**ASSIGNMENT-04**

**Problem Statement :- SMS SPAM Classification**

|  |  |
| --- | --- |
| **Assignment Date** | 5 October 2022 |
| **Student Name** | Bhuvaneshwari M |
| **Student Roll Number** | 113219071004 |
| **Maximum Marks** | 2 Marks |

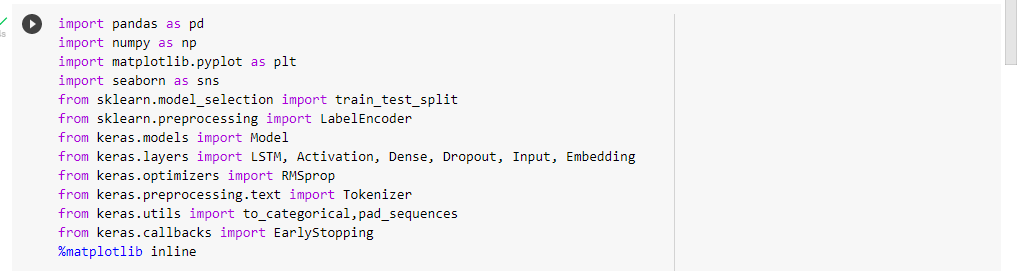
QUESTION 1:

Download the Dataset

Dataset is downloaded and uploaded

QUESTION 2:

Import required library



QUESTION 3:

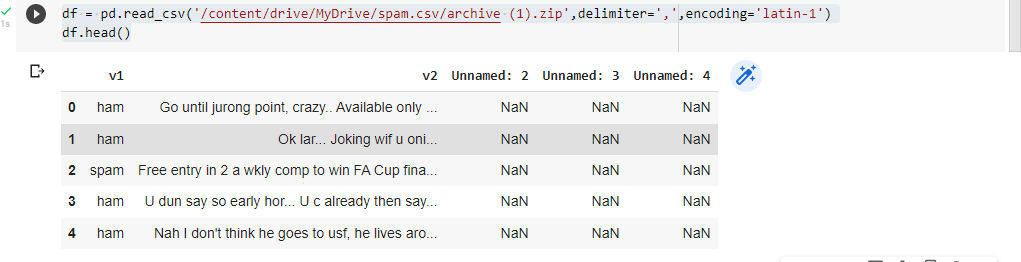
Read dataset and do pre-processing

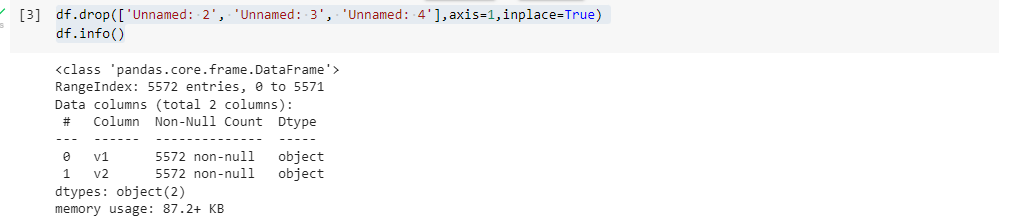
df = pd.read\_csv('/content/drive/MyDrive/spam.csv/archive (1).zip',delimiter=',',encoding='latin-1')

df.head()

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

df.info()





QUESTION 4:

Create model

sns.countplot(df.v1)

plt.xlabel('Label')

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

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=0.15)

max\_words = 1000

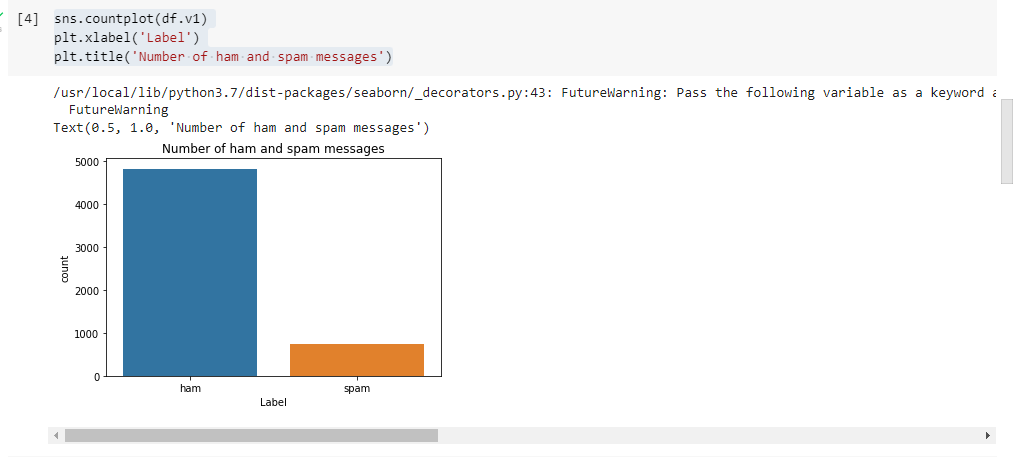
max\_len = 150

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 5:

Add Layers (LSTM, Dense-(Hidden Layers), Output)

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)

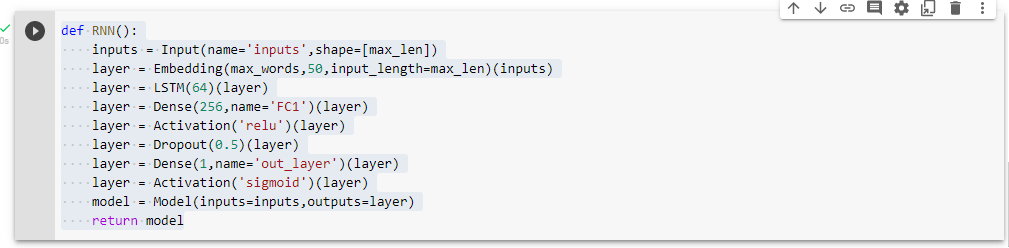
layer = Dropout(0.5)(layer)

layer = Dense(1,name='out\_layer')(layer)

layer = Activation('sigmoid')(layer)

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

return model



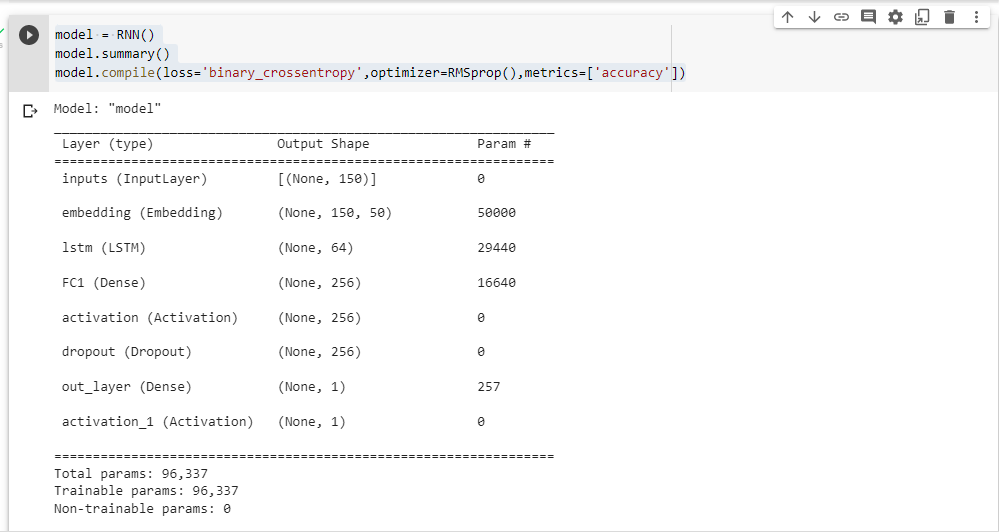
QUESTION 6:

Compile the Model

model = RNN()

model.summary()

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

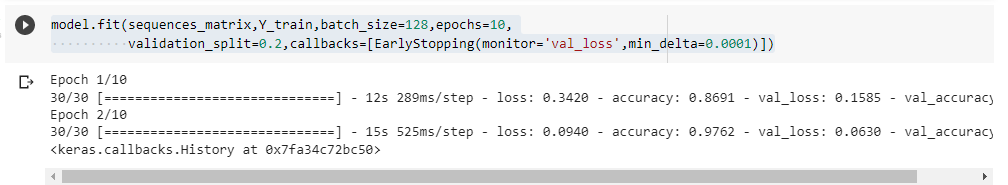


QUESTION 7:

Fit the Model

model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,

validation\_split=0.2,callbacks=[EarlyStopping(monitor='val\_loss',min\_delta=0.0001)])



QUESTION 8:

Test The Model

test\_sequences = tok.texts\_to\_sequences(X\_test)

test\_sequences\_matrix =pad\_sequences(test\_sequences,maxlen=max\_len)

accr = model.evaluate(test\_sequences\_matrix,Y\_test)

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

