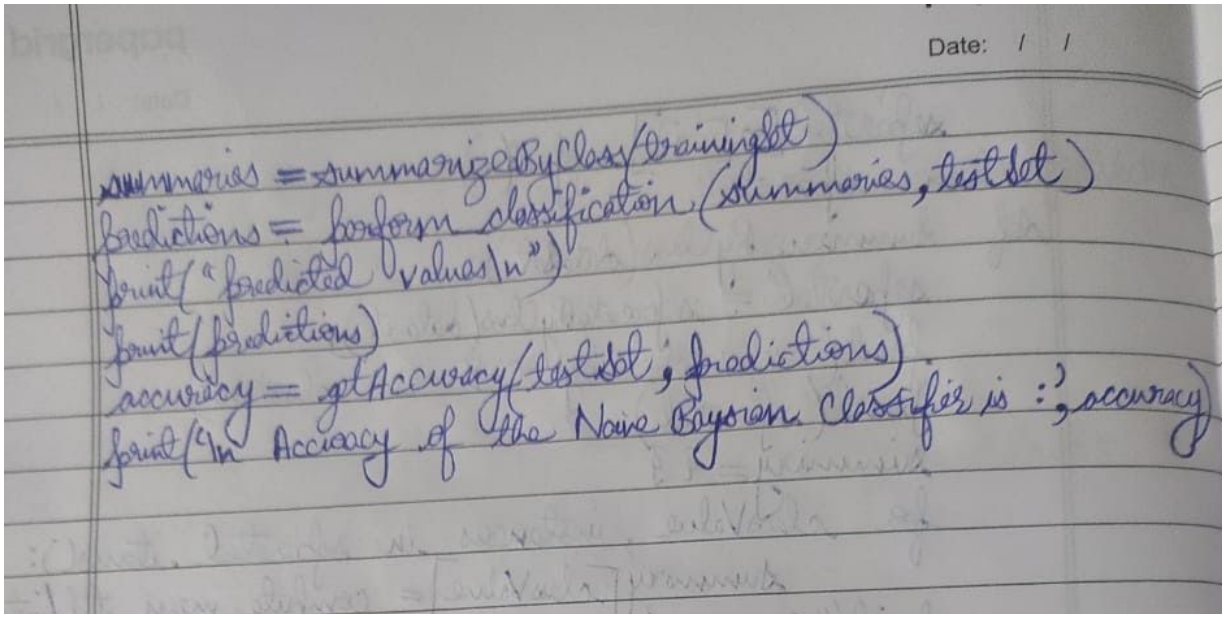
Program-5

Program – 5
(Naive – Bayesian Classifier)

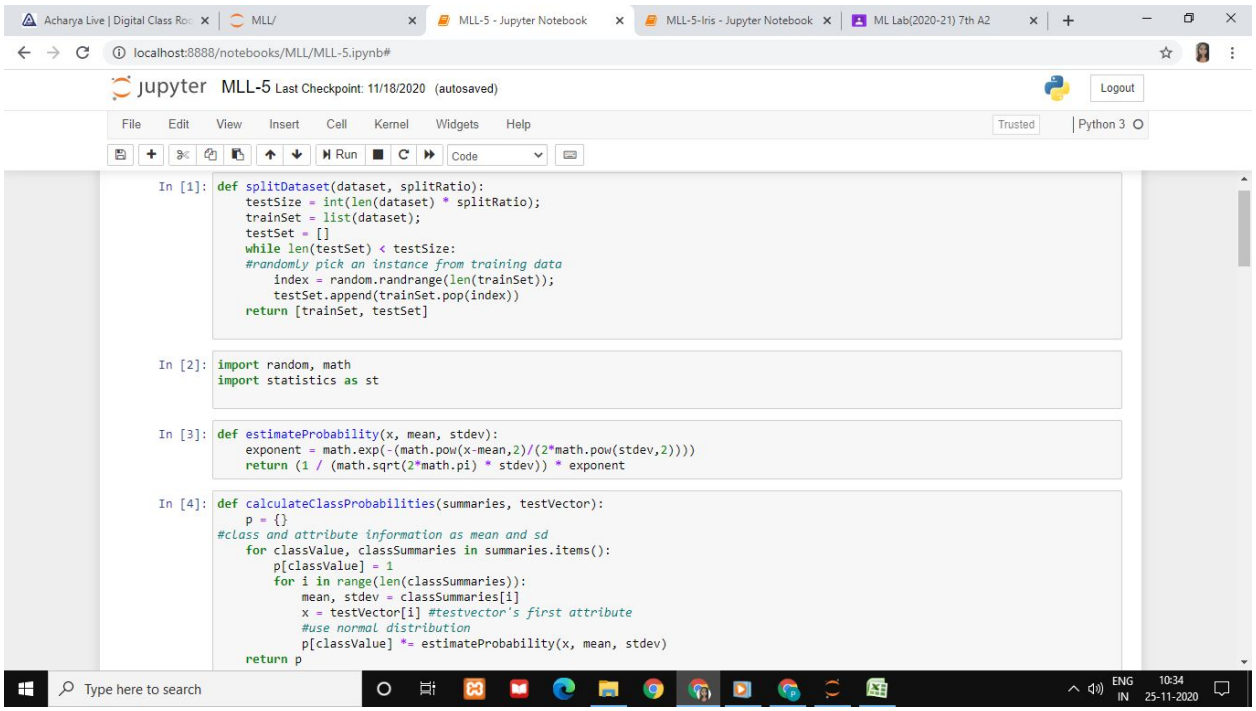
```
def splitDataset(dataset, splitRatio):
    testSize = int(len(dataset) * splitRatio);
    trainSet = list(dataset);
    testSet = []
    while len(testSet) < testSize:
        index = randrange(len(trainSet));
        testSet.append(trainSet.pop(index))
    return [trainSet, testSet]

import random, math
import statistics as st

def estimateProbability(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, testVector):
    p={}
    for classValue, classSummaries in summaries.item():
        p[classValue]=1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = testVector[i]
            p[classValue] *= estimateProbability(x, mean,
                                                 stdev)

    return p
```

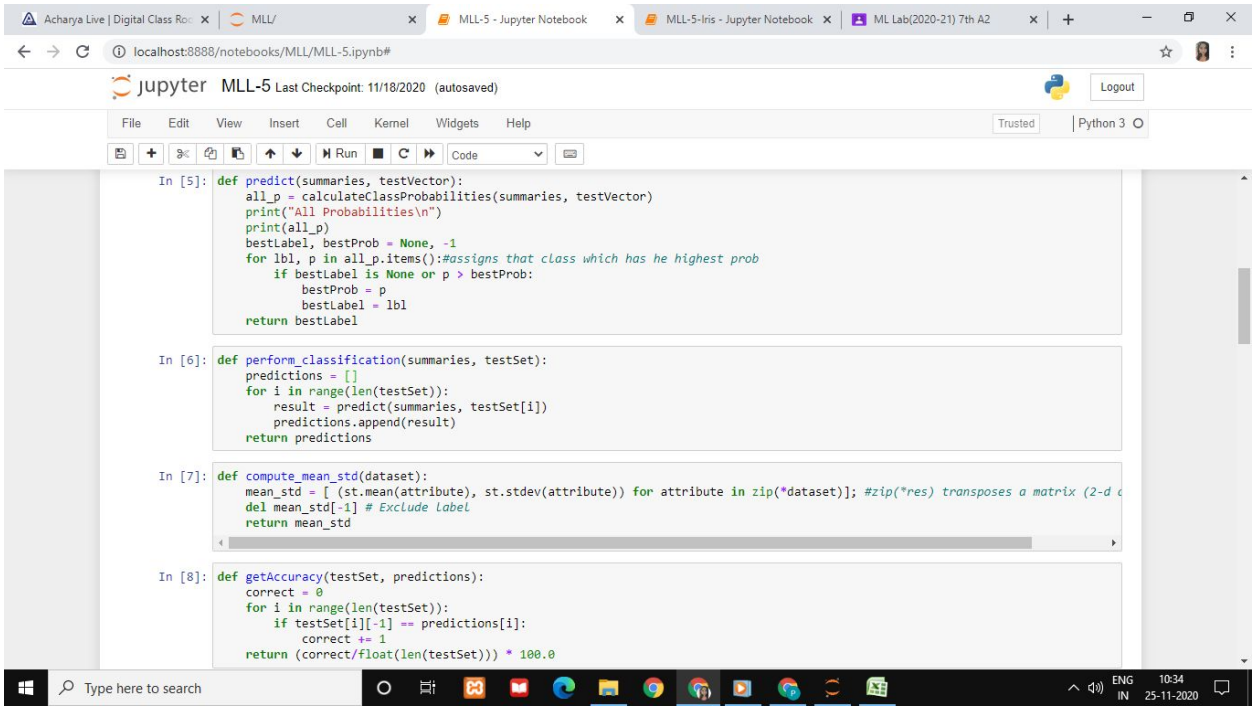
```
def predict(summaries, testVector):
    all_p = calculateClassProbabilities(summaries, testVector)
    print("All Probabilities\n")
    print(all_p)
    bestLabel, bestProb = None, -1
    for lbl, p in all_p.items():
        if bestLabel is None or p > bestProb:
            bestProb = p
            bestLabel = lbl
    return bestLabel
def perform_classification(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions
def compute_mean_std(dataset):
    mean_std = [(st.mean(attribute), st.stdev(attribute))
                for attribute in zip(*dataset)];
    del mean_std[-1]
    return mean_std
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 separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        x = dataset[i]
        if(x[-1] not in separated):
            separated[x[-1]] = []
```

```
        separated[x[-1]].append(x)
    return separated
def summarizeByClass(dataset):
    separated = separateByClass(dataset);
    print("separated by 0 & 1 class\n")
    print(separated)
    summary = {}
    for classValue, instances in separated.items():
        summary[classValue] = compute mean std(instances)
    print("Summary of mean and standard deviation")
    print(summary)
    return summary
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
import numpy as np
import pandas as pd
import csv
dataset = local.loadCsv('data5.csv');
print('Pima Indian Diabetes Dataset loaded...')
print('Total instances available :', len(dataset))
print('Total attributes present :', len(dataset[0]) - 1)
print("First five instances of dataset :")
for i in range(5):
    print(i+1, ':', dataset[i])
splitRatio = 0.2
trainingset, testset = splitDataset(dataset, splitRatio)
print('\n Dataset is split into training and testing set.')
print('Training set size', len(trainingset))
print('Test set size', len(testset))
```


Date: / /
summaries = summarizeByClass(trainingSet)
predictions = getPredictions(summaries, testSet)
print('predicted values\n')
print(predictions)
accuracy = getAccuracy(testSet, predictions)
print('\n Accuracy of the Naive Bayesian Classifier is :', accuracy)

OUTPUT


Acharya Live | Digital Class Ro
MLL/
MLL-5 - Jupyter Notebook
MLL-5-Iris - Jupyter Notebook
ML Lab(2020-21) 7th A2
localhost:8888/notebooks/MLL/MLL-5.ipynb#
jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]:
def splitDataset(dataset, splitRatio):
testSize = int(len(dataset) * splitRatio);
trainSet = list(dataset);
testSet = []
while len(testSet) < testSize:
#randomly pick an instance from training data
index = random.randrange(len(trainSet));
testSet.append(trainSet.pop(index))
return [trainSet, testSet]
In [2]:
import random, math
import statistics as st
In [3]:
def estimateProbability(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 [4]:
def calculateClassProbabilities(summaries, testVector):
p = {}
#class and attribute information as mean and sd
for classValue, classSummaries in summaries.items():
p[classValue] = 1
for i in range(len(classSummaries)):
mean, stdev = classSummaries[i]
x = testVector[i] #testvector's first attribute
#use normal distribution
p[classValue] *= estimateProbability(x, mean, stdev)
return p
Type here to search
ENG
IN
10:34
25-11-2020



jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

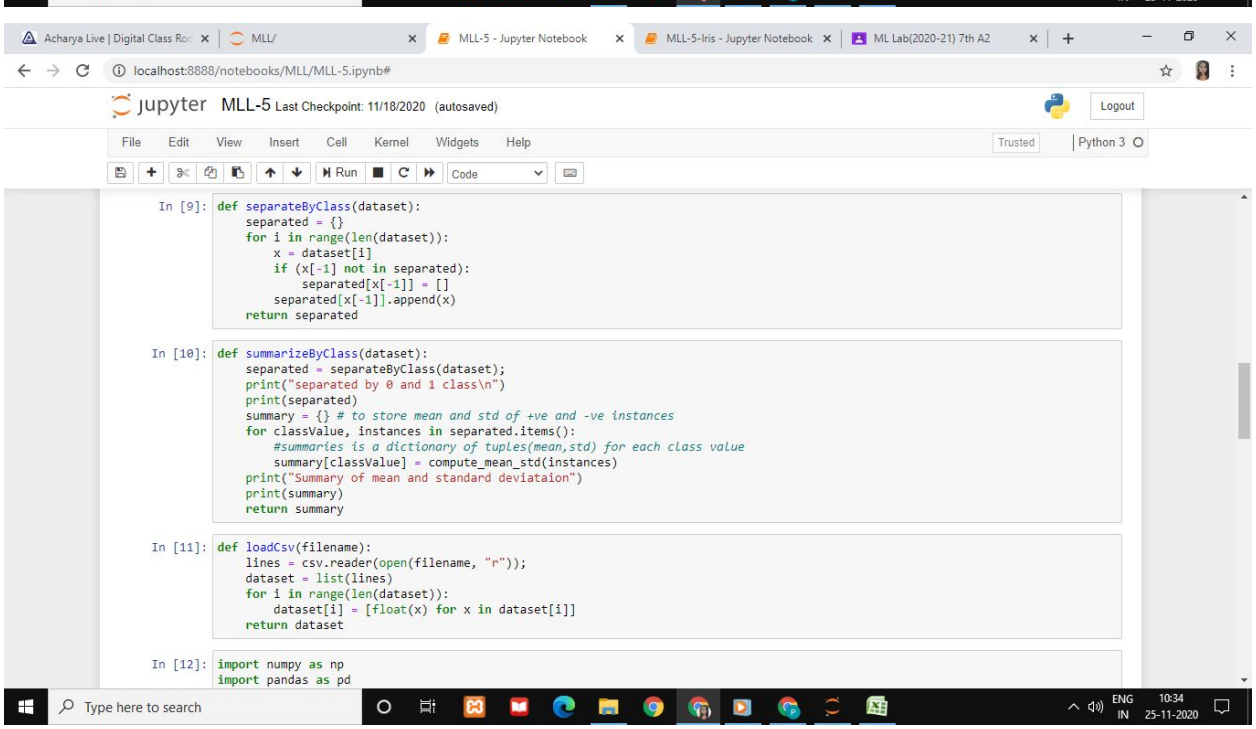
Run Code

In [5]:
```
def predict(summaries, testVector):
    all_p = calculateClassProbabilities(summaries, testVector)
    print("All Probabilities\n")
    print(all_p)
    bestLabel, bestProb = None, -1
    for lbl, p in all_p.items():#assigns that class which has he highest prob
        if bestLabel is None or p > bestProb:
            bestProb = p
            bestLabel = lbl
    return bestLabel
```

In [6]:
```
def perform_classification(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions
```

In [7]:
```
def compute_mean_std(dataset):
    mean_std = [ (st.mean(attribute), st.stdev(attribute)) for attribute in zip(*dataset)]; #zip(*res) transposes a matrix (2-d c
    del mean_std[-1] # Exclude Label
    return mean_std
```

In [8]:
```
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
```



jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

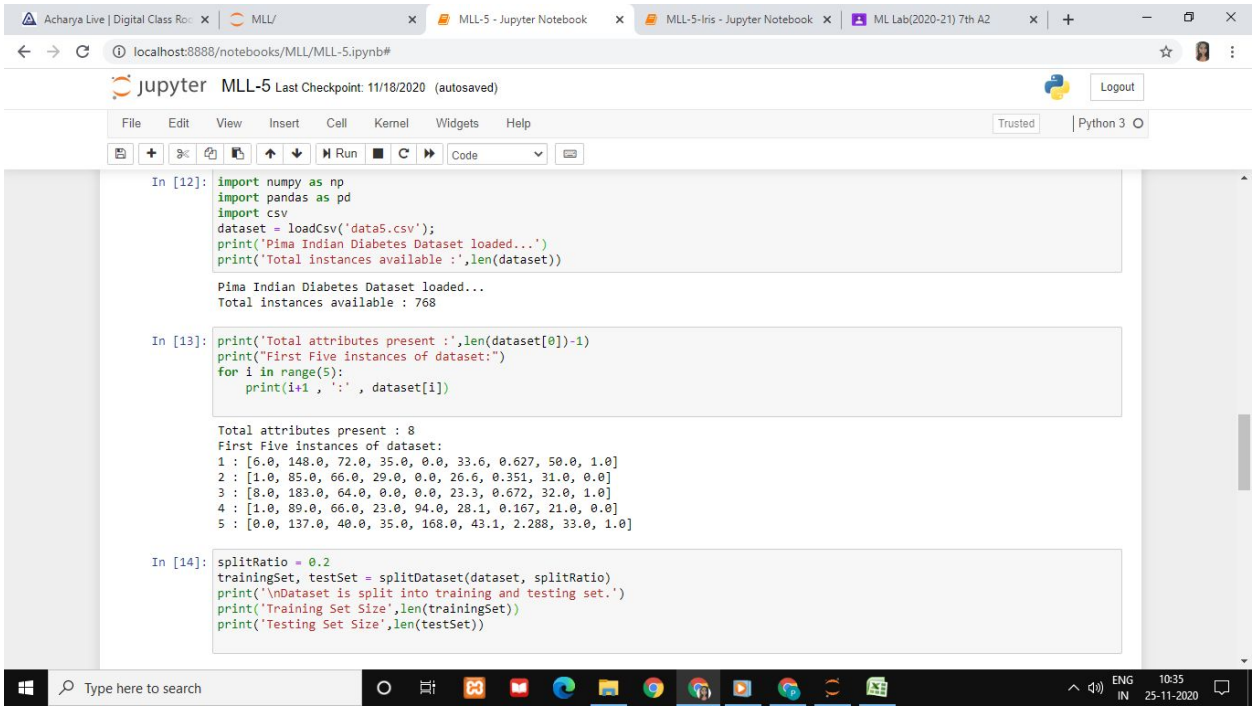
Run Code

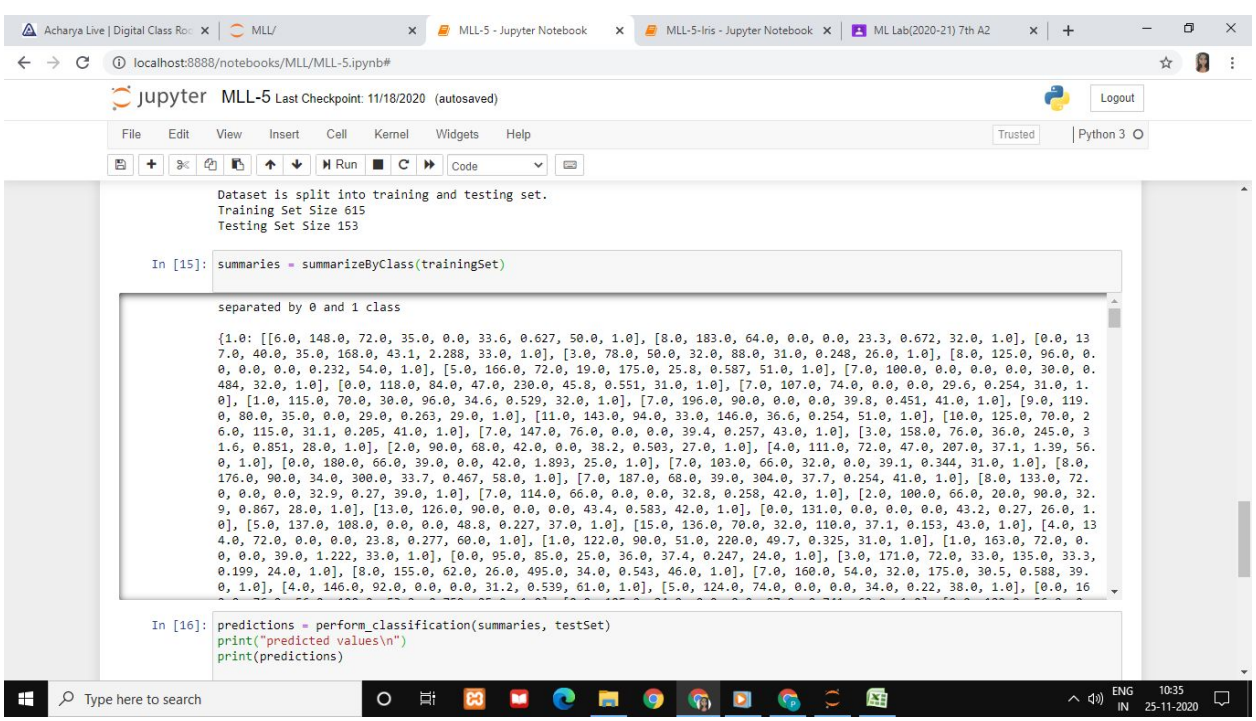
In [9]:
```
def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        x = dataset[i]
        if (x[-1] not in separated):
            separated[x[-1]] = []
        separated[x[-1]].append(x)
    return separated
```

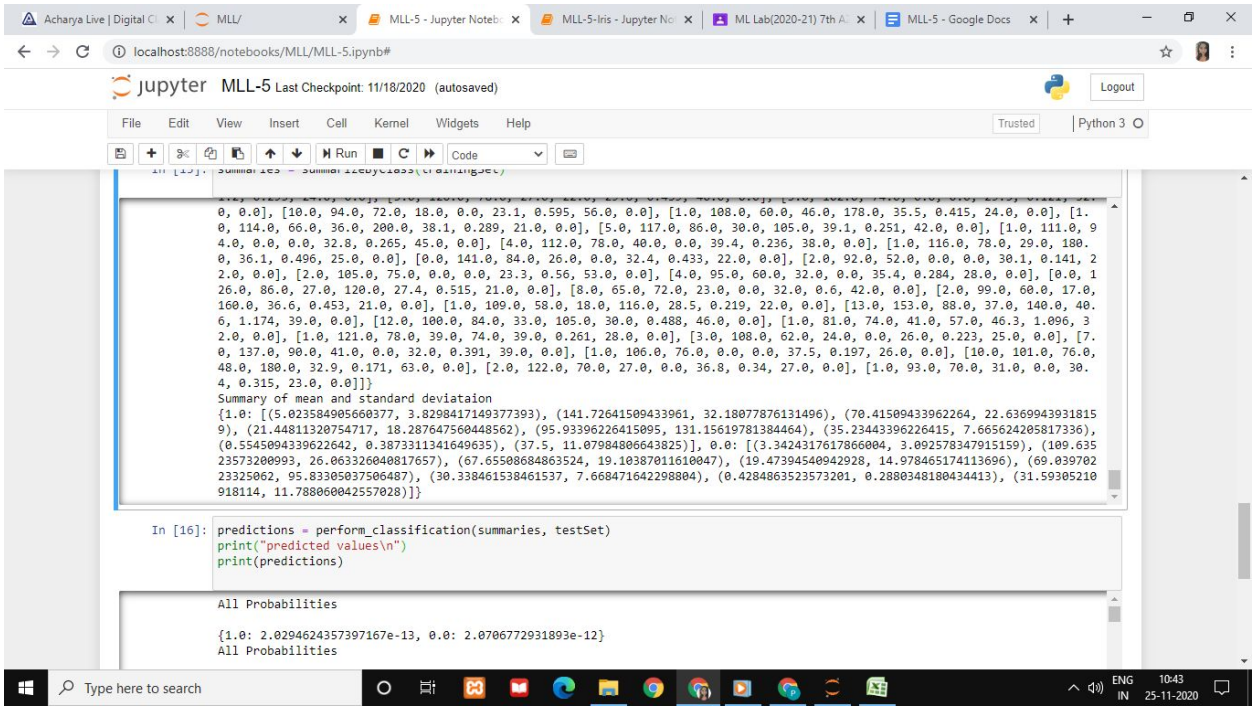
In [10]:
```
def summarizeByClass(dataset):
    separated = separateByClass(dataset);
    print("separated by 0 and 1 class\n")
    print(separated)
    summary = {} # to store mean and std of +ve and -ve instances
    for classValue, instances in separated.items():
        #summaries is a dictionary of tuples(mean,std) for each class value
        summary[classValue] = compute_mean_std(instances)
    print("Summary of mean and standard deviataion")
    print(summary)
    return summary
```

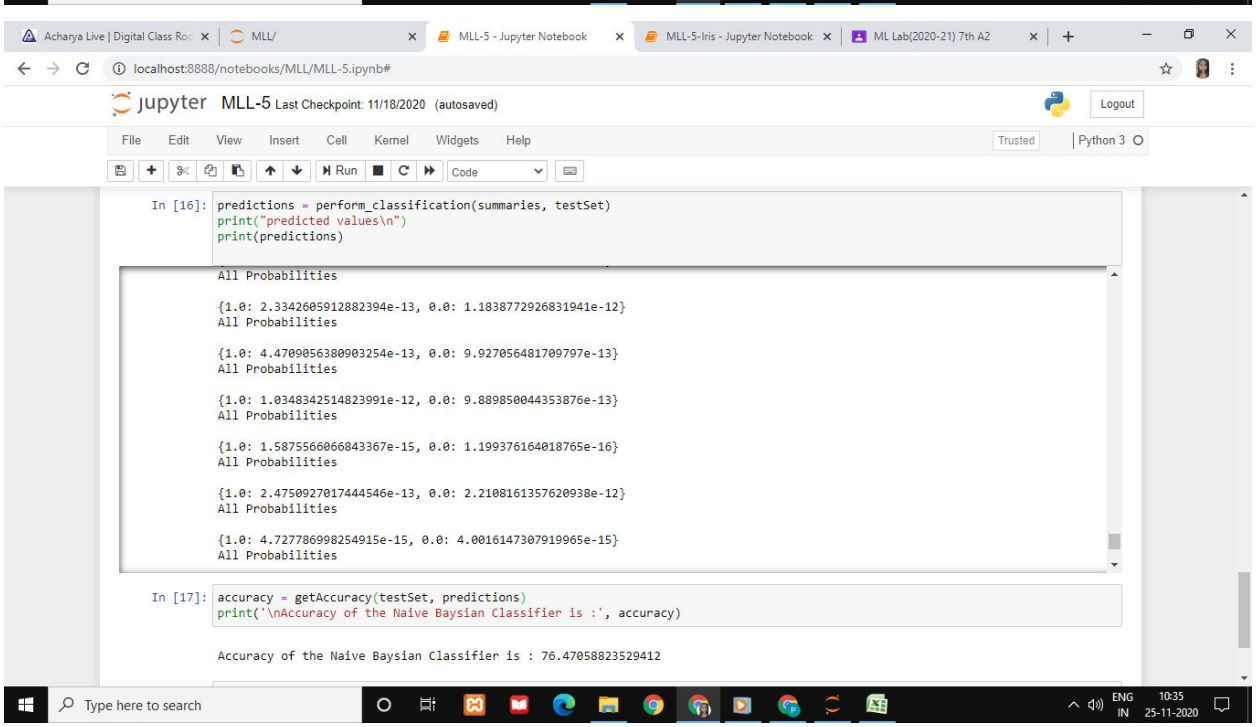
In [11]:
```
def loadCsv(filename):
    lines = csv.reader(open(filename, "r"));
    dataset = list(lines)
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset
```

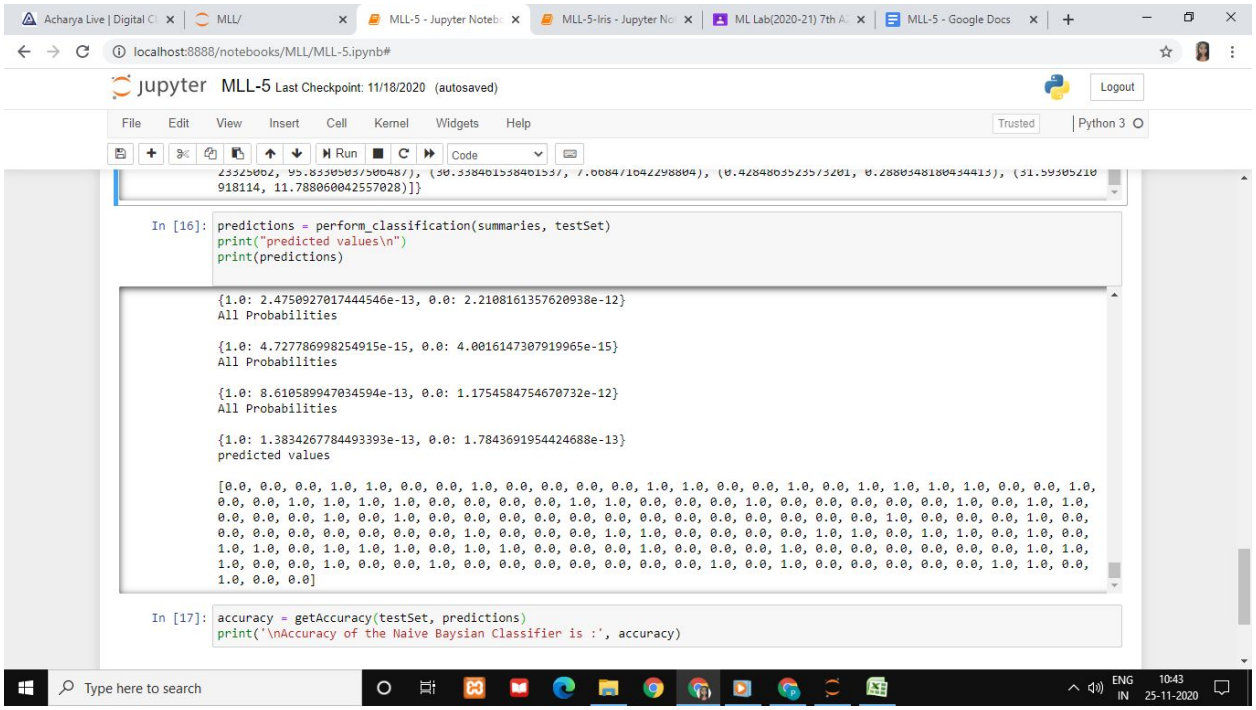
In [12]:
```
import numpy as np
import pandas as pd
```

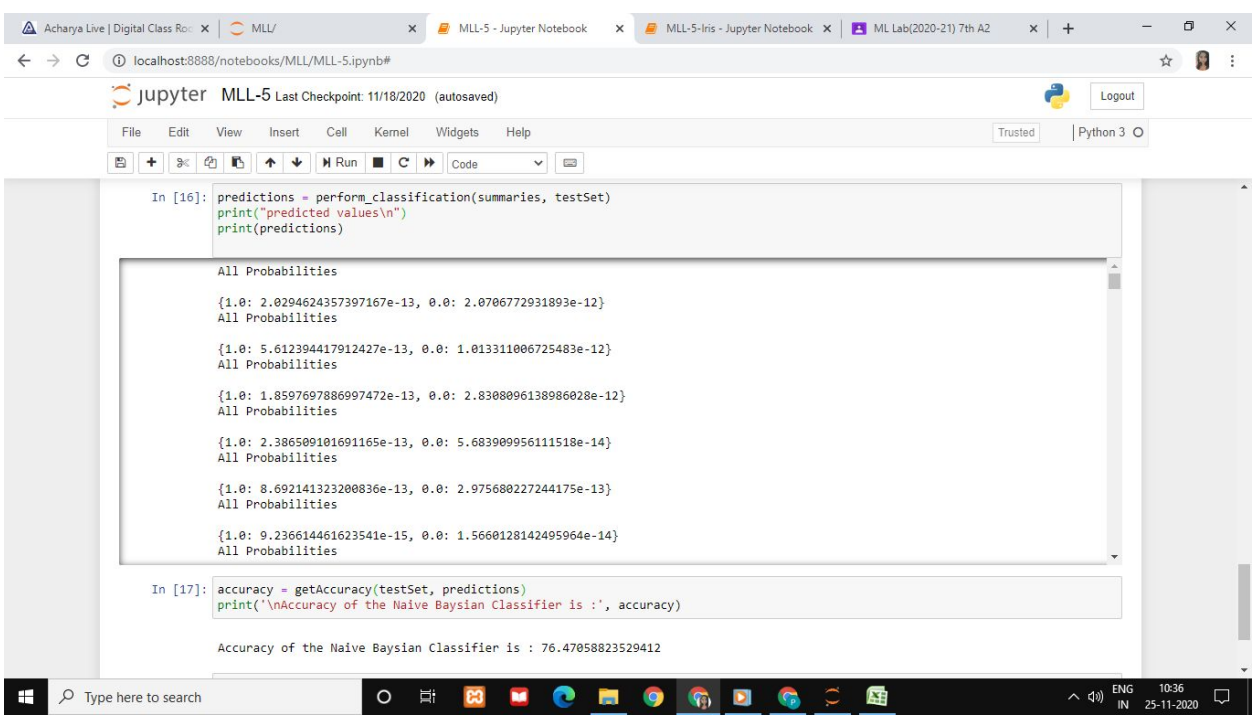

Acharya Live | Digital Class Roc
MLL/
MLL-5 - Jupyter Notebook
MLL-5-Iris - Jupyter Notebook
ML Lab(2020-21) 7th A2
localhost:8888/notebooks/MLL/MLL-5.ipynb#
jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [12]:
import numpy as np
import pandas as pd
import csv
dataset = loadCsv('data5.csv');
print('Pima Indian Diabetes Dataset loaded...')
print('Total instances available :',len(dataset))
Pima Indian Diabetes Dataset loaded...
Total instances available : 768
In [13]:
print('Total attributes present :',len(dataset[0])-1)
print("First Five instances of dataset:")
for i in range(5):
print(i+1 , ':' , dataset[i])
Total attributes present : 8
First Five instances of dataset:
1 : [6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0, 1.0]
2 : [1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0, 0.0]
3 : [8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0, 1.0]
4 : [1.0, 89.0, 66.0, 23.0, 94.0, 28.1, 0.167, 21.0, 0.0]
5 : [0.0, 137.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0, 1.0]
In [14]:
splitRatio = 0.2
trainingSet, testSet = splitDataset(dataset, splitRatio)
print('\nDataset is split into training and testing set.')
print('Training Set Size',len(trainingSet))
print('Testing Set Size',len(testSet))
Type here to search
ENG
IN
10:35
25-11-2020


Acharya Live | Digital Class Roc
MLL/
MLL-5 - Jupyter Notebook
MLL-5-Iris - Jupyter Notebook
ML Lab(2020-21) 7th A2
localhost:8888/notebooks/MLL/MLL-5.ipynb#
jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Dataset is split into training and testing set.
Training Set Size 615
Testing Set Size 153
In [15]:
summaries = summarizeByClass(trainingSet)
separated by 0 and 1 class
{1.0: [[6.0, 148.0, 72.0, 35.0, 0.0, 33.6, 0.627, 50.0, 1.0], [8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0, 1.0], [0.0, 13
7.0, 40.0, 35.0, 168.0, 43.1, 2.288, 33.0, 1.0], [3.0, 78.0, 50.0, 32.0, 88.0, 31.0, 0.248, 26.0, 1.0], [8.0, 125.0, 96.0, 0.
0, 0.0, 0.0, 0.232, 54.0, 1.0], [5.0, 166.0, 72.0, 19.0, 175.0, 25.8, 0.587, 51.0, 1.0], [7.0, 100.0, 0.0, 0.0, 0.0, 30.0, 0.
484, 32.0, 1.0], [0.0, 118.0, 84.0, 47.0, 230.0, 45.8, 0.551, 31.0, 1.0], [7.0, 107.0, 74.0, 0.0, 0.0, 29.6, 0.254, 31.0, 1.
0], [1.0, 115.0, 70.0, 30.0, 96.0, 34.6, 0.529, 32.0, 1.0], [7.0, 196.0, 90.0, 0.0, 0.0, 39.8, 0.451, 41.0, 1.0], [9.0, 119.
0, 80.0, 35.0, 0.0, 29.0, 0.263, 29.0, 1.0], [11.0, 143.0, 94.0, 33.0, 146.0, 36.6, 0.254, 51.0, 1.0], [10.0, 125.0, 70.0, 2
6.0, 115.0, 31.1, 0.205, 41.0, 1.0], [7.0, 147.0, 76.0, 0.0, 0.0, 39.4, 0.257, 43.0, 1.0], [3.0, 158.0, 76.0, 36.0, 245.0, 3
1.6, 0.851, 28.0, 1.0], [2.0, 90.0, 68.0, 42.0, 0.0, 38.2, 0.503, 27.0, 1.0], [4.0, 111.0, 72.0, 47.0, 207.0, 37.1, 1.39, 56.
0, 1.0], [0.0, 180.0, 66.0, 39.0, 0.0, 42.0, 1.893, 25.0, 1.0], [7.0, 103.0, 66.0, 32.0, 0.0, 39.1, 0.344, 31.0, 1.0], [8.0,
176.0, 90.0, 34.0, 300.0, 33.7, 0.467, 58.0, 1.0], [7.0, 187.0, 68.0, 39.0, 304.0, 37.7, 0.254, 41.0, 1.0], [8.0, 133.0, 72.
0, 0.0, 0.0, 32.9, 0.27, 39.0, 1.0], [7.0, 114.0, 66.0, 0.0, 0.0, 32.8, 0.258, 42.0, 1.0], [2.0, 100.0, 66.0, 20.0, 90.0, 32.
9, 0.867, 28.0, 1.0], [13.0, 126.0, 90.0, 0.0, 0.0, 43.4, 0.583, 42.0, 1.0], [0.0, 131.0, 0.0, 0.0, 0.0, 43.2, 0.27, 26.0, 1.
0], [5.0, 137.0, 108.0, 0.0, 0.0, 48.8, 0.227, 37.0, 1.0], [15.0, 136.0, 70.0, 32.0, 110.0, 37.1, 0.153, 43.0, 1.0], [4.0, 13
4.0, 72.0, 0.0, 0.0, 23.8, 0.277, 60.0, 1.0], [1.0, 122.0, 90.0, 51.0, 220.0, 49.7, 0.325, 31.0, 1.0], [1.0, 163.0, 72.0, 0.
0, 0.0, 39.0, 1.222, 33.0, 1.0], [0.0, 95.0, 85.0, 25.0, 36.0, 37.4, 0.247, 24.0, 1.0], [3.0, 171.0, 72.0, 33.0, 135.0, 33.3,
0.199, 24.0, 1.0], [8.0, 155.0, 62.0, 26.0, 495.0, 34.0, 0.543, 46.0, 1.0], [7.0, 160.0, 54.0, 32.0, 175.0, 30.5, 0.588, 39.
0, 1.0], [4.0, 146.0, 92.0, 0.0, 0.0, 31.2, 0.539, 61.0, 1.0], [5.0, 124.0, 74.0, 0.0, 0.0, 34.0, 0.22, 38.0, 1.0], [0.0, 16
In [16]:
predictions = perform_classification(summaries, testSet)
print("predicted values\n")
print(predictions)
Type here to search
ENG
IN
10:35
25-11-2020

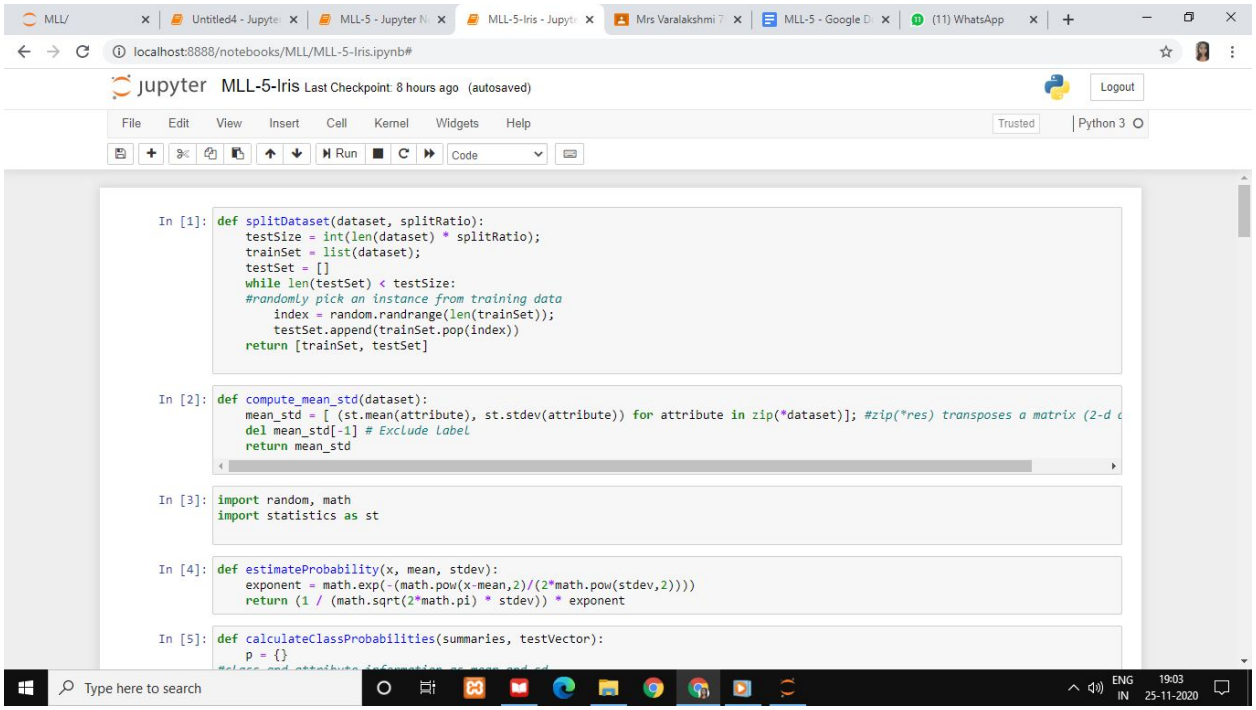

Acharya Live | Digital C
MLL/
MLL-5 - Jupyter Noteb
MLL-5-Iris - Jupyter No
ML Lab(2020-21) 7th A
MLL-5 - Google Docs
localhost:8888/notebooks/MLL/MLL-5.ipynb#
jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
0, 0.0], [10.0, 94.0, 72.0, 18.0, 0.0, 23.1, 0.595, 56.0, 0.0], [1.0, 108.0, 60.0, 46.0, 178.0, 35.5, 0.415, 24.0, 0.0], [1.
0, 114.0, 66.0, 36.0, 200.0, 38.1, 0.289, 21.0, 0.0], [5.0, 117.0, 86.0, 30.0, 105.0, 39.1, 0.251, 42.0, 0.0], [1.0, 111.0, 9
4.0, 0.0, 0.0, 32.8, 0.265, 45.0, 0.0], [4.0, 112.0, 78.0, 40.0, 0.0, 39.4, 0.236, 38.0, 0.0], [1.0, 116.0, 78.0, 29.0, 180.
0, 36.1, 0.496, 25.0, 0.0], [0.0, 141.0, 84.0, 26.0, 0.0, 32.4, 0.433, 22.0, 0.0], [2.0, 92.0, 52.0, 0.0, 0.0, 30.1, 0.141, 2
2.0, 0.0], [2.0, 105.0, 75.0, 0.0, 0.0, 23.3, 0.56, 53.0, 0.0], [4.0, 95.0, 60.0, 32.0, 0.0, 35.4, 0.284, 28.0, 0.0], [0.0, 1
26.0, 86.0, 27.0, 120.0, 27.4, 0.515, 21.0, 0.0], [8.0, 65.0, 72.0, 23.0, 0.0, 32.0, 0.6, 42.0, 0.0], [2.0, 99.0, 60.0, 17.0,
160.0, 36.6, 0.453, 21.0, 0.0], [1.0, 109.0, 58.0, 18.0, 116.0, 28.5, 0.219, 22.0, 0.0], [13.0, 153.0, 88.0, 37.0, 140.0, 40.
6, 1.174, 39.0, 0.0], [12.0, 100.0, 84.0, 33.0, 105.0, 30.0, 0.488, 46.0, 0.0], [1.0, 81.0, 74.0, 41.0, 57.0, 46.3, 1.096, 3
2.0, 0.0], [1.0, 121.0, 78.0, 39.0, 74.0, 39.0, 0.261, 28.0, 0.0], [3.0, 108.0, 62.0, 24.0, 0.0, 26.0, 0.223, 25.0, 0.0], [7.
0, 137.0, 90.0, 41.0, 0.0, 32.0, 0.391, 39.0, 0.0], [1.0, 106.0, 76.0, 0.0, 0.0, 37.5, 0.197, 26.0, 0.0], [10.0, 101.0, 76.0,
48.0, 180.0, 32.9, 0.171, 63.0, 0.0], [2.0, 122.0, 70.0, 27.0, 0.0, 36.8, 0.34, 27.0, 0.0], [1.0, 93.0, 70.0, 31.0, 0.0, 30.
4, 0.315, 23.0, 0.0]]}
Summary of mean and standard deviataion
{1.0: [(5.023584905660377, 3.8298417149377393), (141.72641509433961, 32.18077876131496), (70.41509433962264, 22.6369943931815
9), (21.44811320754717, 18.287647560448562), (95.93396226415095, 131.15619781384464), (35.23443396226415, 7.665624205817336),
(0.5545094339622642, 0.3873311341649635), (37.5, 11.07984806643825)], 0.0: [(3.3424317617866004, 3.092578347915159), (109.635
23573200993, 26.063326040817657), (67.65508684863524, 19.10387011610047), (19.47394540942928, 14.978465174113696), (69.039702
23325062, 95.83305037506487), (30.338461538461537, 7.668471642298804), (0.4284863523573201, 0.2880348180434413), (31.59305210
918114, 11.788060042557028)]}
In [16]:
predictions = perform_classification(summaries, testSet)
print("predicted values\n")
print(predictions)
All Probabilities
{1.0: 2.0294624357397167e-13, 0.0: 2.0706772931893e-12}
All Probabilities
Type here to search
ENG IN
10:43
25-11-2020

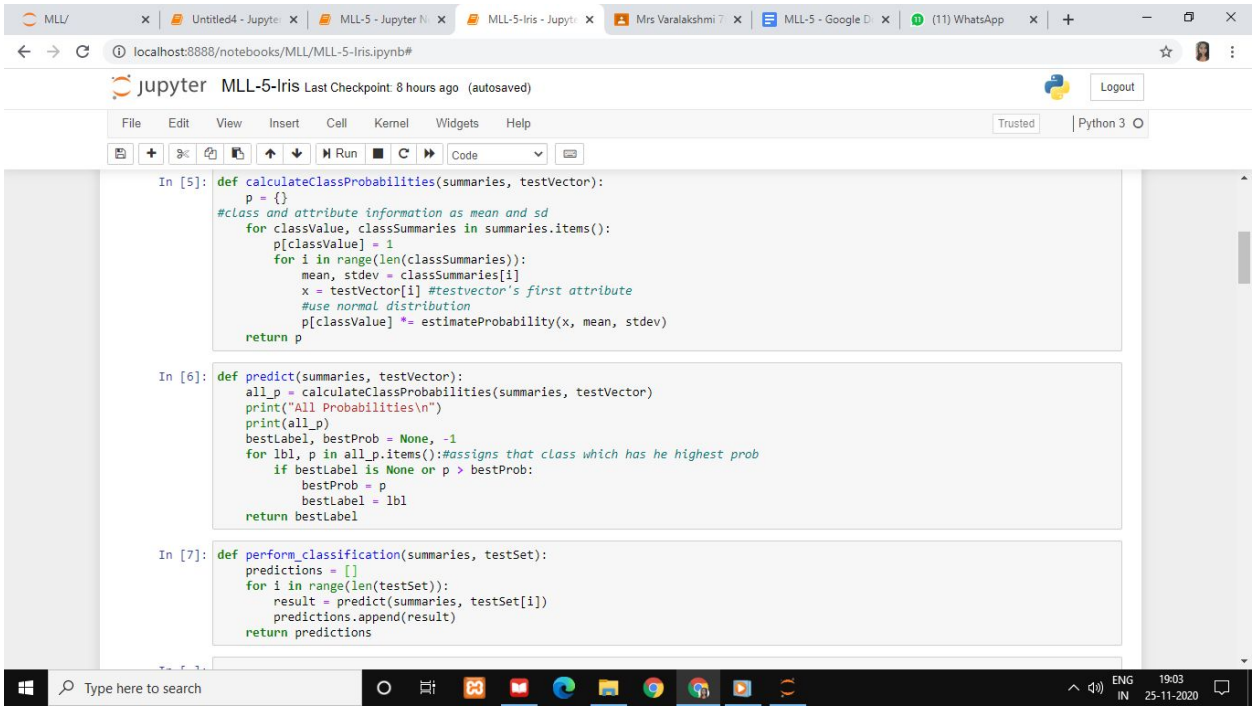

Acharya Live | Digital Class Roc
MLL/
MLL-5 - Jupyter Notebook
MLL-5-Iris - Jupyter Notebook
ML Lab(2020-21) 7th A2
localhost:8888/notebooks/MLL/MLL-5.ipynb#
jupyter MLL-5 Last Checkpoint: 11/18/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [16]:
predictions = perform_classification(summaries, testSet)
print("predicted values\n")
print(predictions)
All Probabilities
{1.0: 2.3342605912882394e-13, 0.0: 1.1838772926831941e-12}
All Probabilities
{1.0: 4.4709056380903254e-13, 0.0: 9.927056481709797e-13}
All Probabilities
{1.0: 1.0348342514823991e-12, 0.0: 9.889850044353876e-13}
All Probabilities
{1.0: 1.5875566066843367e-15, 0.0: 1.199376164018765e-16}
All Probabilities
{1.0: 2.4750927017444546e-13, 0.0: 2.2108161357620938e-12}
All Probabilities
{1.0: 4.727786998254915e-15, 0.0: 4.0016147307919965e-15}
All Probabilities
In [17]:
accuracy = getAccuracy(testSet, predictions)
print('\nAccuracy of the Naive Baysian Classifier is :', accuracy)
Accuracy of the Naive Baysian Classifier is : 76.47058823529412
Type here to search
ENG IN
10:35
25-11-2020

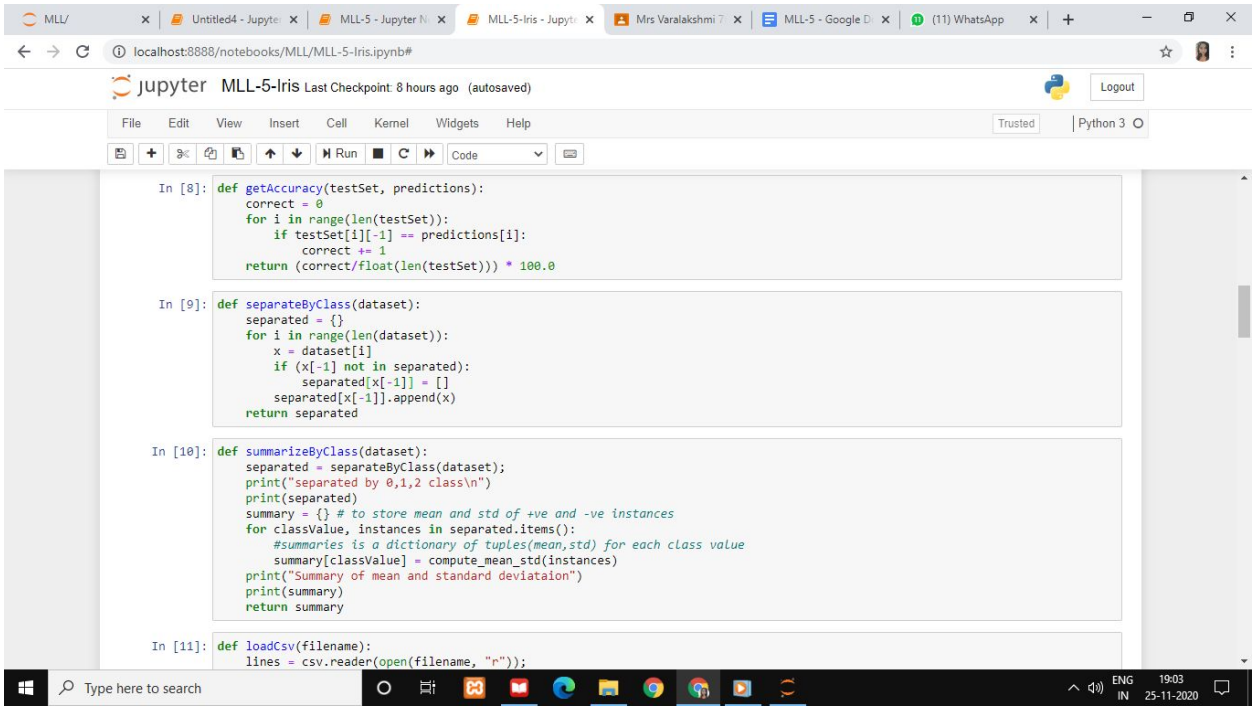




OUTPUT For Iris Dataset


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]:
def splitDataset(dataset, splitRatio):
testSize = int(len(dataset) * splitRatio);
trainSet = list(dataset);
testSet = []
while len(testSet) < testSize:
#randomly pick an instance from training data
index = random.randrange(len(trainSet));
testSet.append(trainSet.pop(index))
return [trainSet, testSet]
In [2]:
def compute_mean_std(dataset):
mean_std = [(st.mean(attribute), st.stdev(attribute)) for attribute in zip(*dataset)]; #zip(*res) transposes a matrix (2-d
del mean_std[-1] # Exclude label
return mean_std
In [3]:
import random, math
import statistics as st
In [4]:
def estimateProbability(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 [5]:
def calculateClassProbabilities(summaries, testVector):
p = {}
Type here to search
ENG
IN
19:03
25-11-2020


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [5]:
def calculateClassProbabilities(summaries, testVector):
p = {}
#class and attribute information as mean and sd
for classValue, classSummaries in summaries.items():
p[classValue] = 1
for i in range(len(classSummaries)):
mean, stdev = classSummaries[i]
x = testVector[i] #testvector's first attribute
#use normal distribution
p[classValue] *= estimateProbability(x, mean, stdev)
return p
In [6]:
def predict(summaries, testVector):
all_p = calculateClassProbabilities(summaries, testVector)
print("All Probabilities\n")
print(all_p)
bestLabel, bestProb = None, -1
for lbl, p in all_p.items():#assigns that class which has he highest prob
if bestLabel is None or p > bestProb:
bestProb = p
bestLabel = lbl
return bestLabel
In [7]:
def perform_classification(summaries, testSet):
predictions = []
for i in range(len(testSet)):
result = predict(summaries, testSet[i])
predictions.append(result)
return predictions
Type here to search
ENG
IN
19:03
25-11-2020



jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [8]:
```
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
```

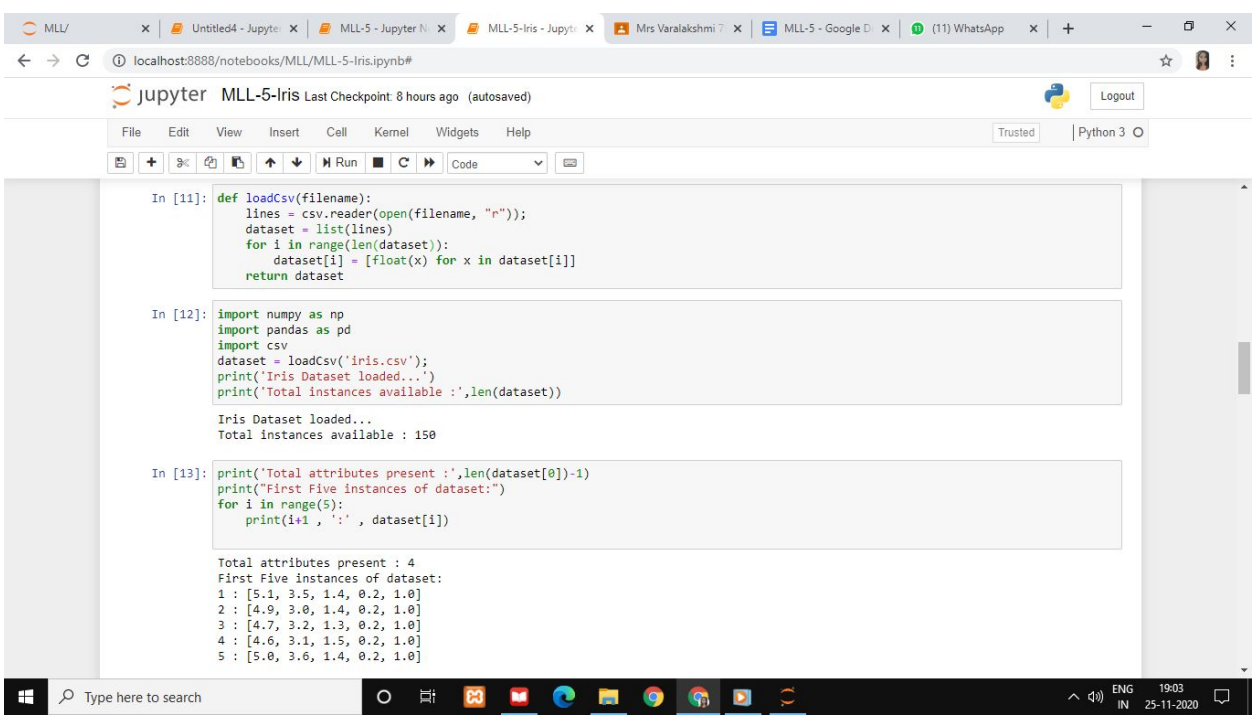
In [9]:
```
def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        x = dataset[i]
        if (x[-1] not in separated):
            separated[x[-1]] = []
        separated[x[-1]].append(x)
    return separated
```

In [10]:
```
def summarizeByClass(dataset):
    separated = separateByClass(dataset);
    print("separated by 0,1,2 class\n")
    print(separated)
    summary = {} # to store mean and std of +ve and -ve instances
    for classValue, instances in separated.items():
        #summaries is a dictionary of tuples(mean,std) for each class value
        summary[classValue] = compute_mean_std(instances)
    print("Summary of mean and standard deviataion")
    print(summary)
    return summary
```

In [11]:
```
def loadCsv(filename):
    lines = csv.reader(open(filename, "r"));
```

Type here to search

ENG IN 19:03 25-11-2020



jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [11]:
```
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 [12]:
```
import numpy as np
import pandas as pd
import csv
dataset = loadCsv('iris.csv');
print('Iris Dataset loaded...')
print('Total instances available :',len(dataset))
```

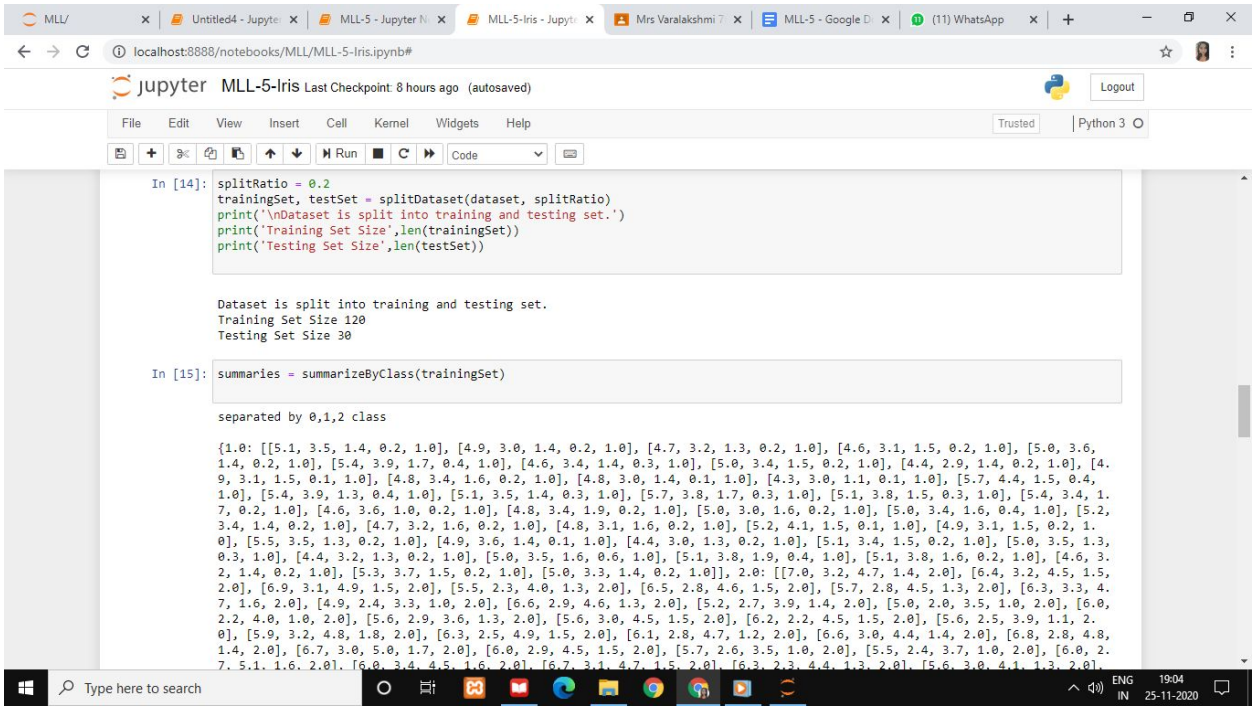
```
Iris Dataset loaded...
Total instances available : 150
```

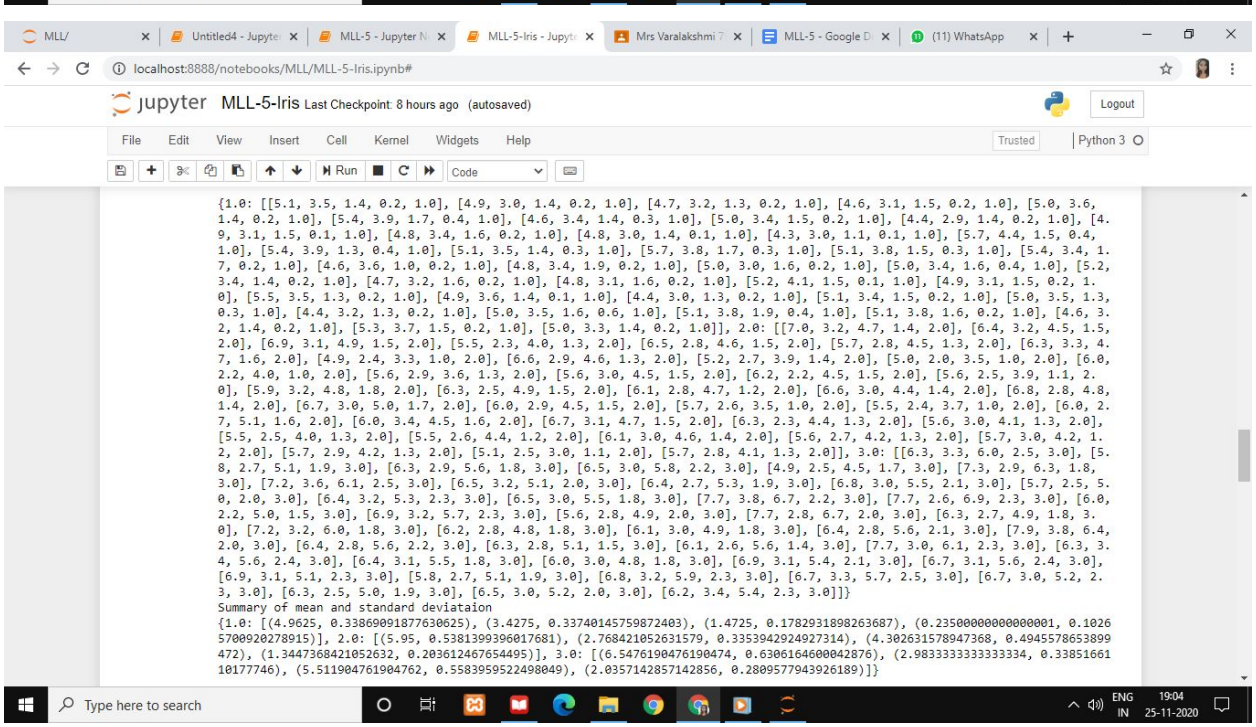
In [13]:
```
print('Total attributes present :',len(dataset[0])-1)
print("First Five instances of dataset:")
for i in range(5):
    print(i+1 , ':' , dataset[i])
```

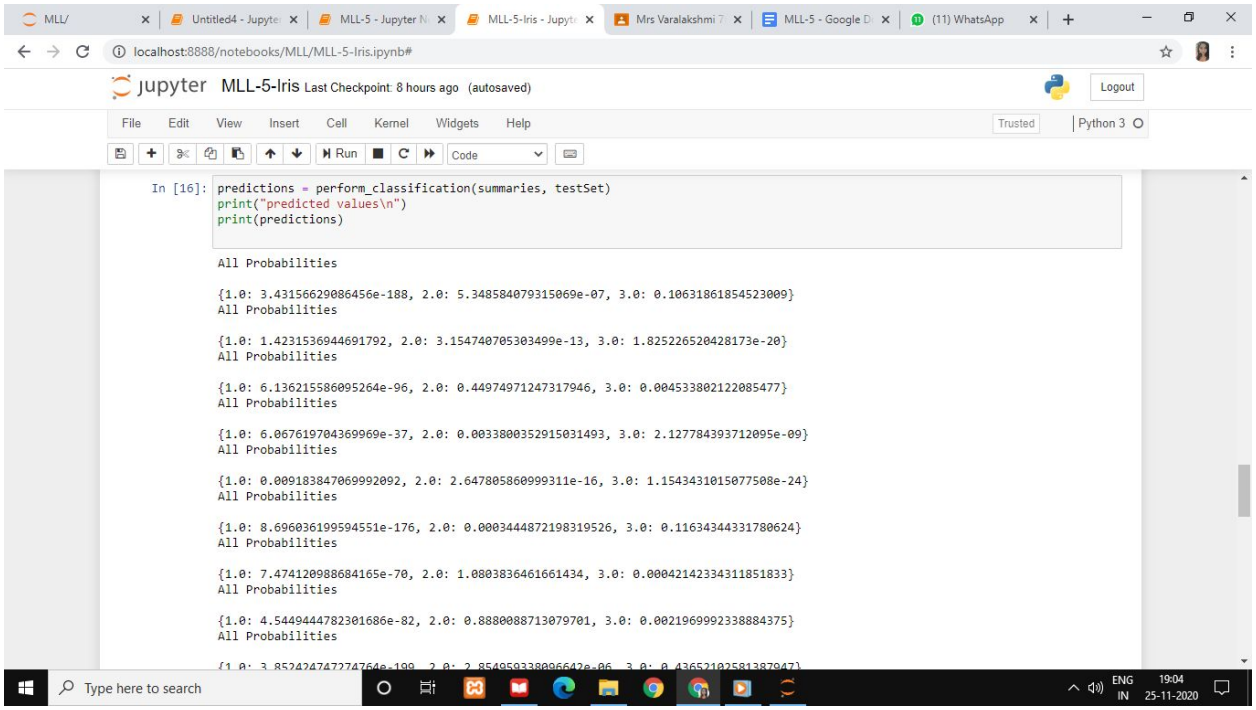
```
Total attributes present : 4
First Five instances of dataset:
1 : [5.1, 3.5, 1.4, 0.2, 1.0]
2 : [4.9, 3.0, 1.4, 0.2, 1.0]
3 : [4.7, 3.2, 1.3, 0.2, 1.0]
4 : [4.6, 3.1, 1.5, 0.2, 1.0]
5 : [5.0, 3.6, 1.4, 0.2, 1.0]
```

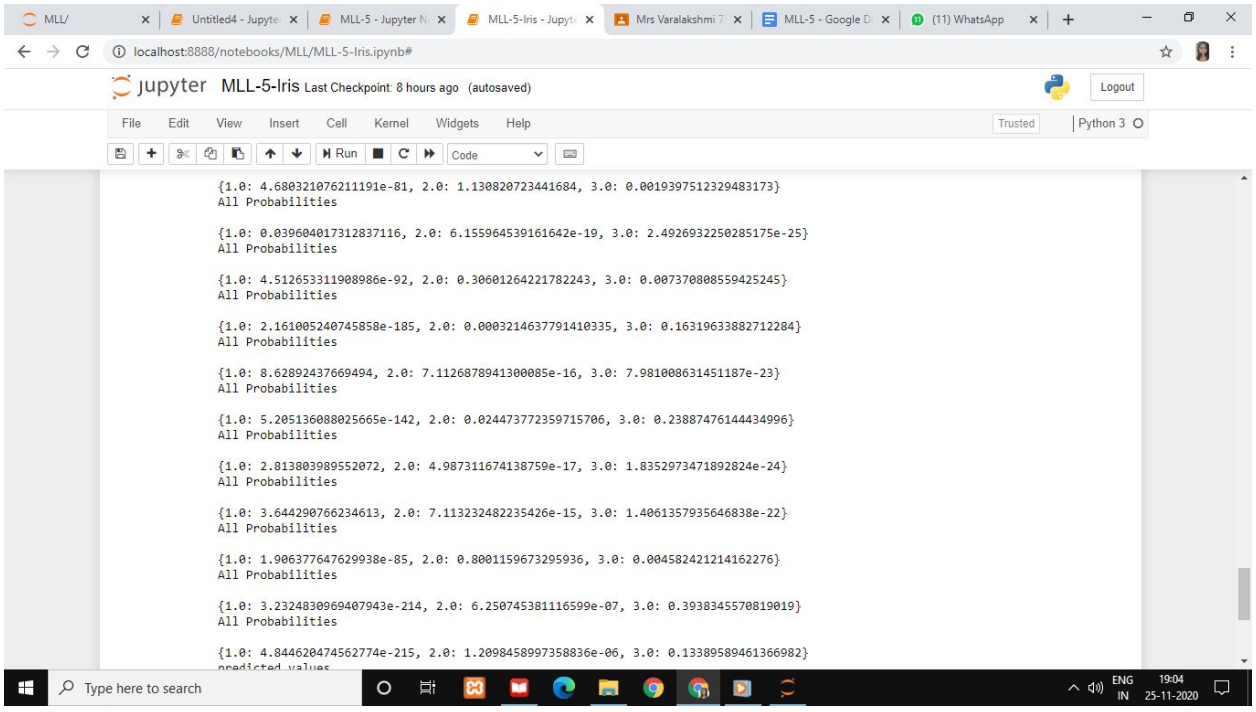
Type here to search

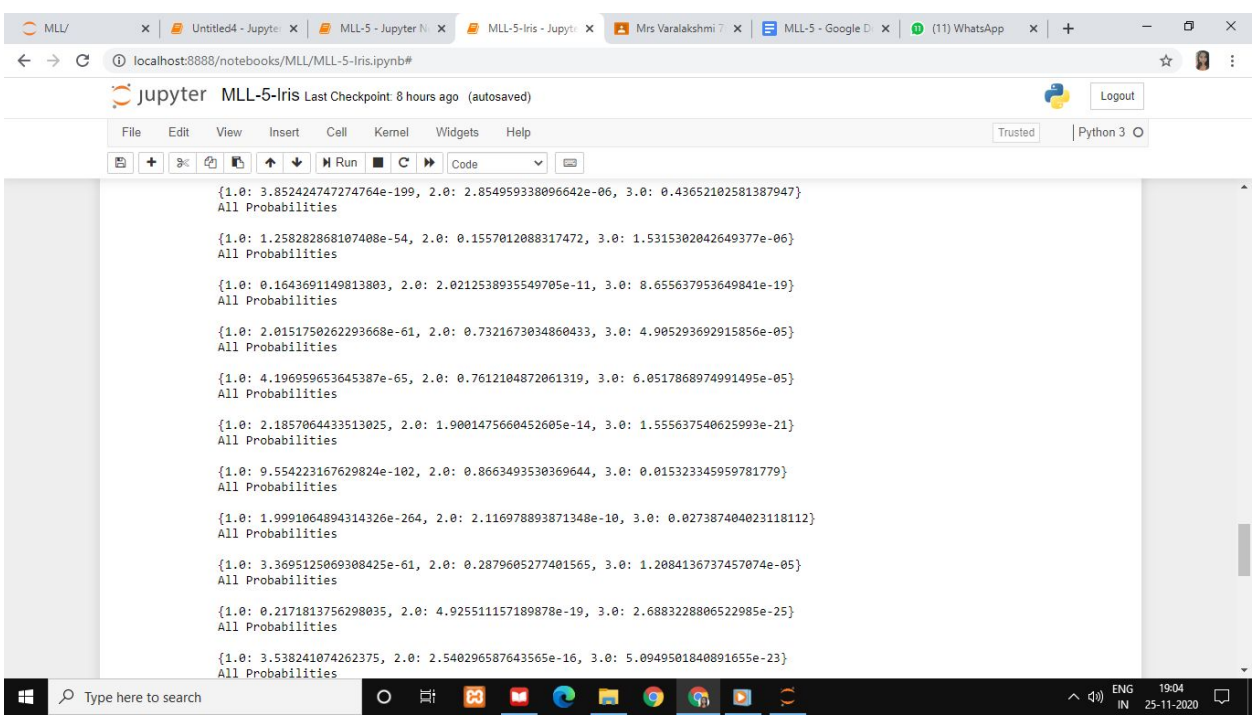
ENG IN 19:03 25-11-2020

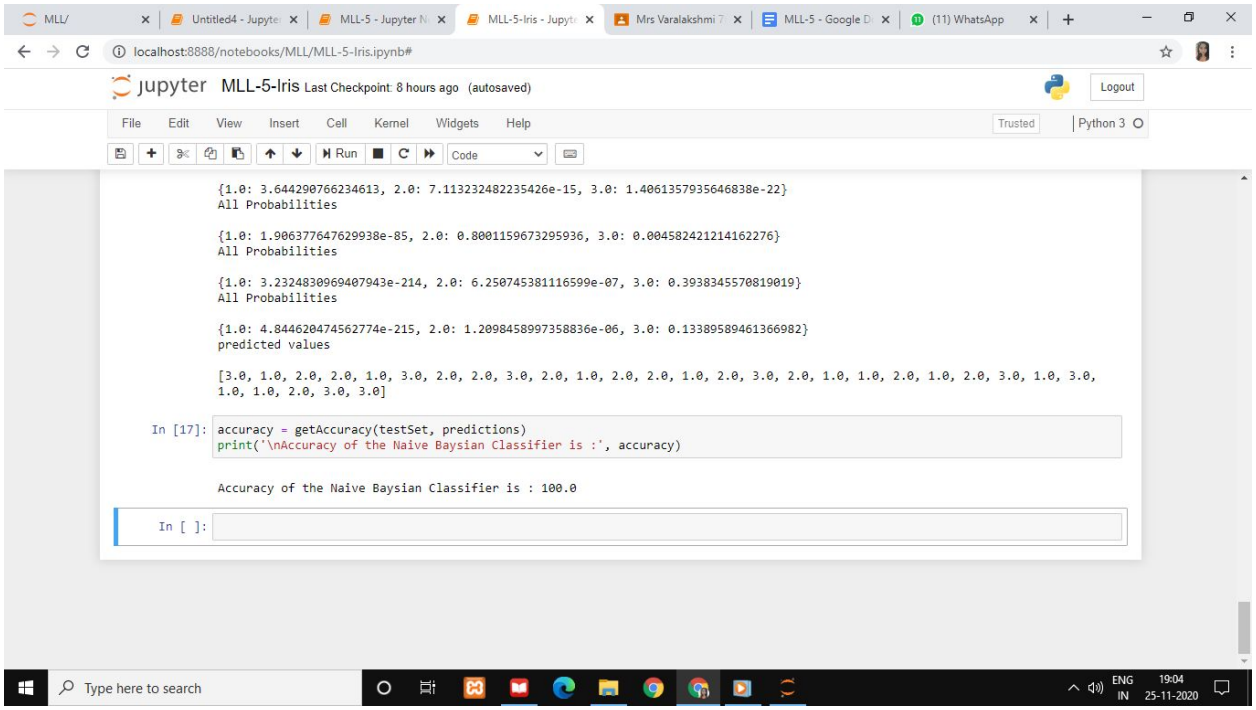

MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [14]:
splitRatio = 0.2
trainingSet, testSet = splitDataset(dataset, splitRatio)
print('\nDataset is split into training and testing set.')
print('Training Set Size',len(trainingSet))
print('Testing Set Size',len(testSet))
Dataset is split into training and testing set.
Training Set Size 120
Testing Set Size 30
In [15]:
summaries = summarizeByClass(trainingSet)
separated by 0,1,2 class
{1.0: [[5.1, 3.5, 1.4, 0.2, 1.0], [4.9, 3.0, 1.4, 0.2, 1.0], [4.7, 3.2, 1.3, 0.2, 1.0], [4.6, 3.1, 1.5, 0.2, 1.0], [5.0, 3.6,
1.4, 0.2, 1.0], [5.4, 3.9, 1.7, 0.4, 1.0], [4.6, 3.4, 1.4, 0.3, 1.0], [5.0, 3.4, 1.5, 0.2, 1.0], [4.4, 2.9, 1.4, 0.2, 1.0], [4.
9, 3.1, 1.5, 0.1, 1.0], [4.8, 3.4, 1.6, 0.2, 1.0], [4.8, 3.0, 1.4, 0.1, 1.0], [4.3, 3.0, 1.1, 0.1, 1.0], [5.7, 4.4, 1.5, 0.4,
1.0], [5.4, 3.9, 1.3, 0.4, 1.0], [5.1, 3.5, 1.4, 0.3, 1.0], [5.7, 3.8, 1.7, 0.3, 1.0], [5.1, 3.8, 1.5, 0.3, 1.0], [5.4, 3.4, 1.
7, 0.2, 1.0], [4.6, 3.6, 1.0, 0.2, 1.0], [4.8, 3.4, 1.9, 0.2, 1.0], [5.0, 3.0, 1.6, 0.2, 1.0], [5.0, 3.4, 1.6, 0.4, 1.0], [5.2,
3.4, 1.4, 0.2, 1.0], [4.7, 3.2, 1.6, 0.2, 1.0], [4.8, 3.1, 1.6, 0.2, 1.0], [5.2, 4.1, 1.5, 0.1, 1.0], [4.9, 3.1, 1.5, 0.2, 1.
0], [5.5, 3.5, 1.3, 0.2, 1.0], [4.9, 3.6, 1.4, 0.1, 1.0], [4.4, 3.0, 1.3, 0.2, 1.0], [5.1, 3.4, 1.5, 0.2, 1.0], [5.0, 3.5, 1.3,
0.3, 1.0], [4.4, 3.2, 1.3, 0.2, 1.0], [5.0, 3.5, 1.6, 0.6, 1.0], [5.1, 3.8, 1.9, 0.4, 1.0], [5.1, 3.8, 1.6, 0.2, 1.0], [4.6, 3.
2, 1.4, 0.2, 1.0], [5.3, 3.7, 1.5, 0.2, 1.0], [5.0, 3.3, 1.4, 0.2, 1.0]], 2.0: [[7.0, 3.2, 4.7, 1.4, 2.0], [6.4, 3.2, 4.5, 1.5,
2.0], [6.9, 3.1, 4.9, 1.5, 2.0], [5.5, 2.3, 4.0, 1.3, 2.0], [6.5, 2.8, 4.6, 1.5, 2.0], [5.7, 2.8, 4.5, 1.3, 2.0], [6.3, 3.3, 4.
7, 1.6, 2.0], [4.9, 2.4, 3.3, 1.0, 2.0], [6.6, 2.9, 4.6, 1.3, 2.0], [5.2, 2.7, 3.9, 1.4, 2.0], [5.0, 2.0, 3.5, 1.0, 2.0], [6.0,
2.2, 4.0, 1.0, 2.0], [5.6, 2.9, 3.6, 1.3, 2.0], [5.6, 3.0, 4.5, 1.5, 2.0], [6.2, 2.2, 4.5, 1.5, 2.0], [5.6, 2.5, 3.9, 1.1, 2.
0], [5.9, 3.2, 4.8, 1.8, 2.0], [6.3, 2.5, 4.9, 1.5, 2.0], [6.1, 2.8, 4.7, 1.2, 2.0], [6.6, 3.0, 4.4, 1.4, 2.0], [6.8, 2.8, 4.8,
1.4, 2.0], [6.7, 3.0, 5.0, 1.7, 2.0], [6.0, 2.9, 4.5, 1.5, 2.0], [5.7, 2.6, 3.5, 1.0, 2.0], [5.5, 2.4, 3.7, 1.0, 2.0], [6.0, 2.
7, 5.1, 1.6, 2.0], [6.0, 3.4, 4.5, 1.6, 2.0], [6.7, 3.1, 4.7, 1.5, 2.0], [6.3, 2.3, 4.4, 1.3, 2.0], [5.6, 3.0, 4.1, 1.3, 2.0],
Type here to search
ENG IN
19:04
25-11-2020


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
{1.0: [[5.1, 3.5, 1.4, 0.2, 1.0], [4.9, 3.0, 1.4, 0.2, 1.0], [4.7, 3.2, 1.3, 0.2, 1.0], [4.6, 3.1, 1.5, 0.2, 1.0], [5.0, 3.6,
1.4, 0.2, 1.0], [5.4, 3.9, 1.7, 0.4, 1.0], [4.6, 3.4, 1.4, 0.3, 1.0], [5.0, 3.4, 1.5, 0.2, 1.0], [4.4, 2.9, 1.4, 0.2, 1.0], [4.
9, 3.1, 1.5, 0.1, 1.0], [4.8, 3.4, 1.6, 0.2, 1.0], [4.8, 3.0, 1.4, 0.1, 1.0], [4.3, 3.0, 1.1, 0.1, 1.0], [5.7, 4.4, 1.5, 0.4,
1.0], [5.4, 3.9, 1.3, 0.4, 1.0], [5.1, 3.5, 1.4, 0.3, 1.0], [5.7, 3.8, 1.7, 0.3, 1.0], [5.1, 3.8, 1.5, 0.3, 1.0], [5.4, 3.4, 1.
7, 0.2, 1.0], [4.6, 3.6, 1.0, 0.2, 1.0], [4.8, 3.4, 1.9, 0.2, 1.0], [5.0, 3.0, 1.6, 0.2, 1.0], [5.0, 3.4, 1.6, 0.4, 1.0], [5.2,
3.4, 1.4, 0.2, 1.0], [4.7, 3.2, 1.6, 0.2, 1.0], [4.8, 3.1, 1.6, 0.2, 1.0], [5.2, 4.1, 1.5, 0.1, 1.0], [4.9, 3.1, 1.5, 0.2, 1.
0], [5.5, 3.5, 1.3, 0.2, 1.0], [4.9, 3.6, 1.4, 0.1, 1.0], [4.4, 3.0, 1.3, 0.2, 1.0], [5.1, 3.4, 1.5, 0.2, 1.0], [5.0, 3.5, 1.3,
0.3, 1.0], [4.4, 3.2, 1.3, 0.2, 1.0], [5.0, 3.5, 1.6, 0.6, 1.0], [5.1, 3.8, 1.9, 0.4, 1.0], [5.1, 3.8, 1.6, 0.2, 1.0], [4.6, 3.
2, 1.4, 0.2, 1.0], [5.3, 3.7, 1.5, 0.2, 1.0], [5.0, 3.3, 1.4, 0.2, 1.0]], 2.0: [[7.0, 3.2, 4.7, 1.4, 2.0], [6.4, 3.2, 4.5, 1.5,
2.0], [6.9, 3.1, 4.9, 1.5, 2.0], [5.5, 2.3, 4.0, 1.3, 2.0], [6.5, 2.8, 4.6, 1.5, 2.0], [5.7, 2.8, 4.5, 1.3, 2.0], [6.3, 3.3, 4.
7, 1.6, 2.0], [4.9, 2.4, 3.3, 1.0, 2.0], [6.6, 2.9, 4.6, 1.3, 2.0], [5.2, 2.7, 3.9, 1.4, 2.0], [5.0, 2.0, 3.5, 1.0, 2.0], [6.0,
2.2, 4.0, 1.0, 2.0], [5.6, 2.9, 3.6, 1.3, 2.0], [5.6, 3.0, 4.5, 1.5, 2.0], [6.2, 2.2, 4.5, 1.5, 2.0], [5.6, 2.5, 3.9, 1.1, 2.
0], [5.9, 3.2, 4.8, 1.8, 2.0], [6.3, 2.5, 4.9, 1.5, 2.0], [6.1, 2.8, 4.7, 1.2, 2.0], [6.6, 3.0, 4.4, 1.4, 2.0], [6.8, 2.8, 4.8,
1.4, 2.0], [6.7, 3.0, 5.0, 1.7, 2.0], [6.0, 2.9, 4.5, 1.5, 2.0], [5.7, 2.6, 3.5, 1.0, 2.0], [5.5, 2.4, 3.7, 1.0, 2.0], [6.0, 2.
7, 5.1, 1.6, 2.0], [6.0, 3.4, 4.5, 1.6, 2.0], [6.7, 3.1, 4.7, 1.5, 2.0], [6.3, 2.3, 4.4, 1.3, 2.0], [5.6, 3.0, 4.1, 1.3, 2.0],
[5.5, 2.5, 4.0, 1.3, 2.0], [5.5, 2.6, 4.4, 1.2, 2.0], [6.1, 3.0, 4.6, 1.4, 2.0], [5.6, 2.7, 4.2, 1.3, 2.0], [5.7, 3.0, 4.2, 1.
2, 2.0], [5.7, 2.9, 4.2, 1.3, 2.0], [5.1, 2.5, 3.0, 1.1, 2.0], [5.7, 2.8, 4.1, 1.3, 2.0]], 3.0: [[6.3, 3.3, 6.0, 2.5, 3.0], [5.
8, 2.7, 5.1, 1.9, 3.0], [6.3, 2.9, 5.6, 1.8, 3.0], [6.5, 3.0, 5.8, 2.2, 3.0], [4.9, 2.5, 4.5, 1.7, 3.0], [7.3, 2.9, 6.3, 1.8,
3.0], [7.2, 3.6, 6.1, 2.5, 3.0], [6.5, 3.2, 5.1, 2.0, 3.0], [6.4, 2.7, 5.3, 1.9, 3.0], [6.8, 3.0, 5.5, 2.1, 3.0], [5.7, 2.5, 5.
0, 2.0, 3.0], [6.4, 3.2, 5.3, 2.3, 3.0], [6.5, 3.0, 5.5, 1.8, 3.0], [7.7, 3.8, 6.7, 2.2, 3.0], [7.7, 2.6, 6.9, 2.3, 3.0], [6.0,
2.2, 5.0, 1.5, 3.0], [6.9, 3.2, 5.7, 2.3, 3.0], [5.6, 2.8, 4.9, 2.0, 3.0], [7.7, 2.8, 6.7, 2.0, 3.0], [6.3, 2.7, 4.9, 1.8, 3.
0], [7.2, 3.2, 6.0, 1.8, 3.0], [6.2, 2.8, 4.8, 1.8, 3.0], [6.1, 3.0, 4.9, 1.8, 3.0], [6.4, 2.8, 5.6, 2.1, 3.0], [7.9, 3.8, 6.4,
2.0, 3.0], [6.4, 2.8, 5.6, 2.2, 3.0], [6.3, 2.8, 5.1, 1.5, 3.0], [6.1, 2.6, 5.6, 1.4, 3.0], [7.7, 3.0, 6.1, 2.3, 3.0], [6.3, 3.
4, 5.6, 2.4, 3.0], [6.4, 3.1, 5.5, 1.8, 3.0], [6.0, 3.0, 4.8, 1.8, 3.0], [6.9, 3.1, 5.4, 2.1, 3.0], [6.7, 3.1, 5.6, 2.4, 3.0],
[6.9, 3.1, 5.1, 2.3, 3.0], [5.8, 2.7, 5.1, 1.9, 3.0], [6.8, 3.2, 5.9, 2.3, 3.0], [6.7, 3.3, 5.7, 2.5, 3.0], [6.7, 3.0, 5.2, 2.
3, 3.0], [6.3, 2.5, 5.0, 1.9, 3.0], [6.5, 3.0, 5.2, 2.0, 3.0], [6.2, 3.4, 5.4, 2.3, 3.0]]}
Summary of mean and standard deviataion
{1.0: [(4.9625, 0.33869091877630625), (3.4275, 0.33740145759872403), (1.4725, 0.1782931898263687), (0.23500000000000001, 0.1026
5700920278915)], 2.0: [(5.95, 0.5381399396017681), (2.768421052631579, 0.3353942924927314), (4.302631578947368, 0.4945578653899
472), (1.3447368421052632, 0.20361246765495)], 3.0: [(6.547619047619047, 0.6306164600042876), (2.9833333333333334, 0.33851661
10177746), (5.511904761904762, 0.5583959522498049), (2.0357142857142856, 0.2809577943926189)]}
Type here to search
ENG IN
19:04
25-11-2020


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [16]:
predictions = perform_classification(summaries, testSet)
print("predicted values\n")
print(predictions)
All Probabilities
{1.0: 3.43156629086456e-188, 2.0: 5.348584079315069e-07, 3.0: 0.10631861854523009}
All Probabilities
{1.0: 1.4231536944691792, 2.0: 3.154740705303499e-13, 3.0: 1.825226520428173e-20}
All Probabilities
{1.0: 6.136215586095264e-96, 2.0: 0.44974971247317946, 3.0: 0.00453380212208547 7}
All Probabilities
{1.0: 6.067619704369969e-37, 2.0: 0.0033800352915031493, 3.0: 2.127784393712095e-09}
All Probabilities
{1.0: 0.009183847069992092, 2.0: 2.647805860999311e-16, 3.0: 1.1543431015077508e-24}
All Probabilities
{1.0: 8.696036199594551e-176, 2.0: 0.0003444872198319526, 3.0: 0.11634344331780624}
All Probabilities
{1.0: 7.474120988684165e-70, 2.0: 1.0803836461661434, 3.0: 0.00042142334311851833}
All Probabilities
{1.0: 4.5449444782301686e-82, 2.0: 0.8880088713079701, 3.0: 0.0021969992338884375}
All Probabilities
Type here to search
ENG
IN
19:04
25-11-2020


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
{1.0: 4.680321076211191e-81, 2.0: 1.130820723441684, 3.0: 0.001939751232948317 3}
All Probabilities
{1.0: 0.039604017312837116, 2.0: 6.155964539161642e-19, 3.0: 2.4926932250285175e-25}
All Probabilities
{1.0: 4.512653311908986e-92, 2.0: 0.30601264221782243, 3.0: 0.007370808559425245}
All Probabilities
{1.0: 2.161005240745858e-185, 2.0: 0.0003214637791410335, 3.0: 0.16319633882712284}
All Probabilities
{1.0: 8.62892437669494, 2.0: 7.1126878941300085e-16, 3.0: 7.981008631451187e-23}
All Probabilities
{1.0: 5.205136088025665e-142, 2.0: 0.024473772359715706, 3.0: 0.23887476144434996}
All Probabilities
{1.0: 2.813803989552072, 2.0: 4.987311674138759e-17, 3.0: 1.8352973471892824e-24}
All Probabilities
{1.0: 3.644290766234613, 2.0: 7.113232482235426e-15, 3.0: 1.4061357935646838e-22}
All Probabilities
{1.0: 1.9063776476299 38e-85, 2.0: 0.8001159673295936, 3.0: 0.004582421214162276}
All Probabilities
{1.0: 3.2324830969407943e-214, 2.0: 6.250745381116599e-07, 3.0: 0.3938345570819019}
All Probabilities
{1.0: 4.844620474562774e-215, 2.0: 1.2098458997358836e-06, 3.0: 0.13389589461366982}
predicted values
Type here to search
ENG IN
19:04 25-11-2020


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
{1.0: 3.852424747274764e-199, 2.0: 2.854959338096642e-06, 3.0: 0.43652102581387947}
All Probabilities
{1.0: 1.258282868107408e-54, 2.0: 0.1557012088317472, 3.0: 1.5315302042649377e-06}
All Probabilities
{1.0: 0.1643691149813803, 2.0: 2.0212538935549705e-11, 3.0: 8.655637953649841e-19}
All Probabilities
{1.0: 2.0151750262293668e-61, 2.0: 0.7321673034860433, 3.0: 4.905293692915856e-05}
All Probabilities
{1.0: 4.1969596536453 87e-65, 2.0: 0.7612104872061319, 3.0: 6.0517868974991495e-05}
All Probabilities
{1.0: 2.1857064433513025, 2.0: 1.9001475660452605e-14, 3.0: 1.555637540625993e-21}
All Probabilities
{1.0: 9.554223167629824e-102, 2.0: 0.8663493530369644, 3.0: 0.015323345959781779}
All Probabilities
{1.0: 1.9991064894314326e-264, 2.0: 2.116978893871348e-10, 3.0: 0.02738740402311 8112}
All Probabilities
{1.0: 3.3695125069308425e-61, 2.0: 0.2879605277401565, 3.0: 1.2084136737457074e-05}
All Probabilities
{1.0: 0.2171813756298035, 2.0: 4.925511157189878e-19, 3.0: 2.6883228806522985e-25}
All Probabilities
{1.0: 3.538241074262375, 2.0: 2.5402965876435 65e-16, 3.0: 5.0949501840891655e-23}
All Probabilities
Type here to search
ENG IN
19:04 25-11-2020


MLL/
Untitled4 - Jupyter
MLL-5 - Jupyter N
MLL-5-Iris - Jupyt
Mrs Varalakshmi 7
MLL-5 - Google D
(11) WhatsApp
localhost:8888/notebooks/MLL/MLL-5-Iris.ipynb#
jupyter MLL-5-Iris Last Checkpoint: 8 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
{1.0: 3.644290766234613, 2.0: 7.113232482235426e-15, 3.0: 1.4061357935646838e-22}
All Probabilities
{1.0: 1.9063776476299 38e-85, 2.0: 0.8001159673295936, 3.0: 0.004582421214162276}
All Probabilities
{1.0: 3.2324830969407943e-214, 2.0: 6.250745381116599e-07, 3.0: 0.3938345570819019}
All Probabilities
{1.0: 4.844620474562774e-215, 2.0: 1.2098458997358836e-06, 3.0: 0.13389589461366982}
predicted values
[3.0, 1.0, 2.0, 2.0, 1.0, 3.0, 2.0, 2.0, 3.0, 2.0, 1.0, 2.0, 2.0, 1.0, 2.0, 3.0, 2.0, 1.0, 1.0, 2.0, 1.0, 2.0, 3.0, 1.0, 3.0,
1.0, 1.0, 2.0, 3.0, 3.0]
In [17]:
accuracy = getAccuracy(testSet, predictions)
print('\nAccuracy of the Naive Baysian Classifier is :', accuracy)
Accuracy of the Naive Baysian Classifier is : 100.0
In [ ]:
Type here to search
ENG
IN
19:04
25-11-2020