Assignment 4

▾ 1. Download the dataset link

- Label - Ham or Spam
- Message - Message

1. NAME: DHIVYAA K S
2. REG NO:611219106017
3. DATE:1.11.22

1.NAME:DHIVYAA 2.REGISTER NUMBER:611219106017 3.DATE:01.11.2022

```
import warnings
warnings.filterwarnings("ignore")
```

▾ 2. Importing Required Library

```
import re
import nltk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from wordcloud import WordCloud,STOPWORDS,ImageColorGenerator
```

▾ 3. Read dataset and do Preprocessing

```
df = pd.read_csv("/content/spam.csv",encoding='ISO-8859-1')


df = df.iloc[:,:2]
df.columns=['label','message']
df.head()
```

	label	message
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...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   label    5572 non-null   object
 1   message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
ms1 = pd.Series((df.loc[df['label']=='ham','message']).tolist()).astype(str)
wordcloud = WordCloud(stopwords=STOPWORDS,width=800,height=600,background_color='black').gene
plt.figure(figsize=(20,10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 799.5, 599.5, -0.5)
```



```
ms2 = pd.Series((df.loc[df['label']=='spam','message']).tolist()).astype(str)
wordcloud = WordCloud(stopwords=STOPWORDS,width=1000,height=400,background_color='black').ger
plt.figure(figsize=(20,10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-0.5, 999.5, 399.5, -0.5)
```



```
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
corpus = []


import nltk
from nltk.corpus import stopwords
nltk.download('all')


for i in range(len(df)):
    review = re.sub('[^a-zA-Z]',' ',df['message'][i])
    review = review.lower()
    review = review.split()
```

```
review = [lemmatizer.lemmatize(i) for i in review if not i in set(stopwords.words('englis
review = ' '.join(review)
corpus.append(review)
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Package abc is already up-to-date!
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Package alpino is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |     to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger_ru is already
[nltk_data]    |     up-to-date!
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package basque_grammars is already up-to-date!
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package biocreative_ppi is already up-to-date!
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package bllip_wsj_no_aux is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Package brown is already up-to-date!
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Package brown_tei is already up-to-date!
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Package cess_cat is already up-to-date!
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Package cess_esp is already up-to-date!
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Package chat80 is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package comparative_sentences is already up-to-
[nltk_data]    |     date!
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    |   Package comtrans is already up-to-date!
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Package conll2000 is already up-to-date!
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Package conll2002 is already up-to-date!
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
```

▾ 4. Create Model

```python
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.layers import Dense,Dropout,LSTM,Embedding
from keras.models import Sequential,load_model


token = Tokenizer()
token.fit_on_texts(corpus)
text_to_seq = token.texts_to_sequences(corpus)


max_length_sequence = max([len(i) for i in text_to_seq])
padded_seq = pad_sequences(text_to_seq, maxlen=max_length_sequence, padding="pre")


padded_seq
```

```
array([[   0,    0,    0, ...,   16, 3551,   70],
       [   0,    0,    0, ...,  359,    1, 1610],
       [   0,    0,    0, ...,  218,   29,  293],
       ...,
       [   0,    0,    0, ..., 7042, 1095, 3547],
       [   0,    0,    0, ...,  842,    1,   10],
       [   0,    0,    0, ..., 2198,  347,  152]], dtype=int32)
```

```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(df['label'])


from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(padded_seq,y,test_size=0.25,random_state=42)


X_train.shape
```

```
(4179, 77)
```

▾ 5. Add Layers

```
TOT_SIZE = len(token.word_index) + 1
model = Sequential()
#IP Layer
model.add(Embedding(TOT_SIZE,32,input_length=max_length_sequence))
model.add(LSTM(units=50, activation = 'relu',return_sequences=True))
model.add(Dropout(0.2))
#Layer2
model.add(LSTM(units=60, activation = 'relu'))
model.add(Dropout(0.3))
#output layer
model.add(Dense(units=1, activation='sigmoid'))
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the crit
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the cr
```
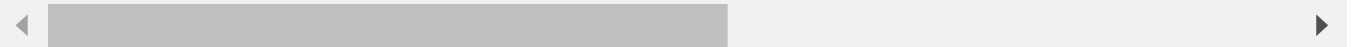
```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 77, 32)	225408
lstm (LSTM)	(None, 77, 50)	16600
dropout (Dropout)	(None, 77, 50)	0
lstm_1 (LSTM)	(None, 60)	26640
dropout_1 (Dropout)	(None, 60)	0
dense (Dense)	(None, 1)	61

```
=================================================================
Total params: 268,709
Trainable params: 268,709
Non-trainable params: 0
```

6 Compile the model

```
model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])
```

7 Fit the model

```
model.fit(X_train, y_train,validation_data=(X_test,y_test), epochs=10)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-26-4461f1aa2b5f> in <module>
----> 1 model.fit(X_train, y_train,validation_data=(X_test,y_test), epochs=10)

NameError: name 'X_train' is not defined
```

SEARCH STACK OVERFLOW

```
model.evaluate(X_test,y_test)
```

```
44/44 [==============================] - 1s 24ms/step - loss: 0.1467 - accuracy: 0.9813
[0.14665858447551727, 0.9813352227210999]
```

▾ 8. Save the Model

```
from pickle import dump,load
tfid = 'tfid.sav'
lstm = 'lstm.sav'
```

```
dump(token,open(tfid,'wb'))
model.save('nlp.h5')
```

▾ 9. Test the Model

```
def preprocess(raw_mess):
    review = re.sub('[^a-zA-Z]',' ',raw_mess)
    review = review.lower()
    review = review.split()
    review = [lemmatizer.lemmatize(i) for i in review if not i in set(stopwords.words('englis
    review = ' '.join(review)
    return review
```

```
def predict(mess):
    vect = load(open(tfid,'rb'))
    classifier = load_model('nlp.h5')
    clean = preprocess(mess)
    text_to_seq = token.texts_to_sequences([mess])
    padded_seq = pad_sequences(text_to_seq, maxlen=77, padding="pre")
```

```python
    pred = classifier.predict(padded_seq)
    return pred



msg = input("Enter a message: ")
predi = predict(msg)
if predi >= 0.6:
    print("It is a spam")
else:
    print("Not a spam")
```

```
    Enter a message: I HAVE A DATE ON SUNDAY WITH WILL!!
    -------------------------------------------------------------------------
    FileNotFoundError                         Traceback (most recent call last)
    <ipython-input-25-ea4400168dfd> in <module>
          1 msg = input("Enter a message: ")
    ----> 2 predi = predict(msg)
          3 if predi >= 0.6:
          4     print("It is a spam")
          5 else:

    <ipython-input-21-7f1c0840f83c> in predict(mess)
          1 def predict(mess):
    ----> 2     vect = load(open(tfid,'rb'))
          3     classifier = load_model('nlp.h5')
          4     clean = preprocess(mess)
          5     text_to_seq = token.texts_to_sequences([mess])

    FileNotFoundError: [Errno 2] No such file or directory: 'tfid.sav'
```

> SEARCH STACK OVERFLOW

```python
msg = input("Enter a message: ")
predi = predict(msg)
if predi >= 0.6:
    print("It is a spam")
else:
    print("Not a spam")
```