République Algérienne Démocratique Et Populaire

Ministère De L'enseignement Supérieur Et De La Recherche Scientifique

Université De Sciences et Technologies Houari Boumediene

Faculté D'électronique et informatique

Département d'Informatique

Rapport

Classification des emails spam

Réalisé par :

- HIMEUR Salah Eddine
- OUHAB Mohamed Ismail
- BENRABBAH Mohammed

M1 SII G2

Soumettre à :

Prof SETITRA

Introduction :

Préparation des données :

Nettoyage de données :

Extraction de caractéristiques et entraînement des modèles :

Résultats :

Conclusion :

Juin 18, 2022

Introduction :

Le courrier électronique est l'un des outils de communication les plus importants. Pour avoir une communication efficace, le filtrage des spams est l'une des fonctions importantes. Et pour ce faire sous python on a procédé comme suit :

Ce projet a été fait en trois phases :

- 1er phase : Préparation des données.
- 2ème phase : Nettoyage de données.
- 3eme phase : l'extraction de caractéristiques et l'entraînement de modèles.

Préparation des données :

on a choisi ces emails comme notre dataset :

Non spam Emails:

- 20021010_hard_ham.tar.bz2
- 20030228_easy_ham_2.tar.bz2
- 20030228_easy_ham.tar.bz2

Spam emails:

- 20021010_spam.tar.bz2
- 20050311_spam_2.tar.bz2

on extraire ces dossier et on lire le contenu de ses fichiers, avec la bibliothèques **email** on peut extraire le contenu d'un email, après l'extraction de le contenu de tous les emails, on supprime les e-mails vides et on affecte à chaque email son label (spam ou ham) et on sauvegarde les résultats dans des fichier csv (dirtyHam.csv et dirtySpam.csv) pour le traitée dans le 2eme phase.

Nettoyage de données :

on charge les fichier csv résultant de 1er phase et on applique les étapes suivants pour le nettoyer :

- **Minuscule**.
- **Normalisation des URL**.
- **Normalisation des adresses e-mail**.
- **Normalisation des nombres**.
- **Normalisation des dollars**.
- **Suppression des non-mots**.

tout ces étapes ont été fait par un seul bibliotheque **`cleantext`**

```
def clean_up(email):
    return clean(email,
    lower=True,                      # lowercase text
    no_line_breaks=True,            # fully strip line breaks as opposed to only normalizing them
    no_urls=True,                   # replace all URLs with a special token
    no_emails=True,                 # replace all email addresses with a special token
    no_phone_numbers=True,          # replace all phone numbers with a special token
    no_numbers=True,                # replace all numbers with a special token
    no_digits=True,                 # replace all digits with a special token
    no_currency_symbols=True,       # replace all currency symbols with a special token
    no_punct=True,                  # remove punctuations
    replace_with_punct="",           # instead of removing punctuations you may replace them
    replace_with_url=" httpadr",
    replace_with_email=" adremail",
    replace_with_phone_number=" phone",
    replace_with_number=" number",
    replace_with_digit=" digit",
    replace_with_currency_symbol=" currencySymbol",
    lang="en"                        # set to 'de' for German special handling
        )
```

Suppression de balises HTML.

Radicalisation de mots: on a fait cette étape par le bibliothèque **NLTK** avec sa fonction **`WordNetLemmatizer`** qui convertit les mots en leur forme radicale.

Après cette dernière étape, on sauvegarde les résultats obtenus sous des fichiers csv.

Extraction de caractéristiques et entraînement des modèles :

Comme notre données sont des chaînes de caractères et les classifieurs manipule des caractéristiques entier ou float, on doit trouver un bon représentation numérique pour notre caractéristique, un bon choix est le comptage de nombres de répétition des mots dans les emails, pour réaliser ça on a deux choix :

CountVectorizer: le CountVectorizer conservera un dictionnaire de chaque mot et de son identifiant respectif. Cet identifiant sera lié au nombre de mots de ce mot dans l'ensemble de l'apprentissage.

Un problème avec cette approche est que certains mots comme "the", "and" apparaissent de nombreuses fois et n'apportent pas vraiment d'informations significatives.

TfidfVectorizer[1]: Une autre alternative populaire est TfidfVectorizer. En plus de prendre le nombre de mots de chaque mot, les mots qui apparaissent souvent dans plusieurs documents ou phrases, le vectoriseur va essayer de les réduire.

On charge les fichiers csv obtenue par la 2eme phase et on extraire les caractéristique par **TfidVectorizer** et on utilise la méthode **train_test_split** de **sklearn** pour diviser l'ensemble des données en données d'entraînement et données de test.

On a testé 4 classifieurs :

un classifieur **bayésien naïf.**

un classifieur de **régression logistique**.

un classifieur **SVM** avec un noyau gaussien (**rbf**).

un classifieur **KNN.**

Résultats :

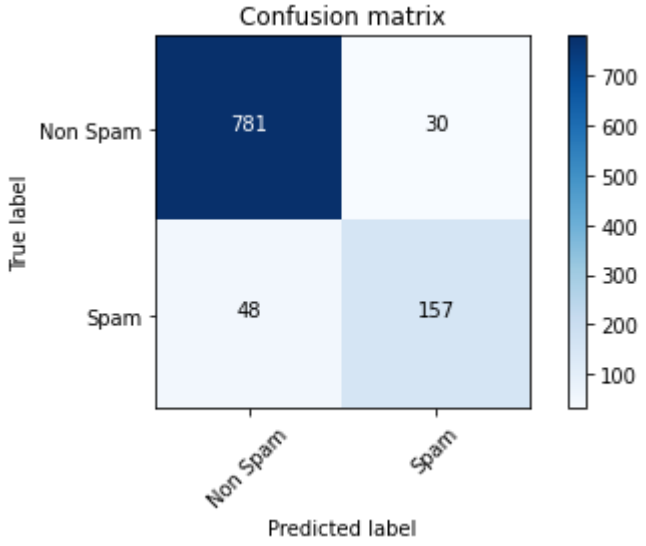
Après l'entraînement de ces algorithmes sur les données d'entraînement on a le testé sur les données de test et on a obtenue les résultats suivants:

Mesures de performance	**bayésien naïf**	**SVM (rbf Kernel)**	**Régression logistique**	**KNN**
Précision	83.96%	97.89%	97.67%	86.61%
Rappel	76.59%	90.73%	81.95%	94.63%
F1-score	80.10%	94.18%	89.12%	90.44%
balanced accuracy	86.44%	95.12%	90.73%	95.47%

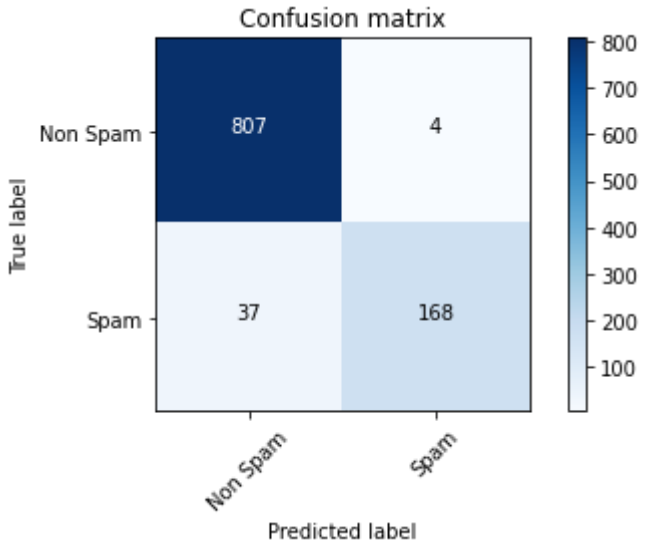
1:https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

et on a obtenue les matrices de confusion suivantes:

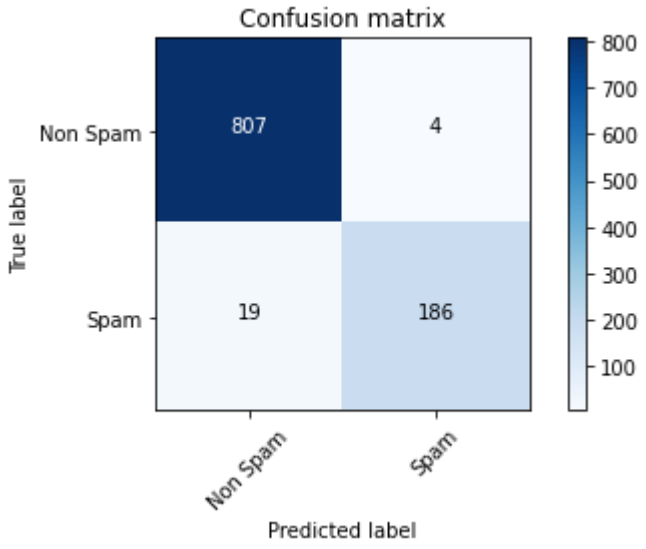
bayésien naïf:



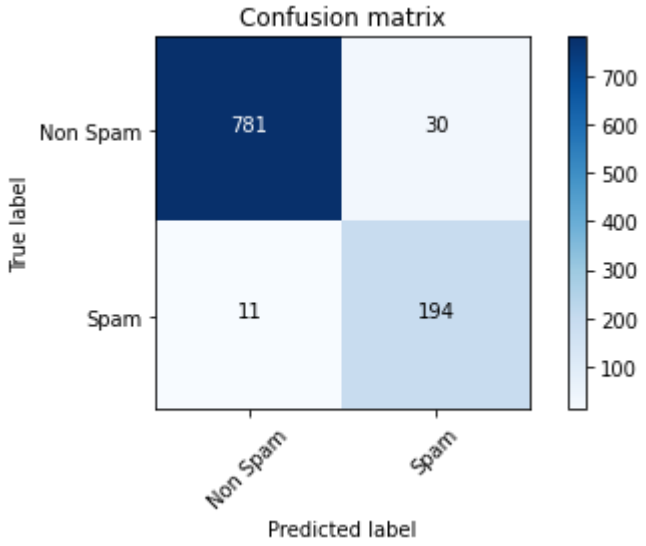
Régression logistique:



SVM:



KNN:



Conclusion :

D'après les résultats obtenus on constate que le modèle SVM avec le Kernel Gaussian « rbf » est le plus efficace pour la classification des emails entre spam et non spam car il nous a donné le meilleur rapport entre la précision et le rappel.

Photos :

BENRABAH Mohamed

OUHAB Mohamed Ismail

Himeur Salah Eddine