Travail présenté à
Mohamed Bouguessa

TRAVAIL PRATIQUE 1

Analyser et critiquer la performance de cinq algorithmes de classification

Distinguer les utilisateurs réguliers des utilisateurs pollueurs de Tweeter en 2006

Par

Mohamad Hawchar

HAWM20039905

et

Émile Pronovost

PROE18049606

Le 07 mars 2022

Université du Québec à Montréal

INF7370 - Apprentissage automatique

Table des matières

1. Introduction

*** À noter que les tables et figures du projet sont générées par le code et bien nommées dans leur dossier respectif sous le format « .png ». À cet effet, ces dossiers feront office d'annexes ***

1.1 Mise en contexte

Le mandat de ce présent travail consiste à faire une analyse comparative entre cinq méthodes de classification supervisées. Ces méthodes seront implémentées afin de distinguer des utilisateurs réguliers des utilisateurs pollueurs de Twitter selon des données de 2006. Il sera alors question de calculer des attributs qui ont à priori une influence sur le type d'utilisateur, appliquer les méthodes de classification et évaluer ces méthodes à l'aide de deux tests. Le test du Chi-2 et le gain d'information.

1.2 Résumé de la méthodologie

Premièrement, nous avons attribué chaque fichier d'entrée à une variable du même nom. Chaque variable représente un tableau (*dataframe)* pour lesquels le nom des colonnes concorde avec le jeu de données « social_honeypot_icwsm_2011 ».

Deuxièmement, on a calculé 15 attributs (*features*) qui peuvent influencer la catégorisation d'un utilisateur pollueur ou non. Une fonction a été créée pour chaque attributs nécessitant d'être calculés. Les fonctions prennent un *dataframe* en entrée et en retourne un autre convenablement calculé. L'ensemble des *dataframe* résultants ont été fusionnés selon le type d'utilisateur, en leur attribuant une classe binaire indiquant s'ils sont pollueurs (1) ou non (0).

Troisièmement, un prétraitement des données a permis de traiter les valeurs manquantes en les remplaçant par la valeur « 0 ». C'est aussi à cette étape qu'on a normalisé les données en fonction de l'écart-type et de la moyenne des valeurs de chaque attribut à l'aide de la cote Z (Z-Score).

Quatrièmement, on a généré les cinq classifications sur l'ensemble des attributs et on a comparé la qualité de chaque classification à l'aide de matrices de confusion et de courbes ROC. Nous avons fait deux tests; celui du gain d'information et celui du Chi-2. Les sept attributs ayant les résultats les plus forts ont été sélectionnés pour chacun de ces tests et on a comparé de nouveau les méthodes classifications en fonction de ces-derniers. Il s'est avéré que la méthode de classification la plus optimale est celle des forêts aléatoires (*Random Forests)*.

2. Méthodologie

2.1 Lecture des fichiers en dataframe et association des noms de colonne

Les données utilisées ont été collectées sur une période temporelle du 30 décembre 2009 au 02 août 2010.

Les six fichiers représentent trois données distinctes par type d'utilisateur (pollueur ou non-pollueur). Le premier jeu de données regroupe **l'information de base** pour chaque profil d'utilisateur tels que le nombre de followers ou la date de création du compte Twitter. Les deux fichiers sont « content_polluters.txt » pour les pollueurs et « legitimate_users.txt » pour les utilisateurs légitimes. Le second type de donnée est le **nombre de *followings*** pour chaque utilisateur. Chaque ligne est constituée du numéro unique de l'utilisateur (UserID) et d'une liste de nombres séparés par des virgules. Cette liste correspond à l'évolution du nombre de *followings* dans le temps. Le troisième correspond à **l'ensemble des *tweets*** récoltés durant la période de collecte.

Le jeu de données utilisé est le dossier « social_honeypot_icwsm_2011 ». Il comporte les six fichiers textes nécessaires pour l'analyse. Sachant que le nombre de fichiers est petit et qu'ils n'ont que quelques colonnes, on a décidé d'entrer le des noms de colonnes en tant que liste dans le script. On a alors nommé une variable pour chaque fichier, et appliqué la fonction Pandas *read_csv()*. Elle a permis d'associer à chaque variable un dataframe avec le bon nom des champs. On a du changé le type de données de certaines colonnes. À cet effet, on a assigné le type *date* pour les attributs ayant des dates comme valeur à l'aide du paramètre *parse_dates* de la fonction *read_csv()*. On a ensuite séparé les valeurs de la colonne *SeriesOfNumberOfFollowings* en fonction des virgules pour en créer une série (format *Series* dans Pandas) de nombres entiers.

2.2 Calcul des attributs

Pour chaque attribut, au moins une fonction a été créée. Chaque fonction prend en paramètre un *dataframe* et en retourne un nouveau qui comprend l'attribut calculé. Le nom de chaque fonction

indique en anglais le *dataframe* qu'elle retourne. Par exemple, le premier attribut(feature) calculé est la longueur du nom de l'usager. La fonction porte alors le nom *lengthName()*. À noter que toutes les fonctions utilisent la fonction DataFrame() de Pandas qui permet de créer un dataframe à partir d'un dictionnaire. On est alors en mesure de spécifier le nom de l'attribut par la clé du dictionnaire.

Plus de détails techniques sont expliqués par des notes dans le code lui-même, mais l'idée générale est de pouvoir concaténer les 15 *features* adéquatement.

1. La longueur du nom de l'usager (*the length of the screen name*)
2. La longueur de la description du profil de l'usager (*the length of description)*

Les deux premières fonctions sont uniquement la sélection des colonnes qui correspondent aux longueurs respectives du nom d'usager et de la description du profil de l'usager. Puisque les fichiers textes ont déjà une colonne du même nom, aucun calcul n'est nécessaire. Il suffit de sélectionner les bonnes colonnes.

3. La durée de vie du compte (the longevity of the account)

On estime que la durée de vie du compte est la différence entre la date de du compte et la date de la collecte des données, en jours. Ces dates sont disponibles dans les deux champs respectifs suivants : *CreatedAt* et *CollectedAt*. Puisque le type de donnée est déjà un objet de date, il suffit de faire la soustraction entre ces colonnes. Cependant, il faut préciser l'unité de calcul et le type de donnée dans le tableau. Nous décidons de calculer la durée de vie en jours et que ce soient des nombres réels. La fonction *timedelta64()* de Numpy permet de le faire. On a donc divisé la soustraction des deux dates par 1 jour qui nous donne le nombre de jours en nombres réels.

4. Le nombre *de following*
5. Le nombre de *followers*

Pour le calcul des quatrième et cinquièmes attributs, se référer aux fonctions 1 et 2.

6. L'écart type des IDs numériques uniques des *following* (*the standard deviation of unique numerical IDs of following*)

Pour calculer l'écart-type, il a suffi d'implémenter la fonction de l'écart-type sur une colonne.

7. Le rapport *following / followers*
8. Le nombre total des tweets envoyés

Se référer aux fonctions 1 et 2.

9. Le nombre de tweets envoyé par jour
10. Le rapport nombre de tweets par rapport à la durée de vie du compte : |Tweets| / durée de vie du compte

11. Le rapport des adresses URL par rapport au nombre de tweets : |TweetURL| / |Tweets|
12. Le nombre moyen d'URL par tweet
13. Le rapport des mentions @ par rapport au nombre de tweets : |@username| in tweets / |Tweets|

Pour le calcul des attributs 11, 12 et 13 on a utilisé la librairie des expressions régulières *Regular Epressions (re)*. Ces expressions ont permis d'aller isoler du texte pour chaque ligne du dataframe correspondant à l'information voulue selon l'attribut. Les fonctions traitent la colonne *Tweet* du *dataframe* des *Tweets.*

Deux fonctions ont été créées pour chaque attribut : Une première fonction qui extrait l'information de chaque tweet et une seconde qui inclue la première et retourne le *dataframe* de l'attribut calculé.

L'attribut 11 a comme première fonction de valider si la colonne *Tweet* contient des URL ou non. Elle retourne un 0 ou un 1. La seconde fonction

L'attribut 12 a comme première fonction de compter combien chaque *Tweet* contient des URL. La deuxième fonction va retourner la moyenne d'URL par Tweets utilisant la première fonction.

L'attribut 13 a comme première fonction de compter les mentions de « @ » et la seconde viens retourner la moyenne des mentions par nombre de Tweet.

14. Le temps moyen (en minutes) entre deux tweets consécutifs
15. La valeur de temps maximal (en minutes) entre deux tweets consécutifs

Les attributs 14 et 15 utilisent la même logique que les trois précédents, mais elle n'utilise pas d'expressions régulières. C'est-à-dire qu'une première fonction est créée pour le calcul de l'attribut et que la seconde fonction utilise la première pour retourner le nouveau *dataframe*.

2.3 Prétraitement

Dans cette section on explique notre démarche au retrait des doublons, au traitement des valeurs manquantes et à la normalisation des données. Il faut savoir que pour pouvoir faire cette étape, on a préalablement fusionné les 15 dataframes d'attributs pour chaque type d'utilisateur, associé la classe binaire pollueur et fusionné les dataframe « pollueurs » et « utilisateurs réguliers ».

2.3.1 Concaténation des dataframes et ajout de la classe binaire (pollueur ou non)

On a concaténé chaque dataframes en sortie des calculs de features, pour en créer deux dataframes avec les bons attributs :

2.3.2 Suppression des instances de données en double.

Pour le retrait des doublons, on a utilisé la méthode *drop_duplicates()* intégrée à la librairie Pandas. Elle prend un dataframe en entrée, en retire toutes les lignes en doubles et retourne un nouveau dataframe corrigé.

2.3.3 Remplacement des valeurs manquantes

Selon la documentation du cours de ce présent travail, voici les méthodes disponibles pour traiter les valeurs manquantes :

- **Ignorer le tuple**. (Cette méthode est pertinente seulement si la part des valeurs manquantes est non-significative pour la plupart des attributs)
- Utiliser une **constante globale**. (Pertinent lorsque les lignes comportant des valeurs manquantes indiquent une certaine homogénéité inter-attribut ou que les valeurs nulles se regroupent sur une même ligne)
- Utiliser la moyenne ou la médiane de l'attribut.
- Utiliser la valeur la plus probable à l'aide de la formule bayésienne ou d'un arbre de décision.

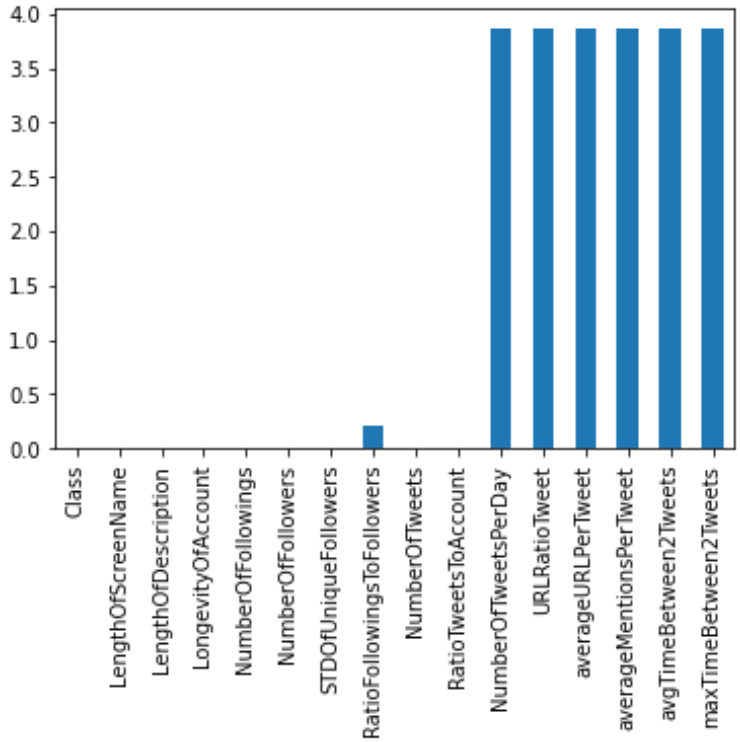
Le choix du traitement des valeurs manquantes s'est fait en deux principales étapes. D'abord, on a vérifié pour chaque *feature* la part que prennent les valeurs manquantes sur l'ensemble des valeurs. Ensuite, on a analysé la tendance globale d'une ligne comportant de telles valeurs. C'est-à-dire de

vérifier quelle semble être la valeur des *features* x2, x3, … x15 lorsque la valeur du *feature* x1 sont manquantes et ainsi de suite.

Avant le calcul de la proportion des valeurs manquantes, nous avons d'abord effectué une uniformité de celles-ci. À cet effet, les valeurs .inf, et -inf ont été remplacées par NaN. Cela permet que les valeurs manquantes soient tous attribuées plus rapidement ce type de valeur.

Le diagramme suivant illustre les proportions des valeurs manquantes pour chaque colonne :

Diagramme 1. Proportion des valeurs manquantes par attribut



On constate que des valeurs manquantes ne sont présentes que pour six des 15 *features*. La proportion n'est cependant pas significative étant d'environ 3,8%. Il est important de souligner que ces six attributs sont tous reliés aux *Tweets*. Le fait qu'ils comportent des valeurs manquantes s'explique par leur méthode de calcul, qui utilise soit des ratios, soit toute division telle que la moyenne. À cet effet, lorsqu'on divise une quelconque valeur par 0, la fonction retourne la valeur *inf* et lorsque qu'on divise un 0 par 0, la valeur retournée est *NaN*. De plus, après une visualisation de chaque ligne des valeurs manquantes, on a constaté que si une valeur est manquante, les mêmes six attributs en avaient à leur tour. Cela explique pourquoi les six attributs ont tous la même proportion.

Avec les constats expliqués ci-haut, on a décidé de remplacer les valeurs manquantes par la valeur 0.

2.3.4 Normalisation des données (approche z-score)

La fonction *zscore* est déjà implémentée dans Pandas et il suffit de l'appliquer sur un *dataframe* pour que toutes les colonnes soient normalisées. Dans notre cas, on veut que ce soit appliqué pour tous les attributs sans compter la classe. On a alors créé une fonction afin de ne pas appliquer le *zscore* sur la colonne des classes en la retirant et en la réinsérant après le processus.

2.3.5 Hypothèse et constats en fonction de la distribution

Une fois normalisée, les données sont maintenant prêtes pour l'implémentation des algorithmes de classification. Nous pouvons cependant avoir une bonne idée initiale sur les attributs qui se démarquent des autres pour la différenciation des classes. Cela peut être visualisé avec les différences de distribution des pollueurs (classe 1) et des non-pollueurs (classe 0). À cet effet, la page 11 comprend deux séries d'histogrammes pour chaque attribut. La première série comprend les histogrammes des utilisateurs réguliers et la seconde, celle des pollueurs. On peut voir que trois histogrammes montrent des différences sur la distribution des attributs interclasses : *LengthOfDescription :* La distribution des utilisateurs pollueurs est coupée pour des longueurs de description plus courtes contrairement aux utilisateurs réguliers qui on des longueurs de descriptions plus variées, plus « naturelles ».

URLRatioTweet : On constate très bien que la proportion des URL dans les tweets est plus élevée pour les pollueurs.

LongevityOfAccount : La longévité des comptes des pollueurs a tendance à être plus courte et l'histogramme nous le montre.

Figure 1. Séries d'histogrammes des attributs des utilisateurs réguliers

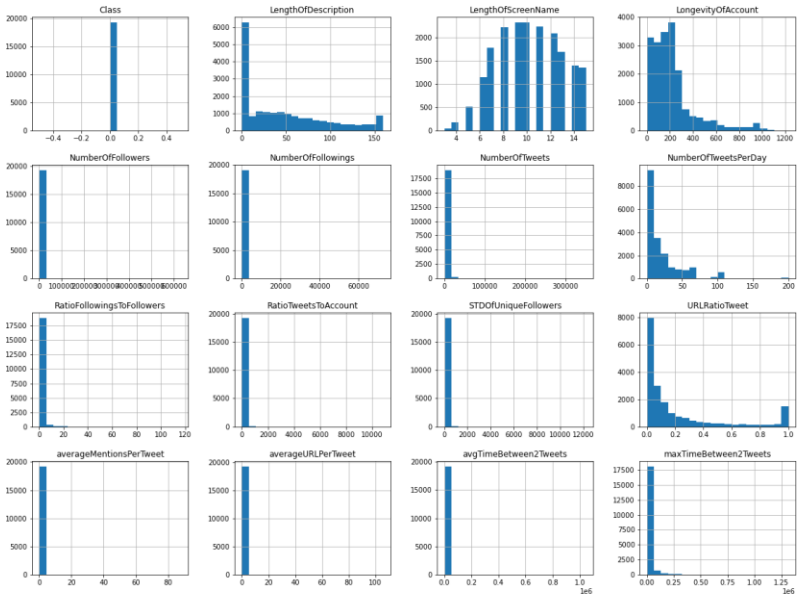
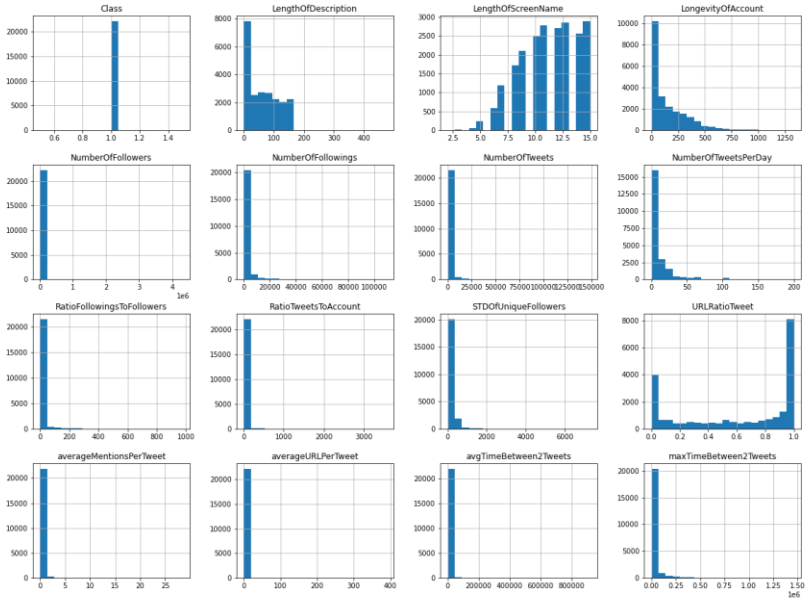


Figure 2. Séries d'histogrammes des attributs des utilisateurs pollueurs



Voici les diagrammes en boîte (*BoxPlot*) de quelques attributs se démarquant ainsi qu'une brève interprétation de ces derniers:

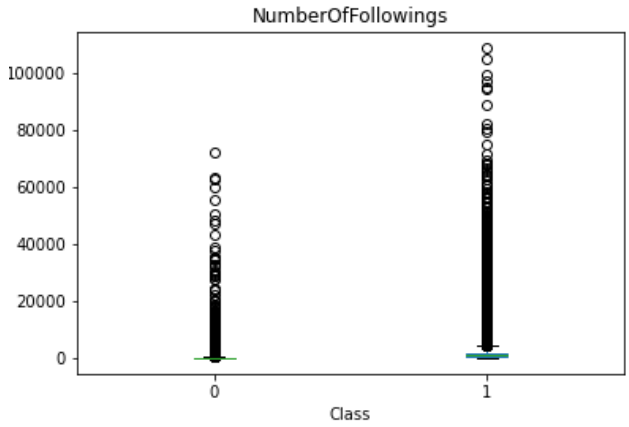


Diagramme en boîte 1. Distribution des utilisateurs selon leur nombre de Followings des classes. On remarque que les pollueurs tendent à avoir plus de « valeurs aberrantes » tant qu'au nombre de followings et semblent avoir une plus grande étendue de distribution. On peut alors penser qu'ils ont tendances à suivre un plus grand échantillon de comptes Twitter que les utilisateurs réguliers, et de façon « anormale ».

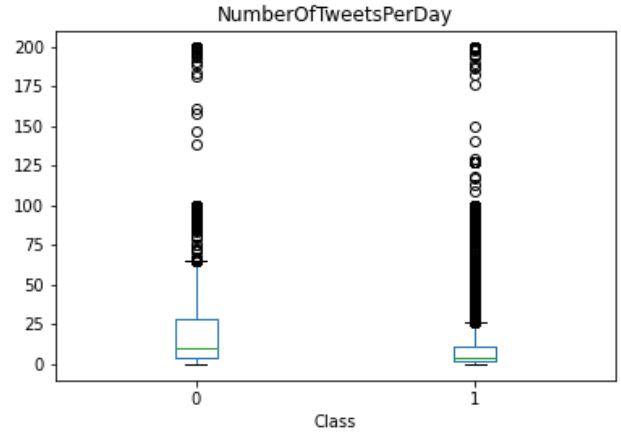


Diagramme en boîte 2. Distribution des utilisateurs selon leur ratio de *Tweets par jour*.

On remarque deux éléments pertinents. D'abord l'étendue des utilisateurs est beaucoup plus grande, ensuite

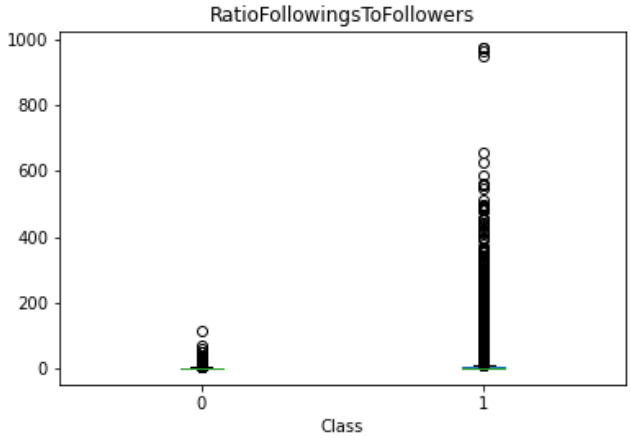


Diagramme en boîte 3. Distribution des utilisateurs selon leur ratio de *followings* / *followers*.

On voit très bien que les valeurs aberrantes des pollueurs sont plus fréquentes et extrêmes que les utilisateurs réguliers. Certains pollueurs ont alors un ratio anormal, c'est-à-dire qu'ils suivent plus de pages qu'ils sont populaires.

3 Entraînement des modèles et analyses comparatives

Cette étape vise en premier temps à comparer les cinq méthodes de classification appliquées sur l'ensemble des attributs. En second lieu, cette comparaison est répétée sur les sept attributs les plus représentatifs selon deux tests distincts : : A. Le gain d'information et B. le test du Chi-2.

Pour chaque méthode de classification de chaque analyse comparative, on a créé une matrice de confusion et une courbe ROC.

La matrice de confusion évalue la qualité de la classification en nous indiquant à quelle proportion l'algorithme a classifié correctement le jeu de données. Elle utilise un modèle de prédiction des classes et compare ces prédictions avec les valeurs initiales. Puisque la classification est binaire, la grille est composée de quatre cases, soient deux pour les classes prédites et deux pour les classes initiales. Voici l'interprétation de ces cases :

True Positive : La classe prédite est de la bonne classe que le jeu de données initial. Dans le cas ci-présent les pollueurs dans le jeu de données sont prédits pollueurs dans le training dataset.

True Negative : Les vrais négatifs sont les utilisateurs réguliers qui ont bien été prédit non-pollueur.

False Positive : Les faux positifs sont les utilisateurs réguliers qui ont été prédit pollueurs. Il y a alors une surreprésentation de la classe pollueurs dépendamment d'un taux de false positives élevé.

False Negative : Les faux négatifs sont les utilisateurs pollueurs qui ont été prédit comme utilisateur réguliers. C'est le type d'erreur à éviter en comparaison avec les *false positives* puisque le modèle oublie alors certains pollueurs.

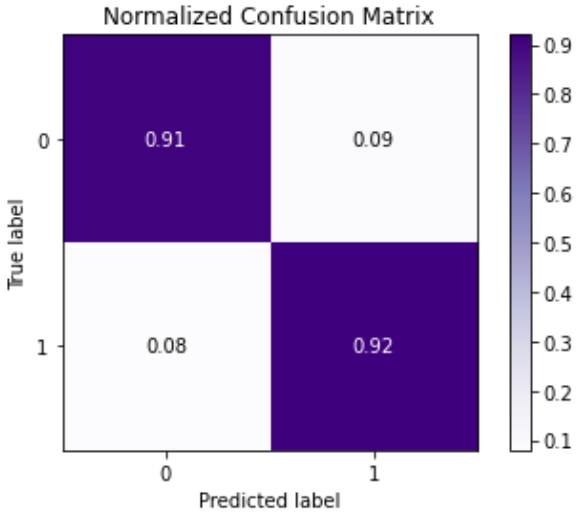
Une fois que l'on comprend la signification de chaque éléments constituants les matrices de confusion, il est facile de décrire, d'interpréter et de comparer chaque matrice.

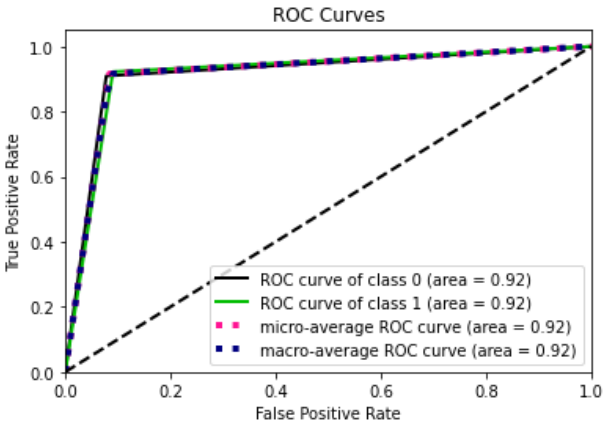
Chaque courbe ROC montre le compromis entre la sensibilité, qui correspond au True Positives Ratio (TPR) et la spécificité qui prend en compte le false positive ratio (1-FPR). Plus la courbe se rapproche du coin supérieur gauche, meilleure est la performance de la classification. Se rajoute à la courbe d'autres lignes qui agissent comme repère visuel. Par exemple, la droite correspond à une classification aléatoire qui devrait donner techniquement des points situés le long de la diagonale. Le cas échéant, cela voudrait dire que chaque valeur FPR est = à la valeur TPR. On veut que la courbe ROC se tienne le plus loin possible de cette diagonale vers le coin supérieur gauche, car plus la courbe se rapproche de la diagonale, moins le test est précis.

Une brève analyse est présente à la fin de chaque tâche en comparant les cinq méthodes classifications entre elles.

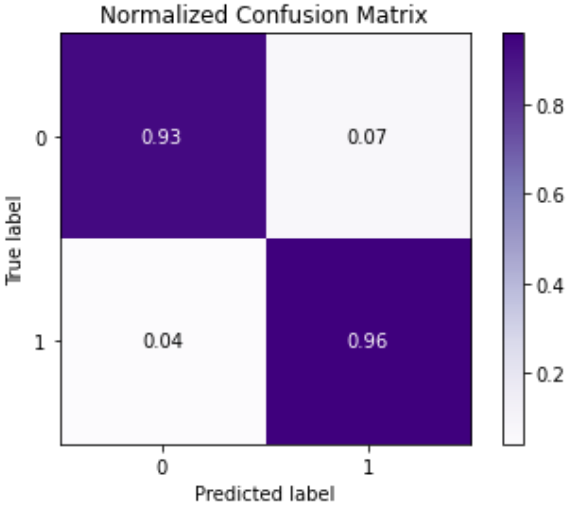
Tâche 1. Analyse comparative des méthodes de classification (incluant les 15 attributs)

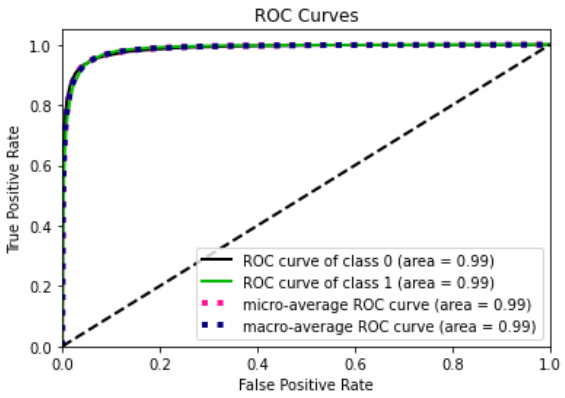
3.1.1 Arbre de décision



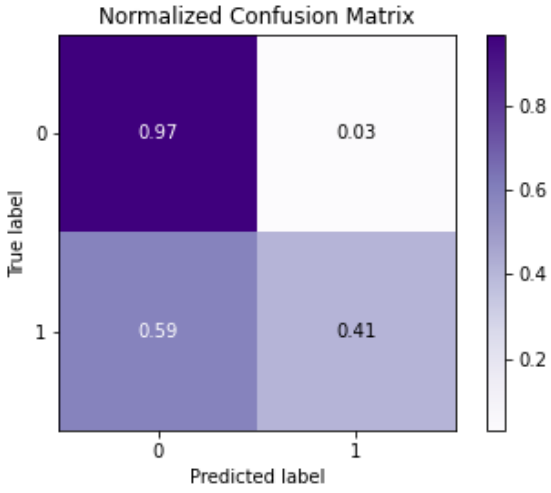


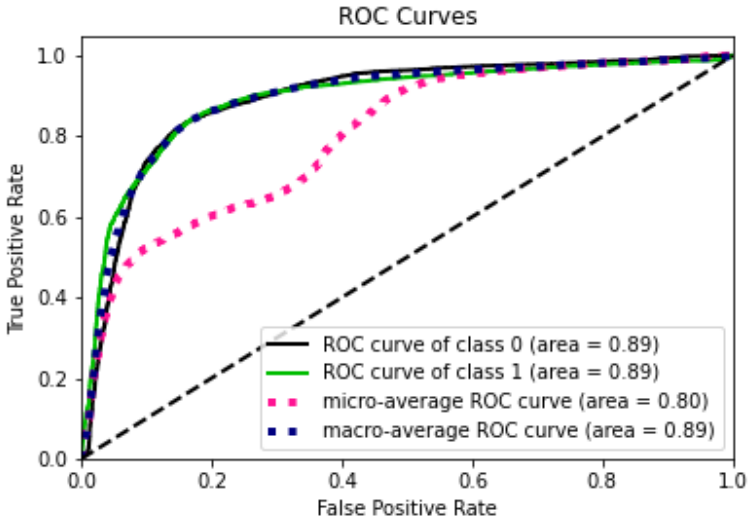
3.1.2 Forêt d'arbres décisionnels (Random Forest)



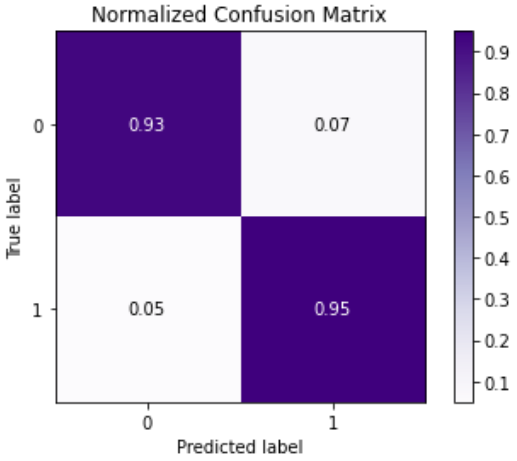


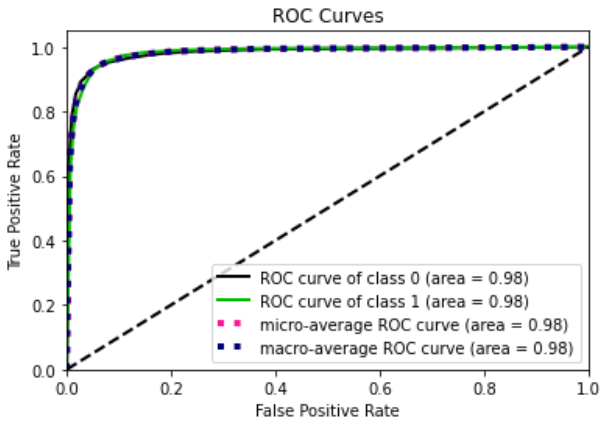
3.1.3 Classification bayésienne naïve



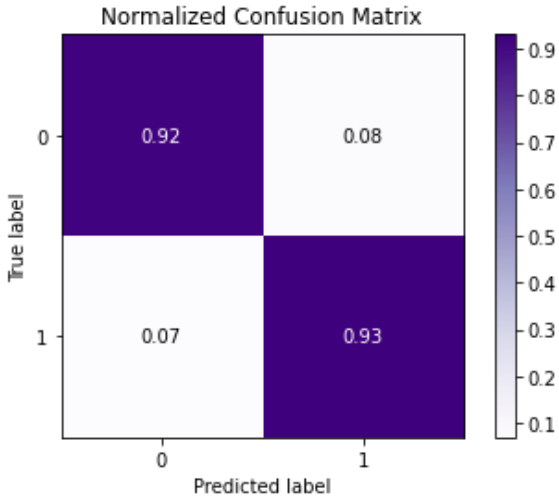


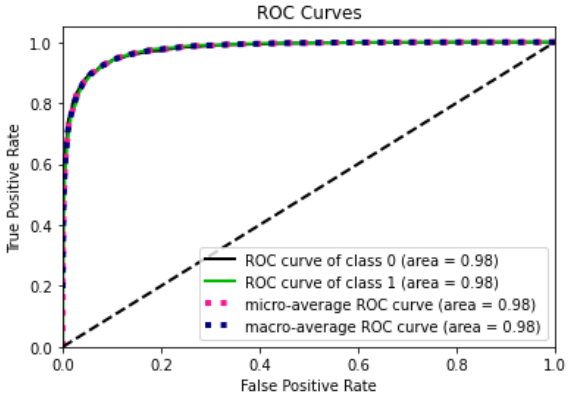
3.1.4 Bagging





3.1.5 AdaBoost





3.1.6 Analyse comparative de la tâche 1.

L'ensemble des tableaux suivants montrent trois indices sur la qualité de la classification.

Tableau 1. Analyse du rappel (*recall*) pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.91	0.93	0.97	0.93	0.92
Pollueurs	0.92	0.96	0.41	0.95	0.93
macro avg	0.92	0.94	0.69	0.94	0.92
weighted avg	0.92	0.95	0.68	0.94	0.92

Tableau 2. Analyse de la précision (*precision*) pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.91	0.95	**0.59**	0.94	0.92
Pollueurs	0.92	0.94	**0.94**	0.94	0.93
accuracy	0.92	0.95	0.68	0.94	0.92
macro avg	0.92	0.95	0.77	0.94	0.92
weighted avg	0.92	0.95	0.78	0.94	0.92

Tableau 3. Analyse du *f1-score* pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.91	**0.94**	0.74	**0.94**	0.92
Pollueurs	0.92	**0.95**	0.58	**0.94**	0.93
macro avg	0.92	0.95	0.66	0.94	0.92
weighted avg	0.92	0.95	0.65	0.94	0.92

Les trois tableaux ci-dessus permettent de comparer la qualité des classifications pour chacune des méthodes utilisées.

Le premier tableau indique le rappel (*Recall*), soit le pourcentage du nombre total de cas positifs prédits correctement par notre modèle. C'est-à-dire la division entre les vrais positifs et la somme

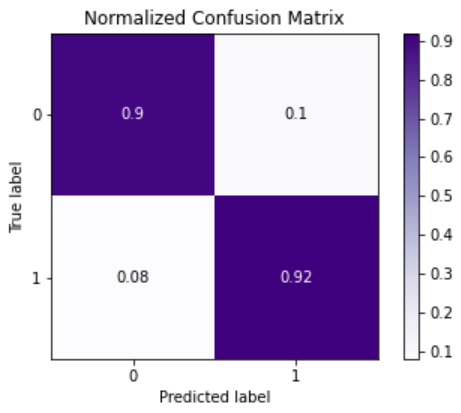
des vrais positifs et des faux positifs. Plus ce pourcentage est élevé, moins les *false negative* sont élevés en proportion. Les pourcentages en police rouge indiquent les nombres qui se démarquent du reste des autres. À cet effet, la classification bayésienne naïve a un pourcentage très élevé soit, de 97% de *recall* pour les utilisateurs légitimes, mais très faibles 41% pour les pollueurs. L'explication pour une telle différence de recall dans la classification est une grande proportion de false positives.

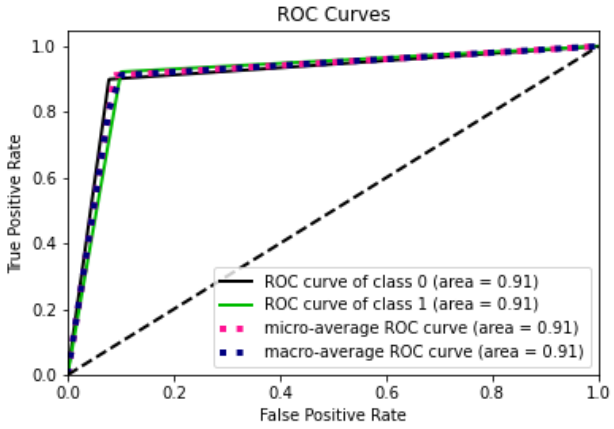
Le second tableau permet l'analyse de la précision qui est calculée en divisant les *True positives* par tous les positives (par tous les positifs, on parle ici de l'addition entre les *True positives* et les *False positives*). Une précision élevée indique que les false positives sont en faible proportion par rapport aux prédictions qui se sont avérées vraies.

Le troisième tableau est la synthèse des « précisions » et des « rappels ». Il représente l'analyse de f1-score, un indicateur, soit la moyenne entre les précisions et rappels des deux premiers tableaux.

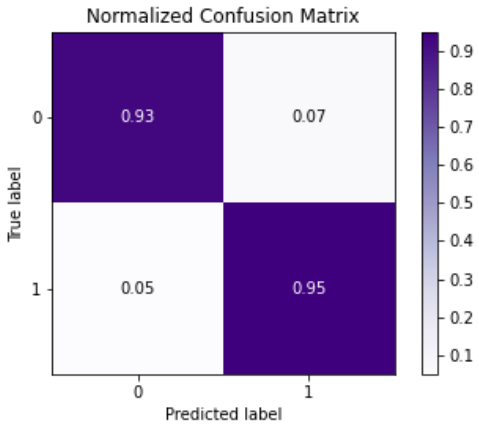
3.2 Tâche 2A Analyse comparative des sept attributs sélectionnés (gain d'*information*)

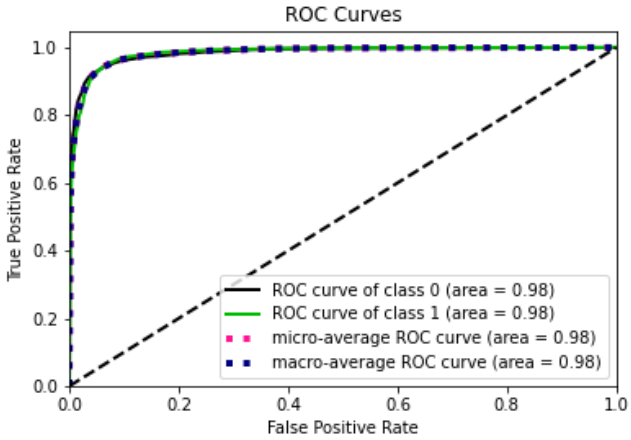
3.2.1 Arbre de décision



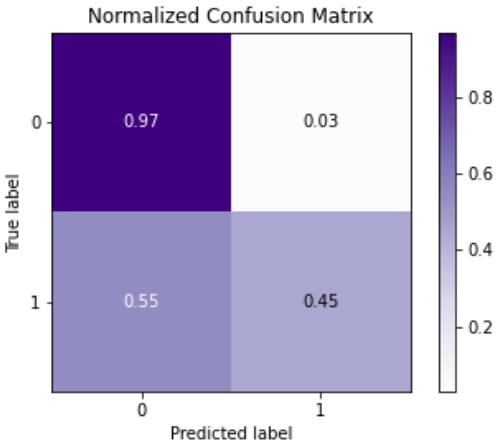


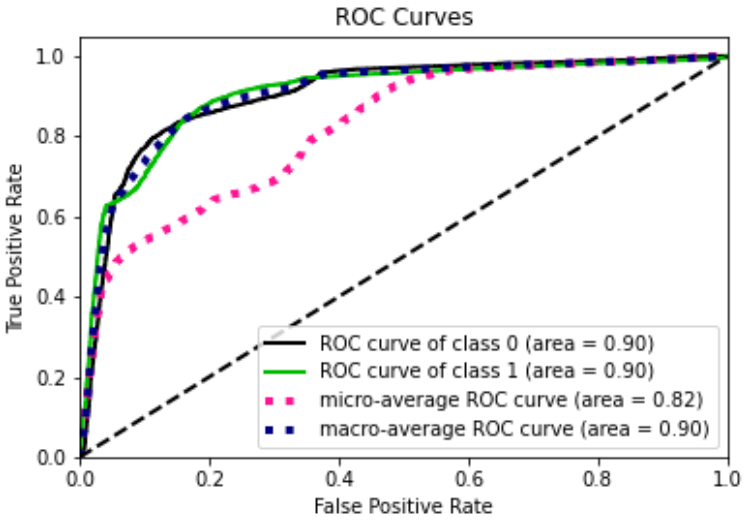
3.2.2 Forêt d'arbres décisionnels (Random Forest)



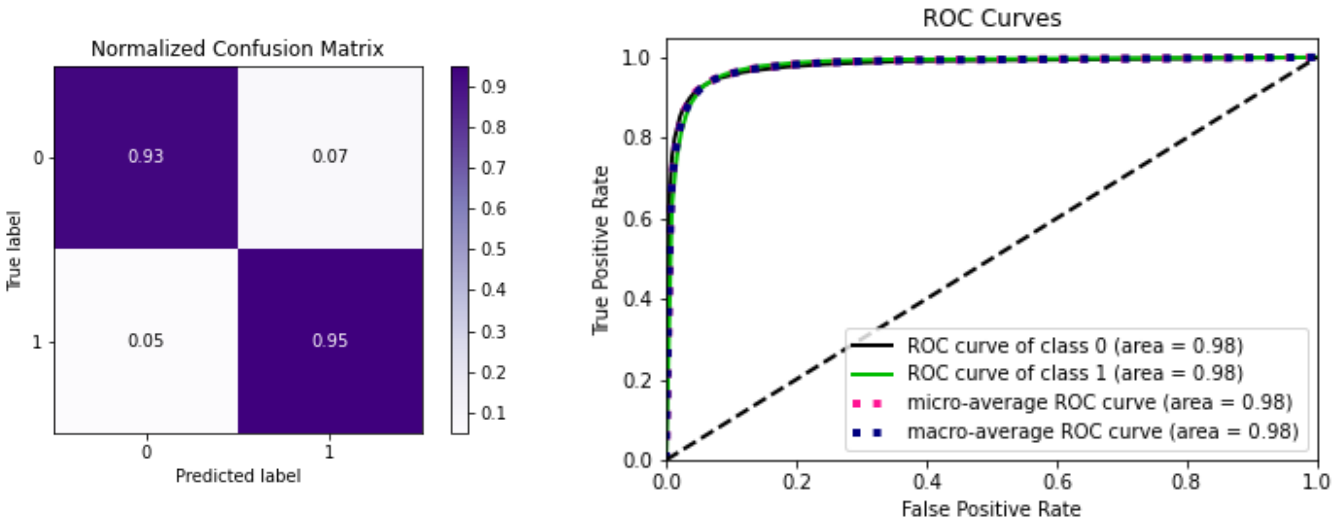


3.2.3 Classification bayésienne naïve





3.2.4 Bagging



3.2.5 AdaBoost

3.2.6 Analyse comparative de la tâche 2.A (gain d'*information*).

Tableau 4. Analyse du rappel (*recall*) pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.90	**0.93**	0.97	**0.93**	0.92
Pollueurs	0.92	**0.95**	0.45	**0.95**	0.92
macro avg	0.91	0.94	0.71	0.94	0.92
weighted avg	0.91	0.94	0.69	0.94	0.92

Tableau 5. Analyse de la précision (*precision*) pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost

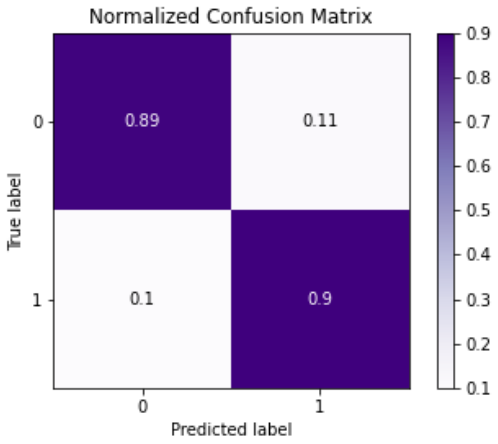
Légitimes	0.91	0.95	0.61	0.94	0.91
Pollueurs	0.91	0.94	0.95	0.93	0.93
accuracy	0.91	0.94	0.69	0.94	0.92
macro avg	0.91	0.94	0.78	0.94	0.92
weighted avg	0.91	0.94	0.79	0.94	0.92

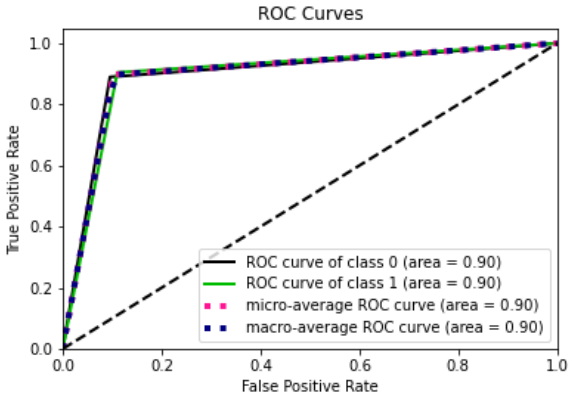
Tableau 6. Analyse du *f1-score* pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.90	**0.94**	0.75	**0.93**	0.91
Pollueurs	0.92	**0.94**	0.61	**0.94**	0.92
macro avg	0.91	0.94	0.68	0.94	0.92
weighted avg	0.91	0.94	0.67	0.94	0.92

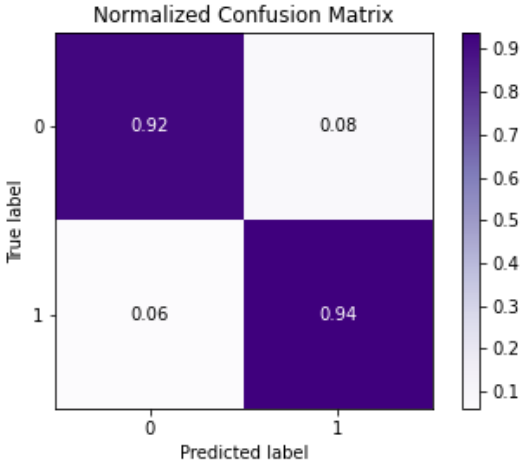
3.3 Tâche 2B Analyse comparative des sept attributs sélectionnés (Test Chi-2)

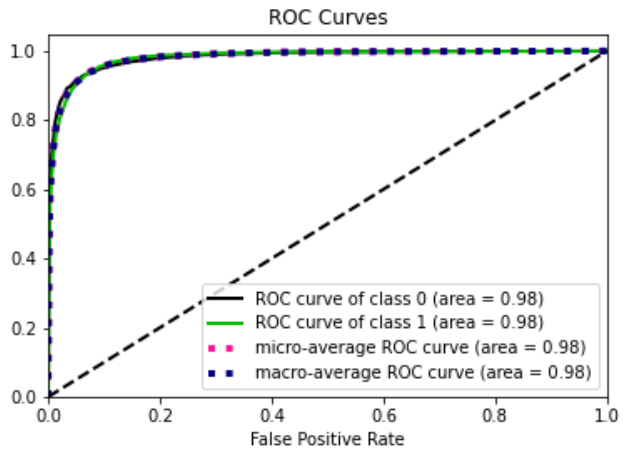
3.3.1 Arbre de décision



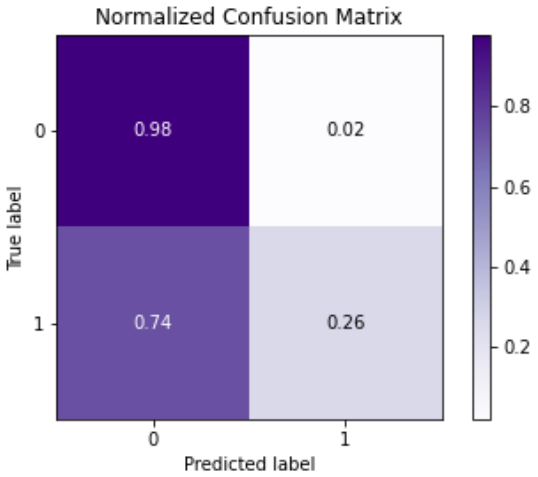


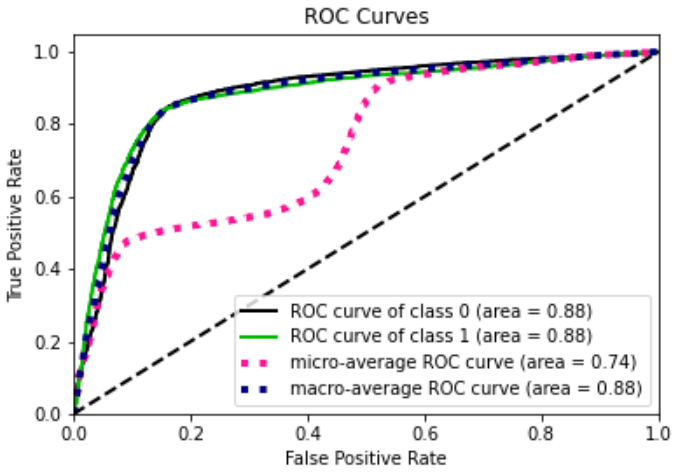
3.3.2 Forêt d'arbres décisionnels (Random Forest)



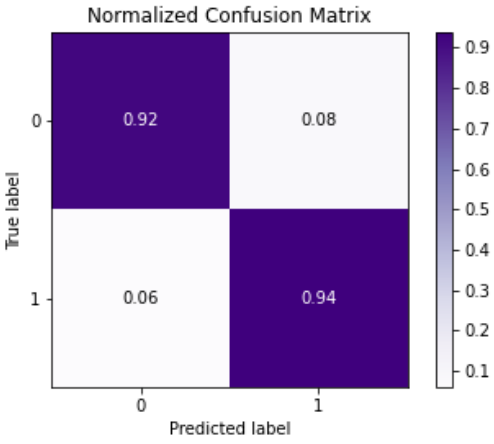


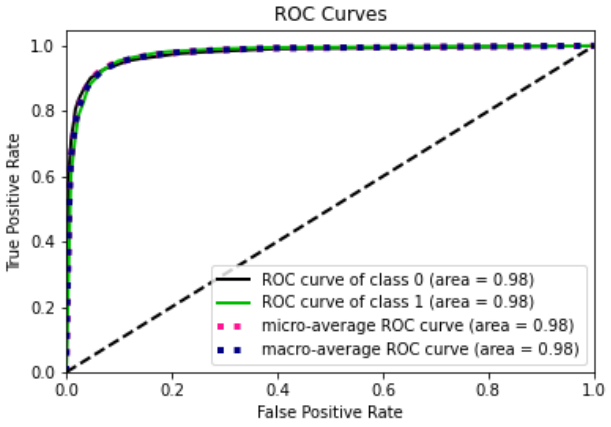
3.3.3 Classification bayésienne naïve



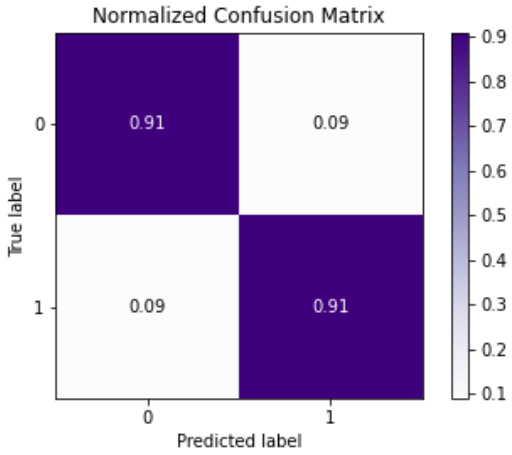


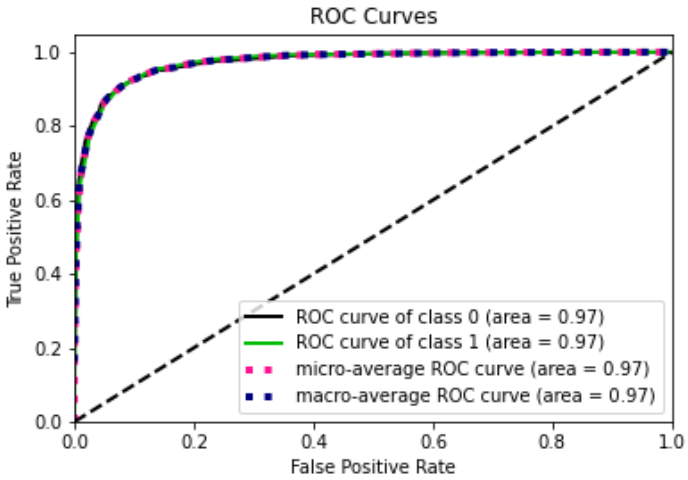
3.3.4 Bagging





3.3.5 AdaBoost





3.3.6 Analyse comparative de la tâche 2.B (Chi-2).

Tableau 7. Analyse du rappel (*recall*) pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.89	0.92	0.98	0.92	0.91
Pollueurs	0.90	0.94	0.26	0.94	0.91
macro avg	0.90	0.93	0.62	0.93	0.91
weighted avg	0.90	0.93	0.60	0.93	0.91

Tableau 8. Analyse de la précision (*precision*) pour chaque méthode de classification

	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.89	0.94	0.54	0.93	0.90
Pollueurs	0.90	0.93	0.93	0.93	0.92
accuracy	0.90	0.93	0.60	0.93	0.91
macro avg	0.90	0.93	0.73	0.93	0.91
weighted avg	0.90	0.93	0.74	0.93	0.91

Tableau 9. Analyse du *f1-score* pour chaque méthode de classification					
	Arbre de décision	Forêts aléatoires	Classification bayésienne naïve	Bagging	AdaBoost
Légitimes	0.89	**0.93**	0.69	**0.92**	0.91
Pollueurs	0.90	**0.94**	0.41	**0.93**	0.92
macro avg	0.90	0.93	0.55	0.93	0.91
weighted avg	0.90	0.93	0.54	0.93	0.91

4 Conclusion

Le but de cet exercice était de comparer cinq types de méthodes de classification supervisée. Il s'est avéré que la méthode des forêts aléatoires était la plus précise des cinq dans les trois tâches, suivi de tout près par le Bagging. Ces deux méthodes se sont démarquées par tous les indices de comparaison calculés. Que ce soit le score F1, que la surface sous la courbe ROC ou bien le *accuracy score*, le Random Forest à 100 estimateurs avec comme criterion le gain d'information et la méthode la plus optimale pour le jeu de données. On a implémenté un code fonctionnel qui calcul adéquatement les 15 attributs, qui a procédé aux analyses et produit des graphiques et tableaux des résultats.

Cependant, il aurait été intéressant de bonifier la section d'explication des tableaux des résultats et de pousser l'analyse. Au niveau du code, il aurait été intéressant de mieux annoter certaines étapes. Par exemple, dans le calcul de la classification par arbre de décision, il aurait été pertinent de limiter la profondeur de l'arbre afin d'éviter de *l'overfitting*.

5 Bibliographie

Lee, K., Eoff, B., & Caverlee, J. (2021). Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter. Proceedings of the International AAAI Conference on Web and Social Media, 5(1), 185-192. Retrieved from https://ojs.aaai.org/index.php/ICWSM/article/view/14106