ASSIGNMENT 4

ASSIGNMENT DATE	8 OCTOBER 2022
STUDENT NAME	HARI @ ARJUN.K
STUDENT ROLL NUMBER	2019504525
MAXIMUM MARKS	2 MARKS

1. Downloading dataset spam.csv

2. Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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 pad_sequences
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
```

3. Read dataset and do pre-processing

```
data=pd.read_csv(r'C:\Users\Karthikeyan\Downloads\
spam.csv',encoding='latin')
df = pd.read_csv(r'C:\Users\Karthikeyan\Downloads\
spam.csv',delimiter=',',encoding='latin-1')
df.head()
```

```
     v1                                                 v2 Unnamed: 2  
\
0   ham  Go until jurong point, crazy.. Available only ...        NaN   

1   ham                      Ok lar... Joking wif u oni...        NaN   

2  spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN   

3   ham  U dun say so early hor... U c already then say...        NaN   

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


  Unnamed: 3 Unnamed: 4  
0        NaN        NaN  
1        NaN        NaN  
2        NaN        NaN  
3        NaN        NaN  
4        NaN        NaN  
```

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

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

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

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

# Test and train split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
# Tokenisation function
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 and 5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
# Creating LSTM model
inputs = Input(name='inputs',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FC1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='out_layer')(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
```

6.Compile the model & 7.Fit the Model

```
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=[
'accuracy'])
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
          validation_split=0.2)
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
30/30 [==============================] - 12s 250ms/step - loss: 0.0124
- accuracy: 0.9974 - val_loss: 0.0601 - val_accuracy: 0.9905
Epoch 2/10
30/30 [==============================] - 6s 215ms/step - loss: 0.0060
- accuracy: 0.9984 - val_loss: 0.0631 - val_accuracy: 0.9895
Epoch 3/10
30/30 [==============================] - 6s 204ms/step - loss: 0.0067
- accuracy: 0.9989 - val_loss: 0.0735 - val_accuracy: 0.9884
Epoch 4/10
30/30 [==============================] - 7s 225ms/step - loss: 0.0044
- accuracy: 0.9987 - val_loss: 0.0619 - val_accuracy: 0.9905
Epoch 5/10
30/30 [==============================] - 6s 212ms/step - loss: 0.0042
```

```
- accuracy: 0.9984 - val_loss: 0.0666 - val_accuracy: 0.9895
Epoch 6/10
30/30 [==============================] - 13s 446ms/step - loss: 0.0038
- accuracy: 0.9987 - val_loss: 0.0755 - val_accuracy: 0.9852
Epoch 7/10
30/30 [==============================] - 19s 580ms/step - loss: 0.0033
- accuracy: 0.9992 - val_loss: 0.0754 - val_accuracy: 0.9884
Epoch 8/10
30/30 [==============================] - 6s 212ms/step - loss: 0.0033
- accuracy: 0.9992 - val_loss: 0.0689 - val_accuracy: 0.9905
Epoch 9/10
30/30 [==============================] - 7s 220ms/step - loss: 0.0018
- accuracy: 0.9997 - val_loss: 0.0942 - val_accuracy: 0.9873
Epoch 10/10
30/30 [==============================] - 23s 799ms/step - loss: 0.0021
- accuracy: 0.9995 - val_loss: 0.1041 - val_accuracy: 0.9852

<keras.callbacks.History at 0x22c5a8baa90>
```

8.Save the Model

```
model.save('sms_classifier.h5')
```

9.Test the model

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
accr = model.evaluate(test_sequences_matrix,Y_test)

27/27 [==============================] - 1s 31ms/step - loss: 0.2302 -
accuracy: 0.9773

print('Test set\n  Loss: {:0.3f}\n  Accuracy:
{:0.3f}'.format(accr[0],accr[1]))

Test set
  Loss: 0.230
  Accuracy: 0.977
```