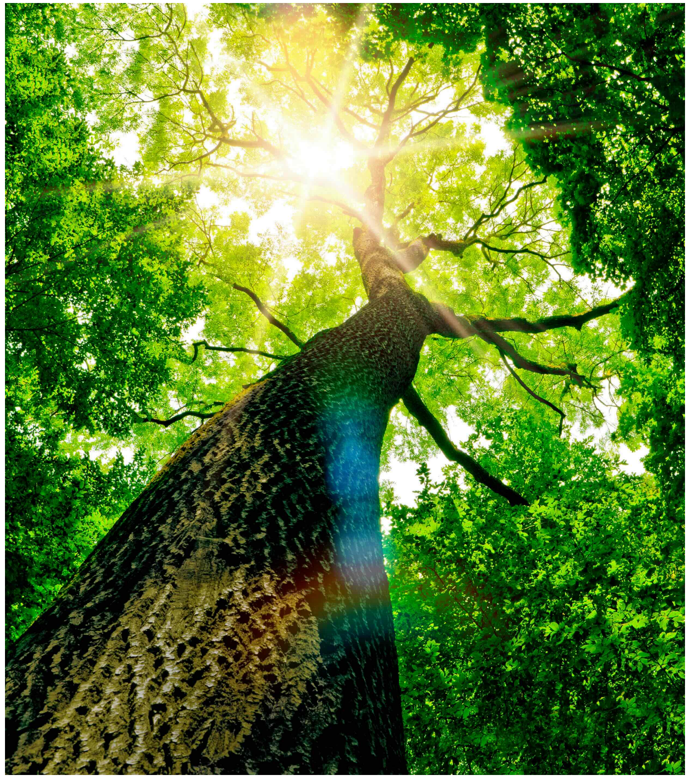
Tree-Based Methods

ACTL3142 & ACTL5110 Statistical Machine Learning for Risk Applications

Overview

Decision trees

- Stratify / segment the predictor space into a number of simple regions
- The set of splitting rules can be summarised in a tree



A tree (stock photo)

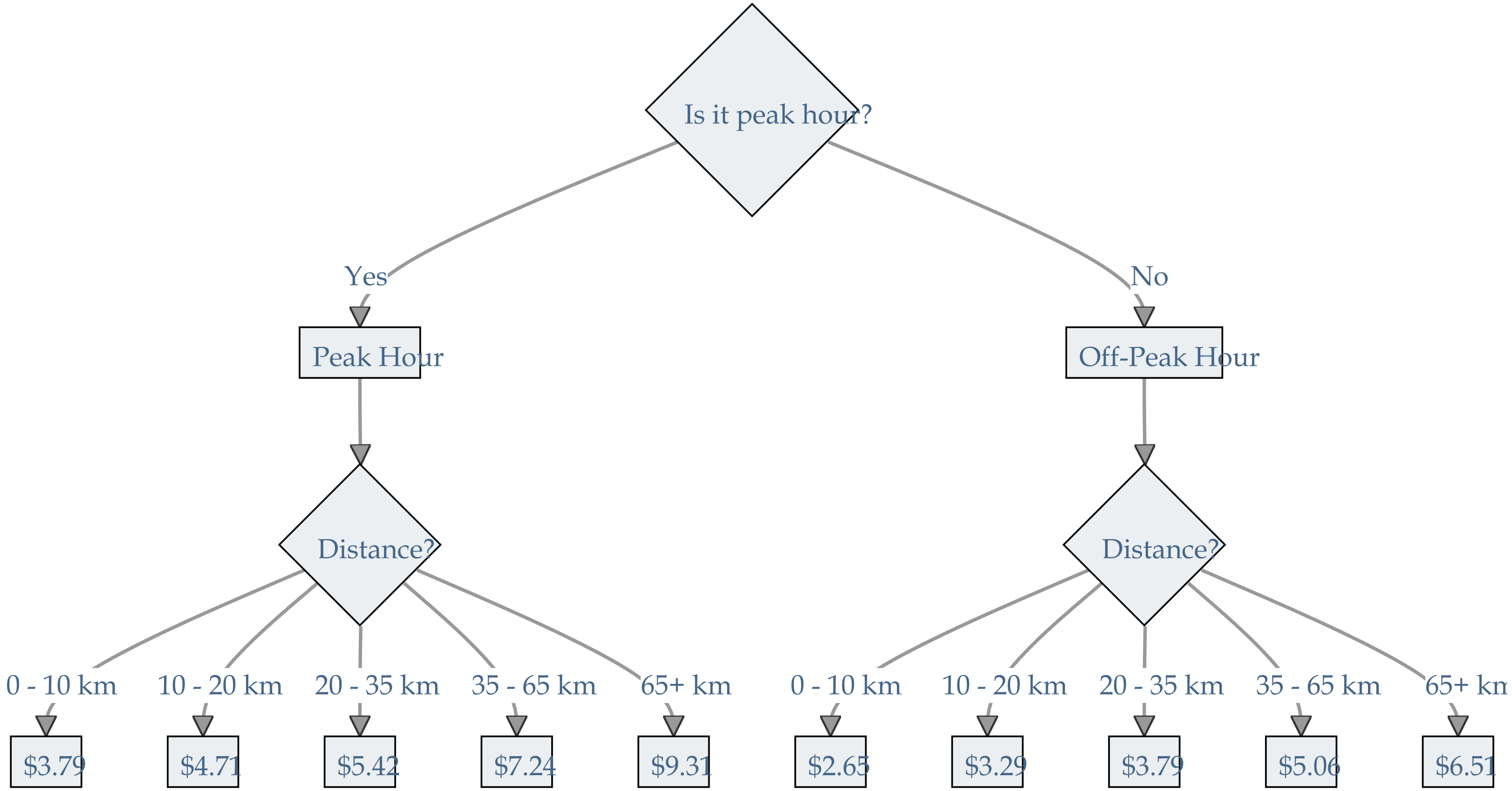
Bagging, random forests, boosting

- Ensemble methods
- Produce multiple trees
- Improve the prediction accuracy of tree-based methods
- Lose some interpretation

Lecture Outline

- **Decision Trees**
- Growing a Tree
- National Flood Insurance Program Demo
- Pruning a Tree
- Bootstrap Aggregation
- Random Forests
- Boosting

How much is a train ticket?



In code

R Python

```
rail_cost <- function(peak_hours, distance) {
  if (peak_hours) {
    if (distance <= 10) {
      cost <- 3.79
    } else if (distance <= 20) {
      cost <- 4.71
    } else if (distance <= 35) {
      cost <- 5.42
    } else if (distance <= 65) {
      cost <- 7.24
    } else {
      cost <- 9.31
    }
  } else {
    if (distance <= 10) {
      cost <- 2.65
    } else if (distance <= 20) {
      cost <- 3.29
    } else if (distance <= 35) {
      cost <- 3.79
    } else if (distance <= 65) {
      cost <- 5.06
    } else {
      cost <- 6.51
    }
  }
  return(cost)
}
```

Hitters dataset

R | Python

```r
data(Hitters)
Hitters
```

	AtBat <int>	Hits <int>	HmRun <int>	Runs <int>	RBI <int>
-Andy Allanson	293	66	1	30	29
-Alan Ashby	315	81	7	24	38
-Alvin Davis	479	130	18	66	72
-Andre Dawson	496	141	20	65	78
-Andres Galarraga	321	87	10	39	42
-Alfredo Griffin	594	169	4	74	51
-Al Newman	185	37	1	23	8
-Argenis Salazar	298	73	0	24	24
-Andres Thomas	323	81	6	26	32
-Andre Thornton	401	92	17	49	66

1-10 of 322 rows | 1-6 of 21 columns Previous 1 2 3 4 5 6 ... 33 Next

Fit a basic tree

R | Python

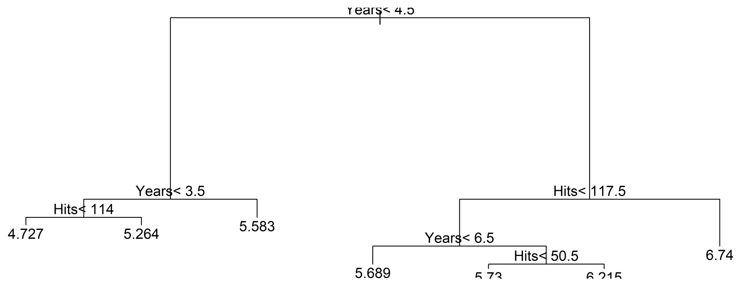
```r
(tree <- rpart(
  log(Salary) ~ Years + Hits,
  data = Hitters))
```

```
n= 263

node), split, n, deviance, yval
      * denotes terminal node

 1) root 263 207.153700 5.927222
   2) Years< 4.5 90  42.353170 5.106790
     4) Years< 3.5 62  23.008670 4.891812
       8) Hits< 114 43  17.145680 4.727386 *
       9) Hits>=114 19   2.069451 5.263932 *
     5) Years>=3.5 28  10.134390 5.582812 *
   3) Years>=4.5 173  72.705310 6.354036
     6) Hits< 117.5 90  28.093710 5.998380
      12) Years< 6.5 26   7.237690 5.688925 *
      13) Years>=6.5 64  17.354710 6.124096
        26) Hits< 50.5 12   2.689439 5.730017 *
        27) Hits>=50.5 52  12.371640 6.215037 *
     7) Hits>=117.5 83  20.883070 6.739687 *
```

```r
plot(tree)
text(tree)
```

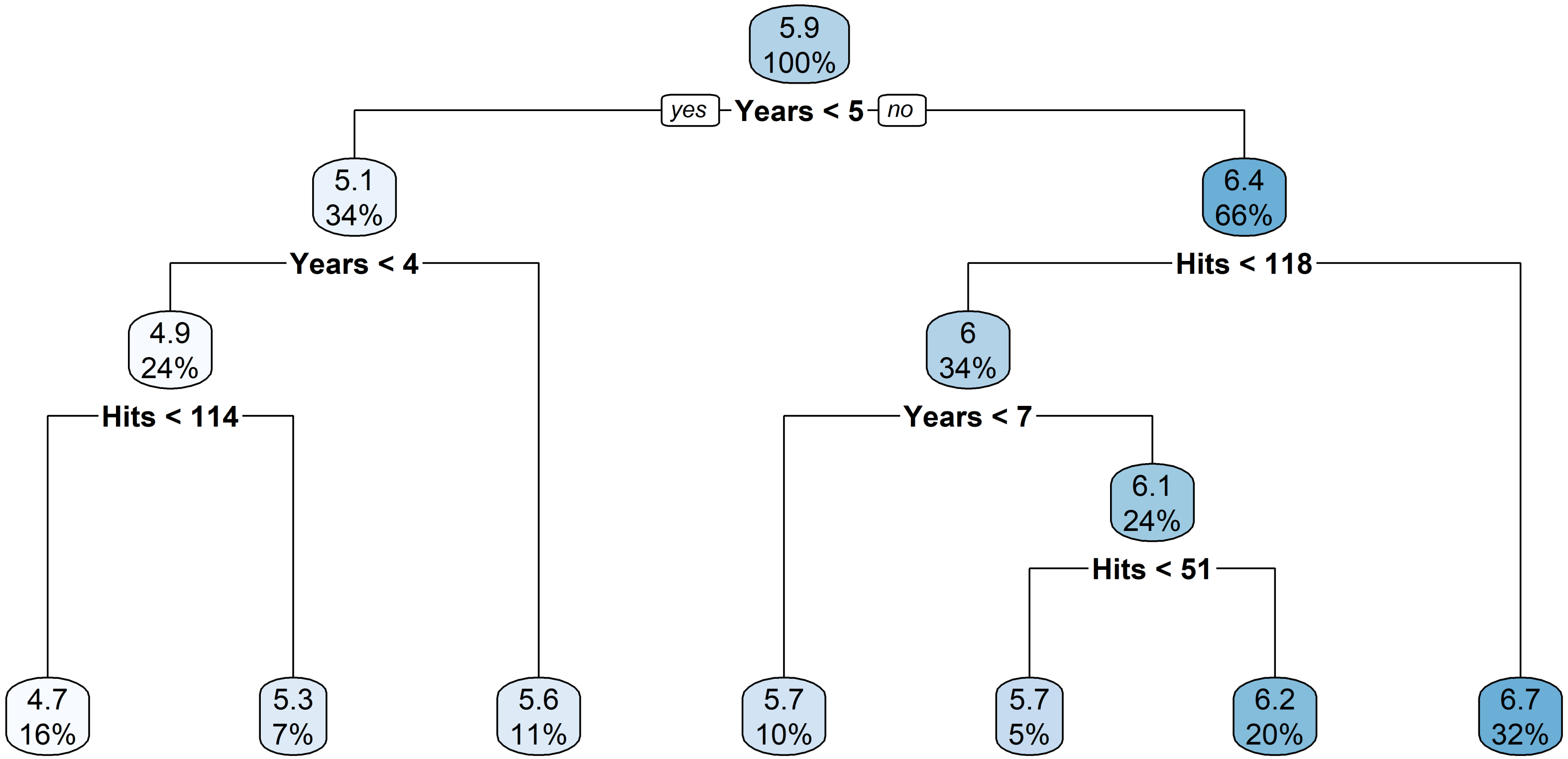


Source: These plots are recreating ISLR2's Figure 8.4.

Nicer plots for decision trees

R | Python

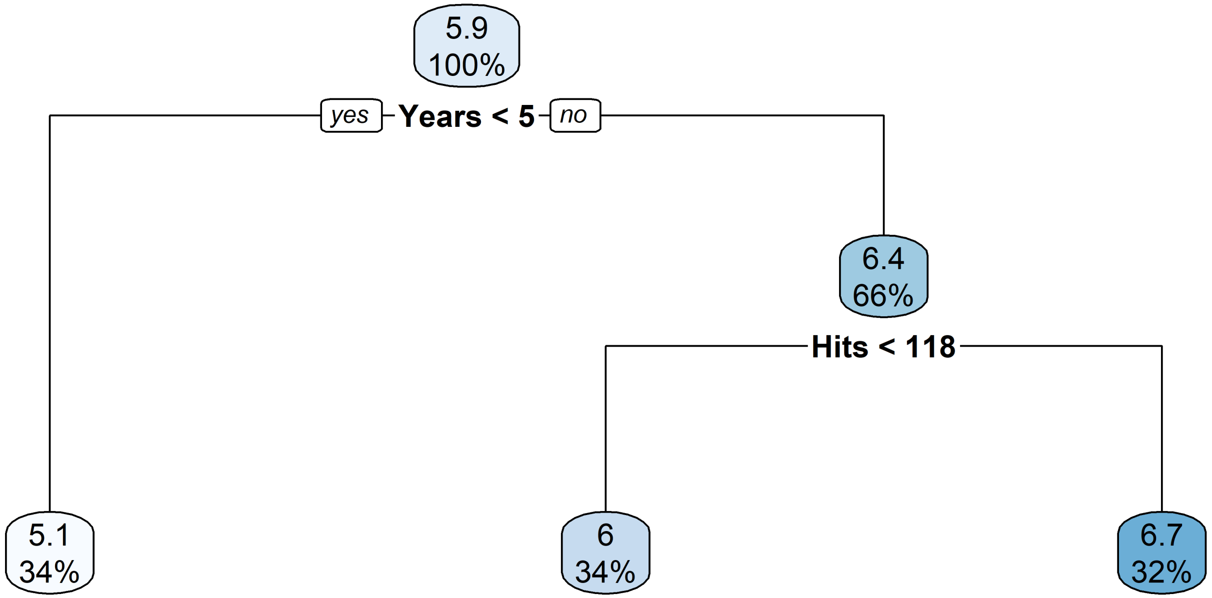
```
rpart.plot(tree)
```



Source: A recreation of ISLR2's Figure 8.4.

After pruning that tree

```
pruned_tree <- prune(tree, cp = tree$cptable[3, "CP"])
rpart.plot(pruned_tree)
```

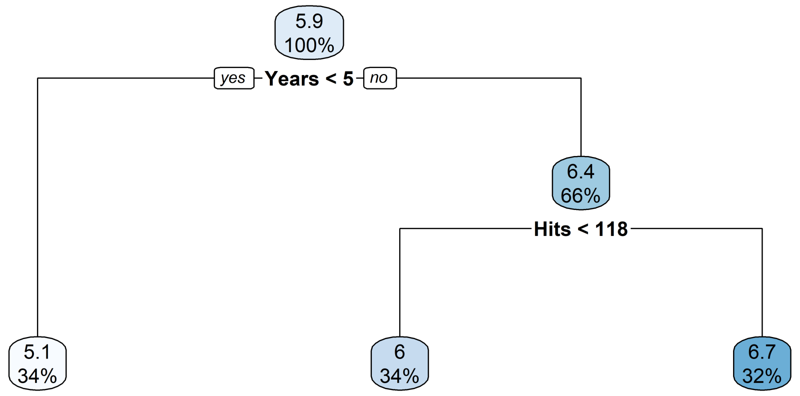


Source: A recreation of ISLR2's Figure 8.1.

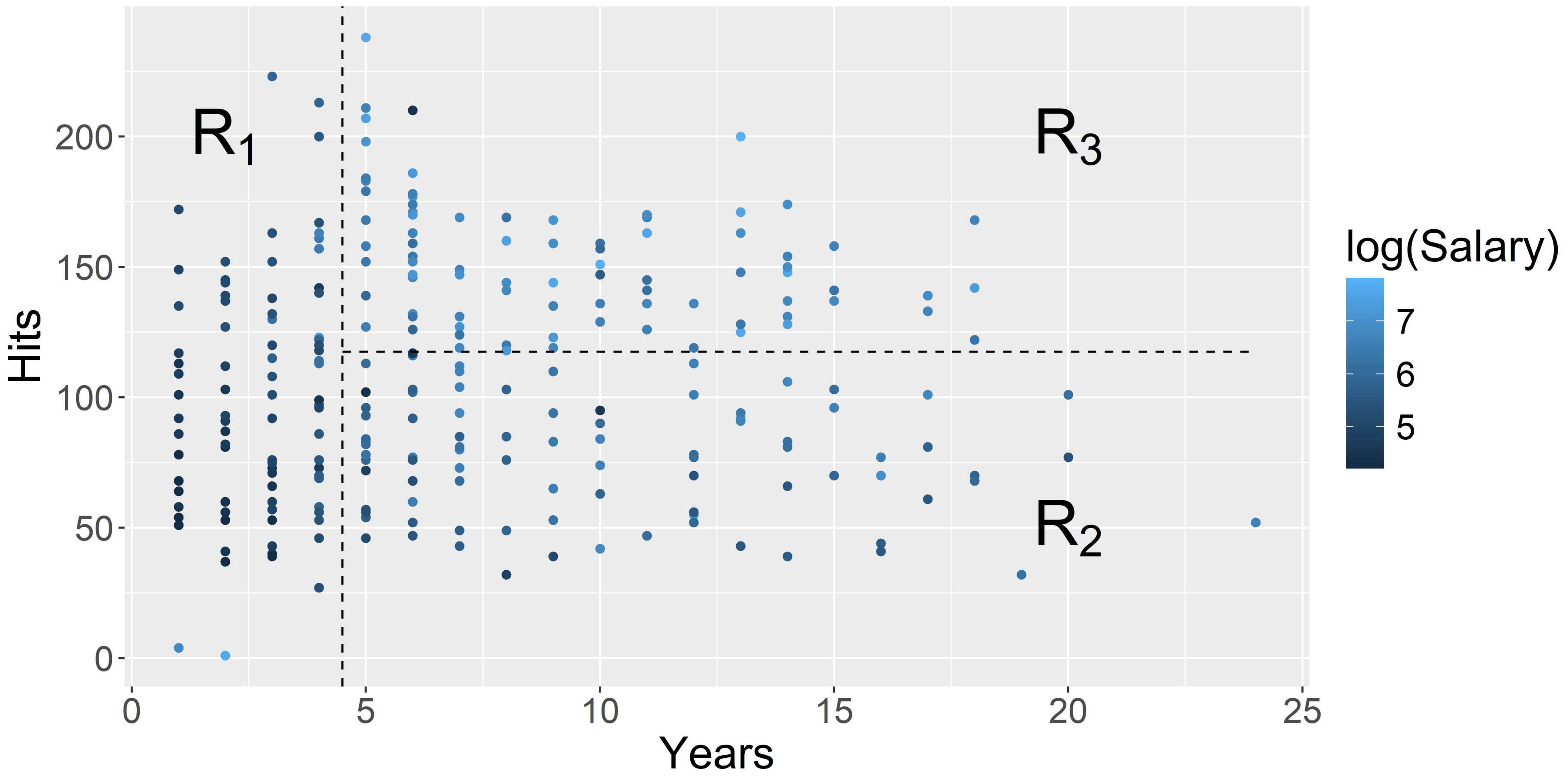
Tree Terminology

- **Internal nodes**
- **Terminal nodes** or **leaves**
- **Branches**
- **Root**

```
rpart.plot(pruned_tree)
```



Regions in the predictor space



Source: A recreation of ISLR2's Figure 8.2.

Tree regions & predictions

A decision tree is made by:

1. Dividing the predictor space (i.e. 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. Making the same prediction for every observation that falls into the region R_j
   - the mean response for the training data in R_j (regression trees)
   - the mode response for the training data in R_j (classification trees)

Example:

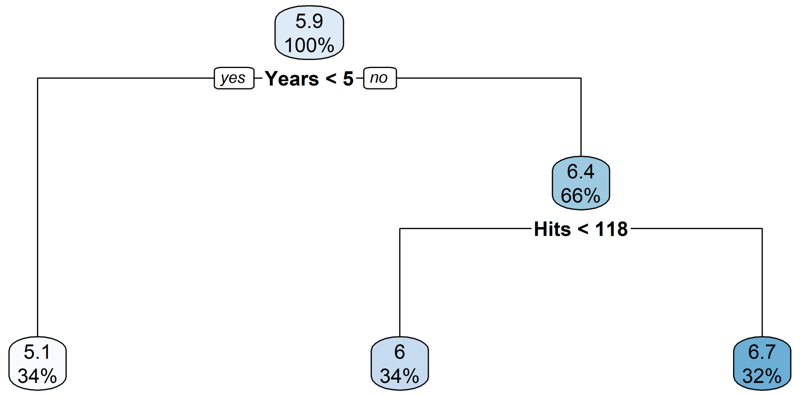
Region	**Predicted salaries**
$R_1 = \{X \mid \texttt{Years} < 4.5\}$	$\$1,000 \times e^{5.107} = \$165,174$
$R_2 = \{X \mid \texttt{Years} \geq 4.5, \texttt{Hits} < 117.5\}$	$\$1,000 \times e^{5.999} = \$402,834$
$R_3 = \{X \mid \texttt{Years} \geq 4.5, \texttt{Hits} \geq 117.5\}$	$\$1,000 \times e^{6.740} = \$845,346$

Discussion

How do you interpret the results of this tree? In particular, consider the following questions

- Which factor is more important in determining `Salary`?
- How does `Hits` affect `Salary`?

```
rpart.plot(pruned_tree)
```

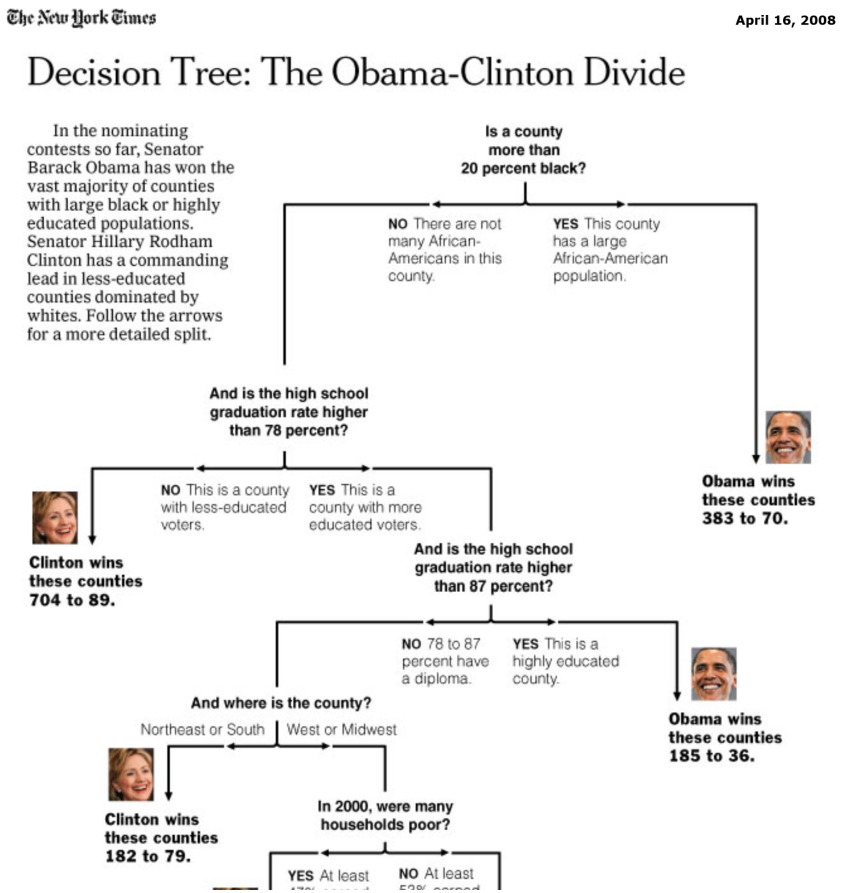


Decision trees: summary

Decision trees are simple, popular, and easy to interpret.

They are not the most accurate method, but they can be great to understand the data.

They do form the basis for more accurate and complex methods like random forests and boosting.

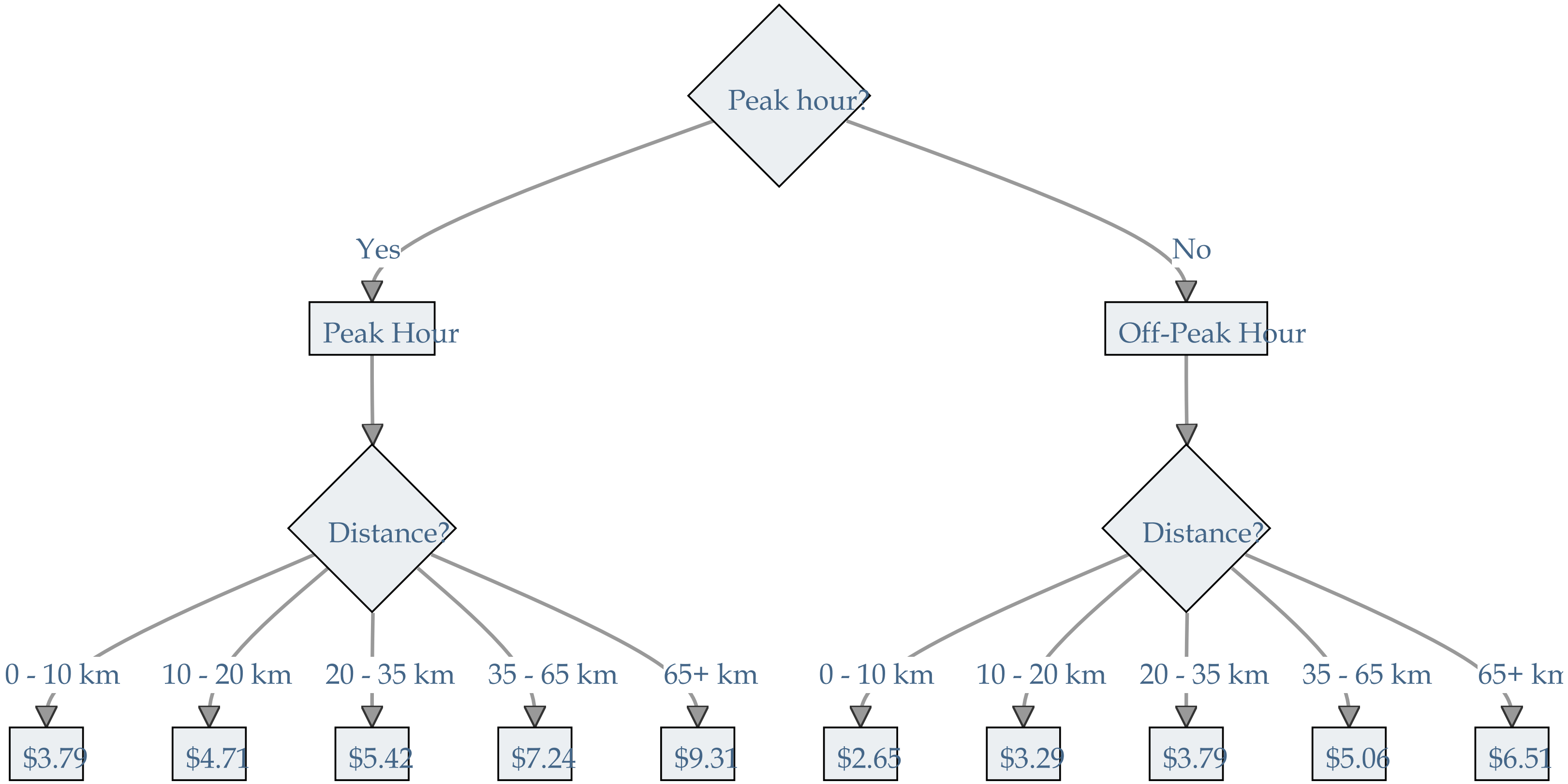


A decision tree in the wild.

Source: New York Times (2008), Decision Tree: The Obama-Clinton Divide.

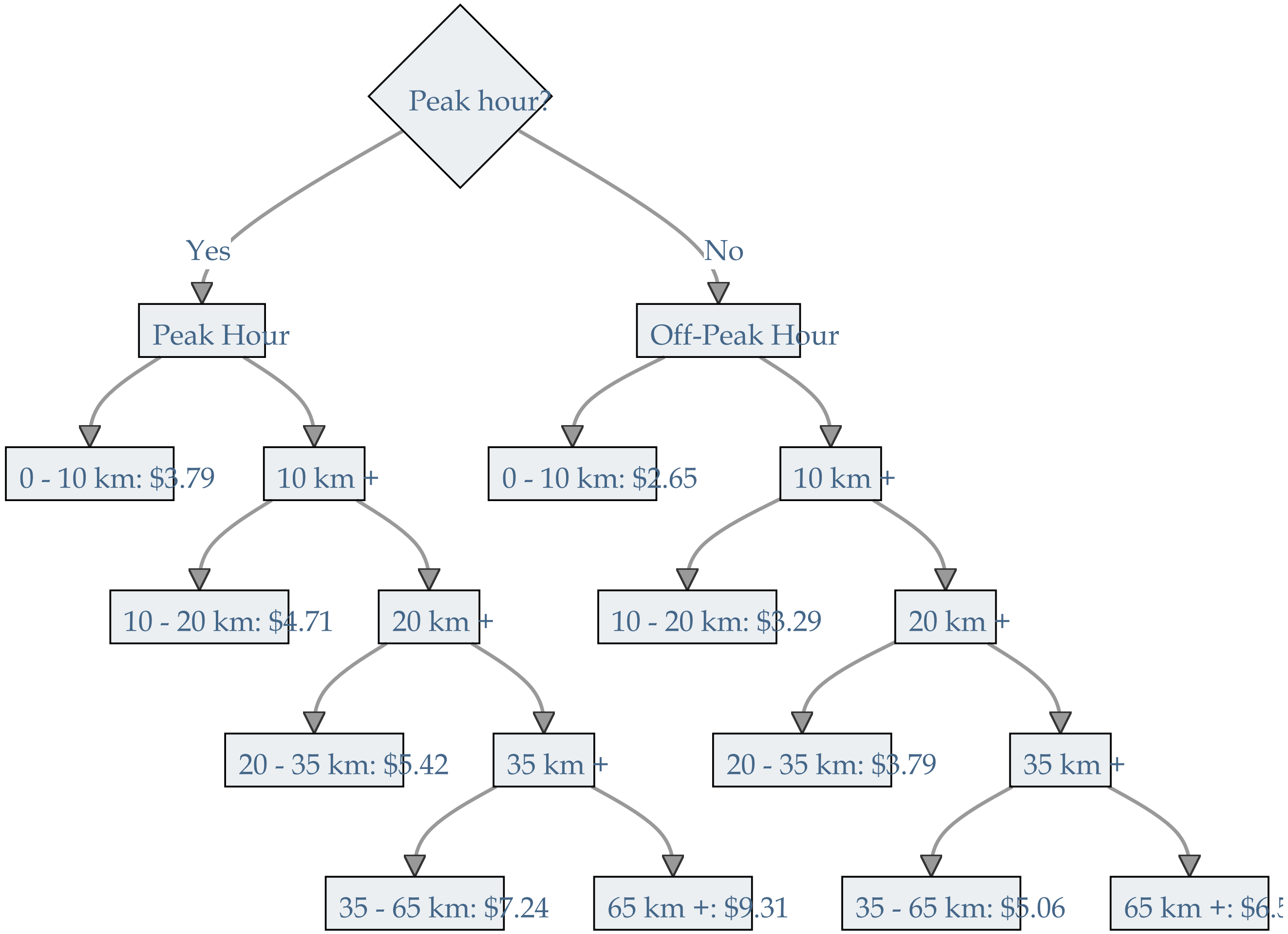
Non-binary train cost tree

A decision tree enforces binary splits…



Binary train cost tree

… but we can still represent non-binary splits in a binary tree.



Popular (IME 2023 abstracts)

Lecture Outline

- Decision Trees
- **Growing a Tree**
- National Flood Insurance Program Demo
- Pruning a Tree
- Bootstrap Aggregation
- Random Forests
- Boosting

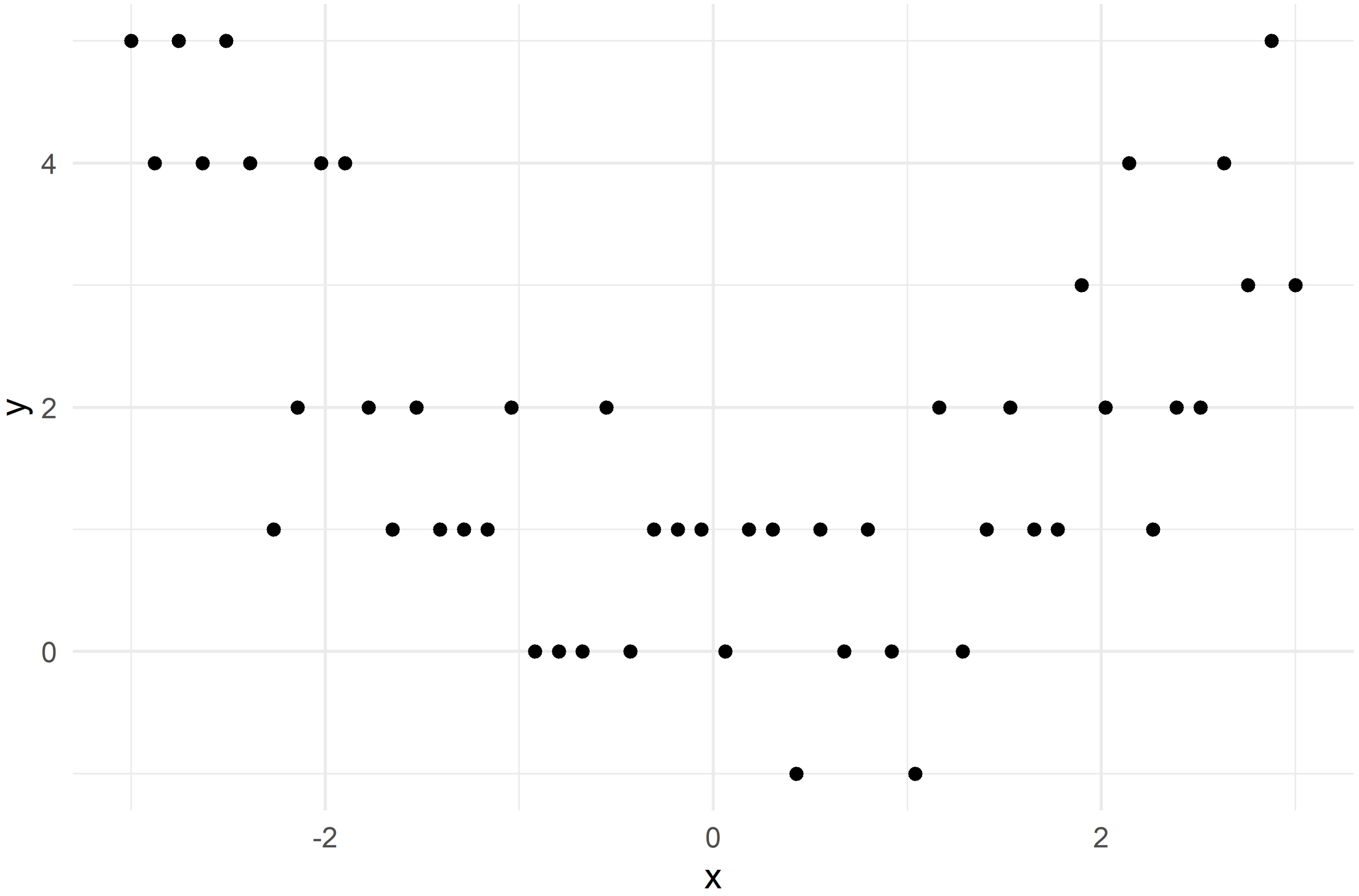
Fitting a regression tree

- Divide the predictor space into high-dimensional rectangles, or boxes
- The goal is to find boxes $R_1, R_2, \ldots, R_J$ that minimise

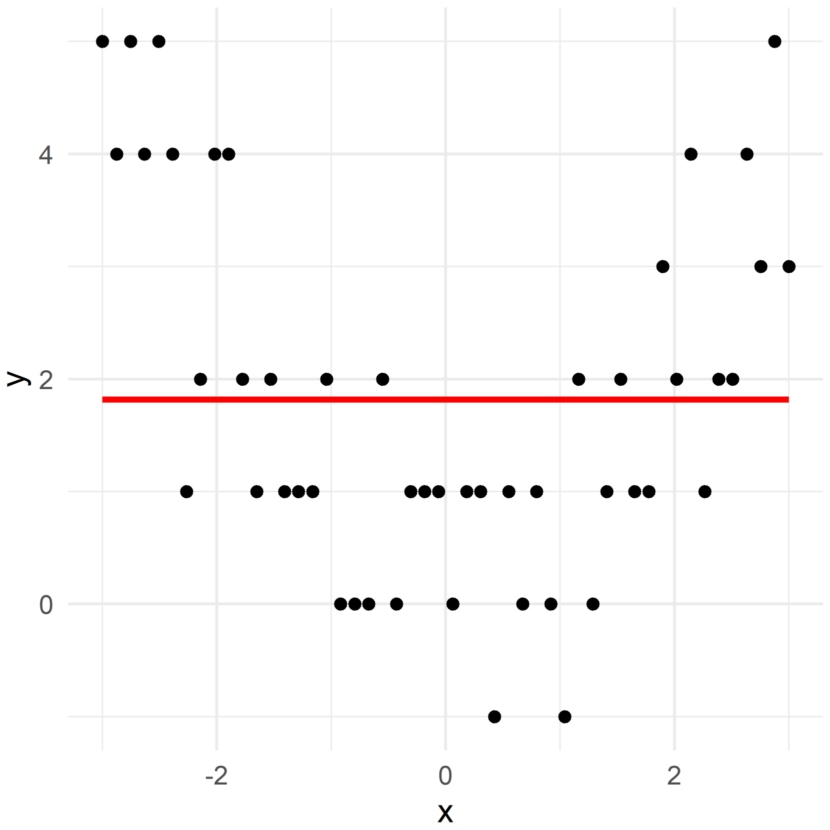
$$\text{RSS} = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

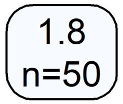
  where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box
- Computationally unfeasible to consider every possible partition
  - take a top-down, greedy approach…

Synthetic regression dataset

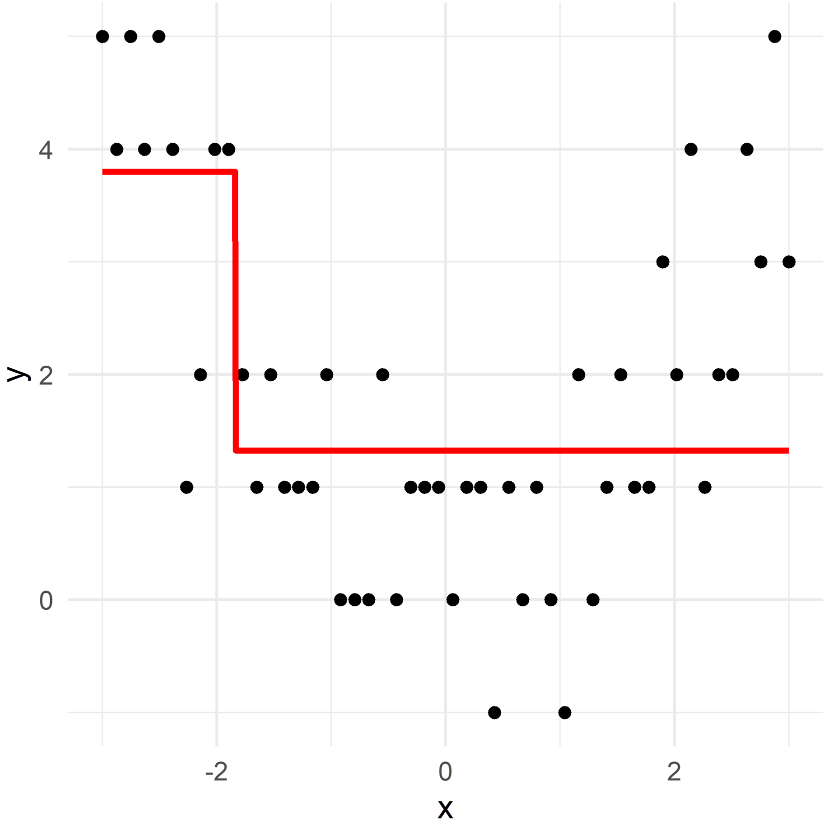


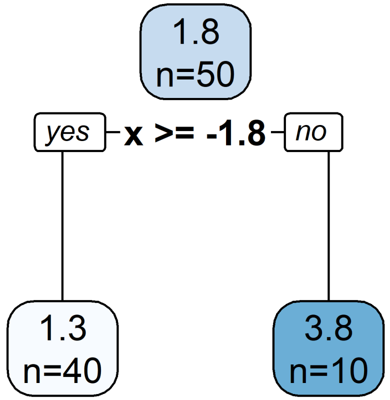
Growing a regression tree I



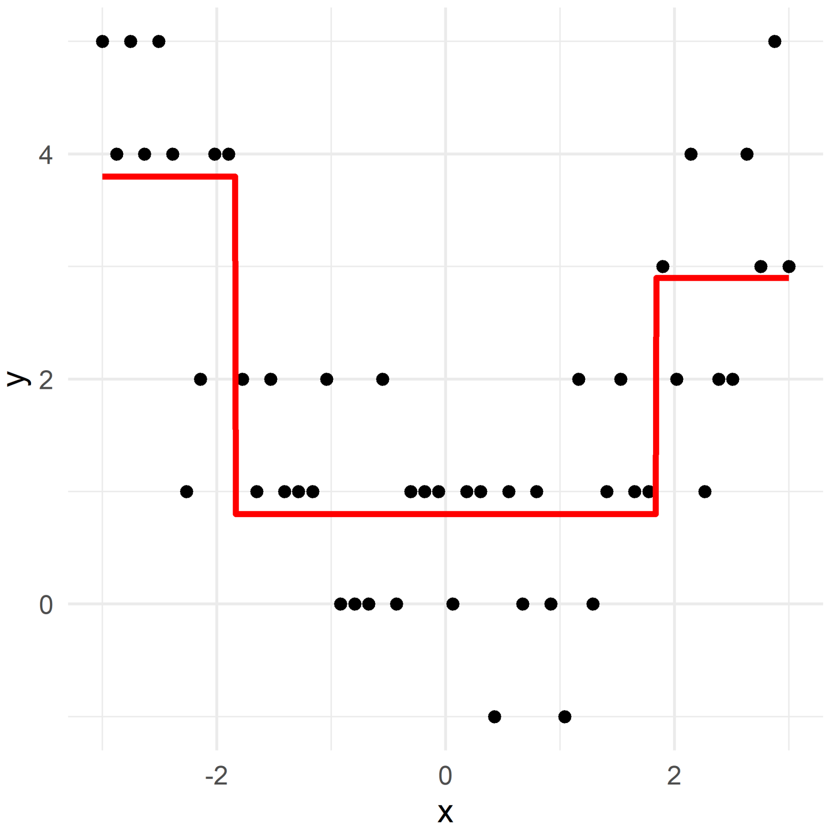


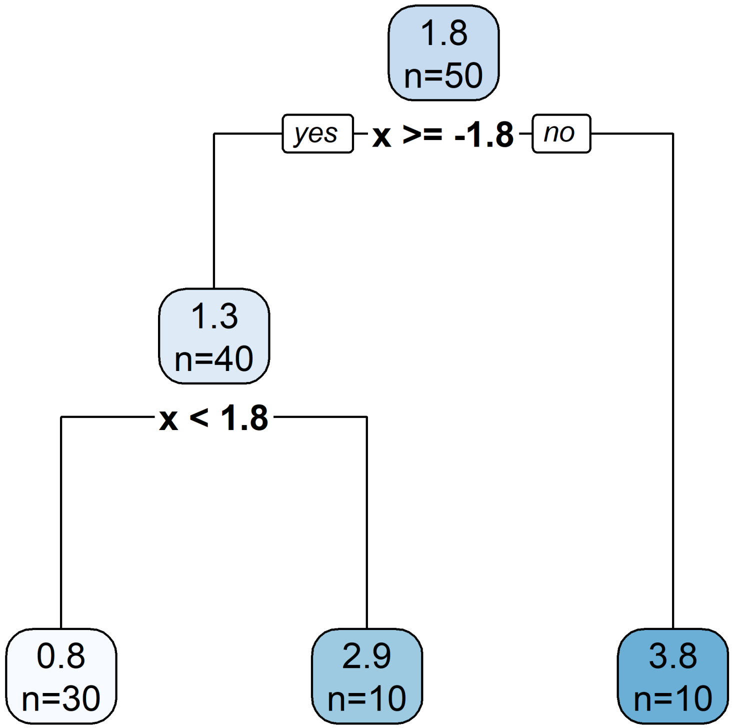
Growing a regression tree II



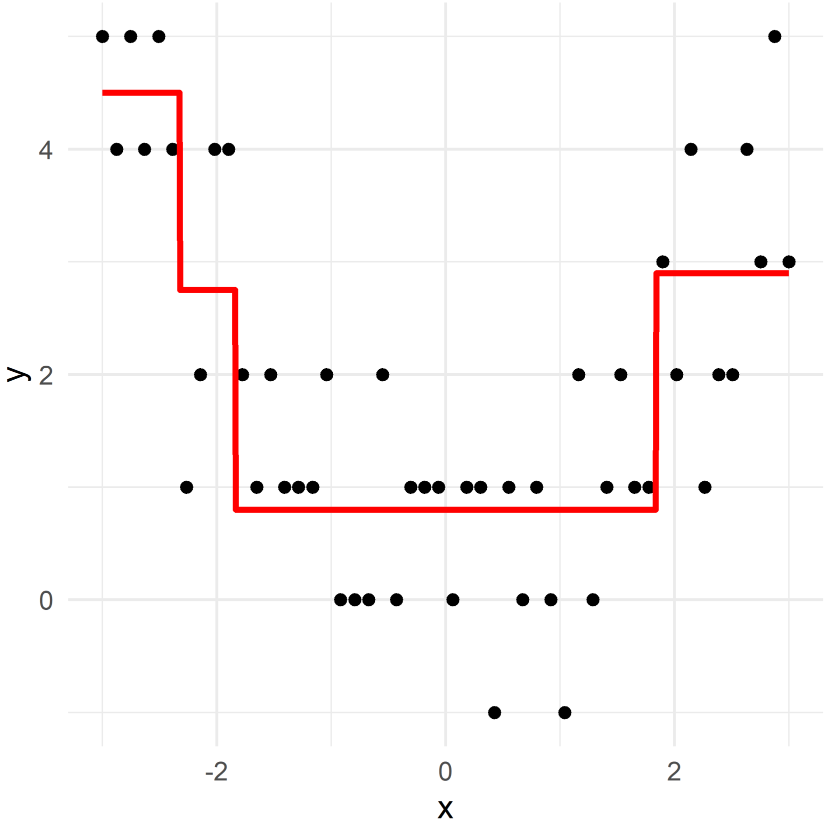


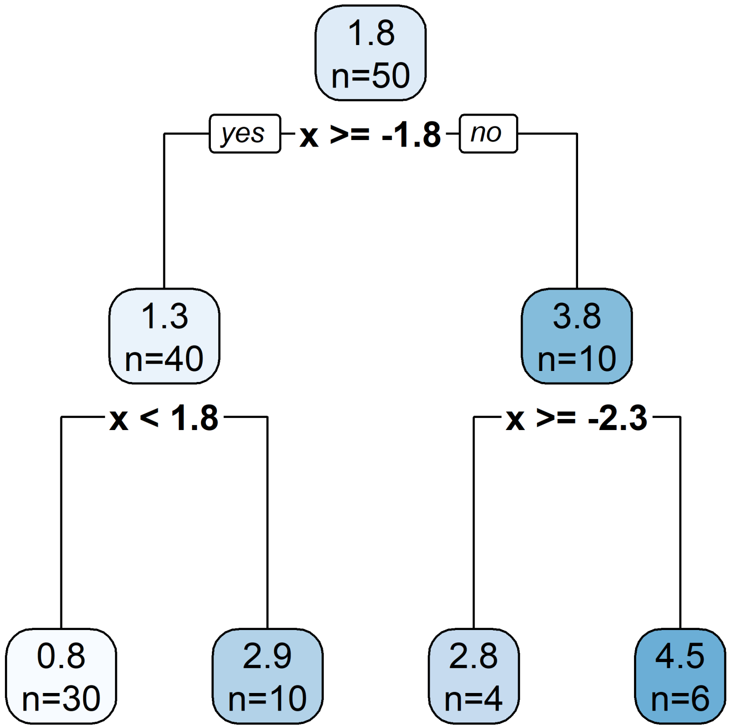
Growing a regression tree III





Growing a regression tree IV





Recursive binary splitting

- Start with the root node, and make new splits greedily one at a time
- Scan through all of the inputs
  - for each splitting variable, the split point s can be determined very quickly
  - The overall solution for this branch (i.e. selection of j) follows.
- Partition the data into the two resulting regions
- Repeat the splitting process on each of the two regions
- Continue the process until a stopping criterion is reached

UNSW SYDNEY

Recursive binary splitting details

- Consider a splitting variable j and split point s

$$R_1(j, s) = \{X | X_j \leq s\} \quad \text{and} \quad R_2(j, s) = \{X | X_j > s\}$$

- Find the splitting variable j and split point s that solve

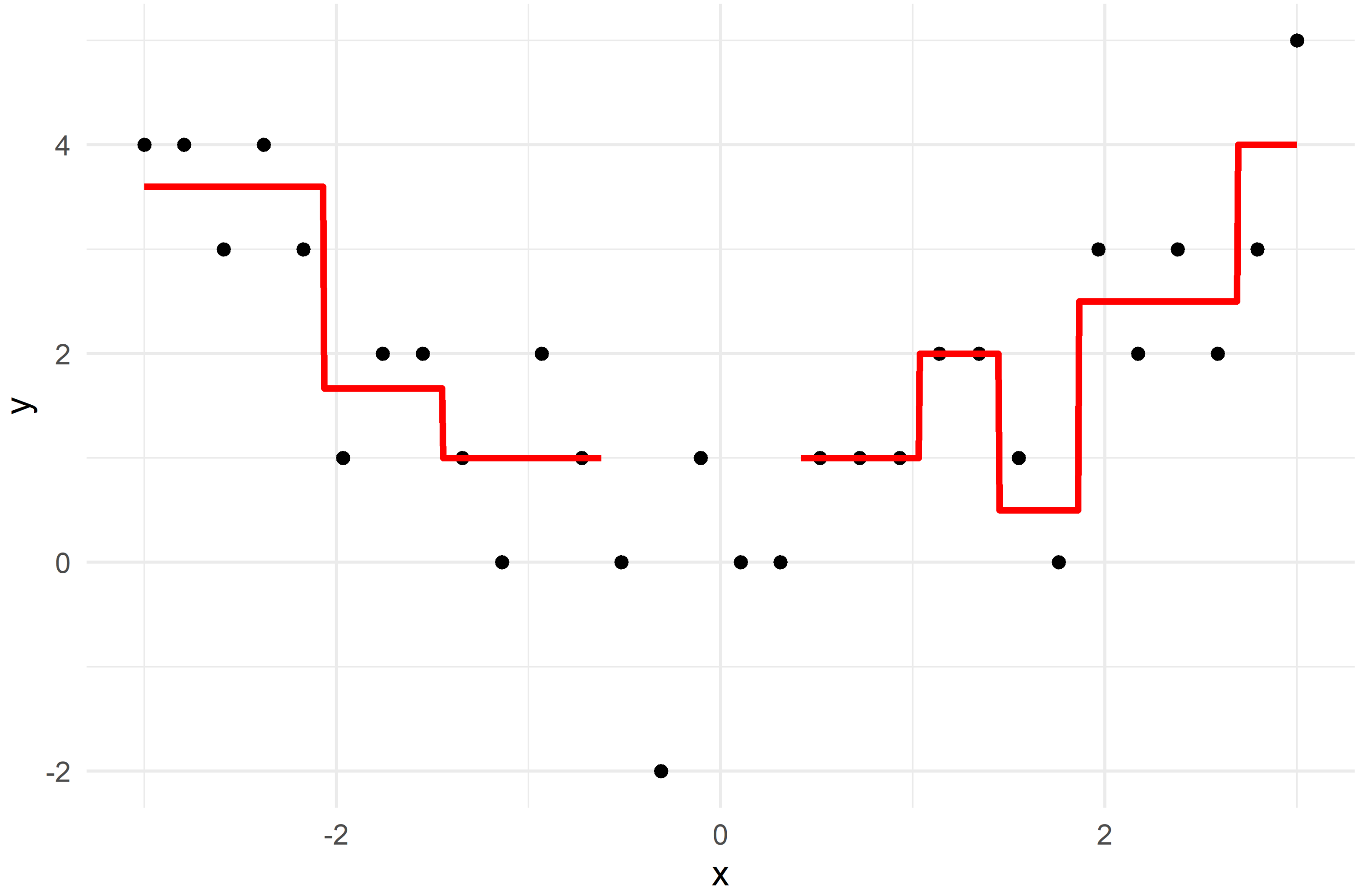
$$\min_{j,\, s} \left[\min_{c_1} \sum_{x_i \in R_1(j,\, s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,\, s)} (y_i - c_2)^2 \right]$$

where the inner mins are solved by

$$\hat{c}_1 = \text{Ave}(y_i | x_i \in R_1(j, s)) \quad \text{and} \quad \hat{c}_2 = \text{Ave}(y_i | x_i \in R_2(j, s))$$

2023 exam question

What would be the tree's predicted value for y at $x = 0$?



Classification trees

Very similar to a regression tree, except:

- Predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs
- RSS cannot be used as a criterion for making the binary splits, instead use a measure of node purity:

Gini index, or

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

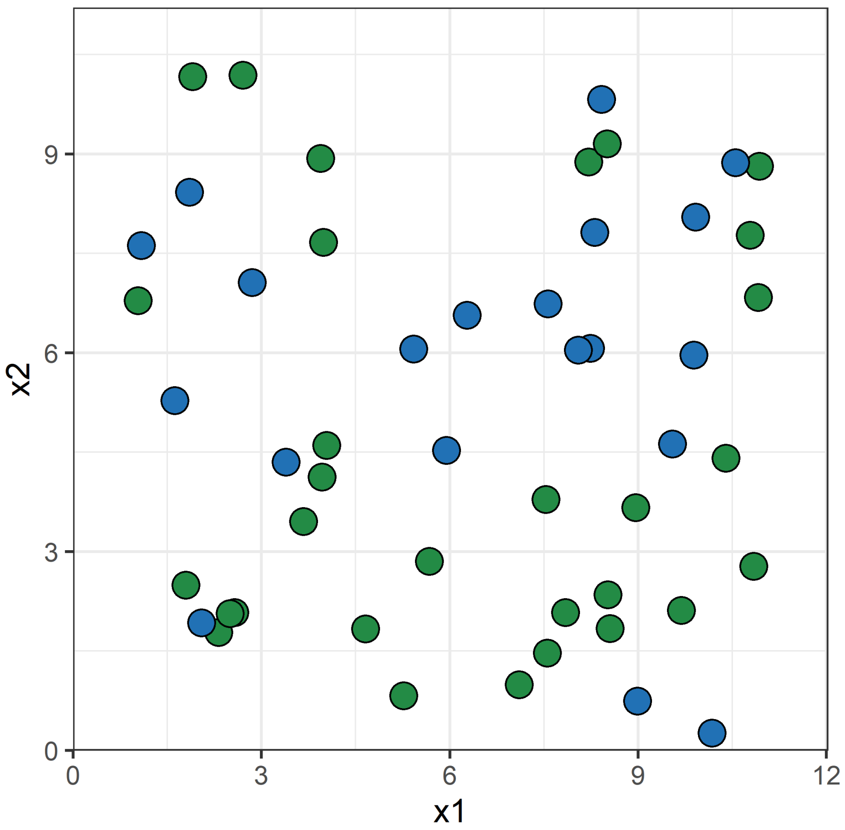
entropy

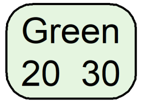
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \ln(\hat{p}_{mk})$$

where

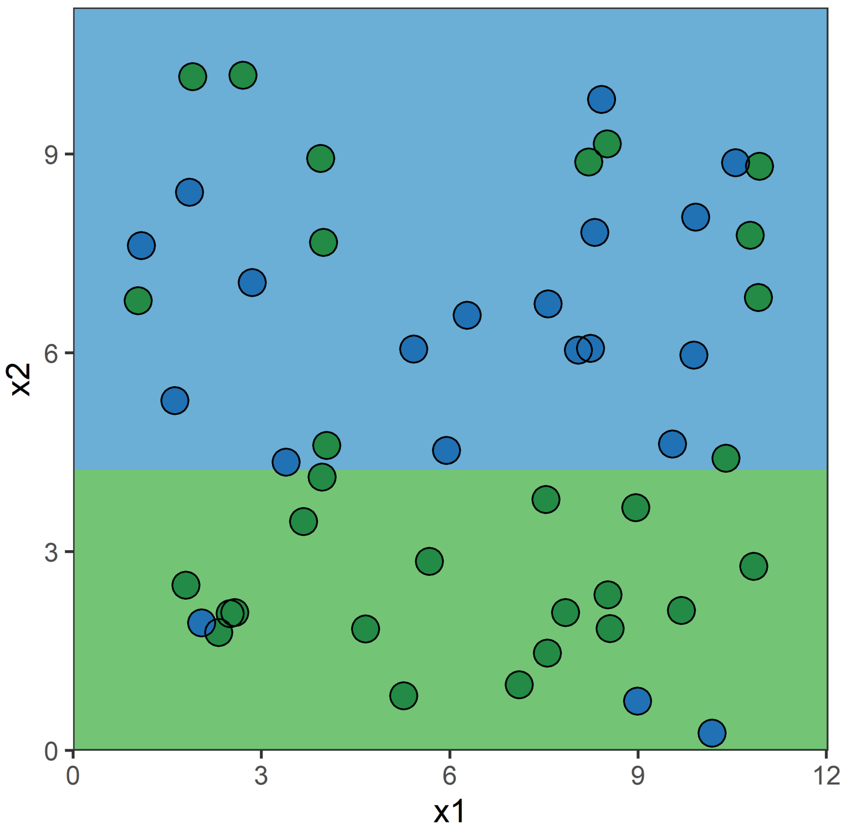
$$\hat{p}_{mk} = \frac{1}{|R_m|} \sum_{x_i \in R_m} I(y_i = k)\,.$$

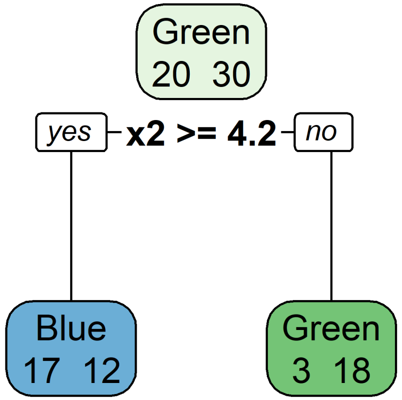
Growing a classification tree I



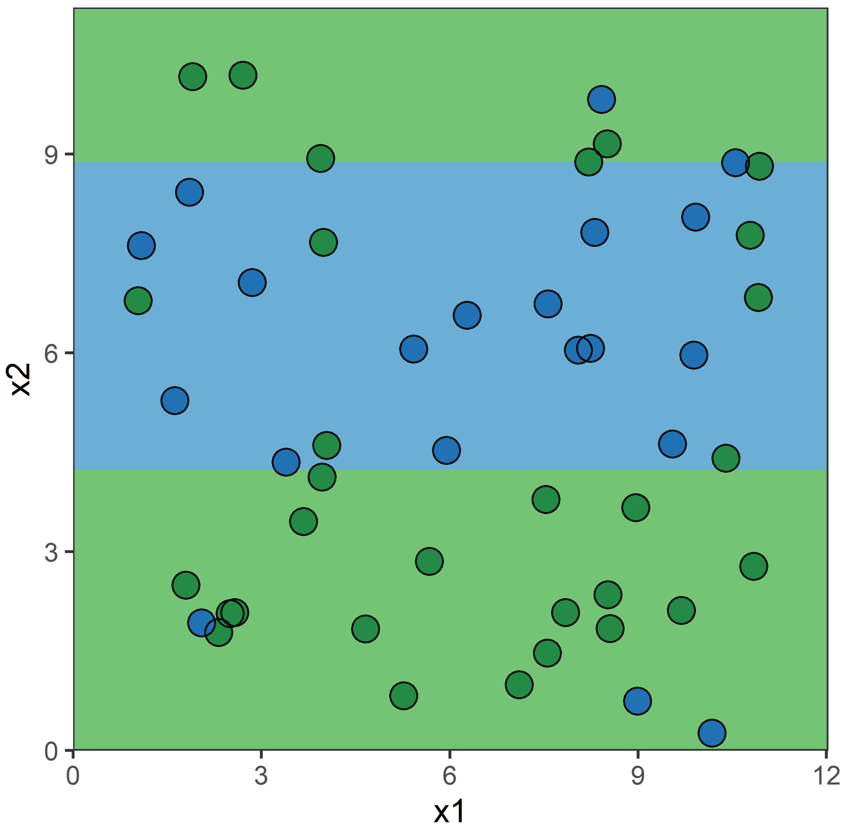


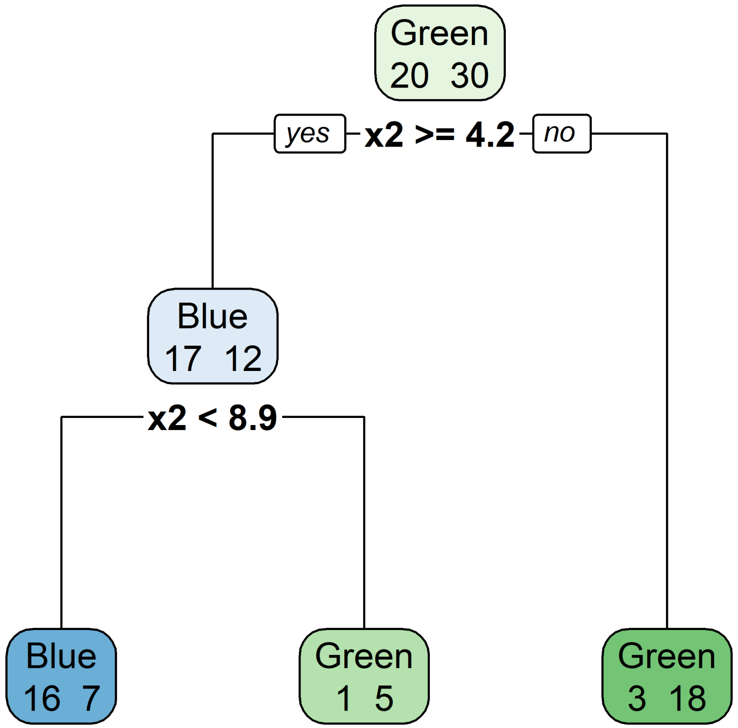
Growing a classification tree II



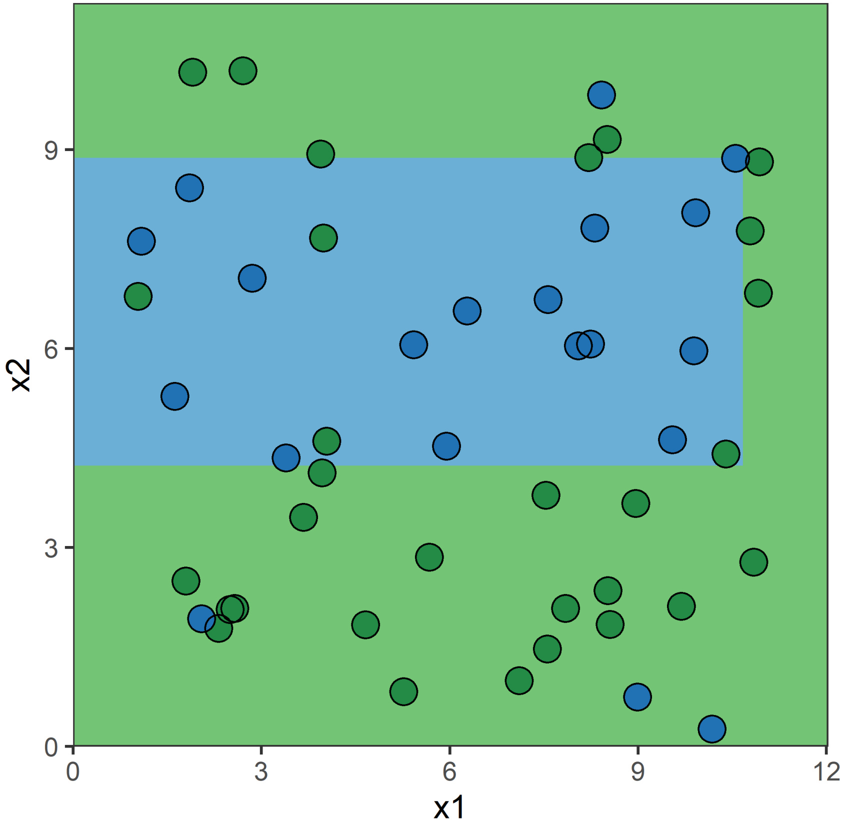


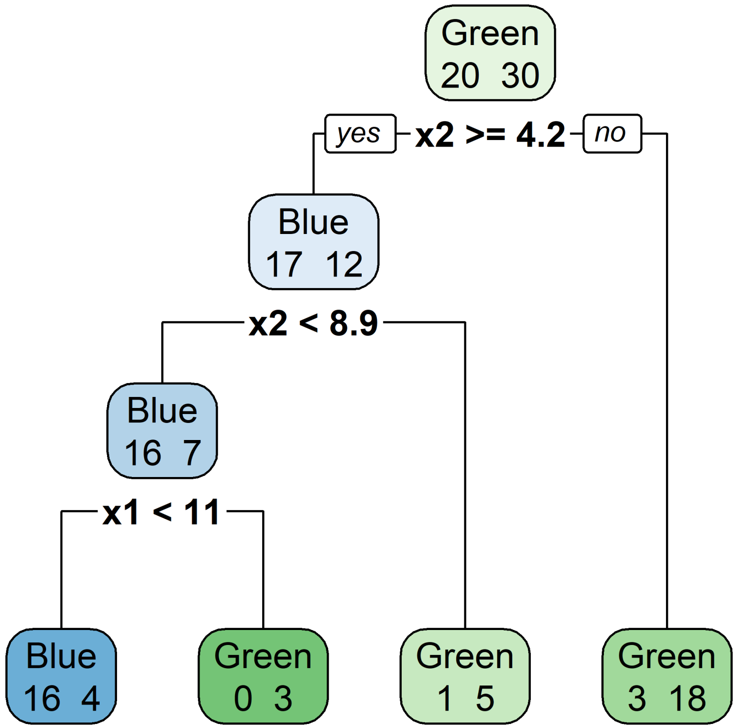
Growing a classification tree III





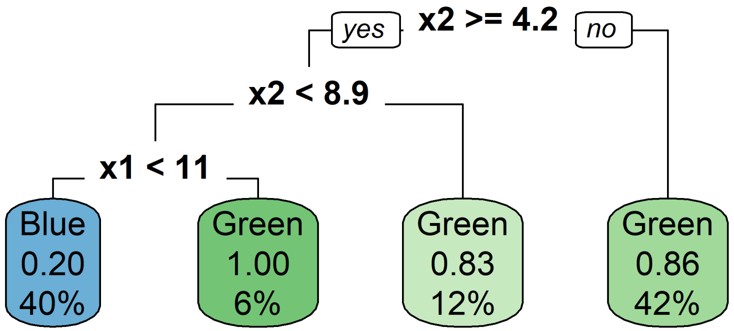
Growing a classification tree IV



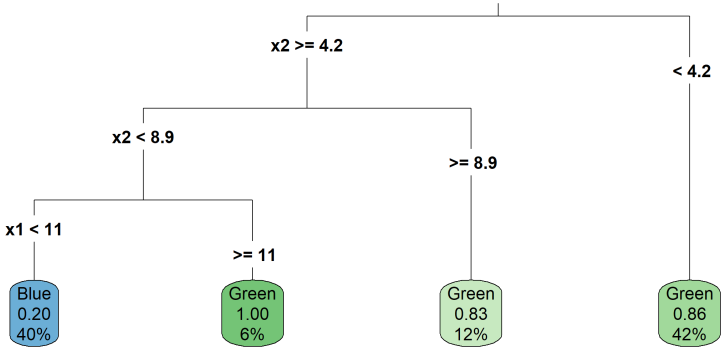


Multiple representations

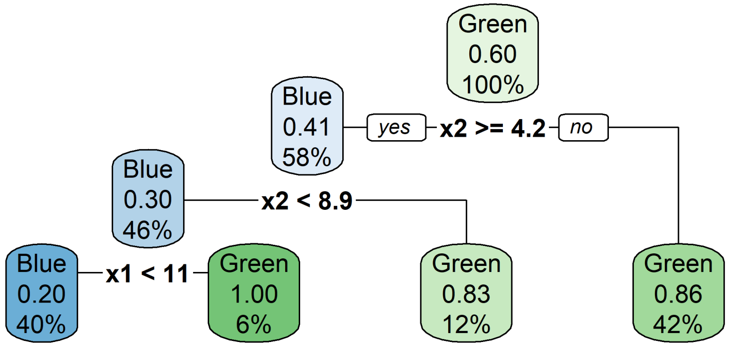
```r
rpart.plot(tree4, type = 0)
```



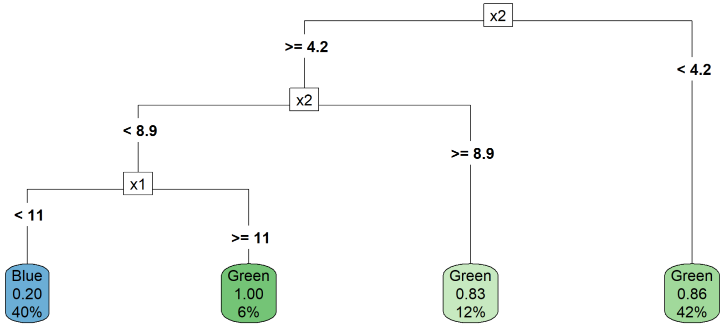

```r
rpart.plot(tree4, type = 3)
```




```r
rpart.plot(tree4, type = 2)
```

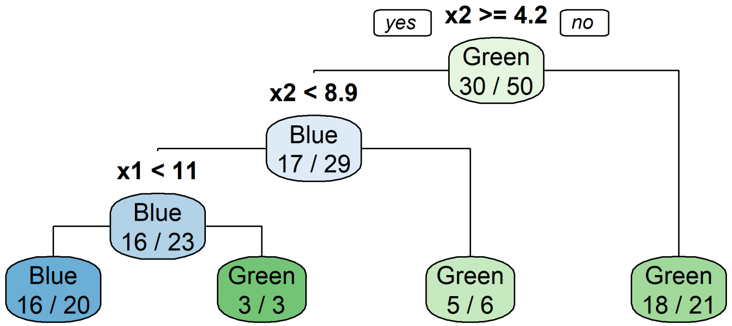



```r
rpart.plot(tree4, type = 5)
```

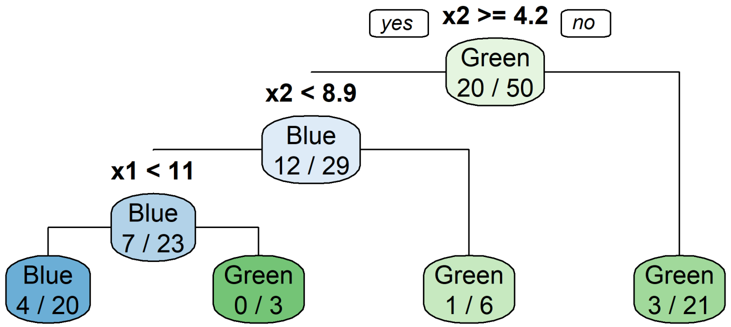


Multiple representations II

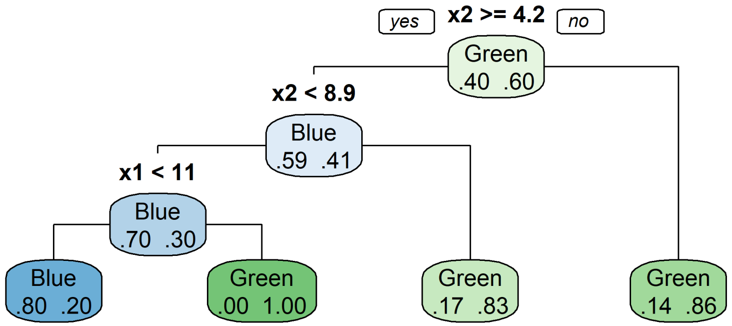
```
rpart.plot(tree4, type = 1, extra = 2)
```



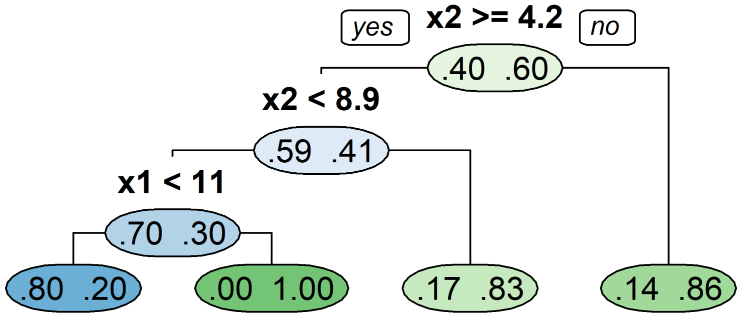

```
rpart.plot(tree4, type = 1, extra = 3)
```




```
rpart.plot(tree4, type = 1, extra = 4)
```




```
rpart.plot(tree4, type = 1, extra = 5)
```



Which one?

> So, should you use Gini impurity or entropy? The truth is, most of the time it does not make a big difference: they lead to similar trees. Gini impurity is slightly faster to compute, so it is a good default. However, when they differ, Gini impurity tends to isolate the most frequent class in its own branch of the tree, while entropy tends to produce slightly more balanced trees.
>
> Footnote: See Sebastian Raschka's interesting analysis for more details.

Source: Géron (2022), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow, 3rd Edition, O'Reilly Media, Chapter 6.

Lecture Outline

- Decision Trees
- Growing a Tree
- **National Flood Insurance Program Demo**
- Pruning a Tree
- Bootstrap Aggregation
- Random Forests
- Boosting

National Flood Insurance Program

Available at OpenFEMA dataset.

```r
claims <- read.csv("FimaNfipClaimsClean.csv")
```

National Flood Insurance Program (NFIP, image source)

See also Zhang and Xu (2023), *Fairness of Ratemaking for Catastrophe Insurance: Lessons from Machine Learning*, Information Systems Research.

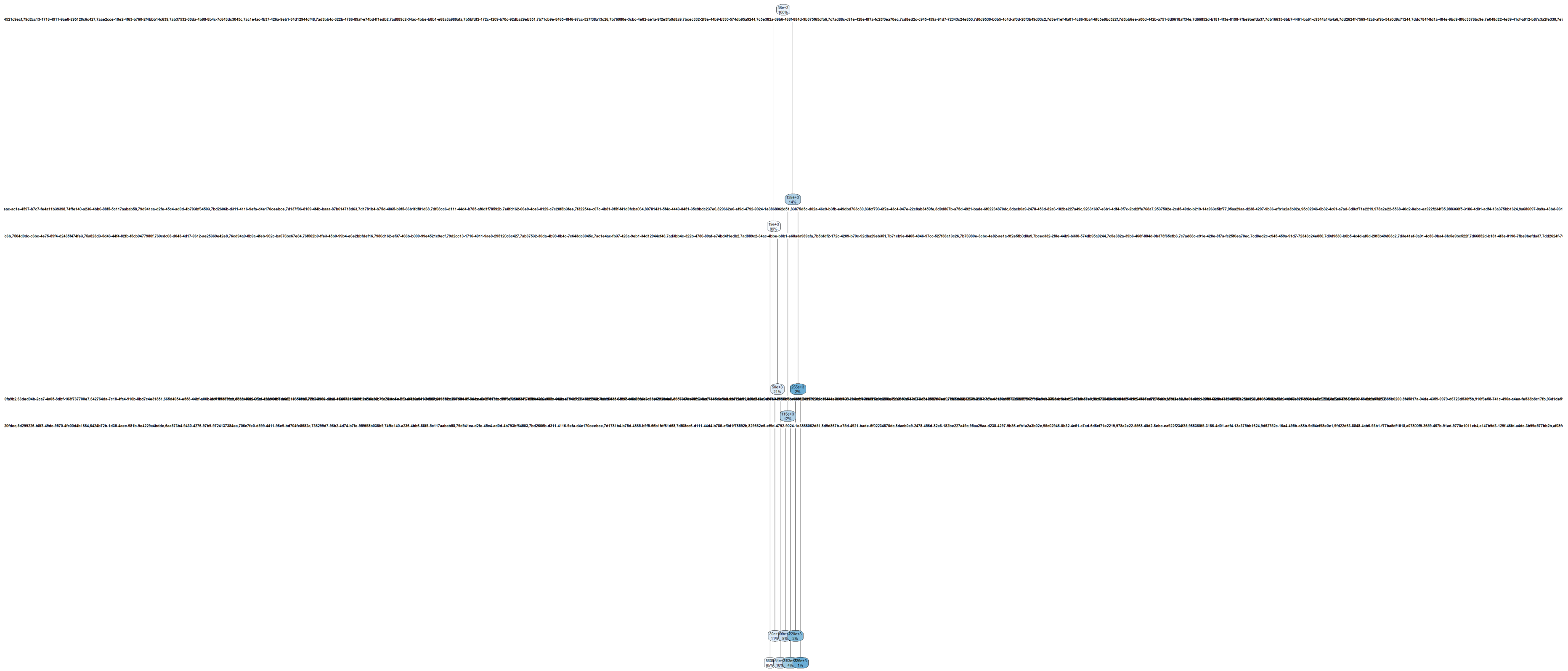
The data dictionary

Name	Title	Type	Description
id	ID	text	Unique ID assigned to the record
amountPaidOnBuildingClaim	Amount Paid on Building Claim	decimal	Dollar amount paid on the building claim. In some instances, a negative amount may appear.
agricultureStructureIndicator	Agriculture Structure Indicator	boolean	Indicates whether a building is reported as being an agricultural structure in the policy application.
policyCount	Policy Count	smallint	Insured units in an active status. A policy contract ceases to be in an active status as of the cancellation date or the expiration date.
countyCode	County Code	text	FIPS code uniquely identifying the primary County (e.g., 011 represents Broward County) associated with the project.
lossDate	Date of Loss	datetime	Date on which water first entered the insured building.
elevatedBuildingIndicator	Elevated Building Indicator	boolean	Indicates whether a building meets the NFIP definition of an elevated building.
latitude	Latitude	decimal	Approximate latitude of the insured building.
locationOfContents	Location of Contents	smallint	Code that indicates the location of contents, (e.g., garage on property, in house).

Name	Title	Type	Description
longitude	Longitude	decimal	Approximate longitude of the insured building.
lowestFloorElevation	Lowest Floor Elevation	decimal	A building's lowest floor is the floor or level that is used as the point of reference when rating a building.
numberOfFloors	Number of Floors	smallint	Code that indicates the number of floors in the insured building.
occupancyType	Occupancy Type	smallint	Code indicating the use and occupancy type of the insured structure.
originalConstructionDate	Original Construction Date	date	The original date of the construction of the building.
originalNBDate	Original NB Date	date	The original date of the flood policy.
postFIRMConstructionIndicator	Post-FIRM Construction Indicator	boolean	Indicates whether construction was started before or after publication of the FIRM.
rateMethod	Rate Method	text	Indicates policy rating method.
state	State	text	The two-character alpha abbreviation of the state in which the insured property is located.
totalBuildingInsuranceCoverage	Total Building Insurance Coverage	integer	Total Insurance Amount in whole dollars on the Building.

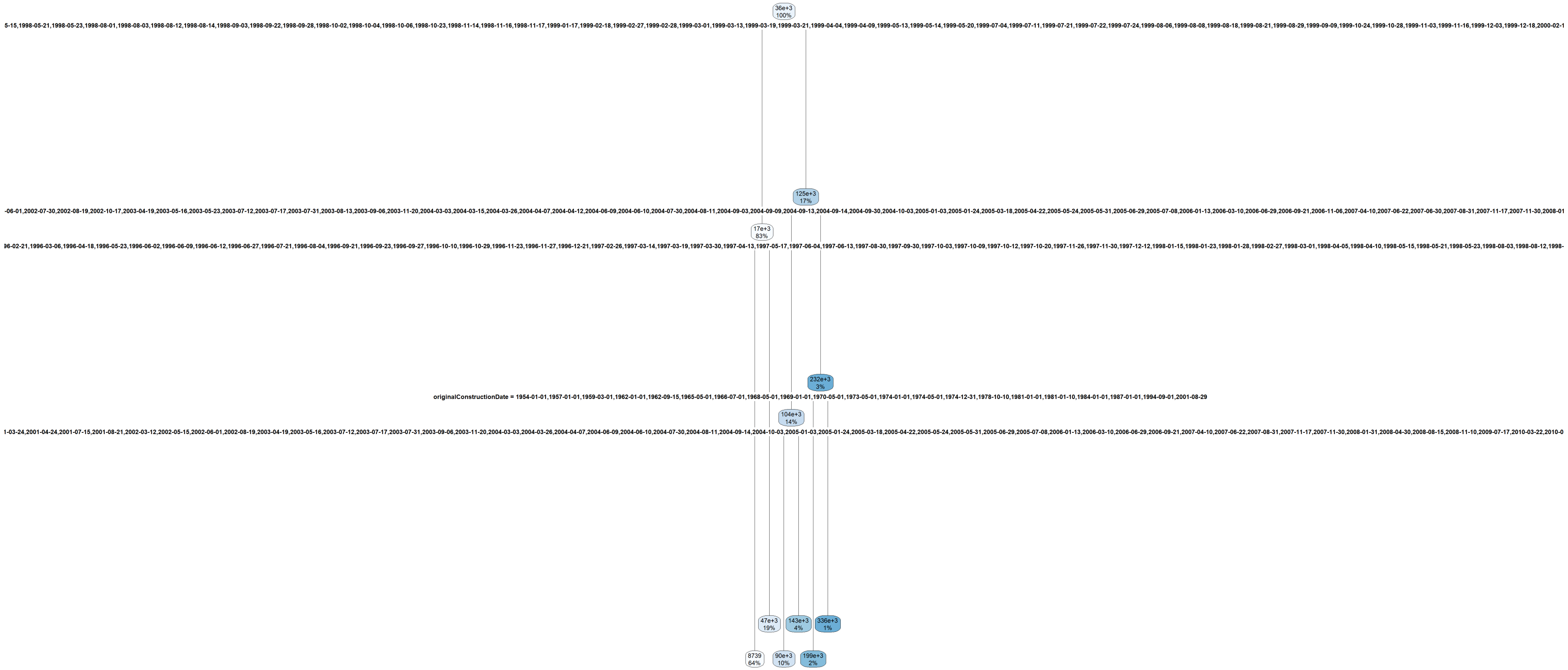
First decision tree

```
tree <- rpart(amountPaidOnBuildingClaim ~ ., data=claims[1:1000,])
rpart.plot(tree)
```



Remove ID column

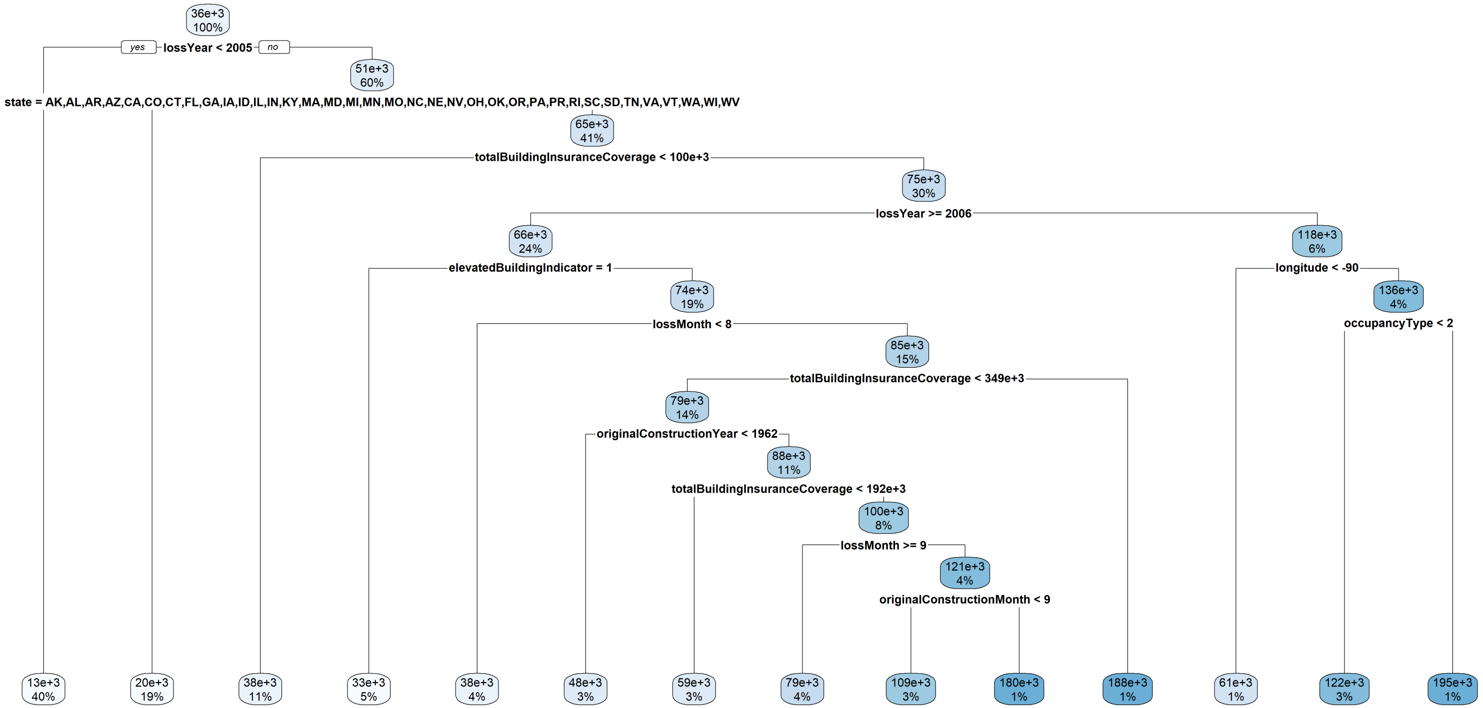
```
claims <- claims %>% select(-id)
tree <- rpart(amountPaidOnBuildingClaim ~ ., data=claims[1:1000,])
rpart.plot(tree)
```



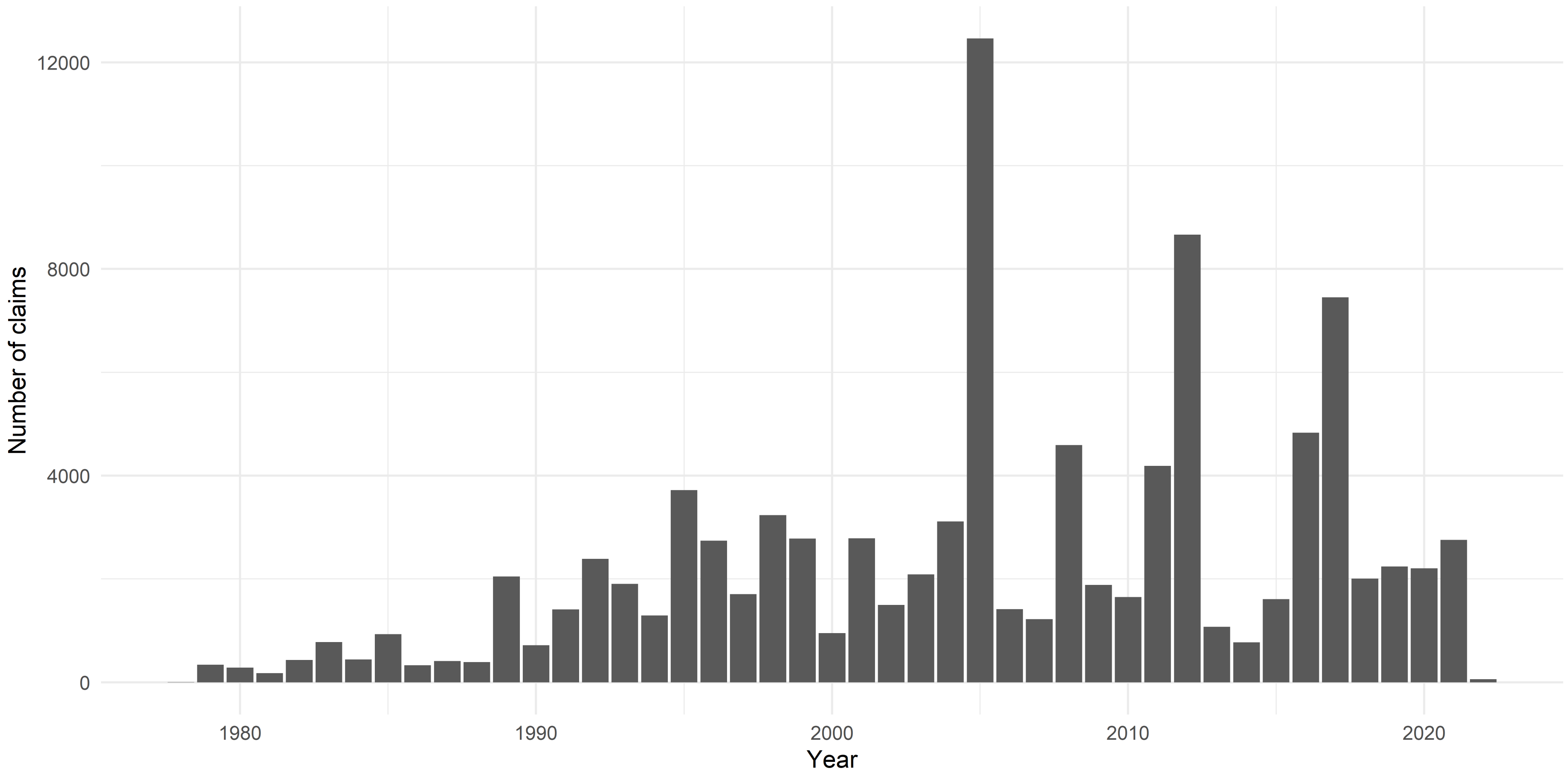
Dates to years and months

```r
claims$lossYear <- year(claims$lossDate) # And so on...
```

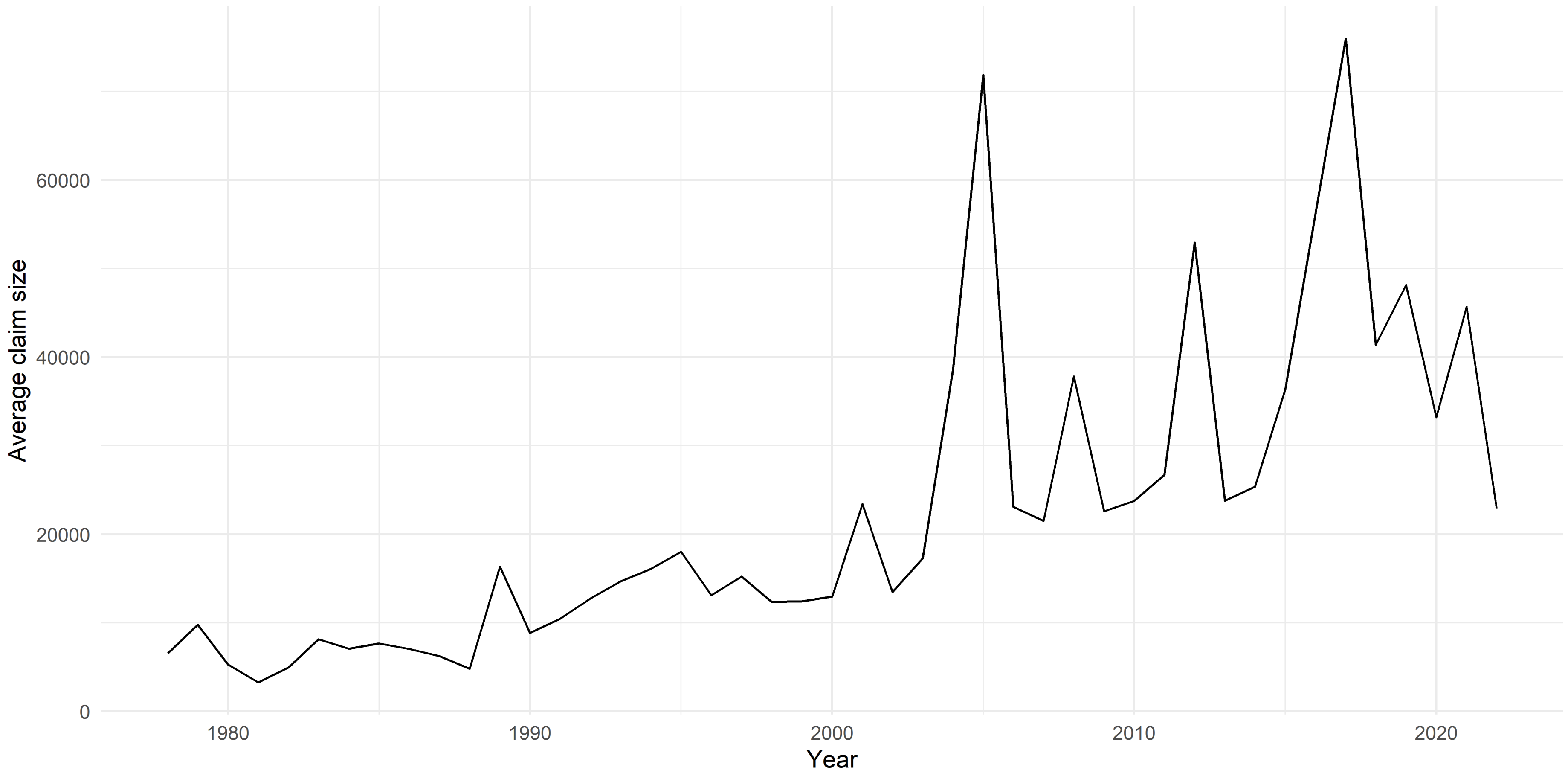
```r
tree <- rpart(amountPaidOnBuildingClaim ~ ., data=claims[1:1000,])
rpart.plot(tree)
```



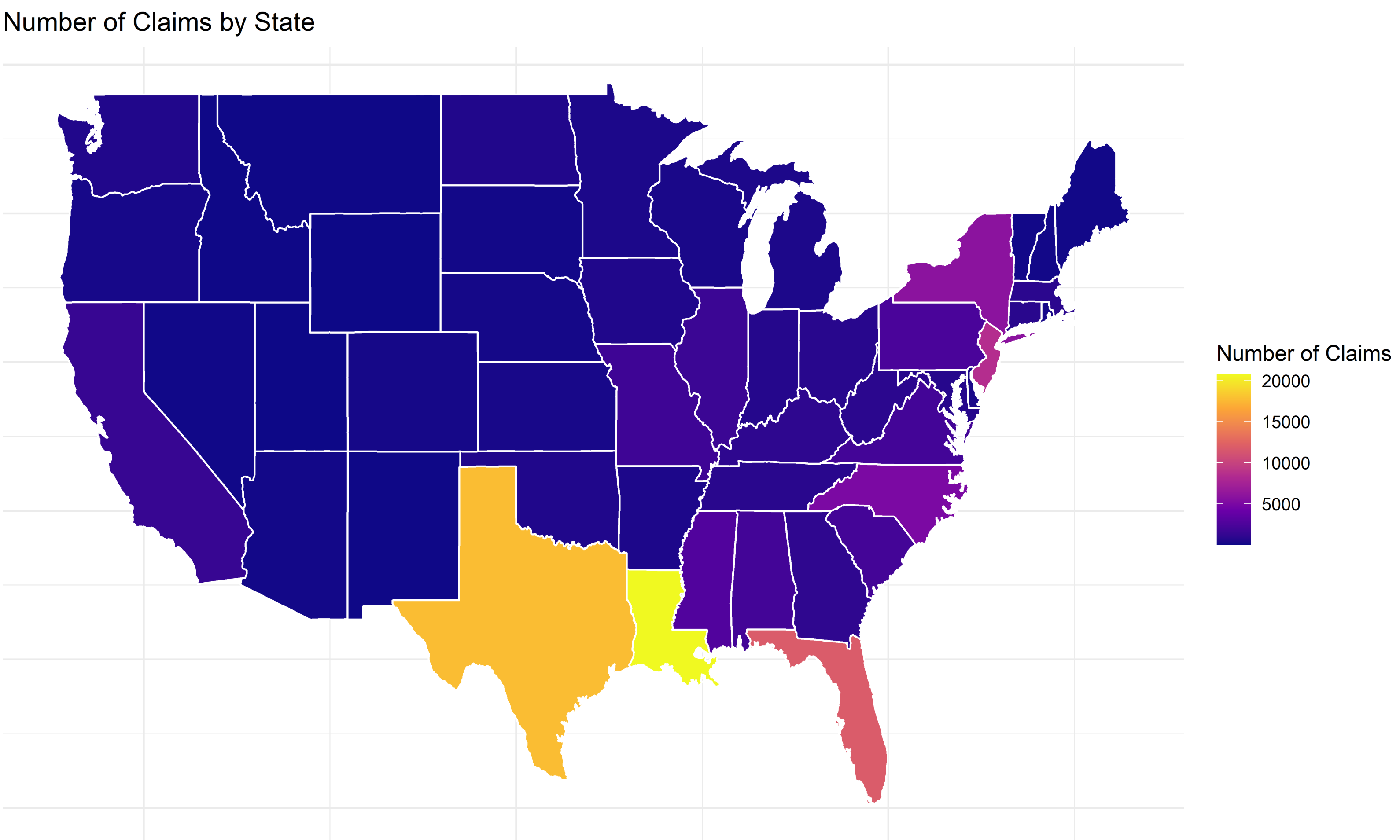
Plot claims by year



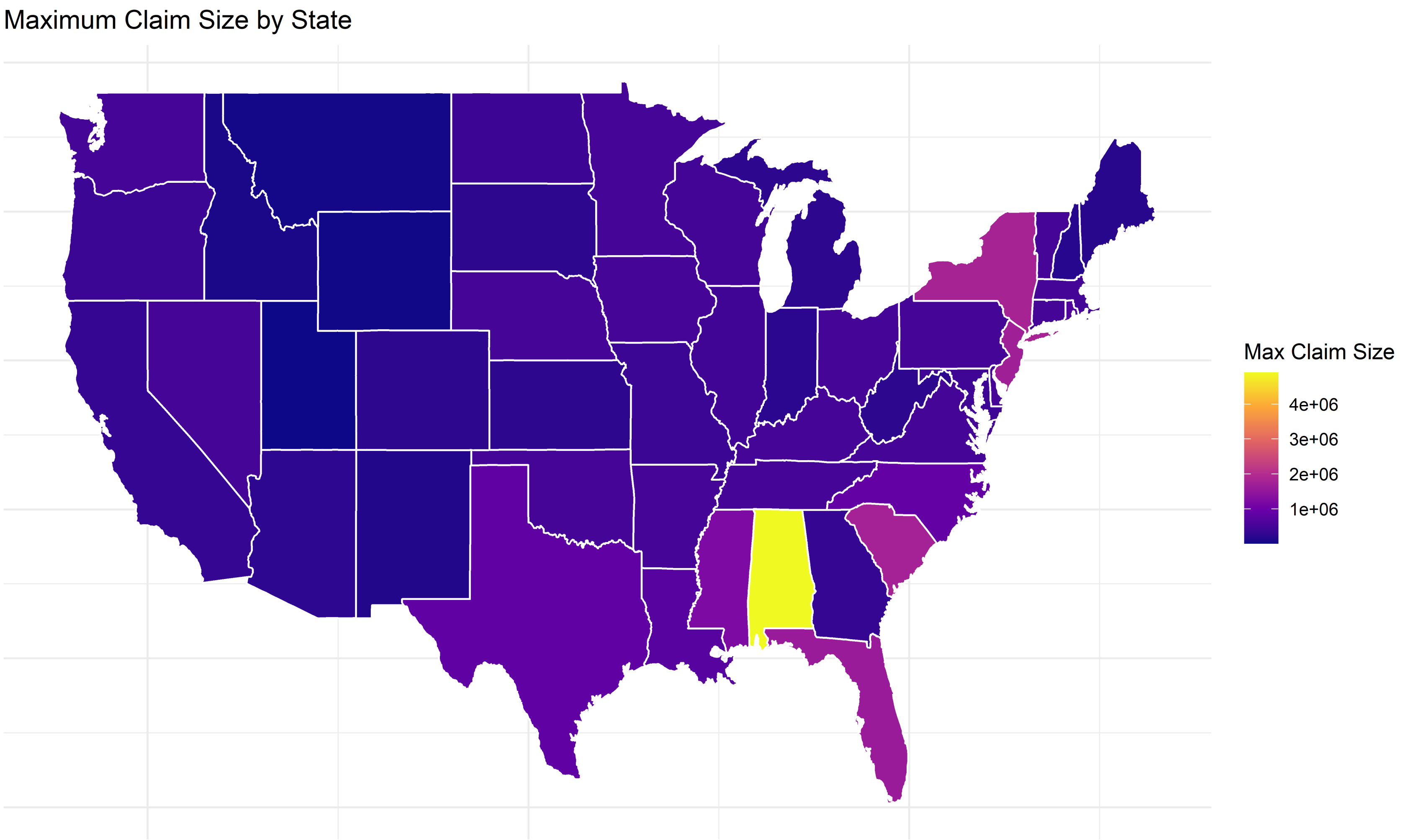
Plot average claim size by year



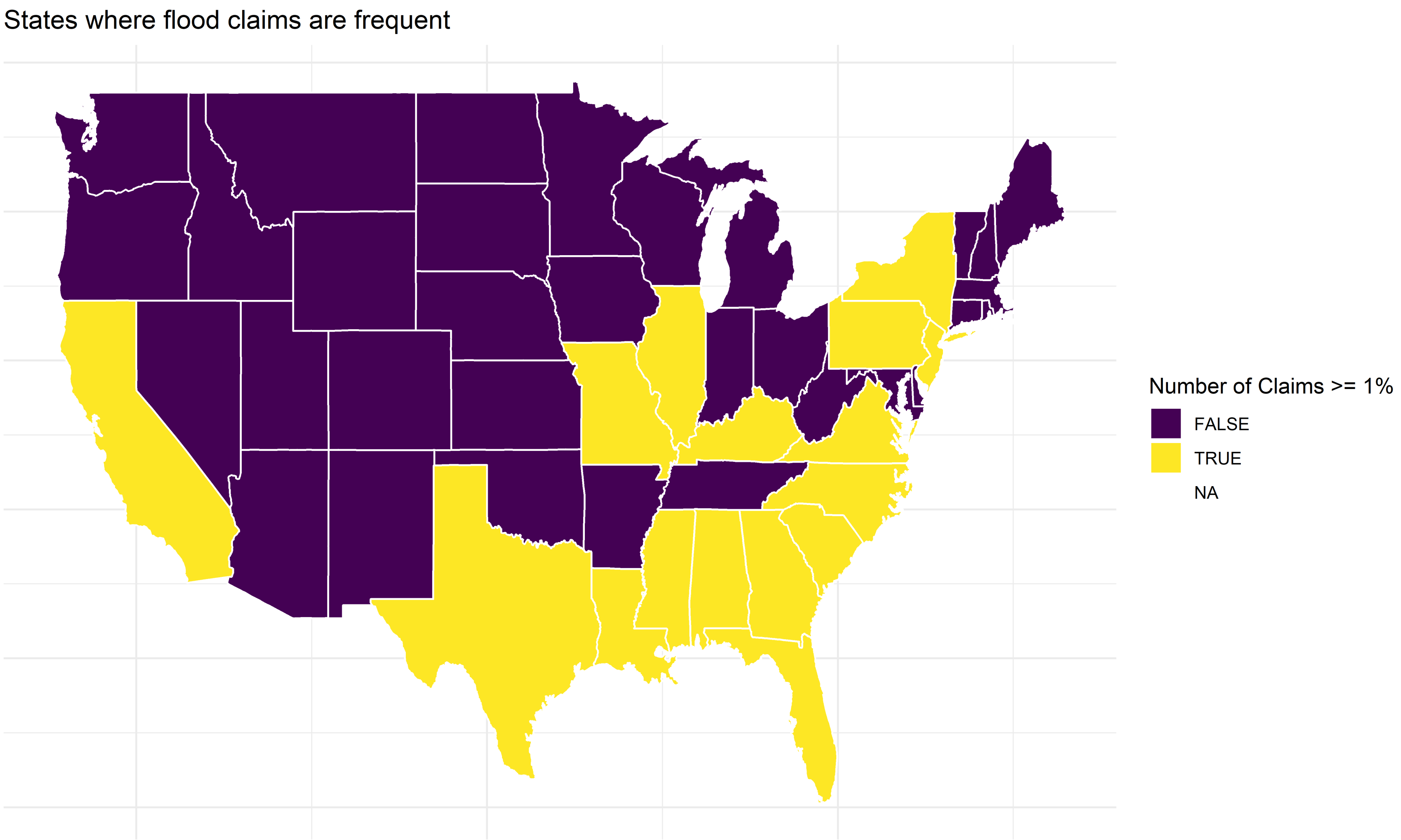
Number of claims by state



Max claim size by state



Some states have very few claims



Geographical distribution of perils

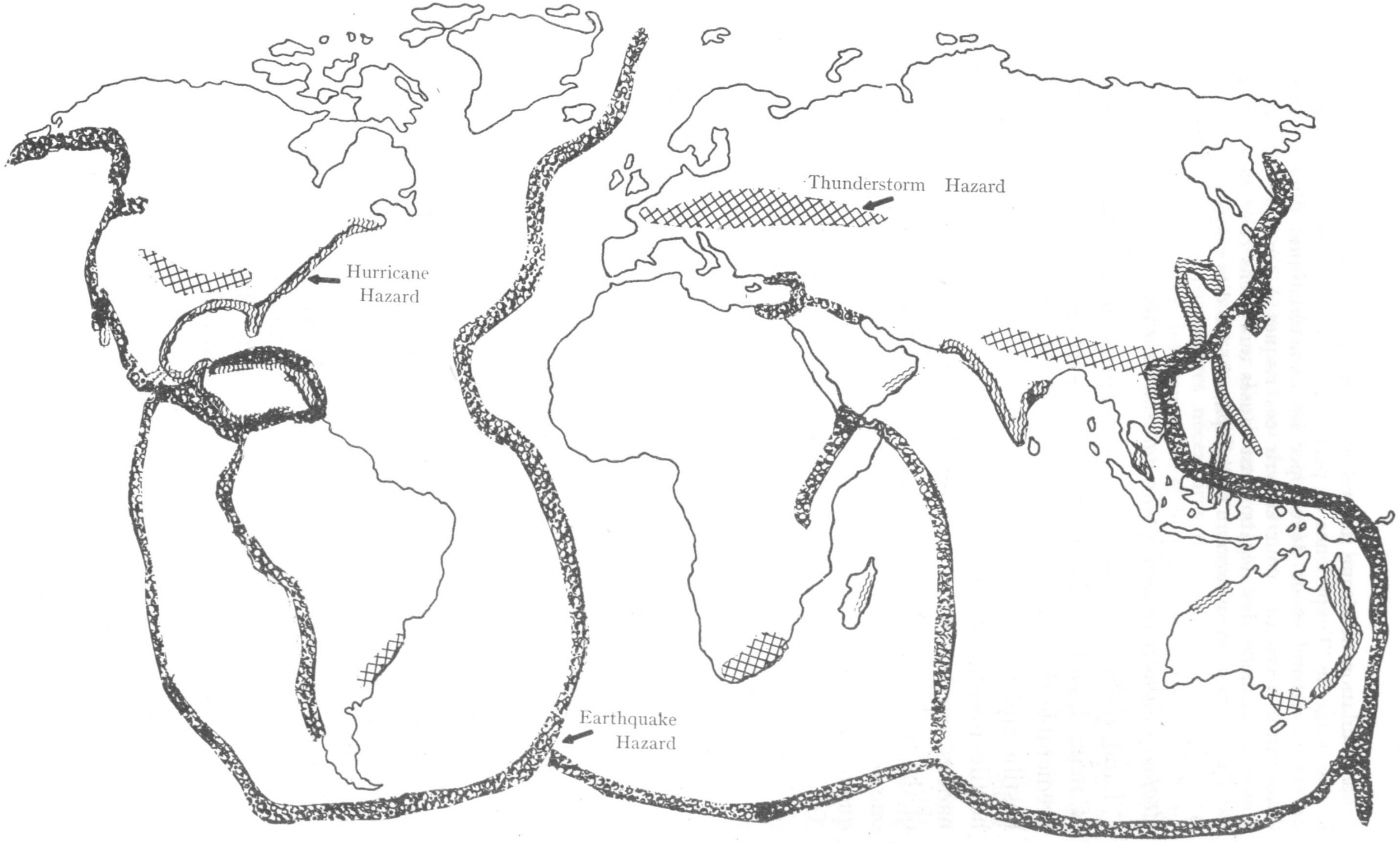


Exhibit 1. Areas prone to the destructive forces of Thunderstorms, Hurricanes, and Earthquakes.

Friedman Exhibit 1 (p. 10).

Source: Friedman, D. G. (1972), *Insurance and the natural hazards*. ASTIN Bulletin: The Journal of the IAA, 7(1), 4-58.

Hot spots

Exhibit 13. Observed and calculated geographical pattern of highest wind associated with movement of Hurricane Flossy along the Gulf Coast in 1956.

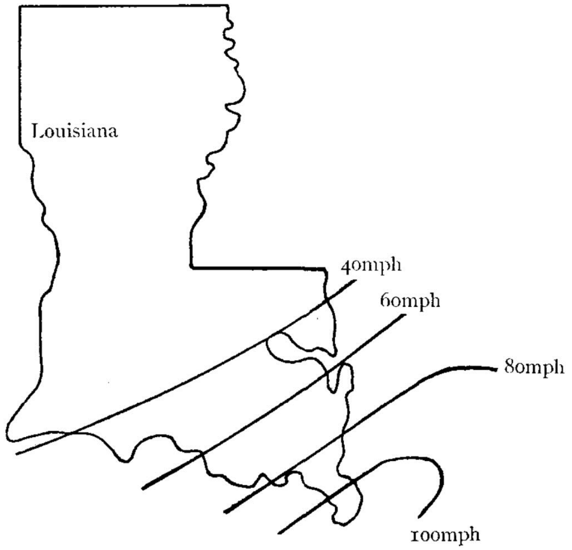


Exhibit 13a. Observed pattern of peak wind gust based upon observations tabulated in the U.S. Department of Commerce publication *Climatological Data Annual issue*—1956 (Vol. 7, No. 13) Superintendent of Documents, Washington, D.C.

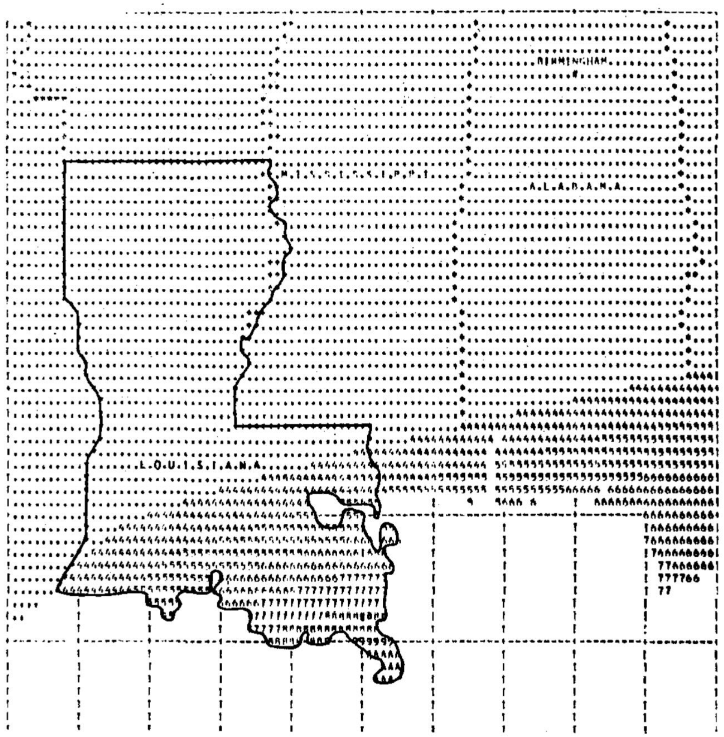


Exhibit 13b. Computed pattern of highest winds. The digit given at each grid point in the affected area represents a wind speed interval. For instance, the digit 5 denotes a wind speed between 50 to 59 miles per hour. For speed intervals above 100 miles per hour, an alphabetic designation is used. The letter A represents the interval from 100 to 109 miles per hour. The State of Louisiana has been outlined on the printout.

Friedman Exhibit 13 (p. 46).

Source: Friedman, D. G. (1972), *Insurance and the natural hazards*. ASTIN Bulletin: The Journal of the IAA, 7(1), 4-58.

Reduce the number of levels

```
table(claims$state)
```

```
   AK    AL    AR    AZ    CA    CO    CT    DC    DE    FL    GA    GU    HI 
   27  2019   410   139  1408   191   819    10   303 11833  1035     6   152 
   IA    ID    IL    IN    KS    KY    LA    MA    MD    ME    MI    MN    MO 
  504    43  1605   662   241  1012 20785   885   696   115   402   368  1752 
   MS    MT    NC    ND    NE    NH    NJ    NM    NV    NY    OH    OK    OR 
 2753    53  5023   513   189   142  8520    48    81  5899   790   490   237 
   PA    PR    RI    SC    SD    TN    TX    UN    UT    VA    VI    VT    WA 
 2363   569   238  2023   136   809 17759    10    11  2006    83   108   522 
   WI    WV    WY 
  313   874    16 
```

```
length(unique(claims$state))
```

```
[1] 55
```

```
# States with fewer than 1% claims
rare_flood_states <- names(which(table(claims[["state"]]) < nrow(claims) / 100))
claims$state <- ifelse(claims$state %in% rare_flood_states, "Other", claims$state)

table(claims$state)
```

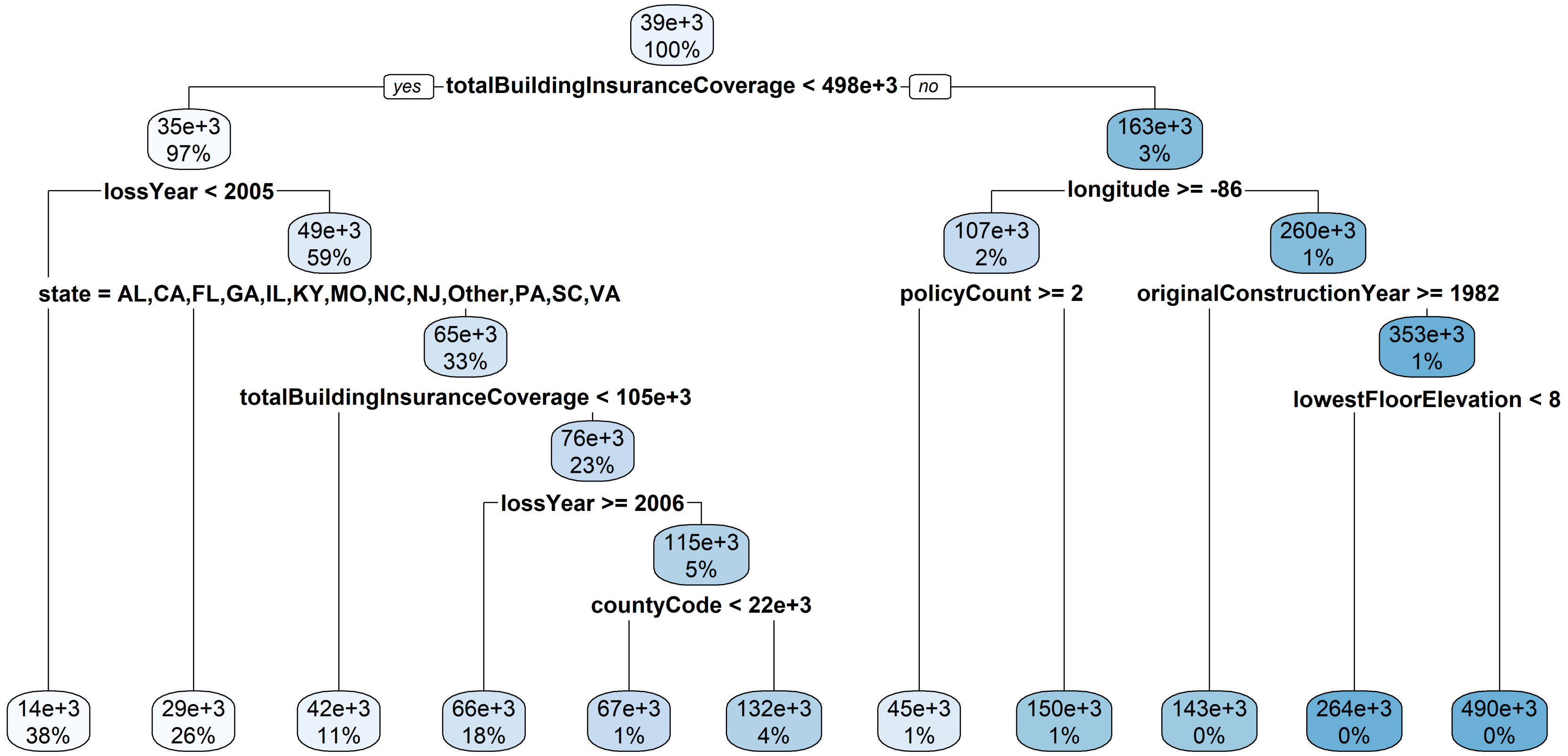
```
   AL    CA    FL    GA    IL    KY    LA    MO    MS    NC    NJ    NY Other 
 2019  1408 11833  1035  1605  1012 20785  1752  2753  5023  8520  5899 12205 
   PA    SC    TX    VA 
 2363  2023 17759  2006 
```

```
length(unique(claims$state))
```

```
[1] 17
```

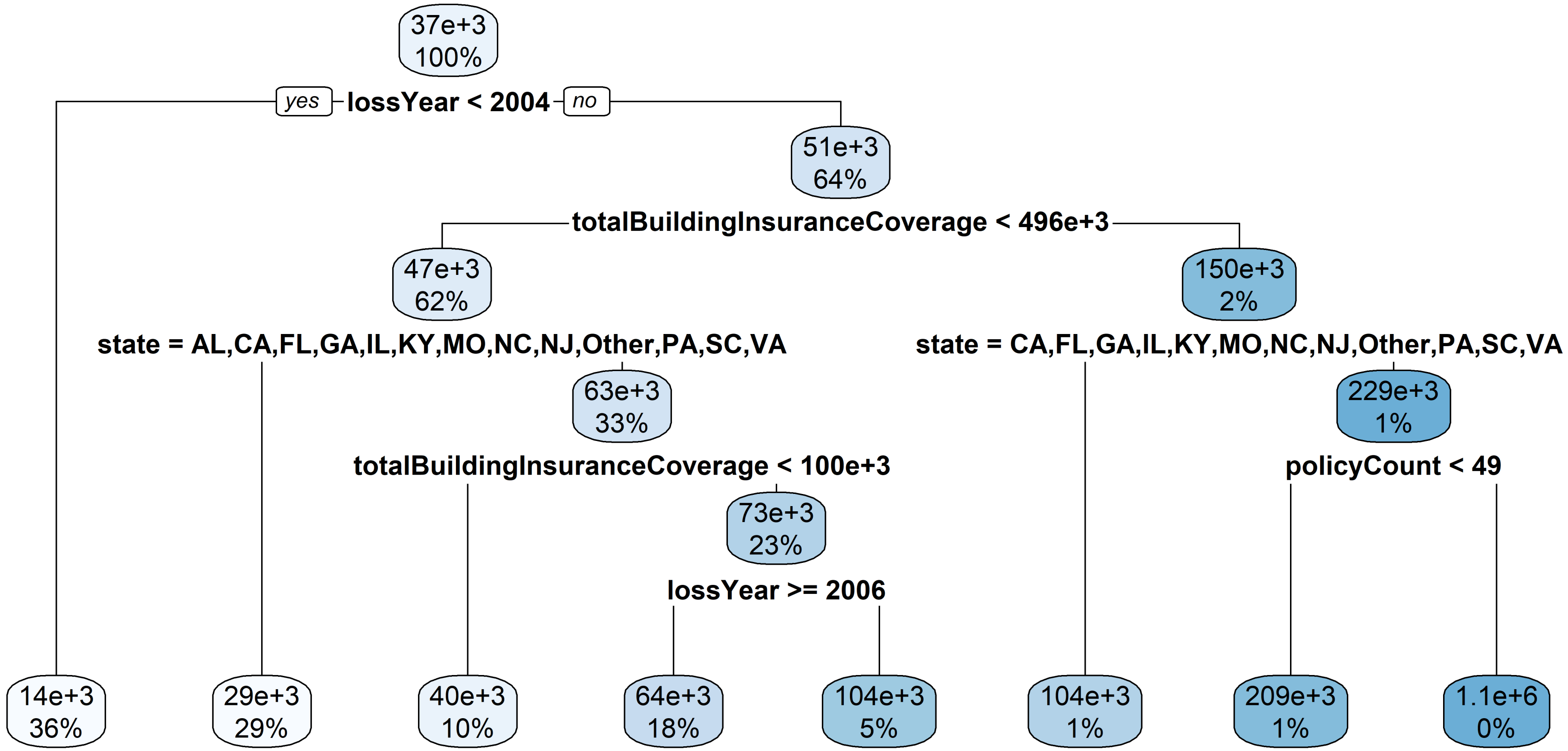
New tree

```r
tree <- rpart(amountPaidOnBuildingClaim ~ ., data=claims[1:5000,])
rpart.plot(tree)
```



More data

```r
tree <- rpart(amountPaidOnBuildingClaim ~ ., data=claims[1:50000,])
rpart.plot(tree)
```



Lecture Outline

- Decision Trees
- Growing a Tree
- National Flood Insurance Program Demo
- **Pruning a Tree**
- Bootstrap Aggregation
- Random Forests
- Boosting

What's the best size of tree

The smallest tree is just a root node (no splits).

The upper limit is to grow until one observation in each region.

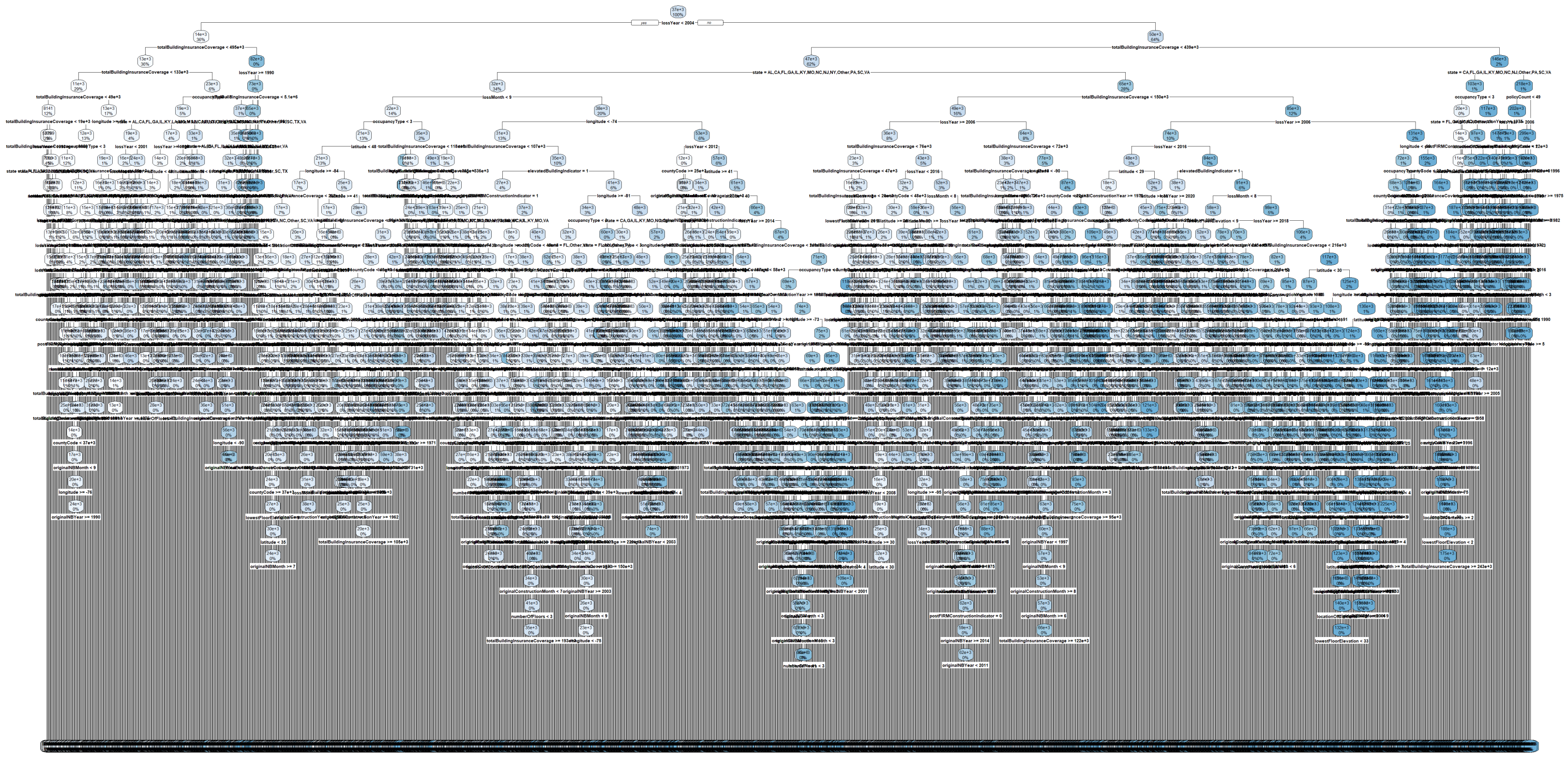
How large should we grow the tree?

- What's wrong if the tree is too small?
- What's wrong if the tree is too large?

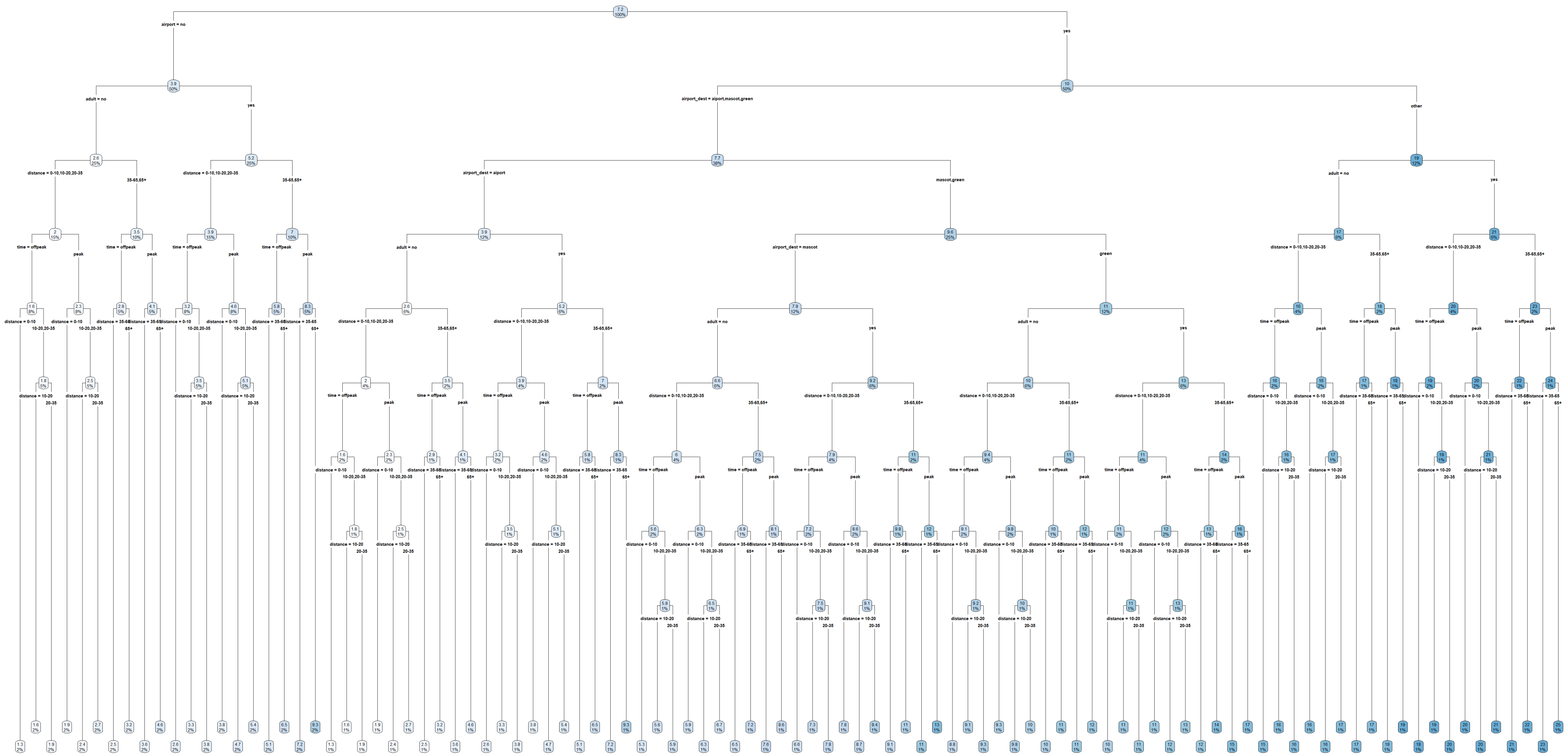
Pruning (stock photo)

A large tree for the flood insurance data

```r
large_tree <- rpart(amountPaidOnBuildingClaim ~ ., data=train_set, control=rpart.control(cp=0.00001))
rpart.plot(large_tree)
```



The full tree for train pricing



Early stopping of training

> "In order to reduce the size of the tree and hence to prevent overfitting, these stopping criteria that are inherent to the recursive partitioning procedure are complemented with several rules. Three stopping rules that are commonly used can be formulated as follows:
>
> - A node t is declared terminal when it contains less than a fixed number of observations.
> - A node t is declared terminal if at least one of its children nodes t_L and t_R that results from the optimal split s_t contains less than a fixed number of observations.
> - A node t is declared terminal when its depth is equal to a fixed maximal depth."

Source: Denuit et al. (2020), *Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions*, Springer, p. 58.

Pruning motivation

> "While the stopping rules presented above may give good results in practice, the strategy of stopping early the growing of the tree is in general unsatisfactory… That is why it is preferable to prune the tree instead of stopping the growing of the tree. Pruning a tree consists in fully developing the tree and then prune it upward until the optimal tree is found."

- A decision rule of considering the decrease in RSS at each step/split (versus a threshold) is too short-sighted.
- Alternate approach of growing a large tree then pruning back to obtain a subtree is a better strategy.
- Cross validation of each possible subtree is however very cumbersome.
- An alternative approach is cost complexity pruning (also known as weakest link pruning)

Source: Denuit et al. (2020), *Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions*, Springer, p. 59.

Cost-Complexity Pruning

Define a subtree $T \subset T_0$ to be any tree than can be obtained by pruning T_0 (a fully-grown tree)

- Terminal node m represents region R_m
- $|T|$: number of terminal nodes in T

Define the cost complexity criterion

$$\text{Total cost} = \text{Measure of Fit} + \text{Measure of Complexity}$$

$$C_\alpha(T) = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_m)^2 + \alpha|T|$$

where $\hat{y}_m$ is the mean y_i in the mth leaf and α controls the tradeoff between tree size and goodness of fit.

Cost-Complexity Pruning

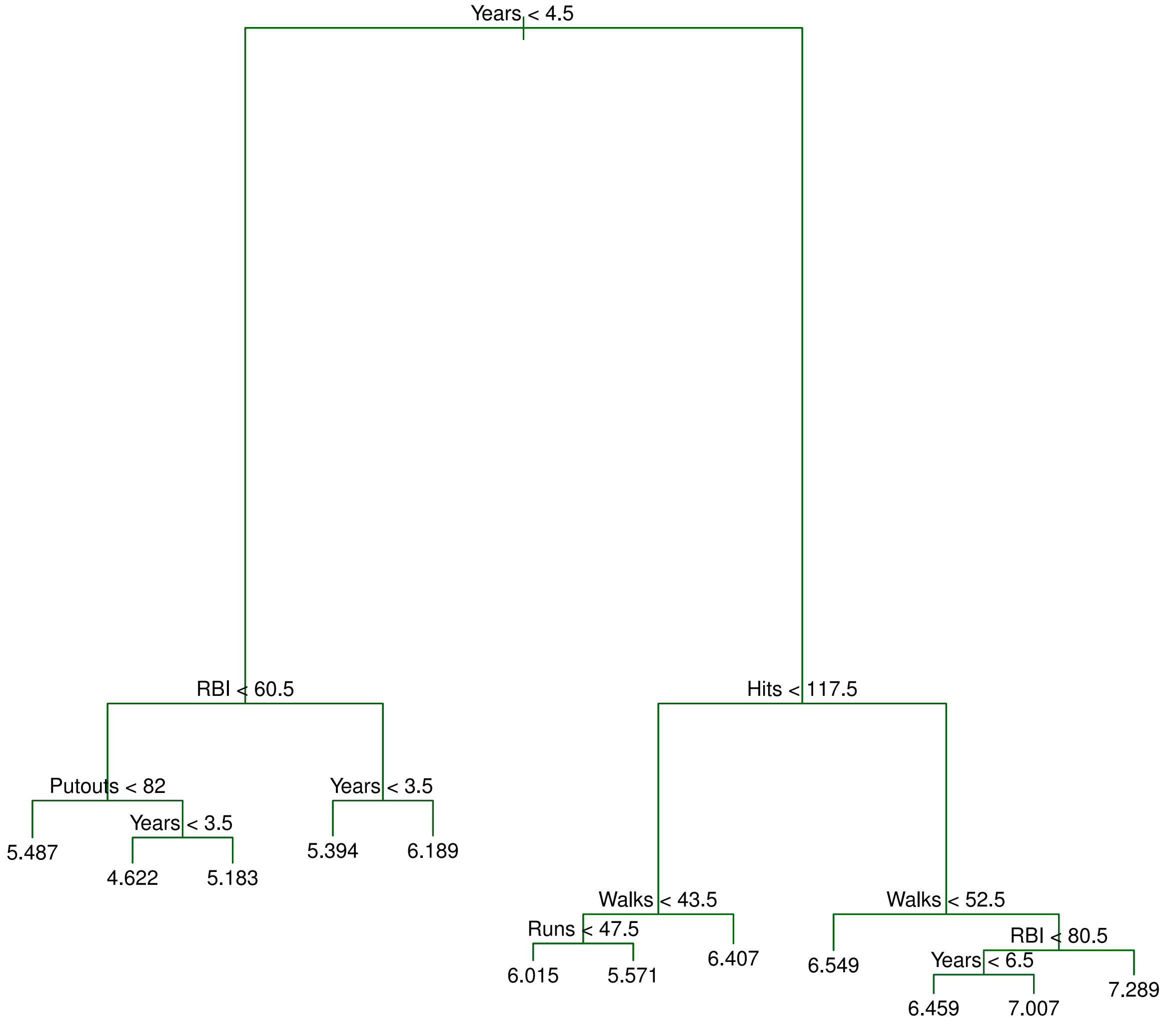
For each α, we want to find the subtree $T_\alpha \subseteq T_0$ that minimises $C_\alpha(T)$

- How to find T_α?
  - "weakest link pruning"
    - For a particular α, find the subtree T_α such that the cost complexity criterion is minimised
- How to choose α?
  - cross-validation

Tree Algorithm Summary

1. Use recursive binary splitting to grow a large tree on the training data
   - stop only when each terminal node has fewer than some minimum number of observations
2. Apply cost complexity pruning to the large tree to obtain a sequence of best subtrees, as a function of α
   - there is a unique smallest subtree T_α that minimises $C_\alpha(T)$
3. Use K-fold cross-validation to choose α
4. Return the subtree from Step 2 that corresponds to the chosen value of α

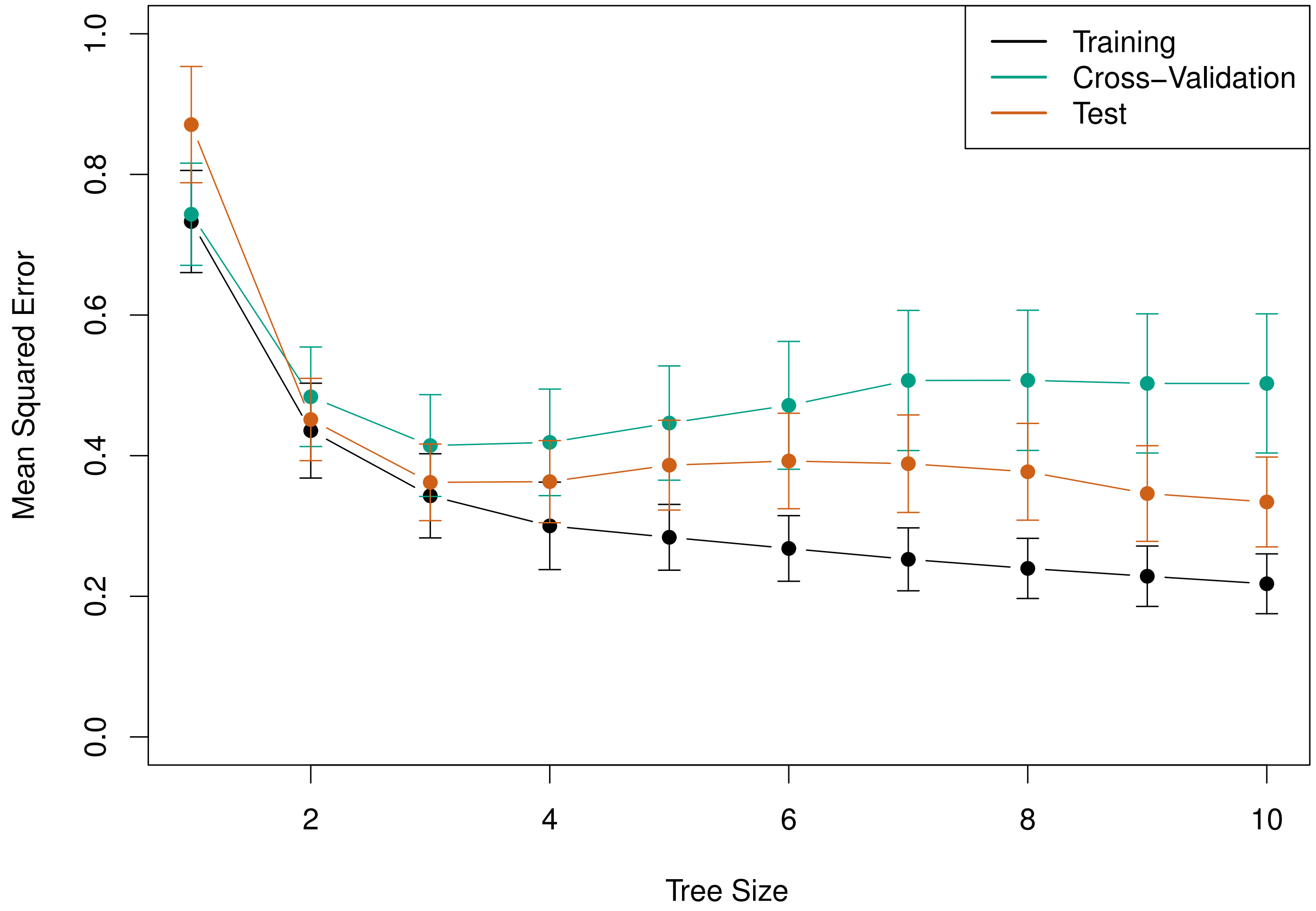
Unpruned `Hitters` tree



The unpruned tree that results from top-down greedy splitting on the training data.

Source: James et al. (2021), An Introduction to Statistical Learning, Figure 8.4.

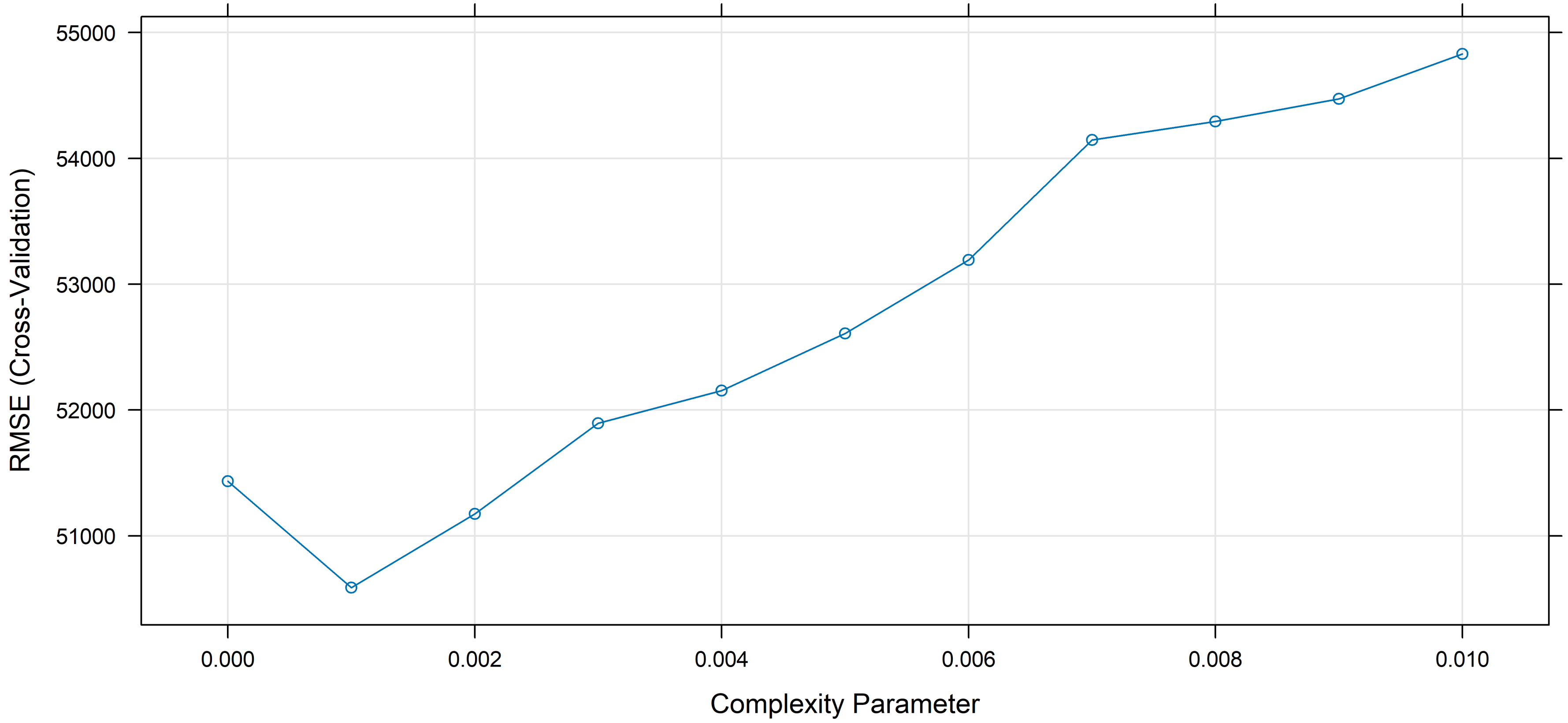
CV to pick α (equiv., $|T|$)



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 of size three.

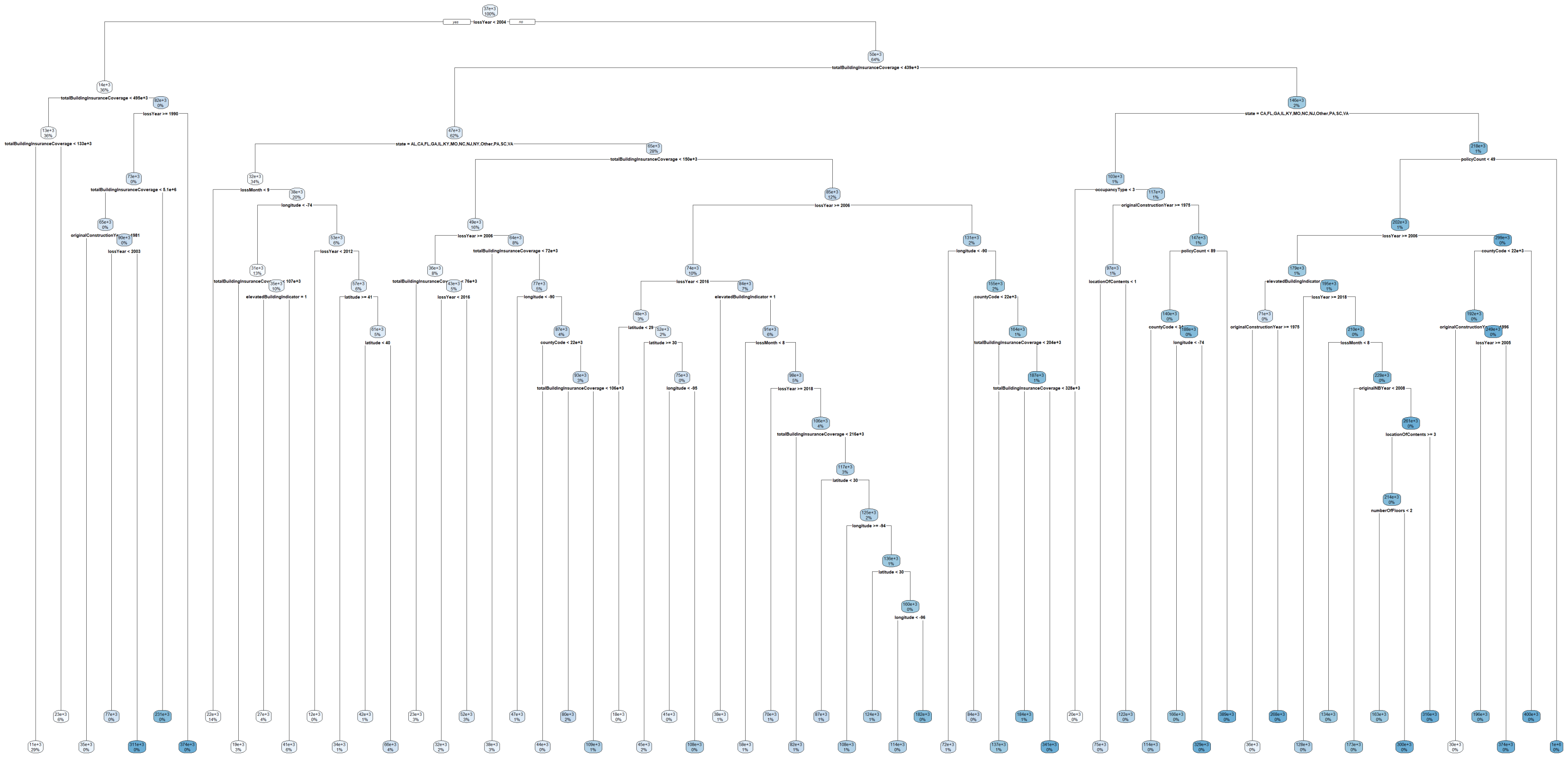
Source: James et al. (2021), An Introduction to Statistical Learning, Figure 8.5.

CV to prune NFIP tree



The pruned tree

```r
pruned_tree <- prune(large_tree, cp=optimal_cp)
rpart.plot(pruned_tree)
```



Linear model

```r
linear <- lm(amountPaidOnBuildingClaim ~ ., data=train_set)
summary(linear)
```

```
Call:
lm(formula = amountPaidOnBuildingClaim ~ ., data = train_set)

Residuals:
    Min      1Q  Median      3Q     Max 
-785390  -27448  -10894   11246 4787232 

Coefficients:
                                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)                    -2.793e+06  6.662e+04 -41.928  < 2e-16 ***
agricultureStructureIndicator   1.662e+04  1.669e+04   0.996 0.319163    
policyCount                     1.463e+03  1.179e+02  12.407  < 2e-16 ***
countyCode                      6.958e-02  3.986e-02   1.746 0.080900 .  
elevatedBuildingIndicator      -1.387e+04  6.202e+02 -22.358  < 2e-16 ***
latitude                        3.955e+02  1.031e+02   3.835 0.000125 ***
locationOfContents              4.392e+02  1.412e+02   3.111 0.001864 ** 
longitude                      -1.455e+02  3.953e+01  -3.681 0.000232 ***
lowestFloorElevation           -4.797e-02  5.281e-01  -0.091 0.927623    
occupancyType                   3.937e+03  1.681e+02  23.413  < 2e-16 ***
postFIRMConstructionIndicator   3.291e+03  7.311e+02   4.501 6.78e-06 ***
stateCA                        -8.250e+03  2.975e+03  -2.773 0.005548 ** 
```

Comparing models

Method	**RMSE**
Linear Model	5.4792406^{4}
Large Tree	4.8683476^{4}
Pruned Tree	4.7371652^{4}

Lecture Outline

- Decision Trees
- Growing a Tree
- National Flood Insurance Program Demo
- Pruning a Tree
- **Bootstrap Aggregation**
- Random Forests
- Boosting

Advantages and disadvantages of trees

Advantages

- Easy to explain
- (Mirror human decision making)
- Graphical display
- Easily handle qualitative predictors

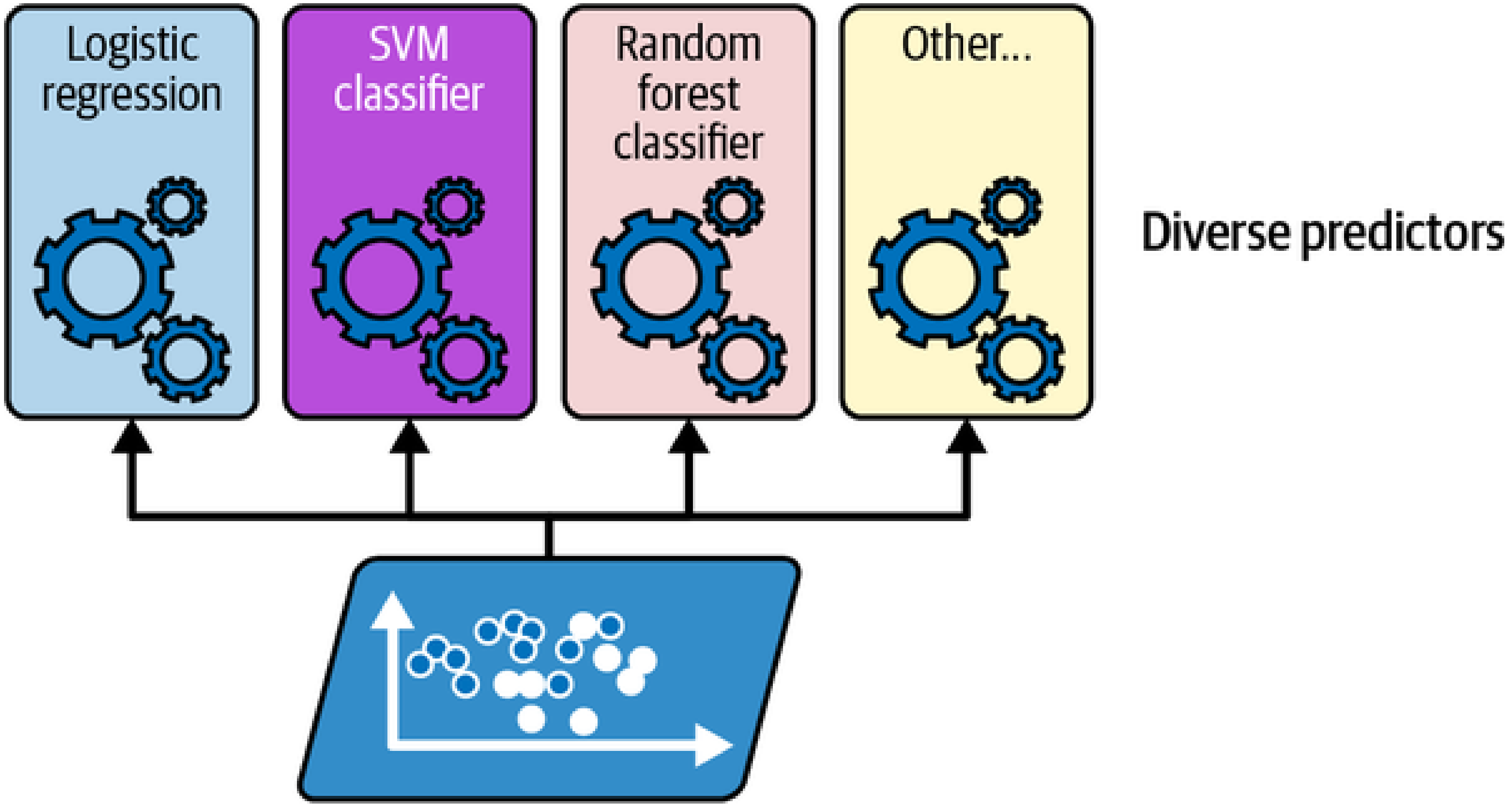
Disadvantages

- Low predictive accuracy compared to other regression and classification approaches
- Can be very non-robust

Is there a way to improve the predictive performance of trees?

- Pruning a decision tree
- Ensemble methods
- Bagging, random forest, boosting

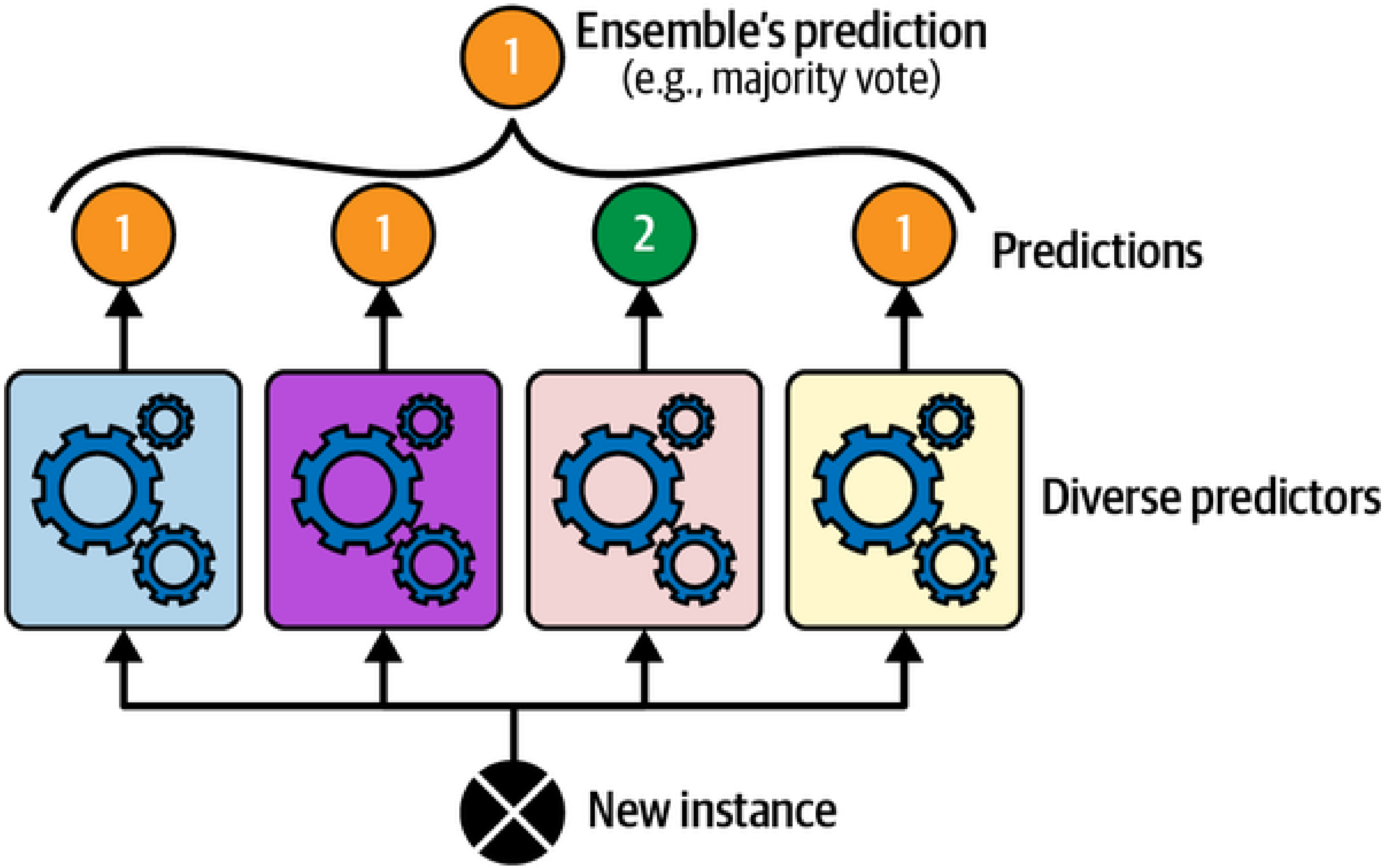
An ensemble is a group of models…



Training various different classifiers on the same dataset.

Source: Geron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd ed., Figure 7-1.

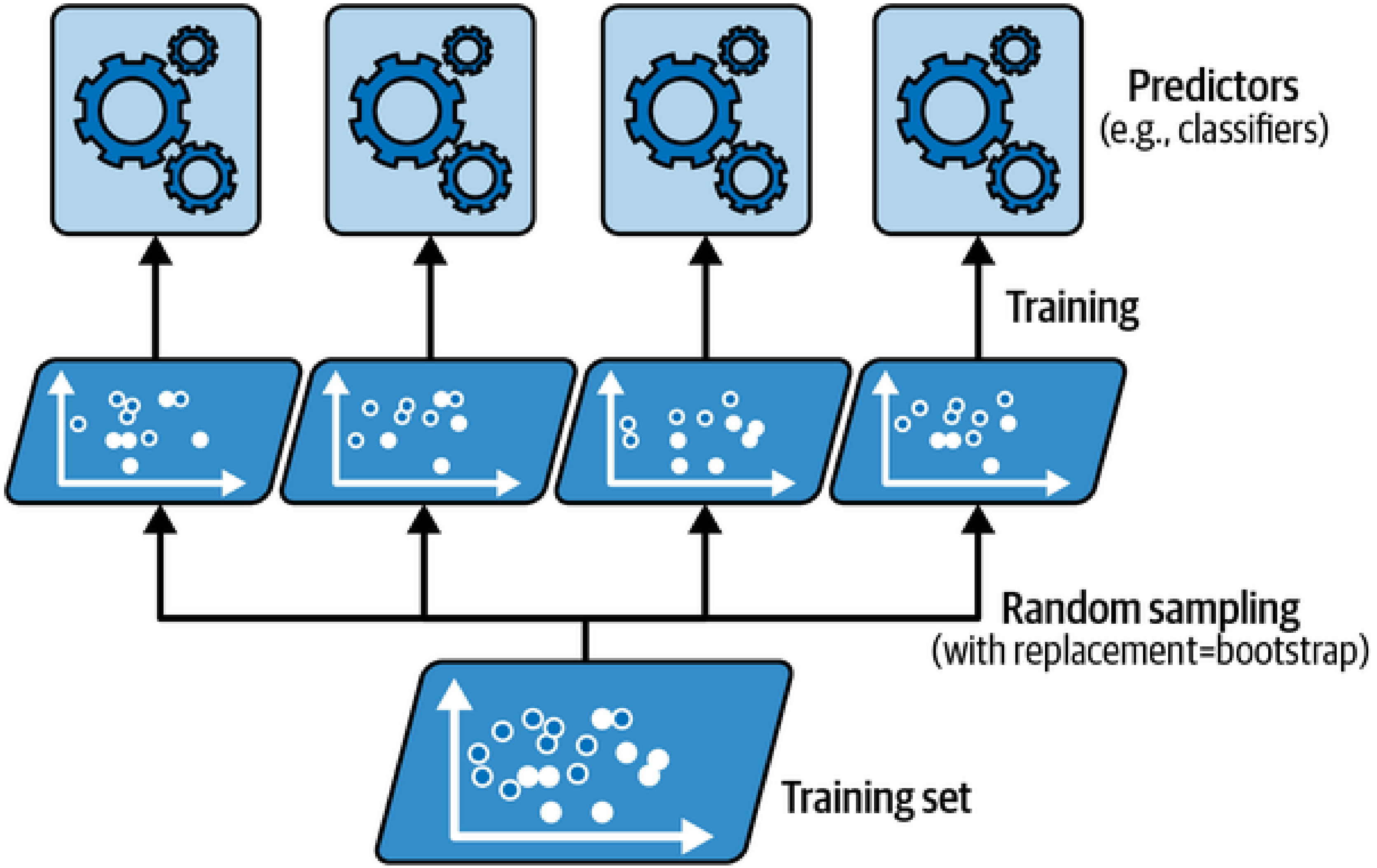
… & you combine their predictions



Make an overall prediction based on the majority vote of the models.

Source: Geron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd ed., Figure 7-2.

Bootstrapping



Train on different versions of the same data.

Source: Geron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd ed., Figure 7-4.

Bootstrap resampling I

Original dataset

```r
# Sort by first column to make
# it easier to see the resampling
# (so not necessary in general).
df %>% arrange(x1)
```

x1 <dbl>	x2 <dbl>
1.032286	6.7859712
1.084090	7.6180446
1.616814	5.2788242
1.793994	2.4958501
1.851577	8.4238348
1.900143	10.1700020
2.044791	1.9265906
2.317750	1.7829508
2.501149	2.0640931
2.571976	2.0820747

1-10 of … Previous **1** 2 3 4 5 Next

A bootstrap resample

```r
set.seed(1)
df %>%
  sample_n(size=nrow(df), replace=TRUE) %>%
  arrange(x1)
```

x1 <dbl>	x2 <dbl>
1.032286	6.7859712
1.032286	6.7859712
1.032286	6.7859712
1.032286	6.7859712
1.084090	7.6180446
1.793994	2.4958501
1.793994	2.4958501
1.793994	2.4958501
1.851577	8.4238348
1.900143	10.1700020

1-10 of … Previous **1** 2 3 4 5 Next

There are 54% of the rows in the original dataset in the bootstrap resample.

Bootstrap resampling II

Original dataset

```
# Sort by first column to make
# it easier to see the resampling
# (so not necessary in general).
df %>% arrange(x1)
```

x1 <dbl>	x2 <dbl>
1.032286	6.7859712
1.084090	7.6180446
1.616814	5.2788242
1.793994	2.4958501
1.851577	8.4238348
1.900143	10.1700020
2.044791	1.9265906
2.317750	1.7829508
2.501149	2.0640931
2.571976	2.0820747

1-10 of … Previous 1 2 3 4 5 Next

A bootstrap resample

```
set.seed(4)
df %>%
  sample_n(size=nrow(df), replace=TRUE) %>%
  arrange(x1)
```

x1 <dbl>	x2 <dbl>
1.032286	6.7859712
1.616814	5.2788242
1.616814	5.2788242
1.616814	5.2788242
1.851577	8.4238348
1.900143	10.1700020
2.044791	1.9265906
2.317750	1.7829508
2.317750	1.7829508
2.571976	2.0820747

1-10 of … Previous 1 2 3 4 5 Next

There are 68% of the rows in the original dataset in the bootstrap resample.

UNSW SYDNEY

Bootstrap Aggregation (Bagging)

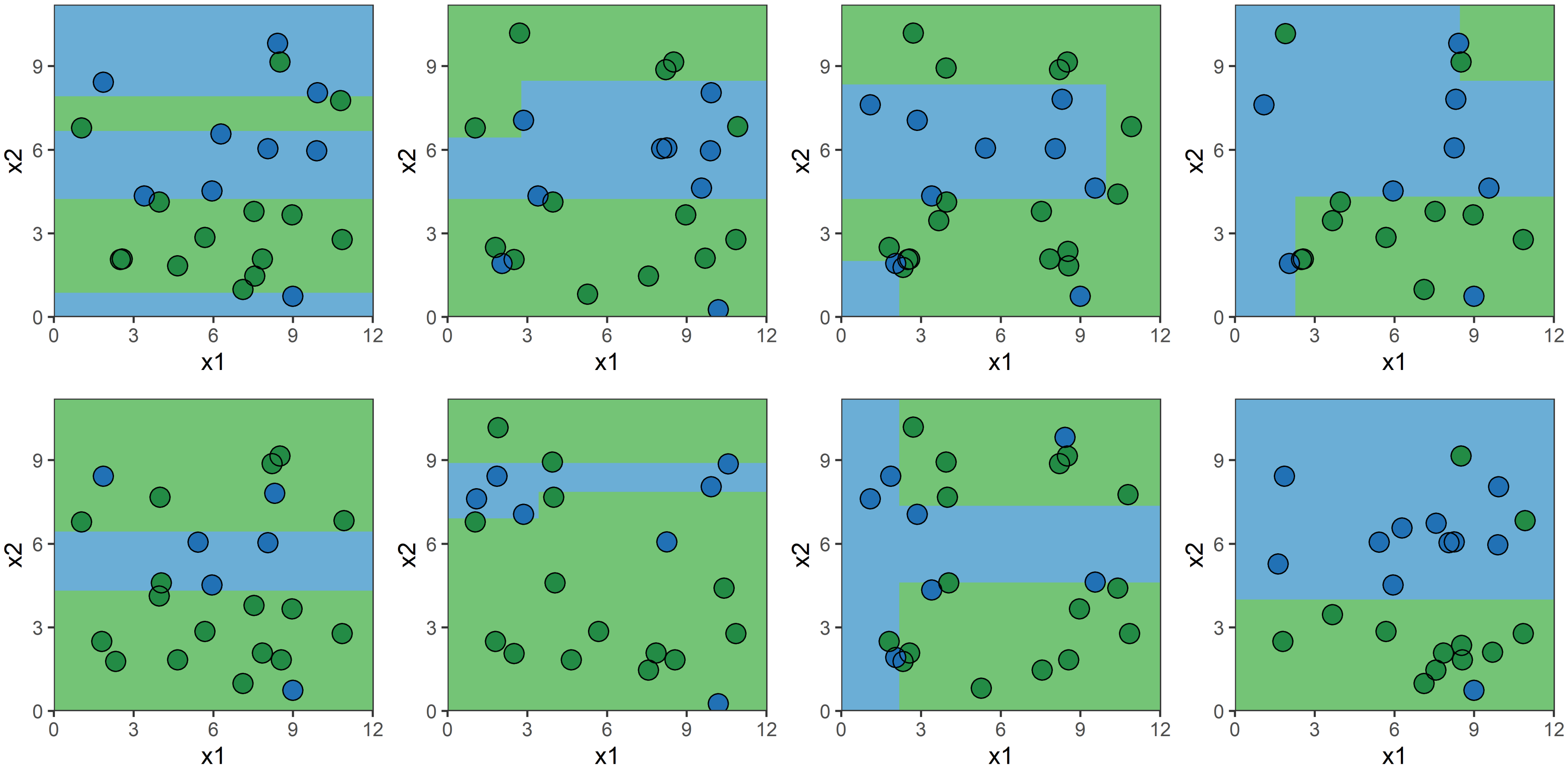
- A **general-purpose** procedure to reduce variance
  - particularly useful and frequently used in the context of decision trees

Bagging procedure:

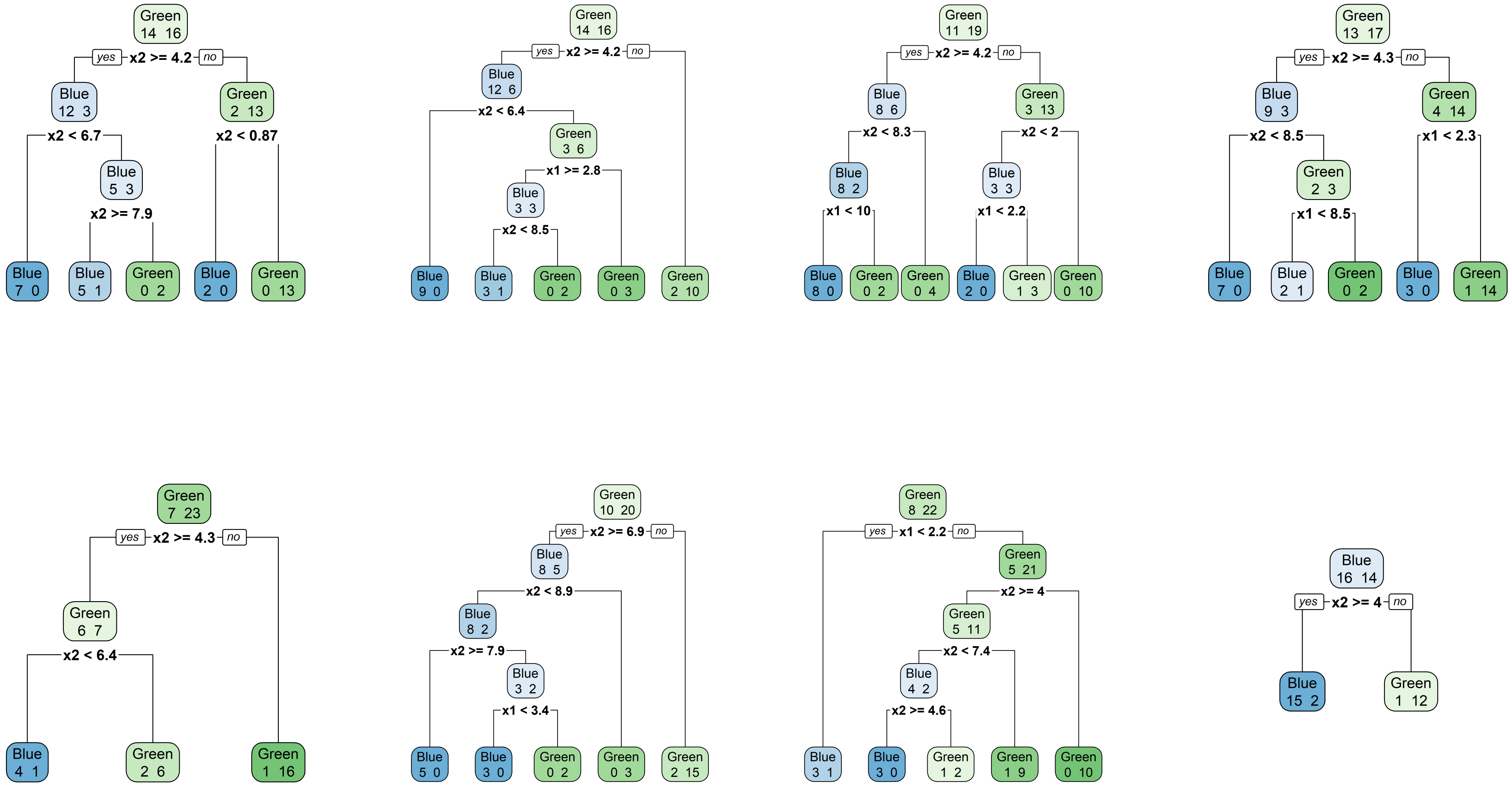
1. Bootstrap
   - sample with replacement repeatedly
   - generate B different bootstrapped training data sets
2. Train
   - train on the bth bootstrapped training set to get $\hat{f}^{*b}(x)$
3. Aggregate (Regression: average, Classification: majority vote)

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

Bagging: Illustration



Bagging: Illustration



Samples that are in the bag

Let's say element i, j of the matrix is 1 if the ith observation is in the jth bootstrap sample and 0 otherwise; i.e. it is "in the bag".

Boot_1 <dbl>	Boot_2 <dbl>	Boot_3 <dbl>	Boot_4 <dbl>	Boot_5 <dbl>
1	0	1	0	0
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	1	1	0	0
0	1	1	1	1
1	1	0	1	1
0	0	0	1	1
1	1	1	1	1
0	1	1	1	1

1-10 of 10 rows

Samples that are out of bag

Now consider the inverse, element i, j of the matrix is 1 if the ith observation is **not** in the jth bootstrap sample, it is "out of the bag".

Boot_1 <dbl>	Boot_2 <dbl>	Boot_3 <dbl>	Boot_4 <dbl>	Boot_5 <dbl>
0	1	0	1	1
0	1	0	1	1
0	1	0	1	0
0	1	0	0	1
0	0	0	1	1
1	0	0	0	0
0	0	1	0	0
1	1	1	0	0
0	0	0	0	0
1	0	0	0	0

1-10 of 10 rows

#OOB <dbl>
3
3
2
2
2
1
1
3
0
1

1-10 of 10 rows

Can perform "out of bag evaluation" by using the out of bag samples as a test set. This is cheaper than cross-validation.

UNSW SYDNEY

Out-of-bag error estimation

There is a very straightforward way to estimate the test error of a bagged model

- On average, each bagged tree makes use of around two-thirds of the observations
- The remaining one-third of the observations are referred to as the out-of-bag (OOB) observations
- Predict the response for the ith observation using each of the trees in which that observation was OOB
  - $\sim B/3$ predictions for the ith observation
- Take the average or a majority vote to obtain a single OOB prediction for the i th observation
- Turns out this is very similar to the LOOCV error.

Bagging: variable selection

- Bagging can lead to difficult-to-interpret results, since, on average, no predictor is excluded
- Variable importance measures can be used
  - Bagging regression trees: RSS reduction for each split
  - Bagging classification trees: Gini index reduction for each split
- Pick the ones with the highest variable importance measure

Lecture Outline

- Decision Trees
- Growing a Tree
- National Flood Insurance Program Demo
- Pruning a Tree
- Bootstrap Aggregation
- **Random Forests**
- Boosting

Random Forests

Random forests decorrelates the bagged trees

- At each split of the tree, a fresh random sample of m predictors is chosen as split candidates from the full set of p predictors
- Strong predictors are used in (far) fewer models, so the effect of other predictors can be properly measured.
  - Reduces the variance of the resulting trees
- Typically choose $m \approx \sqrt{p}$
- Bagging is a special case of a random forest with $m = p$

Random forests (stock photo)

Fitting with `randomForest`

```r
rf_model <- randomForest(amountPaidOnBuildingClaim ~ ., data = train_set,
    ntree=50, importance = TRUE)
```

Method	RMSE
Linear Model	5.4792406^{4}
Large Tree	4.8683476^{4}
Pruned Tree	4.7371652^{4}
Random Forest	4.2828885^{4}

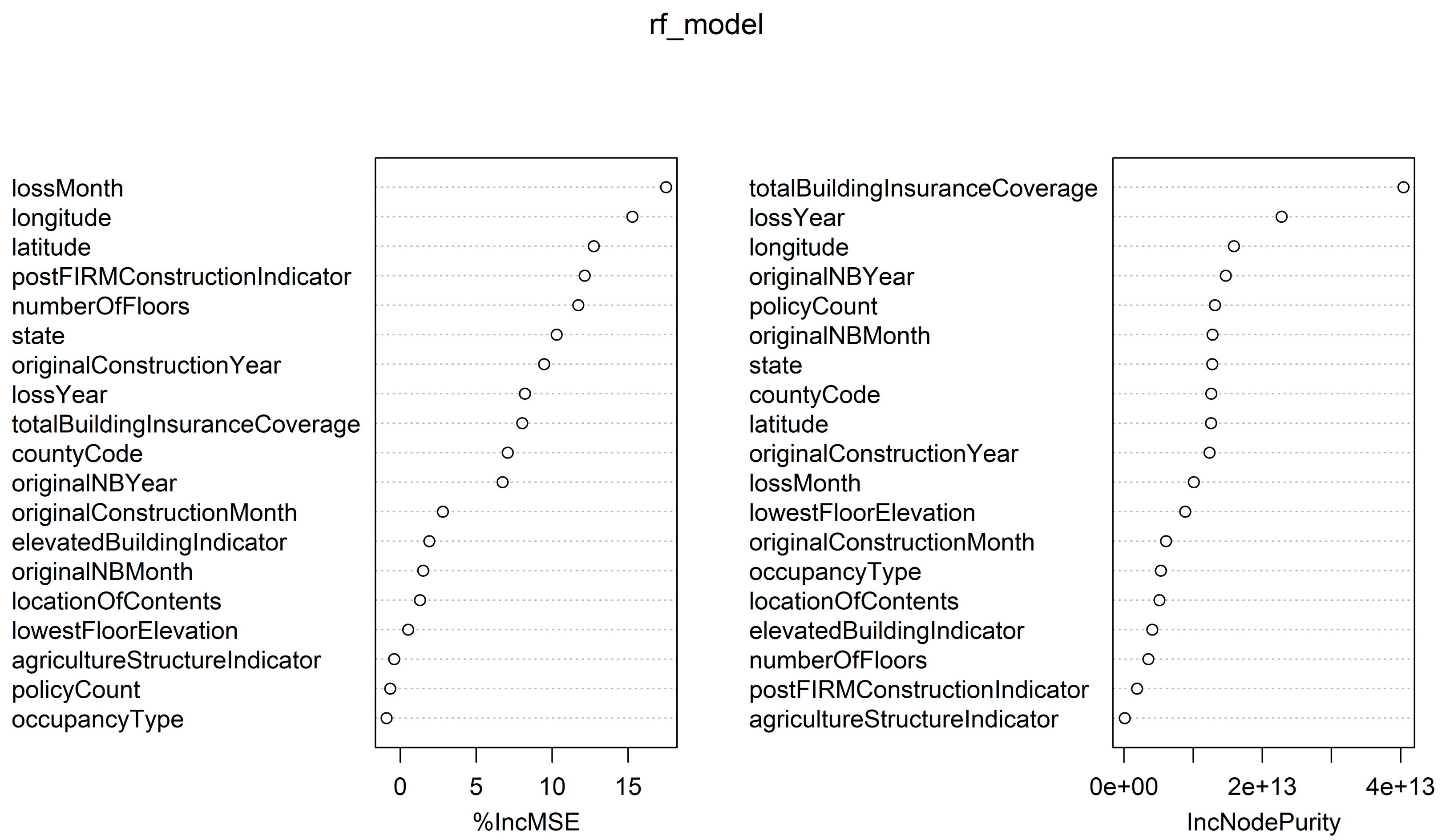
Variable importance

```r
importance(rf_model)
```

```
                                %IncMSE IncNodePurity
agricultureStructureIndicator -0.4092969  7.736630e+10
policyCount                   -0.6649637  1.312233e+13
countyCode                     7.0676498  1.261540e+13
elevatedBuildingIndicator      1.9163476  4.066224e+12
latitude                      12.7351805  1.257793e+13
locationOfContents             1.2897842  5.107086e+12
longitude                     15.2725053  1.588849e+13
lowestFloorElevation           0.5264056  8.837304e+12
occupancyType                 -0.8955214  5.325440e+12
postFIRMConstructionIndicator 12.1436127  1.855385e+12
state                         10.2933401  1.275269e+13
totalBuildingInsuranceCoverage  8.0324394  4.040641e+13
numberOfFloors                11.7156439  3.504747e+12
lossYear                       8.2085826  2.277261e+13
lossMonth                     17.4917355  1.009505e+13
originalConstructionYear       9.4747850  1.235049e+13
originalConstructionMonth      2.8065239  6.062158e+12
originalNBYear                 6.7318799  1.468756e+13
originalNBMonth                1.5024842  1.277571e+13
```

Variable importance plot

```r
varImpPlot(rf_model)
```



Lecture Outline

- Decision Trees
- Growing a Tree
- National Flood Insurance Program Demo
- Pruning a Tree
- Bootstrap Aggregation
- Random Forests
- **Boosting**

Boosting

- A general approach that can be applied to many statistical learning methods for regression or classification
- We focus on boosting for regression trees
- Involves combining a large number of decision trees
  - trees are grown sequentially
  - using the information from previously grown trees
  - no bootstrap - instead each tree is fitted on a modified version of the original data (sequentially)
- Unlike standard trees, boosting learns slowly - by focusing on the residuals and hence focusing on areas the previous tree did not perform well.

Boosting Algorithm for Regression Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set
2. For $b = 1, 2, \cdots, B$, repeat:
   a. Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r)
   b. Update $\hat{f}$ by adding in a shrunken version of the new tree

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

   c. Update the residuals

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

3. Output the boosted model

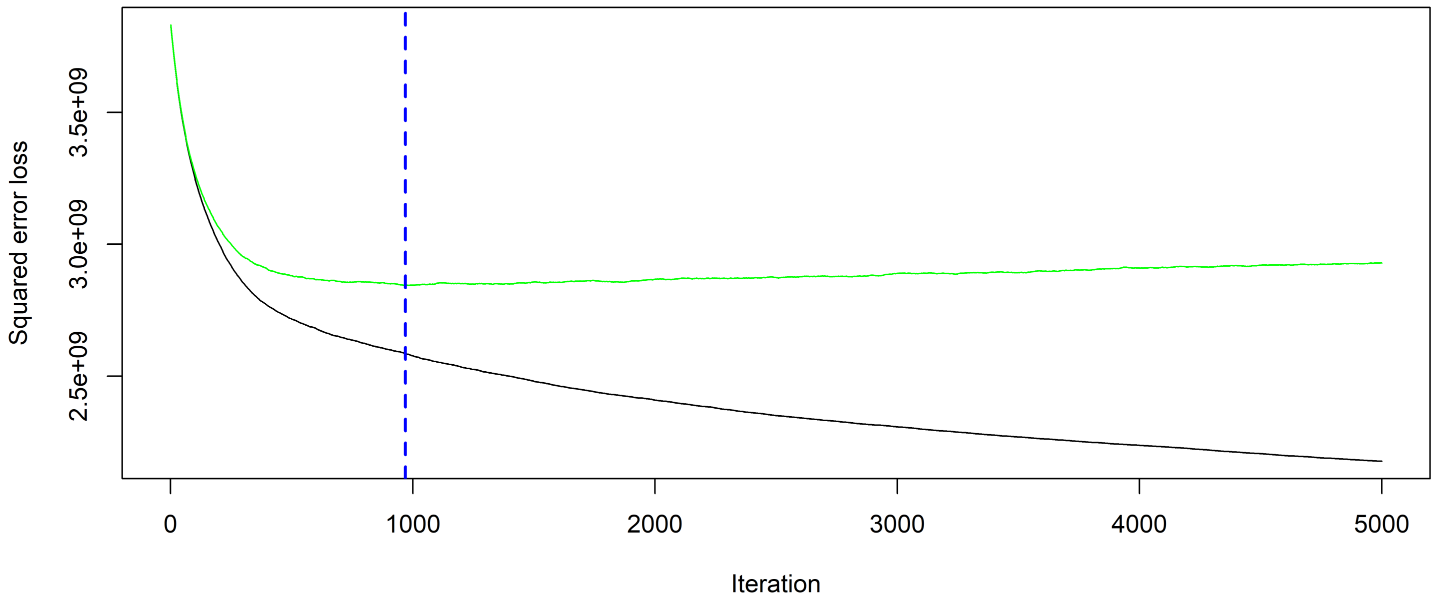
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting Tuning Parameters

- The number of trees B
  - overfit if B is too large
  - use cross-validation to select B
- The shrinkage parameter λ
  - a small positive number
  - controls the rate at which boosting learns
  - typical values are 0.01 or 0.001
- The number d of splits in each tree
  - $d = 1$ often works well, each tree is a stump

Fitting with gbm

```r
# Fit a gradient boosting model
gbm_model <- gbm(amountPaidOnBuildingClaim ~ ., data = train_set,
                 distribution = "gaussian", n.trees = 5000,
                 interaction.depth = 2, shrinkage = 0.01, cv.folds = 5)
best_iter <- gbm.perf(gbm_model, method = "cv")
```




```r
predictions <- predict(gbm_model, newdata = val_set, n.trees = best_iter)
```

Comparing models

Method	RMSE
Linear Model	5.4792406^{4}
Gradient Boosting	4.9819999^{4}
Large Tree	4.8683476^{4}
Pruned Tree	4.7371652^{4}
Random Forest	4.2828885^{4}

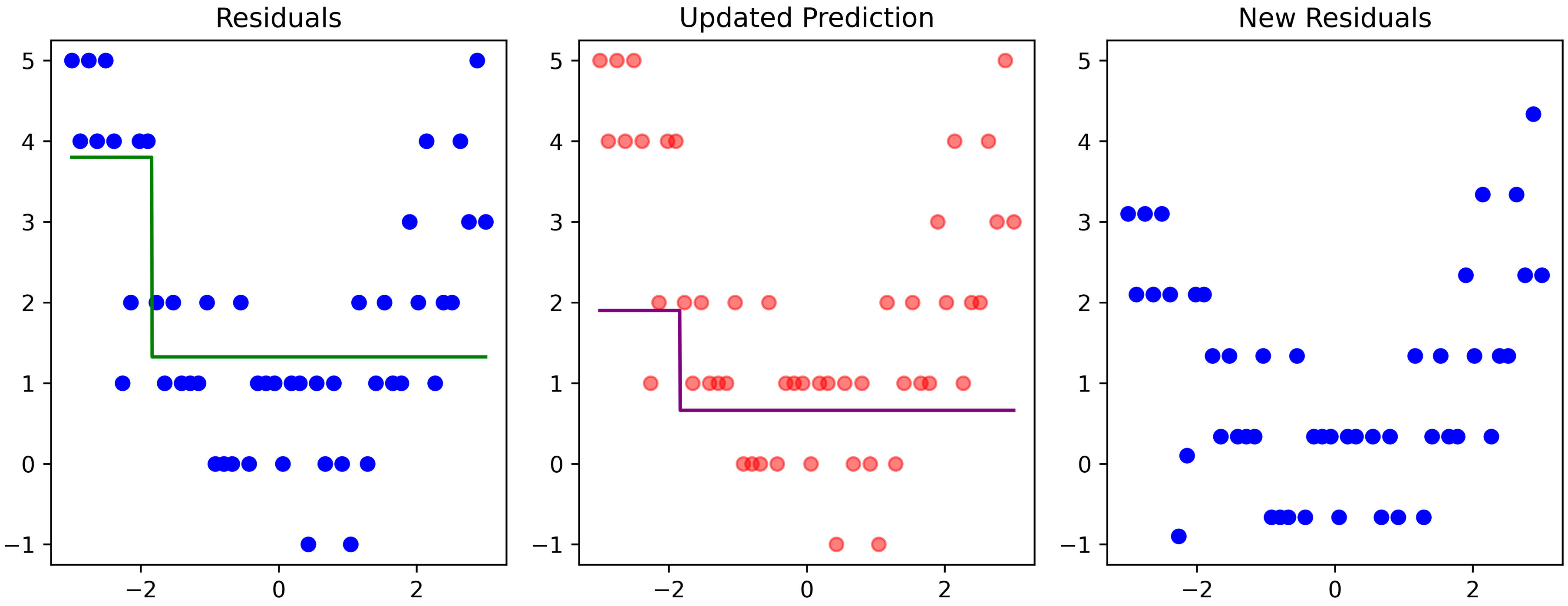
Finally, evaluating the winning model on the test set:

```
if (val_rmse_gbm < val_rmse_rf) {
  predictions <- predict(gbm_model, newdata = test_set, n.trees = best_iter)
  test_rmse <- sqrt(mean(predictions - test_set$amountPaidOnBuildingClaim)^2)
} else {
  predictions <- predict(rf_model, newdata = test_set)
  test_rmse <- sqrt(mean(predictions - test_set$amountPaidOnBuildingClaim)^2)
}
test_rmse
```

```
[1] 965.1979
```

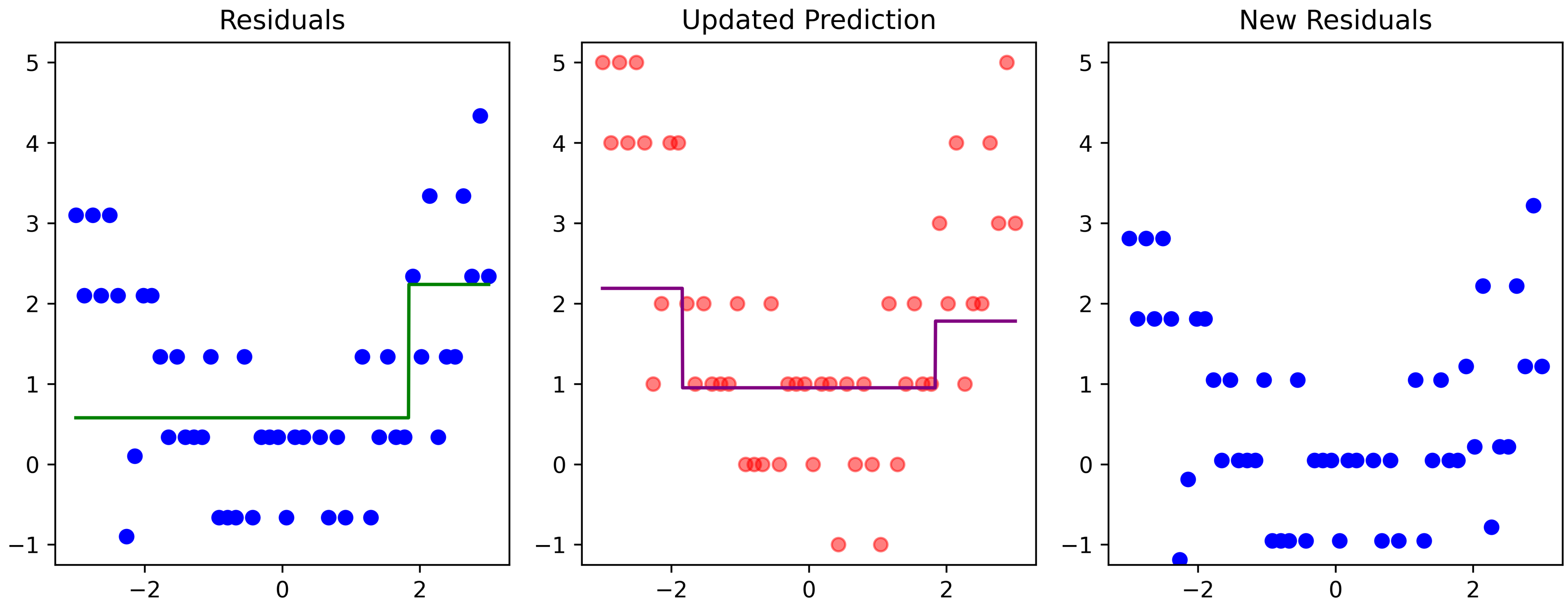
Test set error for the winning model is 965.1979057.

Boosting (iteration 1)



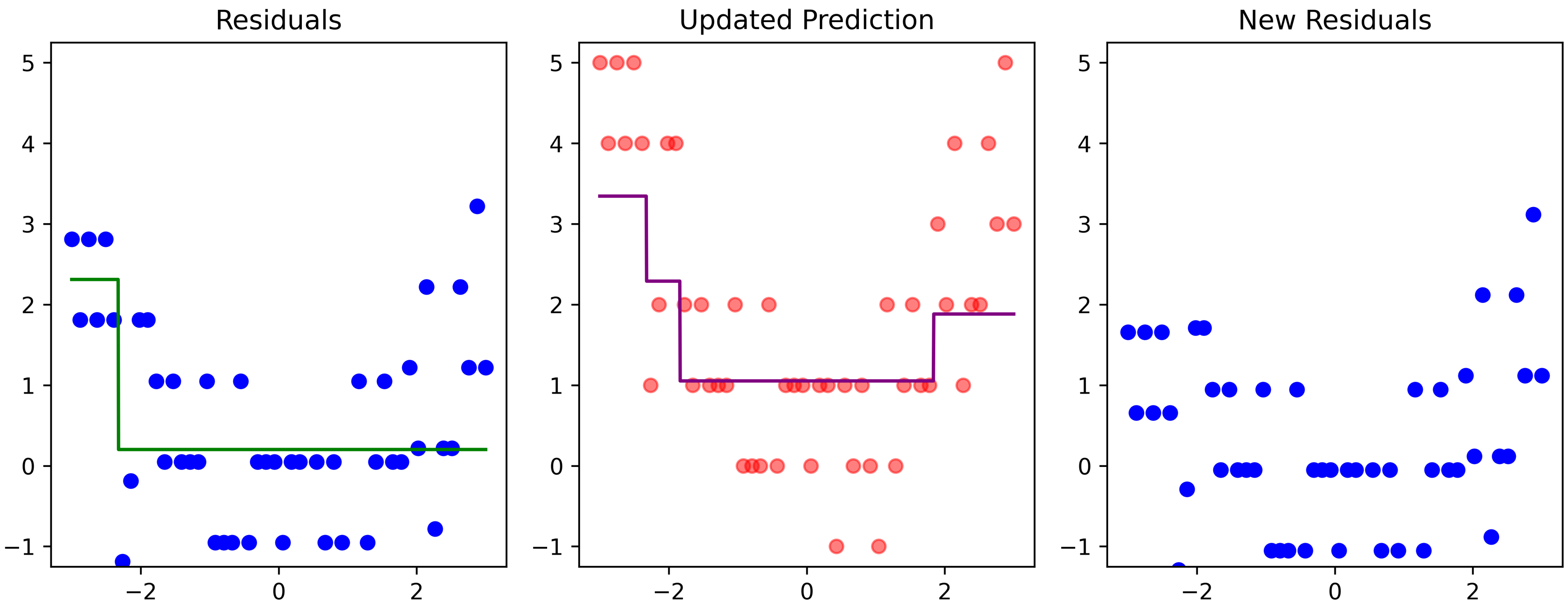
Here, $\lambda = \frac{1}{2}$ is the learning rate.

Boosting (iteration 2)



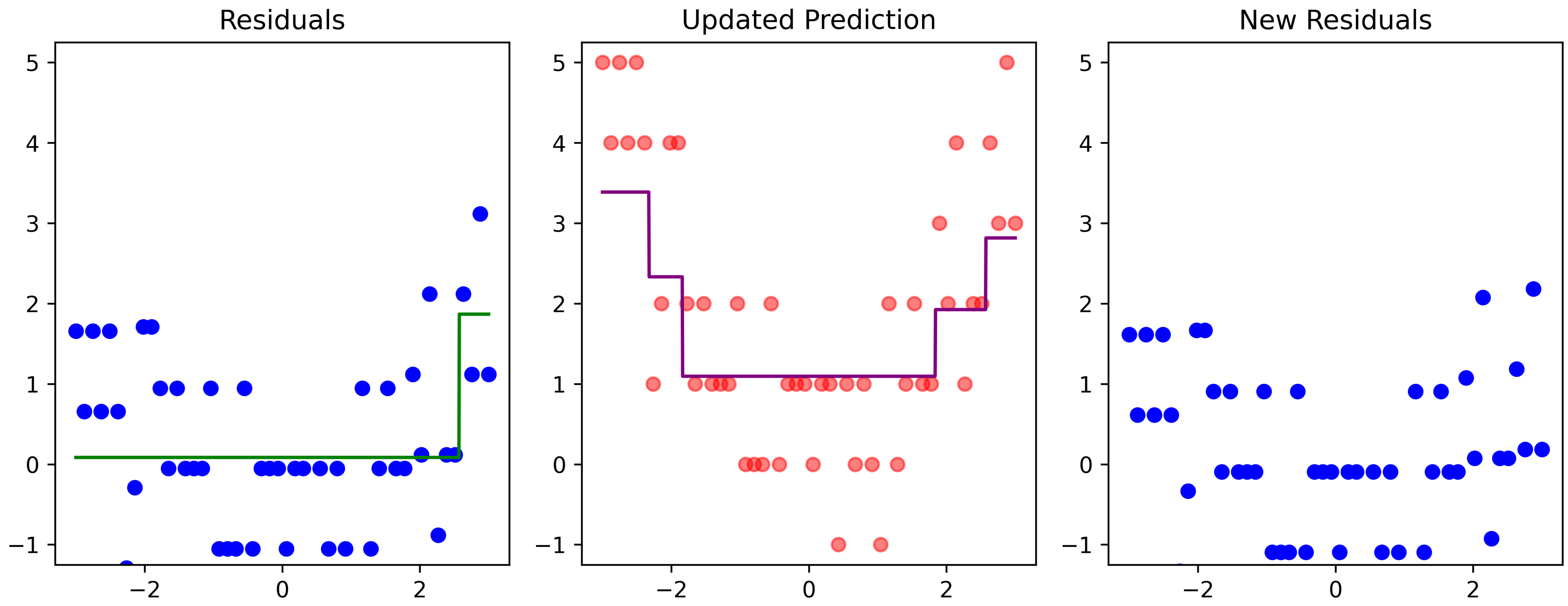
Here, $\lambda = \frac{1}{2}$ is the learning rate.

Boosting (iteration 3)



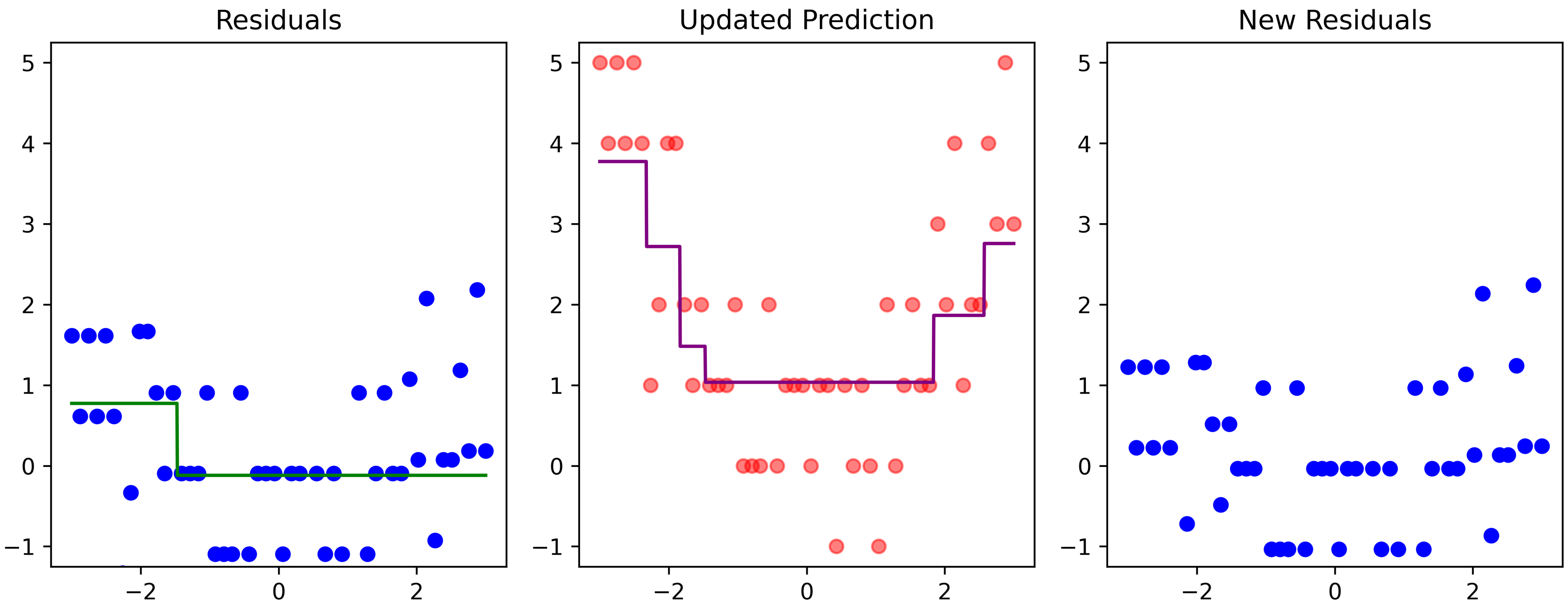
Here, $\lambda = \frac{1}{2}$ is the learning rate.

Boosting (iteration 4)



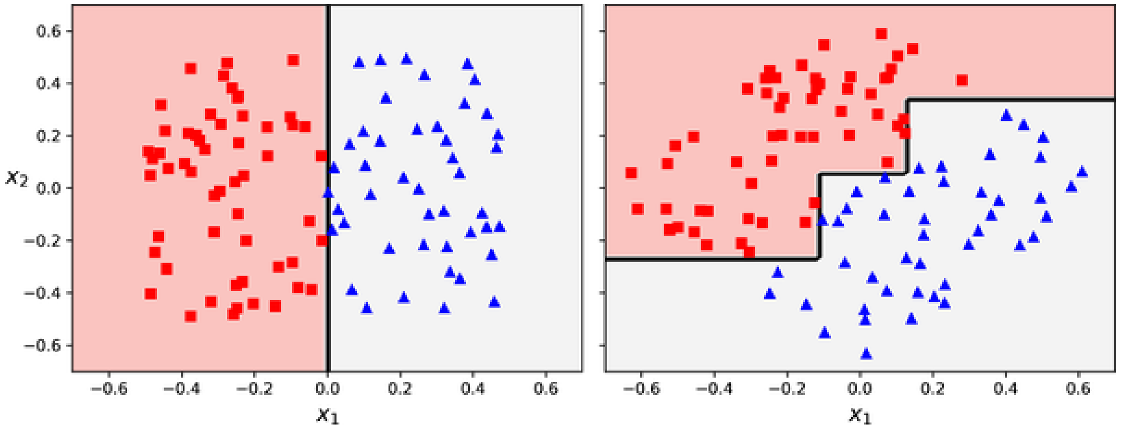
Here, $\lambda = \frac{1}{2}$ is the learning rate.

Boosting (iteration 5)



Here, $\lambda = \frac{1}{2}$ is the learning rate.

Sensitivity to training data orientation



An example of a dataset which is rotated and fit to decision trees.

UNSW SYDNEY