

Write a program to implement the naive Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, considering few test data sets.

Aim:-

To write a program to implement the naive Bayesian classifier for a sample training data set.

Program:-

```
import csv
import random
import math

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):
    trainSize = int(len(dataset) * splitRatio);
    trainSet = []
    copy = list(dataset);
    while len(trainSet) < trainSize:
        index = random.randrange(len(copy));
        trainSet.append(copy.pop(index))
    return [trainSet, copy]

def separateByClass(dataset):
```

```
separated = {}
```

```
for i in range(len(dataset)):
```

```
    vector = dataset[i]
```

```
    if (vector[-1] not in separated):
```

```
        separated[vector[-1]] = []
```

```
        separated[vector[-1]].append(vector)
```

```
return separated
```

```
def mean(numbers):
```

```
    return sum(numbers)/float(len(numbers))
```

```
def stdev(numbers):
```

```
    avg = mean(numbers)
```

```
    variance = sum([pow(x-avg, 2) for x in numbers])/float(len(numbers)-1)
```

```
    return math.sqrt(variance)
```

```
def summarize(dataset):
```

```
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip
                                                           (*dataset)];
```

```
    del summaries[-1]
```

```
    return summaries
```

```
def summarizeByClass(dataset):
```

```
    separated = separateByClass(dataset);
```

```
    summaries = {}
```

```
    for classValue, instances in separated.items():
```

```
        summaries[classValue] = summarize(instances)
```

```
    return summaries
```

```
def calculateProbability(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, inputVector):
    probabilities = {}
    for classValue, classSummaries in summaries.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = inputVector[i]
            probabilities[classValue] *= calculateProbability(x, mean, stdev)
    return probabilities

def predict(summaries, inputVector):
    probabilities = calculateClassProbabilities(summaries, inputVector)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel

def getPredictions(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][0-1] == predictions[i]:
        correct += 1
return (correct / float(len(testSet))) * 100.0

def main():
    filename = '5data.csv'
    splitRatio = 0.67
    dataset = loadCsv(filename);
    trainingSet, testSet = splitDataset(dataset, splitRatio)
    print('Split %03 rows into train-%13 and test-%23 rows'.format(
        len(dataset),
        len(trainingSet), len(testSet)))
    summaries = summarizeByClass(trainingSet);
    predictions = getPredictions(summaries, testSet)
    accuracy = getAccuracy(testSet, predictions)
    print('Accuracy of the classifier is : %03%'.format(accuracy))
main()
```

### Result:-

Thus the program to implement the naïve Bayesian classifier for a sample training data set has been executed successfully.

## Output:-

Split 768 rows into train = 514 and test = 254 rows

Accuracy of the classifier is : 71.65354330708661%