5.Problem: 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.

References : https://gist.github.com/wzyuliyang/883bb84e88500e32b833 https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

In [2]:
```
# Example of Naive Bayes implemented from Scratch in Python
#http://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
import csv
import random
import math

# 1.Data Handling
# 1.1 Loading the Data from csv file of Pima indians diabetes dataset.
def loadcsv(filename):
    lines = csv.reader(open(filename, "r"))
    dataset = list(lines)
    for i in range(len(dataset)):
        # converting the attributes from string to floating point numbers
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset

#1.2 Splitting the Data set into Training Set
def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    while len(trainSet) < trainSize:
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
        index = random.randrange(len(copy)) # random index
        trainSet.append(copy.pop(index))
    return [trainSet, copy]

#2.Summarize Data
#The naive bayes model is comprised of a
#summary of the data in the training dataset.
#This summary is then used when making predictions.
#involves the mean and the standard deviation for each attribute, by cl
ass value

#2.1: Separate Data By Class
#Function to categorize the dataset in terms of classes
#The function assumes that the last attribute (-1) is the class value.
#The function returns a map of class values to lists of data instances.
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

#The mean is the central middle or central tendency of the data,
# and we will use it as the middle of our gaussian distribution
# when calculating probabilities

#2.2 : Calculate Mean
def mean(numbers):
    return sum(numbers)/float(len(numbers))

#The standard deviation describes the variation of spread of the data,
#and we will use it to characterize the expected spread of each attribu
te
#in our Gaussian distribution when calculating probabilities.

#2.3 : Calculate Standard Deviation
def stdev(numbers):
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

#2.4 : Summarize Dataset
#Summarize Data Set for a list of instances (for a class value)
#The zip function groups the values for each attribute across our data
 instances
#into their own lists so that we can compute the mean and standard devi
ation values
#for the attribute.

def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in z
ip(*dataset)]
    del summaries[-1]
    return summaries

#2.5 : Summarize Attributes By Class
#We can pull it all together by first separating our training dataset i
nto
#instances grouped by class.Then calculate the summaries for each attri
bute.

def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries

#3.Make Prediction
#3.1 Calculate Probaility Density Function
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

#3.2 Calculate Class Probabilities
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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

#3.3 Prediction : look for the largest probability and return the assoc
iated class
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

#4.Make Predictions
# Function which return predictions for list of predictions
# For each instance

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

#5. Computing Accuracy
def getAccuracy(testSet, predictions):
    correct = 0
    for i in range(len(testSet)):
        if testSet[i][-1] == predictions[i]:
            correct += 1
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
    return (correct/float(len(testSet))) * 100.0

#Main Function
def main():
    filename = 'C:\\Users\\Dr.Thyagaraju\\Desktop\\Data\\pima-indians-d
iabetes.csv'
    splitRatio = 0.67
    dataset = loadcsv(filename)

    #print("\n The Data Set :\n",dataset)
    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(train
ingSet)))
    print('\n Number of Rows in Testing Set:{0} rows'.format(len(testSe
t)))

    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")

    # prepare model
    summaries = summarizeByClass(trainingSet)
    print("\n Model Summaries:\n",summaries)

    # test model
    predictions = getPredictions(summaries, testSet)
    print("\nPredictions:\n",predictions)

    accuracy = getAccuracy(testSet, predictions)
    print('\n Accuracy: {0}%'.format(accuracy))
main()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
 The length of the Data Set :  768

 The Data Set Splitting into Training and Testing


 Number of Rows in Training Set:514 rows

 Number of Rows in Testing Set:254 rows

 First Five Rows of Training Set:

[4.0, 116.0, 72.0, 12.0, 87.0, 22.1, 0.463, 37.0, 0.0]

[0.0, 84.0, 64.0, 22.0, 66.0, 35.8, 0.545, 21.0, 0.0]

[0.0, 162.0, 76.0, 36.0, 0.0, 49.6, 0.364, 26.0, 1.0]

[10.0, 101.0, 86.0, 37.0, 0.0, 45.6, 1.136, 38.0, 1.0]

[5.0, 78.0, 48.0, 0.0, 0.0, 33.7, 0.654, 25.0, 0.0]


 First Five Rows of Testing Set:

[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0, 0.0]

[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0, 1.0]

[4.0, 110.0, 92.0, 0.0, 0.0, 37.6, 0.191, 30.0, 0.0]

[10.0, 139.0, 80.0, 0.0, 0.0, 27.1, 1.441, 57.0, 0.0]

[7.0, 100.0, 0.0, 0.0, 0.0, 30.0, 0.484, 32.0, 1.0]


 Model Summaries:
 {0.0: [(3.3474320241691844, 3.045635385378286), (111.54380664652568, 2
6.040069054720693), (68.45921450151057, 18.15540652389224), (19.9456193
3534743, 14.709615608767137), (71.50151057401813, 101.04863439385403),
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
(30.863141993957708, 7.207208162103949), (0.4341842900302116, 0.2960911
906946818), (31.613293051359516, 12.100651311117689)], 1.0: [(4.4699453
55191257, 3.7369440851983082), (139.3879781420765, 33.733070931373234),
(71.14754098360656, 20.694403393963842), (22.92896174863388, 18.1519950
92528765), (107.97814207650273, 146.92526156736633), (35.2863387978142
2, 7.783342260348583), (0.5569726775956286, 0.3942245334398509), (36.78
688524590164, 11.174610282702282)]}

Predictions:
 [0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0,
0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0,
0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0,
0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0,
0.0, 1.0, 1.0, 1.0, 1.0, 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, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0,
1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0,
0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0,
1.0, 0.0]

 Accuracy: 80.31496062992126%
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD