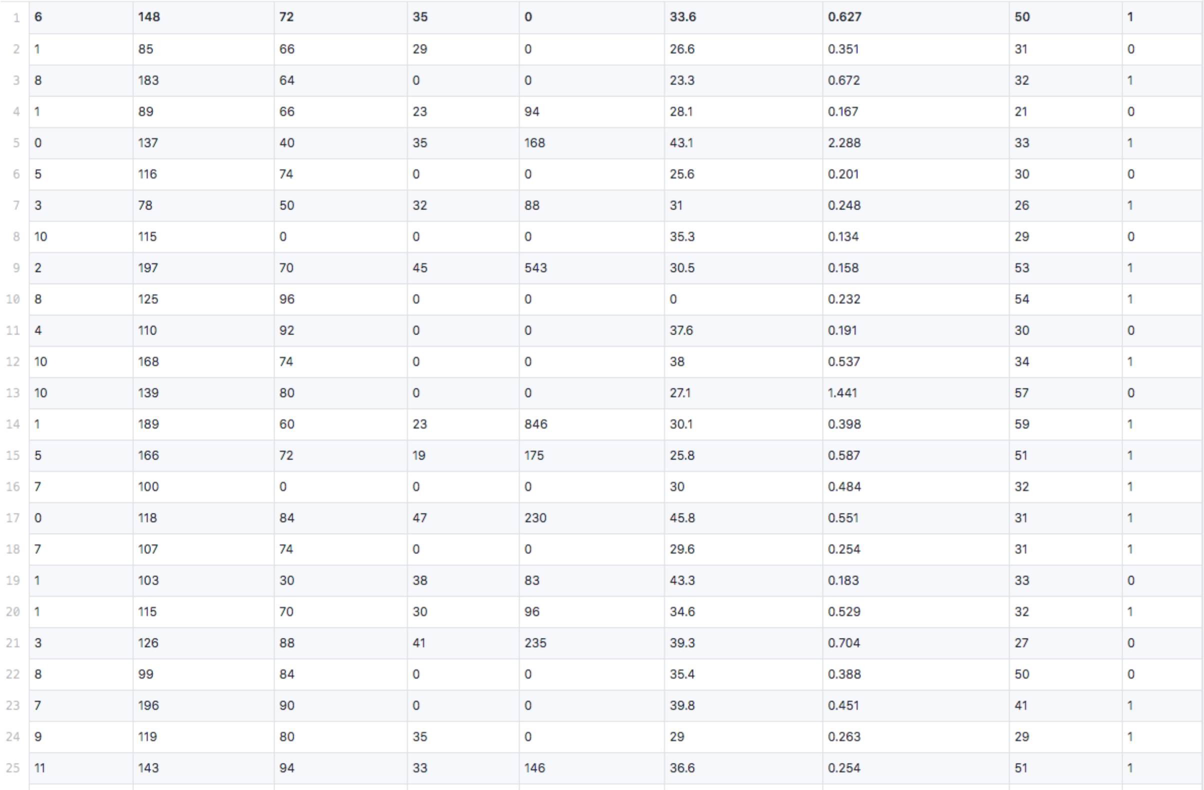
NAÏVE BAYESIAN CLASSIFIER

DEVELOPED BY: VISHRUTH KHARE
ROLL NUMBER: 2K18/CO/393

IDE: PyCharm
Compiler: Terminal (Python 3.6)
Dataset: Diabetes dataset
Sample dataset:



6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1

Source Code

```
import csv
import statistics as st
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 calprob(x, mean, stdev):
    exponent = math.exp(-(math.pow(x-mean,2)/(2*math.pow(stdev,2))))
    return (1 / (math.sqrt(2*math.pi) * stdev*stdev)) * exponent

dataset=loadCsv('pima-indians-diabetes.csv')
size=0.5 #50% distribution
train=[]
test=[]

#splitting the dataset into training and testing
for i in range(int(len(dataset)*size)):
    test.append(dataset[i])
for i in range(int(len(dataset)*size+1),len(dataset)):
    train.append(dataset[i])
print('The lenth of the training set',len(train))
print('The length of the testing set',len(test))

classes=[]
for i in dataset:
    if(i[-1] not in classes):
        classes.append(i[-1]) #list of all unique class values stored in the list named classes
classdict={} #dictionary that is intended to contain all the rows associated with each class value
classdict1={} #dictionary that is intended to contain (mean,standard deviation) of every attribute associated with each class value
classprob={} #dictionary that is intended to contain probabilites of the given sample falling into the class values
#initialization
for i in classes:
    classdict[i]=[]
    classdict1[i]=[]
    classprob[i]=1

#for each class value, all the rows having that class value are appended
for i in classes:
    for row in train:
        if row[-1]==i:
            classdict[i].append(row[:-1])

#for each class value, the tuple(mean, stdev) for each attribute is appended
for classval,datt in classdict.items():
    for col in zip(*datt):
        classdict1[classval].append((st.mean(col),st.stdev(col)))
```

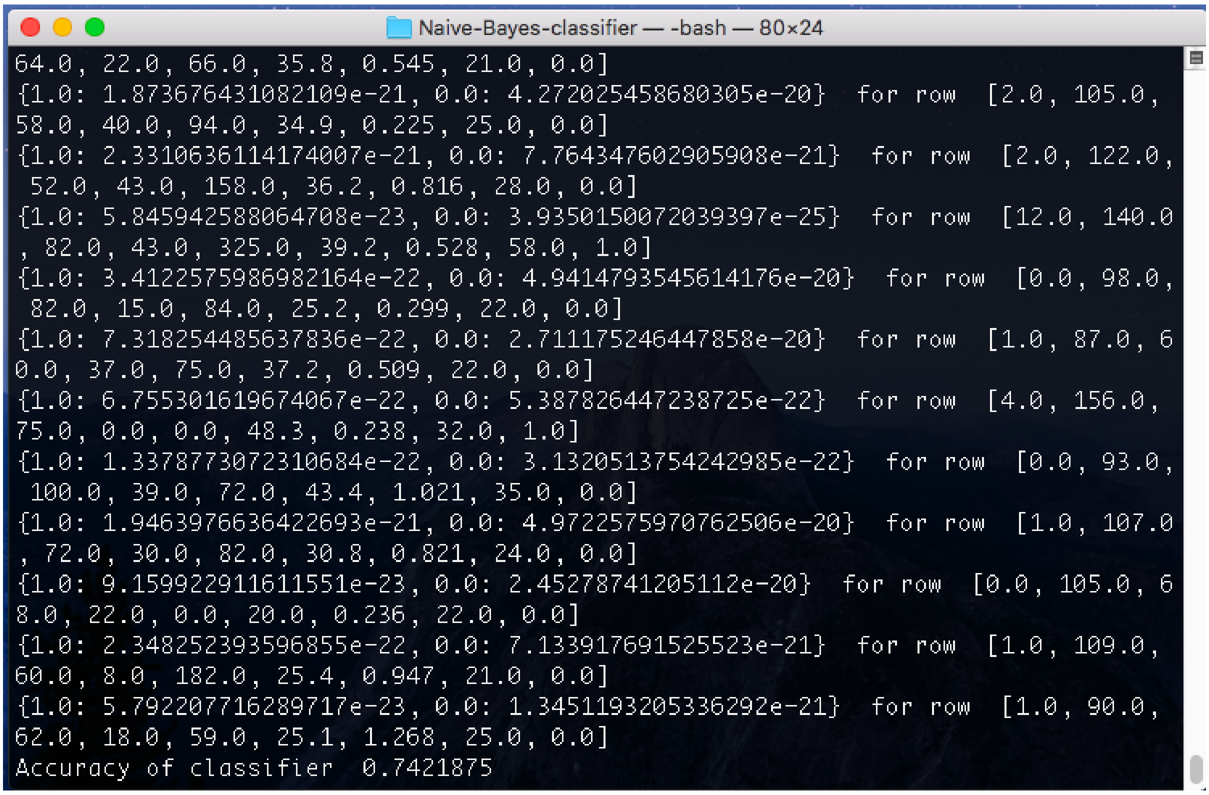
```
count=0 #counter to count the number of correctly classified instances
#calculating class probabilites
for row in test:
    for i in classes:
        classprob[i]=1
    for classval,datt in classdict1.items():
        for i in range(len(row[:-1])):
            mean,std=datt[i]
            x=row[i]
            classprob[classval]*=calprob(x,mean,std) #refer gaussian naive bayes theory
    print(classprob," for row ",row)
    #calculating accuracy
    mini=0
    cl=0
    for c,d in classprob.items():
        if d>mini:
            mini=d
            cl=c

    if row[-1]==cl:
        count+=1

acc=count/len(test)
print("Accuracy of classifier ",acc)
```

Output:

Screenshot with the accuracy for the test-set and training-set divided into 1:1 ratio :



```
64.0, 22.0, 66.0, 35.8, 0.545, 21.0, 0.0]
{1.0: 1.873676431082109e-21, 0.0: 4.272025458680305e-20}  for row  [2.0, 105.0,
58.0, 40.0, 94.0, 34.9, 0.225, 25.0, 0.0]
{1.0: 2.331063611417400 7e-21, 0.0: 7.764347602905908e-21}  for row  [2.0, 122.0,
 52.0, 43.0, 158.0, 36.2, 0.816, 28.0, 0.0]
{1.0: 5.845942588064708e-23, 0.0: 3.935015007203939 7e-25}  for row  [12.0, 140.0
, 82.0, 43.0, 325.0, 39.2, 0.528, 58.0, 1.0]
{1.0: 3.4122575986982164e-22, 0.0: 4.9414793545614176e-20}  for row  [0.0, 98.0,
 82.0, 15.0, 84.0, 25.2, 0.299, 22.0, 0.0]
{1.0: 7.318254485637836e-22, 0.0: 2.711175246447858e-20}  for row  [1.0, 87.0, 6
0.0, 37.0, 75.0, 37.2, 0.509, 22.0, 0.0]
{1.0: 6.755301619674067e-22, 0.0: 5.387826447238725e-22}  for row  [4.0, 156.0,
75.0, 0.0, 0.0, 48.3, 0.238, 32.0, 1.0]
{1.0: 1.3378773072310684e-22, 0.0: 3.1320513754242985e-22}  for row  [0.0, 93.0,
 100.0, 39.0, 72.0, 43.4, 1.021, 35.0, 0.0]
{1.0: 1.9463976636422693e-21, 0.0: 4.9722575970762506e-20}  for row  [1.0, 107.0
, 72.0, 30.0, 82.0, 30.8, 0.821, 24.0, 0.0]
{1.0: 9.159922911611551e-23, 0.0: 2.45278741205112e-20}  for row  [0.0, 105.0, 6
8.0, 22.0, 0.0, 20.0, 0.236, 22.0, 0.0]
{1.0: 2.348252393596855e-22, 0.0: 7.133917691525523e-21}  for row  [1.0, 109.0,
60.0, 8.0, 182.0, 25.4, 0.947, 21.0, 0.0]
{1.0: 5.792207716289717e-23, 0.0: 1.3451193205336292e-21}  for row  [1.0, 90.0,
62.0, 18.0, 59.0, 25.1, 1.268, 25.0, 0.0]
Accuracy of classifier  0.7421875
```