Assignment -4
SMS SPAM Classification

Assignment Date	26 October 2022
Team ID	PNT2022TMID14214
Project Name	AI BASED DISCOURSE FOR BANKING INDUSTRY
Student Name	ACHU SAI GOWTHAM
Student Roll Number	111619104002
Maximum Marks	2 Marks

Question-1. Import required library

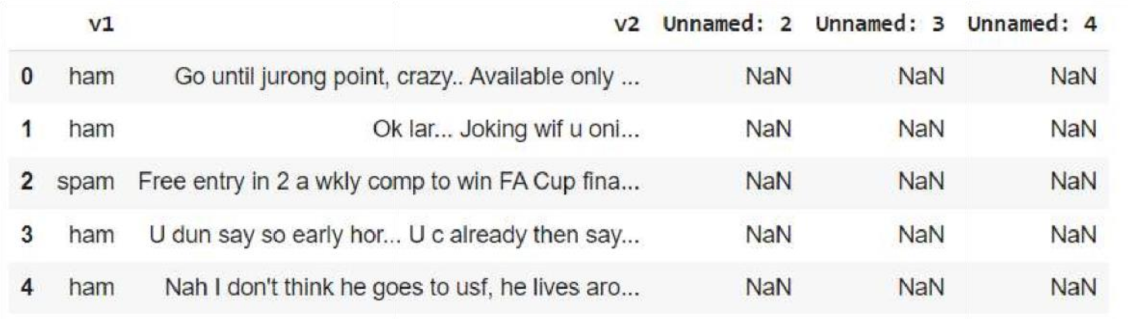
Solution:

```
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
```

Question-2. Read the Dataset

Solution:

```
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

Question-3. Preprocessing the Dataset

Solution:

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

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

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

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.25)

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)
```

Question-4.Create Model

Solution:

```
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)
```

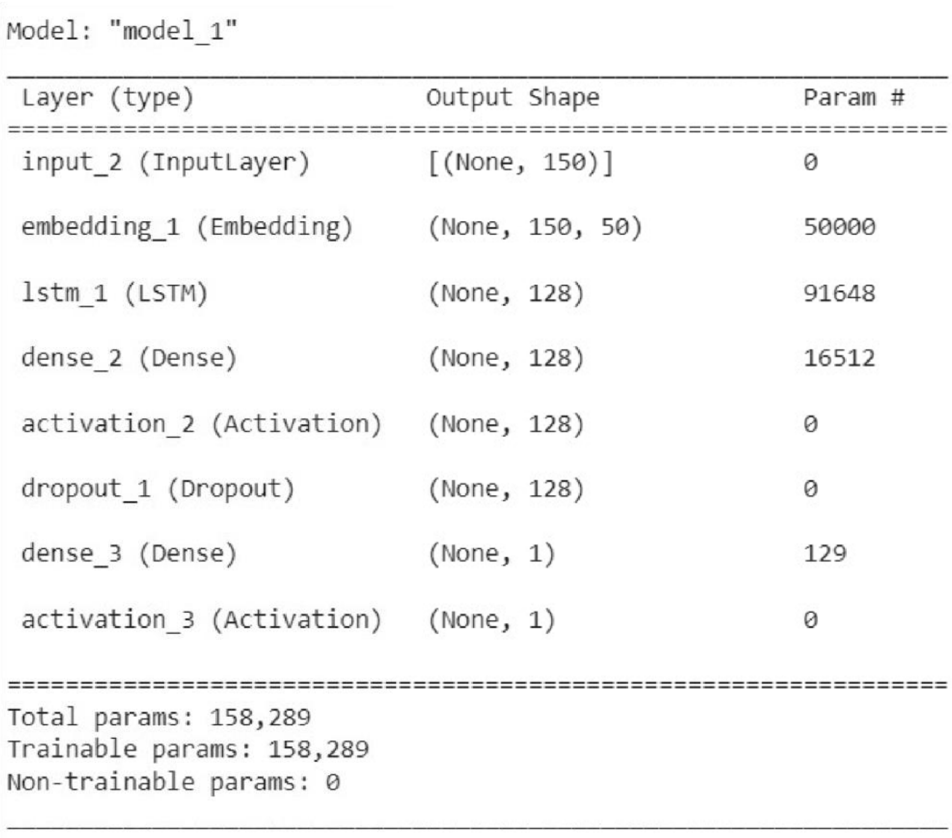
layer = Activation('sigmoid')(layer) model =

Model(inputs=inputs,outputs=layer)

Question-5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

Solution:

model.summary()



```
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
_________________________________________________________________
```

Question-6.Compile the Model

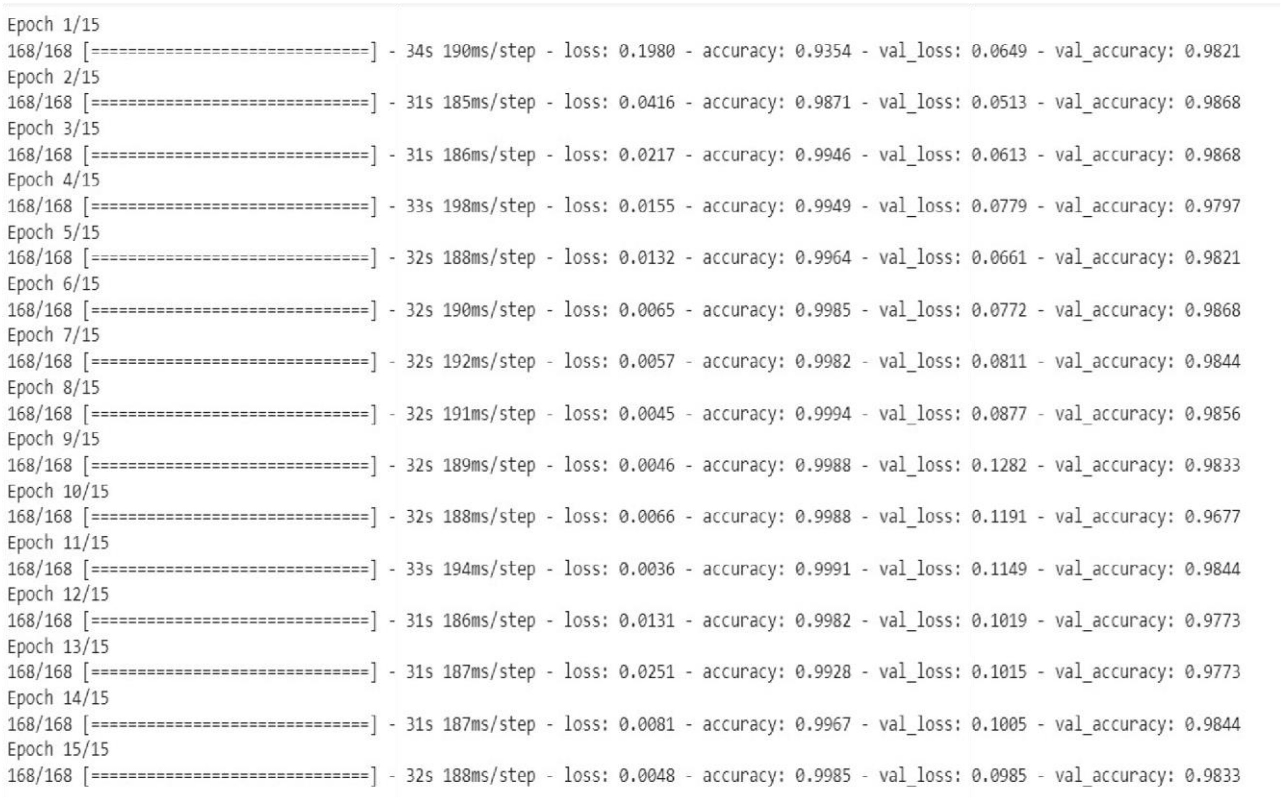
Solution:

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

Question-7. Fit the Model

Solution:

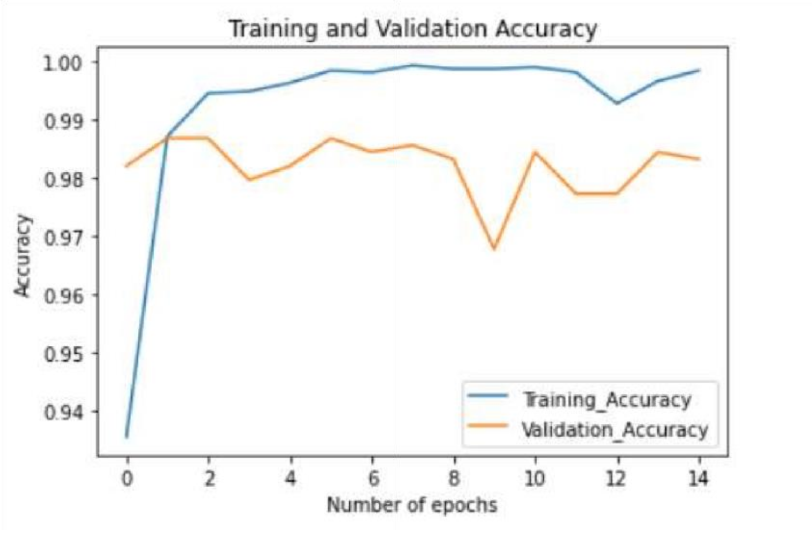
```
history = model.fit(sequences_matrix,Y_train,batch_size=20,epochs=15,
validation_split=0.2)
```



```
Epoch 1/15
168/168 [==============================] - 34s 190ms/step - loss: 0.1980 - accuracy: 0.9354 - val_loss: 0.0649 - val_accuracy: 0.9821
Epoch 2/15
168/168 [==============================] - 31s 185ms/step - loss: 0.0416 - accuracy: 0.9871 - val_loss: 0.0513 - val_accuracy: 0.9868
Epoch 3/15
168/168 [==============================] - 31s 186ms/step - loss: 0.0217 - accuracy: 0.9946 - val_loss: 0.0613 - val_accuracy: 0.9868
Epoch 4/15
168/168 [==============================] - 33s 198ms/step - loss: 0.0155 - accuracy: 0.9949 - val_loss: 0.0779 - val_accuracy: 0.9797
Epoch 5/15
168/168 [==============================] - 32s 188ms/step - loss: 0.0132 - accuracy: 0.9964 - val_loss: 0.0661 - val_accuracy: 0.9821
Epoch 6/15
168/168 [==============================] - 32s 190ms/step - loss: 0.0065 - accuracy: 0.9985 - val_loss: 0.0772 - val_accuracy: 0.9868
Epoch 7/15
168/168 [==============================] - 32s 192ms/step - loss: 0.0057 - accuracy: 0.9982 - val_loss: 0.0811 - val_accuracy: 0.9844
Epoch 8/15
168/168 [==============================] - 32s 191ms/step - loss: 0.0045 - accuracy: 0.9994 - val_loss: 0.0877 - val_accuracy: 0.9856
Epoch 9/15
168/168 [==============================] - 32s 189ms/step - loss: 0.0046 - accuracy: 0.9988 - val_loss: 0.1282 - val_accuracy: 0.9833
Epoch 10/15
168/168 [==============================] - 32s 188ms/step - loss: 0.0066 - accuracy: 0.9988 - val_loss: 0.1191 - val_accuracy: 0.9677
Epoch 11/15
168/168 [==============================] - 33s 194ms/step - loss: 0.0036 - accuracy: 0.9991 - val_loss: 0.1149 - val_accuracy: 0.9844
Epoch 12/15
168/168 [==============================] - 31s 186ms/step - loss: 0.0131 - accuracy: 0.9982 - val_loss: 0.1019 - val_accuracy: 0.9773
Epoch 13/15
168/168 [==============================] - 31s 187ms/step - loss: 0.0251 - accuracy: 0.9928 - val_loss: 0.1015 - val_accuracy: 0.9773
Epoch 14/15
168/168 [==============================] - 31s 187ms/step - loss: 0.0081 - accuracy: 0.9967 - val_loss: 0.1005 - val_accuracy: 0.9844
Epoch 15/15
168/168 [==============================] - 32s 188ms/step - loss: 0.0048 - accuracy: 0.9985 - val_loss: 0.0985 - val_accuracy: 0.9833
```

```
metrics = pd.DataFrame(history.history) metrics.rename(columns = {'loss': 'Training_Loss',
'accuracy': 'Training_Accuracy', 'val_loss': 'Valida tion_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])
```

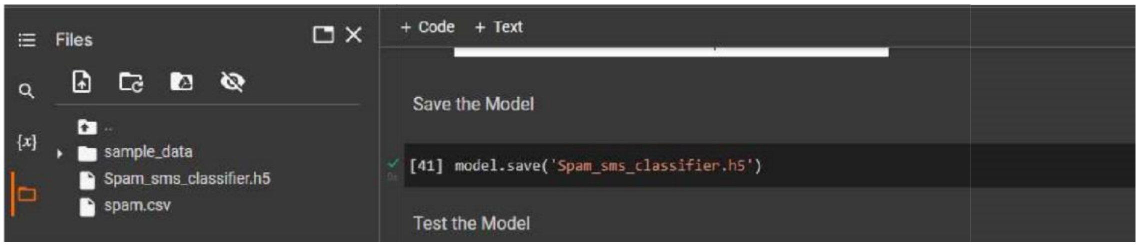
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'Accuracy')



Question-8.Save The Model

Solution:

model.save('Spam_sms_classifier.h5')



Save the Model

```
[41] model.save('Spam_sms_classifier.h5')
```

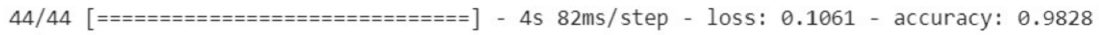
Test the Model

Question-9. Test The Model

Solution:

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

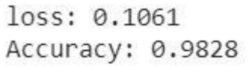
accuracy1 = model.evaluate(test_sequences_matrix,Y_test)



```
44/44 [==============================] - 4s 82ms/step - loss: 0.1061 - accuracy: 0.9828
```

print(' loss: {:0.4f}'.format(accuracy1[0])) print('

Accuracy: {:0.4f}'.format(accuracy1[1]))



```
loss: 0.1061
Accuracy: 0.9828
```