Classification and Regression Trees

PSC 8185: Machine Learning for Social Science

Iris Malone

February 28, 2022

Materials adapted from Sergio Ballacado

Announcements

- Problem Set 4 Released: Due Spring Break-ish

Recap

Where We've Been:

- Cross-validation helps identify tuning parameters
- Bootstrap tells us confidence (SE) around estimate
- Can perform variable selection for regression models using subset selection, shrinkage, or dimensionality reduction techniques

Recap

Where We've Been:

- Cross-validation helps identify tuning parameters
- Bootstrap tells us confidence (SE) around estimate
- Can perform variable selection for regression models using subset selection, shrinkage, or dimensionality reduction techniques

New Terminology:

- Curse of Dimensionality
- Lasso regression
- Ridge regression
- Shrinkage Penalty
- Principal Components

Agenda

1. Why We Need Non-Parametric Models
2. Regression Trees
3. Classification Trees
4. Bagging

Why We Need Non-Parametric Models

Motivation: Bias-Variance Trade-Off

- A central ML challenge is finding a method that minimizes *both* variance and bias.
- Bias-Variance Trade-Off: Models tend to result in either (1) low variance and high bias (under-fitting) or (2) high variance and low bias (over-fitting).
- Parametric and non-parametric methods take different approaches to optimize bias-variance trade-off

Recall: Parametric Models

- More rigid → **low variance**

Recall: Parametric Models

- More rigid $\rightarrow$ **low variance**
- Assumptions:
  - Assumes f has fixed function form (e.g. linear)
  - Assumes fixed number of parameters ($\beta_1, \ldots \beta_p$)

Recall: Parametric Models

- More rigid $\rightarrow$ **low variance**
- Assumptions:
  - Assumes f has fixed function form (e.g. linear)
  - Assumes fixed number of parameters ($\beta_1, \ldots \beta_p$)
- Estimation Goals: f $\rightarrow$ estimating parameters

Recall: Parametric Models

- More rigid $\rightarrow$ **low variance**
- Assumptions:
  - Assumes f has fixed function form (e.g. linear)
  - Assumes fixed number of parameters ($\beta_1, \ldots \beta_p$)
- Estimation Goals: f $\rightarrow$ estimating parameters
- Model Assessment and Selection:
  - Fixed in quality of fit
  - Apply shrinkage/regularization to further reduce variance

Recall: Parametric Models

- More rigid $\rightarrow$ **low variance**
- Assumptions:
  - Assumes f has fixed function form (e.g. linear)
  - Assumes fixed number of parameters ($\beta_1, \ldots \beta_p$)
- Estimation Goals: f $\rightarrow$ estimating parameters
- Model Assessment and Selection:
  - Fixed in quality of fit
  - Apply shrinkage/regularization to further reduce variance
- Common Methods
  - OLS
  - GLMs
  - GAMs
  - Logit Regression/LPM
  - Lasso/Ridge Regression
  - PCR

Overview of Non-Parametric Models

- More flexible → **low bias**
- Requires very few assumptions

Overview of Non-Parametric Models

- More flexible $\rightarrow$ **low bias**
- Requires very few assumptions
- Estimation Goals:
  - No fixed f to describe data
  - $\hat{f}$ is “black box”

Overview of Non-Parametric Models

- More flexible $\rightarrow$ **low bias**
- Requires very few assumptions
- Estimation Goals:
  - No fixed f to describe data
  - $\hat{f}$ is "black box"
- Model Assessment and Selection:
  - Changing in quality of fit.
  - More data $\rightarrow$ better model!

Common Non-Parametric Methods

- KNN (Jan 24)
- Today:
  - Classification and Regression Trees (CART)
  - Bagging
- Next Week:
  - Random Forests
  - Boosting
  - BART
- SVM (March 21)

Why Use Non-Parametric Models?

1. Resolve Common Regression Problems
2. Explore New/Original Data
3. Increased Transparency

Recall: Common Regression Problems

- Common Regression Problems
  - Interaction Effects
  - Non-Normal Residuals
  - Non-Linear Relationships

Recall: Common Regression Problems

- Common Regression Problems
  - Interaction Effects
  - Non-Normal Residuals
  - Non-Linear Relationships
- Potential Solutions:
  - Model interactions
  - Use GLMs
  - Use GAMs

Recall: Common Regression Problems

- Common Regression Problems
  - Interaction Effects
  - Non-Normal Residuals
  - Non-Linear Relationships
- Potential Solutions:
  - Model interactions
  - Use GLMs
  - Use GAMs
  - **Use non-parametric methods**

New/Original Data Collection

- Original Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?

New/Original Data Collection

- Original Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything

New/Original Data Collection

- Original Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting

New/Original Data Collection

- Original Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions

New/Original Data Collection

- Original Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions but could make wrong assumptions

New/Original Data Collection

- Original Data Problems:
  - What's the underlying DGP?
  - What are the most relevant predictors?
  - What's the true f?
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions but could make wrong assumptions
  - **Use non-parametric methods**

Increased Transparency

- Parametric Model Ethics
  - Data Dredging
  - Model P-Hacking
  - Stargazing

Increased Transparency

- Parametric Model Ethics
  - Data Dredging
  - Model P-Hacking
  - Stargazing
- Potential Solutions:
  - Model everything but high risk of overfitting
  - Use existing lit to make predictions but could make wrong assumptions
  - **Use non-parametric methods**

Regression Trees

Overview of Decision Trees

Tree-based methods focus on the use of **decision trees**.

Main Idea:

- Stratify the data into different regions (R_k) using a series of **decision rules** (splitting rules) which maximize differences between classes.
- Summarize the rules segmenting the predictor space as a decision tree.

Overview of Decision Trees

Tree-based methods focus on the use of **decision trees**.

Main Idea:

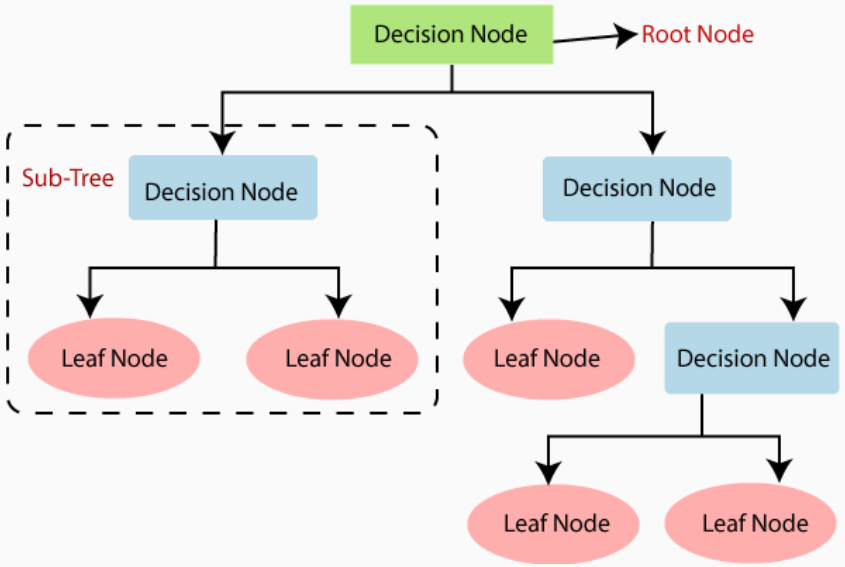
- Stratify the data into different regions (R_k) using a series of **decision rules** (splitting rules) which maximize differences between classes.
- Summarize the rules segmenting the predictor space as a decision tree.

Types of Decision Trees:

- Regression Trees (Numeric Outcome)
- Classification Trees (Categorical Outcome)

Decision Tree Vocabulary

- **Branches:** Each decision (splitting) rule
- **Root:** Topmost Node (Starting Set of Observation)
- **Decision Node:** Predictor variable X_j which is split
- **Subtree:** The tree that is a child of a decision node
- **Leaves:** Terminal Nodes (Final Range of Observations R_k)



Regression Trees

Estimation Goal: Partition set of predictors into different subregions to best explain variation across a continuous outcome.

Regression Trees

Estimation Goal: Partition set of predictors into different subregions to best explain variation across a continuous outcome.

- Select a region R_k (set of observations), a predictor X_j, and a splitting point s
- Loss function minimizes residual sum of squares (RSS)

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2$$

Regression Tree Procedure

Procedure:

- Split R_k using decision rule $X_j < s$ in order to produce the largest decrease in RSS
- Stop partitioning after pre-determined **stopping point**, or number of observations in each leaf

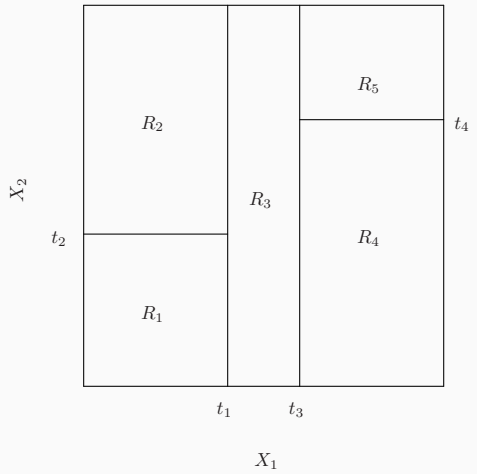


Figure 1: Division of Regional Space

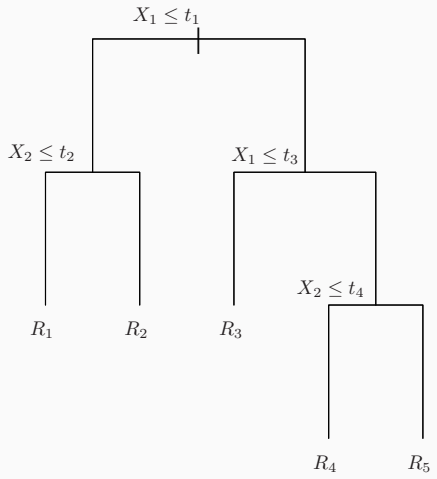


Figure 2: Decision Tree

Recursive Binary Splitting

Partitioning starts at the top and works its way down the tree using procedure known as **recursive binary splitting**

- Find predictor X_1 which explains *most* variation across Y and set as root
- Find predictor X_2 which explains *second most* variation across Y and set as next leaf
- Continue until stopping point is reached.

Example Regression Tree

Motivation: Predict Baseball player salaries based on experience (years) and performance (hits)

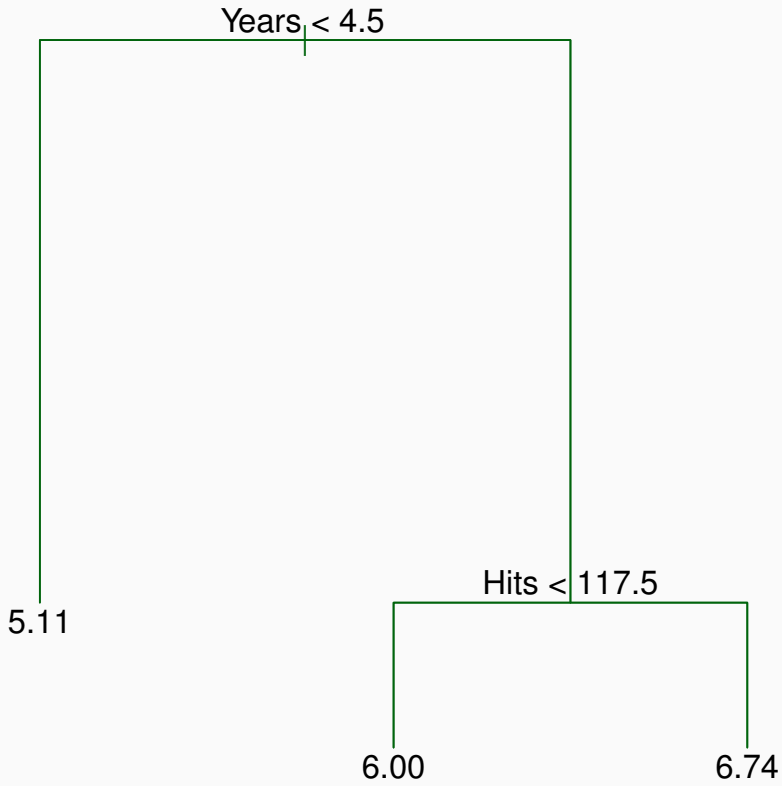


Figure 3: Number in each leaf is mean response for observations

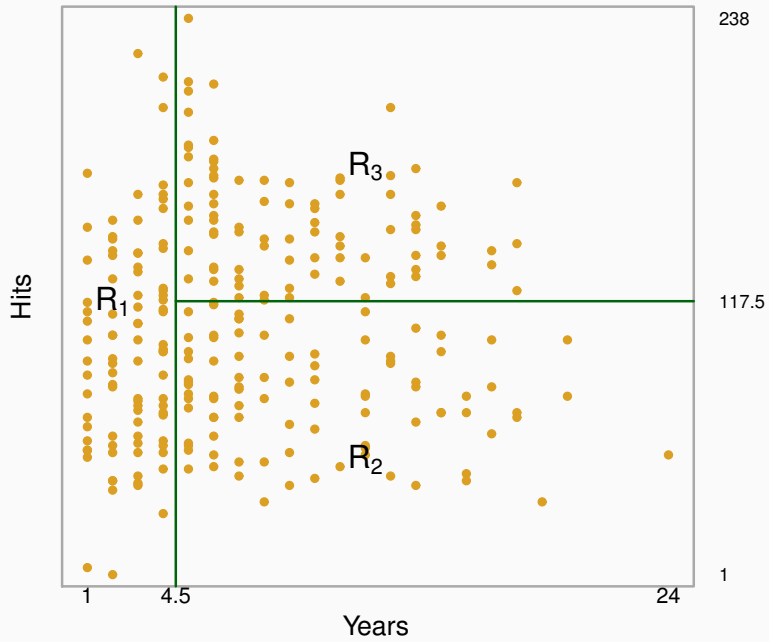


Figure 4: Three region partition R_1, R_2, R_3

Example Regression Tree

Interpretation: The prediction for a point which falls under R_i in test data is the average of the training points in R_i

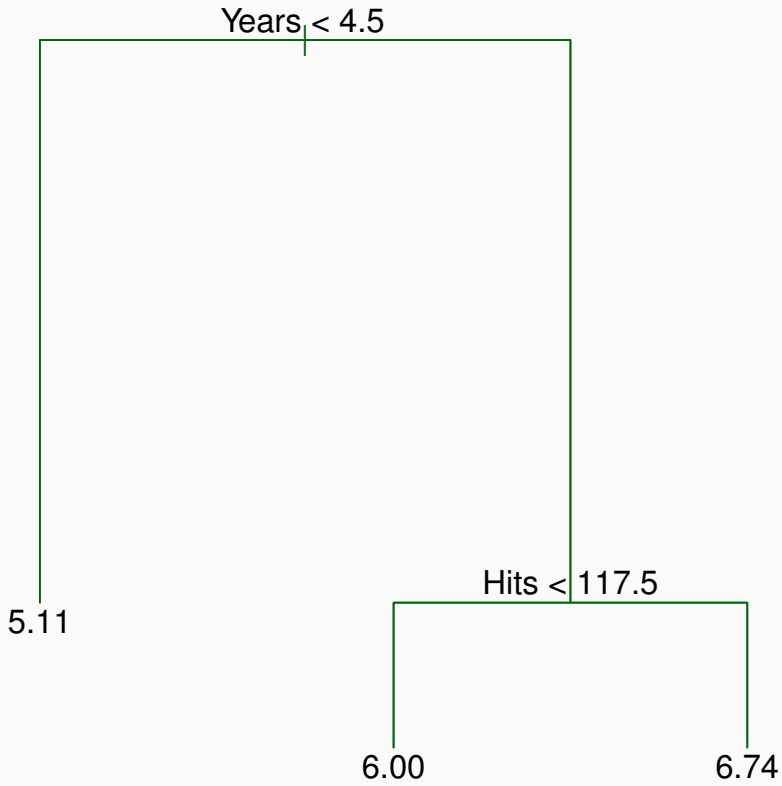


Figure 5: Salary for less than 4.5 years of experience is $e^{5.11} = \$165,670$

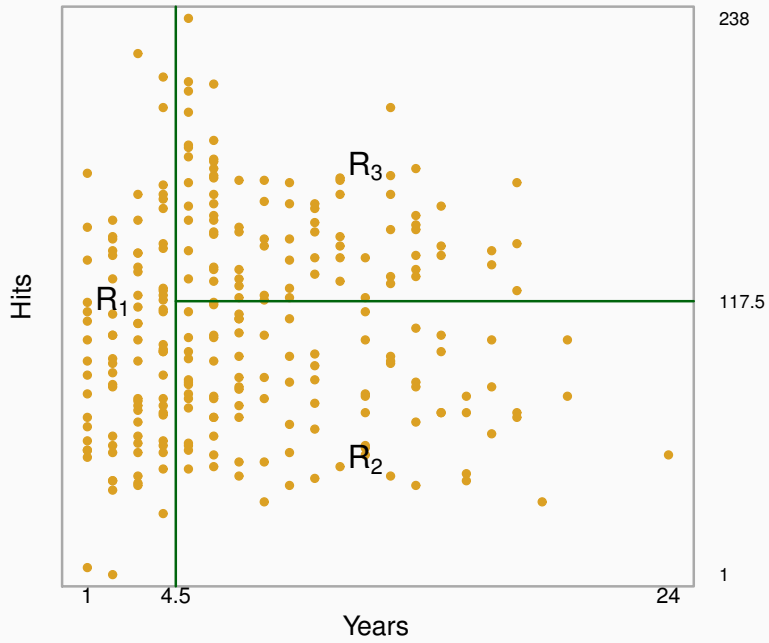


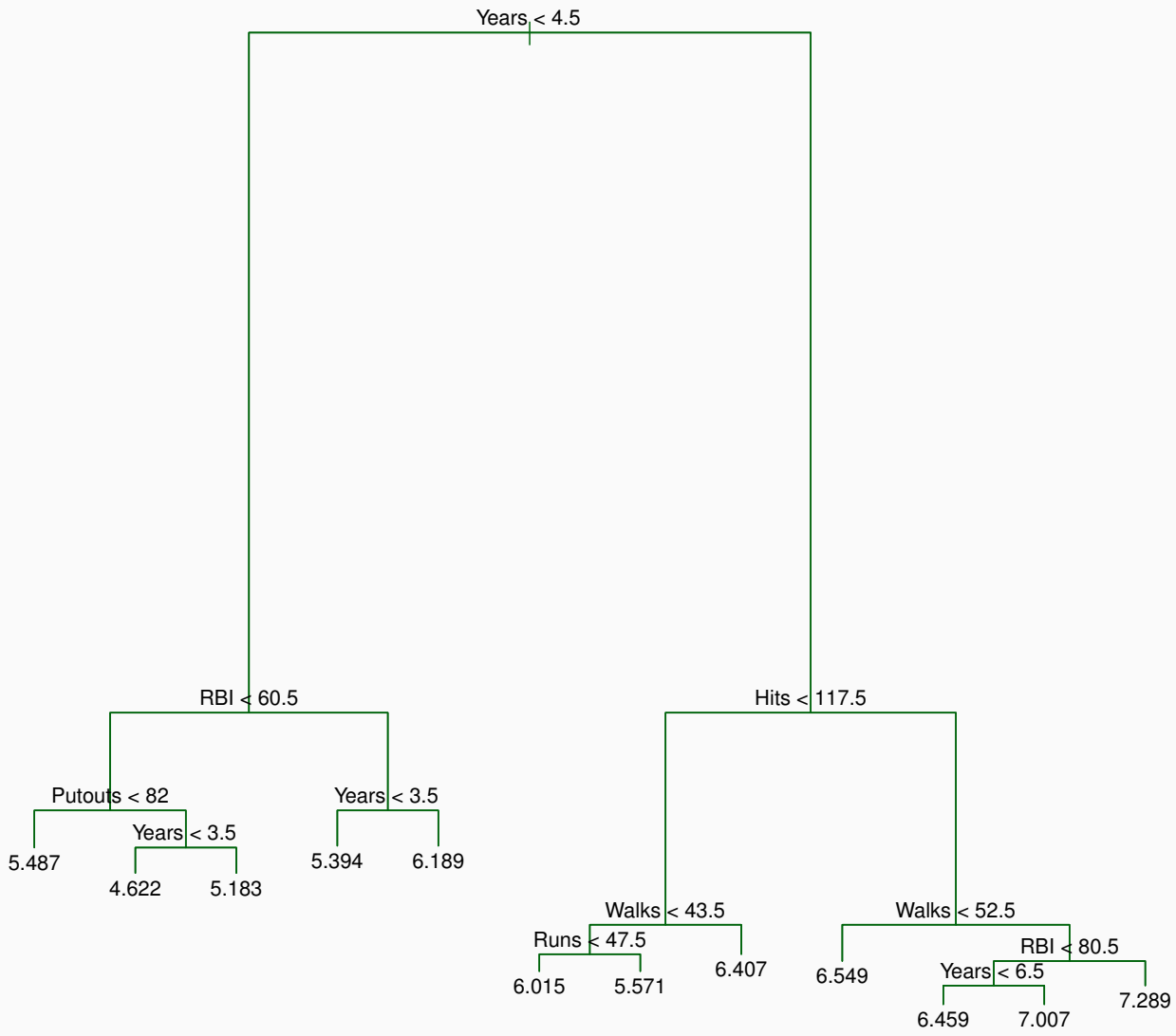
Figure 6: Number of baseball players under each terminal node

Limits to Recursive Binary Splitting

- Produces **top-down greedy** approach - makes best split at given step rather than looking ahead
- Risk of higher misclassification rates within terminal regions because of initial root splits
- Does not consider all possible partition permutations because too computationally expensive

Recursive Splitting → Overfitting

Top-down greedy approach tends to grow overly complex trees → overfit to training data



Solutions to Overfitting

Motivation: Need to find a way to create the simplest tree – or subtree – for given model.

Solutions to Overfitting

Motivation: Need to find a way to create the simplest tree – or subtree – for given model.

Last Class:

- Variable Selection Tools: subset selection, shrinkage/regularization, and dimensionality reduction
- Limits:
  - Those approaches are for parametric models.
  - Make assumptions that may not apply! (e.g. linearity)
- Need alternative!

Solutions to Overfitting

1. Find the optimal subtree by cross-validation

2. Stop growing the tree when the RSS doesn't drop by more than a designated threshold with each cut

3. Grow a very large tree T_0 and prune it back to obtain a simpler subtree

Solutions to Overfitting

1. Find the optimal subtree by cross-validation
   but there are too many partition possibilities so we could still overfit
2. Stop growing the tree when the RSS doesn't drop by more than a designated threshold with each cut

3. Grow a very large tree T_0 and prune it back to obtain a simpler subtree

Solutions to Overfitting

1. Find the optimal subtree by cross-validation

2. Stop growing the tree when the RSS doesn't drop by more than a designated threshold with each cut
   but too short-sighted since a bad split early on might be followed by a good split later on
3. Grow a very large tree T_0 and prune it back to obtain a simpler subtree

Pruning

The preferred solution to mitigate overfitting is **pruning**

Pruning

The preferred solution to mitigate overfitting is **pruning**

Main Idea: Use cross-validation to find the simplest tree which explains the most variation in the outcome

Types of Pruning:

1. Weakest link pruning (cost complexity pruning)
2. Reduced error pruning

Weakest Link Pruning

- Starting with largest tree (T_0), substitute a subtree with a leaf (terminal node) by minimizing:

$$\frac{RSS(T_1) - RSS(T_0)}{|T_0| - |T_1|}$$

- Iterate this pruning to obtain a sequence of trees $T_0, T_1, T_2, \ldots, T_m$ where T_m is the null tree
- Tree size differentiated by number of terminal nodes
- Select the optimal tree T_i via cross-validation

Visual Intuition for Weakest Link Pruning

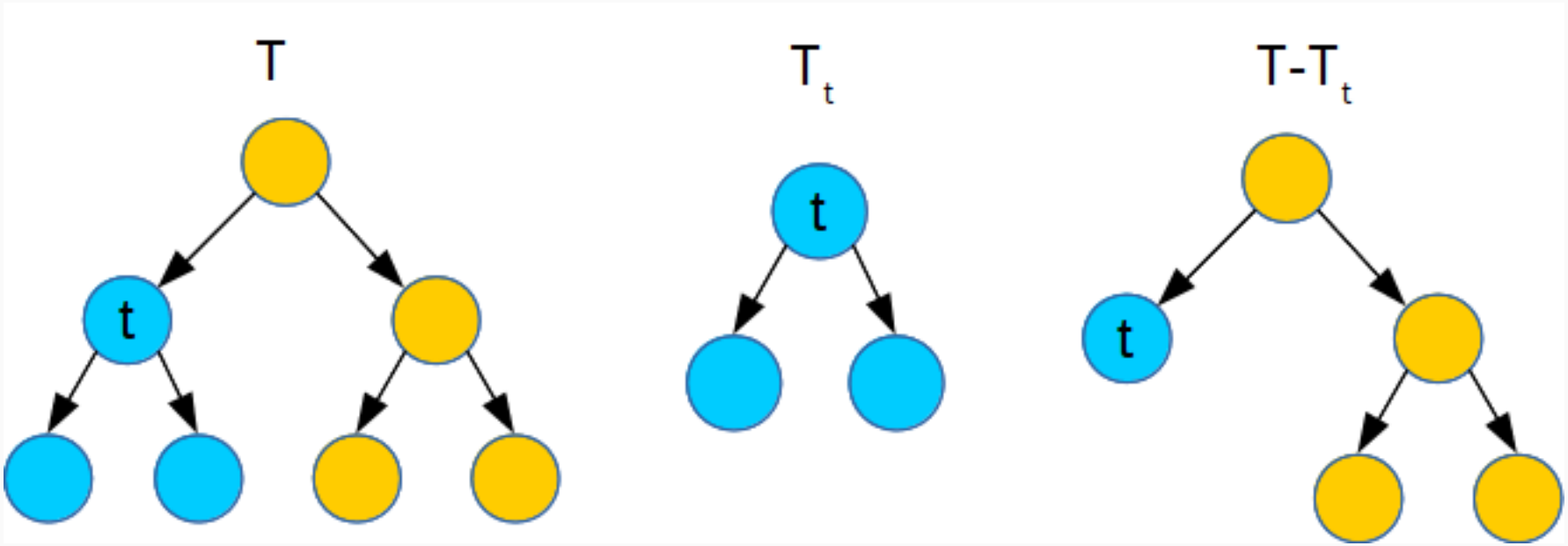


Figure 7: Substitute subtree with leaf and compare error

Weakest Link Pruning

- Solve the optimization problem:

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

- $|T|$ is the number of terminal nodes
- α is a **tuning parameter** which controls the bias-variance tradeoff (model fit relative to complexity)
  - When $\alpha = \infty$ we select the null tree T_m
  - When $\alpha = 0$ we select the full tree T_0
- Choose the optimal α (the optimal T_i) by cross-validation

Cross-Validation for Pruning

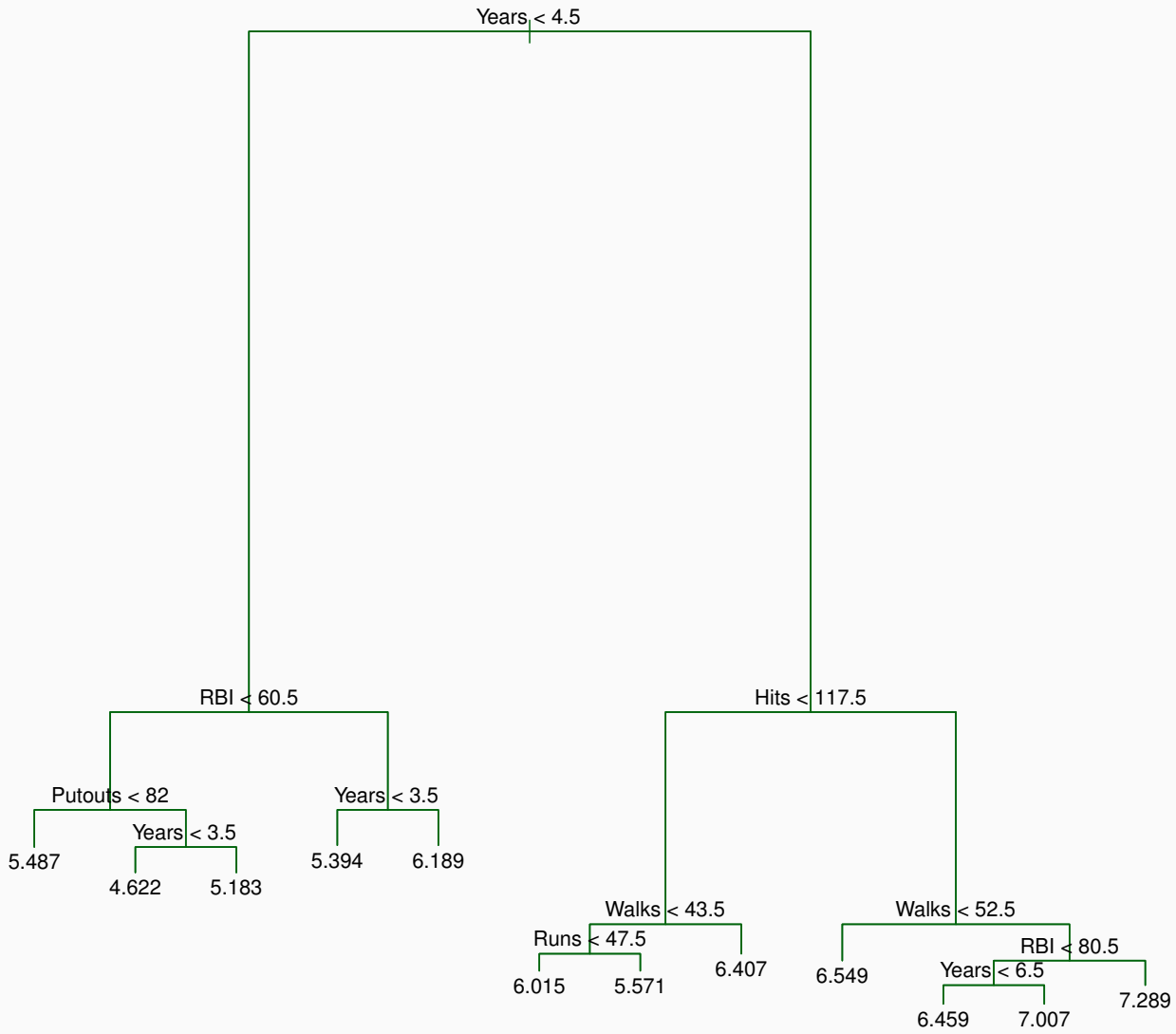
Procedure:

- Split the training observations into k folds
- For $k = 1, 2, \ldots, i$ use every fold except the ith fold
  - Construct a sequence of trees $T_1, T_2, \ldots, T_m$ for a range of values of α and find the prediction for reach region in each one
  - For each tree T_i calculate the RSS on the test (validation) set
- Select the parameter α that minimizes the average test error

Note: We do all fitting, including construction of the trees, using only the training data

Example: Unpruned Hitters Data

This is full tree T_0



Example: 10-fold Cross-Validation

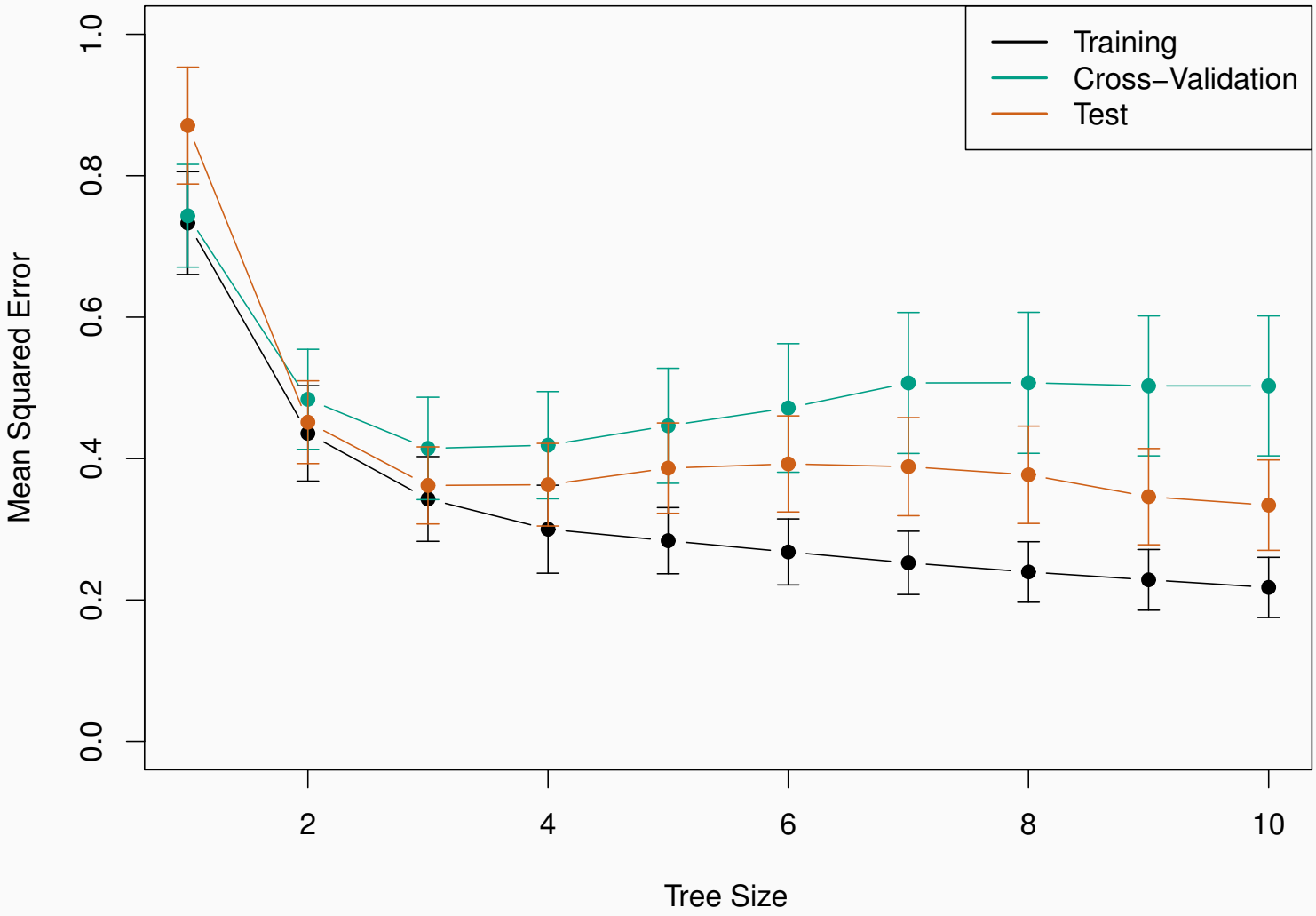
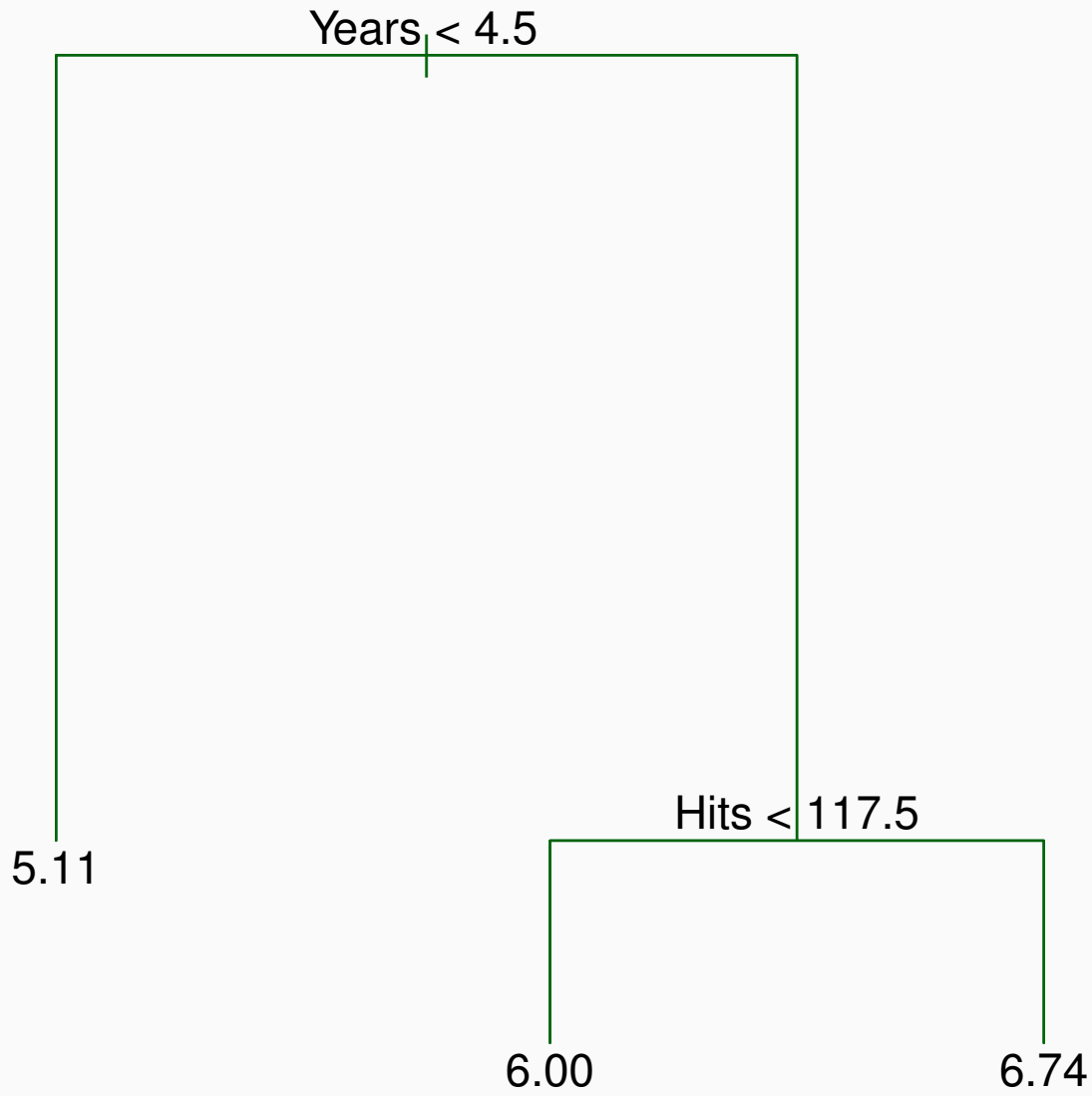


Figure 8: Training, CV, and test MSE as number of terminal nodes (tree size). Optimal CV error at tree size 3

Example: 10-fold Cross-Validation Pruned Tree

This is optimal tree $T_i = T_3$



Pruning

Advantages:

Disadvantages:

Pruning

Advantages:

- Creates more parsimonious tree
- Performs variable selection – removes irrelevant predictors

Disadvantages:

Pruning

Advantages:

- Creates more parsimonious tree
- Performs variable selection – removes irrelevant predictors

Disadvantages:

- Still greedy approach
- Partitioning data → different types of trees
- Risk high misclassification rate within terminal nodes

Classification Trees

Classification Trees

Estimation Goal:

- Partition set of predictors into different subregions to best explain variation across a categorical outcome.
- Predict the response by majority vote, i.e. pick the most common class in each region

Use cross-validation to minimize overfitting and produce optimal tree.

3 Loss Functions Available for Classification Tree

- Misclassification rate (aka 0-1 loss)
- Gini Index
- Cross-Entropy

Classification Tree Loss Function

- **Misclassification rate (aka 0-1 loss)**
- **Gini Index**
- **Cross-Entropy**

Classification Tree Loss Function

- **Misclassification rate (aka 0-1 loss)**

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} 1(y_i \neq \hat{y}_{R_m})$$

- **Gini Index**

- **Cross-Entropy**

Classification Tree Loss Function

- **Misclassification rate (aka 0-1 loss)**

- **Gini Index**

$$\sum_{m=1}^{|T|} \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- **Cross-Entropy**

Classification Tree Loss Function

- **Misclassification rate (aka 0-1 loss)**

- **Gini Index**

- **Cross-Entropy**

$$\sum_{m=1}^{|T|} \sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$$

Gini Index and Cross-Entropy

Motivation: Measure the **purity** of a region with new statistic $\hat{p}_{mk}$

- $\hat{p}_{mk}$ is the proportion of class k in R_m
- Gini Index

$$\sum_{m=1}^{|T|} \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

- Cross-Entropy

$$\sum_{m=1}^{|T|} \sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$$

Gini Index

Main Idea: Gini index captures the expected misclassification rate

Gini Index

Main Idea: Gini index captures the expected misclassification rate

- Measures the probability of incorrectly classifying a random element in the data if it were randomly labeled according to the class distribution in the dataset
- Range $\in [0, 1]$
  - Higher purity scores for more heterogeneous observations
  - Lower purity scores for more homogeneous observations
  - Score reaches its minimum (zero) when all cases in the node fall into a single outcome class
- Lower scores $=$ better model

Example of Gini Index

Motivation: Predict whether students are boys/girls based on above/below average performance

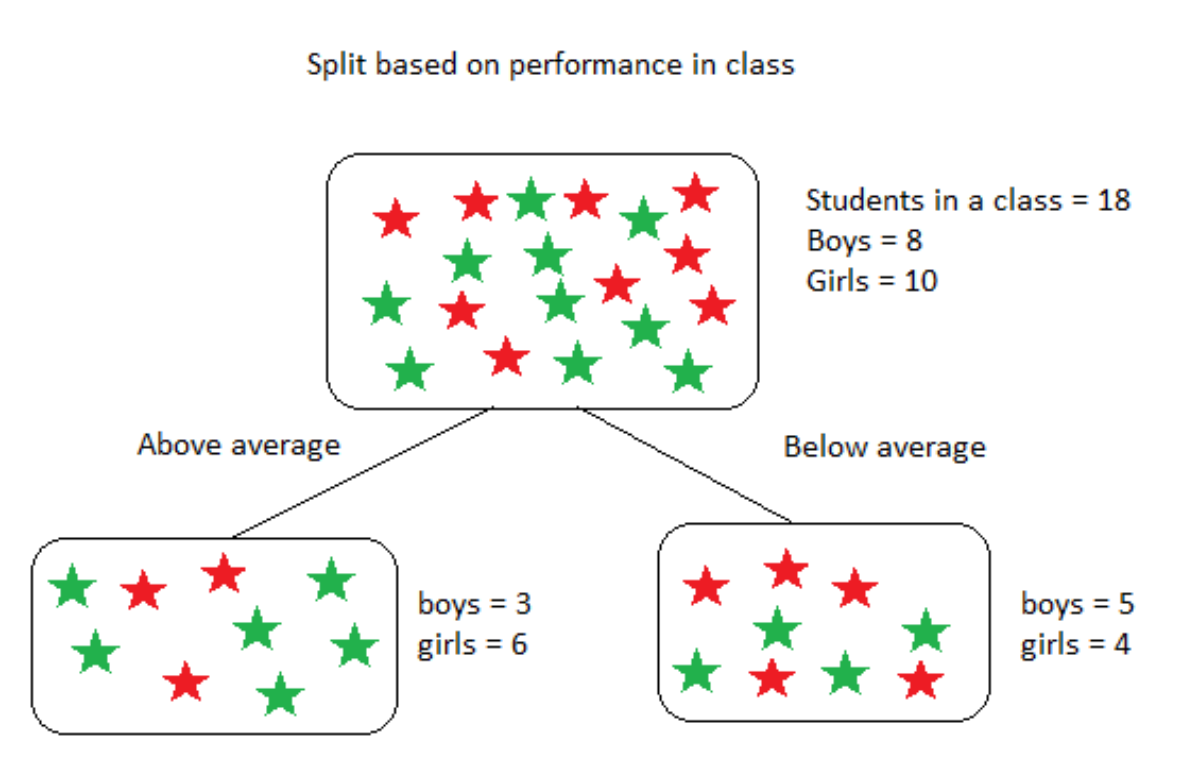


Figure 10: Decision Tree (2 terminal nodes) - note boys and girls distribution across nodes

Source: Bhuvaneswari Gopalan (2020)

Example of Gini Index

$$\sum_{m=1}^{|T|} \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

For "Above average" subnode:

Total Students = 9
Probability of boys = 3/9 = 0.33
Probability of girls = 6/9 = 0.66

Gini Impurity of "above average" subnode =

1 - [(0.33)*(0.33) + (0.66)*(0.66)] = 0.45

For "Below average" subnode:

Total Students = 9
Probability of boys = 5/9 = 0.55
Probability of girls = 4/9 = 0.44

Gini Impurity of "Below average" subnode =

1 - [(0.55)*(0.55) + (0.44)*(0.44) = 0.5

Weighted Gini Impurity of the split based on performance in class = (9/18) * 0.45 + (9/18) * 0.5
= 0.475

Figure 11: Sample Calculations. Find performance not great predictor → medium score size

Cross-Entropy

Main Idea: Cross-entropy is an alternate measure of the expected misclassification rate

- Higher cross-entropy scores for more heterogeneous observations; lower cross-entropy scores for more homogeneous observations
- It reaches its minimum (zero) when all cases in the node fall into a single outcome class
- Preferable to use the Gini index or cross-entropy for growing the tree, while using the misclassification rate when pruning the tree

Example of Classification Tree (ISLR)

Motivation: Predict heart disease using information about 13 possible characteristics

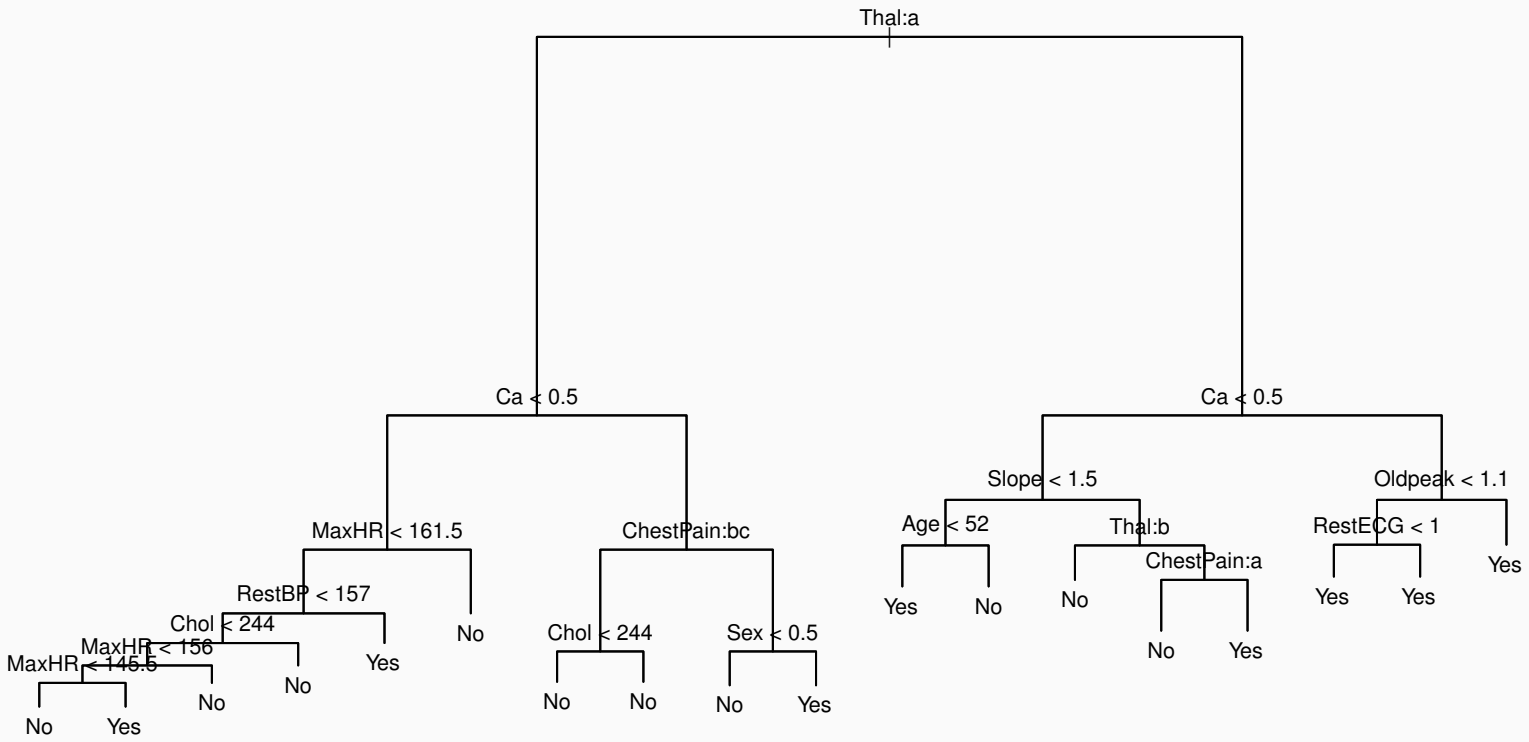


Figure 12: Unpruned tree. Terminal nodes show predicted response based on majority of observations in that region

Example of Classification Tree (ISLR)

Motivation: Predict heart disease using information about 13 possible characteristics

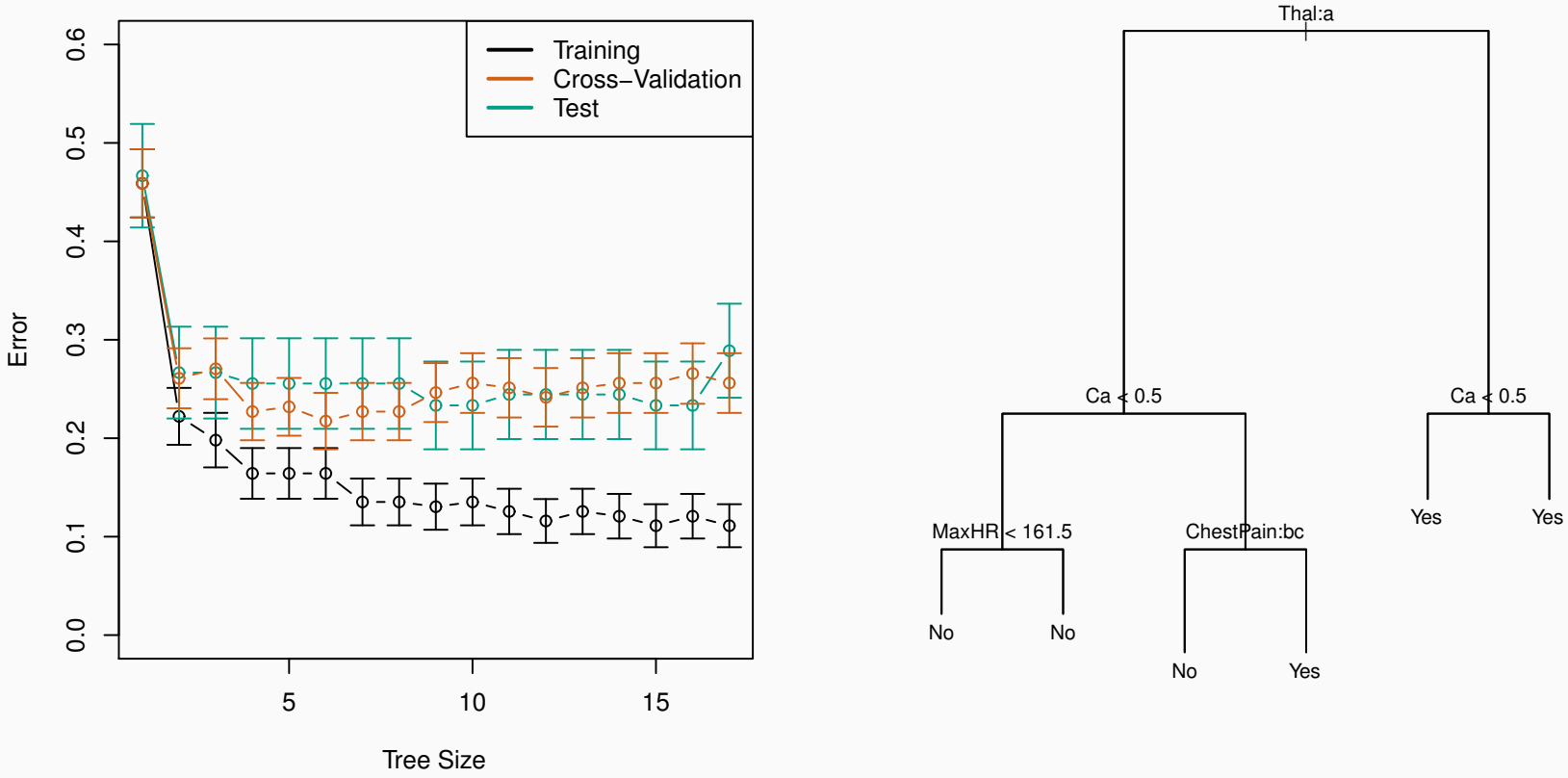


Figure 13: Cross-Validation and pruned tree

Classification Model Toolkit

- Logit
- LDA
- KNN
- CART

Comparison to Alternate Classification Models

When to use CART over Logit/LDA/KNN?

- Using new/original data
- Unsure of interactions or non-linearities
- Unsure of underlying f
- Missing data

Other Decision Tree Considerations

- **Categorical Predictors:** How does the model partition categorical predictors?
- **Continuous vs Binary Predictors:** How does the model treat continuous vs binary predictors?
- **Missing Data:** How does the model handle missing data?

Categorical Predictors

- If there are only 2 categories, then the split is obvious. We don't have to choose the splitting point s for a numeric variable

Categorical Predictors

- If there are only 2 categories, then the split is obvious. We don't have to choose the splitting point s for a numeric variable
- If there are more than 2 categories ...
  - Order the categories according to the average of the response, e.g. if c most common $\in [a, b, c]$, then set level $a > c > b$
  - Transform categories into numeric variable with this ordering and choose a splitting point s

Continuous vs Binary Predictors

- If the model contains one type of predictor (e.g. all continuous or all binary), then no worry

Continuous vs Binary Predictors

- If the model contains one type of predictor (e.g. all continuous or all binary), then no worry
- If the model contains different types of predictors, then worry ...

Continuous vs Binary Predictors

- If the model contains one type of predictor (e.g. all continuous or all binary), then no worry
- If the model contains different types of predictors, then worry ...
  - Model will prioritize continuous over binary variables → easier to partition because more possible splitting points
  - Resulting tree will have continuous variables higher on tree than binary variables
  - **Risk:** False impression continuous variables more important than binary variables!

Continuous vs Binary Predictors

Correct for mixture of continuous and binary predictors using **conditional inference tree**

- Uses alternative recursive partitioning algorithm
- Uses alternative significance test instead of Gini index to find relevant predictors

Missing Data

- **Problem:** If a sample is missing variable X_j and a tree contains a split according to $X_j > s$ then we may not be able to assign the sample to a region

Missing Data

- **Problem:** If a sample is missing variable X_j and a tree contains a split according to $X_j > s$ then we may not be able to assign the sample to a region
- **Solution 1:** Create contingent decision rules
  - Propagate a sample down the tree
  - When choosing a new split with variable X_j (growing the tree)
    - only consider the samples which have the variable X_j
    - in addition to choosing the best split, choose a second best split using a different variable, and a third best, ...
  - If the observation is missing a variable, try the second best decision rule. If missing both variables, try the third best decision rule, etc.
- **Solution 2:** Imputation ($\rightarrow$ next week)

Advantages and Disadvantages to Decision Trees

Advantages:

Disadvantages:

Advantages and Disadvantages to Decision Trees

Advantages:

- Easy to interpret
- Mimic human decision-making
- Easy to visualize
- Easily handle qualitative predictors and missing data

Disadvantages:

Advantages and Disadvantages to Decision Trees

Advantages:

- Easy to interpret
- Mimic human decision-making
- Easy to visualize
- Easily handle qualitative predictors and missing data

Disadvantages:

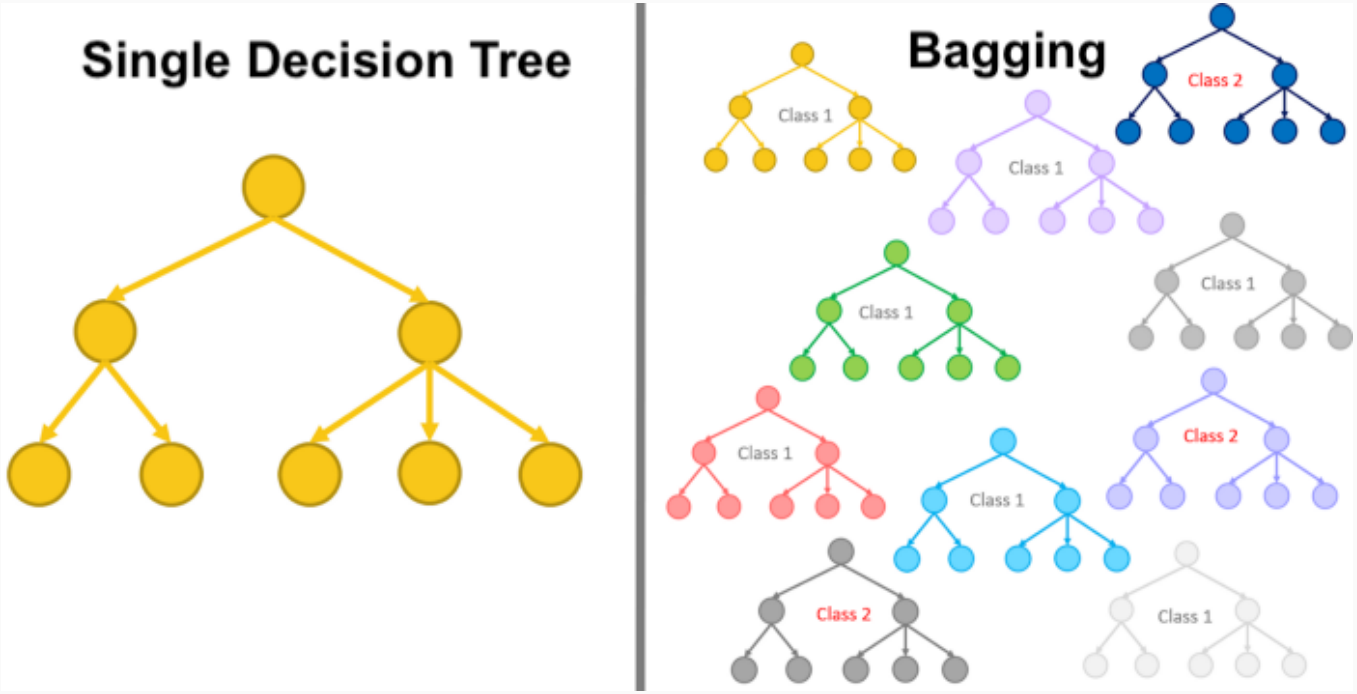
- Likely to overfit to noisy data
- Greedy algorithm can lead to worse fit
- High variance $\rightarrow$ different results

Bagging

Bagging

Bagging = Bootstrap Aggregating

Main Idea: This is an **ensemble method** in which we create lots of different decision trees using bootstrap and average the predictions together



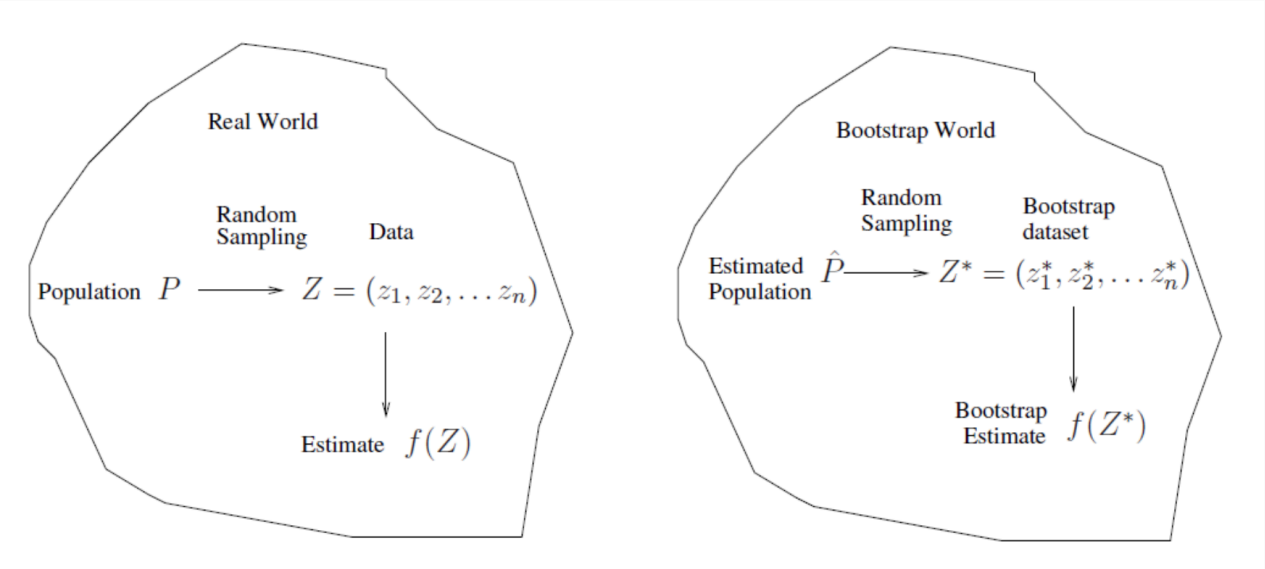
Recall: Bootstrap

In the bootstrap we resample the data by drawing B samples with replacement from training set to create new dataset Z^*

Estimate model on each sample of the data to get coefficient estimates $\hat{\alpha}$

Use variability in coefficient estimates to estimate standard error $\hat{\sigma}_\alpha$

- Original dataset: $x = c(x_1, x_2, \ldots, x_n)$
- Bootstrap samples:



Bagging Algorithm

Algorithm: Average model fit across B different models

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

You can apply bagging to any model type, e.g. Lasso:

- Let $\hat{f}_L^b$ be the prediction of a given Lasso (L) model applied to the b^{th} bootstrap sample
- Bagging Prediction:

$$\hat{f}_L^b = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_L^{*b}(x)$$

Bagging is very useful for decision trees

Bagging helps correct for top-down greedy algorithm which produces high variance CART. **Why?**

Bagging is very useful for decision trees

Bagging helps correct for top-down greedy algorithm which produces high variance CART. **Why?**

- Bagging reduces the variance associated with a single decision tree
- Bootstrap samples are like independent realizations of the data (asymptotics)
- Bagging amounts to averaging the fits from many independent datasets, which reduces the variance by a factor of $1/B$

How to estimate test error of bagging model?

- Cross-Validation:
  - Each time we draw a bootstrap sample, only use a fraction of the samples
  - Estimate test error on rest of the observations

How to estimate test error of bagging model?

- Cross-Validation:
  - Each time we draw a bootstrap sample, only use a fraction of the samples
  - Estimate test error on rest of the observations
- Out of Bag (OOB) error

Out of Bag Error

- For each sample x_i find the prediction $\hat{f}_{bag}(x)$ for all bootstrap samples which do not contain x_i
- Average these predictions to obtain $\hat{f}_{bag}^{oob}(x)$
- Compute the error $(f(x) - \hat{f}_{bag}(x))^2$
- Get MSE by averaging the error over all observations $i = 1, 2, \ldots n$

Out-of-Bag Error

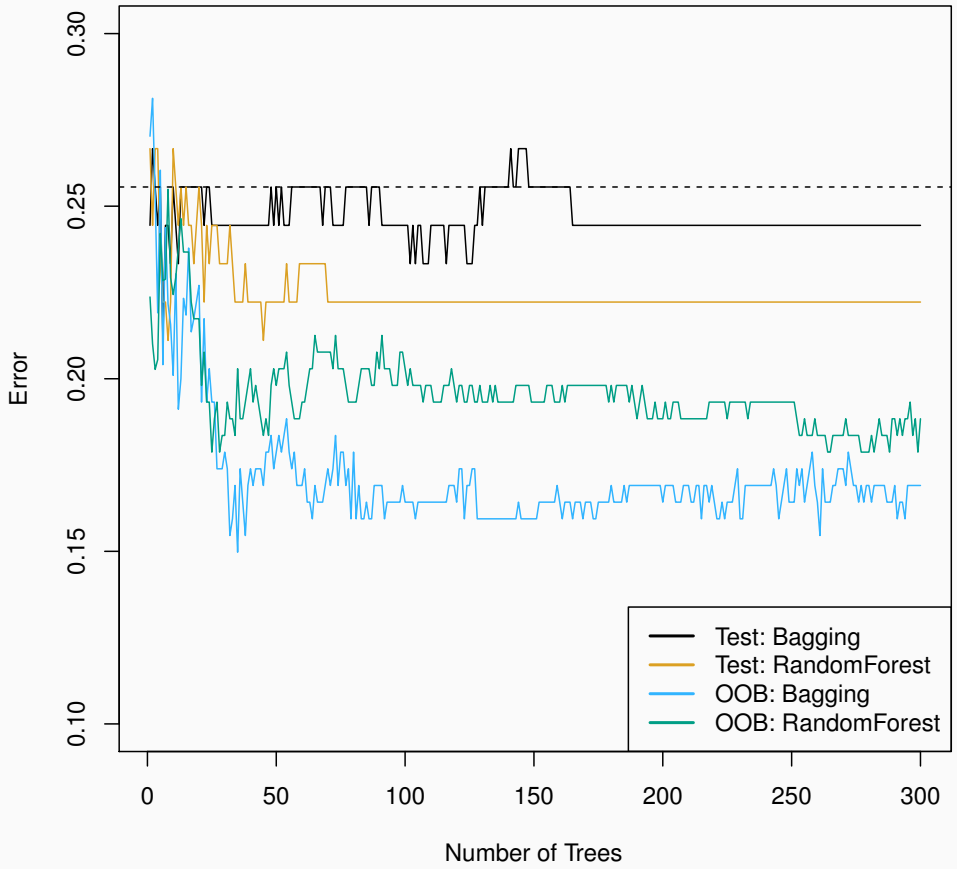


Figure 14: Comparison of CV Test Error and OOB Error as Function of B (number samples). Test error decreases as B increases.

Cross-Validation vs Out of Bag Error

- For small B should see virtually identical results
- OOB better for “large” datasets $\rightarrow$ computationally cheaper
- If B is sufficiently large, OOB error is virtually equivalent to LOOCV

Variable Selection

Problem: Every time we fit a decision tree to a bootstrap sample of data, we get a different tree $T^b \rightarrow$ loss of interpretability. How to identify relevant predictors?

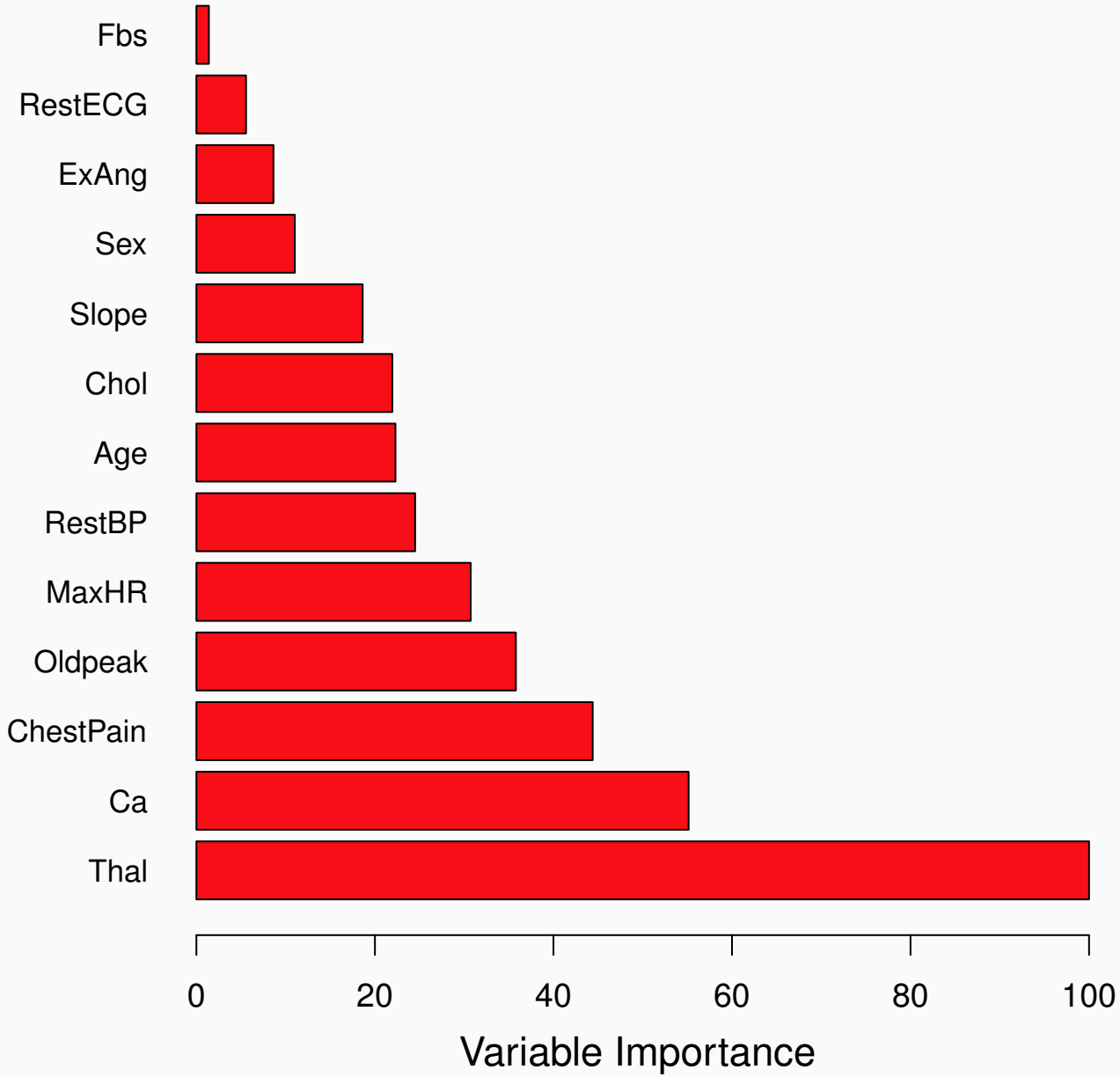
Variable Selection

Problem: Every time we fit a decision tree to a bootstrap sample of data, we get a different tree $T^b \rightarrow$ loss of interpretability. How to identify relevant predictors?

Solution: **Variable Importance Plot (VIP)**

- For each predictor, add up to the total amount by which the Gini index decreases every time we use a predictor in T^b (e.g. model performance worsens with that variable's exclusion)
- Average this total over each bootstrap estimate $T^1, T^2, \ldots, T^B$ (mean decrease in accuracy)

Example of Variable Importance Plot



Limit to Variable Importance Plots

Tells us relative importance, but does not tell us the direction of the relationship between the predictor and the outcome.

Limit to Variable Importance Plots

Tells us relative importance, but does not tell us the direction of the relationship between the predictor and the outcome. **Solution:** **Partial Dependence Plot**

- Show the marginal effect of a feature on an outcome
- Interpretation: Shows how the average prediction in dataset changes when j^{th} feature is changed
- Helps identify non-linearities

PDP Example

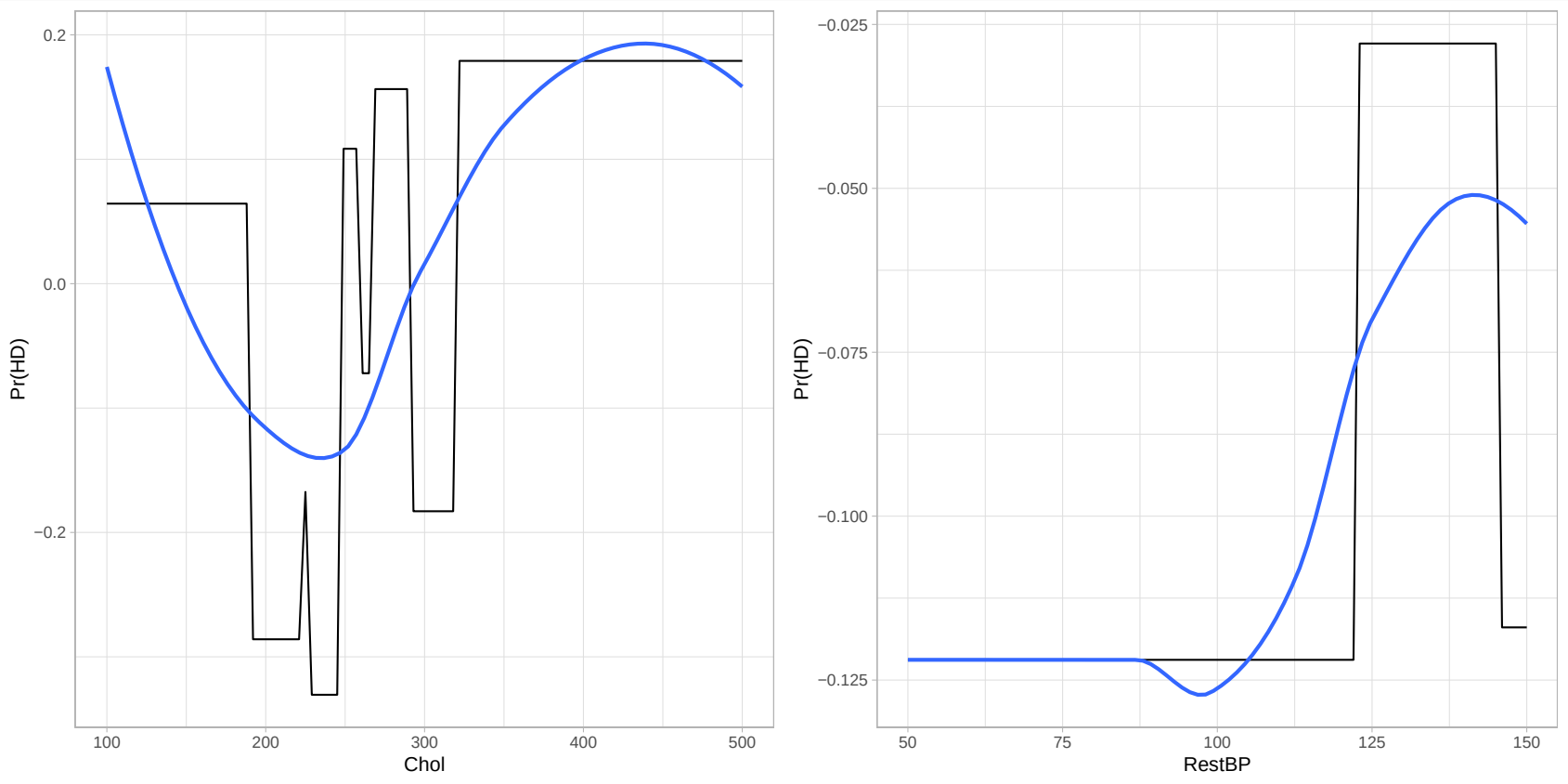


Figure 16: Partial Dependence Plot for Cholesterol Level and Rest BP

Advantages and Disadvantages to Bagging

Advantages:

Disadvantages:

Advantages and Disadvantages to Bagging

Advantages:

- Reduces Variance of Decision Tree
- Better Model Performance than Single Tree
- Performs variable selection
- Good Asymptotics: When n is large the empirical distribution is similar to the true distribution of the sample

Disadvantages:

Advantages and Disadvantages to Bagging

Advantages:

- Reduces Variance of Decision Tree
- Better Model Performance than Single Tree
- Performs variable selection
- Good Asymptotics: When n is large the empirical distribution is similar to the true distribution of the sample

Disadvantages:

- Trees produced by bootstrap look very similar
- Correlated trees produce highly correlated predictions $\rightarrow$ precise, but potentially biased results
- Reduction in variance $1/B$ not that much improvement over CART unless $B \approx \infty$

Conclusion

- Most common non-parametric approaches: KNN, CART, RF, Bagging, GBM, SVM
- Decision trees popular tool to visualize relationships and interactions between variables
- CART top-down greedy algorithm produces high variance results
- Bagging reduces some variance by bootstrapping multiple decision trees together and averaging results