

## Assignment -4

### Python Programming

Assignment Date	20 October 2022
Student Name	Steve John D
Student Roll Number	212219060259
Maximum Marks	2 Marks

1. Download the dataset "spam.csv"
2. Import required library

```
In [23]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.optimizers import Adam
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
```

3. Read the data set

```
In [13]: df = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
df.head()
```

```
Out[13]:
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

#### Preprocessing the dataset

```
In [14]: df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
```

```
In [15]: from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

```
In [16]: X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
In [17]: X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
```

```
In [18]: max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = pad_sequences(sequences,maxlen=max_len)
```

#### 4. CREATE MODEL & ADD LAYERS

```
In [19]: inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
```

```
In [20]: model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 128)	91648
dense (Dense)	(None, 128)	16512
activation (Activation)	(None, 128)	0
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129
activation_1 (Activation)	(None, 1)	0

=====  
Total params: 158,289  
Trainable params: 158,289  
Non-trainable params: 0  
=====

## 5.Compile the Model

```
In [24]: model.compile(loss='binary_crossentropy',optimizer=Adam(),metrics=['accuracy'])
```

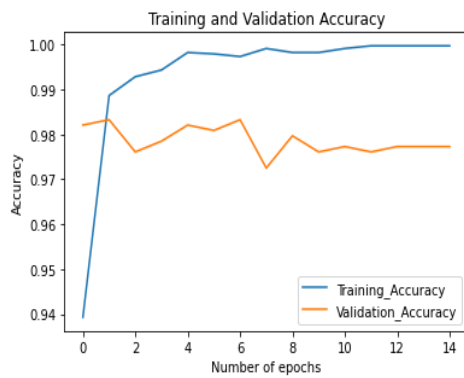
## 6.FIT THE MODEL

```
In [25]: history = model.fit(sequences_matrix,Y_train,batch_size=20,epochs=15,  
validation_split=0.2)
```

```
Epoch 1/15  
168/168 [=====] - 35s 189ms/step - loss: 0.1840 - accuracy: 0.9393 - val_loss: 0.0674 - val_accuracy: 0.9821  
Epoch 2/15  
168/168 [=====] - 33s 196ms/step - loss: 0.0379 - accuracy: 0.9886 - val_loss: 0.0608 - val_accuracy: 0.9833  
Epoch 3/15  
168/168 [=====] - 31s 184ms/step - loss: 0.0189 - accuracy: 0.9928 - val_loss: 0.0805 - val_accuracy: 0.9761  
Epoch 4/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0180 - accuracy: 0.9943 - val_loss: 0.0821 - val_accuracy: 0.9785  
Epoch 5/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0082 - accuracy: 0.9982 - val_loss: 0.0990 - val_accuracy: 0.9821  
Epoch 6/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0062 - accuracy: 0.9979 - val_loss: 0.1117 - val_accuracy: 0.9809  
Epoch 7/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0069 - accuracy: 0.9973 - val_loss: 0.1254 - val_accuracy: 0.9833  
Epoch 8/15  
168/168 [=====] - 33s 198ms/step - loss: 0.0044 - accuracy: 0.9991 - val_loss: 0.1335 - val_accuracy: 0.9725  
Epoch 9/15  
168/168 [=====] - 31s 186ms/step - loss: 0.0097 - accuracy: 0.9982 - val_loss: 0.1148 - val_accuracy: 0.9797  
Epoch 10/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0051 - accuracy: 0.9982 - val_loss: 0.1406 - val_accuracy: 0.9761  
Epoch 11/15  
168/168 [=====] - 31s 186ms/step - loss: 0.0041 - accuracy: 0.9991 - val_loss: 0.1406 - val_accuracy: 0.9773  
Epoch 12/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0025 - accuracy: 0.9997 - val_loss: 0.1444 - val_accuracy: 0.9761  
Epoch 13/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0022 - accuracy: 0.9997 - val_loss: 0.1363 - val_accuracy: 0.9773  
Epoch 14/15  
168/168 [=====] - 33s 197ms/step - loss: 0.0016 - accuracy: 0.9997 - val_loss: 0.1514 - val_accuracy: 0.9773  
Epoch 15/15  
168/168 [=====] - 31s 187ms/step - loss: 0.0022 - accuracy: 0.9997 - val_loss: 0.1462 - val_accuracy: 0.9773
```

```
In [26]: metrics = pd.DataFrame(history.history)  
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation_Accuracy'})  
def plot_graphs1(var1, var2, string):  
    metrics[[var1, var2]].plot()  
    plt.title('Training and Validation ' + string)  
    plt.xlabel ('Number of epochs')  
    plt.ylabel(string)  
    plt.legend([var1, var2])
```

```
In [30]: plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'Accuracy')
```



## 7. SAVE THE MODEL

```
In [33]: model.save('A4Spam_sms_classifier.h5')
```

## 8.TEST THE MODEL

```
In [34]: test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
```

```
In [35]: accuracy1 = model.evaluate(test_sequences_matrix,Y_test)
```

44/44 [=====] - 4s 80ms/step - loss: 0.1045 - accuracy: 0.9864

```
In [40]: print(' loss: {:.4f}'.format(accuracy1[0]))
print(' Accuracy: {:.4f}'.format(accuracy1[1]))
```

loss: 0.1045  
Accuracy: 0.9864