Dataset Importing

**from** google.colab **import** drive

drive**.**mount('/content/drive')

Mounted at /content/drive

In [2]:

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

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

print(dataset**.**head())

print(dataset**.**info())

v1 v2 Unnamed: 2 \

0 ham Go until jurong point, crazy.. Available only ... NaN

1 ham Ok lar... Joking wif u oni... NaN

2 spam Free entry in 2 a wkly comp to win FA Cup fina... NaN

3 ham U dun say so early hor... U c already then say... NaN

4 ham Nah I don't think he goes to usf, he lives aro... NaN

Unnamed: 3 Unnamed: 4

0 NaN NaN

1 NaN NaN

2 NaN NaN

3 NaN NaN

4 NaN NaN

RangeIndex: 5572 entries, 0 to 5571

Data columns (total 5 columns):

# Column Non-Null Count Dtype

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

0 v1 5572 non-null object

1 v2 5572 non-null object

2 Unnamed: 2 50 non-null object

3 Unnamed: 3 12 non-null object

4 Unnamed: 4 6 non-null object

dtypes: object(5)

memory usage: 217.8+ KB

None

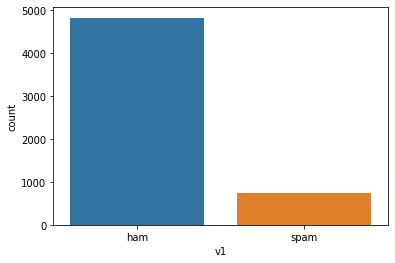
Importing libraries ,Reading dataset and doing pre-processing

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

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

In [4]:

sns**.**countplot(data**=**dataset, x**=**dataset['v1'])



text **=** dataset**.**loc[:, 'v2']

classification **=** dataset**.**loc[:, 'v1']

print(text)

print(classification)

0 Go until jurong point, crazy.. Available only ...

1 Ok lar... Joking wif u oni...

2 Free entry in 2 a wkly comp to win FA Cup fina...

3 U dun say so early hor... U c already then say...

4 Nah I don't think he goes to usf, he lives aro...

...

5567 This is the 2nd time we have tried 2 contact u...

5568 Will Ì\_ b going to esplanade fr home?

5569 Pity, \* was in mood for that. So...any other s...

5570 The guy did some bitching but I acted like i'd...

5571 Rofl. Its true to its name

Name: v2, Length: 5572, dtype: object

0 ham

1 ham

2 spam

3 ham

4 ham

...

5567 spam

5568 ham

5569 ham

5570 ham

5571 ham

Name: v1, Length: 5572, dtype: object

In [10]:

**from** nltk **import** word\_tokenize

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

**import** nltk

nltk**.**download('punkt')

[nltk\_data] Downloading package punkt to /root/nltk\_data...

[nltk\_data] Unzipping tokenizers/punkt.zip.

Out[10]:

True

In [11]:

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(text, classification, test\_size**=**0.2, random\_state**=**42)

In [12]:

text\_length **=** []

**for** i **in** x\_train :

text\_length**.**append(len(word\_tokenize(i)))

In [13]:

print(max(text\_length))

220

In [14]:

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

In [15]:

max\_sequence\_length **=** 38

tok **=** Tokenizer()

tok**.**fit\_on\_texts(x\_train**.**values)

In [16]:

vocab\_length **=** len(tok**.**word\_index)

In [17]:

vocab\_length **=** len(tok**.**word\_index)

In [18]:

x\_train\_sequences **=** tok**.**texts\_to\_sequences(x\_train**.**values)

x\_test\_sequences **=** tok**.**texts\_to\_sequences(x\_test**.**values)

In [19]:

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

In [22]:

x\_train **=** pad\_sequences(x\_train\_sequences, maxlen**=**max\_sequence\_length)

x\_test **=** pad\_sequences(x\_test\_sequences, maxlen**=**max\_sequence\_length)

In [21]:

x\_train[:2]

Out[21]:

array([[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 38, 30, 8,

5, 273, 1989, 81, 116, 26, 11, 1656, 322, 10, 53,

18, 299, 30, 349, 1990],

[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 799, 15, 2555, 1442, 1127, 192, 2556,

171, 12, 98, 1991, 44, 195, 1657, 2557, 1992, 2558, 21,

9, 4, 203, 1025, 225]], dtype=int32)

In [23]:

y\_train**.**values

Out[23]:

array(['ham', 'spam', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)

In [24]:

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

le **=** LabelEncoder()

y\_train **=** le**.**fit\_transform(y\_train)

y\_test **=** le**.**fit\_transform(y\_test)

print(y\_train)

[0 1 0 ... 0 0 0]

In [26]:

**from** keras.models **import** Model, load\_model

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

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

Creating models and Adding layers

In [27]:

**def** create\_model(vocab\_len, max\_seq\_len):

inputs **=** Input(name**=**'inputs', shape**=**[max\_seq\_len]) *#None, 150*

layer **=** Embedding(vocab\_length **+** 1, 50, input\_length**=**max\_seq\_len)(inputs) *#None, 150, 50*

layer **=** LSTM(64)(layer) *#None, 64*

layer **=** Dense(256,name**=**'FC1')(layer) *#None, 256*

layer **=** Activation('relu')(layer) *#None, 256*

layer **=** Dropout(0.5)(layer) *#None, 256*

layer **=** Dense(1,name**=**'out\_layer')(layer) *#None, 1*

layer **=** Activation('sigmoid')(layer) *#None, 1*

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

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

**return** model

model **=** create\_model(vocab\_length, max\_sequence\_length)

model**.**summary()

Model: "model"

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

Layer (type) Output Shape Param #

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

inputs (InputLayer) [(None, 38)] 0

embedding (Embedding) (None, 38, 50) 397750

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: 444,087

Trainable params: 444,087

Non-trainable params: 0

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

Compiling model

In [28]:

**from** keras.callbacks **import** EarlyStopping, ModelCheckpoint, TensorBoard

In [29]:

history **=** model**.**fit(x\_train, y\_train, batch\_size**=**128, epochs**=**20, validation\_split**=**0.2)

Epoch 1/20

28/28 [==============================] - 5s 98ms/step - loss: 0.2984 - acc: 0.8741 - val\_loss: 0.1544 - val\_acc: 0.9552

Epoch 2/20

28/28 [==============================] - 2s 74ms/step - loss: 0.0803 - acc: 0.9820 - val\_loss: 0.0573 - val\_acc: 0.9821

Epoch 3/20

28/28 [==============================] - 2s 75ms/step - loss: 0.0268 - acc: 0.9924 - val\_loss: 0.0419 - val\_acc: 0.9865

Epoch 4/20

28/28 [==============================] - 3s 98ms/step - loss: 0.0151 - acc: 0.9961 - val\_loss: 0.0412 - val\_acc: 0.9843

Epoch 5/20

28/28 [==============================] - 2s 75ms/step - loss: 0.0083 - acc: 0.9969 - val\_loss: 0.0678 - val\_acc: 0.9843

Epoch 6/20

28/28 [==============================] - 2s 74ms/step - loss: 0.0052 - acc: 0.9983 - val\_loss: 0.0690 - val\_acc: 0.9854

Epoch 7/20

28/28 [==============================] - 2s 75ms/step - loss: 4.3604e-04 - acc: 1.0000 - val\_loss: 0.0707 - val\_acc: 0.9865

Epoch 8/20

28/28 [==============================] - 2s 75ms/step - loss: 4.3695e-05 - acc: 1.0000 - val\_loss: 0.0848 - val\_acc: 0.9888

Epoch 9/20

28/28 [==============================] - 2s 74ms/step - loss: 0.0029 - acc: 0.9994 - val\_loss: 0.0913 - val\_acc: 0.9798

Epoch 10/20

28/28 [==============================] - 2s 74ms/step - loss: 2.9656e-04 - acc: 1.0000 - val\_loss: 0.0992 - val\_acc: 0.9832

Epoch 11/20

28/28 [==============================] - 2s 76ms/step - loss: 1.8744e-05 - acc: 1.0000 - val\_loss: 0.1156 - val\_acc: 0.9854

Epoch 12/20

28/28 [==============================] - 2s 76ms/step - loss

0.1156 - val\_acc: 0.9854

Epoch 12/20

28/28 [==============================] - 2s 76ms/step - loss: 2.5507e-06 - acc: 1.0000 - val\_loss: 0.1101 - val\_acc: 0.9888

Epoch 13/20

28/28 [==============================] - 2s 75ms/step - loss: 1.1500e-06 - acc: 1.0000 - val\_loss: 0.1304 - val\_acc: 0.9865

Epoch 14/20

28/28 [==============================] - 2s 75ms/step - loss: 3.2071e-07 - acc: 1.0000 - val\_loss: 0.2487 - val\_acc: 0.9821

Epoch 15/20

28/28 [==============================] - 2s 74ms/step - loss: 0.0046 - acc: 0.9994 - val\_loss: 0.0985 - val\_acc: 0.9832

Epoch 16/20

28/28 [==============================] - 2s 76ms/step - loss: 1.4251e-04 - acc: 1.0000 - val\_loss: 0.1183 - val\_acc: 0.9854

Epoch 17/20

28/28 [==============================] - 2s 75ms/step - loss: 8.2607e-06 - acc: 1.0000 - val\_loss: 0.1241 - val\_acc: 0.9854

Epoch 18/20

28/28 [==============================] - 2s 74ms/step - loss: 3.0136e-06 - acc: 1.0000 - val\_loss: 0.1288 - val\_acc: 0.9843

Epoch 19/20

28/28 [==============================] - 2s 75ms/step - loss: 1.9952e-06 - acc: 1.0000 - val\_loss: 0.1374 - val\_acc: 0.9843

Epoch 20/20

28/28 [==============================] - 2s 75ms/step - loss: 4.4795e-07 - acc: 1.0000 - val\_loss: 0.1438 - val\_acc: 0.9854

Fitting and Saving the model

In [30]:

history\_dict **=** history**.**history

*# list all data in history*

print(history\_dict**.**keys())

*# summarize history for loss*

plt**.**plot(history\_dict['loss'])

plt**.**plot(history\_dict['val\_loss'])

plt**.**title('Training and Validation Loss')

plt**.**ylabel('loss')

plt**.**xlabel('epoch')

plt**.**legend(['train', 'test'], loc**=**'upper left')

plt**.**show()

*# summarize history for accuracy*

plt**.**plot(history\_dict['acc'])

plt**.**plot(history\_dict['val\_acc'])

plt**.**title('Training and Validation Accuracy')

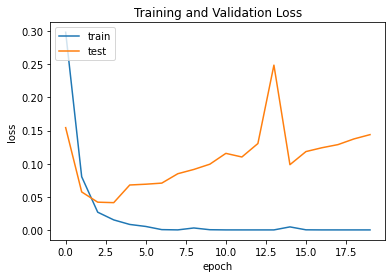
plt**.**ylabel('accuracy')

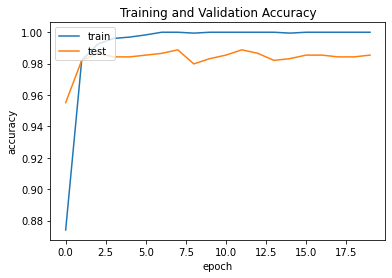
plt**.**xlabel('epoch')

plt**.**legend(['train', 'test'], loc**=**'upper left')

plt**.**show()

dict\_keys(['loss', 'acc', 'val\_loss', 'val\_acc'])





model**.**save('/content/drive/MyDrive/spam.h5')

Testing the model

In [34]:

loaded\_model **=** load\_model('/content/drive/MyDrive/spam.h5')

test\_loss, test\_acc **=** accr **=** loaded\_model**.**evaluate(x\_test, y\_test)

print('Test set\n Loss: {:0.3f}\n Accuracy: {:0.3f}'**.**format(test\_loss, test\_acc))

35/35 [==============================] - 1s 8ms/step - loss: 0.1899 - acc: 0.9848

Test set

Loss: 0.190

Accuracy: 0.985

In [35]:

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

In [36]:

y\_pred\_prob **=** loaded\_model**.**predict(x\_test)

print(np**.**round(y\_pred\_prob, 3))

y\_pred **=** y\_pred\_prob **>** 0.5

y\_pred

35/35 [==============================] - 1s 9ms/step

[[0.006]

[0. ]

[1. ]

...

[0. ]

[0. ]

[1. ]]

Out[36]:

array([[False],

[False],

[ True],

...,

[False],

[False],

[ True]])

In [37]:

**for** i **in** range(5):

print('%s => %d (expected %d)' **%** (x\_test[i]**.**tolist(), y\_pred[i], y\_test[i]))

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1366, 1578, 1432, 19, 7893, 19, 19, 38, 118, 1650, 19, 738, 4, 449, 3023, 35, 1285] => 0 (expected 0)

[1, 188, 11, 6440, 2, 7, 1, 135, 2, 28, 12, 4, 290, 7931, 1, 104, 33, 3, 22, 647, 15, 28, 4, 3607, 18, 374, 191, 224, 2137, 107, 433, 9, 74, 10, 5, 1097, 1806, 1171] => 0 (expected 0)

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 39, 54, 258, 144, 3, 54, 21, 3428, 3, 16, 2, 173, 53, 144, 761, 264, 7182, 208] => 1 (expected 1)

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 64, 33, 3, 1528, 13, 263, 53, 79, 228, 79, 3, 31, 7, 838, 69, 10, 8, 5, 168, 2, 205, 10, 54, 3, 499, 14, 8, 46] => 0 (expected 0)

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 715, 29, 357, 532, 622, 15, 1107, 528, 706, 49, 435, 19, 98, 563, 496, 292, 71, 521, 2, 906, 1546, 138, 1200, 2216] => 1 (expected 1)

In [38]:

**from** sklearn.metrics **import** classification\_report

In [39]:

print(classification\_report(y\_test, y\_pred))

precision recall f1-score support

0 0.98 1.00 0.99 965

1 1.00 0.89 0.94 150

accuracy 0.98 1115

macro avg 0.99 0.94 0.97 1115

weighted avg 0.99 0.98 0.98 1115