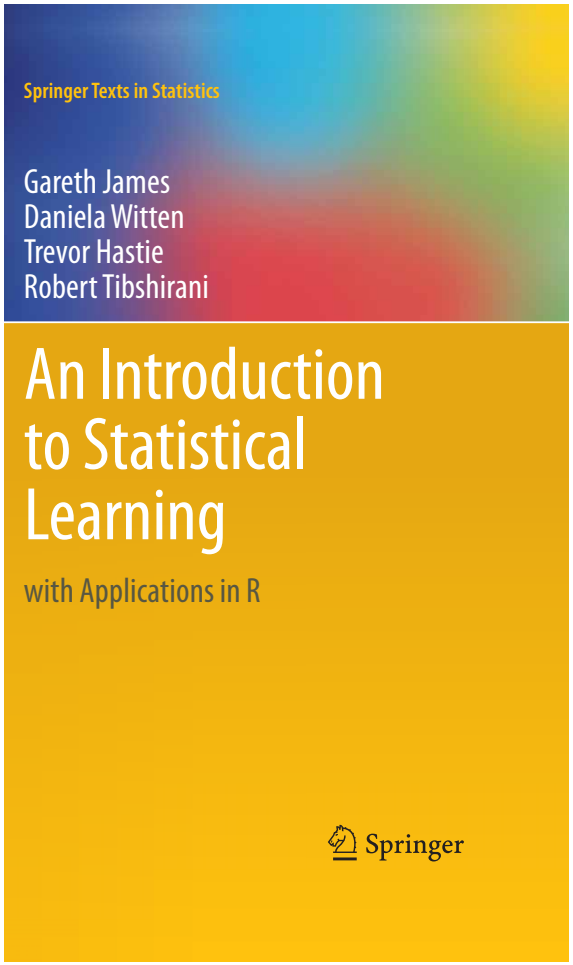
STA4026S – Honours Analytics
Section A – Theory and Application of Supervised Learning
Lecture 5 – Classification and Regression Trees

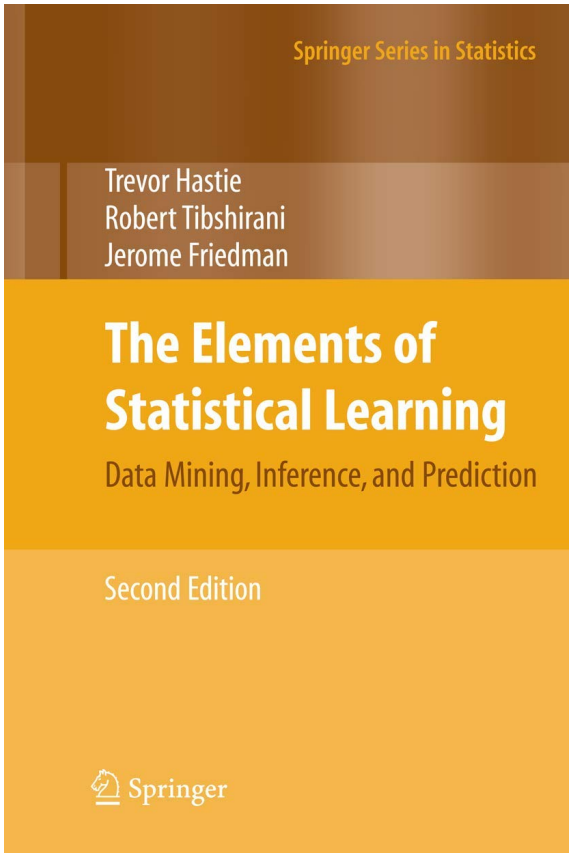
Stefan S. Britz
stefan.britz@uct.ac.za

Department of Statistical Sciences
University of Cape Town

Suggested Reading



Chapter 8



Section 9.2

Tree-based Methods Overview

We will cover the following methods that are based on decision trees:

- Regression Trees
- Classification Trees
- Bootstrap Aggregated Trees (Bagging)
- Random Forests
- Gradient Boosted Trees (Boosting)

Decision Trees Overview

- Decision trees can be applied to both regression and classification problems – hence referred to as CART.
- Decision trees are simple, non-parametric classifiers which utilize a tree structure to model the relationships among the features and the potential outcomes.
- The features can be continuous, discrete, or categorical.
- Decision trees are built using a heuristic called recursive partitioning (divide and conquer).
- We will first consider regression problems (with continuous response Y), and then move on to classification.

Regression Trees

Regression tree basics

- Let Y denote a continuous outcome variable and let $X_1, \ldots, X_p$ be a set of predictors/features, of which we have n observations
- The regression tree algorithm entails repeatedly splitting the p-dimensional feature space into distinct, non-overlapping regions, with the splits orthogonal to the axes
- Suppose that we partition the feature space into M regions, $R_1, R_2, \ldots, R_M$. Then for each region, we model the response as a **constant**, c_m:

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m I(\boldsymbol{x} \in R_m). \tag{1}$$

Regression tree basics

As with other regression models, our goal is still to minimise

$$RSS = \sum_{m=1}^{M} \sum_{i:x_i \in R_m} \left[y_i - \hat{f}(\boldsymbol{x}_i) \right]^2 \tag{2}$$

Therefore, the best $\hat{c}_m$ to minimise this criterion is simply the average of the y_i in each region R_j:

$$\hat{c}_m = \text{ave}(y_i | \boldsymbol{x}_i \in R_m) \tag{3}$$

Exhaustively searching over every possible combination of regions is computationally infeasible

Therefore, we use a top-down, greedy algorithm called **recursive binary splitting**

Recursive Binary Splitting

1. Select the predictor X_j and split point s such that the regions $R_1(j,s) = \{X|X_j < s\}$ and $R_2(j,s) = \{X|X_j \geq s\}$ lead to the greatest possible **reduction in RSS**. That is, we consider all predictors $X_1, \ldots, X_p$, and all possible values of the split point s for each of the predictors, and then choose the predictor and split point such that is minimised.

2. Identify the predictor and split point that best splits one of the previously identified regions, such that there are now three regions.

3. Continue splitting the regions until a **stopping criterion** is reached. For instance, we may continue until no region contains more than 5 observations, or when the proportional reduction in RSS is less than some threshold.

Prediction

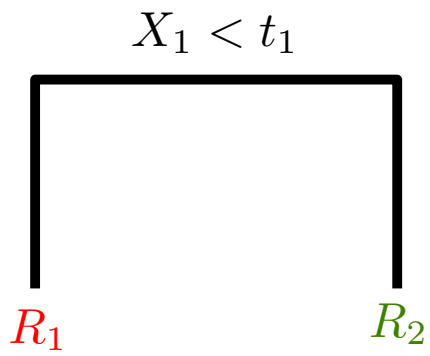
Once the regions $R_1, R_2, \ldots, R_M$ have been created, we predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.

This site/app provides an excellent visual illustration of this

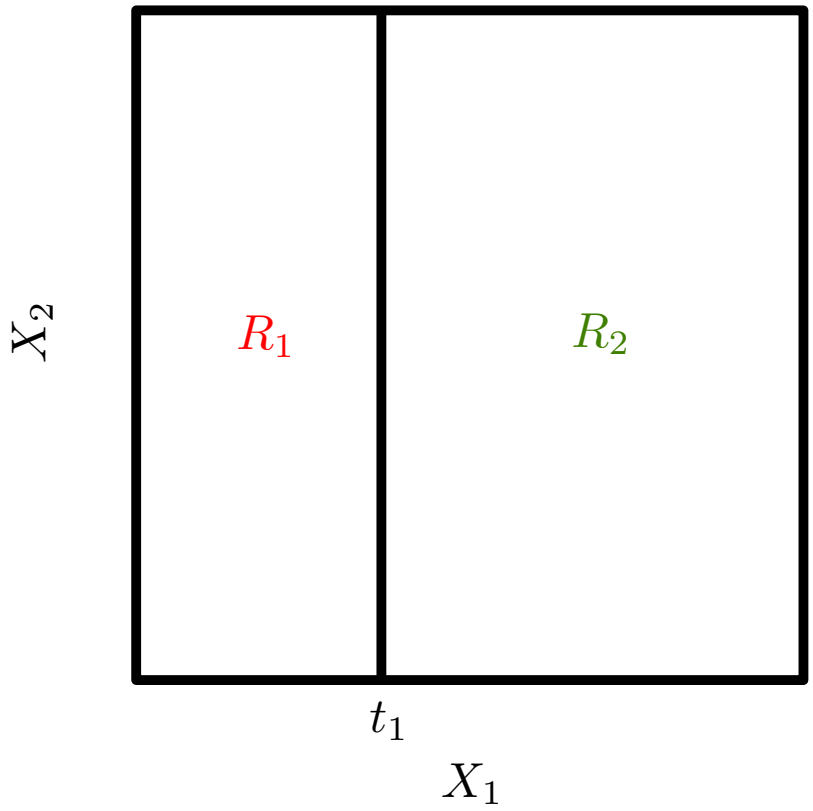
Also check out part 2 on the bias-variance trade-off

Recursive Binary Splitting

Regression Tree

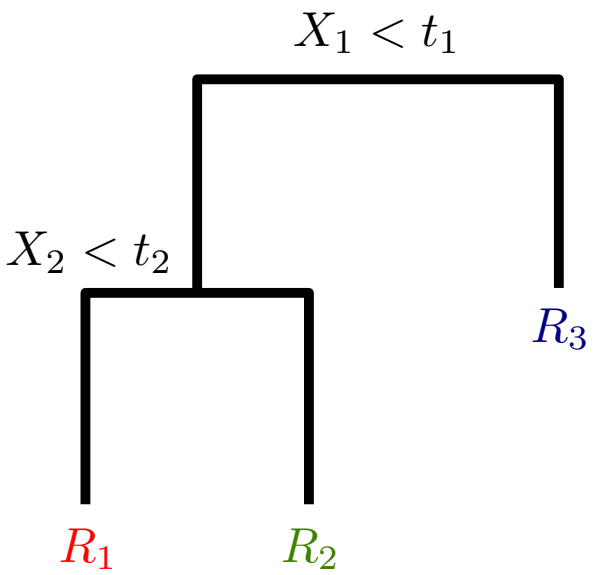


Partitioned Feature Space

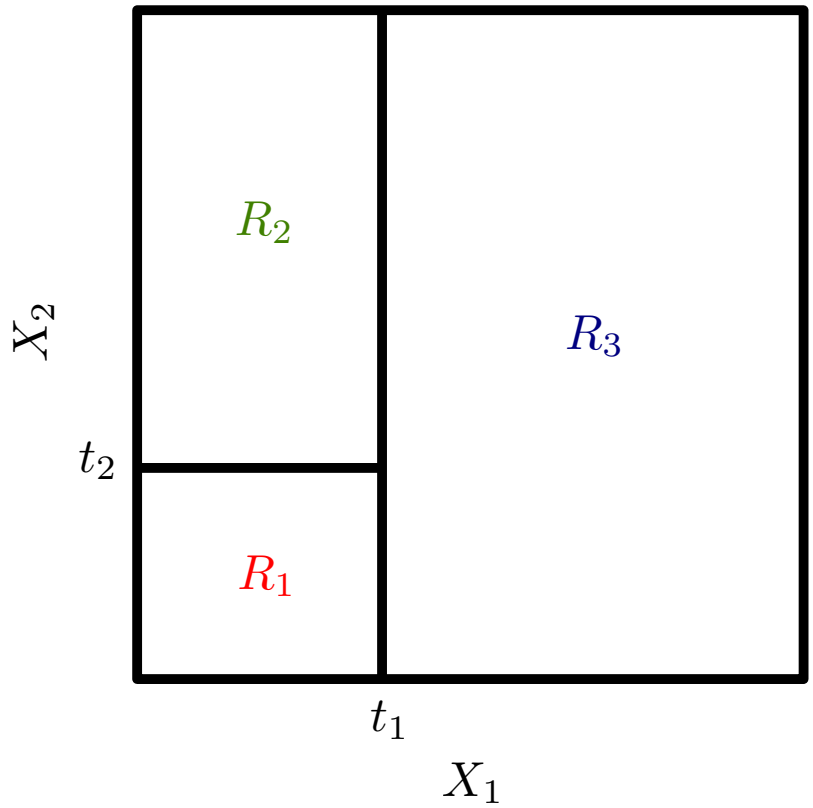


Recursive Binary Splitting

Regression Tree

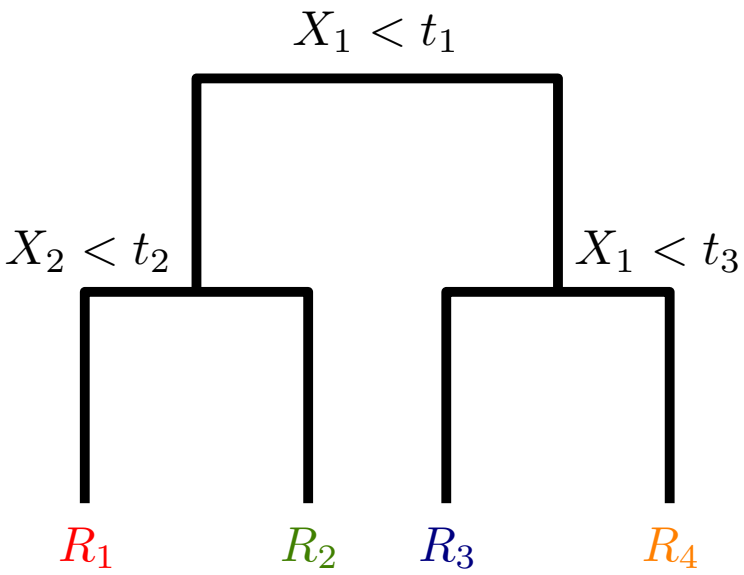


Partitioned Feature Space

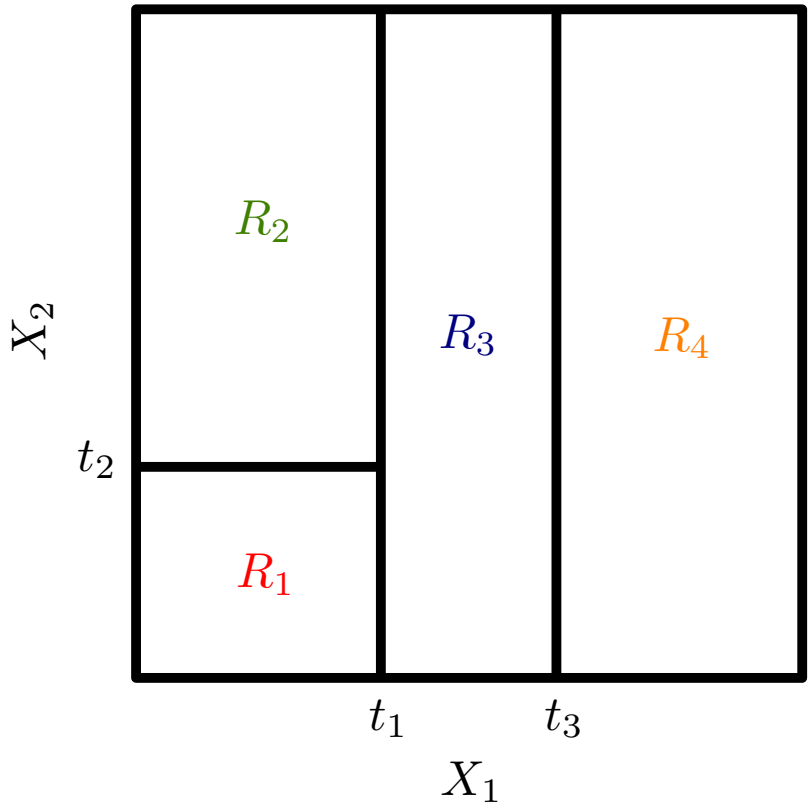


Recursive Binary Splitting

Regression Tree

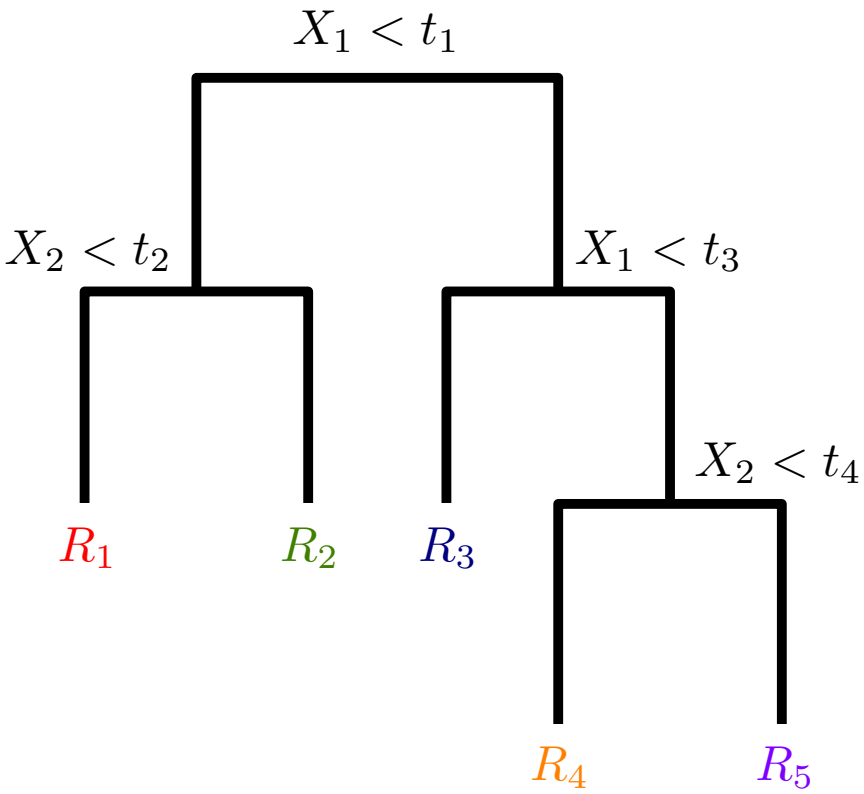


Partitioned Feature Space

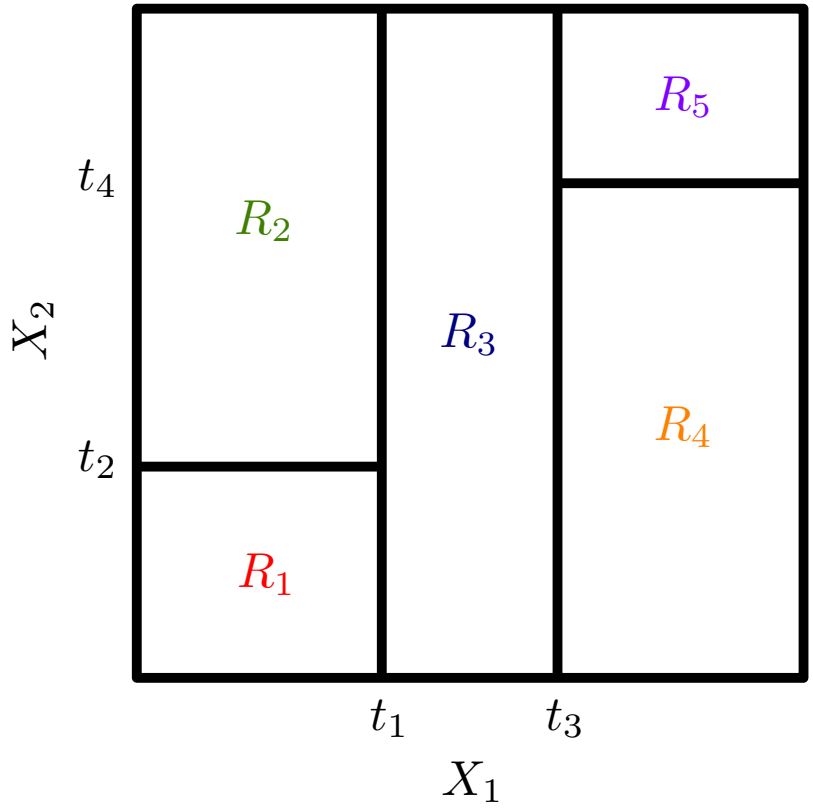


Recursive Binary Splitting

Regression Tree



Partitioned Feature Space



Link with GAMs

As a brief side note, the model from can also be written in the form

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m I(\boldsymbol{x} \in R_m) = \sum_{m=1}^{M} c_m \phi(\boldsymbol{x}; \boldsymbol{v}_m),$$

where $\boldsymbol{v}_m$ encodes the choice of variable to split on (X_j) and the split point value (s) on the path from the start (root) to the m^{th} region (leaf).

When viewed as such, a CART model is just an adaptive basis function model (alluded to in the previous chapter), where the basis functions define the regions, and the weights specify the response value in each region.

Application in R

There are are various R libraries available that contain functions for fitting regression/classification trees:

- tree
- rpart
- party
- maptree
- partykit
- caret
- and more

Each has its own particular strengths and weaknesses, but in general they have the same core elements.

We will be using the (basic) **tree** package for the sake of illustration, by means of another example.

Example 6 – California housing

- In order to motivate regression trees, we begin with a simple example.
- We will look at housing data collected in California in the 1990 census.
- Aggregated data are available for 20,640 neighbourhoods in California.
- Our goal is to predict the median house price in each neighbourhood block using some or all of the following predictor variables:

  Median income; Median house age; Total number of rooms; Total number of bedrooms; Neighbourhood population size; Total number of households; Latitude; Longitude

Example 6 – California housing

```
> calif <- read.csv('cadata.csv')
> head(calif)
```

	HousePrice	Income	HouseAge	Rooms	Bedrooms
1	452600	8.3252	41	880	129
2	358500	8.3014	21	7099	1106
3	352100	7.2574	52	1467	190
4	341300	5.6431	52	1274	235
5	342200	3.8462	52	1627	280
6	269700	4.0368	52	91	213

	Population	Households	Latitude	Longitude
1	322	126	37.88	-122.23
2	2401	1138	37.86	-122.22
3	496	177	37.85	-122.24
4	558	219	37.85	-122.25
5	565	259	37.85	-122.25
6	413	193	37.85	-122.25

Example 6 – California housing

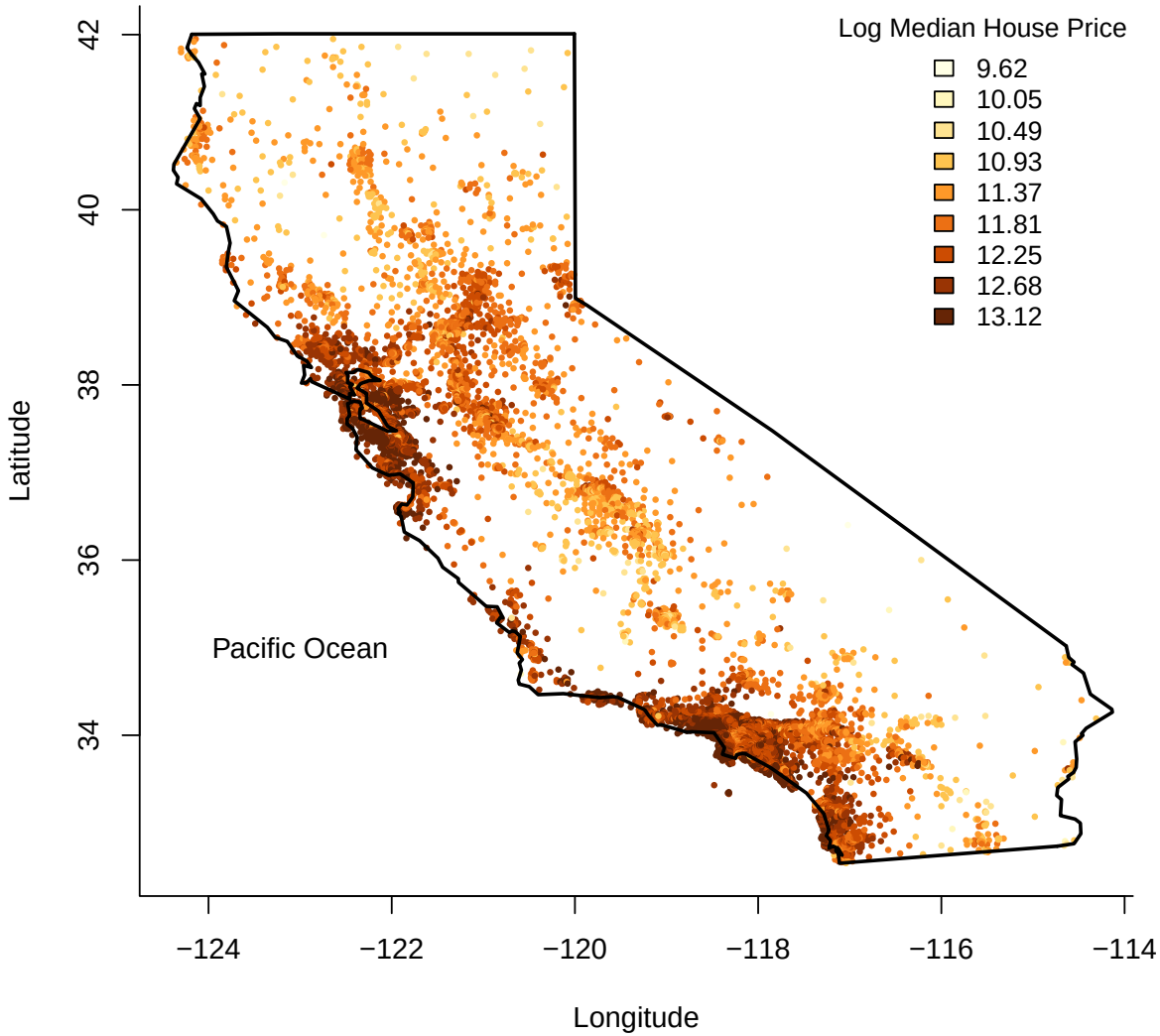
```
> calif <- read.csv('cadata.csv')
> head(calif)
```

	HousePrice	Income	HouseAge	Rooms	Bedrooms
1	452600	8.3252	41	880	129
2	358500	8.3014	21	7099	1106
3	352100	7.2574	52	1467	190
4	341300	5.6431	52	1274	235
5	342200	3.8462	52	1627	280
6	269700	4.0368	52	91	213

	Population	Households	Latitude	Longitude
1	322	126	37.88	-122.23
2	2401	1138	37.86	-122.22
3	496	177	37.85	-122.24
4	558	219	37.85	-122.25
5	565	259	37.85	-122.25
6	413	193	37.85	-122.25

We will begin by considering only **latitude and longitude** as potential predictors

Example 6 – California housing

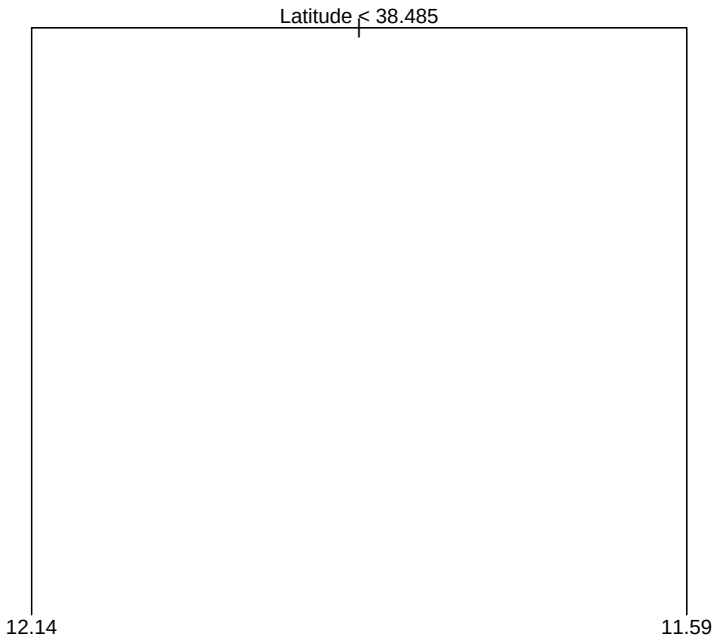


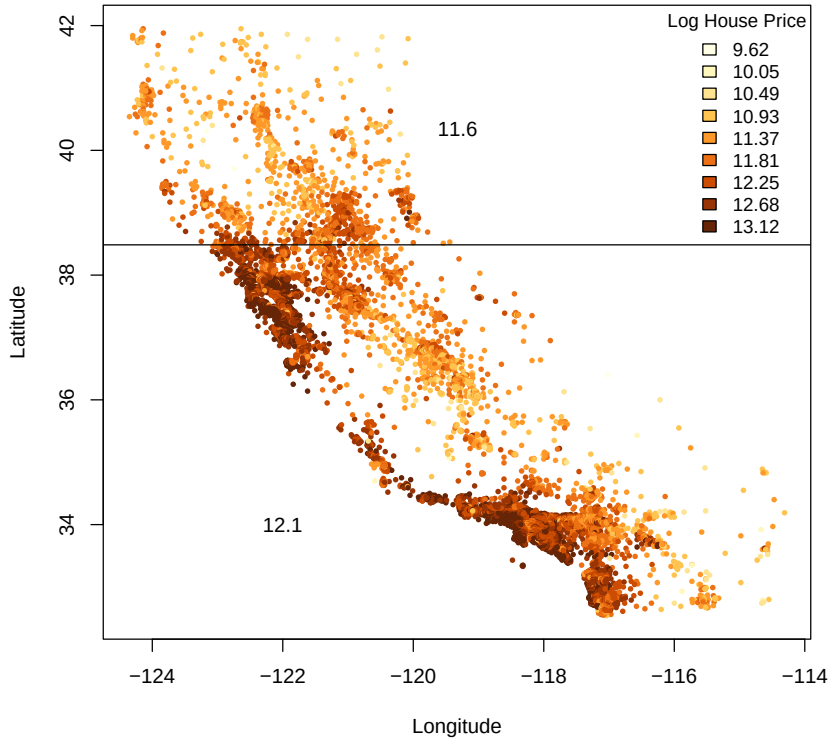
Splitting into Regions

At which latitude or longitude do you think we can distinguish best between higher and lower house prices?

Splitting into Regions

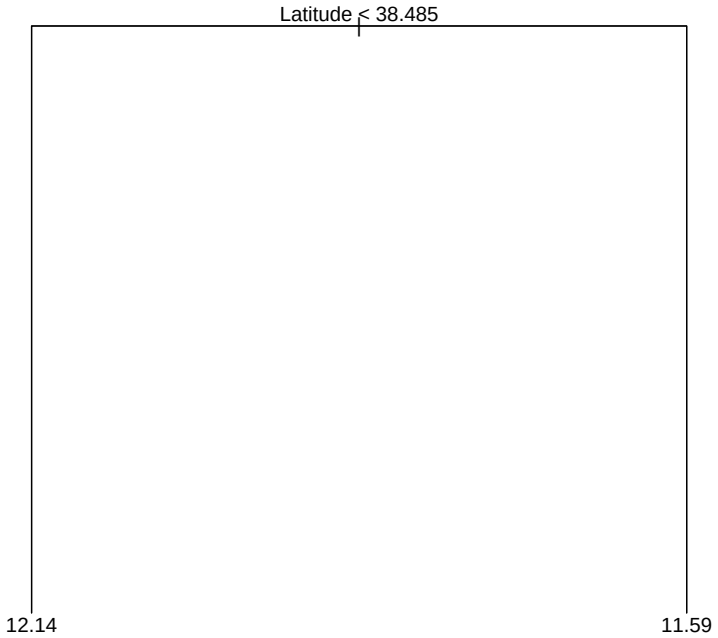
At which latitude or longitude do you think we can distinguish best between higher and lower house prices?

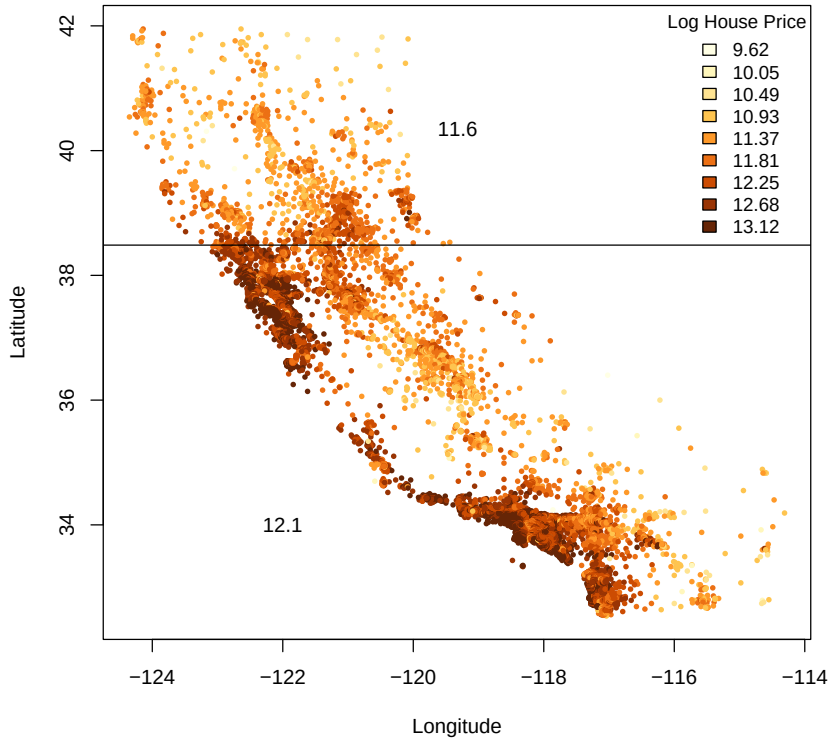




Splitting into Regions

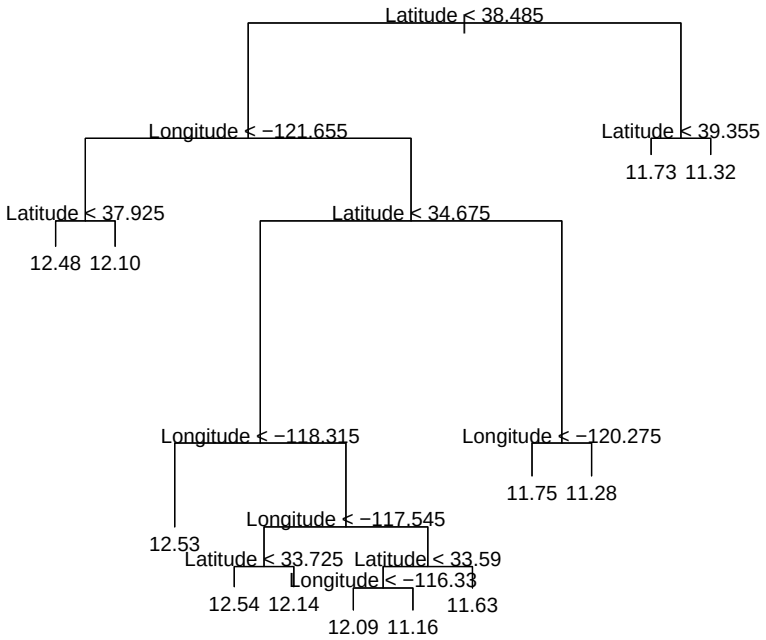
We now ask the same question again for each sub-region.

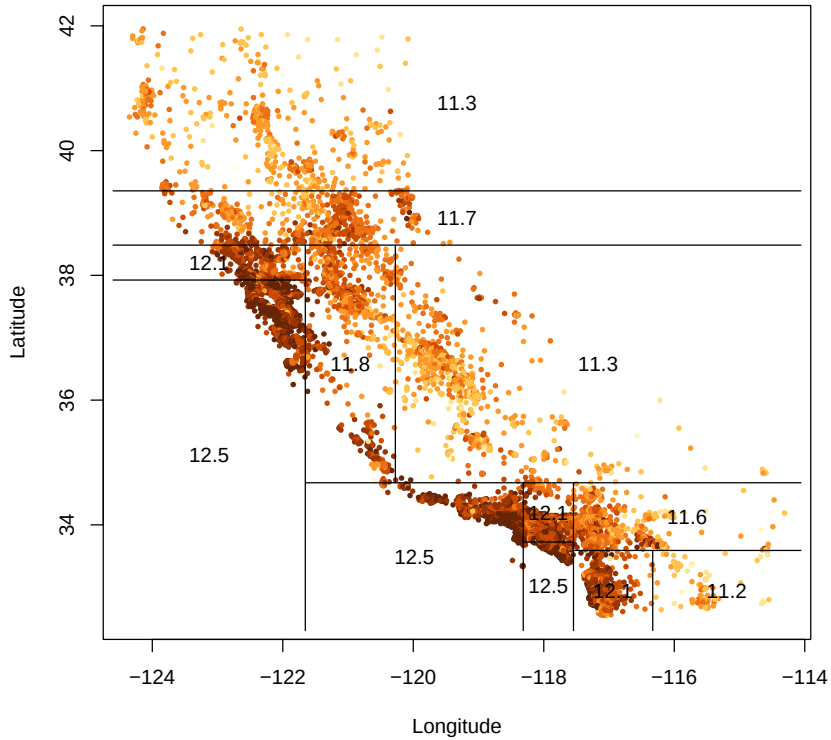




Splitting into Regions

Yielding the following result after 11 iterations:

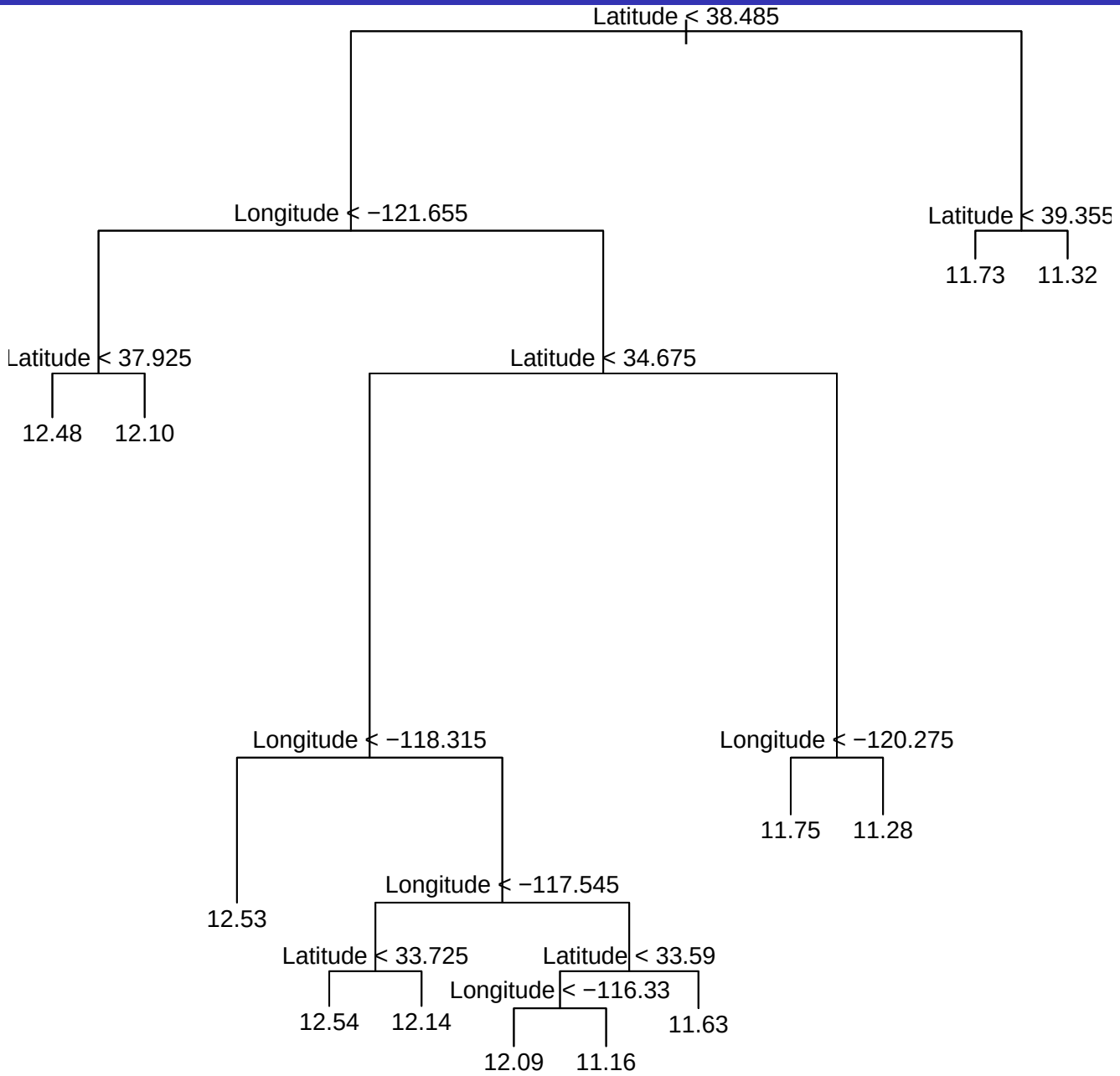




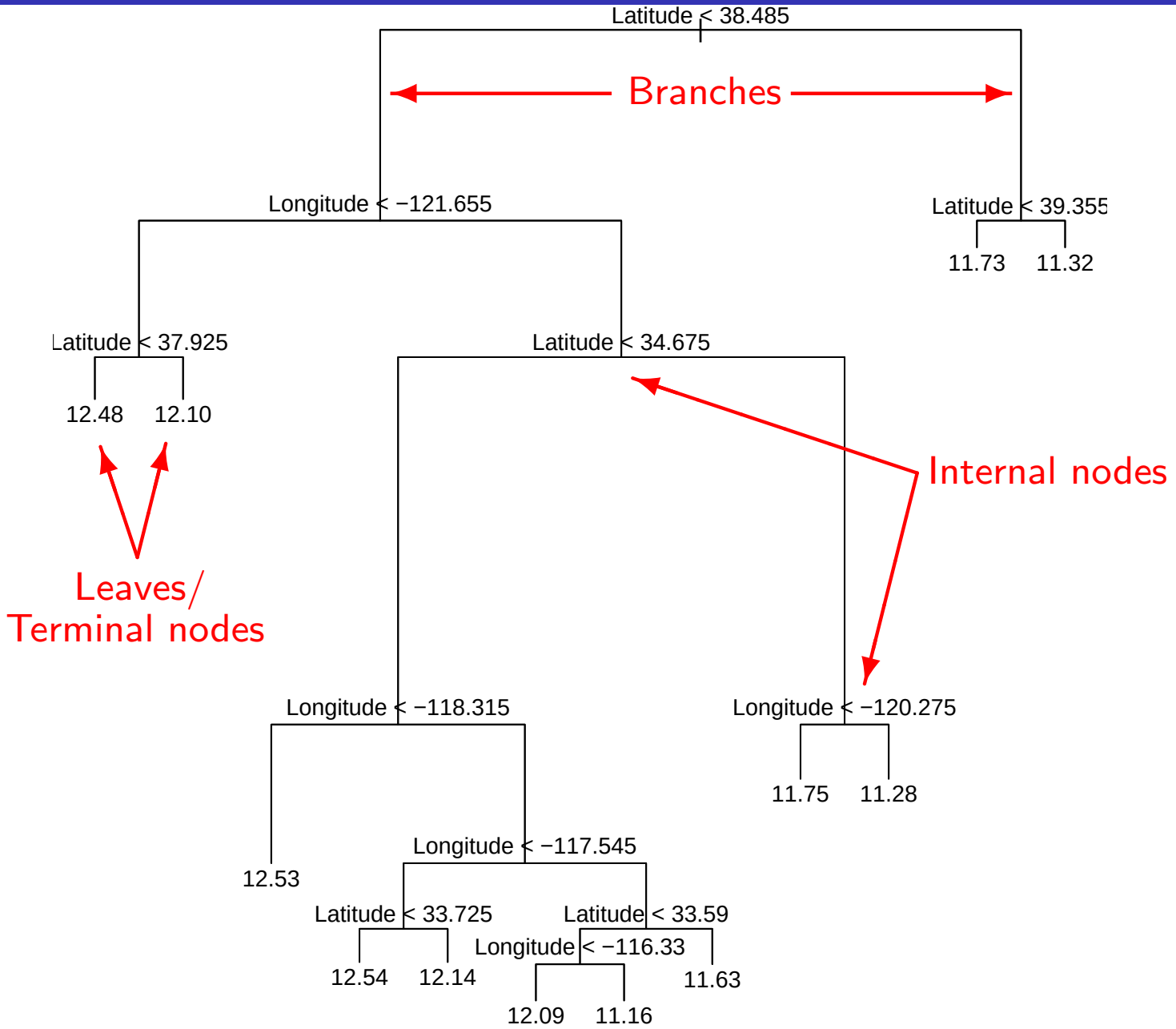
Terminology

- **Parent node**: The immediate predecessor of Node j.
- **Child node**: The immediate successor of a parent node.
- **Root node**: The top node of the tree, all observations are together in this node.
- **Terminal node**: A node that does not have children nodes, decisions are made at these nodes. Also referred to as **leaves**.
- **Internal node**: Any node that is not a terminal node.
- **Depth**: The maximal length of a path from the root node to a terminal node.

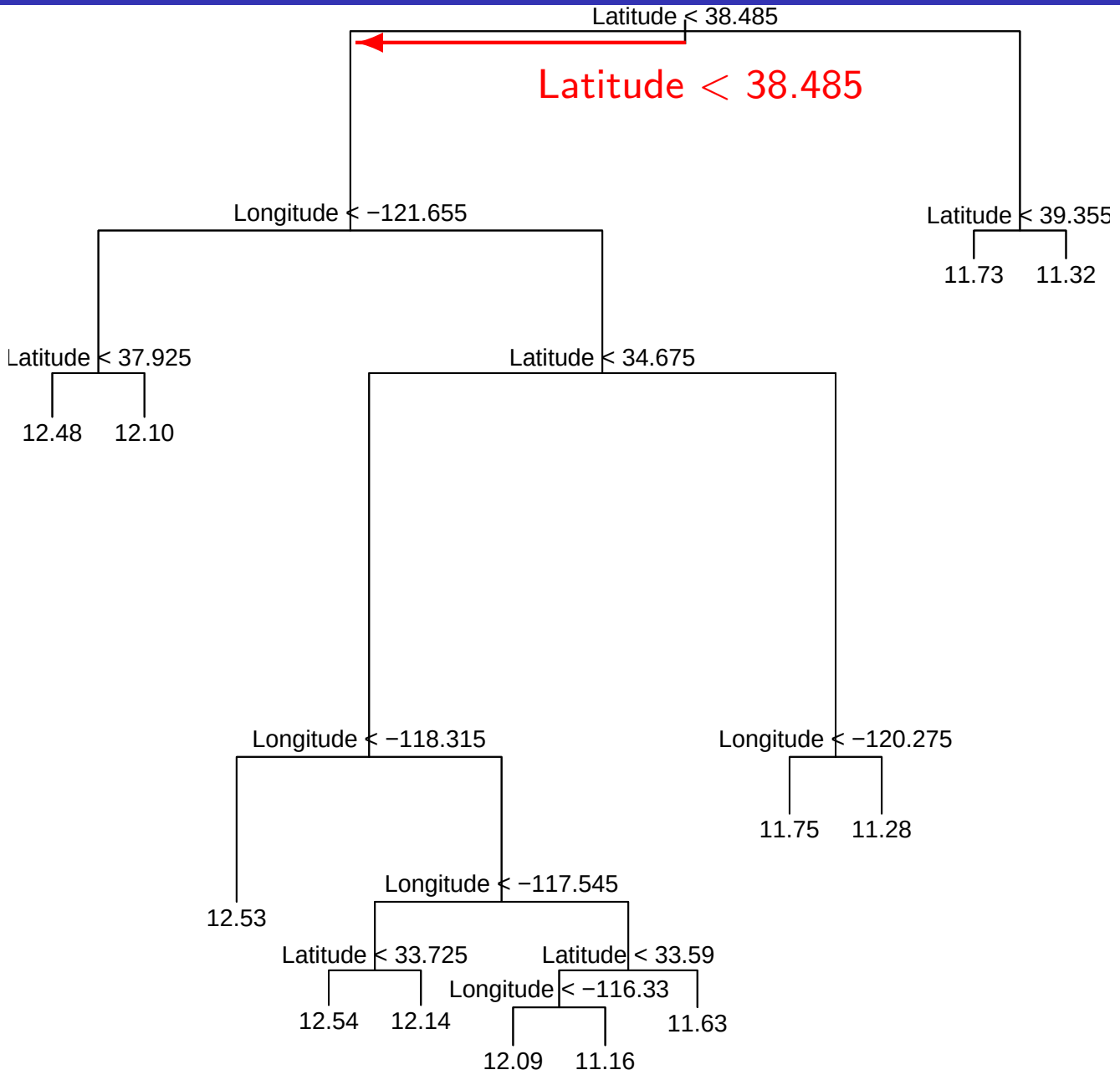
Regression Tree



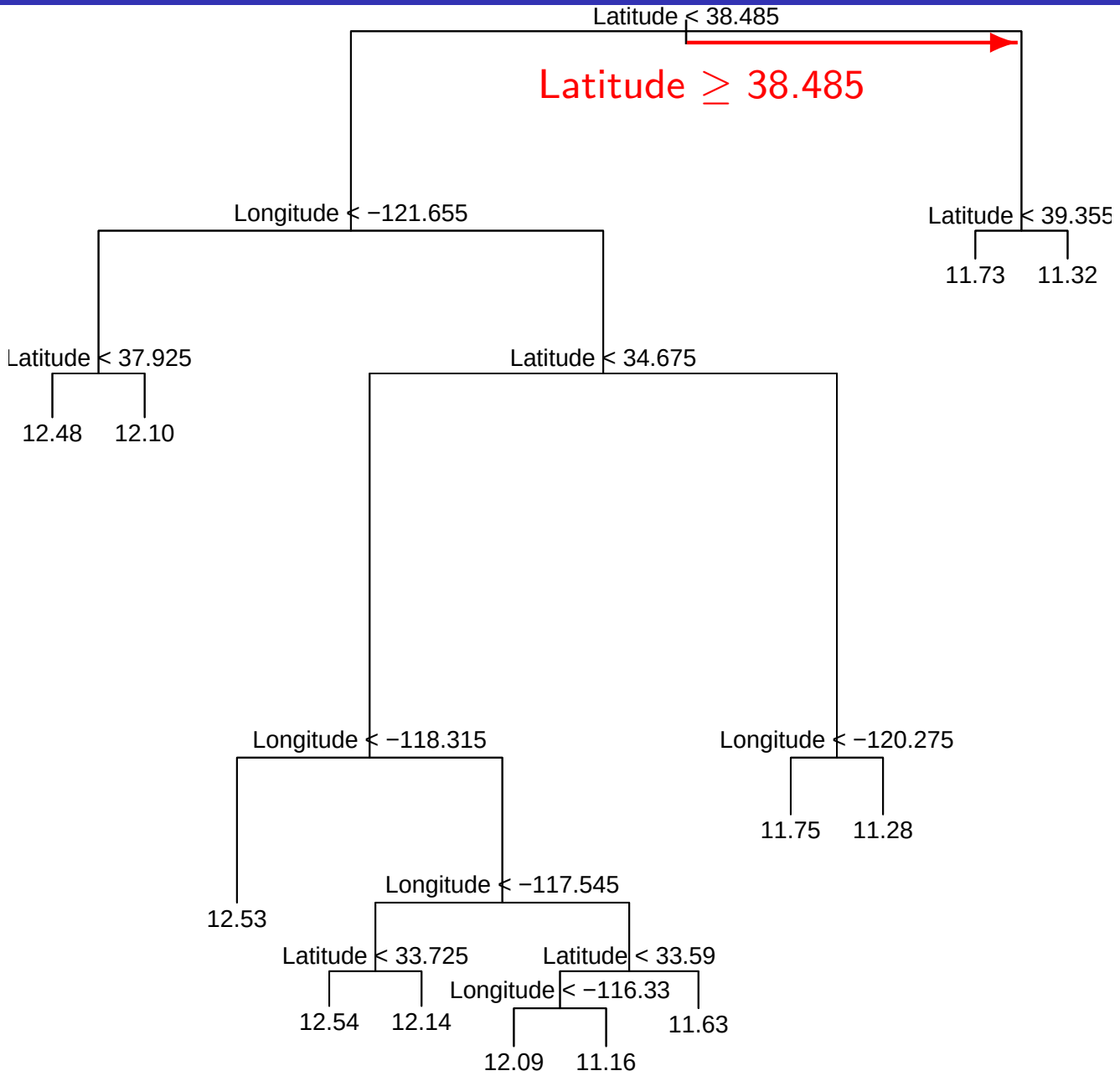
Regression Tree



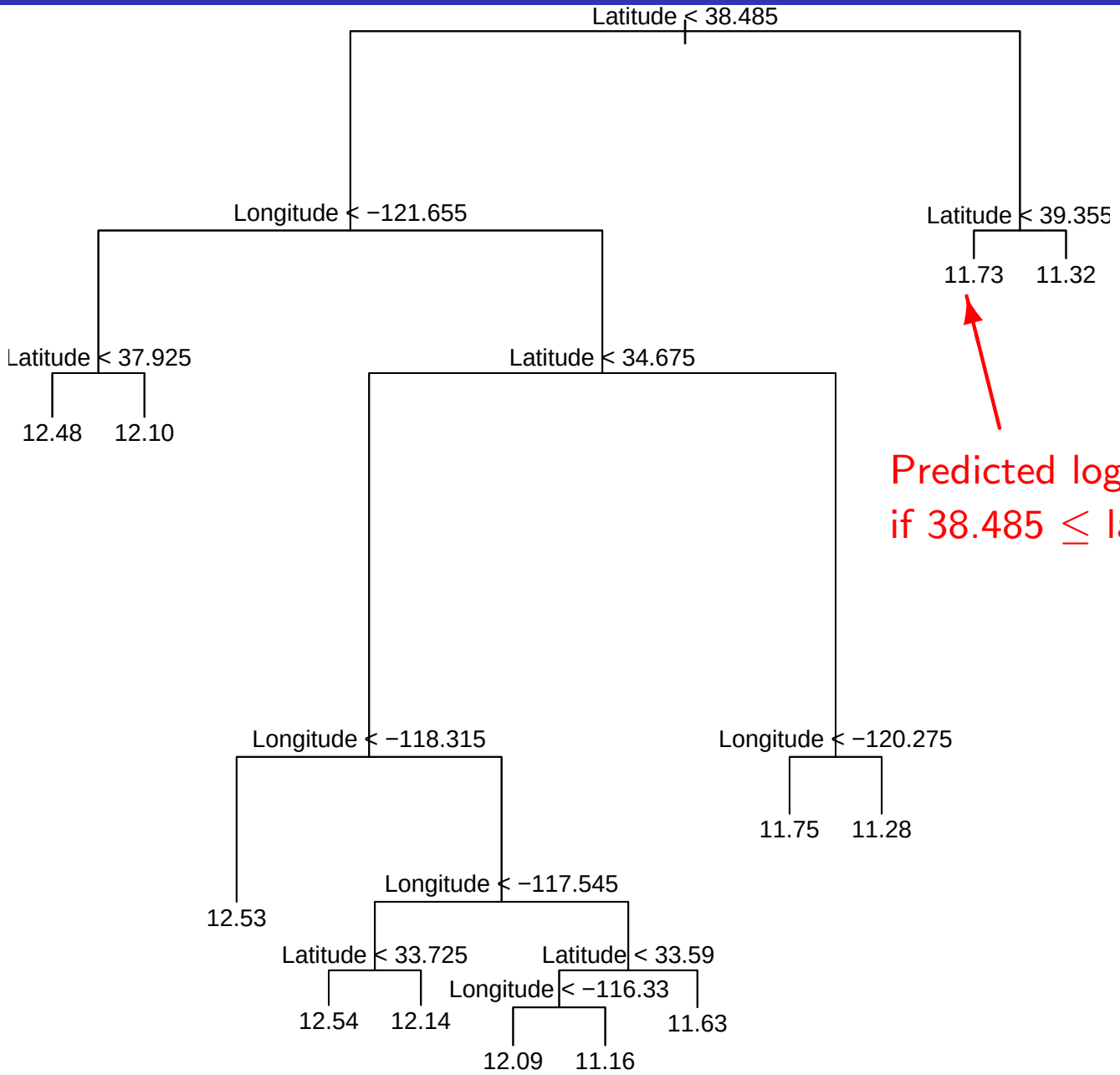
Regression Tree



Regression Tree



Regression Tree



Predicted log house price
if $38.485 \leq \text{lat} < 39.355$

Tree size

- We now need to decide into how many regions the feature space will be split.
- In other words, how large (deep) our tree will be grown. This is also referred to as the model's complexity.
- Is there such a thing as a "too big tree"?
- If we continue splitting until every single observation is in its own region, then we will have perfect classification of all observations.
- However, this would be extreme **overfitting**.

Overfitting of Regression Trees

- Our procedure for fitting a regression tree will produce a complex model with many splits.
- A simpler tree may yield better predictions on out-of-sample data and is easier to interpret.
- One alternative is to continue growing the tree only until the decrease in RSS fails to exceed some high threshold.
- But this is short-sighted since a split later on might produce an even greater reduction in RSS.
- A better strategy is therefore to grow a very large tree and then **prune** it back to obtain a smaller **subtree**.

Tree Pruning

- Our goal is to find the subtree with the lowest test error.
- Given a subtree, we can estimate its test error using cross-validation.
- However, estimating the CV error for every possible subtree is not computationally feasible.
- We will therefore consider only a sequence of trees obtained by pruning the full tree back in a nested manner.
- This tree pruning technique is known as **cost complexity pruning** or weakest link pruning.

Cost Complexity Pruning

- Let T_0 be the full tree and let $T \subseteq T_0$ be a subtree with $|T|$ terminal nodes or, equivalently, regions in the feature space
- Consider the penalised RSS

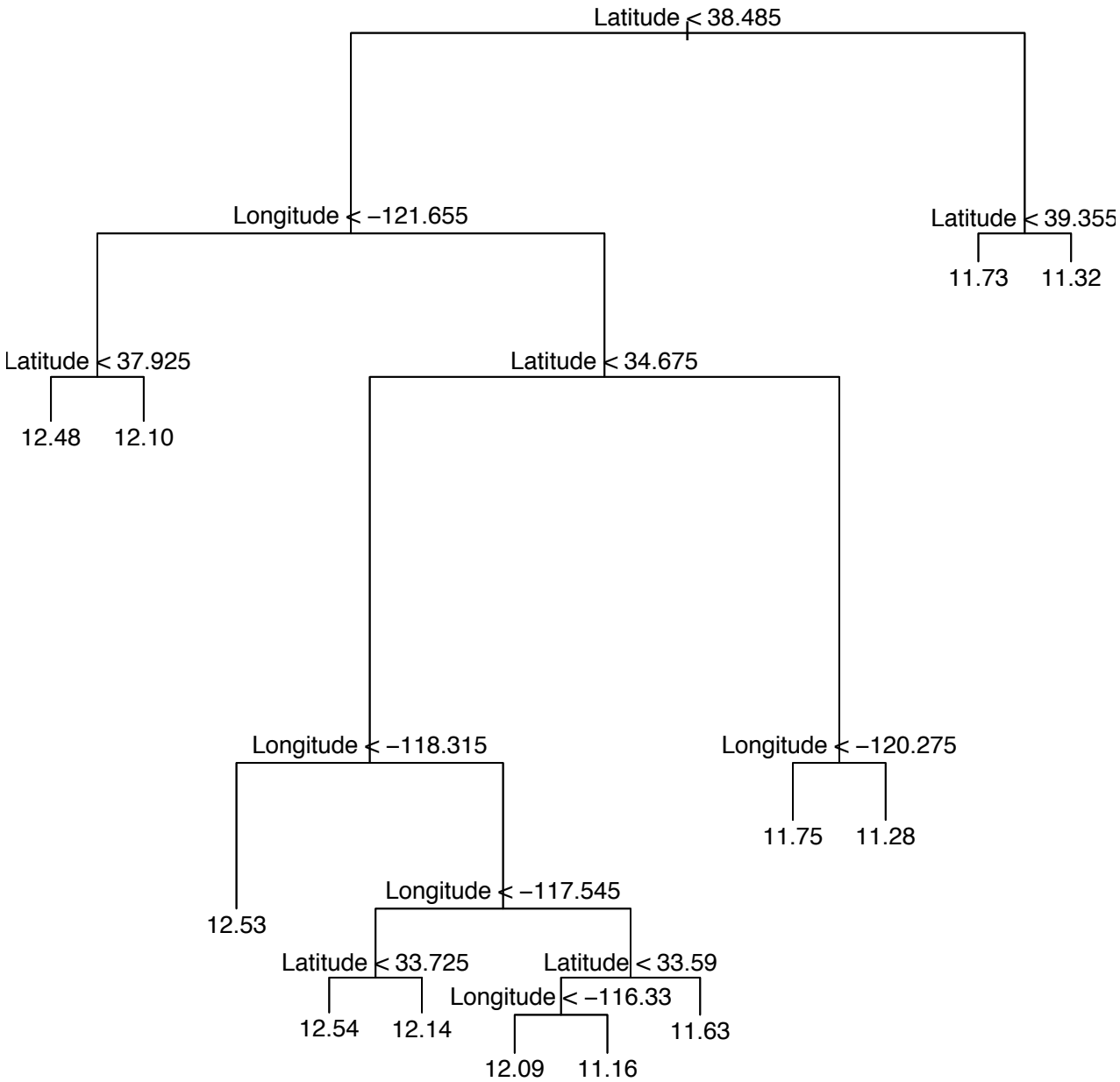
$$\text{RSS}_\alpha = \sum_{j=1}^{|T|} \sum_{i:x_i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha|T|$$

  where $\alpha \geq 0$ is a tuning parameter
- For each value of α, we seek the subtree T that minimises RSS_α
- When $\alpha = 0$, $\text{RSS}_\alpha = \text{RSS}$ and therefore $T = T_0$
- As α increases, subtrees with a large number of leaves are penalised more heavily and a smaller T will be favoured

Cost Complexity Pruning

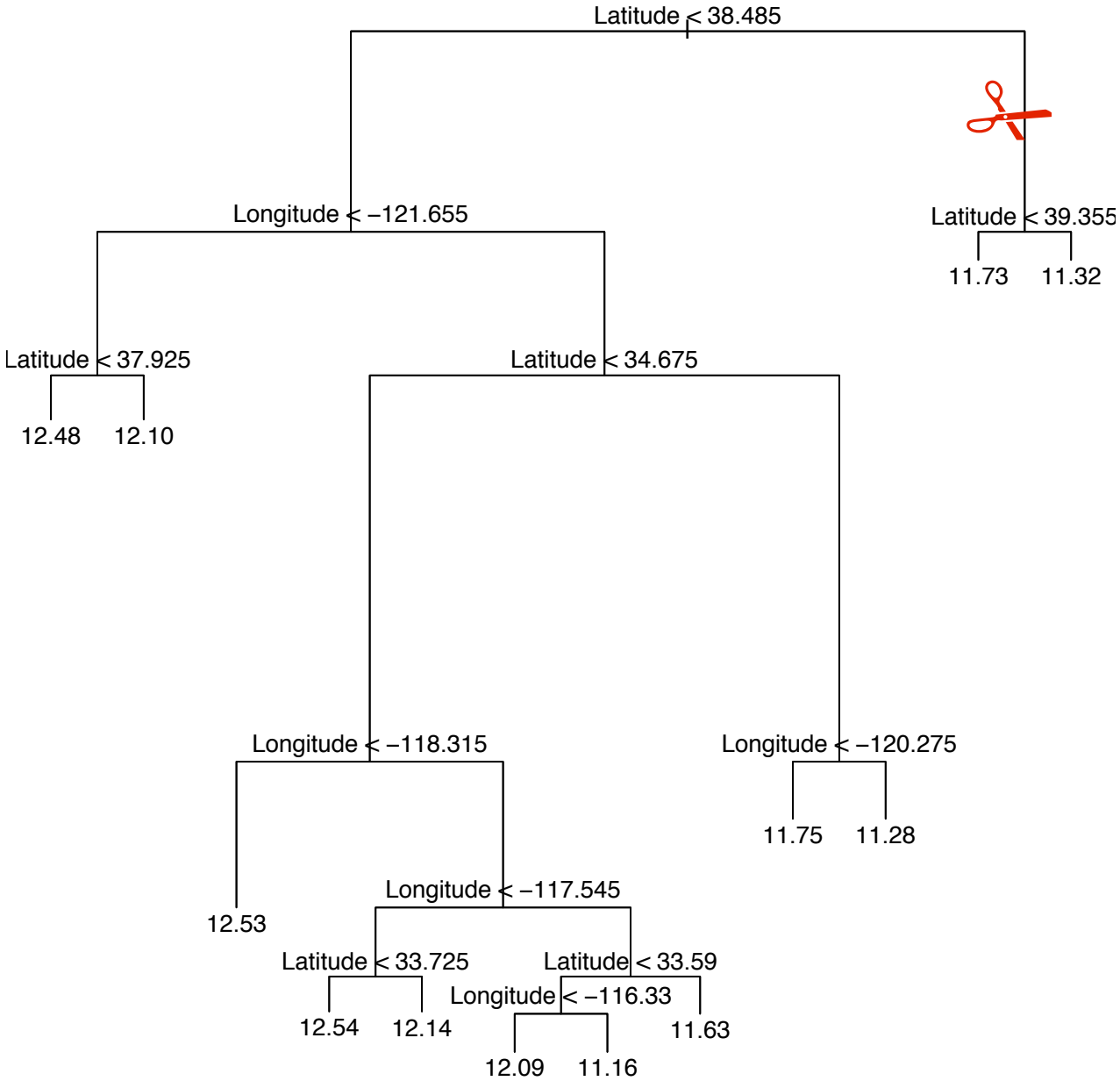
- It turns out that as we increase α from zero, branches get pruned from the tree in a nested fashion
- It is therefore easy to obtain a sequence of subtrees as a function of $\alpha \ldots$

Tree Pruning



$$\alpha = 0$$

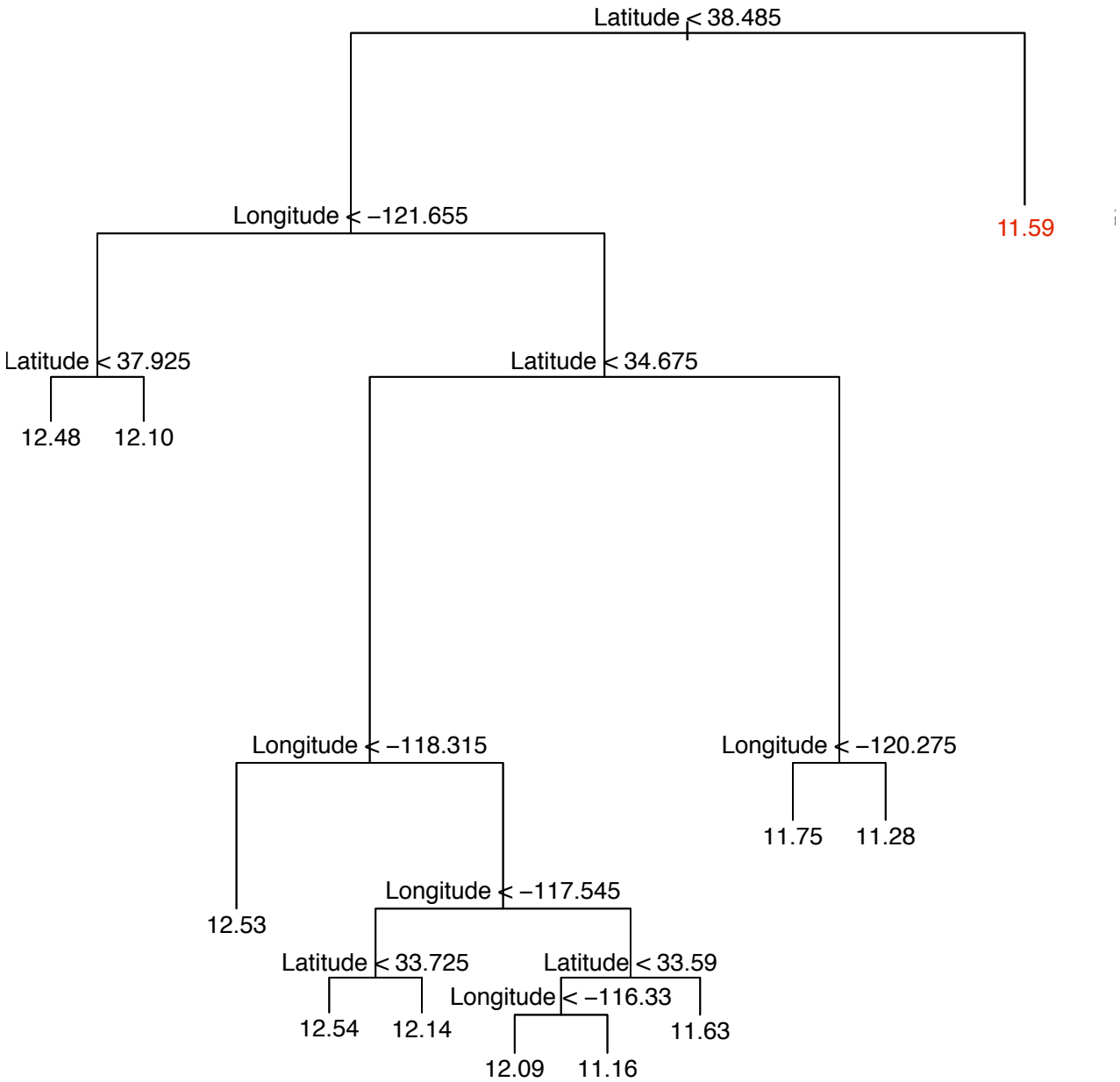
Tree Pruning



77.4

α

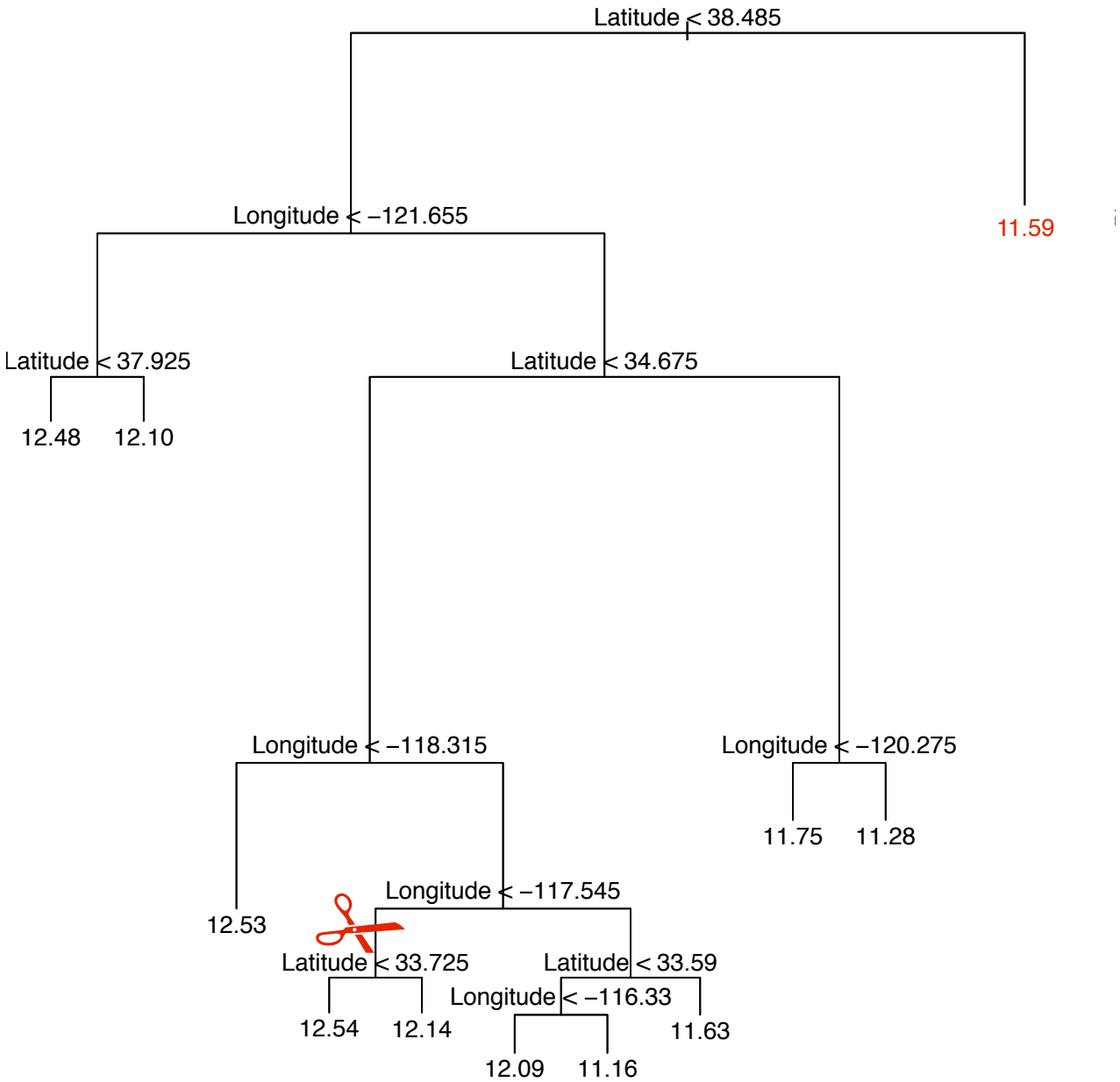
Tree Pruning

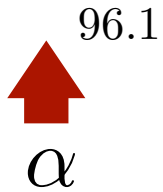


77.4

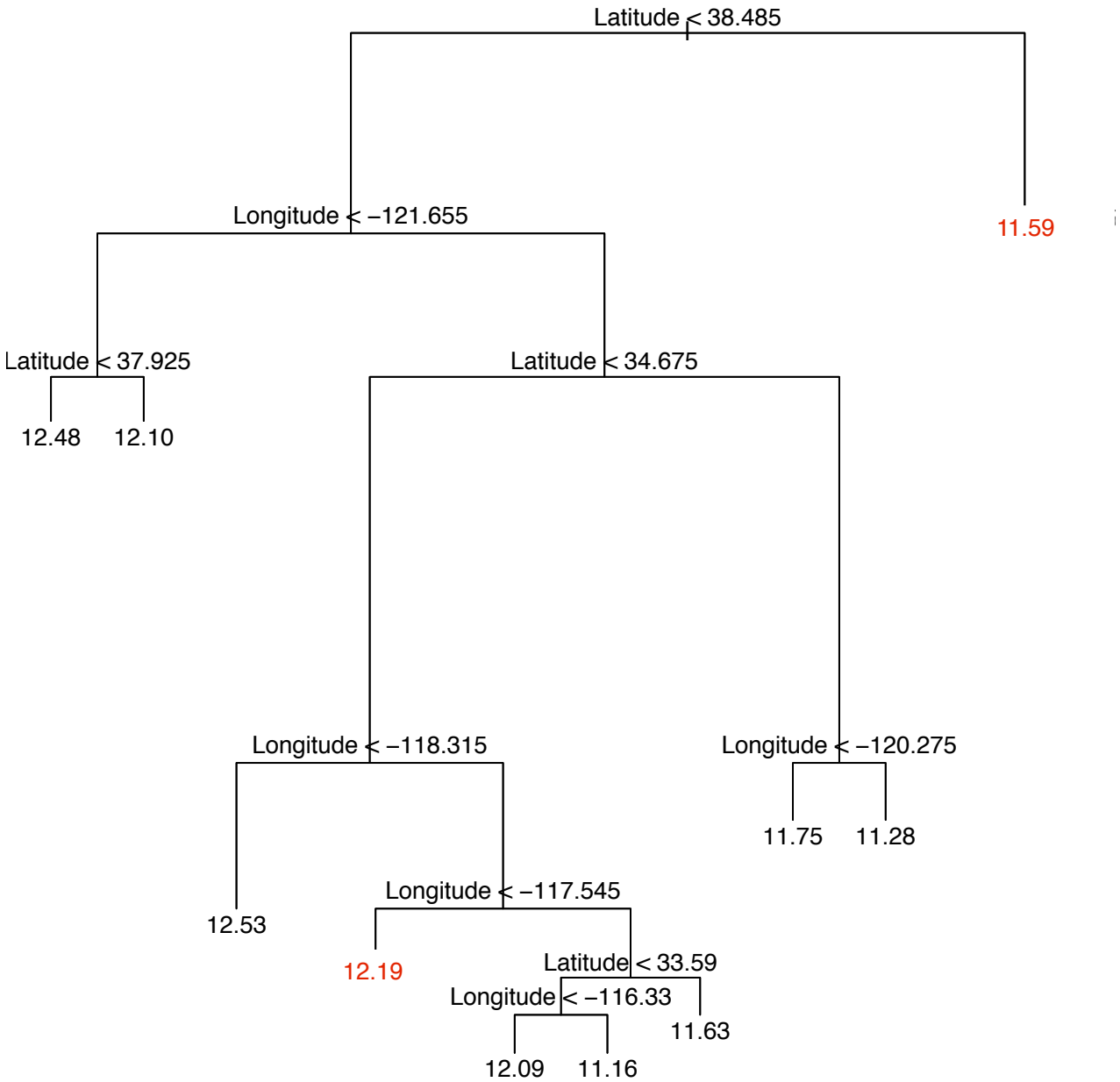
α

Tree Pruning





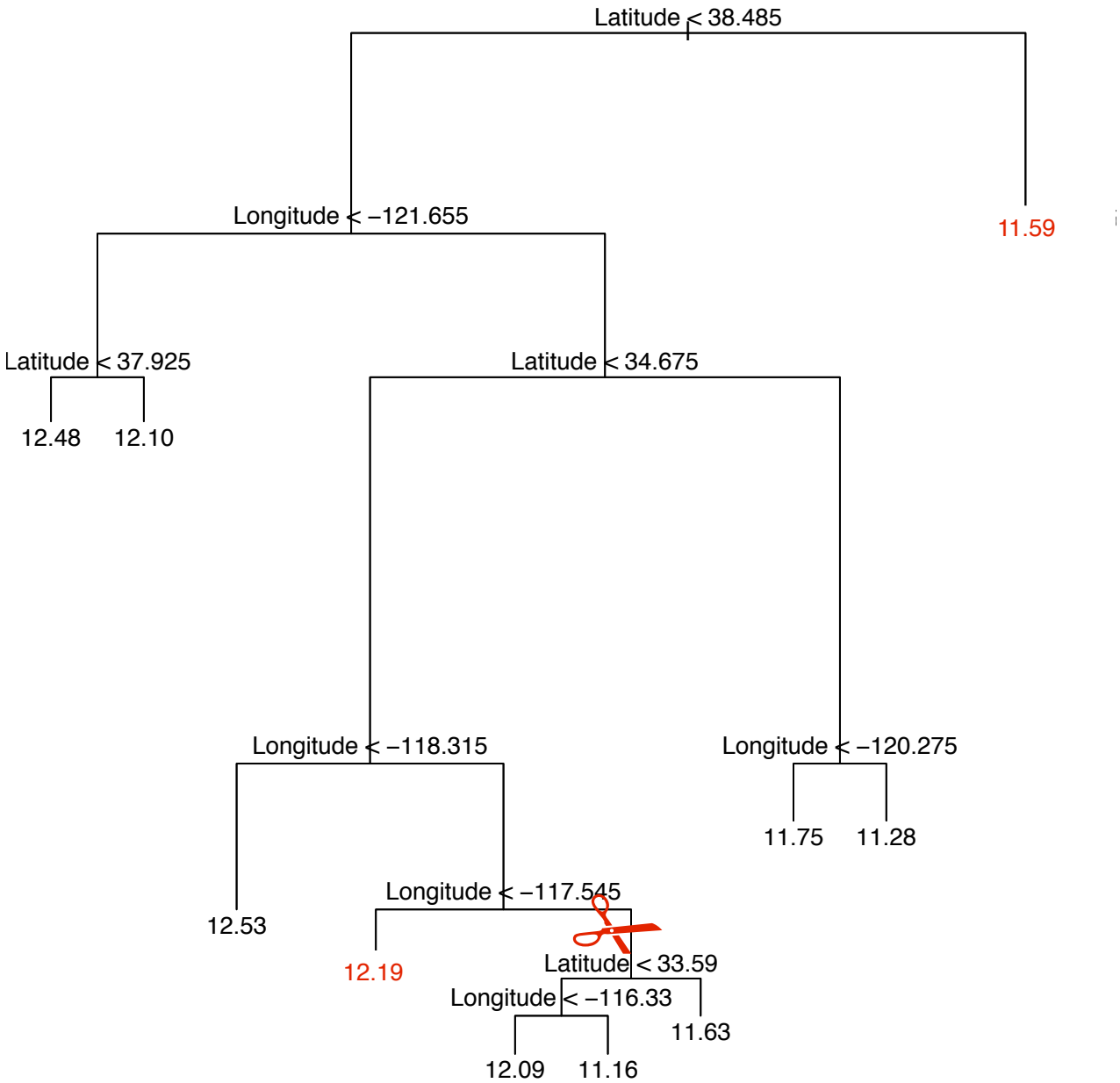
Tree Pruning



96.1

α

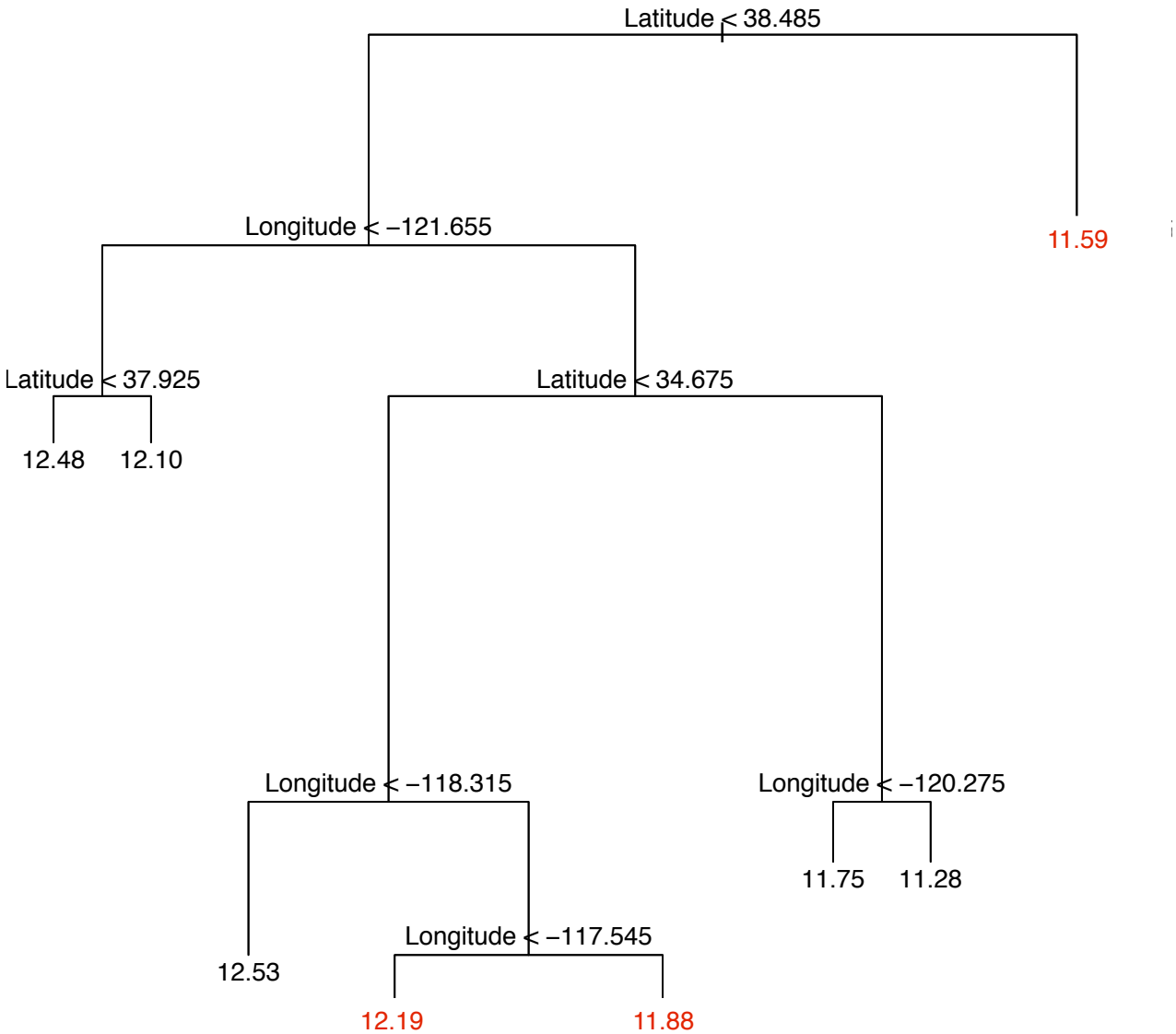
Tree Pruning

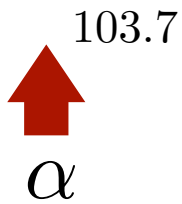


103.7

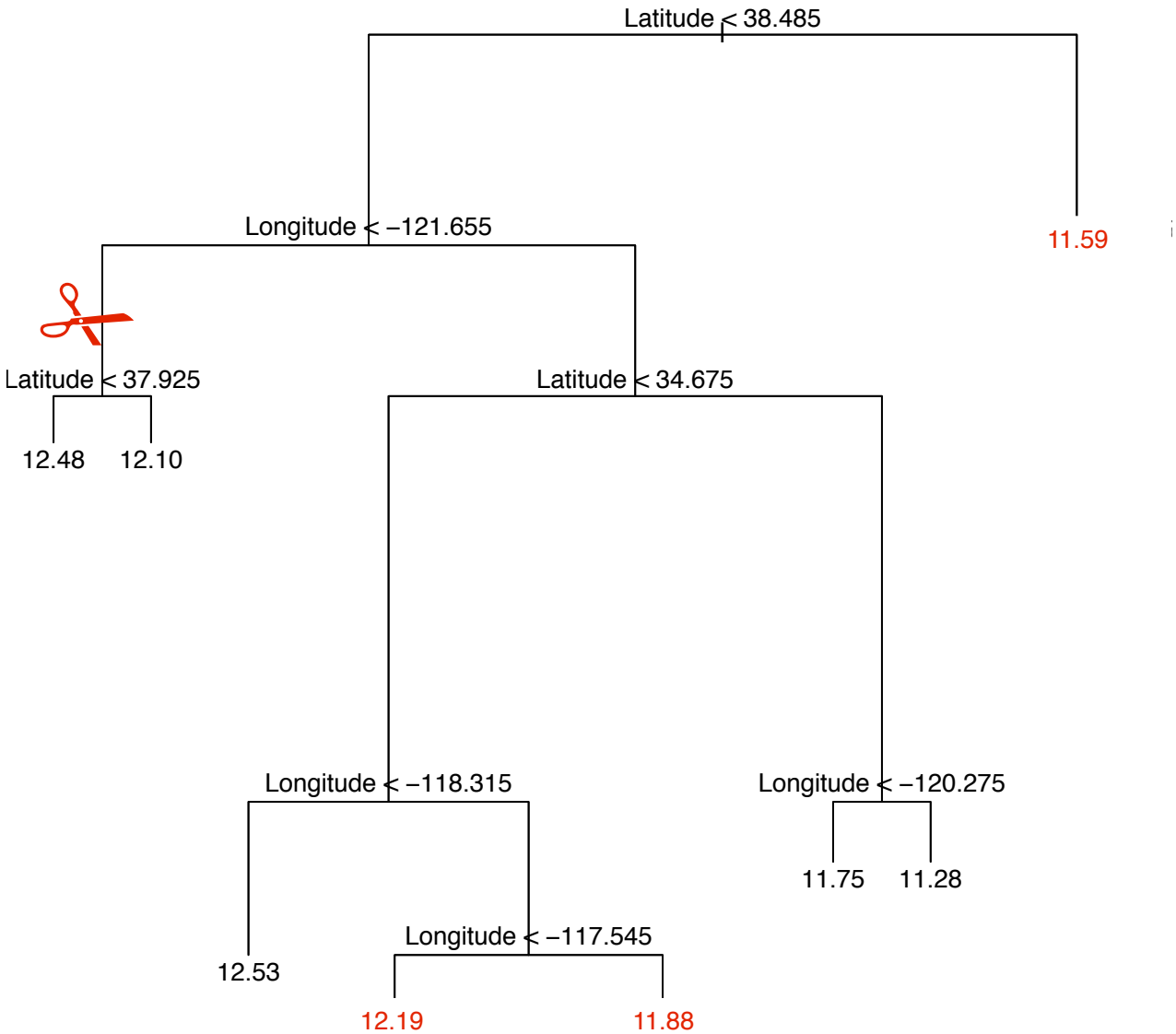
α

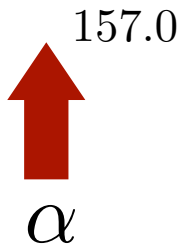
Tree Pruning



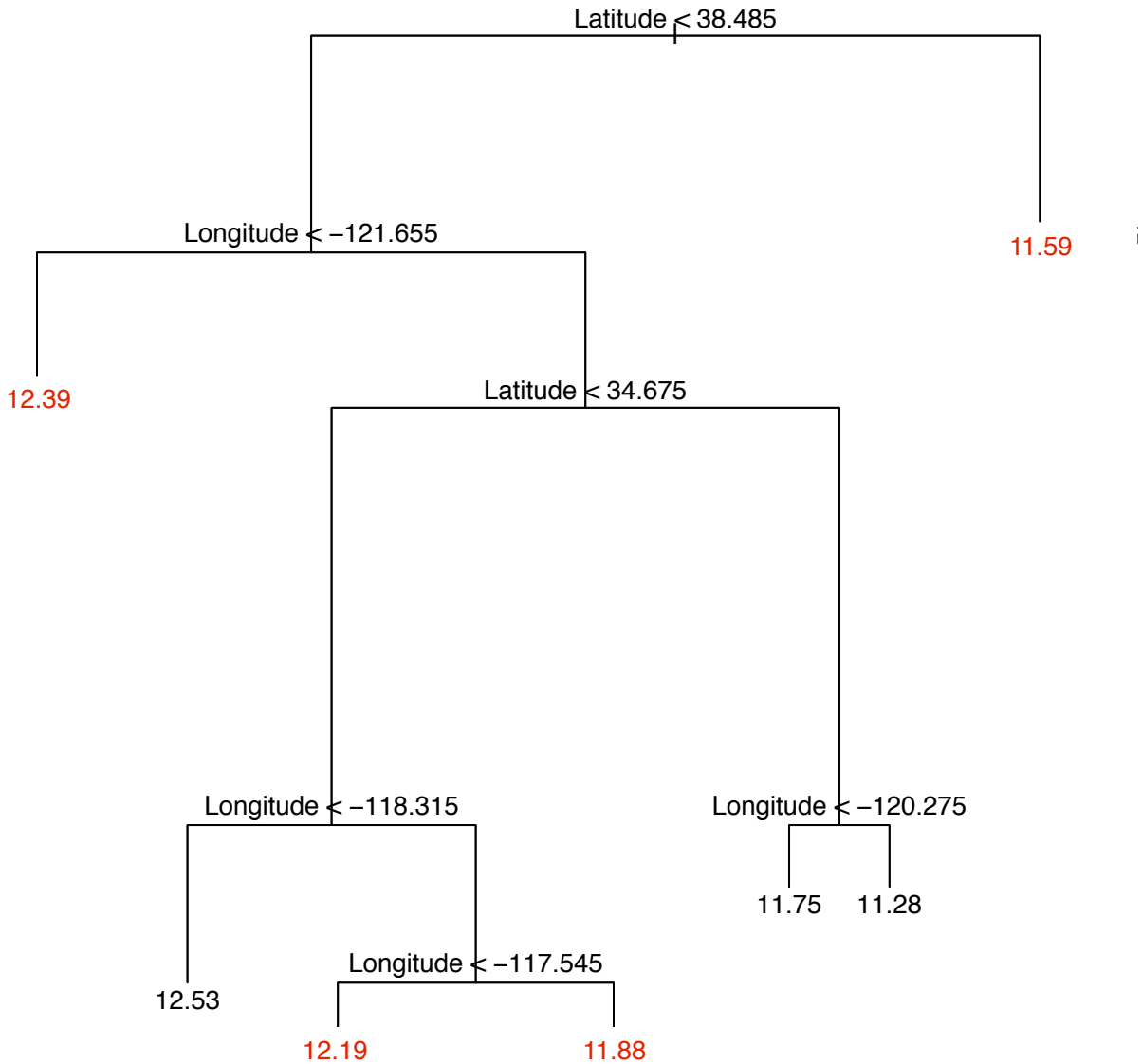


Tree Pruning

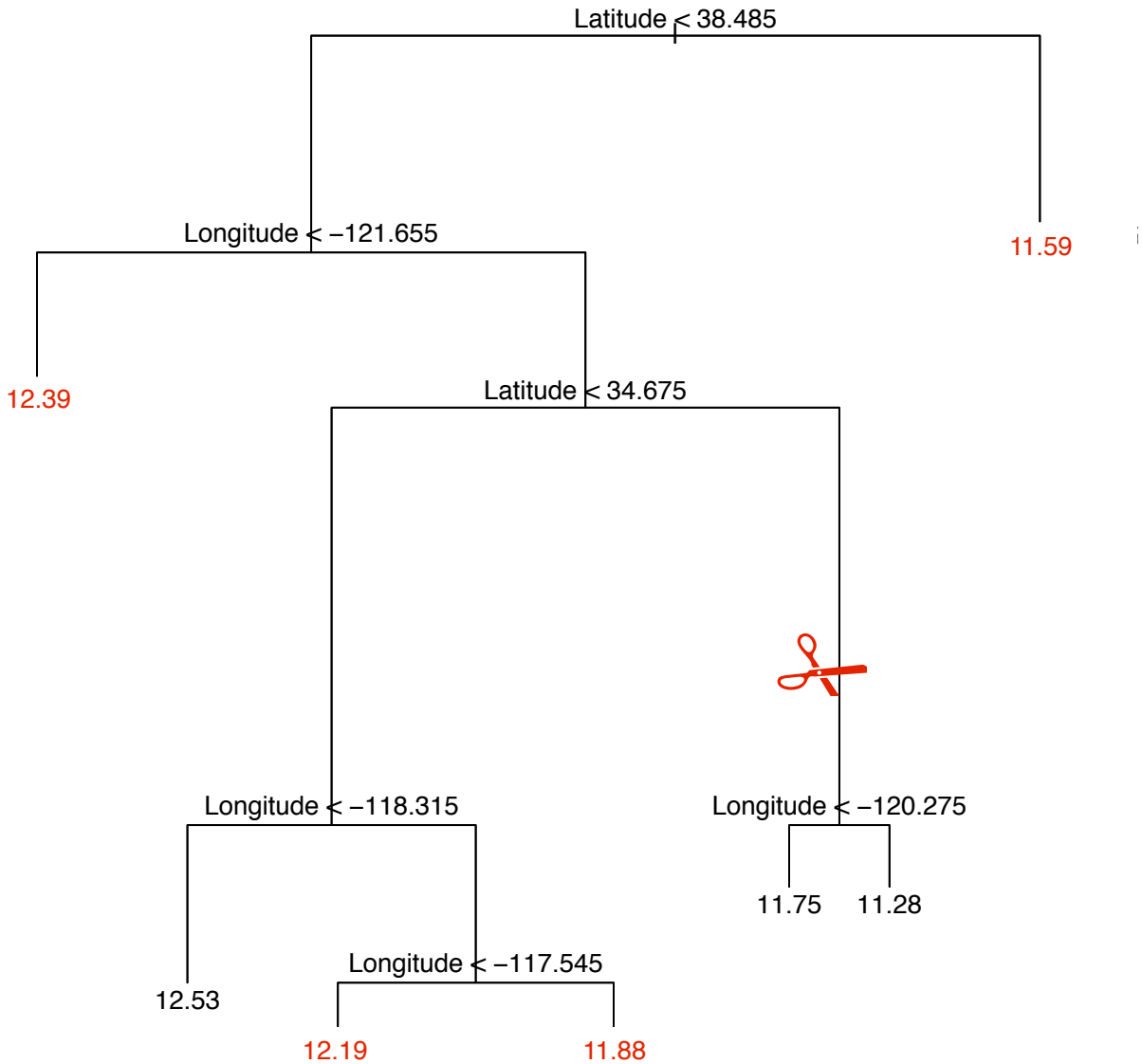


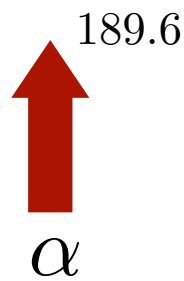


Tree Pruning

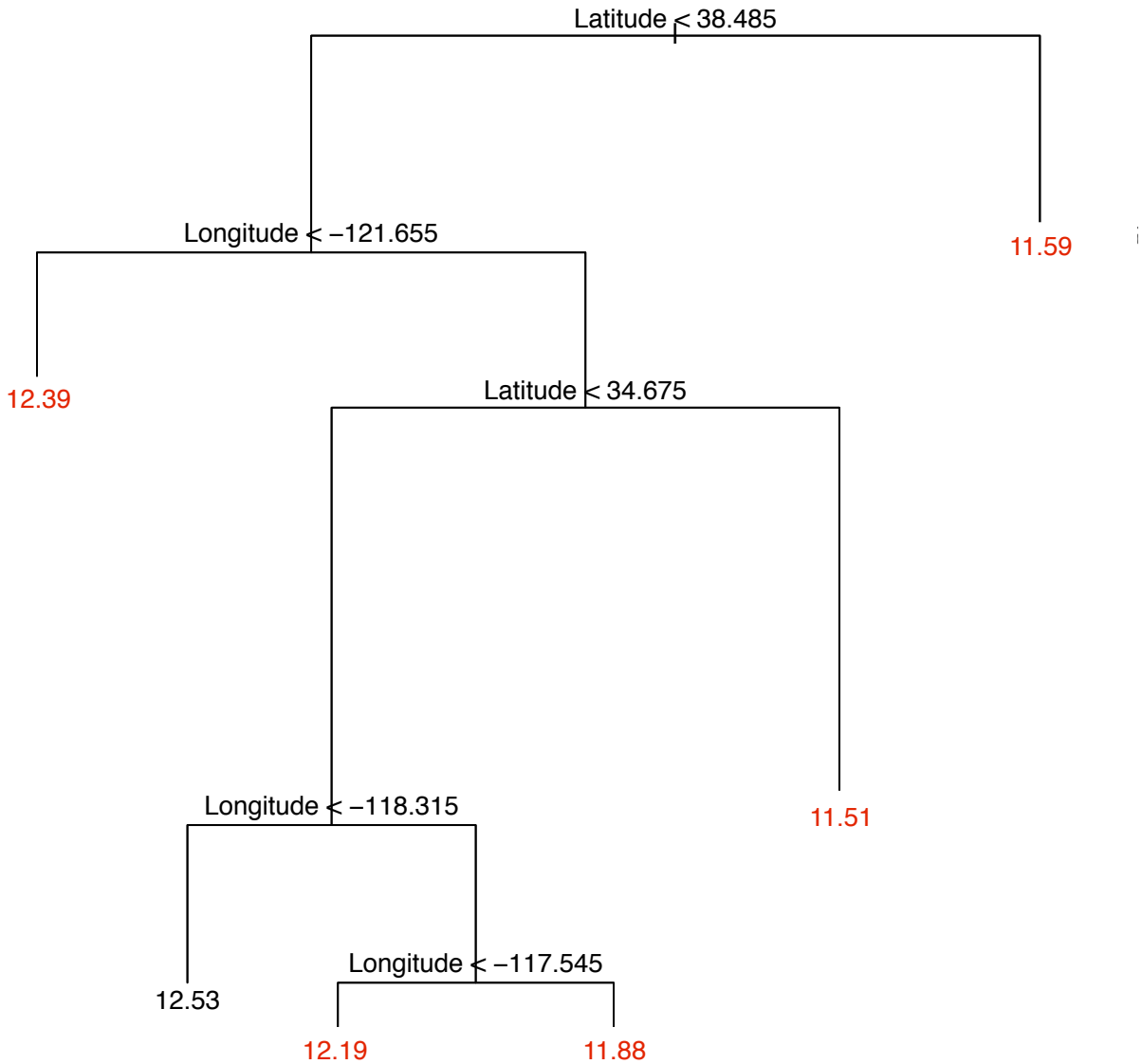


Tree Pruning

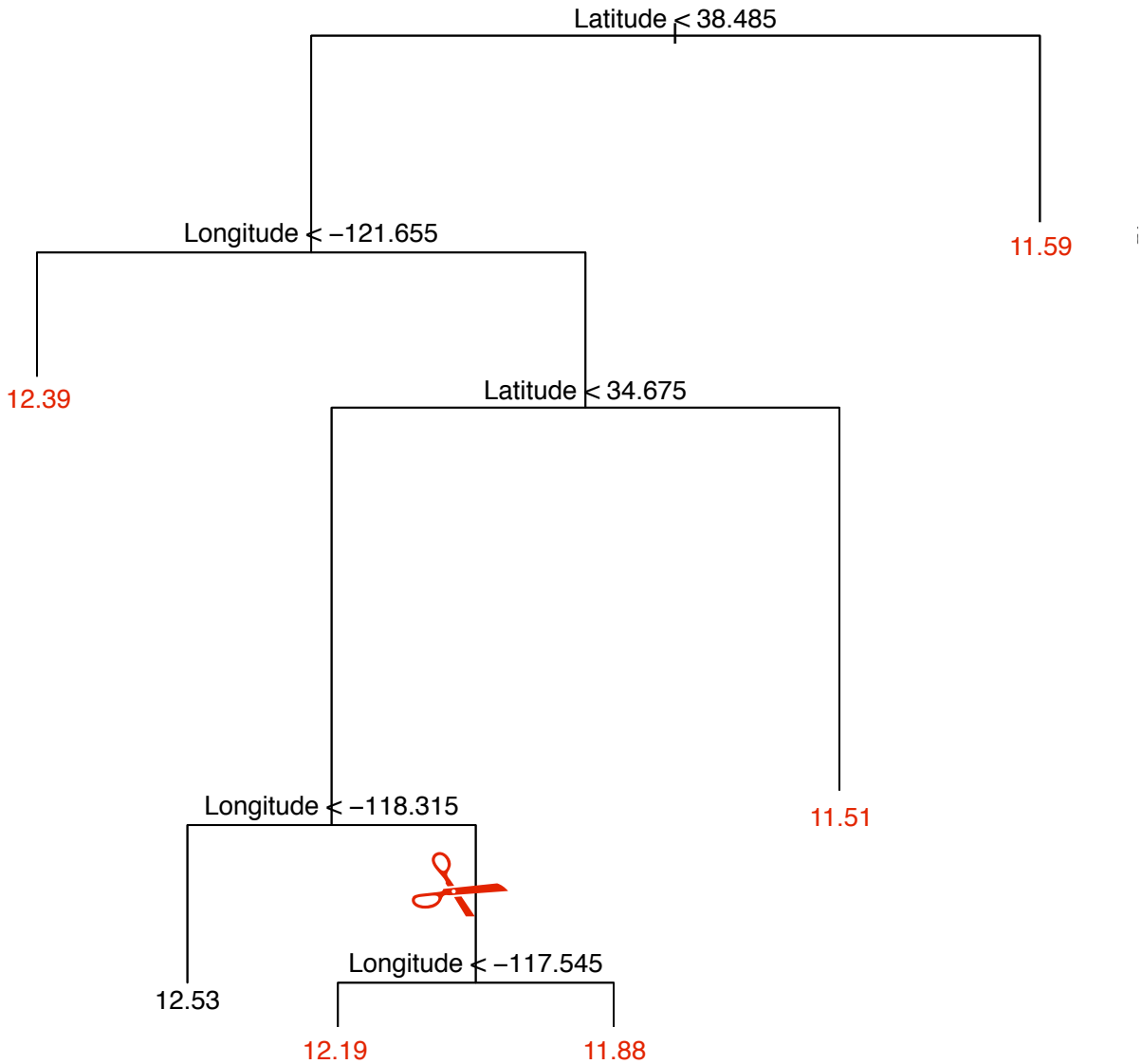




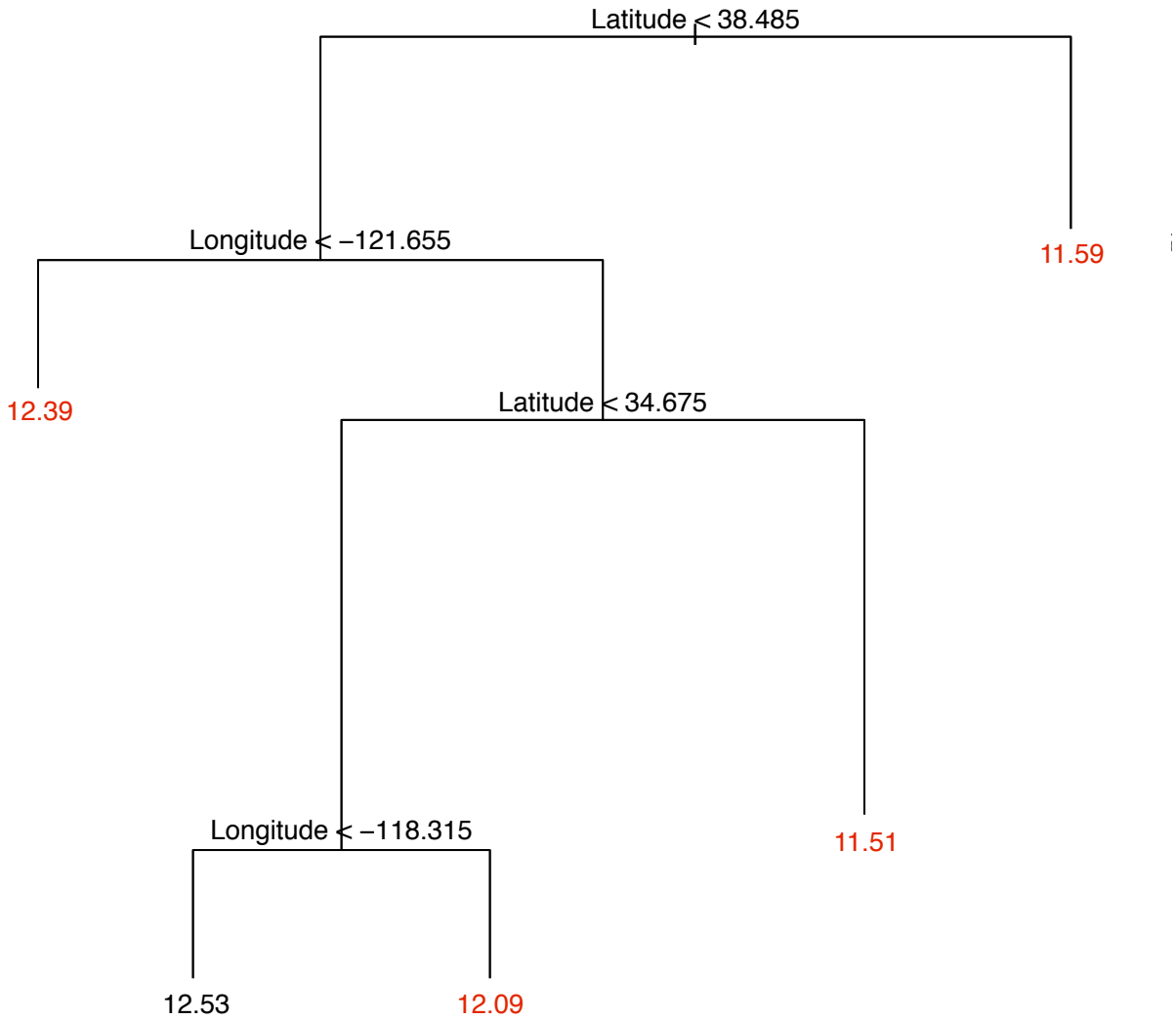
Tree Pruning



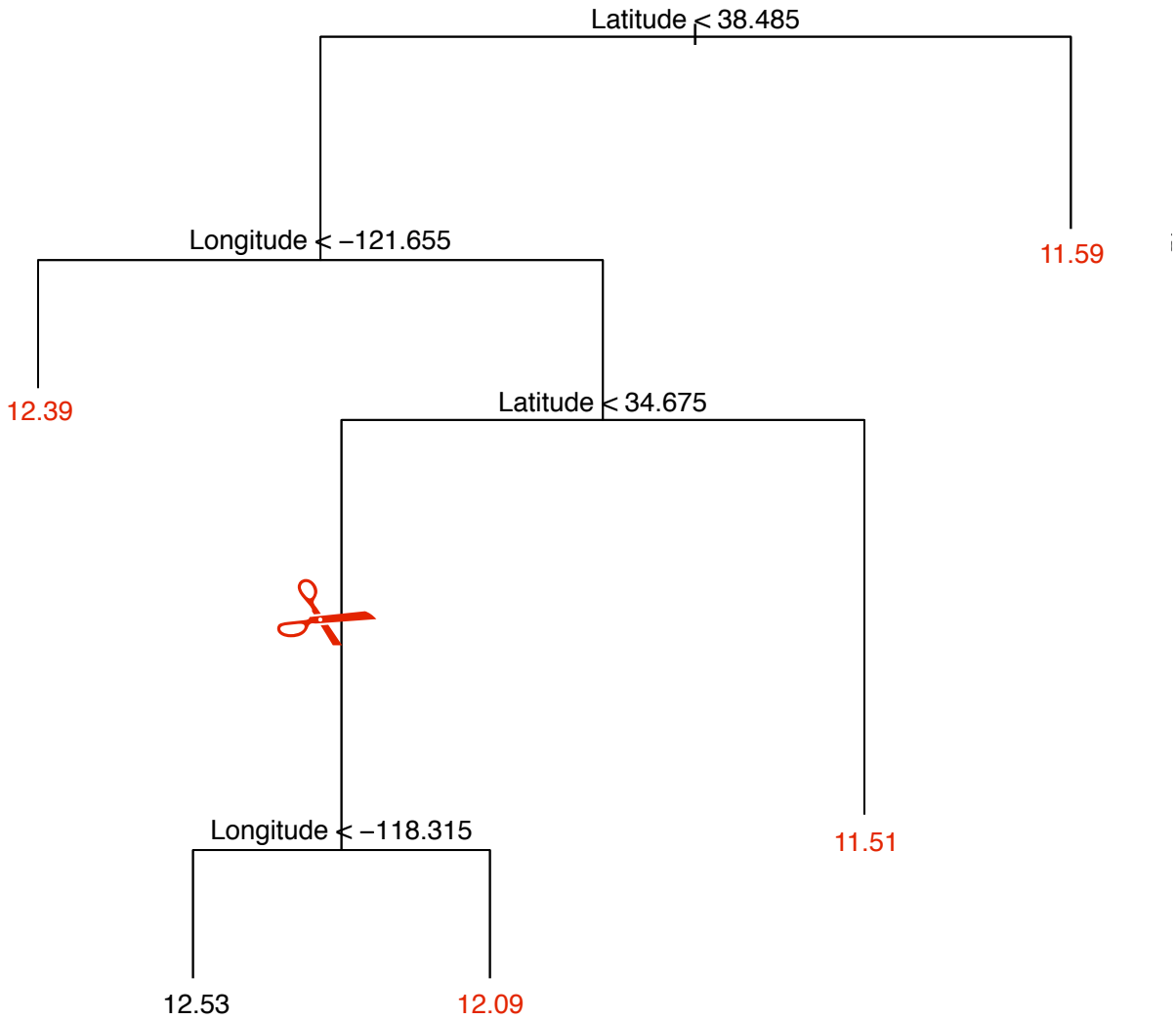
Tree Pruning



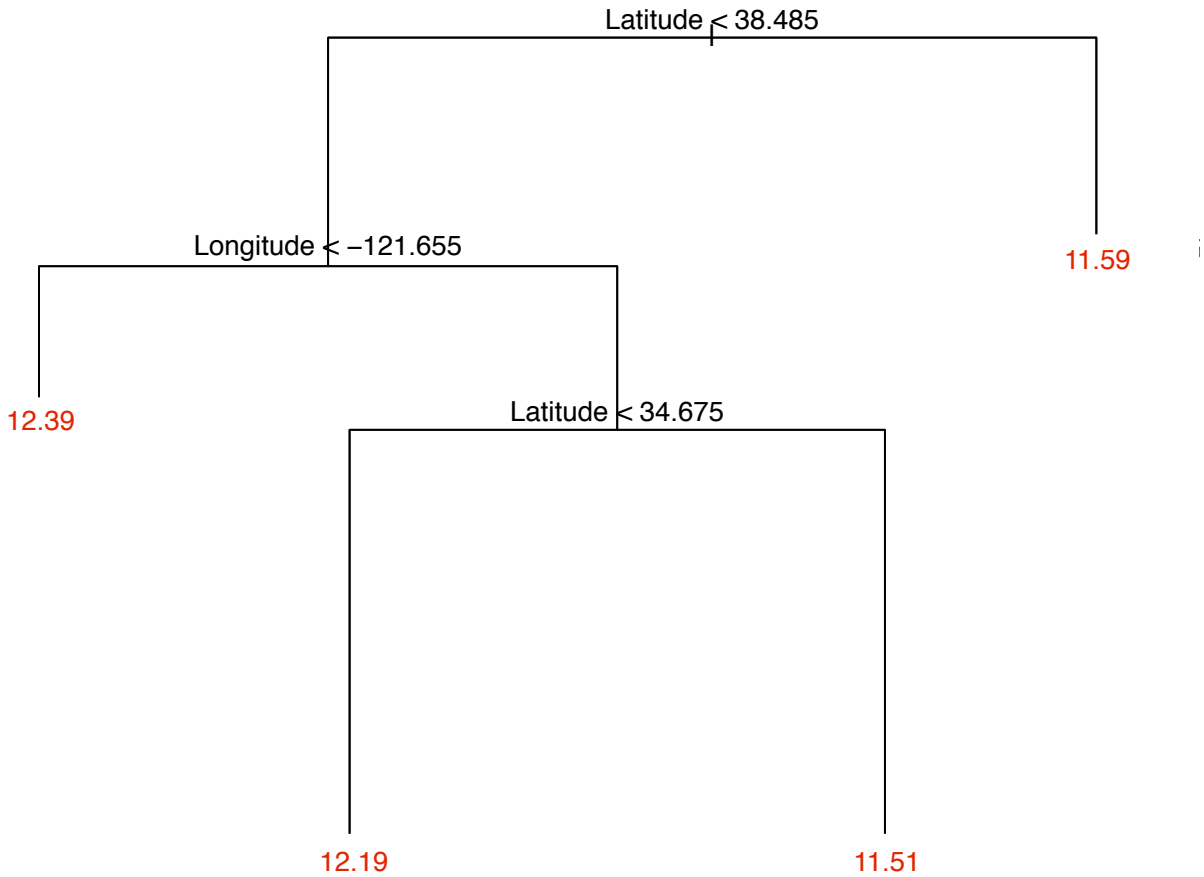
Tree Pruning

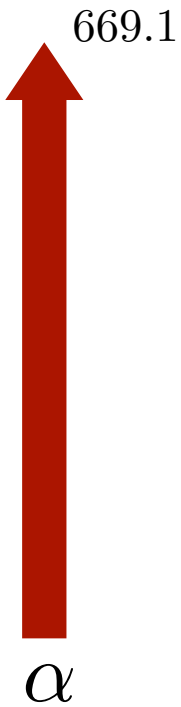


Tree Pruning



Tree Pruning





Tree Pruning

- By choosing α, we are effectively selecting a subtree; that is, a more parsimonious model that does not overfit the sample data.
- How do we choose α?

Tree Pruning

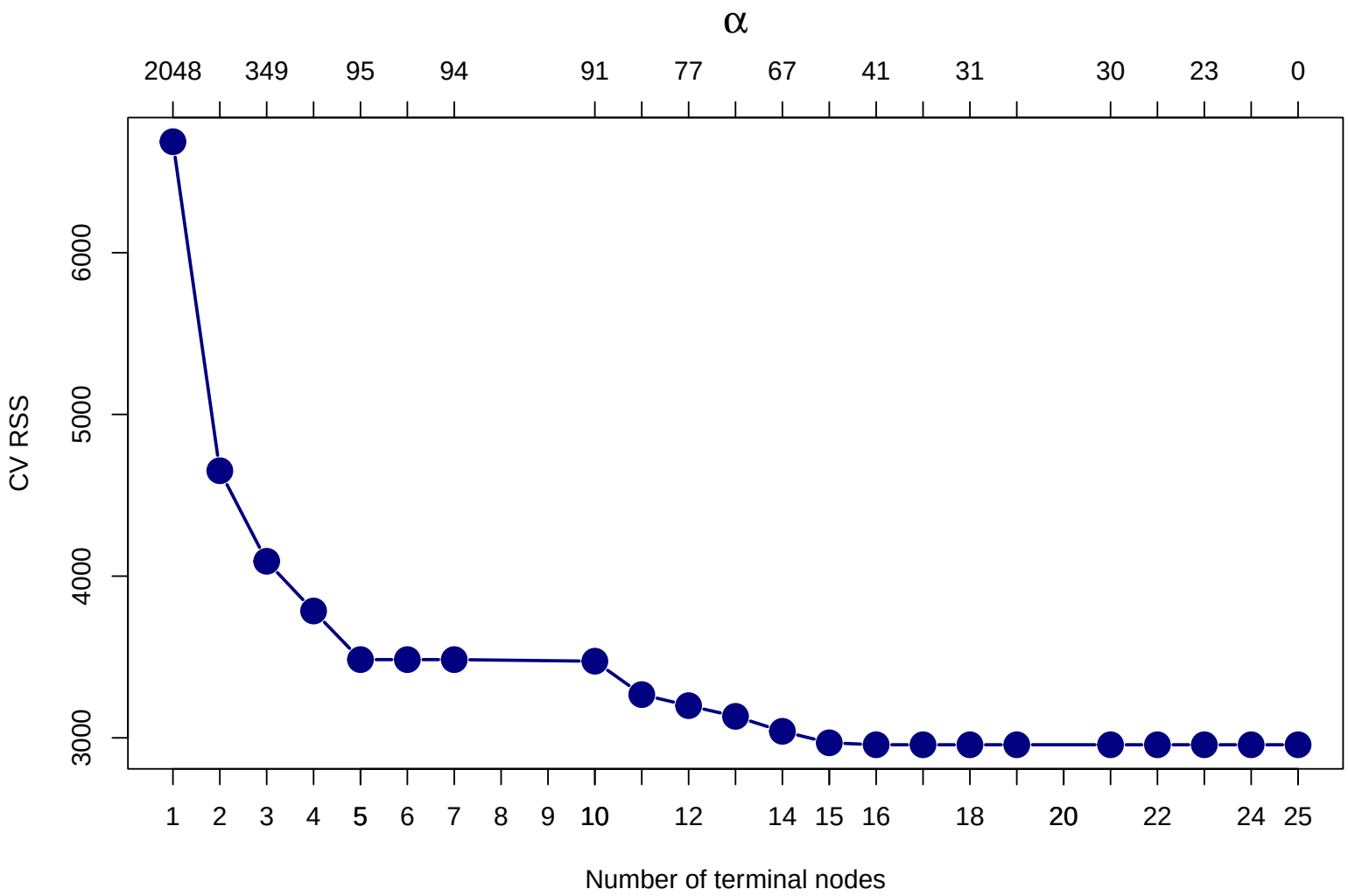
- By choosing α, we are effectively selecting a subtree; that is, a more parsimonious model that does not overfit the sample data.
- How do we choose α?
- Cross-validation, of course!
- We want the subtree (and hence α value) that has the lowest test error and therefore has the best predictive performance on unseen data.

Cross-Validation

The cross-validation procedure is as follows:

1. Using the full dataset, determine the critical values of $\alpha \geq 0$ that produce nested subtrees of different sizes.
2. Compute the test error associated with each of these α values using K-fold cross-validation.
3. Choose the α value with the lowest test error and report the corresponding subtree for the full dataset.

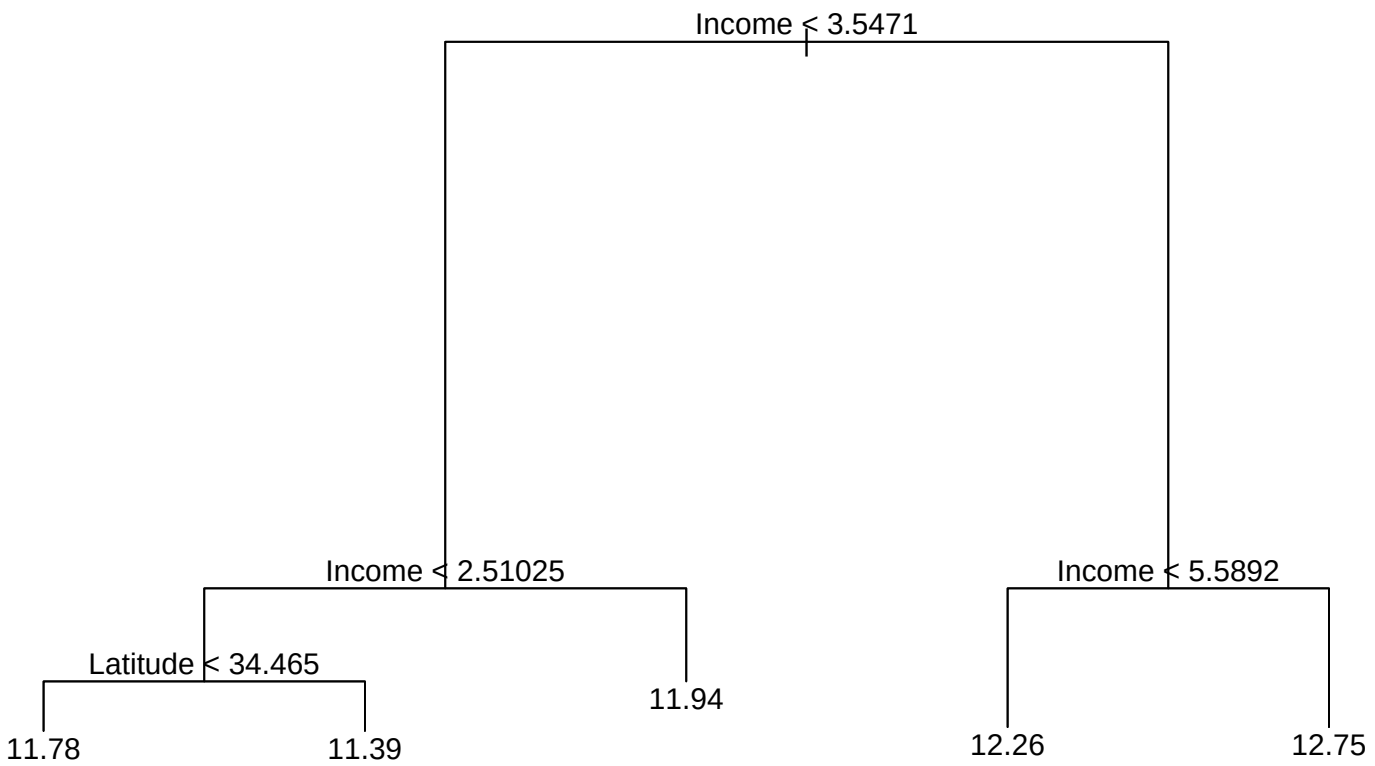
Cross-Validation



Reaches approx min at 15 nodes. Compare the CV error at 5 and 15 leaves. Is the additional complexity of the 15 node model justified?

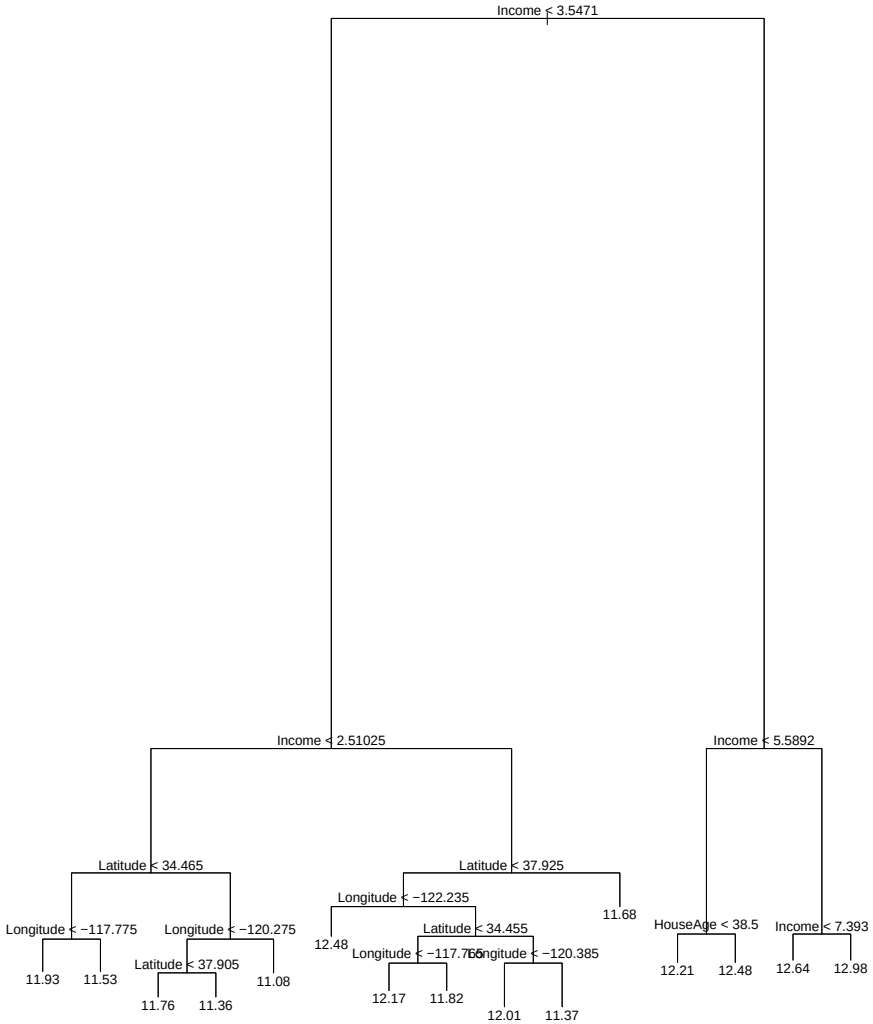
Pruning – 5 nodes

Pruning to 5 leaves yields a tree only using Income and Latitude. This is perhaps overly simplistic.



Pruning – 15 nodes

Trees are more useful in their interpretation than predictive power



We will return to prediction, but for now we will move on to classification

Classification Trees

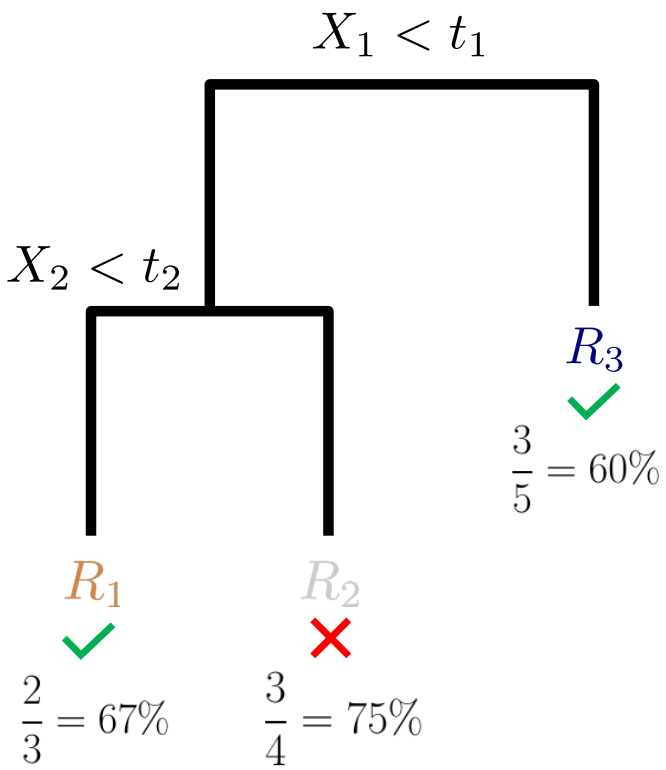
Classification Trees

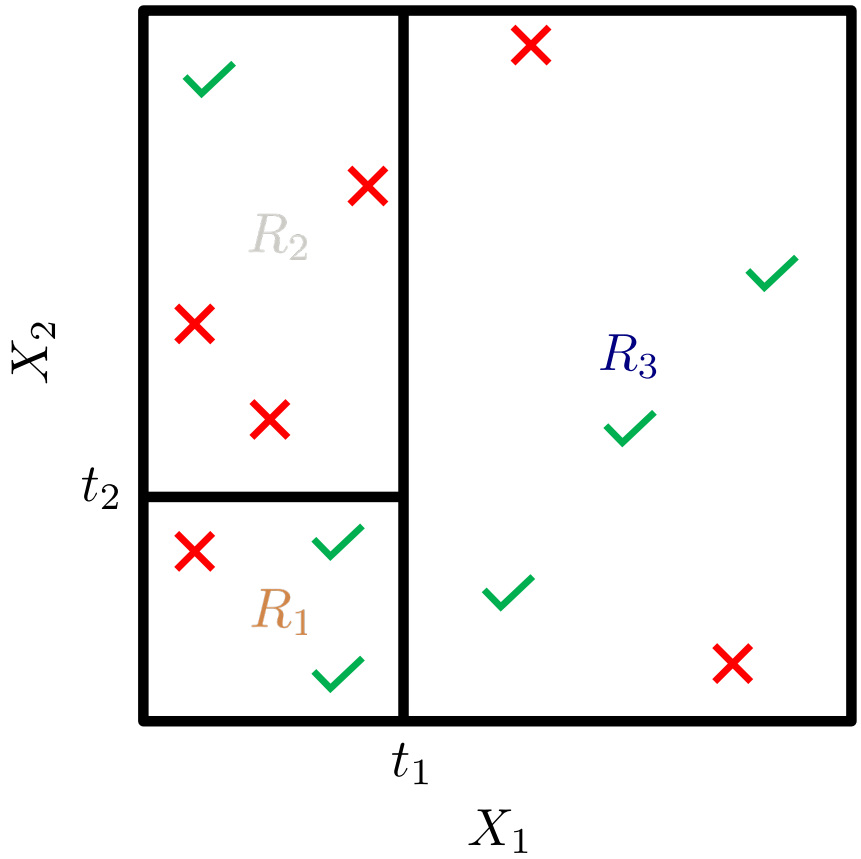
- Classification trees are similar to regression trees, except that they are used to predict a categorical response $Y \in \{1, 2, \ldots, K\}$
- The predicted value of an observation with $X \in R_m$ will now be the most **commonly occurring class** in R_m
- The **class proportions** in each terminal node give us an indication of the reliability of the prediction:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i:x_i \in R_m} I(y_i = k), \tag{4}$$

where N_m denotes the number of observations in region R_m

Classification Trees





Splitting Criteria

- Classification trees are also grown by recursive binary splitting
- But we cannot use the RSS as a criterion for splitting when the response is categorical
- Instead, we could grow the tree by choosing the split that minimises the classification error at each step, i.e. minimise the proportion of incorrectly predicted observations
- In practice, however, splitting on the reduction in classification error does not produce good trees
- We will consider three splitting criteria, only two of which are implemented in the `tree` package

1. Gini Index

The Gini index is a measure of **node impurity** or variability within the leaf nodes:

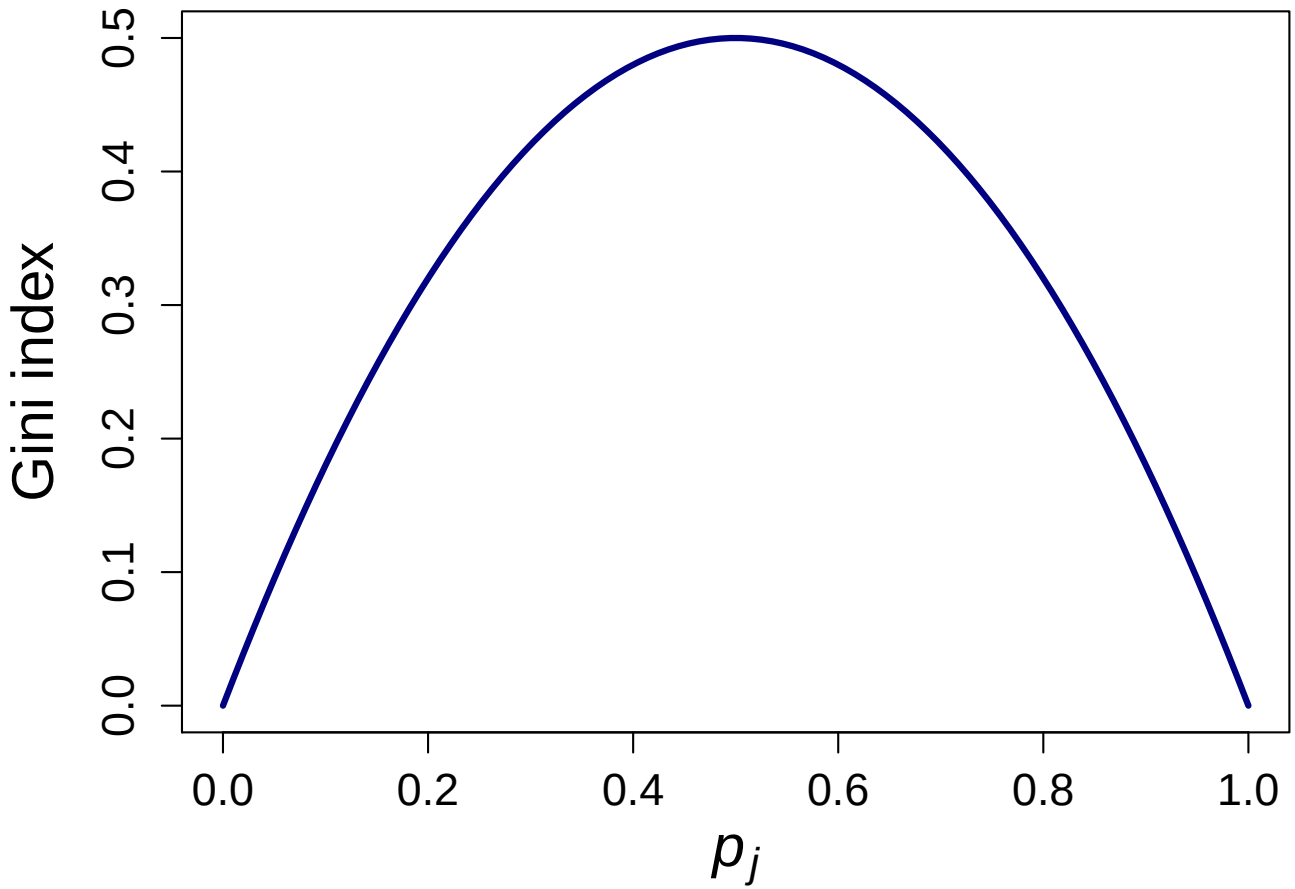
$$G = \sum_{j=1}^{J} G_j \qquad G_j = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$$

where $\hat{p}_{jk}$ is the proportion of observations in response category $k = 1, \ldots, K$ within leaf node $j = 1, \ldots, J$

- We would like the leaf nodes to be as homogeneous or "pure" as possible; ideally, each leaf node would include observations of only one response category
- At each step during tree growth, we therefore choose the split that produces the greatest reduction in the Gini index

1. Gini Index

For a binary response, $G_j = 2\hat{p}_j(1 - \hat{p}_j)$



The Gini index is minimised when leaf node j is homogeneous

2. Entropy

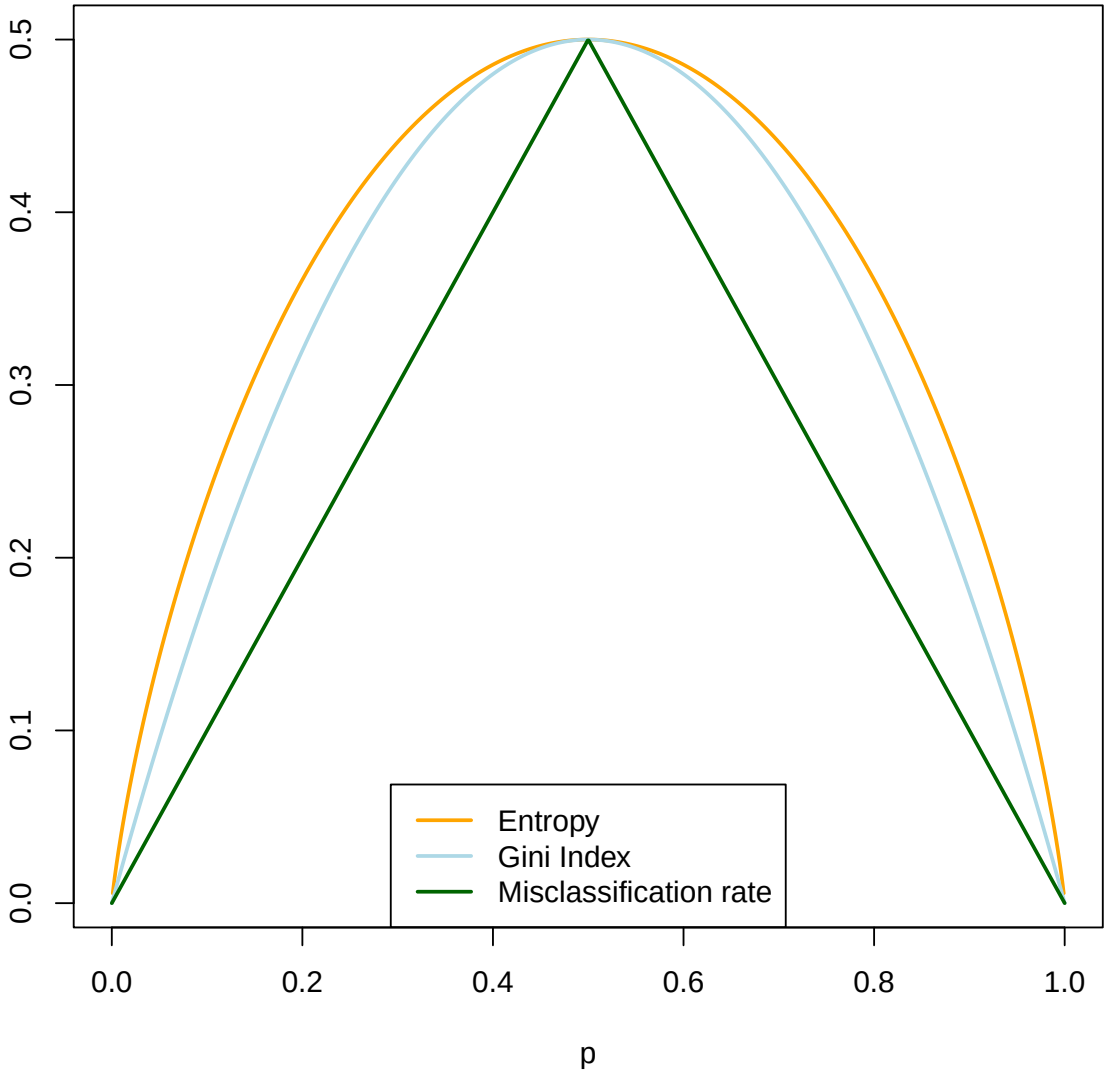
An alternative way to determine node impurity is to calculate the Shannon entropy of each terminal node, and again sum across all J leaves:

$$H = -\sum_{j=1}^{J}\sum_{k=1}^{K} \hat{p}_{jk} \log(\hat{p}_{jk})$$

Note that $\log_2$ is also often used in the definition, depending on the context. This yields units of "bits", whereas the above yields "nats".

- Since $0 \leq \hat{p}_{jk} \leq 1$, it follows that $0 \leq -\hat{p}_{jk} \log(\hat{p}_{jk})$.
- Although stemming from information theory, computationally this is similar to the Gini index.
- Therefore, they will produce similar results when used as a splitting criterion.
- We will be implementing the Gini index in R, but not entropy.

Gini vs Entropy



3. Deviance

- The deviance is constructed by viewing a classification tree as a probability model.
- Let $\boldsymbol{Y}_j$ denote the set of categorical responses in leaf node j.
- $\boldsymbol{Y}_j$ is a random sample of size n_j from the multinomial distribution:

$$p(\boldsymbol{y}_j) = \binom{n_j}{n_{j1} \cdots n_{jK}} \prod_{k=1}^{K} p_{jk}^{n_{jk}}$$

- The likelihood over all J leaf nodes is therefore

$$L = \prod_{j=1}^{J} p(\boldsymbol{y}_j) \propto \prod_{j=1}^{J} \prod_{k=1}^{K} p_{jk}^{n_{jk}}$$

3. Deviance

The deviance is defined as

$$D = -2 \log L = -2 \sum_{j=1}^{J} \sum_{k=1}^{K} n_{jk} \log p_{jk}$$

- We want the model that makes the data most probable, i.e. has the highest likelihood.
- This is equivalent to minimising the deviance.
- When we grow a classification tree, we therefore choose the split that **reduces the deviance** by the most at each step.
- Note that the deviance takes the number of observations in each terminal node into account, unlike Gini/entropy.

3. Deviance

Returning to regression trees for a moment ...

- Suppose that $\boldsymbol{Y}_j$ is a random sample of size n_j from the $N(\mu_j, \sigma^2)$ distribution at leaf node j, then

$$p(\boldsymbol{y}_j) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^{n_j} \exp\left(\frac{-\sum_{i:x_i \in R_j}(y_i - \mu_j)^2}{2\sigma^2}\right)$$

- The likelihood of the means over all J leaf nodes is then

$$L = \prod_{j=1}^{J} p(\boldsymbol{y}_j) \propto \exp\left(-\frac{1}{2}\sum_{j=1}^{J}\sum_{i:x_i \in R_j}(y_i - \mu_j)^2\right)$$

- We therefore have that

$$D = -2\log(L) \propto \sum_{j=1}^{J}\sum_{i:x_i \in R_j}(y_i - \mu_j)^2 = RSS$$

Classification Tree Pruning

- Like regression trees, we must prune a fully grown classification tree to avoid overfitting.
- As before, we cannot use the RSS_α criterion for pruning when the response is categorical.
- We can use any of the splitting criteria above to define a cost complexity criterion for a classification tree T:

$$C_\alpha(T) = C(T) + \alpha|T|$$

- However, if prediction accuracy of the final pruned tree is the goal, then $C(T)$ should be taken as the classification error rate of tree T.

Example 7 – Titanic

- We will look at the Titanic dataset from Kaggle.
- The dataset contains information on 891 passengers aboard the Titanic.
- Our goal is to predict survival (categorical: yes/no) using the following potential predictors:

  Passenger class; Sex; Age; Number of siblings/spouses aboard; Number of parents/children aboard; Fare; Port of embarkation.

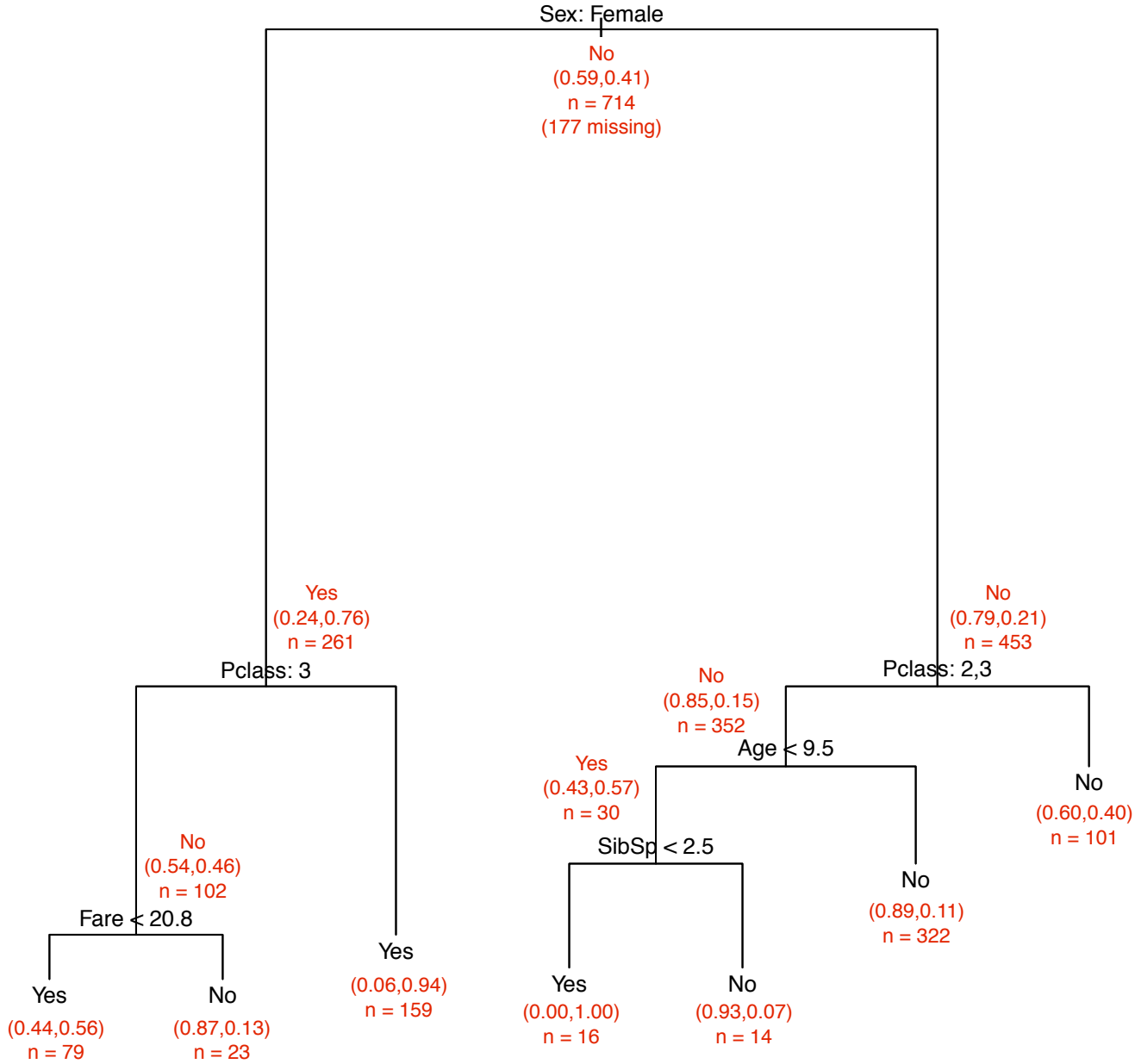
Example 7 – Titanic

```
> titanic <- read.csv('titanic.csv'T)
> head(titanic, 10)
```

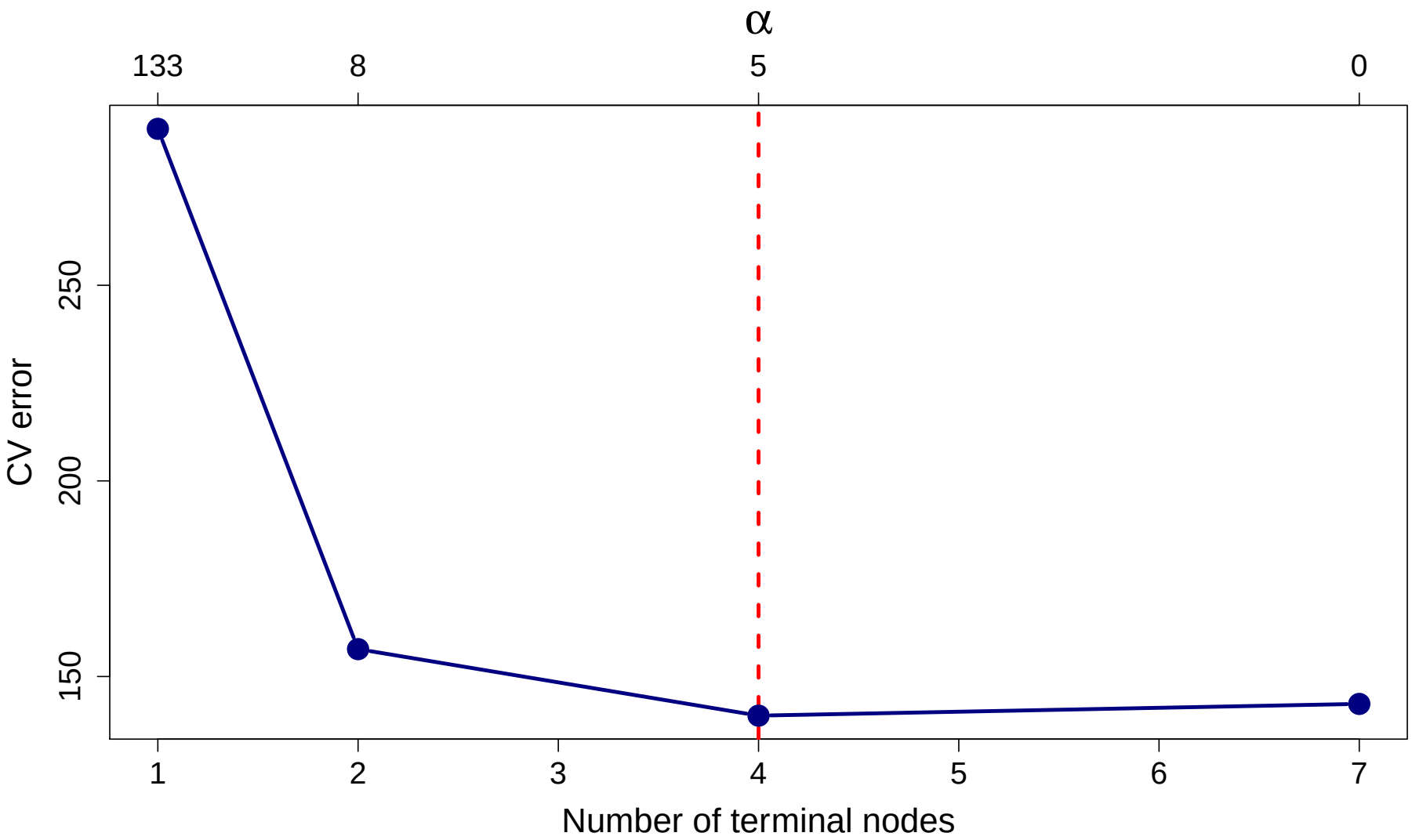
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	No	3	Male	22	1	0	7.25	S
2	Yes	1	Female	38	1	0	71.28	C
3	Yes	3	Female	26	0	0	7.93	S
4	Yes	1	Female	35	1	0	53.10	S
5	No	3	Male	35	0	0	8.05	S
6	No	3	Male	NA	0	0	8.46	Q
7	No	1	Male	54	0	0	51.86	S
8	No	3	Male	2	3	1	21.08	S
9	Yes	3	Female	27	0	2	11.13	S
10	Yes	2	Female	14	1	0	30.07	C

```
> # Embarked: C = Cherbourg; Q = Queenstown; S = Southampton
```

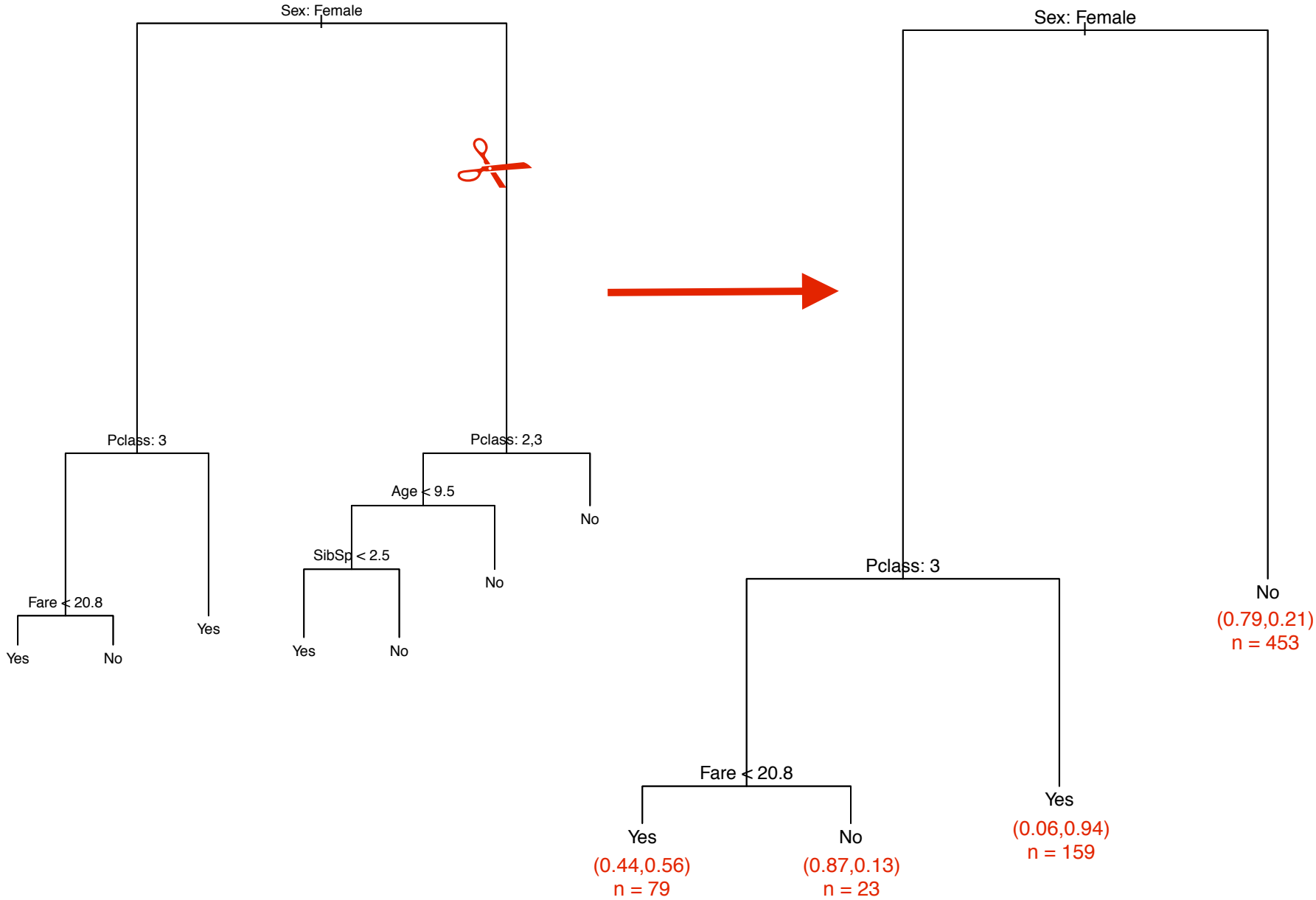
Example 7 – Titanic



Example 7 – Titanic



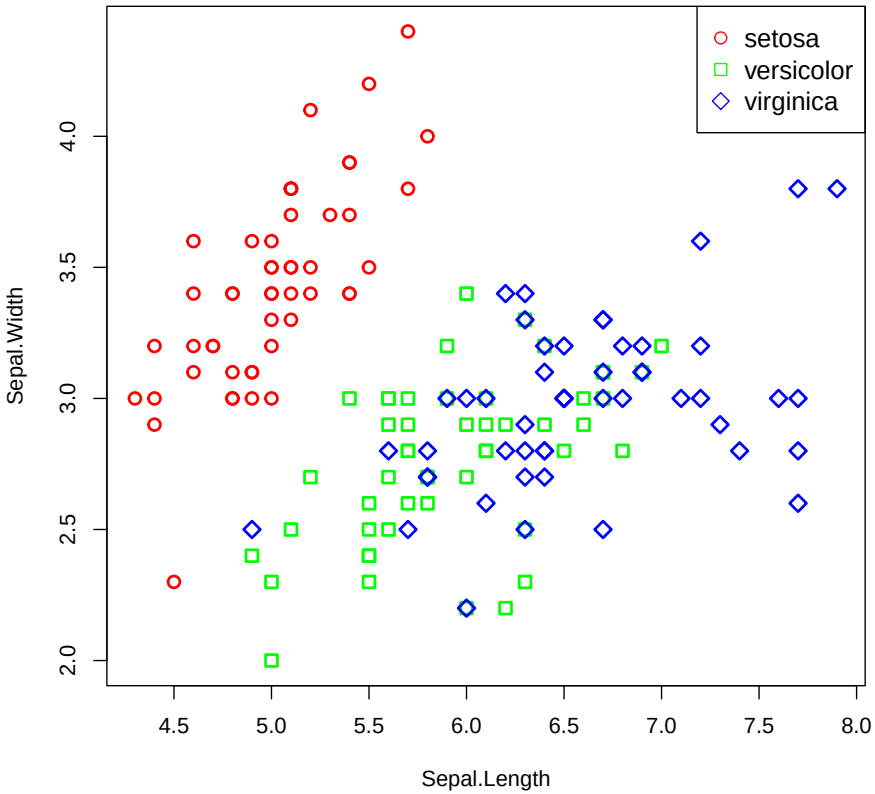
Example 7 – Titanic

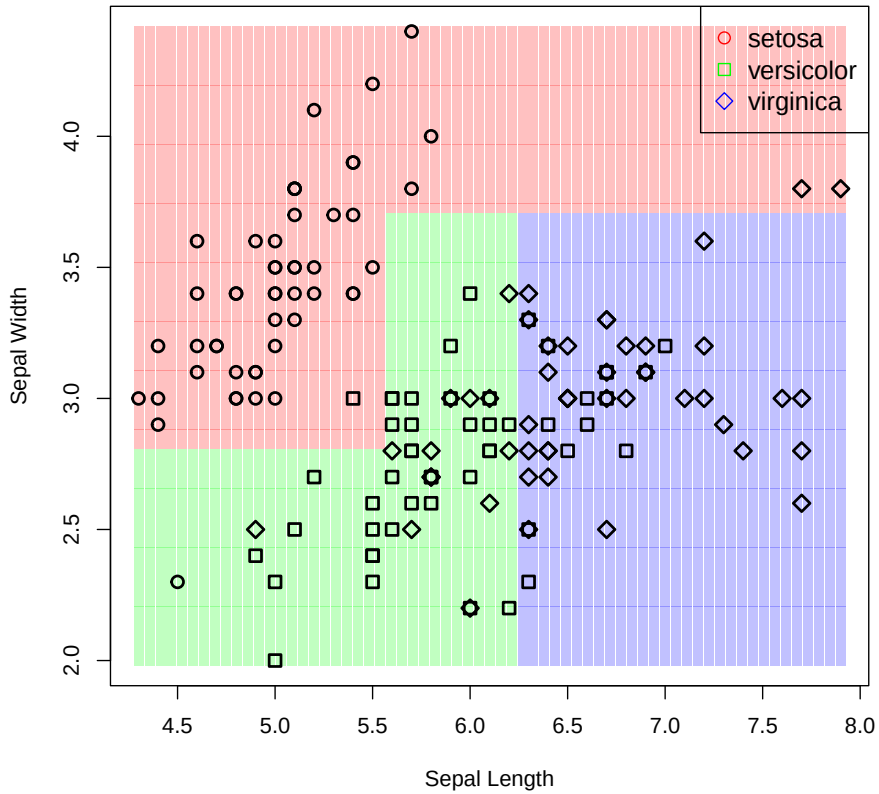


Multiclass classification

- We see that classification trees handle categorical predictors with ease, without the need to code dummy variables
- Another benefit of this approach is that we can just as easily model target variables with multiple classes
- To illustrate this, consider the famous Iris dataset

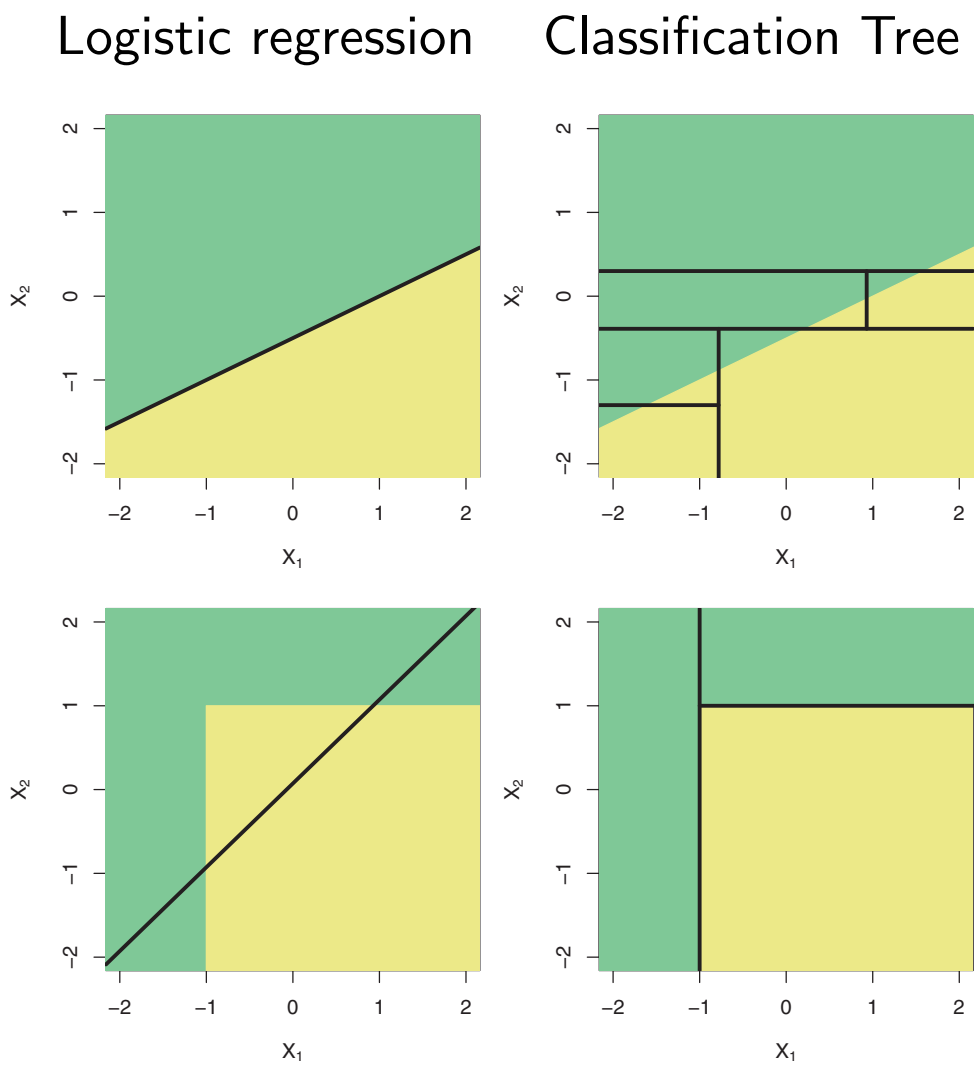
Example 8 – Iris





Trees versus Linear Models

- We could use either logistic regression or decision trees for classification.
- Which model is better?
- It depends on the problem!
- Cross-validation can be used to choose a prediction method.



Advantages and Disadvantages of Trees

- Very easy to explain and interpret – easier than linear regression!
- Mirrors human decision making.
- Easy to handle qualitative predictors without the need to create dummy variables.
- The predictive accuracy of trees is not as good as some other regression and classification approaches.

Next we will consider techniques for improving the predictive performance of decision trees.