CS5691: Pattern Recognition and Machine Learning
Assignment #1

Topics: K-Nearest Neighbours, Naive Bayes, Regression **Deadline:** 28 Feb 2023, 11:55 PM

Teammate 1: Smit Bagul **Roll number:** CS20B011
Teammate 2: Karan Bardhan **Roll number:** CS20B036

Contributions :
CS20B011 : Q1 Ridge regression and Q2
CS20B036 : Q1 and Q3
Plotting, testing and debugging was done collaboratively by both

1. **[Regression]** You will implement linear regression as part of this question for the dataset1 provided here.

   Note that you can only regress over the points in the train dataset and you are not supposed to fit a curve on the test dataset. Whatever solution you get for the train data, you have to use that to make predictions on the test data and report results.

   Google colab notebook solution for Q1 : here

   (a) (2 marks) Use standard linear regression to get the best-fit curve. Split the data into train and validation sets and try to fit the model using a degree 1 polynomial then vary the degree term of the polynomial to arrive at an optimal solution.

   For this, you are expected to report the following -

   - Plot different figures for train and validation data and for each figure plot curve of obtained function on data points for various degree term of the polynomial.(refer to fig. 1.4, Pattern Recognition and Machine Learning, by Christopher M. Bishop).
   - Plot the curve for Mean Square Error(MSE) Vs degree of the polynomial for train and validation data.(refer to fig. 1.5, Pattern Recognition and Machine Learning, by Christopher M. Bishop)
   - Report the error for the best model using Mean Square Error(MSE) for train and test data provided(Use closed-form solution).
   - Scatter plot of best model output vs expected output for both train and test data provided to you.
   - Report the observations from the obtained plots.

   Solution:
   Plot of train and validation data vs degree of polynomial:

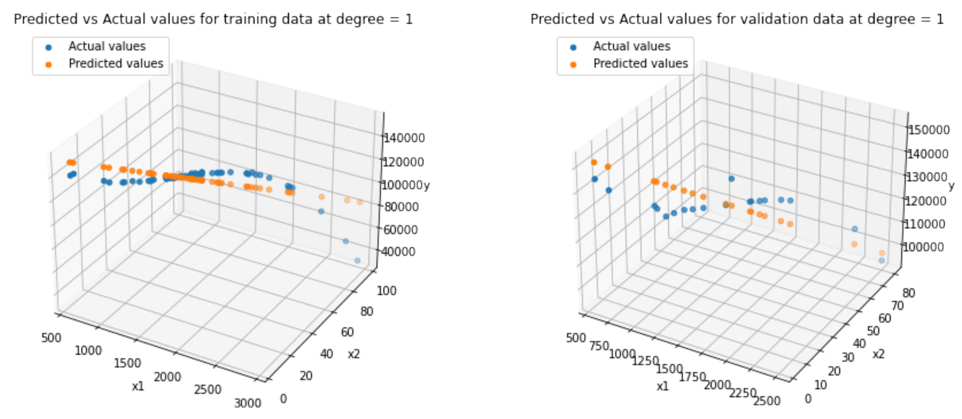


Figure 1: Polynomial degree = 1

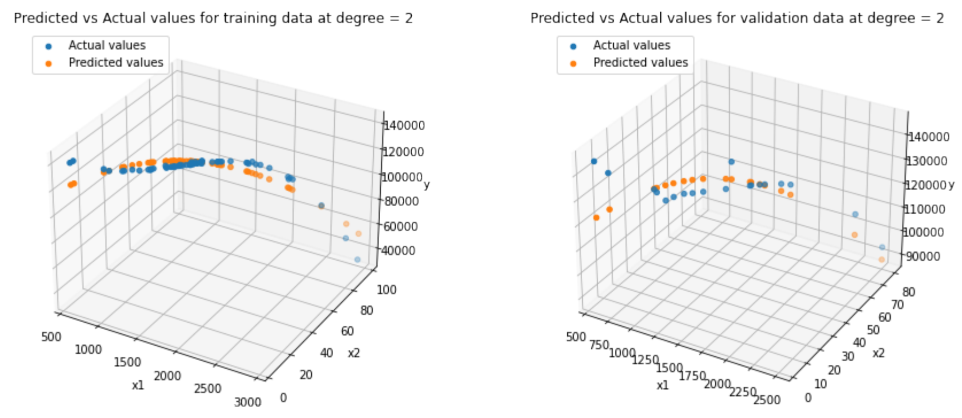


Figure 2: Polynomial degree = 2

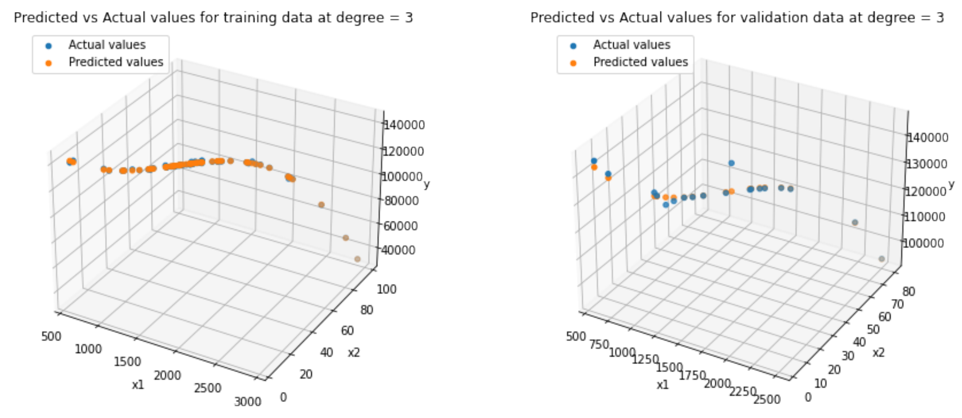


Figure 3: Polynomial degree = 3

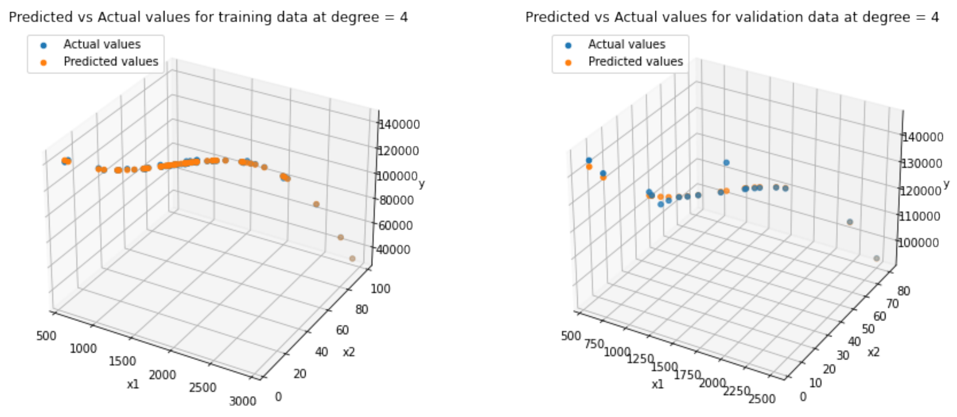


Figure 4: Polynomial degree = 4

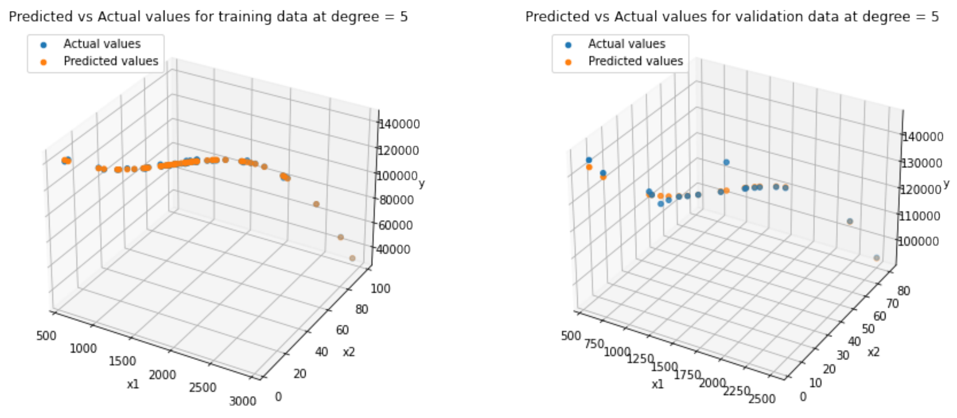


Figure 5: Polynomial degree = 5

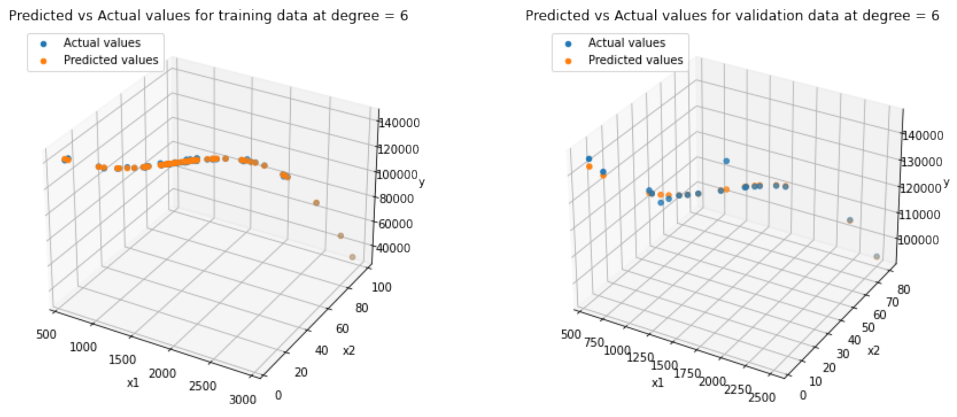


Figure 6: Polynomial degree = 6

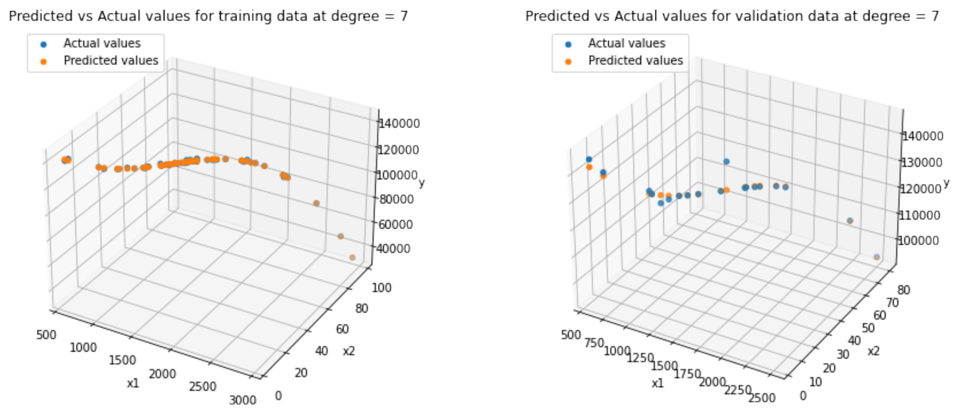


Figure 7: Polynomial degree = 7

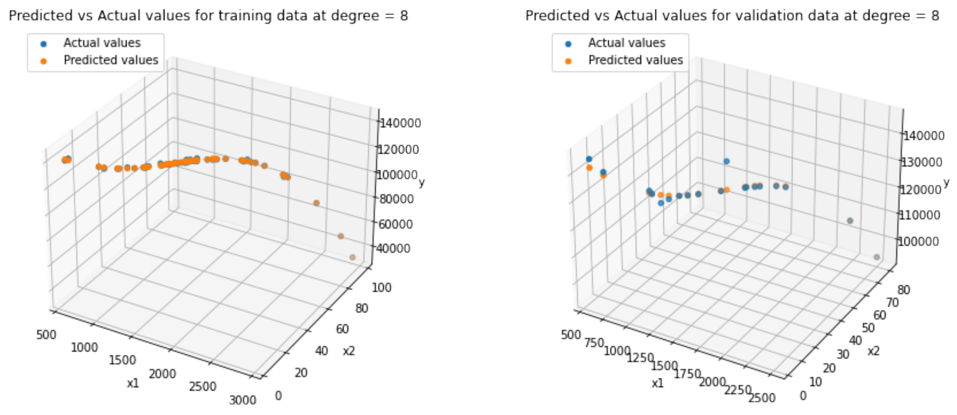


Figure 8: Polynomial degree = 8

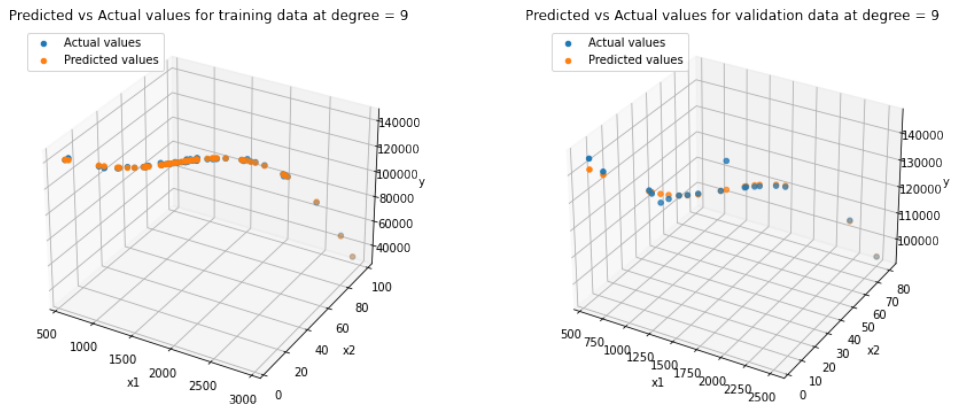


Figure 9: Polynomial degree = 9

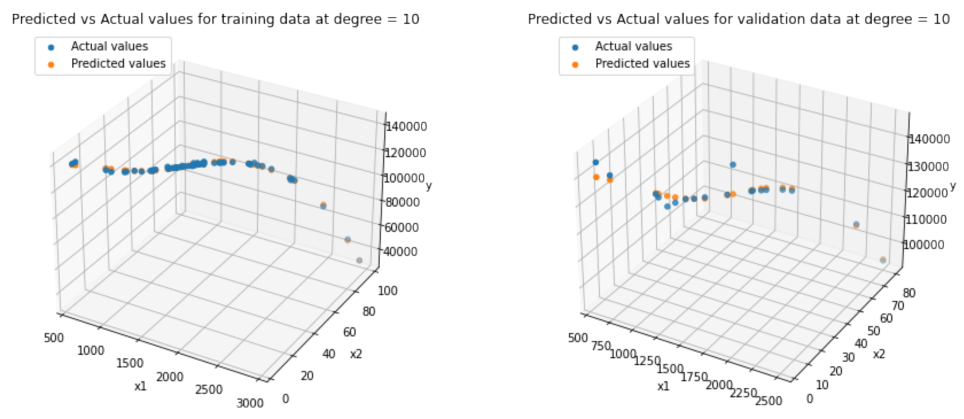


Figure 10: Polynomial degree = 10

Plot of MSE vs degree of polynomial:

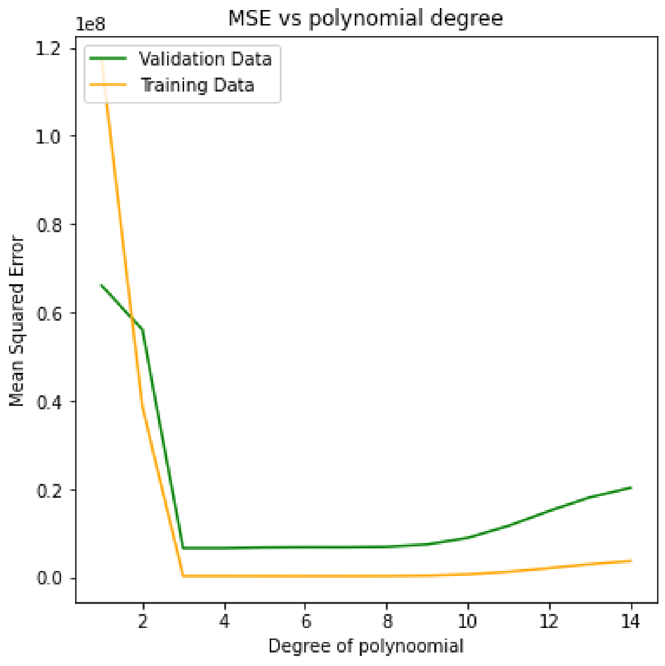
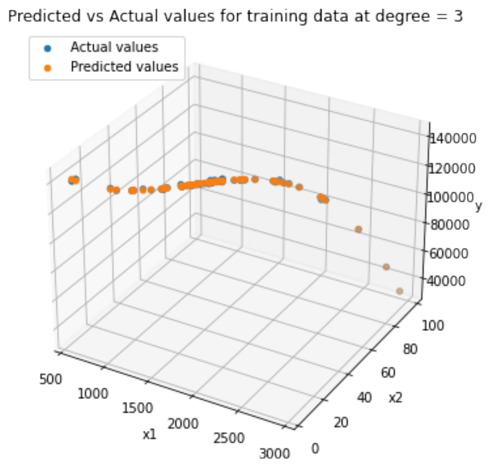


Figure 11: MSE

The Best Model (n = 3) yields an MSE = 361186.1995475413 on the training data
The Best Model (n = 3) yields an MSE = 527459.7402422356 on the testing data

Scatter Plot for best model output vs expected output:



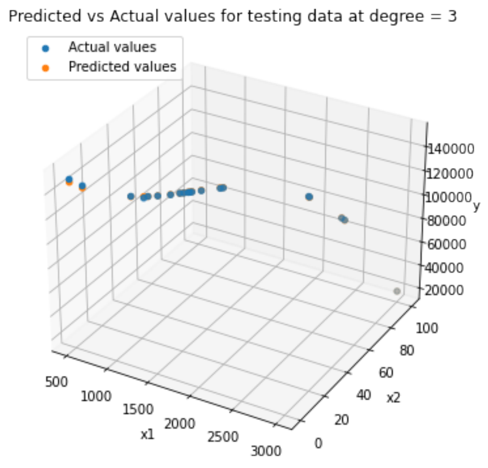


Figure 12: Polynomial degree = 3

Observations :
As Degree increases till n = 3, the MSE decreases after which the MSE on training data remains constant whereas in the validation data the MSE begins to increase after n = 10. This is where the model begins to overfit the data.
The model had a good accuracy for the test data as seen in the plots.

(b) (3 marks) Split the data into train and validation sets and use ridge regression, then report for which value of lambda (λ) you obtain the best fit. For this, you are expected to report the following -

- Choose the degree from part (a), where the model overfits and try to control it using the regularization technique (Ridge regression).
- Use various choices of lambda(λ) and plot MSE test Vs lambda(λ).
- Report the error for the best model using Mean Square Error(MSE) for train and test data provided (Use closed-form solution).
- Scatter plot of best model output vs expected output for both train and test data provided to you.
- Report the observations from the obtained plots.

Solution:
We have chosen degree 12 polynomial because it was overfitting.

Plot of MSE test vs $\log_2(\lambda)$:

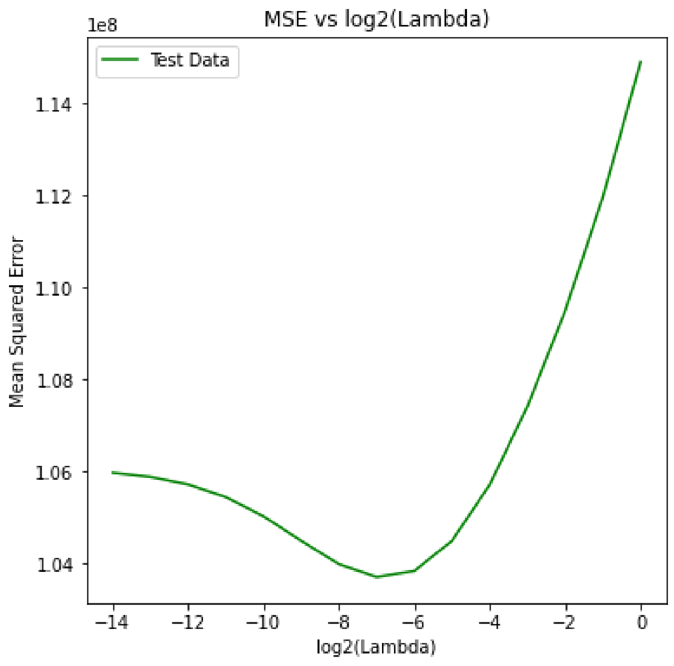


Figure 13: MSE vs $\log_2(\lambda)$

The Best Model ($\lambda = 0.0078125$) yields an MSE = 399973.14290526044 on the training data
The Best Model ($\lambda = 0.0078125$) yields an MSE = 103698729.6069701 on the testing data

Scatter Plot of best model output vs expected output:

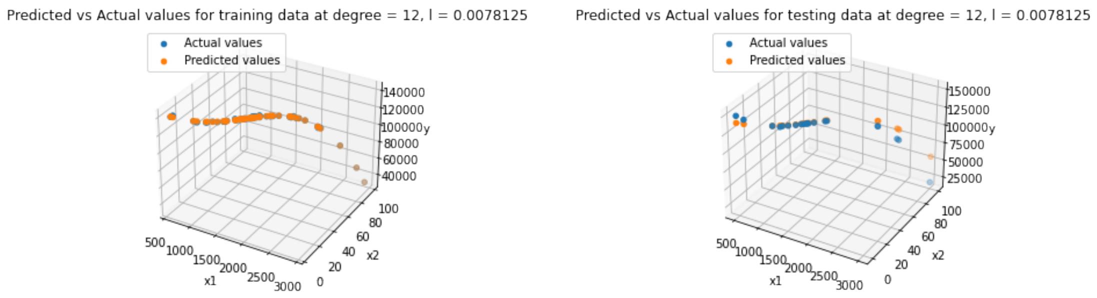


Figure 14: Best Model Plot

Observations :
The MSE value for a degree 12 polynomial initially decreases as we decrease the value of λ and it reaches a minimum value at $\lambda = 0.0078125$ after which it begins to increase again.
The accuracy on both test and training data is greatly improved due to ridge regression.

2. **[Naive Bayes Classifier]** In this Question, you are supposed to build Naive Bayes classifiers for the datasets assigned to your team. Train and test datasets for each team can be found here. For each sub-question below, the report should include the following:

   - Accuracy on both train and test data.
   - Plot of the test data along with your classification boundary.
   - confusion matrices on both train and test data.

   You can refer to sample plots here and can refer Section 2.6 of "Pattern classification" book by [Duda et al. 2001] for theory.

   Google colab notebook solution for Q2 : here

   (a) (1 mark) Implement Naive Bayes classifier with covariance = I on dataset2. where, I denotes the identity matrix.

   Solution:
   Accuracy for training data = 100%
   Accuracy for testing data = 99%

Confusion Matrix for training data :

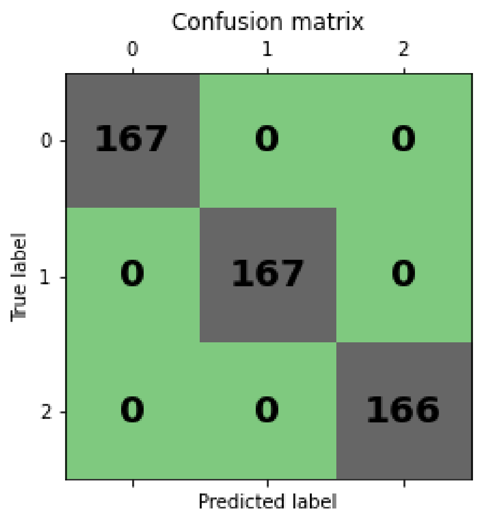


Figure 15: Confusion Matrix train_23

Confusion matrix for test data:

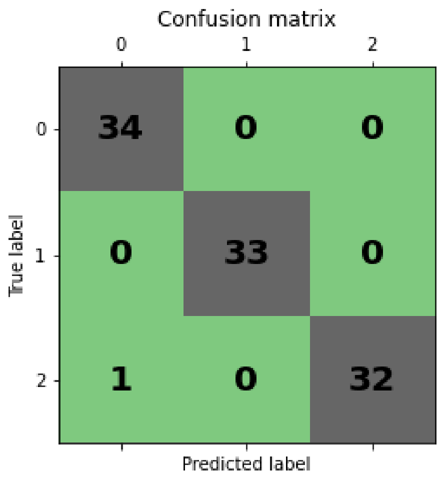


Figure 16: Confusion Matrix test_23

Plot of test data :

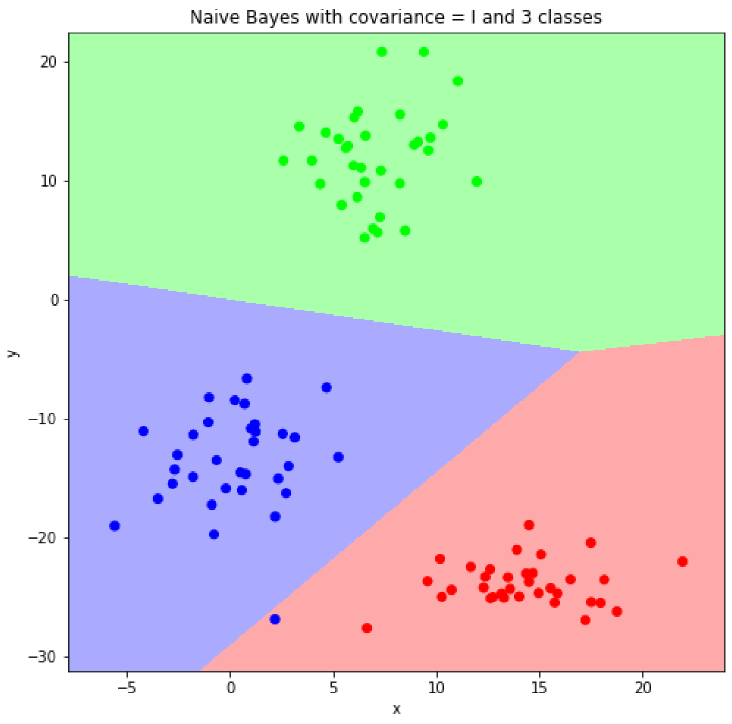


Figure 17: Classification boundary

(b) (1 mark) Implement Naive Bayes classifier with covariance = I on dataset3. where, I denotes the identity matrix.

Solution:
Accuracy for training data = 79.2%
Accuracy for testing data = 76%

Confusion Matrix for training data :

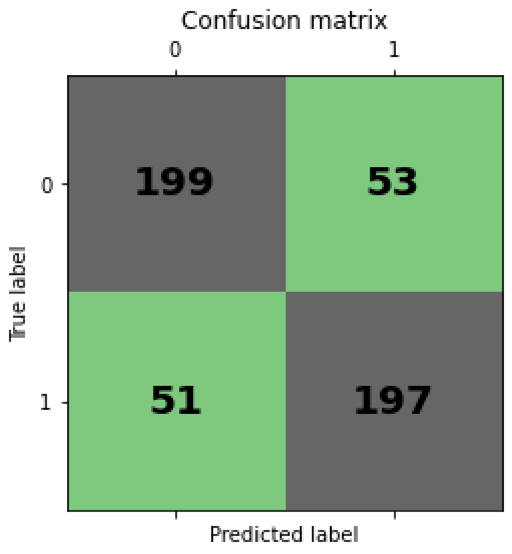


Figure 18: Confusion Matrix train_23

Confusion matrix for test data:

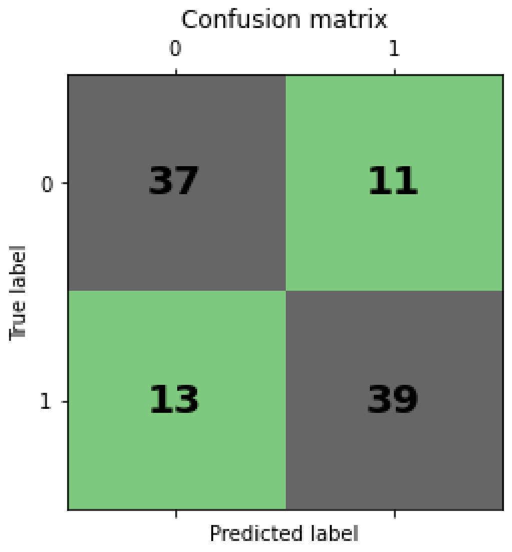


Figure 19: Confusion Matrix test_23

Plot of test data :

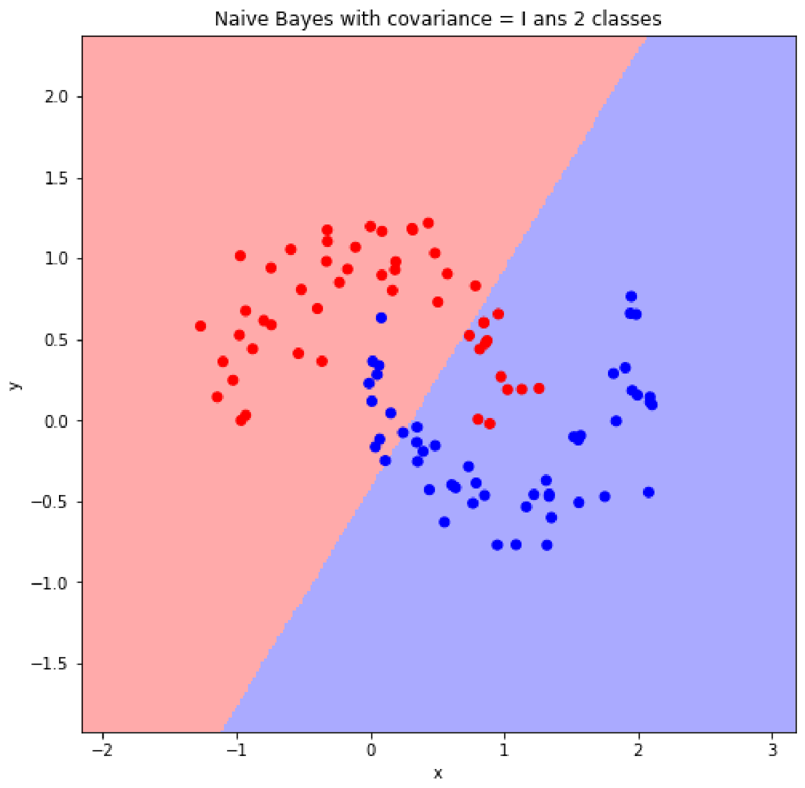


Figure 20: Classification boundary

(c) (1 mark) Implement Naive Bayes classifier with covariance same for all classes on dataset2.

Solution:
Accuracy for training data = 100%
Accuracy for testing data = 100%

Confusion Matrix for training data :

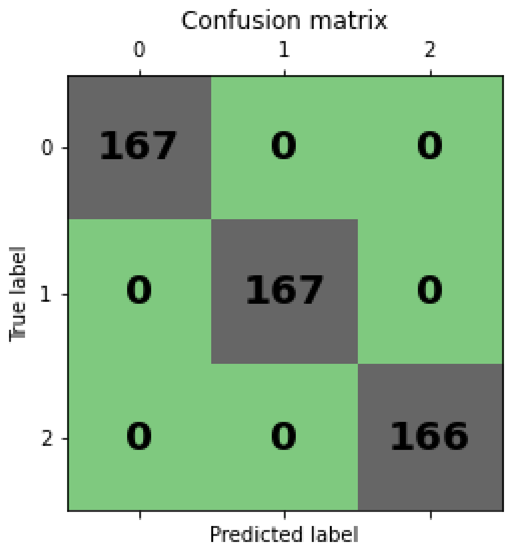


Figure 21: Confusion Matrix train_23

Confusion matrix for test data:

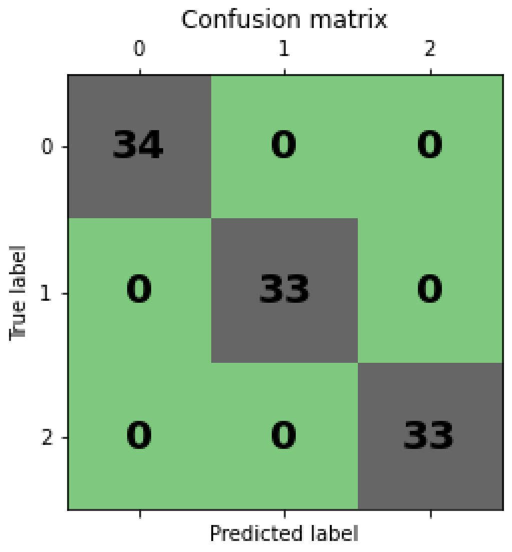


Figure 22: Confusion Matrix test_23

Plot of test data :

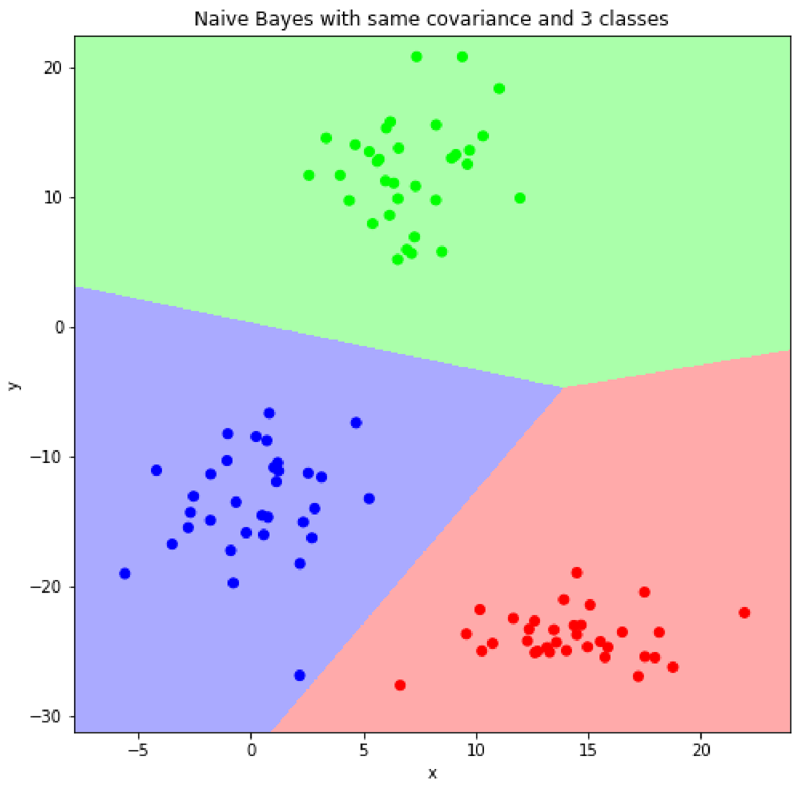


Figure 23: Classification boundary

(d) (1 mark) Implement Naive Bayes classifier with covariance same for all classes on dataset3.

Solution:
Accuracy for training data = 87.6%
Accuracy for testing data = 88%

Confusion Matrix for training data :

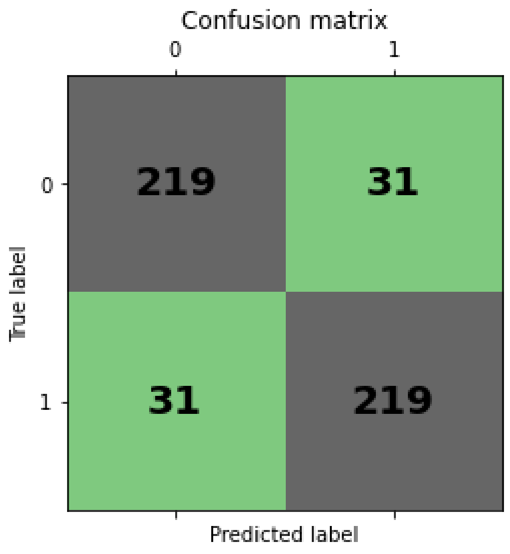


Figure 24: Confusion Matrix train_23

Confusion matrix for test data:

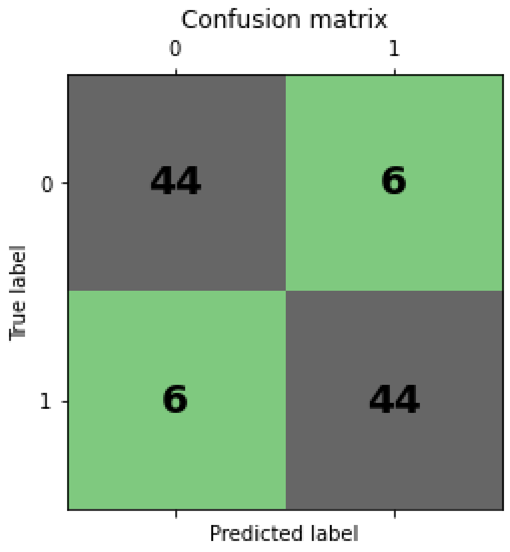


Figure 25: Confusion Matrix test_23

Plot of test data :

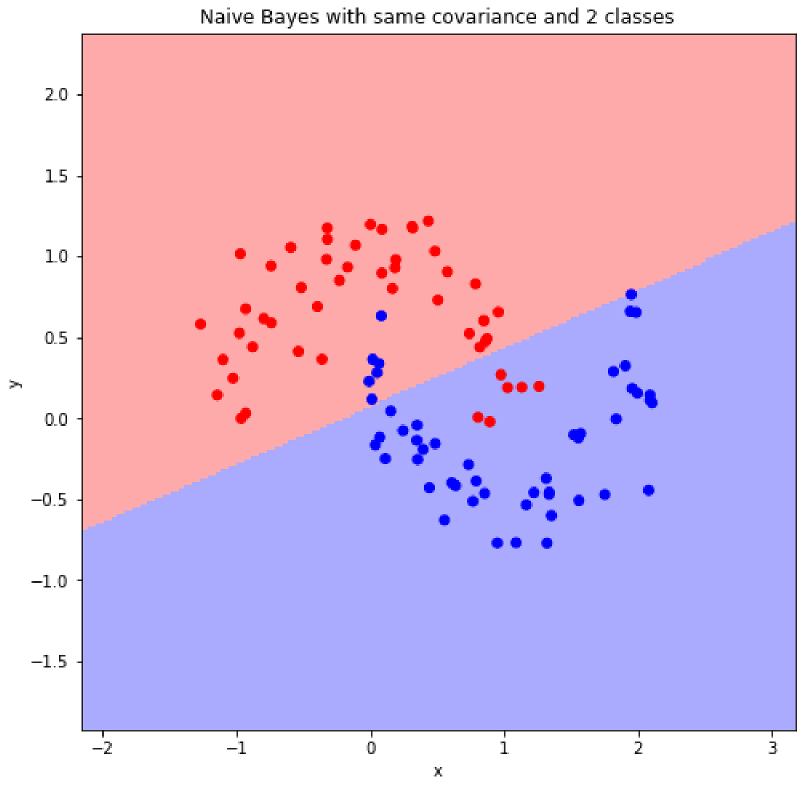


Figure 26: Classification boundary

(e) (1 mark) Implement Naive Bayes classifier with covariance different for all classes on dataset2.

Solution:
Accuracy for training data = 100%
Accuracy for testing data = 100%

Confusion Matrix for training data :

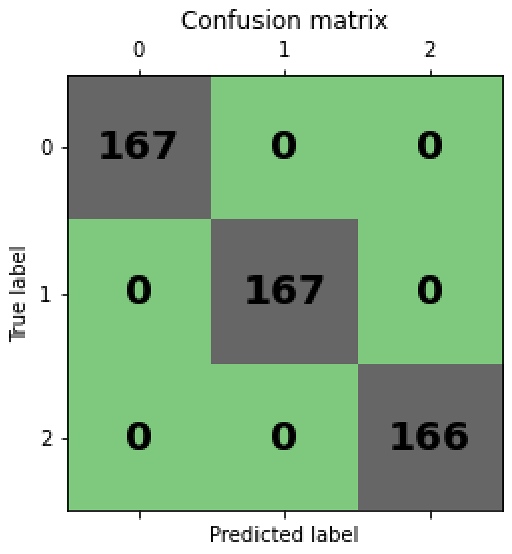


Figure 27: Confusion Matrix train_23

Confusion matrix for test data:

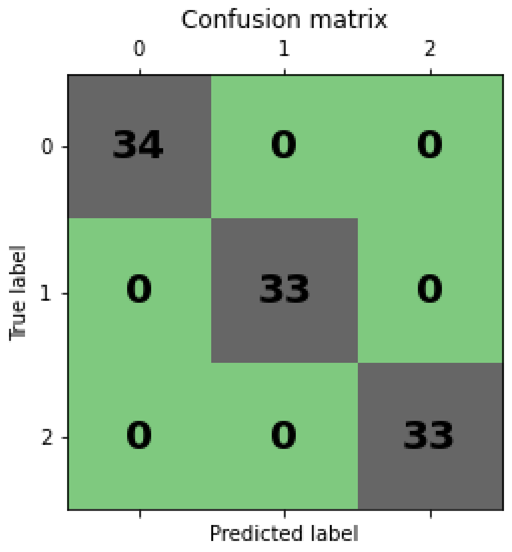


Figure 28: Confusion Matrix test_23

Plot of test data :

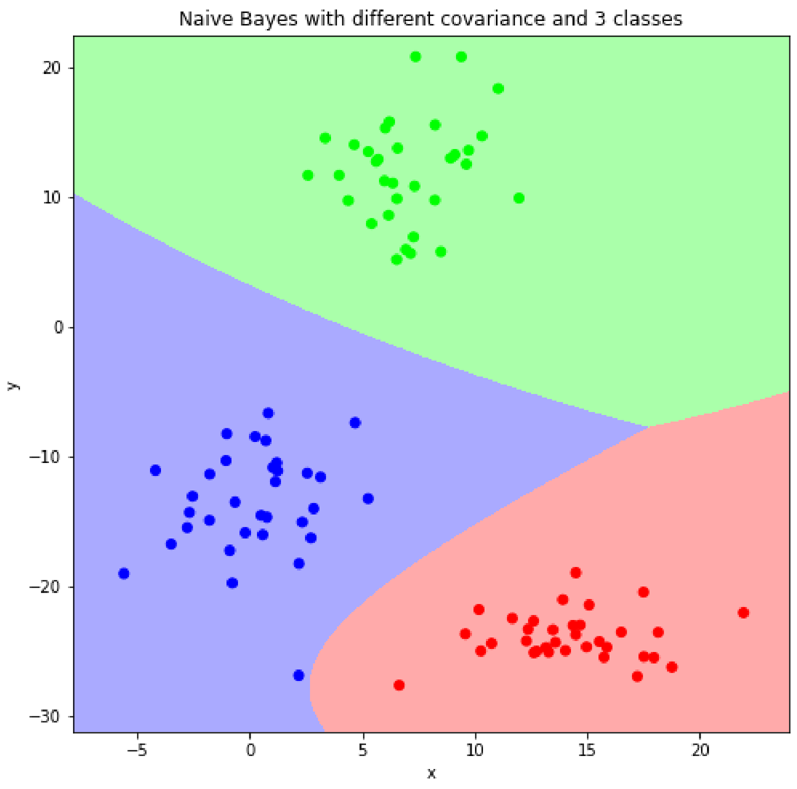


Figure 29: Classification boundary

(f) (1 mark) Implement Naive Bayes classifier with covariance different for all classes on dataset3.

Solution:
Accuracy for training data = 87.6%
Accuracy for testing data = 87%

Confusion Matrix for training data :

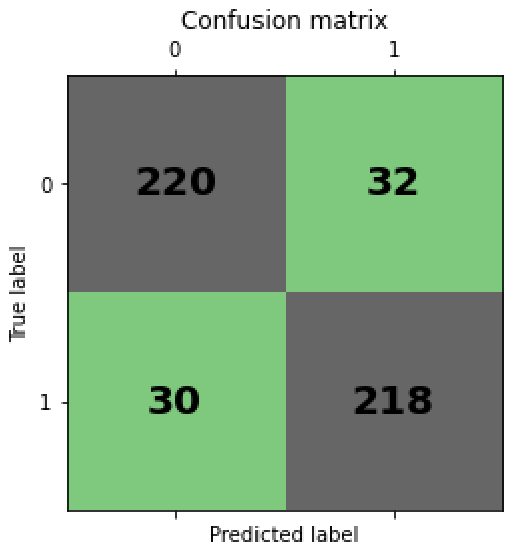


Figure 30: Confusion Matrix train_23

Confusion matrix for test data:

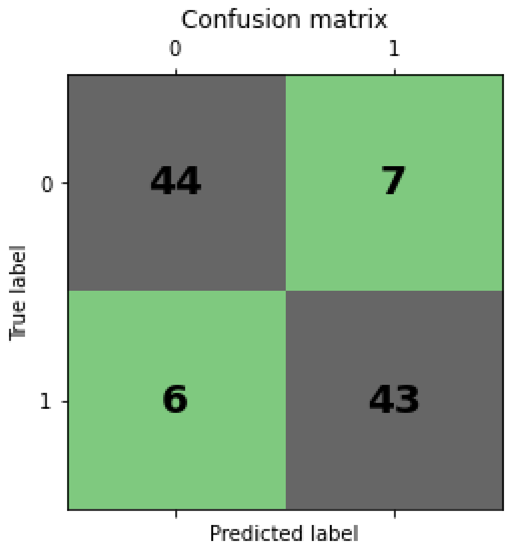


Figure 31: Confusion Matrix test_23

Plot of test data :

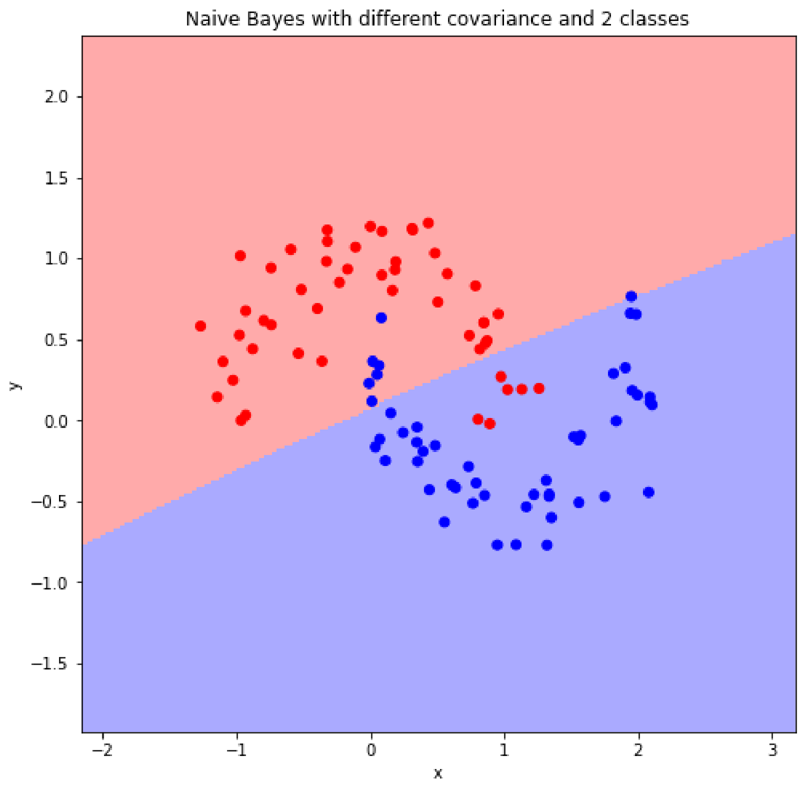


Figure 32: Classification boundary

3. **[KNN Classifier]** In this Question, you are supposed to build the k-nearest neighbors classifiers on the datasets assigned to your team. Dataset for each team can be found here. For each sub-question below, the report should include the following:

   - Analysis of classifier with different values of k (number of neighbors).
   - Accuracy on both train and test data for the best model.
   - Plot of the test data along with your classification boundary for the best model.
   - confusion matrices on both train and test data for the best model.

   Google colab notebook solution for Q3 : here

   (a) (2 marks) Implement k-nearest neighbors classifier on dataset2.

   Solution:
   Best model is k = 1 with accuracy of 100% on both train and test data

Plot of Accuracy vs K:

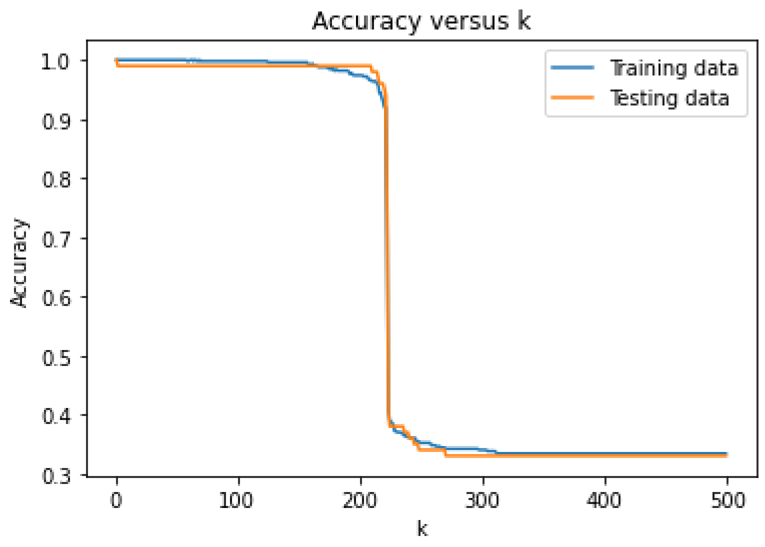


Figure 33: Analysis of K

Confusion Matrix for train data using k=1 :

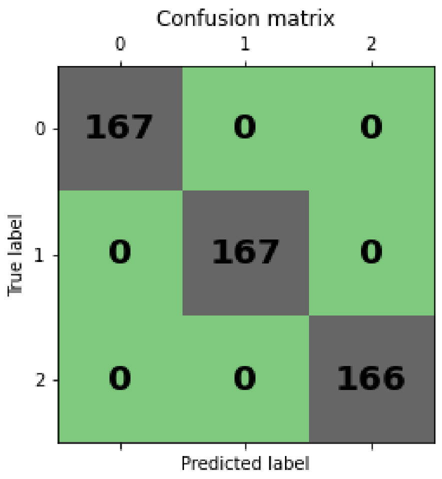


Figure 34: Confusion Matrix train_23

Confusion matrix for test data using k=1:

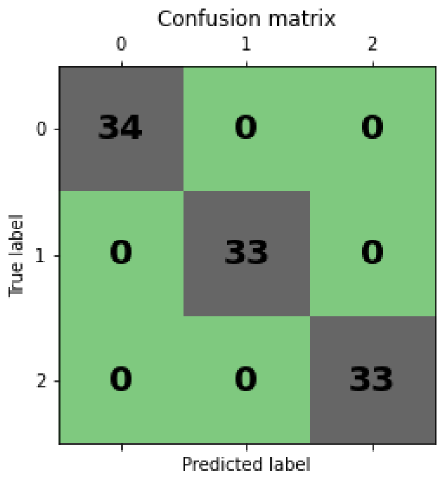


Figure 35: Confusion Matrix test_23

Plot of test data using k=1:

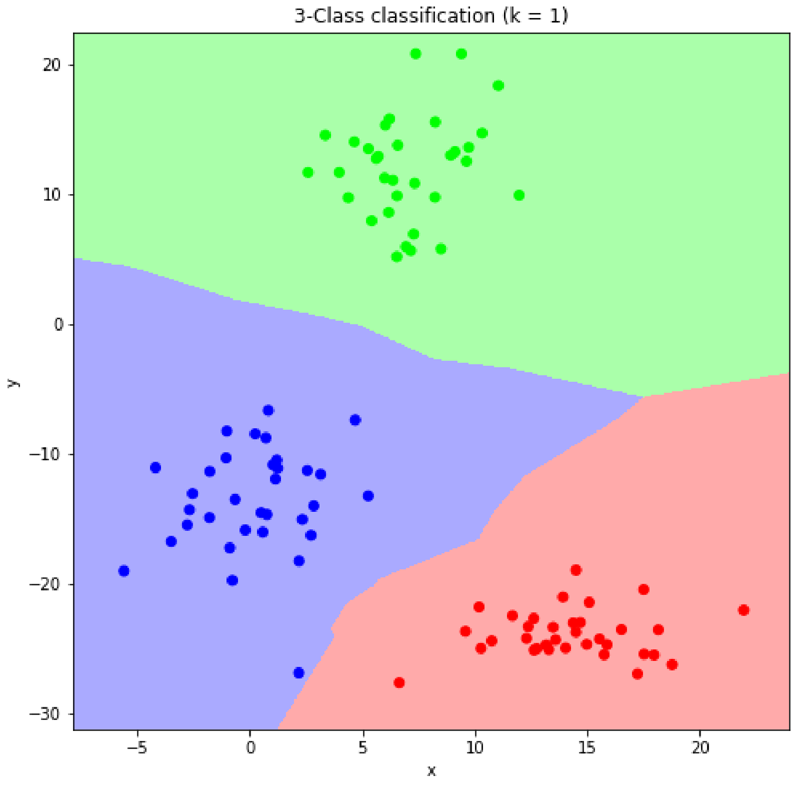


Figure 36: Classification boundary

(b) (2 marks) Implement k-nearest neighbors classifier on dataset3.

Solution:
Best model is k = 1 with accuracy of 100% on both train and test data Plot of Accuracy vs K:

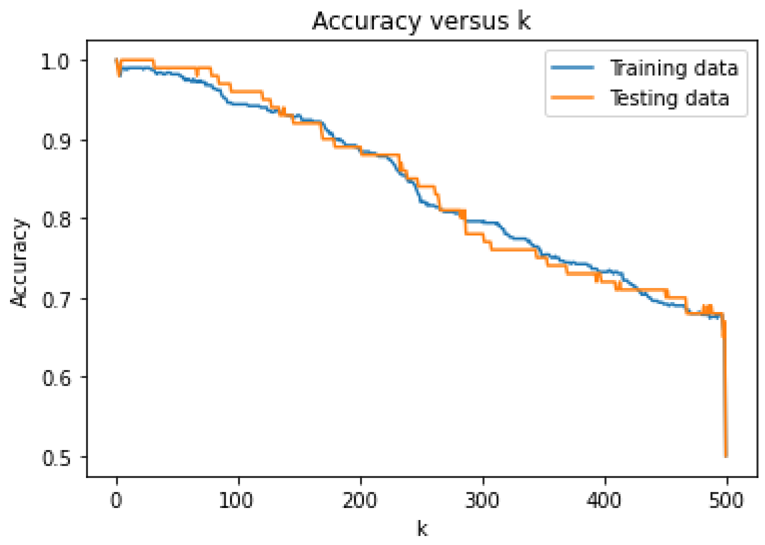


Figure 37: Analysis of K

Confusion Matrix for train data using k=1 :

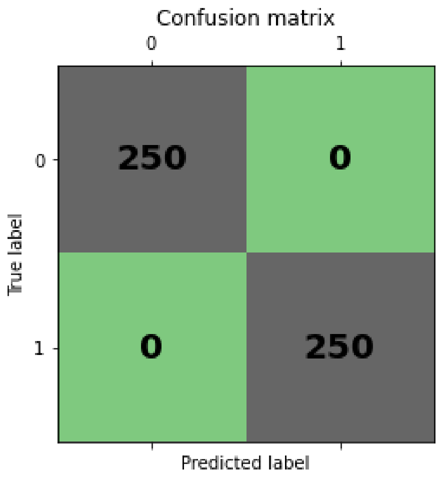


Figure 38: Confusion Matrix train_23

Confusion matrix for test data using k=1:

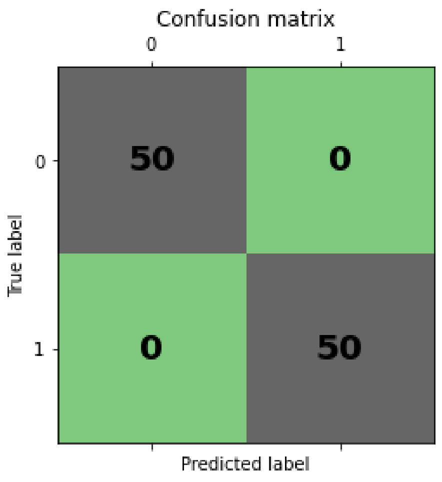


Figure 39: Confusion Matrix test_23

Plot of test data using k=1:

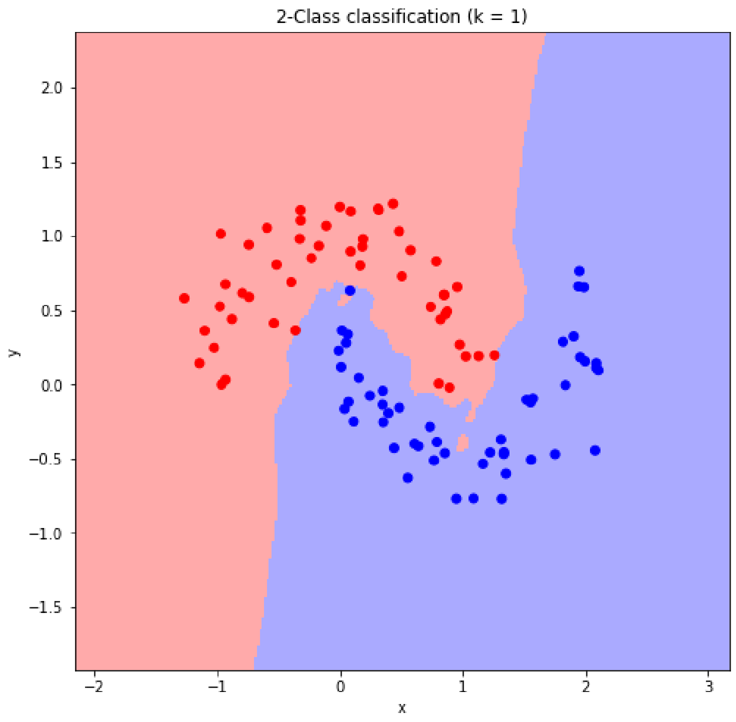


Figure 40: Classification boundary