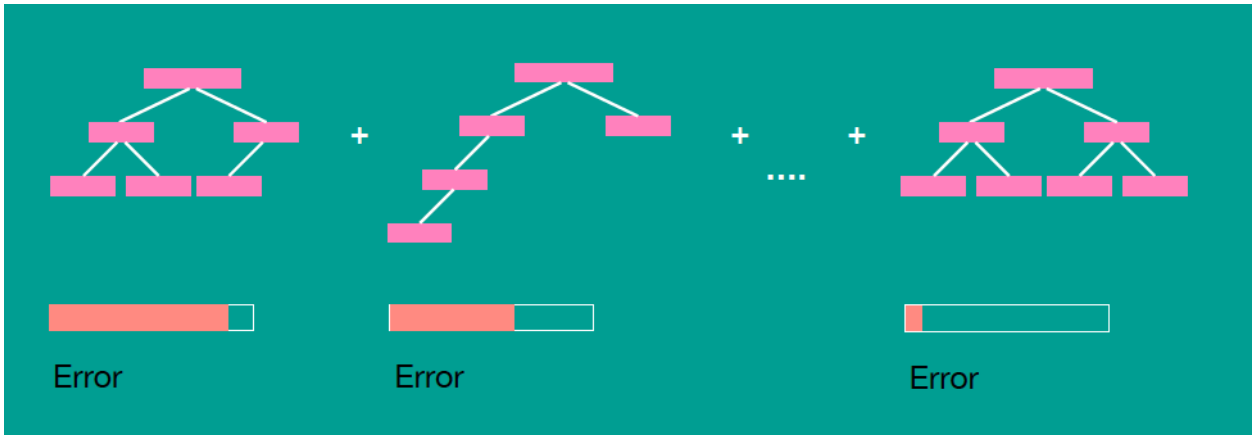
XGBOOST

Extreme Gradient Boosting belongs to a family of boosting algorithms and uses the gradient boosting (GBM) framework. supports both regression and classification predictive modelling problems.

It is called extreme gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. it is similar to gradient boosting framework but more efficient. So what makes it fast is its capacity to do parallel computation on a single machine. This makes XGBoost at least 10 times faster than existing gradient boosting implementations.

XGBoost is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of model performance and computational speed. Specifically, it was engineered to exploit every bit of memory and hardware resources for tree boosting algorithms



- Can be run on both single and distributed systems (Hadoop, Spark).

- Supports parallel processing.

- Cache optimization.

- Has a variety of regularizations which helps in reducing overfitting.

- Auto tree pruning i.e. decision tree will not grow further after certain limits internally.

- Can handle missing values.

- Has Inbuilt Cross-Validation.

- Takes care of outliers to some extent.

HOW IT WORKS

To begin training, a single decision tree is built to predict the label. This first decision tree will predict some instances but will fail for other instances. To improve the model, we can build another decision tree, but this time try to predict the residuals instead of the original labels. This can be thought of as building another model to correct for the error in the current model. After adding the new tree to the model, make new predictions and then calculate residuals again. In order to make predictions with multiple trees, simply pass the given instance through every tree and sum up the predictions from each tree. By building predictors that estimate the residual, we are actually minimizing the gradient of the squared error between the real and predicted labels

How it works in Classification

1. Assign equal weights to all observations. The weight assigned to a data point is the probability that a particular data point will be selected to be fed to the model.

2. Then it uses the Bagging (Bootstrap Aggregating) algorithm to create random samples. Given a data set D1 (m rows and p columns), it creates a new dataset D2(n rows and p columns) by row sampling at random with replacement from the original data where m>n.

3. First weak classifier is made and the data points which are predicted wrongly are now assigned higher weights.

4. Now we again create random samples. But the data points which were predicted wrongly by last week classifier has higher weights and thus have higher chances of getting selected to be fed to next weak classifier. The next weak classifier will work to predict it rightly.

5. Several weak classifiers are grown and the final prediction is obtained by voting.

How it works in regression

1. Fit a decision tree regressor on data.

2. Calculate error residuals which is the difference between actual target value and predicted target value

3. Fit a new model on error residuals as target variable with same input variables

4. Add the predicted residuals multiplied by learning rate to the previous predictions i.e. y_predicted2 = y_predicted1 + learning_rate * e1_predicted

5. Repeat steps 2 to 4 until the number of iterations matches the number specified by the hyperparameter i.e. number of estimators

Example

Suppose, we were trying to predict the price of a house given their age, square footage and location.

age	square footage	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 of the target label

Let's start with a leaf that is the average value of the variable we want to predict. This leaf will be used as a baseline to approach the correct solution in the proceeding steps.

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

Step 2: Calculate the residuals

For every sample, we calculate the residual with the proceeding formula.

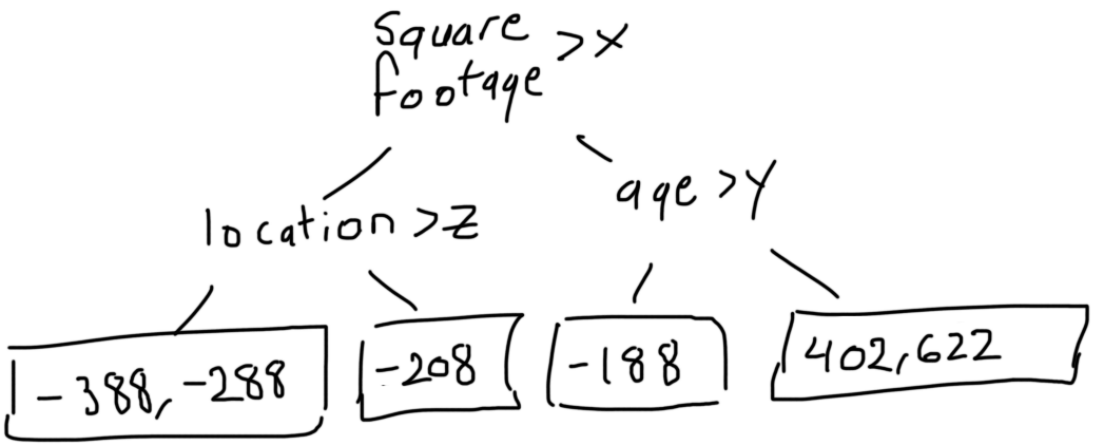
residual = actual value – predicted value

In our example, the predicted value is the equal to the mean calculated in the previous step and the actual value can be found in the price column of each sample. After computing the residuals, we get the following table.

age	square footage	location	price	residuals
5	1500	5	480	-208
11	2030	12	1090	402
14	1442	6	350	-338
8	2501	4	1310	622
12	1300	9	400	-288
10	1789	11	500	-188

Step 3: Construct a decision tree

Next, we build a tree with the goal of predicting the residuals. In other words, every leaf will contain a prediction as to the value of the residual (not the desired label).

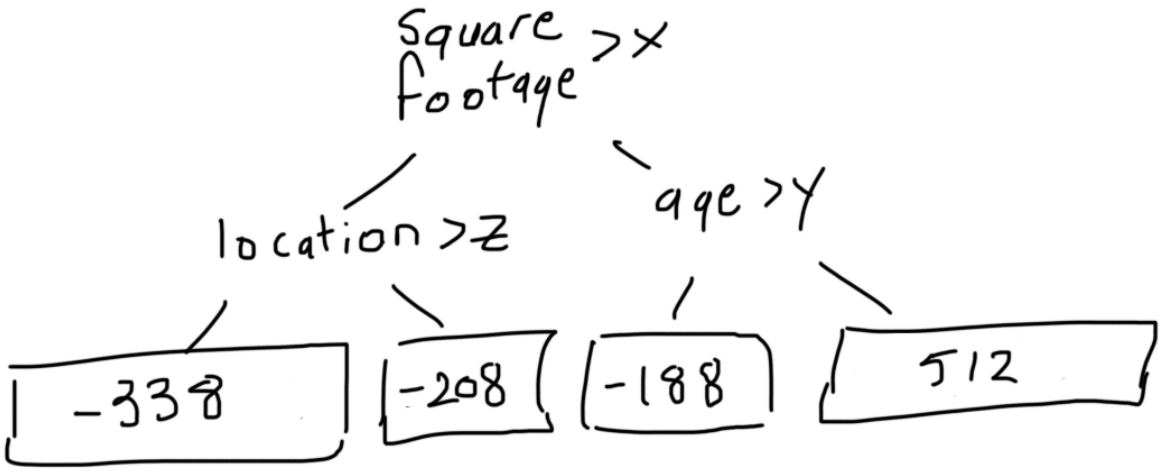


In the event there are more residuals than leaves, some residuals will end up inside the same leaf. When this happens, we compute their average and place that inside the leaf.

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

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

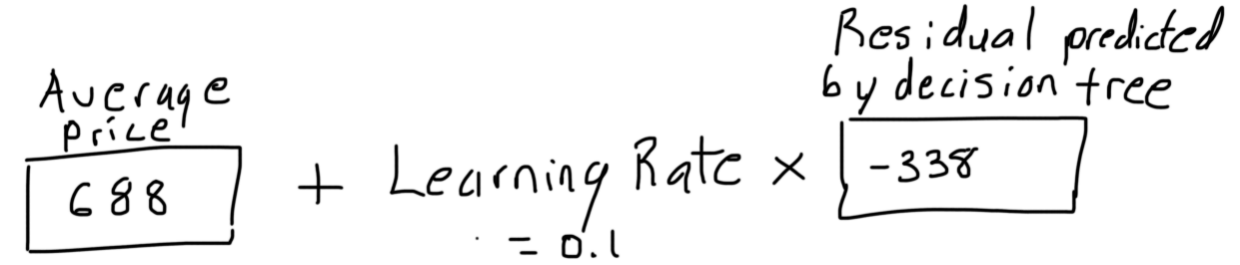
Thus, the tree becomes



Step 4: Predict the target label using all of 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 said leaf is used to predict the house price.

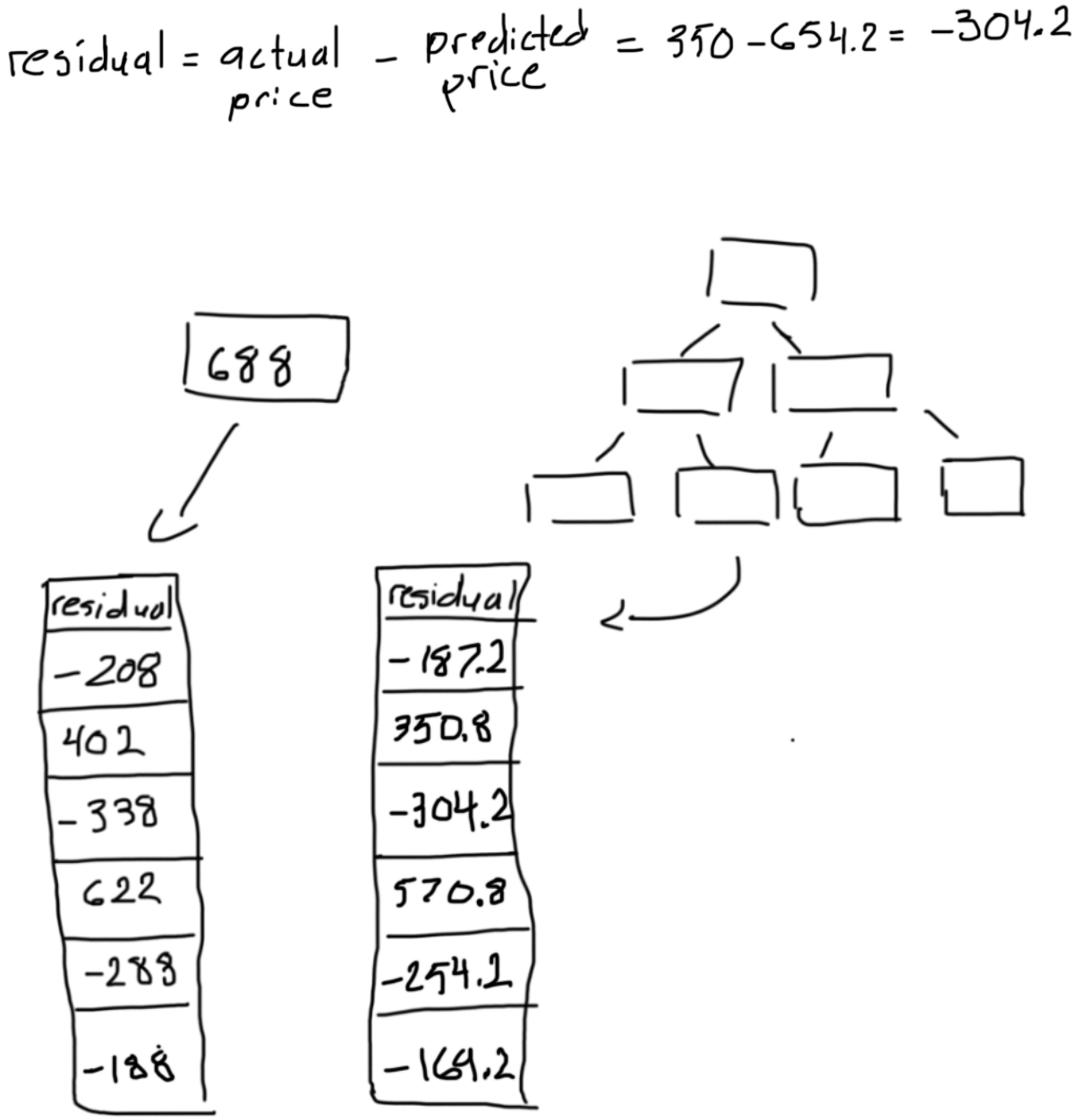
It's been shown through experimentation that taking small incremental steps towards the solution achieves a comparable bias with a lower overall variance (a lower variance leads to better accuracy on samples outside of the training data). Thus, to prevent overfitting, we introduce a hyperparameter called learning rate. When we make a prediction, each residual is multiplied by the learning rate. This forces us to use more decision trees, each taking a small step towards the final solution.



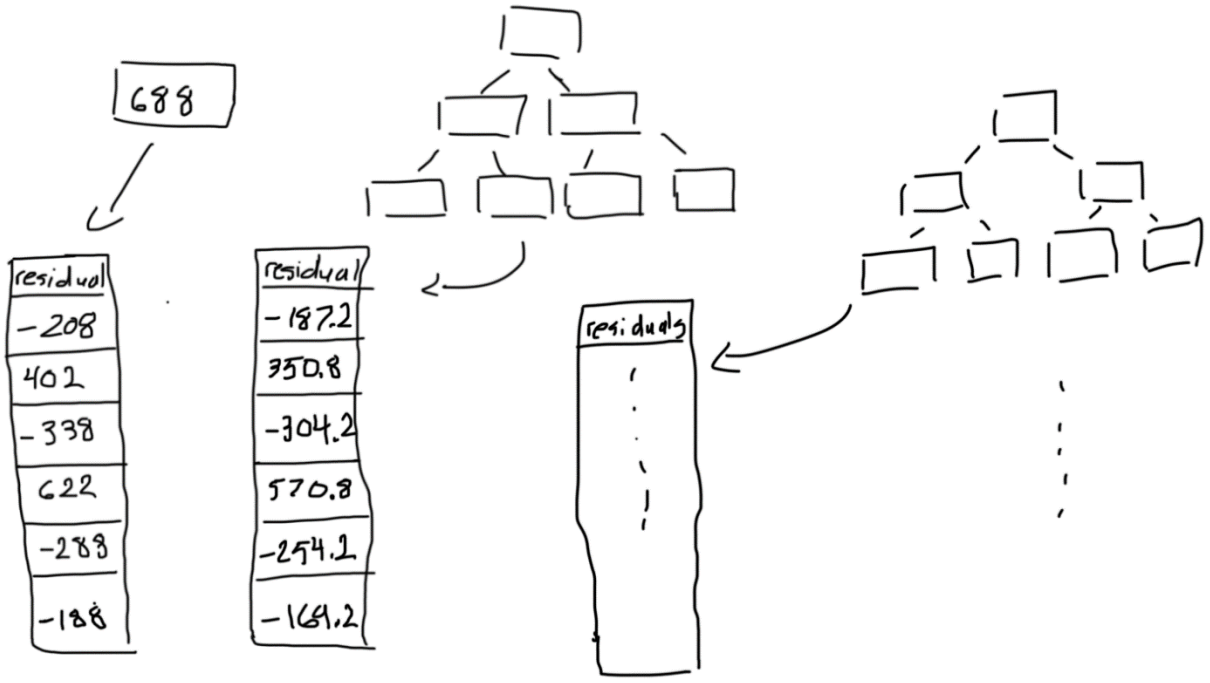
$$\text{predicted price} = 688 + 0.1 \times -338 = 654.2$$

Step 5: Compute the new residuals

We calculate a new set of residuals by subtracting the actual house prices from the predictions made in the previous step. The residuals will then be used for the leaves of the next decision tree as described in step 3.



Step 6: Repeat steps 3 to 5 until the number of iterations matches the number specified by the hyperparameter i.e. number of estimators



Step 7: Once trained, use all of the trees in the ensemble to make a final prediction as to the value of the target variable

The final prediction will be equal to the mean we computed in the first step, plus all of the residuals predicted by the trees that make up the forest multiplied by the learning rate.

Average price: 688 + Learning Rate = 0.1 × Residual predicted by decision tree: -188 + learning Rate = 0.1 × Residual predicted by decision tree: -169.2 + ...