

Assignment 4

▼ 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...
4	ham	Nah I don't think he goes to usf, he lives aro...



```
<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
```

<https://colab.research.google.com/drive/1b2G8kiW2FcXm15KuGPEmZCQhGN6kVU4Z#scrollTo=6e7e96f3&printMode=true>

[illegible]

```
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('eng
    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] | 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=
```

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



```
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 [=====] - 41s 283ms/step - loss: 0.5541 - accuracy:
Epoch 2/10
131/131 [=====] - 37s 280ms/step - loss: 5.3924 - accuracy:
Epoch 3/10
131/131 [=====] - 36s 273ms/step - loss: 0.0553 - accuracy:
Epoch 4/10
131/131 [=====] - 35s 266ms/step - loss: 0.0486 - accuracy:
Epoch 5/10
131/131 [=====] - 36s 273ms/step - loss: 0.0189 - accuracy:
Epoch 6/10
131/131 [=====] - 36s 273ms/step - loss: 0.0145 - accuracy:
Epoch 7/10
131/131 [=====] - 36s 273ms/step - loss: 0.0135 - accuracy:
Epoch 8/10
131/131 [=====] - 39s 296ms/step - loss: 0.0586 - accuracy:
Epoch 9/10
131/131 [=====] - 36s 274ms/step - loss: 0.0238 - accuracy:
Epoch 10/10
131/131 [=====] - 36s 273ms/step - loss: 0.0110 - accuracy:
<keras.callbacks.History at 0x7f808a0b5b10>
```

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

```
44/44 [=====] - 1s 22ms/step - loss: 1.0044 - accuracy: 0.9
[1.004435420036316, 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('eng
    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: Even my brother is not like to speak with me. They treat me like ai
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the c
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the
1/1 [=====] - 0s 266ms/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: "URGENT! You have won a 1 week FREE membership in our £100,000 Pri
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the c
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_functio
1/1 [=====] - 0s 294ms/step
It is a spam
```

[Colab paid products](#) - [Cancel contracts here](#)

✓ 6s completed at 3:35 PM

