ML for Finance: Decision Trees, Random Forests, Boosted Trees

S. Yanki Kalfa

JHU-SAIS

June 17, 2022

Outline

1. Quick Recap of Penalized Regression
2. Decision Trees
3. Bagged Trees
4. Random Forests
5. Boosted Trees

Recap of Penalized Regression

- ARMAX models allow for exogenous regressors that can help us predict future outcomes
- However, they do not provide any selection
- We can use linear machine learning models
- Ridge shrinks all coefficients toward zero
- LASSO sets some variables to zero
- Elastic Net combines both LASSO and Ridge

Decision Trees

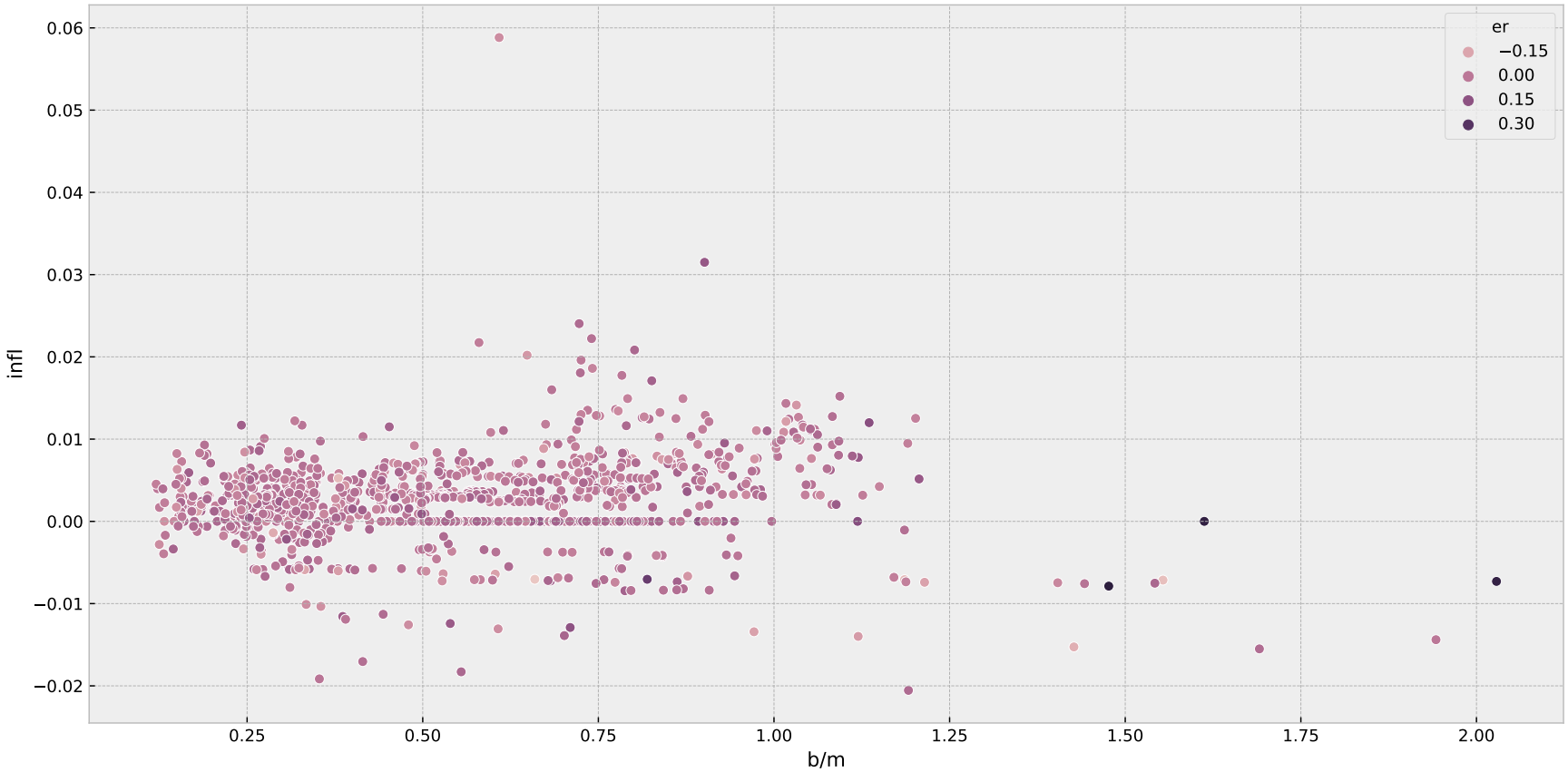
- Trees segement the feature space into rectangular regions
- The spliting rules can be summarised in trees, hence the name decision tree
- Pros:
  - easy to interpret
  - easy to compute
- Cons:
  - Not competitive with other methods
  - High variance (training data dependent)

Decision Trees, why do we care?

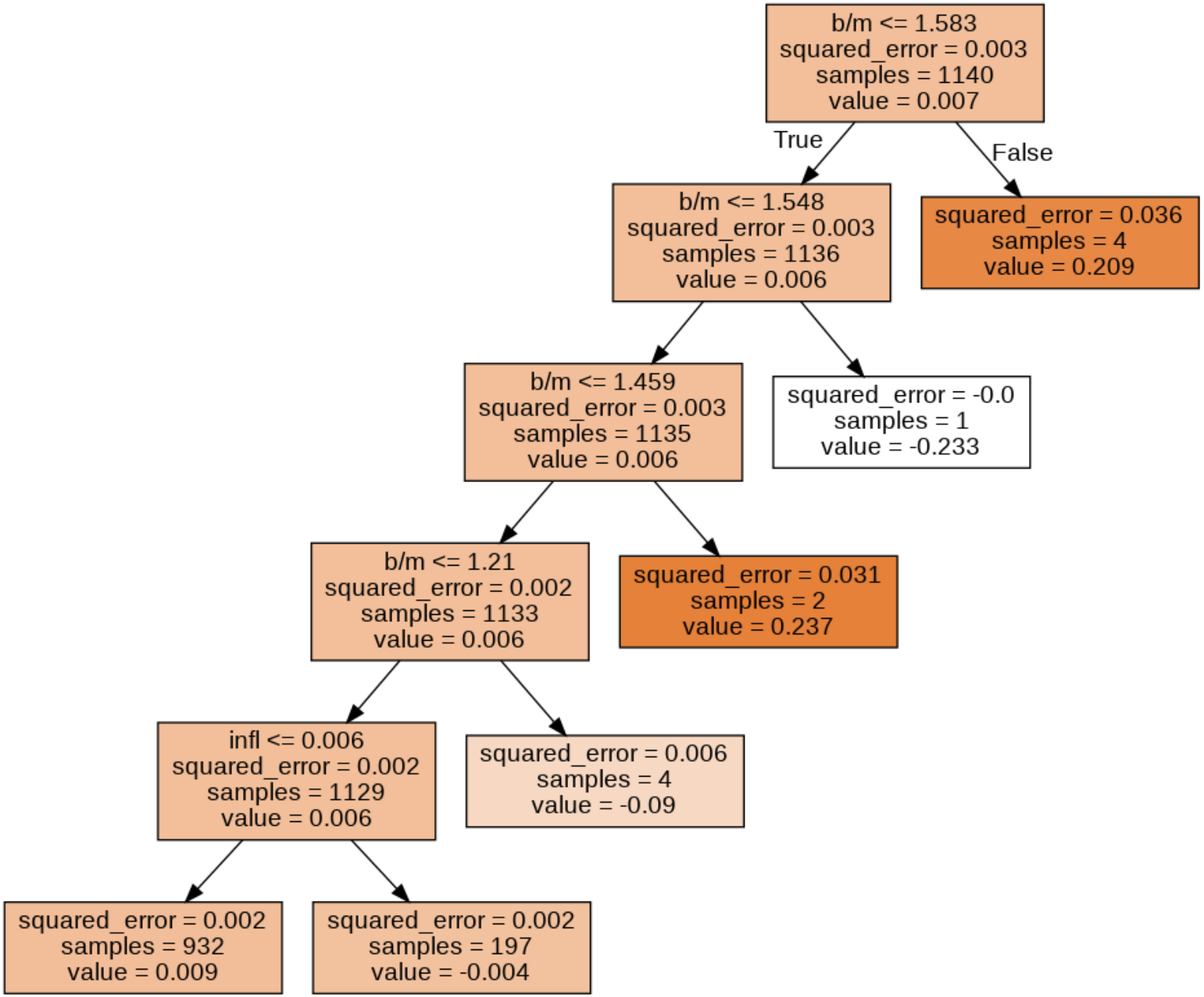
- Decision trees are at the core of:
  1. Bagged Trees
  2. Random Forests
  3. Boosted Trees

Decision Trees: Excess Returns

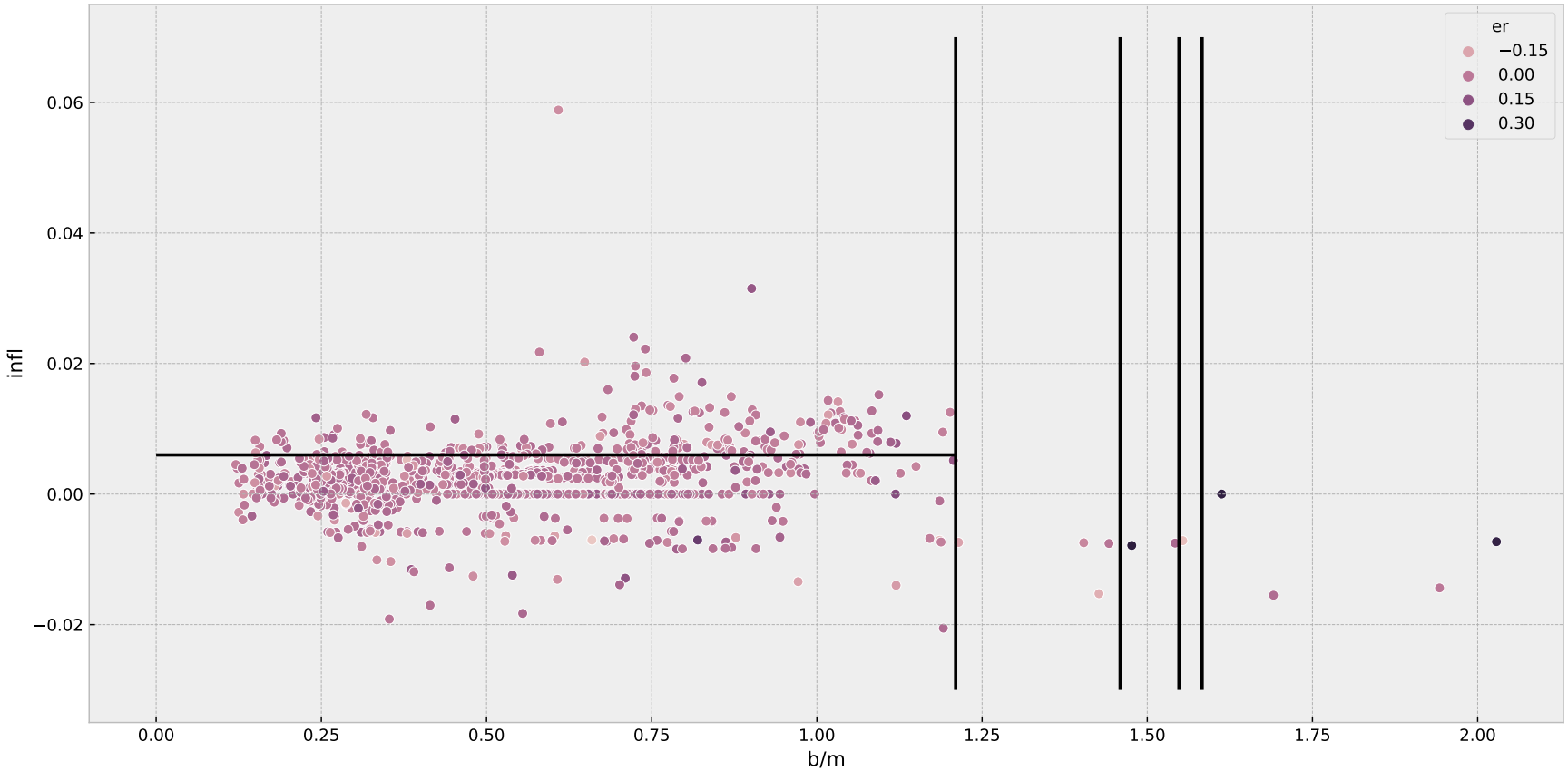
How would you split this data?



Decision Trees: Excess Returns



Decision Trees: Excess Returns



Do you agree with this split?

Decision Trees: Terminology

- We label the regions *terminal nodes*
- We call the nodes where we are splitting *internal nodes*

General Idea

1. Partition the feature space into J distinct and non-overlapping rectangles or high-dimensional rectangles.
2. Make the same prediction for each observation in the same leaf (terminal node).

How to Partition the Feature Space?

We want to minimize the RSS:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{1}$$

Consider for any splitting variable j and cut point s, we have:

$$R_1(j,s) = \{X | X_j \leq s\} \quad R_2(j,s) = \{X | X_j > s\} \tag{2}$$

Then we solve the following equation:

$$\min_{j,s} \left[\min_{\hat{y}_{R_1}} \sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \min_{\hat{y}_{R_2}} \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \right] \tag{3}$$

Stop when a criterion is reached (i.e. leaves are pure, each region has 5 observations)

Explanation

- Computationally infeasible to consider all possible combinations of splits.
- Hence we use *recurisve binary splits*. This is a **top-down**, **greedy** approach.
- It is **top-down** because we start at the top of the tree, then split the feature space.
- It is **greedy** because the splits are made at the *best* point, meaning that it is not made with some look-ahead.

Explanation

- Select predictor X_j and splitpoint s,such that when we split the feature space $\{X|X_j < s\}$ and $\{X|X_j \geq s$ leads to the highest possible reduction SSE.
- Repeat the above process for the resulting terminal nodes
- This process continues until we reach some threshold

Pruning the Tree

- Minimizing the in-sample SSE does not result in good forecasts, it leads to overfitting.
- Maybe we want a smaller tree would lead to smaller variance at the cost of some bias
- One method is to put a hard threshold on SSE
- This is shortsighted, a worthless split in the beginning can lead to a large decrease in later splits

Cost Pruning

- Grow a deep tree (T_0), then cut it back
- Consider a sequence of trees penalized by hyperparameter $\alpha \geq 0$.
- For each α $\exists T \subseteq T_0$. Such that:

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

is as small as possible. $|T|$ is total number of terminal nodes, R_m is the region corresponding to the m^{th} terminal node.

Bagging and Random Forests

Decision Trees have high variance. So, how to deal with that?

- Bootstrap Aggregation
- Random Forests

Bagging: Introduction

- Bootsrap aggregation (bagging) is used to reduce the variance of a forecasting method.
- Suppose we have a set n independent samples from the popuilation, because they come from the same random variable, the variance of each sample is σ^2
- The variance of the mean becomes: σ^2/n
- Hence we reduce the variance
- But we only have one training data
- Hence we take repeated samples from the same training data, called bootstraping

Bagging

1. Construct B bootstrapped decision trees
2. Grow the trees deep
3. Get $\hat{f}^{\star,b}(x)$ prediction at x.
4. Average across the trees

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1} \hat{f}^{\star,b}(x)$$

Bagging will result in low bias and low variance prediction.
Can we improve over bagged trees?

Random Forests - Why do we care?

Bagging seems to solve the variance problem. Why complicate our lives? Consider the following:

- 50 predictors, $p = 50$
- 1 very strong predictor
- 9 moderately strong predictors

Bootstrapped trees will:

- Choose the same predictor at the top split
- Look similar
- Have highly correlated predictions

$$\bar{\sigma}^2 = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{4}$$

Averaging highly correlated predictions does not decrease the variance as much.

Random Forests - OK, I care now

How do we *de-correlate* the trees?

1. Grow B bootstrapped trees
2. At each splitting node consider only a random subset of predictors $m << p$
3. The split only use X_i for $i \in m$ for the split
4. Take a new random sample $m << p$ at each split

A good starting point is $m = \sqrt{p}$.
$(p - m)/p$ of the splits will not consider the strong predictor.
This procedure will result in less variance in the average of the trees.

Boosting

- Boosting can be applied to other forecasting methods
- Boosting is not a parallel method like bagging or random forests
- Boosting is a sequential method
- Each tree is grown based on the previous tree

Boosting: Procedure

- Instead of fitting a deep tree, boosting *learns* slowly
- Given current tree, fit another tree on the residuals
- Add the tree into the fitted function and update residuals
- We want the trees to be shallow (small) with few terminal nodes
- Fitting trees to the residual improves the function in areas where the fit is poor
- Shrinkage parameter λ (learning rate) slows the process for different shaped trees to be fitted

Boosting: Algorithm

1. Set $\hat{f}(x) = 0$ and $r_t = y_t$ for all t
2. For $b = 1, 2, \ldots, B$:
   1. Fit $\hat{f}^b$ with d splits and $(d+1)$ terminal nodes to (X, r)
   2. Update $\hat{f}$ by adding penalized $\hat{f}^b(x)$.

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

   3. Update residuals,

$$r_t \leftarrow r_t - \lambda \hat{f}^b(x_t)$$

3. Final model:

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting: Hyperparameters

- Number of trees: B
- Learning Rate: λ
- Number of splits: d

Source

Source: Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. "An introduction to statistical learning." (2009).