INF5081 Gestion et analyse de données – Hiver 2024
Travail pratique #1
Professeur : Mohamed Bouguessa
Envoyé le 23 janvier 2024
Date de remise : le ~~1er~~ 15 mars 2024 à 09h30

Important

1. Le TP est noté sur 20 points.
2. Le travail peut être fait individuellement ou en équipe de deux étudiant-e-s maximum.
3. Vous devez soumettre :

   a. Le training dataset que vous avez préparé dans un fichier csv. Ne pas remettre les fichiers de données originaux.

   b. Les fichiers de code Python.

   - Donner des noms pertinents à vos fichiers. Exemple *preprocessing.py* (pour le prétraitement de données), *features_extraction.py* (pour le calcul des attributs), *comparaison_full_features.py* (pour la comparaison des algorithmes), *selection_information_gain.py* (pour la sélection d'attributs avec le InfoGain)
   - Il est important de fournir un code bien organisé, et de bien décomposer les différentes tâches du TP dans différents fichiers sources.
   - Inclure des commentaires qui expliquent les fonctions que vous avez implémentées.
   - Inclure un fichier texte (exemple Readme.txt) qui aide le correcteur à exécuter votre code.
   - **Il faut soumettre que des fichiers .py (pas de fichier Jupyter Notebook).**

   c. Votre rapport en format pdf, identifié à votre nom. Le rapport doit suivre les directives mentionnées dans l'énoncé.

   d. Remettre le training dataset (format csv), le rapport **__en pdf seulement__**, les fichiers de code Python dans un seul fichier zip identifié à votre nom. N'utilisez pas un nom générique (exemple : TP1). **__Fichier zip seulement__**. Il est de votre responsabilité de s'assurer que vos fichiers s'ouvrent correctement sans aucun problème.

   e. Une seule soumission par équipe de deux étudiant-e-s.

4. Soigner la présentation et la rédaction de votre rapport. Suivre les directives mentionnées dans l'énoncé. **Ne pas faire une analyse sommaire/générique/superficielle. Ne pas inclure des captures d'écran de vos résultats sans prendre le temps les biens présenter sous la forme de graphiques ou des tableaux.**
5. La date de remise est le 15 mars à 9h30**.** Moodle ferme à 9h30. **Aucun travail ne sera accepté à partir de cette heure, peu importe les raisons techniques.**
6. **La remise se fait via **Moodle uniquement. Aucune remise par courriel**.**
7. **Les soumissions par courriel ne seront pas corrigées ET le TP sera considéré comme non remis.**

Énoncé

Le but de ce travail est d'analyser et critiquer la performance de quelques algorithmes de classification : Arbre de décision, Forêt d'arbres décisionnels (Random Forest) et la classification bayésienne naïve. Pour tester ces algorithmes, vous pouvez utiliser des librairies Python qui représentent l'implémentation de plusieurs algorithmes d'apprentissage automatique. Pour réaliser cette comparaison, vous devez travailler avec des ensembles de données collectées à partir de Twitter et qui représentent deux catégories d'utilisateurs: utilisateurs pollueurs (classe *Content Polluters*) et utilisateurs « normaux/légitimes » (classe *Legitimate Users*). Principalement, les données que vous devez analyser contiennent 2 353 473 tweets postés par 22 223 utilisateurs malicieux (*content polluters*). Et 3 259 693 tweets postés par 19 276 utilisateurs légitimes (*legitimate users*).
Les ensembles de données (6 fichiers textes) sont disponibles dans moodle.

Votre tâche consiste à effectuer une comparaison entre la technique d'arbre de décision, l'approche *Random Forest* et la classification bayésienne naïve. Pour pouvoir réaliser cette analyse comparative, vous devez d'abord utiliser les données collectées (les six fichiers texte) pour bâtir un ensemble d'apprentissage exploitable par les algorithmes d'apprentissage. Pour construire cet ensemble d'apprentissage, il faut calculer, pour chaque utilisateur, un certain nombre de caractéristiques (*features*) qui aident à distinguer les utilisateurs malicieux (*content polluters*) des utilisateurs normaux (*legitimate users*). Vous devez implémenter les *features* suivants :

1. La longueur du nom de l'usager (*the length of the screen name*)
    - Les pollueurs ont une tendance à avoir un nom générique, et alors la taille de leur nom va être plus arbitraire (parfois trop longue) que celle des utilisateurs légitimes.

2. La longueur de la description du profil de l'usager (*the length of description)*
    - Les pollueurs ont une tendance à ne pas avoir une description pour leur profil, ou bien à avoir une description générique/courte.

3. La durée de vie du compte (t*he longevity of the account)*
    - Les comptes des pollueurs ont une tendance à avoir une courte vie.

4. Le nombre *de following*
    - Les pollueurs ont une tendance à suivre beaucoup d'utilisateurs en comparaison avec un utilisateur régulier.

5. Le nombre de *followers*
    - Les pollueurs ont une tendance à être suivis par peu d'utilisateurs que les utilisateurs réguliers.

6. L'écart type des IDs numériques uniques des *following* (*the standard deviation of unique numerical IDs of following*) → utiliser la série des IDs *SeriesOfNumberOfFollowings* disponibles dans les fichiers *content_polluters_followings* et *legitimate_users_followings*
    - Les pollueurs ont une tendance à suivre beaucoup d'utilisateurs et les suppriment rapidement. Ceci rend l'écart type des followings des utilisateurs réguliers plus bas que celui des pollueurs, car les valeurs sont balancées et pas aussi dispersées que celles des pollueurs.

7. Le rapport *following / followers*
    - La proportion entre les followings et followers des utilisateurs réguliers est plus balancée que celle des pollueurs.

8. Le nombre total des tweets envoyés
    - Les pollueurs ont tendance à tweeter beaucoup plus fréquemment que les utilisateurs réguliers.

9. Le nombre de tweets envoyé par jour
    - Les pollueurs ont tendance à créer beaucoup trop de contenu (tweets) en une journée, ce qui n'est généralement pas le cas pour un utilisateur régulier.

10. Le rapport nombre de tweets par rapport à la durée de vie du compte : |Tweets| / durée de vie du compte
    - Les pollueurs ont tendance à créer un compte pour une courte durée, publier beaucoup de tweets et puis restent inactifs ou suppriment le compte.

11. Le rapport des adresses URL par rapport au nombre de tweets : |TweetURL| / |Tweets|
    - La proportion des URLs dans les tweets des pollueurs est plus enlevée que celle des non-pollueurs.

12. Le nombre moyen d'URL par tweet
    - Les pollueurs ont tendance à inclure beaucoup d'URLs dans leurs tweets.

13. Le rapport des mentions @ par rapport au nombre de tweets : |@username| in tweets / |Tweets|
    - Les pollueurs ont tendance de faire mention à d'autres utilisateurs dans leurs tweets plus fréquemment que les utilisateurs réguliers.

14. Le temps moyen (en minutes) entre deux tweets consécutifs
    - Les pollueurs ont tendance à envoyer des tweets en masse en un court laps de temps.

15. La valeur de temps maximal (en minutes) entre deux tweets consécutifs
    - Les pollueurs ont tendance à envoyer des tweets en masse en un court laps de temps.

Une fois les *features* sont calculés (et ce pour chaque utilisateur), il faut préparer votre ensemble d'apprentissage en fonction du format de données accepté par l'implémentation des algorithmes considérés dans la comparaison (il faut regrouper tous les utilisateurs dans un même fichier et ajouter une colonne qui désigne la classe : 1 pour les pollueurs et 0 pour les utilisateurs réguliers). **Une façon simple consiste à utiliser le format CSV.**

Cette étape représente une phase de nettoyage afin de rendre les données cohérentes et prêtes pour l'analyse. Cette étape consiste entre autres :

- Suppression des instances de données en double.
- Remplacement des valeurs manquantes (remplacé par la médiane par exemple).
- Normalisation des données (approche z-score).

Une fois la préparation des données est terminée, on vous demande de réaliser les deux tâches suivantes :

Tâche 1 : analyse comparative en utilisant tous les *features* décrits ci-haut

- Appliquer les algorithmes : Arbre de décision, Random Forest et la classification bayésienne naïve.
- Comparer la performance de ces algorithmes en se basant sur :
    - Le taux des vrais positifs (TP Rate) – de la classe *Content Polluters*
    - Le taux des faux positifs (FP Rate) – de la classe *Content Polluters*
    - F-measure de la classe *Content Polluters*
    - La surface sous la courbe ROC (AUC)
- Illustrer vos résultats dans des tables et/ou graphes. ****Ne pas inclure de simples captures d'écran qui relatent les résultats ****
- Analyser vos résultats. Indiquer les points forts et points faibles de chaque algorithme et expliquer pourquoi. ****Il est très important de bien rédiger cette partie et par l'élaboration d'une analyse consistante et pas une analyse courte, superficielle et générique****

Tâche 2 : analyse comparative avec sélection d'attributs

Identifier les **7 meilleurs features** en utilisant le gain d'information.

- Monter les 7 meilleurs attributs dans un tableau (pas de captures d'écran).
- Appliquer les algorithmes : Arbre de décision, Random Forest et la classification bayésienne naïve **sur l'ensemble de données, mais avec les 7 attributs sélectionnés seulement.**
- Comparer la performance de ces algorithmes en se basant sur
    - Le taux des vrais positifs (TP Rate) – de la classe *Content Polluters*
    - Le taux des faux positifs (FP Rate) – de la classe *Content Polluters*
    - F-measure de la classe *Content Polluters*
    - La surface sous la courbe ROC (AUC)
- Illustrer vos résultats dans des tables et/ou graphes (pas de capture d'écran!)

- Quelles sont vos conclusions par rapport aux tests effectués sur l'ensemble de données avec tous les attributs (tests effectués dans Tâche 1)?

Voici les librairies qui peuvent être utilisées :

1. **Scikit-Learn** : Bibliothèque libre et open source implémentée avec Python, dédié à l'apprentissage automatique. Elle est conçue pour s'harmoniser avec d'autres bibliothèques Python (ou autres), notamment NumPy, SciPy, etc.

2. **Scikit-plot** : Bibliothèque libre et open source implémentée avec Python, utilisé pour l'affichage des graphiques principalement les courbes ROC et AUC.

3. **Numpy** : "Numerical Python" fournit une interface pour stocker et effectuer des opérations sur les données. D'une certaine manière, les tableaux Numpy sont comme les listes en Python, mais Numpy permet de rendre les opérations beaucoup plus efficaces, surtout sur les tableaux de grande taille qui sont au cœur de l'écosystème de la Data Science.

4. **Pandas** : Fournis deux structures de données fondamentales, la "Série" et le "DataFrame". On peut voir ces structures comme une généralisation des tableaux et des matrices de Numpy. La différence entre les structures de Pandas et celles de Numpy c'est la définition explicite par l'utilisateur des indices et des index sur les objets (matrices).

5. **Pickle** : Module Python utilisé pour sérialiser et dé-sérialiser les structures d'objets Python. La sérialisation (ou « pickling ») fait référence au processus de conversion d'un objet en mémoire en un flux d'octets pouvant être stocké sur disque ou envoyé sur un réseau. Plus tard, ce flux de caractères peut être récupéré et dé-sérialiser (ou «unpickling») en retour vers un objet Python.

6. **Matplotlib** : "Mathematic Plot library" est une bibliothèque de traçage Python 2D qui produit des figures de qualité de publication dans une variété de formats papier et d'environnements interactifs entre plates-formes. Matplotlib peut être utilisé dans les scripts Python, les Shells Python et IPython, les notebook Jupyter, etc.