CS5691: Pattern Recognition and Machine Learning
Assignment #1

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

Teammate 1: Soham Tripathy **Roll number:** CS20B073
Teammate 2: Sooraj Srinivasan **Roll number:** CS20B075

- **Sooraj Srinivasan:** Coded the implementations of Linear Regression, Ridge Regression, and Naive Bayes Classifier Algorithms. Plotted the graphs for question 2 and completed the report for the same.
- **Soham Tripathy:** Extended the Linear Regression model to implement polynomial regression, Implemented the K-NN algorithm. Ploted the graphs for question 1, 3 and completed the report for the same.
- All the algorithms were discussed before implementation.
- We also implemented the lasso regression in our .ipynb file

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

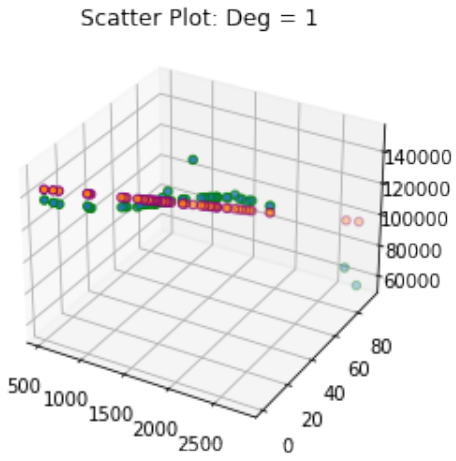
   Note that you can only regress over the points in the train dataset and you are not supposed to fit a curve on the test dataset. Whatever solution you get for the train data, you have to use that to make predictions on the test data and report results.

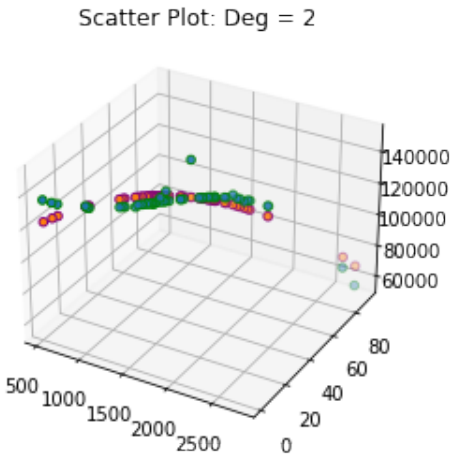
   (a) (2 marks) Use standard linear regression to get the best-fit curve. Split the data into train and validation sets and try to fit the model using a degree 1 polynomial then vary the degree term of the polynomial to arrive at an optimal solution.

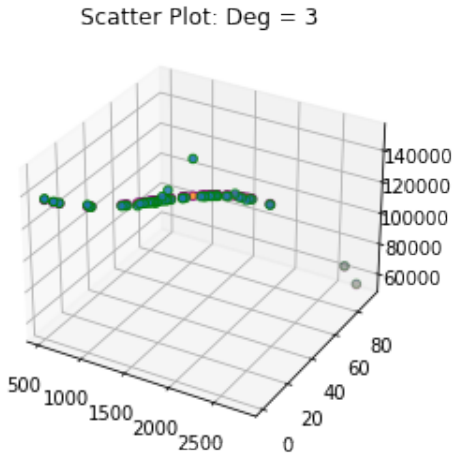
   For this, you are expected to report the following -

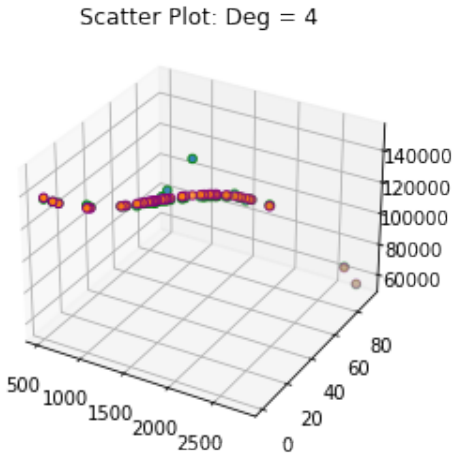
   - Plot different figures for train and validation data and for each figure plot curve of obtained function on data points for various degree term of the polynomial.(refer to fig. 1.4, Pattern Recognition and Machine Learning, by Christopher M. Bishop).
   - Plot the curve for Mean Square Error(MSE) Vs 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.

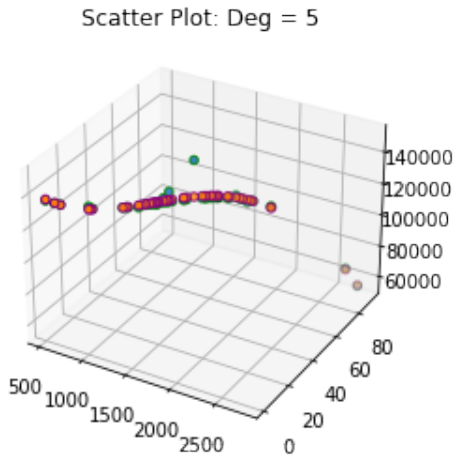
Solution: The green color in the graph indicates the training dataset plot and the red color indicates the prediction plot.

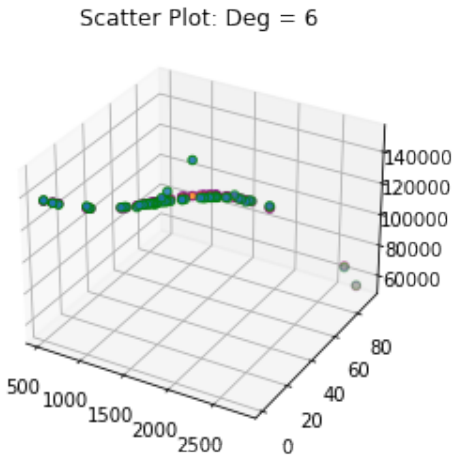


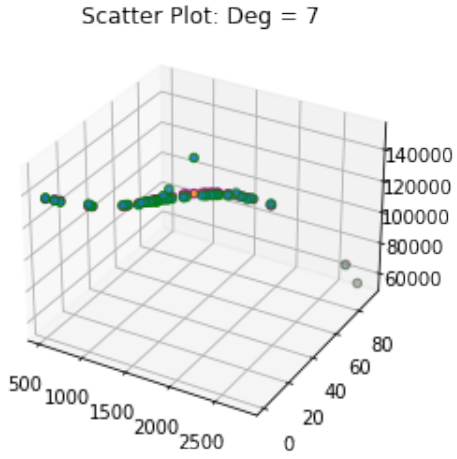




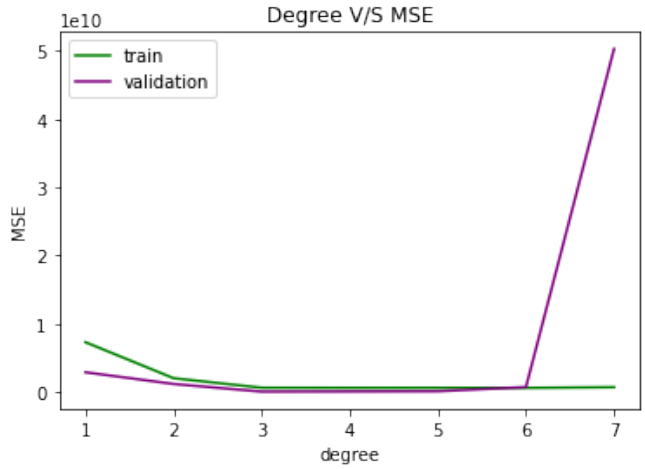








The MSE v/s degree graph is given below.

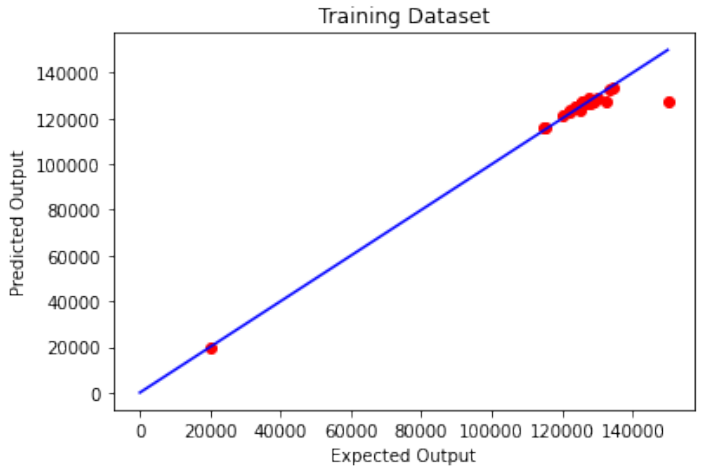


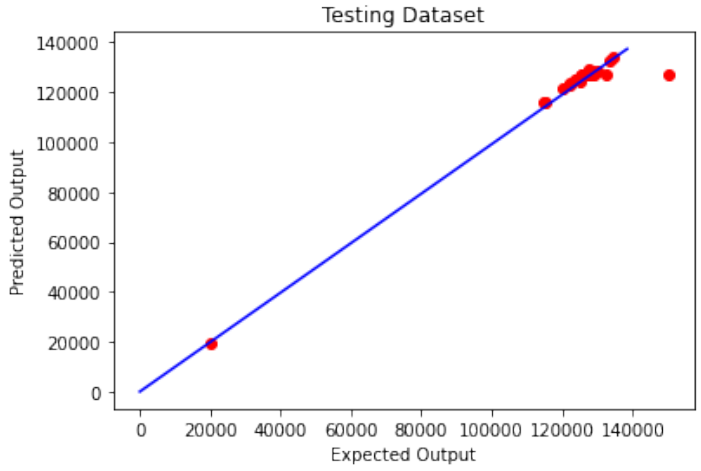
The Best model that fits the dataset is degree 3. The MSE on the training dataset and the test dataset are given below.

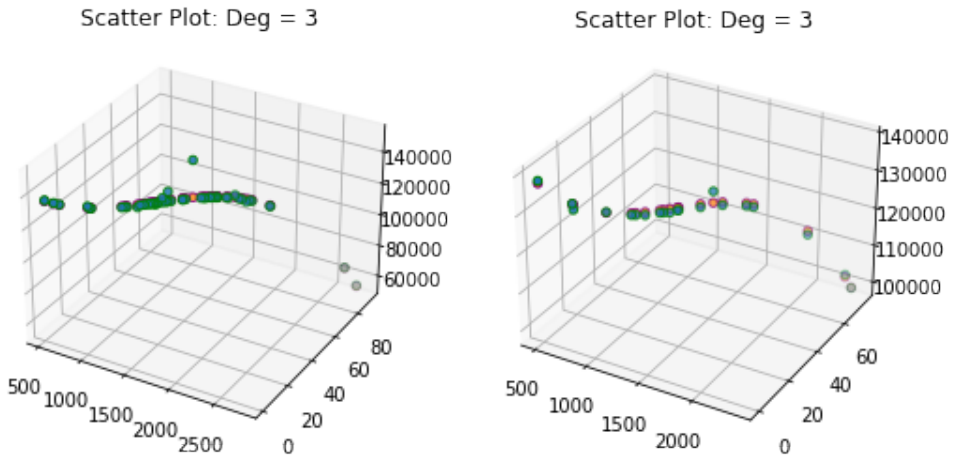
$$MSE_{train} = 592494636.008799$$

$$MSE_{test} = 20014971.76918808$$

The scatter plot of best model output v/s expected output.







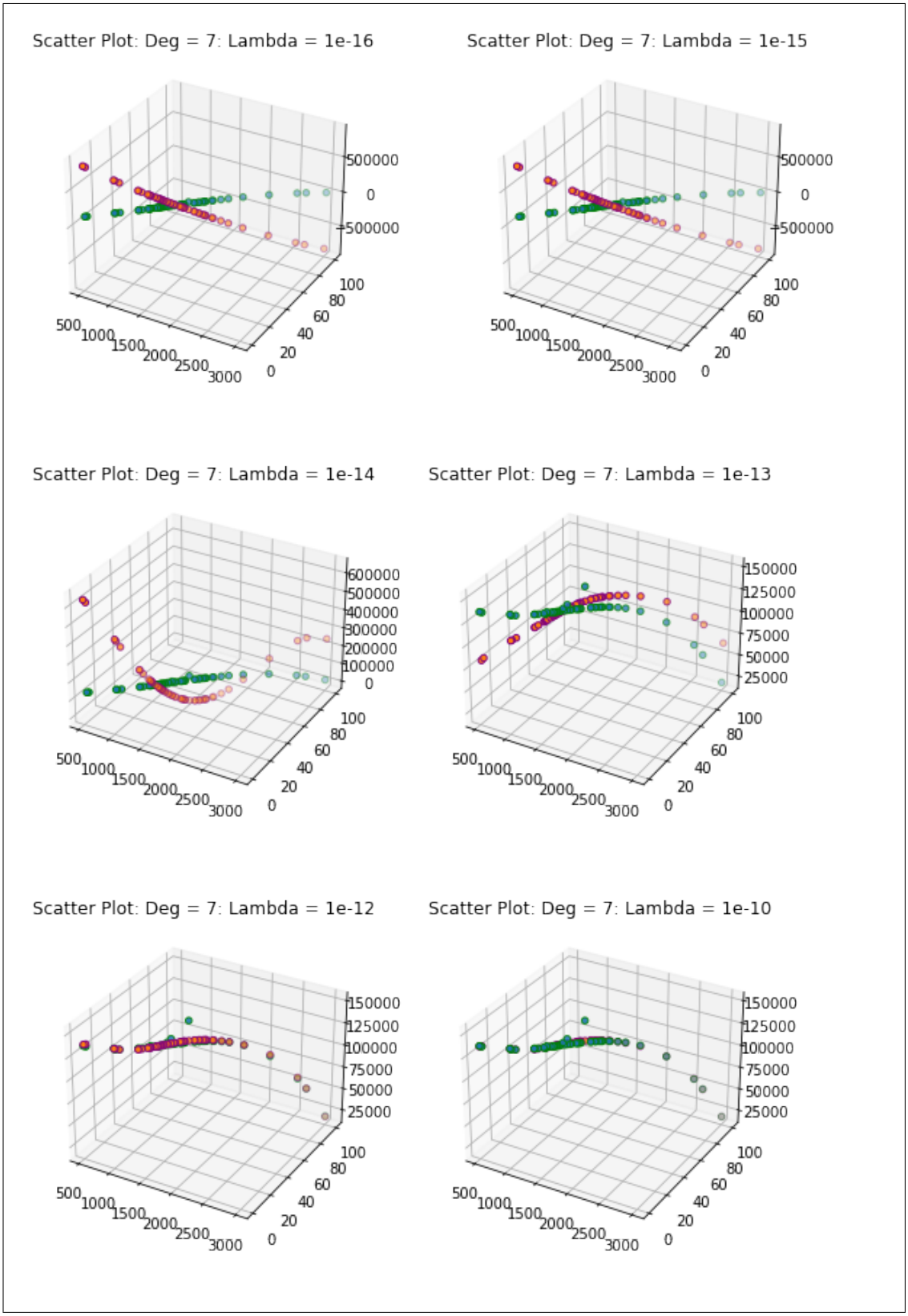
Training Testing

(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 -

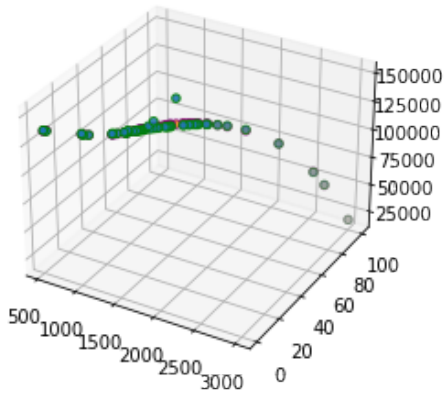
- Choose the degree from part (a), where the model overfits and try to control it using the regularization technique (Ridge regression).
- Use various choices of lambda(λ) and plot MSE test Vs lambda(λ).
- Report the error for the best model using Mean Square Error(MSE) for train and test data provided (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:

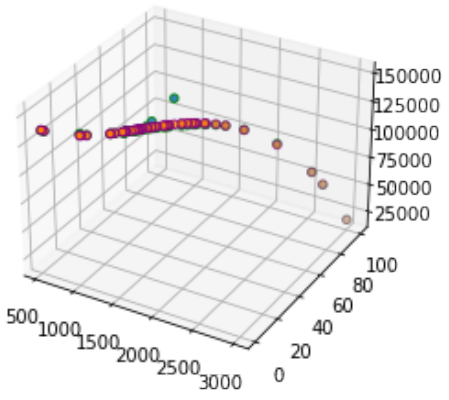
The green color in the graph indicates the training dataset plot and the red color indicates the prediction plot which is regularized using lambda's (mentioned in the graph).


Scatter Plot: Deg = 7: Lambda = 1e-16
Scatter Plot: Deg = 7: Lambda = 1e-15
Scatter Plot: Deg = 7: Lambda = 1e-14
Scatter Plot: Deg = 7: Lambda = 1e-13
Scatter Plot: Deg = 7: Lambda = 1e-12
Scatter Plot: Deg = 7: Lambda = 1e-10

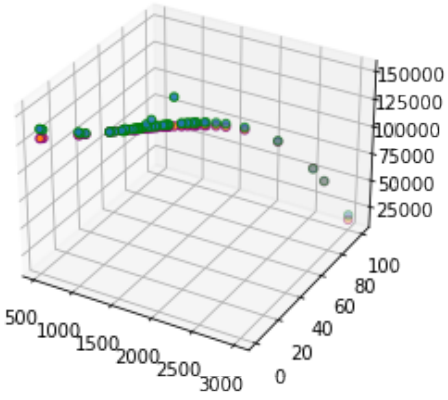
Scatter Plot: Deg = 7: Lambda = 1e-8



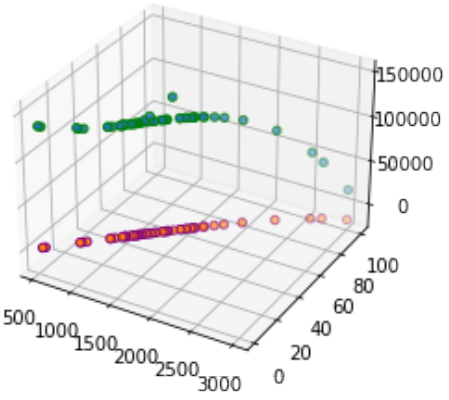
Scatter Plot: Deg = 7: Lambda = 1e-4



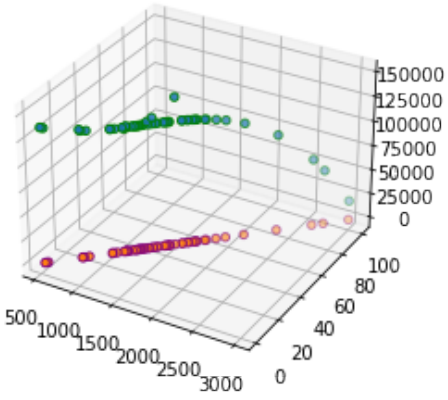
Scatter Plot: Deg = 7: Lambda = 1e0



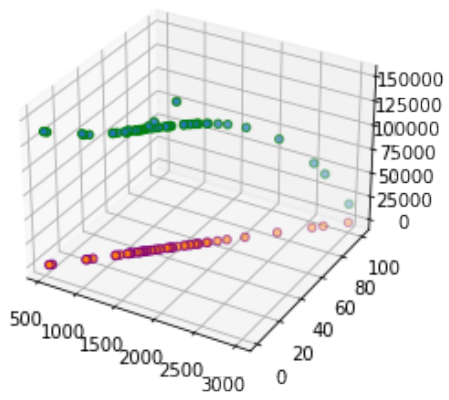
Scatter Plot: Deg = 7: Lambda = 1e4



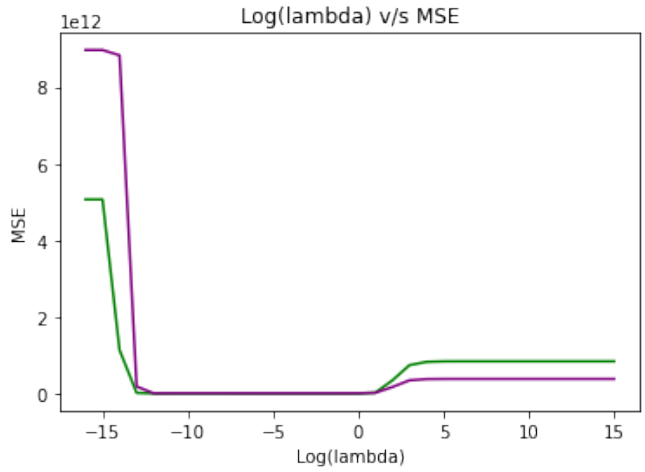
Scatter Plot: Deg = 7: Lambda = 1e8



Scatter Plot: Deg = 7: Lambda = 1e12



The MSE v/s lambda graph is given below.

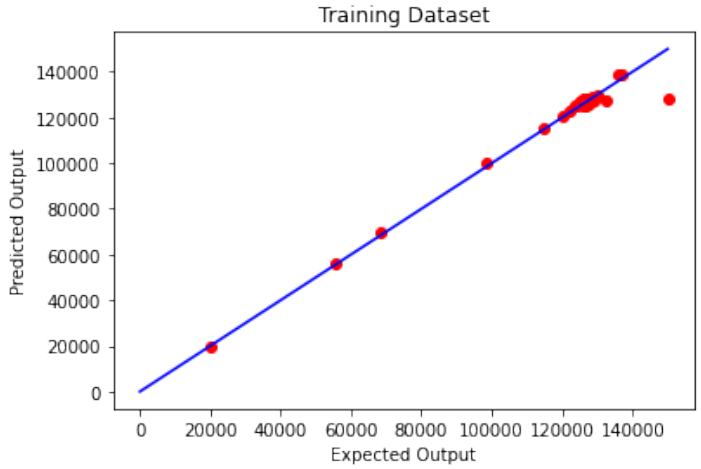


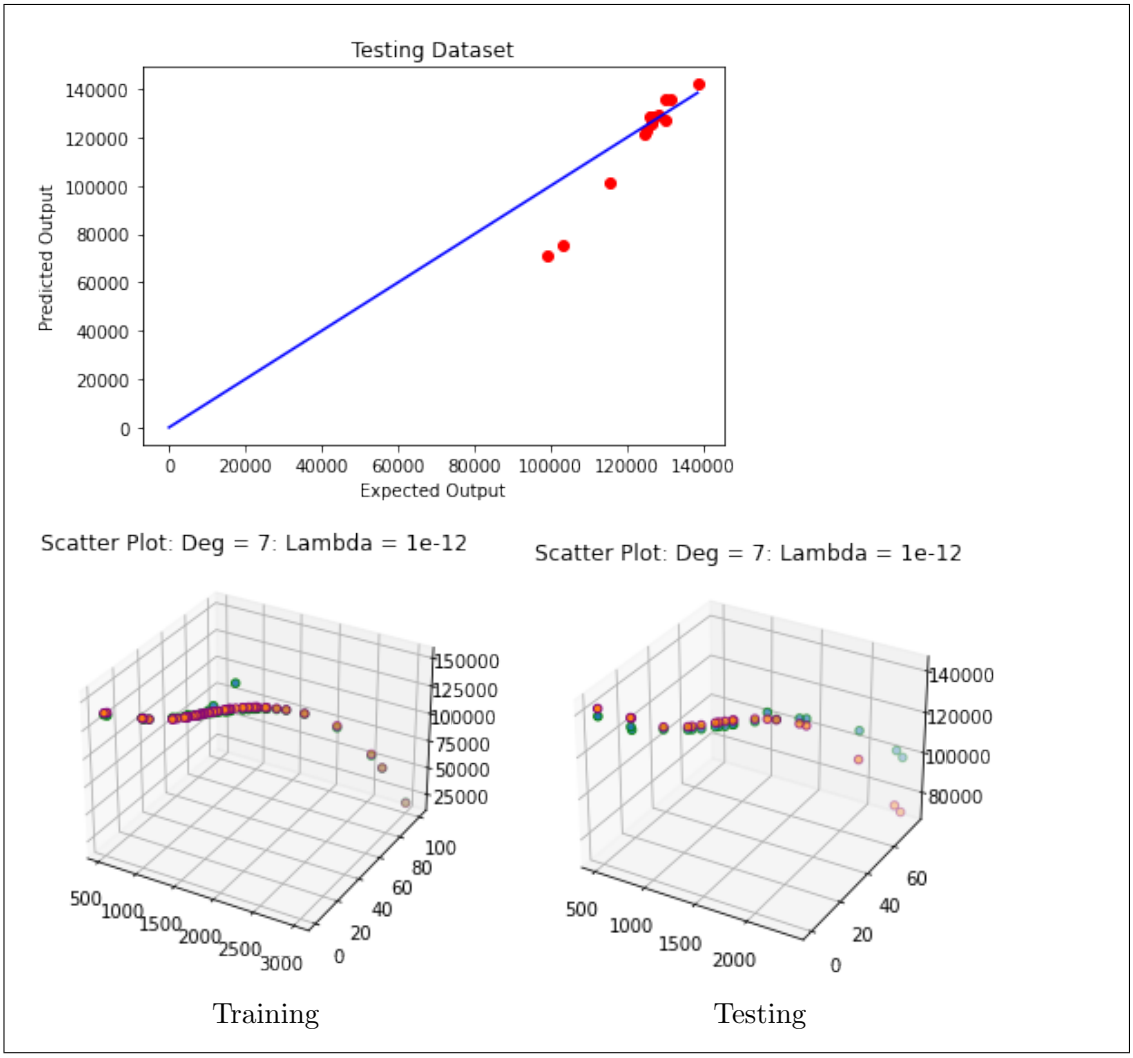
The Best model that fits the dataset is lambda 1e-12 (found using validation set) . The MSE on the training dataset and the test dataset are given below.

$$MSE_{train} = 584435341.1201459$$

$$MSE_{test} = 1869058295.1067886$$

The scatter plot of best model output v/s expected output.





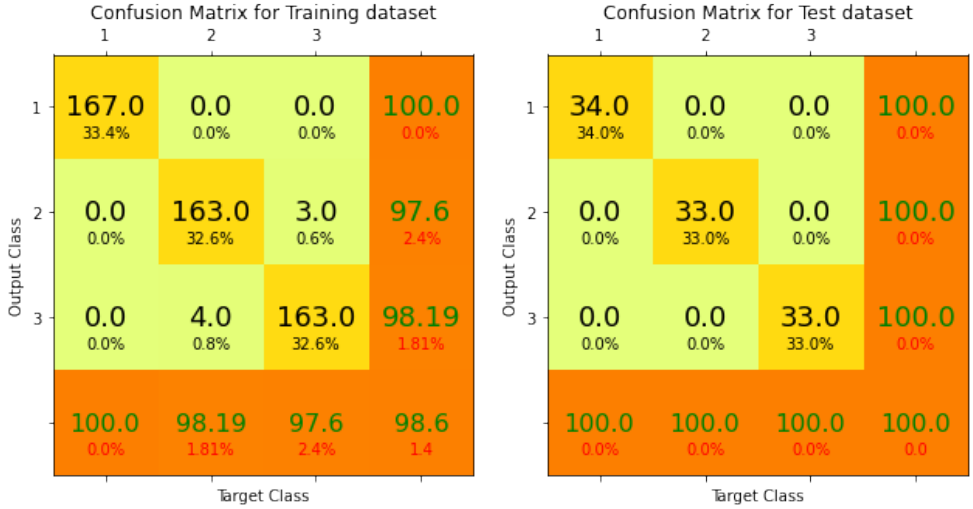
2. **[Naive Bayes Classifier]** In this Question, you are supposed to build Naive Bayes classifiers for the datasets assigned to your team. Train and test datasets for each team can be found here. For each sub-question below, the report should include the following:

   - 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.

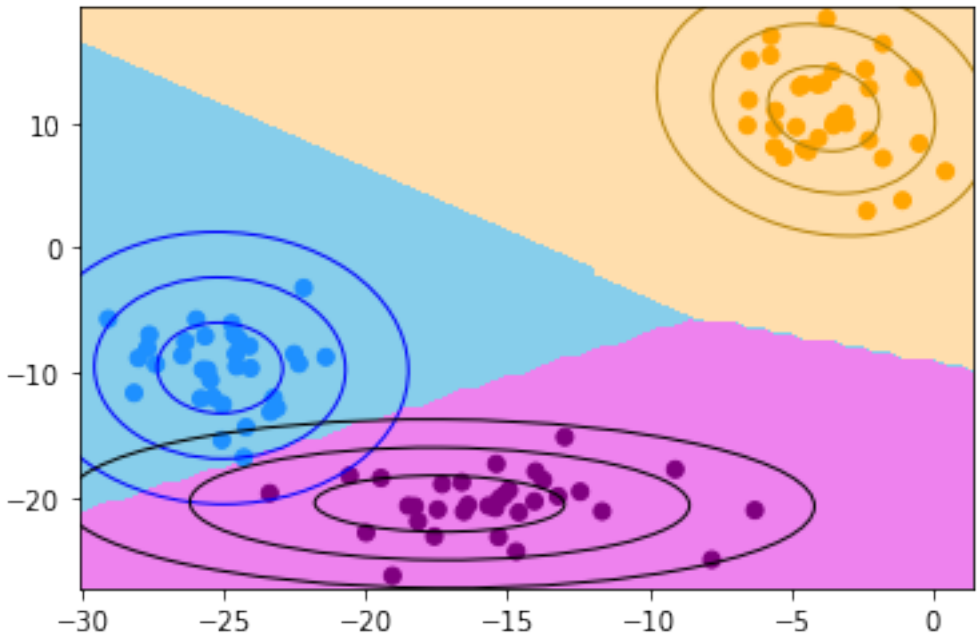
   You can refer to sample plots here and can refer Section 2.6 of "Pattern classification" book by [Duda et al. 2001] for theory.

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

Solution:
Accuracy on training data: 98.6%
Accuracy on test data: 100%



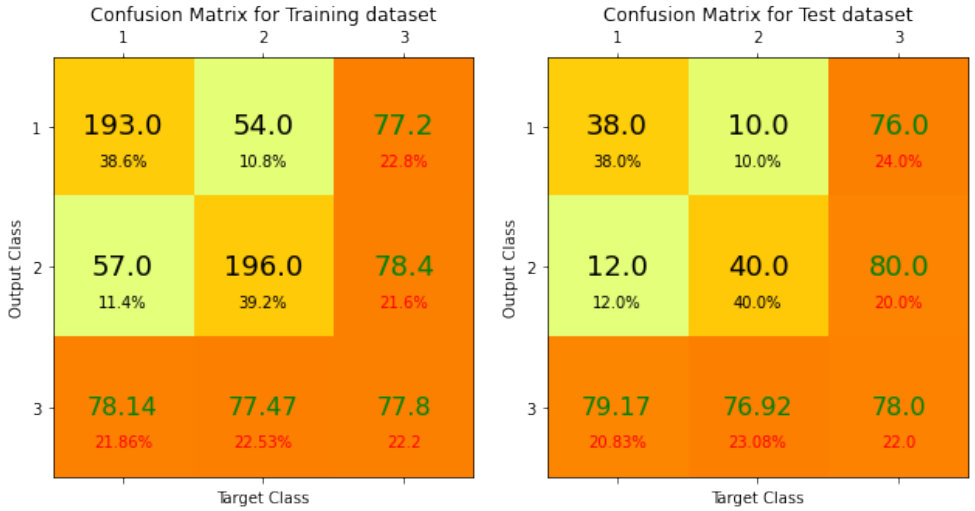
Training Testing



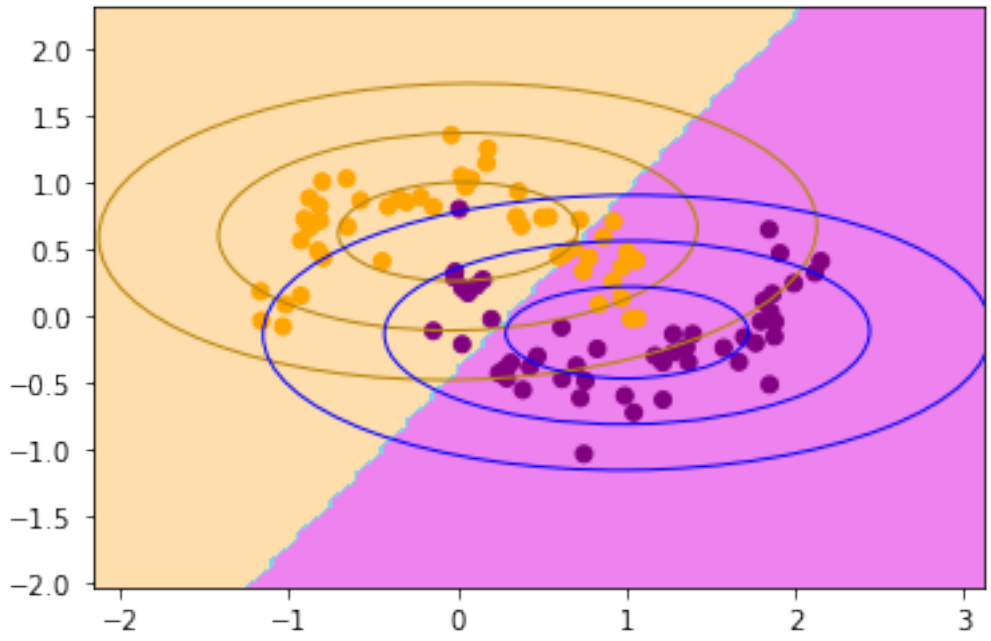
(b) (1 mark) Implement Naive Bayes classifier with covariance = I on dataset3. where, I denotes the identity matrix.

Solution:
Accuracy on training data: 77.8%

Accuracy on test data: 78%

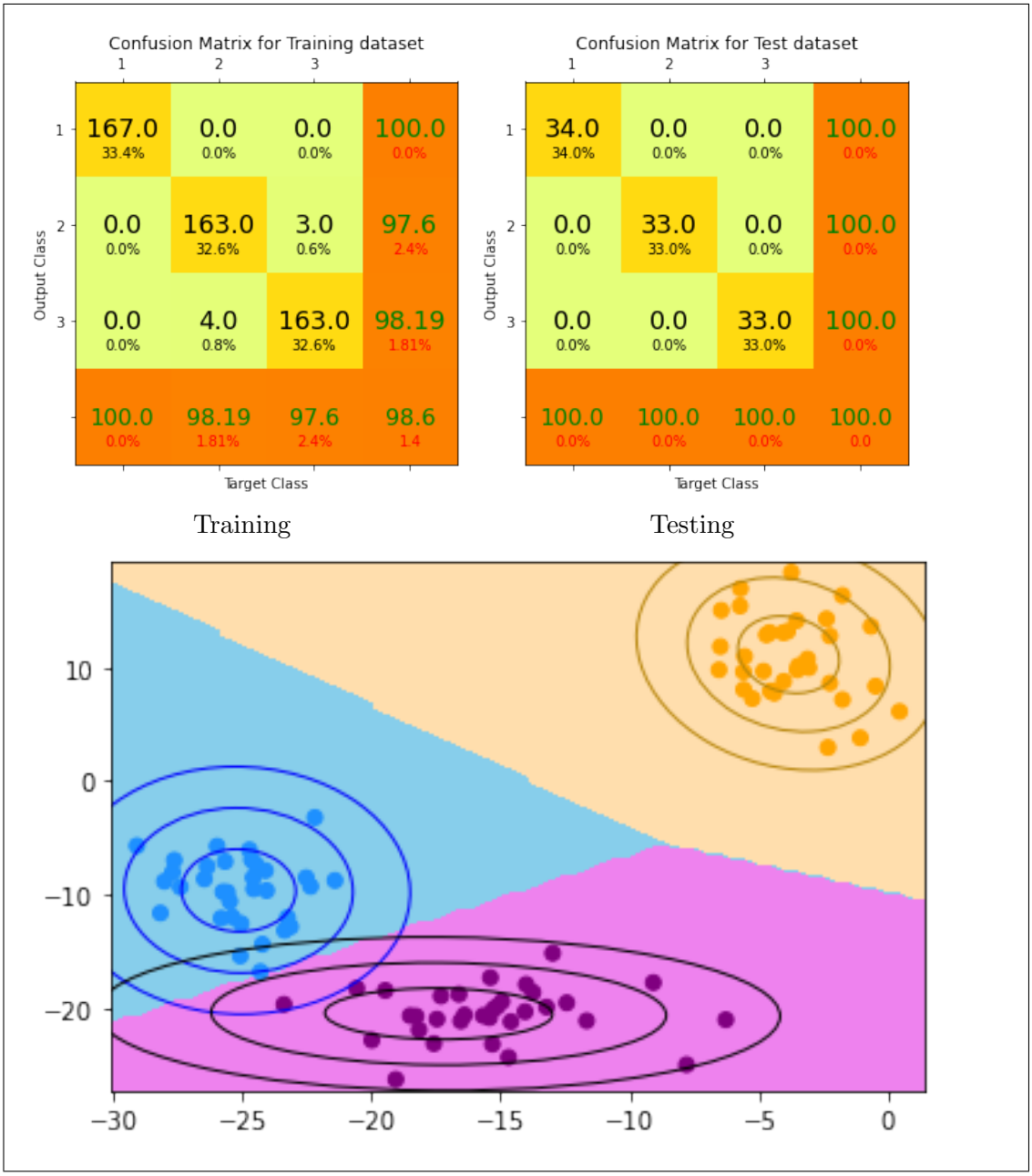


Training Testing



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

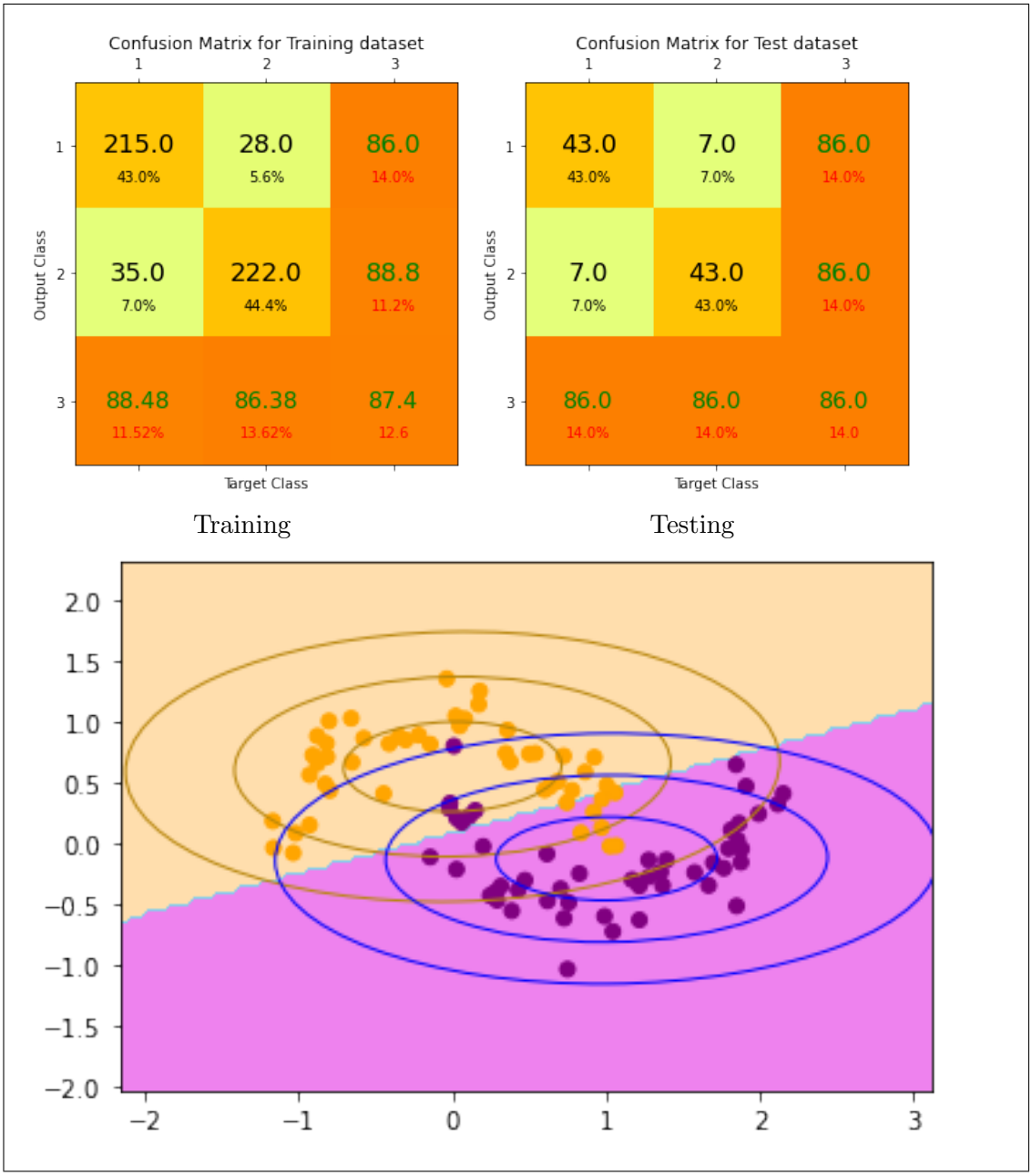
Solution:
Accuracy on training data: 98.6%
Accuracy on test data: 100%



Training Testing

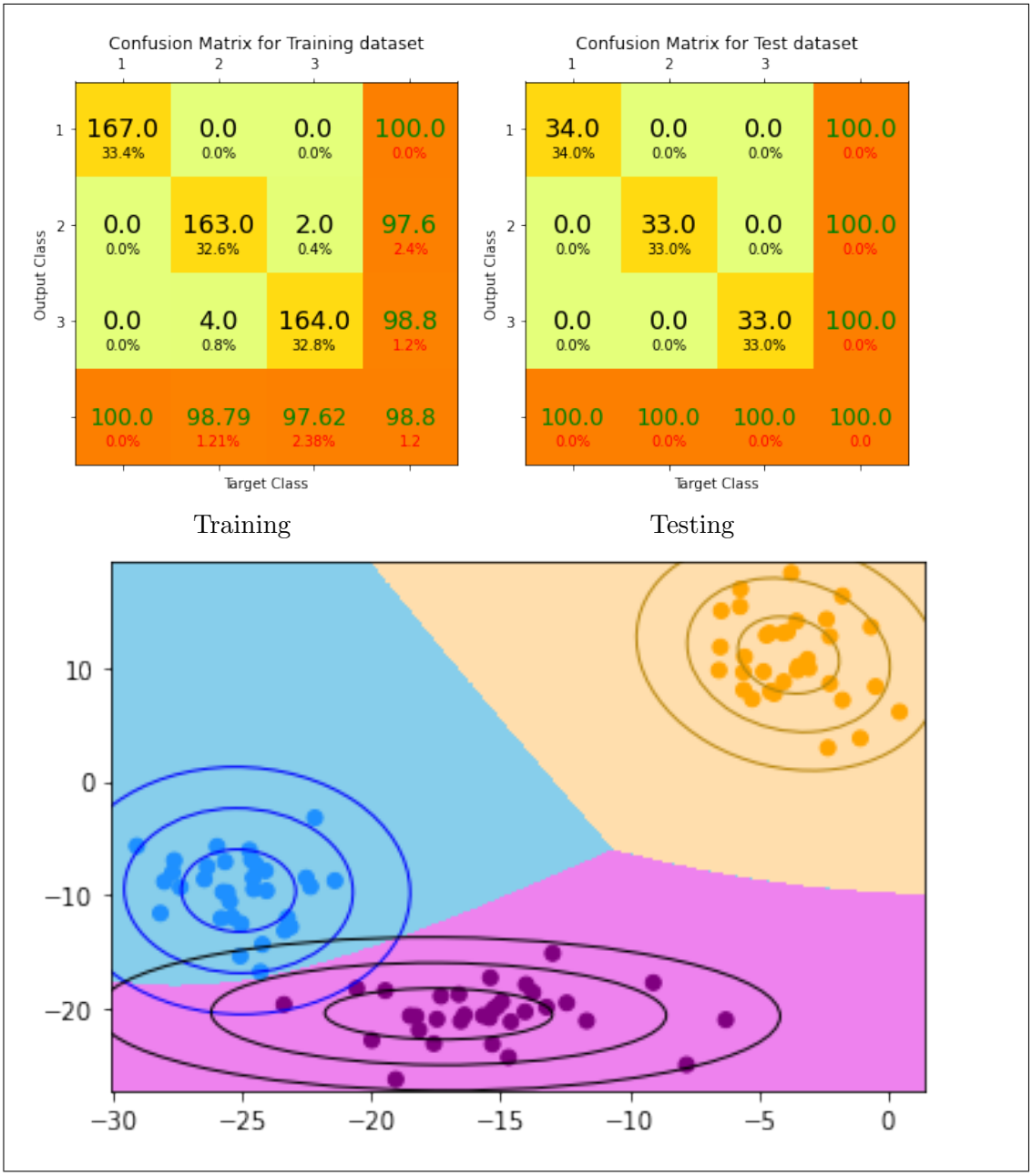
(d) (1 mark) Implement Naive Bayes classifier with covariance same for all classes on dataset3.

Solution:
Accuracy on training data: 87.4%
Accuracy on test data: 86%



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

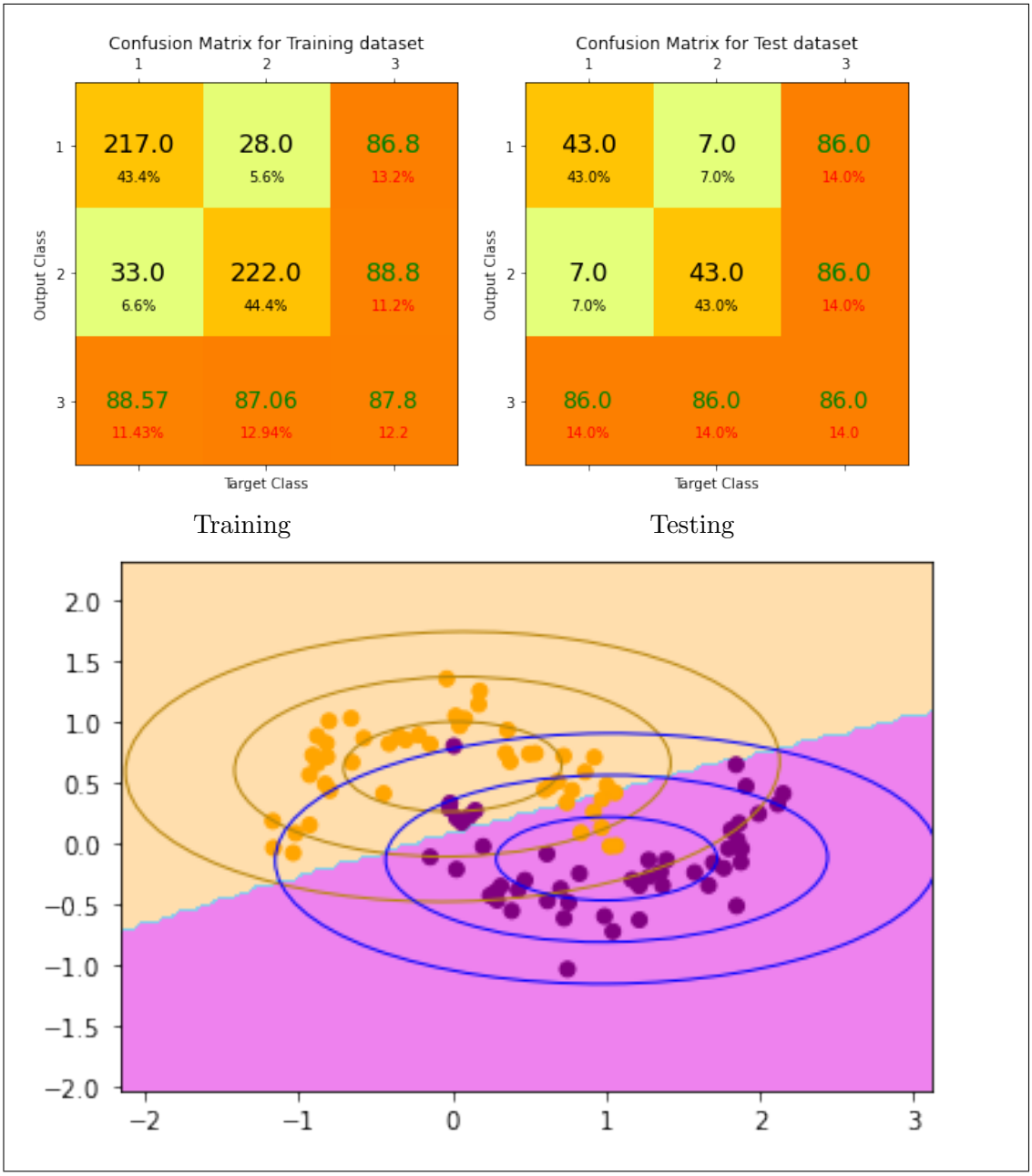
Solution:
Accuracy on training data: 98.8%
Accuracy on test data: 100%



Training Testing

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

Solution:
Accuracy on training data: 87.8%
Accuracy on test data: 86%



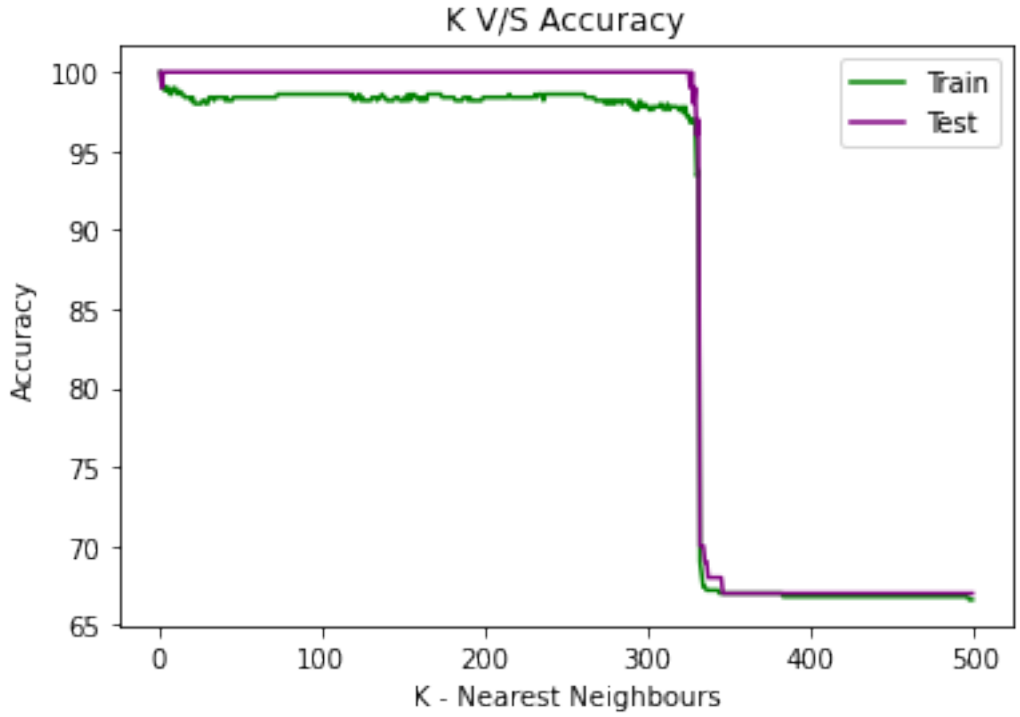
3. **[KNN Classifier]** In this Question, you are supposed to build the k-nearest neighbors classifiers on the datasets assigned to your team. Dataset for each team can be found here. For each sub-question below, the report should include the following:

   - 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.

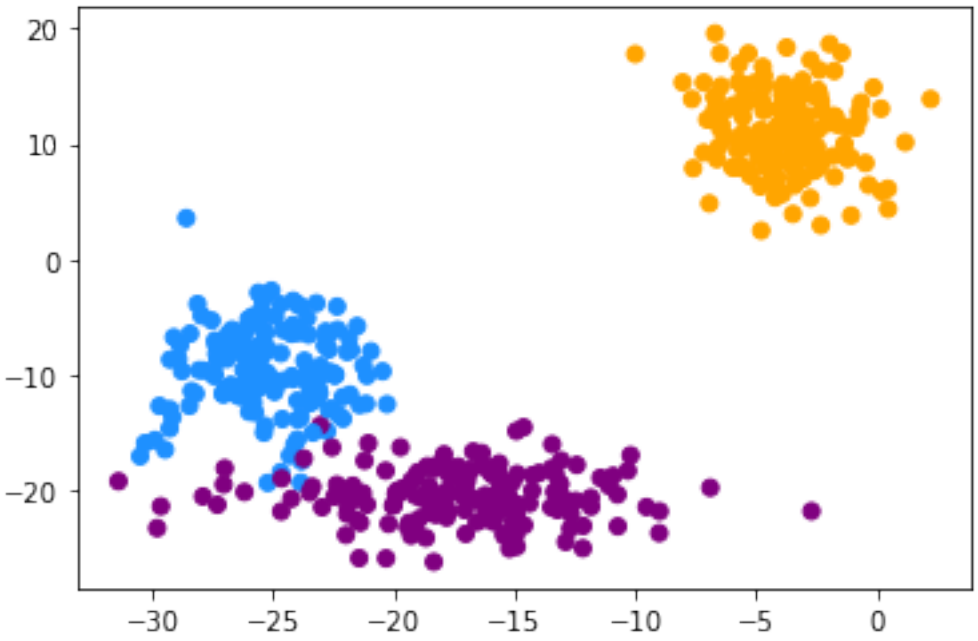
(a) (2 marks) Implement k-nearest neighbors classifier on dataset2.

Solution:

We found the accuracy of training and testing data for various values of K. The plot is given below. The data shows that the accuracy of the training data is below 100 and almost stable for K till 330 and then drops to almost 2/3 of its value, indicating we have two major clusterings, and one of them is split into two. This implies we have three clusters and two of them are very close to each other.



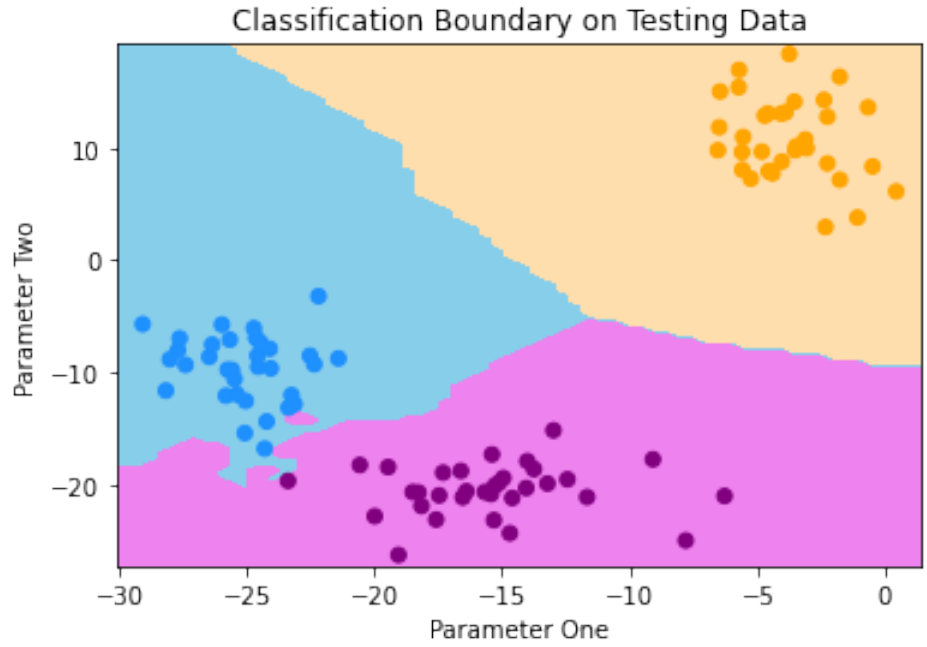
The above analogy can be seen in the scatter plot of the training data.



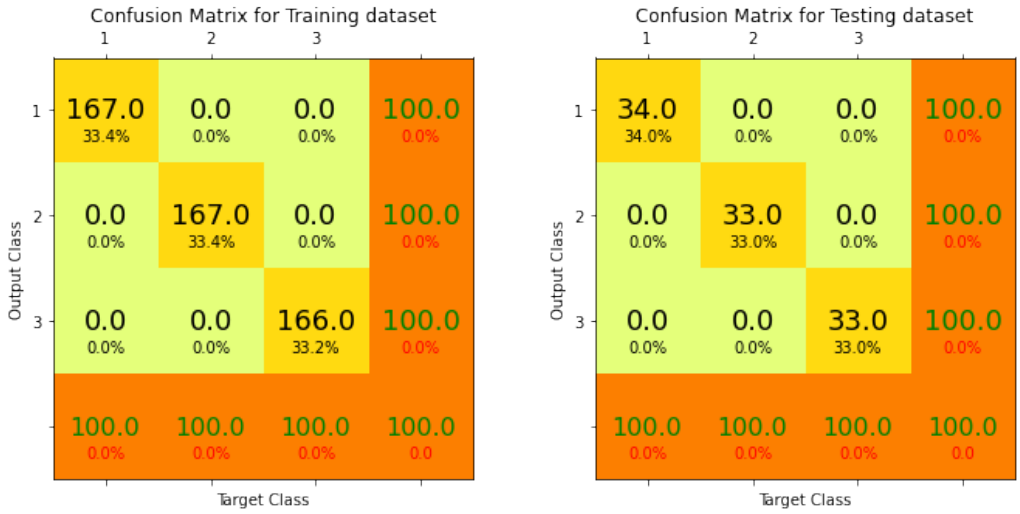
Given below is a screenshot of the accuracy of training and testing data. From this information, we get the best model that fits our training set for K is equal to 1.

```
For K = 1: Train Acc -> 100.0: Test Acc -> 100.0
For K = 2: Train Acc -> 99.0: Test Acc -> 99.0
For K = 3: Train Acc -> 99.2: Test Acc -> 100.0
For K = 4: Train Acc -> 98.8: Test Acc -> 100.0
For K = 5: Train Acc -> 99.0: Test Acc -> 100.0
For K = 6: Train Acc -> 99.0: Test Acc -> 100.0
For K = 7: Train Acc -> 98.6: Test Acc -> 100.0
For K = 8: Train Acc -> 98.6: Test Acc -> 100.0
For K = 9: Train Acc -> 99.0: Test Acc -> 100.0
For K = 10: Train Acc -> 99.0: Test Acc -> 100.0
For K = 11: Train Acc -> 98.8: Test Acc -> 100.0
For K = 12: Train Acc -> 98.6: Test Acc -> 100.0
For K = 13: Train Acc -> 98.8: Test Acc -> 100.0
For K = 14: Train Acc -> 98.6: Test Acc -> 100.0
For K = 15: Train Acc -> 98.6: Test Acc -> 100.0
For K = 16: Train Acc -> 98.4: Test Acc -> 100.0
For K = 17: Train Acc -> 98.4: Test Acc -> 100.0
For K = 18: Train Acc -> 98.4: Test Acc -> 100.0
For K = 19: Train Acc -> 98.4: Test Acc -> 100.0
For K = 20: Train Acc -> 98.2: Test Acc -> 100.0
For K = 21: Train Acc -> 98.4: Test Acc -> 100.0
For K = 22: Train Acc -> 98.0: Test Acc -> 100.0
For K = 23: Train Acc -> 98.0: Test Acc -> 100.0
For K = 24: Train Acc -> 98.0: Test Acc -> 100.0
For K = 25: Train Acc -> 98.0: Test Acc -> 100.0
```

The classification boundary on testing data, with k = 1.

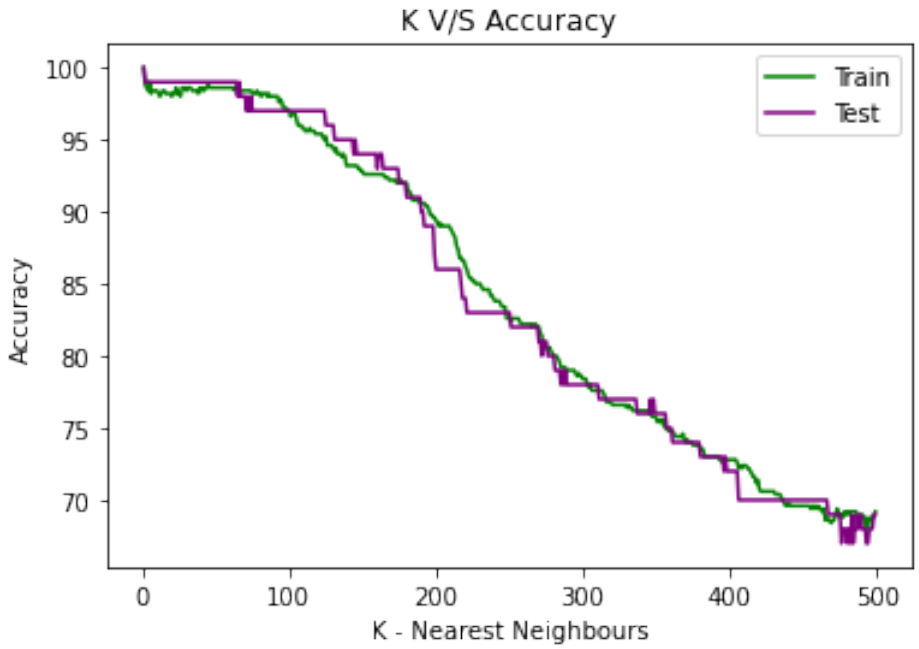


The confusion matrix on training and testing data with k = 1.



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

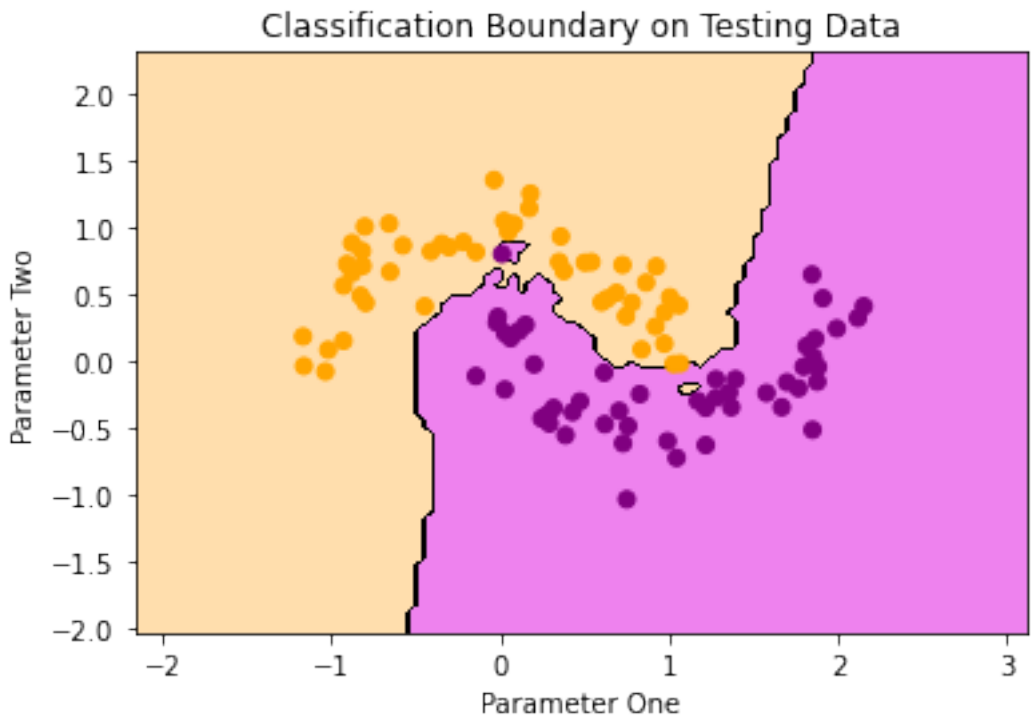
Solution:
We found the accuracy of training and testing data for various values of K. The plot is given below.



Given below is a screenshot of the accuracy of training and testing data. From this information, we get the best model that fits our training set for K is equal to 1.

```
For K = 1:
For K = 1: Train Acc -> 100.0: Test Acc -> 100.0
For K = 2:
For K = 2: Train Acc -> 98.8: Test Acc -> 99.0
For K = 3:
For K = 3: Train Acc -> 98.6: Test Acc -> 99.0
For K = 4:
For K = 4: Train Acc -> 98.4: Test Acc -> 99.0
For K = 5:
For K = 5: Train Acc -> 99.0: Test Acc -> 99.0
For K = 6:
For K = 6: Train Acc -> 98.2: Test Acc -> 99.0
For K = 7:
For K = 7: Train Acc -> 98.4: Test Acc -> 99.0
For K = 8:
For K = 8: Train Acc -> 98.4: Test Acc -> 99.0
For K = 9:
For K = 9: Train Acc -> 98.4: Test Acc -> 99.0
For K = 10:
For K = 10: Train Acc -> 98.4: Test Acc -> 99.0
For K = 11:
For K = 11: Train Acc -> 98.2: Test Acc -> 99.0
For K = 12:
For K = 12: Train Acc -> 98.0: Test Acc -> 99.0
For K = 13:
For K = 13: Train Acc -> 98.2: Test Acc -> 99.0
For K = 14:
For K = 14: Train Acc -> 98.4: Test Acc -> 99.0
For K = 15:
For K = 15: Train Acc -> 98.4: Test Acc -> 99.0
For K = 16:
For K = 16: Train Acc -> 98.2: Test Acc -> 99.0
```

The classification boundary on testing data, with k = 1.



The confusion matrix on training and testing data with k = 1.