Assignment -4

Assignment Date	29 october2022
Student Name	Sanjana .L
Student Roll Number	820419106049
Maximum Marks	2 Marks

Problem Statement :- SMS SPAM Classification

Problem Statement: Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollar industry. At the same time,reduction in the cost of messaging services has resulted in growth in unsolicitedcommercial advertisements (spams) being sent to mobile phones. Due to Spam SMS,Mobile service providers suffer from some sort of financial problems as well as it reducescalling time for users. Unfortunately, if the user accesses such Spam SMS they may facethe problem of virus or malware. When SMS arrives at mobile it will disturb mobile userprivacy and concentration. It may lead to frustration for the user. So Spam SMS is one ofthe major issues in the wireless communication world and it grows day by day.

- Download the Dataset:- Dataset
- Import required library
- Read dataset and do pre-processing
- Create Model
- Add Layers (LSTM, Dense-(Hidden Layers), Output)
- Compile the Model
- Fit the Model
- Save The Model
- Test The Model

- **Import required library:**

```
#IMPORT LIBRARY
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 tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
%matplotlib inline
import csv
```

- **Read dataset :**

```
#READ THE DATASET
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/spam.txt')
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

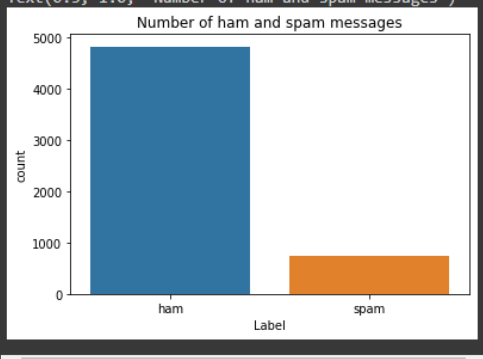
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
df.info()
```

```
<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
```

➢ Pre-processing:

```
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword a
  FutureWarning
Text(0.5, 1.0, 'Number of ham and spam messages')
```




```
[53] #PROCESS THE LABELS
     X = df.v2
     Y = df.v1
     le = LabelEncoder()
     Y = le.fit_transform(Y)
     Y = Y.reshape(-1,1)
```

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

```
[55] 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 = sequence.pad_sequences(sequences,maxlen=max_len)
```

➢ Create Model:

```
[56] #CREATE MODEL  - RNN
     #RNN STURCTURE
     def RNN():
         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)
         return model
```

➢ Compile the Model:

```
[57] #CALL THE FUNCTION AND COMPILE THE MODEL
     model = RNN()
     model.summary()
     model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

 embedding_3 (Embedding)     (None, 150, 50)           50000

 lstm_3 (LSTM)               (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation_6 (Activation)   (None, 256)               0

 dropout_3 (Dropout)         (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_7 (Activation)   (None, 1)                 0

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

➢ Fit , Save and Test The Model:

```
#FIT ON THE TRAINING DATA
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
          validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.0001)])
```

```
Epoch 1/10
30/30 [==============================] - 8s 185ms/step - loss: 0.3101 - accuracy: 0.8928 - val_loss: 0.1542 - val_accuracy: 0.9515
Epoch 2/10
30/30 [==============================] - 5s 160ms/step - loss: 0.0744 - accuracy: 0.9802 - val_loss: 0.0719 - val_accuracy: 0.9821
<keras.callbacks.History at 0x7f1d6718b450>
```

```
[59] #PROCESS THE TEST SET DATA
     test_sequences = tok.texts_to_sequences(X_test)
     test_sequences_matrix = sequence.pad_sequences(test_sequences,maxlen=max_len)
```

```
[60] #EVALUATE THE MODEL ON THE TEST SET
     accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 0s 14ms/step - loss: 0.0552 - accuracy: 0.9880
```

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

```
Test set
  Loss: 0.055
  Accuracy: 0.988
```