7 Assuming a set of documents that need to be classified, use naive Bayesian classifier model to perform this task Built in java classes / API can be used to write the program calculate the accuracy precision & recall of your dataset.

```
import pandas as pd
 msg = pd.read-csv('c:/users/lenovo/
     Desktop/4MT16CS060 - Projwal/
         lab6.csv') names = ['message','label'])
   print('Total instances in the dataset:',
              msg.shape[0])
  msg['labelnum'] = msg.label.map
           [{'pos:1, 'neg':0})
    X = msg.message
    Y = msg.labelnum
    print("\n The message & its label
      of first 5 instances are listed
                below")
   X5, y5 = X[0:5], msg.label[0:5]
    for x, y in zip(X5, y5):
         print(x, ',', y)
```

```
from sklearn.model.selection import
      train-test-split
xtrain, xtest, ytrain, ytest = train-test-split
                                   (x, y).
print ("DataSet is split into Training &
      Testing samples")
print (" Total training instances :', xtrain.
                              shape [0])
print ("Total Testing instances :', xtest.
                         shape [0])
from sklearn.feature-extraction.text
      import Countvectorizer
  Count_vect = Countvectorizer()
  xtrain-dtm = count-vect.fit-transform
                  (xtrain).
  xtest-dtm = count-vect.transform (xtest)
 print ("\n Total features extracted using
   countvectorizer :", xtrain-dtm.shape(1))
  print ("\n Features for first 5
        training instances are listed below")
  df= pd.DataFrame (xtrain-dtm.
       toarray (), columns = Count-vect.
                get-feature_names ())
   print (df[0:5])
```

```
from sklearn.naive.bayes import MultiNominalNB
df= MultiNominalNB().fit(x train-dtm,
                                   ytrain)
predicted = df.predict (xtest - dtm)
print ("\n classification results of testing
        samples are given below")
for doc, p in zip (xtest, predicted):
    pred = 'pos' if p = = 1 else 'neg'
    print ("%s -> %s" %(doc,pred))

from sklearn import metrices
print ("\n Accuracy metrices")
print ("\n Accuracy of the classifeir is",
            matrices.accuracy - score
                          (ytest, predicted))
print ("Recall :", metrices.recall - score
                     (ytest, predicted))
print (" precision :", metrices.precision_
          score (ytest, predicted))
print (" confusion matrix")
print (metrices.confusion_matrix
                  (ytest, predicted))
```

Output :

```
Total instances in the dataset : 18
The message & its label of first 5 insta
are listed below

  I love this sandwich, pos
  This is an amazing place, pos
  I feel very good about these beers, pos
  This is my best work, pos
  what an awesome view, pos

Dataset is split into Training & Testing Samples
  Total training Instances : 13
  Total testing Instances : 5

Total features extracted using countvectorizer:
                                             46
features for first 5 training instances are listed

     about  am  an  awesome  beers  best  boss  can  dead
0      0     0   0     0       0      0     0     0    0
1      0     0   0     0       0      0     0     0    0
2      0     0   0     0       0      0     0     1    1
3      0     0   1     1       0      0     0     0    0
4      0     0   0     0       0      0     0     0    0
```

	do	...	today
0	1	...	0
1	0	...	1
2	0	...	0
3	0	...	0
4	0	...	0

	tomorrow	very	view	we	went	what	will	with	work
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0
2	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

[5 rows X 46 column]

classification results of testing samples are given below.

I love to dance -> pos
I am sick and tired of this place -> neg
This is an amazing place -> pos
what a great holiday -> pos
This is a bad locality to study -> neg

Accuracy metrics

```
Accuracy of the classifier is 1.0
Recall : 1.0
Precision : 1.0
Confusion matrix
[[2 0]
 [0 3]]
```