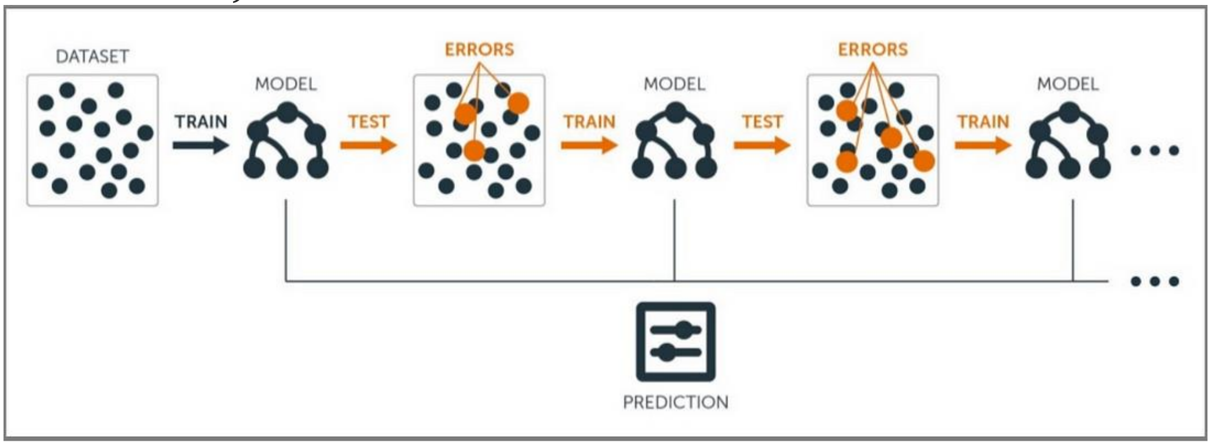
Introduction to the Gradient

The Boosting Algorithm is one of the most powerful learning ideas introduced in the last twenty years. Gradient Boosting is an supervised machine learning algorithm used for classification and regression problems. It is an ensemble technique which uses multiple weak learners to produce a strong model for regression and classification.

Intuition

Gradient Boosting relies on the intuition that the best possible next model , when combined with the previous models, minimizes the overall prediction errors. The key idea is to set the target outcomes from the previous models to the next model in order to minimize the errors. This is another boosting algorithm(few others are Adaboost, XGBoost etc.).



Input requirement for Gradient Boosting:

1. A Loss Function to optimize.

2. A weak learner to make prediction (Generally Decision tree).
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function

The loss function basically tells how my algorithm, models the data set.

It is difference between actual values and predicted values.

Regression Loss functions:

1. L1 loss or Mean Absolute Errors (MAE)
2. L2 Loss or Mean Square Error(MSE)
3. Quadratic Loss

Binary Classification Loss Functions:

1. Binary Cross Entropy Loss
2. Hinge Loss

A gradient descent procedure is used to minimize the loss when adding trees.

2. Weak Learner

Weak learners are the models which is used sequentially to reduce the error generated from the previous models and to return a strong model on the end.

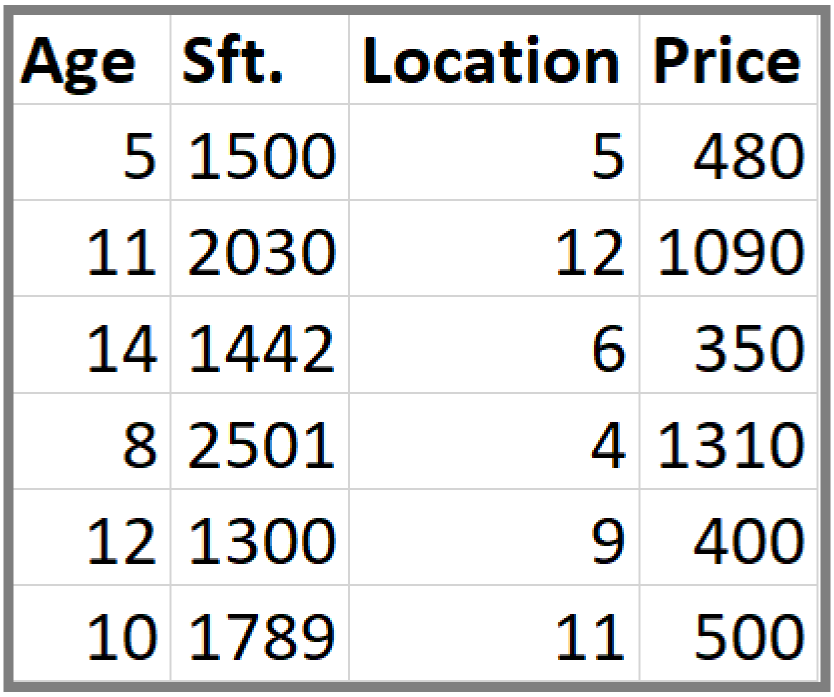
Decision trees are used as weak learner in gradient boosting algorithm.

3. Additive Model

In gradient boosting, decision trees are added one at a time (in sequence), and existing trees in the model are not changed.

Understanding Gradient Boosting Step by Step :

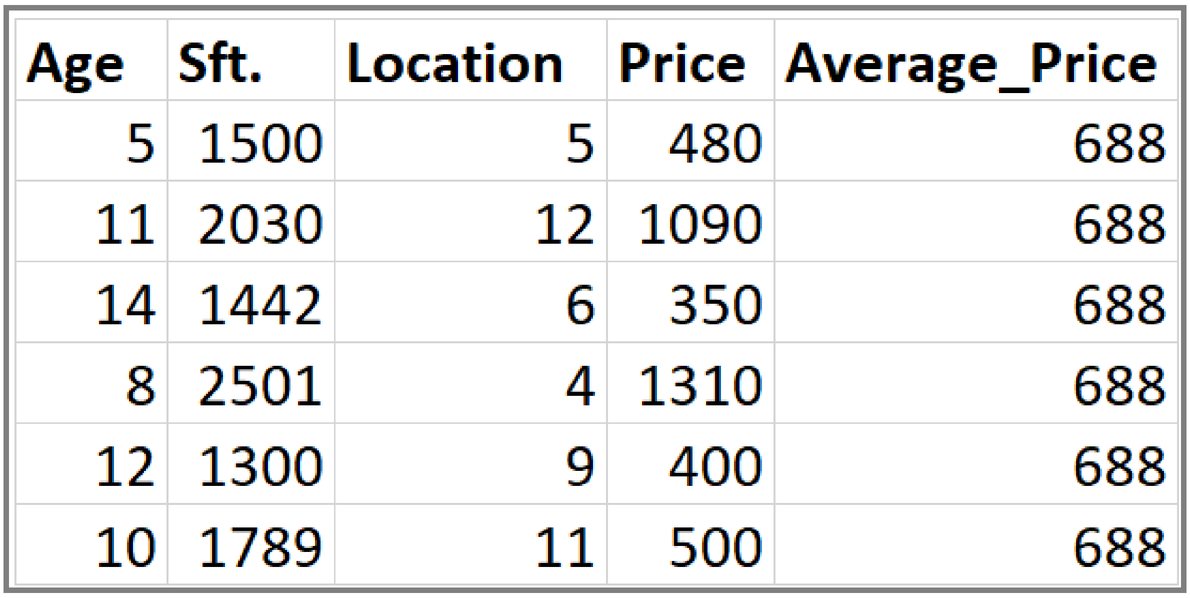
This is our data set. Here Age, Sft., Location is independent variables and Price is dependent variable or Target variable.



Age	Sft.	Location	Price
5	1500	5	480
11	2030	12	1090
14	1442	6	350
8	2501	4	1310
12	1300	9	400
10	1789	11	500

Step 1: Calculate the average/mean of the target variable.

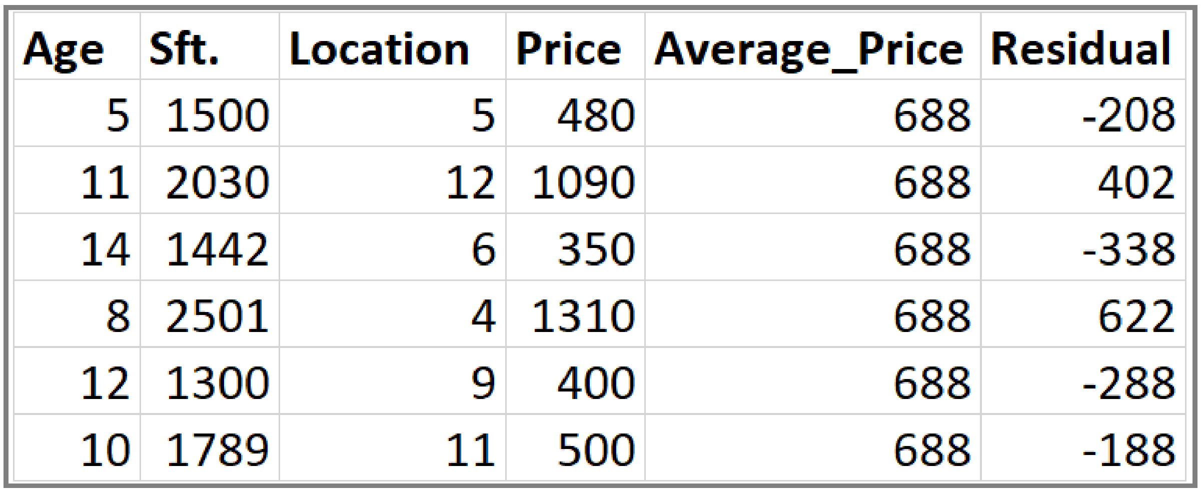
$$\frac{480+1090+350+1310+400+500}{6} = 688$$



Age	Sft.	Location	Price	Average_Price
5	1500	5	480	688
11	2030	12	1090	688
14	1442	6	350	688
8	2501	4	1310	688
12	1300	9	400	688
10	1789	11	500	688

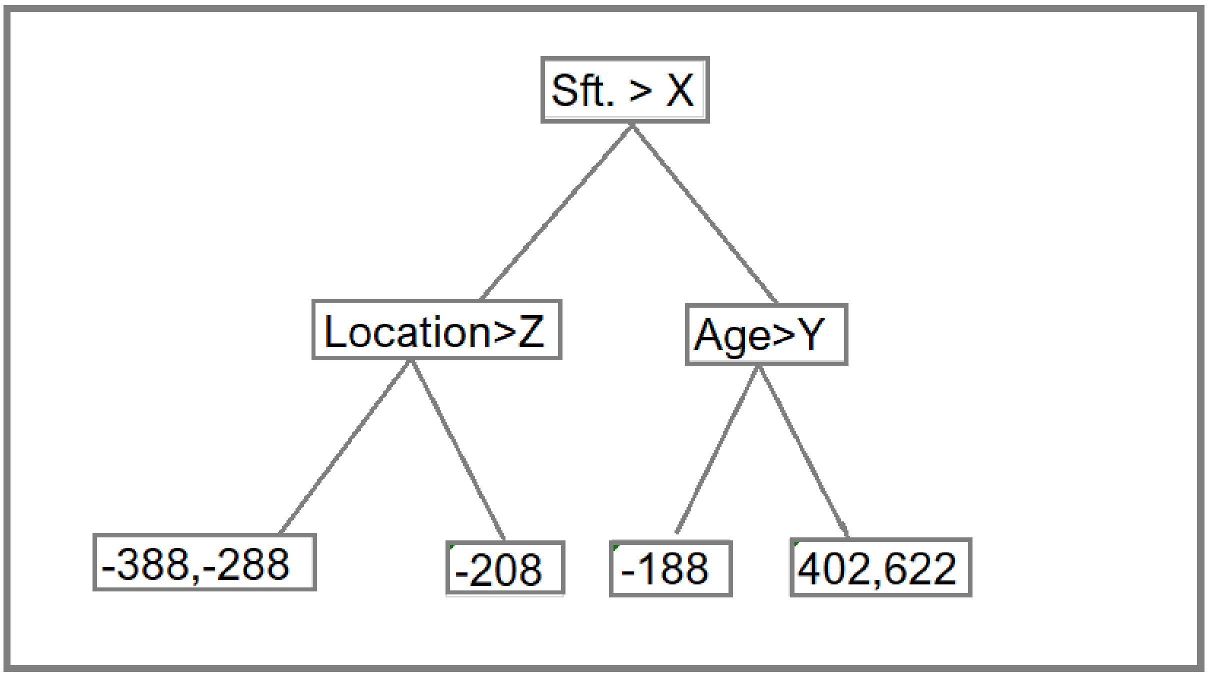
Step 2: Calculate the residuals for each sample.

Residual=Actual Value - Predicted Value



Age	Sft.	Location	Price	Average_Price	Residual
5	1500	5	480	688	-208
11	2030	12	1090	688	402
14	1442	6	350	688	-338
8	2501	4	1310	688	622
12	1300	9	400	688	-288
10	1789	11	500	688	-188

Step 3: Construct a decision tree. We build a tree with the goal of predicting the Residuals.

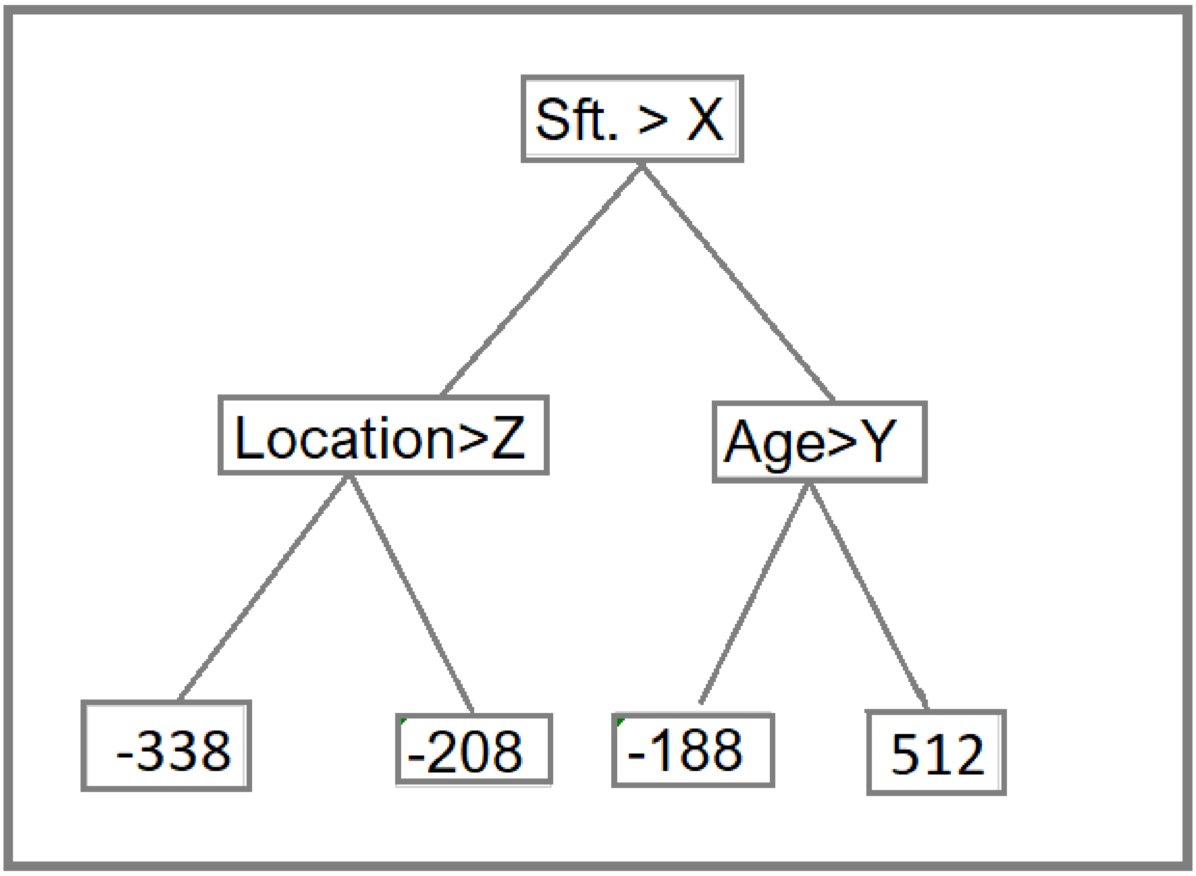


In the event if there are more residuals then leaf nodes(here its 6 residuals),some residuals will end up inside the same leaf. When this happens, we compute their average and place that inside the leaf.

$$\frac{(-388)+(-288)}{2} = -338$$

$$\frac{402+622}{2} = 512$$

After this tree become like this.



Step 4: Predict the target label using all the trees within the ensemble.

Each sample passes through the decision nodes of the newly formed tree until it reaches a given lead. The residual in the said leaf is used to predict the house price.

=>Average Price + Learning Rate * Residual Predicted by decision Tree

=>688 + 0.1 * (-338)

Predicted Price => 654.2

=> 688 + 0.1 * (-208)

Predicted Price=> 667.2

Calculation above for Residual value (-338) and (-208) in Step 2

Same way we will calculate the **Predicted Price** for other values

Note: We have initially taken 0.1 as learning rate.

Step 5 : Compute the new residuals

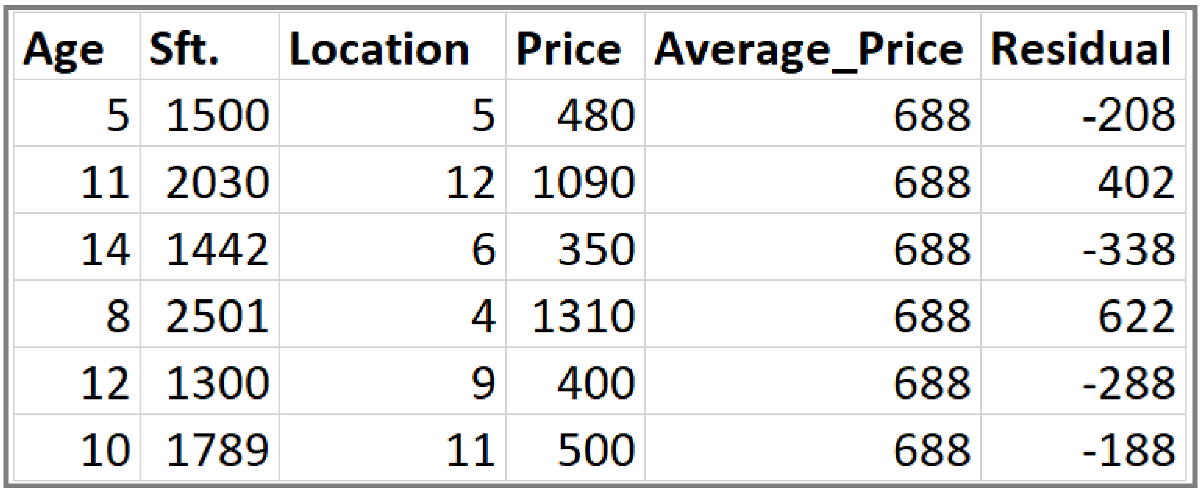
Residual=Actual Value - Predicted Value

=>350-654.2 = -304.2

=>480-667.2 = -187.2

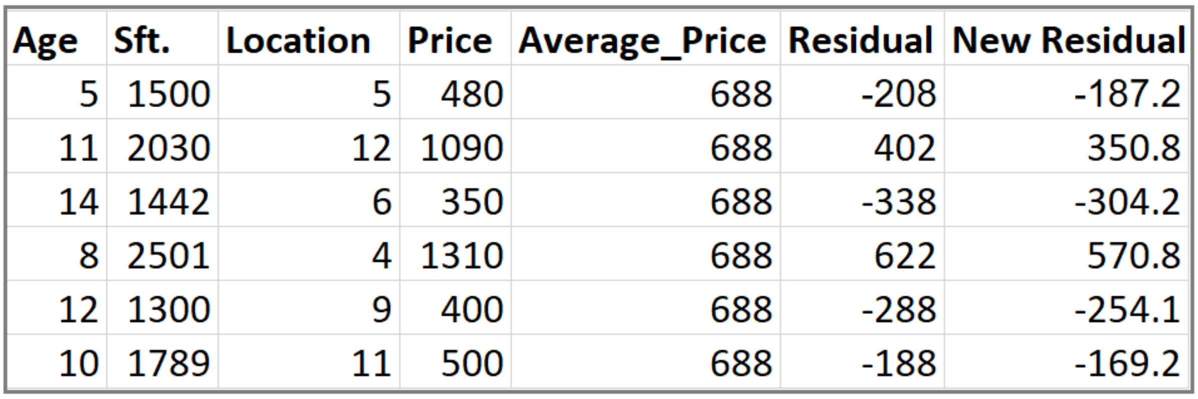
When Price is 350 and 480 Respectively.

With our Single leaf with average value**(688)** we have the below column of Residual.



Age	Sft.	Location	Price	Average_Price	Residual
5	1500	5	480	688	-208
11	2030	12	1090	688	402
14	1442	6	350	688	-338
8	2501	4	1310	688	622
12	1300	9	400	688	-288
10	1789	11	500	688	-188

With our decision tree ,we ended up the below new residuals.



Age	Sft.	Location	Price	Average_Price	Residual	New Residual
5	1500	5	480	688	-208	-187.2
11	2030	12	1090	688	402	350.8
14	1442	6	350	688	-338	-304.2
8	2501	4	1310	688	622	570.8
12	1300	9	400	688	-288	-254.1
10	1789	11	500	688	-188	-169.2

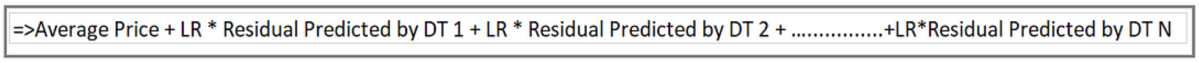
Step 6: Repeat steps 3 to 5 until the number of iterations matches the number specified by the hyper parameter(numbers of estimators)

Step 7: Once trained, use all of the trees in the ensemble to make a final prediction as to value of the target variable. The final prediction will be equal to the mean we computed in Step 1 plus all the residuals predicted by the trees that make up the forest multiplied by the learning rate.

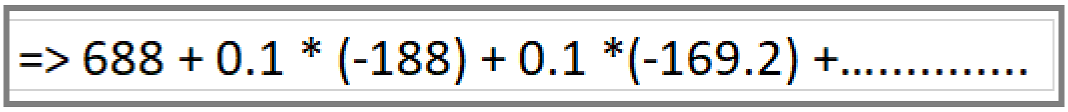
Here,

LR : Learning Rate

DT: Decision Tree



=>Average Price + LR * Residual Predicted by DT 1 + LR * Residual Predicted by DT 2 + ...+LR*Residual Predicted by DT N



=> 688 + 0.1 * (-188) + 0.1 *(-169.2) +.............

Advantages of Gradient Boosting

1. Most of the time predictive accuracy of gradient boosting algorithm on higher side.
2. It provides lots of flexibility and can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible.
3. Most of the time no data pre-processing required.
4. Gradient Boosting algorithm works great with categorical and numerical data.
5. Handles missing data — missing value imputation not required.

Disadvantages of Gradient Boosting

1. Gradient Boosting Models will continue improving to minimize all errors. This can overemphasize outliers and cause over fitting. Must use cross-validation to neutralize.

2. It is computationally very expensive — GBMs often require many trees (>1000) which can be time and memory exhaustive.
3. The high flexibility results in many parameters that interact and influence heavily the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.