REPUBLIQUE DU CAMEROUN
Paix-Travail-Patrie
****
UNIVERSITE DE YAOUNDE I
****
ECOLE NATIONALE SUPERIEUR POLYTECHNIQUE DE YAOUNDE
****
DEPARTEMENT DU GENIE INFORMATIQUE

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
****
UNIVERSITY OF YAOUNDE I
****
NATIONAL ADVANCED SCHOOL OF ENGINEERING OF YAOUNDE
****
DEPARTEMENT OF COMPUTER SCIENCE ENGINEERING

AGENTS INTELLIGENTS

Rapport de Projet : Détection des Langues

Etudiant
NGUAZONG AUREL 20P001

Sous la supervision : M. Bitha

Année académique :
2023/2024

Table des matières

INTRODUCTION 3
I. Méthodologie de conception 4
1. Importation et exploitation de la dataset 4
2. Analyse univariée 4
3. Prétraitement des données 5
4. Séparation des données d'entraînement et de test 6
5. Vectorisation des données 7
6. Entraînement par régression logistique 7
7. Evaluation sur les données test 8
8. Démonstration 9
II. Remarques et commentaires 10

INTRODUCTION

Intitulé du devoir :

Projet AGENTS INTELLIGENTS
A partir du jeu de données Languages basé sur la reconnaissance de la langue en fichier joint, il vous est demandé de mettre en place une solution capable de pouvoir reconnaitre la langue utilisée lors de la saisie d'un texte. Le jeu de données présente plusieurs langues parmi lesquelles :
1) Anglais
2) Malayalam
3) Hindi
4) Tamoul
5) Kannada
6) Français
7) Espagnol
8) Portugais
9) Italien
10) Russe
11) Suédois
12) Néerlandais
13) Arabe
14) Turc
15)Allemand
16) Danois
17) grec

Vous utiliserez à votre convenance des algorithmes d'apprentissage automatique ou en profondeur pour la résolution de la problématique susmentionnée.

I. Méthodologie de conception

1. Importation et exploitation de la dataset

La dataset est importée via pandas et l'on vérifie ces paramètres.

```
# Chargement du fichier
df = pd.read_csv('Languages.csv')
df.head()
```

	Text	Language
0	Nature, in the broadest sense, is the natural...	English
1	"Nature" can refer to the phenomena of the phy...	English
2	The study of nature is a large, if not the onl...	English
3	Although humans are part of nature, human acti...	English
4	[1] The word nature is borrowed from the Old F...	English

Par la suite l'on exploite ces données pour pouvoir mieux les manipuler via un processus simple :

- L'on vérifie les lignes dupliquées

```
#Vérification de valeurs duppliquées
df.duplicated().sum()
```

```
66
```

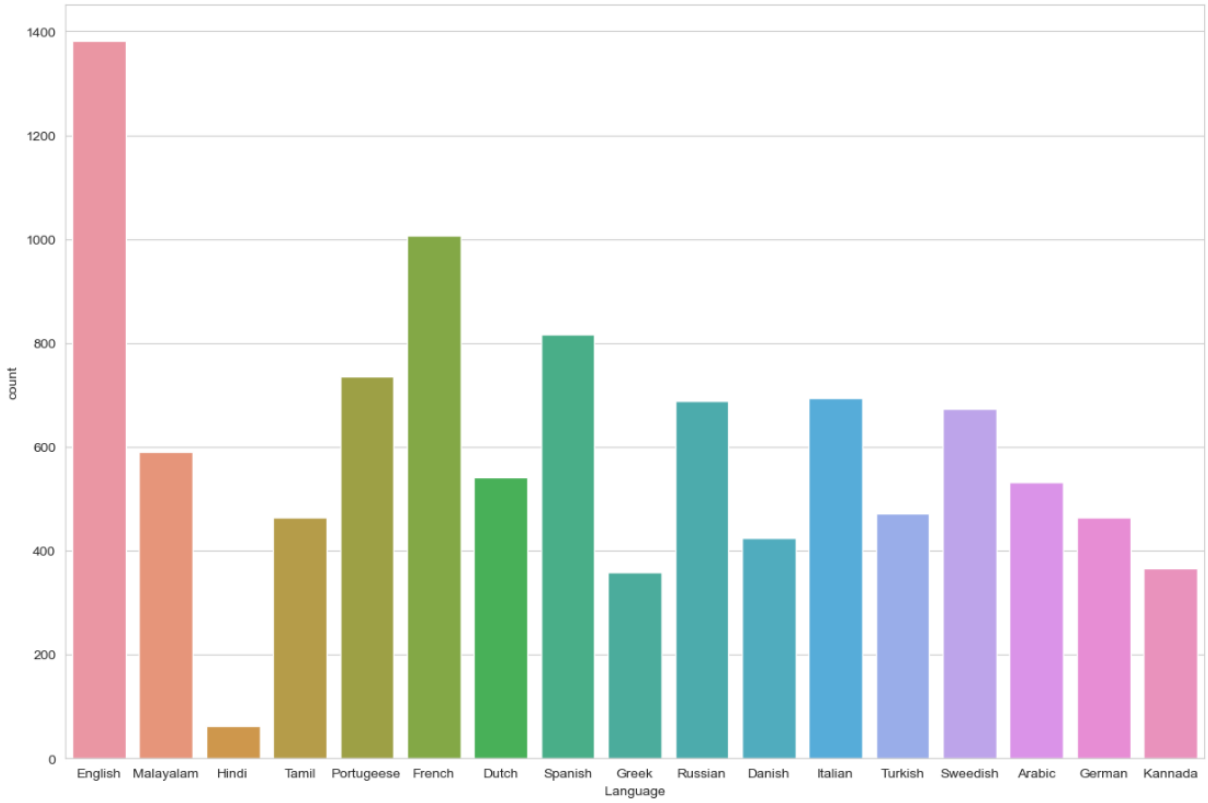
- Puis l'on supprime ces valeurs

```
#Suppression des valeurs duppliquées
df.drop_duplicates(keep='first', inplace=True)
df.duplicated().sum()
```

```
0
```

2. Analyse univariée

Concernant l'analyse univariée de la dataset, l'on remarque que les langues dominantes tel que l'anglais, le français ou l'espagnol auront plus de chance d'être prédite à cause du nombre extrême de features dans la dataset, contrairement à l'Hindi ou le grec qui seront moins performant lors de la prédiction.



3. Prétraitement des données

Après l'analyse, viens le prétraitement de données où l'on doit pouvoir :

- Trier les langues en ordre alphabétique :

```
#Trier les données en ordre croissant en fonction de Language
df_trier = df.sort_values(by=["Language"], ascending = True)
df_trier
```

	Text	Language
9417	كنت أفضل لو لم تفعل.	Arabic
9165	كما إستوحت موسوعة لاروس طريقة عمل ويكيبيديا لإنشاء موسوعة على الإنترنت يحررها مساهمون غير متخصصين، لكن مع اختلاف في التطبيق فموسوعة لاروس تسمح لمستخدمها أن يكتب مقالاً ويحق له وحده أن يحرره، وهذا النموذج يتفق مع ما تطبقه خدمة نول من جوجل.	Arabic
9164	مشاريع أخرى مستقلة عن مؤسسة ويكيميديا استوحت الفكرة من طريقة عمل ويكيبيديا التي تعتمد على نظام الويكي والعمل الجماعي، من هذه المشاريع، موسوعة المعرفة، وموسوعة الحياة.	Arabic
9163	أحدث مشاريع ويكيميديا هو ويكي جامعة مشروع يهدف إلى دعم التعليم الحر واستضافة مصادر تعليمية مجانية [111].	Arabic
9162	ويكيميديا تطلق مشاريع عديدة منذ ذلك الحين.	Arabic
...	...	...
8219	yeterince teşekkür edemem Mühim değil.	Turkish
8220	bahsetme.	Turkish
8221	istediğin zaman.	Turkish
8209	biraz dondurmaya ne dersin?	Turkish
8101	ben çuvalladım ben mahvettim eğer benim yerime birinin çalışmasını istiyorsan, diyebilirsin.	Turkish

10271 rows × 2 columns

- Supprimer les caractères spéciaux, les liens, les chiffres et mettre tout le texte en minuscule :

```python
def trait_text(text):

    #Removes unicode strings like "\u002c"
    text = re.sub(r'(\\u[0-9A-Fa-f]+)',r'', text)

    #Convert to lowercase
    text = text.lower()

    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub('_;__', '', text)

    #remove tags
    text=re.sub("&lt;/?.*?&gt;"," ",text)

    # remove special characters and digits
    text=re.sub("(\\d|\\W)+"," ",text)


    return text
```

- Labéliser les langues pour l'entraînement de prédiction

```python
#On labelise les langues pour mieux manipuler

from sklearn.preprocessing import LabelEncoder

# Vos données de langues
langues = df_trier['Language']

# Créer une instance de LabelEncoder
label_encoder = LabelEncoder()

# Adapter et transformer les données de langues
langues_labelisees = label_encoder.fit_transform(langues)

# Afficher les résultats
for langue, label in zip(langues, langues_labelisees):
    print(f'Langue : {langue} | Label : {label}')
```

4. Séparation des données d'entraînement et de test

Pour l'entraînement, l'on sépare les données d'entraînement de test selon le ratio 8:2.

```
# Separe feature and target form data
X = df_trier.loc[:, 'Cleaned_Text']
y = df_trier.loc[:, 'Language_labelise']

print(f"Shape of X: {X.shape}\nshape of y: {y.shape}")
```

```
Shape of X: (10271,)
shape of y: (10271,)
```

```
# Split in train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=11, stratify=y)
print(f"Train Data Shape: {X_train.shape}\nTest Data Shape: {X_test.shape}")
```

```
Train Data Shape: (8216,)
Test Data Shape: (2055,)
```

5. Vectorisation des données

Pour la vectorisation, l'on doit :

- Dans un premier temps faire un countVectorizer pour dénombrer le vocabulaire à vectoriser

```
# Computer the vocabulary
cVect = CountVectorizer()
cVect.fit(X_train)

# Let's see the vocabulary that has extracted by the count vextorizer
print('NO.of Tokens: ',len(cVect.vocabulary_.keys()))
```

```
NO.of Tokens:  34411
```

- Ensuite vectoriser chaque terme du vocabulaire

```
# document term vector (dtv)
dtv = cVect.transform(X_train)
print(dtv)
```

- Enfin transformer le tout en tableau pour qu'ils soient viables :

```
dtv = dtv.toarray()
print(f"Number of Observations: {dtv.shape[0]}\nTokens/Features: {dtv.shape[1]}")
```

```
Number of Observations: 8216
Tokens/Features: 34411
```

6. Entraînement par régression logistique

```
%%time
lr = LogisticRegression()
lr.fit(dtv, y_train)
```

```
CPU times: total: 2min 10s
Wall time: 57.1 s
```

7. Evaluation sur les données test

```
# Preprocess the test data
test_dtv = cVect.transform(X_test)
test_dtv = test_dtv.toarray()
print(f"Number of Observations: {test_dtv.shape[0]}\nTokens/Features: {test_dtv.shape[1]}")
```

```
Number of Observations: 2055
Tokens/Features: 34411
```

```
%%time
pred = lr.predict(test_dtv)
```

```
CPU times: total: 797 ms
Wall time: 800 ms
```

```
print('Accuracy: ', accuracy_score(y_test, pred) * 100)
```

```
Accuracy:  95.42579075425792
```

L'on remarque dans cette partie que le score au test est de 95,42% ce qui représente un bon score et ne nécessite pas forcement d'optimisation.

Concernant la précision pour chaque langue, elle varie assez mais reste stable avec une moyenne de 95%.

```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       106
           1       0.95      0.92      0.93        85
           2       1.00      0.90      0.95       108
           3       0.99      0.97      0.98       277
           4       0.98      0.97      0.97       202
           5       1.00      0.98      0.99        93
           6       1.00      0.96      0.98        72
           7       1.00      0.83      0.91        12
           8       0.95      0.98      0.96       139
           9       1.00      0.93      0.96        73
          10       1.00      0.97      0.99       118
          11       0.99      0.94      0.97       147
          12       0.70      0.99      0.82       138
          13       0.92      0.94      0.93       163
          14       0.98      0.93      0.95       135
          15       1.00      0.97      0.98        93
          16       0.99      0.94      0.96        94

    accuracy                           0.95      2055
   macro avg       0.97      0.94      0.95      2055
weighted avg       0.96      0.95      0.96      2055
```

8. Démonstration

Cette démo est juste la preuve que le code marche pour la plupart des langues et peut servir comme modèle de reconnaissance.

```
def modele (x):
    text = trait_text(x)
    text = [text]
    t_dtv = cVect.transform(text).toarray()

    pred = lr.predict(t_dtv)

    return pred
```

```
def predict():
    lang = sorted(df["Language"].unique())
    text = input('Enter text: ')
    pred = modele(text)
    y = pred[0]
    print('Ce texte est écrit en : ' + lang[y] )
```

```
predict()
```

```
Enter text: I have a money
Ce texte est écrit en : English
```

II. Remarques et commentaires

Lors de la conception de ce modèle l'on a rencontré un certain nombre de problème et de remarque :

- Était-il nécessaire de supprimer les stopwords ? : J'ai jugé cela pas nécessaire à cause de la réflexion qui dit que même un stopword peut être une indication à la langue

- Était-il nécessaire d'optimiser les hyperparamètres du modèle ? J'ai jugé pas nécessaire à cause des 95% de précision.