|  |  |
| --- | --- |
| **Student Name** | SASIDHARAN S |
| **Student Roll no** | 190701193 |

# Spam Classification Import Libraries

In [1]:

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

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

**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 **import** tensorflow

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

# Read CSV File

In [3]:

df**=**pd**.**read\_csv("spam.csv",encoding**=**'ISO-8859-1')

In [4]:

df**.**head()

Out[4]: **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 |

In [5]:

df**.**drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis**=**1, inplace**=True**) df**.**head(10)

Out[5]:

In [6]:

**v1 v2**

1. ham Go until jurong point, crazy.. Available only ...
2. ham Ok lar... Joking wif u oni...
3. spam Free entry in 2 a wkly comp to win FA Cup fina...
4. ham U dun say so early hor... U c already then say...
5. ham Nah I don't think he goes to usf, he lives aro...
6. spam FreeMsg Hey there darling it's been 3 week's n...
7. ham Even my brother is not like to speak with me. ...
8. ham As per your request 'Melle Melle (Oru Minnamin...
9. spam WINNER!! As a valued network customer you have...
10. spam Had your mobile 11 months or more? U R entitle...

# Model Creation

X **=** df**.**v2 Y **=** df**.**v1

le **=** LabelEncoder()

Y **=** le**.**fit\_transform(Y) Y **=** Y**.**reshape(**-**1,1)

In [7]:

X\_train, X\_test, Y\_train, Y\_test **=** train\_test\_split(X, Y, test\_size**=**0.30, random\_state**=**7)

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 **=** pad\_sequences(sequences,maxlen**=**max\_len)

# Adding Layers

In [9]:

**def** RNN\_model():

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

# Model Compilation

In [10]:

model **=** RNN\_model()

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

In [11]:

model**.**summary()

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 |

In [12]:

In [13]:

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

Total params: 96,337

Trainable params: 96,337

Non-trainable params: 0

data **=** model**.**fit(sequences\_matrix, Y\_train, batch\_size**=**16, epochs**=**10, validation\_split**=**0.25)

Epoch 1/10

183/183 [==============================] - 17s 69ms/step - loss: 0.1828 - accuracy: 0.9395 - val\_loss: 0.0687 - val\_accuracy: 0.9815 Epoch 2/10

183/183 [==============================] - 11s 60ms/step - loss: 0.0322 - accuracy: 0.9911 - val\_loss: 0.0539 - val\_accuracy: 0.9867 Epoch 3/10

183/183 [==============================] - 11s 62ms/step - loss: 0.0145 - accuracy: 0.9969 - val\_loss: 0.0602 - val\_accuracy: 0.9856 Epoch 4/10

183/183 [==============================] - 11s 61ms/step - loss: 0.0045 - accuracy: 0.9990 - val\_loss: 0.0876 - val\_accuracy: 0.9867 Epoch 5/10

183/183 [==============================] - 11s 61ms/step - loss: 0.0015 - accuracy: 0.9997 - val\_loss: 0.0857 - val\_accuracy: 0.9846 Epoch 6/10

183/183 [==============================] - 11s 62ms/step - loss: 0.0070 - accuracy: 0.9983 - val\_loss: 0.0934 - val\_accuracy: 0.9877 Epoch 7/10

183/183 [==============================] - 11s 62ms/step - loss: 0.0021 - accuracy: 0.9997 - val\_loss: 0.0818 - val\_accuracy: 0.9836 Epoch 8/10

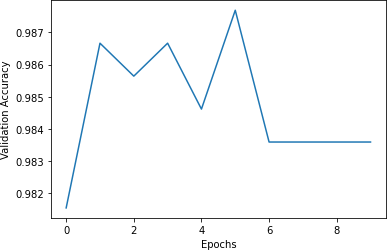
183/183 [==============================] - 11s 61ms/step - loss: 0.0015 - accuracy: 0.9997 - val\_loss: 0.1094 - val\_accuracy: 0.9836 Epoch 9/10

183/183 [==============================] - 11s 61ms/step - loss: 1.7969e-04 - accuracy: 1.0000 - val\_loss: 0.1135 - val\_accuracy: 0.9836 Epoch 10/10

183/183 [==============================] - 11s 62ms/step - loss: 7.7695e-05 - accuracy: 1.0000 - val\_loss: 0.1191 - val\_accuracy: 0.9836

plt**.**figure() plt**.**xlabel('Epochs')

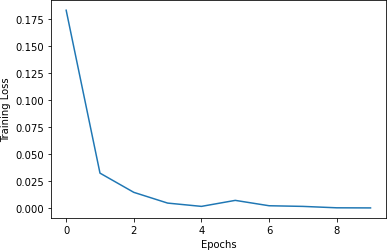
plt**.**ylabel('Validation Accuracy') plt**.**plot(data**.**epoch,data**.**history['val\_accuracy'])

Out[13]: [<matplotlib.lines.Line2D at 0x1f044ed7850>]

In [14]:

plt**.**figure() plt**.**xlabel('Epochs') plt**.**ylabel('Training Loss')

plt**.**plot(data**.**epoch, data**.**history['loss'])

Out[14]: [<matplotlib.lines.Line2D at 0x1f045f88e20>]

In [15]:

In [16]:

# Saving Model

model**.**save('Spam\_Detector\_model.h5')

Testing the Model

test\_sequences **=** tok**.**texts\_to\_sequences(X\_test) test\_sequences\_matrix **=** pad\_sequences(test\_sequences,maxlen**=**max\_len)

In [17]:

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

In [18]:

53/53 [==============================] - 1s 26ms/step - loss: 0.1555 - accuracy: 0.9779

Out[18]: ['loss', 'accuracy']

model**.**metrics\_names

In [19]:

print('Test Loss: {: 0.4f} and Test Accuracy: {: 0.2f}%'**.**format(test\_accuracy[0], test\_accuracy[1]**\***100))

Test Loss: 0.1555 and Test Accuracy: 97.79%