# Tree-Based Methods

Hui Chen

MIT Sloan

15.450, Spring 2023

# Outline

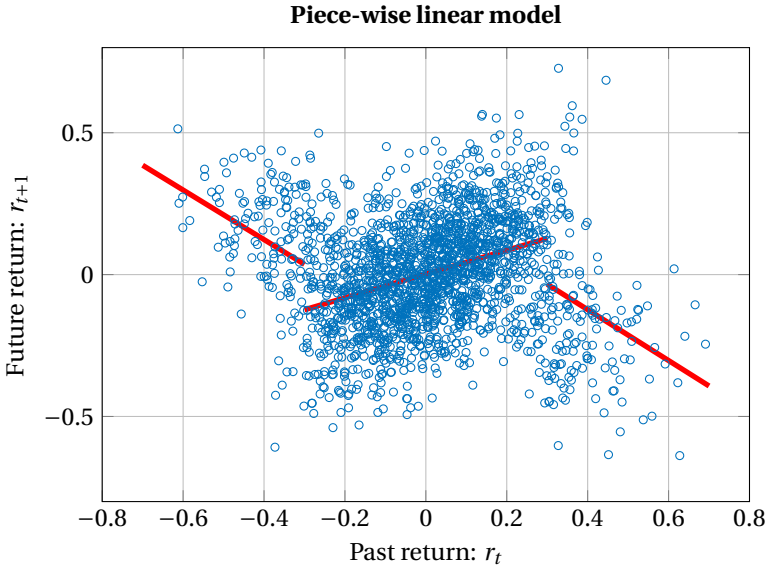
# Do you see any patterns?

# Fitted to a linear model

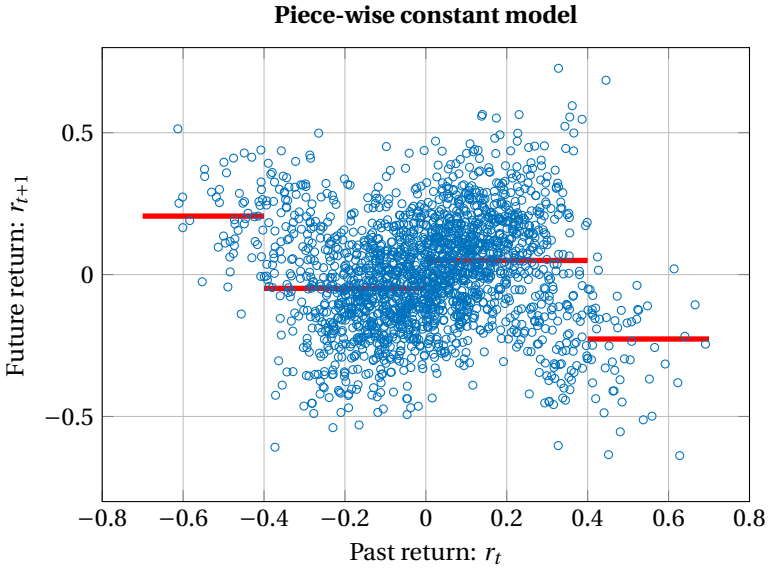
# Do you see any patterns?

# Fitted to a linear model

# Fitted to a piece-wise linear model



# Fitted to a piece-wise constant model



# Decision Trees

- The goal: Predict $y$ using a $p \times 1$ vector of features $\mathbf{x}$.
- $\mathbf{x}$ is in a $p$-dimensional feature space.
- The idea of tree-based methods is to divide up the feature space into a set of rectangles,

$$\{R_1, \cdots, R_m\}$$

- If $\mathbf{x}$ falls into a rectangle $R_j$, the prediction $f(\mathbf{x})$ will be based on the "consensus" of the observations from the training sample that fall into $R_j$.
  - ↪ Regression tree (mean) and classification tree (mode).
  - ↪ Q: Recall KNN – what's the difference?

# Decision Trees

- Regression tree: The prediction will be the mean of the response values for the training data that are in $R_j$.

$$c_j = \frac{1}{n_j} \sum_{i \in R_j} y_i$$

  ↪ $n_j$: the number of observations from the training sample that are in $R_j$.

- Classification tree: The prediction will be the most common response among the training observations in $R_j$.

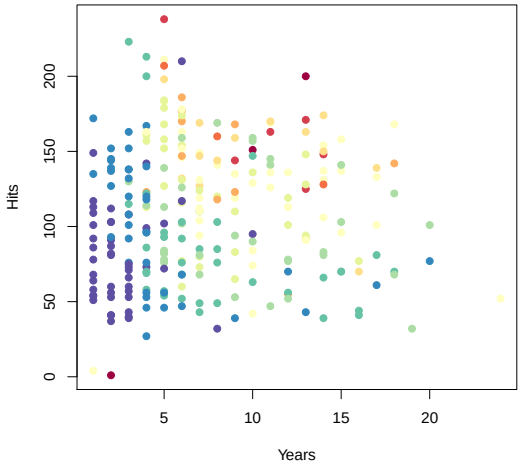
$$c_j = \underset{k=1,\cdots,K}{\arg\max}\, \widehat{p}_k(R_j)$$

  where

$$\widehat{p}_k(R_j) = \frac{1}{n_j} \sum_{i \in R_j} 1\{y_i = k\}$$

  is the fraction of training observations in $R_j$ that are in class $k$.

# Example: Baseball player salary




Salary is color-coded from low (blue, green) to high (yellow, red)

# Example: Baseball player salary


One possible way to divide up the feature space

# How to Divide Up the Feature Space?

- We would like a set of rectangles that give us the best fit in the training data.
- In a regression tree, this can be stated as minimizing RSS (residual sum of squares):

$$\min_{R_1,\cdots,R_m} \sum_{j=1}^{m} \sum_{i \in R_j} (y_i - c_j)^2$$

- Need to consider all possible ways to divide up the feature space using rectangles.
  - ↪ It turns out to be computationally a very difficult problem.

# How to Divide Up the Feature Space?

## A top-down tree-building process

1. Starting at the top of the tree (root), pick one predictor $x_\ell$ out of the $p$ features, and split the feature space into two regions based on some threshold $s$:

$$\{\mathbf{x} : x_\ell \leq s\}, \quad \{\mathbf{x} : x_\ell > s\}$$

2. Keep on splitting in each new region until a certain stopping criterion is met.
   - ↪ For example, when the number of observations remaining in a region is below some threshold.
3. The terminal nodes are called leaves, which correspond to the final set of regions

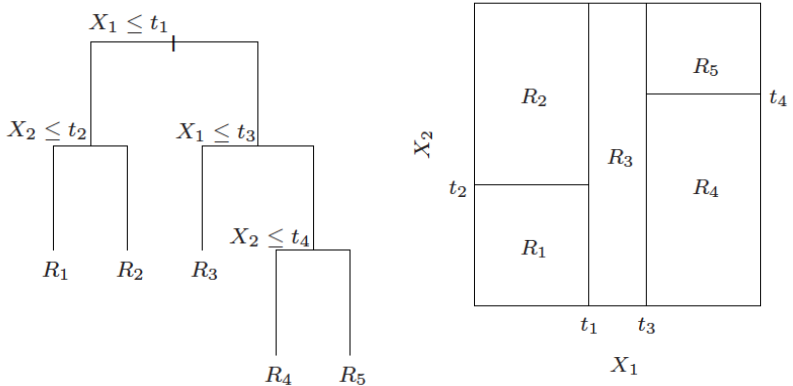
$$R_1, R_2, \cdots, R_m$$

4. The points along the tree where the split occurs are called internal nodes.

- At each internal node, how to choose the splitting variable $x_i$ and threshold $s$? Use a "greedy" strategy. See CART (classification and regression tree).

# Illustration: How to Build a Tree

Source: ISL



# The CART Algorithm: Regression

- A "greedy" strategy: Look for the split that gives the biggest improvement on fitting error.
- For any pair $x_\ell, s$, an old region is split into two new regions:

$$R_1 = \{\mathbf{x} : x_\ell \leq s\}, \quad R_2 = \{\mathbf{x} : x_\ell > s\}$$

- For a regression tree, we choose $\ell, s$ by minimizing the RSS:

$$\arg\min_{\ell, s} \sum_{i \in R_1} (y_i - c_1)^2 + \sum_{i \in R_2} (y_i - c_2)^2$$

  ↪ $c_j$ is the average response in region $R_j$.

- Thus, the greedy strategy favors those splits that give more "pure" regions in the immediate step.

# The CART Algorithm: Classification

- For a classification tree, we can choose $\ell, s$ by minimizing the classification error:

$$\arg\min_{\ell,s} \left( n_1 (1 - \widehat{p}_{c_1}(R_1)) + n_2 (1 - \widehat{p}_{c_2}(R_2)) \right)$$

  - $\widehat{p}_k(R_j)$ is the fraction of training observations in $R_j$ that belong to class $k$;
  - $c_j$ is the most common class in region $R_j$.

- Again, the greedy strategy favors those splits that give more "pure" regions.

- In practice, classification error is often not sensitive enough for tree-growing. Two alternative criteria: Gini index and cross-entropy.

# Pruning a Tree

- What happens if we keep splitting each branch? When should we stop splitting a branch further?
  - ↪ When the decrease in classification error (or RSS) is small enough?
- The standard practice:
  1. Build a large tree $T_0$ through recursive binary splitting, stopping when each terminal node has fewer than certain minimum number of observations.
  2. Prune the tree back.

# Pruning a Tree

- For any tree $T$, denote the number of leaves (terminal nodes) by $|T|$.
- Among all the trees that are a subset of the large tree $T_0$, look for the one that minimizes the classification error but with a penalty on the size of the tree (number of leaves):

$$\min_{T \subseteq T_0} \underbrace{\sum_{j=1}^{|T|} (1 - \widehat{p}_{c_j}(R_j))}_{\text{classification error}} + \alpha |T|$$

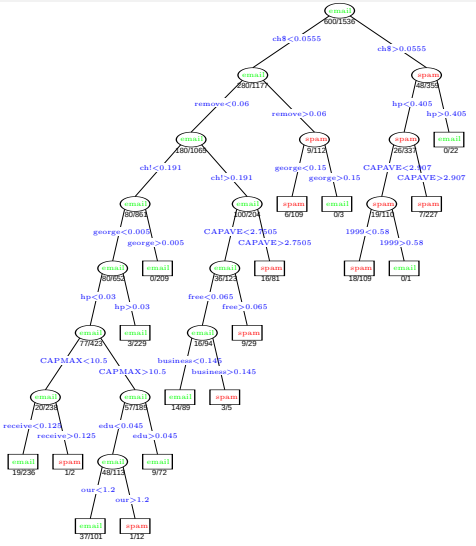
  Q: What does this remind you of?

- Replace classification error with RSS for regression tree.
- $\alpha$: Tuning parameter. We can pick $\alpha$ through cross-validation.
- In Python, we can use the *tree* module from *sklearn* library to implement CART (use either the *DecisionTreeRegressor* or *DecisionTreeClassifier* class).

# Example: SPAM Filter

- A famous example of decision tree learning is to develop an email SPAM filter.
- A public dataset of 4601 emails (ftp.ics.uci.edu), 1813 of which are labelled as spam. For each email there are $p = 57$ predictors:
  - ↪ 54 quantitative predictors – the percentage of words/characters in the email that match a given word/character.
  - ↪ CAPAVE: The average length of uninterrupted sequences of capital letters.
  - ↪ CAPMAX: The length of the longest uninterrupted sequence of capital letters.
  - ↪ CAPTOT: The sum of the length of uninterrupted sequences of capital letters.

# Example: SPAM Filter Tree




Hastie, Tibshirani, Friedman (2013), p315

# Outline

# Bagging

- Trees are very intuitive, easy to display and easy to explain. However, trees tend not to be very accurate in prediction.
- Bagging: Averaging the trees over a collection of bootstrap samples to reduce the variance of prediction. Q: Why would this help?

# Bootstrap: General Principle

- Bootstrap is effectively a Monte Carlo study which uses the empirical distribution as if it were the true distribution.
- Given the training data $(\mathbf{x}_i, y_i)$ with $i = 1, \cdots, n$, under the IID assumption, the empirical distribution function of $(\mathbf{x}, y)$ is
$$\Pr((\mathbf{x}, y) = (\mathbf{x}_i, y_i)) = \frac{1}{n}$$
- Key applications of bootstrap methodology:
  - ↪ Improve the stability of fitted function.
  - ↪ Evaluate distributional properties of complicated estimators and perform bias adjustment.
  - ↪ Improve the precision of asymptotic approximations in small samples (confidence intervals, test rejection regions, etc.)

# Bagging Algorithm

## Bootstrap aggregation

1. Draw $B$ bootstrap samples, each with $n$ observations.
2. Fit a tree $\widehat{f}^{tree,b}$ to bootstrap sample $b$, $b = 1, \cdots, B$ (with or without pruning).

3a. Classification tree: To classify input $\mathbf{x}$, take the majority vote among the $B$ trees:

$$\widehat{f}^{bag}(\mathbf{x}) = \underset{k=1,\cdots,K}{\arg\max} \sum_{b=1}^{B} 1\{\widehat{f}^{tree,b}(\mathbf{x}) = k\}$$

3b. Regression tree: The prediction is the average of the predictions from the $B$ trees:

$$\widehat{f}^{bag}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}^{tree,b}(\mathbf{x})$$

# Random Forests

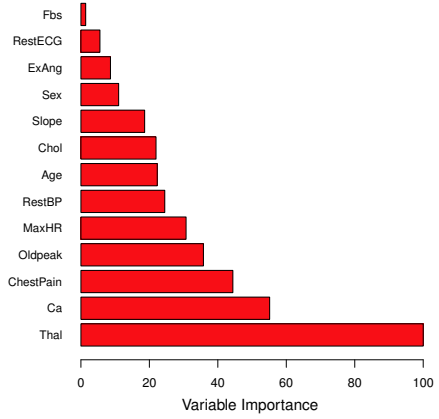
- A further improvement over bagging.
- With bagging, the trees $\widehat{f}^{tree,b}$ are not independent, because they are fit on similar data (bootstrapped from the same training data).
- How to make the trees on different bootstrap samples less correlated? Each time a split in a tree is considered, only consider a random selection of $h$ predictors from the full set of $p$ predictors.
- Typical choice: $h \approx \sqrt{p}$
- How could it be a good idea to throw away data? The benefit of less correlated trees outweighs the costs.
- While RF typically improves the prediction accuracy, it loses much of the appeal of decision tree in interpretability.

# Measuring Predictor Importance

- Ensemble methods are good at reducing variance, but at the expense of interpretability.
- For bagging, RF, and boosting, we can still measure how important each variable is for the overall model's performance.
- For each tee, we can add up the total amount that the RSS (or prediction error) is reduced due to the splits over a given predictor.
- Averaging over the $B$ trees gives us the variable importance measure.

# Measuring Predictor Importance

Source: ISL Figure 8.9

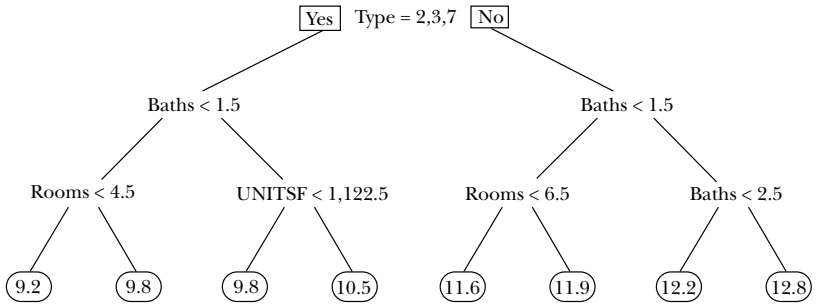



A variable importance plot for the Heart data. Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

# Predicting House Prices

Mullainathan and Spiess (2017)

**A Shallow Regression Tree Predicting House Values**




*Note:* Based on a sample from the 2011 American Housing Survey metropolitan survey. House-value predictions are in log dollars.

# Predicting House Prices

## Mullainathan and Spiess (2017)

**Performance of Different Algorithms in Predicting House Values**

| | *Prediction performance ($R^2$)* | | *Relative improvement over ordinary least squares by quintile of house value* | | | | |
|---|---|---|---|---|---|---|---|
| *Method* | *Training sample* | *Hold-out sample* | 1st | 2nd | 3rd | 4th | 5th |
| Ordinary least squares | 47.3% | 41.7% [39.7%, 43.7%] | – | – | – | – | – |
| Regression tree tuned by depth | 39.6% | 34.5% [32.6%, 36.5%] | −11.5% | 10.8% | 6.4% | −14.6% | −31.8% |
| LASSO | 46.0% | 43.3% [41.5%, 45.2%] | 1.3% | 11.9% | 13.1% | 10.1% | −1.9% |
| Random forest | 85.1% | 45.5% [43.6%, 47.5%] | 3.5% | 23.6% | 27.0% | 17.8% | −0.5% |
| Ensemble | 80.4% | 45.9% [44.0%, 47.9%] | 4.5% | 16.0% | 17.9% | 14.2% | 7.6% |

*Note:* The dependent variable is the log-dollar house value of owner-occupied units in the 2011 American Housing Survey from 150 covariates including unit characteristics and quality measures. All algorithms are fitted on the same, randomly drawn training sample of 10,000 units and evaluated on the 41,808 remaining held-out units. The numbers in brackets in the hold-out sample column are 95 percent bootstrap confidence intervals for hold-out prediction performance, and represent measurement variation for a fixed prediction function. For this illustration, we do not use sampling weights. Details are provided in the online Appendix at http://e-jep.org.

# Are ML Models Better?

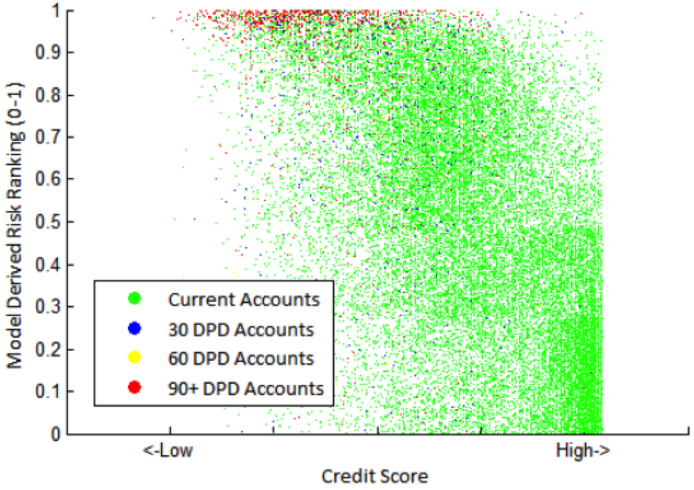



Fig. from Butaru et al. (2016)

# Summary

- Decision-tree based methods are just one class of nonlinear prediction methods.
- Nonlinear methods can lead to significant improvements in prediction accuracy sometimes, but not always.
- What's next? The list keeps growing ...
- Butaru et al. (2016). Risk and risk management in the credit card industry. Journal of Banking and Finance.