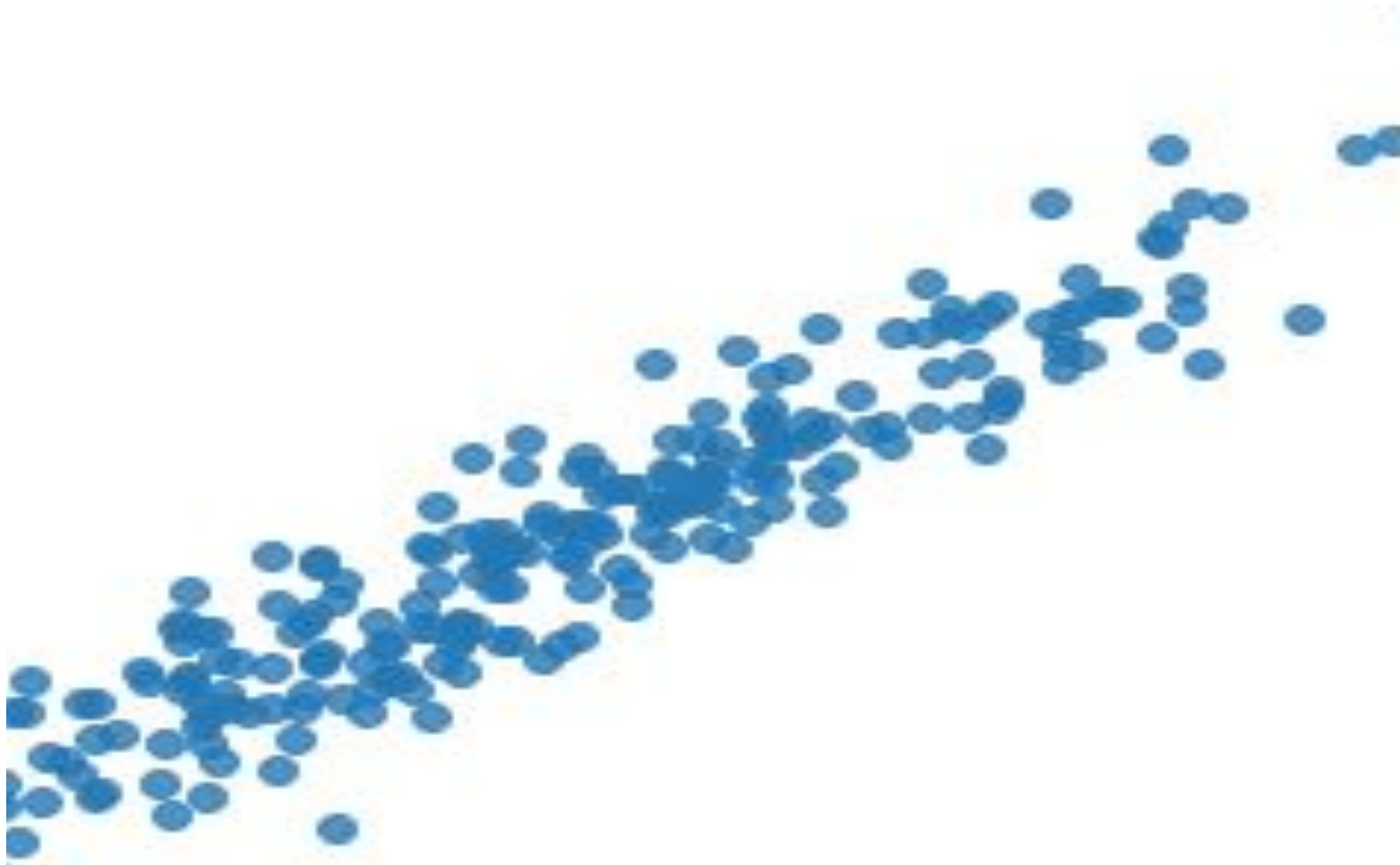


Module 2: Statistical Machine Learning

Introduction,
Regression, and
Classification,
Decision Trees,
Random Forests



Reference: James, G., Witten, D., Hastie, T., Tibshirani, R. (2017).
An Introduction to Statistical Learning: with Applications in R.,
Springer.)



Tree-Based Methods

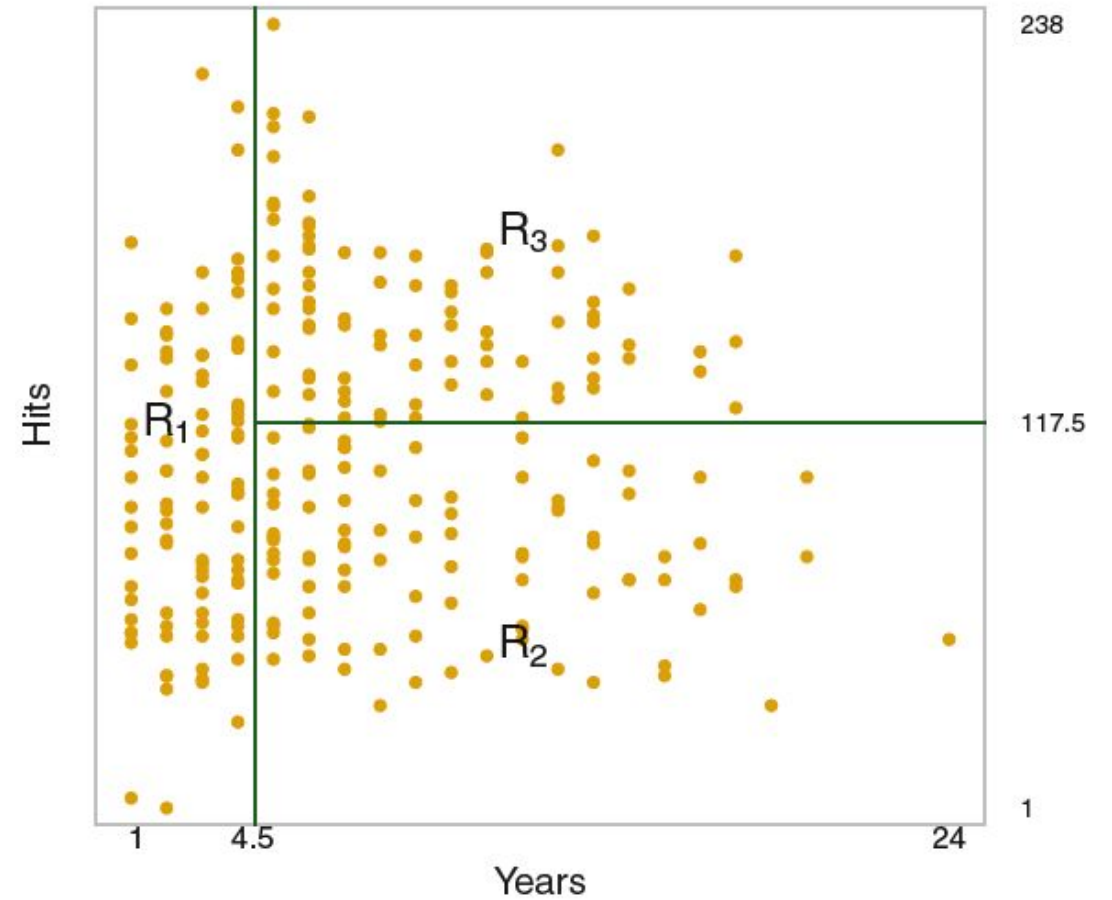
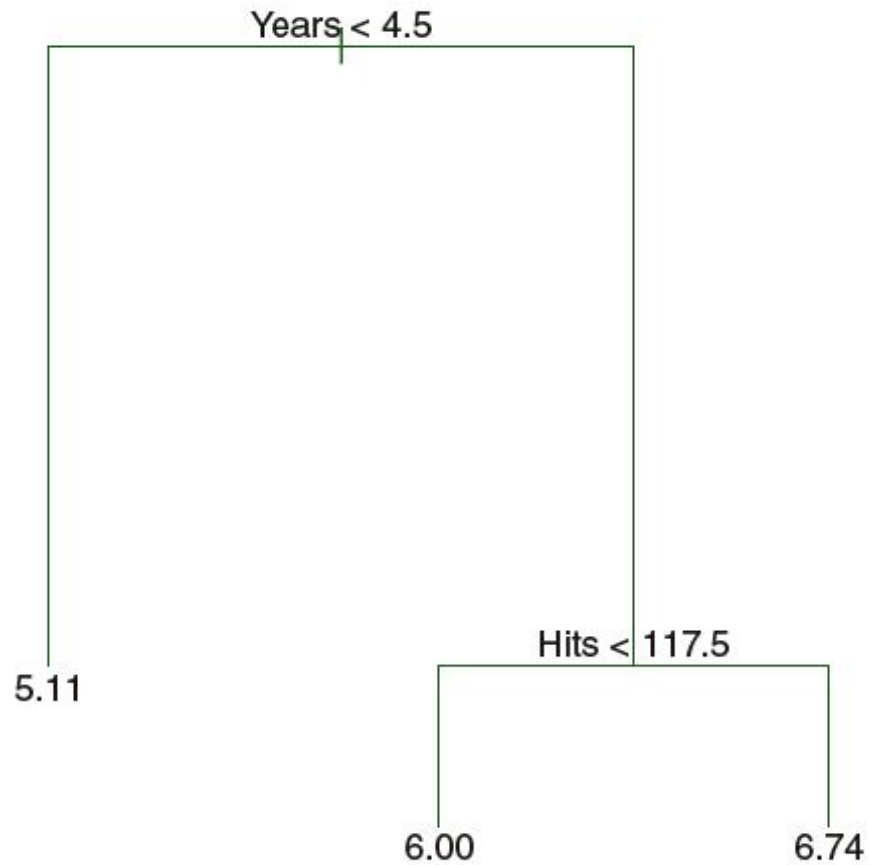
- Tree-based methods for regression and classification.
- Involve **stratifying or segmenting** the predictor space into a number of simple regions.
- In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs.
- The predictor space can be summarized in a tree - decision tree methods.



The Basics of Decision Trees

Decision trees can be applied to both regression and classification problems

Regression Trees





Regression Trees

$$R1 = \{X \mid \text{Years} < 4.5\}$$

$$R2 = \{X \mid \text{Years} \geq 4.5, \text{Hits} < 117.5\}$$

$$R3 = \{X \mid \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$$

The predicted salaries for these three groups are

$$\$1,000 \times e^{(5.107)} = \$165,174,$$

$$\$1,000 \times e^{(5.999)} = \$402,834, \text{ and}$$

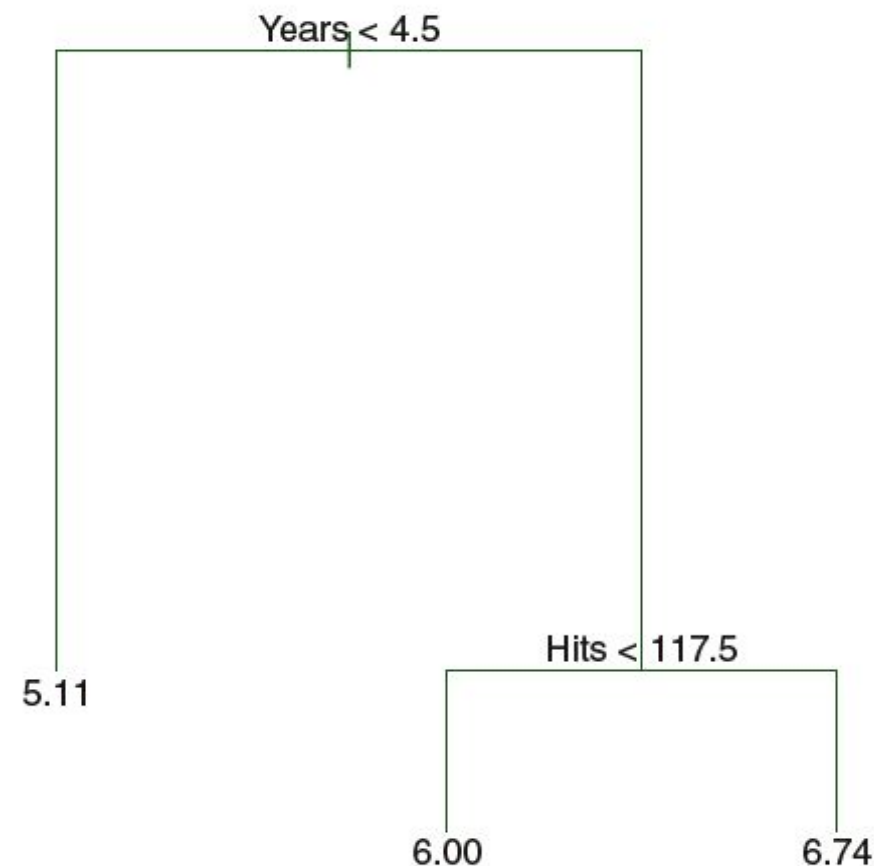
$$\$1,000 \times e^{(6.740)} = \$845,346 \text{ respectively.}$$

In keeping with the tree analogy, the regions R1, R2, and R3 are known as **terminal nodes** or **leaves** of the tree.



Regression Tree

- Terminal Node or leaves
- Internal Nodes
- Branches
- over-simplification of the true relationship
- it is easier to interpret, and has a nice graphical representation





Prediction via Stratification of the Feature Space

Steps

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



Prediction via Stratification of the Feature Space

For instance, suppose that in Step 1 we obtain two regions, $R1$ and $R2$, and that the response mean of the training observations in the first region is 10, while the response mean of the training observations in the second region is 20.

Then for a given observation $X = x$, if $x \in R1$ we will predict a value of 10, and if $x \in R2$ we will predict a value of 20.



Prediction via Stratification of the Feature Space

- Divide the predictor space into high-dimensional rectangles, or boxes
- The goal is to find boxes R_1, \dots, R_J that minimize the RSS, given by

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \quad (8.1)$$

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



Prediction via Stratification of the Feature Space

- **Top-down, greedy approach aka recursive binary splitting**
 - The approach is top-down because it begins at the top of the tree (at which point all observations belong to a single region)
 - 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 that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.



Recursive Binary Splitting

- Select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- The notation $\{X|X_j < s\}$ means the region of predictor space in which X_j takes on a value less than s .
- Consider all predictors X_1, \dots, X_p , and all possible values of the cutpoint s for each of the predictors, and then choose the predictor and cutpoint such that the resulting tree has the lowest RSS.



Recursive Binary Splitting

In greater detail, for any j and s , we define the pair of half-planes

$$R_1(j, s) = \{X | X_j < s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j \geq s\}, \quad (8.2)$$

and we seek the value of j and s that minimize the equation

$$\sum_{i: x_i \in R_1(j, s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j, s)} (y_i - \hat{y}_{R_2})^2, \quad (8.3)$$

where \hat{y}_{R_1} is the mean response for the training observations in $R_1(j, s)$, and \hat{y}_{R_2} is the mean response for the training observations in $R_2(j, s)$.



Recursive Binary Splitting

- Next, we repeat the process, looking for the best predictor and best cutpoint 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.

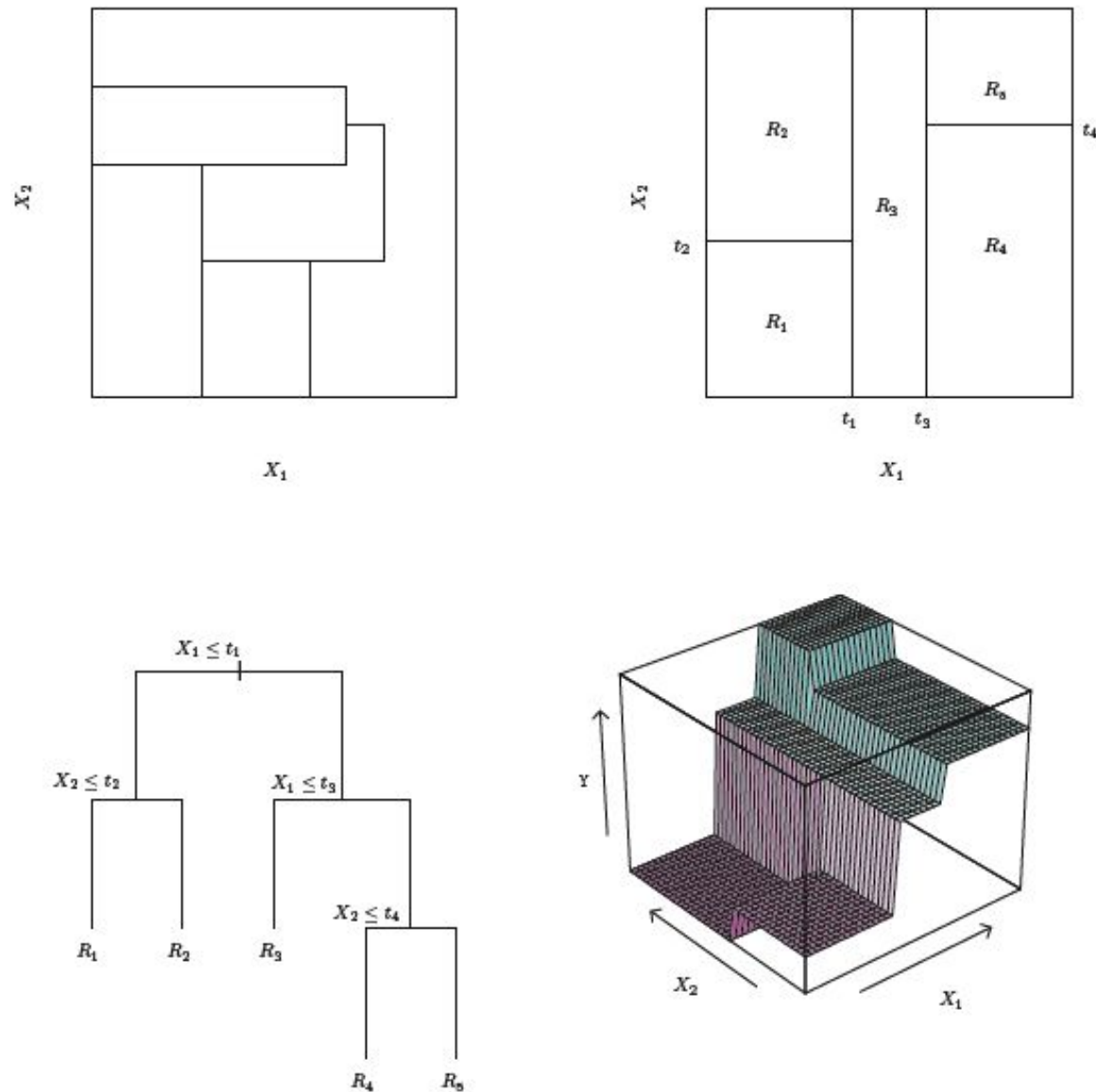


FIGURE 8.3. Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting. Top Right: The output of recursive binary splitting on a two-dimensional example. Bottom Left: A tree corresponding to the partition in the top right panel. Bottom Right: A perspective plot of the prediction surface corresponding to that tree.



Tree Pruning

The process described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.

A smaller tree with fewer splits (that is, fewer regions R_1, \dots, R_J) might lead to lower variance and better interpretation at the cost of a little bias

Build the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold



- Therefore, a better strategy is to grow a very large tree T_0 , and then prune it back in order to obtain a subtree.
- Our goal is to select a subtree that leads to the lowest test error rate
- Estimate its test error using cross-validation or the validation set approach.
- *Cost complexity pruning*—also known as *weakest link pruning*



Cost complexity pruning

Rather than considering every possible subtree, we consider a sequence of trees indexed by a nonnegative tuning parameter α .

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \quad (8.4)$$

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T , R_m is the rectangle (i.e. the subset of predictor space) corresponding to the m th terminal node, and \hat{y}_{R_m} is the predicted response associated with R_m —that is, the mean of the training observations in R_m .



Cost complexity pruning

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- When $\alpha = 0$, then the subtree T will simply equal T_0 , because then (8.4) just measures the training error.
- However, as α increases, there is a price to pay for having a tree with many terminal nodes, and so the quantity (8.4) will tend to be minimized for a smaller subtree.



Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
 2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α .
 3. Use K-fold cross-validation to choose α . That is, divide the training observations into K folds. For each $k = 1, \dots, K$:
 - (a) Repeat Steps 1 and 2 on all but the k th fold of the training data.
 - (b) Evaluate the mean squared prediction error on the data in the left-out k th fold, as a function of α .Average the results for each value of α , and pick α to minimize the average error.
 4. Return the subtree from Step 2 that corresponds to the chosen value of α .
-



Cross-validation

Cross-validation is a technique that splits the data into multiple subsets and uses some of them for training and some of them for testing.

By doing this, you can evaluate how your model performs on different samples of the data and estimate its average accuracy and variability.

Cross-validation also helps you to tune the hyperparameters of your model, such as the maximum depth, minimum samples per leaf, or splitting criterion.

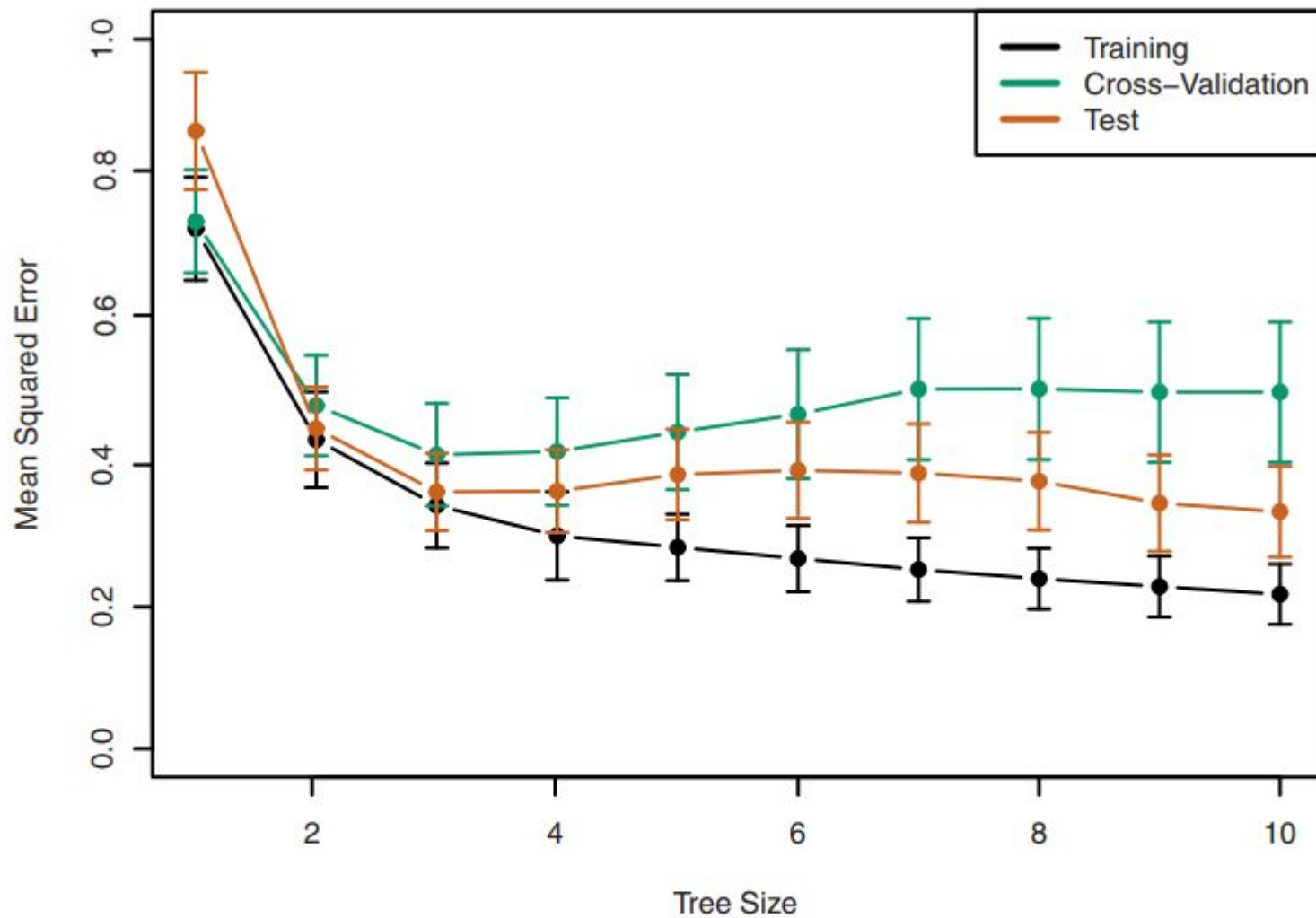


FIGURE 8.5. Regression tree analysis for the **Hitters** data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.



Classification Trees

- A classification tree is very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs
- *Class proportions* among the training observations that fall into that region



Classification Error Rate

Since we plan to assign an observation in a given region to the *most commonly occurring class* of training observations in that region, the *classification error rate* is simply the fraction of the training observations in that region that do not belong to the most common class

$$E = 1 - \max_k(\hat{p}_{mk}). \quad (8.5)$$

Here \hat{p}_{mk} represents the proportion of training observations in the m th region that are from the k th class. However, it turns out that classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable.



The *Gini index* is defined by

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}), \quad (8.6)$$

a measure of total variance across the K classes. It is not hard to see that the Gini index takes on a small value if all of the \hat{p}_{mk} 's are close to zero or one. For this reason the Gini index is referred to as a measure of node *purity*—a small value indicates that a node contains predominantly observations from a single class.



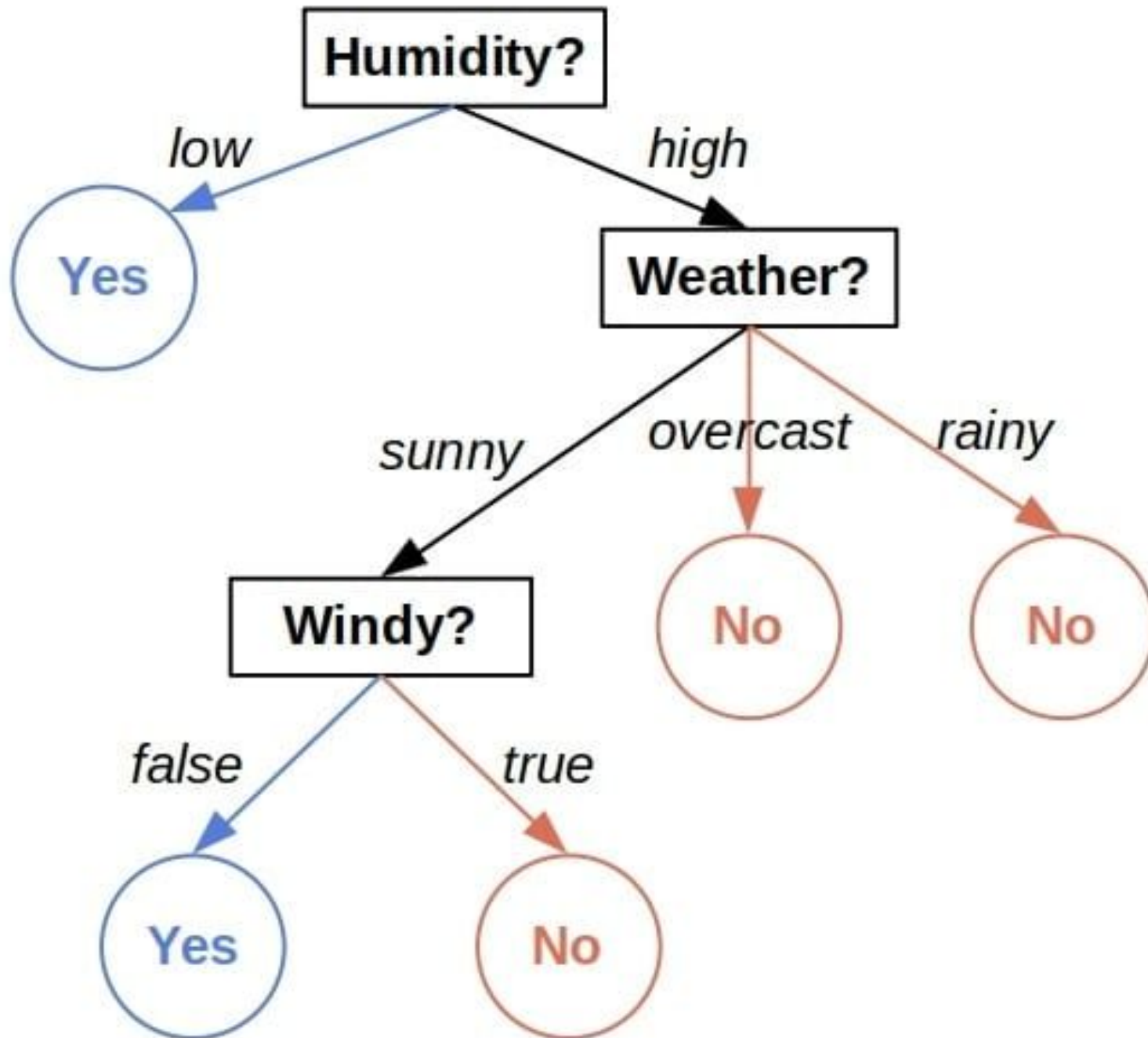
An alternative to the Gini index is *entropy*, given by

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}. \quad (8.7)$$

Since $0 \leq \hat{p}_{mk} \leq 1$, it follows that $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$. One can show that the entropy will take on a value near zero if the \hat{p}_{mk} 's are all near zero or near one. Therefore, like the Gini index, the entropy will take on a small value if the m th node is pure. In fact, it turns out that the Gini index and the entropy are quite similar numerically.

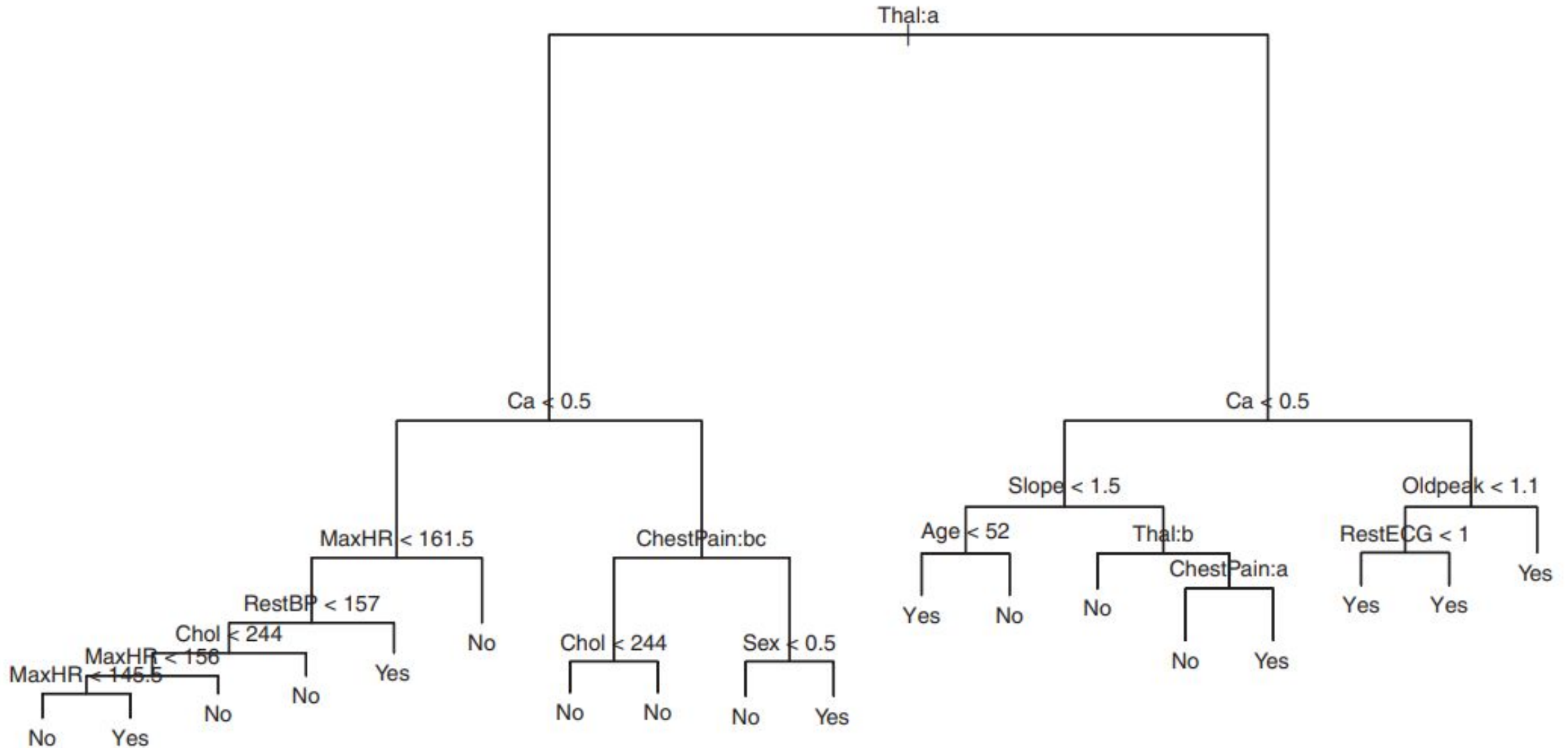


When building a classification tree, either the Gini index or the entropy are typically used to evaluate the quality of a particular split, since these two approaches are more sensitive to node purity than is the classification error rate. Any of these three approaches might be used when *pruning* the tree, but the classification error rate is preferable if prediction accuracy of the final pruned tree is the goal.



Decision tree for classifying days as suitable for playing outside

Heart data. The unpruned tree.



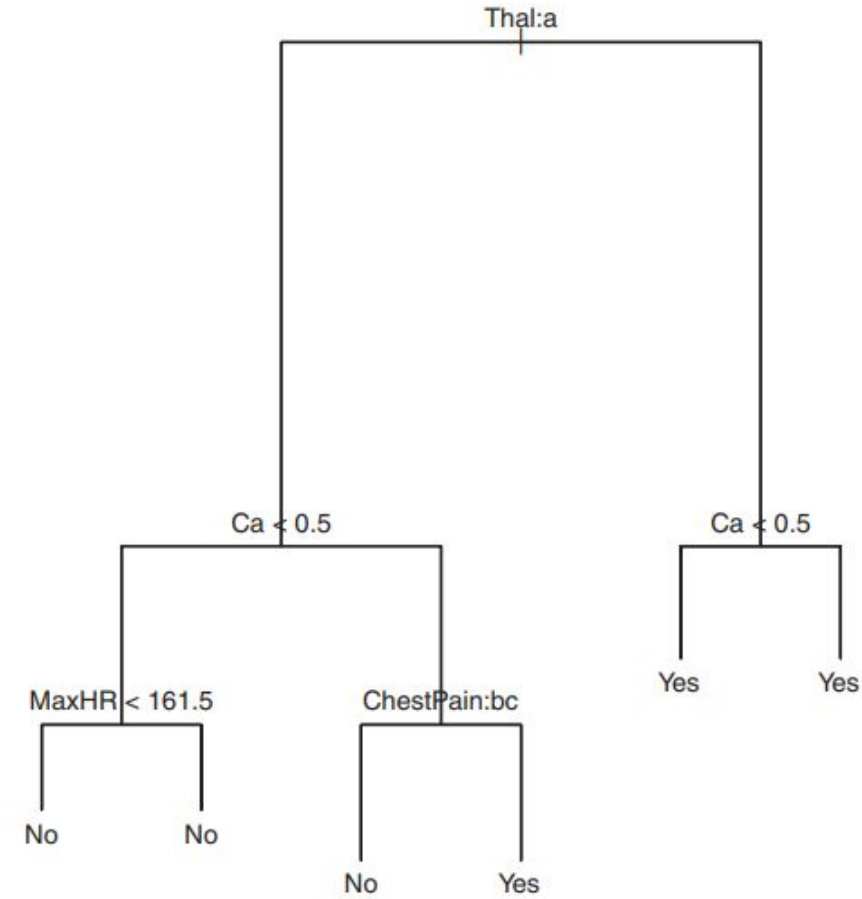
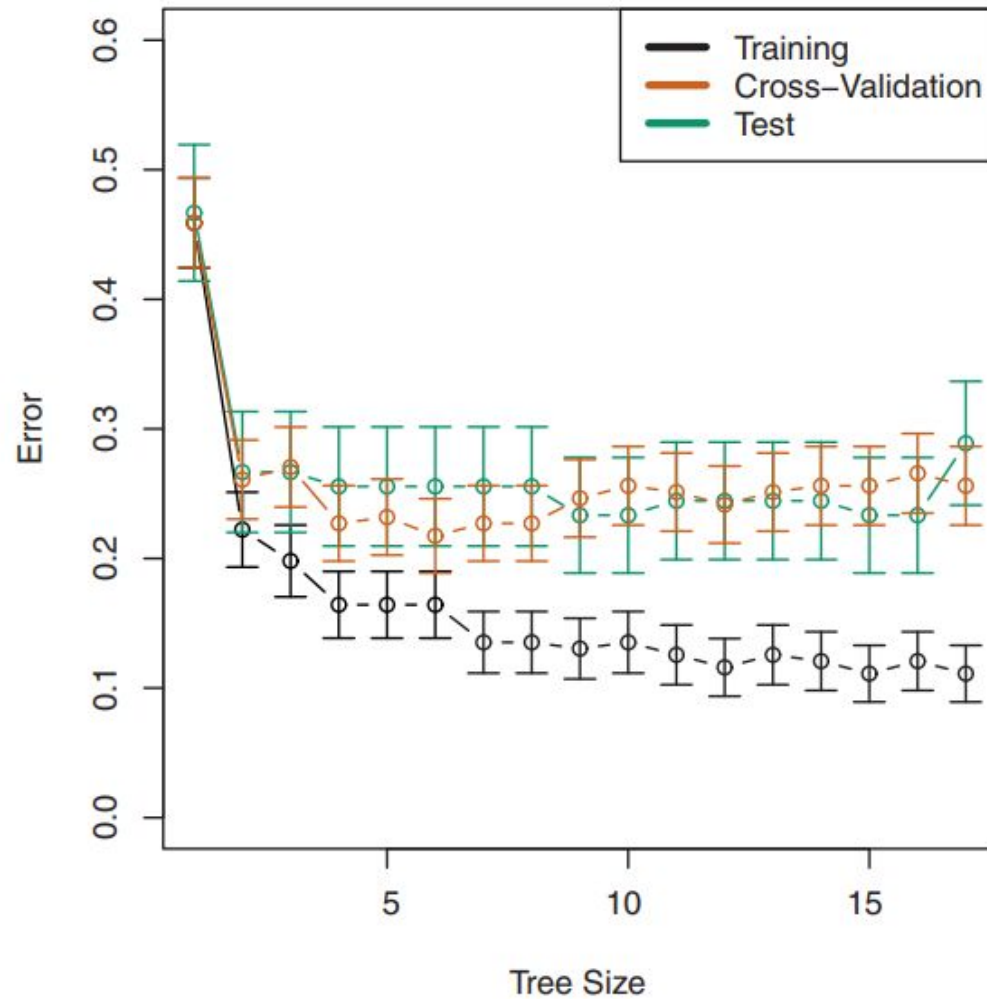
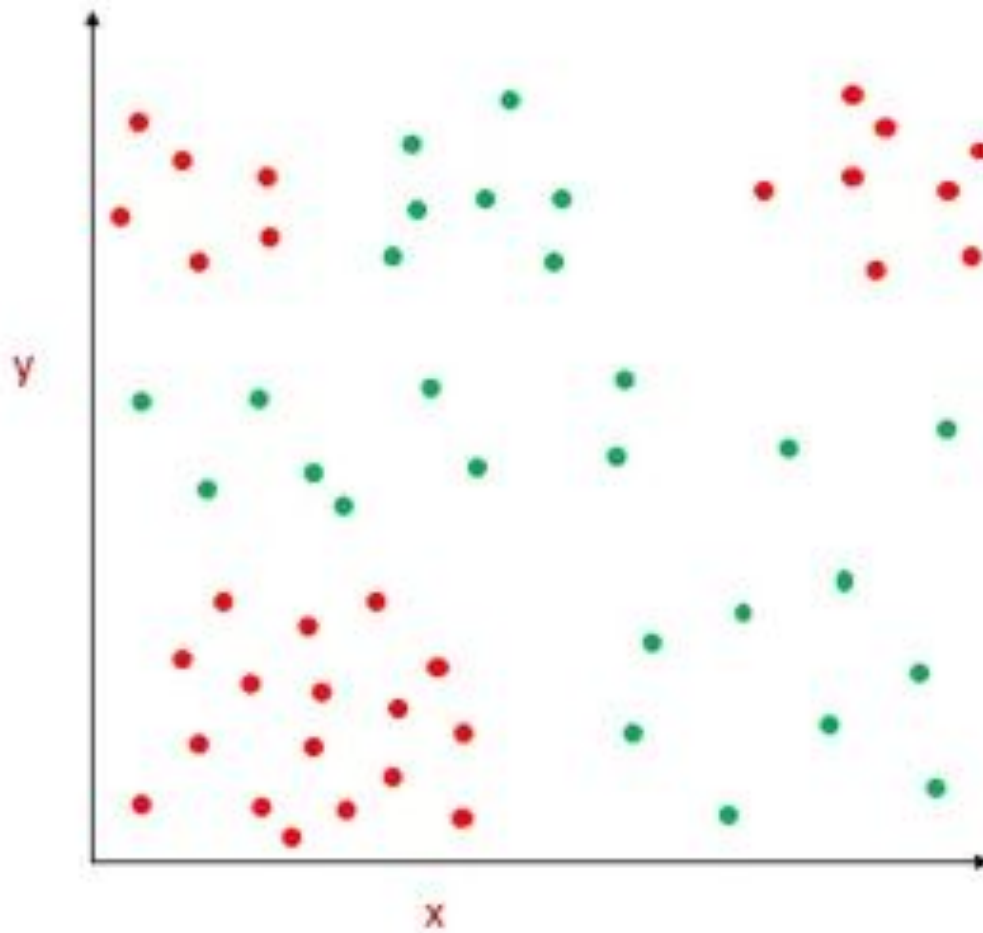
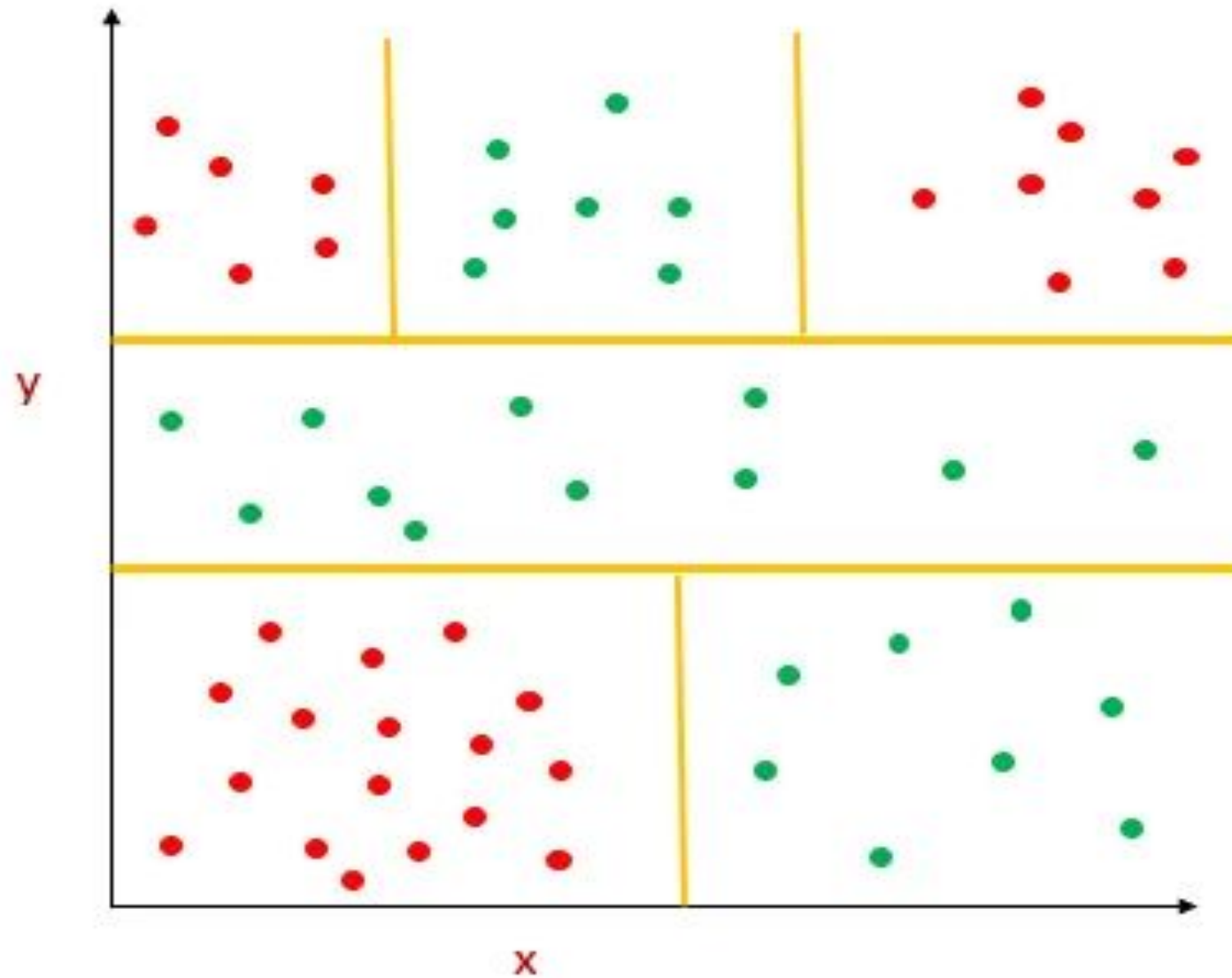


FIGURE 8.6. *Heart* data. Top: The unpruned tree. Bottom Left: Cross-validation error, training, and test error, for different sizes of the pruned tree. Bottom Right: The pruned tree corresponding to the minimal cross-validation error.







Trees Versus Linear Models

Regression and classification trees have a very different flavor from the more classical approaches for regression and classification presented in Chapters 3 and 4. In particular, linear regression assumes a model of the form

$$f(X) = \beta_0 + \sum_{j=1}^p X_j \beta_j, \quad (8.8)$$

whereas regression trees assume a model of the form

$$f(X) = \sum_{m=1}^M c_m \cdot 1_{(X \in R_m)} \quad (8.9)$$

where R_1, \dots, R_M represent a partition of feature space, as in Figure 8.3.

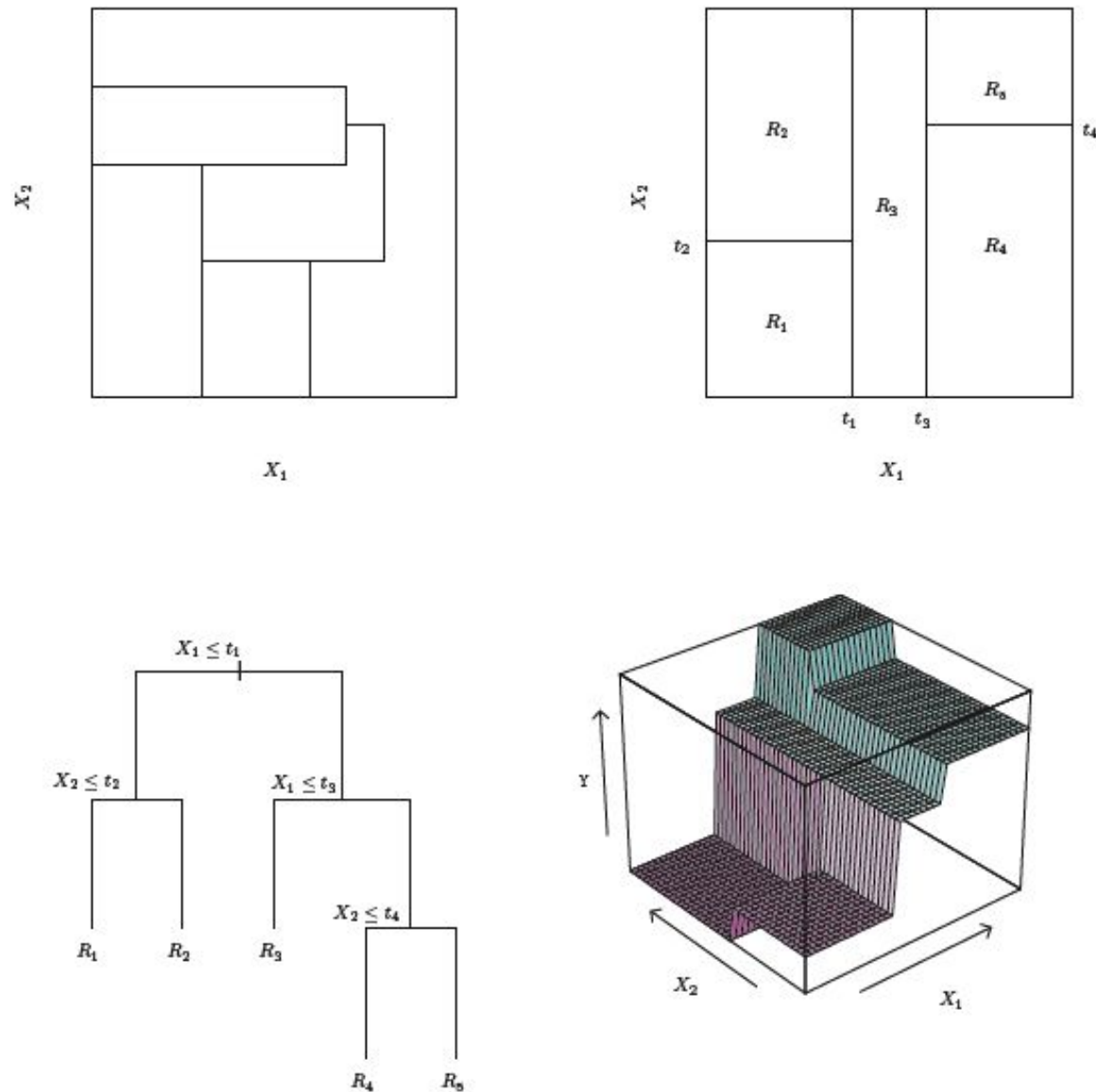
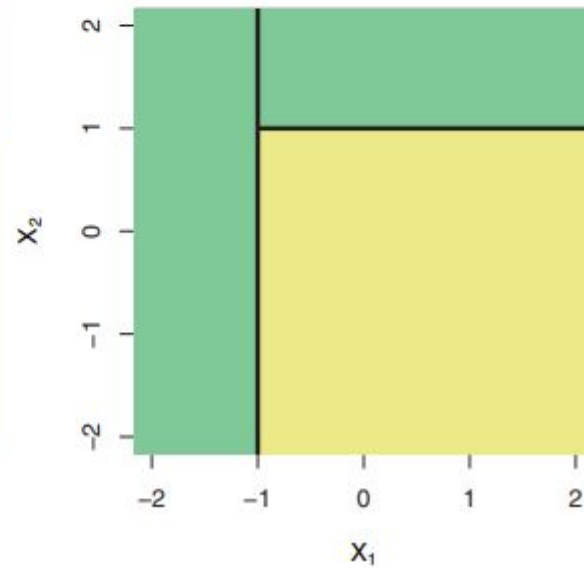
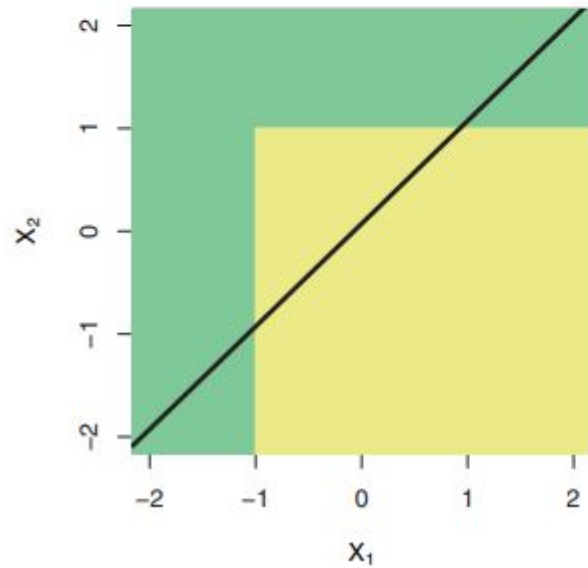
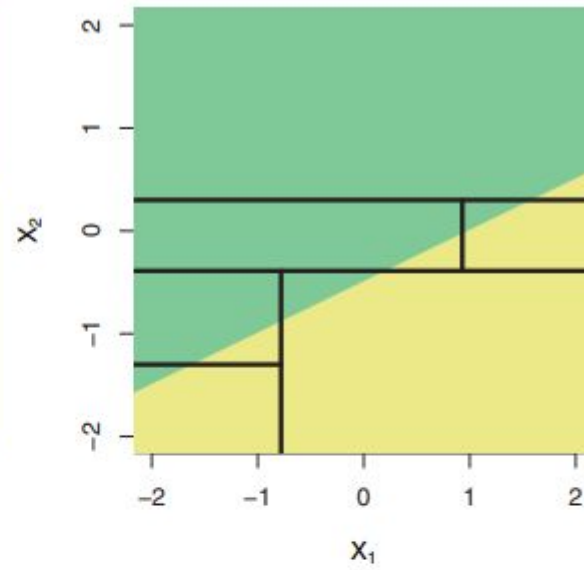
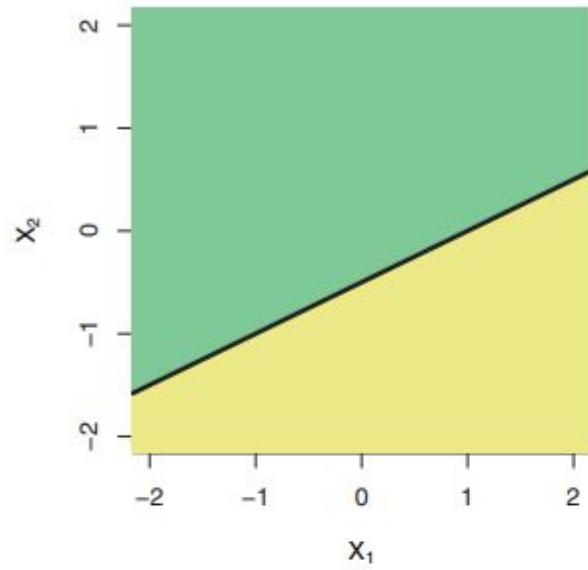


FIGURE 8.3. Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting. Top Right: The output of recursive binary splitting on a two-dimensional example. Bottom Left: A tree corresponding to the partition in the top right panel. Bottom Right: A perspective plot of the prediction surface corresponding to that tree.



Top Row: A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right). Bottom Row: Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).



Advantages of Trees

- Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.



Disadvantages of Trees

Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.

Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree



Bagging

Why bagging is used?

Decision trees are high-variance models. It means a slight change in the training data, will lead to an entirely different model. Decision trees usually overfit. To overcome this situation, ensemble technique — bagging can be used.



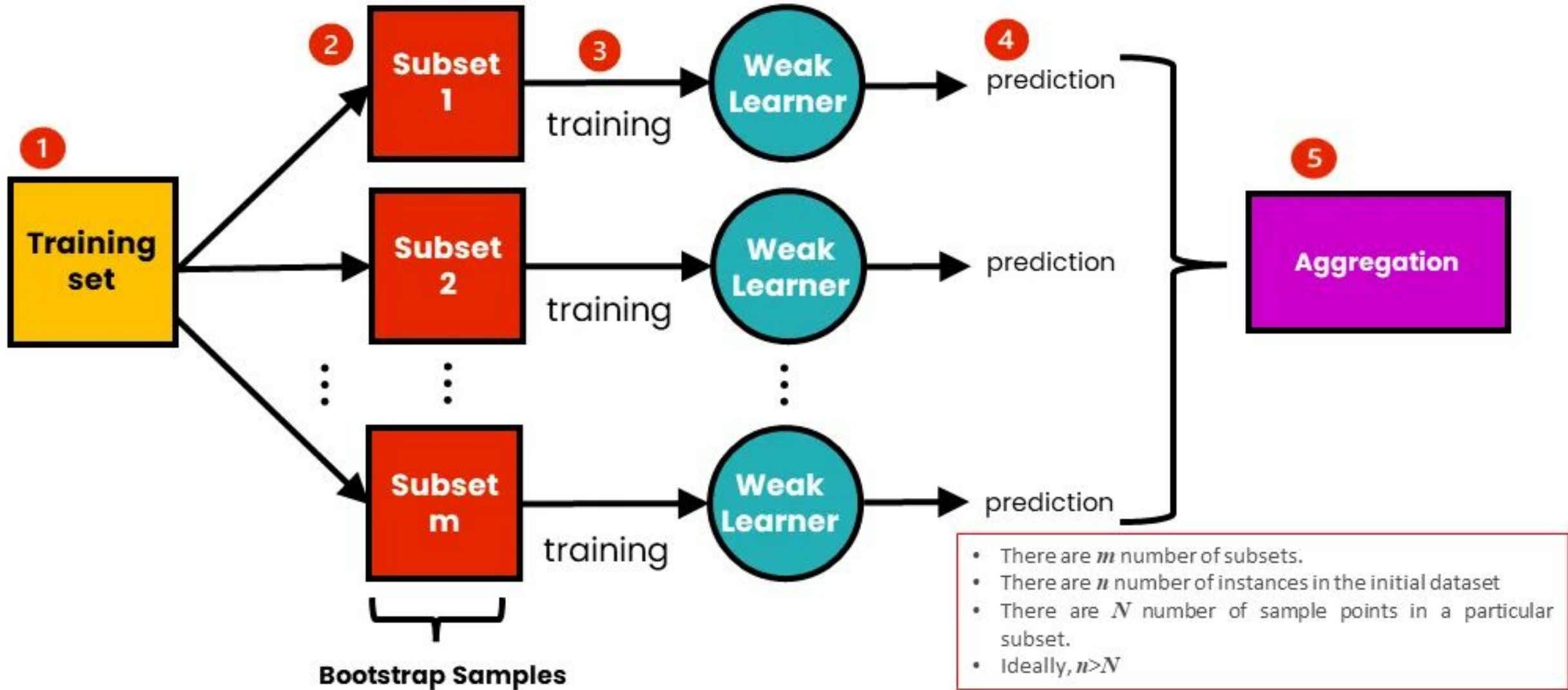
Bagging

What is bagging?

Bagging means Bootstrap Aggregation.

Bagging means building different models using sample subset and then aggregating the predictions of the different models to reduce variance.

The Process of Bagging (Bootstrap Aggregation)





How bagging reduces variance?

- Suppose we have a set of 'n' independent observations say Z_1, Z_2, \dots, Z_n . The variance of individual observation is σ^2 .
- The mean of all data points will be $(Z_1 + Z_2 + \dots + Z_n)/n$
- Similarly, the variance of that mean will be σ^2/n .
- So, if we increase the number of data points, the variance of the mean is reduced. This is the concept behind bagging-decision trees.
- Likewise, if we train multiple decision trees on a given data set and then aggregate the predictions. The variance will be reduced.



we could calculate

$$\hat{f}^1(x), \hat{f}^2(x), \dots, \hat{f}^B(x)$$

using B separate training sets, and average them in order to obtain a single low-variance statistical learning model,

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x).$$



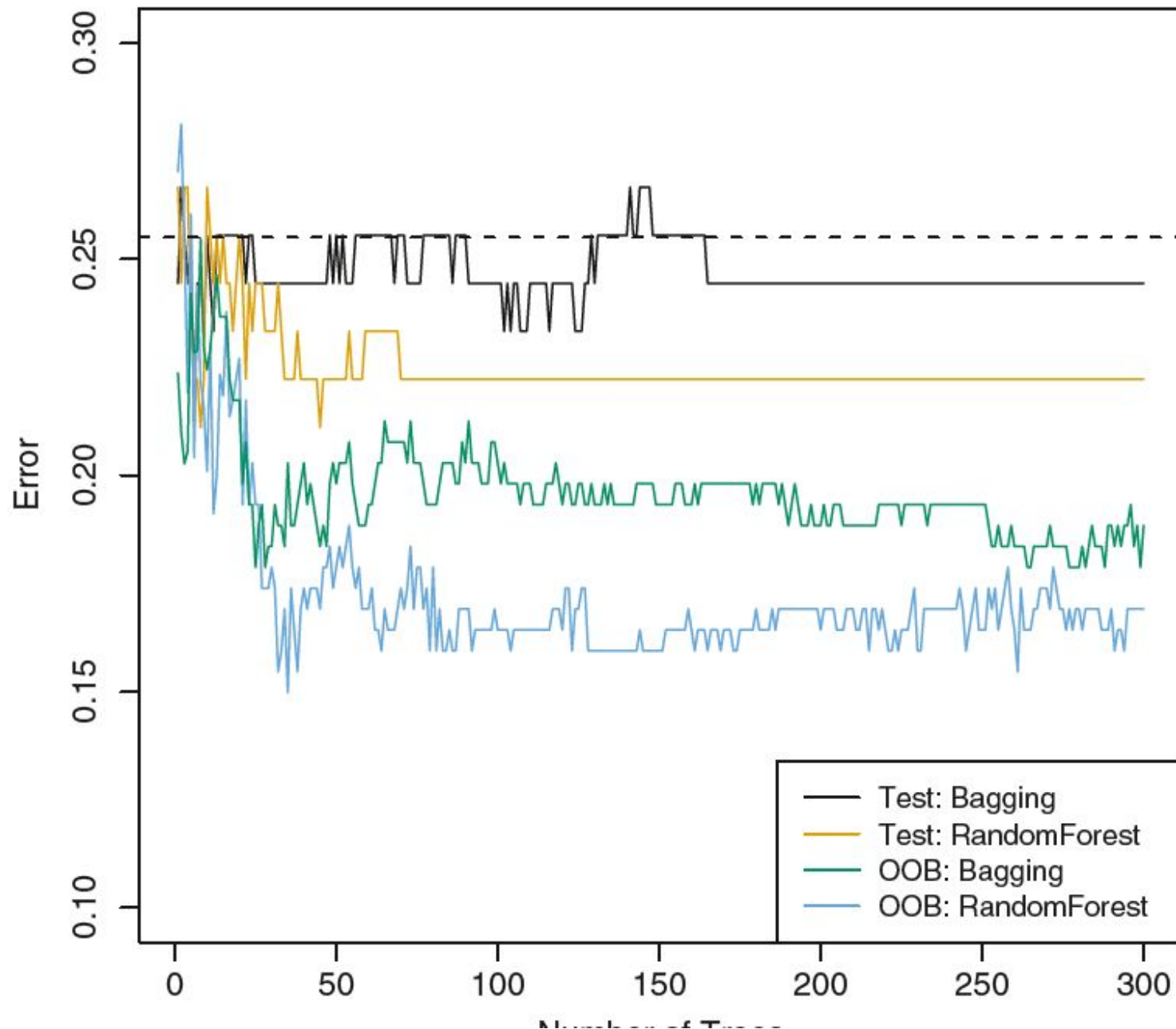
Of course, this is not practical because we generally do not have access to multiple training sets. Instead, we can bootstrap, by taking repeated samples from the (single) training data set. In this approach we generate B different bootstrapped training data sets. We then train our method on the b th bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all the predictions, to obtain

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

This is called bagging.



For a given test observation, we can record the class predicted by each of the B trees, and take a **majority vote**: the overall prediction is the most commonly occurring class among the B predictions.



Bagging and random forest results for the Heart data. The test error (black and orange) is shown as a function of B , the number of bootstrapped training sets used. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.



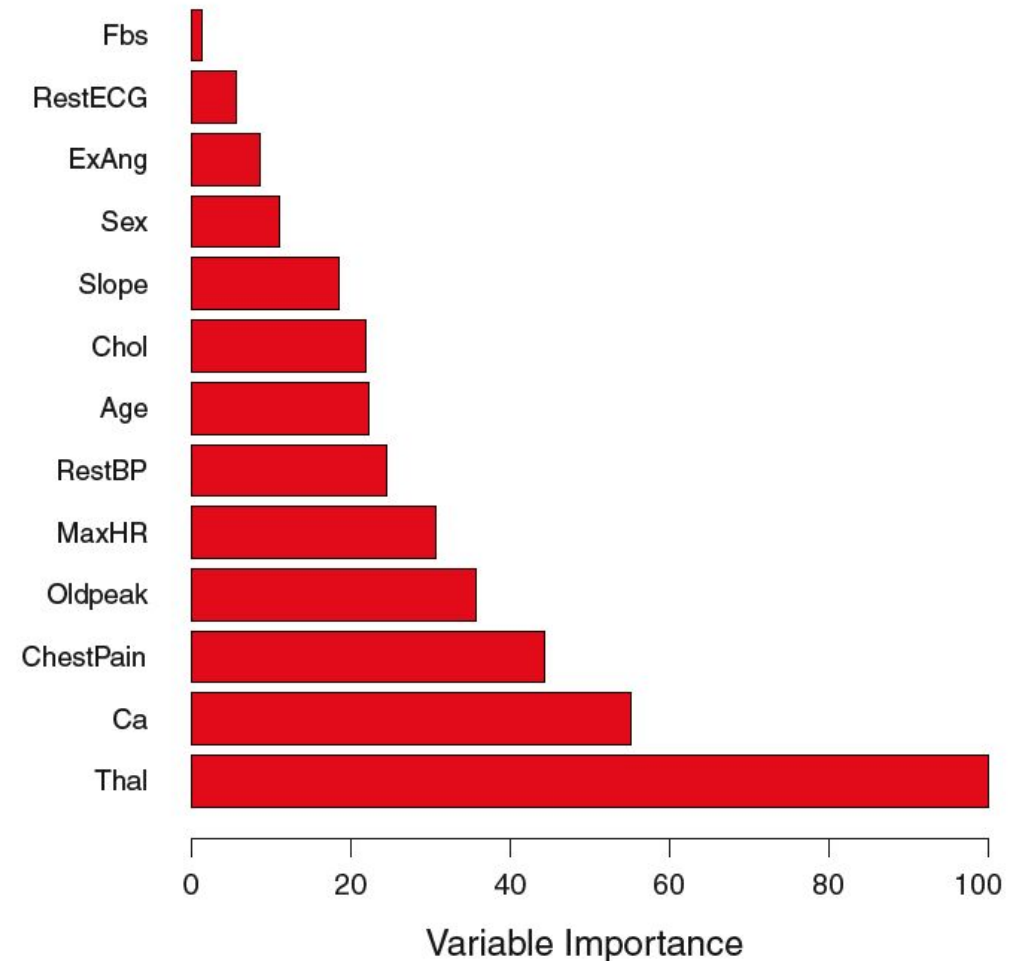
Out-of-Bag Error Estimation

- Out-of-bag (OOB) observations
- Predict the response for the i th observation using each of the trees in which that observation was OOB
- This will yield around $B/3$ predictions for the i th observation.
- Average these predicted responses (if regression is the goal) or can take a majority vote (if classification is the goal).
- This leads to a single OOB prediction for the i th observation.
- The overall OOB MSE (for a regression problem) or classification error (for a classification problem) can be computed.
- The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation.



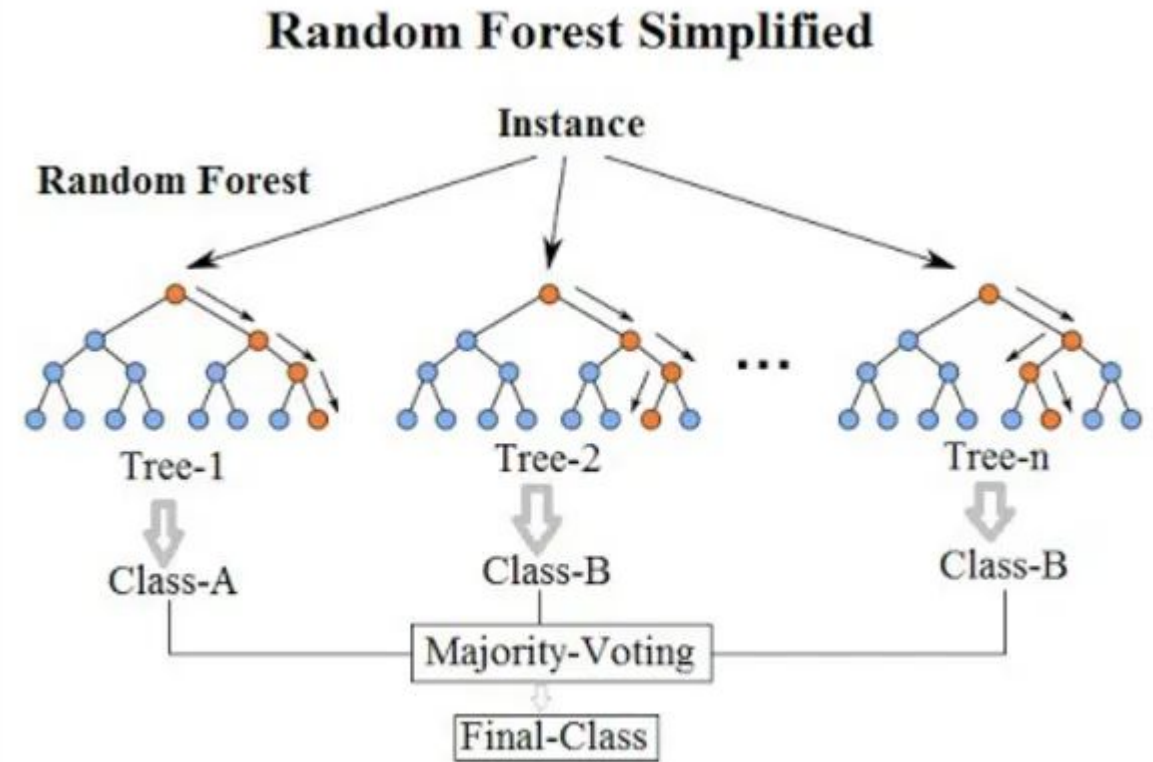
Variable Importance Measures

Although the collection of bagged trees is much more difficult to interpret than a single tree, one can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).



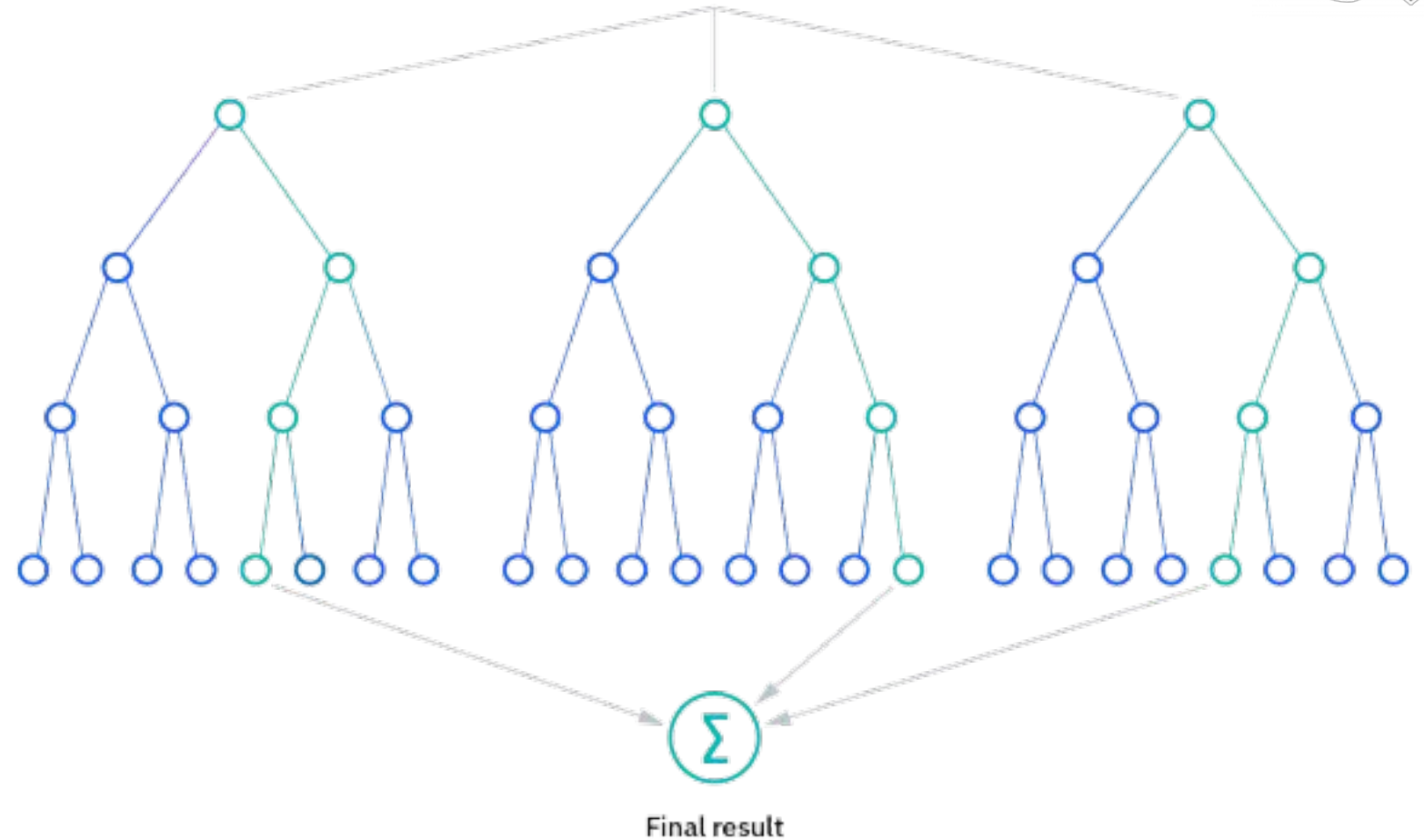
Random Forests

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset

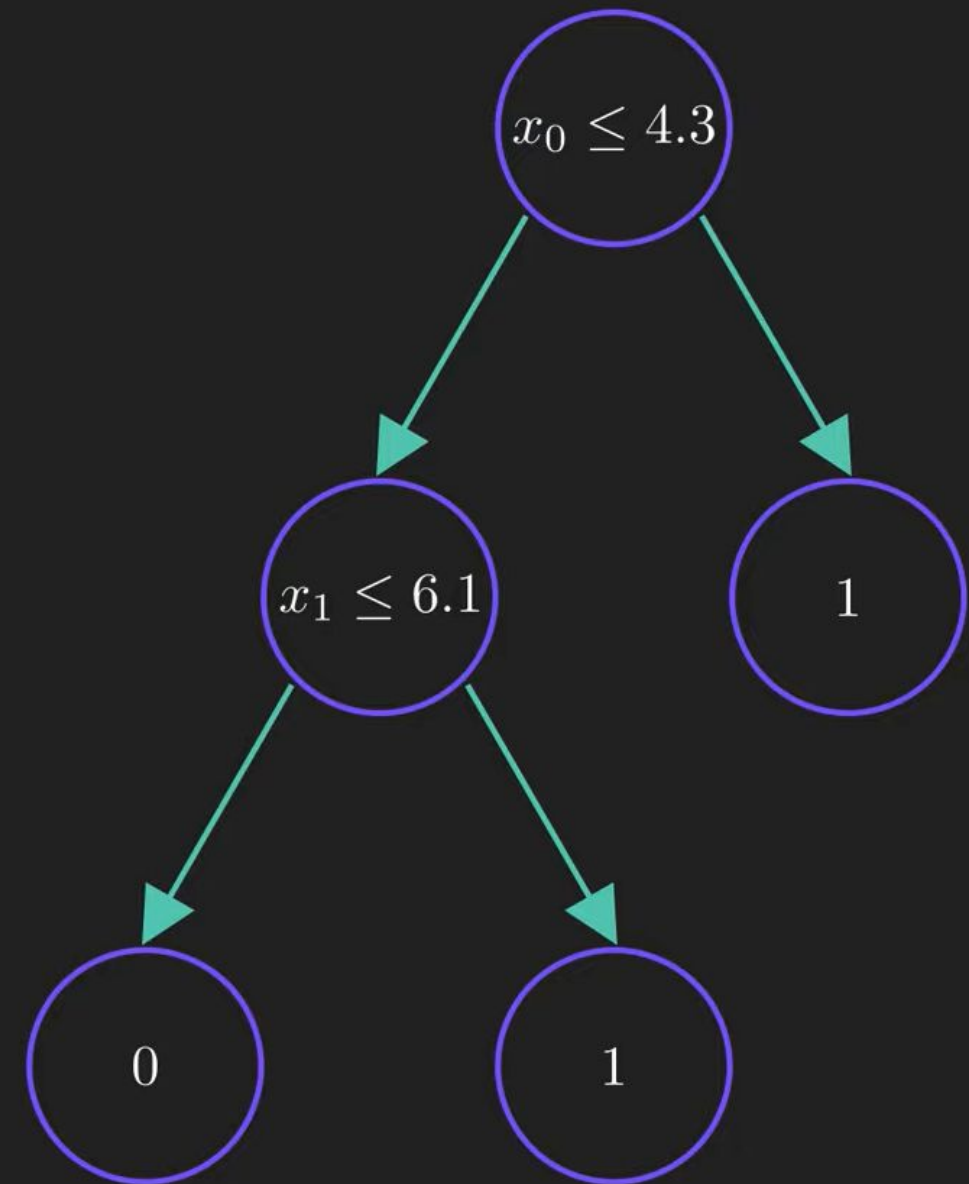


Random Forest

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

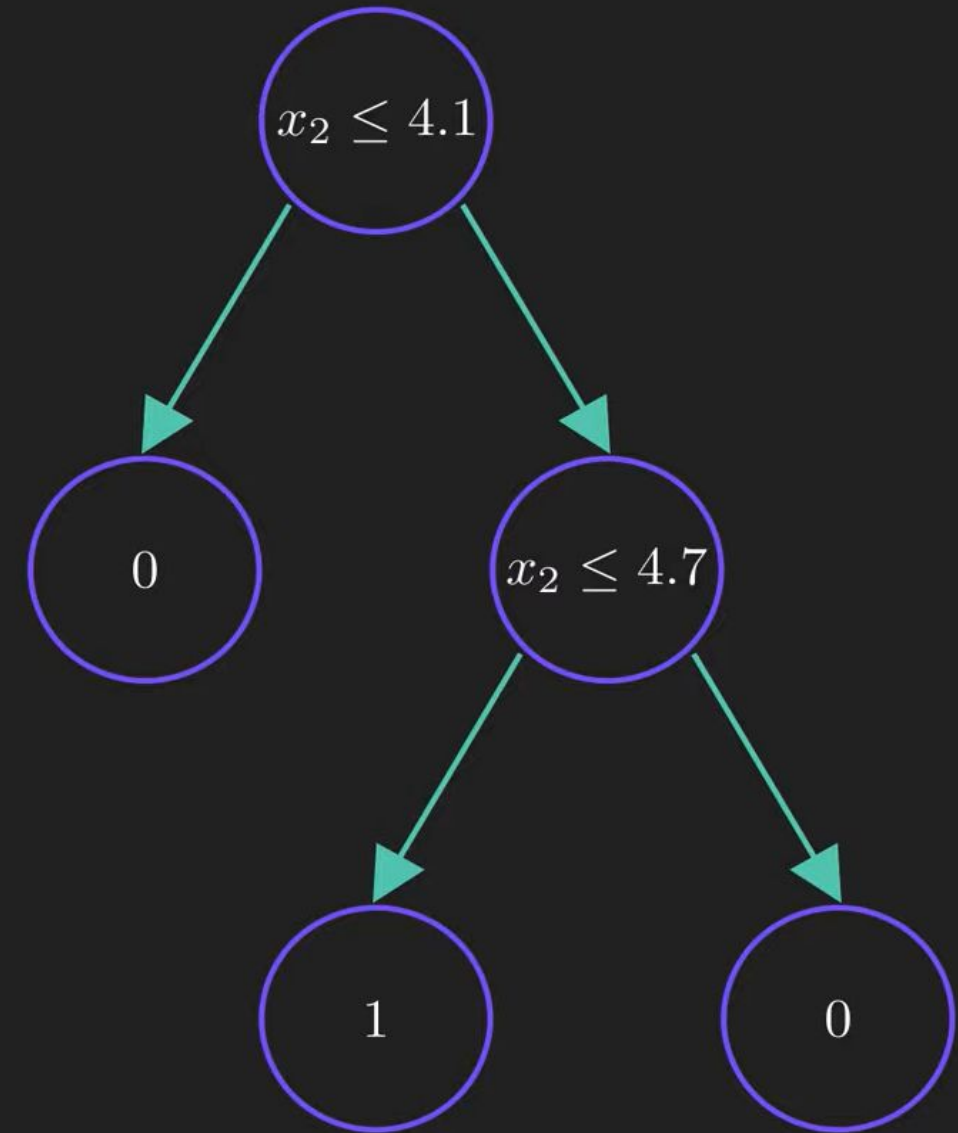


id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1



<https://www.youtube.com/watch?v=v6VJ2RO66Ag>

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	6.5	4.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1



<https://www.youtube.com/watch?v=v6VJ2RO66Ag>

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

id
2
1
3
1
4
4

id
4
1
3
0
0
2

id
3
3
2
5
1
2

<https://www.youtube.com/watch?v=v6VJ2RO66Ag>

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

x_0, x_1

id
2
1
3
1
4
4

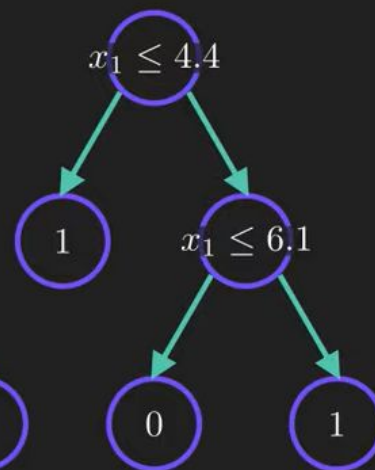
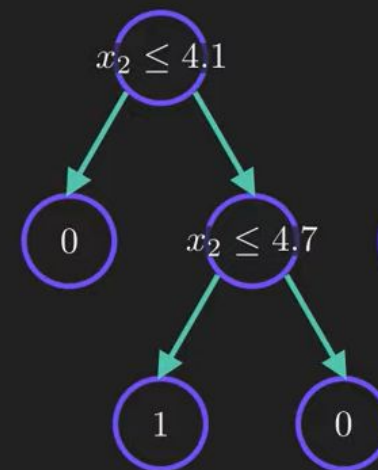
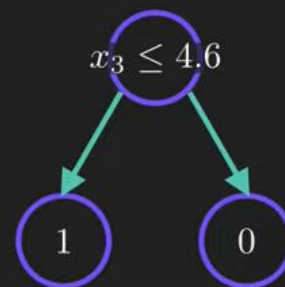
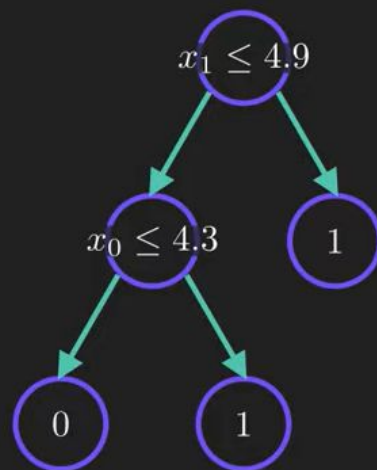
x_2, x_3

id
4
1
3
0
0
2

x_2, x_4

id
3
3
2
5
1
2

x_1, x_3



<https://www.youtube.com/watch?v=v6VJ2RO66Ag>

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

id
2
0
2
4
5
5

x_0, x_1

id
2
1
3
1
4
4

x_2, x_3

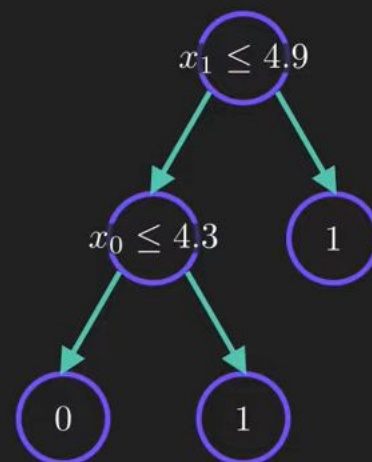
id
4
1
3
0
0
2

x_2, x_4

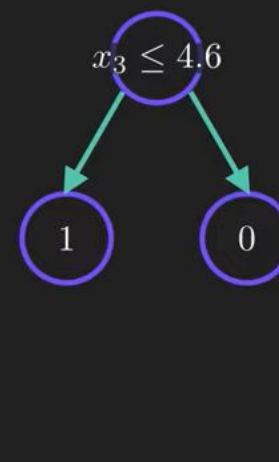
id
3
3
2
5
1
2

x_1, x_3

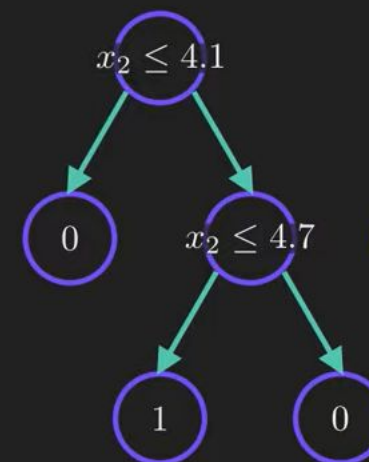
2.8	6.2	4.3	5.3	5.5
-----	-----	-----	-----	-----



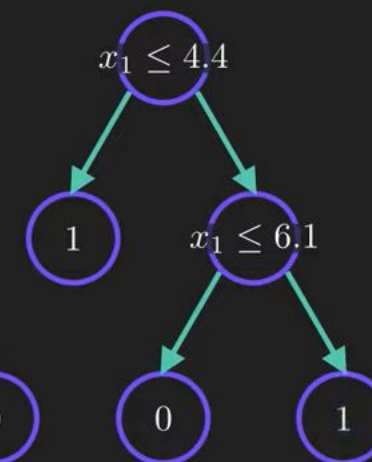
1



0



1



1

<https://www.youtube.com/watch?v=v6VJ2RO66Ag>