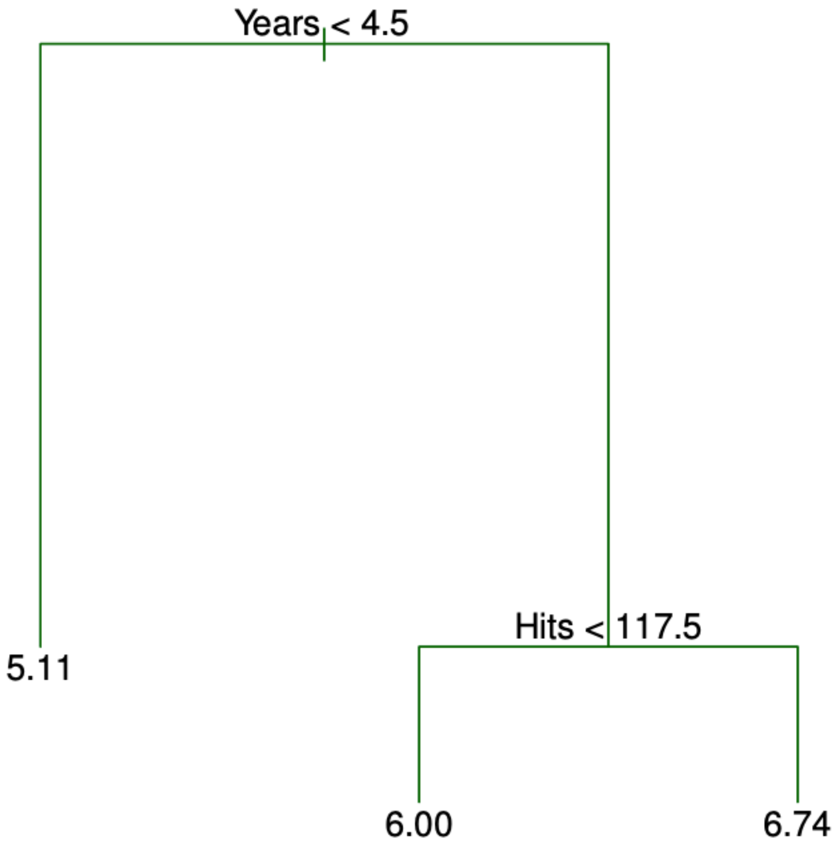
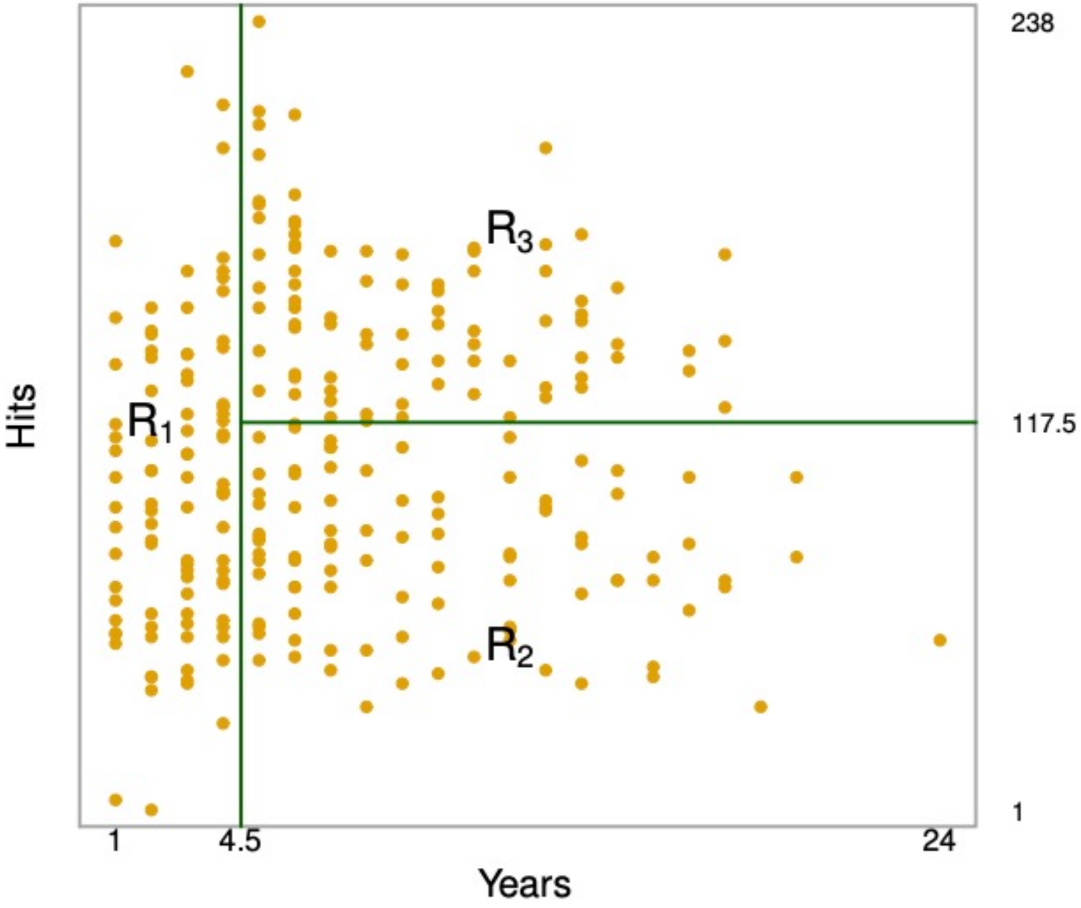
Tree-based Methods

Tianhang Zheng

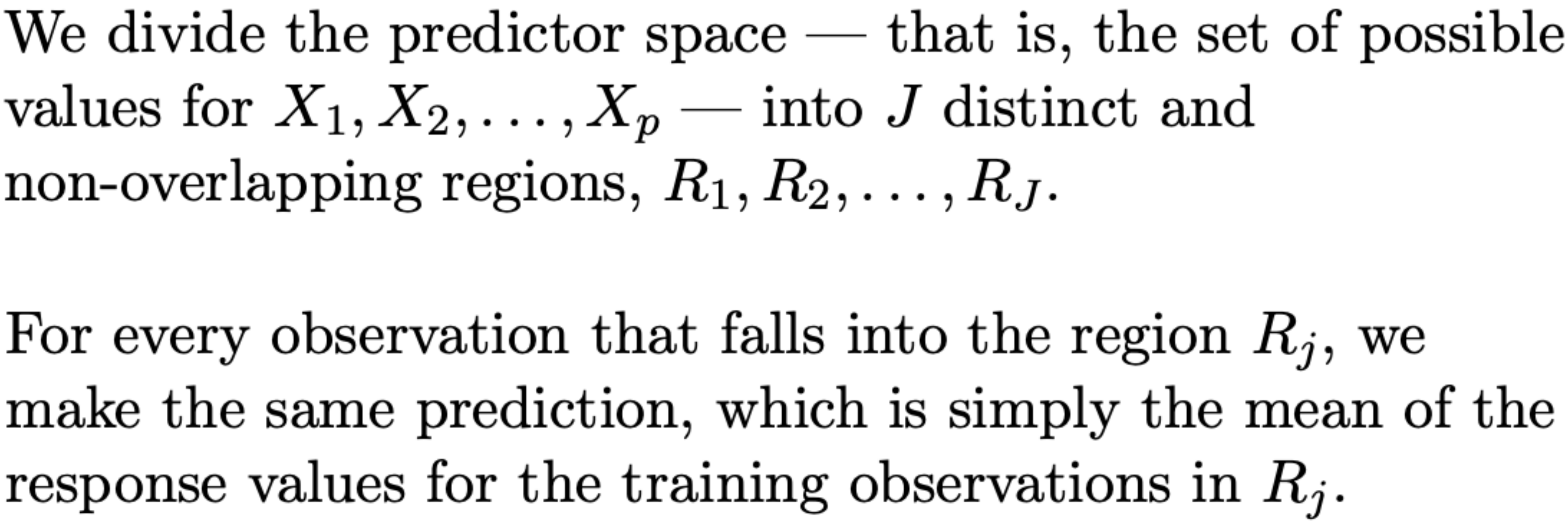
https://tianzheng4.github.io

Interpretation of Decision Tree





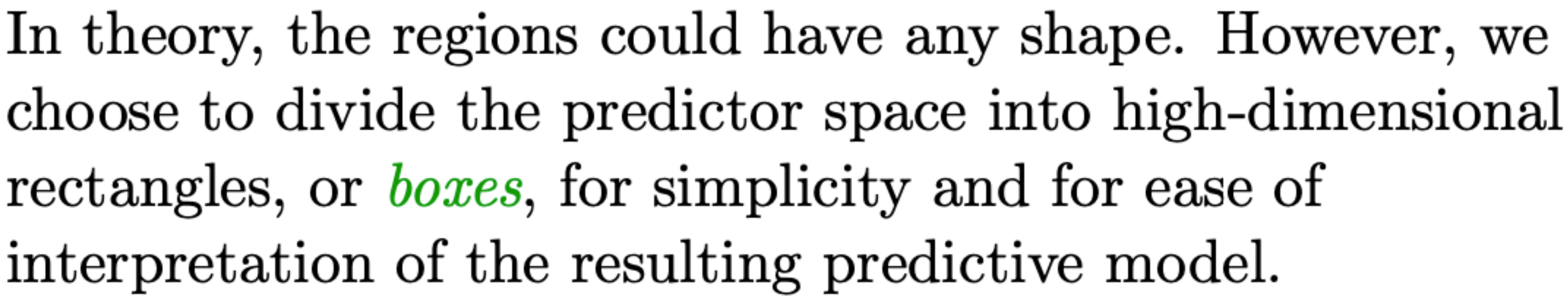
Regression Tree-Building Process



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, $R_1, R_2, \ldots, R_J$.

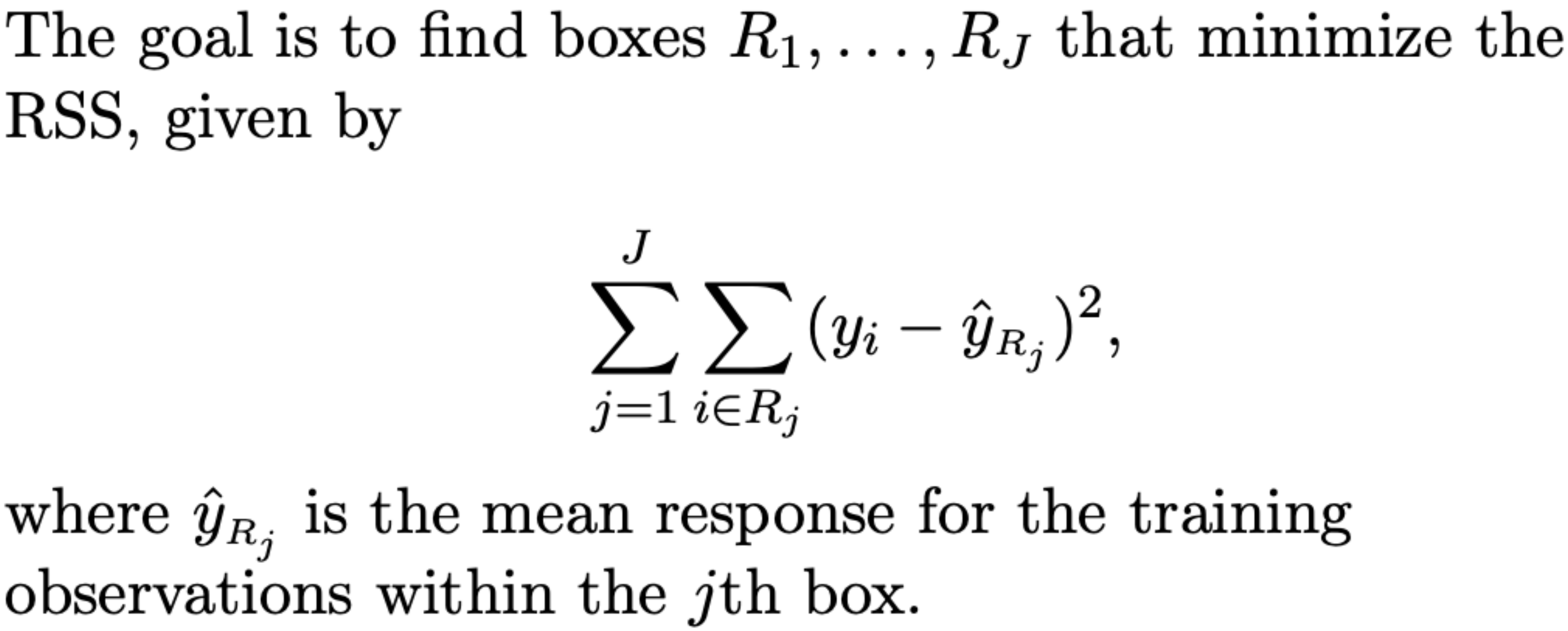
For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Regression Tree-Building Process



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

Regression Tree-Building Process

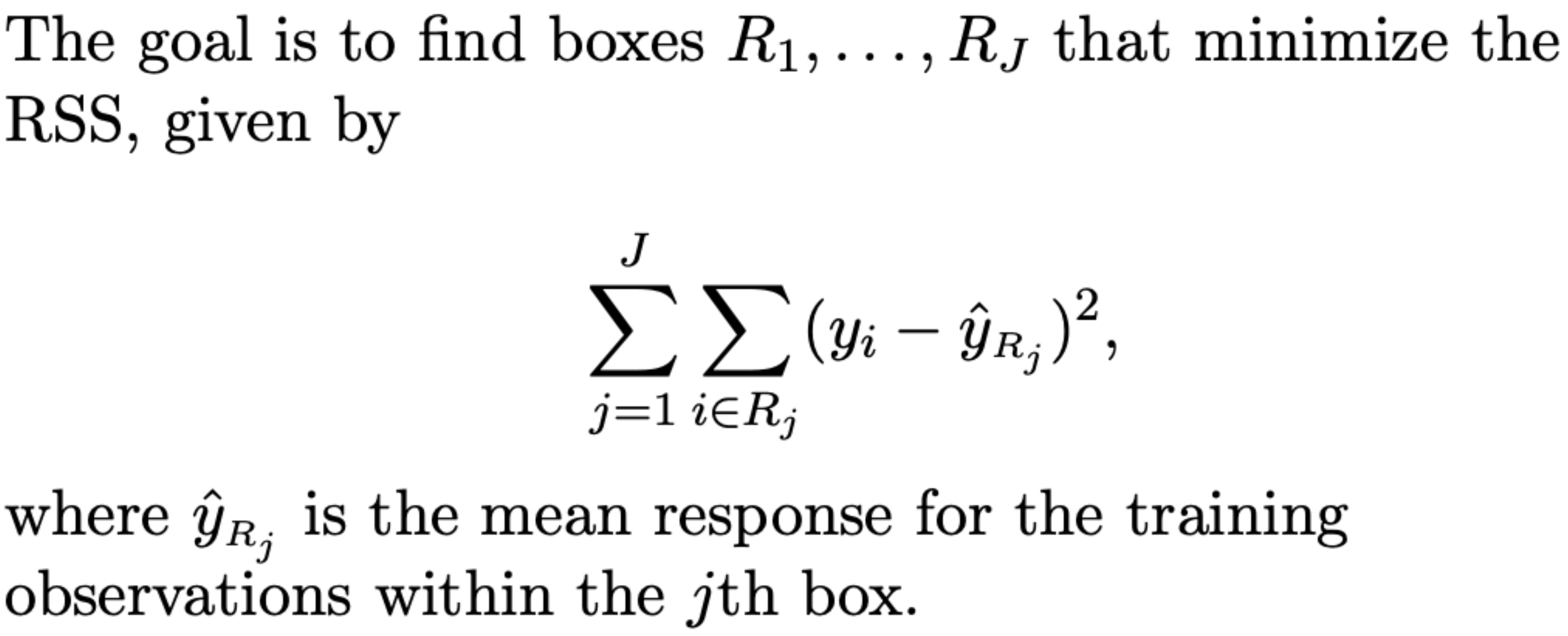


The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

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

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

Regression Tree-Building Process

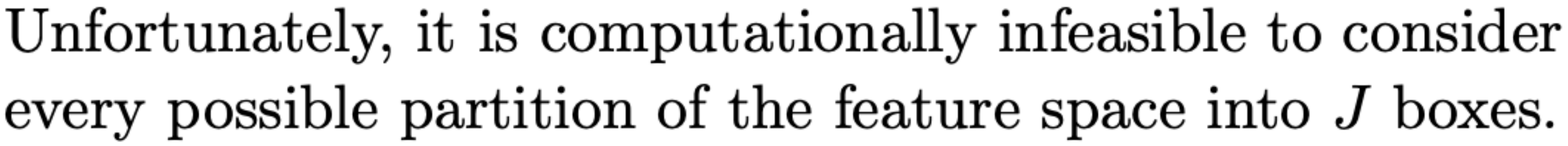


The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

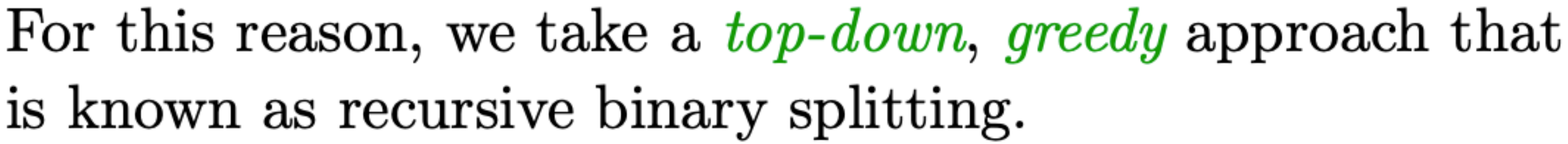
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

Regression Tree-Building Process

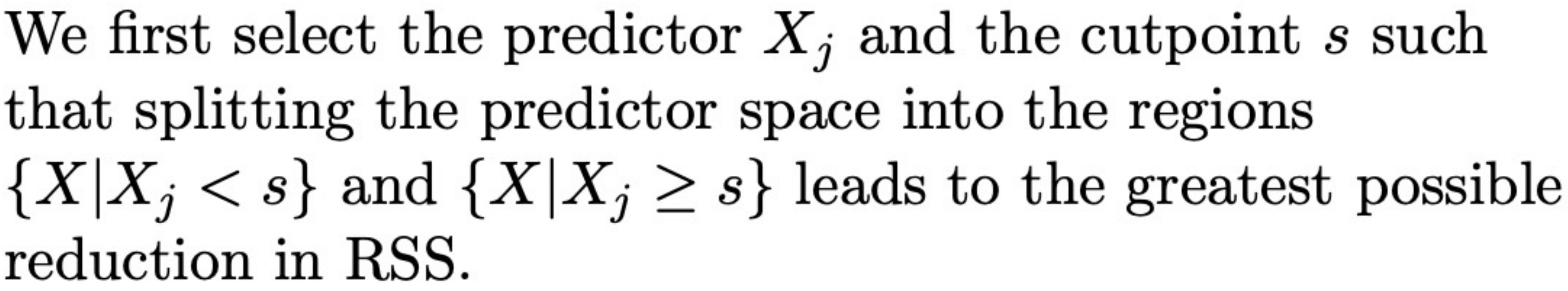


Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.



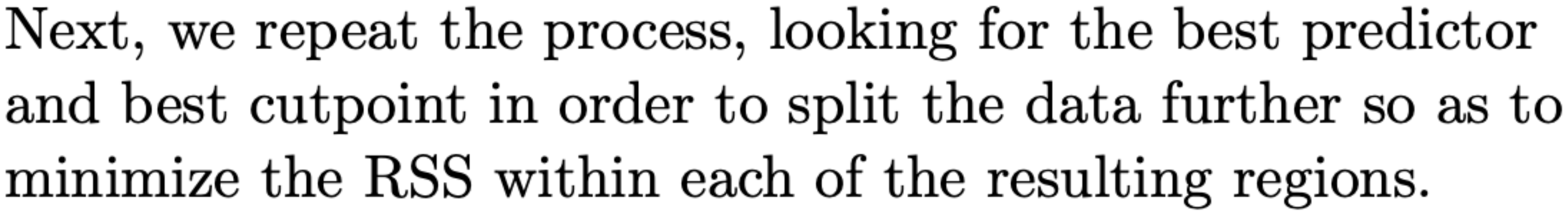
For this reason, we take a *top-down*, *greedy* approach that is known as recursive binary splitting.

Greedy Method

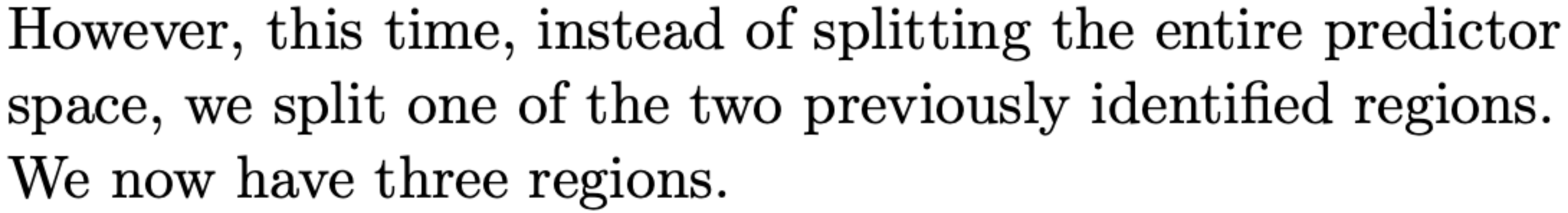


We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.

Greedy Method

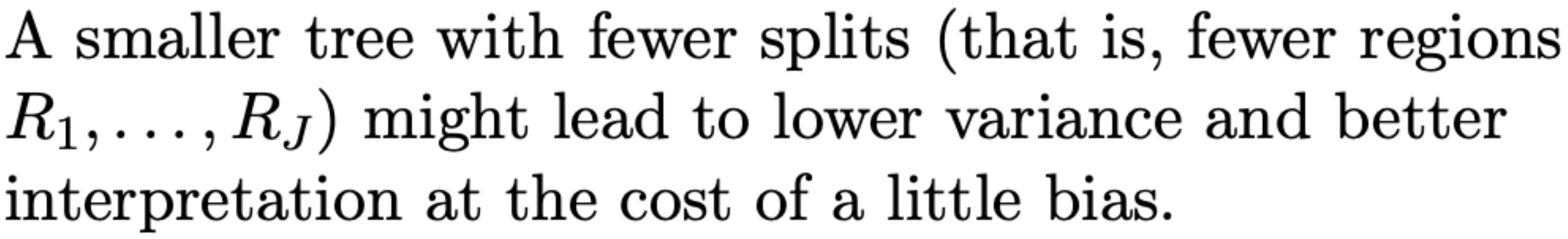


Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.

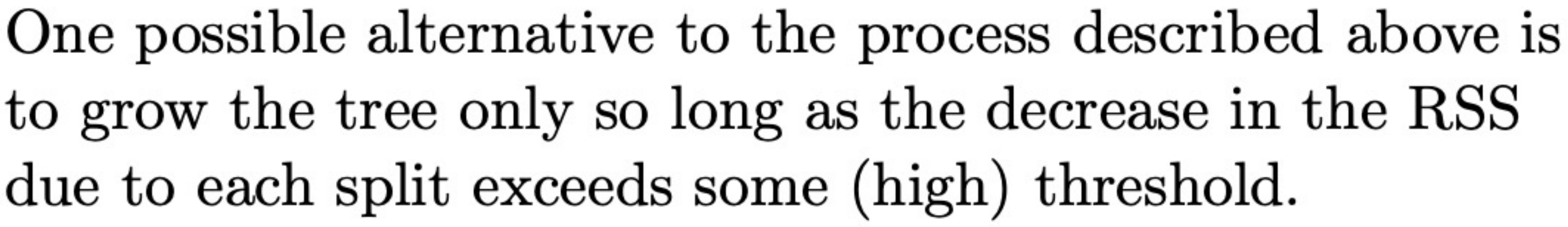


However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.

Tree Pruning

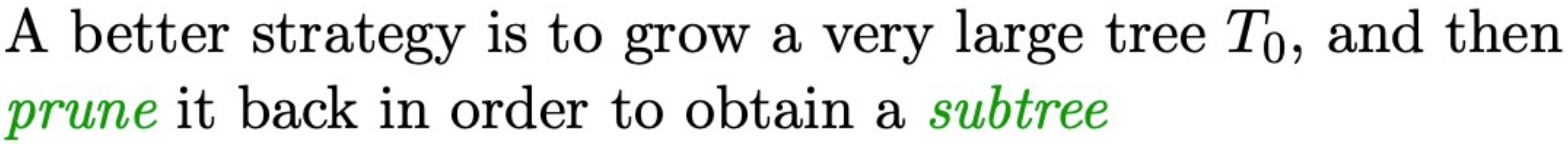


A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.

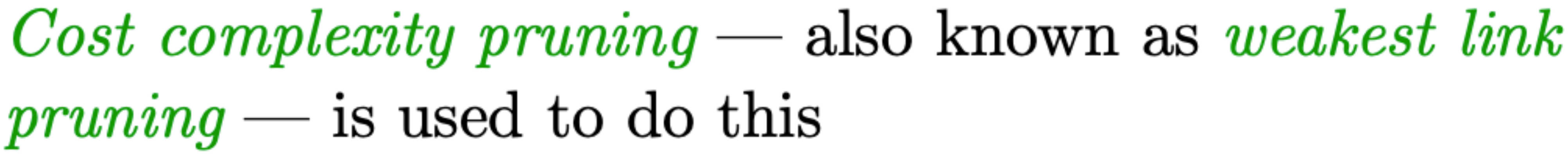


One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.

Tree Pruning



A better strategy is to grow a very large tree T_0, and then *prune* it back in order to obtain a *subtree*



Cost complexity pruning — also known as *weakest link pruning* — is used to do this

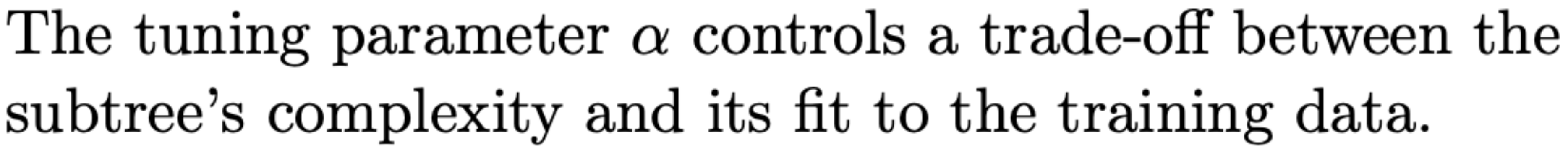
Cost Complexity Pruning

The objective is

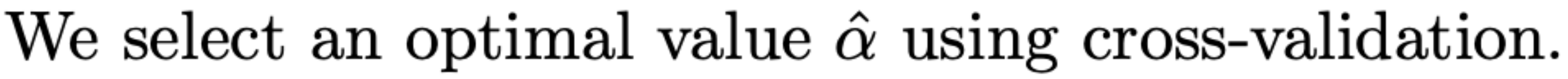
$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Here |T| indicates the number of terminal nodes of the tree T, R_m is the rectangle corresponding to the m-th terminal node, and $\hat{y}_{R_m}$is the mean of the training observations in R_m

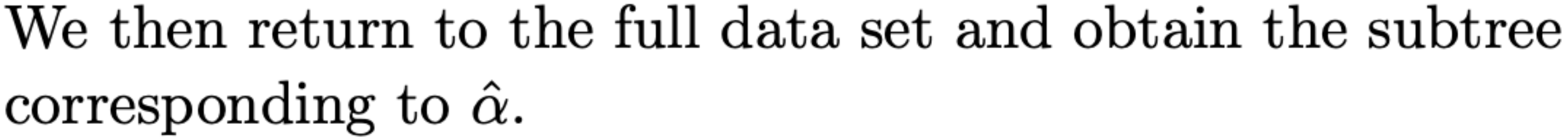
Cost Complexity Pruning



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

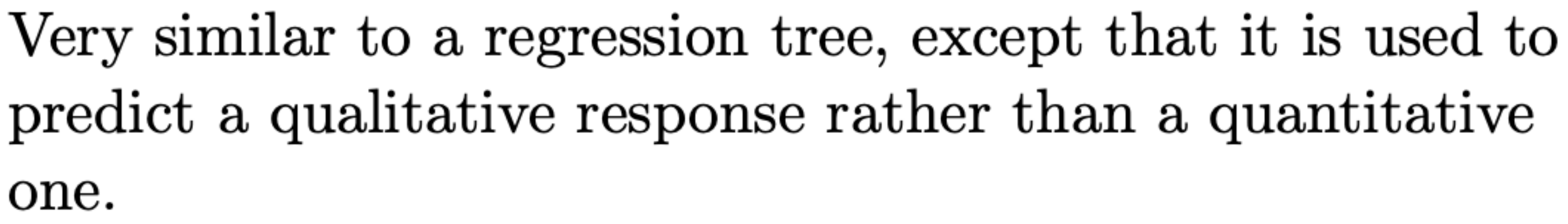


We select an optimal value $\hat{\alpha}$ using cross-validation.

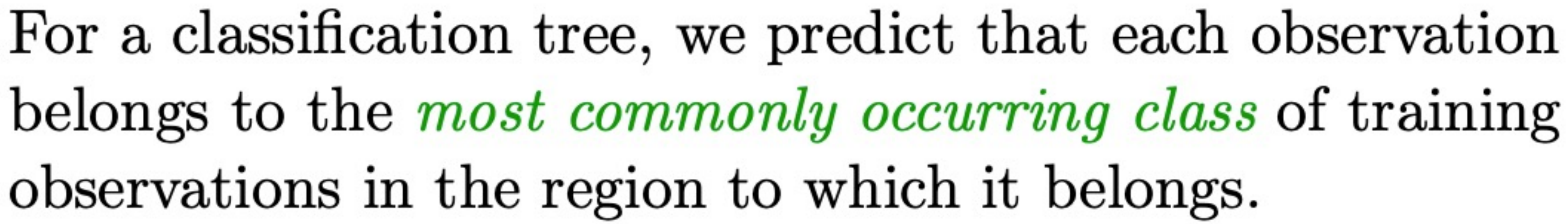


We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

Classification Tree



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.

Classification Tree-Building Process

If a target is a classification outcome taking on values 0,1,…,K-1

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

The Gini index is defined by

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

Classification Tree-Building Process

If a target is a classification outcome taking on values 0,1,…,K-1

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

The Gini index is defined by

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

Gini Index

Gini index is a measure of total variance across the K classes. The Gini index takes on a small value if all $\hat{p}_{mk}$ 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.

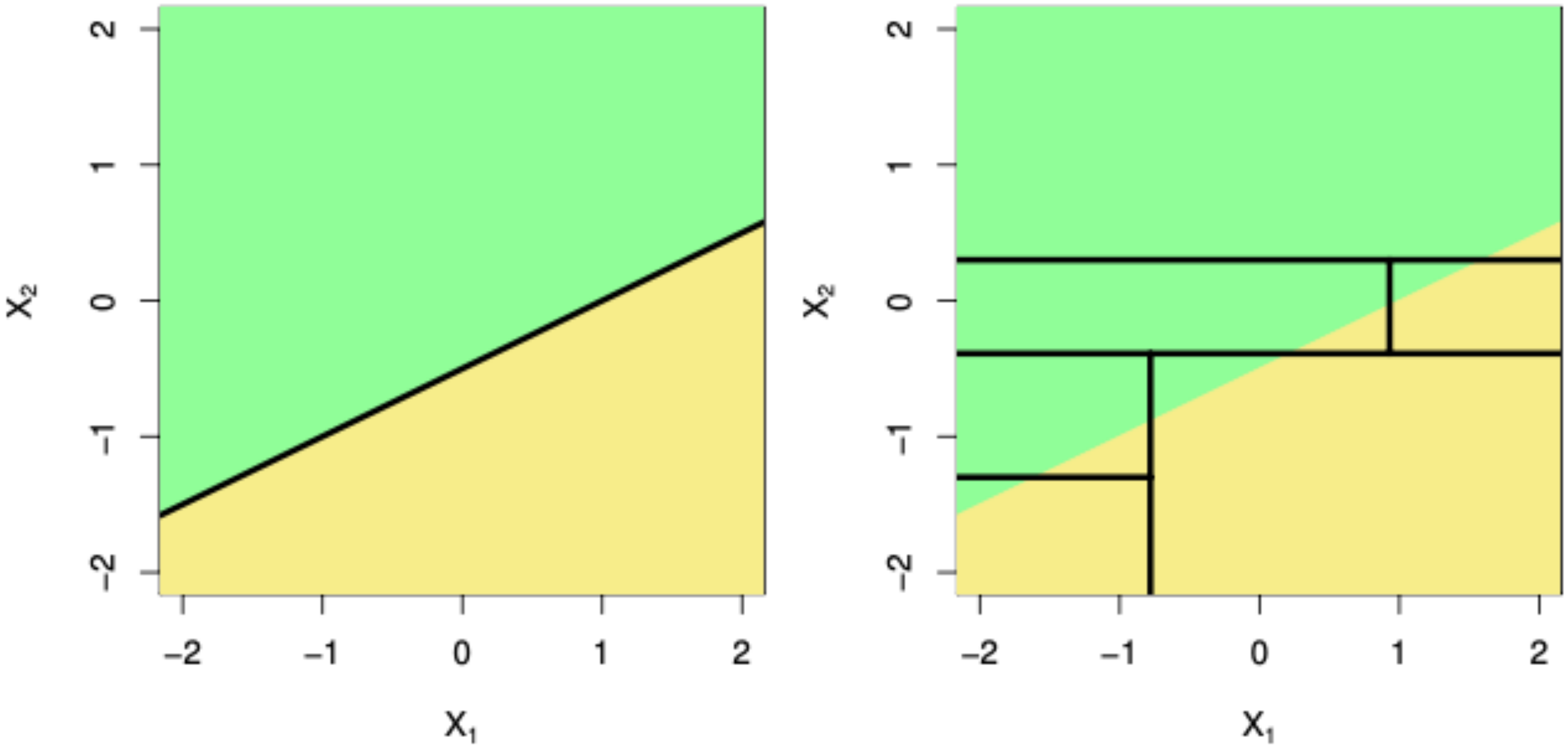
Cross-Entropy

An alternative to the Gini index is cross-entropy, given by

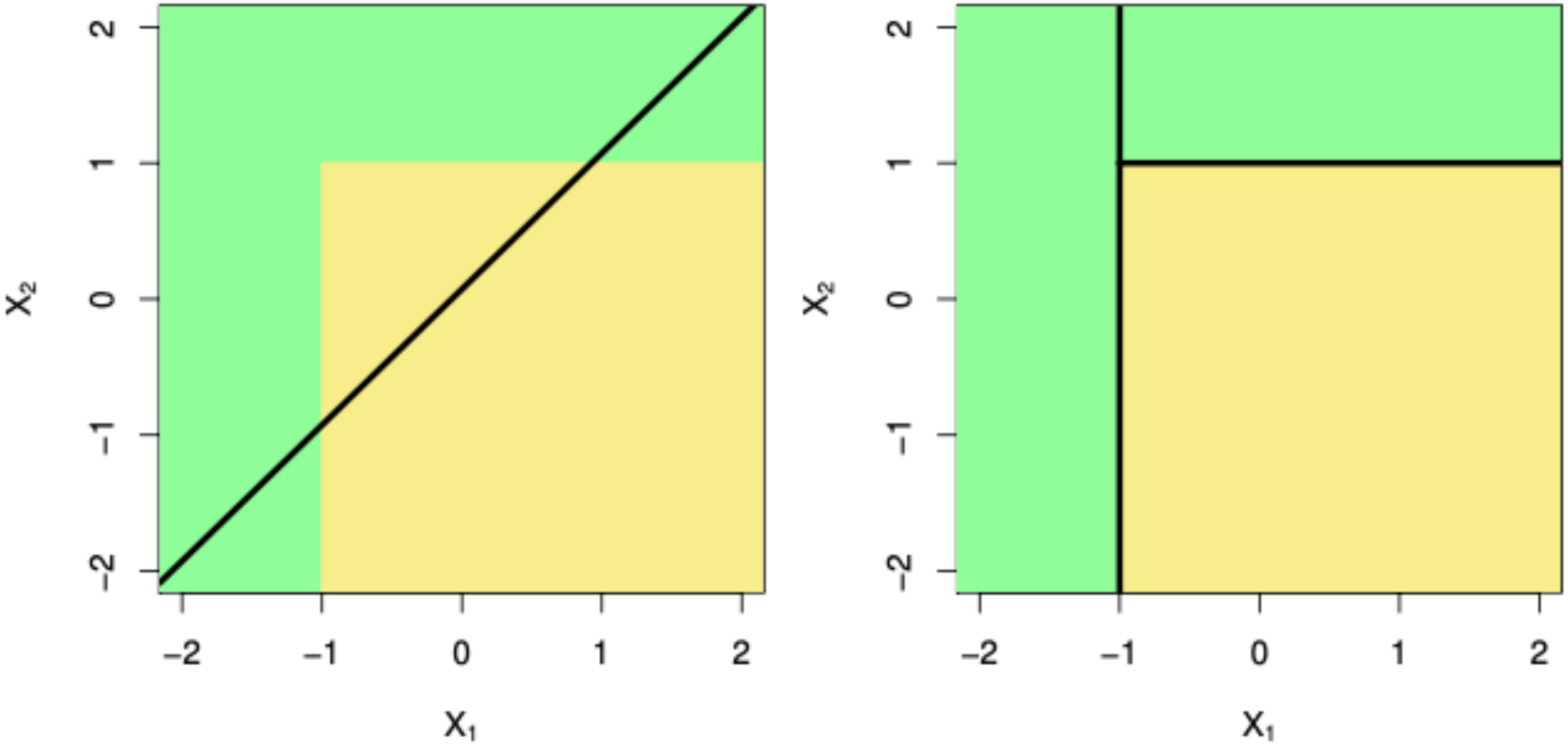
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Cross-entropy is also a commonly-used loss function for deep learning

Tree vs Linear Model



Tree vs Linear Model



Discussion about Tree-based Methods

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

Trees can easily handle qualitative predictors without the need to create dummy variables.

Bagging

Averaging a set of observations reduces variance. But this is not practical because we do not have access to multiple training sets.

Instead, we can bootstrap, by taking repeated samples from the (single) training data set.

Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method.

Bagging

For bagging, we generate B different bootstrapped training data sets. We then train our method on the b-th bootstrapped training set in order to get $f^{b}(x)$, the prediction at a point x. We then average all the predictions to obtain

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

Bagging Classification Trees

For classification trees: for each test observation, we record the class predicted by each of the B trees

Take a majority vote: the overall prediction is the most commonly occurring class among the B predictions.

But the trees are highly correlated and can increase variance

Random Forests

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. This reduces the variance when we average the trees.

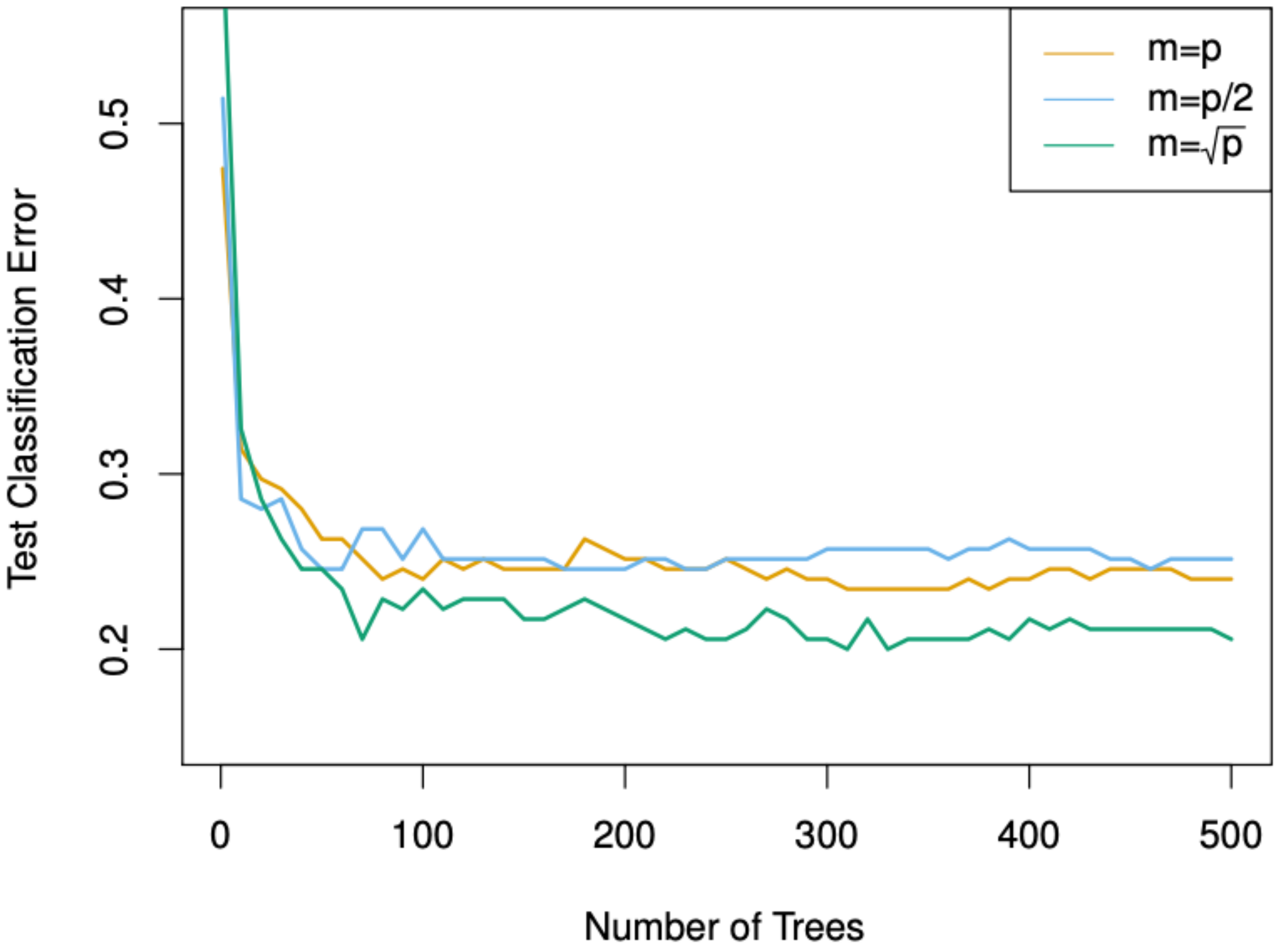
Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. This reduces the variance when we average the trees.

Random Forests

When building these decision trees, each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.

The reason: if one or a few features are very strong predictors for the target output, these features will be selected in many of the B trees, causing them (bagging trees) to become correlated.

Random Forests (How to Select m)

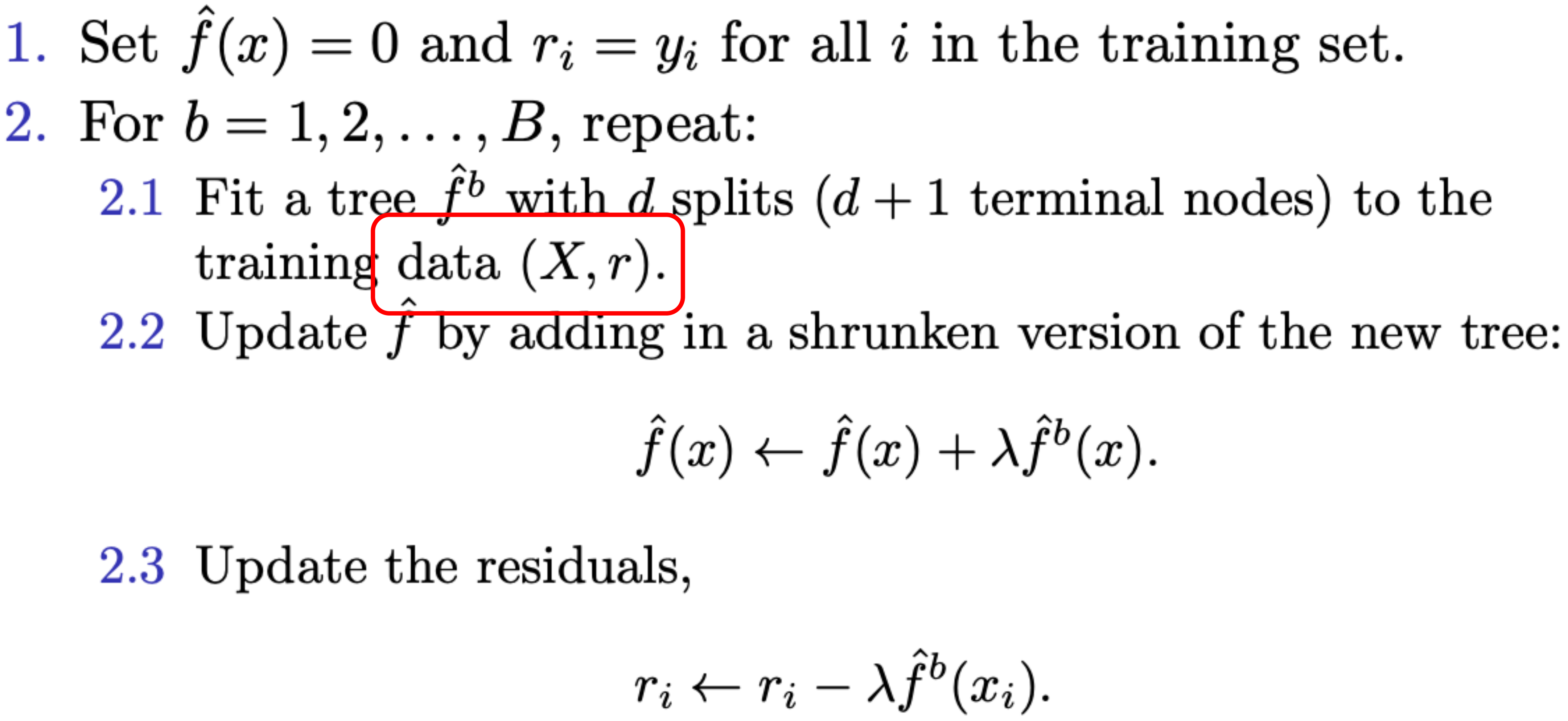


Boosting Trees

Boosting works in a similar way as bagging, except that the trees are grown sequentially: each tree is grown using information from previously grown trees.

Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.

Boosting Regression Trees



1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:

   2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).

   2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

   2.3 Update the residuals,

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

Gradient Boosting

A machine learning algorithm used in regression and classification tasks that gives a prediction model in the form of an ensemble of weak prediction models.

Can outperform random forest but more complicated:
https://en.wikipedia.org/wiki/Gradient_boosting

Hyperparameters for Boosting

The number of trees B: Too large B leads to overfitting (cross validation)

The shrinkage parameter λ: Typically 0.01 or 0.001

The number of splits d in each tree: Often d = 1 works well, in which case each tree is a stump

Q & A