Assignment – 4

SMS Spam Classification

Assignment Date	08 October 2022
Student Name	Ranjith S
Student Roll Number	2019504569
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

Solution:

```
Download the dataset from Google Drive
```

2. Importing Libraries

Solution:

```
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 Dataset and Pre-Processing

Solution:

In [2]:
```
df = pd.read_csv('/content/spam.csv',delimiter=',',encoding='latin-1')
df.head()
```

Out[2]:

	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

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

In [4]:
```
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [5]:
```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

In [6]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)
```

In [7]:
```
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)
```

4. Model Creation and Addition of Layers

Solution:

```
In [8]: 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 = Dense(1)(layer)
        layer = Activation('sigmoid')(layer)
        model = Model(inputs=inputs,outputs=layer)

        model.summary()
```

```
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

 dense_1 (Dense)             (None, 1)                 129

 activation_1 (Activation)   (None, 1)                 0

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

5. Model Compilation

Solution:

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

6. Fit the Model

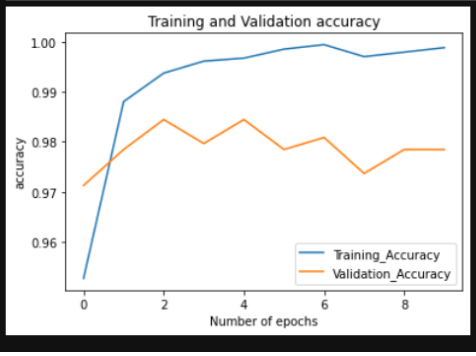
Solution:

```
history = model.fit(sequences_matrix,Y_train,batch_size=20,epochs=10,
          validation_split=0.2)
```

```
Epoch 1/10
168/168 [==============================] - 10s 13ms/step - loss: 0.1588 - accuracy: 0.9527 - val_loss: 0.0922 - val_accuracy: 0.9713
Epoch 2/10
168/168 [==============================] - 2s 13ms/step - loss: 0.0374 - accuracy: 0.9880 - val_loss: 0.0643 - val_accuracy: 0.9785
Epoch 3/10
168/168 [==============================] - 2s 11ms/step - loss: 0.0206 - accuracy: 0.9937 - val_loss: 0.0667 - val_accuracy: 0.9844
Epoch 4/10
168/168 [==============================] - 2s 10ms/step - loss: 0.0120 - accuracy: 0.9961 - val_loss: 0.0841 - val_accuracy: 0.9797
Epoch 5/10
168/168 [==============================] - 2s 10ms/step - loss: 0.0104 - accuracy: 0.9967 - val_loss: 0.0791 - val_accuracy: 0.9844
Epoch 6/10
168/168 [==============================] - 2s 11ms/step - loss: 0.0074 - accuracy: 0.9985 - val_loss: 0.0972 - val_accuracy: 0.9785
Epoch 7/10
168/168 [==============================] - 2s 10ms/step - loss: 0.0031 - accuracy: 0.9994 - val_loss: 0.1078 - val_accuracy: 0.9809
Epoch 8/10
168/168 [==============================] - 2s 11ms/step - loss: 0.0076 - accuracy: 0.9970 - val_loss: 0.1227 - val_accuracy: 0.9737
Epoch 9/10
168/168 [==============================] - 2s 10ms/step - loss: 0.0054 - accuracy: 0.9979 - val_loss: 0.1201 - val_accuracy: 0.9785
Epoch 10/10
168/168 [==============================] - 2s 11ms/step - loss: 0.0031 - accuracy: 0.9988 - val_loss: 0.1830 - val_accuracy: 0.9785
```

In [11]:
```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy': 'Valid
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])
```

In [12]:
```
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```



7. Saving Model

Solution:

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

8. Testing Model

Solution:

Preprocessing the Test Dataset

In [20]:
```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
```

Testing the Model

In [21]:
```
res = model.evaluate(test_sequences_matrix,Y_test)
```

```
44/44 [==============================] - 0s 7ms/step - loss: 0.1179 - accuracy: 0.9813
```