Date .....

Expt. No. .....5..... Page No.....24.....

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

```
import csv
import random
import math
import statistics as st
def loadCsv(filename):
    lines = csv.reader(open("C:\\Users\\PVBhat\\
                Desktop\\csvfiles\\diabetes.csv","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)):
        x = dataset[i]
```

```
        if (x[-1] not in separated):
            separated[x[-1]] = []
        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 calculateClassProbabilities(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] *= calculateProbability(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\P V Bhat\Desktop\
                    csvfiles\diabetes.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, ":", dataset[i])
splitRatio = 0.6
training set, test set = splitDataset(dataset, splitRatio)
print("In Dataset is split into training and testing set.")
print("Training examples = %s In Testing examples = %s."
      %(len(training set), len(test set)))
summaries = summarizeBy(Class(training set))
predictions = perform_classification(summaries, test set)
accuracy = getAccuracy(test set, predictions)
print("In Accuracy of Naive Bayes Classifier is:", accuracy)
```

Dataset

6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

Output :

Pima Indian Diabetis 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, 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 = 460

Testing examples = 308

Accuracy of the Naive Bayes classifier is : 75.0