5. write a program to implement the naive Bayes Bayesian classifier for a sample training data set stored as a .csv file. Compute the accuracy of the classifier, considering few 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 separateByClass(dataset):
    separated = { }
    for i in range(len(dataset)):
        x = dataset[i]
        if (x[-1] not in separated):
            separated[x[-1]] = []
```

Teacher's Signature : _____

```
        separated[x[-1]].append(x)
    return separated

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):
    separated = separateByClass(dataset);
    summary = {}
    for classValue, instances in separated.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 calculateClassProbalities(summaries, testVector):
    p = {}
    for classValue, classSummaries in summaries.items():
        p[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = testVector[i]
```

```
        p[classValue]*.estimateProbability(x,mean,stdev);
    return p

def predict(summaries,testVector):
    all_p = calculateClassProbabilities(summaries,
                                        testVector)
    bestLabel, bestProb = None, -1
    for lbl, p in all_p.items():
        if bestLabel is None or p > bestProb:
            bestProb = p
            bestLabel = lbl
    return bestLabel

def performClassification(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions

def getAccuracy(testSet, predictions):
    correct = 0
    for i in range(len(testSet)):
        if testSet[i][-1] == predictions[i]:
            correct += 1
    return (correct/float(len(testSet)))*100.0
dataset = loadCsv('C:/Users/user1/Desktop/ML Lab1
                  diabetes.csv');
```

Teacher's Signature :

```
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, ':', dataset[i])
SplitRatio = 0.2
trainingSet, testSet = splitDataset(dataset, splitRatio)
print('\n Dataset is split into training and testing set:')
print('Training examples = {0}\n Testing examples = {1}'.
                format(len(trainingSet), len(testSet)))
summaries = SummarizeByClass(trainingSet):
predictions = perform_classification(summaries, testSet)
accuracy = getAccuracy(testSet, predictions)
print('\n Accuracy of the Naive Baysian Classifier
                is:', accuracy)
```

output :-

Pima Indian Diabetes 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, 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 and testing set.

Training examples = 615
Testing examples = 153

Accuracy of the Naive Baysian Classifier is :
0.6535947712418301