14 | Trees

Ivan Corneillet

Data Scientist

Learning Objectives

After this lesson, you should be able to:

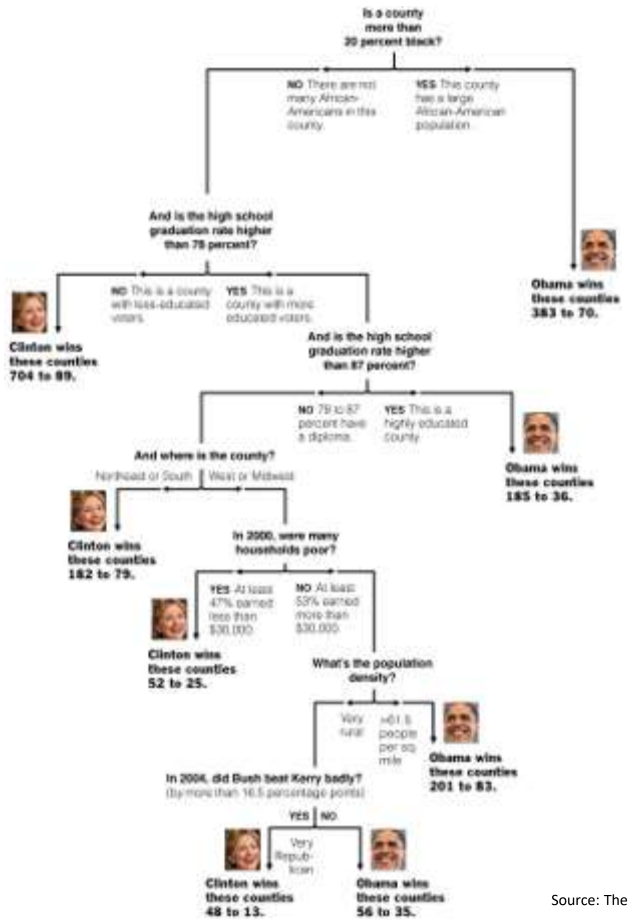
- Understand and build decision tree models for classification and regression
- Understand and build random forest models for classification and regression
- Know how to extract the most important predictors in a random forest model

Decision Trees

The 2008 Democratic Primaries

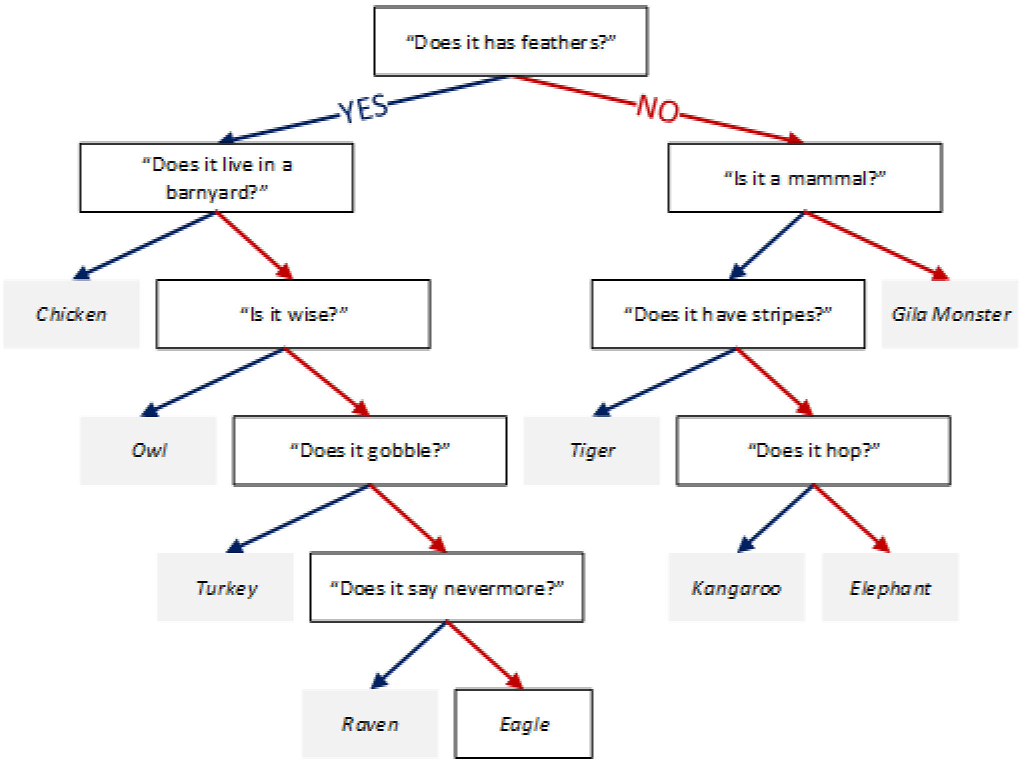
EXAMPLE

- Decision Tree: The Obama-Clinton Divide
  - Published in The New York Time on April 16, 2008 while the Democratic Primaries were still running



Source: The New York Times

Decision trees are like the game "20 questions." They make decision by answering a series of questions, most often binary questions. (yes or no) (cont.)



- We want the smallest set of questions to get to the right answer
- Each question should reduce the search space as much as possible

Using decision trees to make predictions is great but how do we build them in the first place?

Open questions from the previous activity

- ❶ How to choose the split conditions? (variables and threshold values)
  - E.g., why is the threshold for African-American population set at 20%?
- ❷ How to choose the order of the conditions?
  - E.g., why is the first split on African-American population vs. the voters' education level?
- ❸ When to stop?
  - E.g., why didn’t we include other factors such as voters’ age?

Because it isn't computationally feasible to consider every possible partition of the feature space, we take a *top-down*, *greedy* approach known as recursive binary splitting

Top-Down

- The approach begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree

Greedy

- At each step of the tree-building process, the *best* split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step

Decision trees can be applied to both classification and regression problems

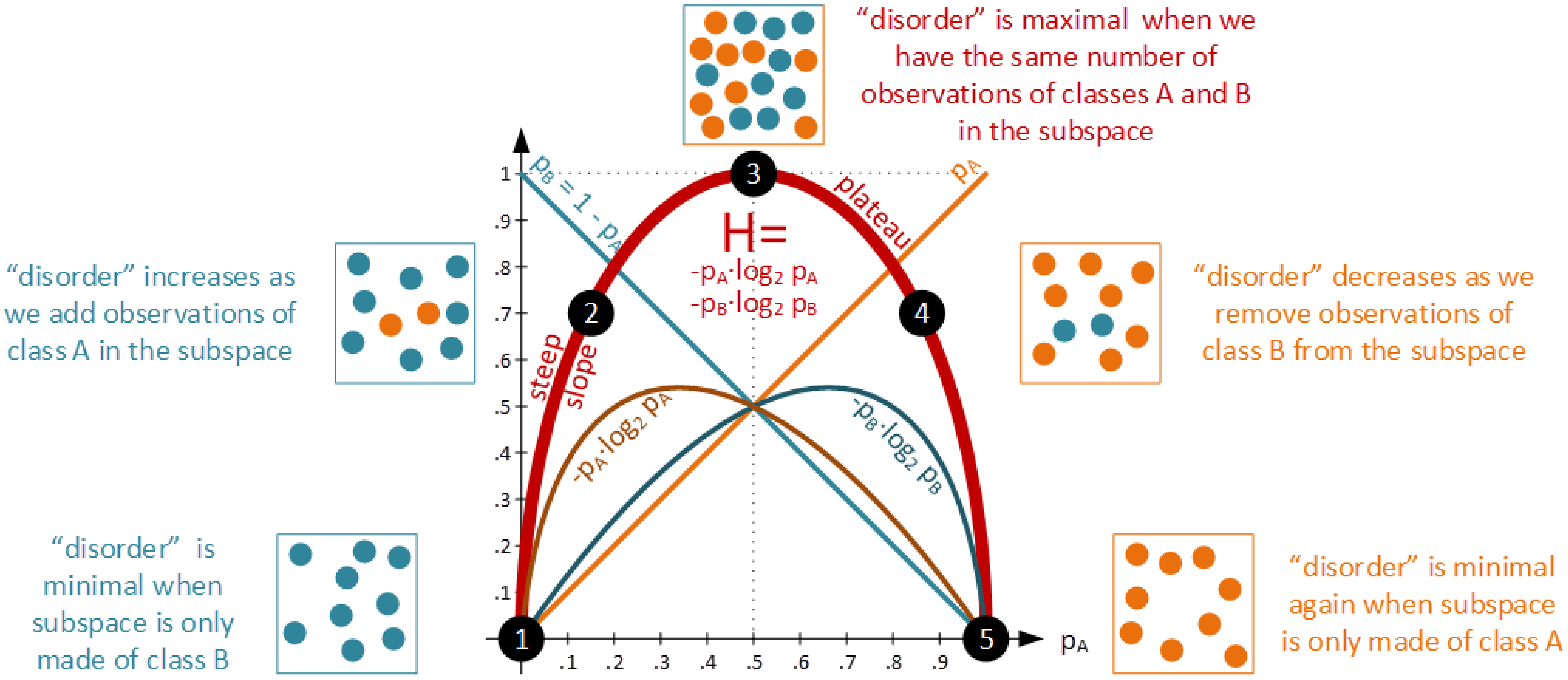
- We first consider classification problems to address ❷ (How to choose the order of the conditions?)

- We'll then move on to regression problems when addressing ❶ (How to choose the split conditions?)

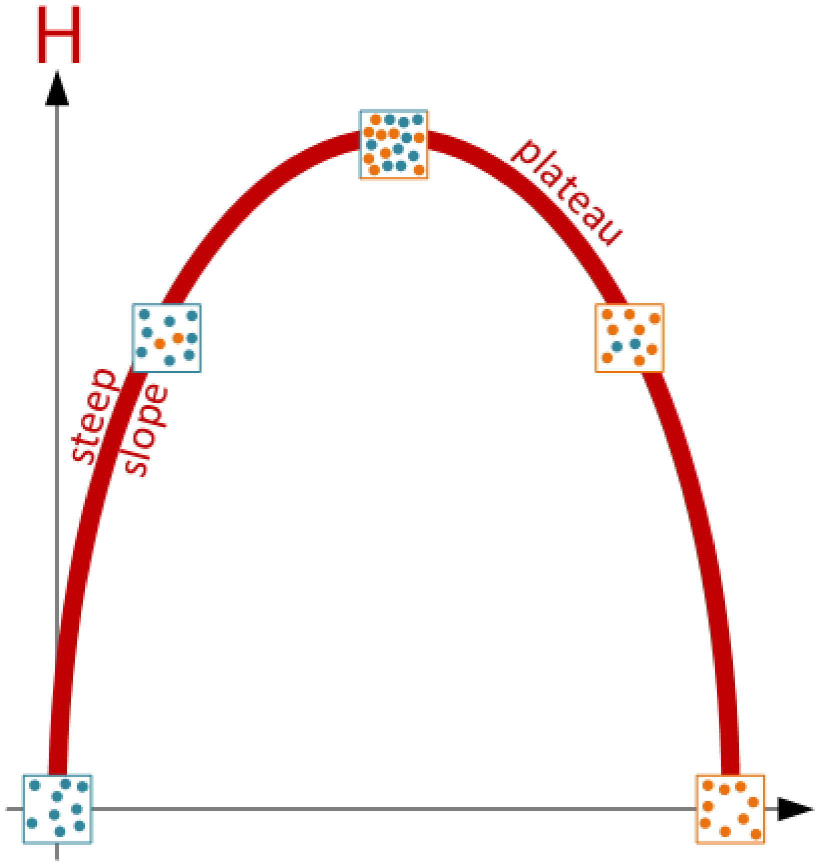
Decision Trees

Entropy

Entropy (H) is a measure of disorder



What to remember about entropy



$$H = -\sum_{i=1}^{k} p_i \cdot log_2(p_i)$$

(p_i represents the proportion of observations in the region that are from the i^{th} class)

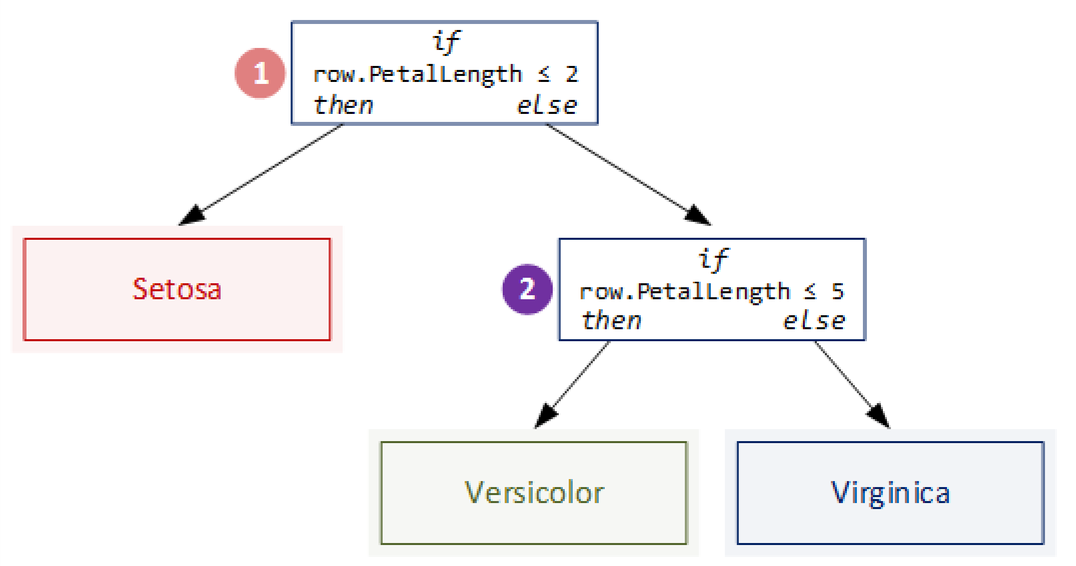
Decision Trees

Training a Classification Decision Tree

❷ *How to choose the order of the conditions?*

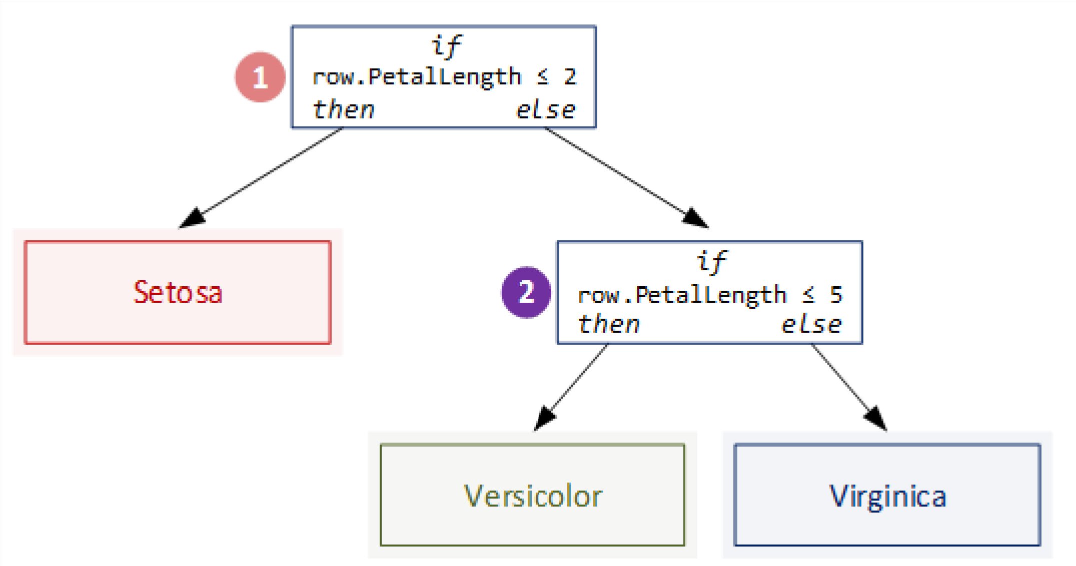
Our first classifiers (class 5) were in fact decision trees

```python
def my_second_classifier(row):
  if row.PetalLength <= 2:
    return 'Setosa'
  elif row.PetalLength <= 5:
    return 'Versicolor'
  else:
    return 'Virginica'
```

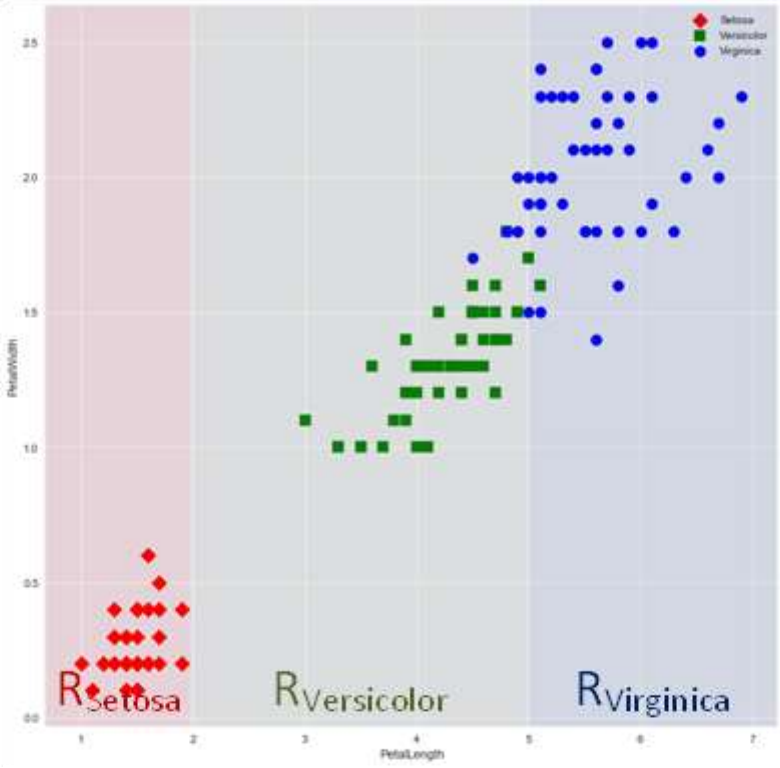


Our first classifiers (class 5) were in fact decision trees (cont.)

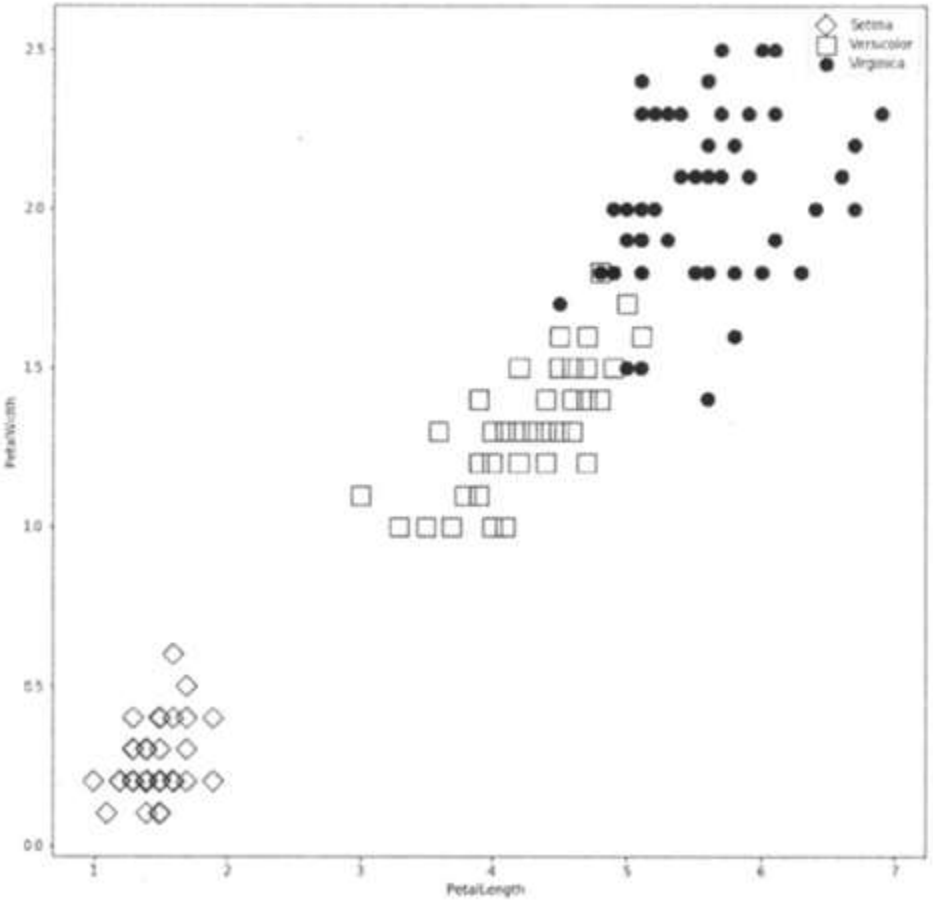
Decision Tree



Feature Space



⓿ What's the entropy of the dataset/root node before the first split? What's your intuition of its level?



root node

	red	green	blue	overall
c_i	50	50	50	150 (n)
p_i	$\frac{50}{150} = \frac{1}{3}$	$\frac{50}{150} = \frac{1}{3}$	$\frac{50}{150} = \frac{1}{3}$	1
H	$-\frac{1}{3} \times \log_e \frac{1}{3} - \frac{1}{3} \log_e \frac{1}{3} - \frac{1}{3} \log_e \frac{1}{3} = \log_e 3 \simeq 1.58$			

$$n = \sum_{i=1}^{k} c_i$$

$$p_i = \frac{c_i}{n}$$

$$H = -\sum_{i=1}^{k} p_i \cdot log_2(p_i)$$

❶ What's the entropy after split 1? Intuition?

left node/lower subspace after split 1

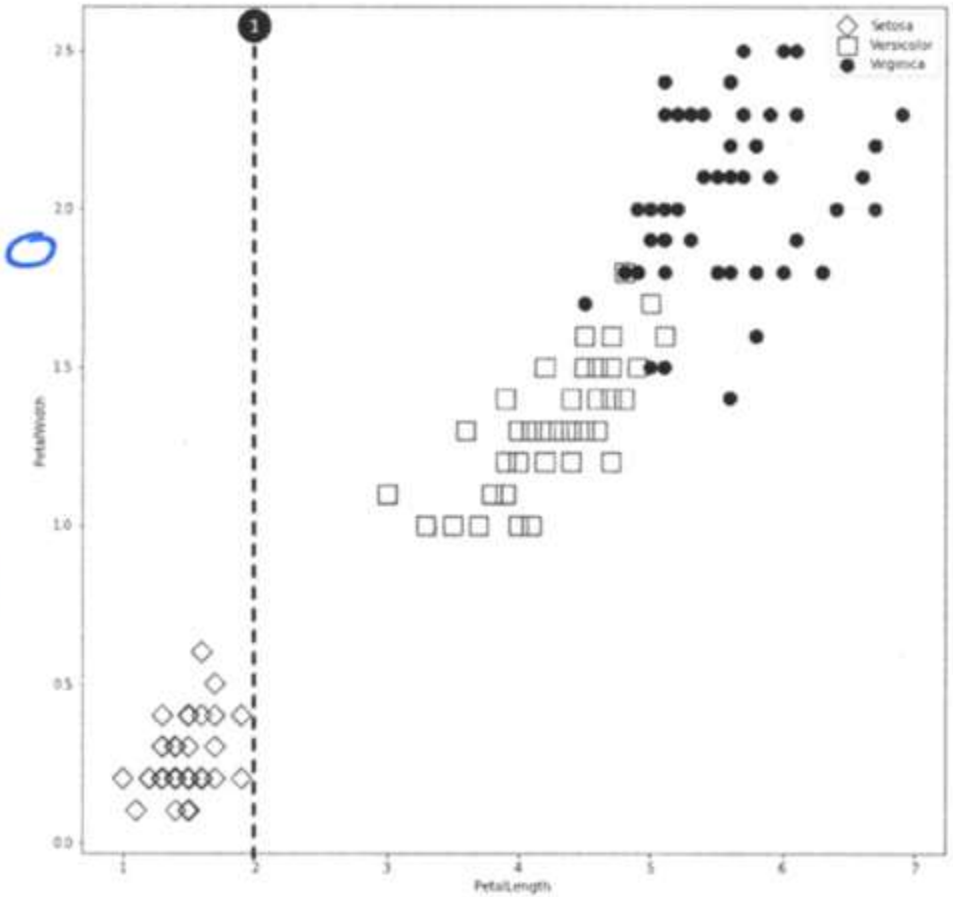
	red	green	blue	overall
c_i	50	0	0	50
p_i	$\frac{50}{50}=1$	$\frac{0}{50}=0$	$\frac{0}{50}=0$	1
H_{left}	$-1\times\log_2 1-0\times\log_2 0-0\times\log_2 0=0$			

right node/higher subspace after split 1

	red	green	blue	overall
c_i	0	50	50	100
p_i	$\frac{0}{100}=0$	$\frac{50}{100}=\frac{1}{2}$	$\frac{50}{100}=\frac{1}{2}$	1
H_{right}	$-0\times\log_2 0-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}=\log_2 2=1$			

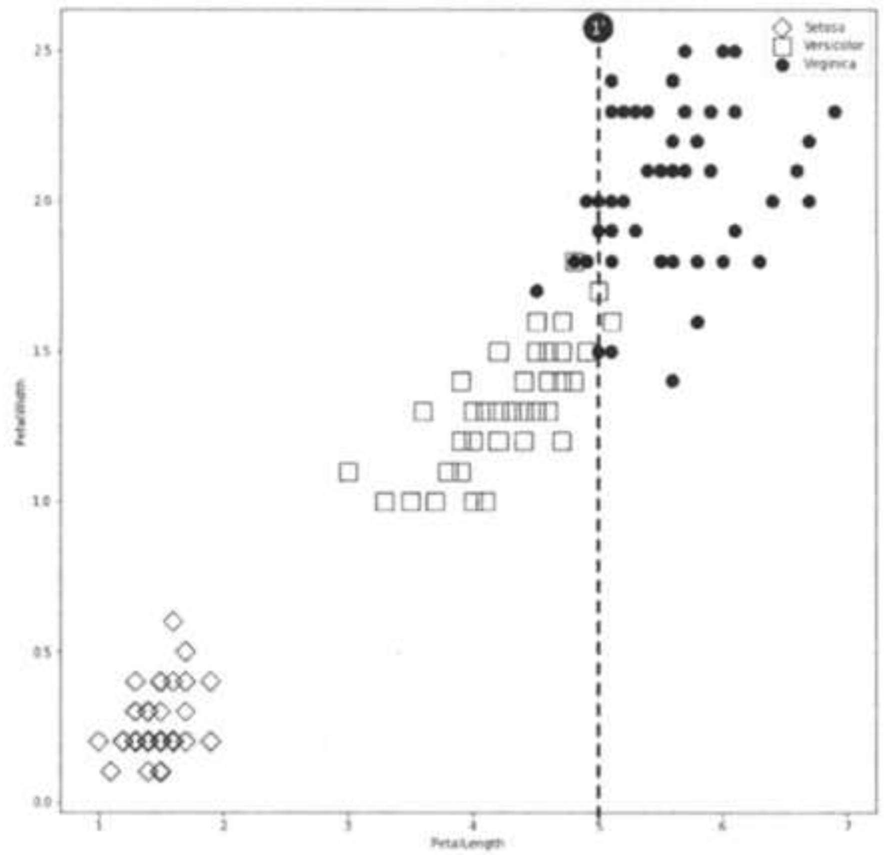
overall after split 1

H_{after}	$\frac{50}{150}\times 0+\frac{100}{150}\times 1=\frac{2}{3}\simeq .667$
IG	$1.58-.667=.918$



❷ What's the entropy after split 1'? Intuition? (Note: 9 "blues" on the left, 1 "green" on the right)

ACTIVITY



left node after split 1'

	red	green	blue	overall
c_i	50	49	9	108
p_i	$\frac{50}{108} = .463$	$\frac{49}{108} = .454$	$\frac{9}{108} = .0833$	1
H_{left}	$-.463 \times \log_e .463 - .454 \times \log_e .454 - .0833 \times \log_e .0833 = 1.33$			

right node after split 1'

	red	green	blue	overall
c_i	0	1	41	42
p_i	$\frac{0}{42} = 0$	$\frac{1}{42} = .0238$	$\frac{41}{42} = .976$	1
H_{right}	$-0 \log_e 0 - .0238 \times \log_e .0238 - .976 \times \log_e .976 = .162$			

overall after split 1'

H_{after}	$\frac{108}{150} \times 1.33 + \frac{42}{108} \times .162 = 1.00$
IG	$1.58 - 1.00 = .582$

Most common occurring class

- In practice, we don't expect each terminal region to hold a single class

- Instead, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs

Regression decision trees

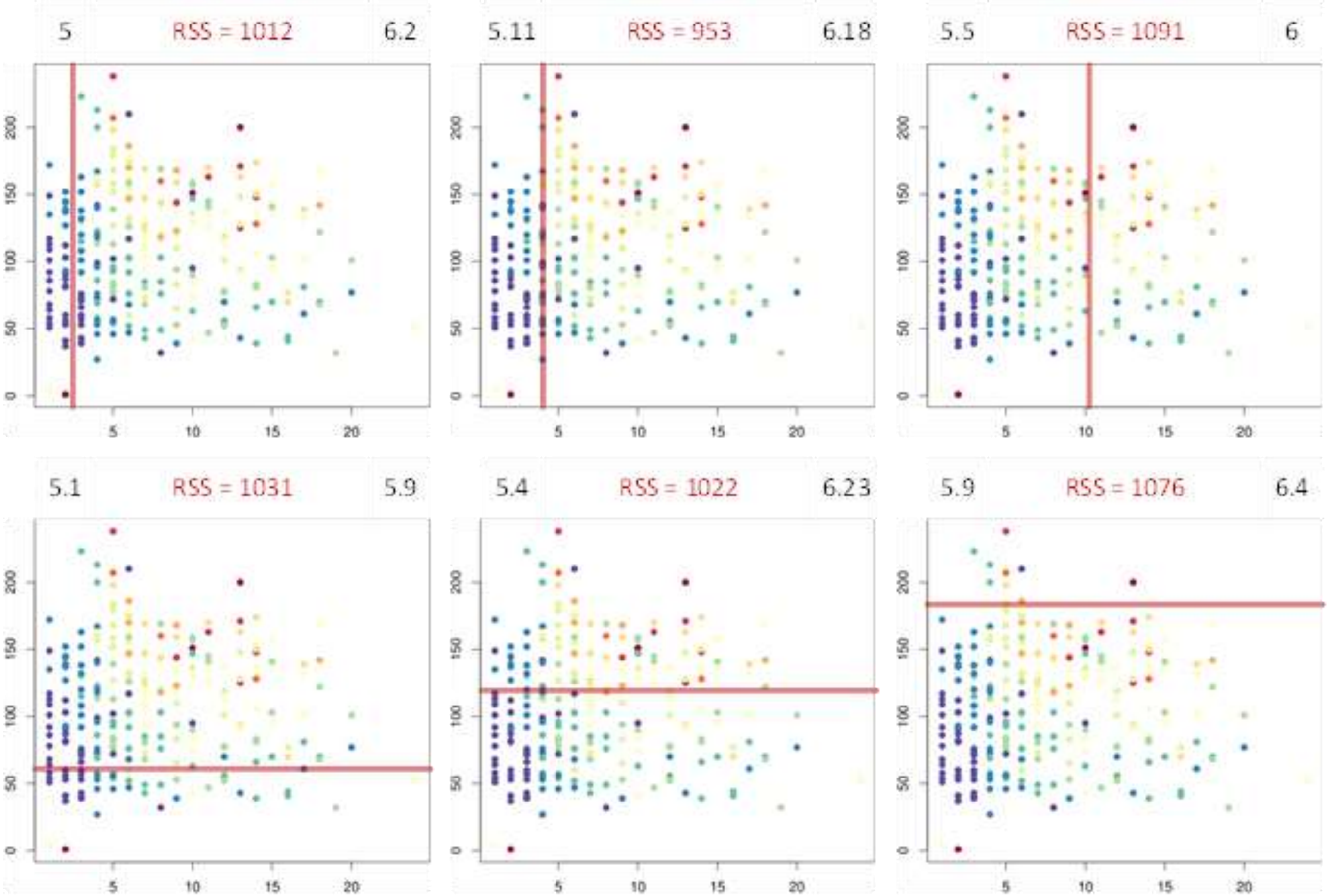
- Just as in the classification setting, we use recursive binary splitting to grow a regression tree

- For every observation that falls into the region R_j, we make the prediction $\hat{y}_{R_j}$, which is the mean of the response values for the training observations in R_j

- In the regression setting, we cannot use entropy for making the binary splits. A natural alternative to H is RSS (residual sum of squares)

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

We first select the feature and the cutpoint such that splitting the feature space into the regions $\{x \mid feature \leq cutpoint\}$ and $\{x \mid feature > cutpoint\}$ leads to the greatest possible reduction in RSS



Source: The Elements of Statistical Learning

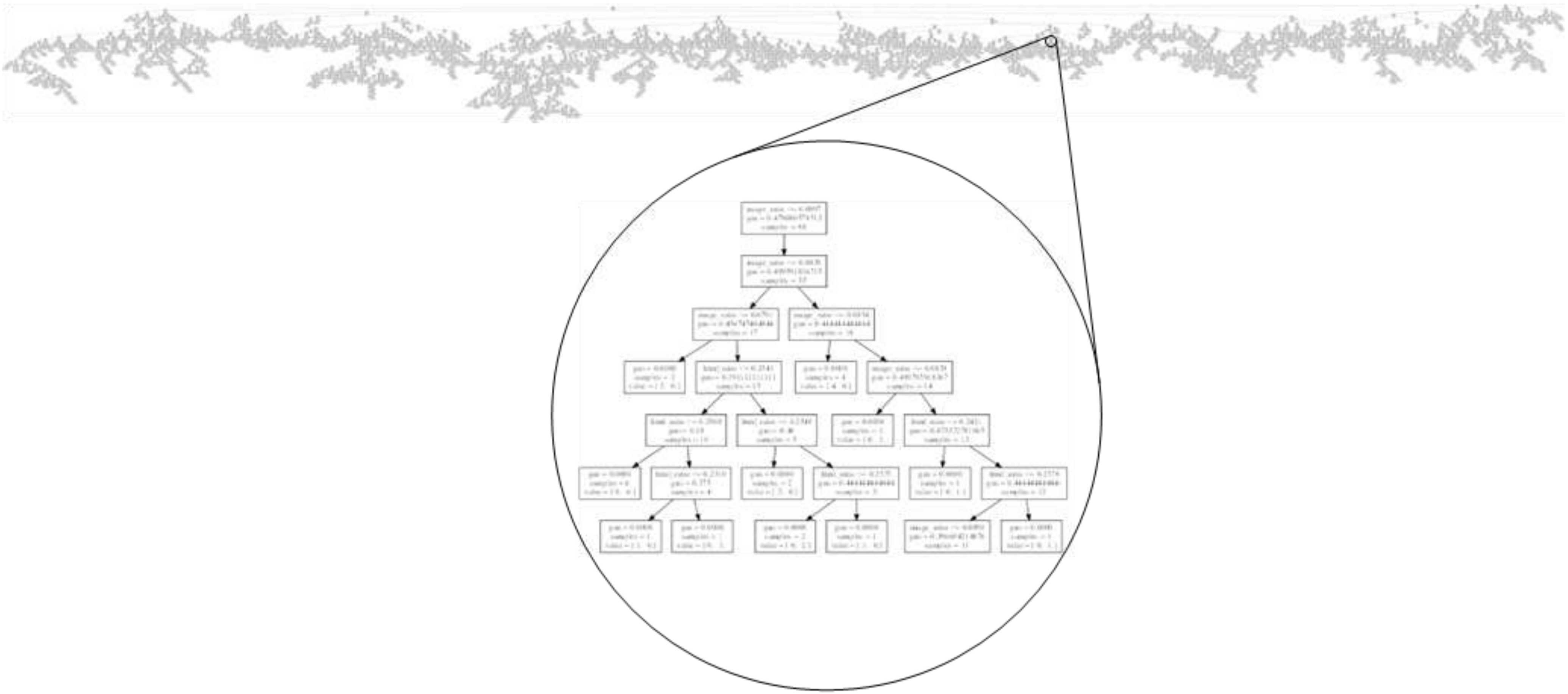
Top-down greedy approach (a.k.a., recursive binary splitting)

- Once the first cut is made, we recursively repeat the process in the two previously identified regions

- For *regression* trees, we would be looking for the best predictor and the *best cutpoint* in order to split the data further so as to minimize the *RSS* within each of the resulting regions

- For *classification* trees, we would be looking for the best predictor and the *highest* information gain in order to split the data further so as to minimize the *entropy* within each of the resulting regions

An fully-grown (i.e., unconstrained) decision tree can memorize a dataset (e.g., below)



Overfitting

- Decision trees tend to be weak models because they can easily memorize or overfit to a dataset
  - A model overfit when it memorizes or bends to a few specific data points rather than picking up general trends in the data

- We can limit our decision trees using a few methods
  - Limit the number of questions (nodes) a tree can have
  - Limit the number of samples in the leaf nodes

Decision Trees | Pros and cons

- Pros
  - Very easy to explain to people; even easier to explain than linear regression
  - Mirror more closely human decision-making than do the regression and classification methods seen so far
  - Can be displayed graphically and are easily interpreted even by non-experts
  - Can easily handle qualitative predictors without the need to create binary variables

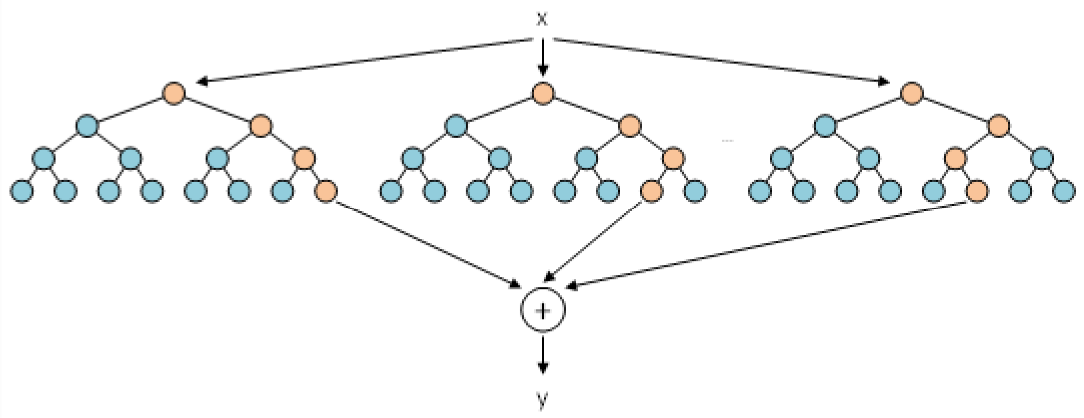
- Cons
  - Do not generally have the same level of predictive accuracy as some of the other regression and classification methods seen so far (higher variance). However, by aggregating many decision trees, the predictive performance of trees can be substantially improved

How can we avoid overfitting and increase predictability?

Random Forests

Random forests are an *ensemble* or collection of individual decision trees

- Random forest models are one of the most widespread classifiers used
- They are relatively simple to use and help avoid overfitting



Random Forests | Pros and cons

- Pros
  - Easy to tune
  - Built-in protection against overfitting
  - Non-linear
  - Built-in interaction effects

- Cons
  - Slow
  - No “coefficients”
  - Black-box
  - Harder to explain

Random Forests | Training

- Training a random forest model involves training many decision tree models
- Since decision trees easily overfit, we use many decision trees together and randomize the way they are created

- Random Forest Training Algorithm
  - Take a bootstrap sample (random sample with replacement) of the dataset
  - Train a decision tree on the bootstrap sample
  - For each split/feature selection, only evaluate a *limited* number of features to find the best one
  - Repeat this for a number of trees

Random Forests | Predicting

- Predictions for a random forest model come from each decision tree
- Make an individual prediction with each decision tree
- Combine the individual predictions and take the majority vote