Assignment -4

Problem Statement :- SMS SPAM Classification

Assignment Date	27 October 2022
Student Name	T.Kiruthika
Student Roll Number	923119106002
Maximum Marks	2 Marks

#1.Download the Dataset

Dataset Downloaded and uploaded to drive
https://www.kaggle.com/code/kredy10/simple-lstm-for-text-classification/data

#2.Import the necessary 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

Read dataset

```
df = pd.read_csv('/content/archive.zip',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
```

#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
```

```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)

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)
```

#4.Create Model

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

```
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

```
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accura
cy'])
```

#7.Train and Fit the Mode

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

```
Epoch 1/10
30/30 [==============================] - 8s 32ms/step - loss: 0.3370 -
accuracy: 0.8720 - val_loss: 0.1678 - val_accuracy: 0.9335
Epoch 2/10
30/30 [==============================] - 0s 13ms/step - loss: 0.0946 -
accuracy: 0.9770 - val_loss: 0.0413 - val_accuracy: 0.9895
Epoch 3/10
30/30 [==============================] - 0s 14ms/step - loss: 0.0449 -
accuracy: 0.9865 - val_loss: 0.0261 - val_accuracy: 0.9947
Epoch 4/10
30/30 [==============================] - 0s 14ms/step - loss: 0.0377 -
accuracy: 0.9876 - val_loss: 0.0275 - val_accuracy: 0.9916
Epoch 5/10
30/30 [==============================] - 0s 14ms/step - loss: 0.0245 -
accuracy: 0.9913 - val_loss: 0.0353 - val_accuracy: 0.9905
Epoch 6/10
30/30 [==============================] - 0s 14ms/step - loss: 0.0224 -
accuracy: 0.9926 - val_loss: 0.0284 - val_accuracy: 0.9916
```

```
Epoch 7/10
30/30 [==============================] - 0s 13ms/step - loss: 0.0156 -
accuracy: 0.9955 - val_loss: 0.0324 - val_accuracy: 0.9916
Epoch 8/10
30/30 [==============================] - 0s 14ms/step - loss: 0.0117 -
accuracy: 0.9963 - val_loss: 0.0477 - val_accuracy: 0.9884
Epoch 9/10
30/30 [==============================] - 0s 13ms/step - loss: 0.0084 -
accuracy: 0.9979 - val_loss: 0.0412 - val_accuracy: 0.9916
Epoch 10/10
30/30 [==============================] - 0s 13ms/step - loss: 0.0074 -
accuracy: 0.9982 - val_loss: 0.0478 - val_accuracy: 0.9916

<keras.callbacks.History at 0x7f3830281c90>
```

#8.Save The Model

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

#Preprocessing the Test Dataset

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

#9.Testing the Model

```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 0s 7ms/step - loss: 0.0989 -
accuracy: 0.9797
```

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

```
Test set
  Loss: 0.099
  Accuracy: 0.980
```