Assignment-4

Reg.No: 611219106303

Name: Mohan R

Date: 01/11/2022

1. Download the dataset link

1. Label - Ham or Spam
2. Message - Message

```
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()
```

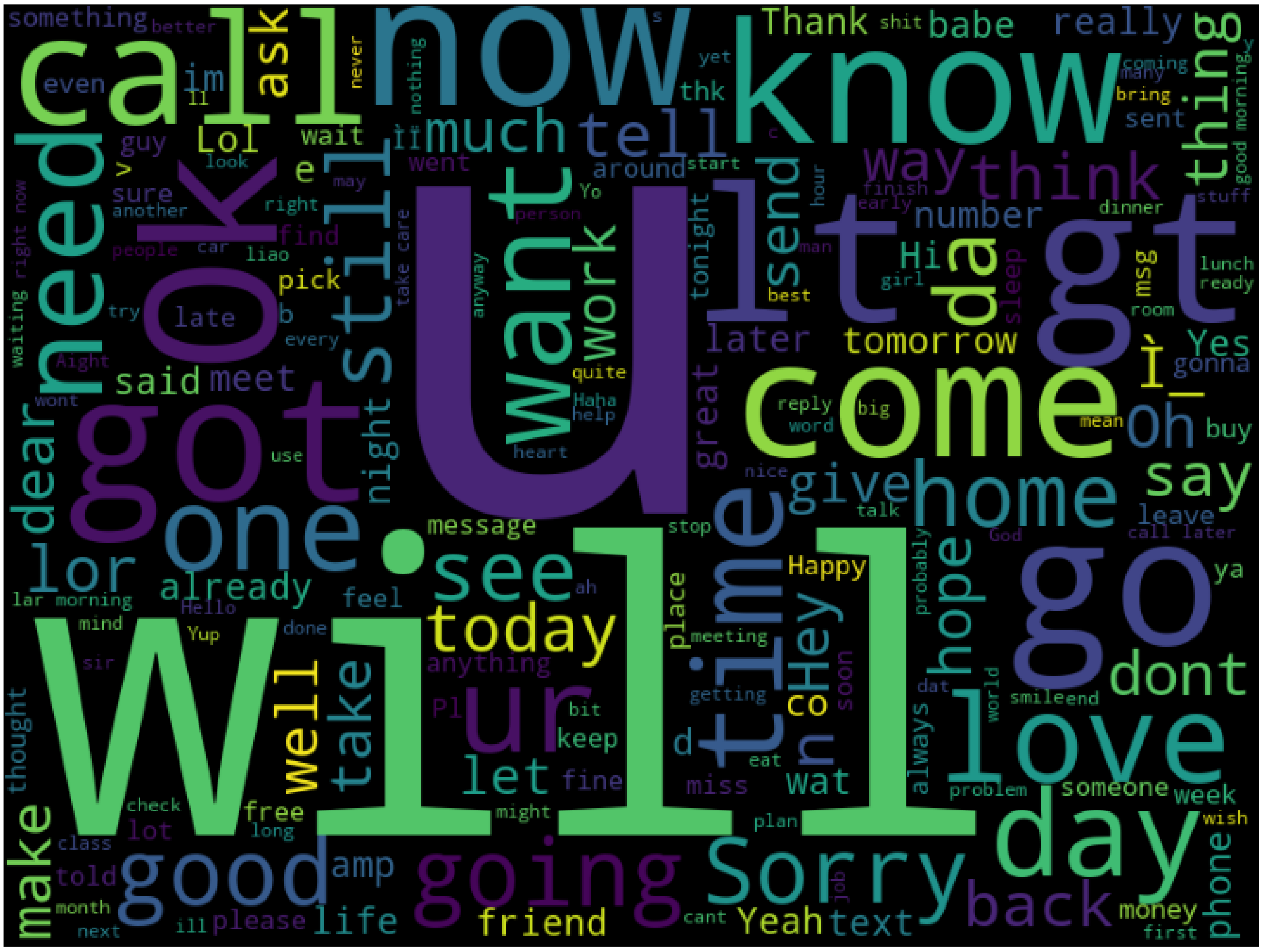
```
label                                   message
```

```
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
```

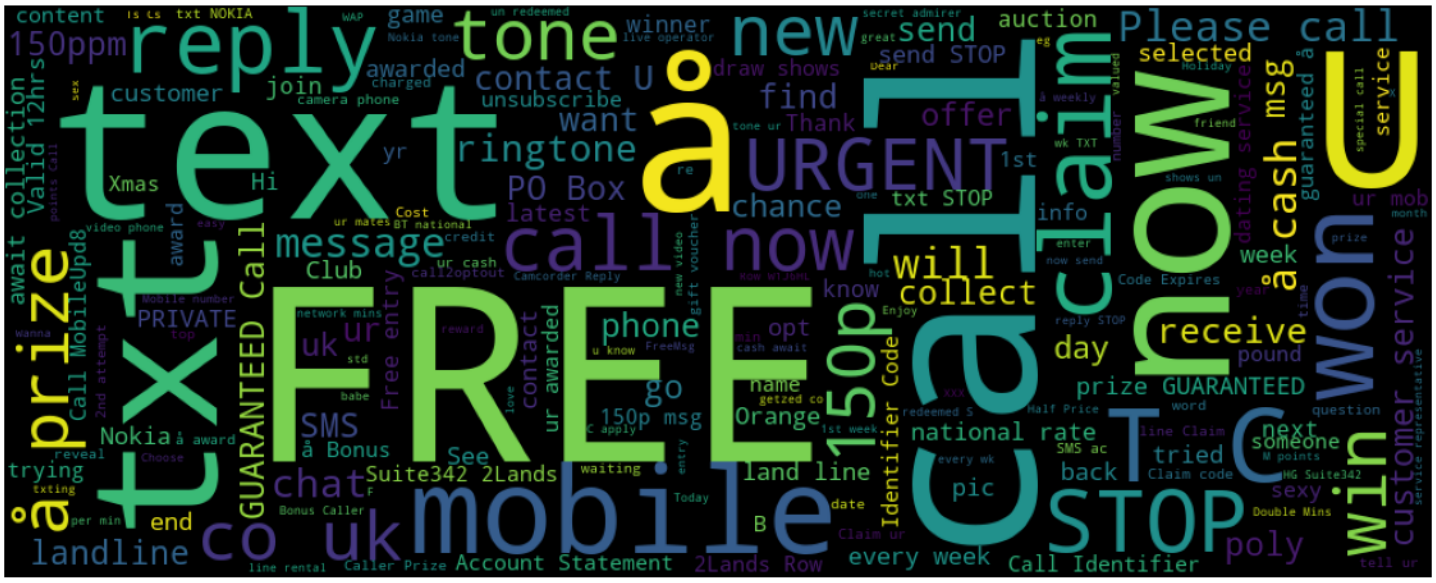
```
ms1 = pd.Series((df.loc[df['label']=='ham','message']).tolist()).astype(str)
wordcloud = WordCloud(stopwords=STOPWORDS,width=800,height=600,background_color='black').generate("
plt.figure(figsize=(20,10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
(-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').generate('
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('english'))]
    review = ' '.join(review)
    corpus.append(review)
```

```
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package subjectivity to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/subjectivity.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package switchboard to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/switchboard.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package twitter_samples to

[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/twitter_samples.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package vader_lexicon to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet.zip.
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet3.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]    |   Unzipping models/wmt15_eval.zip.
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/word2vec_sample.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection all
```

4. Create Model

```
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)
```

```
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'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 77, 32)            225408

 lstm_2 (LSTM)               (None, 77, 50)            16600

 dropout_2 (Dropout)         (None, 77, 50)            0

 lstm_3 (LSTM)               (None, 60)                26640

 dropout_3 (Dropout)         (None, 60)                0

 dense_1 (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)
```

```
Epoch 1/10
131/131 [==============================] - 16s 94ms/step - loss: 0.5379 - accuracy: 0.8612 - v
Epoch 2/10
131/131 [==============================] - 12s 90ms/step - loss: 1390983.1250 - accuracy: 0.89
Epoch 3/10
131/131 [==============================] - 12s 90ms/step - loss: 46.1790 - accuracy: 0.9167 -
Epoch 4/10
131/131 [==============================] - 12s 90ms/step - loss: 0.2061 - accuracy: 0.9746 - v
Epoch 5/10
131/131 [==============================] - 12s 90ms/step - loss: 0.1611 - accuracy: 0.9761 - v
Epoch 6/10
131/131 [==============================] - 12s 90ms/step - loss: 0.5112 - accuracy: 0.9797 - v
Epoch 7/10
131/131 [==============================] - 12s 93ms/step - loss: 0.1295 - accuracy: 0.9828 - v
Epoch 8/10
131/131 [==============================] - 12s 92ms/step - loss: 0.1094 - accuracy: 0.9840 - v
Epoch 9/10
131/131 [==============================] - 12s 90ms/step - loss: 0.0954 - accuracy: 0.9840 - v
Epoch 10/10
131/131 [==============================] - 12s 90ms/step - loss: 0.0826 - accuracy: 0.9859 - v
<keras.callbacks.History at 0x7f748a7aa1d0>
```

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

```
44/44 [==============================] - 1s 22ms/step - loss: 0.1963 - accuracy: 0.9706
[0.19628430902957916, 0.9705671072006226]
```

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('english'))]
    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")
    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: Go until jurong point, crazy.. Available only in bugis n great world la e buf
1/1 [==============================] - 0s 369ms/step
Not a spam
```

```
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: WINNER!! As a valued network customer you have been selected to receivea �90
1/1 [==============================] - 0s 308ms/step
It is a spam
```