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 $v_j$
     - $P(v_j) ← |docs_j| / |Examples|$
     - $Text_j ←$ 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 $w_k$ occurs in $Text_j$
       - $P(w_k | v_j) ← (n_k + 1) / (n + |Vocabulary|)$

**CLASSIFY NAIVE_BAYES_TEXT** (Doc)

Return the estimated target value for the document Doc. $a_i$ denotes the word found in the $i^{th}$ position within Doc.

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

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$
**Data set:**

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

```python
import pandas as pd

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

print(X)
print(y)

#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 ('\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
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 The words or Tokens in the text documents \n')
print(count_vect.get_feature_names())

df=pd.DataFrame(xtrain_dtm.toarray(),columns=count_vect.get_feature_names())

# 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)

#printing accuracy, Confusion matrix, Precision and Recall
from sklearn import metrics
print('\n Accuracy of the classifier 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)
0    I love this sandwich
1    This is an amazing place
2    I feel very good about these beers
3    This is my best work
4    What an awesome view
5    I do not like this restaurant
6    I am tired of this stuff
7    I can't deal with this
8    He is my sworn enemy
9    My boss is horrible
10   This is an awesome place
11   I do not like the taste of this juice
12   I love to dance
13   I am sick and tired of this place
14   What a great holiday
15   That is a bad locality to stay
16   We will have good fun tomorrow
17   I went to my enemy's house today
Name: message, dtype: object
0  1
1  1
2  1
3  1
4  1
5  0
6  0
7  0
8  0
9  0
10 1
11 0
12 1
13 0
14 1
15 0
16 1
17 0
Name: labelnum, dtype: int64

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', 'and', 'awesome', 'beers', 'best', 'can', 'deal', 'do', 'enemy', 'feel', 'fun', 'good', 'great', 'have', 'he', 'holiday', 'house', 'is', 'like', 'love', 'my', 'not', 'of', 'place', 'restaurant', 'sandwich', 'sick', 'sworn', 'these', 'this', 'tired', 'to', 'today', 'tomorrow', 'very', 'view', 'we', 'went', 'what', 'will', 'with', 'work']

Accuracy of the classifier is 0.8

Confusion matrix

[[2 1]
 [0 2]]

The value of Precision 0.6666666666666666

The value of Recall 1.0
Basic knowledge

Confusion Matrix

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive</td>
<td>True Positive</td>
<td>False Positive</td>
</tr>
<tr>
<td>Negative</td>
<td>Positive</td>
<td>False Negative</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

**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

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

\[
= \frac{\text{True Positive}}{\text{Total Actual Positive}}
\]

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>Positive</td>
<td>True Positive</td>
<td>False Positive</td>
</tr>
<tr>
<td>Negative</td>
<td>Positive</td>
<td>False Negative</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

\[
\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}
\]

\[
= \frac{\text{True Positive}}{\text{Total Predicted Positive}}
\]
Example:

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>True Positive</td>
</tr>
<tr>
<td>Negative</td>
<td>False Negative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>1</td>
</tr>
<tr>
<td>Negative</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Precision = \( \frac{TP}{TP + FP} = \frac{1}{1 + 3} = 0.25 \)

Recall = \( \frac{TP}{TP + FN} = \frac{1}{1 + 0} = 1 \)

**Accuracy**: how often is the classifier correct?

\[
\text{Accuracy} = \frac{TP + TN}{\text{Total}} = \frac{1 + 1}{5} = 0.4
\]
Example: Movie Review

<table>
<thead>
<tr>
<th>Doc</th>
<th>Text</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I loved the movie</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>I hated the movie</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>a great movie. good movie</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>poor acting</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>great acting. good movie</td>
<td>+</td>
</tr>
</tbody>
</table>

Unique word

< I, loved, the, movie, hated, a, great, good, poor, acting>

<table>
<thead>
<tr>
<th>Doc</th>
<th>I</th>
<th>loved</th>
<th>the</th>
<th>movie</th>
<th>hated</th>
<th>a</th>
<th>great</th>
<th>good</th>
<th>poor</th>
<th>acting</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doc</th>
<th>I</th>
<th>loved</th>
<th>the</th>
<th>movie</th>
<th>hated</th>
<th>a</th>
<th>great</th>
<th>good</th>
<th>poor</th>
<th>acting</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>2</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

\[ P(+) = \frac{3}{5} = 0.6 \]
\[
P(I |+) = \frac{1 + 1}{14 + 10} = 0.0833 \\
P(a |+) = \frac{1 + 1}{14 + 10} = 0.0833
\]

\[
P(\text{loved} |+) = \frac{1 + 1}{14 + 10} = 0.0833 \\
P(\text{great} |+) = \frac{2 + 1}{14 + 10} = 0.125
\]

\[
P(\text{the} |+) = \frac{1 + 1}{14 + 10} = 0.0833 \\
P(\text{good} |+) = \frac{2 + 1}{14 + 10} = 0.125
\]

\[
P(\text{movie} |+) = \frac{4 + 1}{14 + 10} = 0.2083 \\
P(\text{poor} |+) = \frac{0 + 1}{14 + 10} = 0.0416
\]

\[
P(\text{hated} |+) = \frac{0 + 1}{14 + 10} = 0.0416 \\
P(\text{acting} |+) = \frac{1 + 1}{14 + 10} = 0.0833
\]

<table>
<thead>
<tr>
<th>Doc</th>
<th>I</th>
<th>loved</th>
<th>the</th>
<th>movie</th>
<th>hated</th>
<th>a</th>
<th>great</th>
<th>good</th>
<th>poor</th>
<th>acting</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

\[
P(-) = \frac{2}{5} = 0.4
\]

\[
P(I |-) = \frac{1 + 1}{6 + 10} = 0.125 \\
P(a |-) = \frac{0 + 1}{6 + 10} = 0.0625
\]

\[
P(\text{loved} |-) = \frac{0 + 1}{6 + 10} = 0.0625 \\
P(\text{great} |-) = \frac{0 + 1}{6 + 10} = 0.0625
\]

\[
P(\text{the} |-) = \frac{1 + 1}{6 + 10} = 0.125 \\
P(\text{good} |-) = \frac{0 + 1}{6 + 10} = 0.0625
\]

\[
P(\text{movie} |-) = \frac{1 + 1}{6 + 10} = 0.125 \\
P(\text{poor} |-) = \frac{1 + 1}{6 + 10} = 0.125
\]

\[
P(\text{hated} |-) = \frac{1 + 1}{6 + 10} = 0.125 \\
P(\text{acting} |-) = \frac{1 + 1}{6 + 10} = 0.125
\]
Let’s classify the new document

I hated the poor acting

If \( V_j = + \)
then,
\[
= P(+) P(I | +) P(hated | +) P(the | +) P(poor | +) P(acting | +)
\]
\[
= 0.6 \times 0.0833 \times 0.0416 \times 0.0833 \times 0.0416 \times 0.0833
\]
\[
= 6.03 \times 10^{-2}
\]
If \( V_j = - \)
then,
\[
= P(-) P(I | -) P(hated | -) P(the | -) P(poor | -) P(acting | -)
\]
\[
= 0.4 \times 0.125 \times 0.125 \times 0.125 \times 0.125 \times 0.125
\]
\[
= 1.22 \times 10^{-5}
\]

\[
= 1.22 \times 10^{-5} > 6.03 \times 10^{-2}
\]
So, the new document belongs to \( - \) class