**1.Spam dataset downloaded**

from:[- https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset?resource=download](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset?resource=download)

# 2.Required libararies are imported

In [ ]: **import** numpy **as** np **import** pandas **as** pd **import** keras

**import** matplotlib.pyplot **as** plt **import** seaborn **as** sns **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** RMSprop **from** keras.preprocessing.text **import** Tokenizer **from** keras.preprocessing **import** sequence **from** keras.utils **import** to\_categorical **from** keras.callbacks **import** EarlyStopping

*#from keras.preprocessing.sequence import pad\_sequences*

**%matplotlib** inline

# 3.Read dataset and pre processing

In [ ]: df **=** pd**.**read\_csv('spam.csv',delimiter**=**',',encoding**=**'latin-1') df**.**head()

Out[ ]:

**v1 v2 Unnamed: 2 Unnamed: 3 Unnamed: 4**

1. ham Go until jurong point, crazy.. Available only ... NaN NaN NaN
2. ham Ok lar... Joking wif u oni... NaN NaN NaN
3. spam Free entry in 2 a wkly comp to win FA Cup fina... NaN NaN NaN
4. ham U dun say so early hor... U c already then say... NaN NaN NaN
5. ham Nah I don't think he goes to usf, he lives aro... NaN NaN NaN

drop the unnecessary columns with Nan values

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

In [ ]:

df**.**shape

Out[ ]:

(5572, 2)

In [ ]:

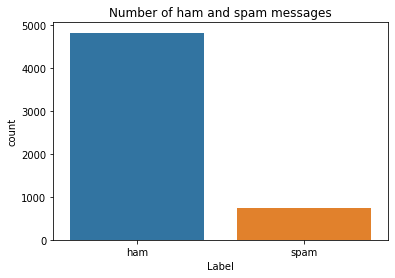
*#plot the ham and spam messages to understand the distribution* sns**.**countplot(df**.**v1) plt**.**xlabel('Label') plt**.**title('Number of ham and spam messages')

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWar ning: Pass the following variable as a keyword arg: x. From version 0.12, t he only valid positional argument will be `data`, and passing other argumen ts without an explicit keyword will result in an error or misinterpretation .

FutureWarning

Out[ ]:

Text(0.5, 1.0, 'Number of ham and spam messages')



In [ ]:

1. **=** df**.**v2
2. **=** df**.**v1

*#label encoding for Y* le **=** LabelEncoder() Y **=** le**.**fit\_transform(Y)

Y **=** Y**.**reshape(**-**1,1)

**Train-test split**

In [ ]:

*#split into train and test sets*

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(X,Y,test\_size**=**0.20)

In [ ]: 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 **=** keras**.**utils**.**pad\_sequences(sequences,maxlen**=**max\_len)

# 4.Create LSTM model, 5.Add layers

In [ ]: 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)

# 6.compile the model

In [ ]: model**.**summary() model**.**compile(loss**=**'binary\_crossentropy',optimizer**=**RMSprop(),metrics**=**['accu racy'])

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # ================================================================= inputs (InputLayer) [(None, 150)] 0 embedding (Embedding) (None, 150, 50) 50000 lstm (LSTM) (None, 64) 29440

FC1 (Dense) (None, 256) 16640 activation (Activation) (None, 256) 0 dropout (Dropout) (None, 256) 0 out\_layer (Dense) (None, 1) 257 activation\_1 (Activation) (None, 1) 0

=================================================================

Total params: 96,337

Trainable params: 96,337

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 7.fit the model

In [ ]: model**.**fit(sequences\_matrix,Y\_train,batch\_size**=**128,epochs**=**10,

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

0.0001)])

Epoch 1/10

28/28 [==============================] - 9s 246ms/step - loss: 0.3549 - acc uracy: 0.8626 - val\_loss: 0.1654 - val\_accuracy: 0.9742

Epoch 2/10

28/28 [==============================] - 4s 153ms/step - loss: 0.0957 - acc uracy: 0.9767 - val\_loss: 0.0468 - val\_accuracy: 0.9821

Out[ ]:

# Save the model

In [ ]: model**.**save('spam\_lstm\_model.h5')

# 9.test the model

In [ ]:

*#processing test data* test\_sequences **=** tok**.**texts\_to\_sequences(X\_test) test\_sequences\_matrix **=**

keras**.**utils**.**pad\_sequences(test\_sequences,maxlen**=**max\_len)

In [ ]:

*#evaluation of our model* accr **=** model**.**evaluate(test\_sequences\_matrix,Y\_test) print('Test set\n Loss: {:0.3f}\n Accuracy: {:0.3f}'**.**format(accr[0],accr[1]))

35/35 [==============================] - 0s 14ms/step - loss: 0.0816 - accu racy: 0.9776 Test set

Loss: 0.082

Accuracy: 0.978