STAT430: Machine Learning for Financial Data

LarryHua.com/teaching
Spring 2019

Ensemble methods

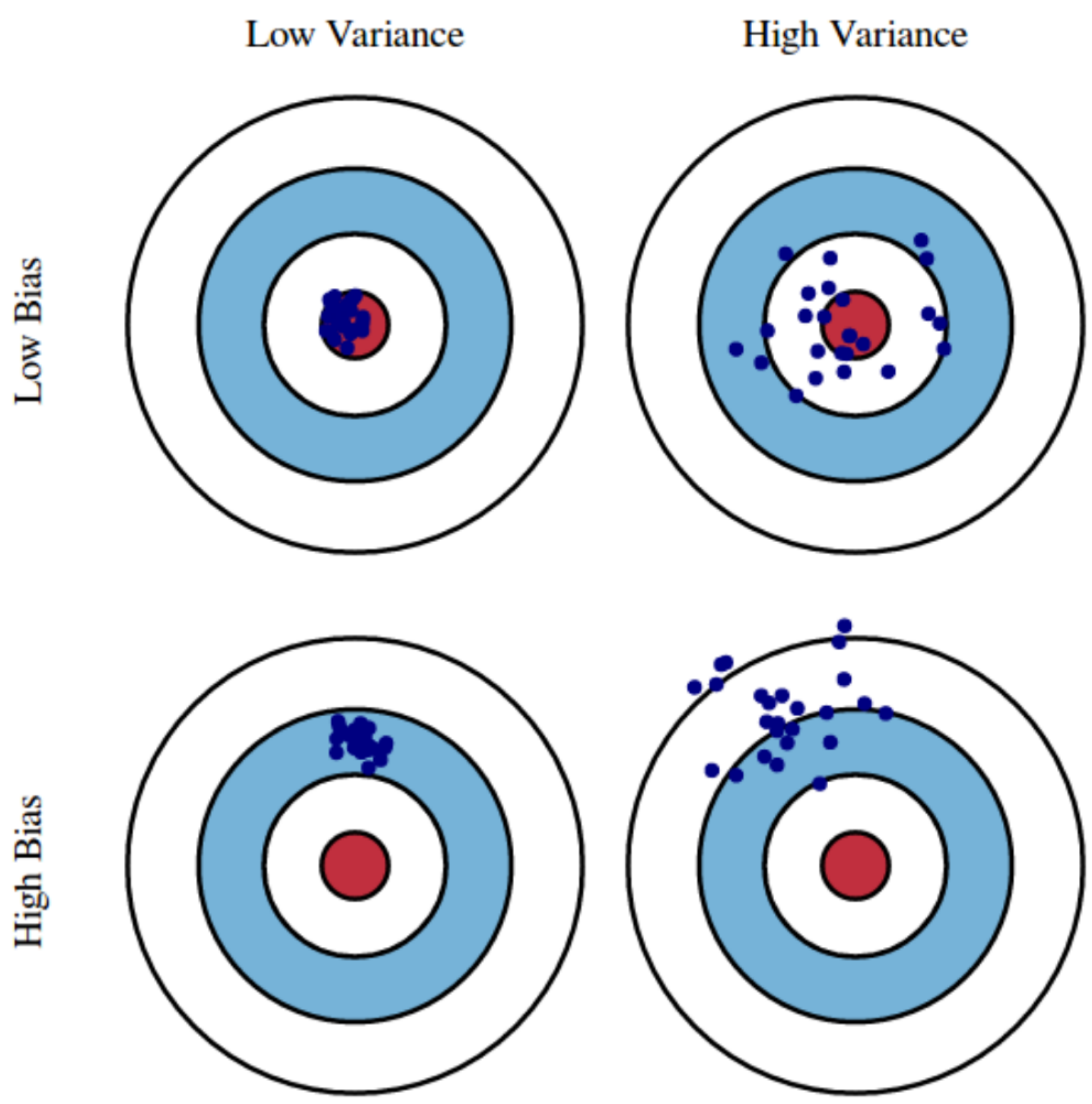
Variance and bias

- Estimate f: $y = f(x) + \epsilon$ with $E(\epsilon) = 0, V(\epsilon) = \sigma_\epsilon^2$
- Mean square error: ($y, \hat{f}(x), \epsilon$ are random variables)

$$\begin{aligned} E((y - \hat{f}(x))^2) &= (E(f(x) - \hat{f}(x)))^2 \quad \text{bias} \\ &+ V(\hat{f}(x)) \quad \text{variance} \\ &+ \sigma_\epsilon^2 \quad \text{irriducible noise} \end{aligned}$$

- An ensemble method is a method that combines weak learners from the same learning algorithm to create a stronger learner.
- Ensemble methods help reduce bias and/or variance.

Variance and bias



Tree-based methods

1. Divide the predictor space into M distinct and non-overlapping regions R_m's
2. For every observation that falls into the region R_m, make the same prediction based on
   - mean of the response values in the same R_m for **regression**
   - majority votes for the same R_m for **classification**
3. Pros and cons
   - Easy to interpret
   - Not competitive with the best supervised learning approaches in terms of prediction accuracy
   - Ensemble methods such as bagging / random forests / boosting can dramatically improve performance

Terminology for trees

- Categorical response variable: **classification trees**
- Continuous response variable: **regression trees**
- Leaves (R_m's) are also called **terminal nodes**
- The other nodes where splits occur are **internal nodes**
- Trees are drawn upside down, with the leaves at the bottom

Pruning a tree

- 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. i.e, weakest link pruning is often used
- For a subtree T, define the loss function $\sum_{m=1}^{|T|} N_m L_m + \alpha|T|$, where N_m is the number of obervations in R_m
  - Regression: $L_m = (1/N_m) \sum_{i:x_i \in R_m} (y_i - \bar{y}_{R_m})^2$
  - Classification: L_m is either Gini index (G_m) or Cross-entropy (D_m)
- The goal is to minimize the loss function in terms of the **complex parameter** α. Since α corresponds to a unique number of terminal nodes (why?), when α is chosen, the number of terminal nodes is chosen.

Gini index

- A measure of total variance across the K classes
- Gini index for the mth region is $G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
- $\hat{p}_{mk}$ is the proportion of training observations in the mth region (i.e., R_m) and are actually from the kth class.
  - For example: two classes: Y and N, and two regions: R_1 and R_2. If YYN are in R_1, and YNNN are in R_2
  - Then $\hat{p}_{11} = 2/3$, $\hat{p}_{12} = 1/3$, $\hat{p}_{21} = 1/4$, $\hat{p}_{22} = 3/4$
  - Then Gini index for R_1 is $G_1 = 2/9 + 2/9 = 4/9$, and for R_2 is $G_2 = 3/16 + 3/16 = 3/8$
- A small value indicates that a node contains predominantly observations from a single class (e.g., 3/8 < 4/9)

Cross-entropy (i.e., Deviance)

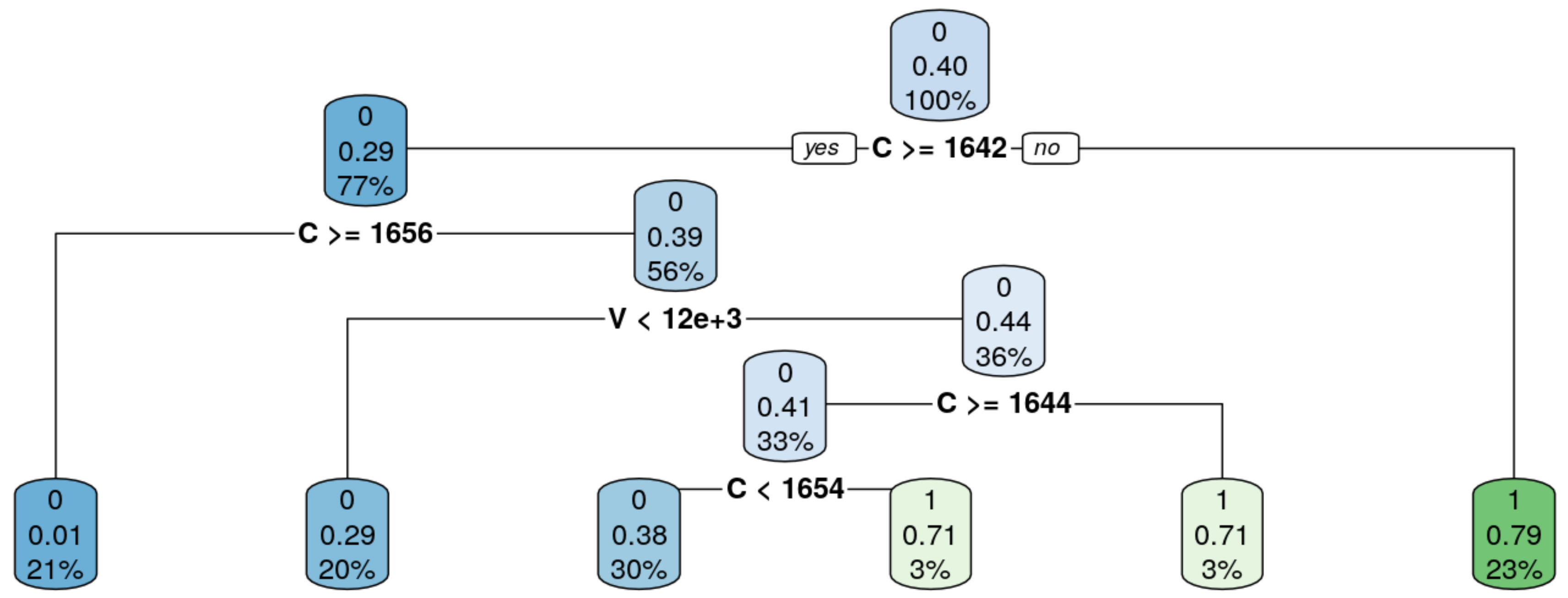
- With the same $\hat{p}_{mk}$ as that for Gini index, cross-entropy for the mth region is $D_m = -\sum_{k=1} \hat{p}_{mk} \log \hat{p}_{mk}$
- Gini index and the cross-entropy are very similar numerically, and can be used alternately.

Implementation of tree pruning

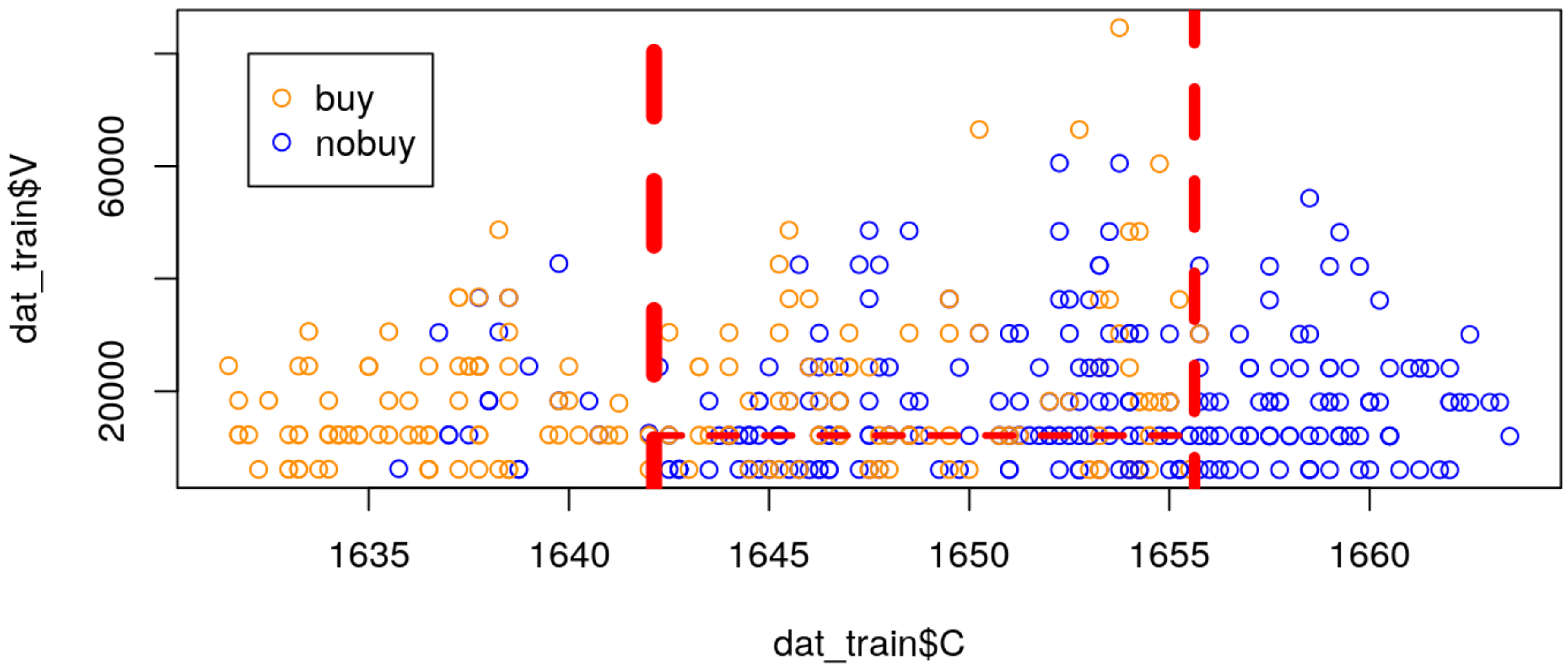
- Cross validation method is used to choose the optimal α
- Refer to page 20 of ISL slides for a summary of tree algorithm.
- Refer to page 12 of An Introduction to Recursive Partitioning Using the RPART Routines for understanding the algorithm.

Classification trees - R examples

```
library(rpart)
library(rpart.plot)
datXY_up <- data.frame(read.csv("~/Dropbox/Teaching/STAT430/slides/datXY_up.csv", header = T))
datXY_up$Y_dir <- as.factor(datXY_up$Y_dir)
dat_train <- subset(datXY_up,Type=="training")
set.seed(0)
tre <- rpart(Y_dir ~ C + V, data = dat_train, method = "class")
rpart.plot(tre)
```



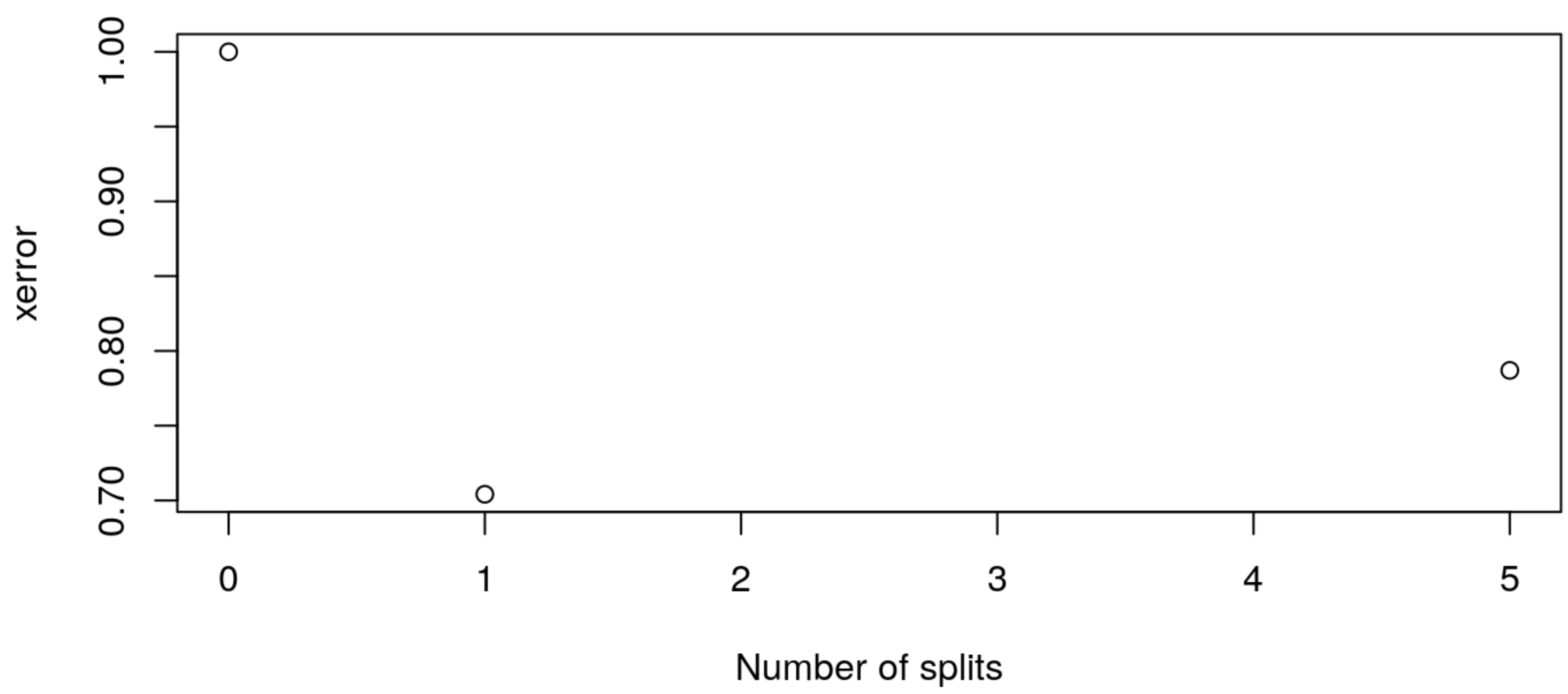
```
plot(dat_train$V~dat_train$C,col=ifelse(dat_train$Y_dir=="0","blue","darkorange"))
legend(1632, 80000, legend=c("buy", "nobuy"), col=c("darkorange","blue"), pch=c(1,1))
segments(1642.125, 0, y1=100000, col = "red", lty=2, lwd = 7)
segments(1655.625, 0, y1=100000, col = "red", lty=2, lwd = 5)
segments(1642.125, 12111.500, x1=1655.625, col = "red", lty=2, lwd = 3)
```



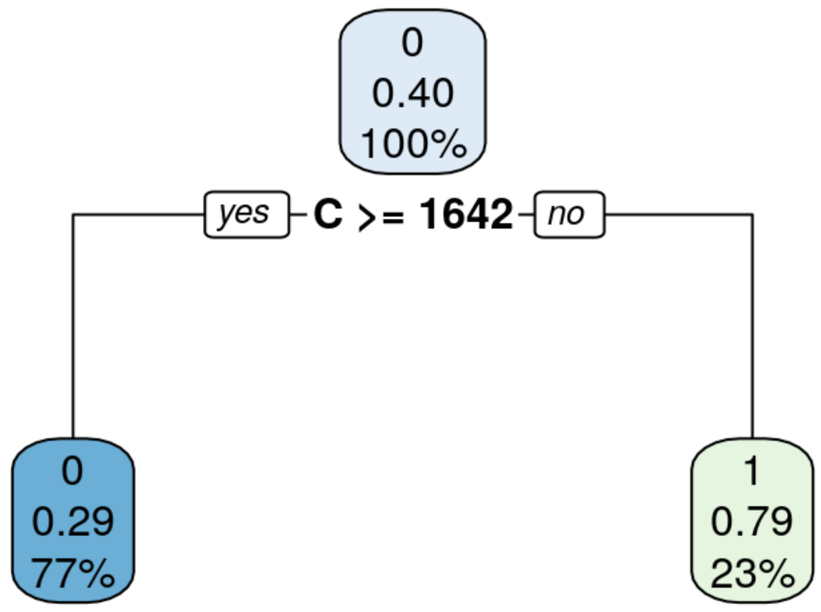
```r
tre$cptable
```

```
##           CP nsplit rel error    xerror       xstd
## 1 0.33727811      0 1.0000000 1.0000000 0.05941822
## 2 0.01775148      1 0.6627219 0.7041420 0.05461858
## 3 0.01000000      5 0.5917160 0.7869822 0.05637871
```

```r
plot(tre$cptable[,4]~tre$cptable[,2], xlab="Number of splits", ylab="xerror")
```

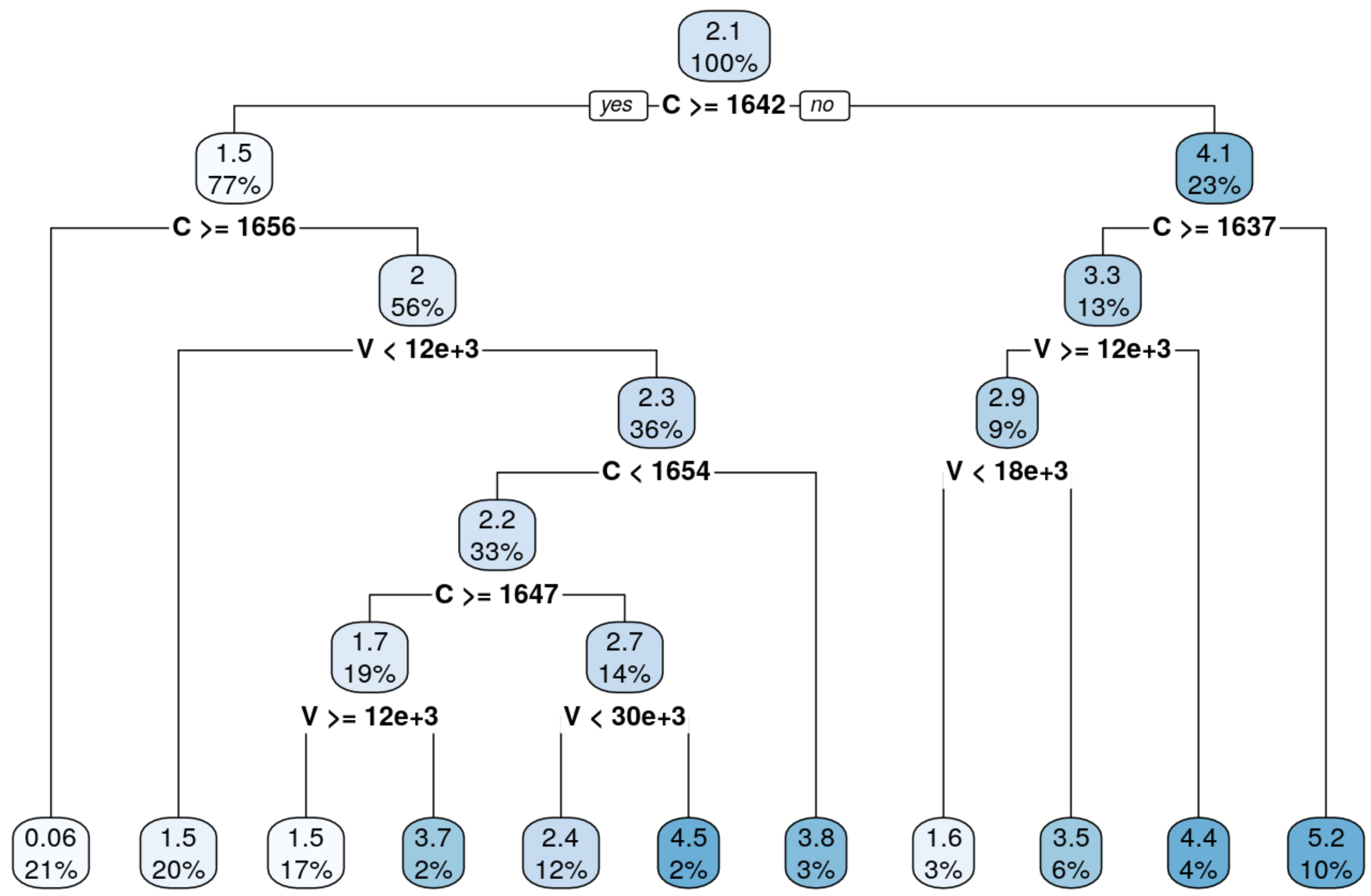


```
tre_pru <- prune(tre, cp=tre$cptable[which.min(tre$cptable[,"xerror"]),"CP"])
rpart.plot(tre_pru)
```

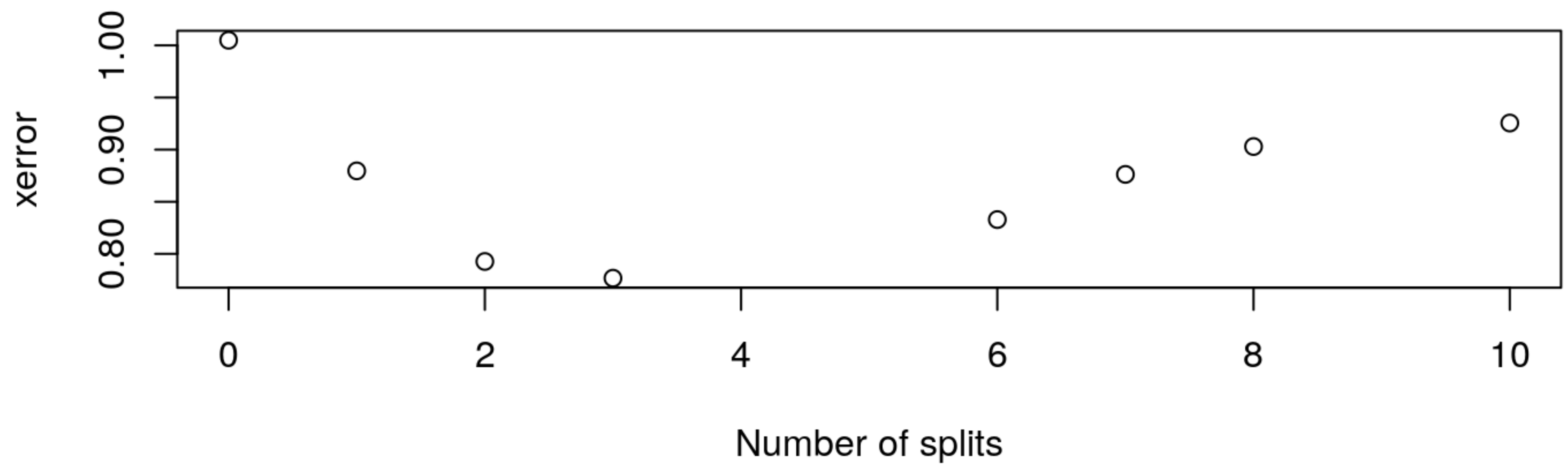


Regression trees

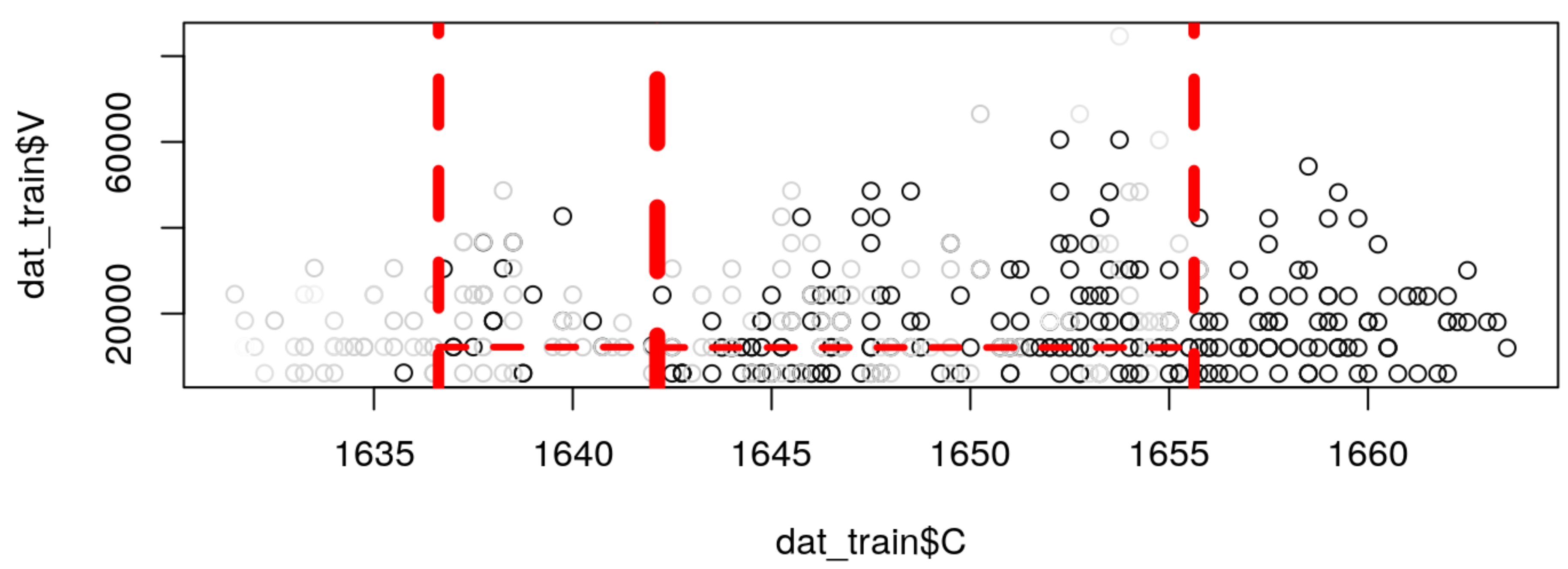
```r
tre <- rpart(Y_ret ~ C + V, data = dat_train, method = "anova")
rpart.plot(tre)
```



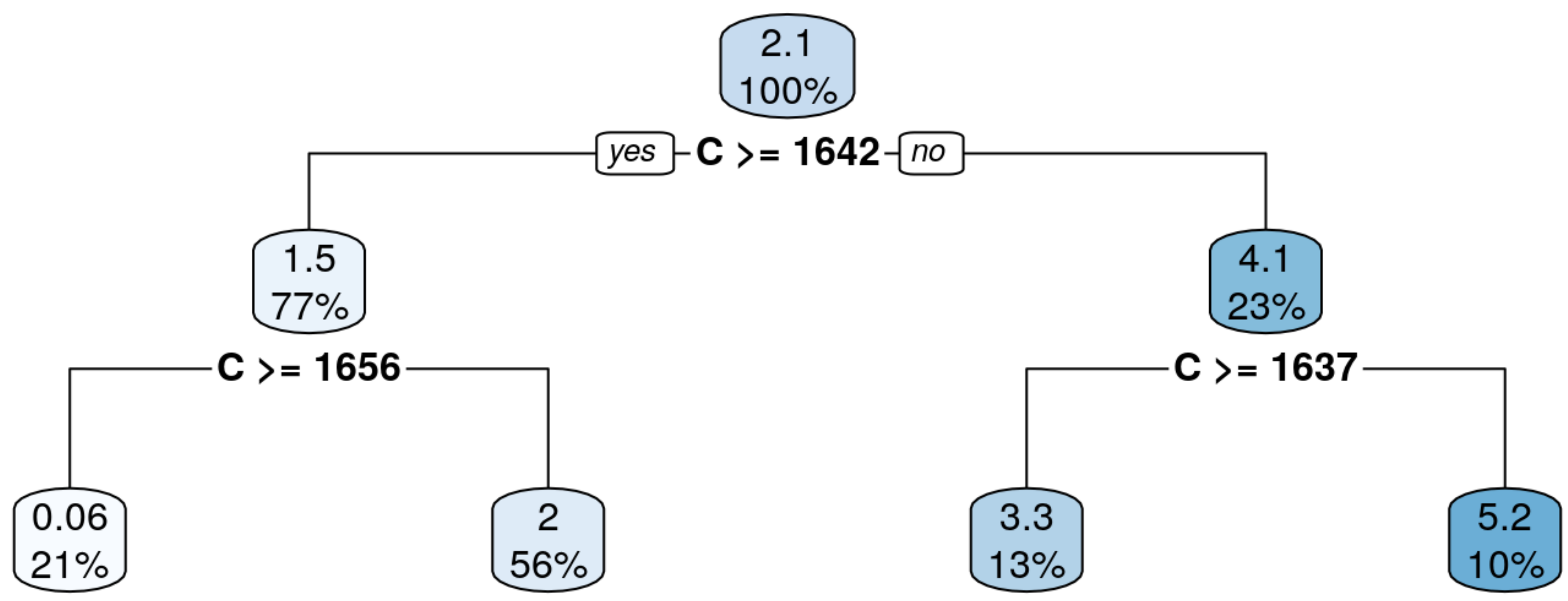
```
plot(tre$cptable[,4]~tre$cptable[,2], xlab="Number of splits", ylab="xerror")
```



```
maxret <- max(dat_train$Y_ret); minret <- min(dat_train$Y_ret)
maxdiff <- maxret - minret
oret <- order(dat_train$Y_ret) # return orders
colvec <- heat.colors(max(oret))
plot(dat_train$V~dat_train$C, col=grey((dat_train$Y_ret-minret+0.3)/(maxdiff+0.3)) )
segments(1642.125, 0, y1=100000, col = "red", lty=2, lwd = 7)
segments(1655.625, 0, y1=100000, col = "red", lty=2, lwd = 5)
segments(1636.625, 0, y1=100000, col = "red", lty=2, lwd = 5)
segments(1642.125, 12111.500, x1=1655.625, col = "red", lty=2, lwd = 3)
segments(1636.625, 12206, x1=1642.125, col = "red", lty=2, lwd = 3)
```



```
tre_pru <- prune(tre, cp=tre$cptable[which.min(tre$cptable[,"xerror"]),"CP"])
rpart.plot(tre_pru)
```



- Back to Course Scheduler