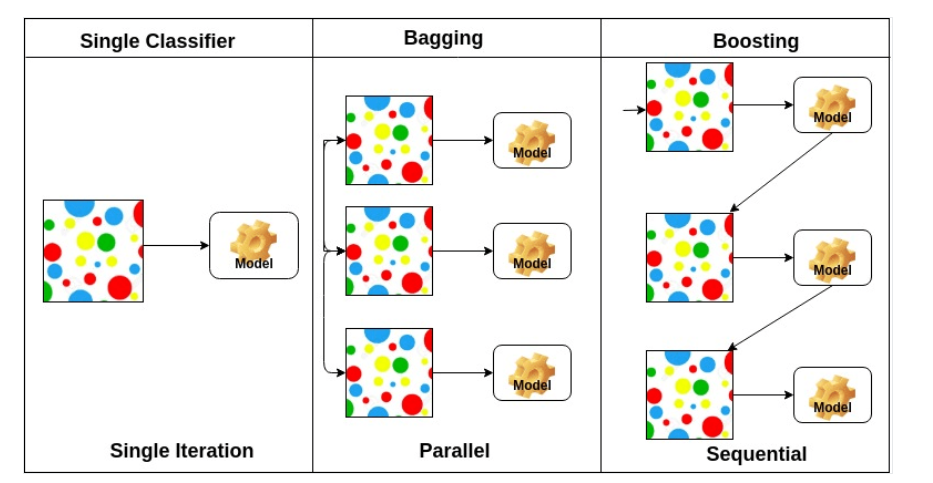
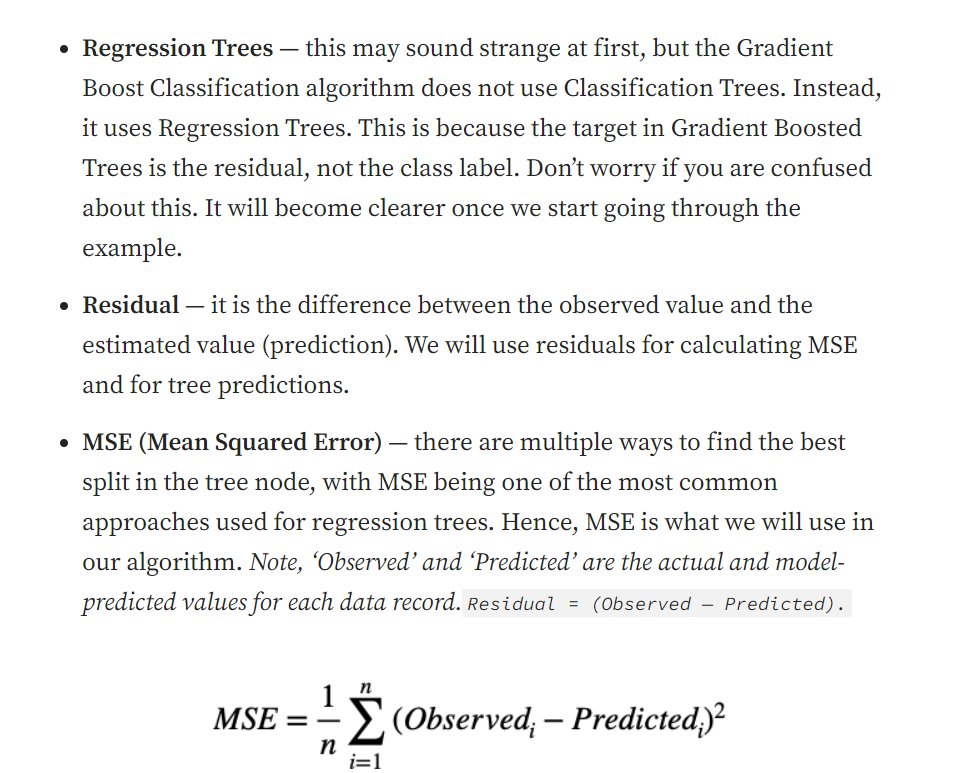
***Gradient boost***

In random forests, the results of decision trees are aggregated at the end of the process, **Parallel**. Gradient boosting doesn't do this and instead aggregates the results of each decision tree along the way to calculate the final result, **Sequential**.



Above:

* 1. decision tree single
  2. Random forest
  3. Gradient boosting.



* Just as humans learn from their mistakes and try not to repeat them further in life, the **Boosting**algorithm tries to build a strong learner (predictive model) from the mistakes of several weaker models. You start by creating a model from the training data. Then, you create a second model from the previous one by trying to reduce the errors from the previous model. Models are added sequentially, each correcting its predecessor, until the training data is predicted perfectly or the maximum number of models have been added.
* Boosting basically tries to reduce the bias error which arises when models are not able to identify relevant trends in the data. This happens by evaluating the difference between the predicted value and the actual value.

.**How Gradient Boosting Works**

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

### 1. Loss Function

The loss function used depends on the type of problem being solved. It must be differentiable, but many standard loss functions are supported and you can define your own. For example, regression may use a squared error and classification may use logarithmic loss.

### 2. Weak Learner (Learning rate)

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

Specifically, regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions.

Trees are constructed in a greedy manner, choosing the best split points based on minimize the loss.

Alpha(α) represents the learning rate. The value of alpha(α) varies between 0 to 1 let alpha(α)=0.1. Learning rate parameter in gradient boosting is used to control the contribution of each weak learner which are added in series.

### Weighted Updates

The predictions of each tree are added together sequentially.

**The contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. This weighting is called a shrinkage or a learning rate.**

### 3) Additive Model

Trees are added one at a time, and existing trees in the model are not changed.

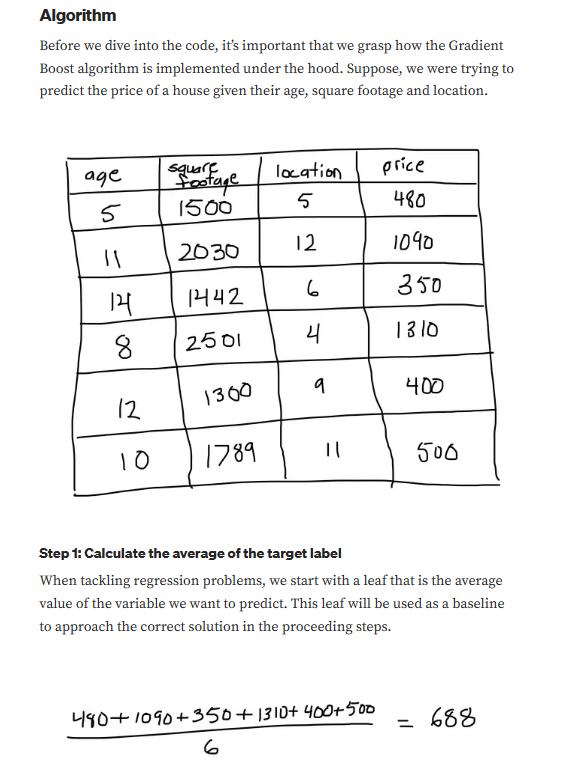
A gradient descent procedure is used to minimize the loss when adding trees.

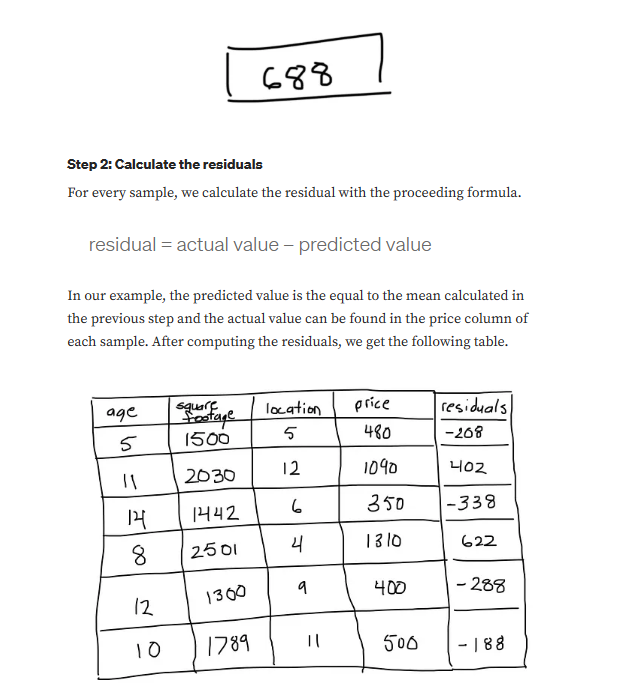
Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.

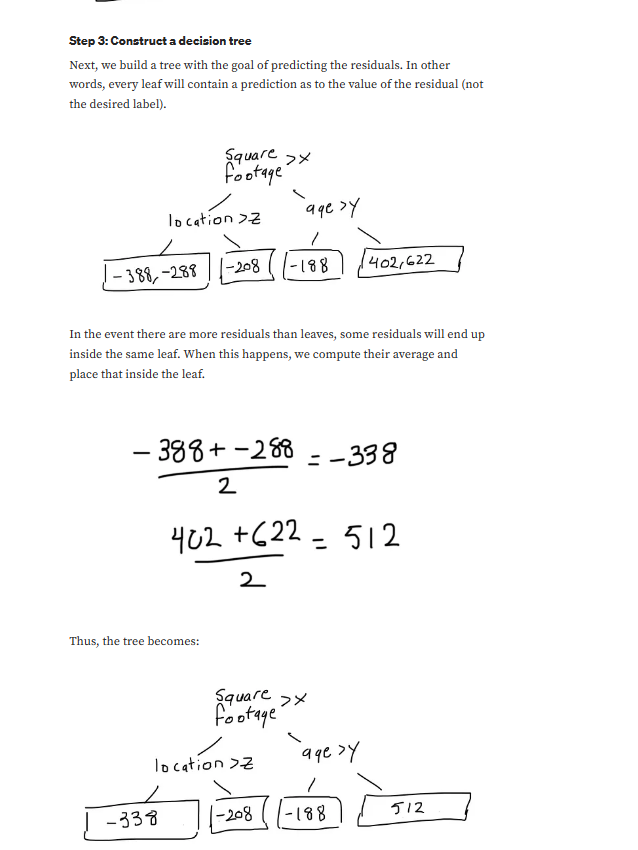
Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by (reducing the residual loss.

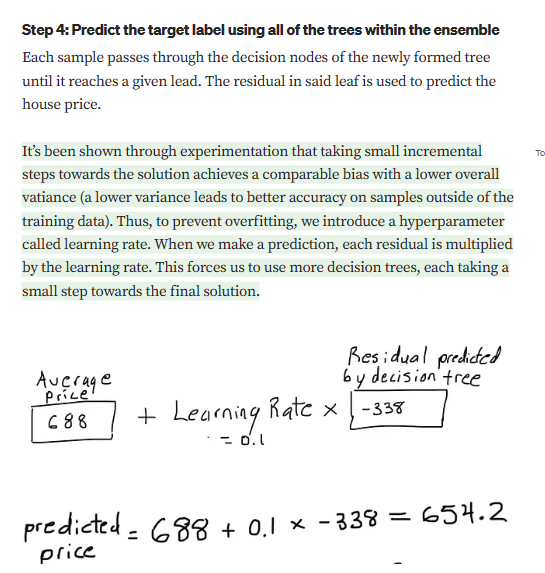
**Generally, this approach is called functional gradient descent**

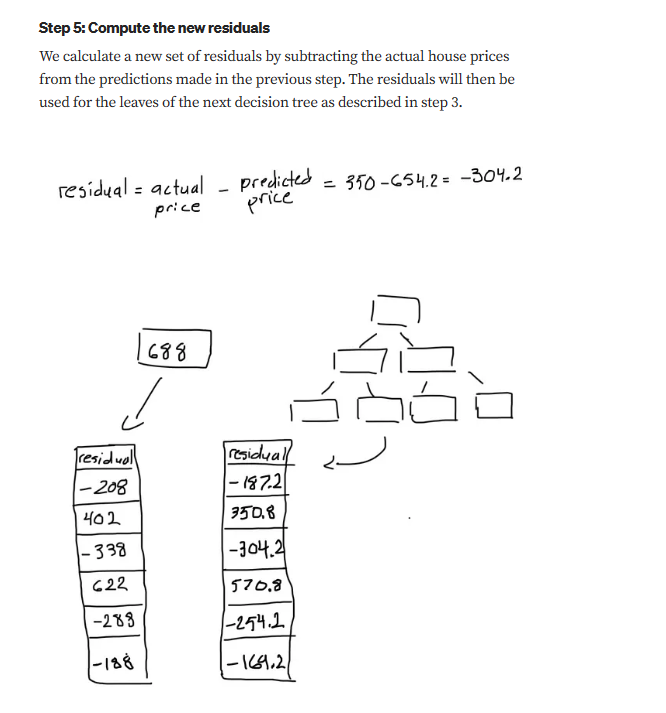
**Example:**

****

****

****

****

****

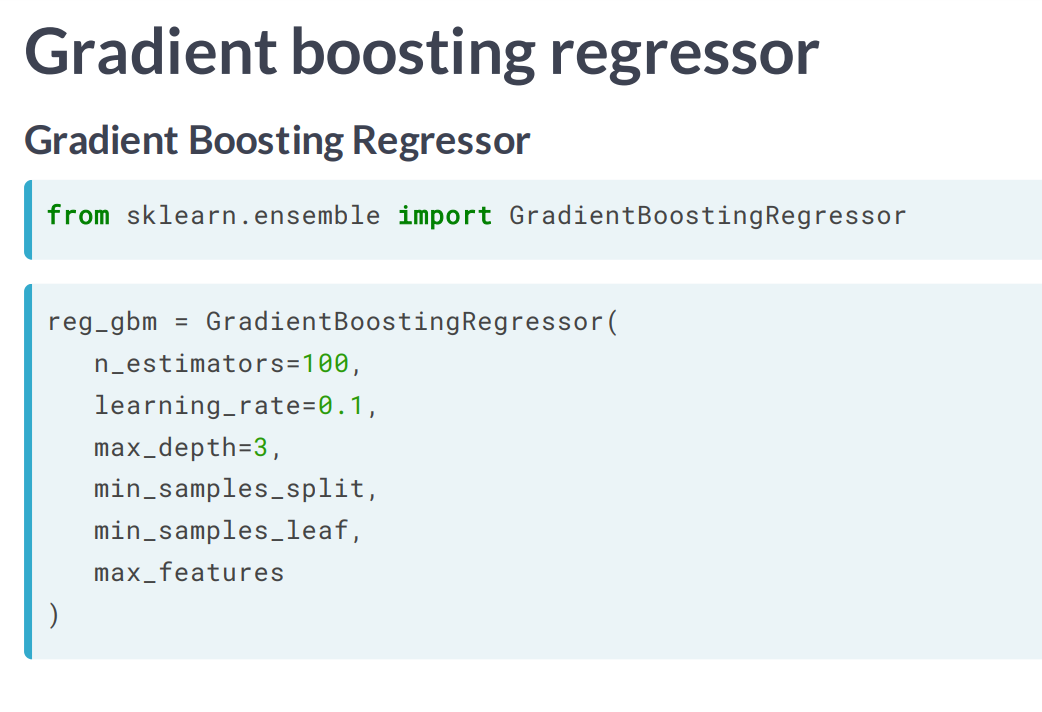
**Gradient boosting Classifier Sklearn:**

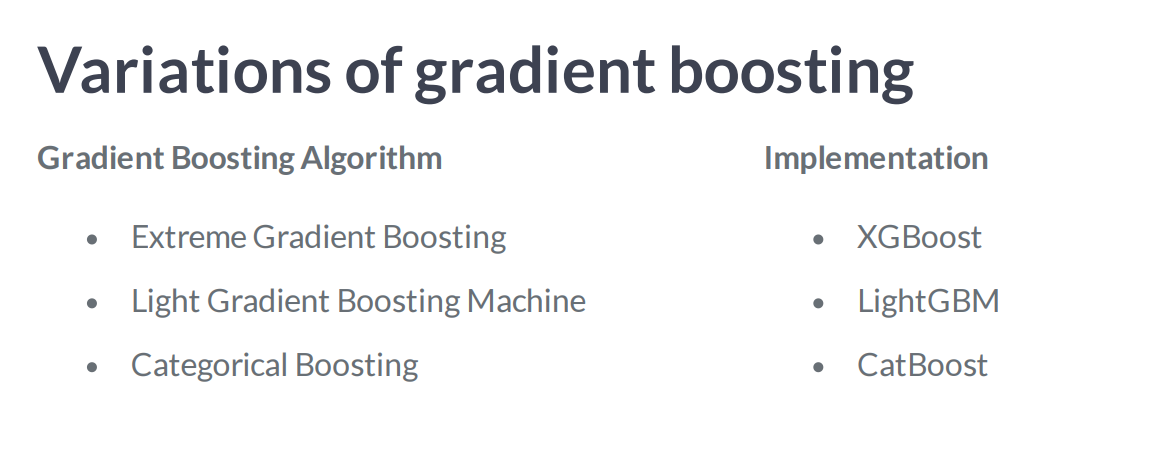
[**https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html**](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html)

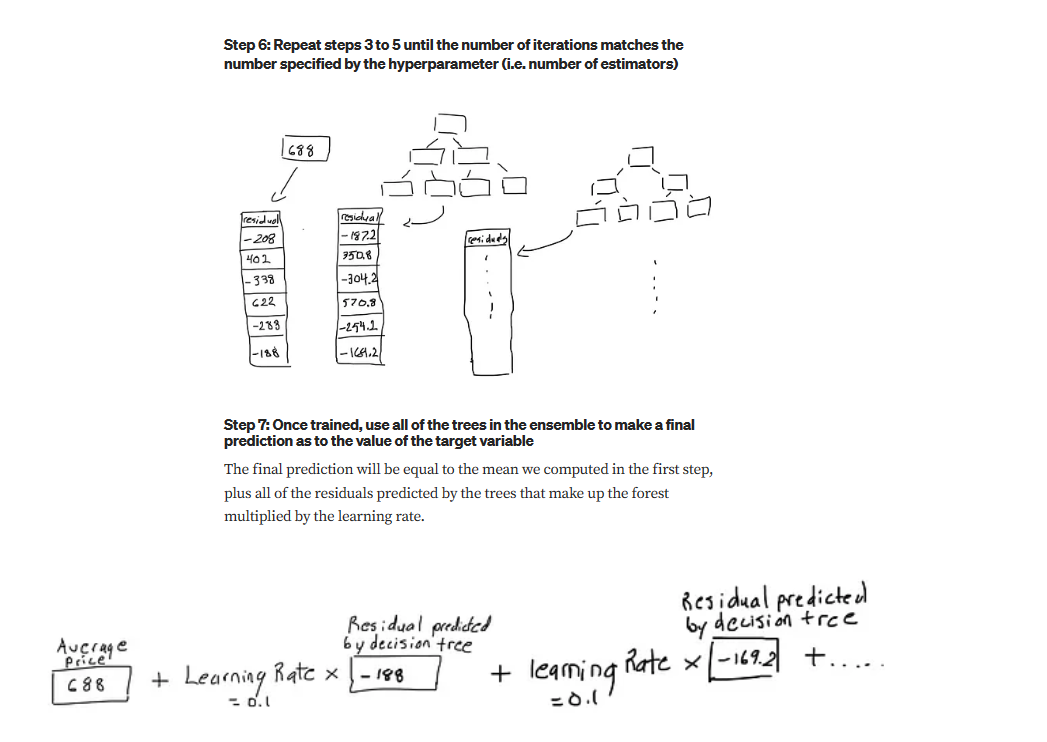


**Gradient boosting Regressor:**

[**https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html**](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html)





****