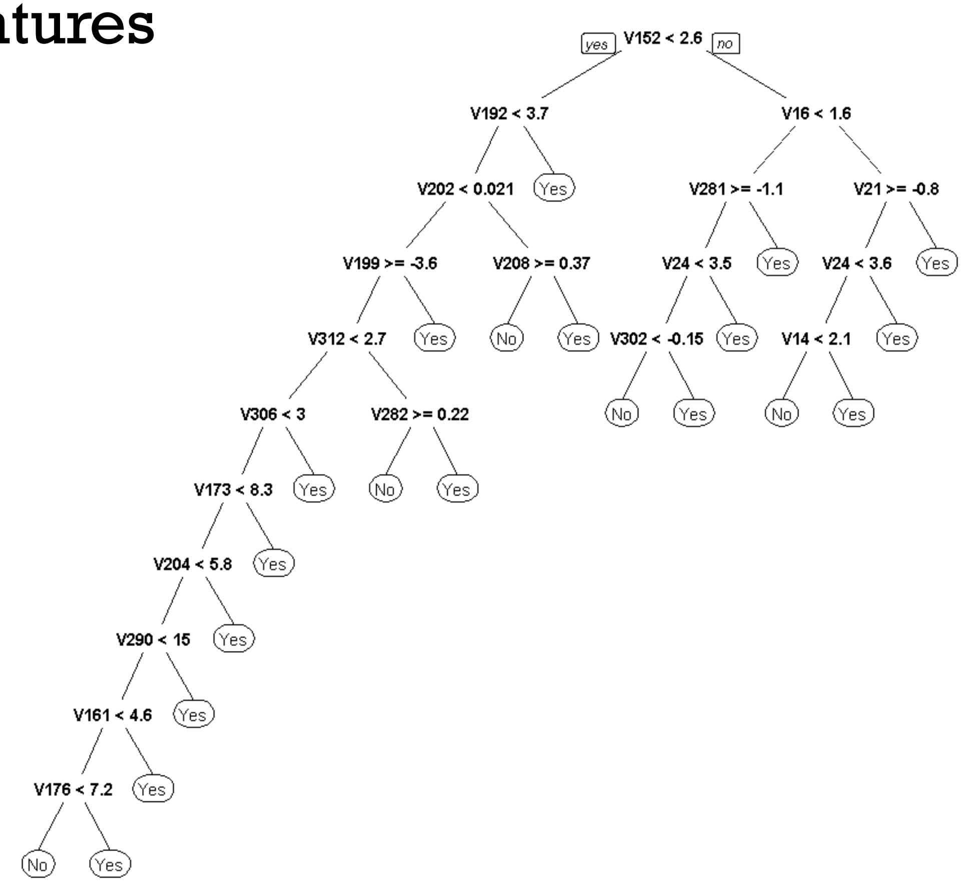
DECISION TREES

OUTLINE

- The Basics of Decision Trees
  - Regression Trees
  - Pruning Trees
  - Classification Trees
- Advanced Prediction Models (Ensemble Learning)
  - Bagging
  - Random Forests
  - Boosting

OUTLINE

- An example of making decisions based on tree representation of incident conditions (yes)
- Analyzing many features



INTRODUCTION

- Tree-based methods for regression and classification
- The idea is to stratify or segment the predictor space into a number of simple regions
- In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods

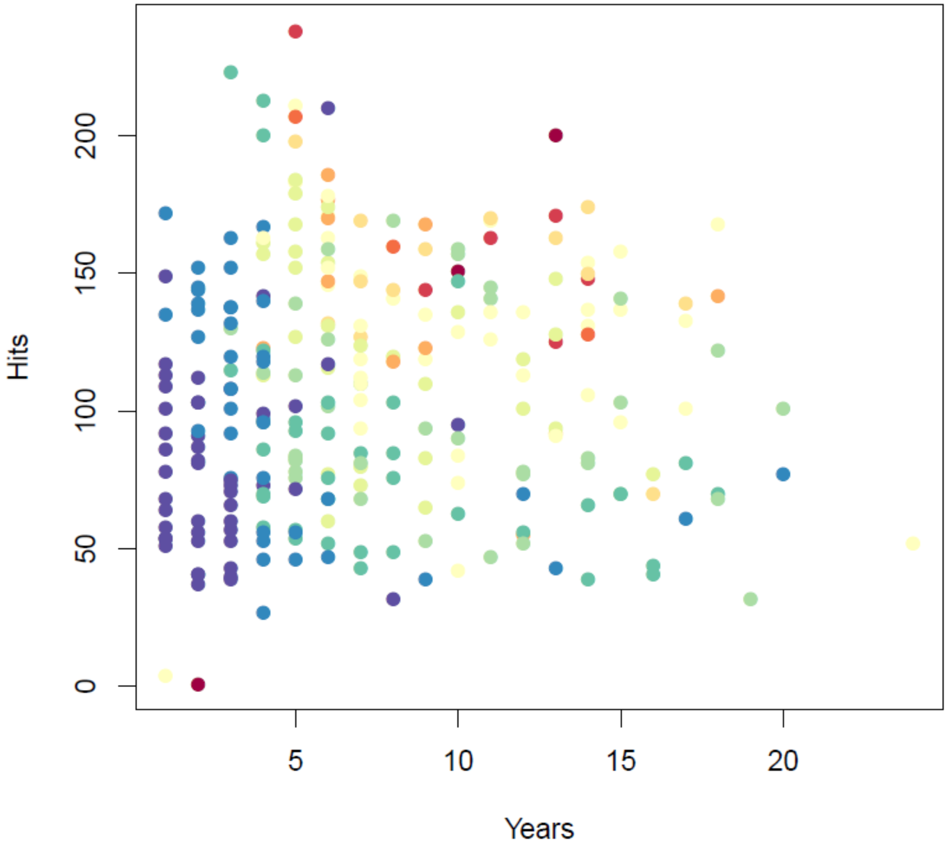
PROS AND CONS

- Tree-based methods are simple and useful for interpretation
- Typically, they are not competitive with the best supervised learning approaches in terms of prediction accuracy
- Hence we also discuss bagging, random forests, and boosting
- These methods grow multiple trees which are then combined to yield a single consensus prediction (ensemble learning)
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy (transforming weak learner into a stronger one), at the expense of some loss in interpretation

REGRESSION TREES

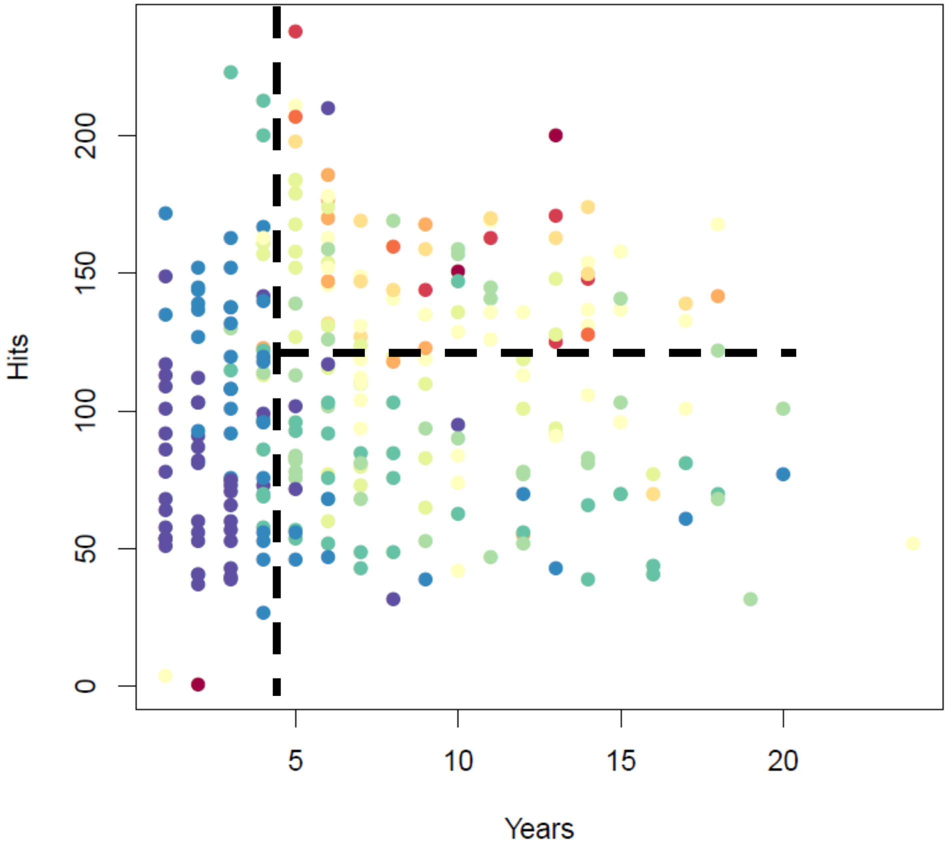
HITTERS DATA: VISUALIZATION

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

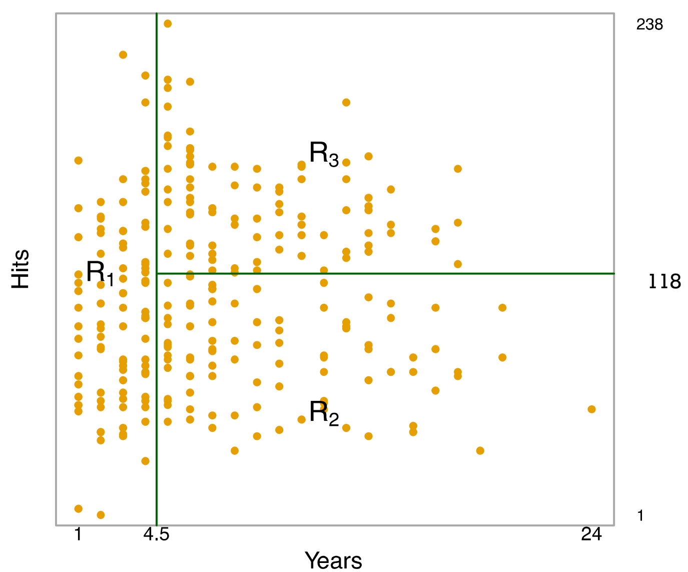


HITTERS DATA: VISUALIZATION

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

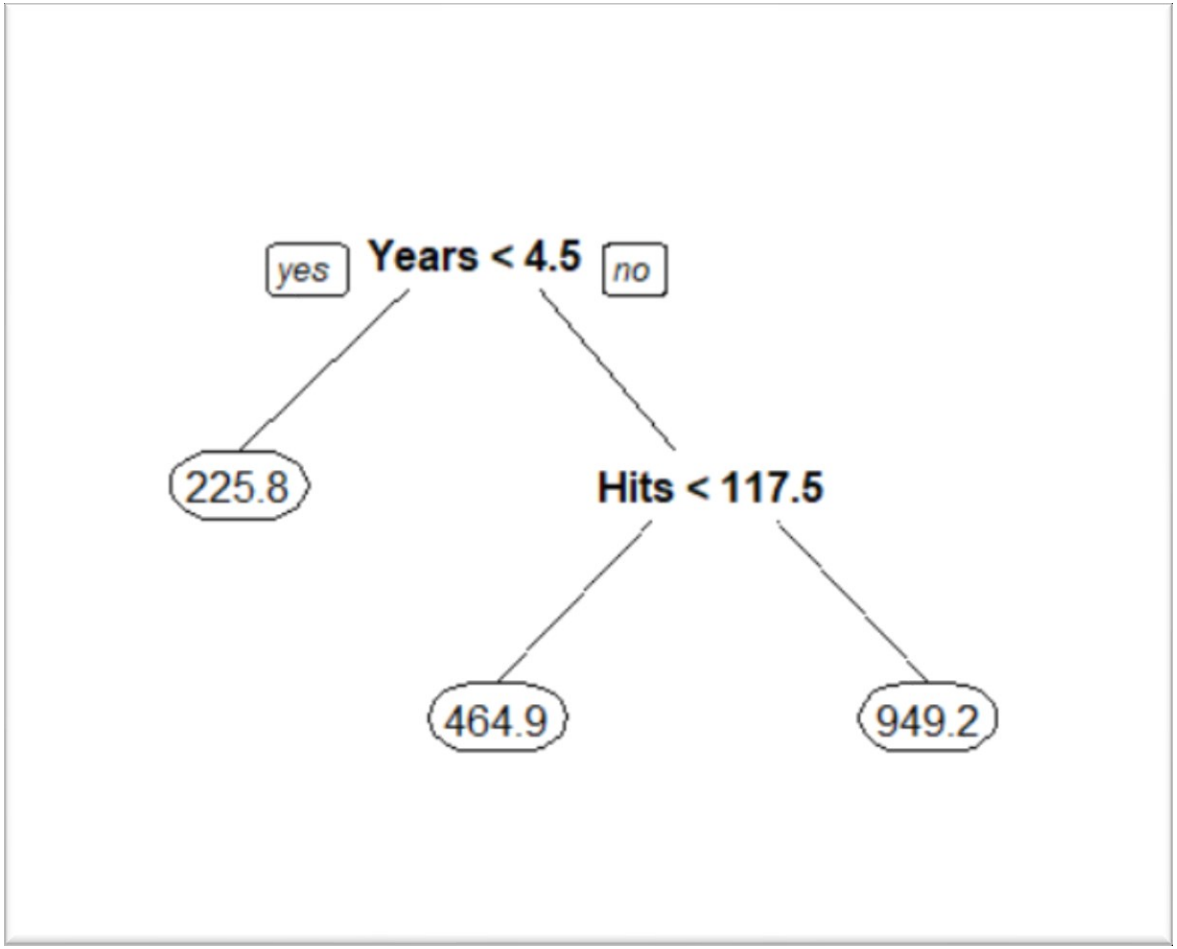


RESULTS



- Overall, the tree segments the players into three regions of predictor space:
- $R_1 = \{X \mid Years < 4.5\}$
- $R_2 = \{X \mid Years \geq 4.5, Hits < 118\}$
- $R_3 = \{X \mid Years \geq 4.5, Hits \geq 118\}$

HITTERS DATA: REGRESSION TREE



TERMINOLOGY

- The regions R_1, R_2, and R_3 are known as terminal nodes or leaves of the tree
- Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree
- The points along the tree where the predictor space is split are referred to as internal nodes
- In the hitters tree, the two internal nodes are indicated by the text $Years < 4.5$ and $Hits < 118$
- We refer to the segments of the trees that connect the nodes as branches

HITTERS DATA: INTERPRETATION

- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players
- Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary
- Among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries
- The predicted Salary for those players is given by the mean response value for the players in the data set belonging to the segments R_1, R_2, and R_3
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain

TREE-BUILDING PROCESS

1. We divide the predictor space - that is, the set of possible values for $X_1, X_2, \ldots, X_p$ - into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$

2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j

EXAMPLE

- Suppose that in Step 1, we obtain two regions, R_1 and R_2
- Suppose, the response mean of the training observations in the first region is 10
- Suppose, the response mean of the training observations in the second region is 20
- If for a given observation $X = x, x \in R_1$, we will predict a value of 10
- If for a given observation $X = x, x \in R_2$, we will predict a value of 20

TREE-BUILDING PROCESS – STEP 1

- In theory, the regions could have any shape
- However, we choose to divide the predictor space into high-dimensional rectangles, or boxes
- This is for simplicity and for ease of interpretation of the resulting predictive model

TREE-BUILDING PROCESS – STEP 1

- The goal is to find boxes $R_1, R_2, \dots, R_J$ that minimize the RSS

$$RSS = \sum_{j=1}^{J} \sum_{i:x_i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the j^{th} box

TREE-BUILDING PROCESS – STEP 1

- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes
- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting
- The approach is top-down because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree
- It is greedy because at each step of the tree-building process, the best split is made at particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step

TREE-BUILDING PROCESS – STEP 1

- We first select the predictor X_j and the cut-point s such that splitting the predictor space into the regions

$$R_1(j, s) = \{X | X_j < s\}$$

and

$$R_2(j, s) = \{X | X_j \geq s\}$$

leads to the greatest possible reduction in RSS

TREE-BUILDING PROCESS – STEP 1

- Before splitting

$$err_0 = \sum_{i:x_i \in R} (y_i - \hat{y}_R)^2$$

- After splitting

$$err_1 = \sum_{i:x_i \in R_1(j,s)} \left(y_i - \hat{y}_{R_1}\right)^2 + \sum_{i:x_i \in R_2(j,s)} \left(y_i - \hat{y}_{R_2}\right)^2$$

TREE-BUILDING PROCESS – STEP 1

- Actually, we seek the values of j and s that minimize err_1 and include X_j in a tree only if the decrease of the error is significant
- Finding the values of j and s that minimize err_1 can be done quite quickly, especially when the number of features p is not too large

TREE-BUILDING PROCESS

- Next, we repeat the process, looking for the best predictor and best cut-point in order to split the data further so as to minimize the RSS within each of the resulting regions
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions
- Again, we look to split one of these three regions further, so as to minimize the RSS
- The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations

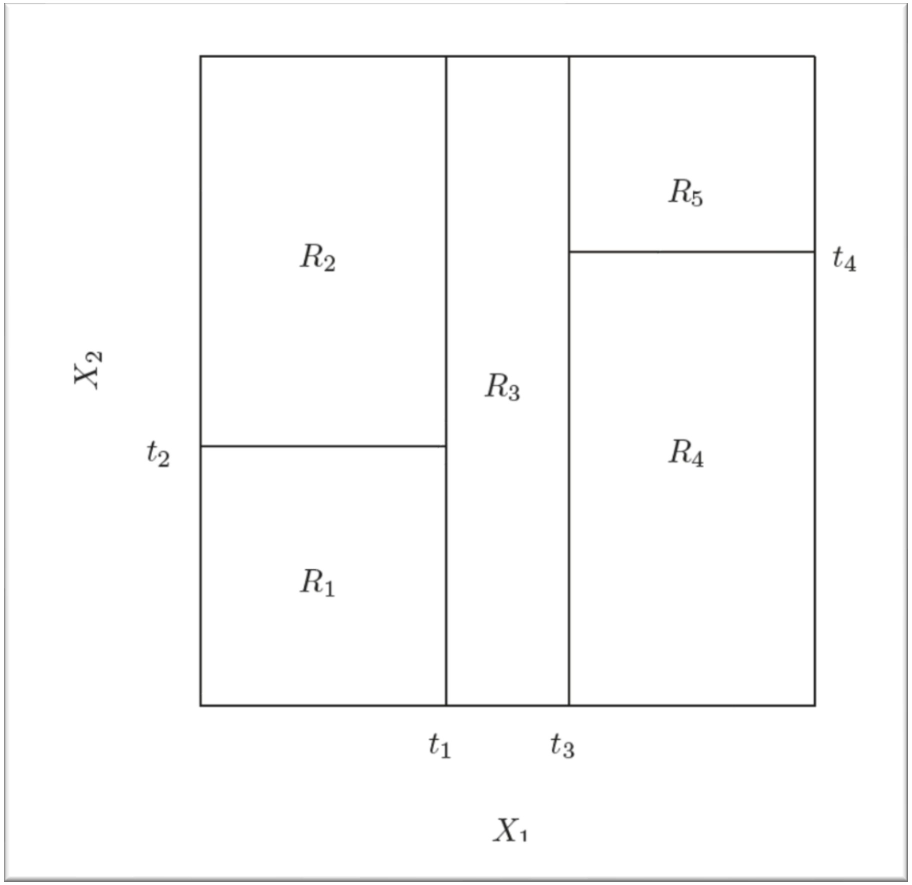
TREE-BUILDING PROCESS – STEP 2

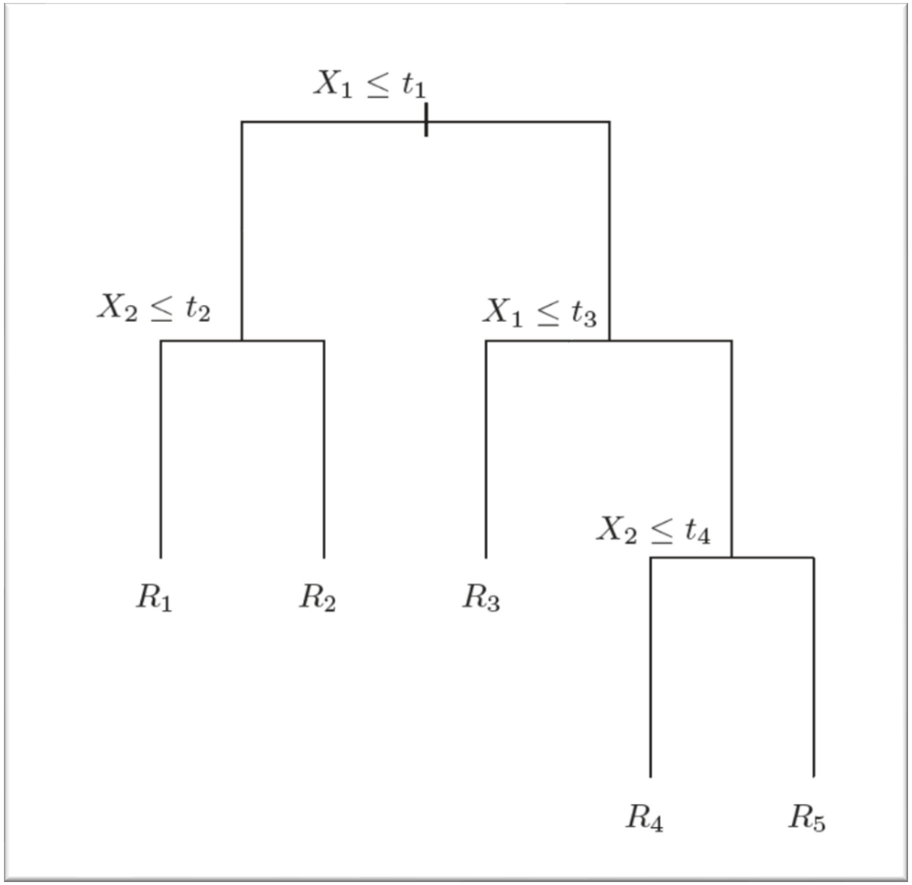
- Once the regions

$$R_1, \dots, R_J$$

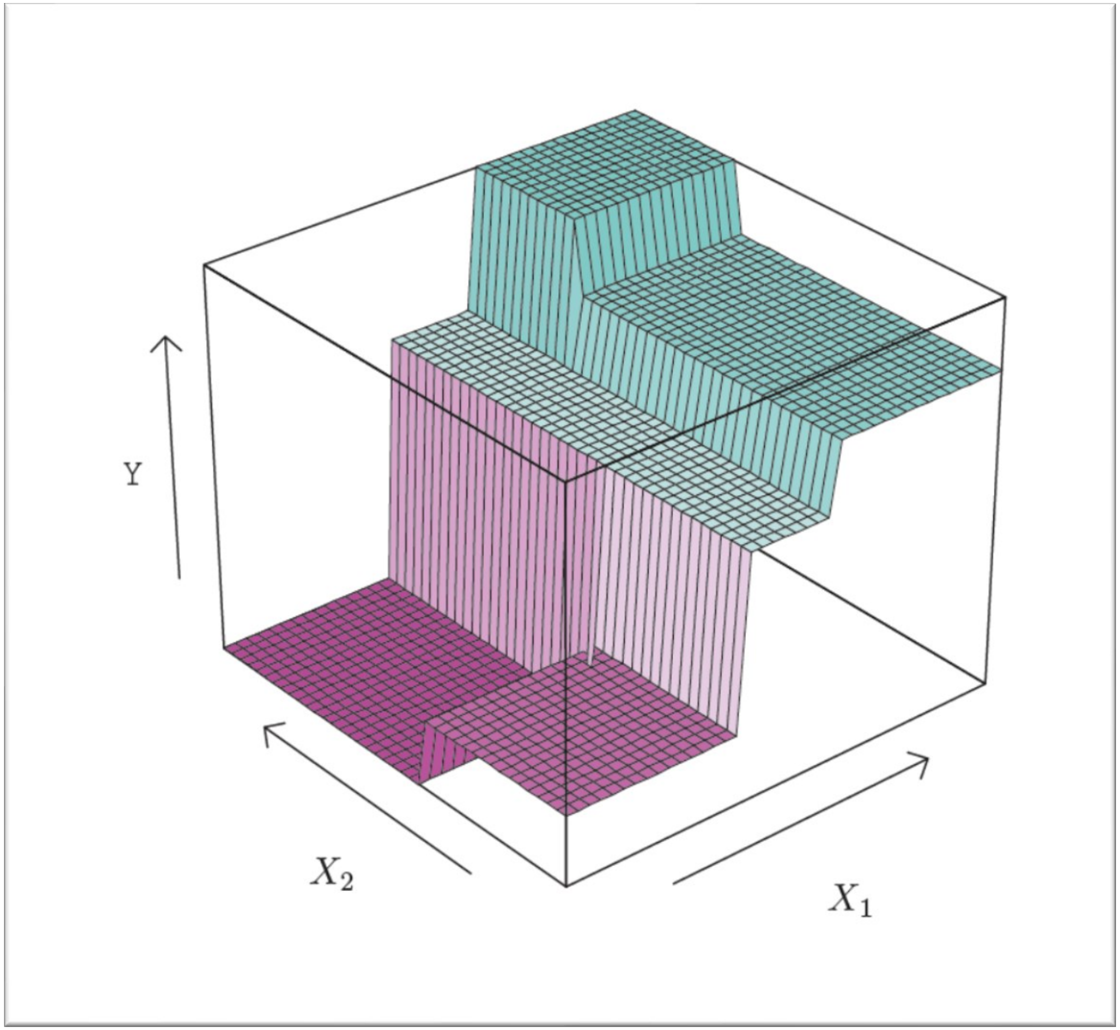
have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs

EXAMPLE OF A RECURSIVE BINARY SPLITTING

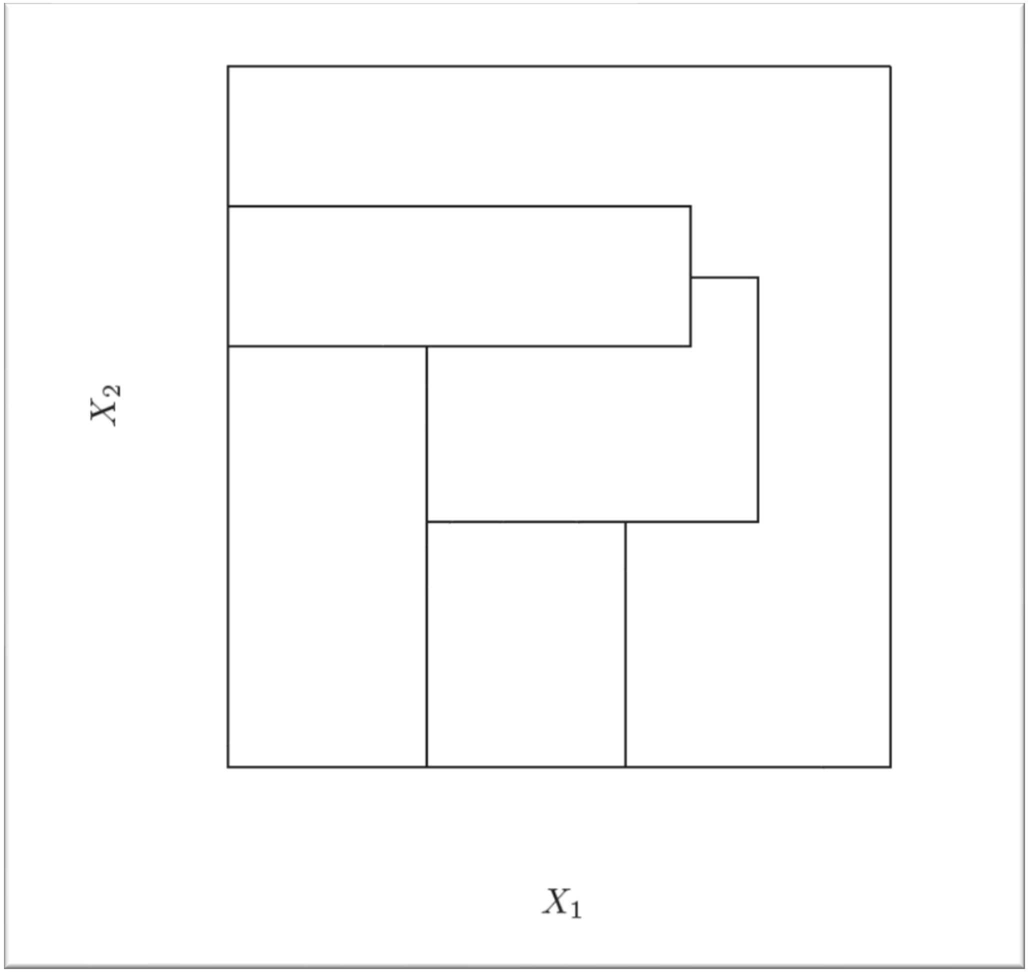




A PERSPECTIVE PLOT OF THE PREDICTION SURFACE CORRESPONDING TO THE PREVIOUS TREE



A PARTITION OF TWO-DIMENSIONAL FEATURE SPACE THAT COULD NOT RESULT FROM RECURSIVE BINARY SPLITTING



TREE PRUNING

IMPROVING TREE ACCURACY

- A large tree (with many terminal nodes) may tend to overfit the training data, leading to poor test set performance
- This is because the resulting tree might be too complex
- Generally, we can improve accuracy by pruning the tree i.e. cutting off some of the terminal nodes
- A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias

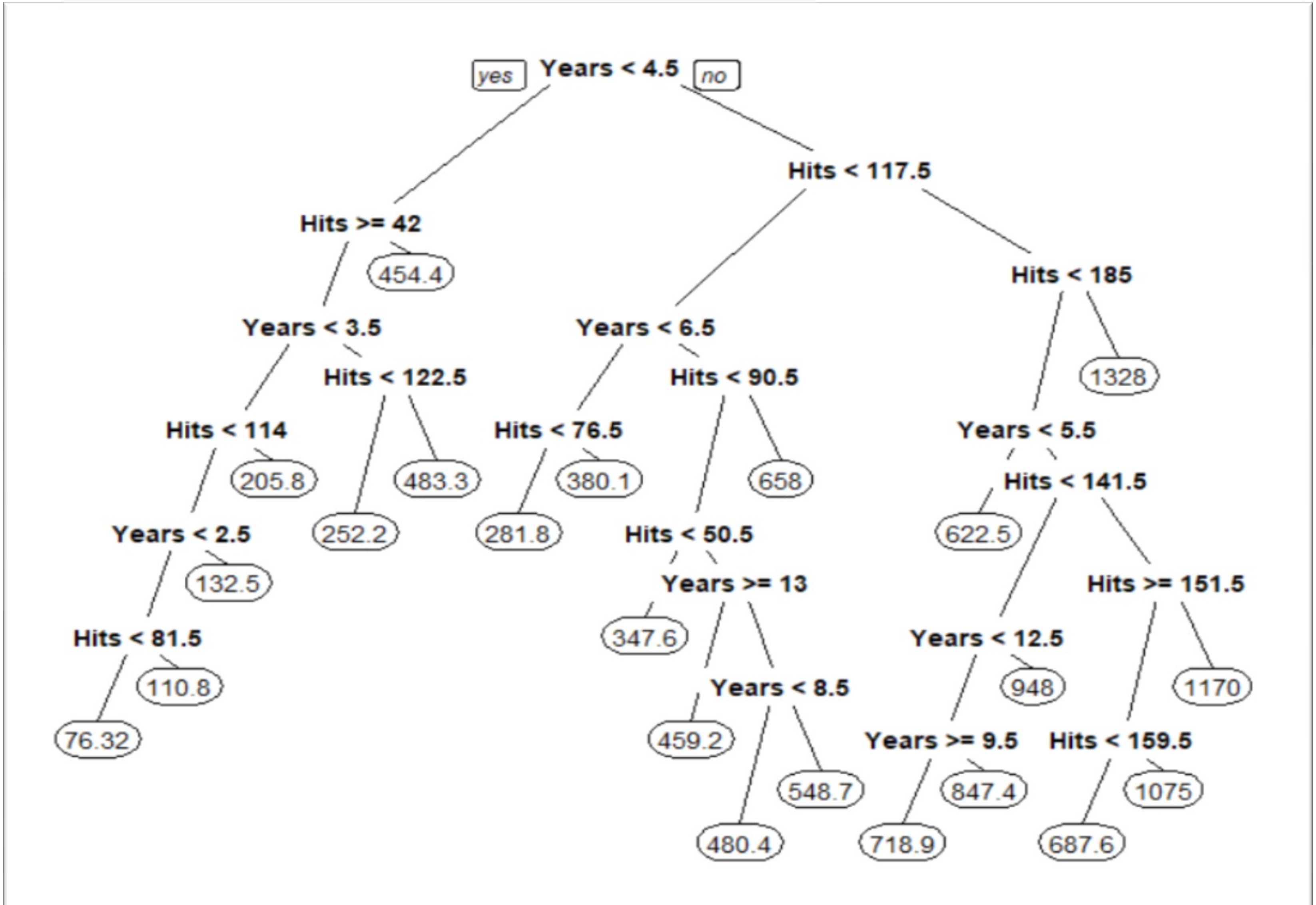
IMPROVING TREE ACCURACY

- One possible alternative to this process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold
- This strategy will result in smaller trees, but is too short-sighted
- A seemingly worthless split early on in the tree might be followed by a very good split - that is, a split that leads to a large reduction in RSS later on

IMPROVING TREE ACCURACY

- A better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree
- How do we know how far back to prune the tree?
- Given a subtree, we can estimate its test error using cross validation
- However, estimating the cross-validation error for every possible subtree would be too cumbersome, since there is an extremely large number of possible subtrees
- Instead, we need a way to select a small set of subtrees for consideration

BIG REGRESSION TREE - T_0



COST COMPLEXITY PRUNING

- Cost complexity pruning - also known as weakest link pruning - is used to do this
- Rather than considering every possible subtree, we consider a sequence of trees indexed by a tuning parameter α
- For each value of α there exists a subtree $T \in T_0$ such that

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

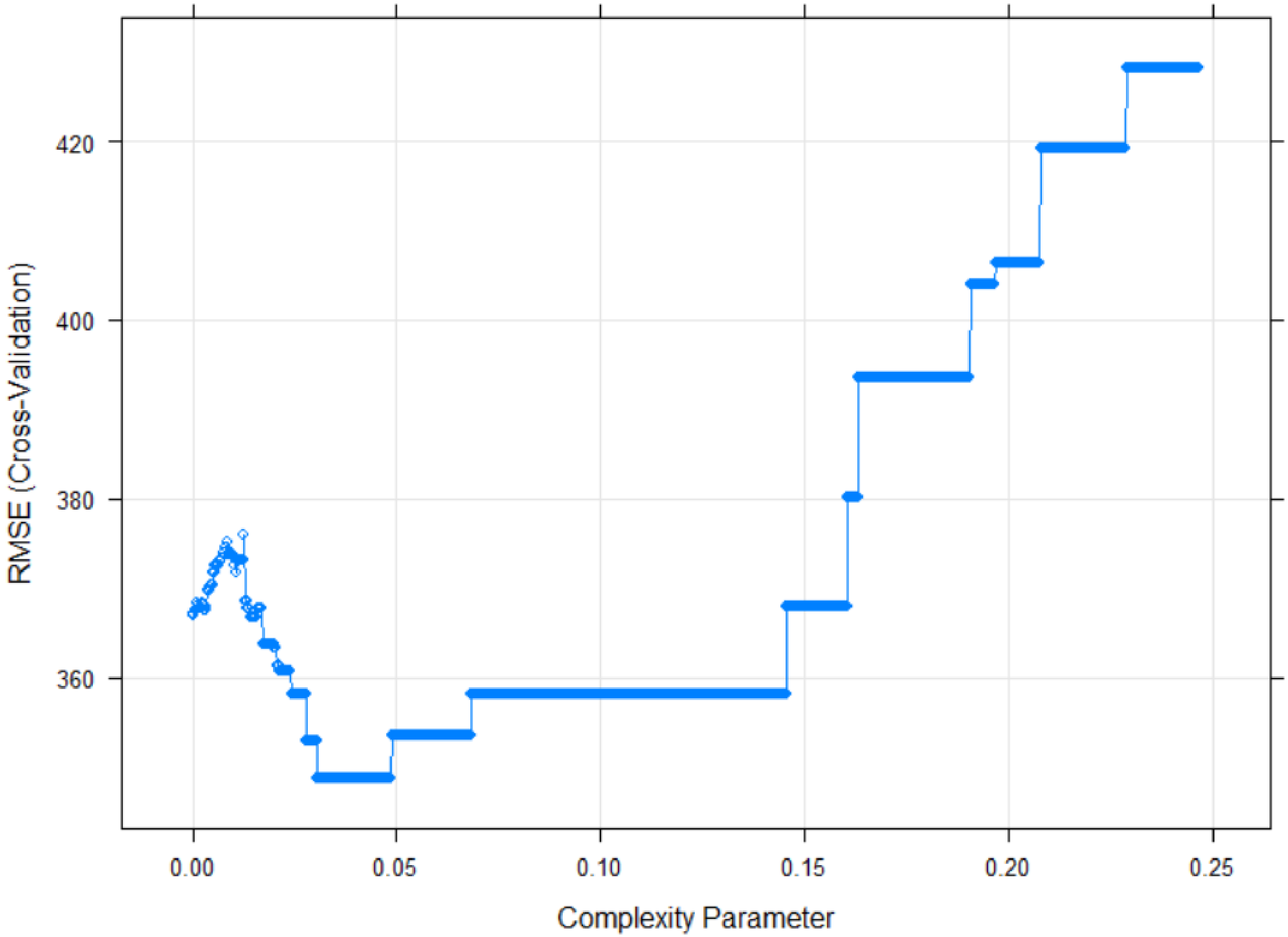
CHOOSING THE BEST SUBTREE

- The tuning parameter controls a trade-off between the subtree's complexity and its fit to the training data

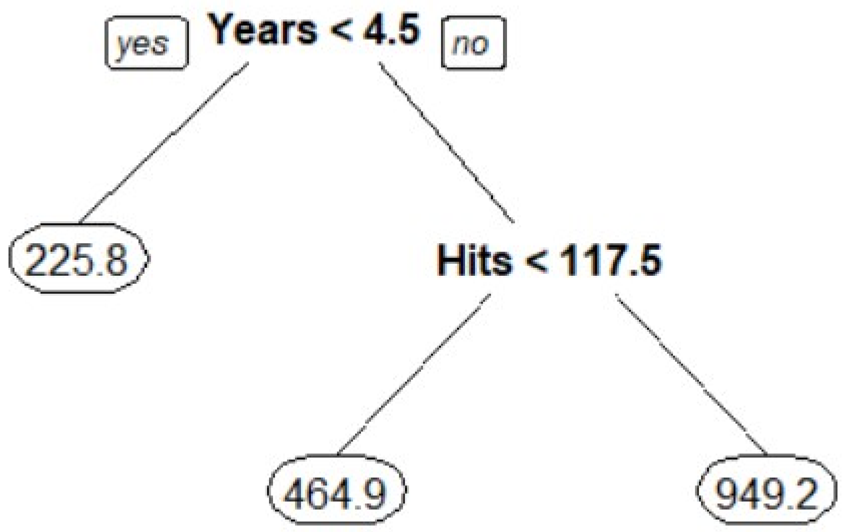
- We select an optimal value $\hat{\alpha}$ using cross-validation

- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$

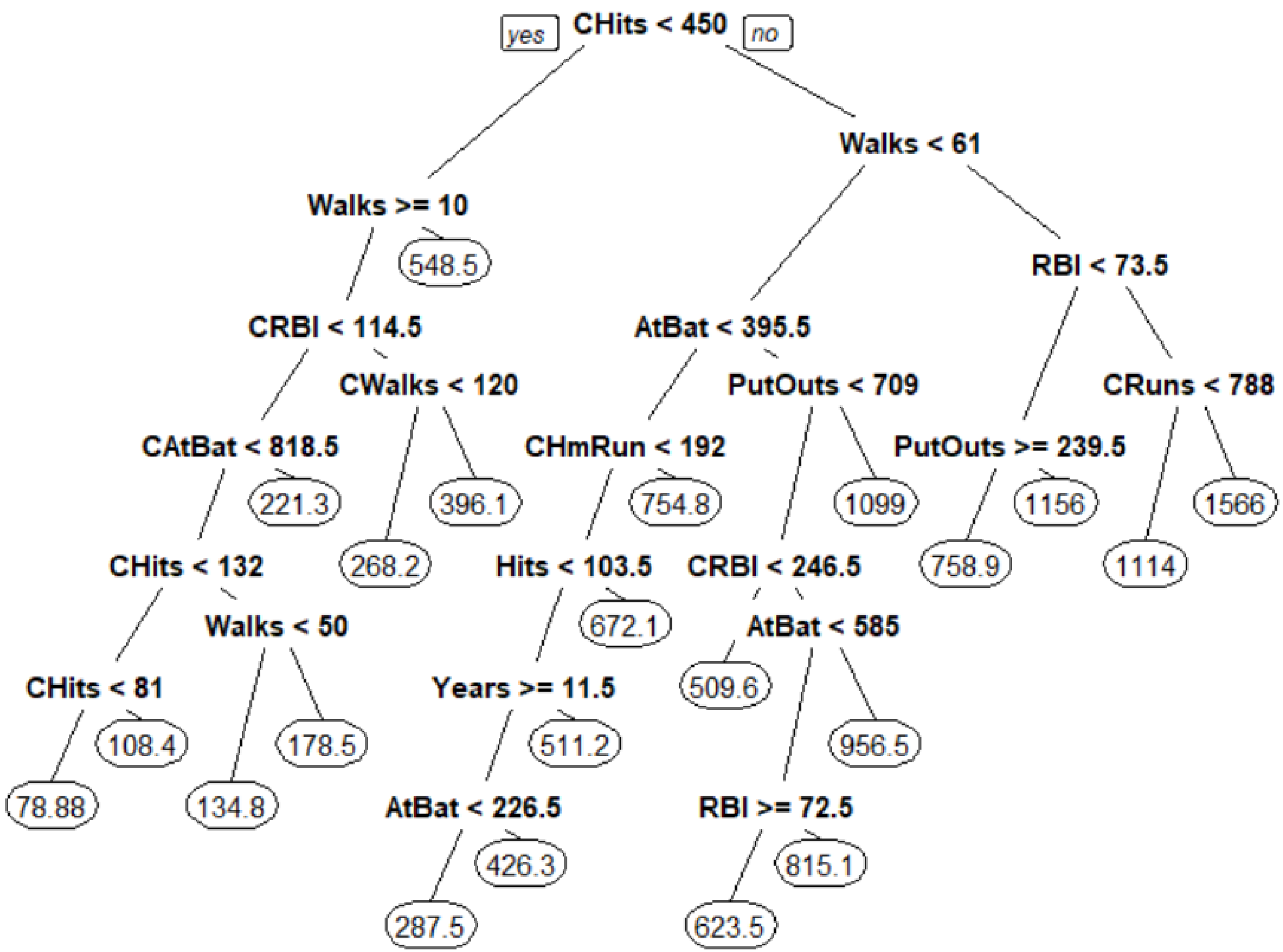
HITTERS DATA: CV RESULTS



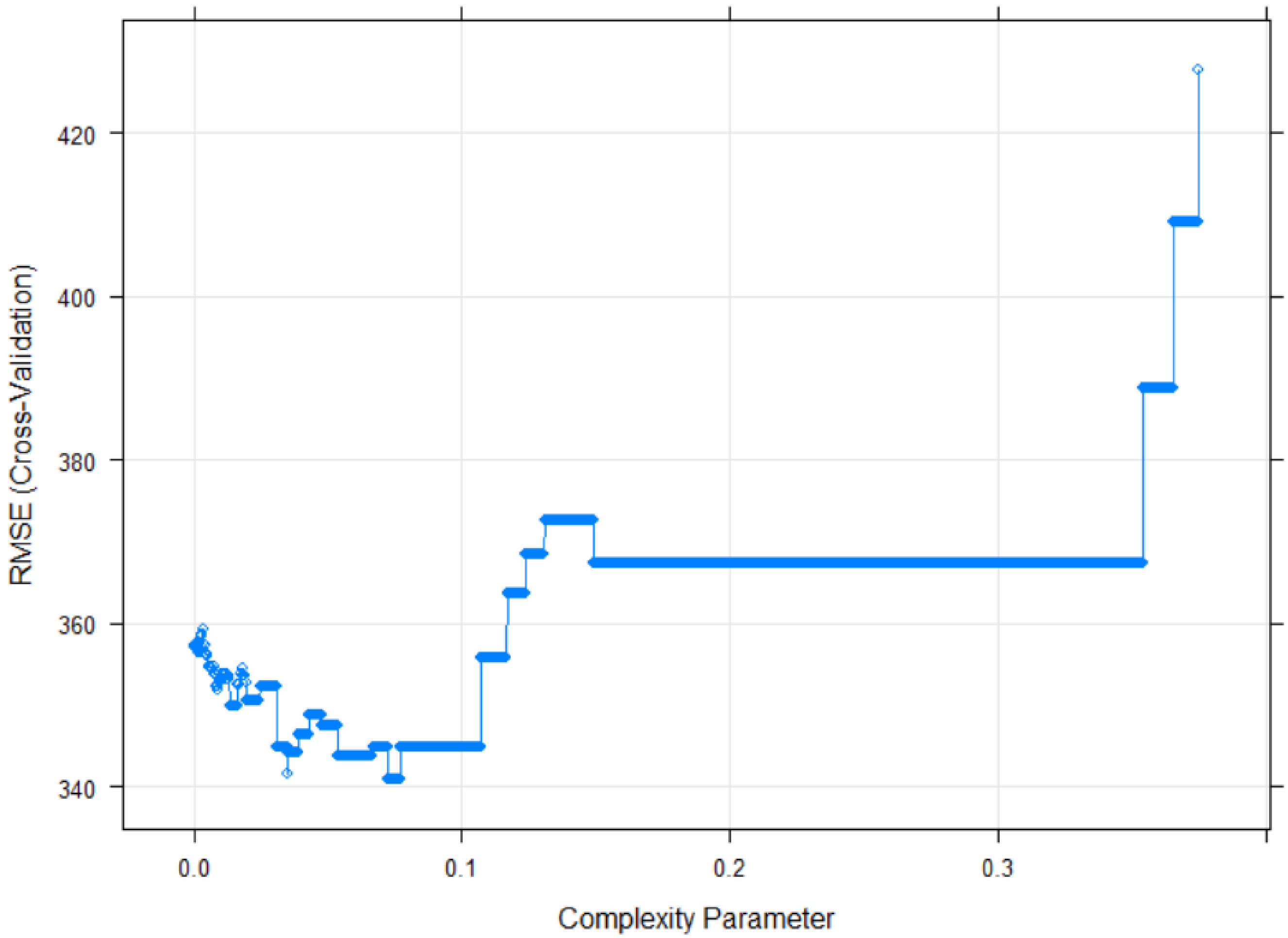
HITTERS DATA: REGRESSION TREE



ENTIRE HITTERS DATA



OPTIMAL TREE PRUNING



OPTIMAL TREE