CS5691: Pattern Recognition and Machine Learning
Assignment #1

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

Teammate 1: Aditya Viraj Rao Ponugoti **Roll number:** CS20B005
Teammate 2: Neel Prabhanjan Rachamalla **Roll number:** CS20B056

- Please refer to the **Additional Resources** tab on the Course webpage for basic programming instructions.
- This assignment has to be completed in teams of 2. Collaborations outside the team are strictly prohibited.
- Any kind of plagiarism will be dealt with severely. These include copying text or code from any online sources. These will lead to disciplinary actions according to institute guidelines. Acknowledge any and every resource used.
- Be precise with your explanations. Unnecessary verbosity will be penalized.
- Check the Moodle discussion forums regularly for updates regarding the assignment.
- You should submit a zip file titled **'rollnumber1_rollnumber2.zip'** on Moodle where rollnumber1 and rollnumber2 are your institute roll numbers. Your assignment will **NOT** be graded if it does not contain all of the following:

    1. Type your solutions in the provided LaTeX template file and title this file as **'Report.pdf'**. **State your respective contributions at the beginning of the report clearly.** Also, embed the result figures in your LaTeX solutions.
    2. Clearly name your source code for all the programs in **individual Google Colab files**. Please submit your code only as Google Colab file (.ipynb format). Also, embed the result figures in your Colab code files.

- We highly recommend using `Python 3.6+` and standard libraries like `NumPy, Matplotlib, Pandas, Seaborn`. Please use `Python 3.6+` as the only standard programming language to code your assignments. Please note: the TAs will only be able to assist you with doubts related to Python.
- You are expected to code all algorithms from scratch. **You cannot use standard inbuilt libraries for algorithms**. Using them will result in a straight zero on coding questions, `import` wisely!
- We have provided different training and testing sets for each team. f.e. train_1 and test_1 denotes training and testing set assigned to team id 1. Use sets assigned to your team only for all questions, reporting results using sets assigned to different team will result in straight zero marks.
- Any graph that you plot is unacceptable for grading unless it labels the x-axis and y-axis clearly.

- **Please start early and clear all doubts ASAP.**
- Please note that the TAs will **only** clarify doubts regarding problem statements. The TAs won't discuss any prospective solution or verify your solution or give hints.
- Please refer to the CS5691 PRML course handout for the late penalty instruction guidelines.
- Post your doubt only on Moodle so everyone is on the same page.

---

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 -

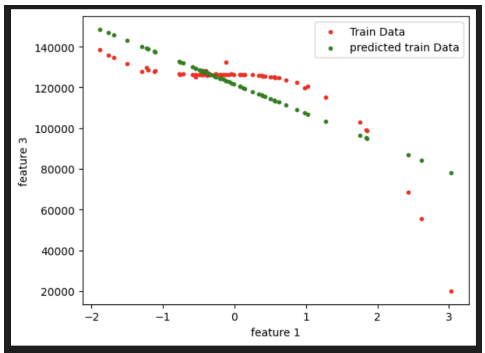
   - Plot different figures for train and validation data and for each figure plot curve of obtained function on data points for various degree term of the polynomial.(refer to fig. 1.4, Pattern Recognition and Machine Learning, by Christopher M. Bishop).
   - Plot the curve for Mean Square Error(MSE) Vs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

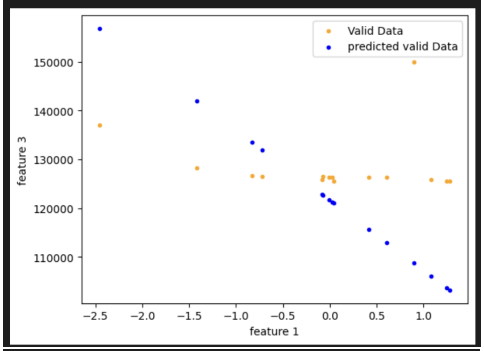
   Please find the link to the google colab file here.

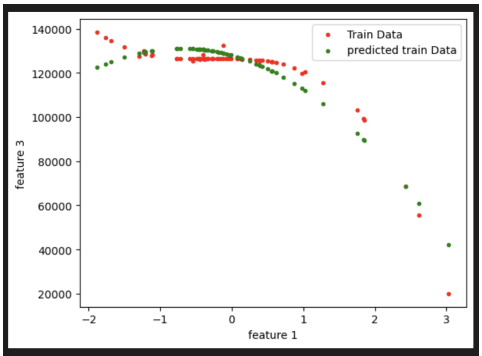
   Upon normalising the data for feature1 and feature2, **both columns turn out to be the same**. So we have dropped the column containing feature2. And now, assuming it is 2D data, we have proceeded further.
   We have worked with the close-form solution to obtain beta, which involves the concept of the pseudo inverse.
   To compute each degree polynomial, I have appended columns to the existing matrix **X** to make **X'**

The following plots have red as the train data and green as the predicted train data. Similarly, the data in orange is the validation data, and blue is the predictions for validation data.

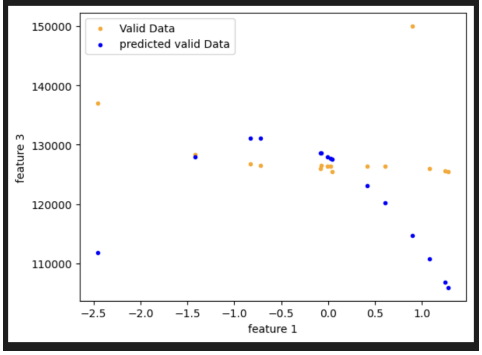


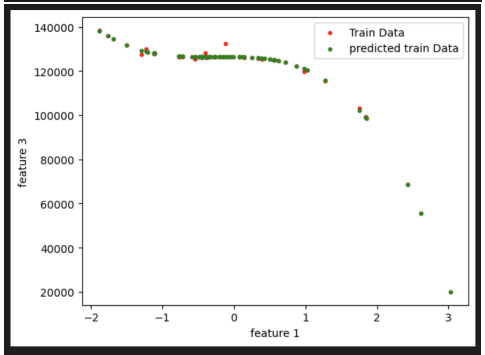
degree: 1



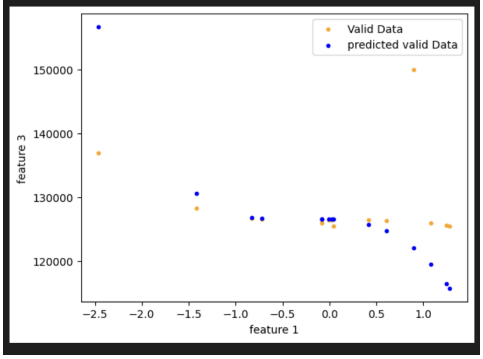


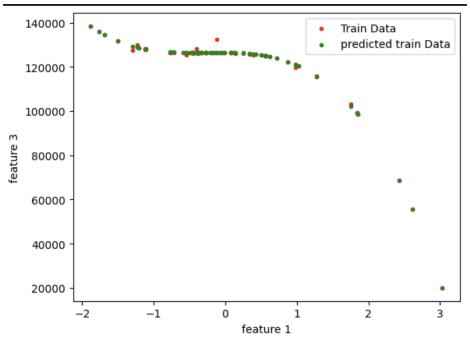
degree: 2



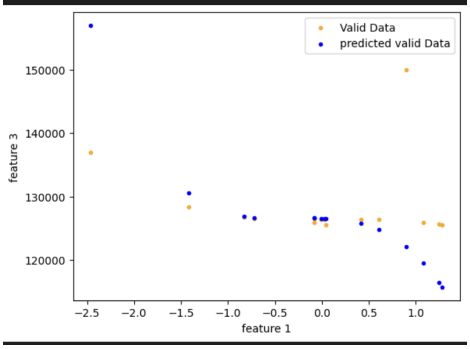


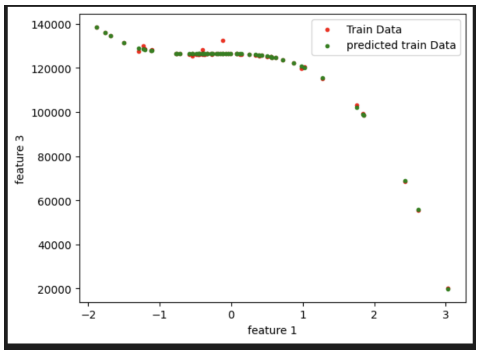
degree: 3

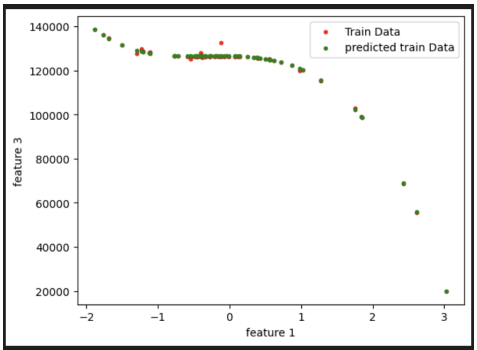




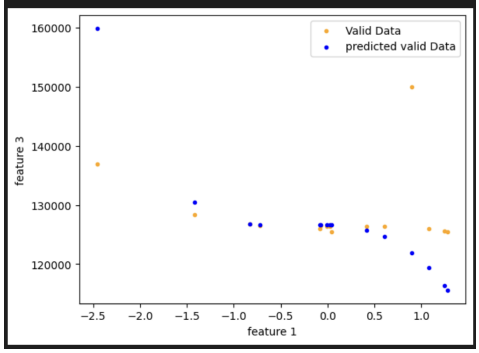
degree: 4



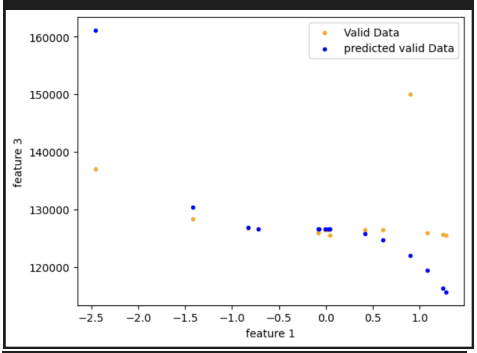


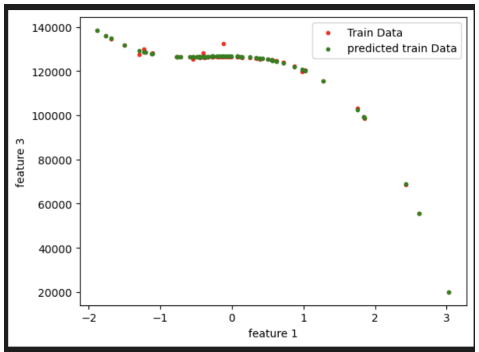


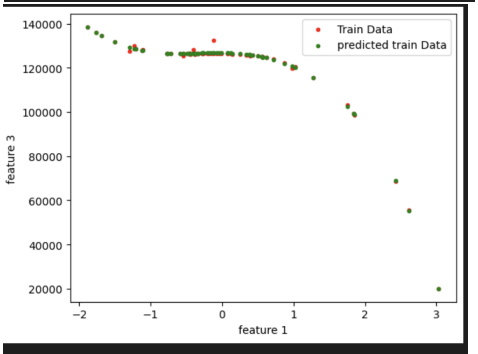
degree: 5



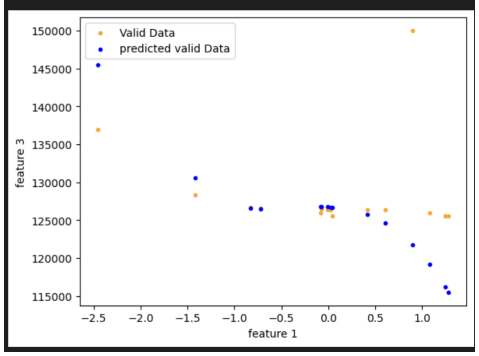
degree: 6



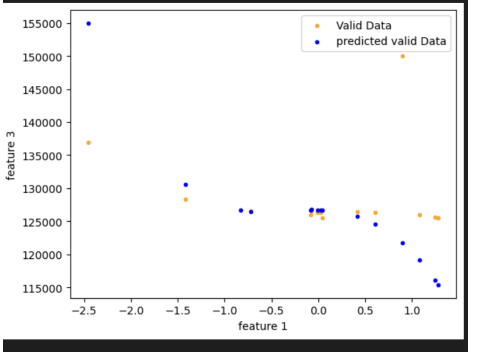


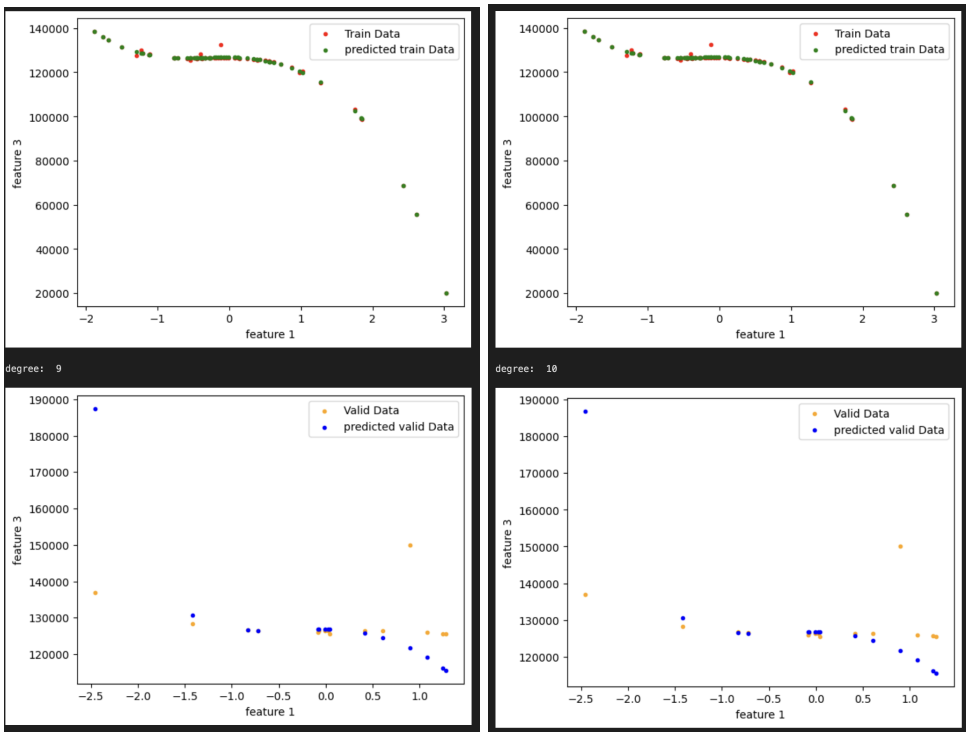


degree: 7



degree: 8

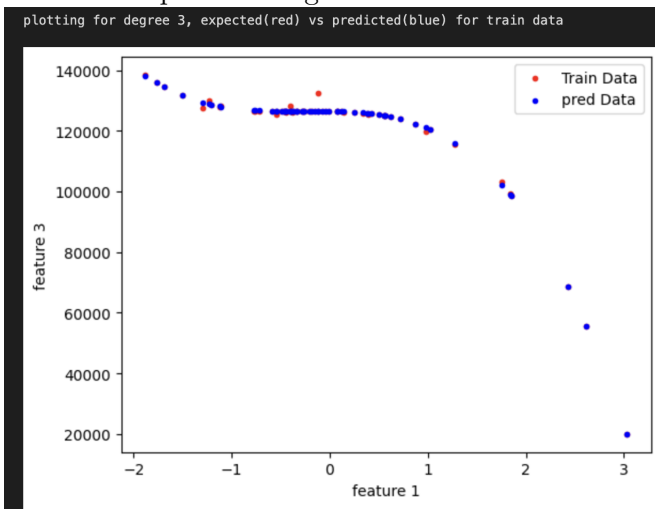


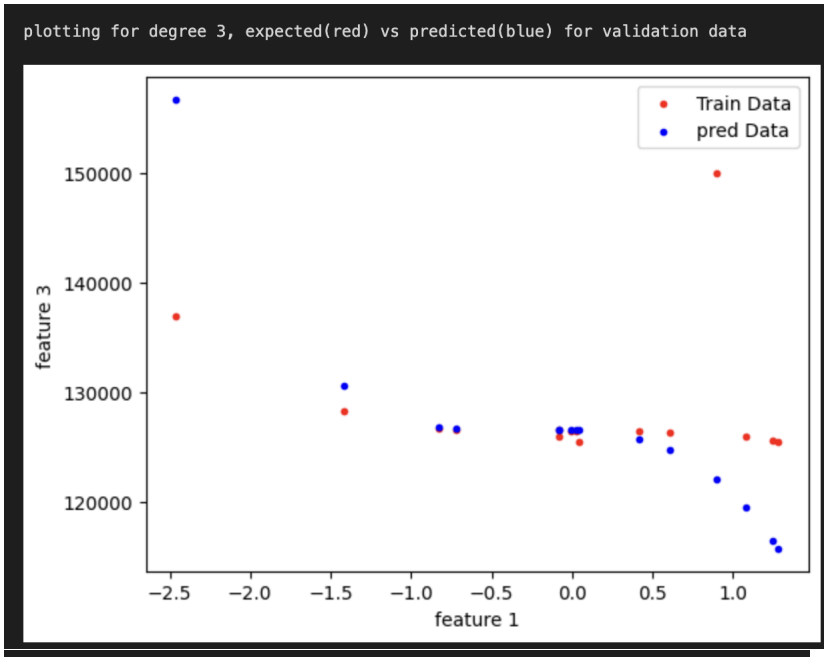


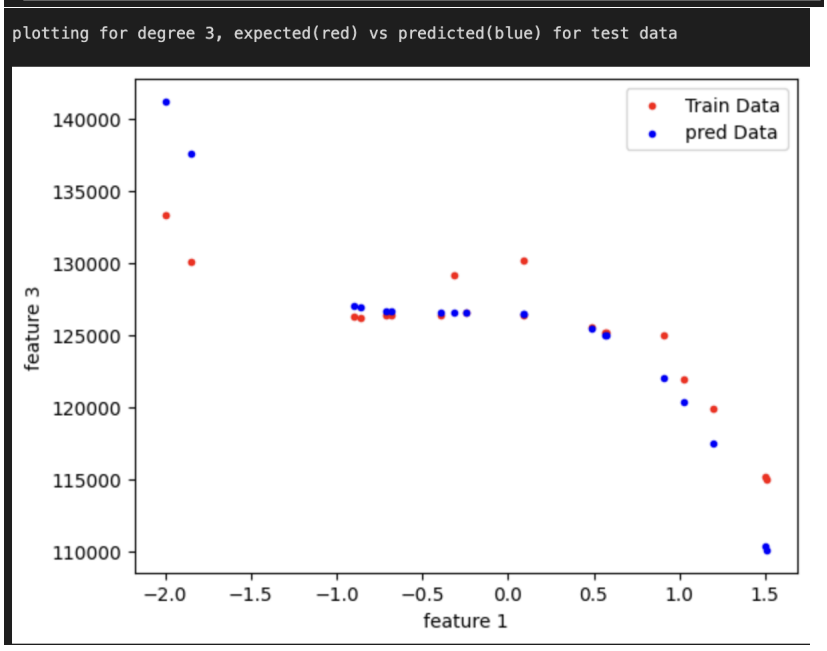
The errors for degree = 3 as the best-fit polynomial to also avoid over-fitting of data.

```
best degree:  3
MSE train error:  765856.867249531 at degree 3
MSE valid error:  93290327.56397326 at degree 3
MSE test error:  10766824.156454507 at degree 3
```

The scatter plots for degree =3 fit is:-







As the data in CSV is already shuffled, we can just take the first 20 percent as validation data and the rest as train data.

On observing the values of MSE for validation data, we have **degree = 5** as the best-fit polynomial.

(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 -

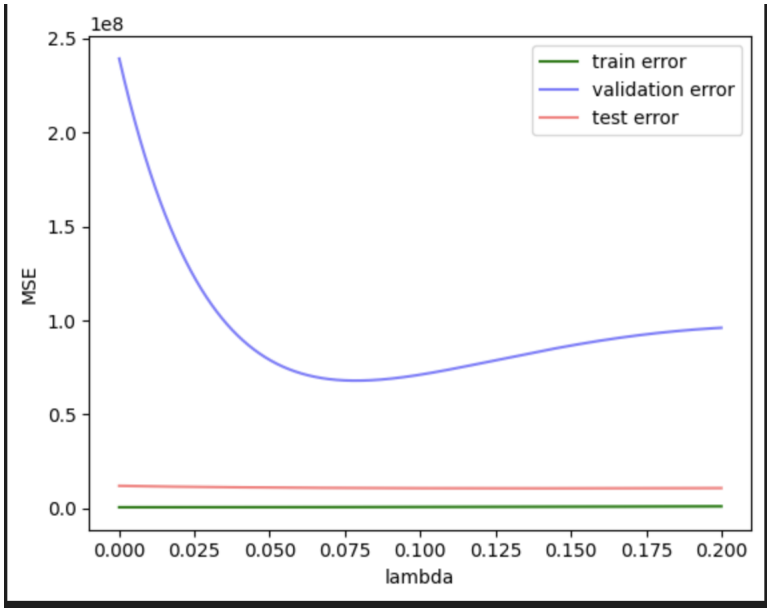
- Choose the degree from part (a), where the model overfits and try to control it using the regularization technique (Ridge regression).
- Use various choices of lambda(λ) and plot MSE test Vs lambda(λ).
  We used the closed form solution where extra lambda*I is in the expression to calculate Beta.

- 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:
Ridge Regression :-

Please find the link to the google colab file here.
From part (a) we got the least MSE at degree=3, but for overfitting the data we take degree = 9. So this over-fitting can be controlled by penalising large beta and shrinkage methods.

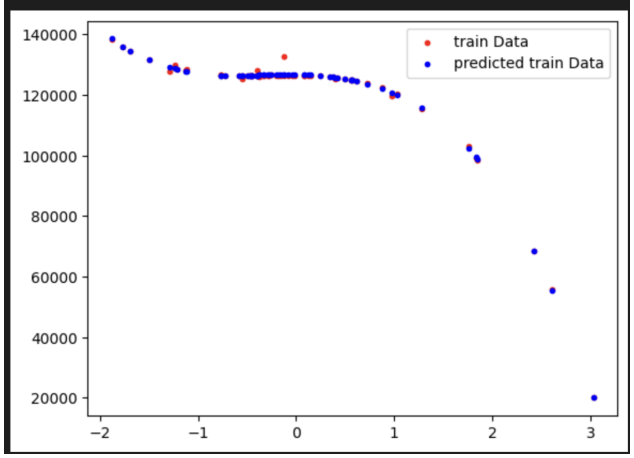
Plot of MSE vs lambda with train error in green, validation error in blue and test error in red.

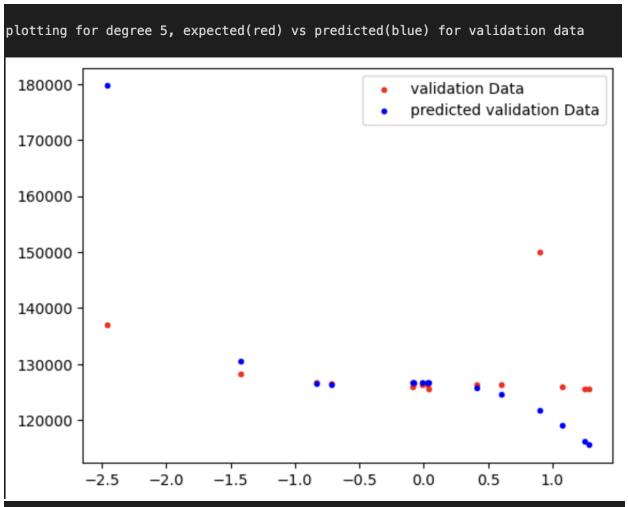


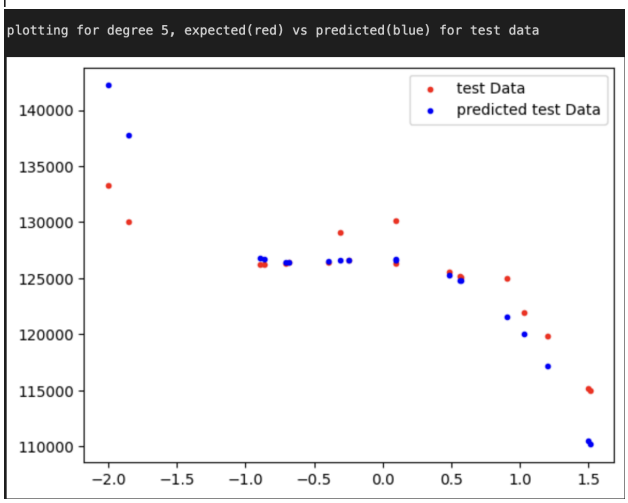

```
min valid error of  68008143.01857471  at  [0.0786]
min test error of  10799172.288745562  at  [0.1389]
min train error of  727844.44571884  at  [0.0001]
```

Above are the MSE values for lambda = 0.0786, which can be taken as the best-fit lambda from the plot.
Below are the scatter plots for lambda=0.0786 with predicted and expected data.

```
plotting for degree 5, expected(red) vs predicted(blue) for train data
```







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 to 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:
Naive Bayes Classifier Using Multivariate Gaussian Distribution :-

The D-dimensional Gaussian distribution is parameterised by a mean vector, $\mu = (\mu 1, ..., \mu D)^T$ and a DXD covariance matrix, $\Sigma = \sigma(ij)$. The probability density function is given by:

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$

where $|\Sigma|$ is the determinant of the covariance matrix Σ.

This conditional probability density function is formulated for each class by constructing the mean and covariance matrices for the data points in the respective classes. After which , we compute probabilities for each test point to fall in that class and assign that class for which it is most probable to fall in.

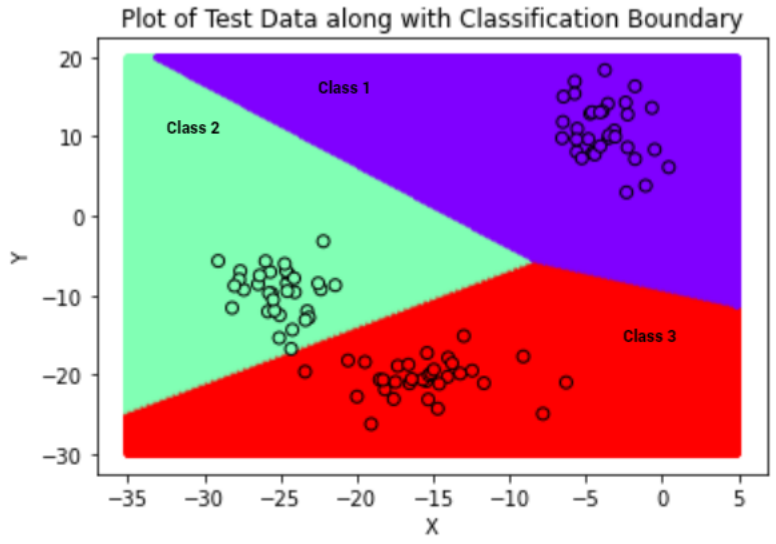
Naive Bayes Classifier with Covariance = I on dataset2 where I is the identity matrix

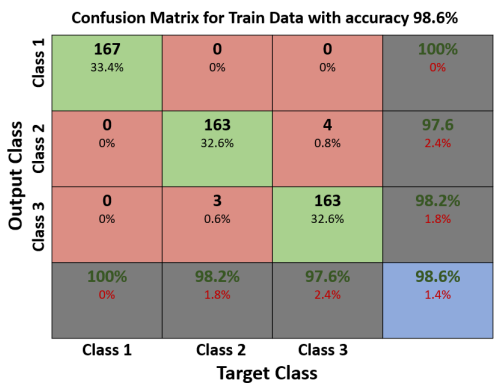
Here, we consider the covariance matrix to be the same for all the classes and also as the Identity matrix.

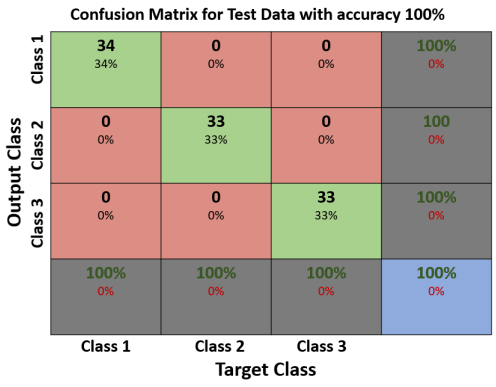
Please find the link to the google colab file here.

Results :-

Accuracy on train data is 98.6%.
Accuracy on test data is 100%.







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

Solution:

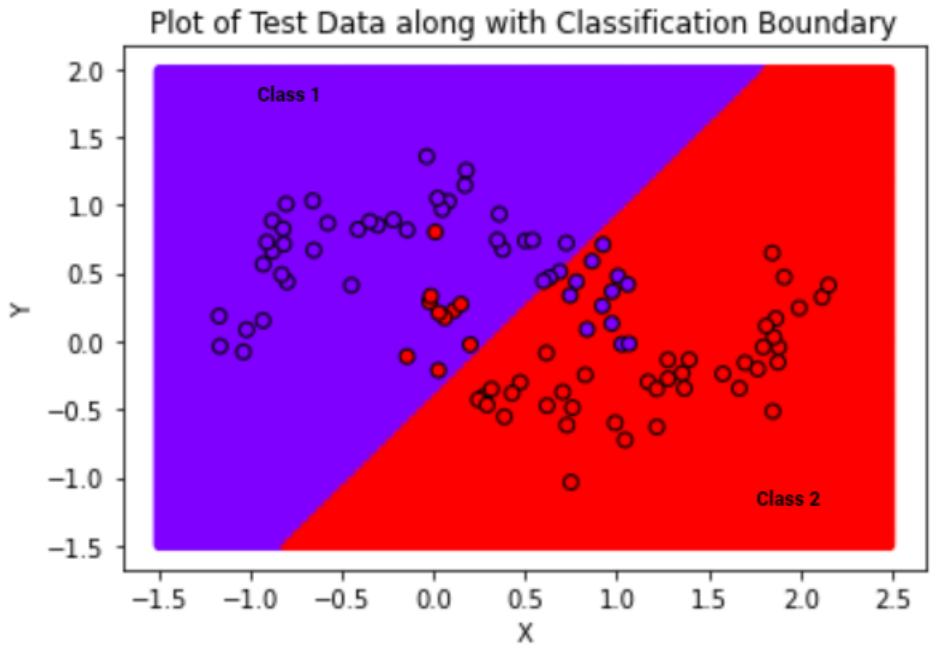
Naive Bayes Classifier with Covariance = I on dataset3 where I is the identity matrix

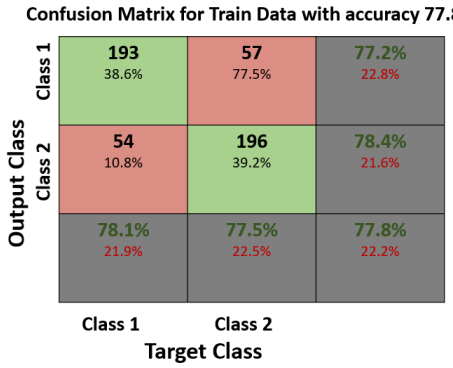
Here , we consider the covariance matrix to be same for all the classes and also as Identity matrix.

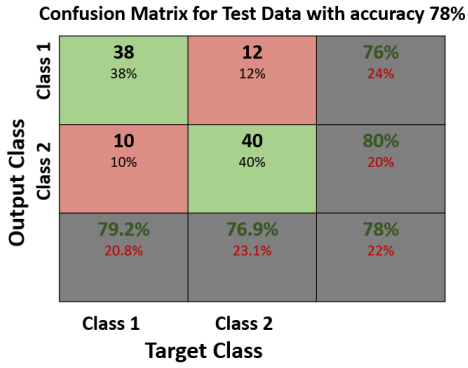
Please find the link to the google colab file here.

Results :-

Accuracy on train data is 77.8%.
Accuracy on test data is 78%.







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

Solution:

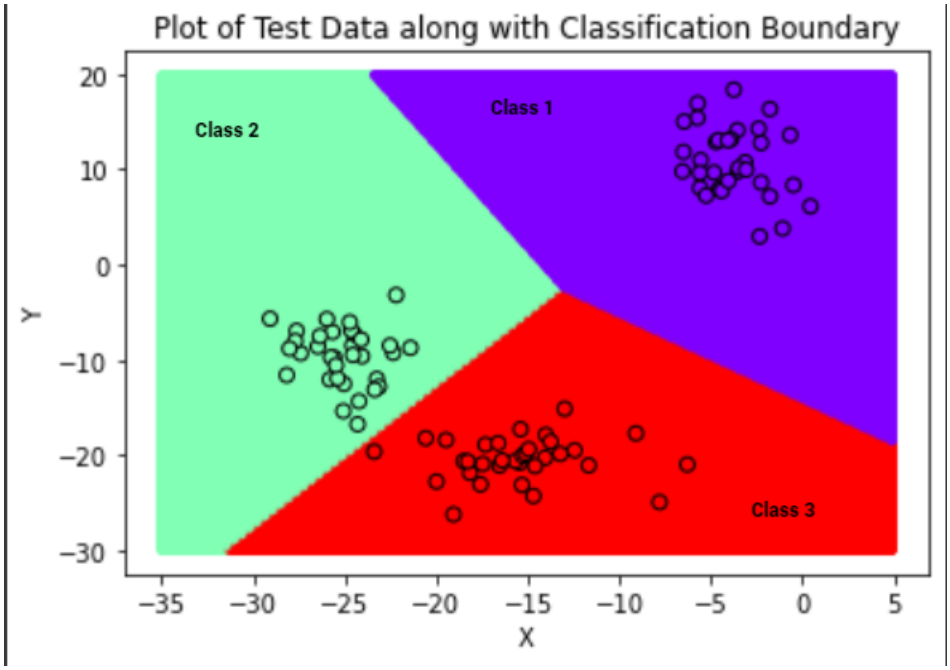
Naive Bayes Classifier with Covariance same for all classes on dataset2

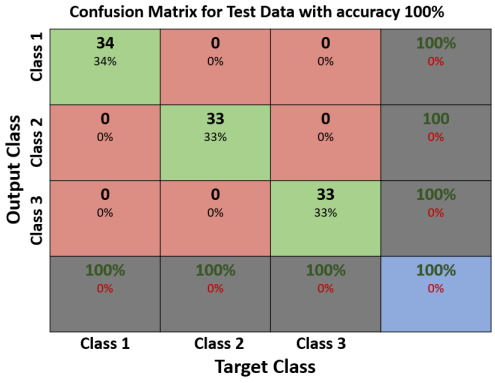
Here , we consider the covariance matrix to be same for all the classes and compute it considering all the data points of the train data.

Please find the link to the google colab file here.

Results :-

Accuracy on train data is 97.4%.
Accuracy on test data is 100%.





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

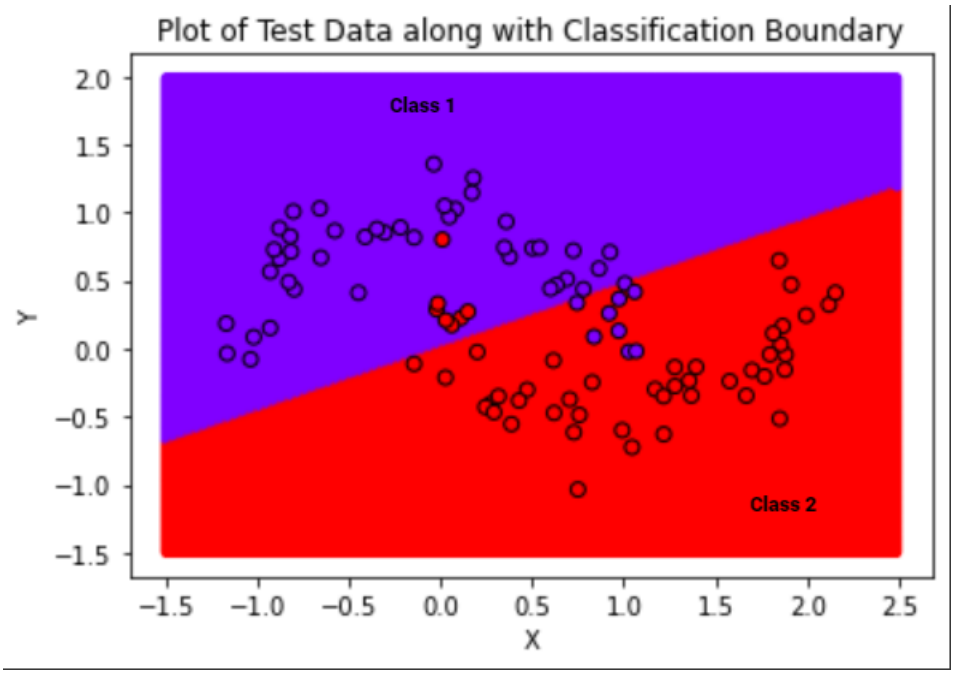
Solution:
Naive Bayes Classifier with Covariance same for all classes on dataset3

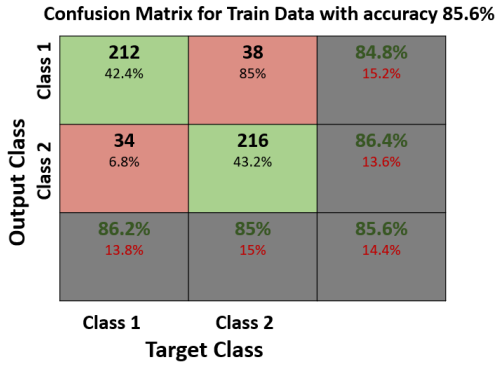
Here , we consider the covariance matrix to be same for all the classes and compute it considering all the data points of the train data.

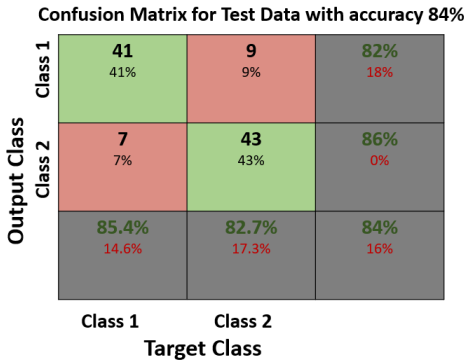
Please find the link to the google colab file here.

Results :-

Accuracy on train data is 85.6%.
Accuracy on test data is 84%.







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

Solution:
Naive Bayes Classifier with Covariance different for different classes on dataset2

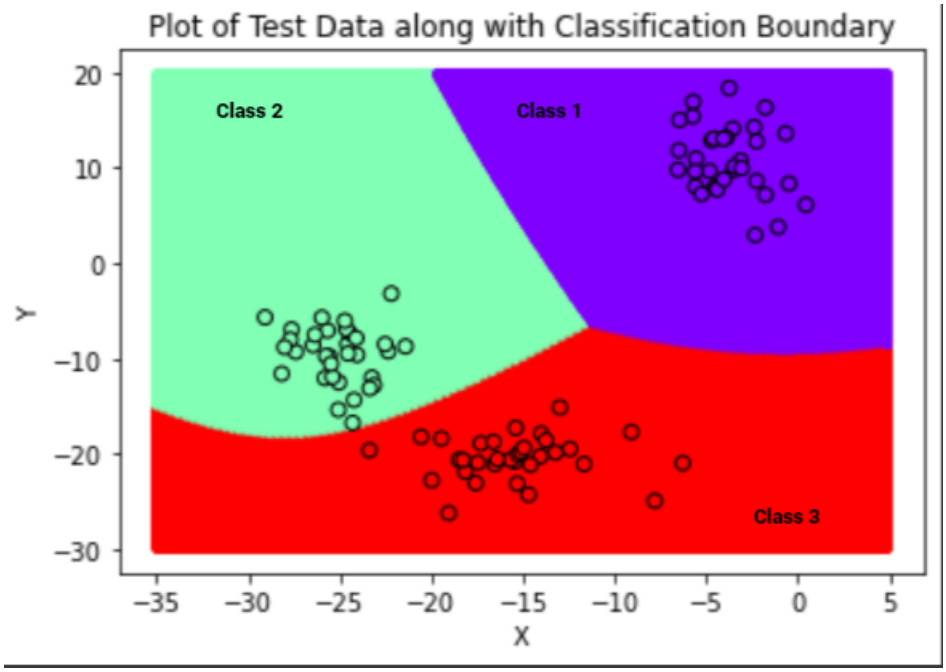
Here , we consider the covariance matrix to be different for different classes and

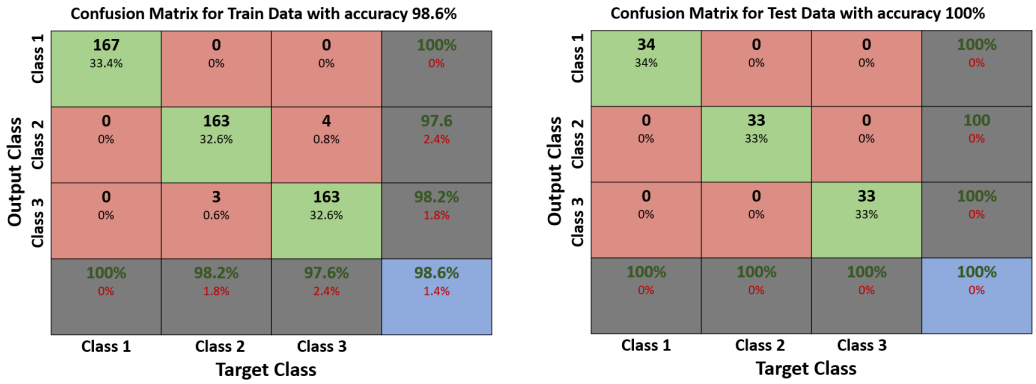
compute it considering corresponding classes' data points of the train data.

Please find the link to the google colab file here.

Results :-

Accuracy on train data is 98.6%.
Accuracy on test data is 100%.





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

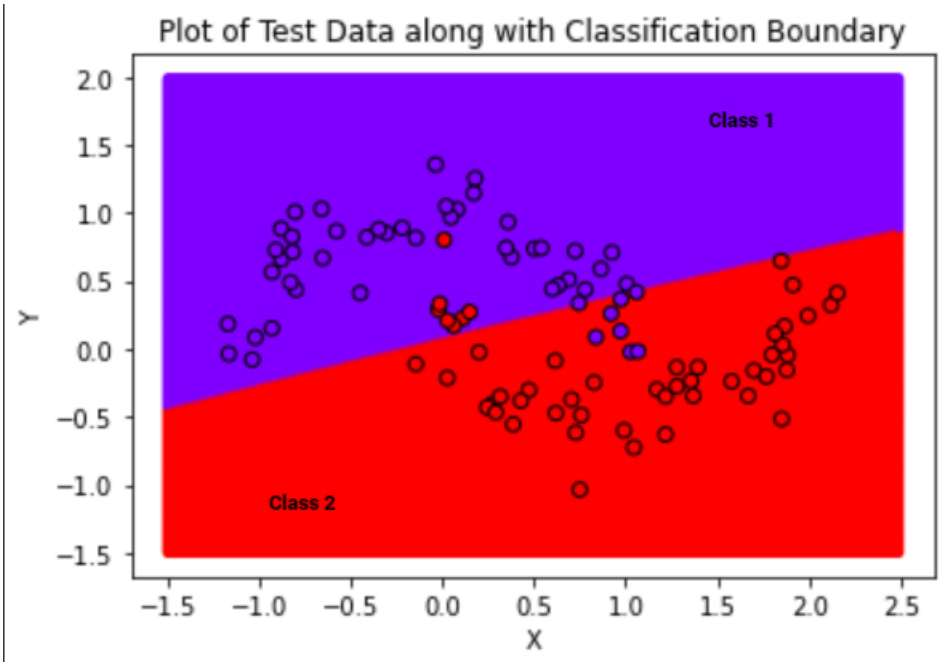
Solution:
Naive Bayes Classifier with Covariance different for different classes on dataset3

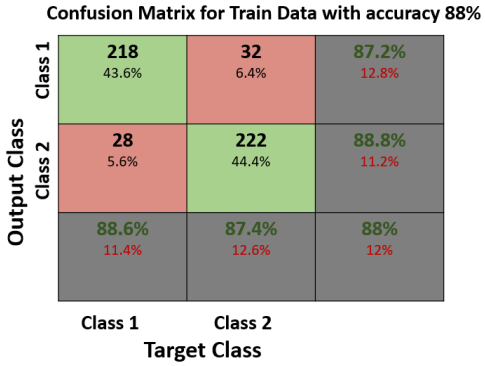
Here , we consider the covariance matrix to be different for different classes and compute it considering corresponding classes' data points of the train data.

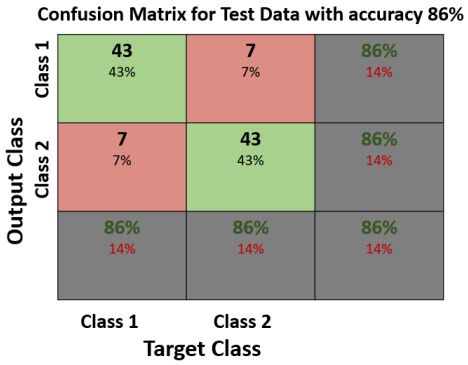
Please find the link to the google colab file here.

Results :-

Accuracy on train data is 88%.
Accuracy on test data is 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:
K-Nearest Neighbours Classifier :-

- Specifically, the k-nearest neighbor fit for $\hat{Y}$ is defined as follows:

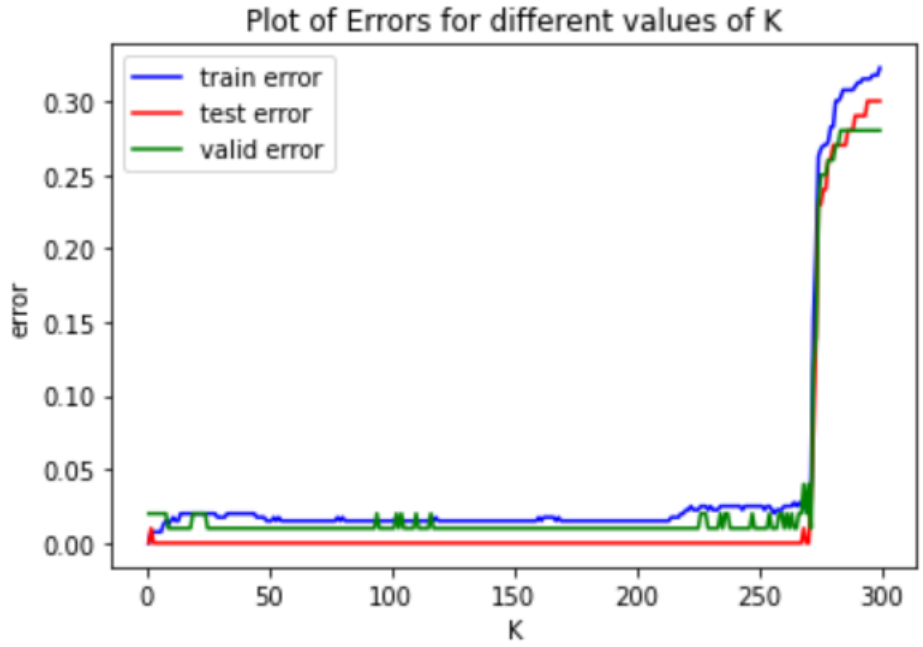
$$\hat{Y}(x) = \frac{1}{K}\sum_{x_i \epsilon N_k(x)} y_i$$

where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training sample. Closeness implies a metric, which for the moment we assume is Euclidean distance.

KNN Classifier on dataset2

Please find the link to the google colab file here.

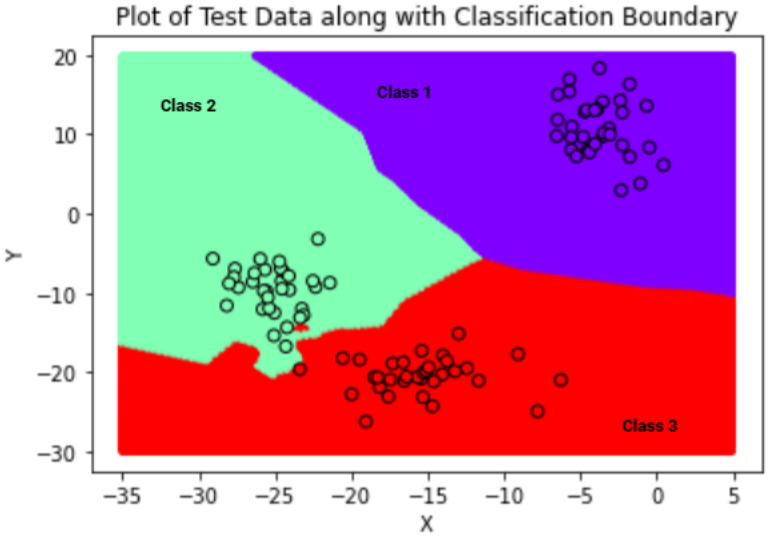
Results :-

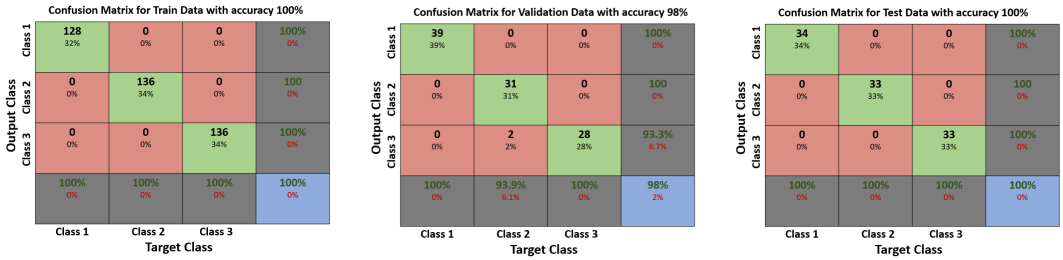


Here , from the plot it is clearly evident that we get the optimum K(least error) in the range 1 to 200 for both train and test data. So , we consider the least among them.
Therefore the Optimum K for both train and test data is K=1.

The results considering K=1 follow :-

Accuracy on train data is 99.25%.
Accuracy on test data is 100%.





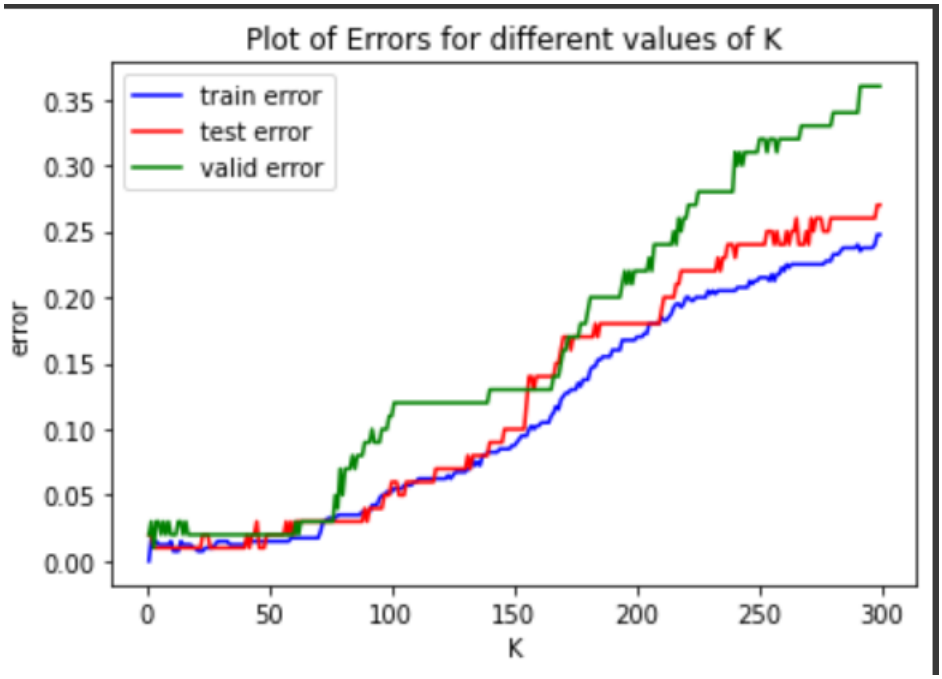
(b) (2 marks) Implement k-nearest neighbors classifier on dataset3.

Solution:

KNN Classifier on dataset3

Please find the link to the google colab file here.

Results :-



Here , from the plot it is clearly evident that we get the optimum K(least error) as 1 for train data and in the range 3 to 25(or so) for test data. So , we consider the least among the range for test data.
Therefore , the Optimum K for train data is K=1 and test data is K=3.

The results considering K=3 follow :-

Accuracy on train data is 98.75%.
Accuracy on test data is 99%.