```
import csv
import random
import math

def loadcsv(filename):
    dataset = list(csv.reader(open(filename,"r")))
    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 = []
    trainSet,testSet = dataset[:trainSize],dataset[trainSize:]
    return [trainSet, testSet]

def mean(numbers):
    return sum(numbers)/(len(numbers))

def stdev(numbers):
    avg = mean(numbers)
    v = 0
    for x in numbers:
        v += (x-avg)**2
    return math.sqrt(v/(len(numbers)-1))

def summarizeByClass(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)
    summaries = {}
    for classValue, instances in separated.items():
        summaries[classValue] = [(mean(attribute), stdev(attribute)) for attribute in zip(*instances)][:-1]
    return summaries

def calculateProbability(x, mean, stdev):
    exponent = math.exp((-(x-mean)**2)/(2*(stdev**2)))
    return (1 / math.sqrt(2*math.pi*(stdev**2))) * exponent

def predict(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)
            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/(len(testSet))) * 100.0

filename = 'diabetes2.csv'
splitRatio = 0.9
dataset = loadcsv(filename)
actual = []
trainingSet, testSet = splitDataset(dataset, splitRatio)
for i in range(len(testSet)):
  vector = testSet[i]
  actual.append(vector[-1])
print('Split {0} rows into train={1} and test={2} rows'.format(len(dataset), len(trainingSet), len(testSet)))
summaries = summarizeByClass(trainingSet) #will have (mean,sd) for all attributes.(for class 1 & 0 separately)
```

```
predictions = getPredictions(summaries, testSet)
print('\nActual values:\n',actual)
print("\nPredictions:\n",predictions)
accuracy = getAccuracy(testSet, predictions)
print("Accuracy",accuracy)
```

```
Split 768 rows into train=691 and test=77 rows

Actual values:
 [1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0,

Predictions:
 [1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0,
Accuracy 76.62337662337663
```

```
dataset = list(csv.reader(open(filename,"r")))
for i in range(len(dataset)):
    dataset[i] = [float(x) for x in dataset[i]]
print(dataset)
```

```
[[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0, 1.0], [1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0, 0.0], [8.0, 183.0, 64.0, 0.
```

Colab paid products - Cancel contracts here

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.