Program 6

Assuming a set of documents that need to be classified, use the naive Bayesian classifier 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 dataset.

```
import pandas as pd
msg = pd.read_csv('data6.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 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)
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(X, Y)
print('\n Dataset is split into Training and Testing
                                        Samples')
print('\n Total Training instances:', xtrain.shape[0])
print('\n 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)
```

Teacher's Signature _____

```
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 MultinomialNB
df = MultinomialNB().fit(xtrain_dtm, ytrain)
predicted = df.predict(xtrain_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 metrics
print("\n Accuracy metrics")
print("Accuracy of the classifier is", metrics.accuracy_score
                                    (ytest, predicted))
print("Recall:", metrics.recall_score(ytest, predicted), "\n Precision:
                                    ", metrics.precision_score):
print("Confusion Matrix")
print(metrics.confusion_matrix(test, predicted))
```

Teacher's Signature

Dataset

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
I do not like this stuff, neg
I can't deal with this, neg
He is my swim enemy, neg
My boss is horrible, neg
This is an awesome place, pos
I do not like the taste of this juice, neg
I love to dance, pos
I am sick and tired of this place, neg
what a great holiday, pos
That is bad locality to stay, neg
We will have good fun tommorows, pos.
I went to my enemy's house today, neg.

Output

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.

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

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

	about	am	an	awesome	beers	but	boss	can	did	do ... total
0	0	0	0	0	0	0	0	0	0	1 ... 0
1	0	0	0	0	0	0	0	0	0	0 ... 1
2	0	0	0	0	0	0	0	1	1	0 ... 0
3	0	0	0	1	0	0	0	0	0	0 ... 0
4	0	0	0	0	0	0	0	0	0	0 ... 0

	tomorrow	cry	view	we	went	what	will	with	
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 columns]

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 stay → neg.

Accuracy metrics
Accuracy of the classifier is 1.0
Recall : 1.0
Precision : 1.0

Confusion Matrix
[[2 0]
[0 3]]