**SMS SPAM CLASSIFICATION**

In [1]:

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

In [2]:

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** LabelEncoder

**from** tensorflow.keras.models **import** Model

**from** tensorflow.keras.layers **import** LSTM, Activation, Dense, Dropout, Input, Embedding

**from** tensorflow.keras.optimizers **import** RMSprop

**from** tensorflow.keras.preprocessing.text **import** Tokenizer

**from** tensorflow.keras.preprocessing **import** sequence

**from** tensorflow.keras.utils **import** to\_categorical

**from** tensorflow.keras.callbacks **import** EarlyStopping

**%**matplotlib inline

**Download the Dataset**

**READ DATASET AND DO PREPROCESSING**

In [3]:

df **=** pd**.**read\_csv(r'spam.csv',encoding**=**'latin-1')

df**.**head()Out[3]:

|  | **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 |

**Remove unwanted data**

In [4]:

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

df**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5572 entries, 0 to 5571

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 v1 5572 non-null object

1 v2 5572 non-null object

dtypes: object(2)

memory usage: 87.2+ KB

**CREATE INPUT VECTORS AND PROCESS LABELS**

In [6]:

X **=** df**.**v2

Y **=** df**.**v1

le **=** LabelEncoder()

Y **=** le**.**fit\_transform(Y)

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

**SPLIT THE TRAINING AND TESTING DATA**

In [7]:

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

**PROCESS THE DATA**

In [8]:

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)

**CREATE MODELS AND ADD LAYERS**

In [9]:

**def** RNN():

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

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

layer **=** LSTM(128)(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('tanh')(layer)

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

**return** model

In [10]:

model **=** RNN()

model**.**summary()

Model: "model"

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

Layer (type) Output Shape Param #

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

inputs (InputLayer) [(None, 150)] 0

embedding (Embedding) (None, 150, 50) 50000

lstm (LSTM) (None, 128) 91648

FC1 (Dense) (None, 256) 33024

activation (Activation) (None, 256) 0

dropout (Dropout) (None, 256) 0

out\_layer (Dense) (None, 1) 257

activation\_1 (Activation) (None, 1) 0

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

Total params: 174,929

Trainable params: 174,929

Non-trainable params: 0

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

In [11]:

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

**FIT THE MODEL**

In [12]:

model**.**fit(sequences\_matrix,Y\_train,batch\_size**=**128,epochs**=**100,

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

Epoch 1/100

28/28 [==============================] - 12s 349ms/step - loss: 0.3517 - accuracy: 0.8867 - mse: 0.0865 - mae: 0.1574 - val\_loss: 0.1572 - val\_accuracy: 0.9540 - val\_mse: 0.0405 - val\_mae: 0.1175

Epoch 2/100

28/28 [==============================] - 9s 332ms/step - loss: 0.0802 - accuracy: 0.9823 - mse: 0.0235 - mae: 0.1035 - val\_loss: 0.1200 - val\_accuracy: 0.9809 - val\_mse: 0.0305 - val\_mae: 0.1298

Out[12]:

<keras.callbacks.History at 0x7f59653822d0>

In [13]:

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

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

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

35/35 [==============================] - 2s 48ms/step - loss: 0.1067 - accuracy: 0.9857 - mse: 0.0268 - mae: 0.1255

In [17]:

print('Test set\nLoss: {:0.3f}\nAccuracy: {:0.3f}'**.**format(accr[0],accr[1]))

Test set

Loss: 0.107

Accuracy: 0.986

**SAVE THE MODEL**

In [14]:

model**.**save(r"Desktop\model\_lSTM.h5")

**TEST THE MODEL**

In [15]:

**from** tensorflow.keras.models **import** load\_model

m2 **=** load\_model(r"Desktop\model\_lSTM.h5")

m2**.**evaluate(test\_sequences\_matrix,Y\_test)

35/35 [==============================] - 2s 47ms/step - loss: 0.1067 - accuracy: 0.9857 - mse: 0.0268 - mae: 0.1255

Out[15]:

[0.10674279928207397,

0.9856502413749695,

0.026829054579138756,

0.12549518048763275]