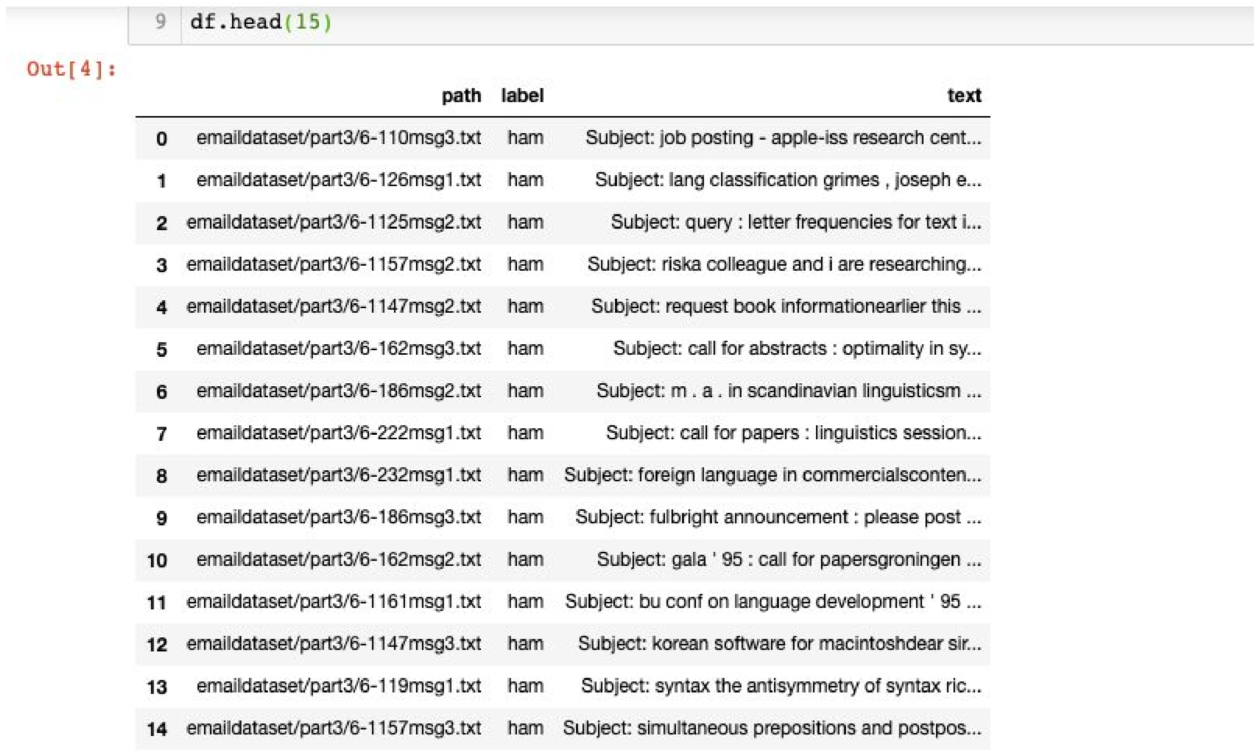
Spam classification

Preprocessing:

1st of all we get all the file path using “glob” library and save into a pandas DataFrame.
We identify true labes of email txt from the names of the files using if 'spmsg' in the name of file its a 'spam' else its 'ham' and save into pd.df, then read all the files text and concatenate into pd.df we got the output like that



```
df.head(15)
```

Out[4]:

	path	label	text
0	emaildataset/part3/6-110msg3.txt	ham	Subject: job posting - apple-iss research cent...
1	emaildataset/part3/6-126msg1.txt	ham	Subject: lang classification grimes , joseph e...
2	emaildataset/part3/6-1125msg2.txt	ham	Subject: query : letter frequencies for text i...
3	emaildataset/part3/6-1157msg2.txt	ham	Subject: riska colleague and i are researching...
4	emaildataset/part3/6-1147msg2.txt	ham	Subject: request book informationearlier this ...
5	emaildataset/part3/6-162msg3.txt	ham	Subject: call for abstracts : optimality in sy...
6	emaildataset/part3/6-186msg2.txt	ham	Subject: m . a . in scandinavian linguisticsm ...
7	emaildataset/part3/6-222msg1.txt	ham	Subject: call for papers : linguistics session...
8	emaildataset/part3/6-232msg1.txt	ham	Subject: foreign language in commercialsconten...
9	emaildataset/part3/6-186msg3.txt	ham	Subject: fulbright announcement : please post ...
10	emaildataset/part3/6-162msg2.txt	ham	Subject: gala ' 95 : call for papersgroningen ...
11	emaildataset/part3/6-1161msg1.txt	ham	Subject: bu conf on language development ' 95 ...
12	emaildataset/part3/6-1147msg3.txt	ham	Subject: korean software for macintoshdear sir...
13	emaildataset/part3/6-119msg1.txt	ham	Subject: syntax the antisymmetry of syntax ric...
14	emaildataset/part3/6-1157msg3.txt	ham	Subject: simultaneous prepositions and postpos...

And store this data in local disk in the form of csv for later usage.

Distribution spam/non-spam

In [14]:

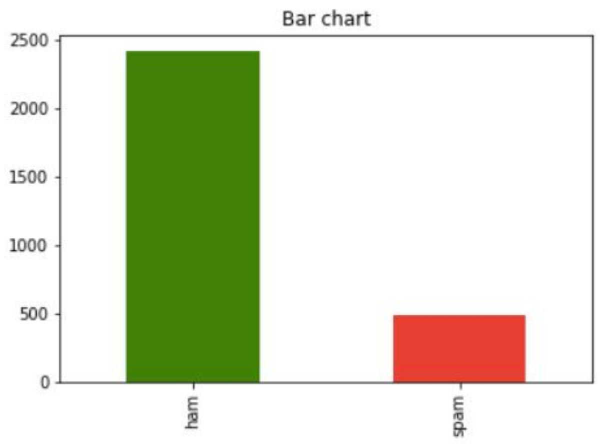
```
df.groupby('label').describe()
```

Out[14]:

	text			
	count	unique	top	freq
label				
ham	2412	2408	Subject: ld ' 98 - call for participationld ' ...	2
spam	481	468	Subject: re := 20 the virtual girlfriend and v...	4

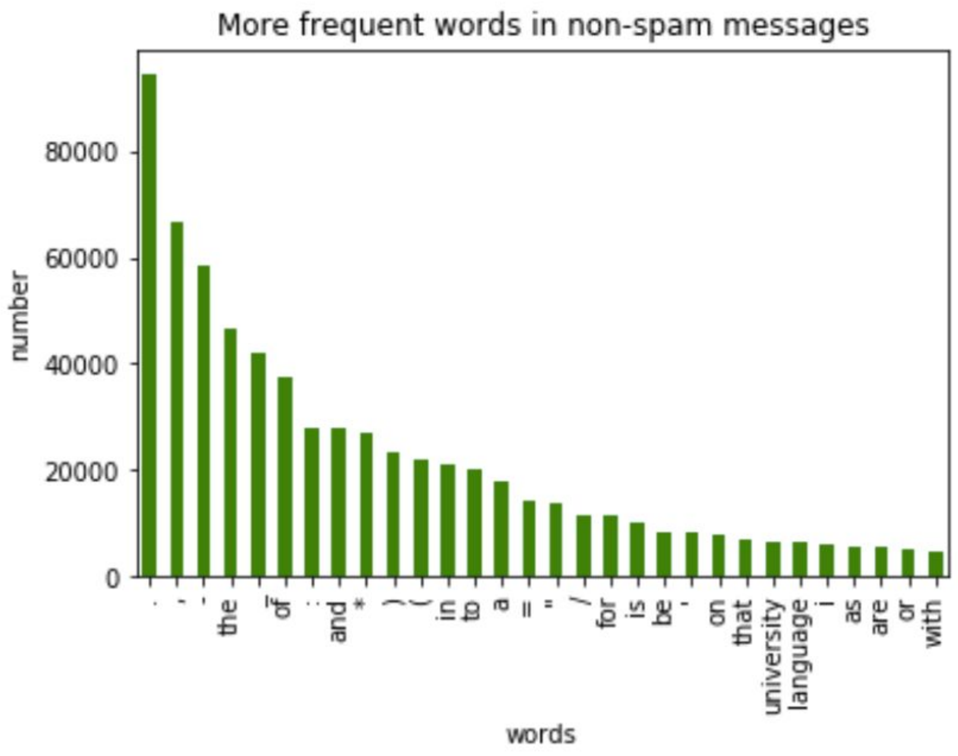
In [15]:

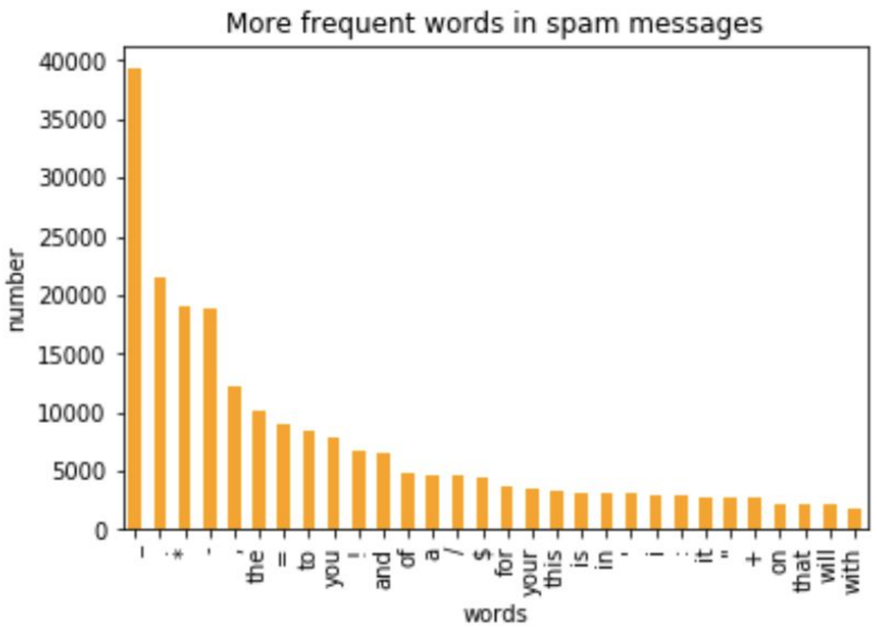
```
count_Class=pd.value_counts(df["label"], sort= True)
count_Class.plot(kind= 'bar', color= ["green", "red"])
plt.title('Bar chart')
plt.show()
```



We can see there are 2412 email are ham and only 481 are spam.

Data analysis:



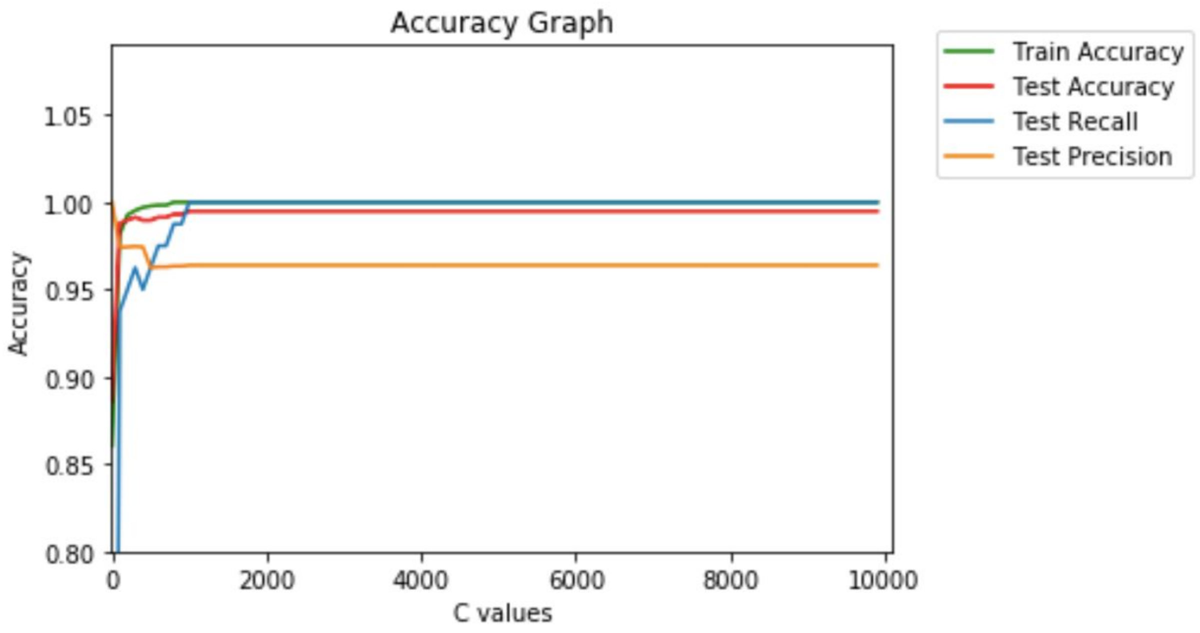


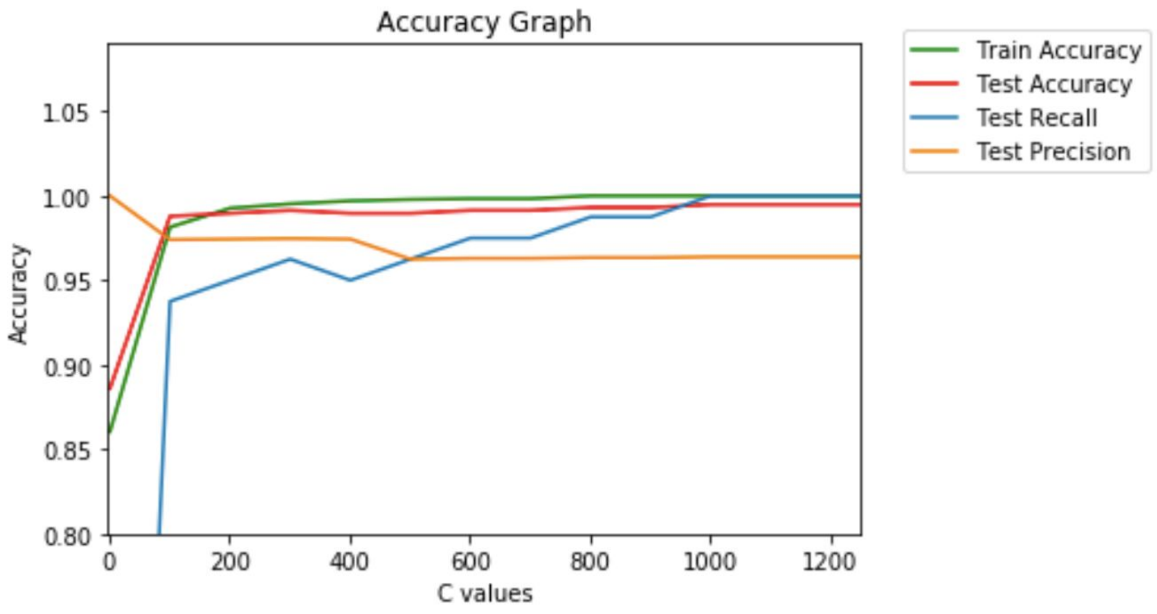
Remove the English stopword and convert the data into vector for efficient processing.

Our goal is to predict whether new email is spam or not. I assume that is much worse misclassify not spam than misclassify an spam. We can see form the above words frequencies we can’t predict whether its spam or not because of overlapping words frequencies. I.e. ‘and’, ‘the’,’for’ are frequent in spam and non-spam emails.

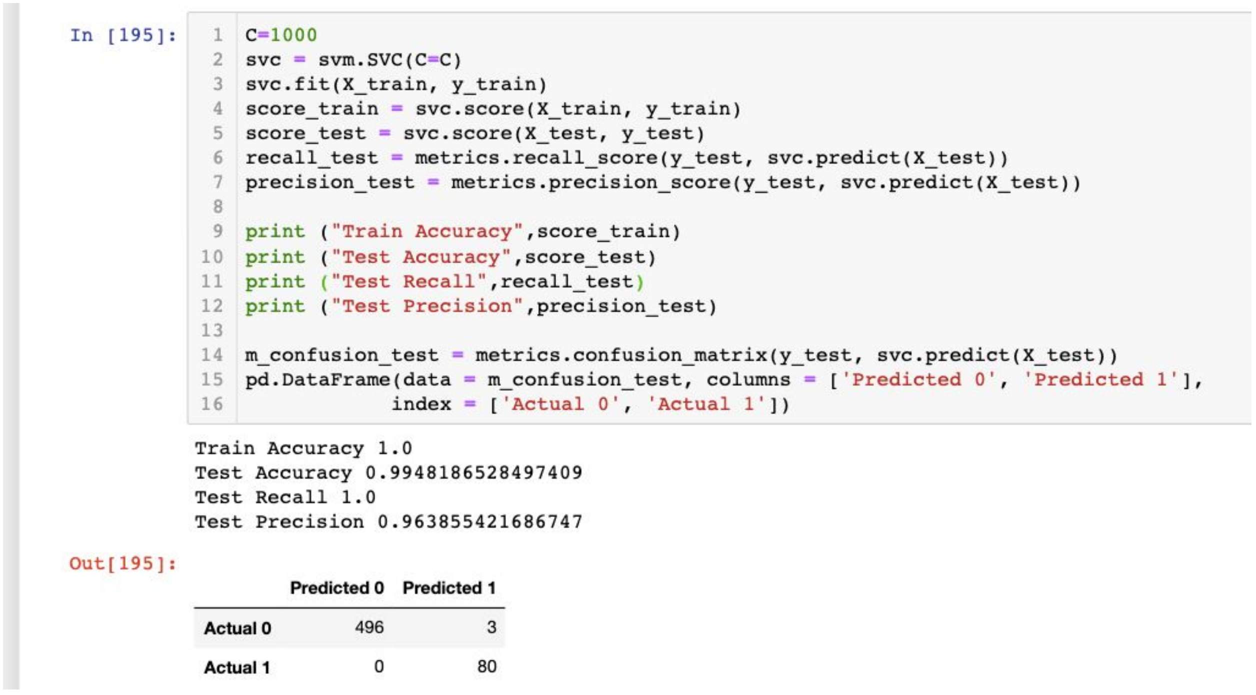
Support Vector Machine:

We are going to apply the support vector machine model with the rbf kernel, train different models changing the regularization parameter C, and evaluate the accuracy, recall and precision of the model with the test set.





We have trained SVM for 1 to 10000 with the step size of 100, but we can see form the graph after the value of C=1000 line shows no change. So, we don't need to go for higher values of C.



In [195]:
```
C=1000
svc = svm.SVC(C=C)
svc.fit(X_train, y_train)
score_train = svc.score(X_train, y_train)
score_test = svc.score(X_test, y_test)
recall_test = metrics.recall_score(y_test, svc.predict(X_test))
precision_test = metrics.precision_score(y_test, svc.predict(X_test))

print ("Train Accuracy",score_train)
print ("Test Accuracy",score_test)
print ("Test Recall",recall_test)
print ("Test Precision",precision_test)

m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
            index = ['Actual 0', 'Actual 1'])
```

```
Train Accuracy 1.0
Test Accuracy 0.9948186528497409
Test Recall 1.0
Test Precision 0.963855421686747
```

Out[195]:

	Predicted 0	Predicted 1
Actual 0	496	3
Actual 1	0	80

We can see at C= 1000 we got the Train Accuracy 100% and Test Accuracy 99.5% . But form confusion matrix we can see we misclassify non-spam to spam witch is not good, we don't want that accuracy if our ham email goes into spam.
So, we need to increase the margin between span and non-spam by decreasing the C value.

After some iterations we reached at point where we set the C=65 and got the required results

```
In [213]:
C=65
svc = svm.SVC(C=C)
svc.fit(X_train, y_train)
score_train = svc.score(X_train, y_train)
score_test = svc.score(X_test, y_test)
recall_test = metrics.recall_score(y_test, svc.predict(X_test))
precision_test = metrics.precision_score(y_test, svc.predict(X_test))

print ("Train Accuracy",score_train)
print ("Test Accuracy",score_test)
print ("Test Recall",recall_test)
print ("Test Precision",precision_test)

m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
            index = ['Actual 0', 'Actual 1'])
```

```
Train Accuracy 0.9714779602420052
Test Accuracy 0.9810017271157168
Test Recall 0.8625
Test Precision 1.0
```

Out[213]:

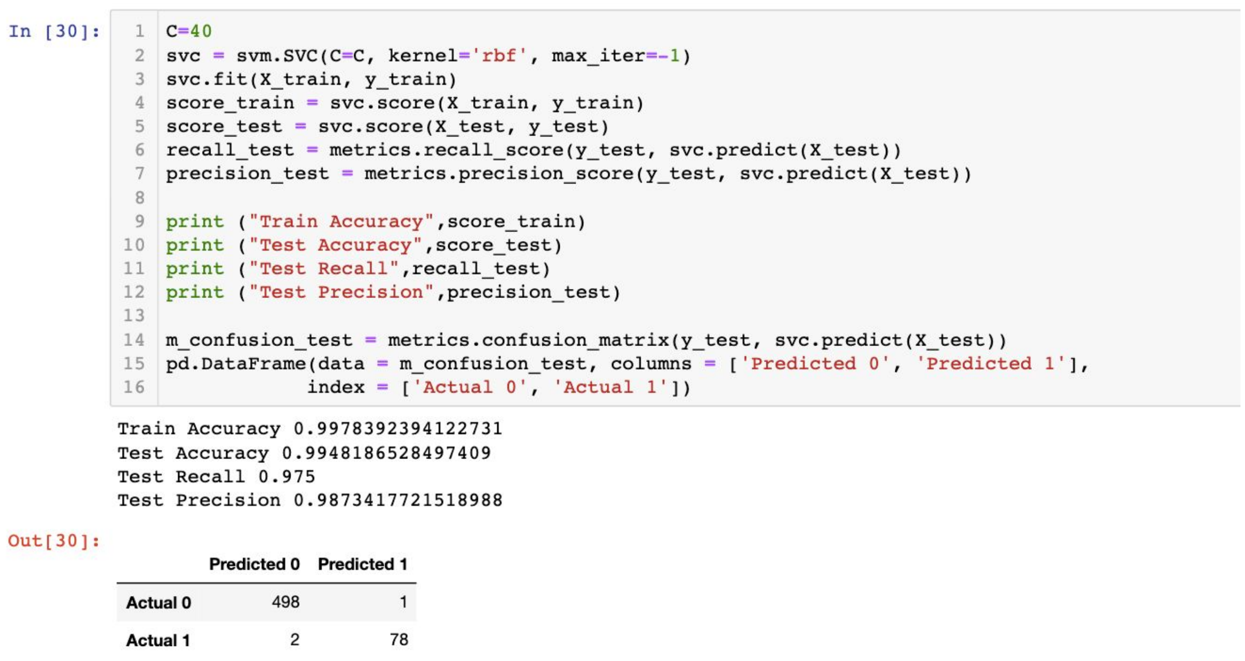
	Predicted 0	Predicted 1
Actual 0	499	0
Actual 1	11	69

We can see at C= 65 our Training Accuracy and Test Accuracy decreased to 97.1% and 98.1%. And we have **11** extra emails in our inbox but no **ham** is classified as **spam**.

Conclusion

The best model we have found for support vector machine with **98.1%** accuracy.

Now I am using 3000 of corpus previously it was more than 60,000



In [30]:

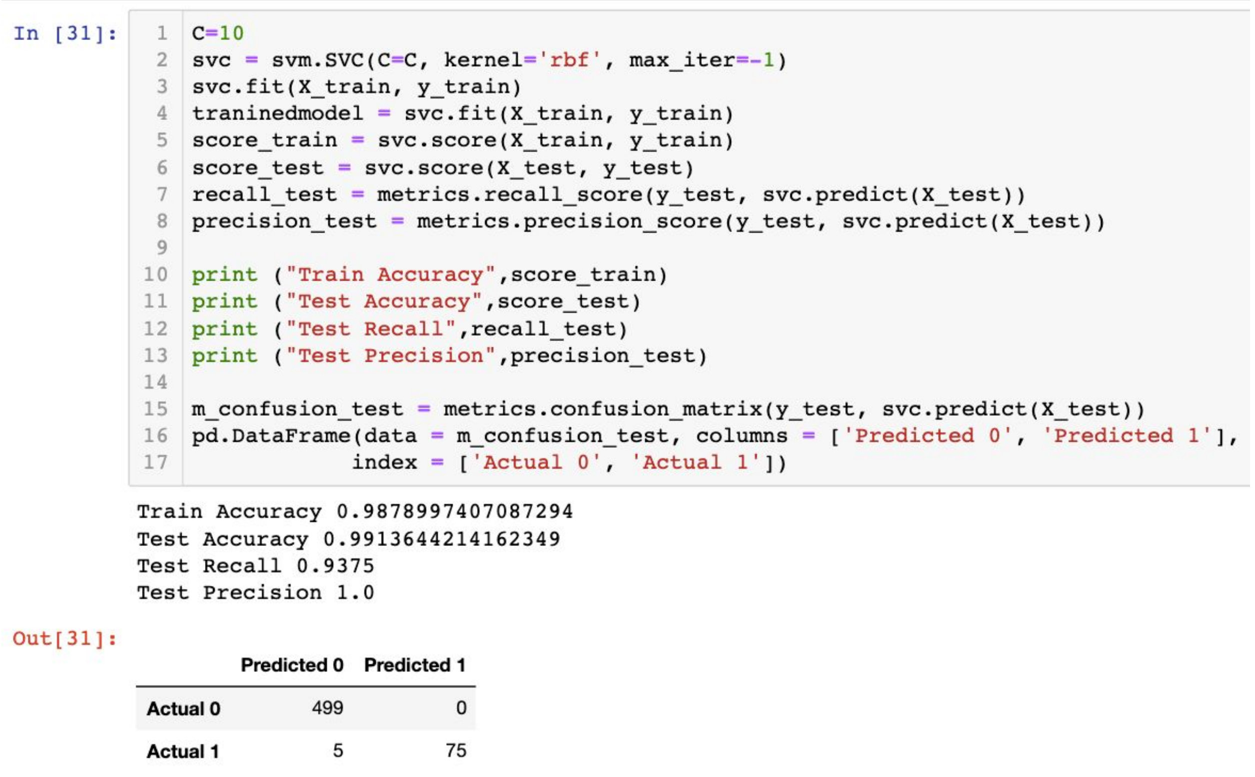
```python
C=40
svc = svm.SVC(C=C, kernel='rbf', max_iter=-1)
svc.fit(X_train, y_train)
score_train = svc.score(X_train, y_train)
score_test = svc.score(X_test, y_test)
recall_test = metrics.recall_score(y_test, svc.predict(X_test))
precision_test = metrics.precision_score(y_test, svc.predict(X_test))

print ("Train Accuracy",score_train)
print ("Test Accuracy",score_test)
print ("Test Recall",recall_test)
print ("Test Precision",precision_test)

m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
            index = ['Actual 0', 'Actual 1'])
```

```
Train Accuracy 0.9978392394122731
Test Accuracy 0.9948186528497409
Test Recall 0.975
Test Precision 0.9873417721518988
```

Out[30]:

	Predicted 0	Predicted 1
Actual 0	498	1
Actual 1	2	78

For C = 40 we got the highest Training and Test Accuracy 99.8% and 99.5%. But form confusion matrix we can see we misclassify non-spam to spam.



In [31]:

```python
C=10
svc = svm.SVC(C=C, kernel='rbf', max_iter=-1)
svc.fit(X_train, y_train)
traninedmodel = svc.fit(X_train, y_train)
score_train = svc.score(X_train, y_train)
score_test = svc.score(X_test, y_test)
recall_test = metrics.recall_score(y_test, svc.predict(X_test))
precision_test = metrics.precision_score(y_test, svc.predict(X_test))

print ("Train Accuracy",score_train)
print ("Test Accuracy",score_test)
print ("Test Recall",recall_test)
print ("Test Precision",precision_test)

m_confusion_test = metrics.confusion_matrix(y_test, svc.predict(X_test))
pd.DataFrame(data = m_confusion_test, columns = ['Predicted 0', 'Predicted 1'],
            index = ['Actual 0', 'Actual 1'])
```

```
Train Accuracy 0.9878997407087294
Test Accuracy 0.9913644214162349
Test Recall 0.9375
Test Precision 1.0
```

Out[31]:

	Predicted 0	Predicted 1
Actual 0	499	0
Actual 1	5	75

For C = 10 we got slightly less accuracy but we got the required results, not to classify spam to a ham email. And very high accuracy 99.1% from the support vector machine with less no. of features.