ASSIGNMENT-4

Problem Statement :- SMS SPAM Classification

Assignment Date	10 November 2022
Student Name	Mr.K.FRANCIS HUBAN
Student Roll Number	913319104018
Maximum Marks	2 Marks

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 unsolicited commercial 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 reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

1. **DOWNLOAD THE DATA SET**

2. **IMPORT REQUIRED 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 keras.models import Model

from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding

from tensorflow.keras.optimizers import RMSprop

from keras.preprocessing.text import Tokenizer

from keras.preprocessing import sequence

from tensorflow.keras.utils import to_categorical

from keras.callbacks import EarlyStopping

%matplotlib inline
```

3. **READ THE DATA SET DO PREPROCESSING**

Load the data into Pandas dataframe

```
df = pd.read_csv('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

Data Preprocessing

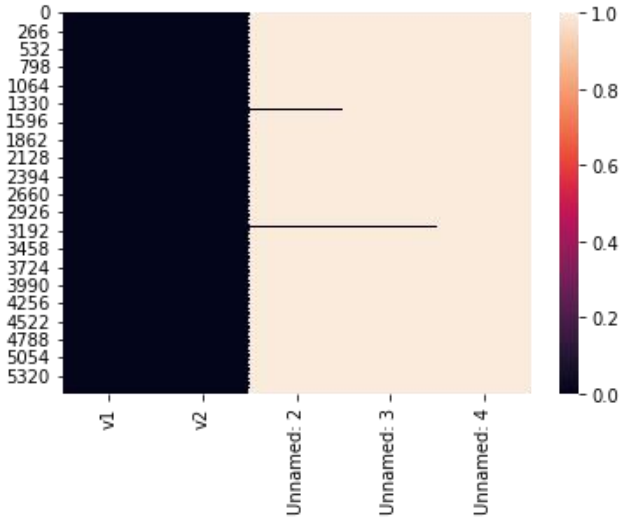
- *checking for null values*
- *Removing the null values*

```
df.shape
```

```
[3]  df.shape

     (5572, 5)
```

```
sns.heatmap(df.isnull())
```



DROPING THE COLUMNS WITH NULL VALUES

```
df.isnull().sum()
```

```
v1              0
v2              0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

DISTRIBUTION OF DATA

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

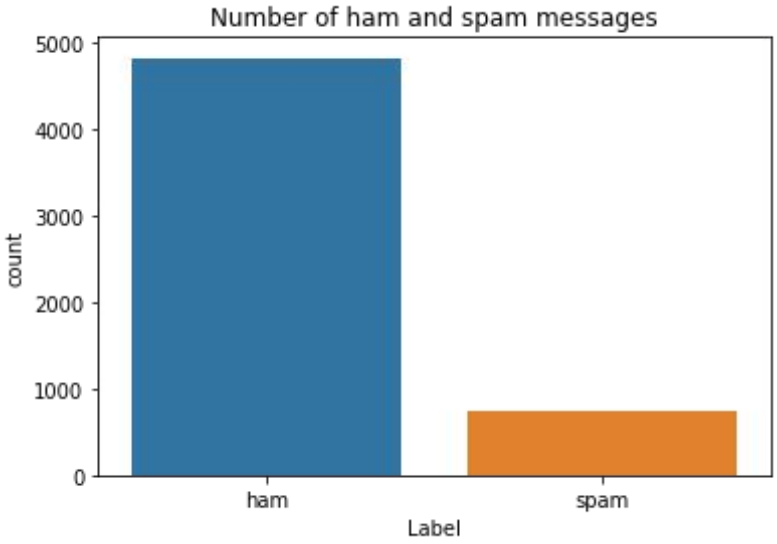
```
df.info()
```

```
<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
```

```
sns.countplot(df['v1'])
```

```
plt.xlabel('Label')
```

```
plt.title('Number of ham and spam messages')
```



Train and Test split

- *Create input and output vectors.*
- *Process the labels.*

```
X = df.v2
```

```
X
```

```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                             Rofl. Its true to its name
Name: v2, Length: 5572, dtype: object
```

Y = df.v1

Y

```
0        ham
1        ham
2       spam
3        ham
4        ham
        ...
5567    spam
5568     ham
5569     ham
5570     ham
5571     ham
Name: v1, Length: 5572, dtype: object
```

Encoding the target column

le = LabelEncoder()

Y = le.fit_transform(Y)

Y = Y.reshape(-1,1)

Y

```
array([[0],
       [0],
       [1],
       ...,
       [0],
       [0],
       [0]])
```

Y[:10]

```
array([[0],
       [0],
       [1],
       [0],
       [0],
       [1],
       [0],
       [0],
       [1],
       [1]])
```

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)

X_train.sample(5)

```
4590    Well done ENGLAND! Get the official poly ringt...
4369    Do you want a new Video handset? 750 any time ...
2839    BABE !!! I miiiiiiissssssssss you ! I need you...
5140    Yeah I imagine he would be really gentle. Unli...
109                        Dont worry. I guess he's busy.
Name: v2, dtype: object
```

4. CREATE THE MODEL

- Tokenize the data and convert the text to sequences.
- Add padding to ensure that all the sequences have the same shape.
- There are many ways of taking the max_len and here an arbitrary length of 150 is chosen.

from keras.preprocessing.text import Tokenizer

from keras.preprocessing import sequence

from tensorflow.keras.preprocessing.sequence import pad_sequences

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)

- built a RNN neural network
- Added LSTM and hidden layers to the neural network

```
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
model = RNN()
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
_________________________________________________________________
```

6.COMPILE THE MODEL

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

7.FIT THE MODEL

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=20,
          validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.0001)])
```

```
Epoch 1/20
30/30 [==============================] - 3s 30ms/step - loss: 0.3424 - accuracy: 0.8735 - val_loss: 0.2031 - val_accuracy:
Epoch 2/20
30/30 [==============================] - 0s 14ms/step - loss: 0.0944 - accuracy: 0.9784 - val_loss: 0.0459 - val_accuracy:
<keras.callbacks.History at 0x7f81d0104890>
```

Model Evaluation

test_sequences = tok.texts_to_sequences(X_test)

test_sequences_matrix =pad_sequences(test_sequences,maxlen=max_len)

test_sequences_matrix

```
array([[  0,   0,   0, ...,   0,   0,   8],
       [  0,   0,   0, ..., 826, 447, 447],
       [  0,   0,   0, ...,  22,  38, 307],
       ...,
       [  0,   0,   0, ..., 237,  90, 237],
       [  0,   0,   0, ...,  93, 846, 349],
       [  0,   0,   0, ...,   2,  12,  65]], dtype=int32)
```

accr = model.evaluate(test_sequences_matrix,Y_test)

```
27/27 [==============================] - 0s 7ms/step - loss: 0.0613 - accuracy: 0.9821
```

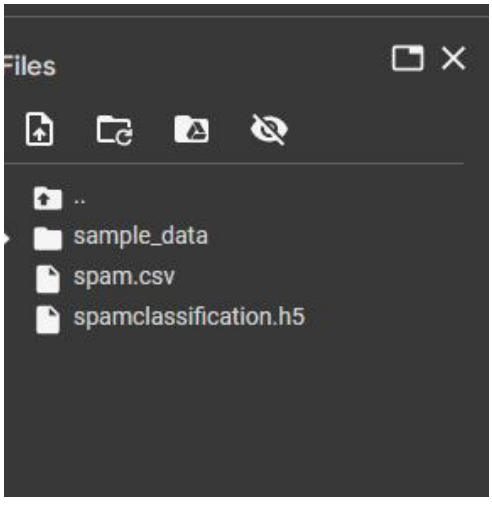
```
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```

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

```
Test set
  Loss: 0.061
  Accuracy: 0.982
```

8.SAVE THE MODEL

model.save("spamclassification.h5")



9.TEST THE MODEL

sample_texts = ["this week is a last dayfor submitting the task"]

txts = tok.texts_to_sequences(sample_texts)

txts = pad_sequences(txts, maxlen=max_len)

preds = model.predict(txts)

Preds

```
[42] model.save("spamclassification.h5")

[45] sample_texts = ["this week is a last dayfor submitting the task"]
     txts = tok.texts_to_sequences(sample_texts)
     txts = pad_sequences(txts, maxlen=max_len)
     preds = model.predict(txts)
     preds

     1/1 [==============================] - 0s 19ms/step
     array([[0.01753583]], dtype=float32)
```