

## Assignment - 4

### SMS SPAM Classification

Assignment Date	27 October2022
Student Name	PRIYADHARSHINI N
Student Roll Number	820419106042
Maximum Marks	2 Marks

1.Import required library

```
[2] import pandas as pd
import numpy as np
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 RMSprop
from keras.preprocessing.text import Tokenizer
from keras_preprocessing import sequence
from keras.utils import to_categorical
from keras.models import load_model
```

2.Read Dataset and do preprocessing

```
[6] df = pd.read_csv('/content/drive/MyDrive/Dataset/spam (1).csv',delimiter=',',encoding='latin-1')
df.head()
```

	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

```
✓ [7] df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True) #dropping unwanted columns
1s df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0    v1      5572 non-null    object
 1    v2      5572 non-null    object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
✓ [8] df.groupby(['v1']).size()
1s
```

```
v1
ham      4825
spam      747
dtype: int64
```

```
✓ [9] # Label Encoding target column
0s X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
✓ [10] # Test and train split
1s X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

```
✓ [12] # Tokenisation function
1s 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 = sequence.pad_sequences(sequences,maxlen=max_len)
```

3. Create Model

4. Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
✓ [13] # Creating LSTM model
0s inputs = Input(name='InputLayer',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FullyConnectedLayer1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='OutputLayer')(layer)
layer = Activation('sigmoid')(layer)
```

## 5. Compile the Model

```
model = Model(inputs=inputs, outputs=layer)
model.summary()
model.compile(loss='binary_crossentropy', optimizer=RMSprop(), metrics=['accuracy'])
```

Model: "model"

Layer (type)	Output Shape	Param #
=====		
InputLayer (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	500000
lstm (LSTM)	(None, 64)	29440
FullyConnectedLayer1 (Dense)	(None, 256)	16640
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
OutputLayer (Dense)	(None, 1)	257
activation_1 (Activation)	(None, 1)	0
=====		
Total params: 96,337		
Trainable params: 96,337		
Non-trainable params: 0		

## 6. Fit the Model

```
model.fit(sequences_matrix, Y_train, batch_size=128, epochs=10,
          validation_split=0.2)
```

```
Epoch 1/10
30/30 [=====] - 11s 280ms/step - loss: 0.3416 - accuracy: 0.8598 - val_loss: 0.1802 - val_accuracy: 0.9736
Epoch 2/10
30/30 [=====] - 8s 258ms/step - loss: 0.0934 - accuracy: 0.9786 - val_loss: 0.0732 - val_accuracy: 0.9789
Epoch 3/10
30/30 [=====] - 8s 261ms/step - loss: 0.0436 - accuracy: 0.9873 - val_loss: 0.0508 - val_accuracy: 0.9873
Epoch 4/10
30/30 [=====] - 8s 259ms/step - loss: 0.0298 - accuracy: 0.9905 - val_loss: 0.0538 - val_accuracy: 0.9863
Epoch 5/10
30/30 [=====] - 8s 257ms/step - loss: 0.0248 - accuracy: 0.9931 - val_loss: 0.0686 - val_accuracy: 0.9842
Epoch 6/10
30/30 [=====] - 8s 260ms/step - loss: 0.0185 - accuracy: 0.9952 - val_loss: 0.0677 - val_accuracy: 0.9863
Epoch 7/10
30/30 [=====] - 8s 259ms/step - loss: 0.0140 - accuracy: 0.9958 - val_loss: 0.0745 - val_accuracy: 0.9842
Epoch 8/10
30/30 [=====] - 8s 258ms/step - loss: 0.0094 - accuracy: 0.9974 - val_loss: 0.0919 - val_accuracy: 0.9800
Epoch 9/10
30/30 [=====] - 9s 291ms/step - loss: 0.0084 - accuracy: 0.9979 - val_loss: 0.0997 - val_accuracy: 0.9821
Epoch 10/10
30/30 [=====] - 8s 260ms/step - loss: 0.0083 - accuracy: 0.9984 - val_loss: 0.0989 - val_accuracy: 0.9789
<keras.callbacks.History at 0x7f3be5fbf2d0>
```

## 7. Save the Model

```
[16] model.save("model_1")
```

```
WARNING:absl:Function `wrapped_model` contains input name(s) InputLayer with unsupported characters which will be renamed to inputlayer in the SavedModel
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_layer_call_and_return_conditional_losses while saving (showing 2 of 2):
```

## 8. Test the Model

```
[17] test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = sequence.pad_sequences(test_sequences,maxlen=max_len)
accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:.03f}'.format(accuracy[1]))
```

```
27/27 [=====] - 1s 22ms/step - loss: 0.0721 - accuracy: 0.9856
Accuracy: 0.986
```

```
[18] y_pred = model.predict(test_sequences_matrix)
print(y_pred[25:40].round(3))
```

```
27/27 [=====] - 1s 23ms/step
[[0.002]
 [0.001]
 [0. ]
 [1. ]
 [0. ]
 [0. ]
 [0. ]
 [0. ]
 [0.016]
 [0. ]
 [1. ]
 [0. ]
 [0. ]
 [0.001]
 [0. ]]
```

```
print(Y_test[25:40])
```

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$