5.Assuming a set of documents that need to be classified, use the naive Bayesian Classifier model to perform this task. Calculate the accuracy, precision, and recall for your data set.

Dataset:

```
i love sandwitch,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 restraunt,neg
i am tired of this stuff,neg
i can't deal with this,neg
he is my sworn 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 tommorrow,pos
i went to my enemy's house today,neg
```

CODE:

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

data=pd.read_csv('textdata.csv',names=['message','label'])
print('The dataset is',data)
print('The dimensions of the dataset',data.shape)
data['labelnum']=data.label.map({'pos':1,'neg':0})
X=data.message
y=data.labelnum
print(X)
print(y)
vectorizer = TfidfVectorizer()
data = vectorizer.fit_transform(X)
print('\n the Features of dataset:\n')
df=pd.DataFrame(data.toarray(),columns=vectorizer.get_feature_names_out()
) df.head()
print('\n Train Test Split')
xtrain,xtest,ytrain,ytest = train_test_split(data,y,test_size=0.3,random_state=42)
```

```
print('\n the total number of training data:',ytrain.shape)
print('\n the total number of test data:',ytest.shape)
clf=MultinomialNB().fit(xtrain,ytrain)
predict=clf.predict(xtest)
predicted=clf.predict(xtest)
print('\n Accuracy of the classifier is',metrics.accuracy_score(ytest,predicted))
print('\nConfusion Matrix is\n',metrics.confusion_matrix(ytest,predicted))
print('\n classification report
is\n',metrics.classification_report(ytest,predicted)) print('\n Value of precision
is\n',metrics.precision_score(ytest,predicted)) print('\n Value of recall
is\n',metrics.recall_score(ytest,predicted))
```

OUTPUT:

```
The dataset is message label
0 i love sandwitch pos
1 this is an amazing place pos
2 i feel very good about these beers pos
3 this is my best work pos
4 what an awesome view pos
5 i do not like this restraunt neg
6 i am tired of this stuff neg
7 i can't deal with this neg
8 he is my sworn enemy neg
9 my boss is horrible neg
10 this is an awesome place pos
11 i do not like the taste of this juice neg
12 i love to dance pos
13 i am sick and tired of this place neg
14 what a great holiday pos
15 that is bad locality to stay neg
16 we will have good fun tommorrow pos
17 i went to my enemy's house today neg
The dimensions of the dataset (18, 2) 0 i love sandwitch
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 bad locality to stay
16 we will have good fun tommorrow
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 Features of dataset: Train Test Split

 the total number of training data: (12,)

 the total number of test data: (6,)

 Accuracy of the classifier is 0.8333333333333334

Confusion Matrix is
 [[3 0]
 [1 2]]

 classification report is
                              precision recall f1-score support

        0 0.75 1.00 0.86 3 1 1.00 0.67 0.80 3

   accuracy 0.83 6 macro avg 0.88 0.83 0.83 6 weighted avg 0.88 0.83 0.83
                                                                                    6 Value of precision is
 1.0

 Value of recall is
 0.6666666666666666
```

6.Construct aBayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set

DATASET:

```
SuperSeniorCitizen,Male,Yes,Medium,Sedetary,High,Yes
SuperSeniorCitizen,Female,Yes,Medium,Sedetary,High,Yes
SeniorCitizen,Male,No,High,Moderate,BorderLine,Yes
Teen,Male,Yes,Medium,Sedetary,Normal,No
Youth,Female,Yes,High,Athlete,Normal,No
MiddleAged,Male,Yes,Medium,Active,High,Yes
Teen,Male,Yes,High,Moderate,High,Yes
SuperSeniorCitizen,Male,Yes,Medium,Sedetary,High,Yes
Youth,Female,Yes,High,Athlete,Normal,No
SeniorCitizen,Female,No,High,Athlete,Normal,Yes
Teen,Female,No,Medium,Moderate,High,Yes
Teen,Male,Yes,Medium,Sedetary,Normal,No
MiddleAged,Female,No,High,Athlete,High,No
MiddleAged,Male,Yes,Medium,Active,High,Yes
Youth,Female,Yes,High,Athlete,BorderLine,No
SuperSeniorCitizen,Male,Yes,High,Athlete,Normal,Yes
SeniorCitizen,Female,No,Medium,Moderate,BorderLine,Yes
Youth,Female,Yes,Medium,Athlete,BorderLine,No
Teen,Male,Yes,Medium,Sedetary,Normal,No
```

CODE:

```
import pandas as pd
col=['Age','Gender','FamilyHist','Diet','LifeStyle','Cholesterol','HeartDisease'
] data = pd.read_csv('lab8.csv',names =col )
print(data)
#encoding
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
for i in range(len(col)):
 data.iloc[:,i] = encoder.fit_transform(data.iloc[:,i])
#spliting data
X = data.iloc[:,0:6]
y = data.iloc[:,-1]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
#prediction
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
#confusion mtx output
```

```
from sklearn.metrics import confusion_matrix
print('Confusion matrix',confusion_matrix(y_test, y_pred))
```

OUTPUT:

```
 Age Gender FamilyHist Diet LifeStyle Cholesterol \ 0 SuperSeniorCitizen Male
           Yes Medium Sedetary High 1 SuperSeniorCitizen Female Yes Medium
              Sedetary High 2 SeniorCitizen Male No High Moderate BorderLine



3 Teen Male Yes Medium Sedetary Normal 4 Youth Female Yes High Athlete
Normal 5 MiddleAged Male Yes Medium Active High 6 Teen Male Yes High
Moderate High 7 SuperSeniorCitizen Male Yes Medium Sedetary High 8 Youth
Female Yes High Athlete Normal 9 SeniorCitizen Female No High Athlete
Normal 10 Teen Female No Medium Moderate High 11 Teen Male Yes Medium
Sedetary Normal 12 MiddleAged Female No High Athlete High 13 MiddleAged
Male Yes Medium Active High 14 Youth Female Yes High Athlete BorderLine 15
SuperSeniorCitizen Male Yes High Athlete Normal 16 SeniorCitizen Female No
Medium Moderate BorderLine 17 Youth Female Yes Medium Athlete BorderLine
18 Teen Male Yes Medium Sedetary Normal

  HeartDisease
0 Yes
1 Yes
2 Yes
3 No
4 No
5 Yes
6 Yes
7 Yes
8 No
9 Yes
10 Yes
11 No
12 No
13 Yes
14 No
15 Yes
16 Yes
17 No
18 No
Confusion matrix [[2 0]
                  [2 0]]
```