In [ ]:

```
Program:6
Assuming a set of documents that need to be classified, use the naïve Bayesian Classifi
model to perform this task. Built-in Java classes/API can be used to write the program.
Calculate the accuracy, precision, and recall for your data set.
```

In [13]:

```
import pandas as pd
```

In [14]:

```
msg=pd.read_csv('document.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('\nThe message and 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)
```

```
Total instances in the dataset: 18

The message and its label of first 5 instances 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
```

In [15]:

```
# Splitting the dataset into train and test data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(X,Y)
print('\nDataset is split into Training and Testing samples')
print('Total training instances :', xtrain.shape[0])
print('Total testing instances :', xtest.shape[0])
```

```
Dataset is split into Training and Testing samples
Total training instances : 13
Total testing instances : 5
```

In [16]:

```
# Output of count vectoriser is a sparse matrix
# CountVectorizer - stands for 'feature extraction'
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
xtrain_dtm = count_vect.fit_transform(xtrain) #Sparse matrix
xtest_dtm = count_vect.transform(xtest)
print('\nTotal features extracted using CountVectorizer:',xtrain_dtm.shape[1])
print('\nFeatures for first 5 training instances are listed below')
df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())
print(df[0:5])#tabular representation

```

```
Total features extracted using CountVectorizer: 42

Features for first 5 training instances are listed below
   am  amazing  an  and  awesome  best  boss  can  dance  deal  ...  the  \
0   0        0   0    0        0     0     0    0      0     0  ...    0
1   1        0   0    1        0     0     0    0      0     0  ...    0
2   0        0   0    0        0     0     0    0      0     0  ...    0
3   0        1   1    0        0     0     0    0      0     0  ...    0
4   0        0   0    0        0     0     0    0      1     0  ...    0

   this  tired  to  tomorrow  we  what  will  with  work
0     0      0   0         0   0     0     0     0     0
1     1      1   0         0   0     0     0     0     0
2     0      0   0         1   1     0     1     0     0
3     1      0   0         0   0     0     0     0     0
4     0      0   1         0   0     0     0     0     0

[5 rows x 42 columns]
```

In [17]:

```
#print(xtrain_dtm) #Same as above but sparse matrix representation
# Training Naive Bayes (NB) classifier on training data.
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)
print('\nClassstification 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))

```

```
Classstification results of testing samples are given below
I am tired of this stuff -> neg
```

In [18]:

```
#printing accuracy metrics
from sklearn import metrics
print('\nAccuracy metrics')
print('Accuracy of the classifer is',metrics.accuracy_score(ytest,predicted))
print('Recall :',metrics.recall_score(ytest,predicted),
'\nPrecison :',metrics.precision_score(ytest,predicted))
print('Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))

```

```
Accuracy metrics
Accuracy of the classifer is 0.8
Recall : 1.0
Precison : 0.6666666666666666
Confusion matrix
[[2 1]
 [0 2]]
```

In [ ]:

```

```