PRML LAB 4

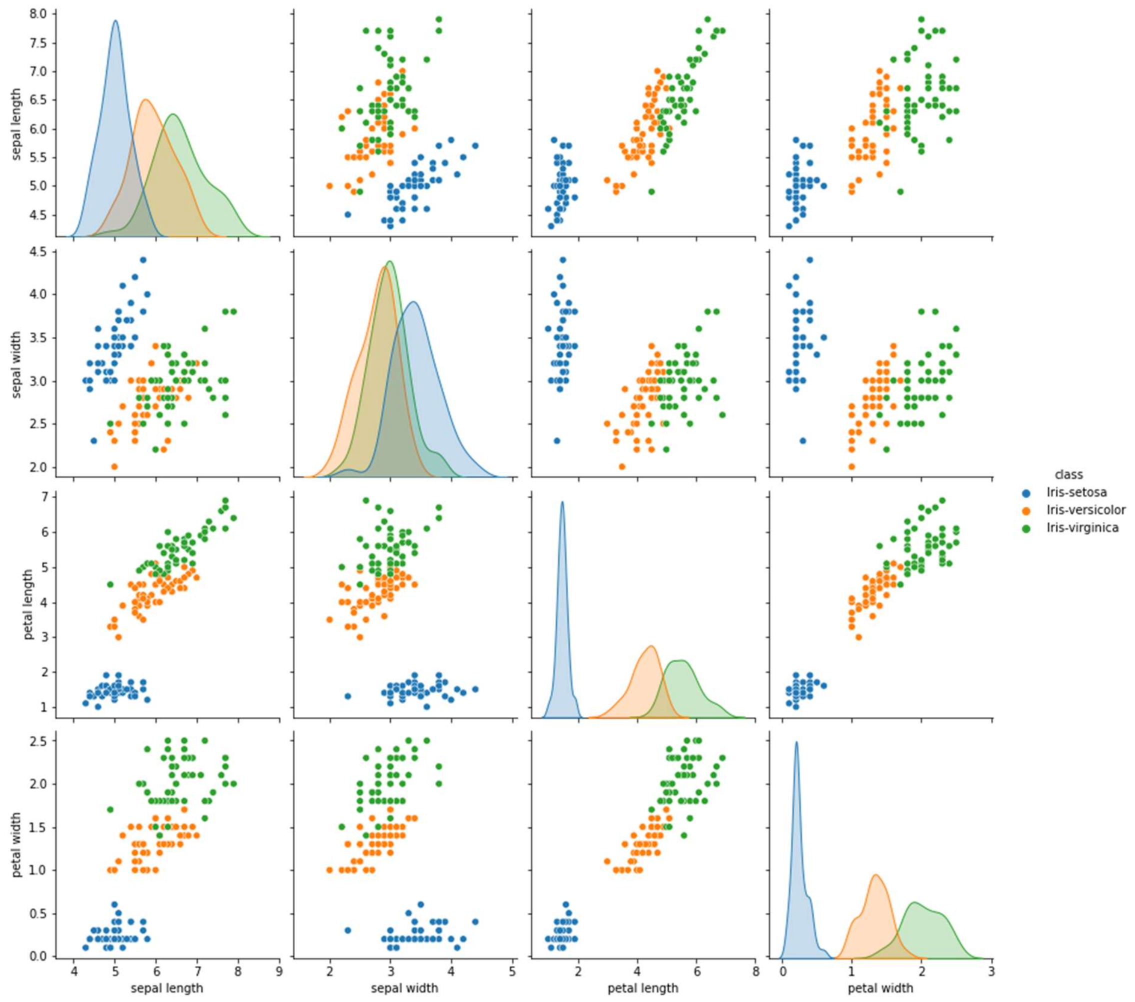
Name: - Shalin Jain

Roll No.: - B21CS070

Problem 1

Part 1

Pre-processing the data and visualizing it with help of sns.pairplot



Now for the Gaussian Classifier I have made a constructor which uses the matrix variable to take the type of case for the covariance matrix.

Part 2

Train Function

Train function uses the X_train , Y_train for training the model. In the function it mainly calculate variance, means of different feature and classes. It also calculate the covariance matrix according to the input in the constructor.

Test Function

It takes X_test, Y_test as input and test on the input. It uses another function known as Predict function which uses a feature vector to calculate the probability and uses all the probability and returns the class for which it gets the maximum probability. This thing is done in a for loop in the test function for the whole X_test. The it checks with the Y_test to get accuracy of the model.

It returns a tuple giving a list of predicted class and a dictionary with 'Score' as key which gives the accuracy.

```
Case 1
{'Score': 0.9333333333333333}
```

Predict Function

It takes a feature vector as an input and predict the class of the feature vector.

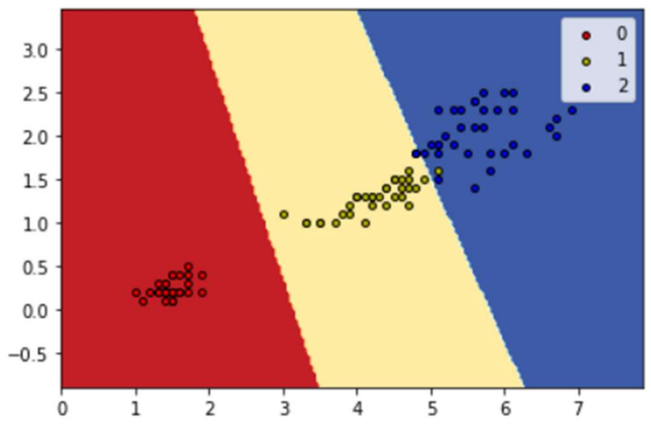
Plot Decision Boundary Function

It plots the decision boundary of the dataset given as input and can also take colours as input for different colours.

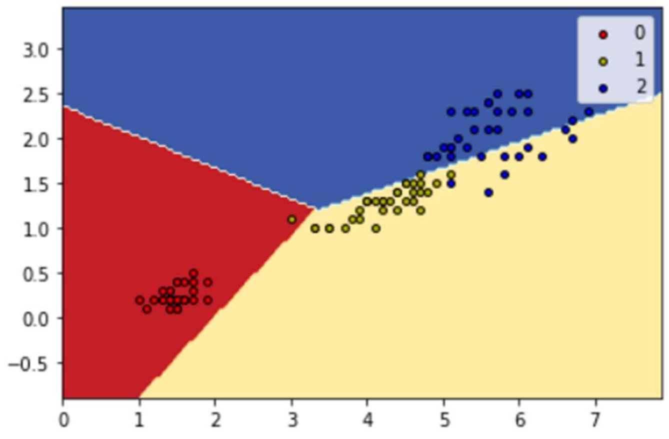
Part 3

Plots of the different cases are given as

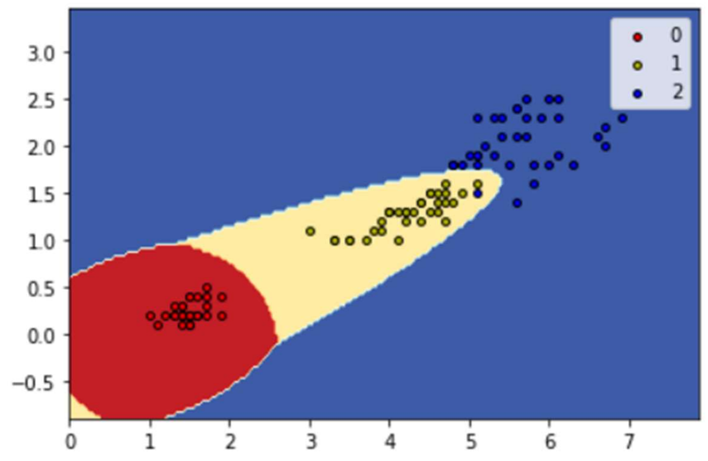
```
Case 1
{'Score': 0.9333333333333333}
```



```
Case 2
{'Score': 0.8}
```



```
Case 3
{'Score': 0.9555555555555556}
```



The decision boundary is precise and is linear in the first two cases and non-linear in third case.

And if we plot means them the line joining the means will only be perpendicular in the first case.

Part 4

For the cross validation I have made a function which divides the dataset into k folds, k-1 folds go for the training and 1 fold is remains for the testing. For k times score is found on the testing fold.

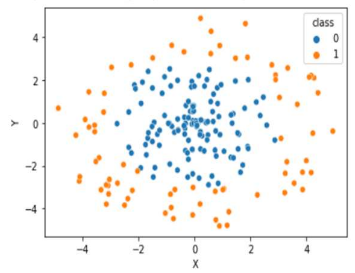
```
Case 1
[0.9, 0.9, 0.9, 0.9666666666666667, 0.9333333333333333]
0.9333333333333332
Case 2
[0.8666666666666667, 0.8666666666666667, 0.9, 0.8, 0.8333333333333334]
0.8266666666666665
Case 3
[1.0, 1.0, 0.9666666666666667, 0.9, 1.0]
0.9733333333333333
```

Part 5

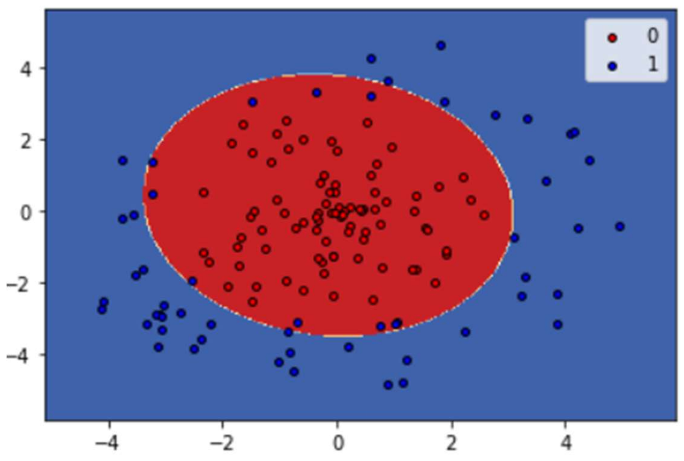
Since the classes are not linearly separable, we get the circular decision boundary as shown. The accuracy and the decision boundary show the generalizability of the model.

```
            X         Y  class
0   -0.598292  0.184666      0
1    1.895588  3.071458      1
2   -2.251511 -1.401709      0
3   -0.892532 -0.015036      0
4   -1.500317 -0.868722      0
..        ...       ...    ...
195 -0.878631  2.529979      0
196 -0.877398 -1.924083      0
197  0.093710 -0.119018      0
198  0.650437  0.530700      0
199  1.310787  0.380045      0

[200 rows x 3 columns]
<matplotlib.axes._subplots.AxesSubplot at 0x7f83ca07a5e0>
```



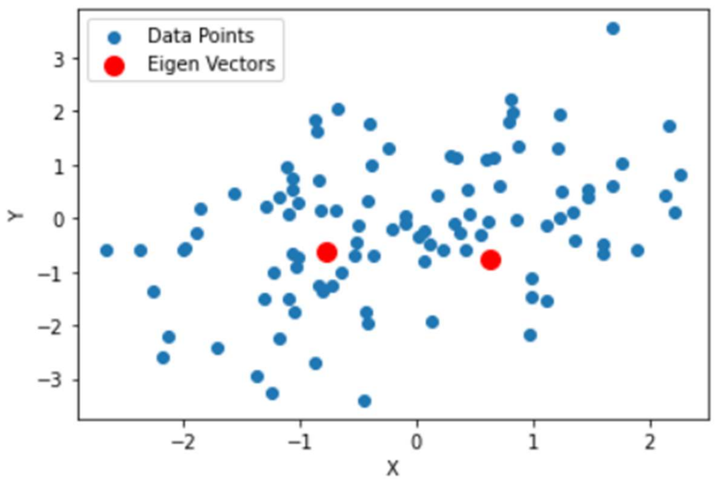
Decision Boundary Plot is: -



Problem 2

Part 1

Plotted random numbers form the given the covariance and the mean matrix. Found the eigen vectors and plotted on the same plot.



Part 2

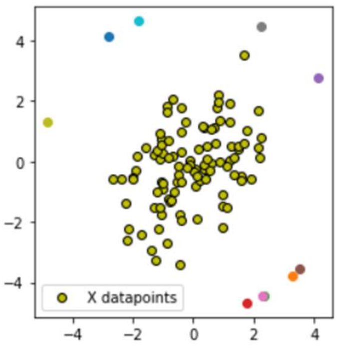
Performed the transformation by finding the inverse matrix and taking the root of the matrix. Found the covariance matrix of the newly generated dataset.

```
Covariance Matrix is
[[1.00000000e+00 6.42723809e-17]
 [6.42723809e-17 1.00000000e+00]]
```

The obtained covariance matrix is nearly an identity matrix.
The purposeof the tranformation is that since the naive bayes assumes that the features are indepen dent of each other so to minimize the correlation between the features we transform the matrix suc h that the covariance matrix becomes an identity matrix.

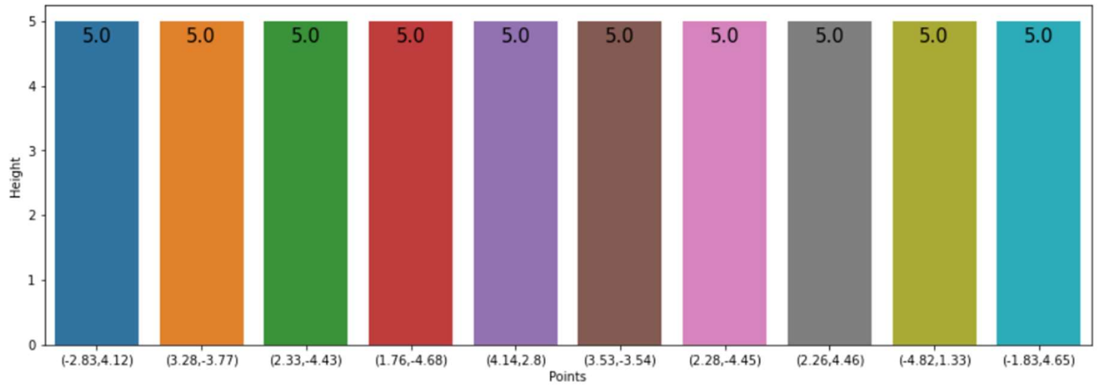
Part 3

Creating a random dataset as describe in question and plotting it as shown.



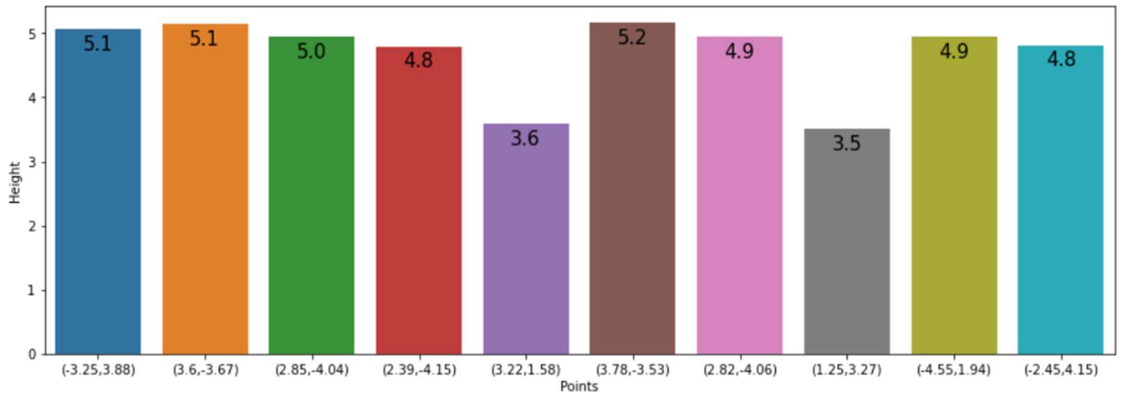
Part 4

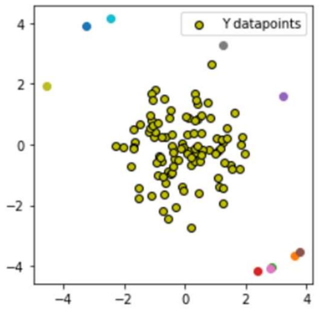
Now finding the distance from the mean as given in the question as plotting the bar graph for the same as shown: -



Part 5

Performing the transformation same as in part 3 and again plotting the distance bar again.


Height
Points
(-3.25,3.88)
(3.6,-3.67)
(2.85,-4.04)
(2.39,-4.15)
(3.22,1.58)
(3.78,-3.53)
(2.82,-4.06)
(1.25,3.27)
(-4.55,1.94)
(-2.45,4.15)


Y datapoints