Gradient Boosting

According to wikipedia

Gradient boosting is a machine learning technique for regression and classification problems, which produce a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperform random forest. It builds the model in a stage wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable function.

One of the common uses of gradient boost is estimating continuous variables.

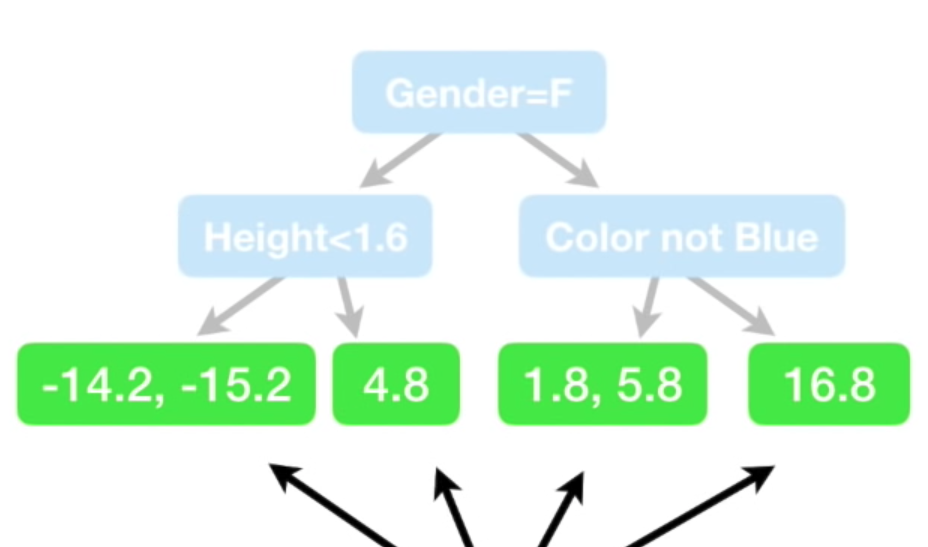
When gradient boost ise used for regression we start with the initial leaf that is the mean of the variable that we want to predict. Then we add trees based on residual values. Then we scale trees with learning rate to prevent overfitting. And we keep adding trees based on the error made by the previous tree.

How does it work?

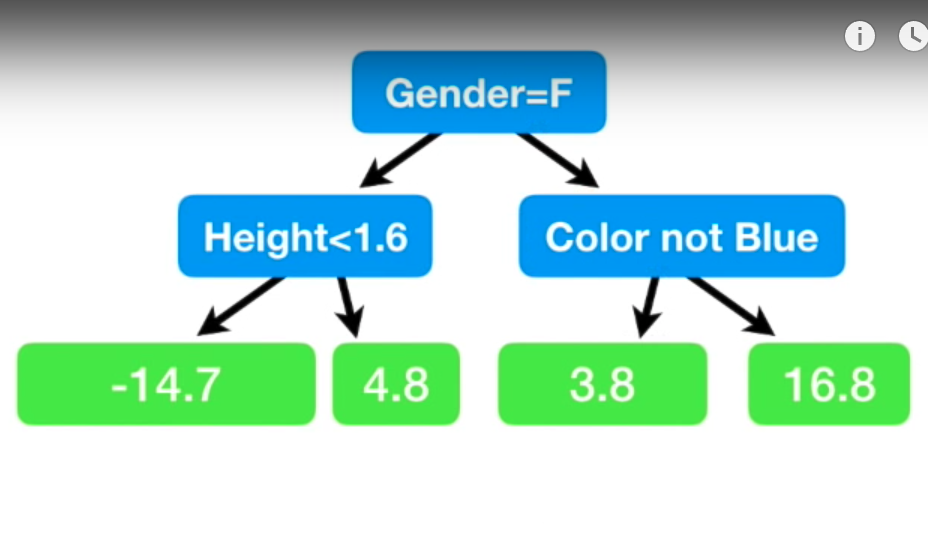


We are trying to estimate weight and the first initial estimate ist mean of the weight that is 71.2. Then we find the residual that is actual values - predicted.

Then we built a tree by using futures gender, height, color. In this example we are using upto 4 leaves but usually number of leaves are preferred between 8 to 32.



If there are leaves with more that one value like the one on the most left -14.2 and -15.2 we take the average -14.7 etc.



Then average weight + build tree

71.2 + 16.8 =88 which is a perfect prediction. However, we have low bias but high variance. Therefore we use the learning rate when we update trees.

We shove learning rate of 0.1

71.2+ (0.1\*16.8) = 72.9 which is better than initial guess

Then we repeat the same process with next tree

Advantages

GBDT are better learners than random forest

Prone to overfitting compare to random forest

Handle missing data

No data preprocessing required often works great with categorical and numerical values as is

Disadvantages

Tuning parameter is harder than random forest

Takes longer than random forest because it is built sequentially

Computationally expensive memory exhaustive

http://uc-r.github.io/gbm\_regression

Python Sklearn Implementation

Import libraries

Load data

Separate futures and target

(Split data into train and test for some cases to find accuracy, Also we may need to normalize input for some ML algorithm)

Define the model

Fit the Model

Predict