Assignment -4 SMS SPAM Classification

**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.layersimport LSTM,Activation, Dense, Dropout, Input, Embedding from keras.optimizers import Adam fromkeras.preprocessing.textimportTokenizer

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()



**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 =

tok.fit\_on\_texts(X\_tr

Tokenizer(num\_wordain)

s=max

\_

words)

sequences = tok.texts\_to\_sequences(X\_train) sequences\_matrix = pad\_sequenc

es(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)

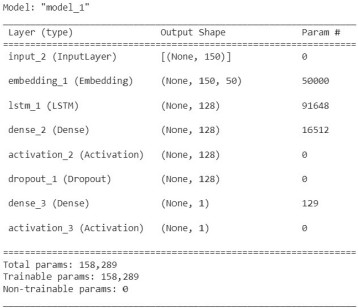
layer = Activation('sigmoid')(layer)

model = Model(inputs=inputs,outputs=layer)

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

Solution:

model.summary()



**Question-6.**Compile the Model

Solution:

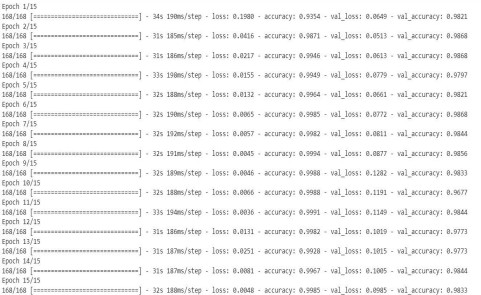
model.compile(loss='binary\_crossentropy',optimizer=Adam(),metrics=['accuracy'])

**Question-7.** Fit the Model

Solution:

history = model.fit(sequences\_matrix,Y\_train,batch\_size=20,epochs=15,

validation\_split=0.2)



metrics = pd.DataFrame(history.history)

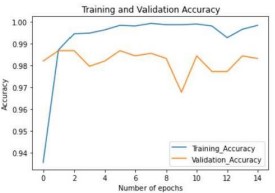
metrics.rename(columns ={'loss':'Training\_Loss', 'accuracy':'Training\_Accuracy', 'val\_loss': 'Valida tion\_Loss', 'val\_accuracy': 'Validation\_Accuracy'}, inplace = True)

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')



**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)



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

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

