Assignment -4

SMS SPAM Classification

Assignment Date	29 October 2022
Student Name	Ms.SUGASHINI.T
Student Roll Number	820419104076
Maximum Marks	2 Marks

Question-1:

1.Import the Required 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
```

2.Read dataset and do pre-processing

```
[18] df = pd.read_csv("/content/drive/MyDrive/spam.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

3.Preprocessing the Dataset

```
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
```

```
[20] X= df.v2

     Y = df.v1

     le=LabelEncoder()

     Y=le.fit_transform(Y)
```

4.Create Model

```
[21] X_train,X_test,Y_train, Y_test=train_test_split(X,Y, test_size=0.15)
```

```
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)
```

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

```
[30] 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)

     model.summary()
```

```
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
_________________________________________________________________
```

6.Compile the Model

```
[36] model.compile(loss= "binary_crossentropy",optimizer=RMSprop(), metrics=["accuracy"])
```

7.Fit the Model

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

```
Epoch 1/10
30/30 [==============================] - 8s 190ms/step - loss: 0.3187 - accuracy: 0.8762 - val_loss: 0.1664 - val_accuracy: 0.9420
Epoch 2/10
30/30 [==============================] - 5s 174ms/step - loss: 0.0786 - accuracy: 0.9813 - val_loss: 0.0776 - val_accuracy: 0.9789
Epoch 3/10
30/30 [==============================] - 5s 172ms/step - loss: 0.0409 - accuracy: 0.9900 - val_loss: 0.0685 - val_accuracy: 0.9821
Epoch 4/10
30/30 [==============================] - 5s 173ms/step - loss: 0.0333 - accuracy: 0.9913 - val_loss: 0.0815 - val_accuracy: 0.9800
Epoch 5/10
30/30 [==============================] - 5s 172ms/step - loss: 0.0266 - accuracy: 0.9918 - val_loss: 0.0785 - val_accuracy: 0.9821
Epoch 6/10
30/30 [==============================] - 5s 175ms/step - loss: 0.0178 - accuracy: 0.9952 - val_loss: 0.0902 - val_accuracy: 0.9821
Epoch 7/10
30/30 [==============================] - 6s 216ms/step - loss: 0.0135 - accuracy: 0.9971 - val_loss: 0.0935 - val_accuracy: 0.9778
Epoch 8/10
30/30 [==============================] - 5s 172ms/step - loss: 0.0113 - accuracy: 0.9976 - val_loss: 0.1017 - val_accuracy: 0.9810
Epoch 9/10
30/30 [==============================] - 5s 173ms/step - loss: 0.0075 - accuracy: 0.9979 - val_loss: 0.1355 - val_accuracy: 0.9821
Epoch 10/10
30/30 [==============================] - 5s 174ms/step - loss: 0.0057 - accuracy: 0.9987 - val_loss: 0.1478 - val_accuracy: 0.9810
<keras.callbacks.History at 0x7f3d1a538890>
```

8.Save the Model

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

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

9.Test the Model

```
[46] accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 0s 16ms/step - loss: 0.0705 - accuracy: 0.9856
```

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

```
Test set
 Loss: 0.071
 Accuracy: 0.986
```

+ Code