PROJET TECHNIQUES AVANCEES d'APPRENTISSAGE STATISTIQUE

Rapport – Mai 2021

Forêts Aléatoires et Boosting d'arbres de Décision

Auteurs:
Lilia BEN BACCAR
Emmanuel KAMLA FOTSING

Encadrant:
Stéphan CLEMENCON

Année scolaire 2020 - 2021

Contents

Introduction

Deux types de techniques ont été proposées comme développement à partir des arbres de décision : les forêts aléatoires et le boosting d'arbres de décision. Le but de ce projet est de comparer les niveaux de performances obtenu avec ces deux méthodes.

1 Forêts Aléatoires

Le bagging ou l'agrégation boostrap est une technique permettant de réduire la variance d'une fonction de prédiction estimée. Cette approche semble particulièrement bien fonctionner pour les méthodes à haute variance et à faible biais, telles que les arbres. Pour la régression, il suffit d'ajuster plusieurs fois le même arbre de régression à des versions échantillonnées des données d'entraînement, et de faire la moyenne des résultats. Pour la classification, un ensemble d'arbres émet chacun un vote pour la classe prédite.

L'approche des forêts aléatoires (Breiman, 2001) est une modification du bagging qui construit une grande collection d'arbres dé-corrélés, puis en fait la moyenne. Sur de nombreux problèmes, les performances des forêts aléatoires sont très similaires à celles du boosting et elles sont plus simples à entraîner et à paramétrer.

Algorithm 1: Forêts aléatoires pour la régression ou la classification

1. Pour $b = 1$ jusqu'à B:
   (a) Tirer un échantillon bootstrap Z^* de taille N à partir des données d'entraînement.
   (b) Construire un arbre de forêt aléatoire T_b pour l'échantillon tiré, en répétant récursivement les étapes suivantes pour chaque nœud terminal de l'arbre, jusqu'à ce que la taille minimale des nœuds n_{min} soit atteinte.
      i. Selectionner aléatoirement m variables à partir des p variables.
      ii. Choisir la meilleure variable / le meilleur point de séparation parmi les m.
      iii. Séparer le noeud en deux noeuds filles.
2. Produire l'ensemble d'arbres $\{T_b\}_1^B$.

Pour faire la prédiction d'une nouvelle observation x:

Regression: $\hat{f}_{rf}^B(x) = 1/B \sum_{b=1}^B T_b(x)$

Classification: Soit $\hat{C}_b(x)$ la prédiction de classe du $b$ème arbre de la forêt aléatoire. Alors : $\hat{C}_{rf}^B(x) =$ *vote majoritaire* $\{\hat{C}_b(x)\}_1^B$

Soit $\left(\hat{h}(\cdot, \Theta_1), \ldots, \hat{h}(\cdot, \Theta_q)\right)$ une collection de prédicteurs par arbres, avec $\Theta_1, \ldots, \Theta_q$ q variables aléatoires i.i.d. indépendantes de $\mathcal{L}_n$. Le prédicteur des forêts aléatoire $\hat{h}_{RF}$ est

obtenu en agrégant cette collection d'arbres aléatoires de la façon suivante:

- $\hat{h}_{RF}(x) = \frac{1}{q}\sum_{l=1}^{q}\hat{h}(x,\Theta_l)$ (moyenne des prédictions individuelles des arbres) en régression,
- $\hat{h}_{RF}(x) = \arg\max_{1\leq k\leq K}\sum_{l=1}^{q}\mathbb{1}_{\hat{h}(x,\Theta_l)=k}$ (vote majoritaire parmi les prédictions individuelles des arbres) en classification,

Le terme forêt aléatoire vient du fait que les prédicteurs individuels sont explicitement des prédicteurs par arbre et que chaque arbre dépend d'une variable aléatoire supplémentaire (c'est-à-dire en plus de $\mathcal{L}_n$). Une forêt aléatoire est l'agrégation d'une collection d'arbres aléatoires.

2 Boosting d'arbres de décision

Les arbres de décision sont connus pour souffrir de biais et de variance (Large Bias : arbres simples ; Large Variance : arbres complexes). Pour remédier à ce problème deux types de techniques ont été proposées notamment le Boosting d'arbres de décision. Introduite par Schapire(1990), le Boosting est une technique ensembliste qui consiste à agréger des modèles élaborés séquentiellement sur un échantillon d'apprentissage dont les poids des individus mal prédits sont corrigés au fur et à mesure. En d'autres termes le Boosting d'arbres de décision combine plusieurs arbres de décision pour produire de meilleures performances prédictives qu'en utilisant un seul arbre de décision. "Un groupe d'apprenants faibles se réunit pour former un apprenant fort".

2.1 AdaBoost

Algorithm 2: AdaBoost

1. Initialisation les poids des observations $\omega_i = \frac{1}{N}$, $i = 1, 2, ..., N$
2. Pour $m = 1, ..., M$ avec M le nombre maximal d'arbres.
   (a) Ajuster un classifieur $G_m(x)$ sur les données d'entraînement en utilisant les poids ω_i
   (b) Calcul: $err_m = \frac{\sum_{i=1}^{N}\mathbb{I}(y_i \neq G_m(x_i))}{\sum_{i=1}^{N}\omega_i}$
   (c) calcul: $\alpha_m = \log((1 - err_m)/err_m)$
   (d) mise à jour $\omega_i = \omega_i \exp\left[\alpha_m \mathbb{I}(y_i \neq G_m(x_i)\right]$
3. Output $G(x) = sign\left[\sum_{m=1}^{M}\alpha_m G_m(x_i)\right]$

Introduit par Freund and Schapire (1997), AdaBoost est l'algorithme de Boosting le plus populaire. En considerant un problème de classification et une matrice X de variables explicatives, un classifieur G(X) prend une des deux valeurs $-1, 1$. Le taux d'erreur sur l'échantillon d'entraînement est:

$$\bar{err} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i \neq G(x_i))$$

Plus généralement, les expansions de fonctions de base prennent la forme:

$$f(x) = \sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

où $\beta_m, m = 1, 2, ..., M$ sont les coefficients d'expansions, et $b(x, \gamma) \in \mathbf{R}$ sont des simples fonctions multivariées.

Algorithm 3: Algorithme Forward Stagewise Additive Modeling

1. Initialisation $f_0(x) = 0$
2. Pour $m = 1, 2, ..., M$:
   (a) Calculer:
   $$(\beta_m, \gamma_m) = \arg\min \sum_{i=1}^{N}\mathbb{L}(y_i, f_{m-1}(x_i) + \beta b(x_i;\gamma)$$
   (b) Fixer $f_m(x) = f_{m-1}(x) + \beta_m b(x;\gamma_m)$

En effet, en utilisant une fonction de perte exponentielle $L(y, f(x)) = \exp(-yf(x))$ il y a une équivalence entre l'algorithme Adaboost et algorithme Forward Stagewise Additive Modeling.

2.2 Gradient Boosting

Cet algorithme prend en compte à la fois le modèle additif, la fonction de coût et les apprenants faibles. Contrairement à AdaBoost, le Gradient Boosting peut être utlisé avec toute fonction différentiable et est très robuste aux "outliers".
Soit L une fonction de perte différentiable pour le modèle F (F est considéré ici comme le modèle Classification And Regression Trees).

Algorithm 4: Gradient Boosting

1. Initialisation du modèle par une constante: $F_0(x) = argmin \sum_{i=1}^{n} L(y_i, \gamma)$
2. Pour $m = 1, ..., M$ avec M le nombre maximal d'arbres. On calcule :
   (a)
   $$r_{im} = -\Big[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\Big]_{F(x)=F_{m-1}(x)}$$
   où r_{im} représente le residu de l'individu i pour le modèle à $m^{ième}$ arbre.
   (b) Ajustement d'un arbre aux valeurs des residus r_{im} puis détermine les ensembles R_{jm} (représente la feuille j du m^ième arbre) pour $j = 1, .., J_m$. Avec J_m le nombre de feuilles de l'arbre.
   (c) Pour $j = 1, .., J_m$ on calcule:
   $$\gamma_{jm} = \arg\min \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$
   (d) Calculer:
   $$F_m(x) = F_{m-1}(x) + \nu \sum_{i=1}^{J_m} \gamma_{jm} \mathbf{1}(x \in R_{jm})$$
   Où ν est appelé taux d'apprentissage (permet de réduire l'effet qu'à chaque arbre sur la prédiction finale, et améliore la précision).
3. Calculer $F_M(x)$

2.3 Variantes de Boosting

Plusieurs variantes de Boosting ont été mises sur pied notamment: Extreme Gradient Boosting, Light Gradient Boosting et le CatBoost avec des spécificités particulières.

L'Extreme Gradient Boosting proposé par Tianqi Chen and Carlos Guestrin est un assemblage de "weak learners" qui prédisent les résidus, et corrigent les erreurs des "weak learners" précédents. La particularité d'XGBoost réside dans le type de "weak learner" utilisé. Les arbres qui ne sont pas assez bons sont "élagués", c'est à dire qu'on leur coupe des branches, jusqu'à ce qu'ils soient suffisamment performant. Sinon ils sont complètement supprimés.Ainsi, XGBoost s'assure de ne conserver que de bons weak learners. De plus, XGBoost est informatiquement optimisé pour rendre les différents calculs nécessaires à l'application d'un Gradient Boosting rapides.

Light Gradient Boosting ne construit pas un arbre par niveau - ligne par ligne mais par les feuilles. Il met en œuvre un algorithme d'apprentissage d'arbre de décision basé sur un

histogramme hautement optimisé, qui présente de grands avantages en termes d'efficacité et de consommation de mémoire. L'algorithme LightGBM utilise deux nouvelles techniques appelées Gradient-Based One-Side Sampling (GOSS) et Exclusive Feature Bundling (EFB), qui permettent à l'algorithme de fonctionner plus rapidement tout en maintenant un haut niveau de précision et surtout tient compte des variables catégorielles.

Le CatBoost est une variante du Boosing ayant pour particularité la capacité de déterminer la façon optimale d'encoder les variables catégorielles en utilisant un target encodeur c'est-à dire que l'algorithme imput automatiquement la valeur de la target au niveau des classes.

3 Applications et Comparaison

Dans cette section nous allons implémenter quelques uns des algorithms présentés ci-dessus afin de comparer leur performance.

3.1 Presentation des données

Les données utilisées dans ce travail sont téléchargeables via ce lien https://web.stanford.edu/~hastie/CASI_files/DATA/SPAM.html. La base utilisée a été élaborée pour construire un filtre anti-spam personnalisé. Elle est constituée de 4601 messages électroniques dont 1813 comme spam, le reste étant du bon courriel (ham). L'ensemble de caractéristiques(57 features) suit 57 des mots non triviaux les plus couramment utilisés dans le corpus, en utilisant un modèle de sac de mots. La fréquence relative de chacun de ces mots et jetons est enregistrée pour chaque message électronique. Sont également inclus trois enregistrements différents de lettres majuscules.

3.2 Approche Analytique

La variable cible, celle que nous chercher à prédire est le message électronique(spam ou ham). Il est question d'élaborer et de mener une analyse comparative des algorithmes détaillés précédemment et basés sur les différentes caractéristiques du message.

Afin de trouver les paramètres optimaux de chaque algorithmes et d'éviter l'overfitting(modèle s'adapte un peu trop aux données d'entrainement et peine à se généraliser sur les données test) et l'underfitting(modèle s'adapte mal aux données donc difficile de capturer le pouvoir prédictif) une validation croisée est faite sur l'échantillon d'apprentissage.

4 Resultats

Différentes métriques sont utilisées pour mesurer la performance des modèles sur les données de test notamment: Accuracy, Recall and AUC.

4.1 Bagging des Arbres

Plusieurs algorithmes de Bagging sont implémentées sur l'échantillon train(80% des données) puis évaluer sur l'échantillon text(20% des données). Nous commençons par un arbre simple puis réalisons un voting Bagging et enfin un Random Forest.

Table 1: Performance des modèles

Modèles	**Accuracy(%)**	**Precision(%)**	**Recall(%)**
Simple Decision Tree	90.89	90.73	84.86
Bagging vote	92.01	94.58	84.86
Random Forest	95.37	96.10	92.15

On constate que le Random Forest surpasse tous les autres modèles sur les trois métriques avec une précision de 96.1%. Ce modèle sera dont retenu comme modèle Bagging pour la comparaison.

4.2 Boosting des Arbres

Les algorithmes tels que l'Adaboost, l'extrême gradient boosting(xgboost) et le light gradient boosting machine(lgbm) ont été ajustés sur les données d'entrainement puis évaluer sur les données test.

Table 2: Performance des modèles

Modèles	**Accuracy(%)**	**Precision(%)**	**Recall(%)**
Adaboost	94.62	95.13	91.21
Xgboost	95.52	95.24	93.46
lgbm			

Comme le montre le tableau 2, le Xgboost surpasse "à la fois l'Adaboost et le light Gradient Boosting Machine" avec un recall de 95.24% et un accuracy de 95.52%.

Pour les données utilisées dans cette analyse le **Random Forest** et **Extrême Gradient Boosting** sont les deux algorithmes qui surpassent tous les autres modèles respectivement dans le cadre du Bagging des arbres et du Boosting des arbres. Cependant, d'une part, en terme de recall et d'accuracy le xgboost est plus performant que le Random Forest et d'autre part le Random Forest surpasse le xgboost en terme de précision. En d'autre terme, le xgboost détecte mieux(93.46%) les spams et de manière générale classe bien les spams et les hams(95.52%) des messages électroniques que le Random Forest(95.37%).

References

[1] Hastie, Tibshirani et Friedman. **Elements of Statistical Learning Theory** http://statweb.stanford.edu/~tibs/ElemStatLearn/.

[2] Leo Breiman, **Random forests:** http://oz.berkeley.edu/~breiman/randomforest2001.pdf.

[3] Harvard IACS Github **2019-CS109A** https://harvard-iacs.github.io/2019-CS109A/

Annexe

A- Confusion Matrix et courbe roc

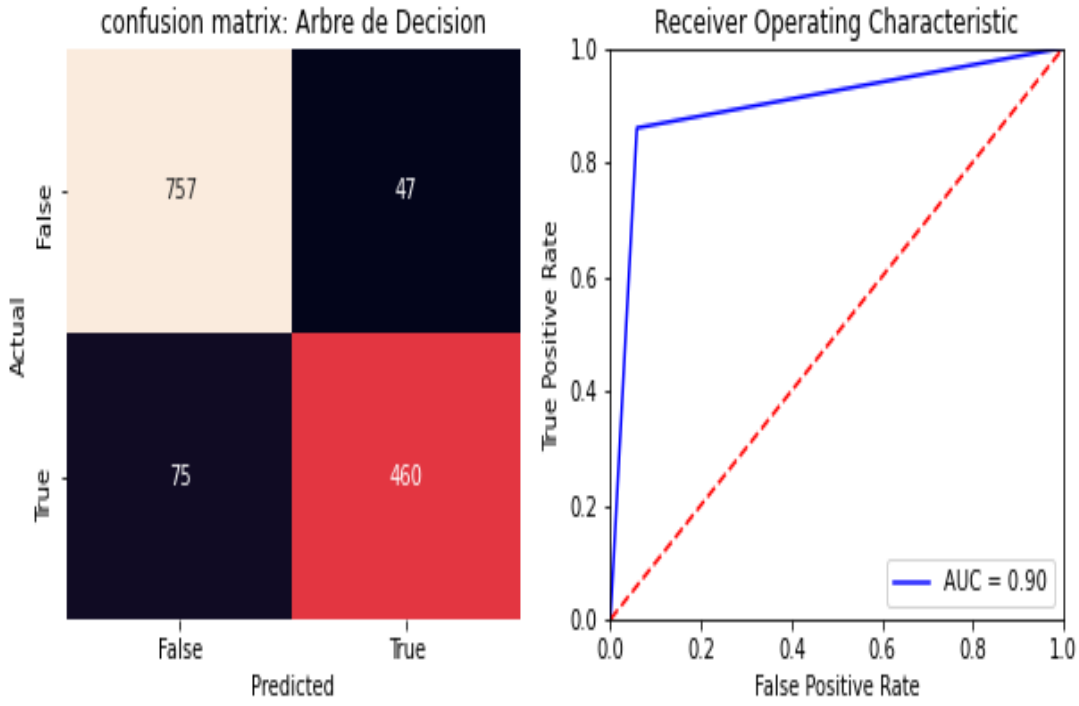

Figure 1: Matrice de confusion et courbe roc: Simple Decision Tree

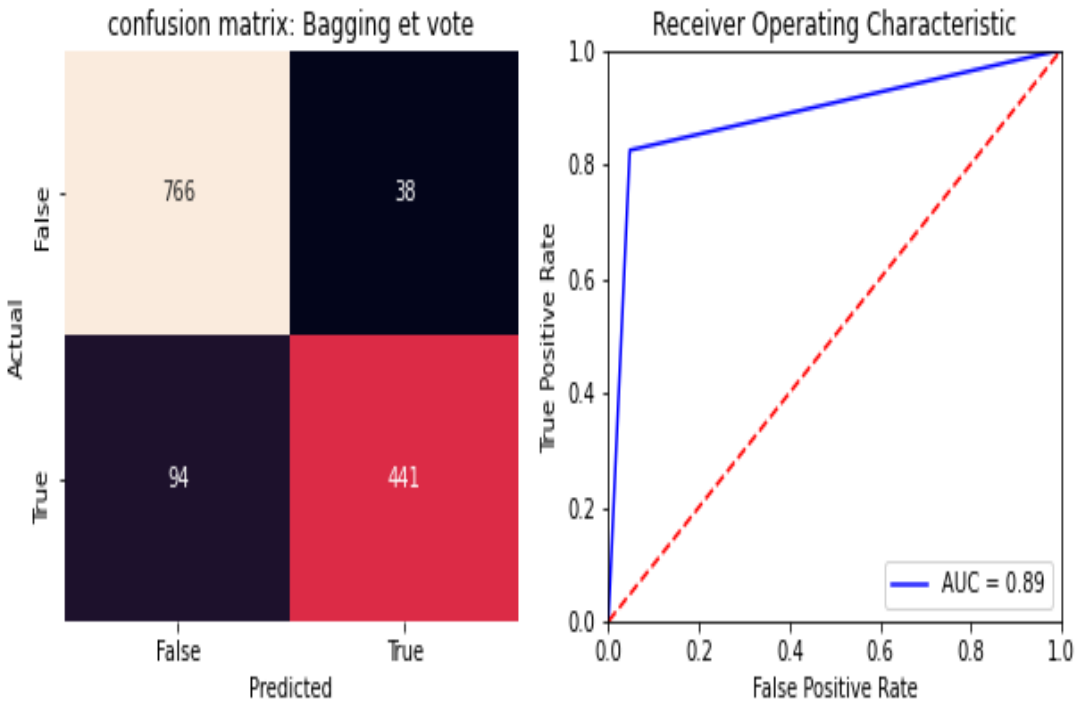

Figure 2: Matrice de confusion et courbe roc: Bagging Arbre

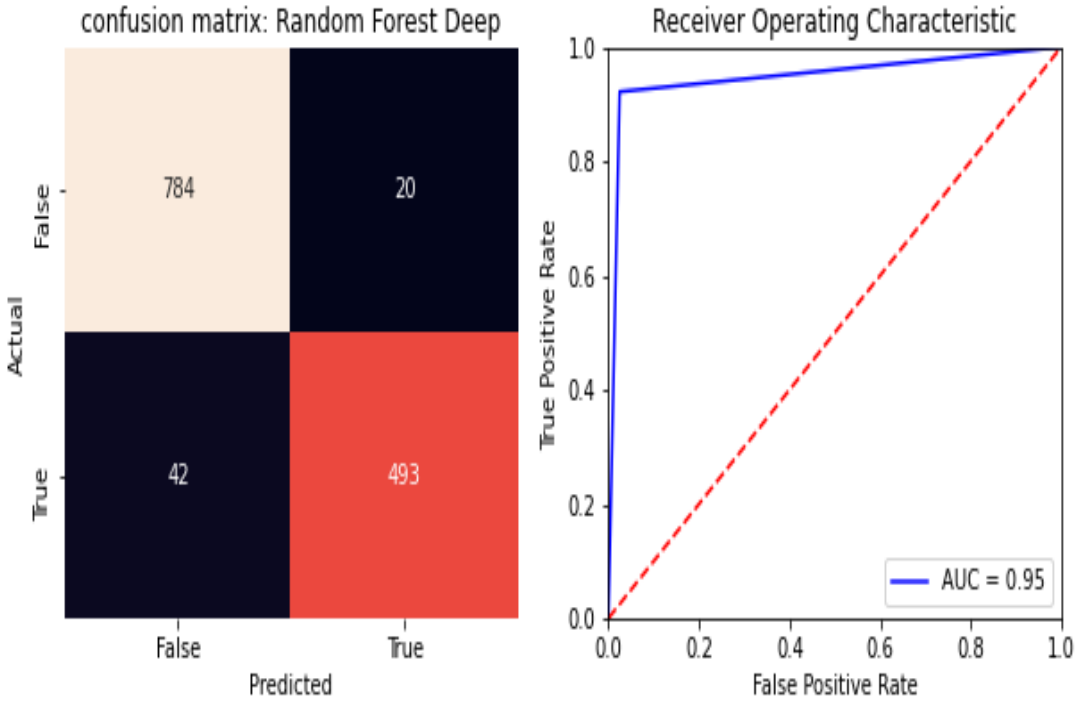

Figure 3: Matrice de confusion et courbe roc: Random Forest

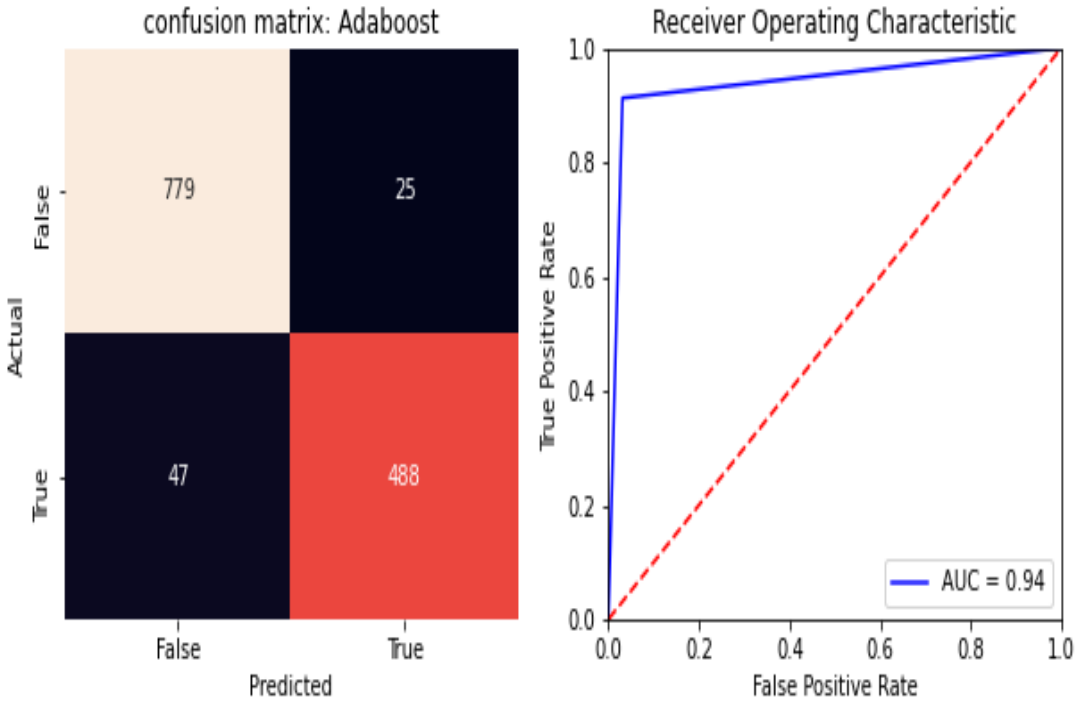

Figure 4: Matrice de confusion et courbe roc: AdaBoost

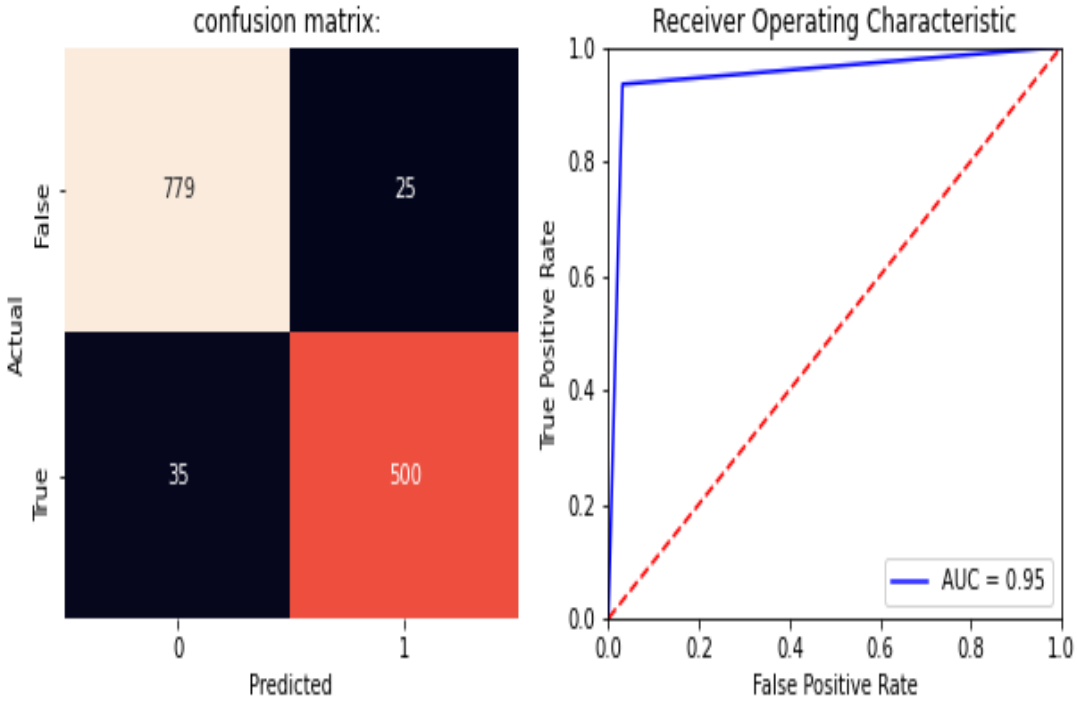

Figure 5: Matrice de confusion et courbe roc: XGboost