Pattern Recognition and Machine Learning

(Winter 2022)

Assignment 4: Bayes Classification

Report

Question - 1:

Task - 1: Perform pre-processing and visualization of the dataset. Split the data into train and test sets. Also identify the useful columns and drop the unnecessary ones.

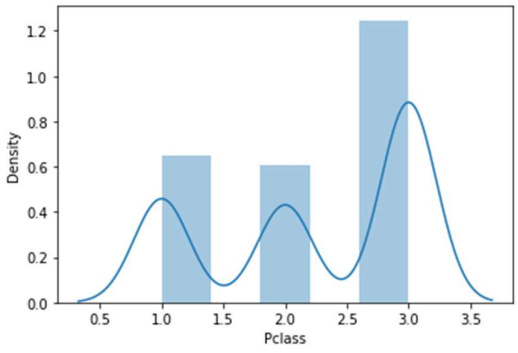
I first took the input of the given Titanic dataset. Then I performed the data pre-processing on the given dataset.

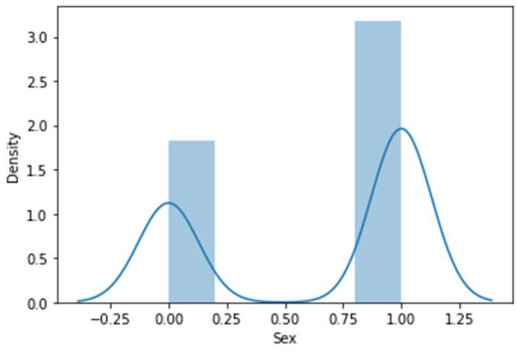
I first identified the useful columns and dropped the unnecessary ones. The reason for dropping the columns "PassengerId", "Name", and "Ticket" was that they are not the relevant features for determining the target (Survival). The reason of dropping the column "Cabin" was that it contains a lot of missing/Nan values.

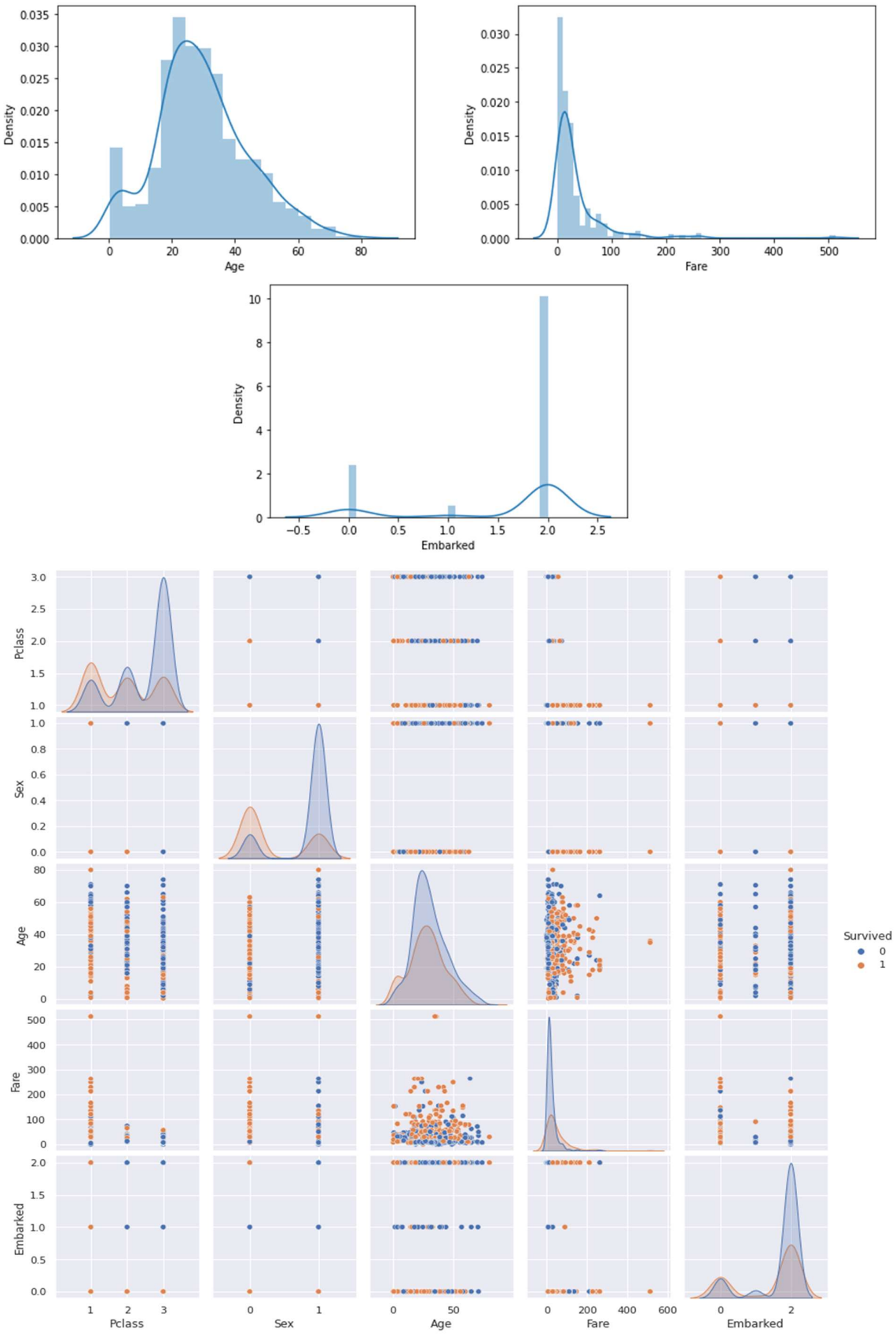
Then, after dropping the unnecessary columns. I checked if there are any null values present in the dataset. It appears that there are some null values present in the dataset. So, I just dropped the rows containing that missing/Nan values.

Then, I performed categorical encoding wherever applicable in the data. The categorical data was present in the columns "Sex" and "Embarked" of the dataset.

Then, I did the visualization of the dataset. I plotted the graph of density v/s features of the dataset.







Finally, I performed the splitting of the dataset into training sets and testing sets.

<u>Task - 2:</u> Identify the best possible variant of naive bayes classifier for the given dataset. Justify your reason for the same.

To identify the best possible variant of the naïve bayes classifier. I first looked upon the possible naïve bayes classifier that are Gaussian Naïve Bayes Classifier, Multinomial Naïve Bayes Classifier, and Bernoulli Naïve Bayes Classifier. So, on observing the dataset, datatypes of the features present in the dataset and the visualization of the plots plotted in Task- 1. I observed that the features "Age", and "Fare" appears to follow gaussian (normal) distribution. Hence, I decided that the best possible variant of naïve bayes classifier for this dataset should be the Gaussian Naïve Bayes Classifier. As the Gaussian naïve bayes classifier is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

<u>Task - 3:</u> Implement the identified variant of Naive Bayes Classifier from scratch. [you may not use any 3rd-party library's classifier function, such as scikit-learn. However, Built-in functions, such as train/test split, can be used for supplementary tasks.].

I implemented the Gaussian Naïve Bayes Classifier from scratch. I first calculated the mean and variance of each of the features for the two classes separately. Then, I prepared lists that contains mean and variance of each of the features for the two classes and appended these lists in two new lists i.e., one for survived class and another for not survived class.

Then after this I wrote the function named probability that helps to calculate the likelihood. Then I made a function named _Predictions that gives the likelihood of the values and simultaneously I stored this values in two lists i.e., PredictionsSurv, PredictionsNSurv for storing likelihood of survival class and non-survival class respectively.

Then I wrote a function to calculate the prior probability. Then I wrote a function to calculate the evidence .

Finally, I wrote the function posterior probability. And, based on the posterior probability to predict the class, I wrote a function named Predictions that helps to determine whether the according to given features person survived or not survived.

Finally, I also calculated the accuracy of my implemented gaussian naïve bayes classifier. The accuracy came out to be: 76.92307692307693%

<u>Task - 4:</u> Perform 5-fold cross-validation using the entire training feature set.

I performed 5-fold cross validation using the entire training set. I got the accuracy of the 5 models as:

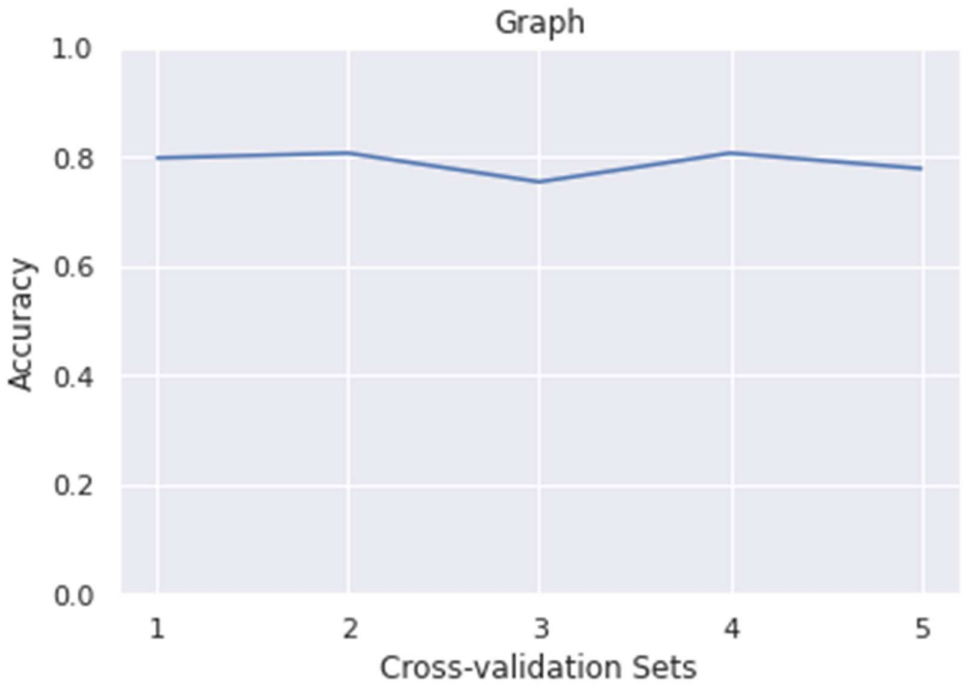
{1: 0.7982456140350878, 2: 0.8070175438596491, 3: 0.7543859649122807, 4: 0.8070175438596491, 5: 0.7787610619469026}

I got mean accuracy of all the 5 folds as 78.90855457227138%.

This clearly specifies that the naïve bayes classifier implemented from scratch is correct.

<u>Task - 5:</u> Visualize and summarize the results across the cross-validation sets. Compute the probability of the top class for each row in the testing dataset.

For, the visualization of the results across the cross-validation sets, I plotted a graph that helps to visualize the accuracy of cross-validation sets. The graph that I got is:



To compute the probability of the top class of each row in the testing dataset for which it has been classified to either survived class or not survived class. I just used the made a function named “probabilityOfTopClass” that returns the probability of top class of each row in the given dataset.

<u>Task - 6:</u> Compare your scratch implementation with scikit-learn in terms of the performance.

For this task I imported the Gaussian Naïve Bayes Classifier from the sklearn library and first instantiated the GaussianNB(). Then, I took the training sets and trained the Gaussian Naïve Bayes Classifier using fit() method. Then I predicted the values of testing sets from the model trained using the predict() method.

Hence, for the predictions that we got, I computed the accuracy came out to be: 76.223776223776 21%.

The accuracy came similar to the classifier that I implemented from scratch and hence it specifies that the implementation is correct.

<u>Task - 7:</u> Implement any other model of your choice [not necessarily from scratch] and perform 5-fold cross-validation and summarize the results. Compare it with the Naive Bayes Classifier you have implemented and justify your results.

I implemented the decision tree classifier; I Imported the decision tree classifier from the sklearn library and first instantiated the decision tree classifier. Then, I took the training sets and trained the decision tree classifier using the fit() method. Then I predicted the values of testing sets from the model trained using the predict() method.

Hence, for the predictions that we got, I computed the accuracy came out to be: 0.7622377622377622.

I also applied the 5-fold cross validation and summarize the results and got the mean accuracy for 5 folds as: 74.68250271696941%

Since, the accuracy of both predictions and the folds came out to be similar to the classifier that I implemented from scratch and hence it specifies that the naïve bayes classifier and decision tree classifier both helps in classification and hence if I would have tuned the hyperparameters of the decision tree, I would have got higher accuracy but still the accuracy I got depicts that the naive bayes classifier gives correct results of predictions when compared to the decision tree.

Question - 2:

- **Data Pre-Processing.**

I first took the input of the given dataset. Then I performed the data pre-processing on the given dataset.

I checked if there are any null values present in the dataset. It appears that there are no null values present in the dataset.

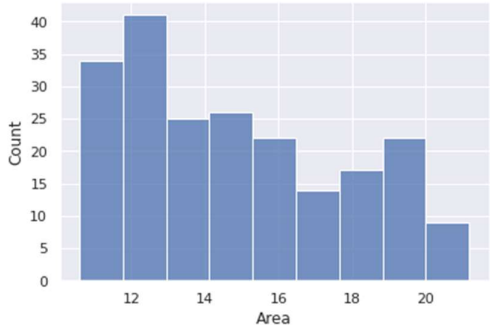
```
Area                      0
Perimeter                 0
Compactness               0
Length of kernel          0
Width of Kernel           0
Asymmetry coefficient     0
Length of kernel groove   0
Classes                   0
dtype: int64
```

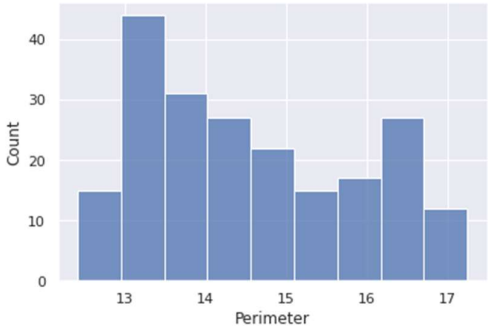
Since no categorical data was present in the dataset, hence, no categorical encoding was required.

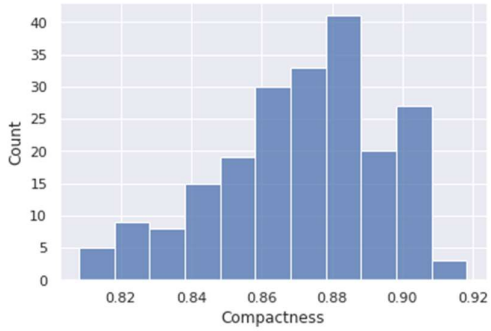
Task - 1: Use histogram to plot the distribution of samples.

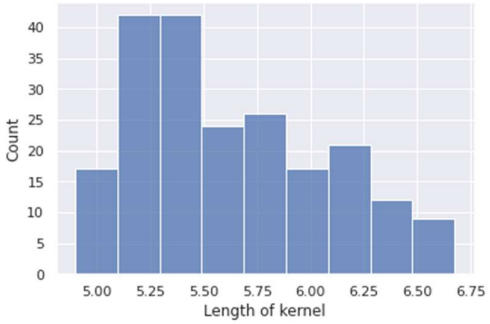
For this task we have to plot the histograms to plot the distribution of samples. So, for plotting the distribution of samples, I plotted the histogram using the histplot() method of seaborn library. histplot() method helps to show the distribution of the dataset.

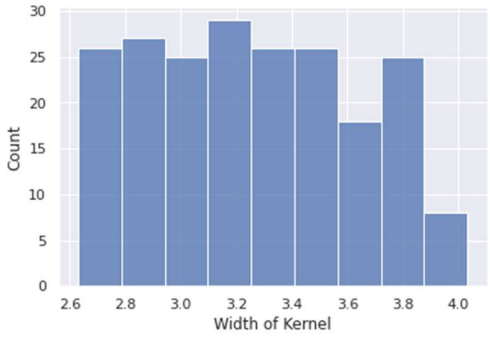
The distribution plot of samples that I got for different features of dataset is:

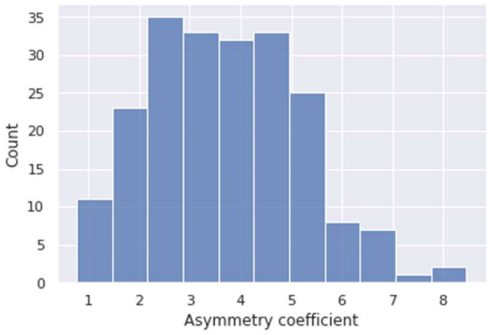


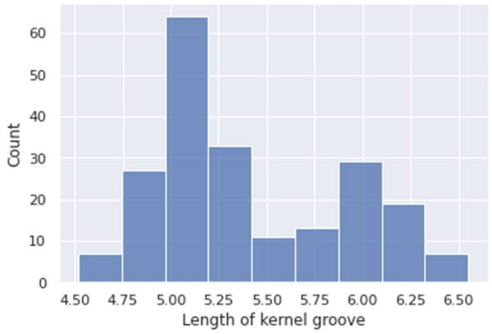












<u>Task - 2:</u> Determine the prior probability for all the classes.

For calculating the prior probability for each of the class. I just applied the concept/formula of prior probability. The prior probability is defined as count of samples of each class/ total number of samples.

So, for calculating the count of samples of each class, I just iterated over the column "Classes" and stored the count of samples of each class. Then, I just applied the formula of prior probability to calculate for all the classes.

I got the prior probability as:

```
Prior Class 1: 0.3333333333333333
Prior Class 2: 0.3333333333333333
Prior Class 3: 0.3333333333333333
```

Where Prior Class 1, Prior Class 2, Prior Class 3 represents the prior probability of each class.

<u>Task - 3:</u> Discretize the features into bins from scratch. Use of pandas, scikit learn and scipy is not allowed for this subpart.

For this task i.e., to discretize the features into bins from scratch. I made a function binning which takes dataset and features (features of which binning is to be done) as parameter.

So, I iterated over each feature and calculated the maximum and minimum value of that feature/column and also calculated the range (i.e., maximum value of that column - minimum value of that column) and divided the range by 10 ('10' here represents the number of bins in which the feature has been discretized). Then, I just iterated over dataset column and checked in which bin the value exists. So, I just updated the value to the bin value. Hence, the features of dataset was discretized into bins successfully.

Then for further tasks I just split the data into train and test sets.

<u>Task - 4:</u> Determine the likelihood/class conditional probabilities for all the classes.

For determining the likelihood/class conditional probabilities for all the classes. I made a function likelihood that takes the dataset as the parameter. In this function I first separated the dataset into three sub datasets (i.e., dataset corresponding to class1, dataset corresponding to class2 and dataset corresponding to class3).

Then, I iterated over each sub datasets, for each sub dataset I iterated over its columns and calculated the probability of bins present in that column. For example, in dataset corresponding to class 1, I stored the probability (count of particular bin/ length of sub dataset) in list and appended the list into the main list. I also applied the concept that minimize the error that if count of a bin is 0, I just increased the count of bin by 1 and also updated the probability.

I finally returned the main list that contains probability of each class, each feature, each bin. Hence, I calculated/implemented the function to determine the likelihood or the class conditional probabilities for all the classes.

I have also printed the likelihood for each feature of each class in a better way such that one can visualize, in my colab file.

<u>Task - 5:</u> Plot the count of each unique element for each class. Compare the plot with the plot of distribution.

For plotting the plot that describes the count of each unique element for each class. I just made the sub datasets corresponding to each class (i.e., dataset corresponding to class1, dataset corresponding to class2 and dataset corresponding to class3). Then I just iterated over each sub dataset and for each feature, I plotted a histogram that shows the count on y axis and feature on x-axis. I plotted these graphs for each class.

<u>Task - 6:</u> Calculate the posterior probabilities and plot them in a single graph. Analyse the plot.

To calculate the posterior probabilities, I made a function posterior that takes index, dataset and likelihood (computed in the previous task) as input parameters. So, in this function I made an array that contains the value of features of that row whose posterior probability is to be calculated. For each sample_index whose posterior is to be calculated, I updated the value of features corresponding to that row. I initialized the posterior probability for each class to the prior probabilities (computed in one of the previous tasks). Then, I iterated over each feature and updated the posterior probability of each class over each iteration. Finally, I got the probabilities (i.e., likelihood*prior). So, for calculation final posterior probabilities for each class we need to divide the computed probabilities with the evidence. Hence, divided each of the probabilities with evidence and hence we got the posterior probabilities of each class. Finally, I returned those posterior probabilities.

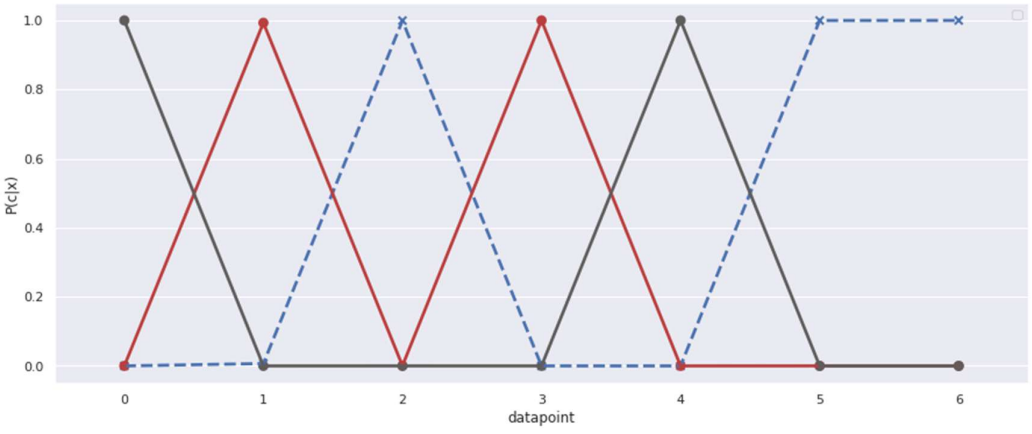
Now, I just calculated the probabilities by running the implemented function for test dataset.

For plotting the graph, I made three list (i.e., for each class) and appended the posterior probabilities for that class in their particular list. Then for graph I just plotted these posterior probabilities v/s datapoints in my colab file.

This graph shows that for each datapoint one of the classes gives higher posterior probability which implies that particular class corresponds to that datapoint, helps in classification. These graphs show the naïve bayes classification that how the best class is chosen from the posterior probabilities of three classes that gives the highest posterior probability.

So, I plotted two graphs one for 7 datapoints and the other for whole testing dataset. I got the results as follows:

Graph-1 (for 7 datapoints):



Graph-2 (for the complete dataset)