**République Algérienne Démocratique et Populaire**
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**
**Université des Sciences et de la Technologie Houari Boumediene**

**Faculté d'Informatique**
Département d'Intelligence Artificielle et Sciences des Données

Projet de fin de semestre

# Apprentissage automatique et réseaux de neurones

**Groupe**:

**BOULARIACHE Abdessamed, G3, 171731033493.**

**KEMOUM Meroua, G3, 171731053329**

**MAHIDDINE Mohamed Amine, G3, 201704000012.**

**TAZIR Mohamed Reda, G3, 161631076578.**

**Professeurs:**

Mme. Setitra
Mme. Belhadi

Date 17/06/2022

# Introduction

De nombreux services de messagerie fournissent aujourd'hui des filtres anti-spam capables de classer les e-mails dans les spams et les non-spams avec une grande précision. Pour ce projet, l'objectif est d'utiliser les notions apprises durant ce semestre pour résoudre un problème de sciences de données. Pour ce projet nous devront implémenter un détecteur de Spam apprenant sur un ensemble de données.

Le classificateur devra classer si un e-mail donné, x, en spam (y = 1) ou non-spam (y = 0). Il faut convertit chaque email en un vecteur de caractéristiques $x \in R$ . L'ensemble de données utilisé $n$ est SpamAssassin Public Corpus. https://spamassassin.apache.org/old/publiccorpus/

## Étape 1 : préparation des données

Nous avons deux groupes de données: Ham et spam.
La figure suivante montre un exemple de chaque groupe (Ham et Spam)

```
  | I'm hoping that all people with no additional sequences will notice are
  | purely cosmetic changes.

Well, first, when exmh (the latest one with your changes) starts, I get...

can't read "flist(totalcount,unseen)": no such element in array
    while executing
"if {$flist(totalcount,$mhProfile(unseen-sequence)) > 0} {
	FlagInner spool iconspool labelup
    } else {
	FlagInner down icondown labeldown
    }"
    (procedure "Flag_MsgSeen" line 3)
    invoked from within
"Flag_MsgSeen"
    (procedure "MsgSeen" line 8)
    invoked from within
```

Figure 01: Exemple d'email data/ham

```
Greetings!

You are receiving this letter because you have expressed an interest in
receiving information about online business opportunities. If this is
erroneous then please accept my most sincere apology. This is a one-time
mailing, so no removal is necessary.

If you've been burned, betrayed, and back-stabbed by multi-level marketing,
MLM, then please read this letter. It could be the most important one that
has ever landed in your Inbox.

MULTI-LEVEL MARKETING IS A HUGE MISTAKE FOR MOST PEOPLE
```

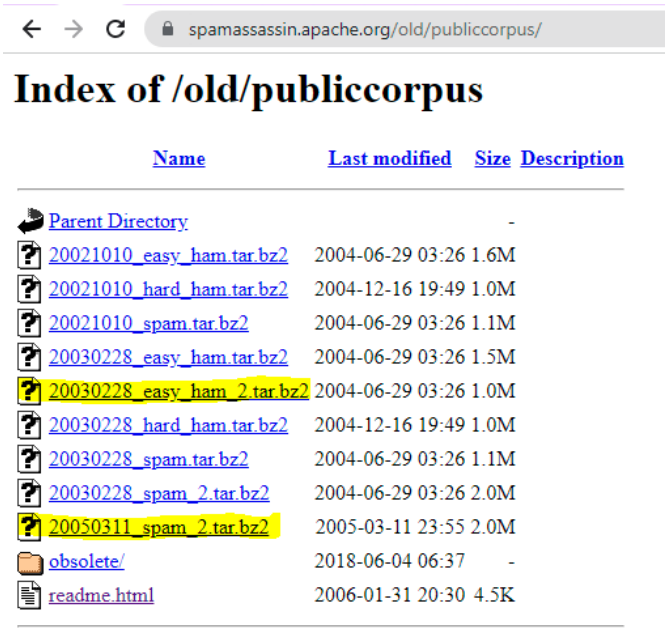
```
------------------------------------------------------------
This email is NEVER sent unsolicited.  THIS IS NOT "SPAM". You are receiving
this email because you EXPLICITLY signed yourself up to our list with our
online signup form or through use of our FFA Links Page and E-MailDOM
systems, which have EXPLICIT terms of use which state that through its use
you agree to receive our emailings.  You may also be a member of a Altra
Computer Systems list or one of many numerous FREE Marketing Services and as
such you agreed when you signed up for such list that you would also be
receiving this emailing.
Due to the above, this email message cannot be considered unsolicitated, or
spam.
------------------------------------------------------------
```

Figure 02: Exemple d'email data/spam

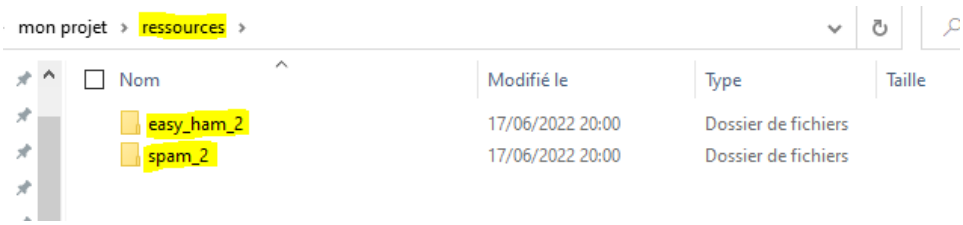
## Choix de données :

Nous avons utilisé les ensembles donnés ci-dessous :

[ ] 20030228_easy_ham_2.tar.bz2 2004-06-29 03:26 1.0M
[ ] 20050311_spam_2.tar.bz2 2005-03-11 23:55 2.0M




Pour des raisons de sécurité nous n'avons pas pu l'uploader sur Google Drive.
Pour tester notre code avec les mêmes conditions, vous pouvez les extraire, et les mettre dans le dossier « ressources »




Nous avons aussi supprimé le fichier « cmds » qui ne contient pas de mails.

## Importation des emails :

En utilisant la méthode import_emails_from_directory(path) nous avons parcouru l'ensemble des emails en faisant l'extraction de leur corps grâce à la librairie «email » .

## Nettoyage des emails :

Nous avons implémenté une fonction qui fait le prétraitement nécessaire du corps de chaque email, en utilisant des expressions régulières et le module « SnowballStemmer » pour la radicalisation

Cette fonction traite les notions suivantes :

**Minuscule :** l'intégralité de l'e-mail devra être convertie en minuscules.

**Suppression de balises HTML :** Toutes les balises HTML devront être supprimées des e-mails. De nombreux e-mails sont souvent accompagnés d'un formatage HTML ; toutes les Balises HTML devront être supprimées, de sorte que seul à garder uniquement le contenu de l'email.

**Normalisation des URL :** Toutes les URL devront être remplacées par le texte « httpaddr ».

**Normalisation des adresses e-mail :** toutes les adresses e-mail devront être remplacées avec le texte "emailaddr".

**Normalisation des nombres :** Tous les nombres devront être remplacés par le texte "nombre".

**Normalisation des dollars :** Tous les signes dollar ($)devront être remplacés par le texte "dollar".

**Radicalisation de mots :** Les mots devront être réduits à leur forme radicale. Par exemple, "discount", "discounts", "discounted" et "discounting" devront être tous remplacé par " discount", et "include", "includes", "included", et "ncluded" devront être tous remplacés par « includ ».

**Suppression des non-mots :** les non-mots et la ponctuation devront être supprimés. Tous les espaces blancs (onglets, nouvelles lignes, espaces) devront être remplacés par un seul espace.

## Organisation des caractéristiques

Dans cette étape nous avons combiné nos emails prétraités dans un seule nparray pour avoir l'ensemble de nos features,
On a aussi rempli notre tableau de labels en utilisant l'entier 1 pour indiquer si le mail est un spam et 0 pour les mails non-spam.

Par la suite nous avons diviser nos données en données d'entraînement et de test.

## Construction du vocabulaire

Vectorization en utilisant le module TfidfVectorizer :
Comme les données textuelles ne peuvent pas être utilisés directement, nous avons utilisé la Fonction de vectorisation "TF-IDF" présente dans la librairie scikit-learn. Cette technique nous permet de traduire les données textuelles sous formes de vecteur numérique représentant la fréquence d'apparition des mots par un indicateur de similarité (si ce mot est commun ou rare dans tous les documents).
Dans notre implémentation nous avons éliminé les mots qui se produisent rarement dans l'ensemble de nos données.
Cette fonction nous a permis aussi de filtrer les Stop-words (comme : “and”, “the”, “him” )

A la fin nous avons sauvegarder la liste de vocabulaire qu’on a généré dans un fichier texte « vocab.txt », ainsi que la table de nos données finales dans un fichier qu’on a nommé « data.csv »

# Étape 2 : Classification

Puisqu’on travaille avec des étiquettes de classe (spam, non-spam), il s'agit d'un problème d'apprentissage supervisé, par conséquent nous avons choisi les modèles d’apprentissage suivant :

**Naive Bayes**

Naive Bayes Classifier est un algorithme populaire en Machine Learning. C’est un algorithme du Supervised Learning utilisé pour la classification. Il est particulièrement utile pour les problématiques de classification de texte, adapté à notre problème

**SVM**
Ce sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de discrimination et de régression. Les SVM sont une généralisation des classifieurs linéaires.

Dans notre implémentation nous avons utilisé un SVM linéaire.

**Régression logistique**

La régression logistique est un modèle statistique permettant d'étudier les relations entre un ensemble de variables qualitatives Xi et une variable qualitative Y. Il s'agit d'un modèle linéaire généralisé utilisant une fonction logistique comme fonction de lien.

# Librairies Utilisé

Veuillez trouver ci-dessous, les librairies que nous avons utilisé dans notre projet:

| Librarie | Utilisation |
|---|---|
| **Scikit-learn** | Scikit-learn est une bibliothèque d'apprentissage automatique gratuite. Elle comporte divers algorithmes de classification, de régression et de clustering, y compris les machines à vecteurs, les random forests, les k-means… On a utilisé cette librairie pour l'entraînement des algorithmes utilisés et aussi dans la phase d'évaluation et comparaison de résultat obtenu. |
| email | email est une bibliothèque pour gérer les e-mails. On a utilisé le module « message_from_bytes » pour extraire le corps |
| **matplotlib** | Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques<br>On a l'utiliser pour tracer les courbes nécessaires pour analyser les résultat |
| NLTK | NLTK est une bibliothèque dédiée au Traitement Naturel du Langage, on a utilisé le module « SnowballStemmer » pour radicaliser les mots |
| **metrics** | Utilisé pour calculé l'accuracy et la courbe de roc |

# Analyse des résultats

Mesurer les performances : prédire les étiquettes de classe dans l'ensemble de données de test et compter le nombre de prédictions correctes pour évaluer la précision (accuracy) de la prédiction.

Le tableau suivant contient les résultats des accuracy obtenu en calculant la prédiction sur l'ensemble de d'entrainement et test pour chaque algorithme:

| Modèle | Régression logistique | Naive Bayes | SVM |
|---|---|---|---|
| Accuracy | 98.83% | 97.98% | 98.79% |

Table 01: Accuracy sur l'ensemble d'entrainement

| Modèle | Régression logistique | Naive Bayes | SVM |
|---|---|---|---|
| Accuracy | 97.50 | 96.96% | 97.67% |

Table 02: Accuracy sur l'ensemble de test

On remarque que les résultats sont relativement proches, et que SVM et la régression logistique ont de meilleurs résultats par rapport le Native Bayes.

# Projet réalisé par :

BOULARIACHE Abdessamed

KEMOUM Meroua

MAHIDDINE Mohamed Amine

TAZIR Mohamed Reda