Forêts aléatoires et Boosting d'arbres de décision

Techniques avancées d'apprentissage

Lilia BEN BACCAR et Emmanuel KAMLA FOTSING

May 14, 2021

ENSAE
2020-2021

Sommaire

Forêt aléatoire

UNE APPROCHE BAGGING

Bagging Le bagging crée des sous-ensembles d'entrainement à l'aide d'échantillonnage bootstrap [Elfron et Tibshirani, 1993].

Pour créer un nouveau "base learner":

- ◇ On tire aléatoirement avec remise n observations de l'ensemble d'entrainement
- ◇ On entraine notre méthode (ex : CART) sur cet ensemble d'observations
  - chaque base learner contient un sous ensemble des observations de l'ensemble d'entrainement.
  - la performance d'un "base learner" est obtenu par l'erreur out-of-bag.

ALGORITHME

Algorithm 1: Forêts aléatoires

1. Pour $b = 1$ jusqu'à B:
   - 1.1 Tirer un échantillon bootstrap de taille N à partir du train.
   - 1.2 Construire un arbre 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.
     - 1.2.1 Selectionner aléatoirement m variables à partir des p variables.
     - 1.2.2 Choisir le meilleur point de séparation parmi les m.
     - 1.2.3 Séparer le noeud en deux noeuds filles.
2. Produire l'ensemble d'arbres $\{T_b\}_1^B$.

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* $\{\hat{C}_b(x)\}_1^B$

Définition mathématique

Forêt aléatoire

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

Boosting d’arbres de décision

BOOSTING

[Schapire, 1990] 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.

AdaBoost [Freund and Schapire, 1997]

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.
   - 2.1 Ajuster un classifieur $G_m(x)$ sur les données d'entraînement en utilisant les poids ω_i
   - 2.2 Calcul: $err_m = \frac{\sum_{i=1}^{N} \mathbb{I}(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \omega_i}$
   - 2.3 calcul: $\alpha_m = \log((1 - err_m)/err_m)$
   - 2.4 mise à jour $\omega_i = \omega_i \exp\left[\alpha_m \mathbb{I}(y_i \neq G_m(x_i)\right]$
3. Output $G(x) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(x_i)\right]$

GRADIENT BOOSTING [FRIEDMAN, 2001]

Algorithm 3: Gradient Boosting

1. Initialisation du modèle: $F_0(x) = argmin \sum_{i=1}^{n} L(y_i, \gamma)$
2. Pour $m = 1, ..., M$ avec M le nombre maximal d'arbres.
   - 2.1 Résidu de l'individu i pour le modèle à $m^{ième}$ arbre: $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$
   - 2.2 Ajuster un arbre aux valeurs des residus r_{im} puis déterminer les ensembles R_{jm} pour $j = 1, .., J_m$.
   - 2.3 Pour $j = 1, .., J_m$:

$$\gamma_{jm} = \arg\min \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

   - 2.4 $F_m(x) = F_{m-1}(x) + \nu \sum_{i=1}^{J_m} \gamma_{jm} \mathbf{1}(x \in R_{jm})$
3. Calculer $F_M(x)$

Autres variantes

Extreme Gradient Boosting Assemblage de "weak learners" qui prédisent les résidus, et corrigent les erreurs des "weak learners" précédents. XGBoost s'assure de ne conserver que de bons weak learners en en élagant ou supprimant certains.

Light Gradient Boosting Construit un arbre par les feuilles basé sur un histogramme hautement optimisé. Deux nouvelles techniques : Gradient-Based One-Side Sampling (GOSS) et Exclusive Feature Bundling (EFB).

CatBoost Capacité de déterminer la façon optimale d'encoder les variables catégorielles.

Expérimentation

Données

- Classification binaire : spam / non-spam
- 57 variables.

	spam	testid	make	address	all	3d	our	over	remove	internet	order	mail
0	True	True	0.00	0.64	0.64	0.0	0.32	0.00	0.00	0.00	0.00	0.00
1	True	False	0.21	0.28	0.50	0.0	0.14	0.28	0.21	0.07	0.00	0.94
2	True	True	0.06	0.00	0.71	0.0	1.23	0.19	0.19	0.12	0.64	0.25
3	True	False	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63
4	True	False	0.00	0.00	0.00	0.0	0.63	0.00	0.31	0.63	0.31	0.63

Figure 1: Echantillon données

Données

- 4601 messages électroniques ($\sim$ 40% spam)

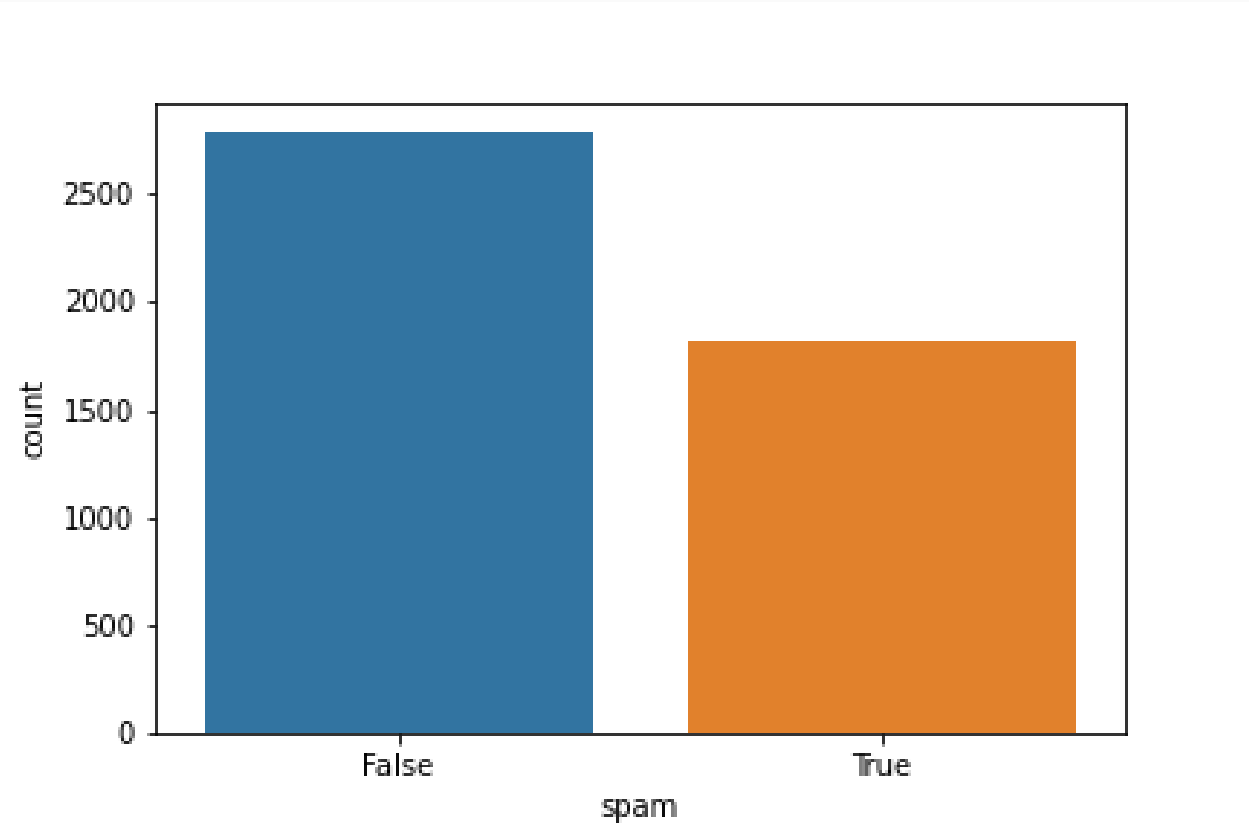

Figure 2: Répartition spam / non-spam

Approche

Analyse comparative des algorithmes présentés

- Split du dataset : train (70%), validation (10%), test (20%)
- Cross-validation : paramètres optimaux (nombre d'arbres, sous-échantillons à utiliser,...)
- Implémentation de l'algorithme avec les paramètres optimaux
- Mesure de la performance (accuracy, précision, recall,...)
- Comparaison

Résultats

RÉSULTATS

Bagging

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

RÉSULTATS

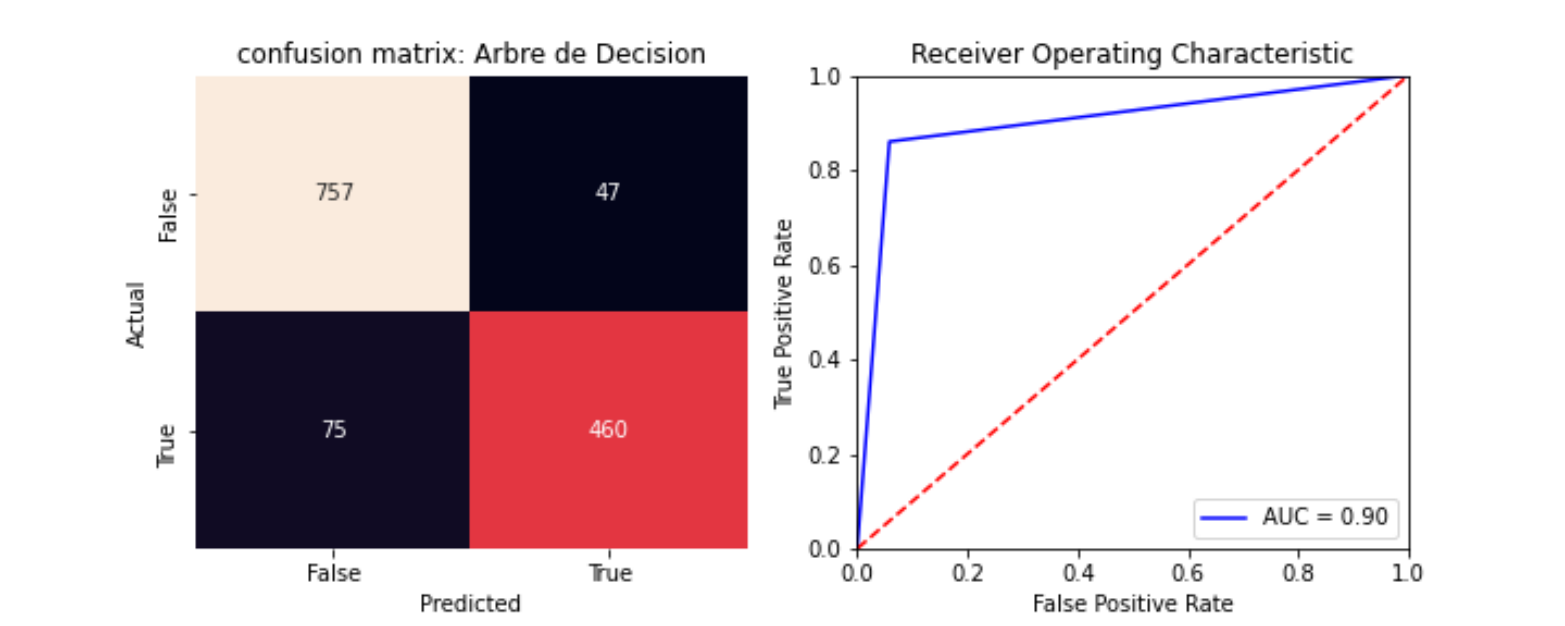

Figure 3: Matrice de confusion et courbe roc: Simple Decision Tree

RÉSULTATS

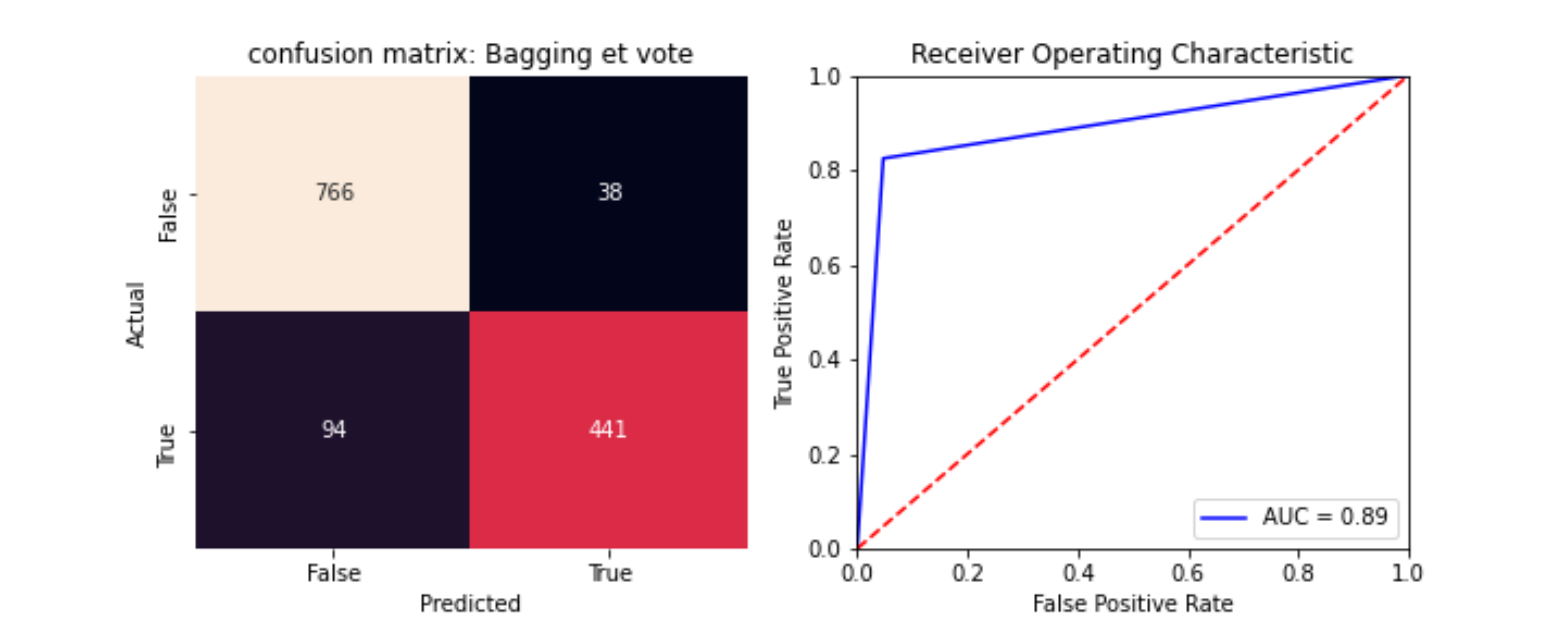

Figure 4: Matrice de confusion et courbe roc: Bagging vote

RÉSULTATS

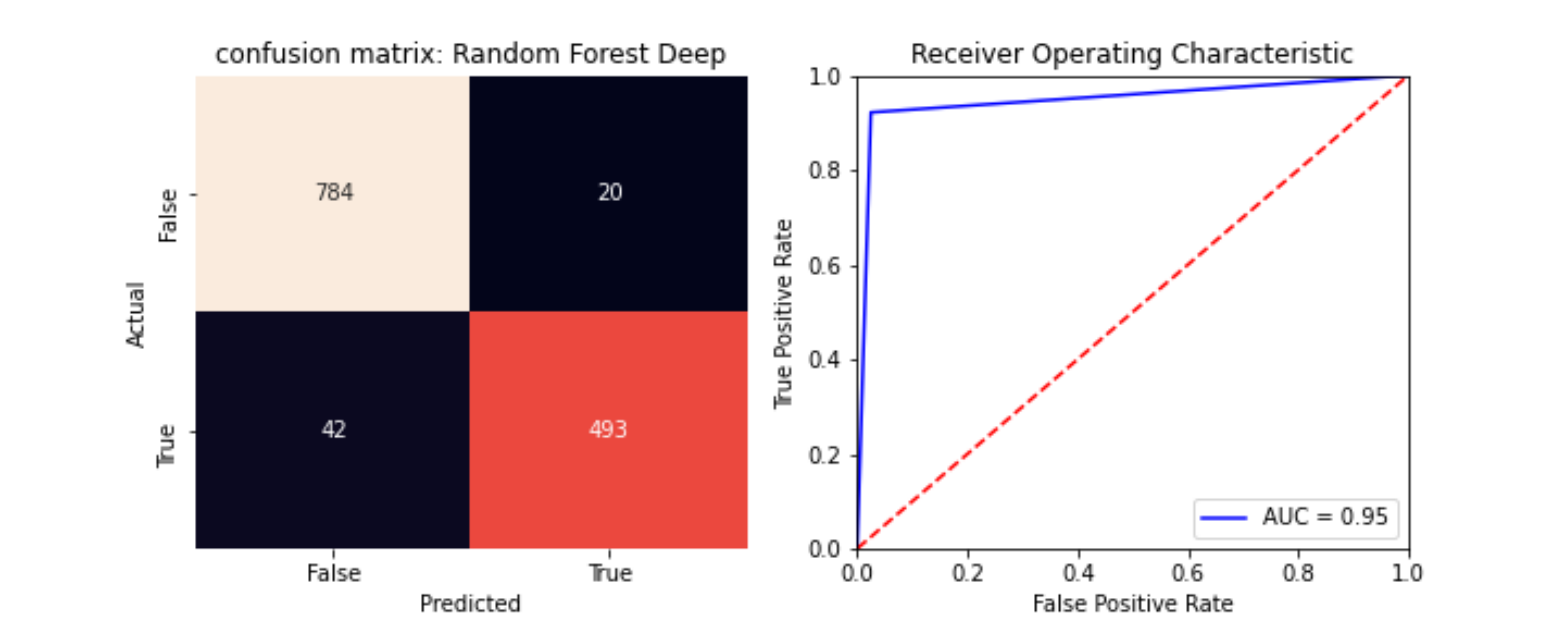

Figure 5: Matrice de confusion et courbe roc: Random Forest

RÉSULTATS

Boosting

Table 2: Performance des modèles

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

RÉSULTATS

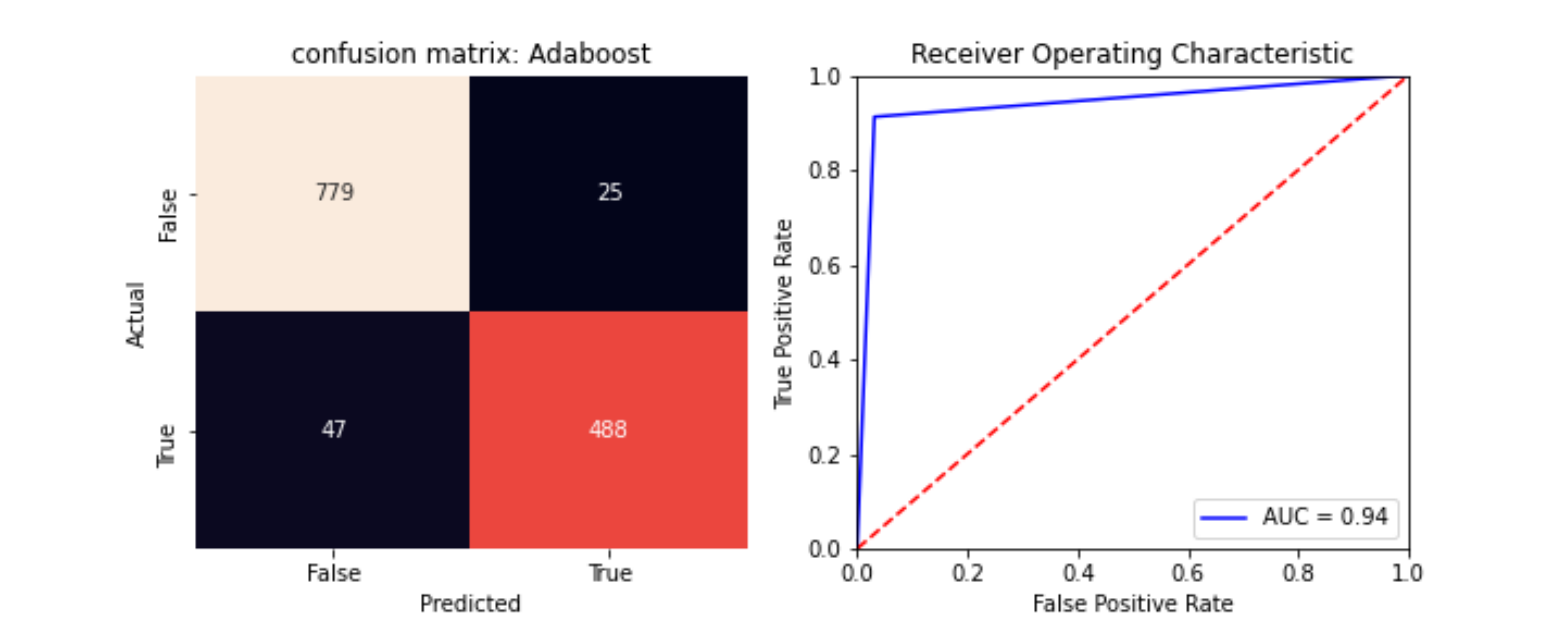

Figure 6: Matrice de confusion et courbe roc: AdaBoost

RÉSULTATS

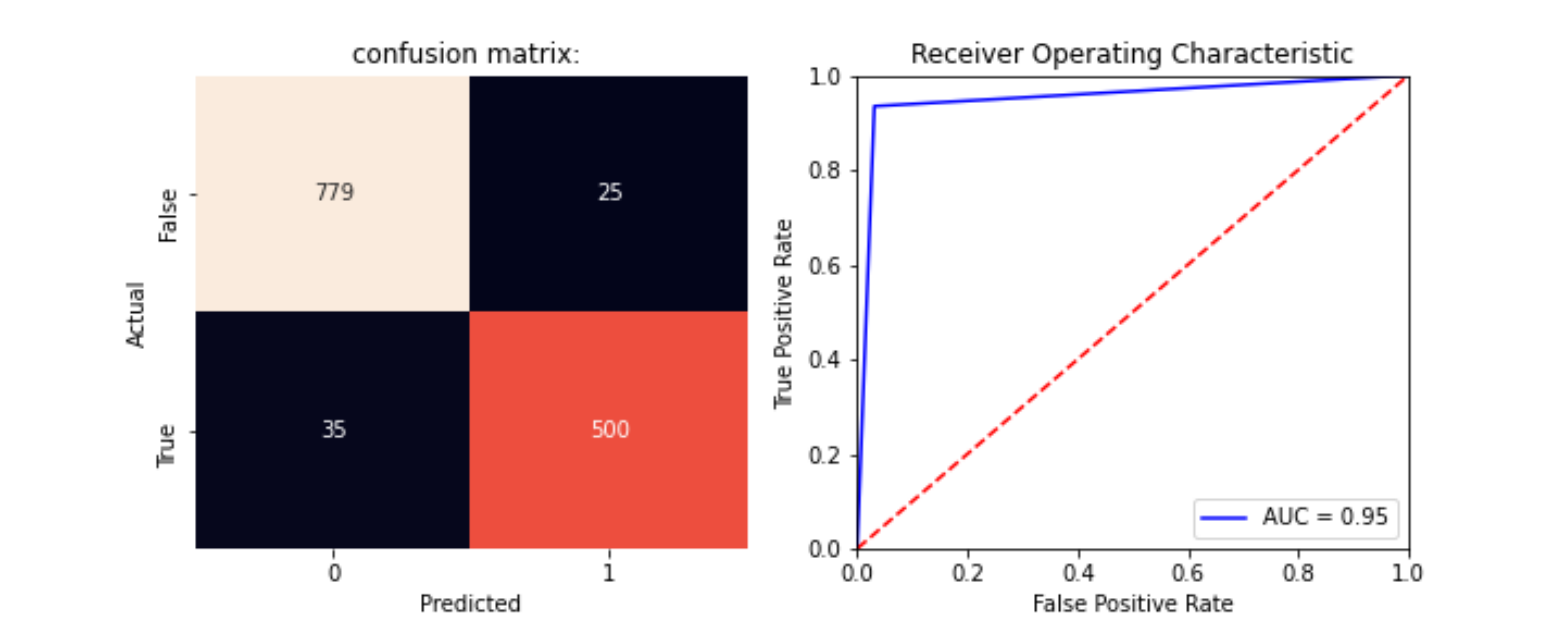

Figure 7: Matrice de confusion et courbe roc: XGboost

Conclusion

- Le Random Forest et le XGBoost surpassent les autres modèles.
- En terme de rappel et accuracy, le XGBoost surpasse le Random Forest.
- En terme de précision, le Random Forest surpasse le XGBoost.

XGBoost détecte mieux **(93.46%)** les spams et de manière générale classe bien les spams et les non-spam **(95.52%)** des messages électroniques que le Random Forest **(95.37%)**.

Références

Références

- Hastie, Tibshirani et Friedman. **Elements of Statistical Learning Theory** `http://statweb.stanford.edu/~tibs/ElemStatLearn/`.
- Leo Breiman, **Random forests:** `http://oz.berkeley.edu/~breiman/randomforest2001.pdf`.
- Harvard IACS Github **2019-CS109A** `https://harvard-iacs.github.io/2019-CS109A/`.