Suryank Tiwari - MT19019

SML Report – Assignment 5

Question 1.

1. Data Analysis and Visualization

These plots are constructed on the pre-processed dataset. The pre-processing steps involved:

- Dropping the ***'No'*** column.
- Filling the null values in the ***'pm2.5'*** column with the median values.
- Replacing the string values in ***'cbwd'*** with distinct integer samples.
  NW = 1, cv = 2, NE = 3, SE = 4

1.1 Pairwise Plots:

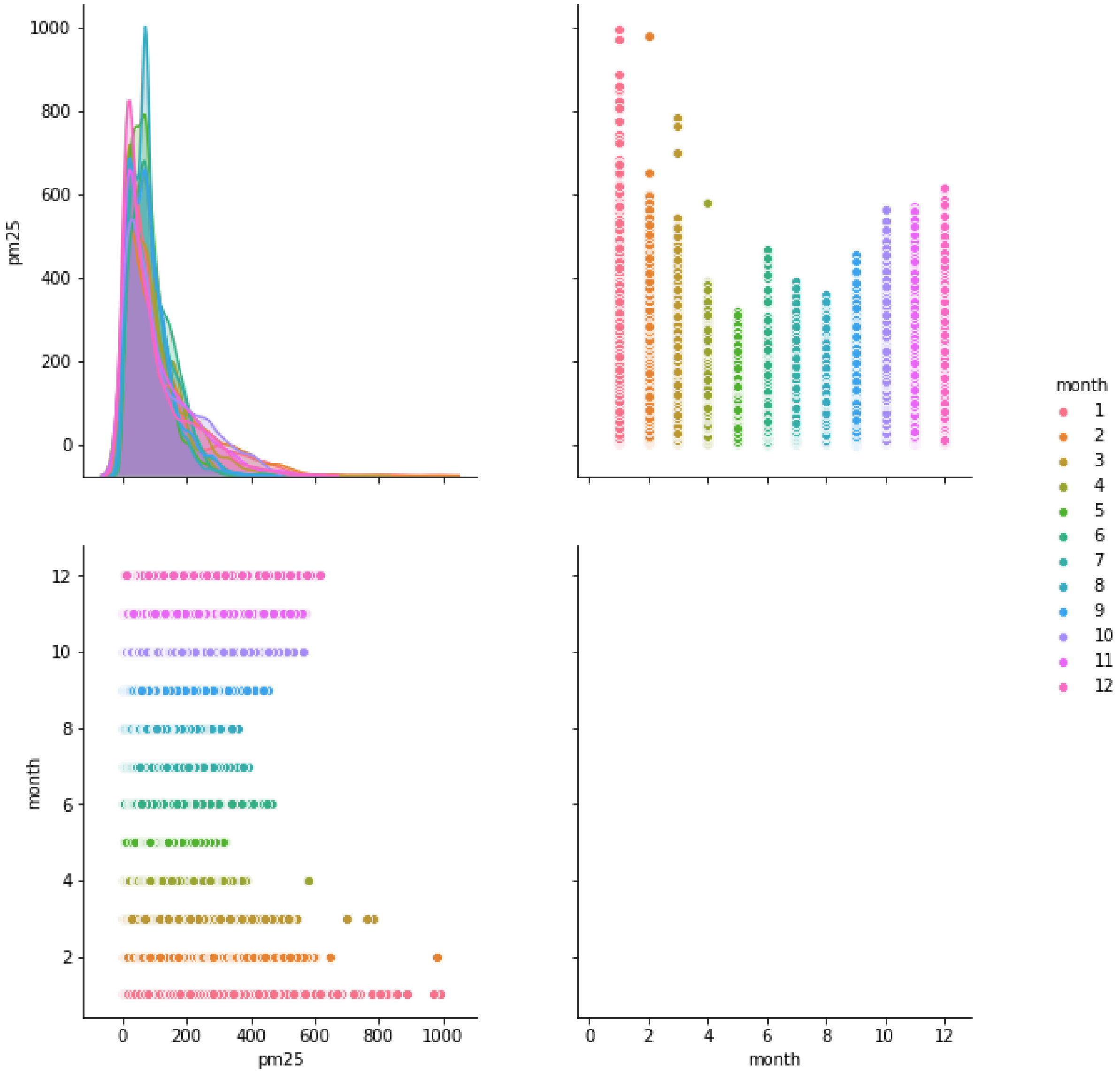


Fig: pairwise plot 'pm2.5' vs 'month'

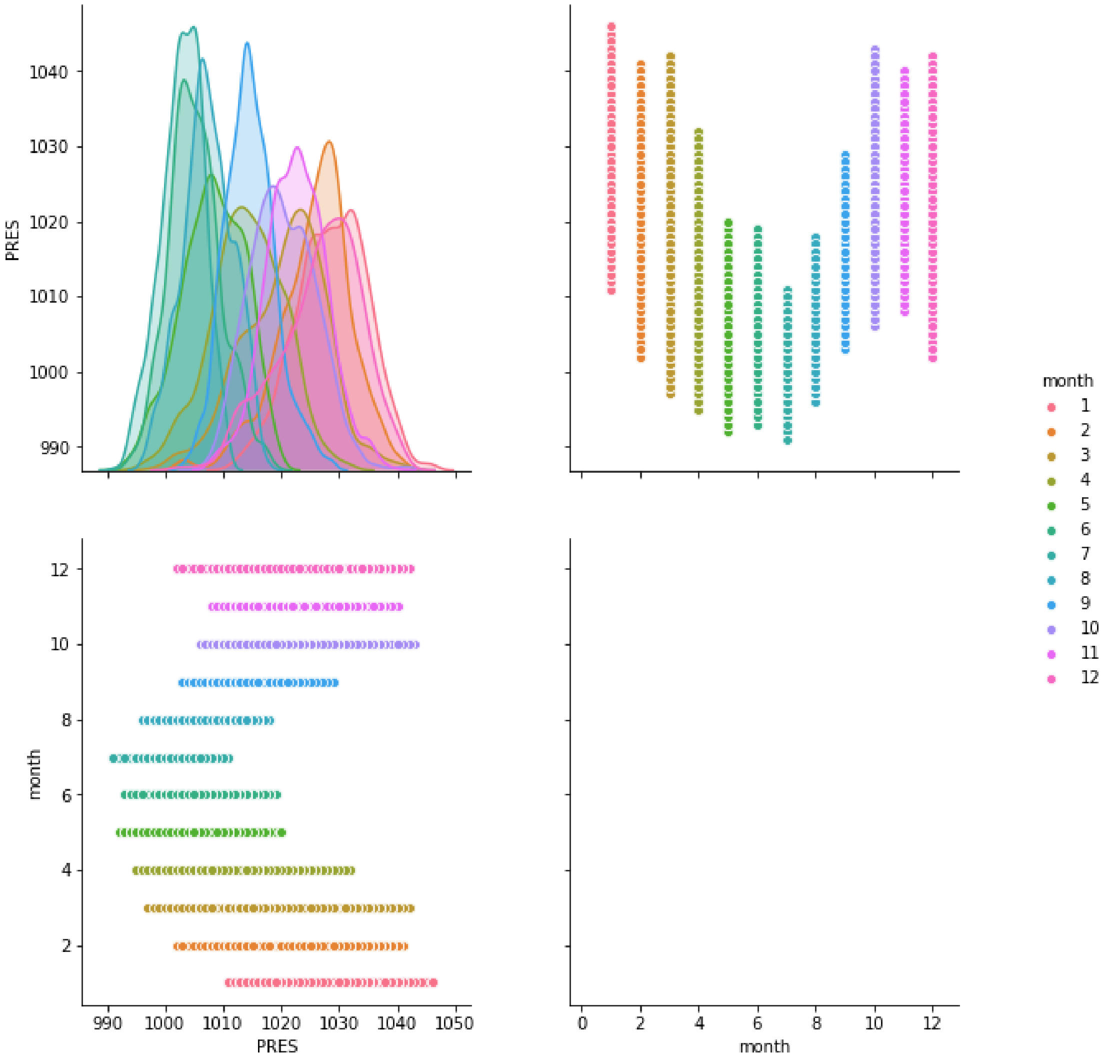
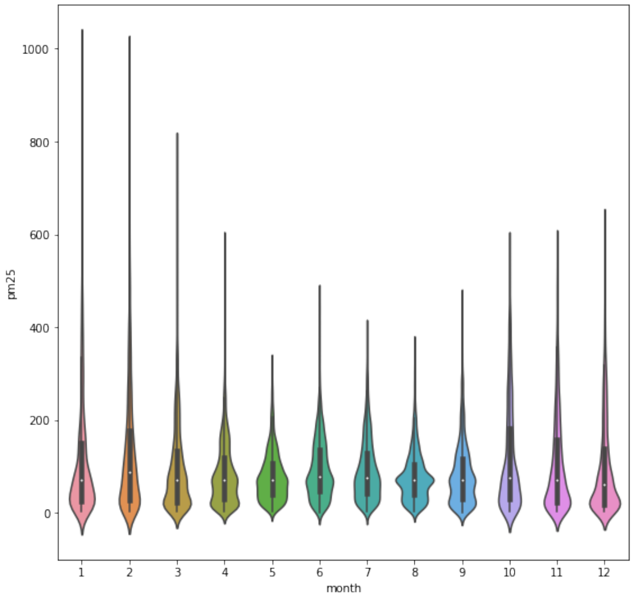
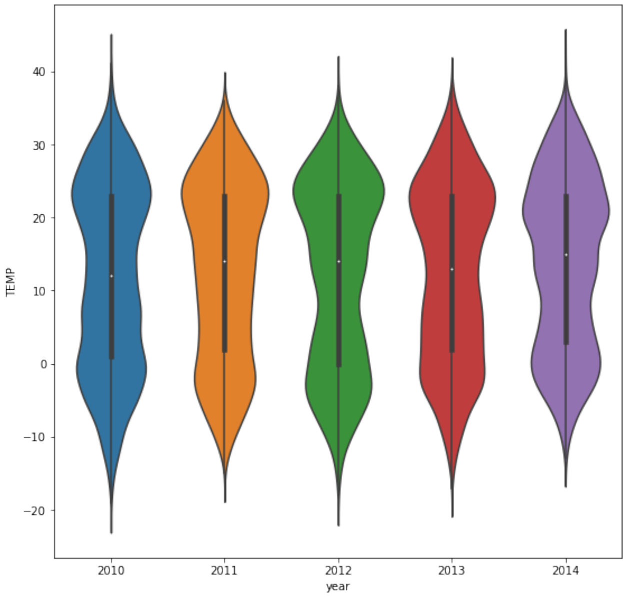


Fig: pairwise plot 'PRES' vs 'month'

1.2 Violin Plots:



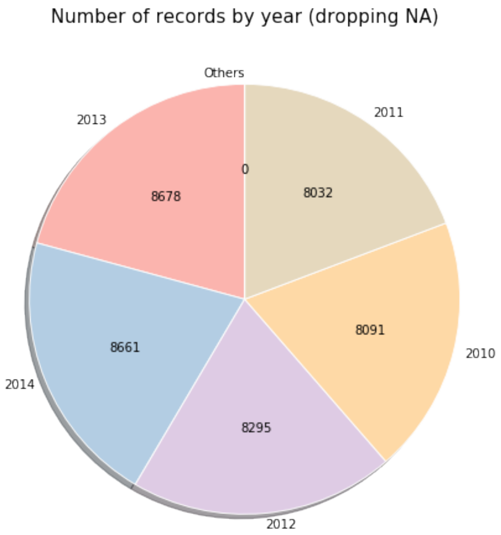
'pm2.5' vs 'month'

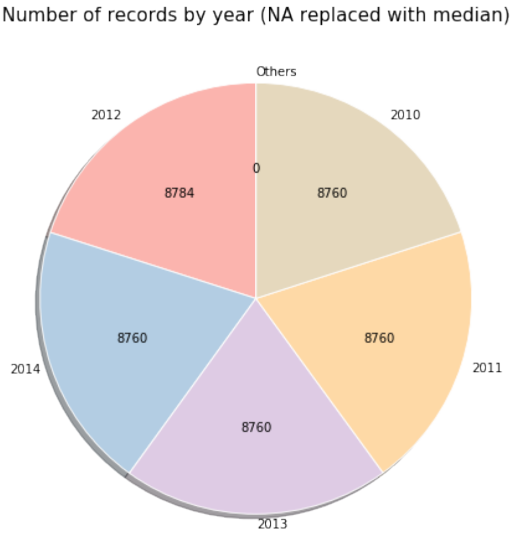


'TEMP' vs 'year'

1.3 Other Plots:

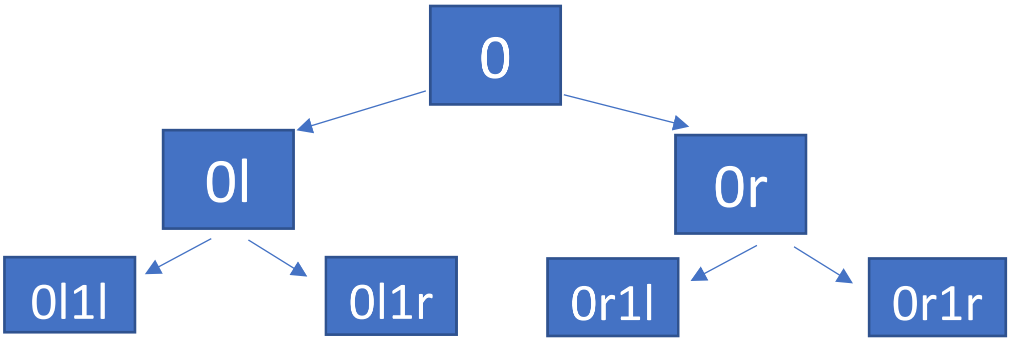
Number of records:





2. Methodology

2.1 Region Division and notation nomenclature: The initial training data is considered to be region '0'. Each region



Every split is denoted by a suffix **(Level of Split) + (Direction of split)**. For each left split, suffix 'l' is added and for each right split suffix 'r' is added. So, 0 gets split into 0l and 0r. Each of which split into 0l1l, 0l1r and 0r1l, 0r1r respectively.
If we consider the leftmost splits only, the pattern would go like. 0 -> 0l -> 0l1l -> 0l1l2l and so on.

The region ids at the leaves of the tree represent the regions present in the tree finally.
Each region is a data frame consisting of rows as entries in that region.

2.2 Function and Class Descriptions

- **Class Tree:** Represents a single decision tree. Constructor takes tree type as argument. Type=1 for classification and Type = 2 for Regression.
    - **train (data, rf):** This function is called to train the decision tree given on the data frame **'data.'** The value of **'rf'** is false by default, unless stated otherwise while calling, and represents whether or not random predictors will be chosen for each split.
    - **make_split (predictor):** This function takes in a column key – like 'TEMP' and will return best split possible based on lowest error found among the '**no_samples'** number of samples checked in all the regions each for the predictor specified.
    - **compute_error (regions):** Takes in a list of regions, computes all their respective errors (talked in later section) and returns a **weighted average** of them.

- **compute_accuracy (region, rid):** This function returns accuracy of the region **'region'** with region id **'rid'**. in case of classification and MSE in case of regression.
- **predict (data):** Computes accuracy or MSE for the testing data frame **'data.'**
- **predict_point (point):** Computes the predicted class or MSE for a sample point.

- **Class BaggedTrees:** Represents a set of decision trees trained on bootstrapped datasets from our training data. Takes the training data in the constructor out of which the bootstrapped training datasets are generated. Other parameters of the class constructor are number of trees to train and the type of those trees as a single integer (1 or 2).
    - **train (rf):** This function trains all the decision trees on their respective bootstrapped datasets. If rf value is true then **Random forest** is applied in the decision trees otherwise not.
    - **predict (data):** Predicts the test data – **'data'**, and returns the accuracy or the MSE.

2.3 Methodology

For Decision Tree:

- A Tree object is created with it's type 0 or 1. 0 for classification, 1 for regression.
- Training data is passed to the **train()** function.
- Till number of iterations (pre-decided) have been reached, the following steps are repeated.
    - Predictors are decided that will be used for the split - either all or on a random basis.
    - For all predictors the **make_split()** function is called.
    - To make a split, total **no_samples** random values are taken from a region. That region is divided into two parts based on that value for the pre-specified predictor, and entropy for resultant regions is computed by calling the **compute_error()** function. This is done for all possible values decided earlier. This process is repeated for all regions. The minimum **error region, predictor, split value** is returned to the train function.
    - The minimum split is decided and the regions are split on that basis. The region which has been split is deleted and the two sub regions are added in the region list.

For all the samples in a region:
Classification Rule: The class that is maximally present (month) is decided as the class for all the points in that region.
Regression Rule: The mean of their values (pm2.5) is returned as the result.

Then predict() is called for accuracy/MSE computation.

For Bagging:

- Bootstrapped datasets are created using df.sample(frac=1, sample=True) to create repeating values and the same size of dataset. Number of datasets created is equal to the number of trees we have.
- All the trees are trained on different datasets.
- For each point in training data:
    - For classification: predict each point by all the trees and get their results using **predictPoint()**. Majority voting is performed and the result is stored. Accuracy is computed by number of correctly classified points.
    - For regression: MSE of each point is computed by all the trees and added and averaged out. It is the resultant output.

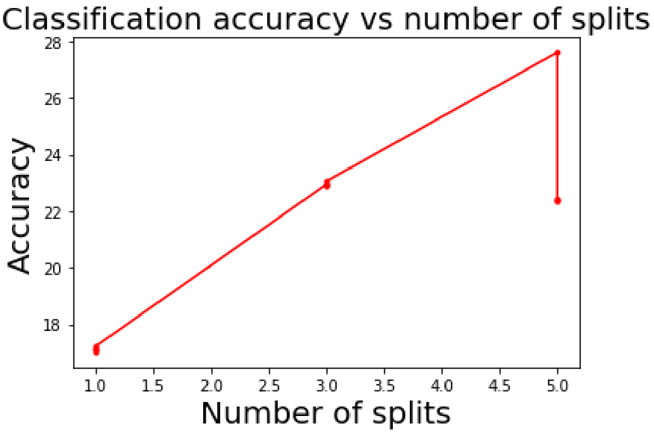
For Random Forests:

While initializing the bagging object, pass rf=True or pass rf=True while training each decision Tree.

3. Results

Single Decision Tree Classification:

	No of Samples	20	100	500	1000
Splits=1	Accuracy	**17.11%**	**17.11%**	**17.02%**	**17.22%**
	Time	4.16s	11.53s	100.98s	183.14s
Splits=3	Accuracy	**22.94%**	**22.92%**	**22.92%**	**23.05%**
	Time	12.53s	93.21s	434.05s	707.67s
Splits=5	Accuracy	**23.60%**	**22.39%**	**22.39%**	**22.40%**
	Time	35.95s	199.39s	817.15s	1364.23s



Single Decision Tree Regression:

	No of Samples	20	100	500	1000
Splits=1	MSE	**8346.56**	**8448.97**	**8448.97**	**8444.25**
	Time	12.62s	53.56s	286.03s	635.51s
Splits=3	MSE	**7204.76**	**7217.37**	**7207.69**	**7217.37**
	Time	29.77s	153.86	947.75s	1501.20s
Splits=5	MSE	**6193.51**	**6031.80**	**6031.80**	**6004.65**
	Time	61.01s	334.71s	1454.37s	3096.28s

Bagging Classification:

Number trees = 5

Number of regions: 4

Number of iterations 3

Number of samples 100

TRAIN DATA REGIONS

0r

0l1r

0l1l2l

0l1l2r

Bagging result: 21.90639269406393 %

Bagging Classification - Random Forest:

Number trees = 5

Number of regions: 4

Number of iterations 3

Number of samples 100

TRAIN DATA REGIONS

0l

0r1l

0r1r2l

0r1r2r

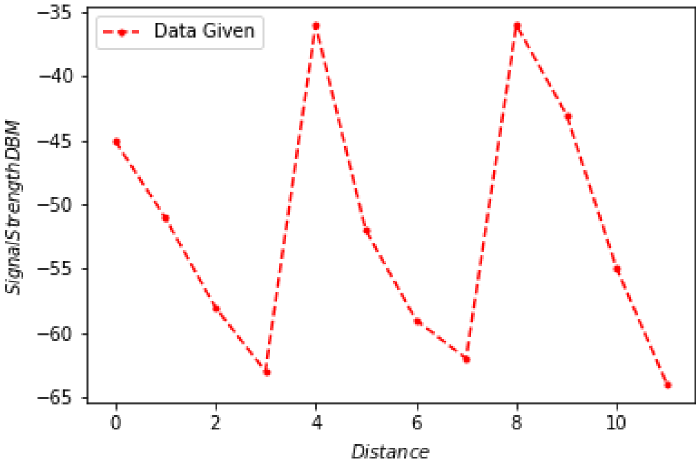
Bagging result: 22.123287671232877 %

4. Assumptions

- Some regions might be empty based on data.
- Stopping criterion is train_itrs, i.e. number of splits to be performed in the tree.
- **No_samples** is taken as a variable because code becomes too slow when we consider all the points.

Question 2.

1. Data Analysis and Visualization



2. Methodology

- Splitting the data into train and test as guided.
- Converting the data into proper format using **np.atleast()**.
- Using RBF and Constant Kernel imported from sklearn with values:

  Constant Kernel (1.0, (1e-1, 1e3)) * RBF(10.0, (1e-3, 1e3))
- Obtaining y_predicted on x_test and the covariance matrix.
- Obtaining confidence values with 95% confidence.
- Plotting

3. Results

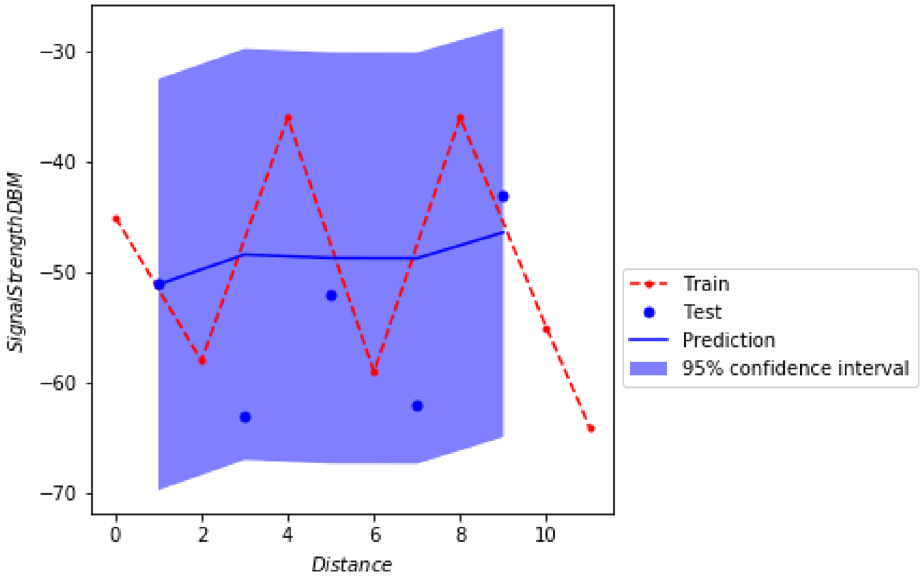
Calculated Value at Test points [-51.1019041, -48.37792764, -48.70005987, -48.71708437, -46.38371664]

Expected Value at Test points [-51, -63, -52, -62, -43]

MSE: 82.51807506410579

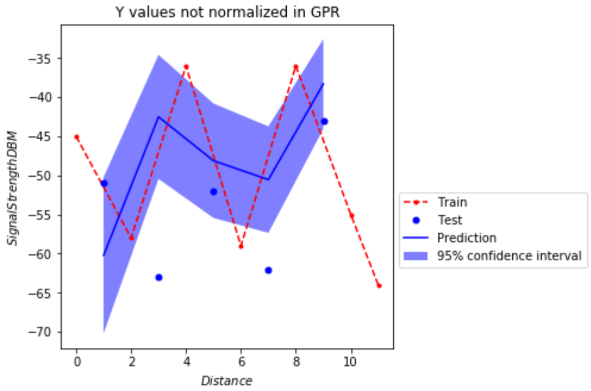
Confidence at Test point predictions: [18.60058931 18.6005239 18.60052389 18.60051752 18.51805098]

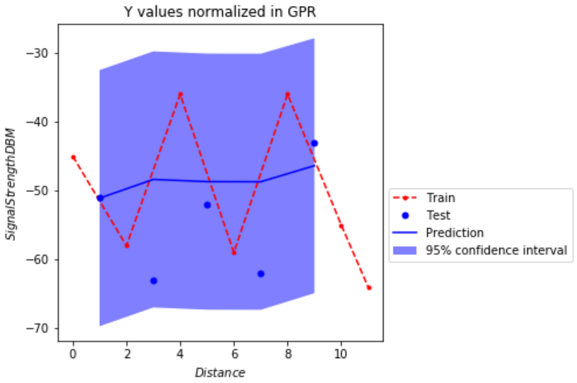
Plot of results obtained after GPR:



4. Inferences

Comparison on normalization vs non normalization of y values:





MSE: 134.69226313768107

MSE: 82.5181210439411