

**Assignment Date: 03 November 2022**

**Student Name: P.Nelson Jebas**

## ▼ 1. Download the dataset [link](#)

- Label - Ham or Spam
- 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()
```

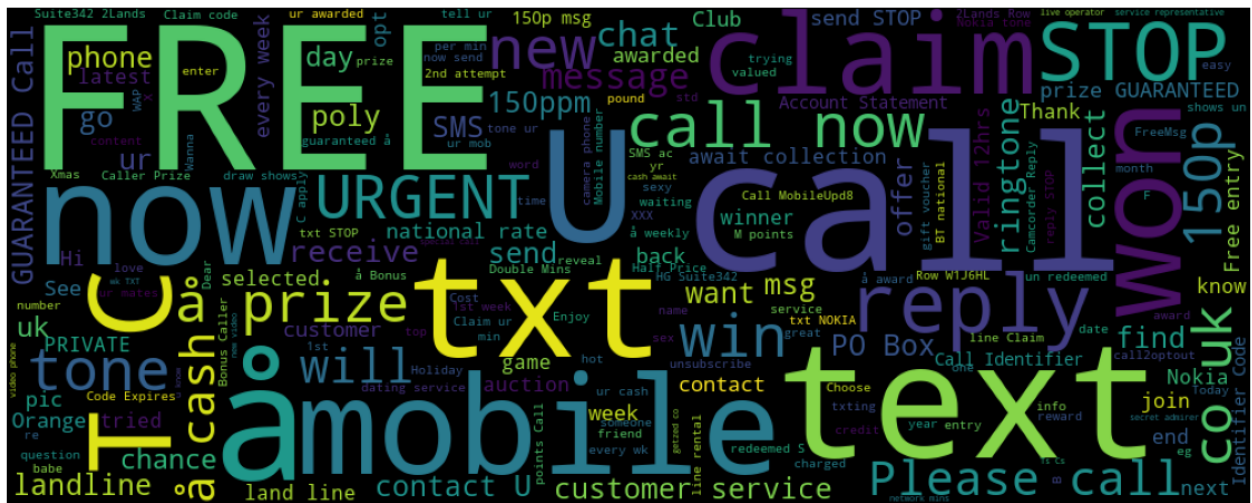
	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...



```
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')
```

$$(-0.5, 999.5, 399.5, -0.5)$$


```
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] | Unzipping corpora/pe08.zip.
[nltk_data] | Downloading package perluniprops to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping misc/perluniprops.zip.
[nltk_data] | Downloading package pil to /root/nltk_data...
[nltk_data] | Unzipping corpora/pil.zip.
[nltk_data] | Downloading package pl196x to /root/nltk_data...
[nltk_data] | Unzipping corpora/pl196x.zip.
[nltk_data] | Downloading package porter_test to /root/nltk_data...
[nltk_data] | Unzipping stemmers/porter_test.zip.
[nltk_data] | Downloading package ppattach to /root/nltk_data...
[nltk_data] | Unzipping corpora/ppattach.zip.
[nltk_data] | Downloading package problem_reports to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/problem_reports.zip.
[nltk_data] | Downloading package product_reviews_1 to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/product_reviews_1.zip.
[nltk_data] | Downloading package product_reviews_2 to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/product_reviews_2.zip.
[nltk_data] | Downloading package propbank to /root/nltk_data...
[nltk_data] | Downloading package pros_cons to /root/nltk_data...
[nltk_data] | Unzipping corpora/pros_cons.zip.
[nltk_data] | Downloading package ptb to /root/nltk_data...
[nltk_data] | Unzipping corpora/ptb.zip.
[nltk_data] | Downloading package punkt to /root/nltk_data...
[nltk_data] | Unzipping tokenizers/punkt.zip.
[nltk_data] | Downloading package qc to /root/nltk_data...
[nltk_data] | Unzipping corpora/qc.zip.
[nltk_data] | Downloading package reuters to /root/nltk_data...

[nltk_data] | Downloading package rslp to /root/nltk_data...
[nltk_data] | Unzipping stemmers/rslp.zip.
[nltk_data] | Downloading package rte to /root/nltk_data...
[nltk_data] | Unzipping corpora/rte.zip.
[nltk_data] | Downloading package sample_grammars to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping grammars/sample_grammars.zip.
[nltk_data] | Downloading package semcor to /root/nltk_data...
[nltk_data] | Downloading package senseval to /root/nltk_data...
[nltk_data] | Unzipping corpora/senseval.zip.
[nltk_data] | Downloading package sentence_polarity to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/sentence_polarity.zip.
[nltk_data] | Downloading package sentiwordnet to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/sentiwordnet.zip.
[nltk_data] | Downloading package shakespeare to /root/nltk_data...
[nltk_data] | Unzipping corpora/shakespeare.zip.

```

```
[nltk_data] | Downloading package sinica_treebank to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/sinica_treebank.zip.
[nltk_data] | Downloading package smultron to /root/nltk_data...
[nltk_data] | Unzipping corpora/smultron.zip.
[nltk_data] | Downloading package snowball_data to
[nltk_data] | /root/nltk_data...
[nltk_data] | Downloading package spanish_grammars to
[nltk_data] | /root/nltk_data...
```

## ▼ 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'))
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria
```

```
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)
```

```
Epoch 1/10
131/131 [=====] - 37s 249ms/step - loss: 0.3597 - accuracy: 0.8
Epoch 2/10
131/131 [=====] - 33s 254ms/step - loss: 67528.7734 - accuracy
Epoch 3/10
131/131 [=====] - 32s 248ms/step - loss: 3482.1545 - accuracy:
Epoch 4/10
131/131 [=====] - 33s 248ms/step - loss: 0.5708 - accuracy: 0.9
Epoch 5/10
131/131 [=====] - 34s 260ms/step - loss: 0.0747 - accuracy: 0.9
Epoch 6/10
131/131 [=====] - 32s 245ms/step - loss: 0.0580 - accuracy: 0.9
Epoch 7/10
131/131 [=====] - 33s 249ms/step - loss: 0.0427 - accuracy: 0.9
Epoch 8/10
131/131 [=====] - 32s 244ms/step - loss: 0.1001 - accuracy: 0.9
Epoch 9/10
131/131 [=====] - 32s 247ms/step - loss: 0.0357 - accuracy: 0.9
Epoch 10/10
131/131 [=====] - 32s 246ms/step - loss: 0.0289 - accuracy: 0.9
<keras.callbacks.History at 0x7feed0948990>
```



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

```
44/44 [=====] - 1s 21ms/step - loss: 0.0839 - accuracy: 0.9806
[0.08388189226388931, 0.980617344379425]
```

## ▼ 8. Save the Model

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

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

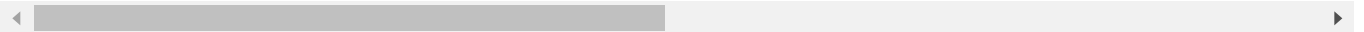
## ▼ 9. Test the Model

```
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([clean])
    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: "Thanks for your Ringtone Order, Reference T91. You will be charged GBP
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria
1/1 [=====] - 0s 284ms/step
It is 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: Keep my payasam there if rinu brings,,,
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria
1/1 [=====] - 0s 250ms/step
Not a spam
```

