Solar Wind Propagation Delay Prediction using Ensemble Regressor and RNN

In this project, we are trying to predict the propagation delay of a solar wind using an ensemble of regression algorithms (random forest, KNN , extra trees and gradient boosting algorithms) and neural network (RNN).

Packages used:

1. Numpy
2. Pandas
3. Matplotlib
4. Scipy
5. Scikit-learn
6. Pytorch

The regression problem was solved using the following steps:

1. Data Visualization
2. Data Preprocessing and Cleaning
3. Apply algorithms
4. Evaluation
5. Validation on Test set

Data sets were created in 3 different ways in order to figure out the best way to create the data for reducing prediction error. So, the model training has been divided into 2 parts based on the algorithms applied and the type of data set used:
Part 1. Using machine learning algorithms (ensemble of random forest, knn, extra trees and gradient boosting algorithms) to predict the propagation delay.
Part 2. Using neural network (RNN) to predict the propagation delay.

For Part 1, the data sets were created using 2 methods and the ensemble regressor was applied on each of those 2 sets. Data sets were created by:
Method 1. 80 by 11 matrix was converted to 1 by 880 vector as scikit-learn algorithms can only work on vector.
Method 2. 1 by 11 vector was created by taking the average of each 11 columns of 80 by 11 matrix.

For Part 2, the data was taken in the matrix form itself.

1. **Data Visualization**

   For data visualization and getting more information about the data, firstly, all the datasets were concatenated and a histogram plot of each attribute was plotted.

   The plots obtained are:

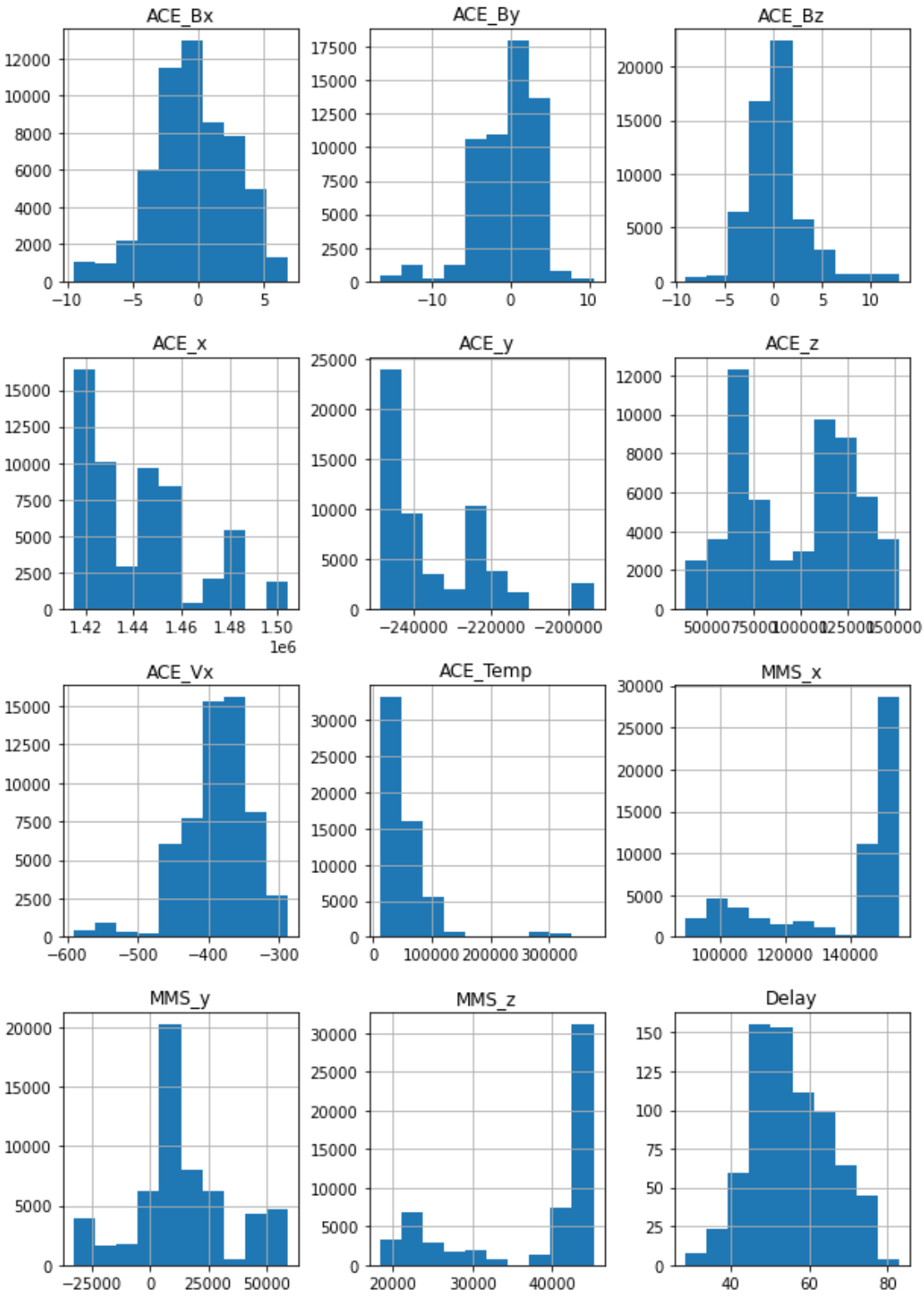


Figure 1: Attribute plot

From the histogram plot, we can see that the distribution is quite different in each of the attributes and none of them are normalized so, on each part (Part 1 and Part 2), we have normalized the data before applying any algorithms. Also, we have looked at the distribution of the entire data set on each part instead of only the attributes in order to figure out the total distribution range of each of the values in the dataset and making correct normalizing decisions.

Distribution of all the data used in Method 1 of Part 1.

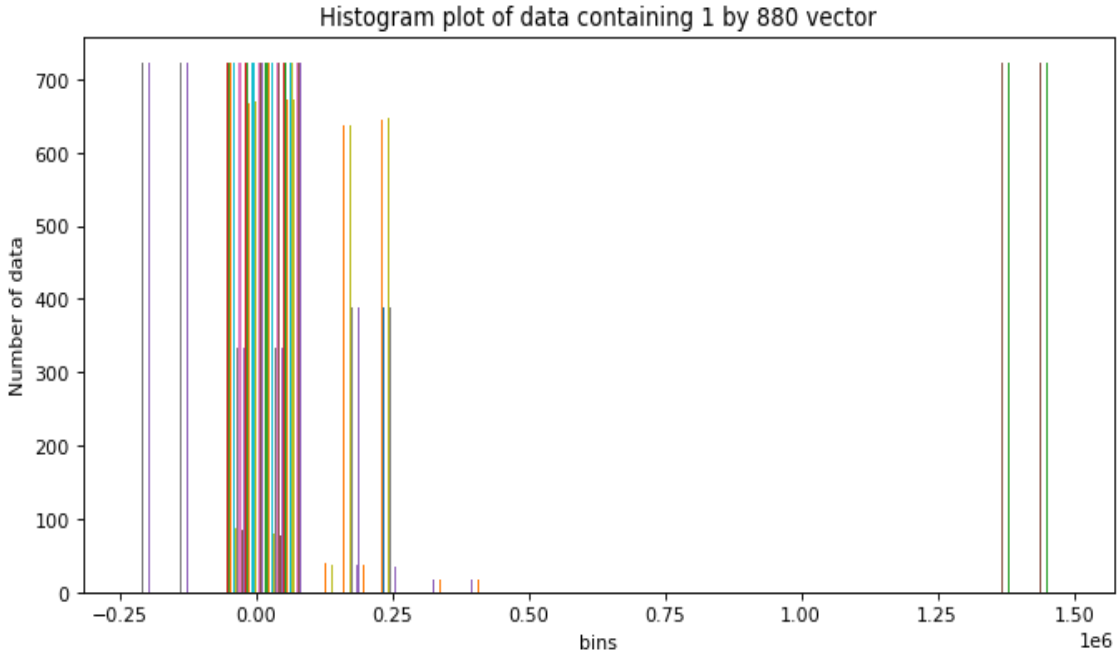


Figure 2: Distribution of dataset each containing data of 1 by 880 vector

The data has many values nearer to 0. Also, there are many huge values like 1.35 * 1e6. For normalizing the data, we subtracted each data by mean of the whole dataset and divided by the standard deviation.

Distribution of all the data used in Method 2 of Part 1.

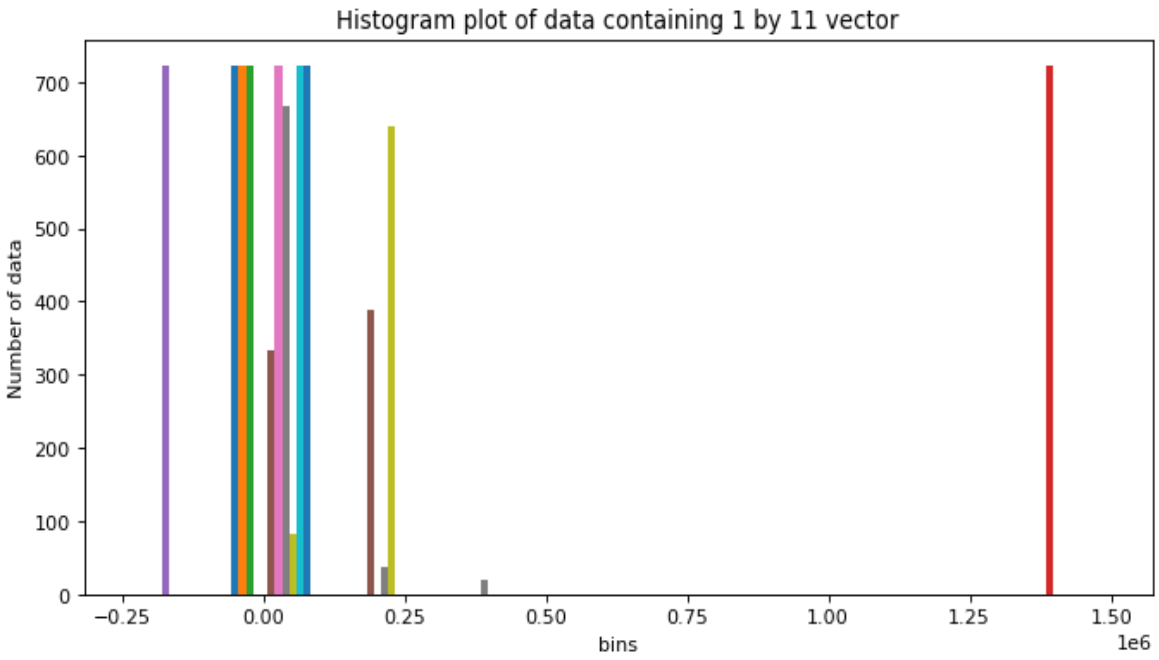


Figure 3: Distribution of dataset each containing 1 by 11 vector.

Though method 2 dataset has less data, it has similar distribution as method 1.

Plots after normalizing both the data:

Method 1:

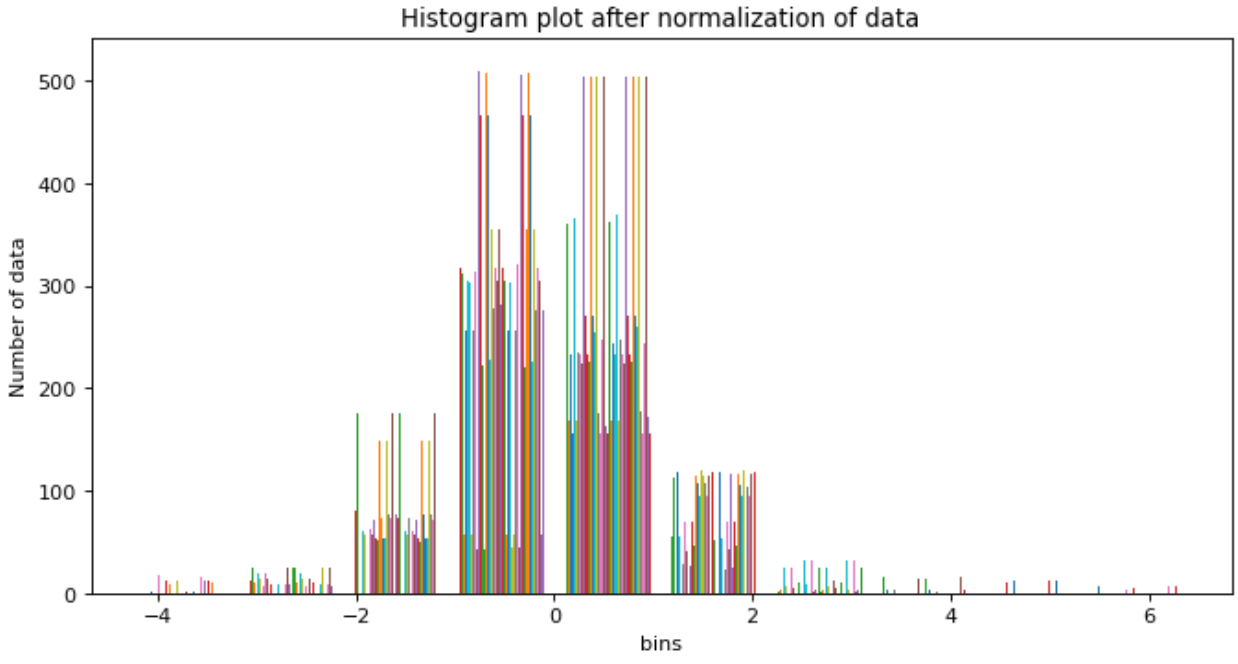


Figure 4: Histogram plot after normalization of data used in Method 1

Method 2:

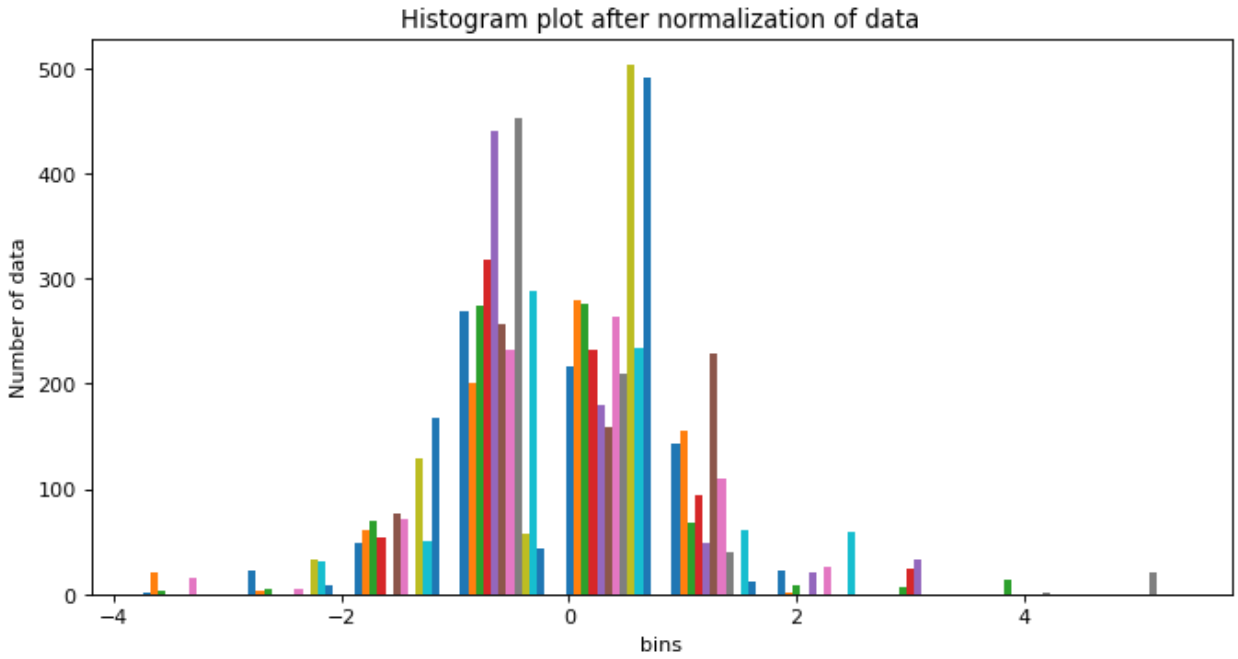


Figure 5: Histogram plot after normalization of data used in Method 2

Normalization reduced the range of data, however, we still have a large number of features. So, we reduced the dimension next.

Plot after applying dimensionality reduction on both the data

Method 1

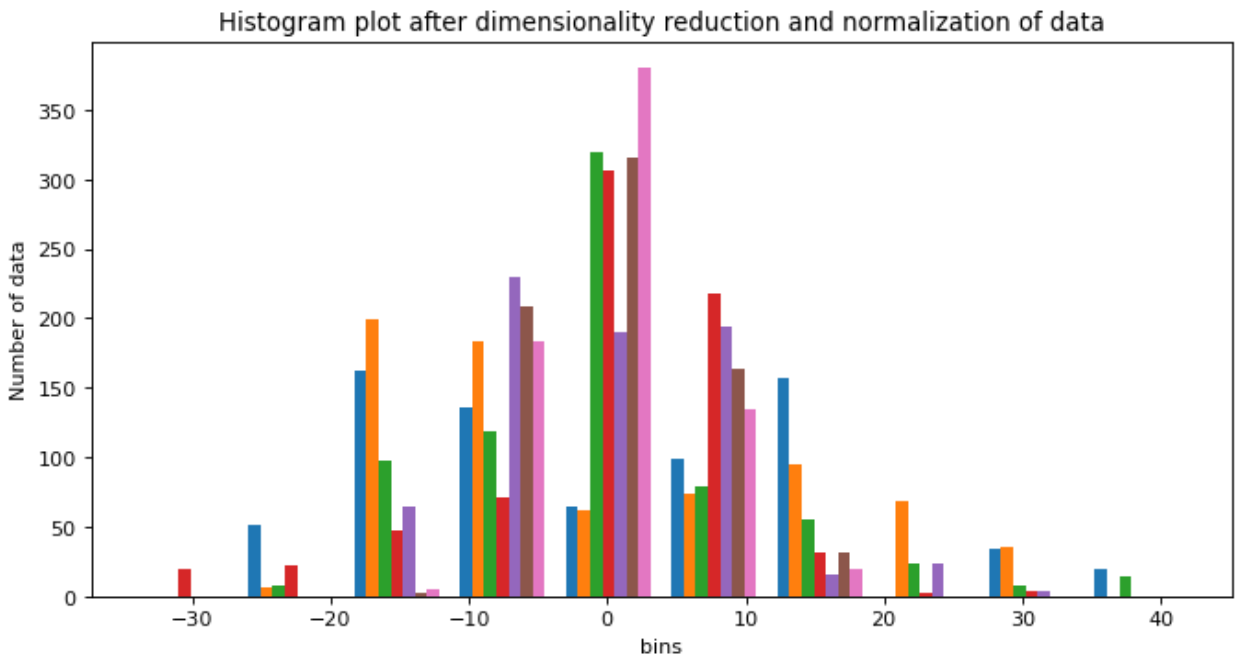


Figure 6: Histogram plot after dimensionality reduction of data used in Method 1

Method 2

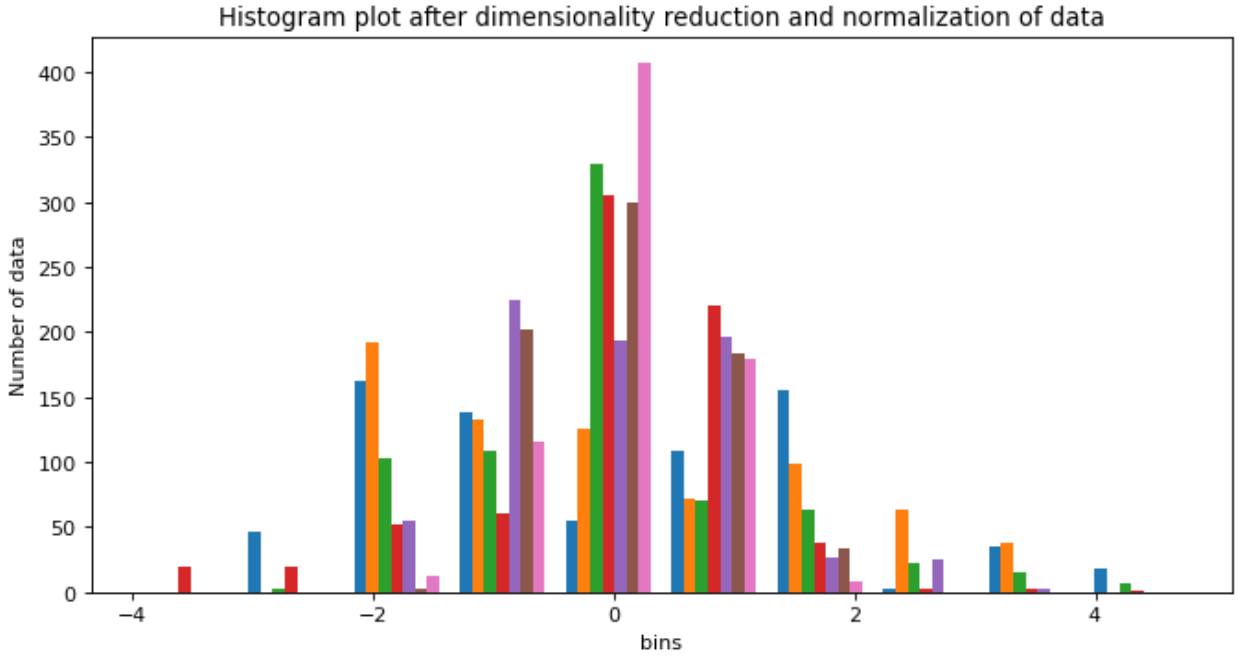


Figure 7: Histogram plot after dimensionality reduction of data used in Method 2

With PCA, we reduced the dimension, however, we can still see that data has large values greater than 30 and greater than 4 in method 1 and method 2 respectively. So, we renormalized the data in the next step using MinMaxScaler.

Data Preprocessing and Cleaning:

Data Preprocessing has been carried out in following 5 steps:

1. First csv file had 79 by 11 dimensions while all other sets had 80 by 11 dimensions. So, at first, we interpolated one additional row in each matrix in the first csv using linear interpolation so as to make the dimension equal in all the sets. After that,

2. Multiple excel files were merged. We found that there are 722 data in total.

3. Data concatenated from multiple excel files were randomly shuffled to reduce the bias in the model.

4. Data was normalized by subtracting each data by mean and then dividing by standard deviation. (Figure 4 and 5)

5. Dimensionality reduction with PCA. (Figure 6 and 7)

6. Data was re-normalized so as to bring all the values between 0 and 1.

4. Data was splitted into 3 chunks in the ratio 70:20:10 in train, test and validation set.

Applying algorithms:

The following two algorithms have been used for training:

1. Ensemble of regression algorithms (random forest, knn, extra trees and gradient boosting algorithms).
2. Recurrent Neural Network (RNN)

For selecting the appropriate algorithm, MSE Loss versus hyperparameter plot was used.

Method 1:

Choosing an appropriate n_estimator and max_depth in Random Forest Regressor:

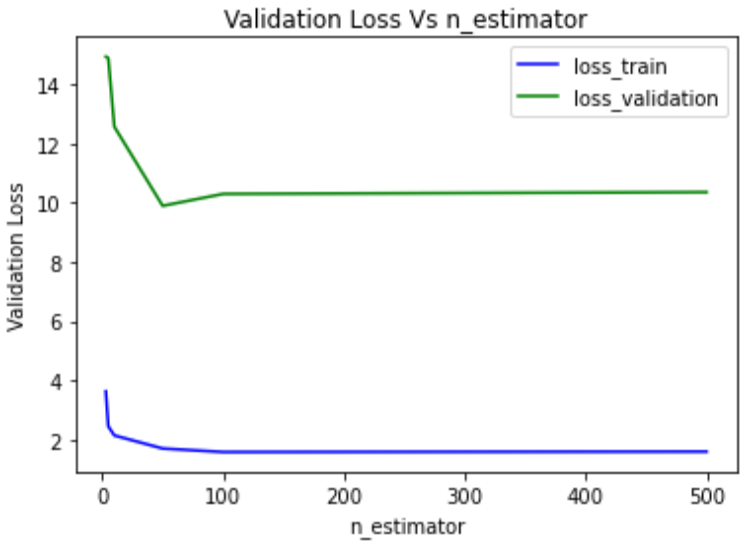


Figure 8: Loss versus n_estimator plot for Random Forest Regressor

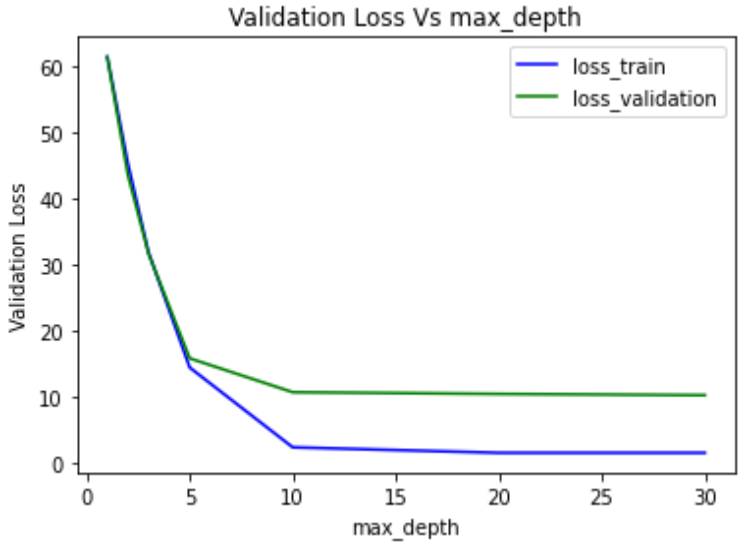


Figure 9: Loss versus max_depth plot for Random Forest Regressor

Thus, the appropriate n_estimator and max_depth for random forest regressor in Method 1 was found to be 50 and 10 respectively.

Choosing an appropriate max_depth in ExtraTreesRegressor:

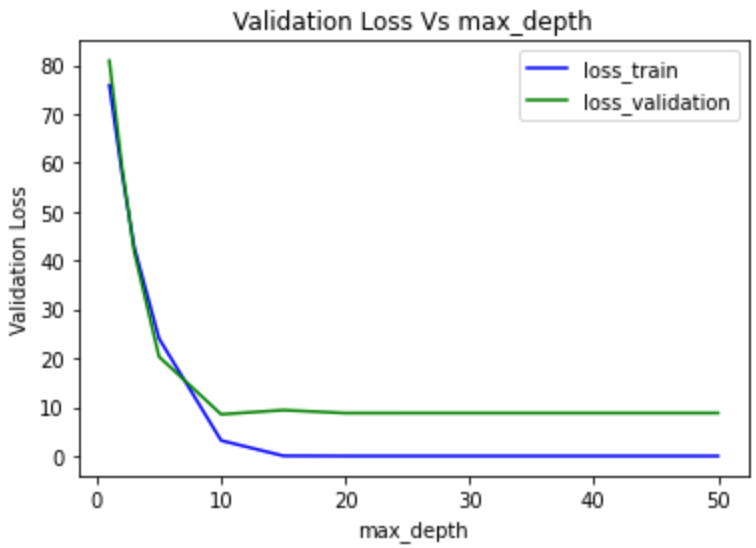


Figure 10: Loss versus max_depth plot for Extra Trees Regressor

Thus, the appropriate max_depth in extra trees regressor was found to be 10.

Choosing an appropriate max_depth for Gradient Boosting Regressor:

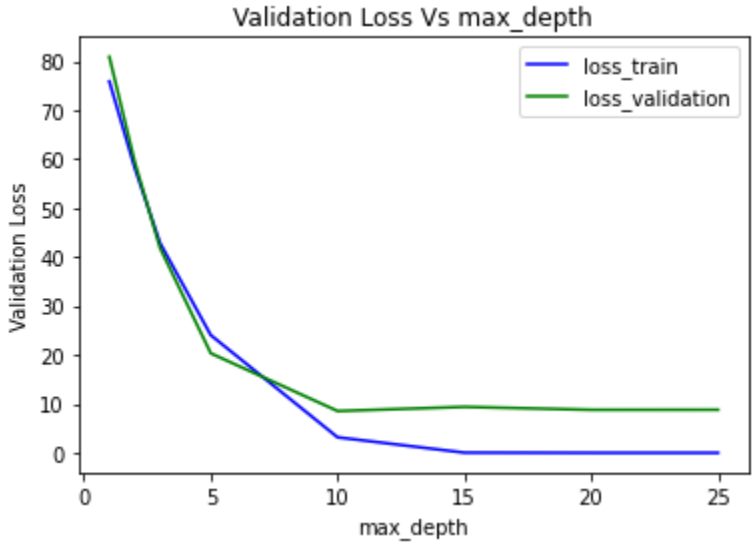


Figure 11: Loss versus max_depth plot for Gradient Boosing Regressor

Thus, the appropriate max_depth was found to be 10.

Method 2:

Choosing an appropriate n_estimators for random forest regressor:

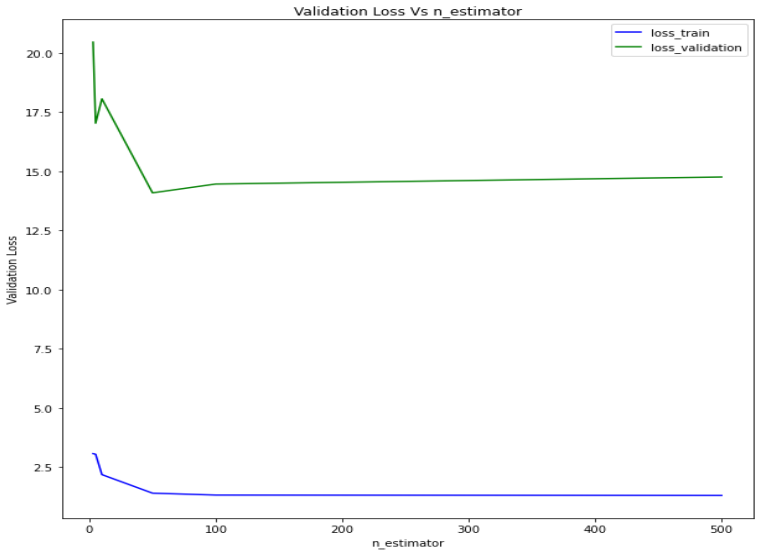


Figure 12: Loss versus n_estimator plot for Random Forest Regressor

Thus, the appropriate n_estimator was found to be 50.

Choosing an appropriate max_depth for random forest regressor:

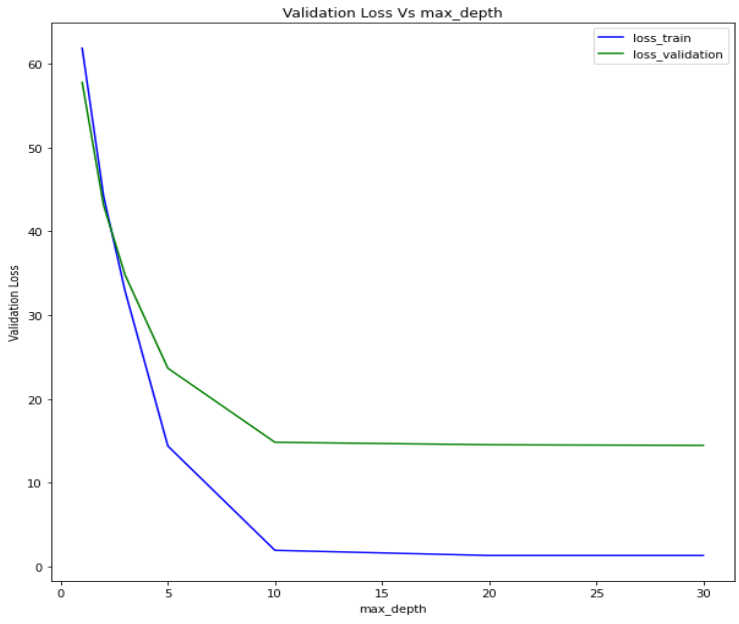


Figure 13: Loss versus max_depth plot for Random Forest Regressor

Thus, appropriate max_depth was found to be 10.

Choosing an appropriate max_depth for Extra Trees Regressor:

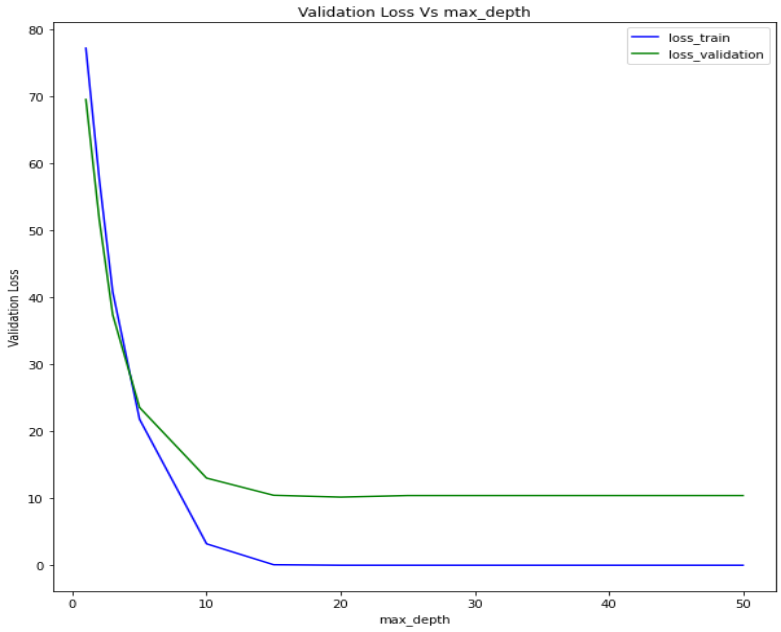


Figure 14. Loss versus max_depth plot for Extra Trees Regressor

Thus, the appropriate value of max_depth was found to be 10 for extra trees regressor.

Choosing an appropriate max_depth for Gradient Boosting Regressor:

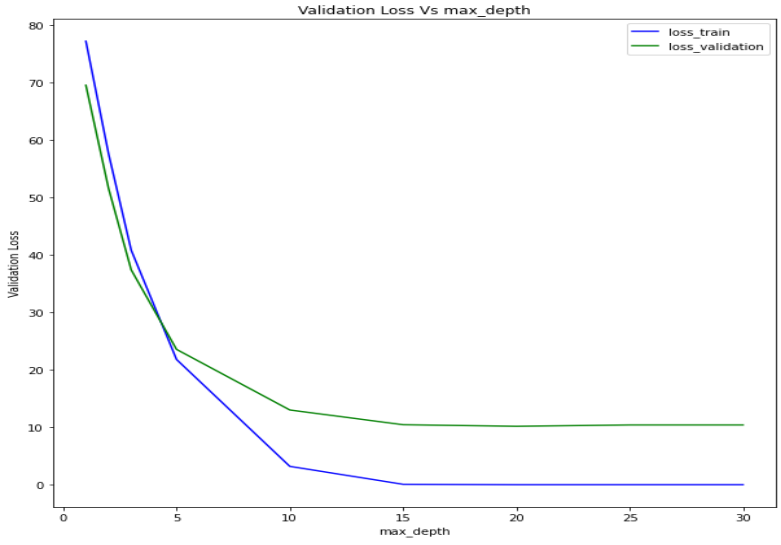
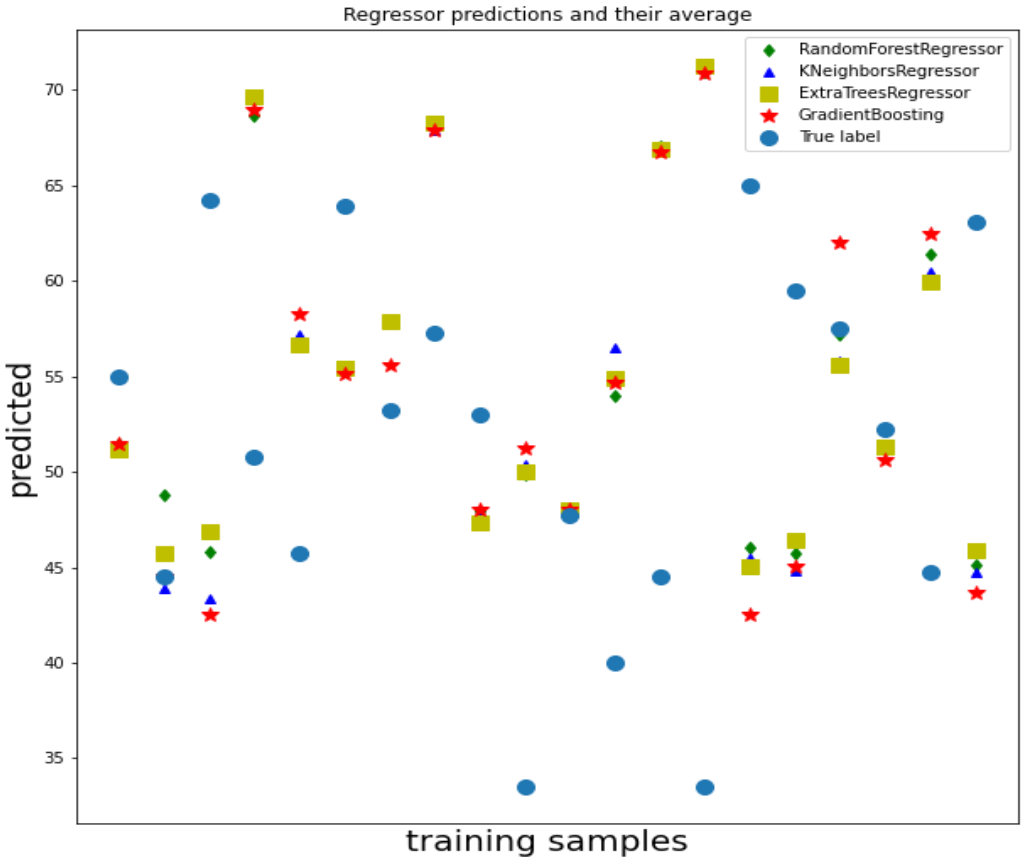


Figure 15. Loss versus max_depth plot for Gradient Boosting Regressor

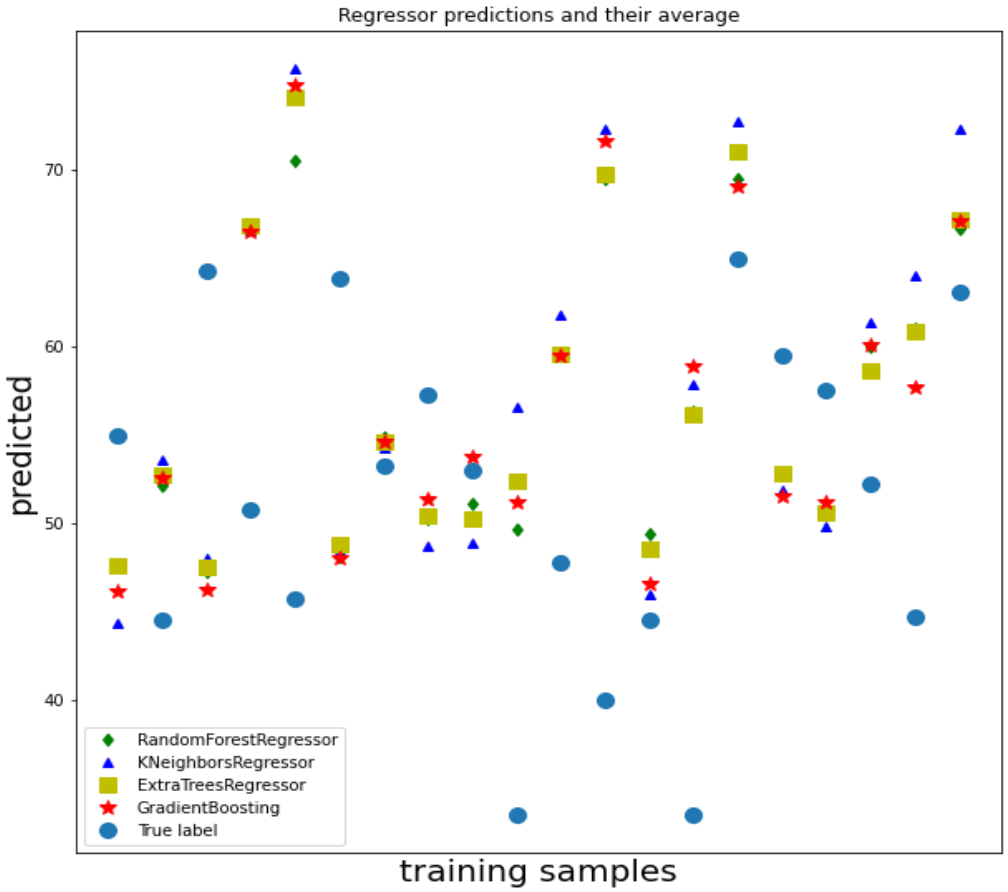
Thus, the appropriate max depth was found to be 10 for gradient boosting regressor.

Performance and Evaluation:

Plotting test prediction from models included in ensemble regressor in method 1 data:



Plotting test prediction from models included in ensemble regressor in method 2 data:



For measuring the performance of algorithms, R2 score on each of the sets was calculated.

	Ensemble of regressors on Method 1 dataset. (R2 score)	**Ensemble of regressors on Method 2 dataset. (R2 score)**
Train	0.98	0.98
Validation	0.90	0.87
Test	0.95	0.94

Mean Squared Error of ensemble regressors on Test set:

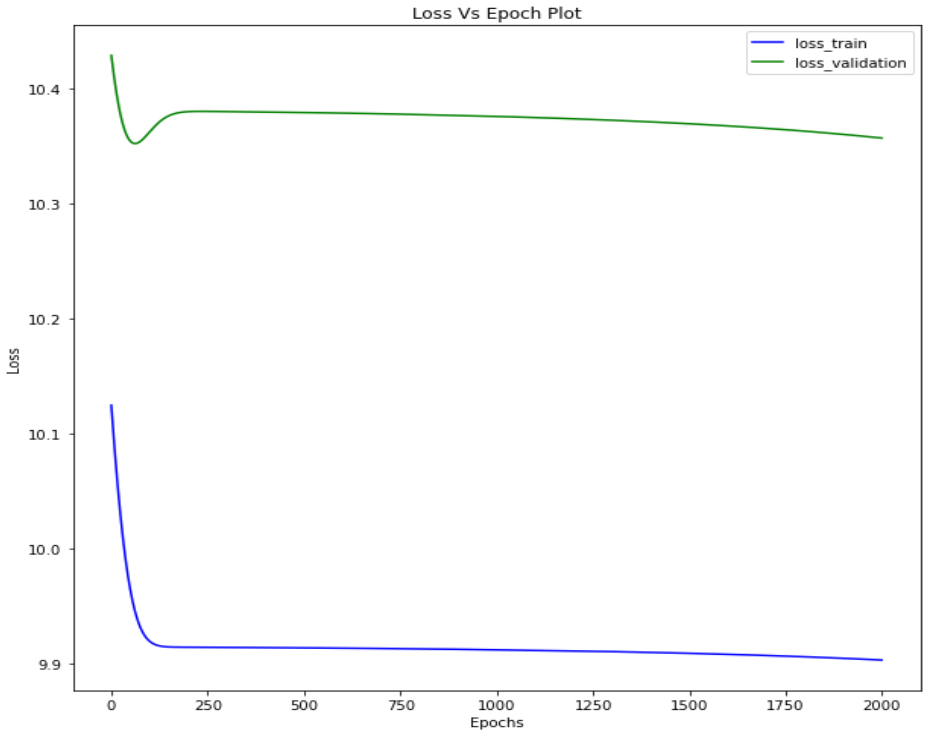
	Test Loss
Method 1	4.35
Method 2	5.48

Root mean squared error of the RNN model:

[Only the loss from last 10 epochs has been displayed here. For loss from the first epoch, please refer to .ipynb or .html file.]

epoch 1990, Average Train Loss: 9.902744, Average Validation Loss: 10.356894
epoch 1991, Average Train Loss: 9.902728, Average Validation Loss: 10.356860
epoch 1992, Average Train Loss: 9.902713, Average Validation Loss: 10.356827
epoch 1993, Average Train Loss: 9.902697, Average Validation Loss: 10.356793
epoch 1994, Average Train Loss: 9.902681, Average Validation Loss: 10.356759
epoch 1995, Average Train Loss: 9.902664, Average Validation Loss: 10.356725
epoch 1996, Average Train Loss: 9.902649, Average Validation Loss: 10.356690
epoch 1997, Average Train Loss: 9.902633, Average Validation Loss: 10.356657
epoch 1998, Average Train Loss: 9.902617, Average Validation Loss: 10.356623
epoch 1999, Average Train Loss: 9.902601, Average Validation Loss: 10.356588
epoch 2000, Average Train Loss: 9.902585, Average Validation Loss: 10.356554

Choosing an appropriate model for RNN:



Thus, the most appropriate model was found to be on epoch 150.

Average train, validation and test loss on the selected model [epoch 150]:

Average Train Loss	9.913957
Average Validation Loss	10.375898
Average Test Loss	9.26

Final Selected model:

Based on the performance of ensemble regressor trained on method 1 dataset, method 2 dataset and the performance of RNN, ensemble regressor trained on method 1 dataset was chosen as the final model.

Test set Validation on the final selected model:

True label	71.75
Predicted label	71.11

Conclusion:

Therefore, we tried ensemble regressors on two types of data sets. We also tried RNN on our dataset. As there is less data, ensemble regressors gave less loss than RNN. Then, we selected model produced using method 1 as our final model and infered and validated a random test data from the model.