5. write a program to implement the naive Bayesian classifier for a sample training dataset stored as a .csv file. Compute the accuracy of the classifier, considering few test data sets

```
import csv, random, math
import statistics as st
def loadcsv ( filename):
        lines = csv. reader (open( filename,"r"))
        dataset = list (lines)
         for i in range ( len ( dataset )):
              dataset [i] = [ float (x) for x
                              in dataset [i]]
         return dataset

def splitdataset ( dataset , splitRatio):
     testsize = int ( len ( dataset ) * splitRatio);
     trainset = list (dataset );
     testset = [ ]
     while len ( testset) < testsize :
          index = random . randrange (len(
                        trainset ));
        testset .append (trainset.pop(index))
    return ( trainset ,testset)
```

```
def seperateByclass (dataset):
    seperated = {}
    for i in range (len(dataset)):
        x = dataset [i]
        if [ x[-1] not in seporated :
            seperated [x[-1]]= [ ]
        seperated [x[-1] ].append (x)
        return seperated

def compute_mean_std (dataset) :
    mean_std = [(st.mean (attribute). st.
            stdev (attribute))
    for attribute in zip [* dataset ]] ;
    del mean_std [-1]
    return mean_std

def summarizeByClass (dataset):
    seperated = seperateByClass (dataset)
    summary = {}
    for classvalue , instances in seperated.items():
        summary [classvalue ]= compute_mean_
            std (instances)
    return summary.
```

```
def estimateProbability (x, mean, stdev):
  exponent = math.exp (-(math.pow
        (x-mean, 2) / (2* math.pow(stdev,2)))
  return (1/(math.sqrt (2*math.pi) *
             stdev)) * exponent.

def calculateClassProbabilities (summaries, testVector):
    p = {}
     for classValue, class Summaries in Summaries.
                items():
     p [classValue] = 1
     for i in range (len (class sumaries)):
          mean, stdev = classSumaries [i];
       x = testVector [i]
        p [classValue] *= estimateProbability
                 (x, mean stdev);
     return p.

def predict (summaries, testVector):
     all-p = Calculate Class Probabilities
              (Summaries, testVector)
     bestlabel, besprob = None, -1
     for lbl, p in all-p.items():
          if bestlabel is None or p > bestProb:
              bestProb = p
              bestlabel = lbl
```

```
return bestlabel

def perform_classification (summaries, test set);
    predictions = [ ]
    for i in range (len (testSet)):
        result = predict (summaris, testset [i])
        predictions . append (result)
        return predictions

def getAccuracy (test set, predictions):
    correct = 0
    for i in range (len (testset ]):
        if testSet [i][-1] = = predictions [i]:
            Correct + = 1
        return (correct / float (len (test ~~set~~)))
                        * 100.0

dataset = loadcsv ('c:/users/lenovo / Desktop
        / 4MT16CS060 - Projwct / diabetis.csv');
print ('Pima Indian Diabetes Dataset loaded..')
print (' Total instances available :', len(dataset))
print (' Total attributes present :', len(dataset
                                    [0] -1)
print ('First Five instances of dataset :")
```

```
for i in range (5) :
    print (i+1, f':' , dataset [i])
splitRatio = 0.2
training set, test set = splitDataSet (dataset,
                         splitRatio)
print ('\n Dataset is split into training &
   testing set ")
print (" Training examples = {0} \n Testing
     examples = {1} ". format ( len (trainingset),
                         len (test set)))
Summaries = Summarize ByClass (trainingset);
predictions = perform - classification (summaries,
                   testset).
accuracy = get Accuracy (test set, prediction)
print ('\n Accuracy of the Naive Bayesian
     classifier is ", accuracy)
```

output :

```
Pima Indian Diabetes Dataset loaded :..
Total instances available : 768
Total attributes present : 8

First five instances of dataset .
  1 : [6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627,
                                   50.0, 1.0]
  2 : [1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351,
                               31.0, 0.0]
  3 : [ 8.0, 183.0, 64.0, 0.0, 0.0, 0.0, 23.3, 0.672
            32.0, 1.0]
  4 : [1.0, 89.0, 66.0, 23.0, 94.0, 28.1,
              0.167, 21.0, 0.0]
  5 : [0.0, 137.0, 40.0, 35.0, 168.0,
              43.1, 2.288, 33.0, 1.0]

Dataset is split into training & testing set

    Training examples = 615
    Testing examples = 153

Accuracy of the Naive Bayesian classifier is :
              75.16339869281046
```