ASSIGNMENT – 4

Problem Statement :- SMS SPAM Classification

Assignment Date	23 October 2022
Student Name	Gunupudi Venkata Lakshmi Durga Sunaina
Student Roll Number	113219041033
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 dataset: Dataset

Downloaded and uploaded the dataset

2.

Import the necessary libraries

```
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
```

24]

Python

3. Read dataset and do pre-processing

Read the Dataset

+ Code + Markdown

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

[5]

	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

Preprocessing the Dataset

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

[26]

```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

[27]

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

[28]

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)
```

[29]

```
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)
```

[30]

4.

Create Model and Add Layers

```
inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)

model.summary()
```

31]

In [9]:
```
df.columns
```

Out[9]:
```
Index(['v1', 'v2'], dtype='object')
```

In [10]:
```
data=df.rename(
{
    "v1":"Category",
    "v2":"Message"
},
    axis=1
)
```

In [11]:
```
data["Message Length"]=data["Message"].apply(len)
```

In [12]:
```
data.describe(include="all")
```

Out[12]:

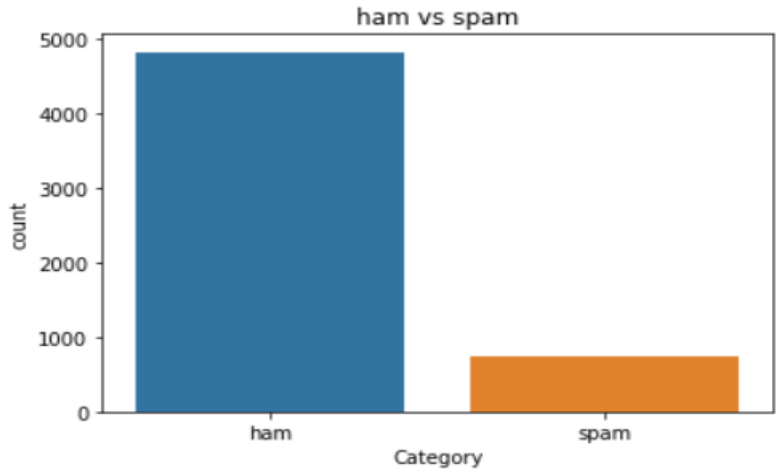
	Category	Message	Message Length
count	5572	5572	5572.000000
unique	2	5169	NaN
top	ham	Sorry, I'll call later	NaN
freq	4825	30	NaN
mean	NaN	NaN	80.118808
std	NaN	NaN	59.690841
min	NaN	NaN	2.000000
25%	NaN	NaN	36.000000
50%	NaN	NaN	61.000000
75%	NaN	NaN	121.000000
max	NaN	NaN	910.000000

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 150)]             0

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

 lstm_1 (LSTM)               (None, 128)               91648

 dense_2 (Dense)             (None, 128)               16512

 activation_2 (Activation)   (None, 128)               0

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 1)                 129

 activation_3 (Activation)   (None, 1)                 0

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

```
data["Category"].value_counts()
```

```
ham     4825
spam     747
Name: Category, dtype: int64
```

```
sns.countplot(
    data=data,
    x="Category"
)
plt.title("ham vs spam")
plt.show()
```



```
ham_count=data["Category"].value_counts()[0]
spam_count=data["Category"].value_counts()[1]

total_count=data.shape[0]

print("Ham contains:{:.2f}% of total data.".format(ham_count/total_count*100))
print("Spam contains:{:.2f}% of total data.".format(spam_count/total_count*100))
```

```
Ham contains:86.59% of total data.
Spam contains:13.41% of total data.
```

```
minority_len=len(data[data["Category"]=="spam"])
majority_len=len(data[data["Category"]=="ham"])
minority_indices=data[data["Category"]=="spam"].index
majority_indices=data[data["Category"]=="ham"].index
random_majority_indices=np.random.choice(
    majority_indices,
    size=minority_len,
    replace=False
)
undersampled_indices=np.concatenate([minority_indices,random_majority_indices])
df=data.loc[undersampled_indices]
df=df.sample(frac=1)

df=df.reset_index()
df=df.drop(
    columns=["index"],
)
```

```
df.shape
```

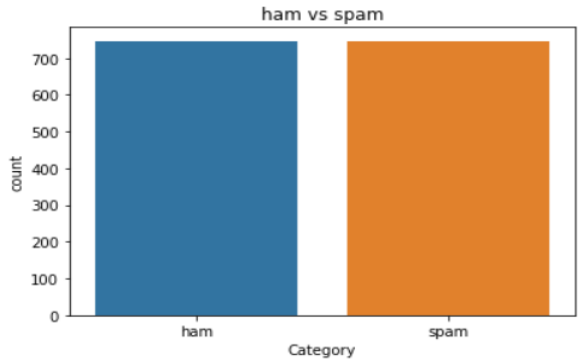
```
(1494, 3)
```

```
df["Category"].value_counts()
```

```
ham     747
spam    747
```

```
ham     747
spam    747
Name: Category, dtype: int64
```

```
sns.countplot(
    data=df,
    x="Category"
)
plt.title("ham vs spam")
plt.show()
```




```
df.head()
```

	Category	Message	Message Length
0	ham	Not yet. Just i'd like to keep in touch and it...	137
1	ham	Kent vale lor... ÌÏ wait 4 me there ar?	39
2	spam	URGENT! We are trying to contact U. Todays dra...	160
3	ham	Now only i reached home. . . I am very tired n...	71
4	spam	Guess who am I?This is the first time I create...	152

5.

Compiling the Model

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

6.

Training the Model

```
history = model.fit(sequences_matrix,Y_train,batch_size=0,epochs=10,
          validation_split=0.2)
```

Python

```
Epoch 1/10
112/112 [==============================] - 4s 19ms/step - loss: 0.2037 - accuracy: 0.9324 - val_loss: 0.0518 - val_accuracy: 0.9854
Epoch 2/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0450 - accuracy: 0.9865 - val_loss: 0.0396 - val_accuracy: 0.9888
Epoch 3/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0298 - accuracy: 0.9910 - val_loss: 0.0548 - val_accuracy: 0.9843
Epoch 4/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0172 - accuracy: 0.9955 - val_loss: 0.0414 - val_accuracy: 0.9877
Epoch 5/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0106 - accuracy: 0.9975 - val_loss: 0.0572 - val_accuracy: 0.9821
Epoch 6/10
112/112 [==============================] - 2s 14ms/step - loss: 0.0100 - accuracy: 0.9983 - val_loss: 0.0489 - val_accuracy: 0.9854
Epoch 7/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0056 - accuracy: 0.9986 - val_loss: 0.0511 - val_accuracy: 0.9877
Epoch 8/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0076 - accuracy: 0.9980 - val_loss: 0.0586 - val_accuracy: 0.9888
Epoch 9/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0040 - accuracy: 0.9992 - val_loss: 0.0547 - val_accuracy: 0.9865
Epoch 10/10
112/112 [==============================] - 2s 15ms/step - loss: 0.0034 - accuracy: 0.9994 - val_loss: 0.0658 - val_accuracy: 0.9854
```

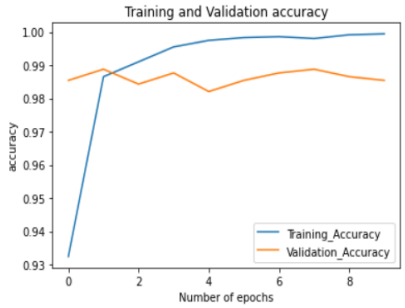
```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy':
'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy':
'Validation_Accuracy'}, inplace = True)
def plot_graphs1(var1, var2, string):
    metrics[[var1, var2]].plot()
    plt.title('Training and Validation ' + string)
    plt.xlabel ('Number of epochs')
    plt.ylabel(string)
    plt.legend([var1, var2])
```

```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation
def plot_graphs1(var1, var2, string):
    metrics[[var1, var2]].plot()
    plt.title('Training and Validation ' + string)
    plt.xlabel ('Number of epochs')
    plt.ylabel(string)
    plt.legend([var1, var2])
```

34] Python

```
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```

35] Python



7.

Saving the Model

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

36]

8.

Preprocessing the Test Dataset

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

37] Python

Testing the Model

```
accuracy1 = model.evaluate(test_sequences_matrix,Y_test)
```

38] Python

```
35/35 [==============================] - 0s 7ms/step - loss: 0.0835 - accuracy: 0.9874
```

```
print(' Accuracy: {:0.3f}'.format(accuracy1[0],accuracy1[1]))
```

39] Python

```
 Accuracy: 0.084
```