NLP_Lab-6_205229118_Mahalakshmi.S

April 27, 2021

0.0.1 Lab6. Spam Filtering using Multinomial NB

In this lab, you will build Naïve Bayes classifier using SMS data to classify a SMS into spam or not. Once the model is built, it can be used to classify an unknown SMS into spam or ham.

0.0.2 STEPS

0.0.3 1. Open "SMSSpamCollection" file and load into DataFrame. It contains two columns "label" and "text"

[1]:
```
import pandas as pd
from nltk.corpus import stopwords
```

[2]:
```
df = pd.read_csv("SMSSpamCollection.csv",encoding='ISO-8859-1')
```

[3]:
```
spam=df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
spam
```

[3]:
```
     label                                               text
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…
…    …                                                   …
5567  spam  This is the 2nd time we have tried 2 contact u…
5568   ham              Will Ì_ b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So…any other s…
5570   ham  The guy did some bitching but I acted like i'd…
5571   ham                        Rofl. Its true to its name

[5572 rows x 2 columns]
```

0.0.4 2. How many sms messages are there?

[4]:
```
len(spam)
```

[4]:
```
5572
```

0.0.5 3. How many "ham" and "spam" messages?. You need to groupby() label column.

[5]:
```
lab=spam.groupby('label').count()
lab
```

[5]:
```
       text
label
ham    4825
spam    747
```

0.0.6 4. Split the dataset into training set and test set (Use 20% of data for testing).

[6]:
```
X = spam.text
y = spam.label
```

[7]:
```
from sklearn.model_selection import train_test_split
```

[8]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.8,test_size=0.
↪2)
```

0.0.7 5. Create a function that will remove all punctuation characters and stop words, as below

[9]:
```
def process_text(msg):
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
    nopunc =[char for char in msg if char not in punctuations]
    nopunc=''.join(nopunc)
    return [word for word in nopunc.split()
    if word.lower() not in stopwords.words('english')]
```

0.0.8 6. Create TfIdfVectorizer as below and perform vectorization on X_train, using fit_perform() method.

[10]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[11]:
```
tfv=TfidfVectorizer(use_idf=True,analyzer=process_text,ngram_range=(1,3),min_df␣
↪= 1,stop_words = 'english')
tfv
```

[11]:
```
TfidfVectorizer(analyzer=<function process_text at 0x00000180F9A2D8B0>,
                ngram_range=(1, 3), stop_words='english')
```

[17]:
```
t1=tfv.fit_transform(X_train)
tf2=tfv.transform(X_test)
```

[18]:
```
t1.shape
```

[18]: (4457, 9895)

[19]:
```
tf2.shape
```

[19]: (1115, 9895)

0.0.9 7. Create MultinomialNB model and perform training on X_train and y_train using fit() method

[20]:
```
x_train,x_test,Y_train,Y_test = train_test_split(X,y,train_size=0.8,test_size=0.
↪2)
```

[21]:
```
from sklearn.naive_bayes import MultinomialNB
```

[22]:
```
clf = MultinomialNB()
```

[23]:
```
clf.fit(t1,y_train)
```

[23]: MultinomialNB()

0.0.10 8. Predict labels on the test set, using predict() method

[24]:
```
y_predict = clf.predict(tf2)
y_predict
```

[24]: array(['ham', 'ham', 'ham', …, 'spam', 'ham', 'spam'], dtype='<U4')

0.0.11 9. Print confusion_matrix and classification_report

[25]:
```
from sklearn.metrics import confusion_matrix
```

[26]:
```
confusion_matrix(y_test,y_predict)
```

[26]:
```
array([[969,   0],
       [ 51,  95]], dtype=int64)
```

[27]:
```
from sklearn.metrics import classification_report
```

[29]:
```
target_names = ['class 0', 'class 1']
print(classification_report(y_test,y_predict,target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.95      1.00      0.97       969
     class 1       1.00      0.65      0.79       146

    accuracy                           0.95      1115
   macro avg       0.97      0.83      0.88      1115
```

```
weighted avg       0.96      0.95      0.95      1115
```

0.0.12 10. Modify ngram_range=(1,2) and perform Steps 7 to 9.

[30]:
```
tf_2=TfidfVectorizer(use_idf=True,analyzer=process_text,ngram_range=(1,2),min_df
 ↪= 1,stop_words = 'english')
tf_2
```

[30]:
```
TfidfVectorizer(analyzer=<function process_text at 0x00000180F9A2D8B0>,
                ngram_range=(1, 2), stop_words='english')
```

[31]:
```
t3=tf_2.fit_transform(X_train)
tf_ng=tf_2.transform(X_test)
```

[32]:
```
t3.shape
```

[32]:
```
(4457, 9895)
```

[33]:
```
tf_ng.shape
```

[33]:
```
(1115, 9895)
```

[35]:
```
clf.fit(t3,y_train)
```

[35]:
```
MultinomialNB()
```

[36]:
```
y_predict2 = clf.predict(tf_ng)
y_predict2
```

[36]:
```
array(['ham', 'ham', 'ham', …, 'spam', 'ham', 'spam'], dtype='<U4')
```

[37]:
```
confusion_matrix(y_test,y_predict2)
```

[37]:
```
array([[969,   0],
       [ 51,  95]], dtype=int64)
```

[38]:
```
target_names = ['class 0', 'class 1']
print(classification_report(y_test,y_predict2,target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.95      1.00      0.97       969
     class 1       1.00      0.65      0.79       146

    accuracy                           0.95      1115
   macro avg       0.97      0.83      0.88      1115
weighted avg       0.96      0.95      0.95      1115
```