

# Classification des documents

Analyse sentimentale des  
gens vis-à-vis de films

Projet FoDo

Réalisé par :

*Kamel MESSAOUDENE*

*Mohand Ameziane ZAIDI*



**01**

Dictionnaire de données

**02**

Importation de données et prétraitement  
de texte

**03**

Ensemble de formation et de test

**04**

Algorithme et évaluation de modèle

**05**

Résultats et répartition des tâches

# Dictionnaire de données

- Source: <http://www.cs.cornell.edu/people/pabo/movie-review-data>
- L'ensemble des données comprend un total de 2000 documents dont la moitié contient des critiques positives concernant un film tandis que la moitié restante contient des critiques négatives.

- *Importation de données:*

Dans le but d'exploiter l'ensemble de données qui se trouvent dans les différents fichiers nous avons utilisé la fonction `LOAD_FILES` de la Bibliothèque `SKLEARN_DATASETS`

```
movie_data = load_files(r"C:\txt_sentoken")  
X, y = movie_data.data, movie_data.target
```

La variable `X` contient l'ensemble des données (fichiers positifs et négatifs) tandis que les catégories cibles (0: négatif et 1: positif) sont stocké dans `Y`.

# Importation de données et prétraitements

## Nettoyage de données

5

### Nettoyage:

- Suppression des caractères uniques (début, milieu et fin)
- Suppression des caractères spéciaux...
- Ecrire en minuscule.
- Lemmatisation grâce a la fonction split.

Nous avons utilisé pour cela la bibliothèque python RE pour effectuer les diverses tâches de prétraitement de texte.

```
for sen in range(0, len(X)):  
    # Supprimer tous les caractères spéciaux  
    document = re.sub(r'\W', ' ', str(X[sen]))  
  
    # Supprimer tous les caractères uniques  
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)  
  
    # Supprimer les caractères uniques du début  
    document = re.sub(r'^\s+[a-zA-Z]\s+', ' ', document)  
  
    # Remplacement de plusieurs espaces par un seul espace  
    document = re.sub(r'\s+', ' ', document, flags=re.I)  
  
    # Suppression du préfixe «b»  
    document = re.sub(r'^b\s+', '', document)  
  
    # Conversion en minuscules  
    document = document.lower()  
  
    # Lemmatisation  
    document = document.split()  
  
    document = [stemmer.lemmatize(word) for word in document]  
    document = ' '.join(document)  
  
    documents.append(document)
```

# Importation de données et prétraitements

## Conversion de texte en nombre

6

### Conversion de texte en nombre:

Dans cette étape nous avons convertie le texte brut des fichiers en nombres en appliquant le modèle du **sac de mots**.

```
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))  
X = vectorizer.fit_transform(documents).toarray()
```

Dans le bout de code ci-dessus nous avons utilisé la fonction CountVectorizer qui prend les paramètres suivants:

**max\_features:** il représente le nombre souhaité des mots fréquents.

**min\_df:** le nombre de document dans lesquels les mots doivent être présents impérativement afin de les sélectionner.

**max\_df:** le pourcentage maximum de présence de mots afin de les sélectionner.

**stop\_words:** supprime les mots vides.

# Encoding « TF-IDF »

### Conversion de texte en nombre:

Tel que vu en cours le TF représente le « Terme Frequency », et l'IDF c'est la fréquence inverse d'un document « Inverse Document Frequency »

```
tfidfconverter = TfidfTransformer()  
X = tfidfconverter.fit_transform(X).toarray()
```

Afin de diviser nos données en ensemble de formation et de test nous avons utilisé la fonction **train\_test\_split** qui prend en argument:

Les données, les catégories, pourcentage de l'ensemble de test et le reste pour l'ensemble de formation.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```



## **Algorithme de forêt aléatoire**

A présent nous avons divisé nos données en ensemble de formation et de test. Pour entraîner notre modèle nous avons implémenté l'algorithme de forêt aléatoire.

```
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
```

Nous avons utilisé la fonction **RandomForestClassifier** de la bibliothèque **SKLEARN**. De plus on a utilisé la méthode **fit** afin d'entraîner l'algorithme.

Ensuite, on a prédit grâce à la fonction **predict** méthode de **RandomForestClassifier** tel que présenté ci-dessus.

## Evaluation du modèle

Dans le but d'évaluer la performance de notre modèle de classification on a utilisé la matrice de confusion, la mesure F1 et la précision.

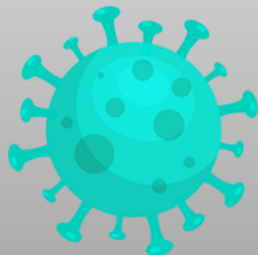
Afin de trouver ces valeurs sous python on a utilisé « `classification_report` », « `confusion_matrix` » et les « `accuracy_score` » services publics de la « `sklearn.metrics` ».

```
print(confusion_matrix(y_test,y_pred))  
  
print(classification_report(y_test,y_pred))  
print(accuracy_score(y_test, y_pred))
```

## Résultats obtenus :

```
PS C:\Users\kamel> cd .\OneDrive\Bureau\  
PS C:\Users\kamel\OneDrive\Bureau> cd .\project_python\  
PS C:\Users\kamel\OneDrive\Bureau\project_python> python main.py  
[nltk_data] Downloading package stopwords to  
[nltk_data]      C:\Users\kamel\AppData\Roaming\nltk_data...  
[nltk_data]   Package stopwords is already up-to-date!  
[[180  28]  
 [ 30 162]]  
  
          precision    recall  f1-score   support  
  
         0         0.86      0.87      0.86         208  
         1         0.85      0.84      0.85         192  
  
    accuracy                   0.85         400  
   macro avg         0.85      0.85      0.85         400  
weighted avg         0.85      0.85      0.85         400  
  
0.855  
PS C:\Users\kamel\OneDrive\Bureau\project_python>
```

## Difficultés rencontrées



Le confinement



Pas assez de  
temps



*J'ai sommeil !*

Charge de travail  
assez élevée

*Par la fenêtre*

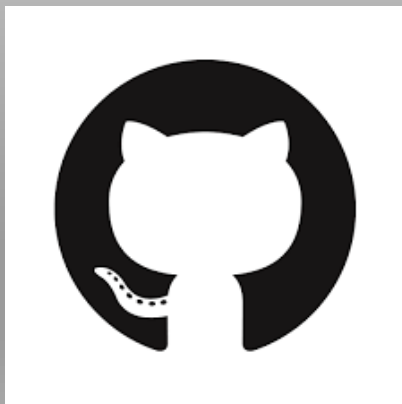


Problèmes d'installations  
(Windows, Linux)

## Code:

Lien GitHub :

<https://github.com/ZAIDIMDAMZ/documentsClassifier>



## **Contributions:**

- Idée du projet : Kamel MESSAOUDEN et Mohand Ameziane ZAIDI
- Code :
  - ☐ Partie “Choix de sujet” : Les deux en collaboration avec le prof.
  - ☐ Partie “Nettoyage et importation de données” : Kamel MESSAOUDENE
  - ☐ Partie “Algorithme de classification” : Mohand Ameziane ZAIDI.
  - ☐ Partie “Evaluation et visualisation” : Kamel MESSAOUDENE
  - ☐ Partie “Slides et rédaction” : Mohand Ameziane ZAIDI.
- Nettoyage des données : Kamel MESSAOUDENE
- Algo des forêts aléatoire : Mohand Ameziane ZAIDI
- Evaluation et vérification de travail : Kamel MESSAOUDENE
- Slides : Mohand Ameziane ZAIDI