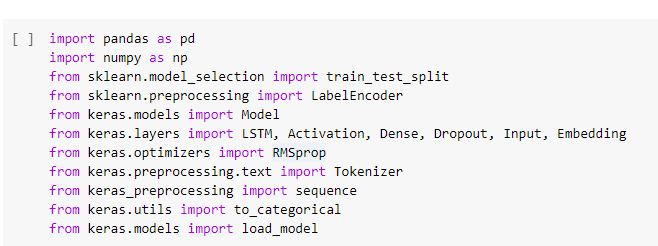
Assignment -4

SMS Spam Classification

| Assignment Date | 08 October 2022 |
| --- | --- |
| Team ID | PNT2022TMID27826 |
| Project Name | AI-powered Nutrition Analyzer for Fitness Enthusiasts |
| Student Name | SANJANA.S |
| Student Roll Number | 311519104050 |
| Maximum Marks | 2 Marks |

# **1.Download the dataset**

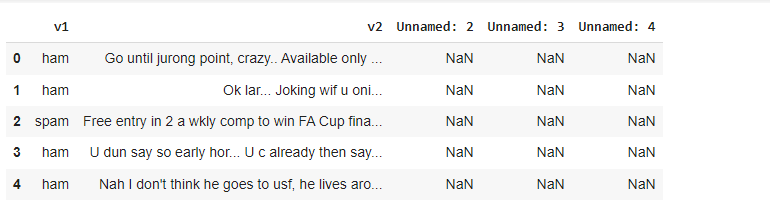
2.Import required library



3.Read Dataset and do preprocessing

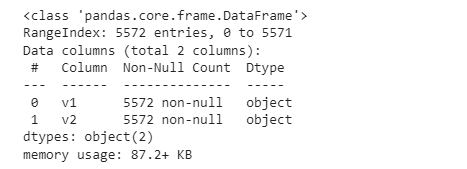
df = pd.read\_csv('/content/spam (1).csv',delimiter=',',encoding='latin-1')

df.head()



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

df.info()



# Count of Spam and Ham values

df.groupby(['v1']).size()



# Test and train split

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.15)

# Label Encoding target column

X = df.v2

Y = df.v1

le = LabelEncoder()

Y = le.fit\_transform(Y)

Y = Y.reshape(-1,1)

# Tokenisation function

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 = sequence.pad\_sequences(sequences,maxlen=max\_len)

4.Create Model and 5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

# Creating LSTM model

inputs = Input(name='InputLayer',shape=[max\_len])

layer = Embedding(max\_words,50,input\_length=max\_len)(inputs)

layer = LSTM(64)(layer)

layer = Dense(256,name='FullyConnectedLayer1')(layer)

layer = Activation('relu')(layer)

layer = Dropout(0.5)(layer)

layer = Dense(1,name='OutputLayer')(layer)

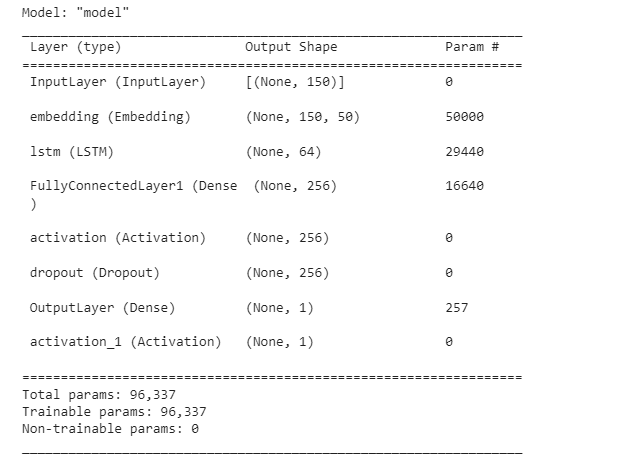
layer = Activation('sigmoid')(layer)

6.Compile the model

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

model.summary()

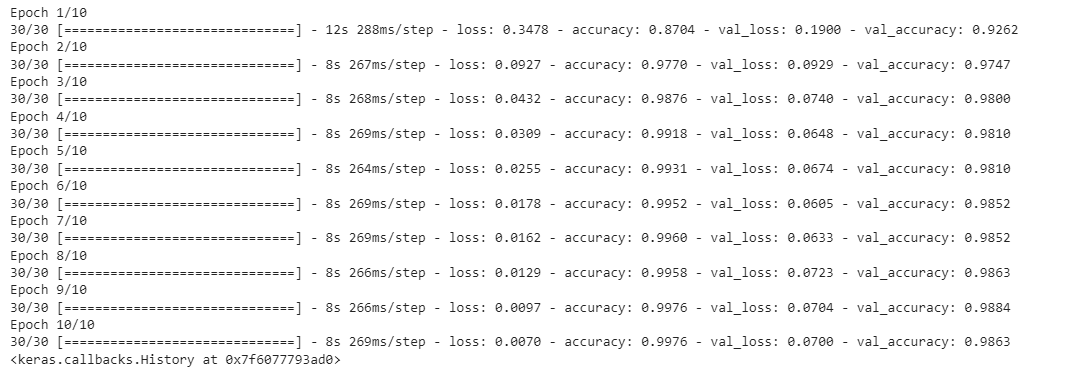
model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])



7.Fit the Model

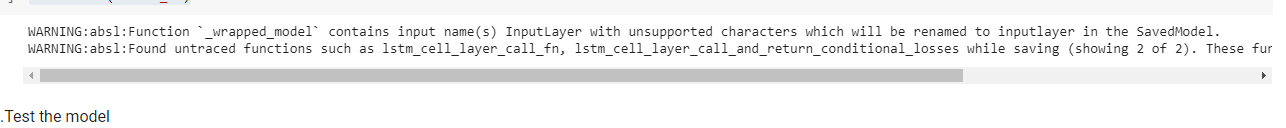
model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,

validation\_split=0.2)



8.Save the Model

model.save("model\_1")



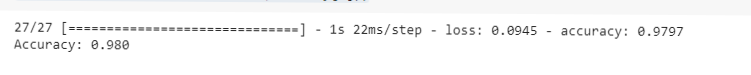
9.Test the model

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

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

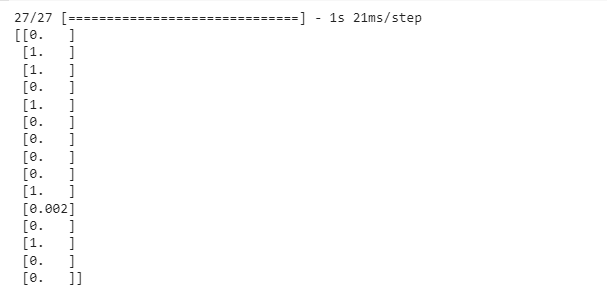
accuracy = model.evaluate(test\_sequences\_matrix,Y\_test)

print('Accuracy: {:0.3f}'.format(accuracy[1]))



y\_pred = model.predict(test\_sequences\_matrix)

print(y\_pred[25:40].round(3))



print(Y\_test[25:40])

