6. Assuming a set of documents that need to be classified, use the naïve 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 data set.

Naive Bayes algorithms for learning and classifying text

LEARN_NAIVE_BAYES_TEXT (Examples, V)

Examples is a set of text documents along with their target values. V is the set of all possible target values. This function learns the probability terms $P(w_k | v_j,)$, describing the probability that a randomly drawn word from a document in class v_j will be the English word w_k. It also learns the class prior probabilities $P(v_j)$.

1. *collect all words, punctuation, and other tokens that occur in Examples*
   - *Vocabulary* ← ***c*** the set of all distinct words and other tokens occurring in any text document from *Examples*

2. *calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms*

   - For each target value v_j in *V* do
     - $docs_j$ ← the subset of documents from *Examples* for which the target value is *vj*
     - $P(v_j) \leftarrow | docs_j | / |\text{Examples}|$
     - $Text_j$ ← a single document created by concatenating all members of $docs_j$
     - *n* ← total number of distinct word positions in $Text_j$
     - for each word w_k in *Vocabulary*
       - n_k ← number of times word $\boldsymbol{w_k}$ occurs in $Text_j$
       - $P(w_k|v_j) \leftarrow (n_k + 1) / (n + |Vocabulary|)$

CLASSIFY_NAIVE_BAYES_TEXT (Doc)

Return the estimated target value for the document Doc. ai denotes the word found in the ith position within Doc.

- *positions* ← all word positions in *Doc* that contain tokens found in *Vocabulary*
- Return V_{NB}, where

$$v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

Examples:

	Text Documents	Label
1	I love this sandwich	pos
2	This is an amazing place	pos
3	I feel very good about these beers	pos
4	This is my best work	pos
5	What an awesome view	pos
6	I do not like this restaurant	neg
7	I am tired of this stuff	neg
8	I can't deal with this	neg
9	He is my sworn enemy	neg
10	My boss is horrible	neg
11	This is an awesome place	pos
12	I do not like the taste of this juice	neg
13	I love to dance	pos
14	I am sick and tired of this place	neg
15	What a great holiday	pos
16	That is a bad locality to stay	neg
17	We will have good fun tomorrow	pos
18	I went to my enemy's house today	neg

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

msg=pd.read_csv('naivetext.csv',names=['message','label'])
print('The dimensions of the dataset',msg.shape)
msg['labelnum']=msg.label.map({'pos':1,'neg':0})
X=msg.message
y=msg.labelnum


#splitting the dataset into train and test data
xtrain,xtest,ytrain,ytest=train_test_split(X,y)

print ('\n The total number of Training Data :',ytrain.shape)
print ('\n The total number of Test Data :',ytest.shape)


#output of count vectoriser is a sparse matrix
cv = CountVectorizer()
xtrain_dtm = cv.fit_transform(xtrain)
xtest_dtm=cv.transform(xtest)
print('\n The words or Tokens in the text documents \n')
print(cv.get_feature_names())

df=pd.DataFrame(xtrain_dtm.toarray(),columns=cv.get_feature_na
mes())

# Training Naive Bayes (NB) classifier on training data.
clf = MultinomialNB().fit(xtrain_dtm,ytrain)
predicted = clf.predict(xtest_dtm)

#printing accuracy, Confusion matrix, Precision and Recall
print('\n Accuracy of the classifer is’,
metrics.accuracy_score(ytest,predicted))

print('\n Confusion matrix')
print(metrics.confusion_matrix(ytest,predicted))


print('\n The value of Precision' ,
metrics.precision_score(ytest,predicted))
```

```
print('\n The value of Recall' ,
metrics.recall_score(ytest,predicted))
```

Output:

The dimensions of the dataset (18, 2)

The total number of Training Data : (13,)

The total number of Test Data : (5,)

The words or Tokens in the text documents

['about', 'am', 'amazing', 'an', 'awesome', 'bad', 'beers', 'boss', 'can', 'deal', 'do', 'enemy', 'feel', 'fun', 'good', 'great', 'have', 'holiday', 'horrible', 'house', 'is', 'juice', 'like', 'locality', 'love', 'my', 'not', 'of', 'place', 'restaurant', 'sandwich', 'stay', 'stuff', 'taste', 'that', 'the', 'these', 'this', 'tired', 'to', 'today', 'tomorrow', 'very', 'view', 'we', 'went', 'what', 'will', 'with']

Accuracy of the classifier is 0.6

Confusion matrix

[[2 0]

[2 1]]

The value of Precision 1.0

The value of Recall 0.3333333333333333

Basic knowledge

Confusion Matrix

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

True positives: data points labelled as positive that are actually positive

False positives: data points labelled as positive that are actually negative

True negatives: data points labelled as negative that are actually negative

False negatives: data points labelled as negative that are actually positive

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$= \frac{True\ Positive}{Total\ Actual\ Positive}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{Total\ Predicted\ Positive}$$

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Example:

		Actual	
		Positive	Negative
Predicted	Positive	1 TP	3 FP
	Negative	0 FN	1 TN

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{1}{1+3} = 0.25$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{1}{1+0} = 1$$

Accuracy: how often is the classifier correct?

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total}} = \frac{1+1}{5} = 0.4$$