```
import csv
import random
import math
def loadCsv(filename):
   lines = csv.reader(open(filename, "r"));
   dataset = list(lines)
   for i in range(len(dataset)):
#converting strings into numbers for processing
      dataset[i] = [float(x) for x in dataset[i]]
   return dataset
def splitDataset(dataset, splitRatio):
#67% training size
   trainSize = int(len(dataset) * splitRatio);
   trainSet = []
   copy = list(dataset);
   while len(trainSet) < trainSize:
#generate indices for the dataset list randomly to pick ele for training
data
      index = random.randrange(len(copy));
      trainSet.append(copy.pop(index))
   return [trainSet, copy]
def separateByClass(dataset):
   separated = {}
#creates a dictionary of classes 1 and 0 where the values are the
instacnes belonging to each class
   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 is a dic of tuples(mean,std) for each class value
      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():#class and
attribute information as mean and sd
    probabilities[classValue] = 1
    for i in range(len(classSummaries)):
      mean, stdev = classSummaries[i] #take mean and sd of every
attribute for class 0 and 1 seperaely
      x = inputVector[i] #testvector's first attribute
      probabilities[classValue] *= calculateProbability(x, mean,
stdev);#use normal dist
  return probabilities
def predict(summaries, inputVector):
  probabilities = calculateClassProbabilities(summaries, inputVector)
  bestLabel, bestProb = None, -1
  for classValue, probability in probabilities.items():#assigns that
class which has he highest prob
    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 = '6.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 of the classifier is : {0}%'.format(accuracy))
main()
```