 | FN = 66 | TP= 35 |

TABLE 10 – Matrice de confusion - Decision tree

| | |
|---|---|
| Sensitivity | 34.65 |
| Specificity | 81.75 |
| Precision | 31.81 |
| F1-score | 33.64 |
| Accuracy | 72.46 |

TABLE 11 – Scores de performance - Decision tree

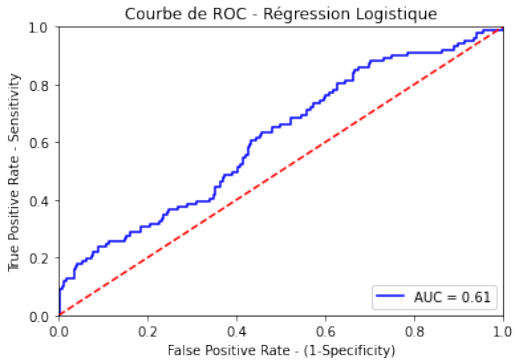


FIGURE 7

l'AUC vaut 58 %.

### 3.7 Bagging

Il s'agit d'un algorithme qui peut être utilisé pour la regression comme pour la classification et qui permet d'éviter le surapprentissage. Pour l'implémentation, il y a trois étapes principales. La première, à travers le bootstrapping, plusieurs échantillons sont crées. À travers une sélection aléatoire avec reemplacement, cela produit plusieurs sous-ensembles du jeu de données d'apprentissage. Ces-derniers servent ensuite à créer des modèles de base. La deuxième étape s'agit d'un jeu de données d'apprentissage en parallelle du échantillon provenant du bootstrapping à travers l'utilisation de *weak learners* ou *base learners*. Finallement, la dernière étape d'aggregation, des predictions finales sont faits à travers le mélange des predictions de chaque modèle. Un exemple de cette méthode est le Random Forest. [6] [2]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 396 | FP= 15 |
| Valeur Actuelle = 1 | FN = 86 | TP= 15 |

TABLE 12 – Matrice de confusion - Bagging

| | |
|---|---|
| Sensitivity | 14.85 |
| Specificity | 96.35 |
| Precision | 50 |
| F1-score | 22.9 |
| Accuracy | 80.27 |

TABLE 13 – Scores de performance - Bagging

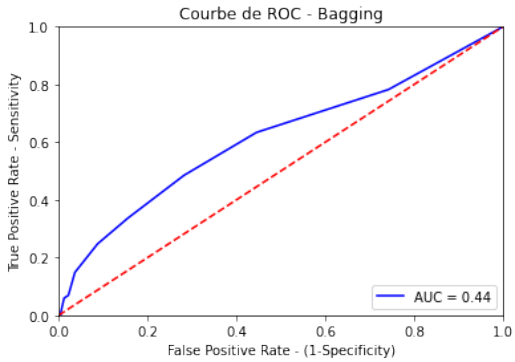


FIGURE 8

l'AUC vaut 61.56 %.

### 3.8 Random Forest

Cette méthode s'agit d'une extension de Bagging et de Decision Trees. Un random forest est constitué d'un ensemble d'arbres de décision indépendants. Chaque arbre dispose d'une vision parcellaire du problème du fait d'un double tirage aléatoire : un tirage aléatoire sur les observations selon lequel est construit l'arbre et un autre sur les variables sur lesquelles sont effectuées la segmentation. A la fin, tous ces arbres de décisions indépendants sont assemblés. La prédiction faite par le random forest pour des données inconnues est alors celle qui a eu la majorité de vote.

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 396 | FP= 15 |
| Valeur Actuelle = 1 | FN = 85 | TP= 16 |

TABLE 14 – Matrice de confusion - Random Forest

| | |
|---|---|
| Sensitivity | 15.84 |
| Specificity | 96.35 |
| Precision | 51.61 |
| F1-score | 24.24 |
| Accuracy | 80.46 |

TABLE 15 – Scores de performance - Random Forest

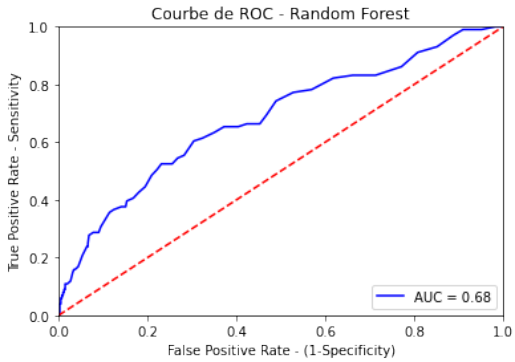


FIGURE 9

l'AUC vaut 68.18 %.
L'indice de Gini mesure le niveau d'inégalité de la répartition de la variable réponse dans le noeud. Il indique donc l'hétérogénéité ou l'homogénéité des noeuds. Plus il est élevé, plus les données sont uniformes. Ainsi, il permet de choisir la variable de séparation à chaque noeud. La diminution moyenne de l'indice de Gini est une mesure de la façon dont chaque variable contribue à l'homogénéité des noeuds et des feuilles dans la forêt aléatoire résultante. Elle indique la diminution de l'impureté du noeud. Plus la diminution moyenne du score de Gini est élevée, plus l'importance de la variable dans le modèle est élevée.

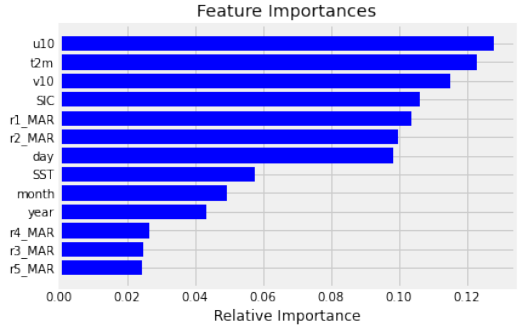


FIGURE 10 – Les variables importantes

Dans notre cas, on voit que la variable la plus importante pour faire la discrimination entre les variables par rapport à Y4 est u10 qui mesure la vitesse du vent, et la moins importante est $r3_{MAR}$ qui indique la région.

## 3.9 Extra Trees

Extra Trees est une methode ensembliste qui prend un ensemble de decision trees, elle est associé aussi au Bagging et Random Forest. Cet algorithme crée un large nombre de decision trees non taillés, ensuite les predictions sont faites à travers majority voting

pour la classification et la moyenne des predictions des decision trees pour la regression. La différence avec Random Forest et Bagging est que cet algorithme crée les decision trees à partir de l'ensemble de données, et non pas à partir du bootstrap sampling.

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 382 | FP= 29 |
| Valeur Actuelle = 1 | FN = 73 | TP= 28 |

Table 16 – Matrice de confusion - Extra trees

| | |
|---|---|
| Sensitivity | 27.72 |
| Specificity | 92.94 |
| Precision | 49.12 |
| F1-score | 35.44 |
| Accuracy | 80.07 |

Table 17 – Scores de performance - Extra trees

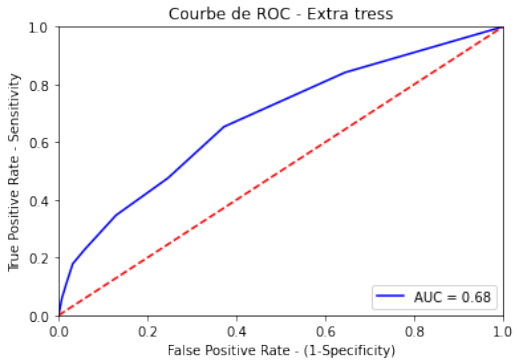


Figure 11

l'AUC vaut 68 %.

### 3.10 Adaboost

Il s'agit d'un algorithme de type Boosting, et reçoit son nom de l'anglais *Adaptative Boosting*. Cet algorithme crée un modèle et donne en un premier temps le même poids à l'ensemble de points de la base de données. Ensuite il donne un poids plus grand aux points qui ne sont pas bien classifiés. Ensuite les points avec un poids supérieur ont priorité sur les autres lors de la création de l'algorithme suivant. L'algorithme repète ce procès et continue à entrainer des modèles jusqu'à ce que l'erreur est minimale. [8]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 403 | FP= 8 |
| Valeur Actuelle = 1 | FN = 91 | TP= 10 |

Table 18 – Matrice de confusion - Adaboost

| | |
|---|---|
| Sensitivity | 9.9 |
| Specificity | 98.05 |
| Precision | 55.55 |
| F1-score | 16.80 |
| Accuracy | 80.66 |

Table 19 – Scores de performance - Adaboost

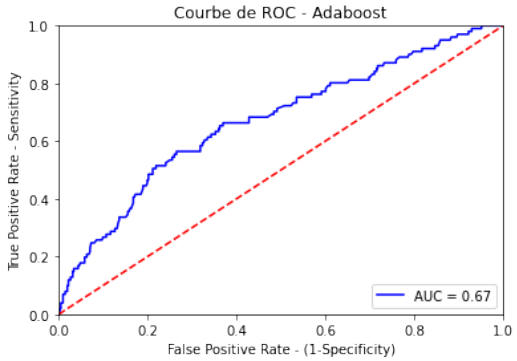


Figure 12

l'AUC vaut 67.07 %.

### 3.11 Gradient boosting

Cette méthode est une généralisation d'AdaBoost. On construit des ensembles à partir des modèles d'arbres de decision. Chaque arbre est rajouté individuelement de sorte qu'on puisse les adapter et corriger les erreurs de prediction faites par les modèles précedents. L'adaptation est faite à travers une fonction objectif et un algorithme d'optimisation par *gradient descent*. De cette forme, le gradient est minimisé au fur et à mésure que le modèle est ajousté. [**GragBoost**]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 406 | FP= 5 |
| Valeur Actuelle = 1 | FN = 97 | TP= 4 |

Table 20 – Matrice de confusion - Gradient boosting

| | |
|---|---|
| Sensitivity | 3.96 |
| Specificity | 98.78 |
| Precision | 44.4 |
| F1-score | 7.27 |
| Accuracy | 80.07 |

TABLE 21 – Scores de performance - Gradient boosting

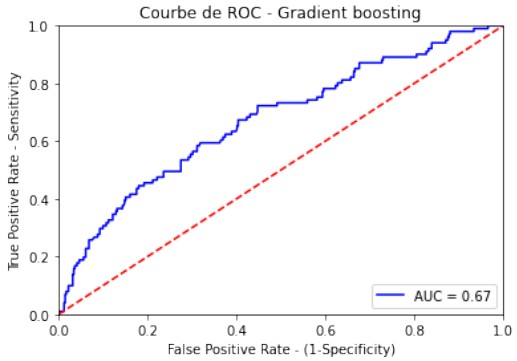


FIGURE 13

l'AUC vaut 67.32 %.

### 3.12 Stacking

Un dernier modèle ensembliste s'agit du *Stacking*. Cet modèle fait un mélange de plusieurs models de classification ou de regression à travers un meta-classifier ou un meta-regresseur. Pour implémenter cette méthode il faut commencer par diviser le jeu de données d'apprentissage en utilisant *K-fold cross validation*. Ensuite, un modèle de base est ajusté pour les premières K-1 parties et des predictions sont faites pour la dernière partie (K). Le modèle de base est ensuite ajusté sur l'ensemble du jeu de données d'apprentissage pour calculer les résultats. Ainsi, les predictions du jeu de données d'apprentissage sont utilisées comme *features* pour le deuxième niveau du modèle. Le deuxième modèle est utilisé pour faire des prédictions du jeu de données d'apprentissage.[5]

| | Valeur Prédite = 0 | Valeur Prédite = 1 |
|---|---|---|
| Valeur Actuelle = 0 | TN = 407 | FP= 4 |
| Valeur Actuelle = 1 | FN = 94 | TP= 7 |

TABLE 22 – Matrice de confusion - Stacking

| | |
|---|---|
| Sensitivity | 6.93 |
| Specificity | 99.02 |
| Precision | 63.63 |
| F1-score | 12.5 |
| Accuracy | 80.85 |

TABLE 23 – Scores de performance - Stacking

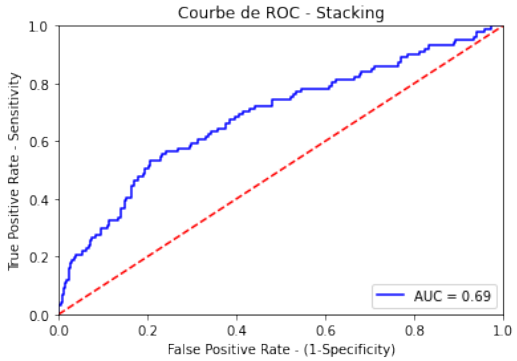


FIGURE 14

L'AUC vaut 69 %

## 4 Conclusion

On peut remarquer qu'il existe des modèles dont leur performance est bonne et d'autres qui ne l'est pas. Pour comparer la performance de tous les modèles, on a préféré de nous basé sur le F1-score car les données sont déséquilibrées et le TN qui est pris en considération par l'accuracy fausse la perception de la performance du modèle dans notre cas. Donc, parmi les modèles qui ont une bonne performance, on cite : Naive Bayes, Decision tree et le plus performant Extra Trees qui possède le plus grand F1-score(48.93 %).

Par contre, le modèle "linear discriminant analysis" est le plus mauvais parmi les modèles, car il n'arrive à prédire que la classe majoritaire. De plus, l'algorithme gradient boosting révèle aussi un F1-score qui n'est pas beaucoup satisfaisant(7.27 %).

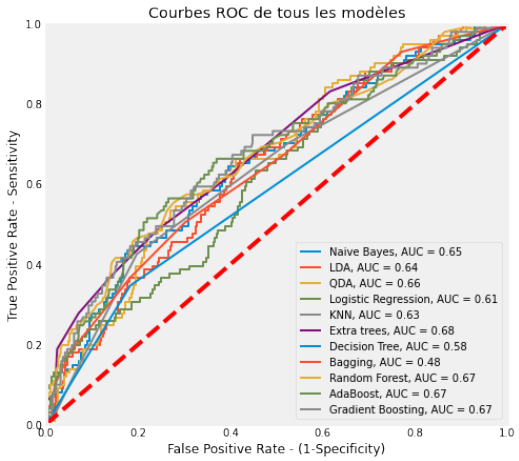


FIGURE 15

De plus, en comparant l'aire sous la courbe de ROC des modèles, on peut remarquer que le modèle Extra Trees possède le plus grand AUC, ce qui nous confirme qu'il est le plus performant. Puis, en comparant les AUC, on peut connaitre et comparer les modèles entre eux.

Ainsi, on conclut que le déséquilibre d'un jeu de données peut biaisé la performance d'un algorithme de machine learning. On peut appliquer la méthode de sous-échantillonnage des données d'apprentissage puis prédire sur les données test et évaluer les résultats. Ou bien, on peut faire la méthode de sur-échantillonnage (oversampling) de la classe minoritaire en créant des observatons de classe 1. Enfin, on peut faire comme on a fait pour la régression logistique, changer le seuil de classification pour améliorer les prédictions.

## Références


[1] @pawangfg. *Quadratic Discriminant Analysis*. 2022. URL : https://www.geeksforgeeks.org/quadratic-discriminant-analysis/ (visité le 2022).

[2] @soumya7. *Bagging vs Boosting in Machine Learning*. 2022. URL : https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/ (visité le 2022).

[3] Jason Brownlee. *Linear Discriminant Analysis with Python*. 2020. URL : https://machinelearningmastery.com/linear-discriminant-analysis-with-python/ (visité le 2022).

[4] Jason Brownlee. *Naive Bayes Classifier from Scratch in Python*. 2019. URL : https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/ (visité le 2022).

[5] Avik Dutta. *Stacking in Machine Learning*. 2019. URL : https://www.geeksforgeeks.org/stacking-in-machine-learning/ (visité le 2022).

[6] IBM Cloud Education. *Bagging*. 2021. URL : https://www.ibm.com/cloud/learn/bagging (visité le 2022).

[7] IBM Cloud Education. *K-Nearest Neighbors Algorithm*. URL : https://www.ibm.com/topics/knn (visité le 2022).

[8] Anshul Saini. *AdaBoost Algorithm - A Complete Guide for Beginners*. 2021. URL : https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/ (visité le 2022).

[9] Roberto Iriondo Saniya Parveez. *Decision Trees in Machine Learning(ML) with Python Tutorial*. 2021. URL : https://pub.towardsai.net/decision-trees-in-machine-learning-ml-with-python-tutorial-3bfb457bce67 (visité le 2022).