Name : Nandhini C S M

Roll No :19ITR051

Date : 05/11/2022

ASSIGNMANT - 4

1.Download the dataset from /content/spam.csv

2.Importing library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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 Adam
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 the dataset

```
data = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
data.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

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

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
M = data.v2
N = data.v1
le = LabelEncoder()
N = le.fit_transform(N)
N = N.reshape(-1,1)
M_train,M_test,N_train,N_test = train_test_split(M,N,test_size=0.25)
max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(M_train)
sequences = tok.texts_to_sequences(M_train)
sequences_matrix = pad_sequences(sequences,maxlen=max_len)
```

4.Creating a Model

```
inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
```

5.Add layer

```
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1.5)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
model.summary()
Model: "model"
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 150)]             0

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

 lstm (LSTM)                 (None, 128)               91648

 dense (Dense)               (None, 128)               16512

 activation (Activation)     (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

 activation_1 (Activation)   (None, 1)                 0
```

```
=================================================================
Total params: 158,289
Trainable params: 158,289
Non-trainable params: 0
_________________________________________________________________
```

6.Compile the model

```
model.compile(loss='binary_crossentropy',optimizer=Adam(),metrics=['accuracy'])
```

7.Fit the model

```
model.fit(sequences_matrix,N_train,batch_size=15,epochs=15,validation_split=0.2)
```

```
Epoch 1/15
223/223 [==============================] - 24s 107ms/step - loss: 0.0229 - accuracy:
Epoch 2/15
223/223 [==============================] - 23s 104ms/step - loss: 0.0115 - accuracy:
Epoch 3/15
223/223 [==============================] - 23s 104ms/step - loss: 0.0098 - accuracy:
Epoch 4/15
223/223 [==============================] - 24s 106ms/step - loss: 0.0086 - accuracy:
Epoch 5/15
223/223 [==============================] - 23s 104ms/step - loss: 0.0063 - accuracy:
Epoch 6/15
223/223 [==============================] - 23s 105ms/step - loss: 0.0049 - accuracy:
Epoch 7/15
223/223 [==============================] - 24s 106ms/step - loss: 0.0047 - accuracy:
Epoch 8/15
223/223 [==============================] - 23s 105ms/step - loss: 0.0027 - accuracy:
Epoch 9/15
223/223 [==============================] - 23s 104ms/step - loss: 0.0054 - accuracy:
Epoch 10/15
223/223 [==============================] - 23s 104ms/step - loss: 0.0033 - accuracy:
Epoch 11/15
223/223 [==============================] - 24s 106ms/step - loss: 0.0027 - accuracy:
Epoch 12/15
223/223 [==============================] - 24s 105ms/step - loss: 0.0027 - accuracy:
Epoch 13/15
223/223 [==============================] - 24s 106ms/step - loss: 0.0021 - accuracy:
Epoch 14/15
223/223 [==============================] - 23s 104ms/step - loss: 0.0017 - accuracy:
Epoch 15/15
223/223 [==============================] - 23s 103ms/step - loss: 0.0014 - accuracy:
<keras.callbacks.History at 0x7fc5f2209950>
```

8.Save the model

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

9.Test the model

```
test_sequences = tok.texts_to_sequences(M_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
accuracy1 = model.evaluate(test_sequences_matrix,N_test)
```

```
44/44 [==============================] - 2s 45ms/step - loss: 0.1242 - accuracy: 0.9
```