ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

**EXPERIMENT 13: Implementation of NLP problem**

**NAME:** Praveen Kumar K

**REG NO:** RA1911030010069

## **AIM:** To Implement NLP programs.

**LANGUAGE:** Python

# THEORY:

NLP stands for **Natural Language Processing**, which is a part of **Computer Science, Human language,** and **Artificial Intelligence**. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as **translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction,** and **topic segmentation**.

# CODE:

**import** pandas **as** pd **import** sqlite3 **import** regex **as** re

**import** matplotlib.pyplot **as** plt

**from** wordcloud **import** WordCloud

df **=** pd**.**read\_csv('emails.csv') df**.**head()

print("spam count: " **+**str(len(df**.**loc[df**.**spam**==**1]))) print("not spam count: " **+**str(len(df**.**loc[df**.**spam**==**0]))) print(df**.**shape)

df['spam'] **=** df['spam']**.**astype(int)

df **=** df**.**drop\_duplicates() print(df**.**shape)

df **=** df**.**reset\_index(inplace **= False**)[['text','spam']]

print(df**.**shape) df['spam']**.**unique() df**.**head()

clean\_desc **=** []

**for** w **in** range(len(df**.**text)): desc **=** df['text'][w]**.**lower()

*#remove punctuation*

desc **=** re**.**sub('[^a-zA-Z]', ' ', desc)

*#remove tags*

desc**=**re**.**sub("&lt;/?.\*?&gt;"," &lt;&gt; ",desc)

*#remove digits and special chars*

desc**=**re**.**sub("(\\d|\\W)+"," ",desc) clean\_desc**.**append(desc)

*#assign the cleaned descriptions to the data frame*

df['text'] **=** clean\_desc df **=** df**.**reset\_index() df**.**head(3)

df1 **=**df**.**loc[df**.**spam**==**0] df2 **=**df**.**loc[df**.**spam**==**1]

stop\_words **=** ['is','you','your','and', 'the', 'to', 'from', 'or', 'I',

'for', 'do', 'get', 'not', 'here', 'in', 'im', 'have', 'on', 're', 'new', 'subject']

*#set the word cloud parameters*

wordcloud **=** WordCloud(width **=** 800, height **=** 800, background\_color **=**

'black', stopwords **=** stop\_words, max\_words **=** 1000

, min\_font\_size **=** 20)**.**generate(str(df['text']))

*#plot the word cloud*

fig **=** plt**.**figure(figsize **=** (8,8), facecolor **= None**) plt**.**imshow(wordcloud)

plt**.**axis('off') plt**.**show()

wordcloud **=** WordCloud(width **=** 800, height **=** 800, background\_color **=**

'black', stopwords **=** stop\_words, max\_words **=** 1000

, min\_font\_size **=** 20)**.**generate(str(df2['text']))

*#plot the word cloud*

fig **=** plt**.**figure(figsize **=** (8,8), facecolor **= None**) plt**.**imshow(wordcloud)

plt**.**axis('off') plt**.**show()

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn **import** ensemble

**from** sklearn.metrics **import** classification\_report, accuracy\_score

*#list of sentences*

text **=** ["the dog is white", "the cat is black", "the cat and the dog are friends"]

*#instantiate the class*

cv **=** CountVectorizer()

*# tokenize and build vocab*

cv**.**fit(text)

*# summarize*

print(cv**.**vocabulary\_)

*# encode document*

vector **=** cv**.**transform(text)

*# summarize encoded vector*

print(vector**.**toarray())

**from** sklearn.feature\_extraction.text **import** CountVectorizer text\_vec **=** CountVectorizer()**.**fit\_transform(df['text'])

**from** sklearn.model\_selection **import** train\_test\_split X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(text\_vec, df['spam'], test\_size **=** 0.45

, random\_state **=**

42, shuffle **= True**)

**from** sklearn **import** ensemble

classifier **=** ensemble**.**GradientBoostingClassifier(

n\_estimators **=** 100, *#how many decision trees to build*

learning\_rate **=** 0.5, *#controls rate at which additional decision trees influes overall prediction*

max\_depth **=** 6,

*# min\_samples\_split = 21, # min\_samples\_leaf = 19,*

*#max\_features = 0.9, #loss = 'huber'*

)

classifier**.**fit(X\_train, y\_train) predictions **=** classifier**.**predict(X\_test)

print(classification\_report(y\_test, predictions))

**from** sklearn.metrics **import** classification\_report,confusion\_matrix, accuracy\_score

pred **=** classifier**.**predict(X\_train) print(classification\_report(y\_train ,pred )) print('Confusion Matrix: \n',confusion\_matrix(y\_train,pred)) print()

print('Accuracy: ', accuracy\_score(y\_train,pred))

pred **=** classifier**.**predict(X\_test) print(classification\_report(y\_test ,pred )) print('Confusion Matrix: \n', confusion\_matrix(y\_test,pred))

print()

print('Accuracy: ', accuracy\_score(y\_test,pred))

**from** textblob **import** TextBlob

*#load the descriptions into textblob*

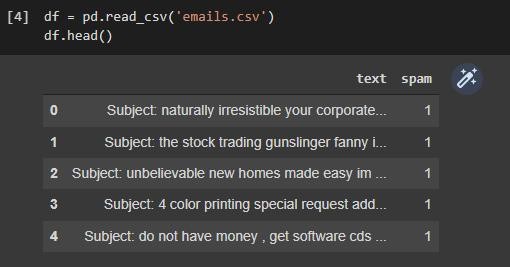
email\_blob **=** [TextBlob(text) **for** text **in** df['text']]

*#add the sentiment metrics to the dataframe*

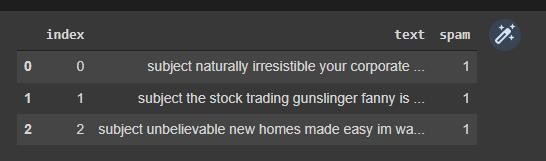
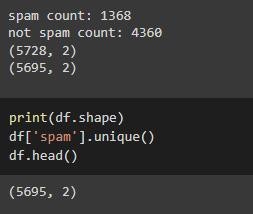
df['tb\_Pol'] **=** [b**.**sentiment**.**polarity **for** b **in** email\_blob] df['tb\_Subj'] **=** [b**.**sentiment**.**subjectivity **for** b **in** email\_blob] *#show dataframe*

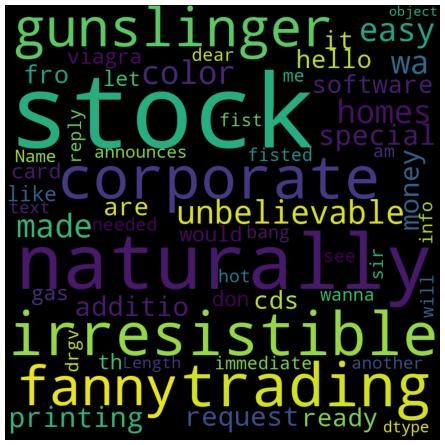
df**.**head(3)

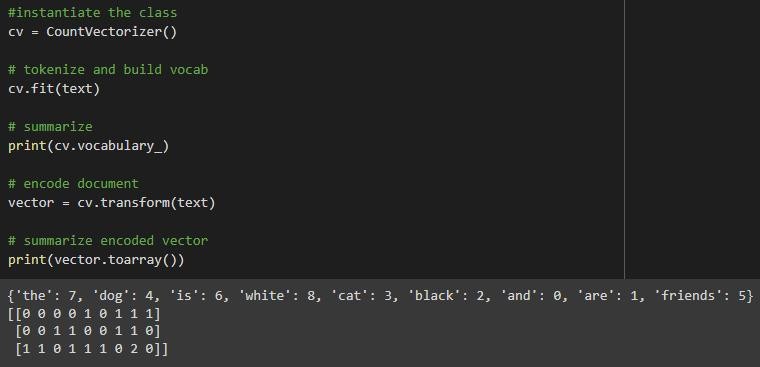
# OUTPUT:

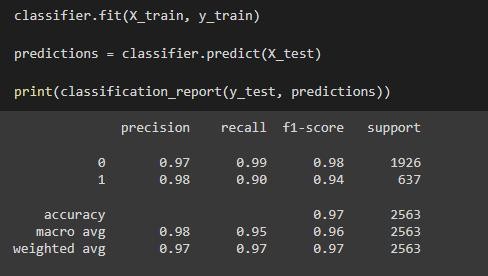
****

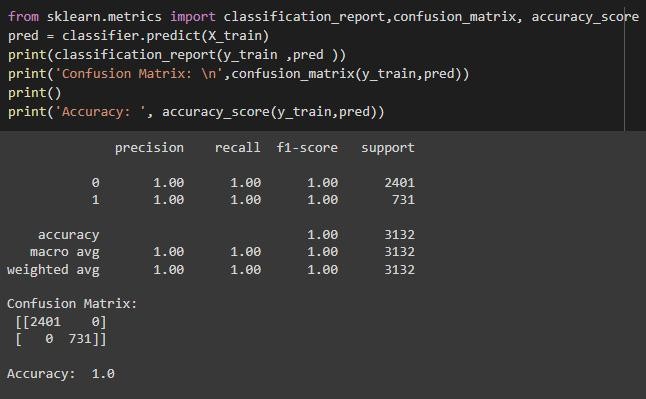


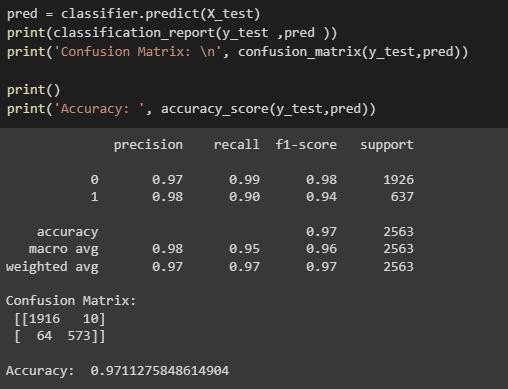


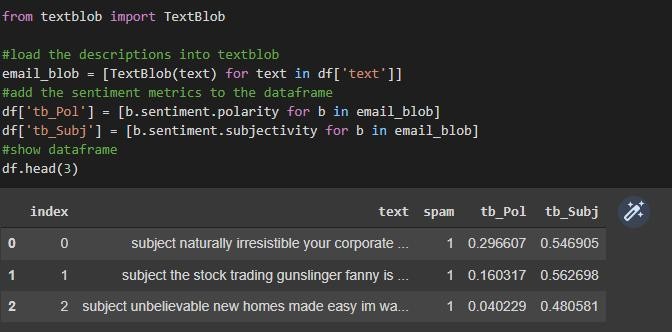


****



****



****

## **RESULT:** Thus, successfully implemented NLP problem.