Decision Trees

Playing Tennis

- Is it raining out?
  - Probably shouldn't play
- Is it really hot?
  - Yes: Maybe, is it also windy?
    - If yes, sure!
    - Otherwise, I'll pass
  - No: Sounds like a nice day, let's play!

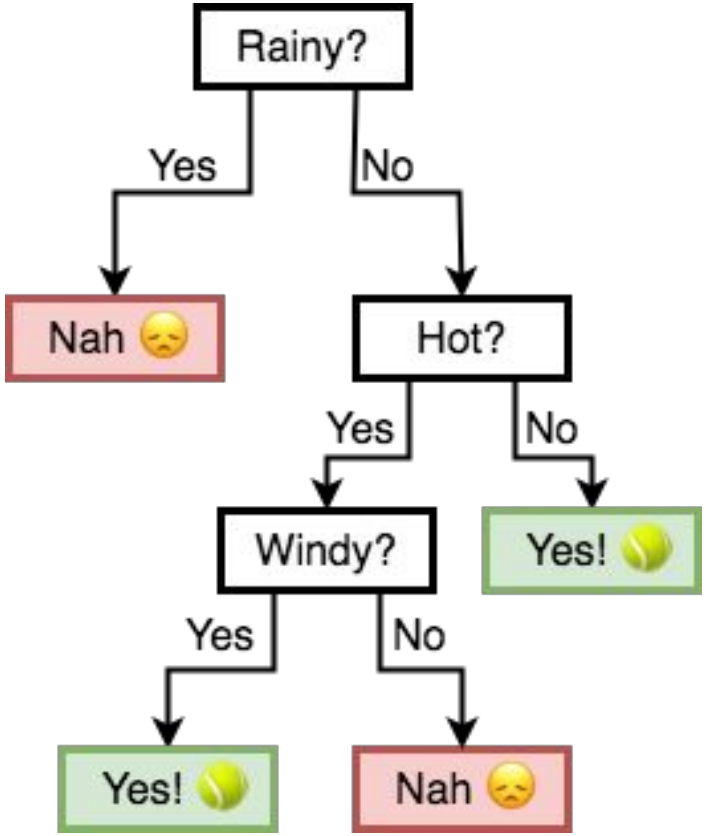


Image Credit: Will Povell (TA '17)

Decision Trees

- Modelled after flowcharts
- Main idea: Ask Yes/No questions until you learn enough to make a decision
- 20 questions: Is it bigger than a breadbox?
- Strategy: What questions should you ask first?

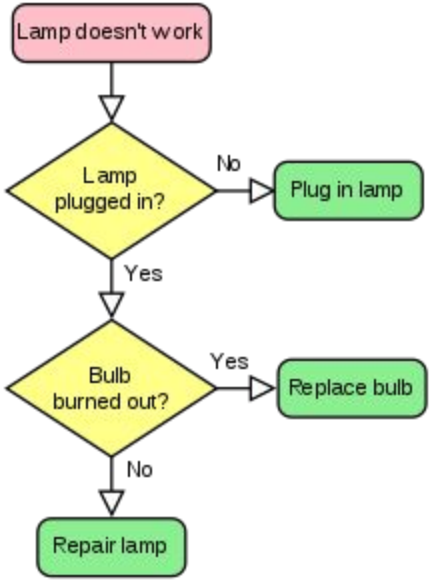
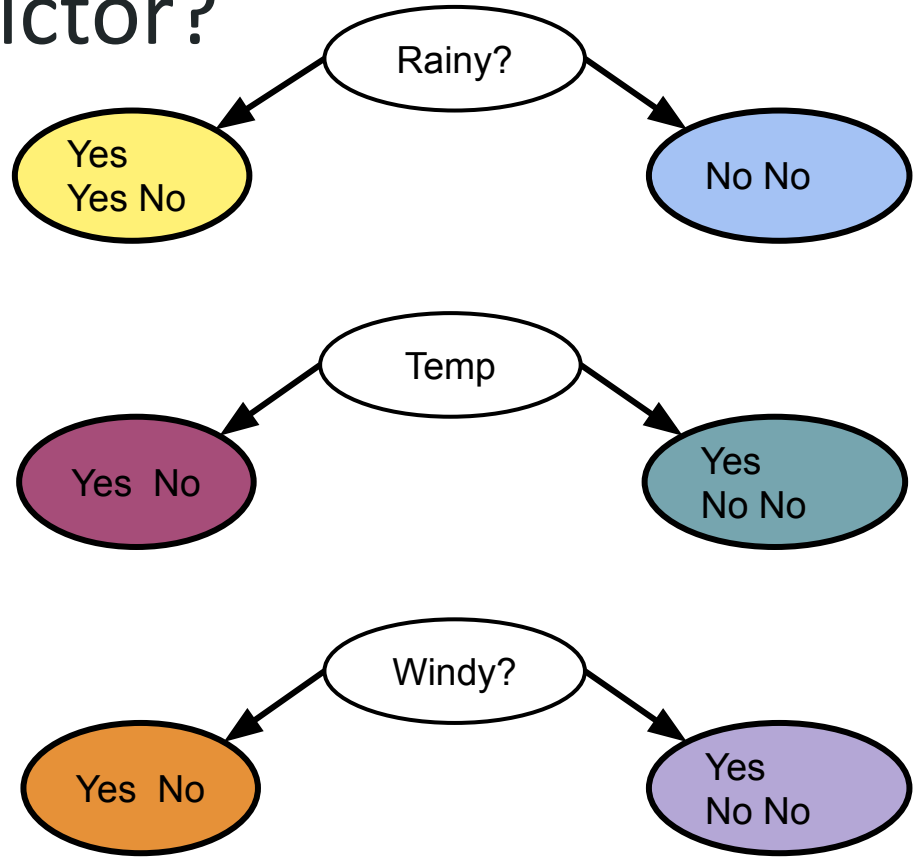


Image Source

Which is the best predictor?

Rainy?	Temp	Windy?	Play?
Sunny	Hot	Yes	Yes
Sunny	Cold	No	Yes
Sunny	Hot	No	No
Rainy	Cold	No	No
Rainy	Cold	Yes	No



Majority Vote

Sunny	67% Play 33% Don't Play
Rainy	0% Play 100% Don't Play

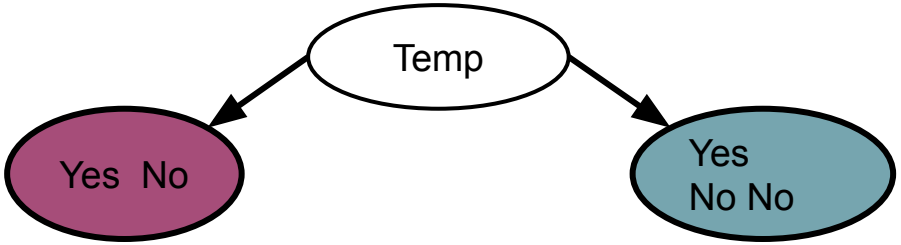
If Sunny, Play!
If Rainy, Don't Play!

The data say never play tennis on Rainy days,
and usually play on Sunny days.

Decision Heuristic: Majority vote

- None of the Rainy observations are classified incorrectly.
- But 33% of the Sunny observations are classified incorrectly.

Majority Vote



Hot	50% Play 50% Don't Play
Cold	33% Play 67% Don't Play

If Hot, (say) Play!
If Cold, Don't Play!

The data say sometimes play tennis on Hot days, and don't usually play on Cold days.

Decision Heuristic: Majority vote

- 50% of the Hot observations are classified incorrectly.
- 33% of the Cold observations are classified incorrectly.

Majority Vote

Windy	50% Play 50% Don't Play
Not Windy	33% Play 67% Don't Play

If Windy, (say) Play!
If Not Windy, Don't Play!

The data say sometimes play tennis on Windy days, and don't usually play on non-Windy days.

Decision Heuristic: Majority vote

- 50% of the Windy observations are classified incorrectly.
- 33% of the Not Windy observations are classified incorrectly.

Measure of Impurity

Majority vote classification rule: classify via the mode, $\max(p_1, p_2)$.

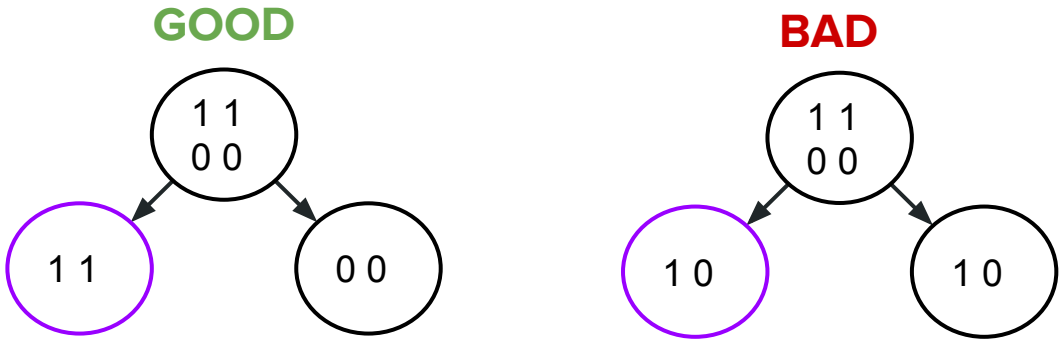
Here, p_1 and p_2 are the percentages in class 1 (YES, play) and 2 (NO, don't play), respectively, at a node after a split.

We then calculate the misclassification error:

- $\max(p_1, p_2)$ are classified correctly
- $1 - \max(p_1, p_2)$ are classified incorrectly

In binary classification, assuming majority vote, the misclassification error $1 - \max(p_1, p_2)$ is necessarily $\leq ½$.

To ask good questions, minimize impurity!



- p_1 = % belonging to class 1
- p_2 = % belonging to class 2

- **GOOD**: Error = 1 - max(100%, 0%) = 1 - 1 = 0
- **BAD**: Error = 1 - max(50%, 50%) = 1 - 0.5 = 0.5

Note that the error at *both* **GOOD** nodes and the error at *both* **BAD** nodes are equal, although of course the errors at the **GOOD** nodes and the errors at the **BAD** nodes differ.

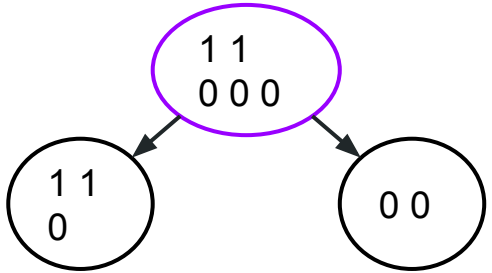
To ask good questions, minimize impurity!

Rainy?

Error(Left) = 1 - max(67%, 33%) = 0.33
Error(Right) = 1 - max(0%, 100%) = 0.0

Weight(Left) = 60%
Weight(Right) = 40%

Weighted Error = (0.33 x 0.6) + (0.0 x 0.4) = 0.2



Weather	Temp	Windy?	Play?
Sunny	Hot	Yes	Yes
Sunny	Cold	No	Yes
Sunny	Hot	No	No
Rainy	Cold	No	No
Rainy	Cold	Yes	No

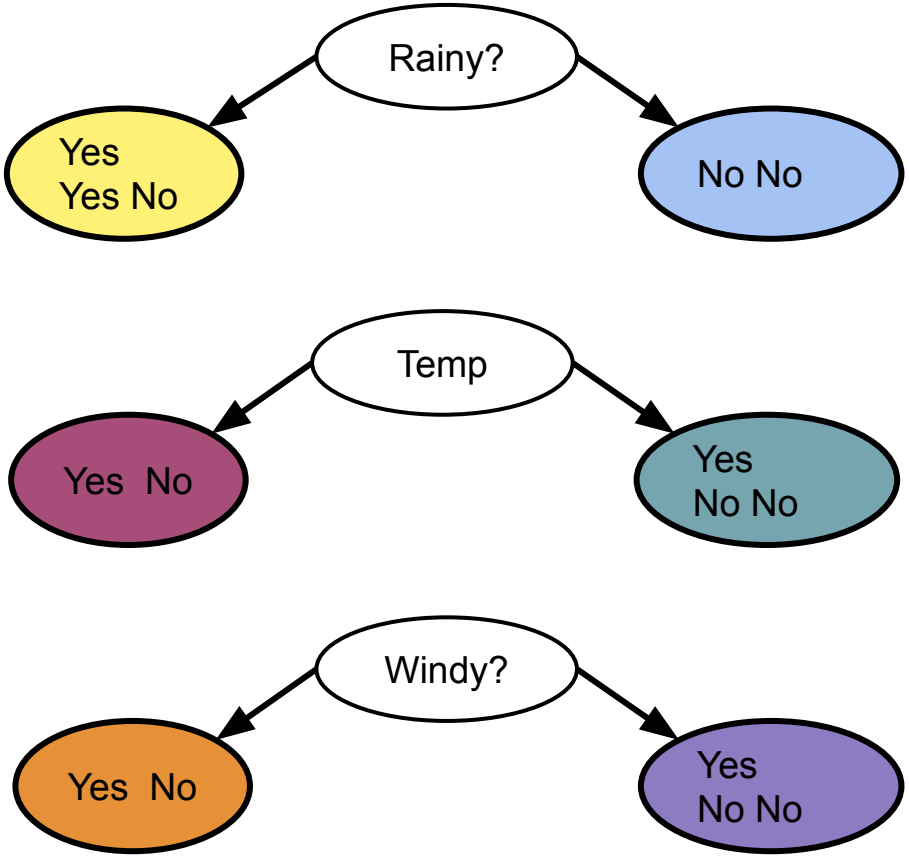
Which is the best predictor?

Sunny	67% Play 33% Don't Play
Rainy	0% Play 100% Don't Play

Hot	50% Play 50% Don't Play
Cold	33% Play 67% Don't Play

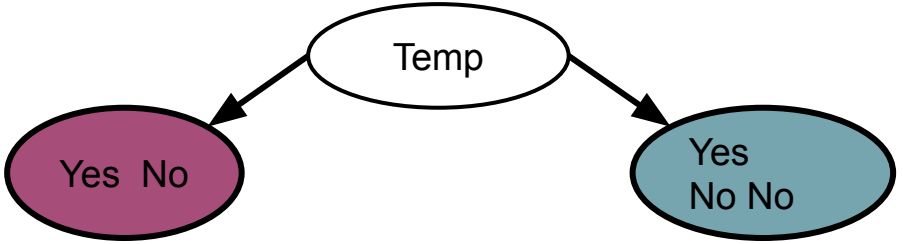
Windy	50% Play 50% Don't Play
Not Windy	33% Play 67% Don't Play

Error(Left) = 1 - max(66%, 33%) = 0.33
Error(Right) = 1 - max(0%, 100%) = 0.0
Weight(Left) = 0.6, Weight(Right) = 0.4
I = (0.6 * 0.33) + (0.4 * 0.0) = 0.2



Error(Left) = 1 - max(66%, 33%) = 0.33
Error(Right) = 1 - max(0%, 100%) = 0.0
Weight(Left) = 0.6, Weight(Right) = 0.4
I = (0.6 * 0.33) + (0.4 * 0.0) = 0.2

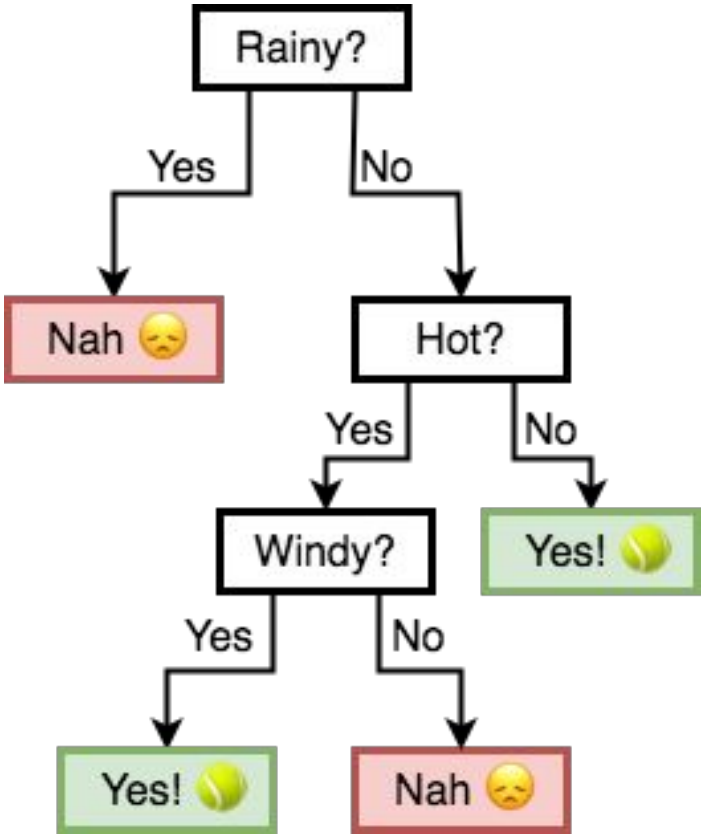
Error(Left) = 1 - max(50%, 50%) = 0.5
Error(Right) = 1 - max(33%, 66%) = 0.33
Weight(Left) = 0.4, Weight(Right) = 0.6
I = (0.4 * 0.5) + (0.6 * 0.33) = 0.4



Error(Left) = 1 - max(50%, 50%) = 0.5
Error(Right) = 1 - max(33%, 66%) = 0.33
Weight(Left) = 0.4, Weight(Right) = 0.6
I = (0.4 * 0.5) + (0.6 * 0.33) = 0.4

The Algorithm

- Start at the root of the tree, with all observations
- Score all the questions using current set of observations
- Split current set of observations by the question with the best score
- Repeat until all observations are contained in just one class, or all observations' answers are identical (i.e., no further information is available to differentiate among classes)
- Given a new observation, classify by walking the tree according to the answers to the questions



Pros and Cons of Decision Trees

Pros:

- Interpretable
- Suitable for both quantitative and categorical features (i.e., questions)
- Suitable for both quantitative and categorical labels (i.e., regression)
  (Regression trees coming soon!)

Cons:

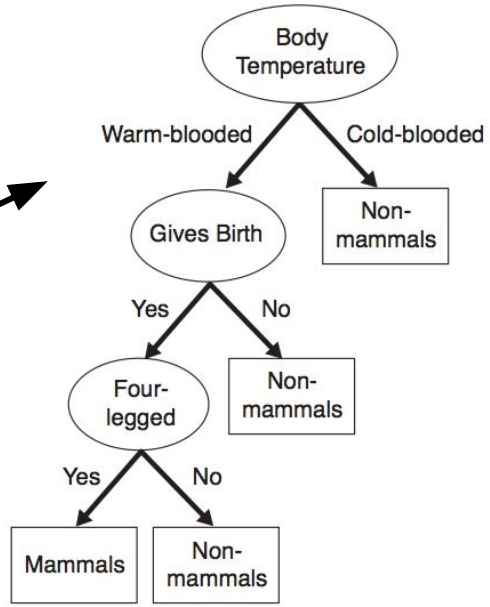
- Low bias
- High variance: a small change in the training data leads to a very different tree
- Easy to overfit!
  - Learn perfectly on training data, but generalize poorly to test data

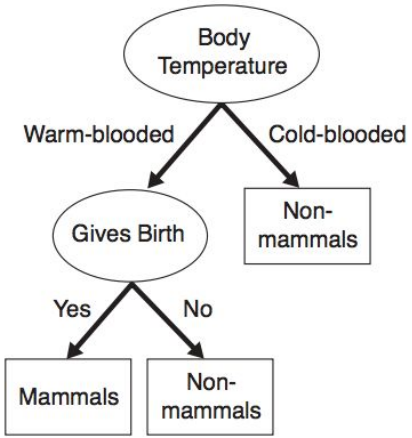
Classifying Mammals vs. Non-mammals

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

Image Source

Overfitting





Fits training data perfectly; accuracy is poor on test data!

Accuracy is less than perfect on training data; but tree is simpler, and accuracy is better on test data!

Image Source

Classifying Mammals vs. Non-mammals

Mislabeled!

Training data

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

Test data

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
human	warm-blooded	yes	no	no	yes
pigeon	warm-blooded	no	no	no	no
elephant	warm-blooded	yes	yes	no	yes
leopard shark	cold-blooded	yes	no	no	no
turtle	cold-blooded	no	yes	no	no
penguin	cold-blooded	no	no	no	no
eel	cold-blooded	no	no	no	no
dolphin	warm-blooded	yes	no	no	yes
spiny anteater	warm-blooded	no	yes	yes	yes
gila monster	cold-blooded	no	yes	yes	no

Image Source

Overfitting

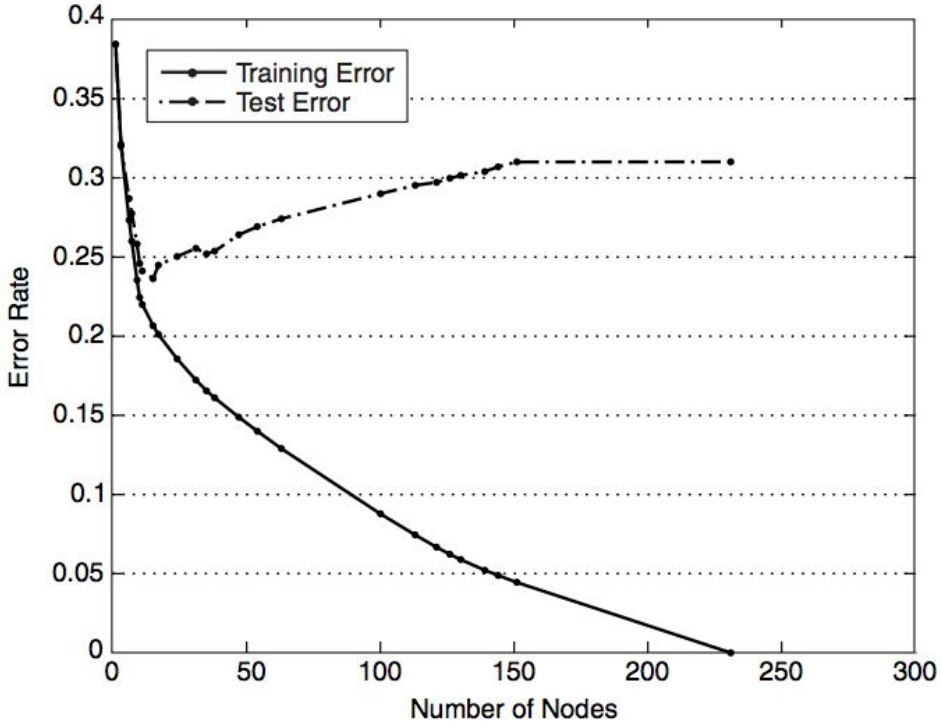


Image Source

Key Design Decisions

What is the best size for the tree?

- Pre-pruning
  - Stop growing the tree when:
    - its depth reaches some threshold
    - there are fewer than some threshold number of observations at a node
    - when the impurity measure no longer decreases by “enough”
      - But how much is “enough”? It is difficult, if not impossible, to know.
- Post-pruning
  - Replace small subtrees with leaf nodes
  - Determine class by majority vote among observations in the subtree

Model Selection

Find a model that appropriately balances complexity and generalization capabilities: i.e., that optimizes the bias-variance tradeoff.

- Low bias, high variance
  - Trees of unlimited depth and (no minimum) node size
- High bias, low variance
  - Set some minimum node size, only adding predictors whose children are big enough
  - Set some minimum improvement threshold, only adding predictors that are good enough

Other complications

- How to fill in missing values
- How to split on numerical values
  - Temperature is in degrees rather than hot vs cold
  - One option is to bin values (> 75 vs. < 75)
- How to handle noisy labels: identical observations with different labels

Decision Trees in R

Decision Trees in R

- R provides the `rpart` package for decision trees
- The package create trees as follows:

```
> library(rpart)
> fit <- rpart(city ~ elevation + beds + bath + sqft)
```

- Trees can be visualized using the `rpart.plot` library:

```
> library(rpart.plot)
> rpart.plot(fit, type = “class”)
```

Controlling `rpart` models

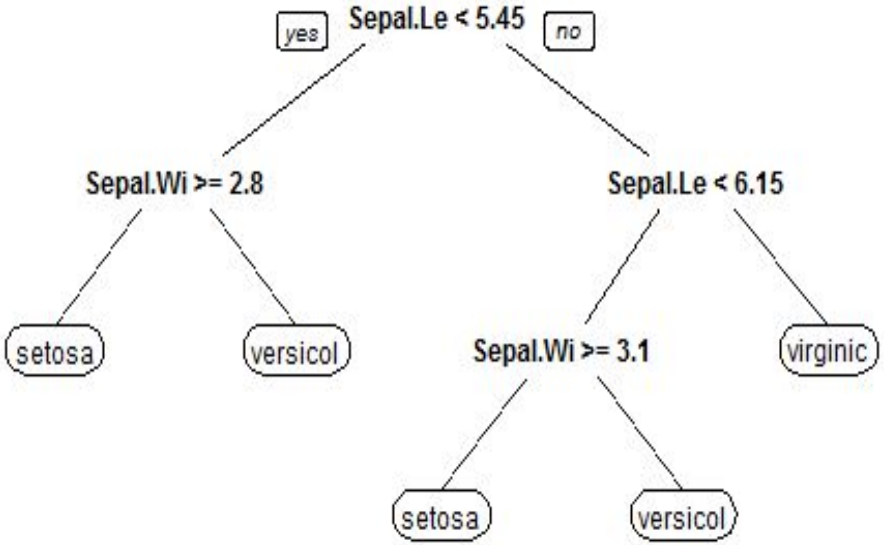
- Often, we can improve performance by tweaking parameters
- `rpart.control` provides these parameters for decision trees
  - `minsplit` is the minimum number of observations that must exist to split
  - `minbucket` is the minimum number of observations that must exist in each leaf
  - `maxdepth` is the maximum depth of the decision tree
  - `xval` is the number of cross validations to perform
- Usage:

```
fit <- rpart(y ~ x, data = frame,
             control = rpart.control(maxdepth = 10))
```

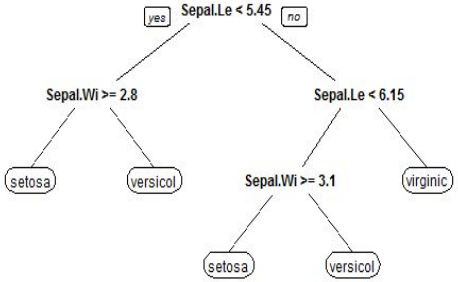
Missing data

- Sometimes certain features will be missing in the training data
- `rpart` automatically handles missing data using surrogate splits
- Surrogates are fake values that `rpart` substitutes for NAs
- They are used on missing test data, as well as missing training data
- It is sometimes difficult to find appropriate surrogates for missing data

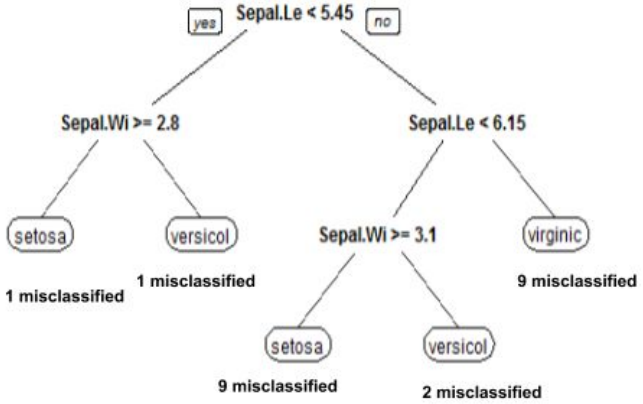
A decision tree for `iris`



A decision tree for `iris`



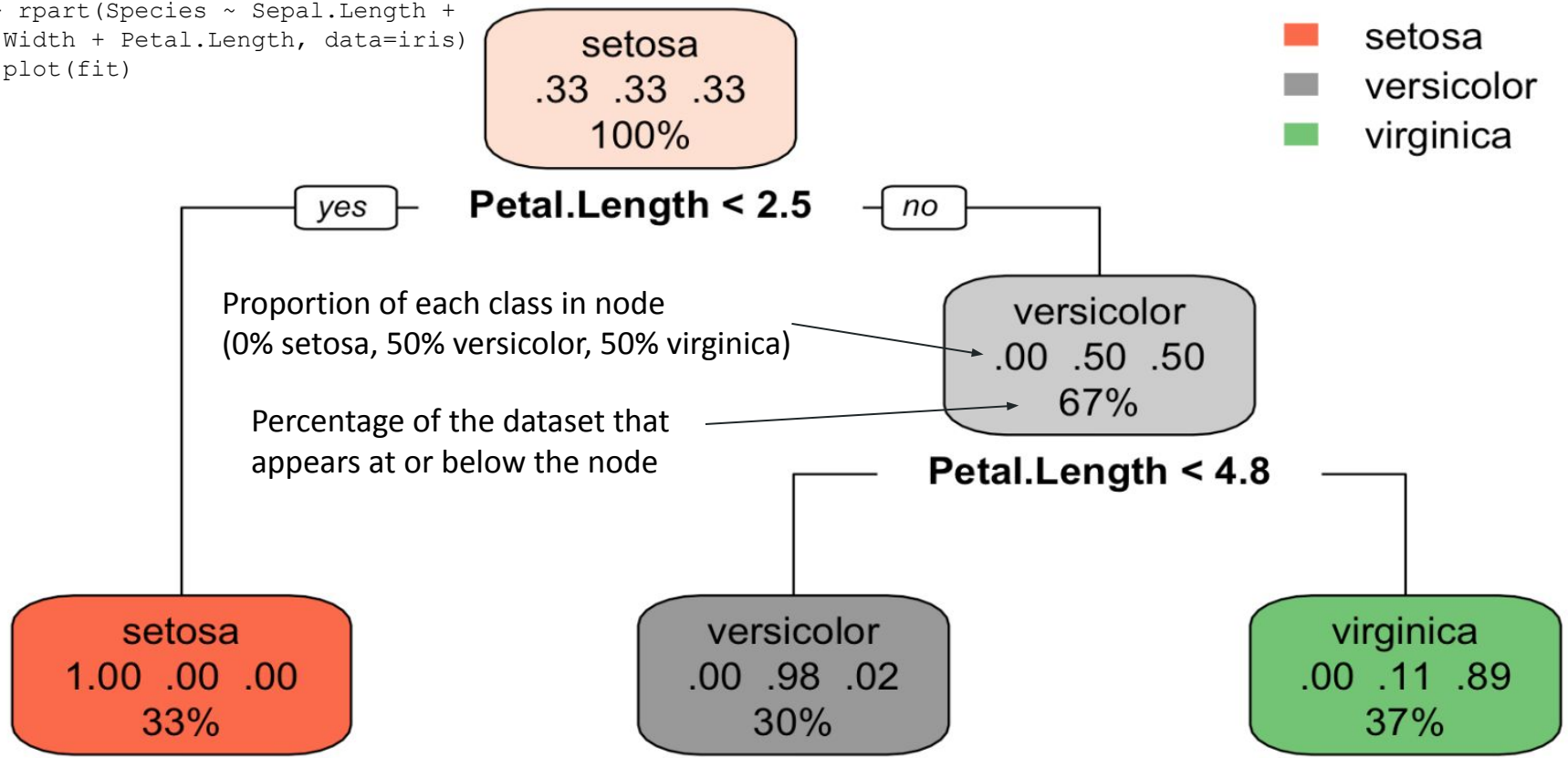
Misclassification error for the decision tree for `iris`



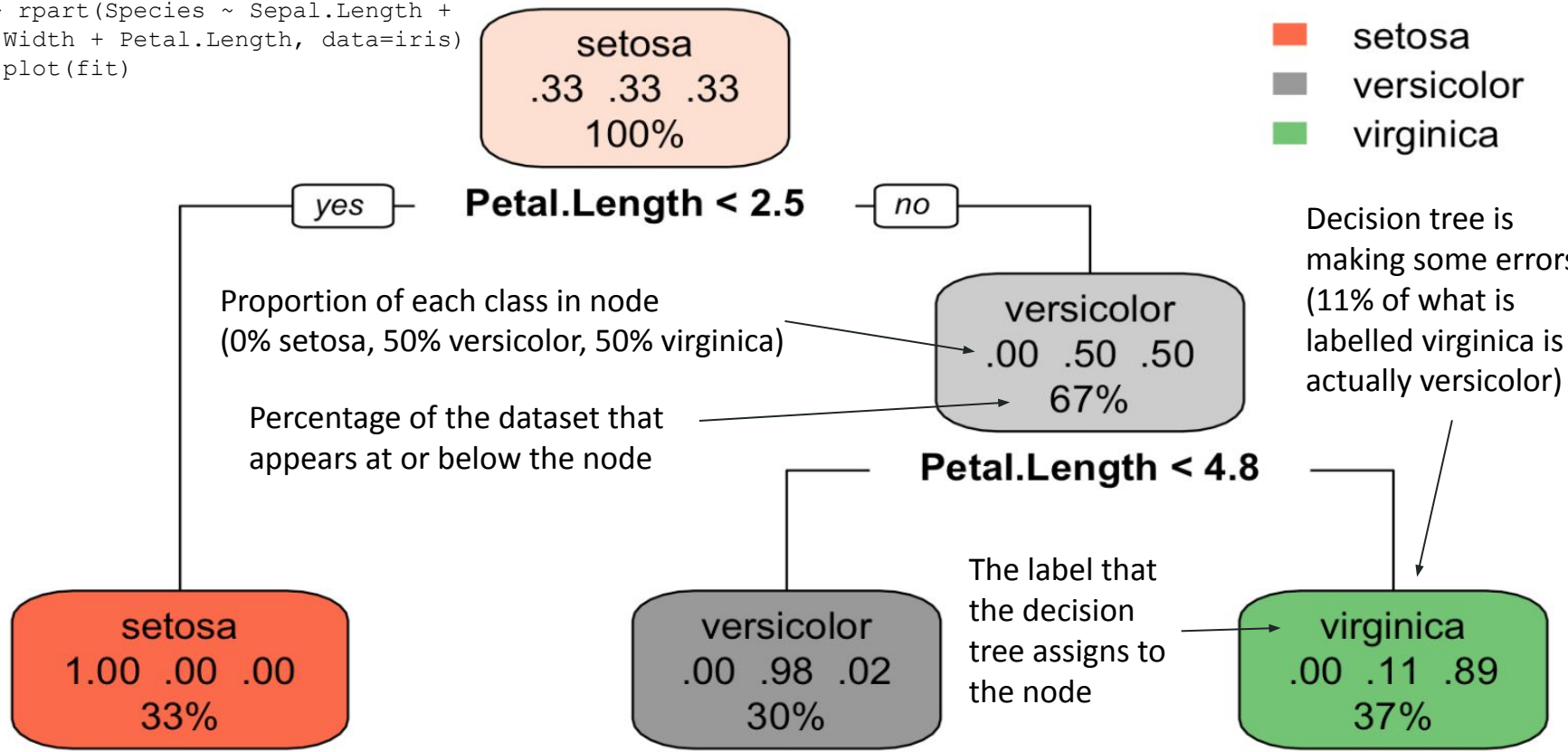
= 22 irises that have been misclassified

22/150 irises misclassified = 14.67% misclassification rate

```
library(rpart)
library(rpart.plot)
fit <- rpart(Species ~ Sepal.Length +
Sepal.Width + Petal.Length, data=iris)
rpart.plot(fit)
```



```
library(rpart)
library(rpart.plot)
fit <- rpart(Species ~ Sepal.Length +
Sepal.Width + Petal.Length, data=iris)
rpart.plot(fit)
```



Extras

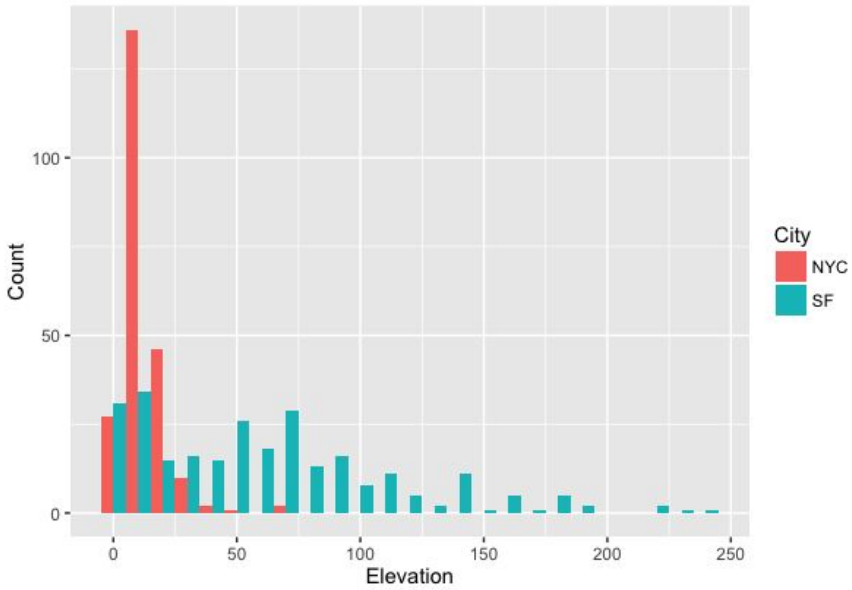
Adapted from Visual Intro to Machine Learning

San Francisco and New York

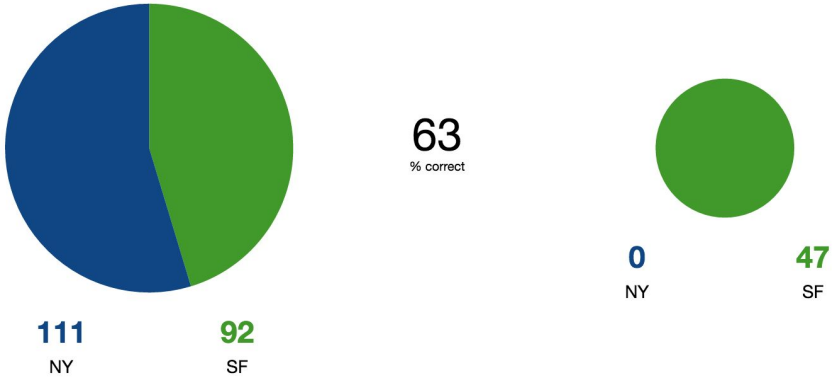
- Data on NYC and SF apartments
- Green = SF & Blue = NYC
- We can look at a scatterplot matrix, a set of scatterplots comparing different features
- We can already see some patterns in the data
- Let’s look at elevation on its own first

Elevation

- NYC is lower than SF
- We could pick a convenient point, like the highest NYC house at 73m, and classify using that
- Houses above 73m are in SF, below 73m are in NYC



Elevation

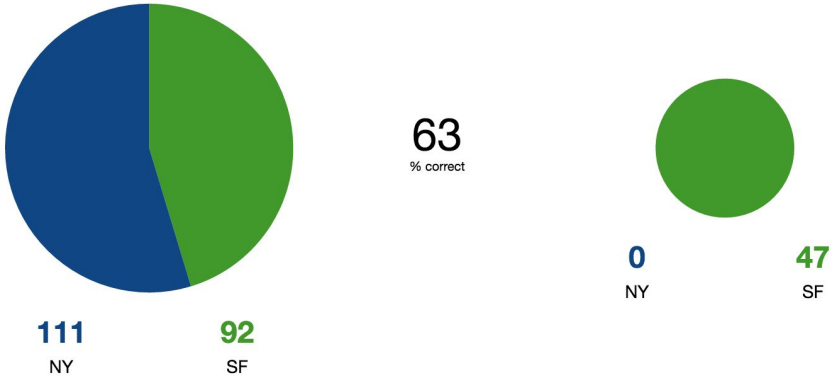


- NYC < 73m, SF > 73m

```
preds <- ifelse(apartments$elevation <= breakpoint, "NYC", "SF")
```

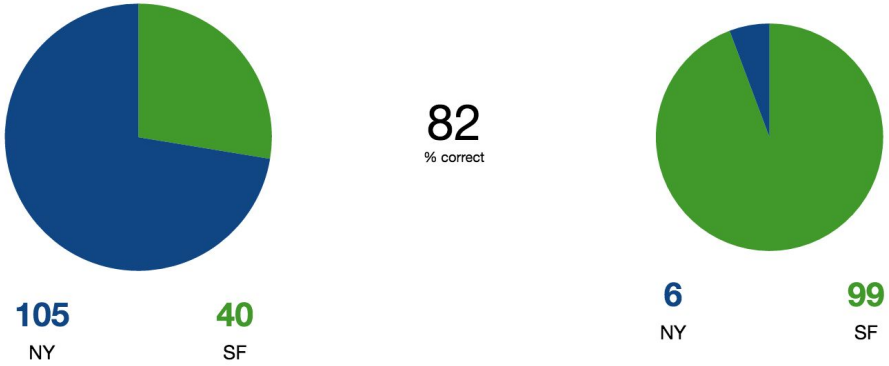
- Accuracy on training data is only 63%
- Barely better than guessing

Elevation



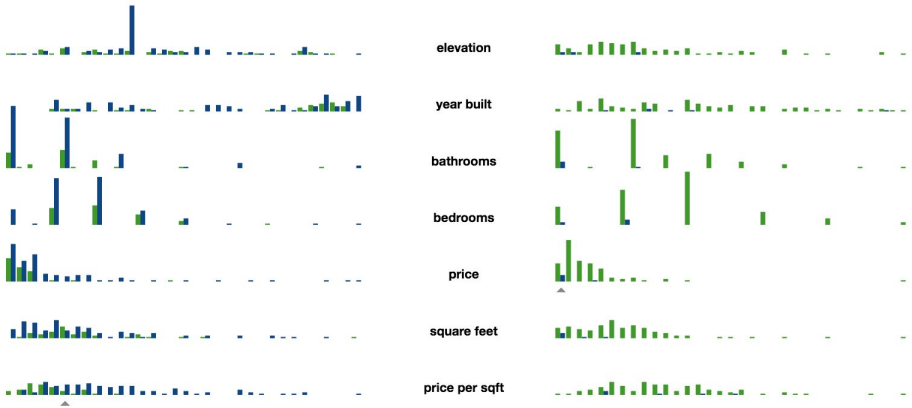
- We classify all houses about 73m correctly, but misclassify a lot below that height, called “false negatives”
- If we split on a lower height, we then misclassify many NYC homes, but we could get better accuracy overall

Elevation



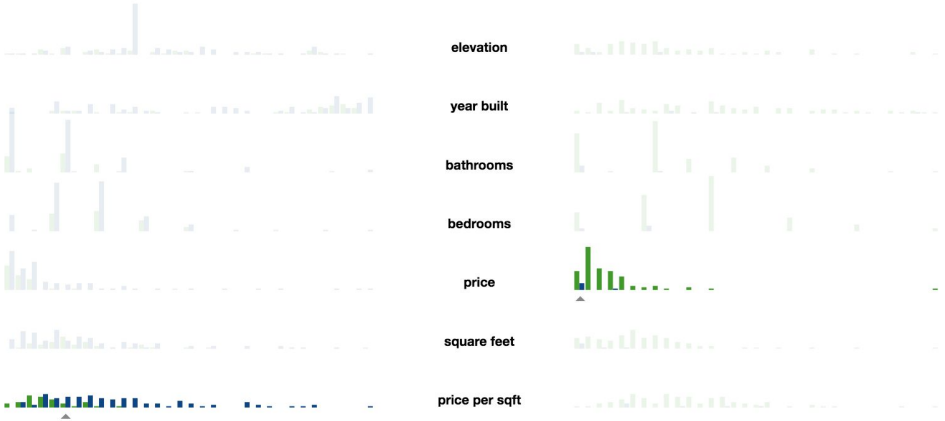
- We accept some false positives, incorrectly classified NYC homes, in order to get a better overall error rate
- We can still improve our accuracy

After the split



- Here are histograms for each side of the split, lower elevation on the left and higher elevation on the right
- We can see more patterns arising in these additional features
- What if we kept splitting?

Split all the things!



- We’ve partitioned our data once, why not split on different features?
- For low elevation houses, splitting on price per square foot gives the best results; same as price for high elevation houses

Just keep splitting

- For each split, we keep splitting and eventually make a decision tree