Rapport de projet d'introduction à l'apprentissage statistique

Projet Réalisée par

BERKENNOU Brahim

BOUKARI Idir

OUKSILI Samy

NAIT-ALI Idir

1. Sujet du projet :

Réalisation d'un filtre anti-spam.

2. Dataset utilisé :

- https://www.kaggle.com/venky73/spam-mails-dataset

3. Liens consultés :

- https://towardsdatascience.com/how-to-identify-spam-using-natural-language-processing-nlp-af91f4170113
- https://www.analyticsvidhya.com/blog/2021/06/automated-spam-e-mail-detection-modelusing-common-nlp-tasks/
- https://medium.com/analytics-vidhya/build-email-spam-classification-model-using-python-and-spacy-a0c914a83f4d
- https://towardsdatascience.com/how-to-build-and-apply-naive-bayes-classification-for-spam-filtering-2b8d3308501

4. Objectif de projet :

L'objectif est de détecter si un mail est indésirable ou légitime à partir de son objet et corps donnés dans le dataset. Ce dernier est composé d'une seule principale feature qui est le mail lui-même.

Cette tâche se repose sur l'apprentissage supervisé, comme on a deux classes :
spam et non-spam, on adoptera la classification binaire pour notre dataset afin de Différencier un spam d'un mail.
On a choisi pour cette fin le modèle Naïve Bayes pour les raisons suivantes :
Le fonctionnement de ce modèle s'appuie sur le calcul de probabilité qu'un mail soit un spam en établissant une corrélation entre la présence de certains éléments (en général des mots) récurrents dans les courriers électroniques indésirables (spam) et le fait qu'ils soient aussi présents Dans le corps de ce mail.
Cette technique répondra parfaitement à nos attentes puisqu'elle nous permet d'assurer un taux de faux positifs très bas (mails indésirables Déclarés comme légitimes).
Cette méthode reste très facile à construire et s'applique sur de larges dataset, ce qui est clairement notre cas ici.

Les classes de notre dataset ne sont pas équilibrées (29% spam, 71% non-spam).
C'est très évident qu'un faux non-spam est un danger pour l'utilisateur. De même, un Faux spam est embêtant aussi, car on peut jeter en spam un mail qui peut être très Important.

5. Modèles utilisés :

5.1. Modèle Naïve-Bayes n°1 :

5.1.1 Étapes d'implémentations :

5.1.1.1. Nettoyage des données :

- Supprimer les colonnes inutiles.
- Supprimer la syntaxe.
- Mettre toutes les lettres en minuscule.
- Transformer les mails en liste de mots.
- Mélanger notre ensemble de données.

5.1.1.2. Extraction des features :

- Construction d'un vocabulaire de mot unique.
- Calcule du nombre d'apparition des mots dans les spam et ham.

5.1.1.3. Utilisation des formules mathématique :

- P_spam/(w1,w2……,wn) ≃ P_Spam * ∏ P_wi/spam Probabilité qu'un mail soit un spam sachant les mots qui le compose.
- P_spam = (nombre de spam) / (taille du data set) ratio des spam dans notre ensemble d'entraînement.
- P_wi/spam = (Nombre d'apparition du mot dans les spam + Alpha)/ (Nombre de mot dans les spam + Alpha ***** ((le nombre de mots unique dans le data set)) : probabilité de trouver wi dans des spams.
- Alpha : un hyper paramètre pour gérer les cas où un mot n'a jamais été vue.
- L'inverse e pour les ham.

5.1.1.4. Optimisation des hypers paramètres :

- Alpha : ce paramètre évite d'avoir une probabilité = 0 qui fausserait les calculs.
- Après les est effectuer nous avons choisis la valeur 1 pour ce Paramètre en ce basant sur le score que fait le modèle.

5.2. Modèle Naïve-Bayes n°2 :

5.2.1. Étapes d'implémentations :

5.2.1.1. Nettoyage des données :

- Supprimer les caractères spéciaux en utilisant la librairie re.
- Convertir tous texte des mails en minuscule.
- Découpage de chaque texte d'émail en une liste de mots en découpant par rapport aux espaces.
- Supprimer les stopWords qui n'ont pas de poids sur la probabilité de décision si ce mot est est fréquent dans les mails ou les spams, à l'aide de librairie ntlk qui fournie tous les stopWords de la langue anglaise.
- Convertir les mots restants en leurs base significative à l'aide de l'outil WordNetLemmatizer fournis par la librairie ntlk.
- Reformer le texte du message nettoyé.

5.2.1.2. Vectoriser les mots de chaque message :

- Pour pouvoir calculer la probabilité de présence de chaque mot dans un message, et pouvoir calculer également la probabilité qu'un mot soit fréquent dans les spam ou les mails par la suite. On vectorise les mots en utilisant la méthode de TF-IDF pour calculer la probabilité de présence de chaque mot dans chaque message.

5.2.1.3. Échantillonnage :

- Maintenant qu'on possède un tableau de tableau (Le corpus) de probabilité représentant la probabilité de présence de chaque mot dans les spams et dans les mails, et vu qu'on possède un ensemble de données déséquilibré on procédera à l'échantillonnage de ce dernier pour équilibrer ces deux classes avant de passer à l'entraînement et la prédiction du modèle.

Et pour ce fait on distingue différentes méthodes :

5.2.1.3.1 Génération d'échantillons synthétiques : qui consiste à générer des échantillons synthétiques de manière automatique et là également on distingue deux types d'algorithmes :

- Les algorithmes de sur-échantillonnage en créant des échantillons synthétiques à partir de la classe minoritaire au lieu de créer de simples copies.
- Les algorithmes de Sous-échantillonnage en générant un certain nombre de centroïdes à partir des données d'origine, afin de perdre le moins d'information possible sur la classe majoritaire, lorsque celle-ci doit être réduite afin d'arriver à un ratio (classe minoritaire/ classe majoritaire) satisfaisant.

5.2.1.3.2 Échantillonnage Classique :

- Sur-échantillonnage : Oversampling qui consiste à augmenter le nombre d'éléments de la classe minoritaire afin d'arriver à un ratio classe minoritaire/ classe majoritaire satisfaisant.
- Sous-échantillonnage : Undersampling qui consiste à diminuer le nombre d'éléments de la classe majoritaire afin d'arriver à un ratio classe minoritaire/ classe majoritaire satisfaisant.

5.2.1.4 Choix du meilleur ratio pour chaque méthode :

- Ces méthodes d'échantillonnage possèdent un hyper paramètre important qui est le ratio, donc par la suite on fera plusieurs échantillonnages sur le corpus en variant le ratio pour en tirer le meilleur ratio d'échantillonnage de chaque méthode qui permet d'avoir le plus haut score en accuracy.

5.2.1.5 Entraînement, Prédiction, Résultats :

- On applique chacune de ces méthodes avec son meilleur ratio et puis on lance l'entraînement du modèle MultipolynomialNB sur les données échantillonnées.
- Après l'entraînement du modèle on passe à la prédiction et l'évaluation de cette dernière.
- Remarques et analyses des résultats de chaque prédiction avec chaque méthode d 'échantillonnage.
- Comme toutes les méthodes ont était appliquer avec la meilleure valeur du ratio de chacune, on va pourvoir désigner la meilleure méthode d'échantillonnage dans notre cas avec laquelle on a pu avoir la meilleure valeur des scores : accuracy, précision, rappel, auc,..

5.3. TF-IDF : est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes. Elle permet d'évaluer la pertinence d'un mot dans un document relativement à un corpus, en se basant sur la formule suivante :

tf-idf = tf * idf

où :

- TF représente la fréquence du terme dans le document.
- TF se calcule de la manière suivante : le nombre d'occurrences du terme dans un document divisé par le nombre total de termes de ce document.
- IDF est la fréquence inverse de document, elle mesure l'importance du terme dans l'ensemble du corpus.
- Dans le schéma TF-IDF, idf vise à donner un poids plus important aux termes les moins fréquents. Elle consiste à calculer le logarithme de l'inverse de la proportion de documents du corpus qui contiennent le terme

5.3. Modèle NLP (Natural Language Processing):

5.3.1. Étapes d'implémentations :

5.3.1.1. Récupération des features nécessaires.

5.3.1.2. Nettoyage des données :

- On construit un vocabulaire en ignorant les termes qui apparaissent dans plus de 50% des messages du dataset car ils sont peu susceptibles d'aider à prédire si le message est un spam ou pas.
- On élimine également les stop words qui n'ajoutent pas beaucoup de sens aux messages puisqu'ils peuvent être ignorés en toute sécurité sans sacrifier la probabilité qu'un message soit un email ou un spam.

5.3.1.3. Vectorization :

On transforme les messages en une matrice de vecteurs :

- Colonne : tous les mots uniques qui restent dans le dataset après le nettoyage.
  - Ligne : chaque message est représenté par la fréquence de chaque mot unique du dataset dans ce dernier.

5.3.1.4. TF-IDF:

On calcule les scores tf-idf pour chaque vecteur message.

- Objectif : réduire l'impact des tokens qui se produisent très fréquemment et qui sont les moins informatifs.

5.3.1.5. Entraînement et test :

- On divise la matrice de vecteur tf-idf en sous-ensembles d'entraînement et de tests aléatoires.
- On répartit le dataset comme suivant : 70% pour le train et 30% pour le test.

5.3.1.6. Model Support Vector Classification :

- On utilise la méthode SVM pour ajuster le modèle en fonction des données d'entraînement données.

5.3.1.7. Score et matrice de confusion :

- Après les tests effectués sur plus de 1500 messages, on a réussi à avoir un très bon score de 0.99.
- On a uniquement un spam déclaré comme ham et 9 ham déclaré comme spam.