First Step: the data loaded and The Labels changed to Class and Text.
describing data shows that only about 15% of the text messages is classified as a spam.

```
In [95]: dataset.head()
```

Out[95]:

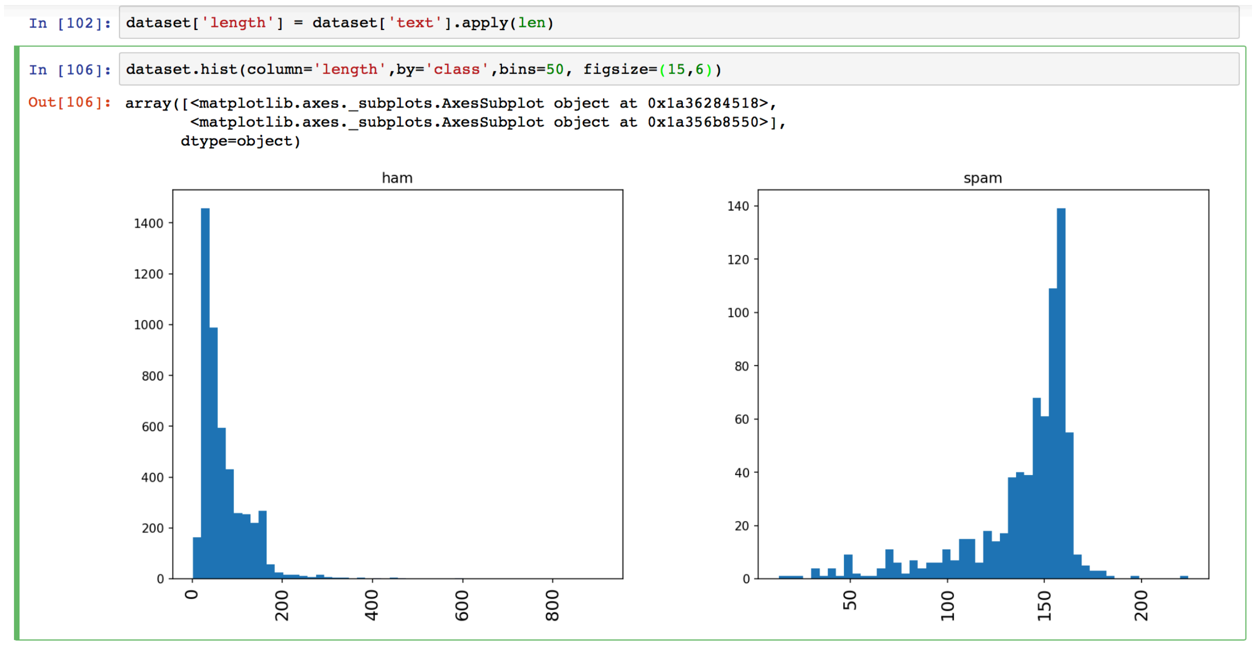
	class	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
In [96]: dataset.groupby('class').describe()
```

Out[96]:

	text			
	count	**unique**	**top**	**freq**
class				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

next A histogram of message length separated by class shows the correlation of the length of each text messages with the text classified as a spam or not.

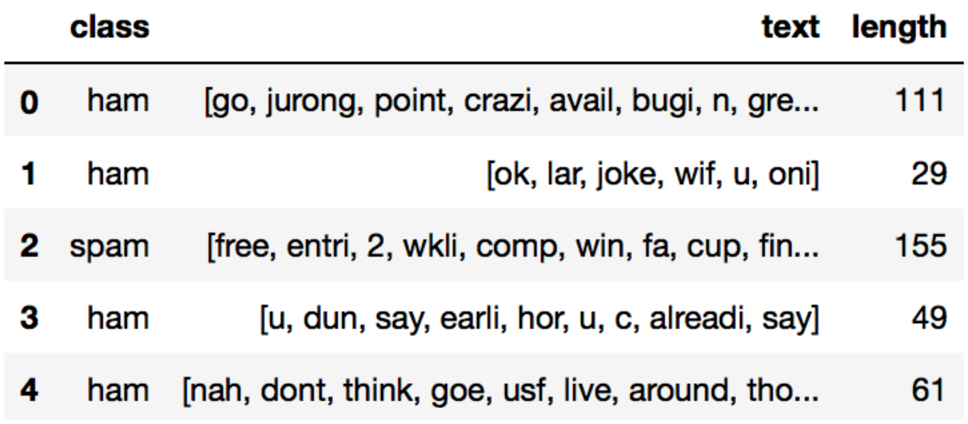


```
In [102]: dataset['length'] = dataset['text'].apply(len)

In [106]: dataset.hist(column='length',by='class',bins=50, figsize=(15,6))

Out[106]: array([<matplotlib.axes._subplots.AxesSubplot object at 0x1a36284518>,
                 <matplotlib.axes._subplots.AxesSubplot object at 0x1a356b8550>],
                dtype=object)
```

From above figure we understand that the ham messages have length about 100 and the spam messages have higher length above 130 or 140 approximately.

next we use text mining technics such as stemming and TF_IDF and finally Naive Bayes classifier which has implemented by scikit as MultinomialNB Class.

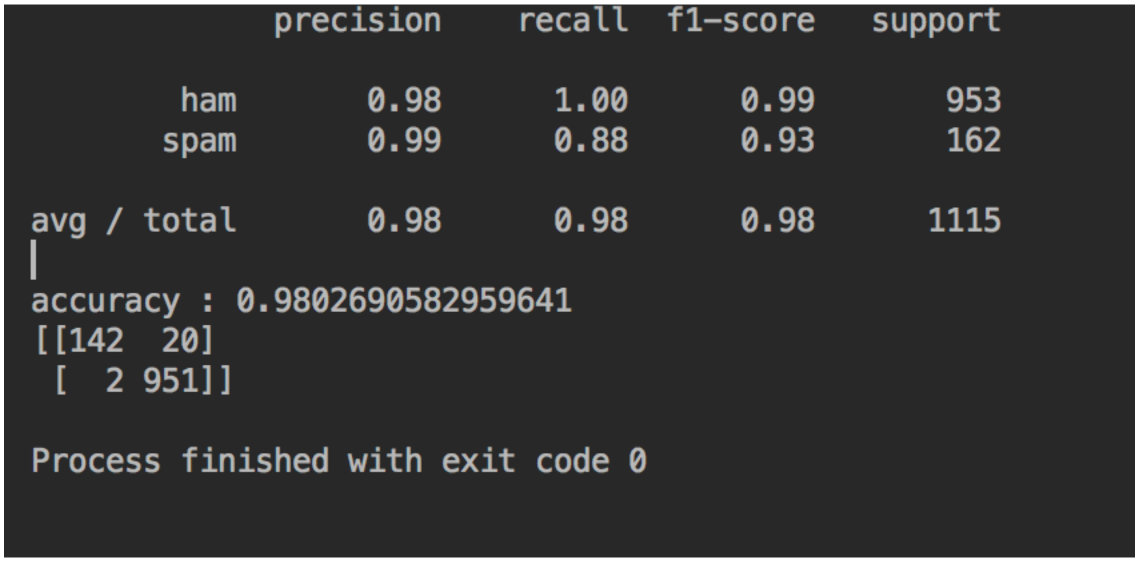
The messages were tokenized then the stop words removed and the the words were stemmed.
I try three different stemmer and they didn't have a lot of difference in performance and final accuracy :
PorterStemmer
SnowballStemmer
LancasterStemmer



	class	text	length
0	ham	[go, jurong, point, crazi, avail, bugi, n, gre...	111
1	ham	[ok, lar, joke, wif, u, oni]	29
2	spam	[free, entri, 2, wkli, comp, win, fa, cup, fin...	155
3	ham	[u, dun, say, earli, hor, u, c, alreadi, say]	49
4	ham	[nah, dont, think, goe, usf, live, around, tho...	61

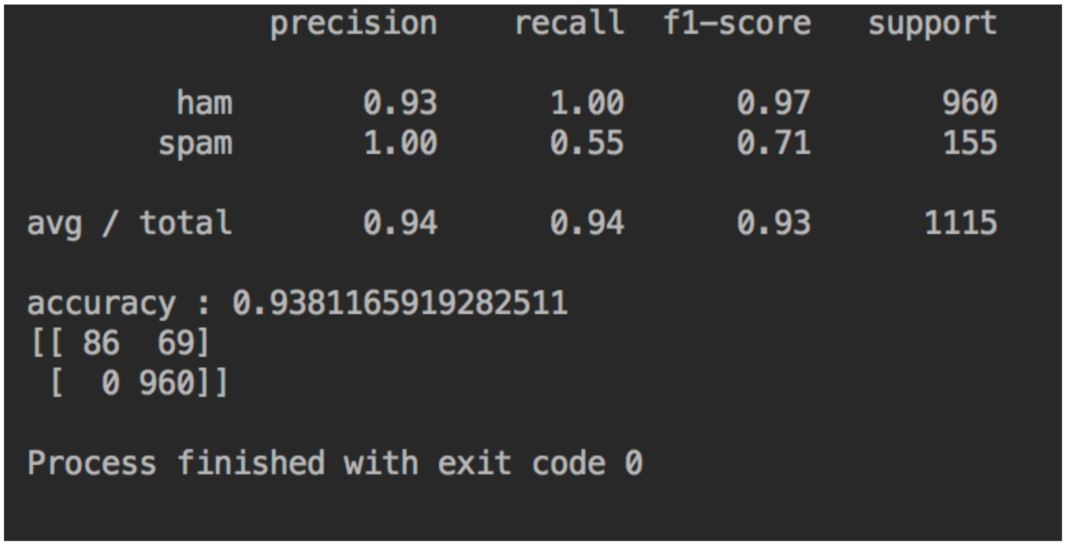
then the strings converted to integer counts and integer counts to weighted TF-IDF scores.
TF-IDF vectors trained with Naive Bayes classifier.
I tried different alpha Parameters and the best was 0.1.

alpha=0.1:



```
             precision    recall  f1-score   support

        ham       0.98      1.00      0.99       953
       spam       0.99      0.88      0.93       162

avg / total       0.98      0.98      0.98      1115
|
accuracy : 0.9802690582959641
[[142  20]
 [  2 951]]

Process finished with exit code 0
```

alpha =2.



```
             precision    recall  f1-score   support

        ham       0.93      1.00      0.97       960
       spam       1.00      0.55      0.71       155

avg / total       0.94      0.94      0.93      1115

accuracy : 0.9381165919282511
[[ 86  69]
 [  0 960]]

Process finished with exit code 0
```