Name : Uddip Yalamanchili
USN : 1PE16CS170

PROGRAM :

```
import csv
import math
import random
#Handle data
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

#Split dataset with ratio
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]

#Separate by Class
#function assumes that the last attribute (-1) is the class value
#The function returns a map of class values to lists of data instances.
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


#Calculate Mean
#We need to calculate the mean of each attribute for a class value
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)

#Summarize Dataset
#The zip function groups the values for each attribute across our data instances into their own lists
def summarize(dataset):
	summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
	del summaries[-1]
	return summaries

#Summarize attributes by class
def summarizeByClass(dataset):
	separated = separateByClass(dataset)
	summaries = {}
	for classValue, instances in separated.items():
		summaries[classValue] = summarize(instances)
	return summaries

#Calculate Gaussian Probability Density Function
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
#Calculate Class Probabilities
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
#Make a prediction
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
#Get predictions
def getPredictions(summaries, testSet):
        predictions = []
        for i in range(len(testSet)):
                result = predict(summaries, testSet[i])
                predictions.append(result)
        return predictions
#Get Accuracy
def getAccuracy(testSet, predictions):
        correct = 0
        for x in range(len(testSet)):
                if testSet[x][-1] == predictions[x]:
                        correct += 1
        return (correct/float(len(testSet)))*100.0

def main():
        filename = 'Data5.csv'
        splitRatio = 0.68
        dataset = loadCsv(filename)
        trainingSet, testSet = splitDataset(dataset, splitRatio)
        print('Split {0} rows into train = {1} and test = {2} rows'.format(len(dataset),len
(trainingSet),len(testSet)))
        #prepare model
        summaries = summarizeByClass(trainingSet)
        #test model
        predictions = getPredictions(summaries, testSet)
        accuracy = getAccuracy(testSet, predictions)
        print('Accuracy: {0}%'.format(accuracy))

main()
```

output :

```
Split 768 rows into train = 522 and test = 246 rows
Accuracy: 64.22764227642277%
uddipyalamanchili@pes:~/programs/pg5$
```