

---

# 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 Co until i want point crazy Available only

```
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(ms1)
plt.figure(figsize=(20,10))
plt.imshow(wordcloud)
plt.axis('off')
```

```
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')
```

```
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 collection 'all'
[nlTK_data] |
[nlTK_data] | Downloading package abc to /root/nlTK_data...
[nlTK_data] | Unzipping corpora/abc.zip.
```

```
[nltk_data] | Downloading package alpino to /root/nltk_data...
[nltk_data] | Unzipping corpora/alpino.zip.
[nltk_data] | Downloading package averaged_perceptron_tagger to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] | Downloading package averaged_perceptron_tagger_ru to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping
[nltk_data] | taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data] | Downloading package basque_grammars to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping grammars/basque_grammars.zip.
[nltk_data] | Downloading package biocreative_ppi to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/biocreative_ppi.zip.
[nltk_data] | Downloading package bllip_wsj_no_aux to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data] | Downloading package book_grammars to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping grammars/book_grammars.zip.
[nltk_data] | Downloading package brown to /root/nltk_data...
[nltk_data] | Unzipping corpora/brown.zip.
[nltk_data] | Downloading package brown_tei to /root/nltk_data...
[nltk_data] | Unzipping corpora/brown_tei.zip.
[nltk_data] | Downloading package cess_cat to /root/nltk_data...
[nltk_data] | Unzipping corpora/cess_cat.zip.
[nltk_data] | Downloading package cess_esp to /root/nltk_data...
[nltk_data] | Unzipping corpora/cess_esp.zip.
[nltk_data] | Downloading package chat80 to /root/nltk_data...
[nltk_data] | Unzipping corpora/chat80.zip.
[nltk_data] | Downloading package city_database to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/city_database.zip.
[nltk_data] | Downloading package cmudict to /root/nltk_data...
[nltk_data] | Unzipping corpora/cmudict.zip.
[nltk_data] | Downloading package comparative_sentences to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/comparative_sentences.zip.
[nltk_data] | Downloading package comtrans to /root/nltk_data...
[nltk_data] | Downloading package conll2000 to /root/nltk_data...
[nltk_data] | Unzipping corpora/conll2000.zip.
[nltk_data] | Downloading package conll2002 to /root/nltk_data...
[nltk_data] | Unzipping corpora/conll2002.zip.
[nltk_data] | Downloading package conll2007 to /root/nltk_data...
[nltk_data] | Downloading package crubadan to /root/nltk_data...
[nltk_data] | Unzipping corpora/crubadan.zip.
[nltk_data] | Downloading package dependency_treebank to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/dependency_treebank.zip.
[nltk_data] | Downloading package dolch to /root/nltk_data...
[nltk_data] | Unzipping corpora/dolch.zip.
[nltk_data] | Downloading package europarl_raw 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

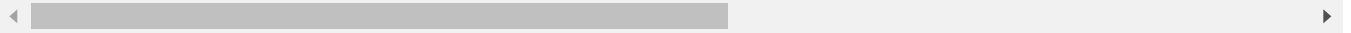
```
TOT_SIZE = len(token.word_index) + 1
model = Sequential()
#IP Layer
model.add(Embedding(TOT_SIZE,32,input_length=max_length_sequence))
model.add(LSTM(units=50, activation = 'relu',return_sequences=True))
```

```

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 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)
```

```

Epoch 1/10
131/131 [=====] - 43s 292ms/step - loss: 0.4032 - accuracy: 0.
Epoch 2/10
131/131 [=====] - 34s 261ms/step - loss: 249.8932 - accuracy:

```

```
Epoch 3/10
131/131 [=====] - 34s 262ms/step - loss: 0.2565 - accuracy: 0.
Epoch 4/10
131/131 [=====] - 39s 295ms/step - loss: 0.1319 - accuracy: 0.
Epoch 5/10
131/131 [=====] - 35s 269ms/step - loss: 0.0814 - accuracy: 0.
Epoch 6/10
131/131 [=====] - 35s 267ms/step - loss: 0.0520 - accuracy: 0.
Epoch 7/10
131/131 [=====] - 35s 266ms/step - loss: 285.4002 - accuracy:
Epoch 8/10
131/131 [=====] - 34s 263ms/step - loss: 0.2325 - accuracy: 0.
Epoch 9/10
131/131 [=====] - 34s 260ms/step - loss: 0.1270 - accuracy: 0.
Epoch 10/10
131/131 [=====] - 34s 260ms/step - loss: 0.0486 - accuracy: 0.
<keras.callbacks.History at 0x7fa460decd90>
```

◀  ▶

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

```
44/44 [=====] - 1s 23ms/step - loss: 0.1682 - accuracy: 0.9749
[0.1681801825761795, 0.9748743772506714]
```

◀  ▶

## ▼ 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")
```

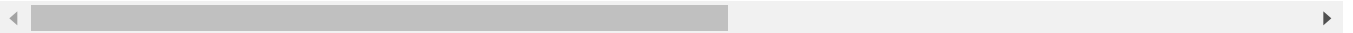
Enter a message: spam

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

1/1 [=====] - 0s 286ms/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")
```



✓ 58s completed at 10:14 PM

