GAAWIN BAITO 4MT17CS039

PAGE NO : 15
DATE : 22/12/20
EXP.NO. : 5

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

```
import csv, random, math
import statistics as st
def load csv (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)
```

GALWIN BRITO 4MT17CS039

PAGE NO : 15
DATE : 22/12/20
EXP.NO. : 5

```
def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        v = dataset[i]
        if (v[-1] not in separated):
            separated[v[-1]] = []
        separated[v[-1]].append(v)
    return separated

def compute_mean_std(dataset):
    mean_std = [(sb.mean(attribute),
                 sb.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
```

ARUN'S

GALWIN BRITO 4MT17CS039

PAGE NO : 16
DATE : 22/12/20
EXP.NO. : 5

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

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

ARUN'S

GAUSSIAN NAIVE BAYES CLASSIFICATION

PAGE NO : 17
DATE : 22/12/20
EXP.NO. : 5

```
        bestlabel = 1bl
    return bestlabel

def perform_classification(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('diabetes.csv')
print('Data Loaded Diabetes Database
                                    Loaded...')
print('Total instances available:',
                                    len(dataset))
print('Total attributes present:',
                                    len(dataset[0])-1)
```

ARUN'S

GALWIN BRITO 4MT17CS039

PAGE NO : 18
DATE : 22/12/20
EXP.NO. : 5

```
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 testingset")
print("Training examples = {0} \n
   Test examples = {1}".format(len(training
                       set), len(testset)))
summaries = summarizeByClass(trainingset).
predictions = perform_classification
              (summaries, testsets)
accuracy = getAccuracy(testset, predictions)
print('\n Accuracy of the Naive Bayesian
        classifier is:", accuracy)
```

ARUN'S

output:

Pima Indians 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, 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 Bayesian classifier is: 75.16339869281046