In [1]:
```
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))
        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))
    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='naviebayes.csv'
    splitRatio=0.99
    dataset=loadcsv(filename)
    trainingset,testset=splitDataset(dataset,splitRatio)
    print(f'Split {len(dataset)} rows into train={len(trainingset)} and test={len(testset)}')
    summaries=summarizeByClass(trainingset)
    predictions=getPredictions(summaries,testset)
    accuracy=getAccuracy(testset,predictions)
    print('Accuracy = {0}%'.format(accuracy))
    print('Prediction = {0}'.format(predictions))
    expres=[]
    for row in testset:
        expres.append(row[-1])
    print('Expected result: ',expres)
main()
```

```
Split 768 rows into train=760 and test=8
Accuracy = 75.0%
Prediction = [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
Expected result:  [1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0]
```