Tree-Based Methods

ACTL3142 & ACTL5110 Statistical Machine Learning for Risk and Actuarial Applications

UNSW SYDNEY

Overview

Decision trees

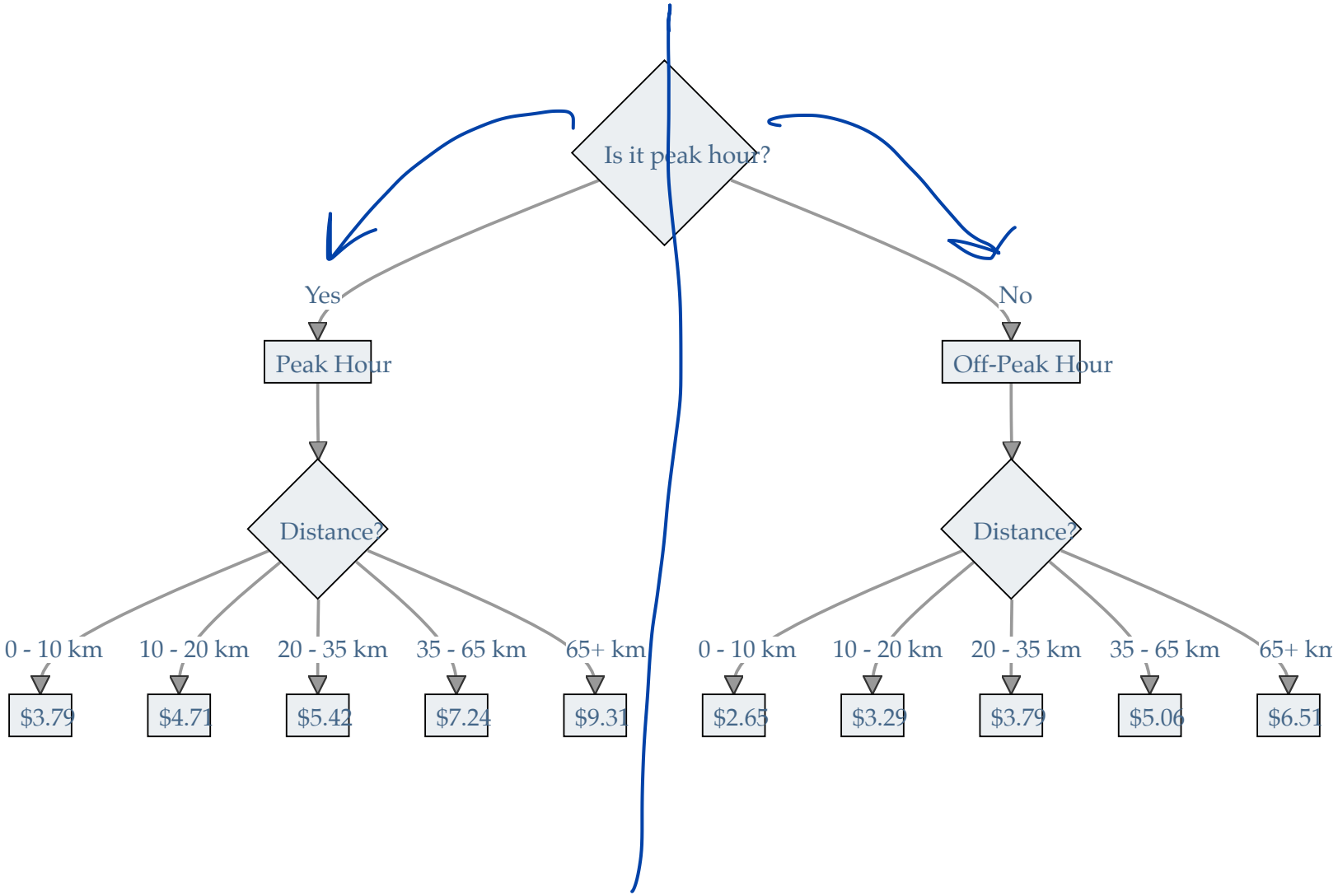
- Stratify / segment the predictor space into a number of simple regions
- The set of splitting rules can be summarised in a tree

Bagging, random forests, boosting

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

Trees

How much is a train ticket?



In code

R | Python

```r
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

```
1 library(ISLR2)
2
3 # Load the hitters dataset
4 data(Hitters)
5
6 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

UNSW SYDNEY

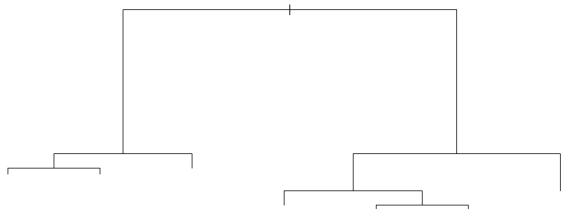
Fit a basic tree

R | Python

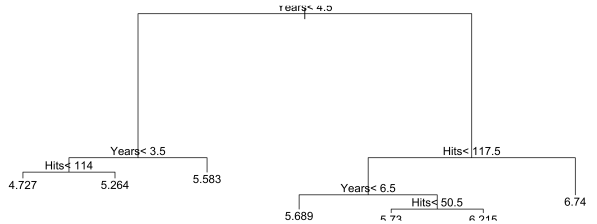
```r
library(rpart)

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

tree
```

```
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)
```



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



As usual, default print/plots in R are 🤮…

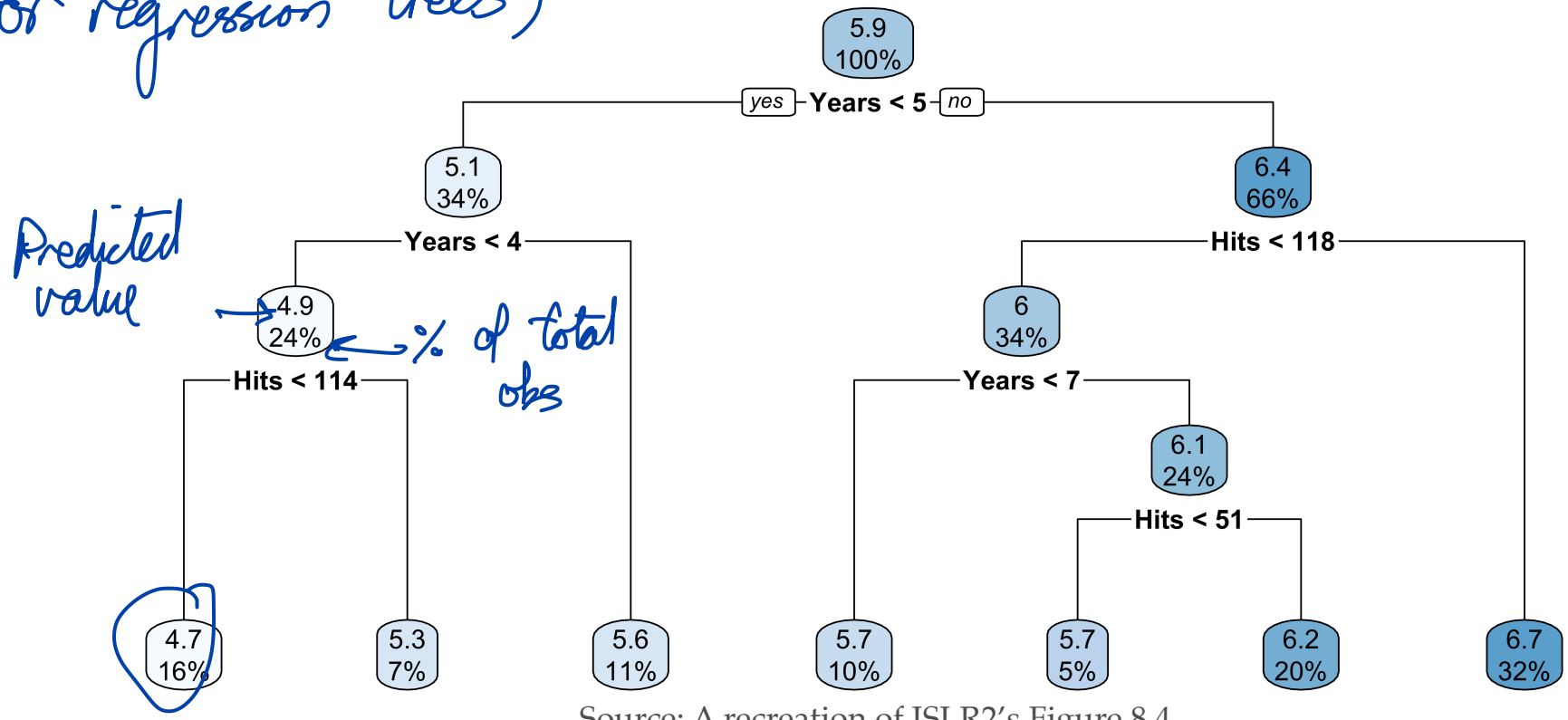
Source: These plots are recreating ISLR2's Figure 8.4.

Plot a tree

R | Python

```r
library(rpart.plot)

rpart.plot(tree)
```

(for regression trees)



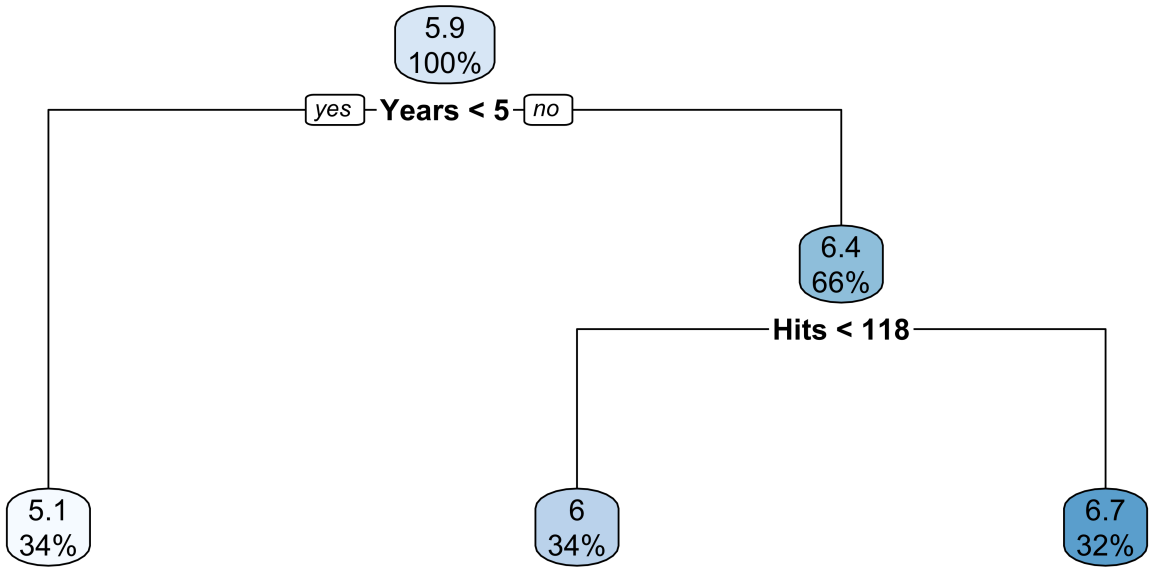
Predicted value

% of total obs

Source: A recreation of ISLR2's Figure 8.4.

A simplified "pruned" tree

```
# Prune the tree down to 3 terminal nodes
pruned_tree <- prune(tree, cp = tree$cptable[3, "CP"])

rpart.plot(pruned_tree)
```

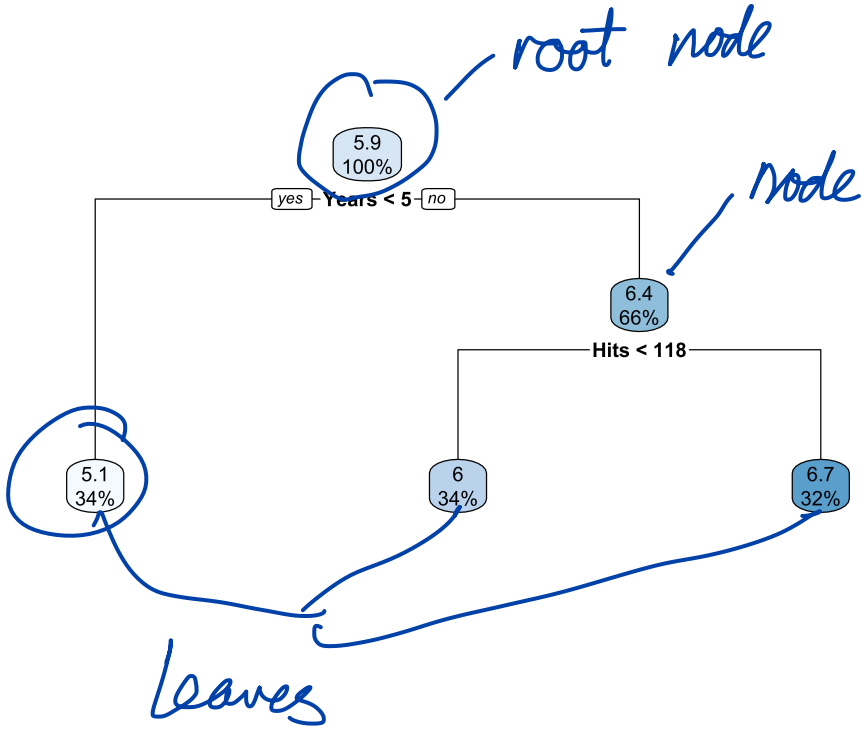


Source: A recreation of ISLR2's Figure 8.1.

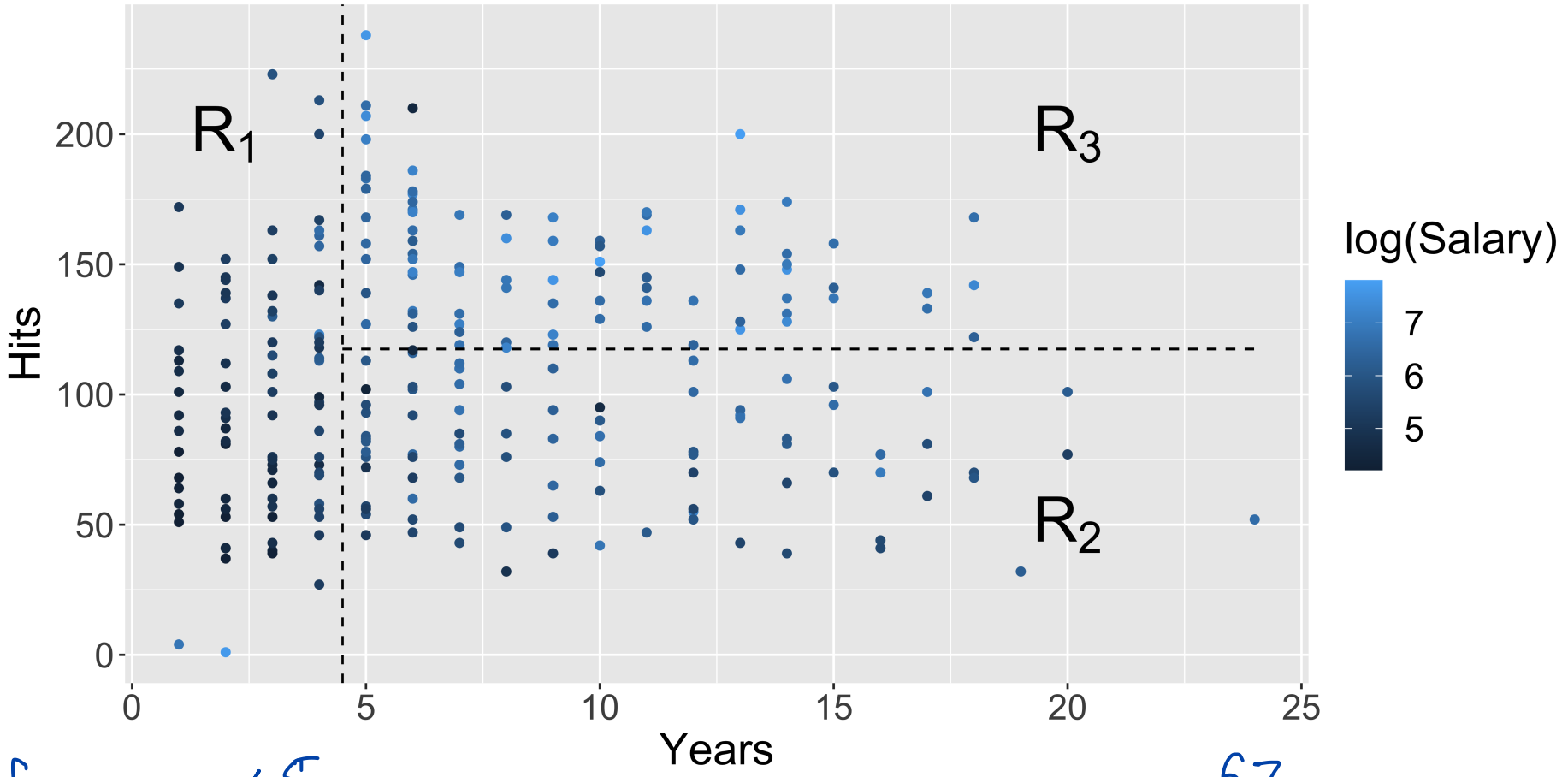
Tree Terminology

Internal nodes (& root node)	◯
Branches	↓
Terminal nodes or leaves	□

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



Regions in the predictor space



If years < 5

$R_1 = e^{5.1}$

$R_2 = e^{6}$

$R_3 = e^{6.7}$

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

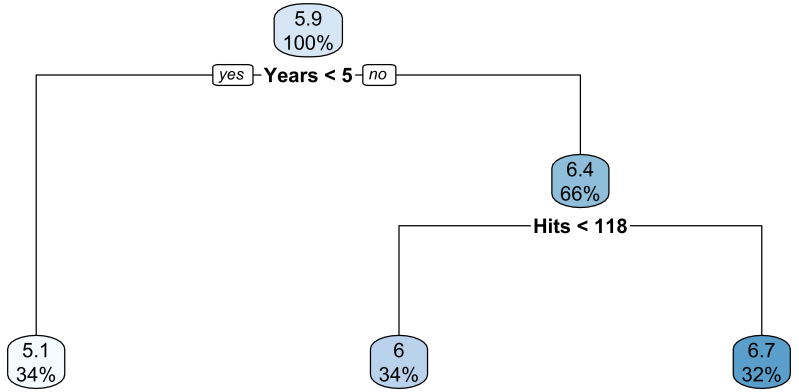
UNSW SYDNEY

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`?

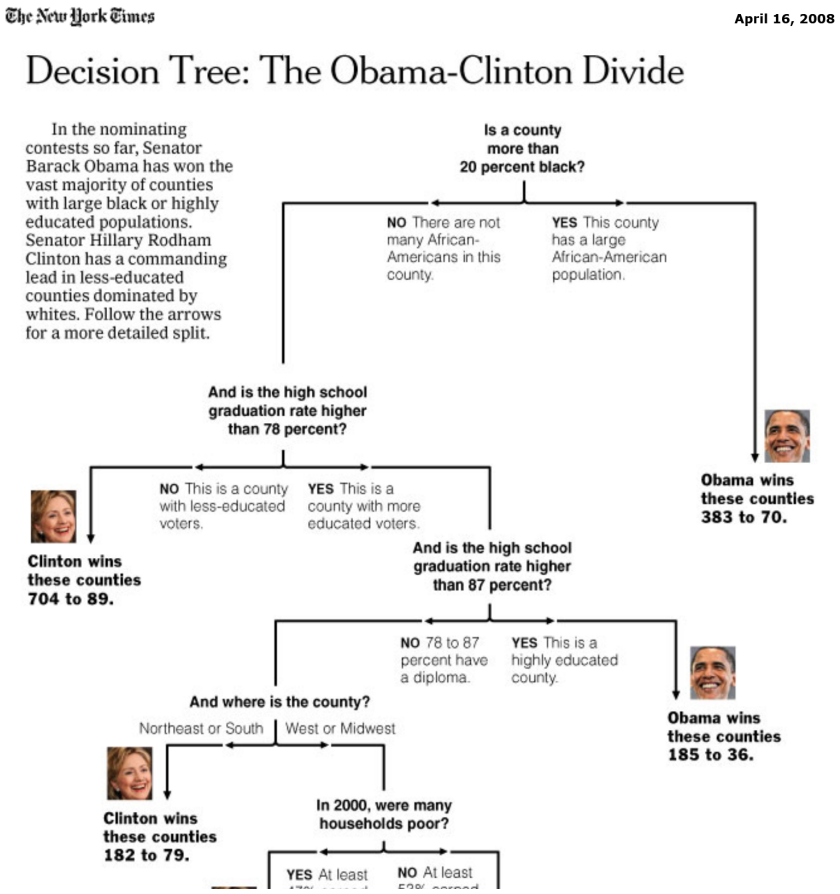
```
1 rpart.plot(pruned_tree)
```



Decision trees: summary

Trees are

- Simple
- Useful for interpretation
- Very common



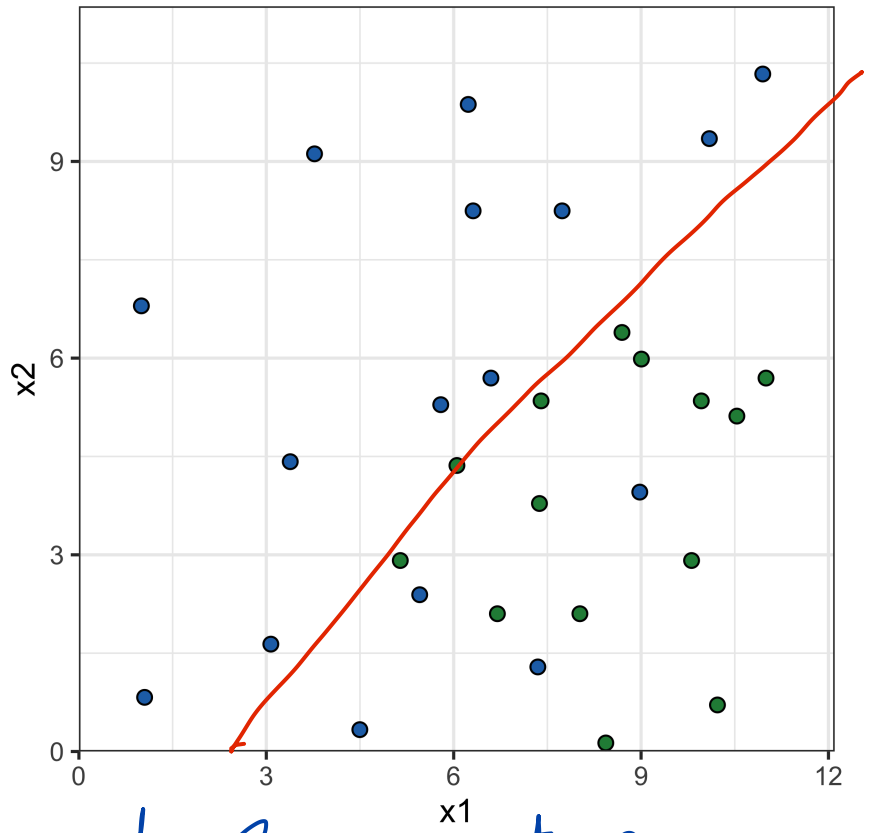
A decision tree in the wild.

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

Popular (IME 2023 abstracts)

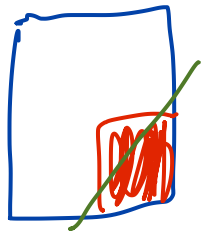
Growing a Tree

Growing a Tree I



Look at 2022 past exam, some data are not well suited to a linear model

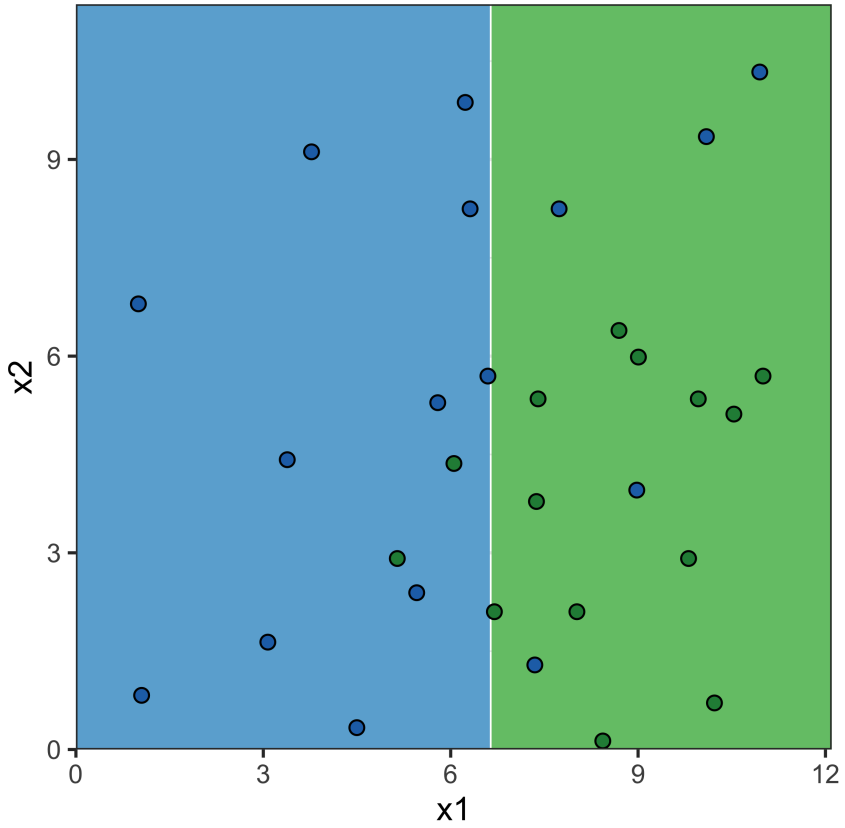
Regression trees – predict a value if in fixed the region

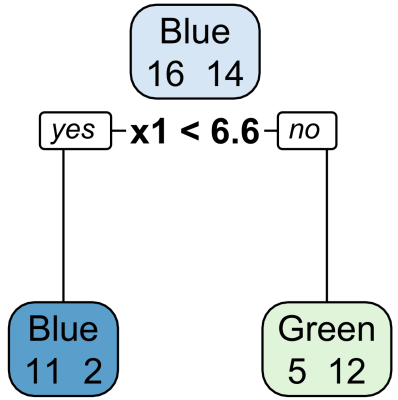


Blue
16 14

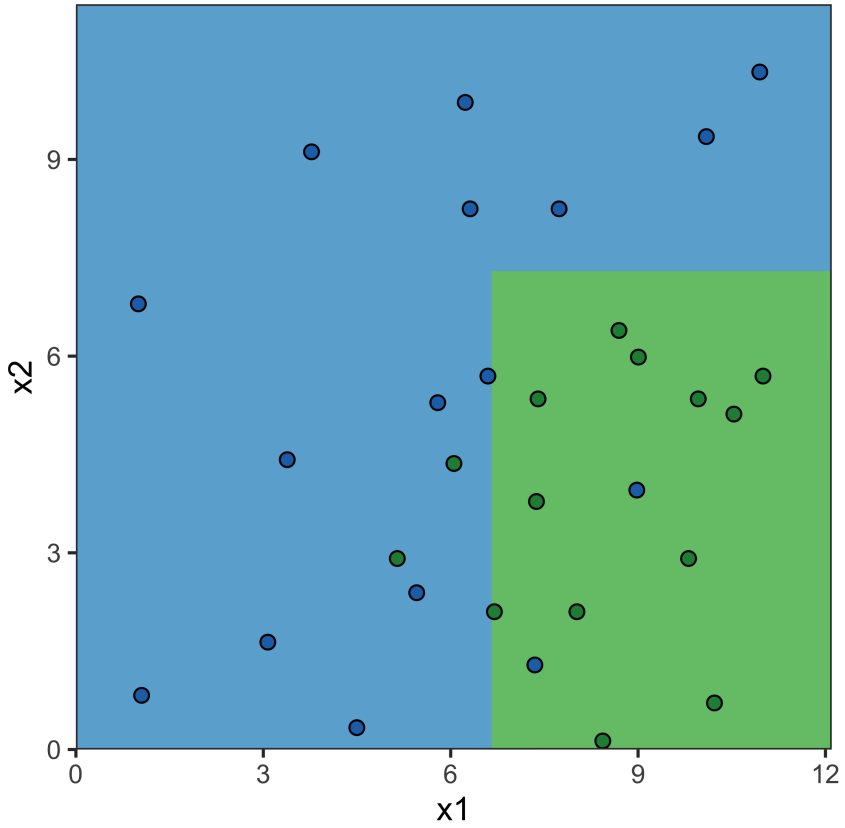
Classification tree – predict a class based on threshold and region.

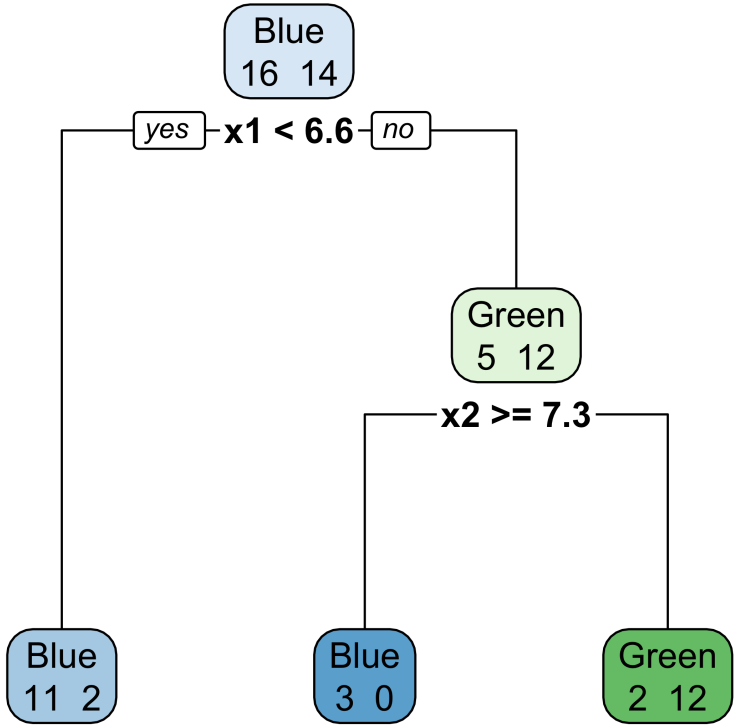
Growing a Tree II



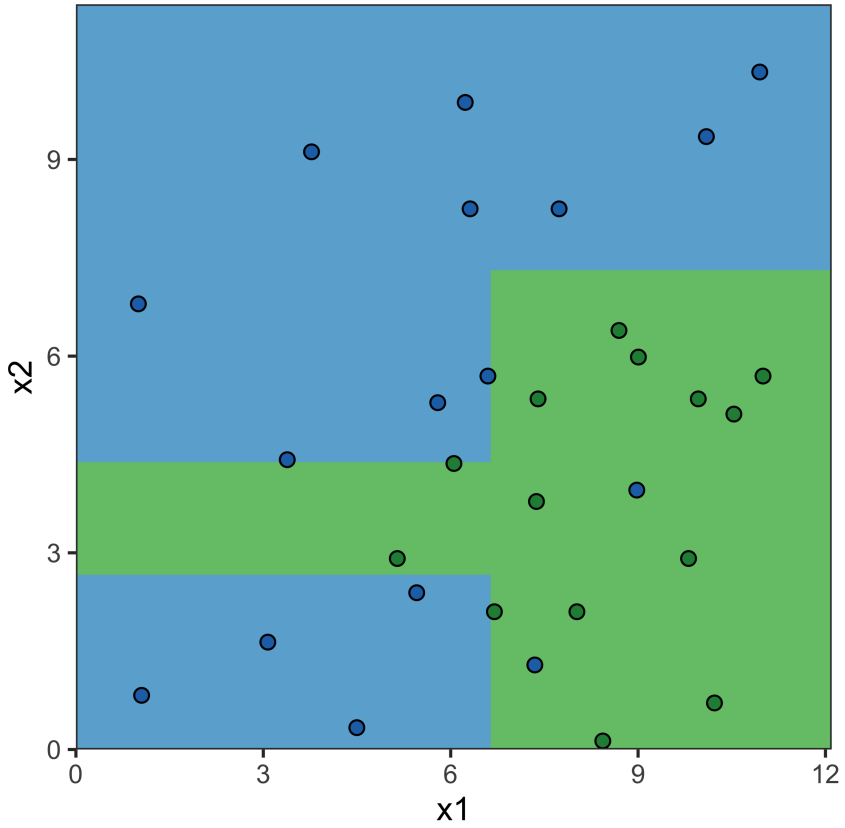


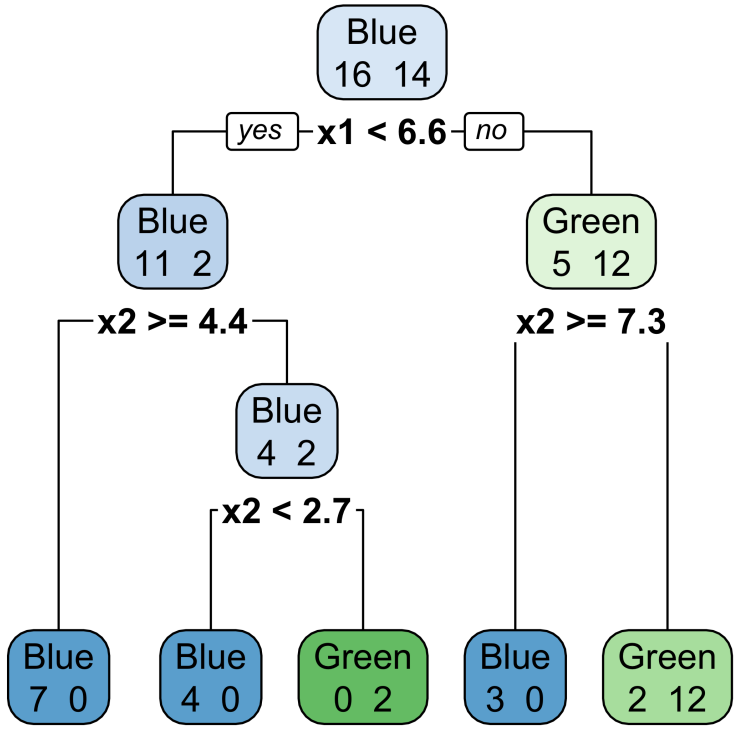
Growing a Tree III





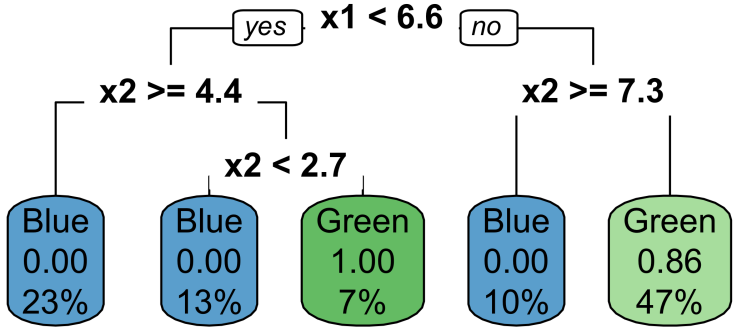
Growing a Tree IV



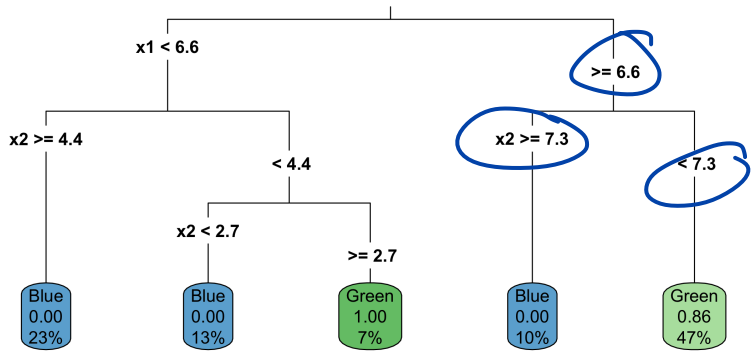


- If you keep dividing, you will regions with only 1 point

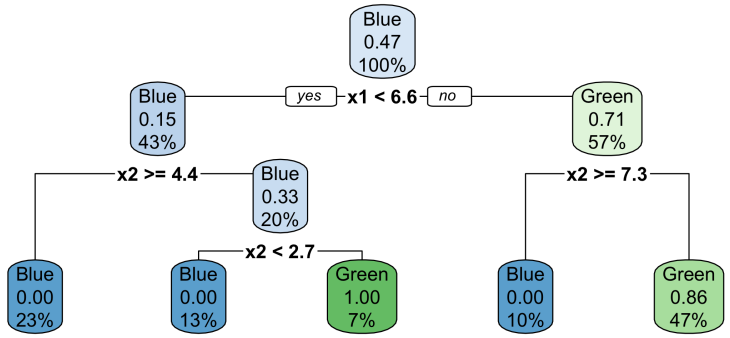
```
1 rpart.plot(tree4, type = 0)
```



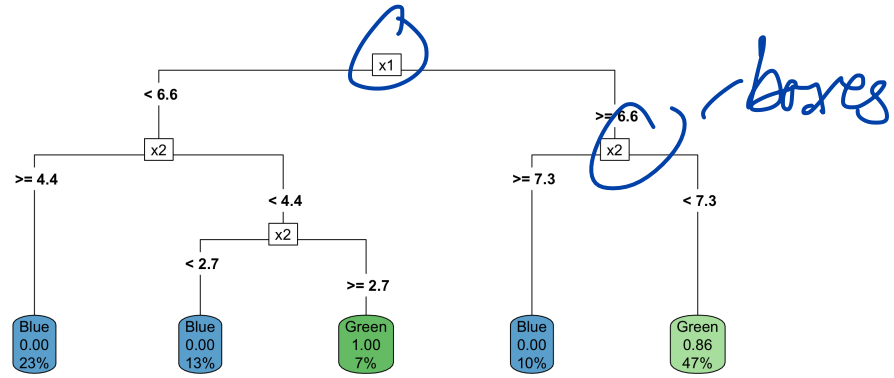

```
1 rpart.plot(tree4, type = 3)
```



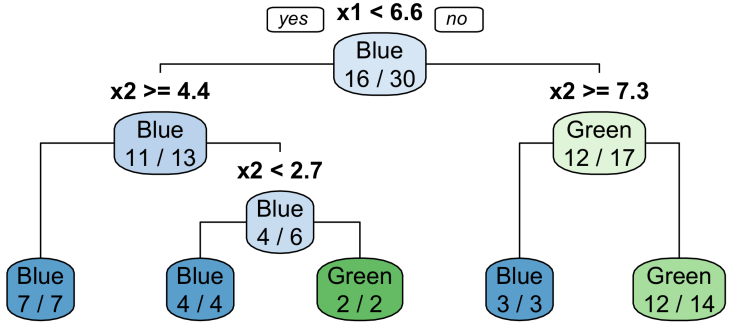

```
1 rpart.plot(tree4, type = 2)
```



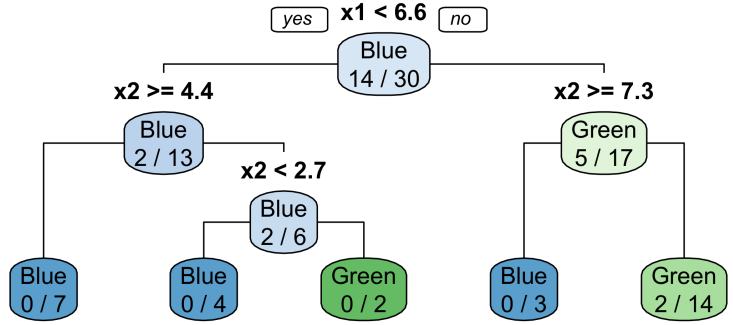

```
1 rpart.plot(tree4, type = 5)
```



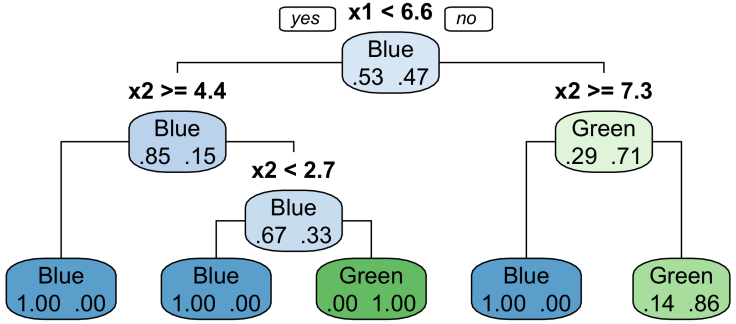
```
1 rpart.plot(tree4, type = 1, extra = 2)
```



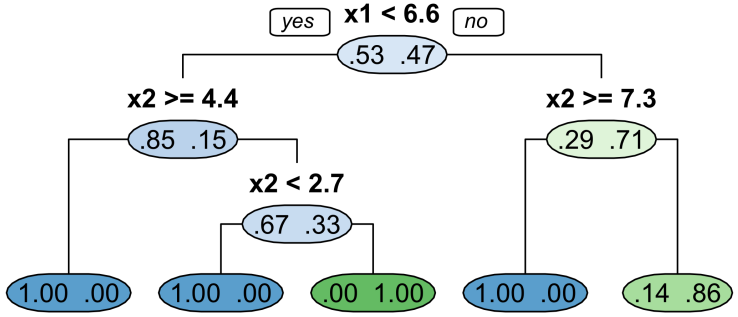

```
1 rpart.plot(tree4, type = 1, extra = 3)
```




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




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



Recursive Binary Splitting

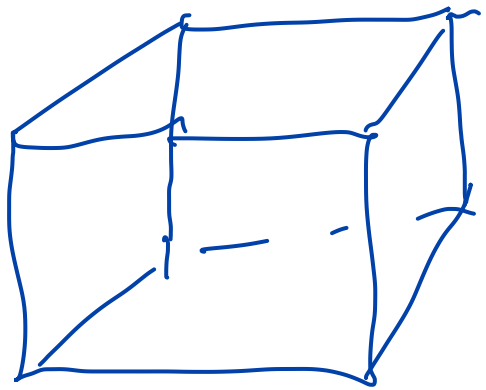
Construct Regions

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



Recursive Binary Splitting I

- 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

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

Recursive Binary Splitting II

- 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

– Is this the best possible tree?

* Probably not. Could be a better division out there.

– We do this because

* It is numerically feasible

Classification Trees

Classification error = $1 - \hat{p}_{mn}$ — not very sensitive

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

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

cross-entropy

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

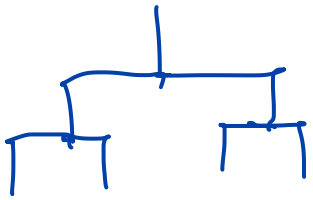
If $\hat{p}_{mn}$ is 0 or 1, $G \approx 0$

where

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

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.

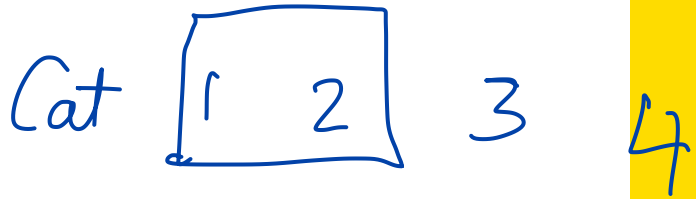


Source: Géron (2022), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow, 3rd Edition, O'Reilly Media, Chapter 6.

Decision Trees Discussion

Discussion Questions

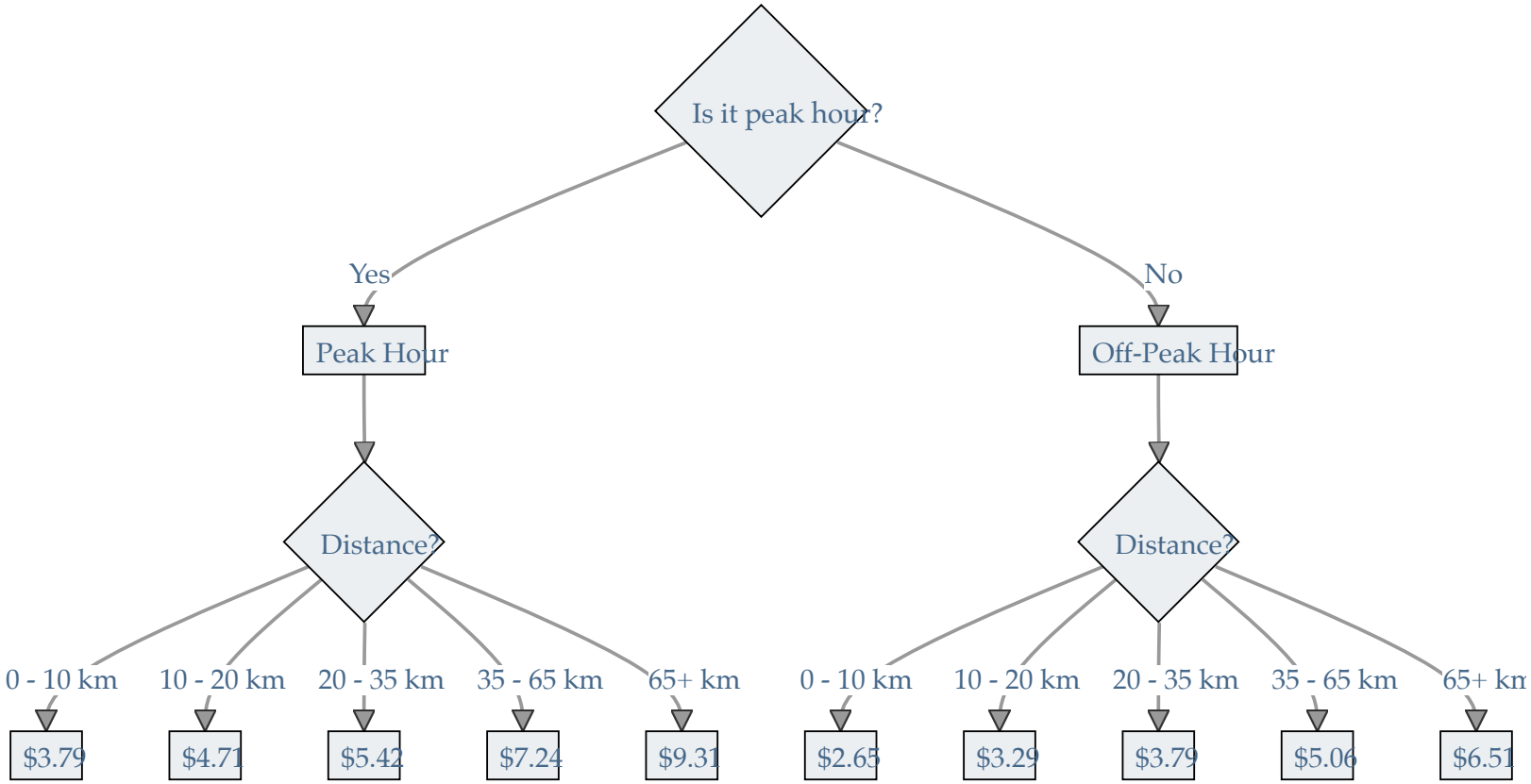
- Why binary splitting? → Already discussed
- What about splitting of categorical predictors?



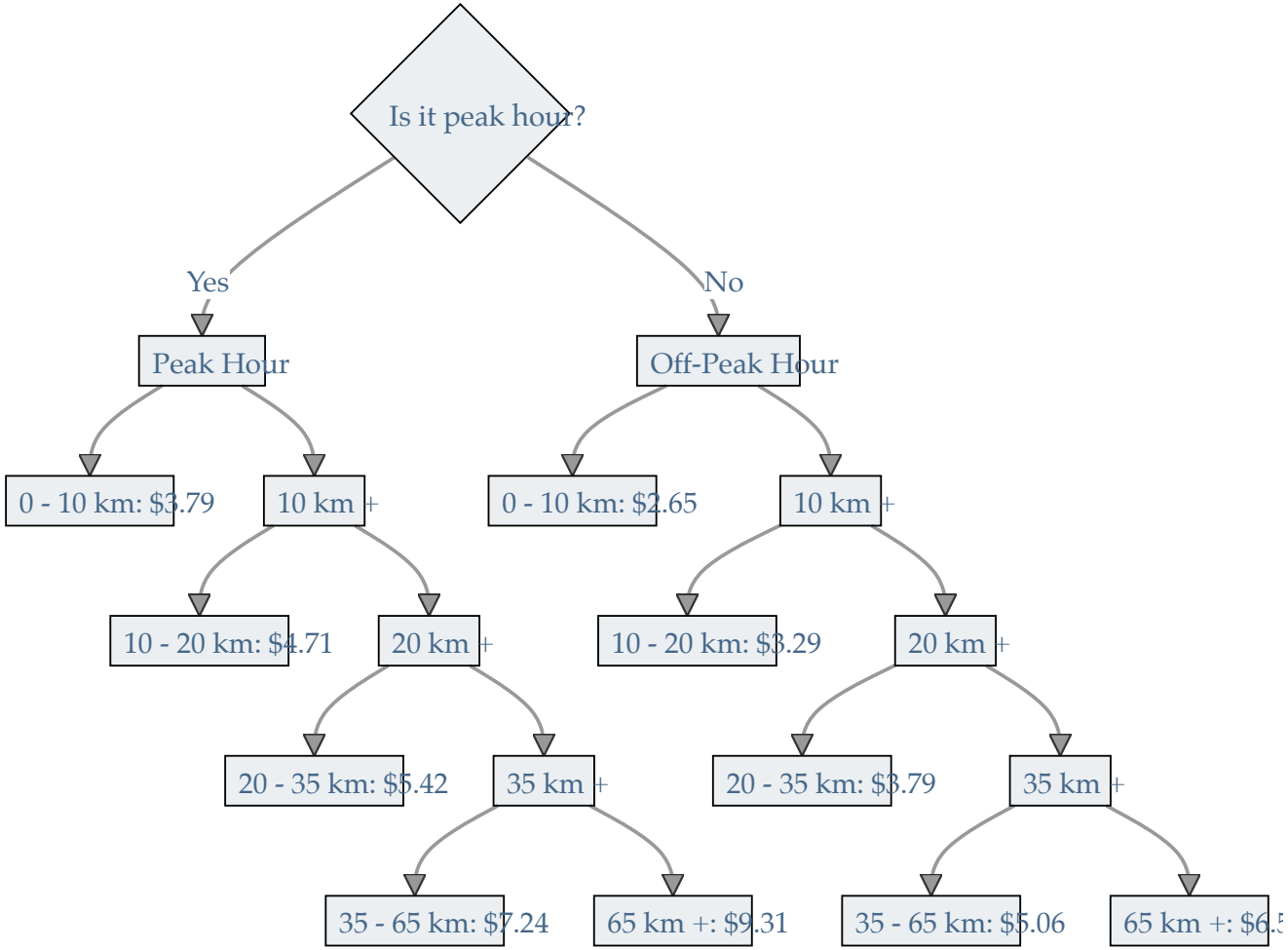
How large should we grow the tree?

- What's the problem if the tree is too large?
- What if the tree is too small?
  - Too simple

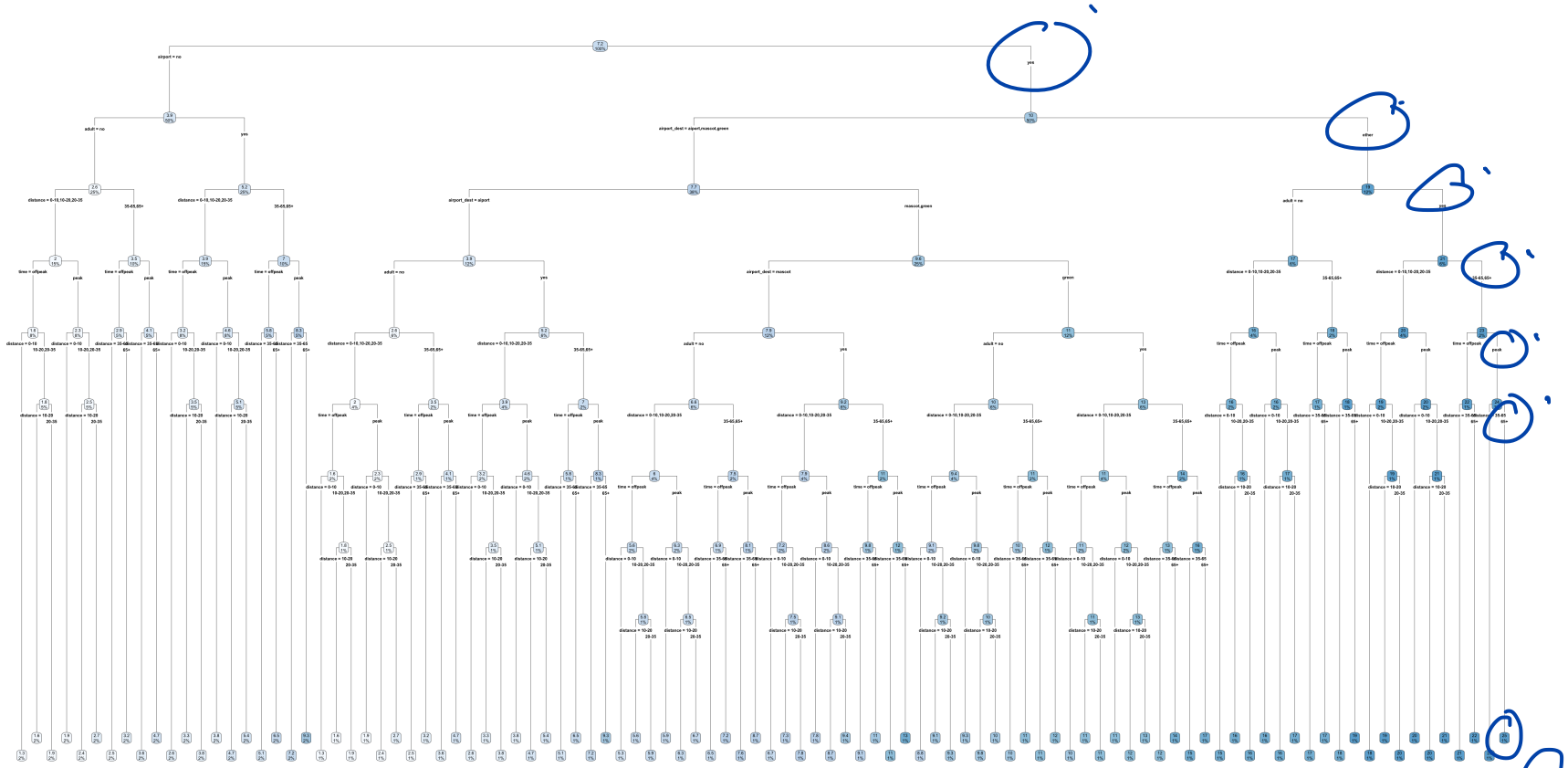
Non-binary train cost tree



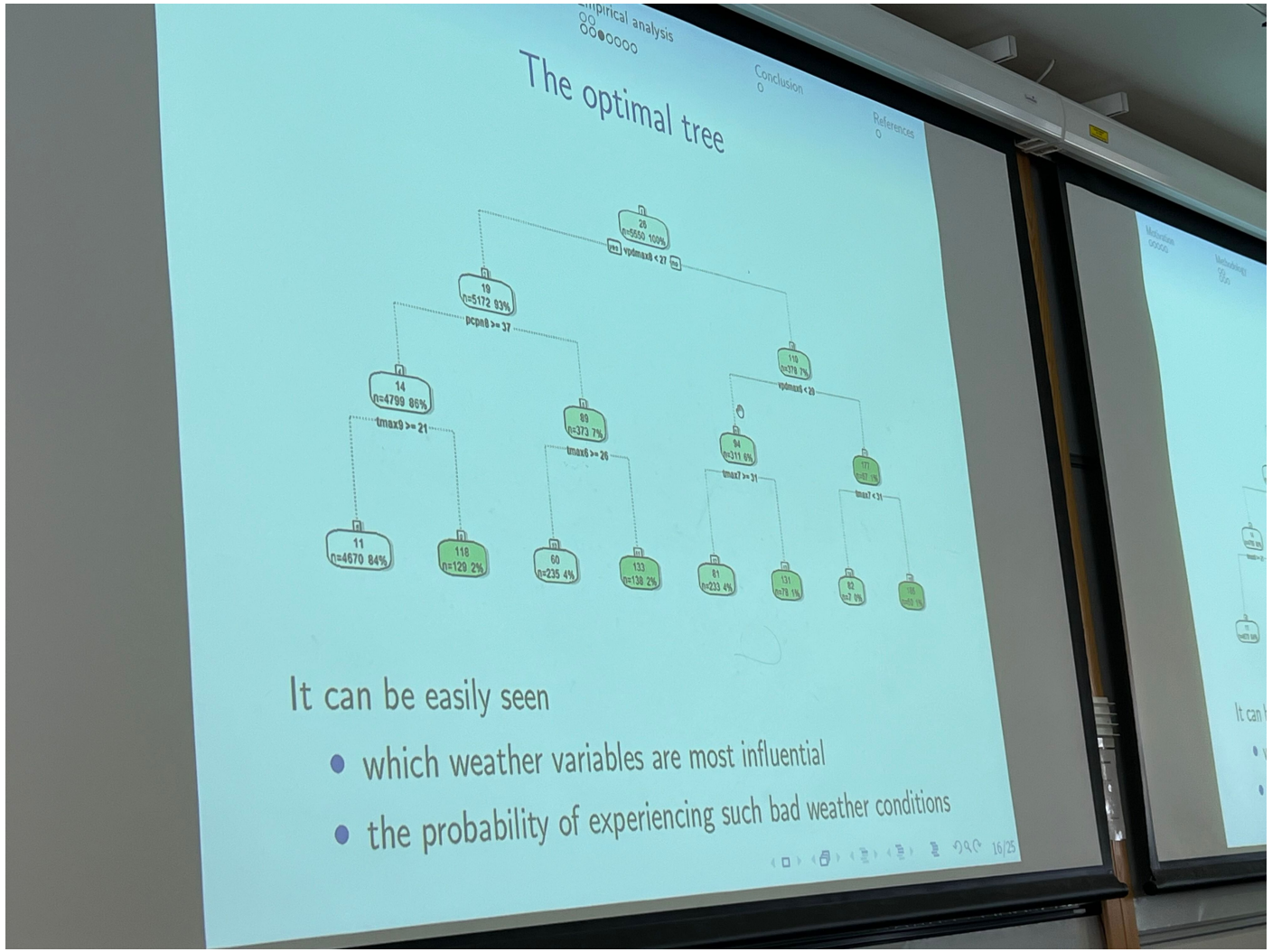
Binary train cost tree



Train prices: interpretable?



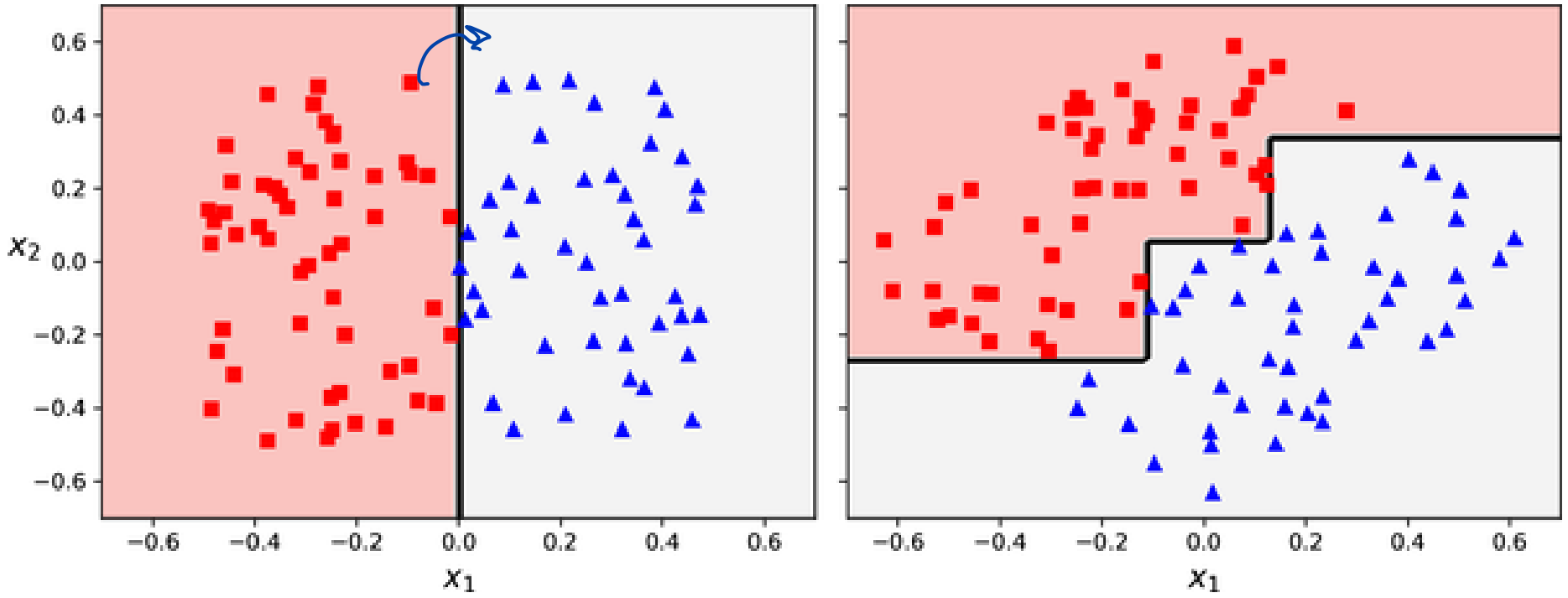
Interpretability?



Hand a decision tree to farmers to describe the payout structure of weather-based index/parametric insurance.

Source: Jinggong Zhang (2023), Integrated Design for Index Insurance, IME Presentation

Sensitivity to training data orientation



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

Source: Aurélien Geron (2022), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3rd Edition, Figure 6-7

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

model with high variance

(Bias - variance)

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

- Pruning a decision tree — Variable selection of GLM
- Ensemble methods
- Bagging, random forest, boosting

] General methods that work for GLM's as well

Pruning a Tree

Pruning

- Wish to have a tree that has a good balance of variance vs bias.
- A decision rule of considering the decrease in RSS at each step/split (vs a threshold) is too short sighted. ← A bad split now may result in a good split later
- 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)

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

$|T|$ = # leaves

Define the cost complexity criterion

If $\alpha = 0$, $|T| = 0$

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

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

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.

UNSW SYDNEY

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

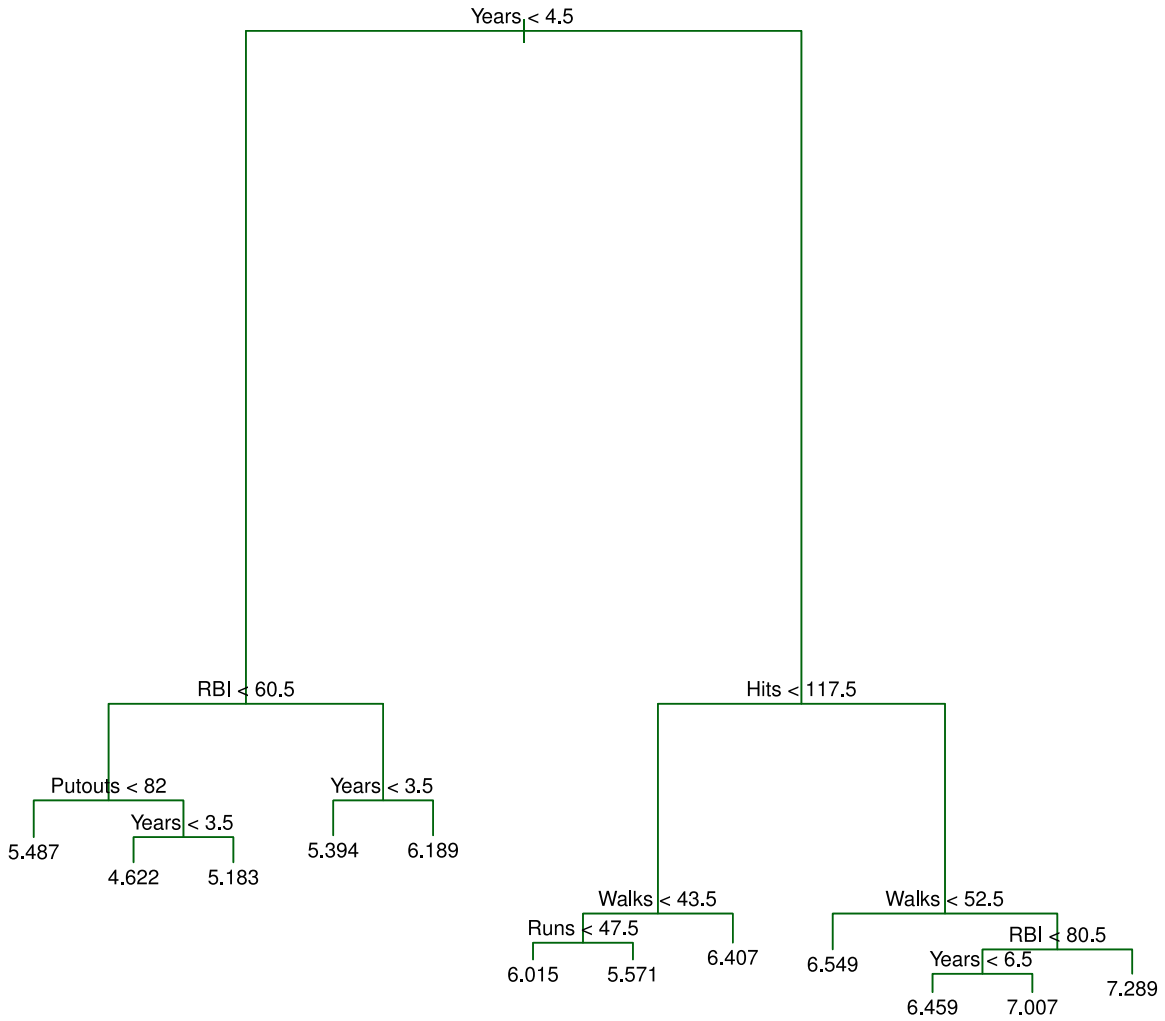
That as α changes, it prunes the tree in a nested and predictable way.

$T_0 \supseteq T_1 \supseteq T_{1.5} \supseteq T_{20}$

Tree Algorithm Summary – Pruned Trees

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

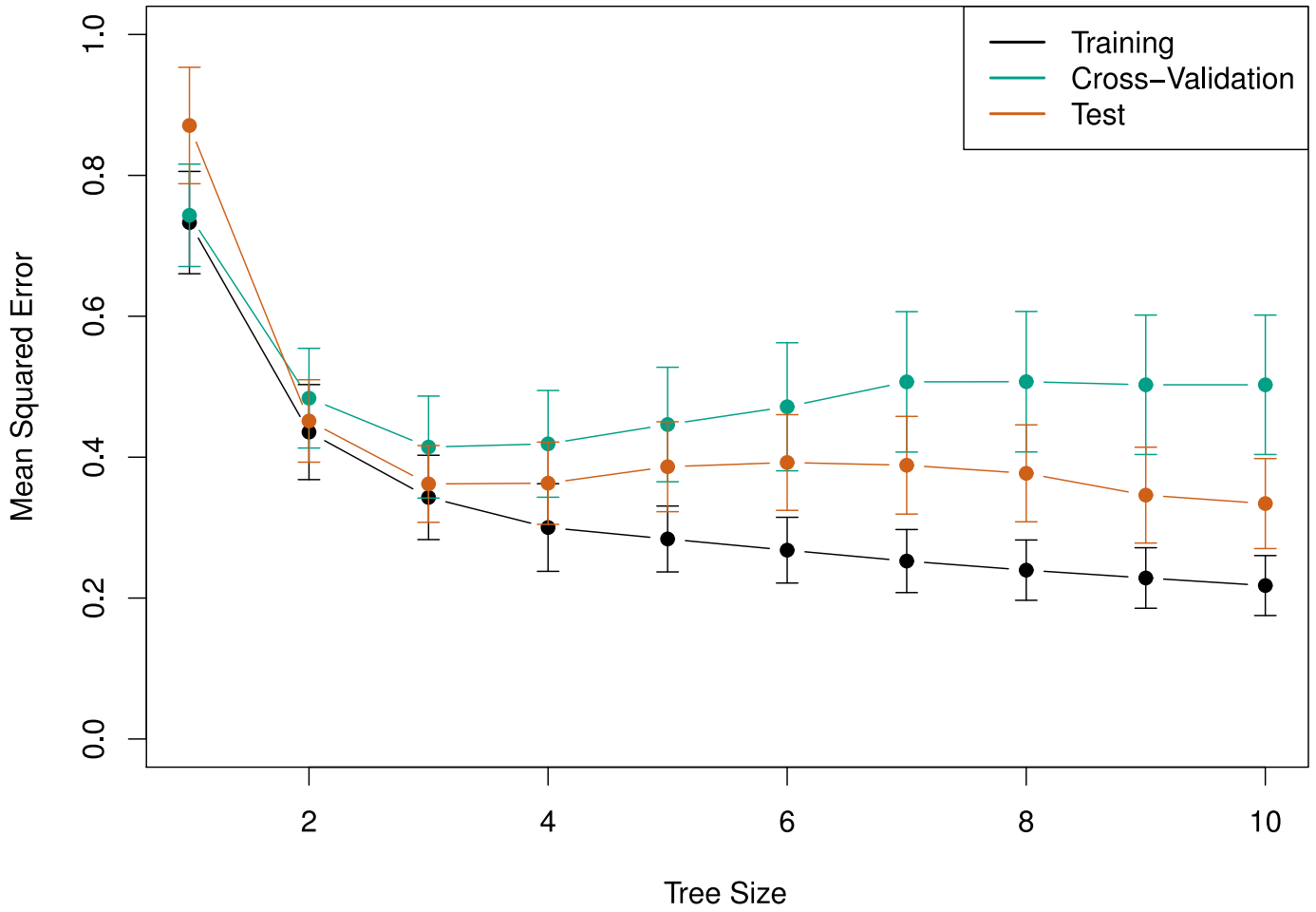


The unpruned tree that results from top-down greedy splitting on the training data.

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

UNSW SYDNEY

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.

Decision Tree Analysis Demo

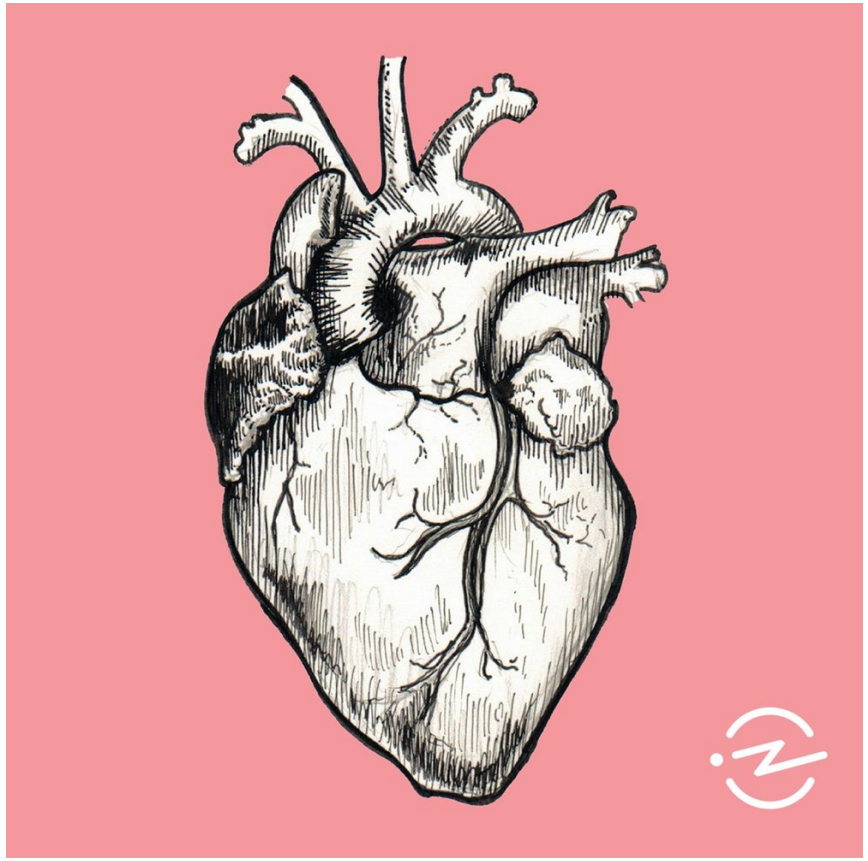
Does a patient have a heart disease?

Output (Y):

- Has heart disease; doesn't have heart disease

Input (X):

- Age
- Sex
- Chest pain
- Resting blood pressure
- Cholesterol
- …



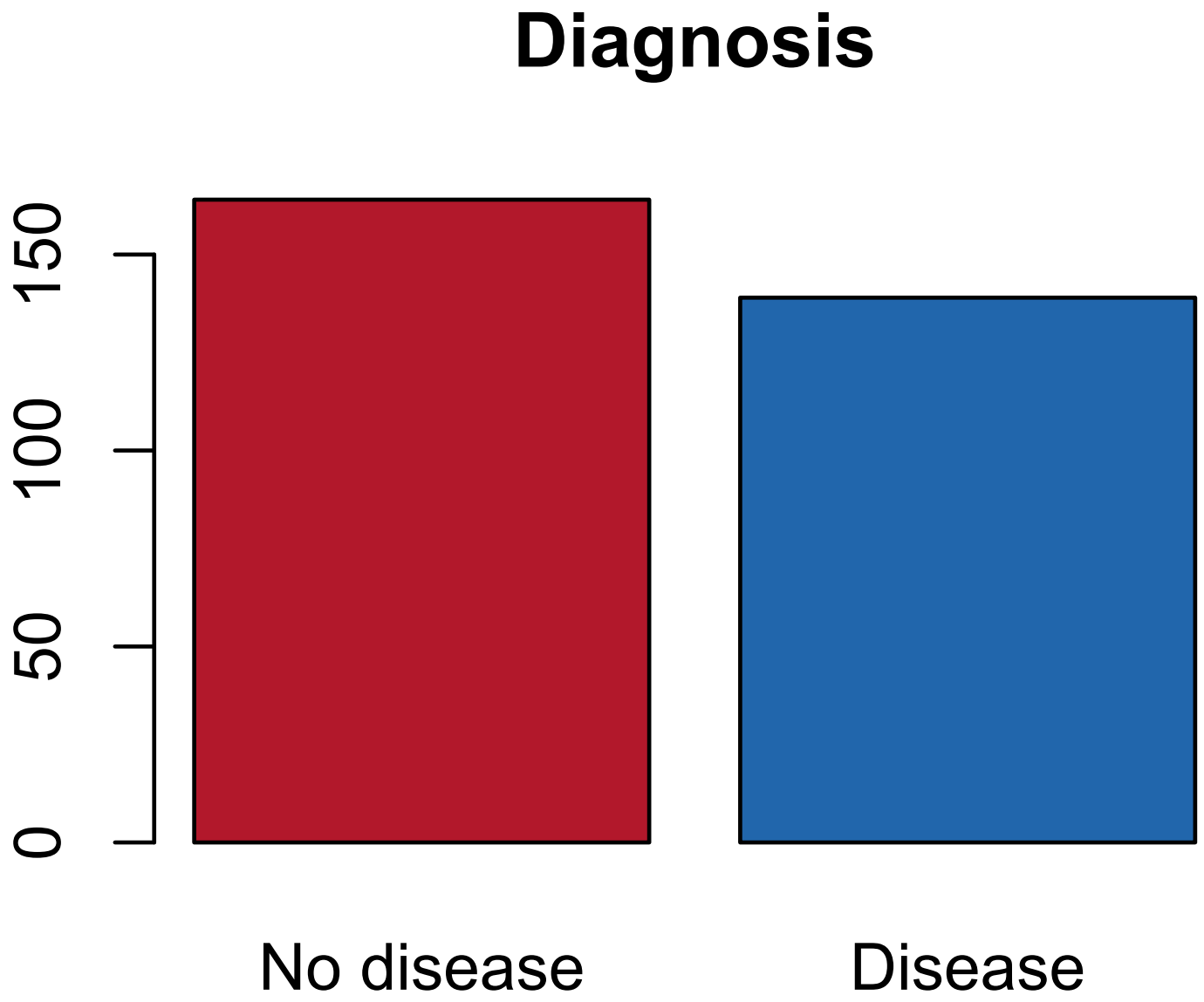
image

Source: https://www.radiotopia.fm/podcasts/the-heart

Cleveland Clinic Heart Disease

Dataset from RPubs of 313 patients and 14 variables in a Cleveland Clinic (RPubs, GitHub)

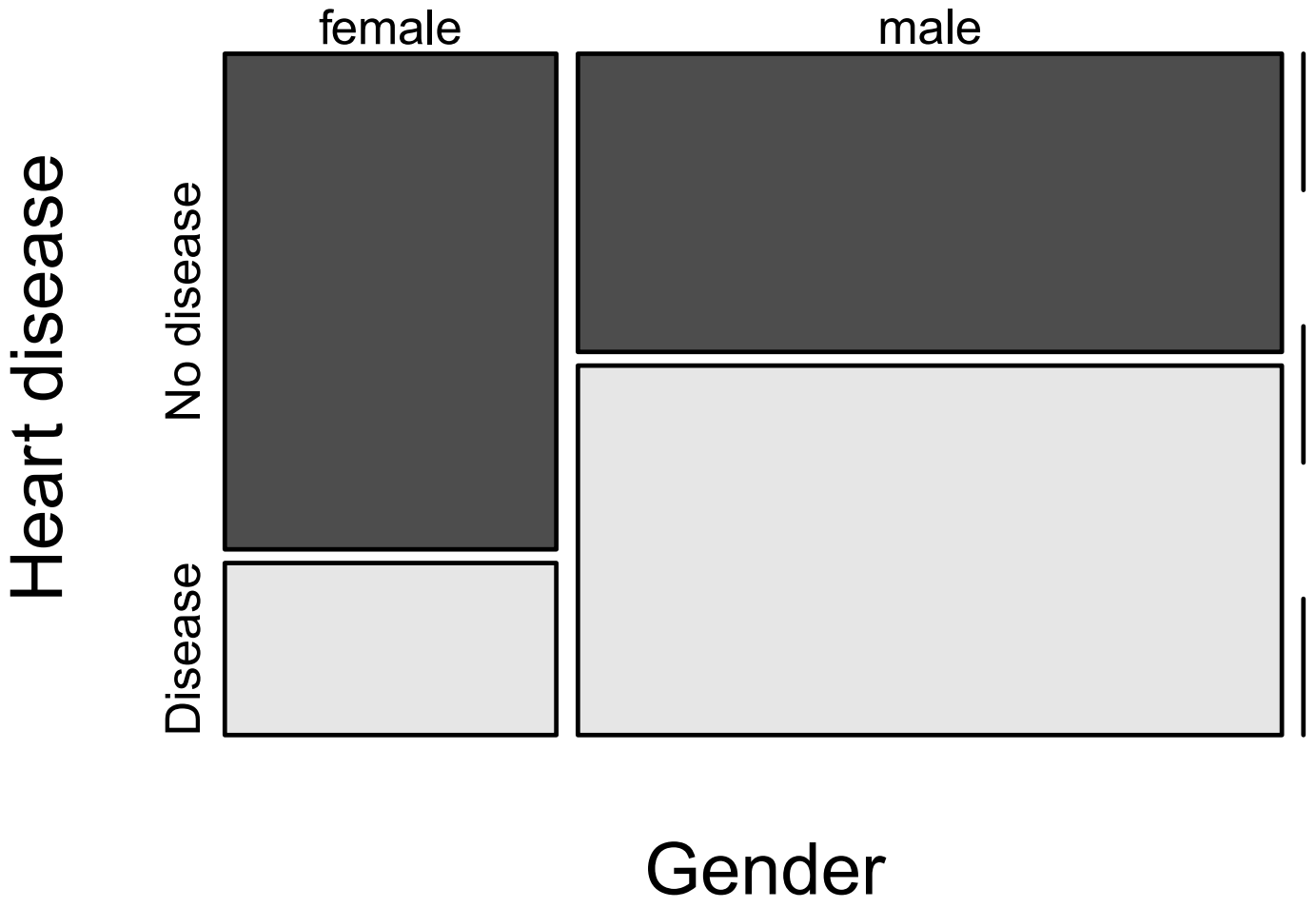
- `age`: Age of patient
- `sex`: Sex of patient
- `cp`: Chest pain
- `trestbps`: resting blood pressure
- `chol`: serum cholesterol
- `fbs`: fasting blood sugar larger 120mg/dl
- `restecg`: resting electroc. result anomality
- `thalach`: maximum heart rate achieved
- `exang`: exercise induced angina
- `oldpeak`: ST depression induc. ex.
- `slope`: slope of peak exercise ST
- `ca`: number of major vessel
- `thal`: thalassemia (3 normal; 6 fixed defect; 7 reversable defect)
- `num`: diagnosis of heart disease (angiographic disease status)



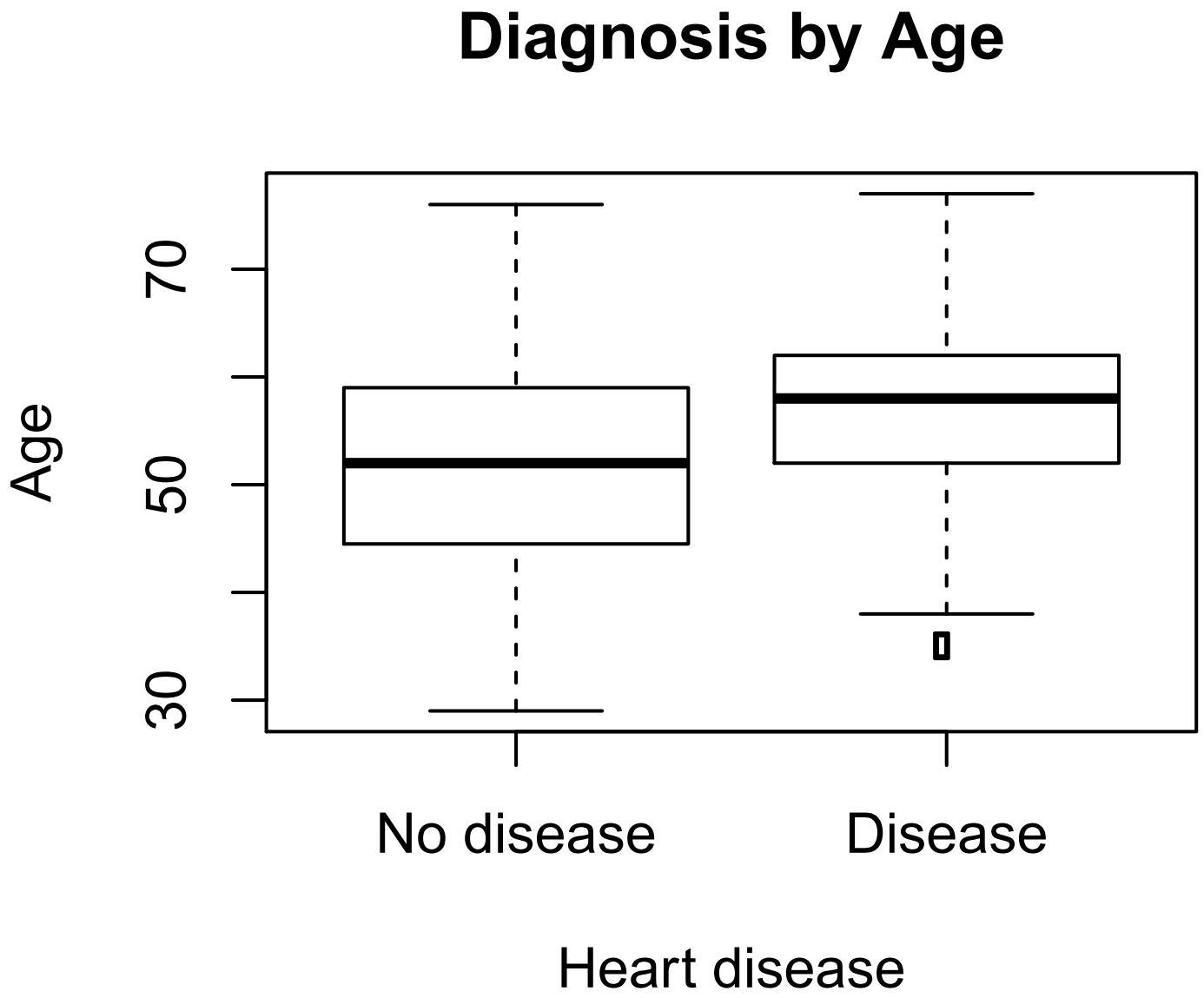
Histogram of diagnosis (target variable).

UNSW SYDNEY

Diagnosis by Gender

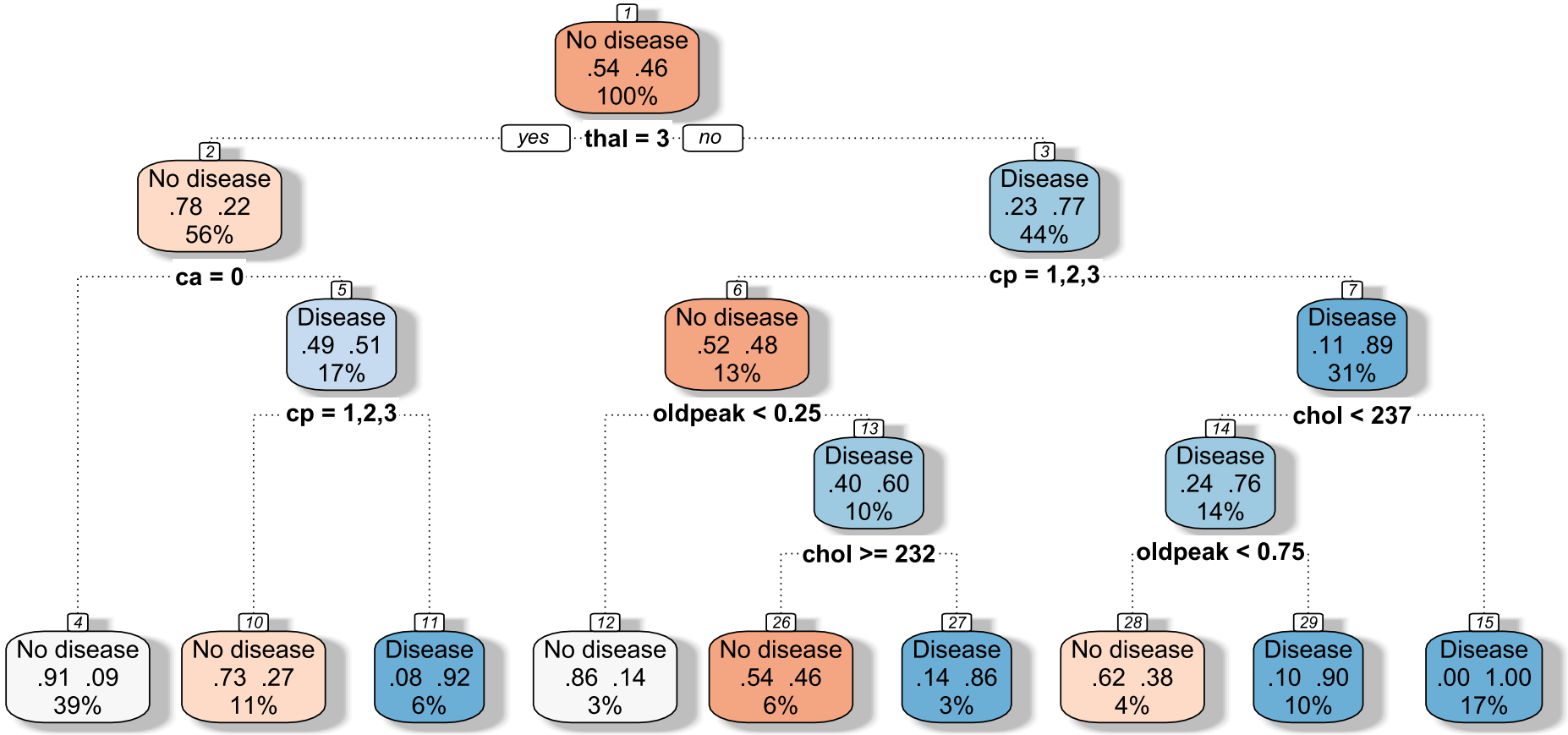


Histogram of diagnosis by gender.



Boxplot of age by diagnosis.

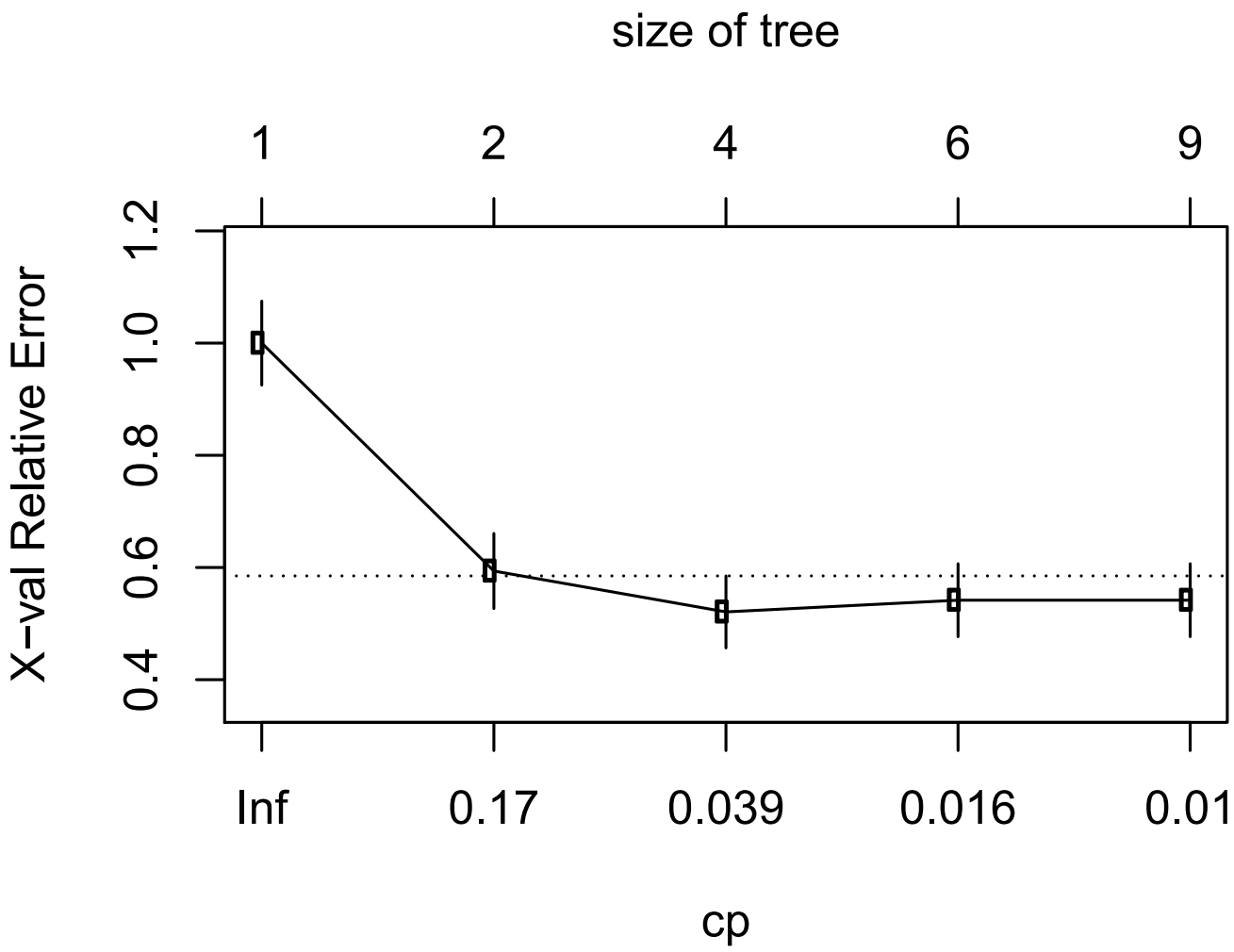
Unpruned tree



We first fit a relatively large decision tree to the training data.

UNSW SYDNEY

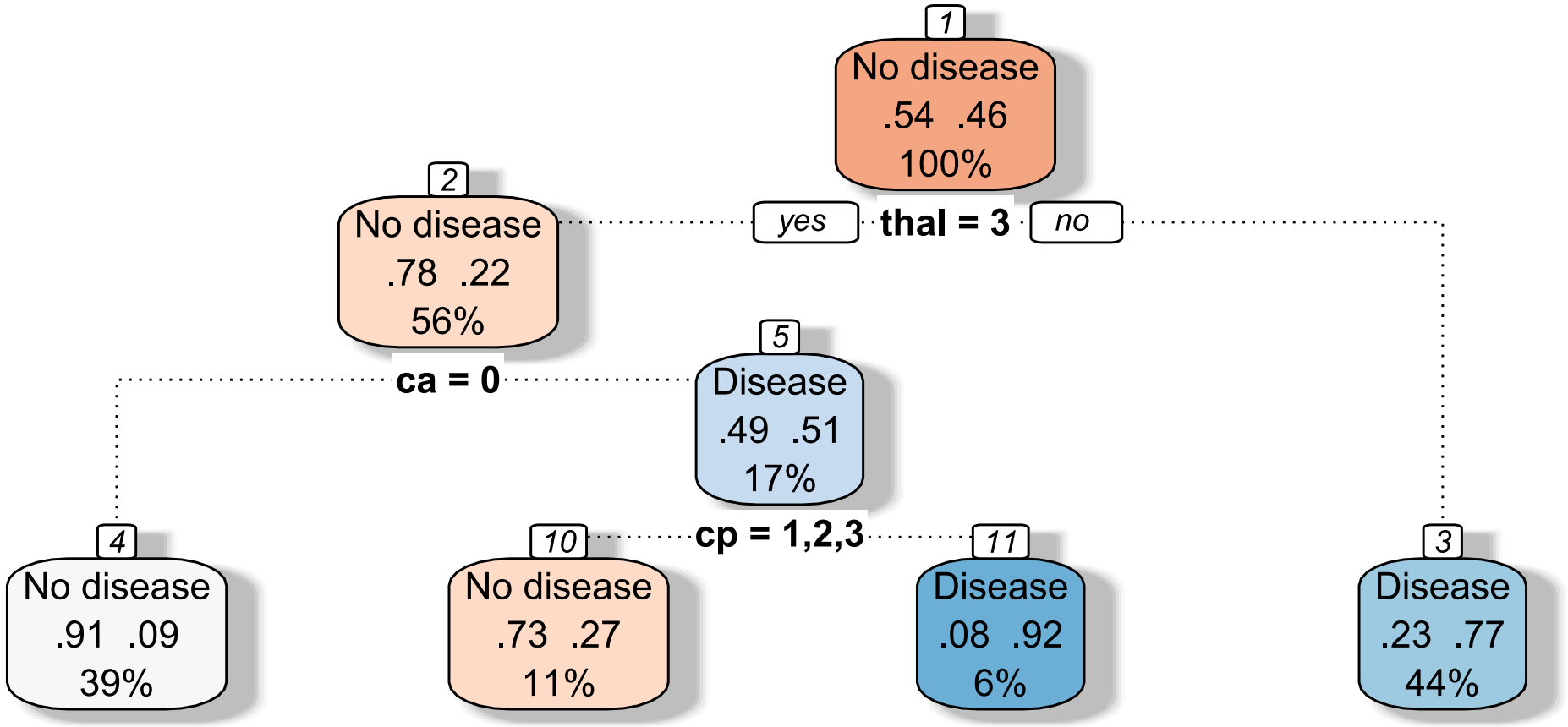
Pick the best subtree size



Using cross-validation we can find the value of α (equiv. `cp`) and hence $|T|$ that gives the best validation set accuracy.

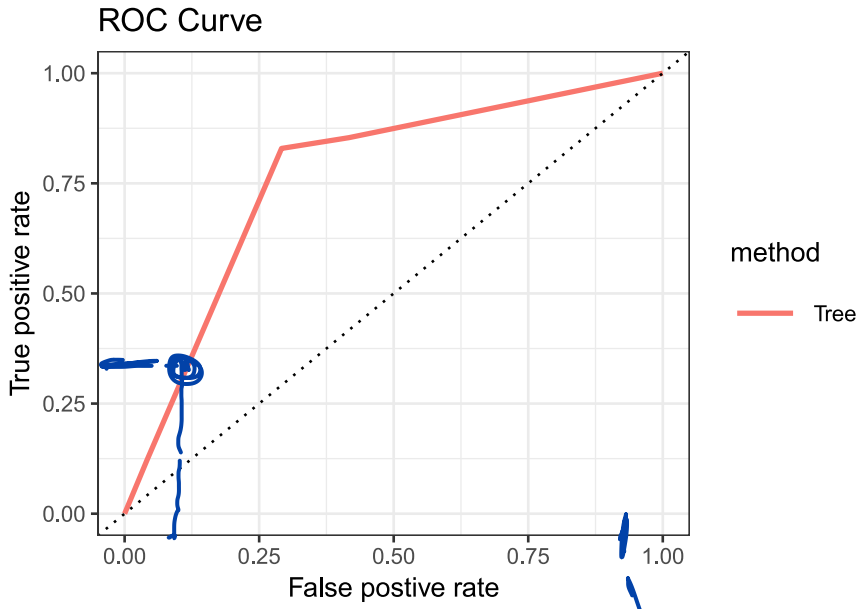
UNSW SYDNEY

Pruned tree



This is the resulting pruned tree.

Classification tree: Heart data (Accuracy)



Method	AUC	Accuracy
Tree	0.77	0.76

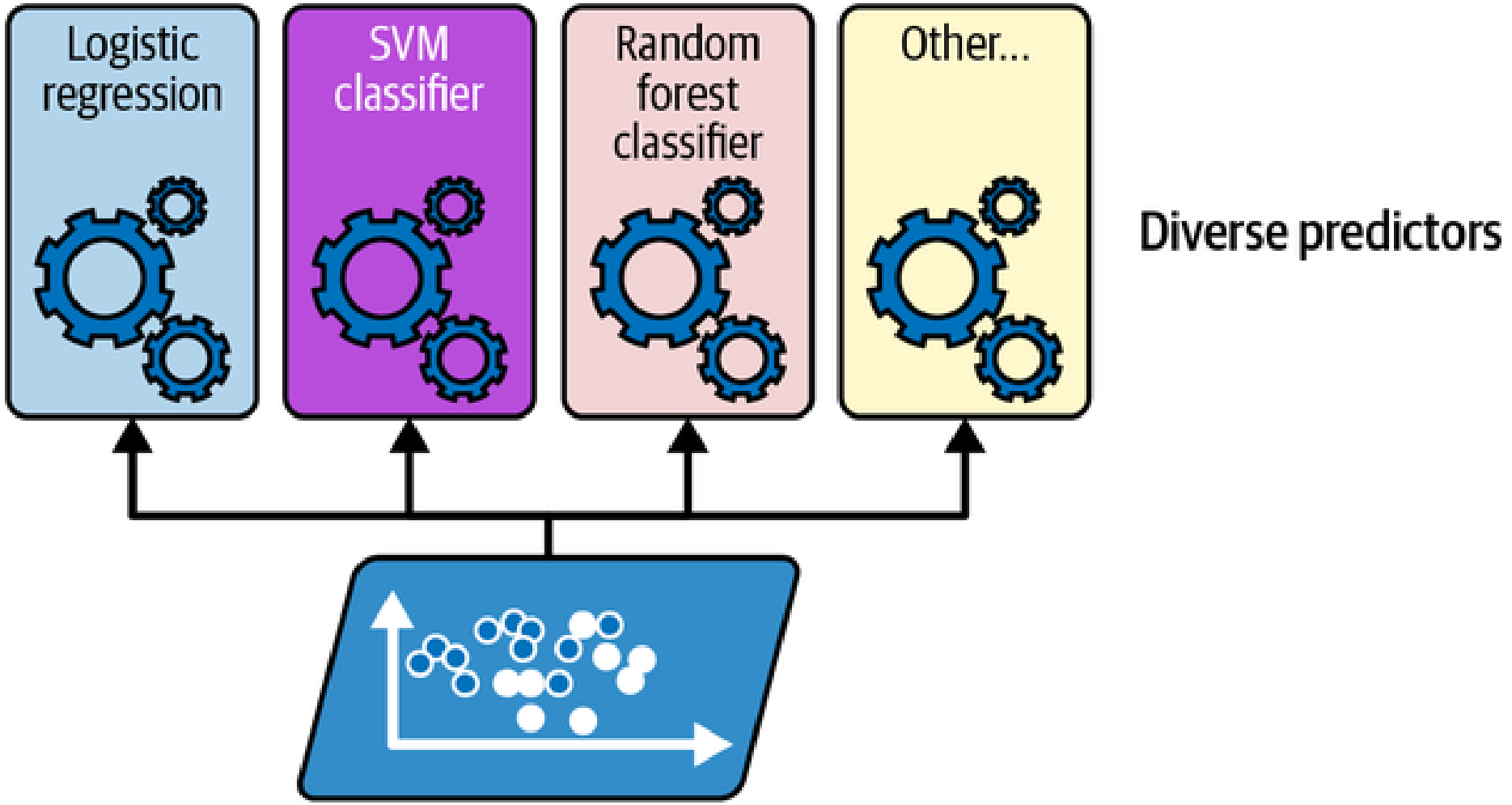
Evaluate the classification performance using a receiver operator characteristic curve.

Medical diagnosis
Criminal law system

Seeing a password is too common.

Ensembles & Bootstrap

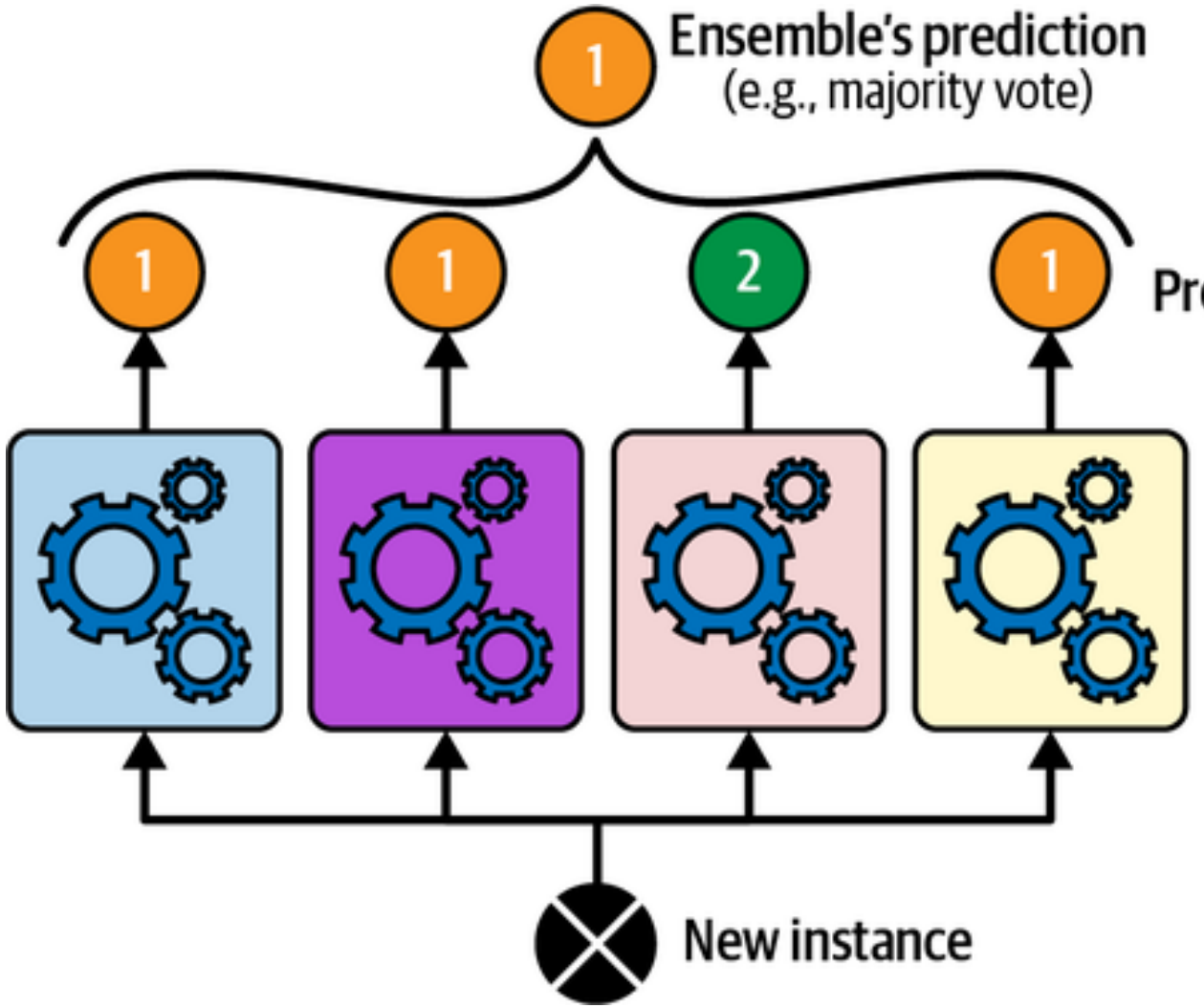
An ensemble is a group of models…



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



— Or on a threshold

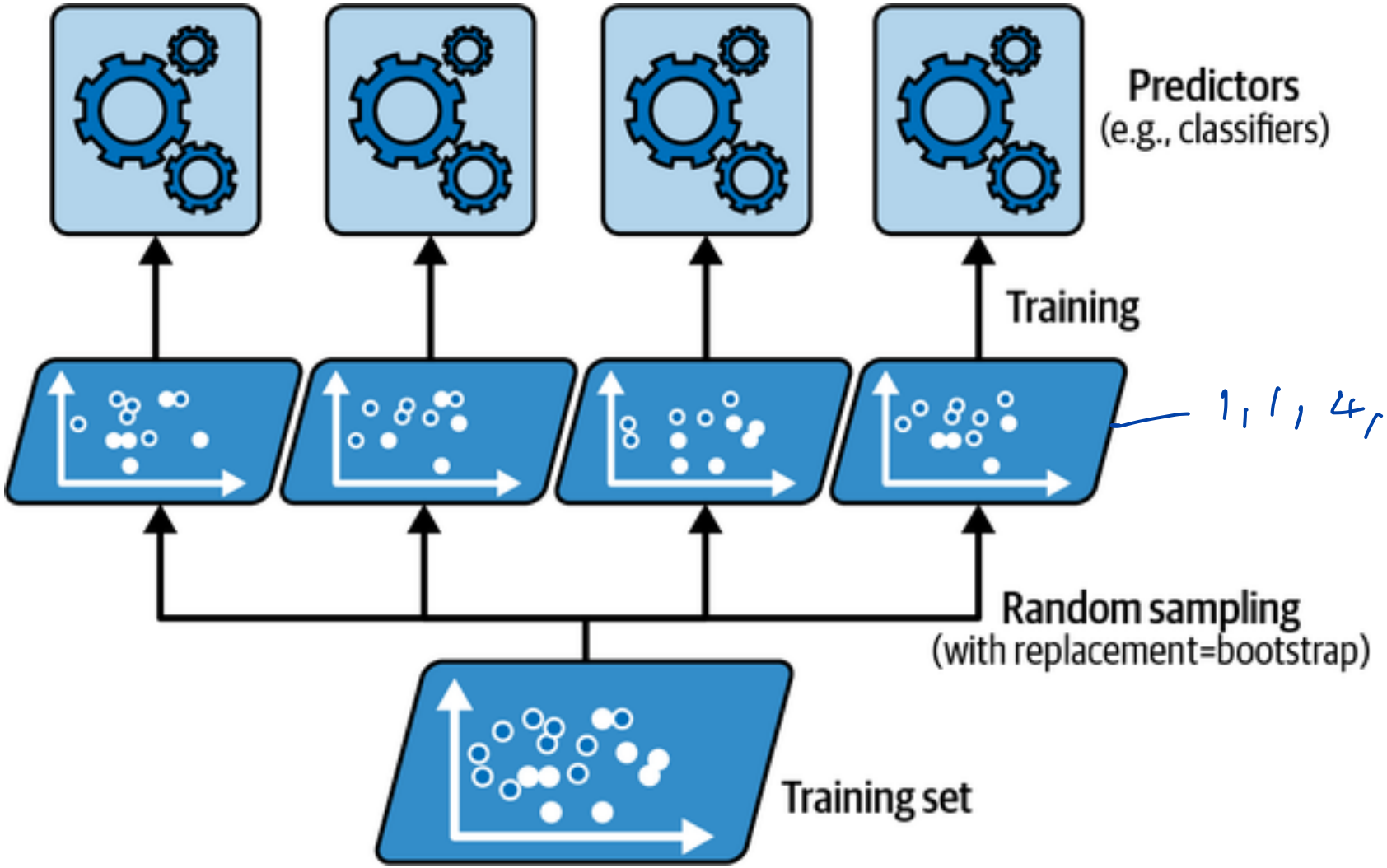
$$\mathbb{E}[\bar{X}] = \mu$$

$$Var(\bar{X}) = \frac{\sum \sigma_i^2}{n}$$

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

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

x1 <dbl>	x2 <dbl>
1.000000	6.7971014
1.051813	0.8260870
3.072539	1.6376812
3.383420	4.4202899
3.772021	9.1159420
4.497409	0.3333333
5.145078	2.9130435
5.455959	2.3913043
5.792746	5.2898551
6.051813	4.3623188

1-10 of 30 ro… Previous 1 2 3 Next

A bootstrap resample

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

x1 <dbl>	x2 <dbl>
1.000000	6.7971014
1.000000	6.7971014
3.072539	1.6376812
3.383420	4.4202899
3.383420	4.4202899
3.772021	9.1159420
3.772021	9.1159420
5.455959	2.3913043
5.455959	2.3913043
5.455959	2.3913043

1-10 of 30 ro… Previous 1 2 3 Next

There are 60% of the rows in the original dataset in the bootstrap resample.

UNSW SYDNEY

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>
1.000000	6.7971014
1.051813	0.8260870
3.072539	1.6376812
3.383420	4.4202899
3.772021	9.1159420
4.497409	0.3333333
5.145078	2.9130435
5.455959	2.3913043
5.792746	5.2898551
6.051813	4.3623188

1-10 of 30 ro… Previous 1 2 3 Next

A bootstrap resample

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

x1 <dbl>	x2 <dbl>
1.000000	6.7971014
1.051813	0.8260870
1.051813	0.8260870
3.072539	1.6376812
3.072539	1.6376812
3.772021	9.1159420
3.772021	9.1159420
4.497409	0.3333333
6.233161	9.8695652
6.233161	9.8695652

1-10 of 30 ro… Previous 1 2 3 Next

There are 66.67% of the rows in the original dataset in the bootstrap resample.

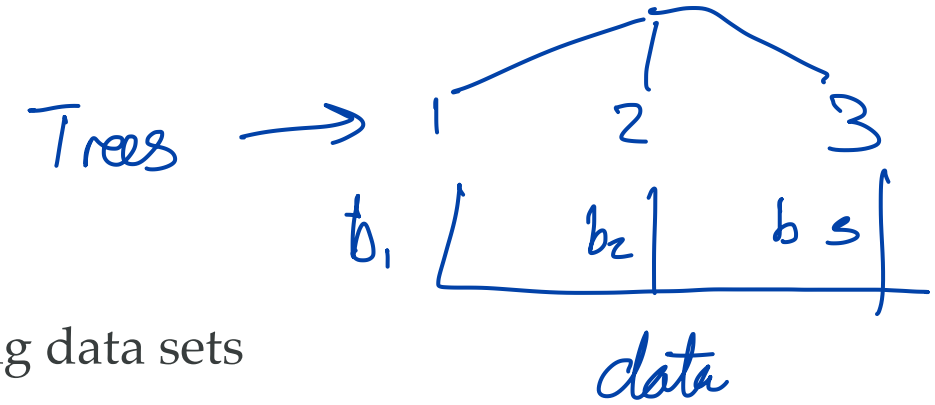
UNSW SYDNEY

Bagging and Random Forests

Bootstrap Aggregation (Bagging)

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

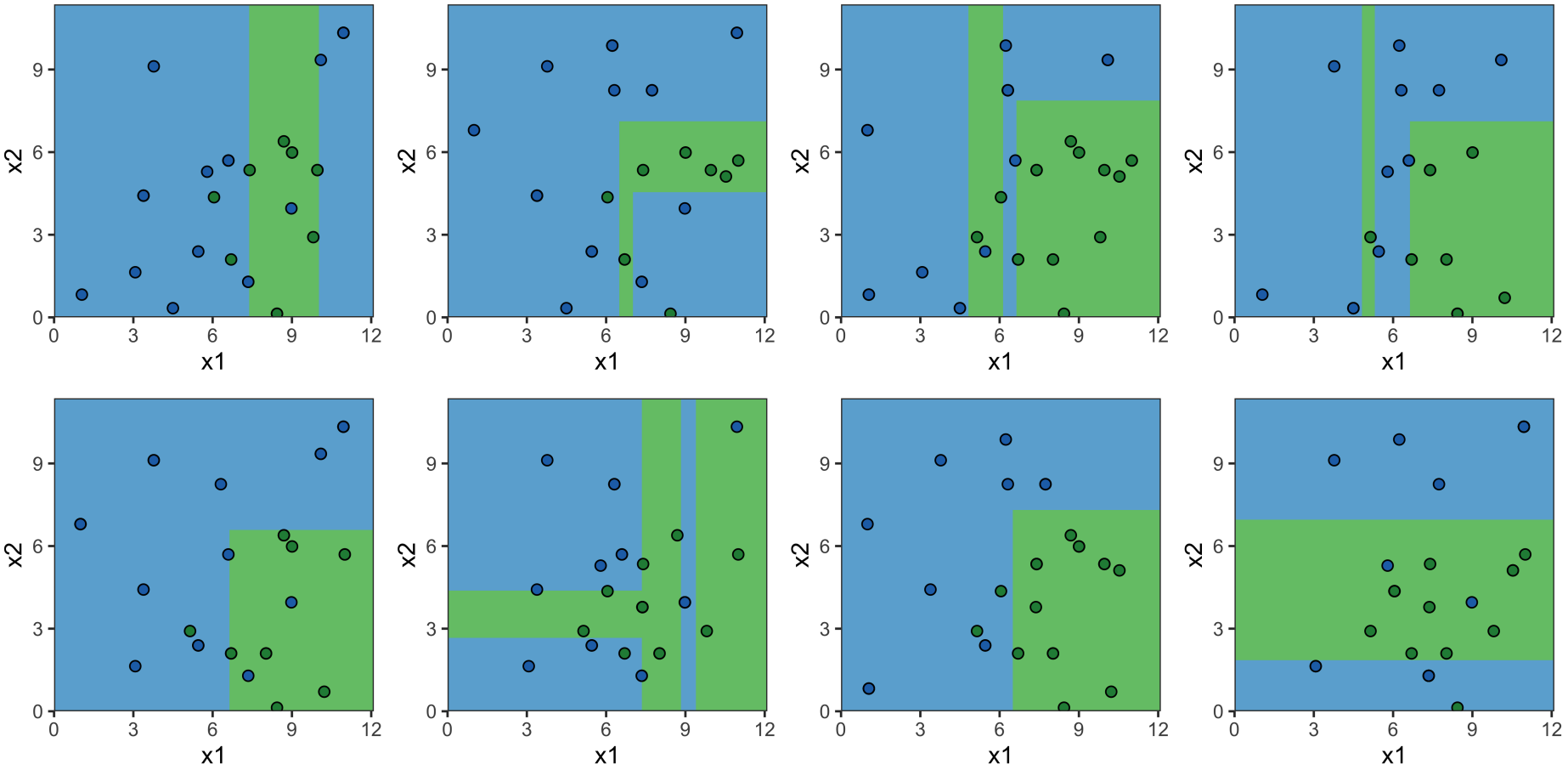
Bagging procedure:



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

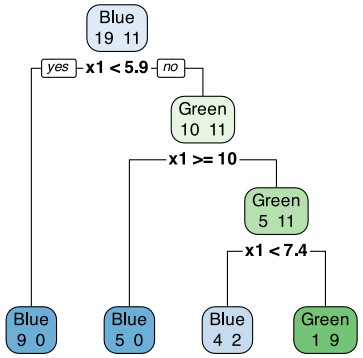
$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

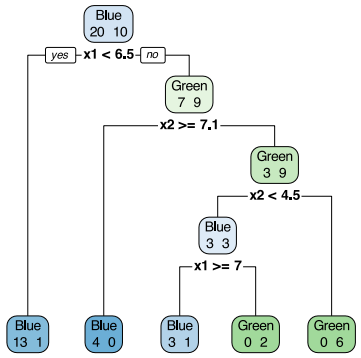
Bagging: Illustration

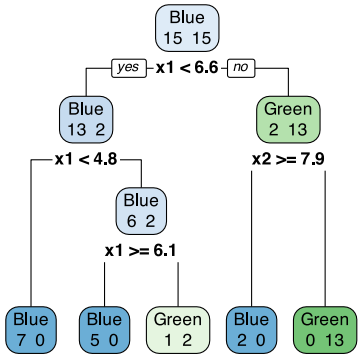


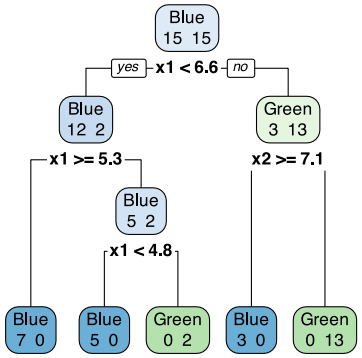
Each tree is probably different

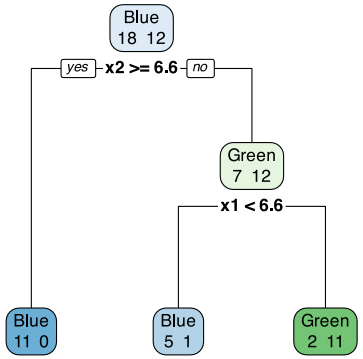
Bagging: Illustration

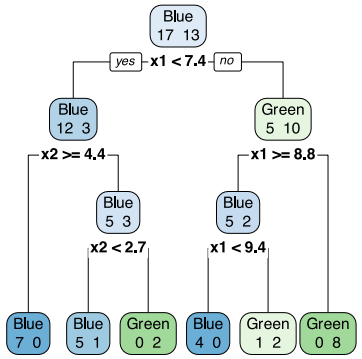


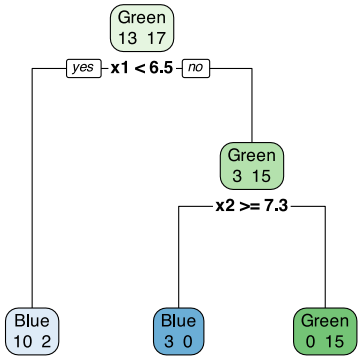


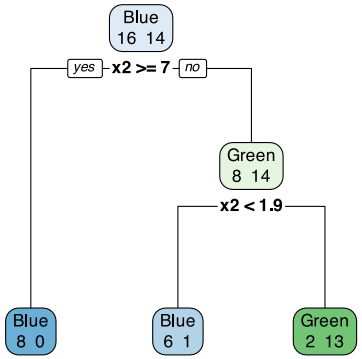












UNSW SYDNEY

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

On avg 1/3 of data is not included

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.

Out-of-Bag Error Estimation

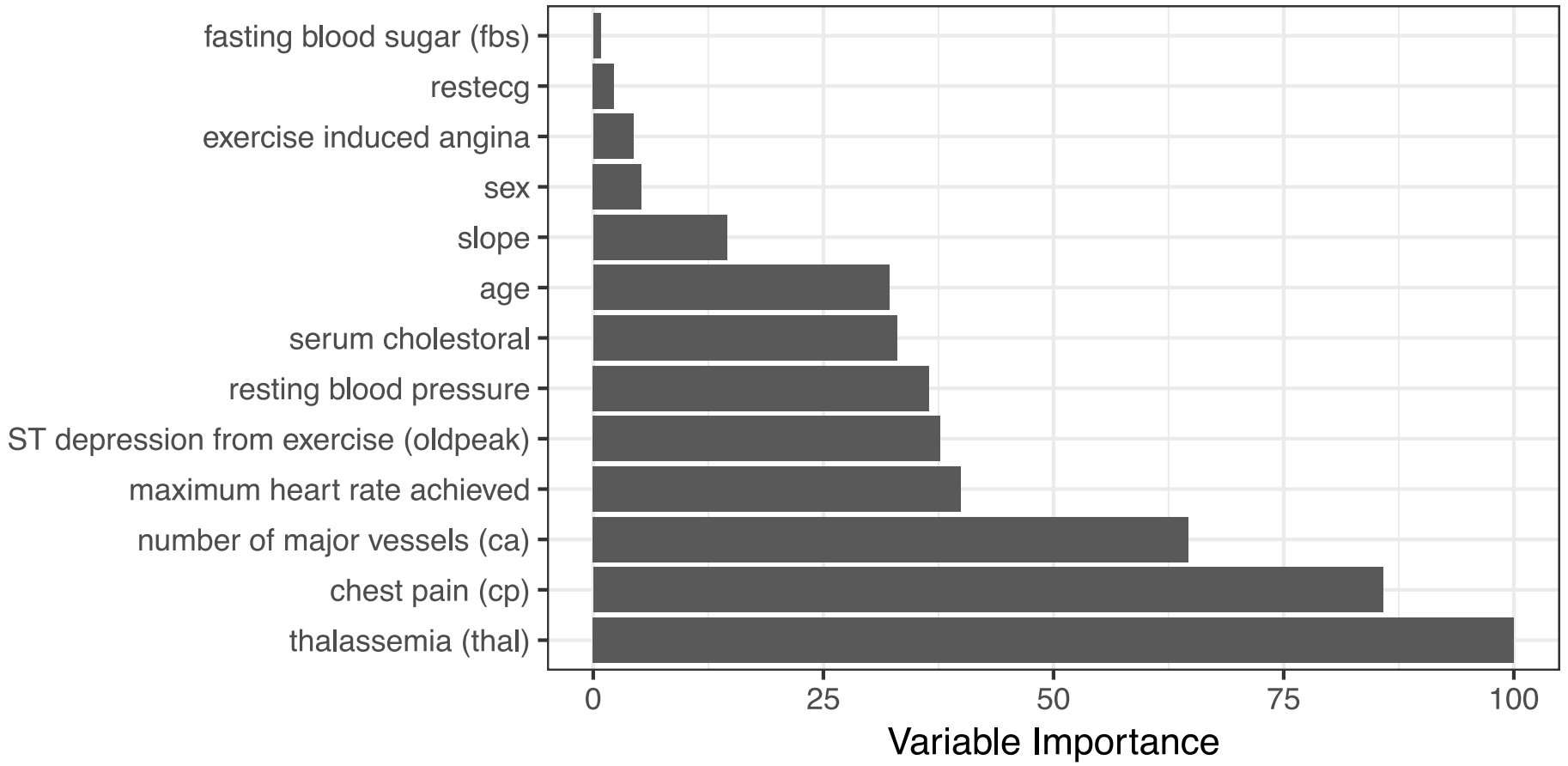
There is a very straightforward way to estimate the test error of a bagged model

- 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

Example of variable importance plot



Variable importance on the `Heart` dataset

Chest pain reduces Gini around 85% of Thal

Random Forests

Random forests decorrelates the bagged trees

, subset of predictors

- 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

chosen randomly

- 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 $m = p$

Now each tree should be less correlated

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

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

If $Y = f(x) + \varepsilon$

Then fit

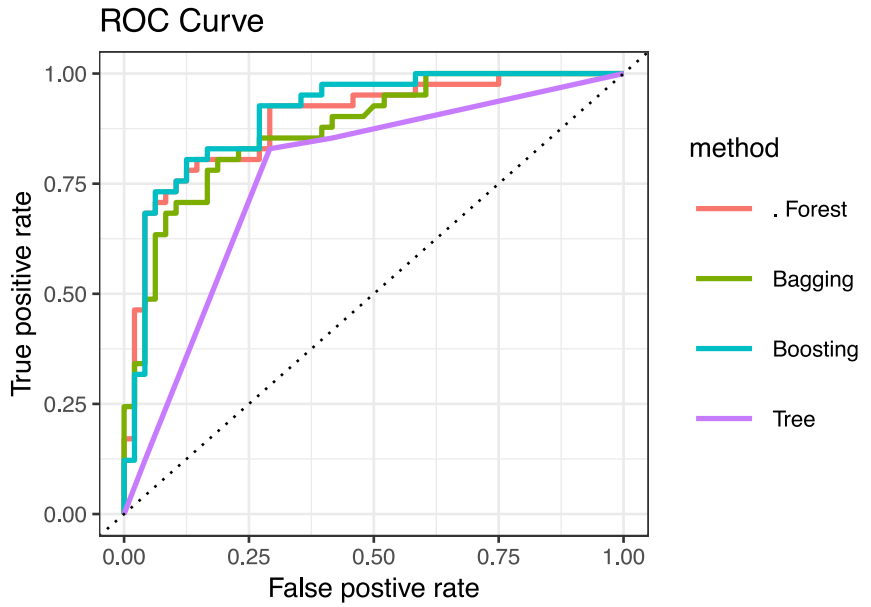
$\varepsilon = g(x) + \varepsilon^*$

$\varepsilon^* = h(x) + \varepsilon'$

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

Comparison of methods: Heart data



image

Method	AUC	Accuracy
Tree	0.77	0.76
Bagging	0.88	0.80
R. Forest	0.89	0.82
Boosting	0.91	0.82