

Assignment - 4
Artificial Intelligence

Assignment Date	12 October 2022
Student Name	RAVILLA RAMA RAO
Student Roll Number	111519104113
Maximum Marks	2 Marks

1.Download the dataset

2.Import required library

```
In [ ]: 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
```

3.Read Dataset and do preprocessing

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

```
Out[ ]:      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...
```

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

```
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
```

```
In [ ]: # Count of Spam and Ham values
df.groupby(['v1']).size()
```

```
Out[ ]: v1
ham      4825
spam      747
dtype: int64
```

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

```
In [ ]: # Test and train split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

```
In [ ]: # Tokenisation function
max_words = 10000
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)
```

4.Create Model and 5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
In [ ]: # Creating LSTM model
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)
```

6.Compile the model

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

Model: "model_2"

Layer (type)	Output Shape	Param #
InputLayer (InputLayer)	[(None, 150)]	0
embedding_2 (Embedding)	(None, 150, 50)	500000
lstm_2 (LSTM)	(None, 64)	29440
FullyConnectedLayer1 (Dense)	(None, 256)	16640
activation_4 (Activation)	(None, 256)	0
dropout_2 (Dropout)	(None, 256)	0
OutputLayer (Dense)	(None, 1)	257
activation_5 (Activation)	(None, 1)	0
Total params: 96,337		
Trainable params: 96,337		
Non-trainable params: 0		

7.Fit the Model

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

Epoch 1/10
30/30 [=====] - ETA: 0s - loss: 0.3374 - accuracy: 0.8712
```

8.Save the Model

```
In [ ]: model.save("model_1")
```

9. Test the model

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

```
In [ ]: accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:.3f}'.format(accuracy[1]))
```

```
27/27 [-----] - 1s 36ms/step - loss: 0.1163 - accuracy: 0.9856
Accuracy: 0.986
```

```
In [ ]: y_pred = model.predict(test_sequences_matrix)
         print(y_pred[25:48].round(3))
```

```
27/27 [-----] - 1s 28ms/step
[[0. ]
 [0. ]
 [0. ]
 [0. ]
 [0. ]
 [0.882]
 [0. ]
 [0.824]
 [0. ]
 [0. ]
 [0. ]
 [0. ]
 [0. ]
 [0. ]
 [0. ]]
```

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

[illegible]