Download the dataset "spam.csv" Import required library

In [44]:

**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.optimizers **import** Adam

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

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

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

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

**from** keras.utils **import** pad\_sequences

**%**matplotlib inline

Read the data set

In [6]:

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

df**.**head()

Out[6]:

|  | **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 |

Preprocessing the dataset

In [7]:

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

df**.**info()

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

In [8]:

**from** wordcloud **import** WordCloud, STOPWORDS, ImageColorGenerator

In [9]:

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

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

le **=** LabelEncoder()

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

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

In [14]:

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(X,Y,test\_size**=**0.25)

In [39]:

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)

Create Model and Add Layers

In [15]:

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

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

layer **=** LSTM(128)(layer)

layer **=** Dense(128)(layer)

layer **=** Activation('relu')(layer)

layer **=** Dropout(0.5)(layer)

layer **=** Dense(1)(layer)

layer **=** Activation('sigmoid')(layer)

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

In [16]:

model**.**summary()

Model: "model"

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

Layer (type) Output Shape Param #

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

input\_1 (InputLayer) [(None, 150)] 0

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

lstm (LSTM) (None, 128) 91648

dense (Dense) (None, 128) 16512

activation (Activation) (None, 128) 0

dropout (Dropout) (None, 128) 0

dense\_1 (Dense) (None, 1) 129

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

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

Total params: 158,289

Trainable params: 158,289

Non-trainable params: 0

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

Compile the Model

In [45]:

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

Fit the Model

In [46]:

history **=** model**.**fit(sequences\_matrix,Y\_train,batch\_size**=**20,epochs**=**15,

validation\_split**=**0.2)

Epoch 1/15

168/168 [==============================] - 37s 204ms/step - loss: 0.1812 - accuracy: 0.9408 - val\_loss: 0.0619 - val\_accuracy: 0.9821

Epoch 2/15

168/168 [==============================] - 38s 229ms/step - loss: 0.0452 - accuracy: 0.9877 - val\_loss: 0.0522 - val\_accuracy: 0.9809

Epoch 3/15

168/168 [==============================] - 34s 200ms/step - loss: 0.0247 - accuracy: 0.9925 - val\_loss: 0.0441 - val\_accuracy: 0.9868

Epoch 4/15

168/168 [==============================] - 34s 201ms/step - loss: 0.0104 - accuracy: 0.9976 - val\_loss: 0.0570 - val\_accuracy: 0.9880

Epoch 5/15

168/168 [==============================] - 34s 201ms/step - loss: 0.0061 - accuracy: 0.9991 - val\_loss: 0.0652 - val\_accuracy: 0.9868

Epoch 6/15

168/168 [==============================] - 33s 199ms/step - loss: 0.0144 - accuracy: 0.9970 - val\_loss: 0.0574 - val\_accuracy: 0.9856

Epoch 7/15

168/168 [==============================] - 34s 200ms/step - loss: 0.0045 - accuracy: 0.9991 - val\_loss: 0.0532 - val\_accuracy: 0.9892

Epoch 8/15

168/168 [==============================] - 35s 209ms/step - loss: 0.0048 - accuracy: 0.9988 - val\_loss: 0.0613 - val\_accuracy: 0.9844

Epoch 9/15

168/168 [==============================] - 33s 198ms/step - loss: 0.0023 - accuracy: 0.9994 - val\_loss: 0.0771 - val\_accuracy: 0.9821

Epoch 10/15

168/168 [==============================] - 33s 197ms/step - loss: 0.0095 - accuracy: 0.9997 - val\_loss: 0.0797 - val\_accuracy: 0.9856

Epoch 11/15

168/168 [==============================] - 33s 198ms/step - loss: 1.5724e-04 - accuracy: 1.0000 - val\_loss: 0.0860 - val\_accuracy: 0.9856

Epoch 12/15

168/168 [==============================] - 34s 200ms/step - loss: 9.8532e-05 - accuracy: 1.0000 - val\_loss: 0.0889 - val\_accuracy: 0.9856

Epoch 13/15

168/168 [==============================] - 35s 209ms/step - loss: 9.1964e-05 - accuracy: 1.0000 - val\_loss: 0.0916 - val\_accuracy: 0.9844

Epoch 14/15

168/168 [==============================] - 33s 198ms/step - loss: 5.8093e-05 - accuracy: 1.0000 - val\_loss: 0.0953 - val\_accuracy: 0.9844

Epoch 15/15

168/168 [==============================] - 33s 198ms/step - loss: 3.9150e-05 - accuracy: 1.0000 - val\_loss: 0.0993 - val\_accuracy: 0.9844

In [47]:

metrics **=** pd**.**DataFrame(history**.**history)

metrics**.**rename(columns **=** {'loss': 'Training\_Loss', 'accuracy': 'Training\_Accuracy', 'val\_loss': 'Validation\_Loss', 'val\_accuracy': 'Validation\_Accuracy'}, inplace **=** **True**)

**def** plot\_graphs1(var1, var2, string):

metrics[[var1, var2]]**.**plot()

plt**.**title('Training and Validation ' **+** string)

plt**.**xlabel ('Number of epochs')

plt**.**ylabel(string)

plt**.**legend([var1, var2])

Save The Model

In [48]:

model**.**save('A4Spam\_sms\_classifier.h5')

In [49]:

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

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

In [50]:

accuracy1 **=** model**.**evaluate(test\_sequences\_matrix,Y\_test)

44/44 [==============================] - 4s 83ms/step - loss: 0.1532 - accuracy: 0.9821

In [51]:

print(' loss: {:0.4f}'**.**format(accuracy1[0]))

print(' Accuracy: {:0.4f}'**.**format(accuracy1[1]))

loss: 0.1532

Accuracy: 0.9821

Download the dataset "spam.csv" Import required library

In [44]:

**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.optimizers **import** Adam

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

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

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

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

**from** keras.utils **import** pad\_sequences

**%**matplotlib inline

Read the data set

In [6]:

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

df**.**head()

Out[6]:

|  | **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 |

Preprocessing the dataset

In [7]:

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

df**.**info()

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

In [8]:

**from** wordcloud **import** WordCloud, STOPWORDS, ImageColorGenerator

In [9]:

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

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

le **=** LabelEncoder()

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

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

In [14]:

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(X,Y,test\_size**=**0.25)

In [39]:

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)

Create Model and Add Layers

In [15]:

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

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

layer **=** LSTM(128)(layer)

layer **=** Dense(128)(layer)

layer **=** Activation('relu')(layer)

layer **=** Dropout(0.5)(layer)

layer **=** Dense(1)(layer)

layer **=** Activation('sigmoid')(layer)

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

In [16]:

model**.**summary()

Model: "model"

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

Layer (type) Output Shape Param #

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

input\_1 (InputLayer) [(None, 150)] 0

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

lstm (LSTM) (None, 128) 91648

dense (Dense) (None, 128) 16512

activation (Activation) (None, 128) 0

dropout (Dropout) (None, 128) 0

dense\_1 (Dense) (None, 1) 129

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

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

Total params: 158,289

Trainable params: 158,289

Non-trainable params: 0

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

Compile the Model

In [45]:

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

Fit the Model

In [46]:

history **=** model**.**fit(sequences\_matrix,Y\_train,batch\_size**=**20,epochs**=**15,

validation\_split**=**0.2)

Epoch 1/15

168/168 [==============================] - 37s 204ms/step - loss: 0.1812 - accuracy: 0.9408 - val\_loss: 0.0619 - val\_accuracy: 0.9821

Epoch 2/15

168/168 [==============================] - 38s 229ms/step - loss: 0.0452 - accuracy: 0.9877 - val\_loss: 0.0522 - val\_accuracy: 0.9809

Epoch 3/15

168/168 [==============================] - 34s 200ms/step - loss: 0.0247 - accuracy: 0.9925 - val\_loss: 0.0441 - val\_accuracy: 0.9868

Epoch 4/15

168/168 [==============================] - 34s 201ms/step - loss: 0.0104 - accuracy: 0.9976 - val\_loss: 0.0570 - val\_accuracy: 0.9880

Epoch 5/15

168/168 [==============================] - 34s 201ms/step - loss: 0.0061 - accuracy: 0.9991 - val\_loss: 0.0652 - val\_accuracy: 0.9868

Epoch 6/15

168/168 [==============================] - 33s 199ms/step - loss: 0.0144 - accuracy: 0.9970 - val\_loss: 0.0574 - val\_accuracy: 0.9856

Epoch 7/15

168/168 [==============================] - 34s 200ms/step - loss: 0.0045 - accuracy: 0.9991 - val\_loss: 0.0532 - val\_accuracy: 0.9892

Epoch 8/15

168/168 [==============================] - 35s 209ms/step - loss: 0.0048 - accuracy: 0.9988 - val\_loss: 0.0613 - val\_accuracy: 0.9844

Epoch 9/15

168/168 [==============================] - 33s 198ms/step - loss: 0.0023 - accuracy: 0.9994 - val\_loss: 0.0771 - val\_accuracy: 0.9821

Epoch 10/15

168/168 [==============================] - 33s 197ms/step - loss: 0.0095 - accuracy: 0.9997 - val\_loss: 0.0797 - val\_accuracy: 0.9856

Epoch 11/15

168/168 [==============================] - 33s 198ms/step - loss: 1.5724e-04 - accuracy: 1.0000 - val\_loss: 0.0860 - val\_accuracy: 0.9856

Epoch 12/15

168/168 [==============================] - 34s 200ms/step - loss: 9.8532e-05 - accuracy: 1.0000 - val\_loss: 0.0889 - val\_accuracy: 0.9856

Epoch 13/15

168/168 [==============================] - 35s 209ms/step - loss: 9.1964e-05 - accuracy: 1.0000 - val\_loss: 0.0916 - val\_accuracy: 0.9844

Epoch 14/15

168/168 [==============================] - 33s 198ms/step - loss: 5.8093e-05 - accuracy: 1.0000 - val\_loss: 0.0953 - val\_accuracy: 0.9844

Epoch 15/15

168/168 [==============================] - 33s 198ms/step - loss: 3.9150e-05 - accuracy: 1.0000 - val\_loss: 0.0993 - val\_accuracy: 0.9844

In [47]:

metrics **=** pd**.**DataFrame(history**.**history)

metrics**.**rename(columns **=** {'loss': 'Training\_Loss', 'accuracy': 'Training\_Accuracy', 'val\_loss': 'Validation\_Loss', 'val\_accuracy': 'Validation\_Accuracy'}, inplace **=** **True**)

**def** plot\_graphs1(var1, var2, string):

metrics[[var1, var2]]**.**plot()

plt**.**title('Training and Validation ' **+** string)

plt**.**xlabel ('Number of epochs')

plt**.**ylabel(string)

plt**.**legend([var1, var2])

Save The Model

In [48]:

model**.**save('A4Spam\_sms\_classifier.h5')

In [49]:

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

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

In [50]:

accuracy1 **=** model**.**evaluate(test\_sequences\_matrix,Y\_test)

44/44 [==============================] - 4s 83ms/step - loss: 0.1532 - accuracy: 0.9821

In [51]:

print(' loss: {:0.4f}'**.**format(accuracy1[0]))

print(' Accuracy: {:0.4f}'**.**format(accuracy1[1]))

loss: 0.1532

Accuracy: 0.9821