Assignment -4

Python Programming

Assignment Date	15 October 2022
Student Name	RESHMA P
Student Roll Number	111519104114
Maximum Marks	2 Marks

Problem Statement :- SMS SPAM Classification

Import the necessary libraries

Solution:

```
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
```

Download the Dataset

Solution:

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

Read dataset and do pre-processing

Solution:

Read dataset

```
In [21]: df = pd.read_csv('/content/drive/MyDrive/spam.csv',delimiter=',',encoding='latin-1')
         df.head()
```

Out[21]:

	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

Pre-processing the Dataset

In [22]:
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
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 [23]:
```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

In [24]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

In [25]:
```
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)
```

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

In [26]:
```
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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inputs (InputLayer)          [(None, 150)]             0

embedding_1 (Embedding)      (None, 150, 50)           50000

lstm_1 (LSTM)                (None, 64)                29440

FC1 (Dense)                  (None, 256)               16640

activation_2 (Activation)    (None, 256)               0

dropout_1 (Dropout)          (None, 256)               0

out_layer (Dense)            (None, 1)                 257

activation_3 (Activation)    (None, 1)                 0

=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
```

Compile the Model

In [27]:
```
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

Train and Fit the Model

In [28]:
```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
          validation_split=0.2)
```

```
Epoch 1/10
30/30 [==============================] - 10s 264ms/step - loss: 0.3182 - accuracy: 0.8788 - val_loss: 0.1571 - val_accuracy: 0.9715
Epoch 2/10
30/30 [==============================] - 7s 247ms/step - loss: 0.0805 - accuracy: 0.9786 - val_loss: 0.0742 - val_accuracy: 0.9778
Epoch 3/10
30/30 [==============================] - 7s 237ms/step - loss: 0.0403 - accuracy: 0.9881 - val_loss: 0.0670 - val_accuracy: 0.9821
Epoch 4/10
30/30 [==============================] - 7s 245ms/step - loss: 0.0272 - accuracy: 0.9929 - val_loss: 0.0806 - val_accuracy: 0.9778
Epoch 5/10
30/30 [==============================] - 7s 242ms/step - loss: 0.0220 - accuracy: 0.9937 - val_loss: 0.0820 - val_accuracy: 0.9800
Epoch 6/10
30/30 [==============================] - 7s 240ms/step - loss: 0.0178 - accuracy: 0.9955 - val_loss: 0.0787 - val_accuracy: 0.9789
Epoch 7/10
30/30 [==============================] - 7s 243ms/step - loss: 0.0150 - accuracy: 0.9958 - val_loss: 0.0969 - val_accuracy: 0.9800
Epoch 8/10
30/30 [==============================] - 7s 241ms/step - loss: 0.0162 - accuracy: 0.9958 - val_loss: 0.0901 - val_accuracy: 0.9768
Epoch 9/10
30/30 [==============================] - 7s 246ms/step - loss: 0.0099 - accuracy: 0.9968 - val_loss: 0.1284 - val_accuracy: 0.9789
Epoch 10/10
30/30 [==============================] - 7s 247ms/step - loss: 0.0355 - accuracy: 0.9905 - val_loss: 0.1264 - val_accuracy: 0.9726
```

Out[28]:

Save The Model

In [29]:
```
model.save('sms_classifier.h5')
```

Preprocessing the Test Dataset

In [30]:
```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
```

Testing the Model

In [31]:
```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 1s 20ms/step - loss: 0.0886 - accuracy: 0.9821
```

In [32]:
```
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

```
Test set
  Loss: 0.089
  Accuracy: 0.982
```