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 :

```
In [3]: # Chargement du dataset
        data = pd.read_csv('spam.csv', encoding="ISO-8859-1")

        # Vérification des informations du dataset
        data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

3.) 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.

```
In [4]: # Suppression des colonnes inutiles et mise à jour des noms des colonnes
        data.rename(columns={'v1':'label','v2':'mail'}, inplace=True)
        data.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'], axis=1, inplace=True)
```

```
In [5]: # Affichage d'un aperçu du DataSet
        data.head()
```

Out[5]:

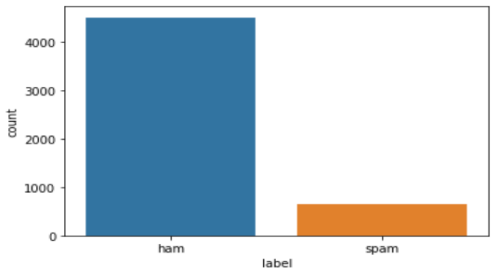
	label	mail
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

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

```
In [9]: # Vérification de l'équilibrage des données
        print(data.label.value_counts())
        # Puis sous forme d'un Graphe
        sns.countplot(x='label', data=data)

        ham     4516
        spam     653
        Name: label, dtype: int64

Out[9]: <AxesSubplot:xlabel='label', ylabel='count'>
```



4.) 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.

```
In [11]: # Création de caractéristiques
         data['N° de caractères'] = data.mail.apply(lambda x: len(x))
         data['N° de mots'] = data.mail.apply(lambda x: len(nltk.word_tokenize(x)))
         data['N° de phrases'] = data.mail.apply(lambda x: len(nltk.sent_tokenize(x)))

         data.describe()
```

Out[11]:

	N° de caractères	N° de mots	N° de phrases
count	5169.000000	5169.000000	5169.000000
mean	78.977945	18.455407	1.961308
std	58.236293	13.322448	1.432583
min	2.000000	1.000000	1.000000
25%	36.000000	9.000000	1.000000
50%	60.000000	15.000000	1.000000
75%	117.000000	26.000000	2.000000
max	910.000000	220.000000	38.000000

5.) Le Pre-processing qui permet de rendre les données plus compactes et bien plus utilisables par le modèle. Ceci consiste à :

a. La Tokenization,
b. La suppression des mots vides,
c. La Vectorisation,
d. La Lemmatisation,
e. La conversion en tableau…

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

6.) La division du DataSet en Train Data et Test Data

7.) 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 :

```
In [25]: # Construction du modèle

         mnb = MultinomialNB()
         mnb.fit(X_train,y_train)

         # Vérification du Cross Validation Score
         cv_score = cross_val_score(mnb, X_train,y_train,scoring='accuracy',cv=10)
         print(cv_score.mean())

         # Métriques du modèle
         pred_test = mnb.predict(X_test)
         print(classification_report(y_test,pred_test))

         0.9789932814006519
                       precision    recall  f1-score   support

                    0       0.99      0.99      0.99      1331
                    1       0.94      0.93      0.93       220

             accuracy                           0.98      1551
            macro avg       0.96      0.96      0.96      1551
         weighted avg       0.98      0.98      0.98      1551
```

8.) 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*'