**Rapport de Mini-Projet**

# **Sujet :** Détection de SPAM – NLP

## **Introduction :**

Le SPAM ou courriel indésirable est une communication électronique non sollicitée, en premier lieu via le courrier électronique. Il s'agit en général d'envois en grande quantité effectués à des fins publicitaires. Le présent rapport décrit minutieusement le processus de création de système de détection de SPAM.

## **Technologies utilisées :**

La mise en œuvre de ce projet a été effectuée sur ‘***Jupyter Notebook***’ en Python 3, et en utilisant les outils suivants :

* **NumPy**
* **Pandas**
* **Matplotlib**
* **Seaborn**
* **Regex**
* **NLTK**
* **SKLearn**

## **DataSet :**

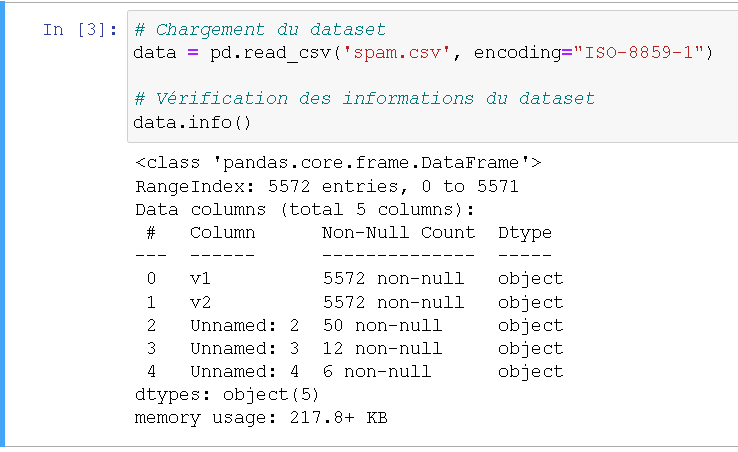
Le fichier **« spam.csv »** joint à ce rapport, contient un message par ligne et chaque ligne est composée de deux colonnes :

* La première contient le label (**HAM** / **SPAM**)
* La deuxième contient le texte brut.

## **Procédure :**

L’élaboration de ce système de détection de SPAM consiste à utiliser les données du DataSet pour entrainer le modèle, afin qu’il soit capable de déterminer si un message donné peut être considéré comme SPAM ou pas. Ceci se fait en plusieurs étapes comme suit :

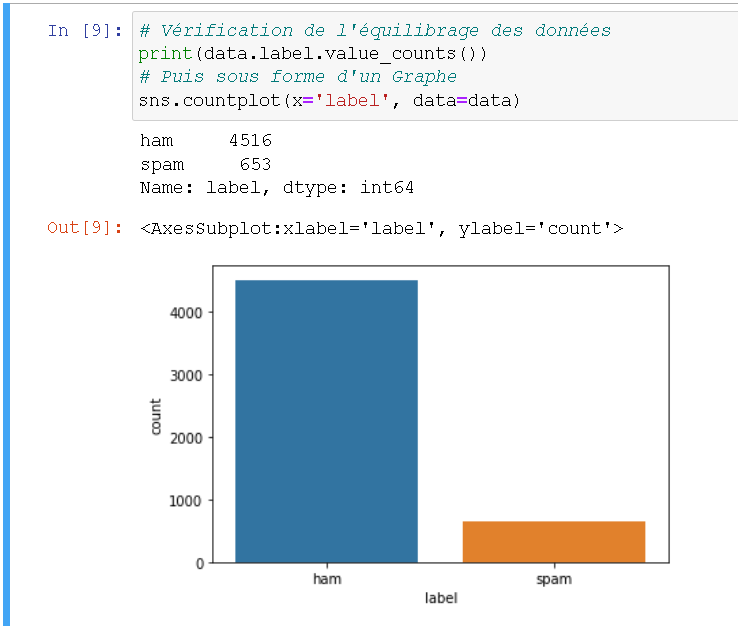
1. Importation des bibliothèques nécessaires : Numpy, NLTK, SKLearn …
2. Chargement du DataSet depuis le fichier .csv et affichage des informations le concernant :



1. Nettoyage du DataSet afin de le minimaliser. Ceci consiste à la suppression des colonnes inutiles, la suppression des redondances et la vérification de l’équilibre des données.



La figure suivante représente la vérification de l’équilibrage des données :



1. La création de caractéristiques permettant d’appliquer l’ingénierie des fonctionnalités afin d’améliorer les performances en sélectionnant les plus importantes et en réduisant la taille de l'ensemble des caractéristiques ce qui rend le calcul plus rapide et efficace.

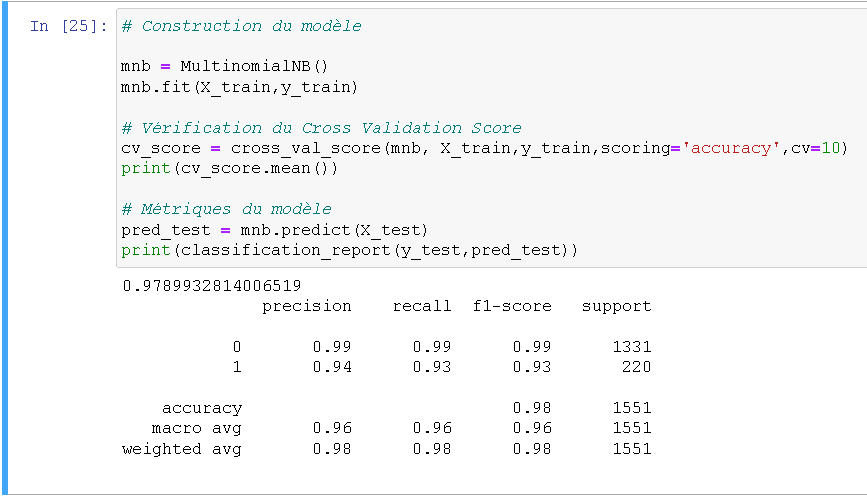


1. Le Pre-processing qui permet de rendre les données plus compactes et bien plus utilisables par le modèle. Ceci consiste à :
   1. La Tokenization,
   2. La suppression des mots vides,
   3. La Vectorisation,
   4. La Lemmatisation,
   5. La conversion en tableau…

(Le fichier ‘.ipynb’ joint à ce rapport illustre ce Pre-processing sous 2 formes différentes : manuellement, et en utilisant ‘CountVectorizer’)

1. La division du DataSet en Train Data et Test Data
2. La création du modèle et le calcul du Cross Validation Score et des métriques du modèle (la précision, le ‘recall’ et le score F1 pour chaque classe. Ainsi que la précision, la moyenne macro et la moyenne pondérée)

La figure suivante illustre les résultats retournés :



1. Utilisation de nouvelles données de Test afin de vérifier l’efficacité du modèle. Ces données doivent être passées en Pre-processing avant la prédiction.

## **Conclusion et récapitulatif :**

Un modèle dont le score F1 est de 98 % est un bon modèle à suivre. N'oubliez pas, cependant, que ces résultats sont basés sur les données d'entraînement que nous avons utilisées. Lorsque nous appliquons un modèle

comme celui-ci à des données réelles, nous devons surveiller activement les performances du modèle au fil du temps. Nous pouvons également continuer à améliorer le modèle en réagissant aux résultats et aux commentaires, par exemple en ajoutant des fonctionnalités et en supprimant les mots mal orthographiés.

Dans ce rapport, nous avons développé un modèle basé sur le ‘Multinomial Naive Bayes classifier’ qui a été largement utilisé dans les problèmes NLP par rapport aux autres algorithmes d'apprentissage automatique, tels que les SVM et les réseaux de neurones, en raison de leur taux d'apprentissage rapide et de leur conception simple. Dans la classification de textes, ils donnent un taux de précision plus élevé malgré leur forte hypothèse naïve.

**Fichiers Joints au rapport :**

‘*Détection de SPAM (NLP) - LAZHAR Abdellah IDDLo T.A.ipynb*’

‘*Spam.csv’*