Introduction

An ordinary man's first dream is to buy a house for himself, and everyone wants to buy the nicest property according to their budget, location, and so on. Every year, housing prices fluctuate based on three primary factors: location, condition of the property, and size of the house, creating a strong desire to develop a technique for forecasting housing prices. This method will be very useful for those who want to buy a house, when should a person sell a house, what prices should they pay based on the size of the house, and so on. Forecasting housing prices can be accomplished using various Machine Learning models, which can be extremely beneficial in addressing this challenge.

In this research, we will concentrate on the Gradient Boosting Model for home prices prediction. This model is being applied to the Boston Housing dataset, which was taken from the 1970 Census and contains 13 independent variables and one dependent(target) variable, which we are using to forecast house prices. In the dataset, characteristics influencing house prices include crime rate, house size, distance from prominent places in the area, air quality, and so on. We will examine how these various factors affect home prices using the Gradient Boosting model, which provides a final forecast which combines the prediction of numerous decision trees. We are using RMSE as a loss function in our approach. GBM model provides predictive accuracy. It's very flexible, can optimize on many loss functions and offers several hyperparameter adjustment possibilities, making the function fit more flexible. It works great with categorical and numerical values as it is and handles missing data very well.

Methods

Data Description

The Boston housing Dataset consists of 506 properties from the 1970 Census with 13 independent variables which are the factors for predicting house price and has the target variable 'mdev' which we will be predicting. Different variables used in our dataset are described below:

1. Independent variable:

Sr. No	Variable	Description	Data Type
1.	Crim	per capita crime rate by town	Numerical
2.	Indus	The proportion of non-retail business acres per town	Numerical
3.	Chas	Charles River dummy variable (=1 if tract bounds river; 0 otherwise)	Numerical

4.	Rm	the average number of rooms per dwelling	Numerical
5.	Zn	The proportion of residential land zoned for lots over 25,000 sq. ft	Numerical
6.	Nox	Nitric oxides concentration (parts per 10 million)	Numerical
7.	Rad	Index of accessibility to radial highways	Numerical
8.	Tax	full-value property-tax rate per USD 10,000	Numerical
9.	Lstat	Percentage of the lower status of the population	Numerical
10.	B	(1000(B – 0.63) ^2) where B is the proportion of blacks by town	Numerical
11.	Ptratio	pupil-teacher ratio by town	Numerical
12.	Dis	weighted distances to five Boston employment centers	Numerical
13.	Age	The proportion of owner-occupied units built before 1940	Numerical

Table 1: Independent Variables

2. Target variable:

Variable	Description	Data Type
medv	The median value of owner-occupied homes in USD 1000's	Numerical

Table 2: Target Variable

Data Preprocessing

There is always a certain level of changes that needs to make on data before applying any Machine Learning model. Here data is being divided into 2 subsets which are the training dataset and testing dataset using a Heuristic partition method. It helps to keep the distribution of data similar in both sub-sets after partition.

The testing dataset is constructed so that we may evaluate how well the model performed on the training dataset. We can see how well the model predicts the values of the target variable in the testing dataset, which is completely unknown to the model. As a result, by determining the variances between predicted and actual values, we may evaluate the correctness of the Gradient Boost model. As a result, it is critical to split the data. The data is being divided as below:

Dataset Type	Description	Partition
Training Dataset	Used to train model	**70%**
Testing Dataset	Used to evaluate trained model	**30%**

Table 3: Splitting of Dataset

Below are some of the steps taken for model accuracy:

1. Scaling of data is being done on all the variables except target so that the variables which have large values as compared to other variables should not dominate the outcome of the model.

2. All independent variables are taken into account for training the GBM model.
3. K-fold Cross Validation is being performed on a training dataset. With the help of k-fold cross-validation, every data point gets tested exactly once and is used in training k-1 times. This is done by dividing training data into k parts(folds) where different combinations are created of train-test partitions to train the model in the best possible way.[2]

Model training and evaluation

After tuning, the GBM model is trained using the best set of parameters which has the lowest RMSE and then validated by running the model on the test dataset. After that evaluation is being done by checking the accuracy of the model which is the RMSE value is increased or decreased. We also check which independent variable has the most impact on the target variable and which variable has the least or no impact. Using this we can also check can we skip any of the independent variables to increase accuracy and reduce computational power.

RESULTS

Gradient boosting is an effective technique for developing predictive models. The concept of boosting arose from the question of whether a weak learner may be transformed to become a better learner. A weak hypothesis (weak learner) is one whose performance is only marginally better than random chance. The aim behind hypothesis boosting was to filter observations, leaving those that the weak learner could manage and focusing on producing new weak learners to address the remaining challenging observations. Gradient boosting involves three elements:

1. A loss function to be optimized.
   It must be differentiable. For example, regression may have a squared error, but classification may have a logarithmic loss. The gradient boosting framework has the advantage of not requiring the development of a new boosting algorithm for each loss function that may be used; instead, it is a generic enough framework that any differentiable loss function can be utilized.[1]

2. A weak learner to make predictions.
   In gradient boosting, decision trees serve as the weak learner. Trees are built in a greedy manner, selecting the best split spots. It is usual to limit the weak learners in certain

ways, such as limiting the number of layers, nodes, splits, or leaf nodes. This is done to guarantee that the learners remain weak but can still be built greedily.[1]

3. An additive model to add weak learners to minimize the loss function.
   Existing trees in the model are not modified, and new trees are inserted one at a time. When adding trees, a gradient descent approach is applied to minimize the loss. The new tree's output is then added to the output of the existing series of trees to fix or improve the model's ultimate output. When the loss no longer improves on an external validation dataset, a specified number of trees are added.[1]

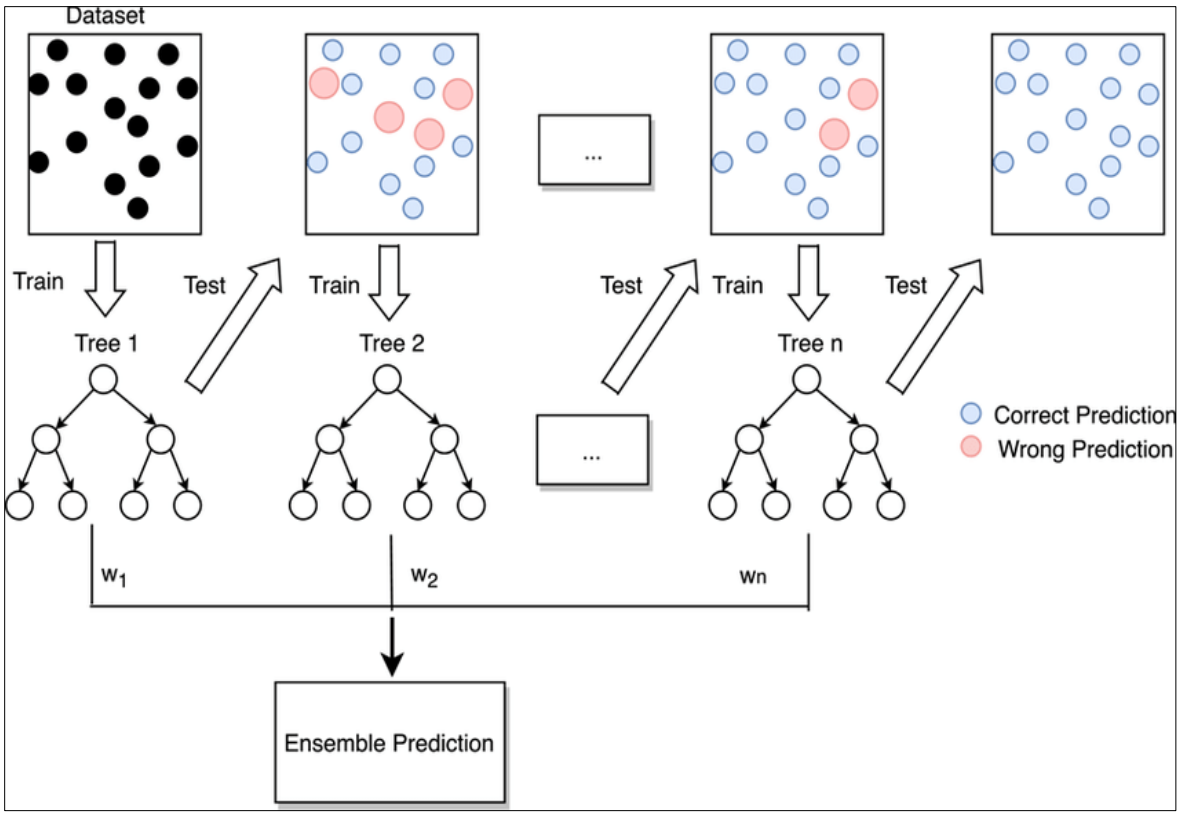

Figure 1[4]: Flow diagram of gradient boosting machine learning method

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. Therefore, tuning the hyperparameters is important because they directly control the behaviour of the training algorithm and have an impact on the performance of the model being trained. The beauty of GBM is that it provides various hyperparameters, mostly used include:

Hyperparameters	Description
Number of trees	The total number of trees to fit. The total number of trees that can be accommodated. GBMs frequently require many trees, therefore the goal is to determine the ideal number of trees that minimize the loss function.[3]

Depth of trees	Integer specifying the maximum depth of each tree (i.e., the highest level of variable interactions allowed) [5]
Learning rate	Controls how quickly the algorithm proceeds down the gradient descent. Smaller values reduce the chance of overfitting but also increases the time to find the optimal fit. This is also called shrinkage.[3]
Max leaf nodes	Integer specifying the minimum number of observations in the terminal nodes of the trees.[5]

Table 4: Tuning Parameters description

For predicting the housing price which is our target variable 'medv' we need to find the best combination of hyperparameters by trying out different combinations. And identify the best set of combinations of hyperparameters after tuning using the caret grid technique by running the model on a training dataset. We finalize the best parameters on basis of the RMSE metric. After running the model, we get different RMSE for a different combination of parameters. **We choose the parameters for which RMSE value is lowest** which is the best fit for our model.

As RMSE (Root Mean Squared Error) is the square root of the mean of the difference between predicted and observed values.

After tuning parameters, we need to specify the type of resampling which is k-fold cross-validation in our approach. Using resampling is important as it removes the chances of overfitting the model. The steps taken for tuning the model are as follows:

1. GBM model is tuned by setting the hyperparameters are as follows:
    a. Number of trees(n.trees) : 0 to 40 by 50
    b. Depth of trees(interaction.depth): 1,3,4,6
    c. Learning rate(shrinkage): 0.01, 0.03
    d. Max leaf nodes(n.minobsinnode): 2
2. K-fold cross-validation is used for resampling where k=10.
3. We get the best parameters after running the 328 combinations of hyperparameters defined by us.
4. The parameters which are the best fit for our model are as below:

Number of trees	Interaction Depth	Learning rate	Max leaf nodes	RMSE	R squared	MAE	RMSE-SD	R squared-SD	MAE-SD
800	6	0.03	2	3.1746	0.8862	2.1759	0.8075	0.0639	0.4007

Table 5: Best fit Model Parameters

We can observe from the above ggplot that as the number of Boosting iterations increases the value of RMSE decreases for all values of depth of trees. The RMSE is lowest when interaction depth is 6 for learning rate 0.03 and number of trees is 800 which are the same results that we got in the above table. Here we can see why we have chosen the learning rate as 0.03 because the RMSE is lowest for 0.03 than 0.01 which is evident from the below Figure 2.

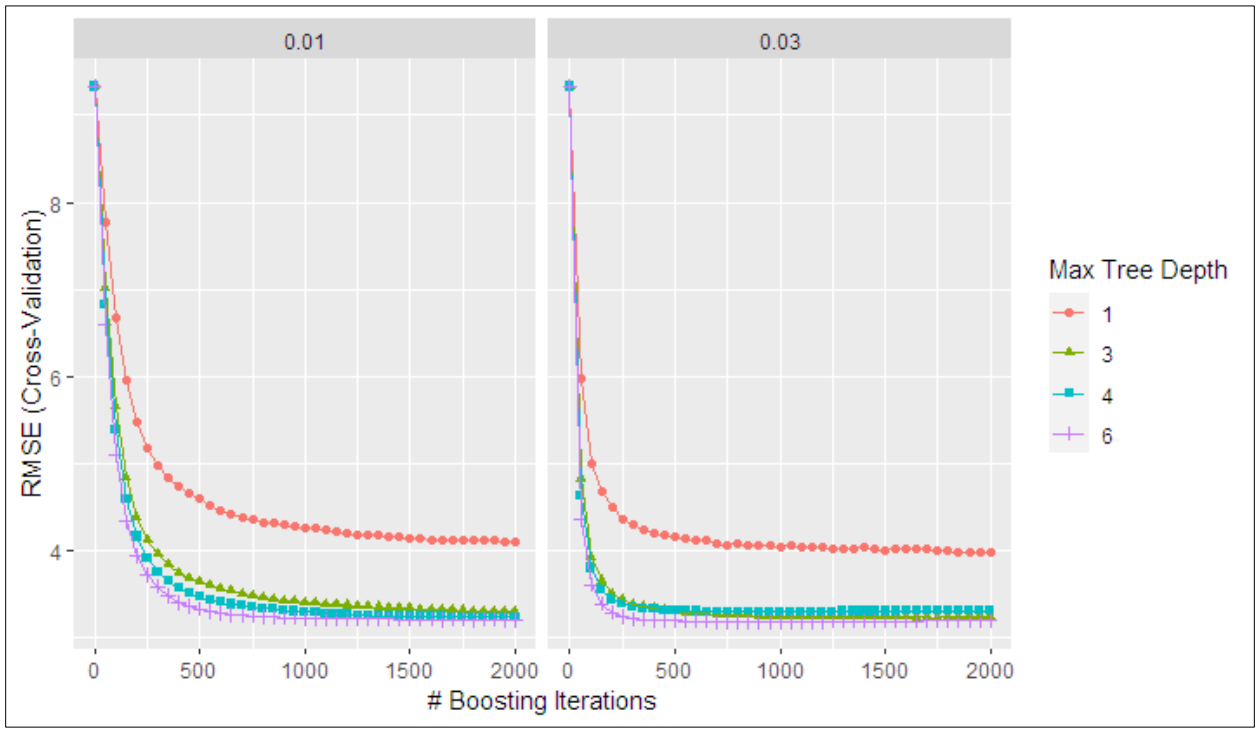

Figure 2: ggplot of GBM model tuning

By tuning the hyperparameters carefully we can increase or decrease the combinations which can impact the execution time, but if RMSE is consistent enough then we can try a different combination and save execution time. After tuning the model correctly on training data, we need to check how the model performs on the testing dataset which we haven't used yet.

Model Evaluation

After the model tuning, evaluation of the model is done on a testing dataset to check the accuracy of the model, or how well our model has been tuned. As the testing dataset is new for a model as it was separated from the main dataset in data preprocessing stage, now the model will work on a completely new dataset which will give us a fair idea of how the model will perform in the real-world test cases or on an unknown dataset. As we can see from the graph below and evaluate how well our model has performed on the testing dataset where predicted values against observed values have been plotted for the GBM model. The red lines indicate the variation in predicted and observed values, and the blue lines indicate how well our model is

fitting. As visible from the Figure 3 we can say that our model fits considerably good as almost all the data points of predicted Vs observed are near the blue line which indicates the model fit. The red line aligns closely with the blue lines which tell us that the variation between predicted and observed values are lowest as it runs along with the model fitting line(blue).

The RMSE value for a testing dataset is even lower than of the training dataset which is again an indication of the model performing well.

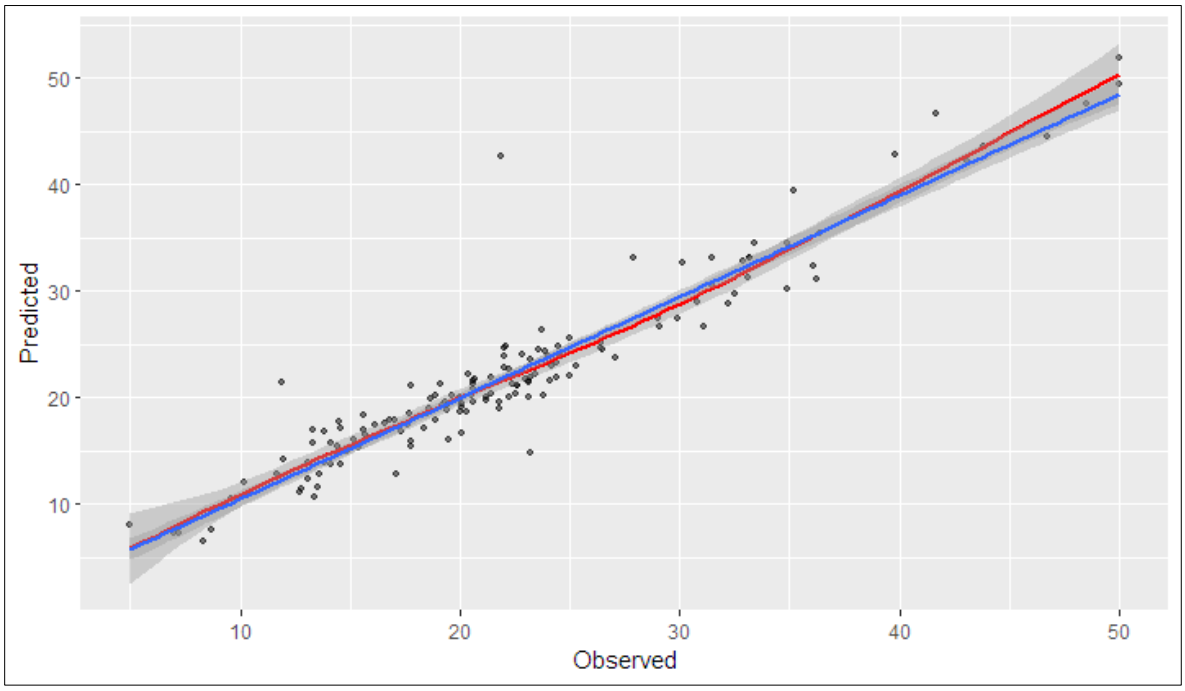

Figure 3: Graph of Predicted V/s Observed values of Target Variable

	RMSE	R-squared	MAE
Values	2.77	0.9	1.7786

Table 6: Accuracy measure on Testing Dataset

At last, we find out which independent variable impacts most for any change in target variable and which impacts less. This helps us in reducing dimensionality and execution time if we can eliminate any variable which has negligible impact on our target variable or very less impact when there are a greater number of variables that are having a significant impact. As we can observe from the above graph the 'lstat' variable has the most impact on the target variable which says that percentage of the lower status of the population impacts most on house prices, followed by 'rm' which is an average number of rooms per house. And the least impactful independent variable is 'zn' which is the proportion of residential land zoned for lots over 25,000 sq. ft.

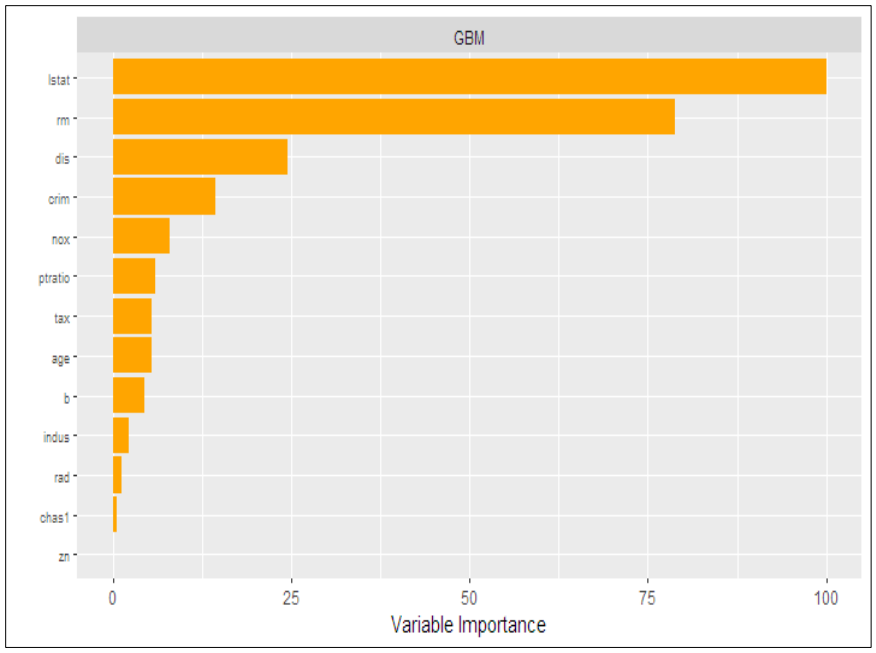

Fig 4: Varibale Importance

GBM variables	Importance Percentage
lstat	100.00
rm	78.86
dis	24.45
crim	14.29
nox	7.88
ptratio	5.93
tax	5.38
age	5.34
b	4.39
indus	2.17
rad	1.11
chas1	0.51
zn	0.00

Table 7: Impact percentage(scaled)

Discussion

Evaluating the results

We are using K-fold cross-validation for resampling to avoid overfitting the model. It is generally used when we have limited data in the machine learning model. It is initially used in machine learning to estimate the skill of Machine Learning models on the testing dataset. Here we have used k=10 which are the number of groups created on a training dataset. It has been found through experiments that using k=10 results in estimating model skill with low bias and modest variance.[2]

After that, we tune the model using a set of hyperparameters. It is very important to select the best combination of hyperparameters because it makes a huge impact on our learned model. If we can optimize it perfectly then we can make the model shine. This is a trial and error method to choose the best hyperparameters. For finding the best model parameters from hyperparameters we use the Grid Search technique which runs all combinations of hyperparameters on training data to find the best set of model parameters. We got the best set of model parameters on basis of the lowest value of RMSE which is 3.1746. After getting the best model parameters we again train the model using the best set of parameters.

After a model is tuned we evaluate the result by applying the best fit model on the testing dataset. As we can see from figure 3 how our model has performed on the testing dataset. As we can see there is no significant level of difference between observed and predicted values as the model fit line (blue line) runs very close to variation(line) as almost all the data points are near the model fit line. We can see one

extreme outlier, which we cant eliminate as it's a part of our dataset with believable values therefore we can call it a natural outlier. As we can see the variation line is going far from the model fit line at the end which can be due to not enough data in the last quantile of the target variable in comparison to the first 3 quantiles. 380 observations(out of 506) of target variable ranges from 0 to 25 thousand dollars which is a good amount of data for training the dataset but we have only 125 observations which range between 25 to 50 thousand dollars.

At last, we find the impact of our independent variables on the target variable which can be observed from Figure 4 and Table 6. We can see the most impactful variable is "lstat" followed by "rm" and "dis". From this, we can say that:

1. The percentage of the lower status of the population impacts most on our target variable. As we can say that the status of the people living in that area drives the house price of that area.
2. The least impactful variable is "zn" which is the proportion of residential land zoned for lots over 25,000 sq. ft. This result justifies our 1st point as more people with lower status will buy or have houses of smaller size.
3. The second most impactful variable is "rm" which is the number of rooms in a house or property.

As we can observe from Figure 4 and Table 6 that "zn" has the least impact which is 0% after scaling and "chas1" with 0.5%. After seeing this we can remove these columns and try running the model again and still our RMSE will be approximately the same as these variables have negligible impact on our target variable. By removing the low impact variables we can reduce the computation power and execution time.

We can also say that due to limited data points our impact variables can be different in a real-world scenario. Maybe the variables can change if we have got the data points in multiple thousands.

Limitations

1. We should use the value of k which can divide the data into groups with an equal amount of data points. As I have chosen k=10 which is not a good factor of 506 data points therefore anyone fold will have unequal data points.[2]
2. We don't have high computing machines to run a large set of combinations. If more iterations would have been performed there is a possibility to find better tuning parameters that can reduce our execution time and our model would have been more accurate.
3. If the distribution of data would have been equal among our dataset then our model fit would have been more accurate as the variation line(red) will have been more aligned to the model fit line(blue) at the end.
4. There are certain limitations to the GBM model as well which are:
    a. It is more sensitive to overfitting if the data is noisy.
    b. Training generally takes longer because trees are built sequentially.
    c. These are harder to tune as there are more hyperparameters.

d. GBM model is considered more resource consuming that its competitor model.

5. This model will not be a for predicting home prices in today's day and age as the factors impacting home prices will be different in 2022 as the dataset is from 1970.

Future Scope

1.	In Future, we can get more independent variables for the prediction of house prices. We can remove the variables which are not relevant and less impactful.
2.	We can try different machine learning models such as k-means clustering, Support vector machines, etc and after comparing the accuracy we can decide which model to select.
3.	Furthermore, each model uniquely determines variable importance. A different variable might be at the top of the list of factors influencing house prices in another model.
4.	We can assess the model based on different parameters rather than RMSE such as R square or MAE.

References

1. Brownlee, Jason. “A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning.” Machine Learning Mastery, (20. 11. 2018).
2. Brownlee, Jason. “A Gentle Introduction to K-Fold Cross-Validation.” Machine Learning Mastery, (21. 05. 2018).
3. University of Cincinati and Bradley Boehmke. “Gradient Boosting Machines · UC Business Analytics R Programming Guide.” Github.io,(2017).
4. Zhang, Tao & Lin, Wuyin & Vogelmann, Andrew & Zhang, Minghua & Xie, Shaocheng & Qin, Yi & Golaz, Jean-Christophe. (2021). Improving Convection Trigger Functions in Deep Convective Parameterization Schemes Using Machine Learning. Journal of Advances in Modeling Earth Systems. (13. 10. 2020).
5. Ridgeway, Greg. Package “Gbm.” (15. 07. 2020).