In [114…
```
import csv
import random
import math
```

In [115…
```
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
```

In [116…
```
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]
```

In [26]:
```
def seperateByClass(dataset):
    seperated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if( vector[-1] not in seperated):
            seperated[vector[-1]] = []
        seperated[vector[-1]].append(vector)
    return seperated
```

In [118…
```
def mean(numbers):
    return sum(numbers)/float(len(numbers))
```

In [119…
```
def stdev(numbers):
    avg = mean(numbers)
    variance = sum([pow(x-avg,2) for x in numbers])/float(len(numbers)-1)
    return math.sqrt(variance)
```

In [120…
```
def summarize(dataset) :
    summaries = [(mean(attribute),stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries
```

In [121…
```
def summarizeByClass(dataset):
    separated=seperateByClass(dataset)
    summaries = {}
    for classValue , instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries
```

In [122…
```
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
```

In [123…
```
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
```

In [124…
```
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
```

In [125…

```
def getPredictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions
```

In [126…

```
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
```

In [128…

```
def main():
    filename = 'P5_haberman_dataset.csv'
    splitRatio = 0.67
    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()
```

```
Split 306 rows into train=205 and test=101 rows
Accuracy: 72.27722772277228%
```