PROGRMA-5: Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

|  |
| --- |
| **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)) *# random index*  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][-1] == predictions[i]:  correct += 1  **return** (correct/float(len(testSet))) \* 100.0  **def** main():  filename = 'C:**\\**Users**\\**ISE**\\**Desktop**\\**Python-naive-Bayesian-Classifier1-master**\\**pima-indians-diabetes.csv'  splitRatio = 0.67  dataset = loadcsv(filename)  print("**\n** The length of the Data Set : ",len(dataset))    print("**\n** The Data Set Splitting into Training and Testing **\n**")  trainingSet, testSet = splitDataset(dataset, splitRatio)    print('**\n** Number of Rows in Training Set:**{0}** rows'.format(len(trainingSet)))  print('**\n** Number of Rows in Testing Set:**{0}** rows'.format(len(testSet)))    print("**\n** First Five Rows of Training Set:**\n**")  **for** i **in** range(0,5):  print(trainingSet[i],"**\n**")    print("**\n** First Five Rows of Testing Set:**\n**")  **for** i **in** range(0,5):  print(testSet[i],"**\n**")  summaries = summarizeByClass(trainingSet)  print("**\n** Model Summaries:**\n**",summaries)  predictions = getPredictions(summaries, testSet)  print("**\n**Predictions:**\n**",predictions)    accuracy = getAccuracy(testSet, predictions)  print('**\n** Accuracy: **{0}**%'.format(accuracy))  main() |