CS60050
Machine Learning

Decision Trees

Somak Aditya
Assistant Professor
Department of CSE
August 8, 2024

Sudeshna Sarkar
Professor
CoE AI, Department of CSE

Decision Trees

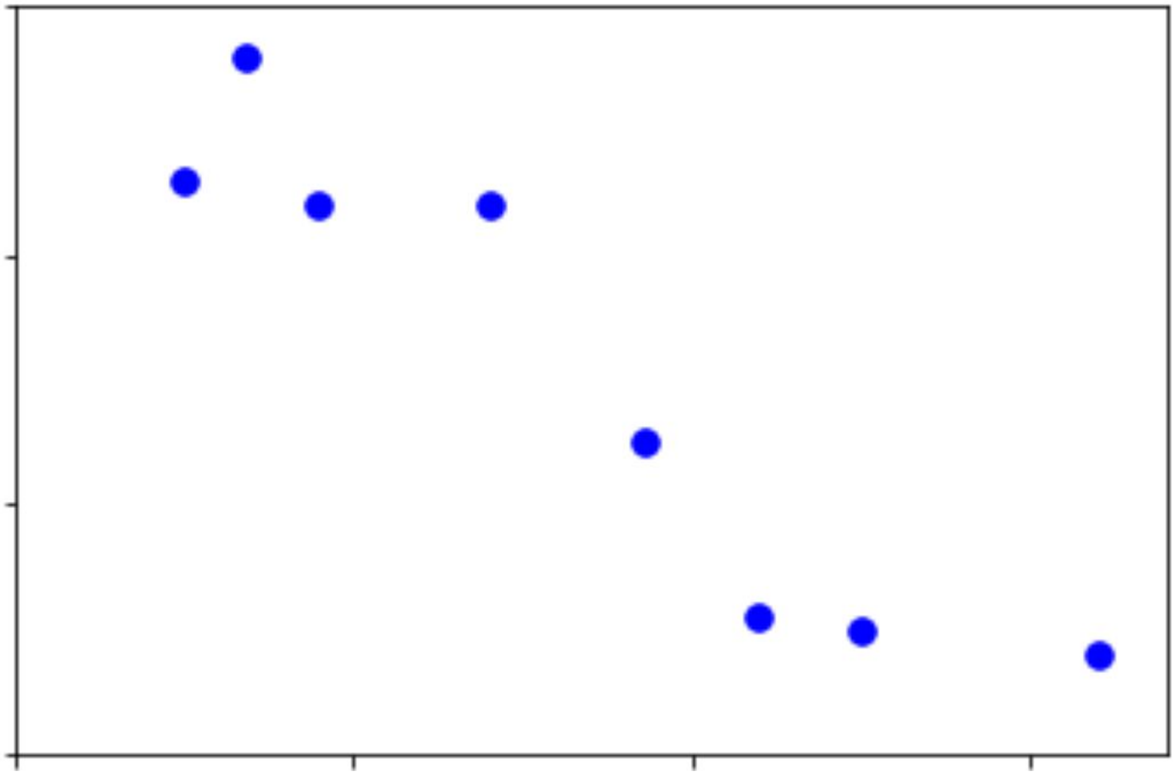
- Why are we talking about decision trees?

- Minimal prereqs: Doesn't rely on a ton of linear algebra, probability, or calc.
  - So we can focus on some important ML concepts and notation, including model selection, overfitting/underfitting
- Explainability
  - Decision trees can be incredibly useful as they can more easily be interpreted and altered by humans than other ML algorithms
- Basis of very powerful set of techniques: Random Forests
  - Random forests train many simple decision trees (ML topic: ensemble learning)
  - While powerful, random forests unfortunately have poor explainablility

Regression Model

- Regression: learning a model to predict a numerical output (but not numbers that just represent categories, that would be classification)

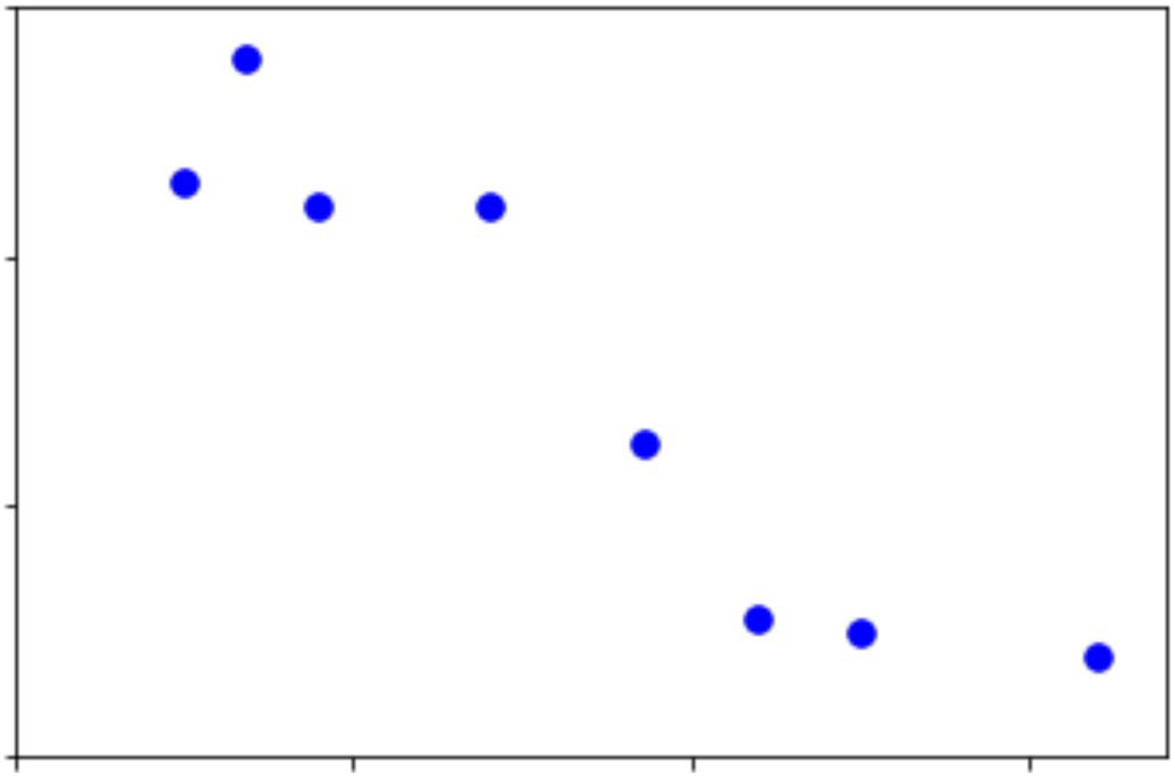
Model



Regression Model

- Regression: learning a model to predict a numerical output (but not numbers that just represent categories, that would be classification)

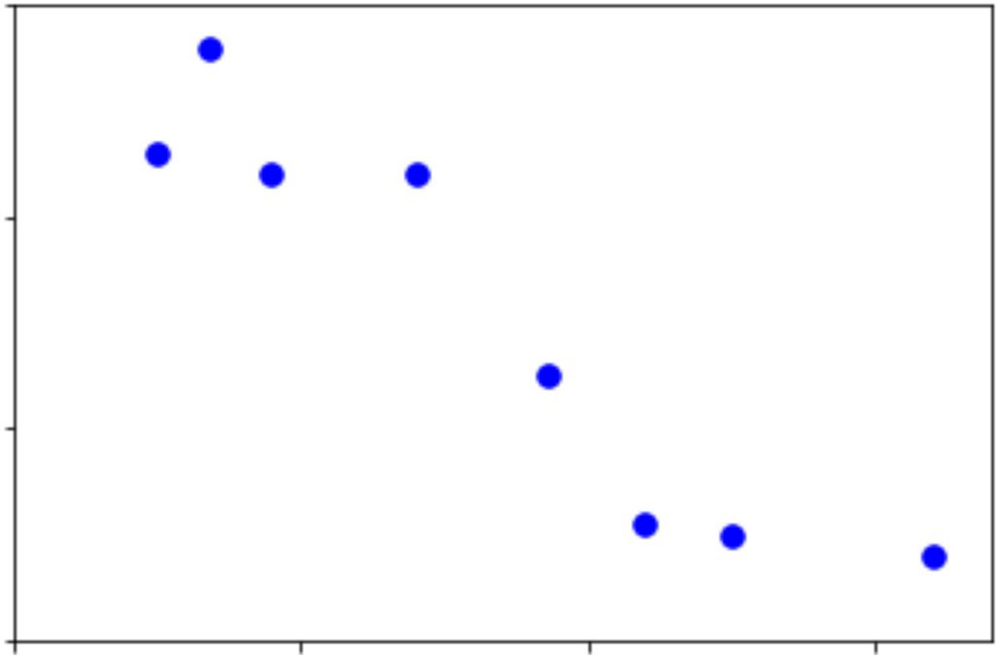
Model: Nearest neighbor



Poll 1

- Does the memorization algorithm learn?

A. Yes
B. No
C. I have no clue



Recap

- Learning problem (classification)

  Find a function that

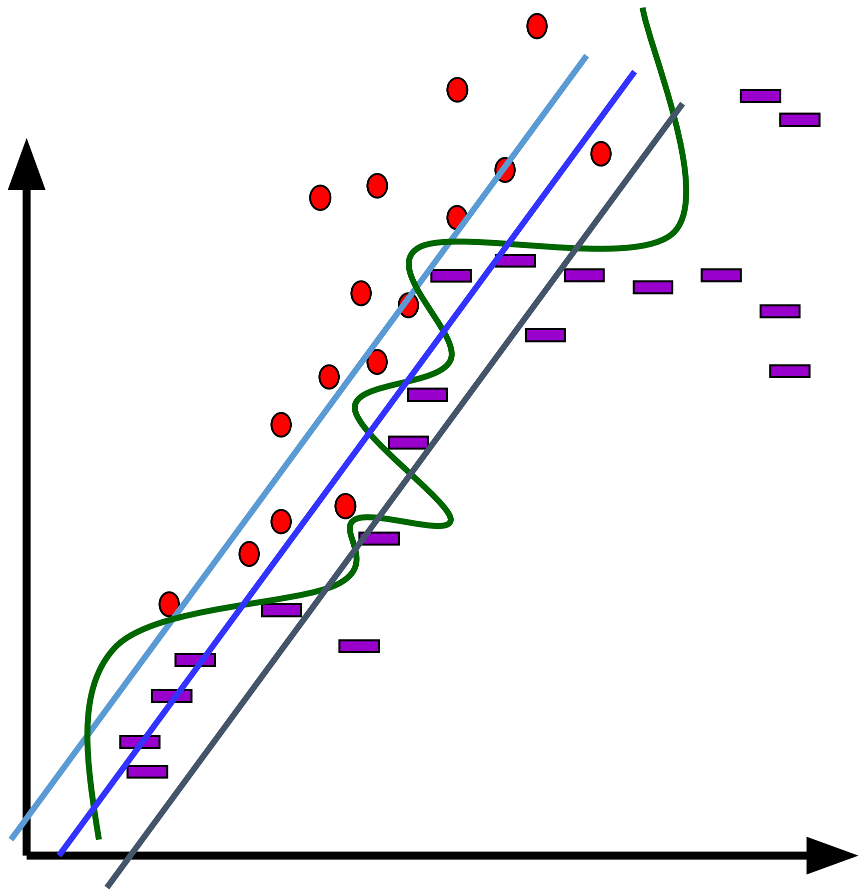
  best separates the data

- What function?

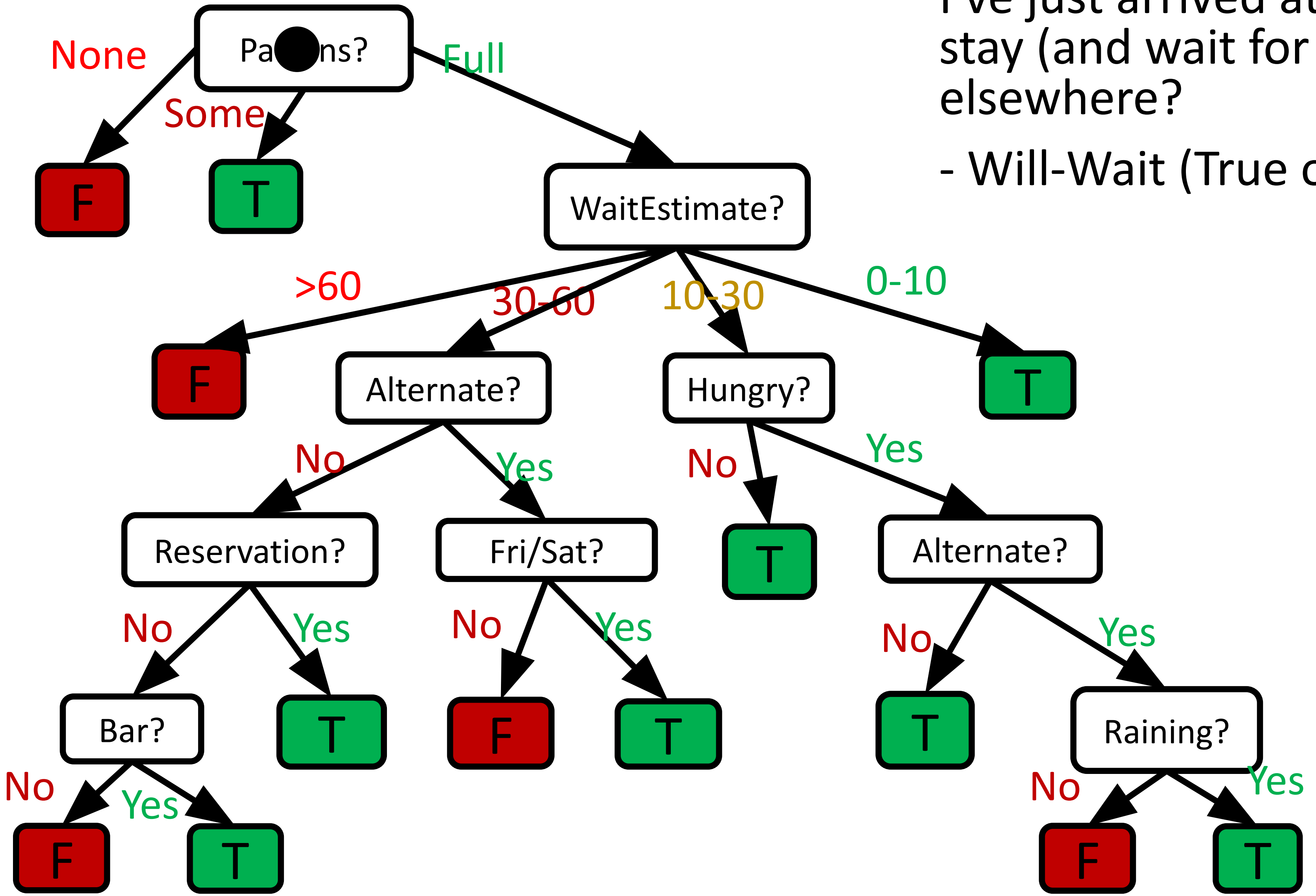
Linear:

$$Y = \mathbf{sign}\left(\boldsymbol{\theta}^T \boldsymbol{X}\right)$$

- **Limitations of Linear Functions**



Decision Trees

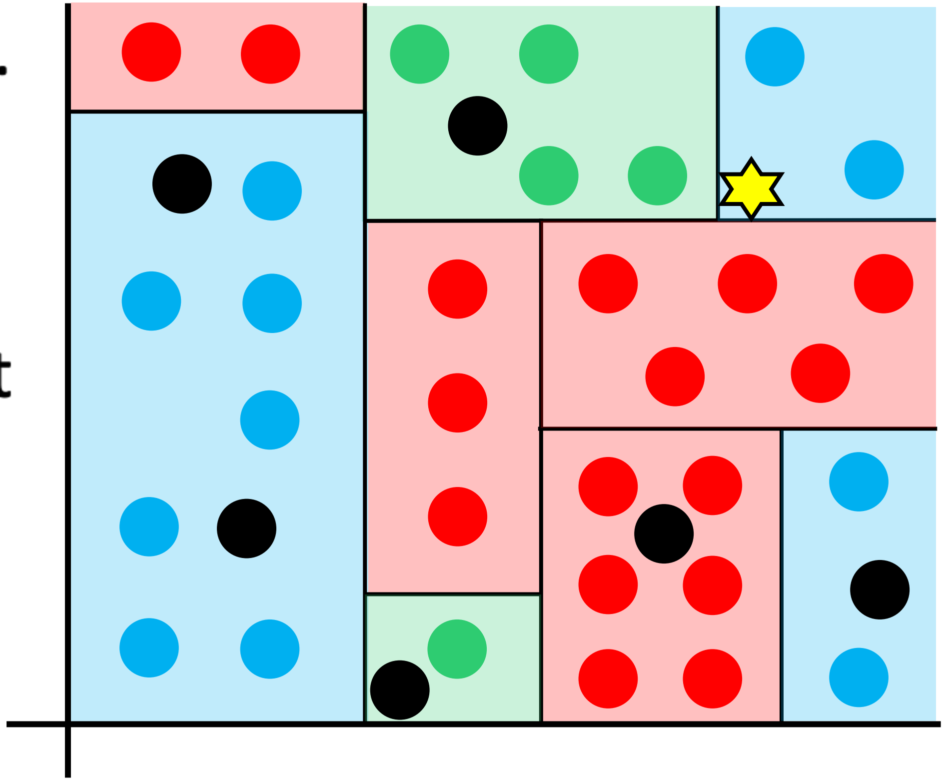


I've just arrived at a restaurant: should I stay (and wait for a table) or go elsewhere?

- Will-Wait (True or False?)

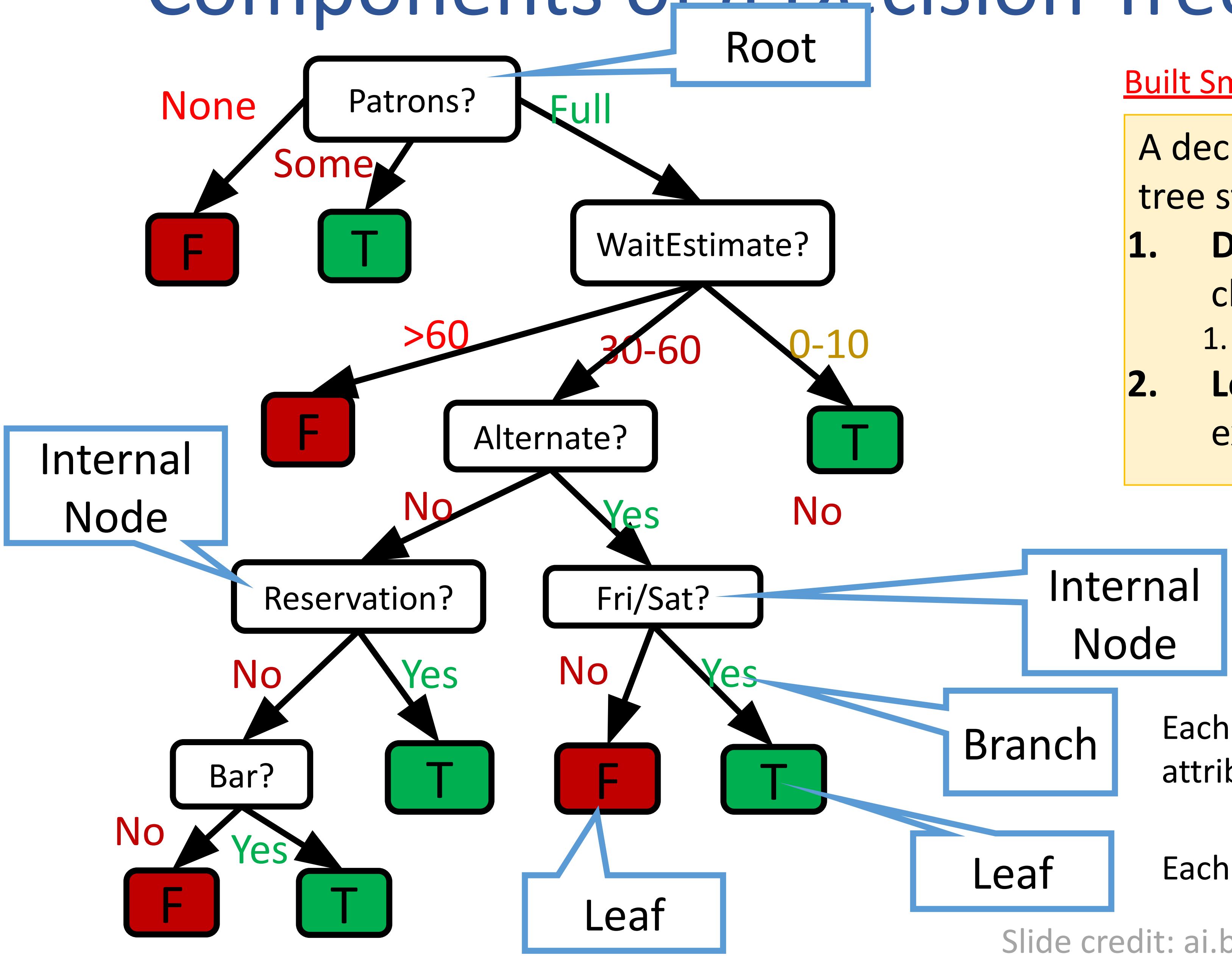
Decision Trees for Classification

- Find "many" lines that best "separates" the data.
- Repeatedly partition feature space $\mathbb{R}^d$
- Assign a label to each partition.
- For test data point, easy to find which partition it lies



This illustration's credit: Purushottam Kar

Components of a Decision Tree



<u>Built Smaller for Illustration</u>

A decision tree is a classifier in the form of a tree structure with two types of nodes:

1. **Decision node (Internal)**: Specifies a choice or test of some attribute
    1. An outcome □ A branch
2. **Leaf node**: Indicates classification of an example

Each internal node tests an attribute

Each branch corresponds to an attribute value node

Each leaf node assigns a classification

Slide credit: ai.berkeley.edu, Illustration: Purushottam Kar, IITK

Type of Internal Tests

Binary Feature

$x_i = 0?$

False | True

Categorical Feature

$x_i?$

$Value_1$ … $Value_n$

Numeric Feature

$x_i < 101.2?$

False | True

Types of Leaves

Classification

$y = POS$

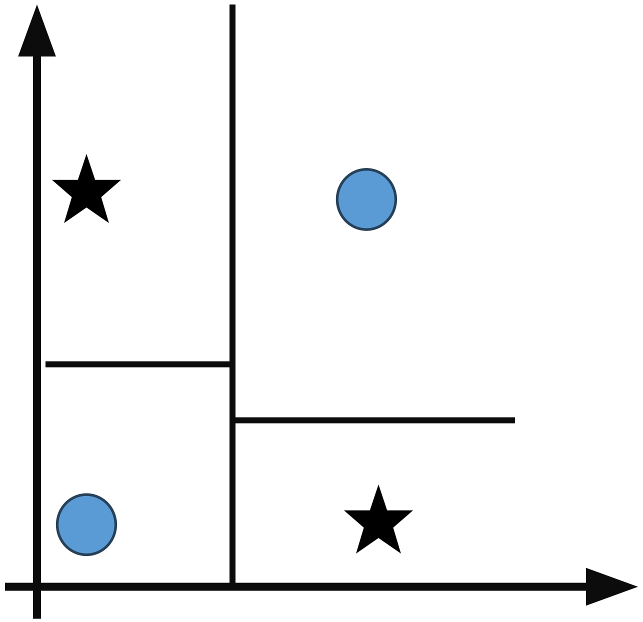
Regression

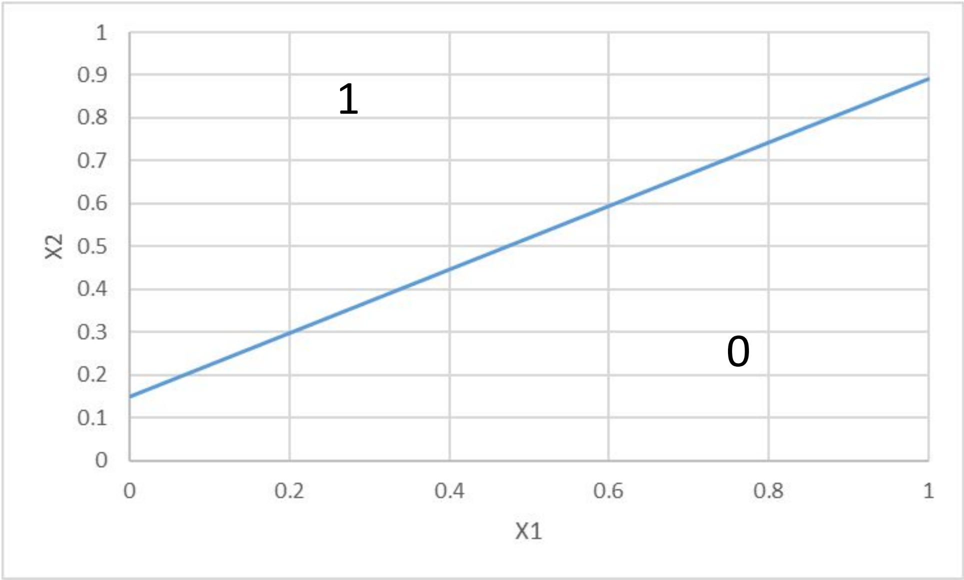
$y = 5.2$

Probability Estimate

$$\begin{aligned} \mathrm{p}(y = 0) &= .3 \\ p(y = 1) &= .3 \\ p(y = 2) &= .4 \end{aligned}$$

Decision Trees vs Linear Models



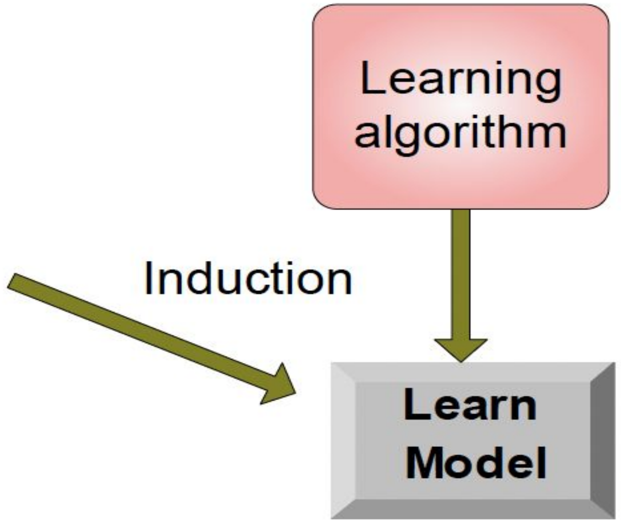


Adding nodes (structure) allows more powerful representation

Classification in a Nutshell

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

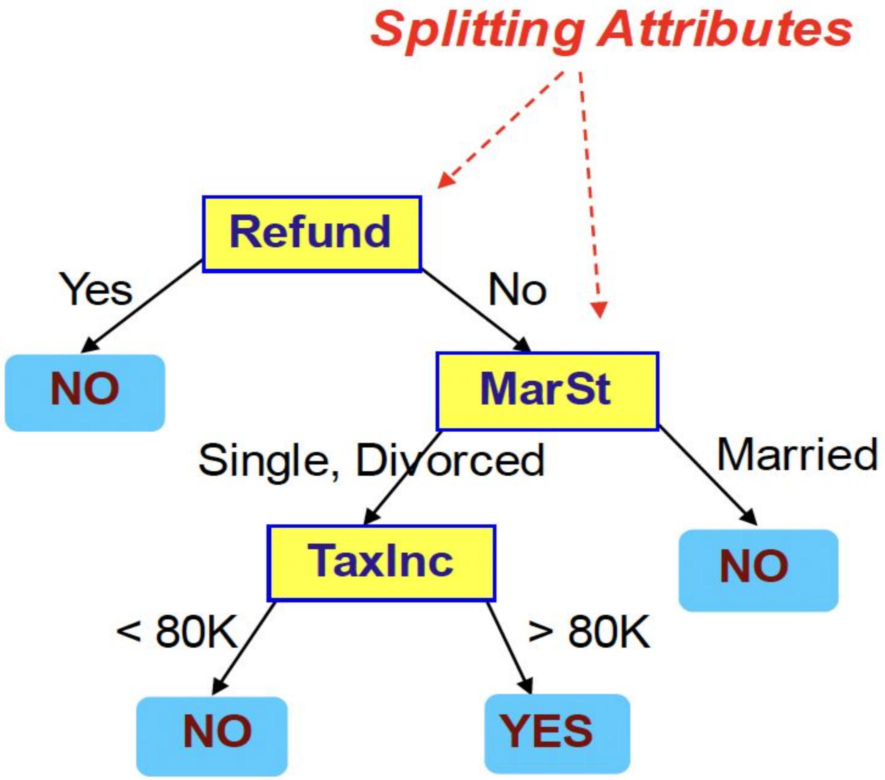
Test Set

Learning Decision Tree from Data

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

Issues

- Given some training examples, what decision tree should be generated?
- One proposal: learn the smallest tree that is fits the data
- Possible method:
    - Exhaustive search over the space of decision trees
    - This is NP-hard.

- *Efficient algorithms available to learn a reasonably accurate (potentially suboptimal) decision tree in reasonable time*
    - *Employs greedy strategy*
    - *Locally optimal choices about which attribute to use next to partition the data*

Building a Decision Tree (GREEDY)

Function BuildTree(*dataset*,*attributes*) returns a DT

  # *dataset* : dataset at current node, *attributes*: current set of attributes

  If *attributes* is empty OR

    all labels in *dataset* are the same:

    # Leaf node

    class = most common class in *dataset*

  else

    # Internal node

    att ⇐ CHOOSE-BEST-ATTRIBUTE(*dataset*, *attributes*)

    tree ← A new DT with root test *att*

    LeftNode = BuildTree(*dataset*(*att* =1), *attributes* −{*att*})

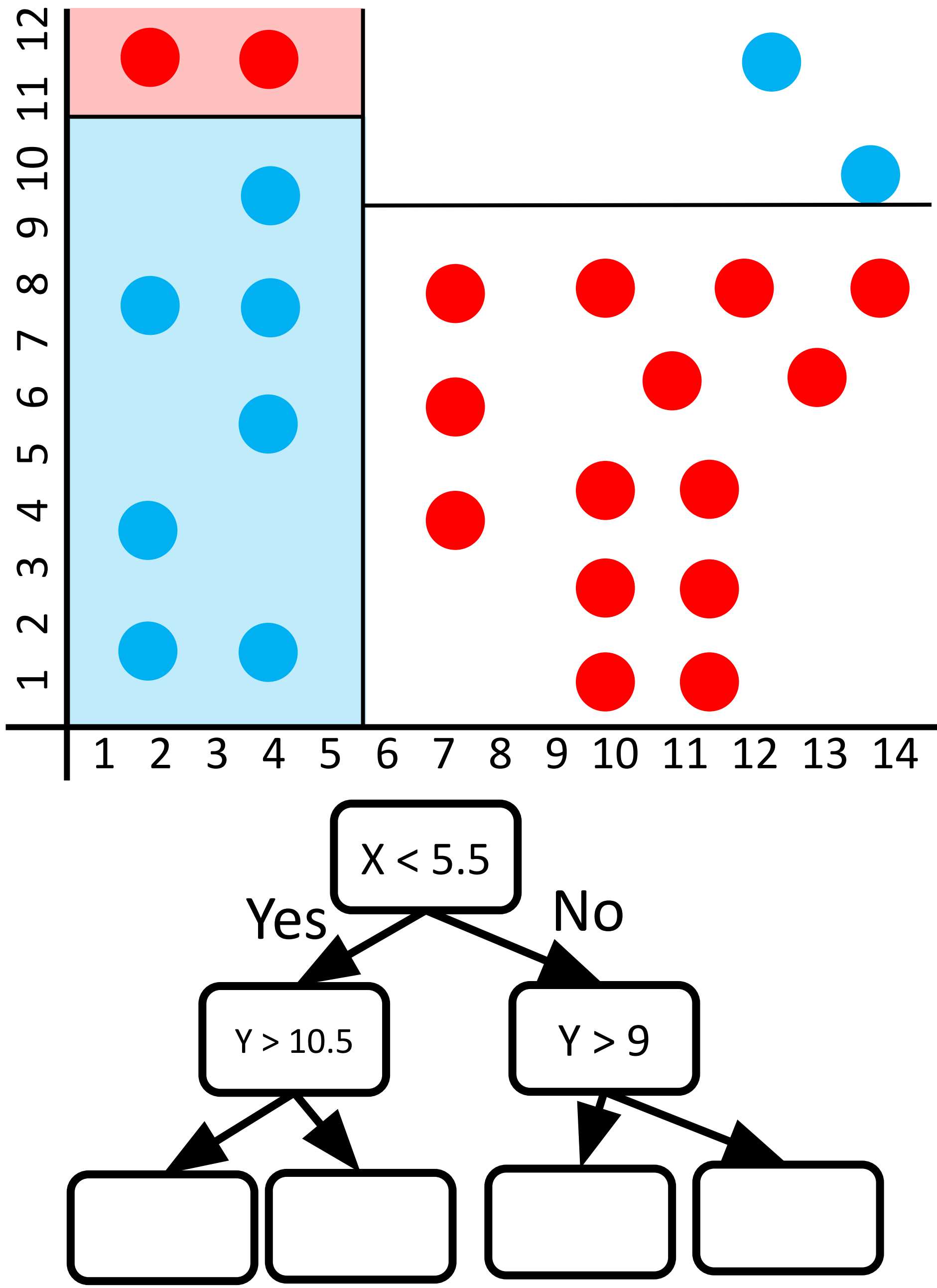
    RightNode = BuildTree(*dataset*(*att* =0), *attributes* −{*att*})

    add branch to tree with value 1, subtree LeftNode

    add branch to tree with value 0, subtree RightNode

    # generalize for multiple values

Binary Test



Decision Tree (Learn & Predict)

Function BuildTree(dataset,attributes) returns a DT

dataset : dataset at current node, *attributes*: current set of attributes

If *attributes* is empty OR

all labels in *dataset* are the same:

Leaf node

class = most common class in *dataset*

else

Internal node

att ⇐ CHOOSE-BEST-ATTRIBUTE(*dataset*, *attributes*)

tree ← A new DT with root test *att*

For each value v_j of attribute *att*:

ChildNode$_j$ = BuildTree(D(*att*= v_j), attributes−{att})

add branch to tree with value v_j, subtree ChildNode$_j$

return tree

Prediction

def f(x'):

Let *current node* = root

while(true):

- if *current node* is internal (non-leaf):
  - Let a = attribute associated with current node
  - Go down branch labeled with value x'_a
- if *current node* is a leaf:
  - return label y stored at that leaf

Decision Tree (Choices)

```
Function BuildTree(dataset,attributes) returns a DT
   # dataset : dataset at current node, attributes: current set of attributes
   If attributes is empty OR
         all labels in dataset are the same:
         # Leaf node
         class = most common class in dataset
   else
          # Internal node
          att ⇐ CHOOSE-BEST-ATTRIBUTE(dataset, attributes)
          tree ← A new DT with root test att
          For each value v_j of attribute att:
                 ChildNode_j = BuildTree(D(att= v_j), attributes—{att})
                 add branch to tree with value v_j, subtree ChildNode_j
         return tree
```

Choices

1. When to stop
   - no more input features
   - all examples are classified the same
   - too few examples to make an informative split
2. Which test to split on
   - split gives smallest error.

Decision Stumps

- Split data based on a single attribute
- Majority vote at leaves

Dataset:

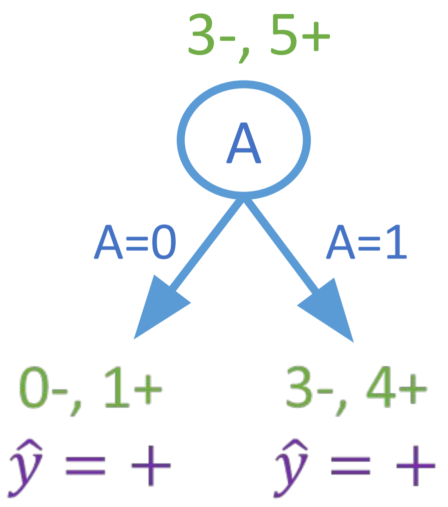
Output Y, Attributes A, B, C

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	o
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Slide credit: CMU MLD Matt Gormley

Poll 2

- Splitting on which attribute {A, B, C} creates a decision stump with the lowest training error?



Error: (0 + 3) / 8

= 3/8

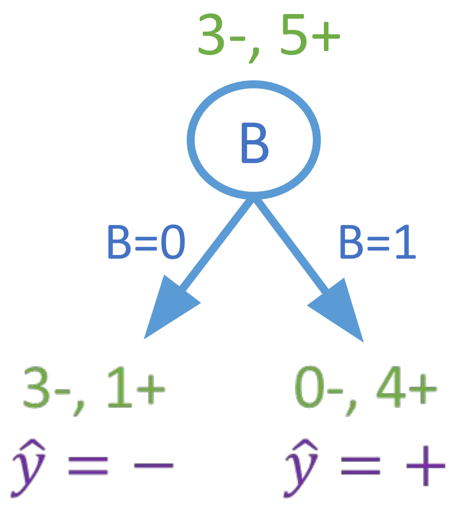
Dataset:

Output Y, Attributes A, B, C

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	o
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Poll 2

- Splitting on which attribute {A, B, C} creates a decision stump with the lowest training error?
- Answer: B



Error: (1 + 0) / 8

= 1/8

Dataset:

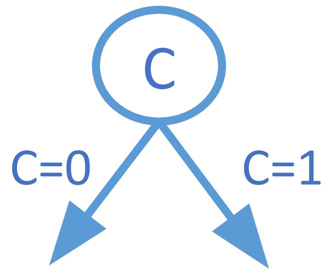
Output Y, Attributes A, B, C

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	o
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Poll 2

- Splitting on which attribute {A, B, C} creates a decision stump with the lowest training error?
- Answer: B

3-, 5+



2-, 2+ $\hat{y} = +/-$

1-, 3+ $\hat{y} = +$

Error: (2 + 1) / 8

= 3/8

Dataset:

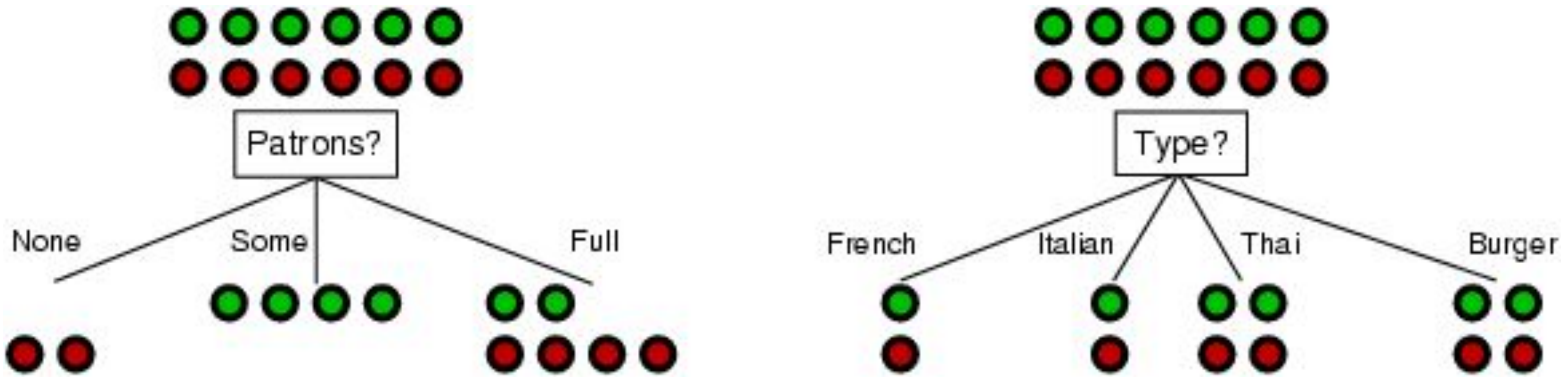
Output Y, Attributes A, B, C

Y	A	B	C
-	1	0	0
-	1	0	1
-	1	0	o
+	0	0	1
+	1	1	0
+	1	1	1
+	1	1	0
+	1	1	1

Slide credit: CMU MLD Matt Gormley

Choosing an attribute

Idea: good attribute splits examples into subsets that are (ideally) *all positive* or *all negative*



Which is better: *Patrons?* or *Type?*

Heuristic based on "information gain".

- We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.

Heuristic to Pick the Attribute

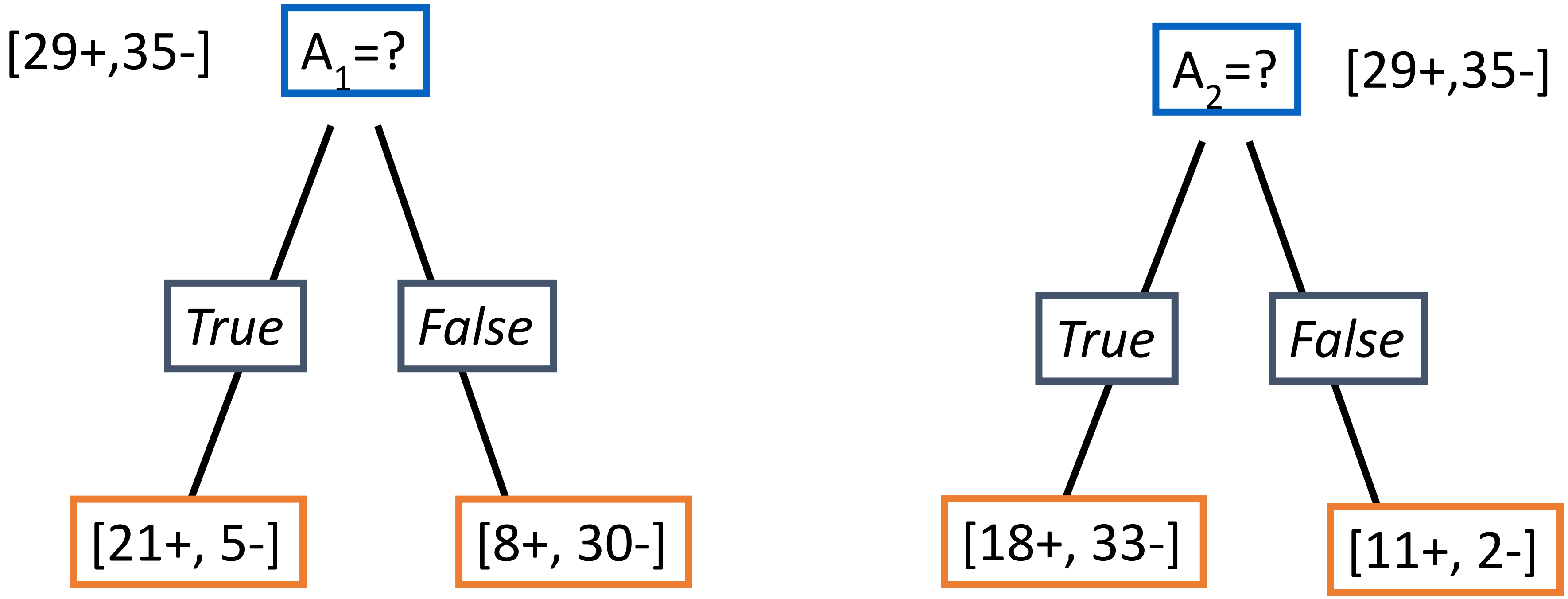
- Preference bias: Occam's Razor
- William of Ockham (1285-1347)

"non sunt multiplicanda entia praeter necessitatem"

entities are not to be multiplied beyond necessity

- Learning the minimal decision tree consistent with data is NP-hard
- Heuristic based on "information gain".
  - We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.

Which Attribute is ”best”?



How to determine the Best Split?

- *Greedy approach:*
  - *Nodes with homogeneous class distribution are preferred*

- *Need a measure of node impurity:*

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

Information gain: measures how well a given attribute separates the training examples according to their target classification

Gini Index: At each node measures, what is the error if you use the most prevalent label

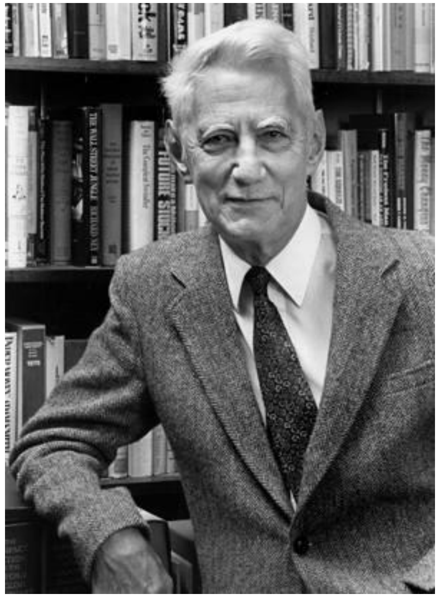
Background: Information theory

- Claude Shannon's seminal work: Mathematical Theory of Communication in 1948
- Information in a message (information entropy)
  - ➔ minimum #bits needed to store/send using a good encoding
- If probability distribution $P(p_1, p_2, \dots, p_n)$ for n messages, its information (or *entropy*) is:

$$H(P) = -(p_1 \log p_1 + p_2 \log p_2 + \cdots + p_n \log p_n)$$

Think we need to send message using A, B, C, D

- If probability is 25% each, we need 2-bits for each. Expected number of bits – 0.25 * 2 + 0.25 * 2 + 0.25 * 2 + 0.25 * 2 = 2
- If probability is 70%, 26%, 2%, 2%. We can use variable-length codes: 0, 10, 110, 111. Expected number of bits = 0.7 * 1 + 0.26*2 + 0.04*3 = 1.34



Claude Shannon

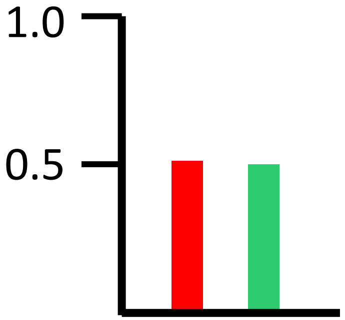
Work done in Bell Labs

Entropy of a Distribution

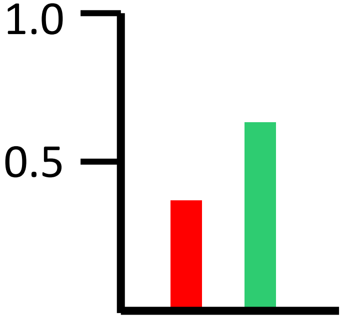
- Quantifies the amount of uncertainty associated with a specific probability distribution

$$H(X) = \sum_x p(X = x) \log_2 \frac{1}{p(X = x)}$$

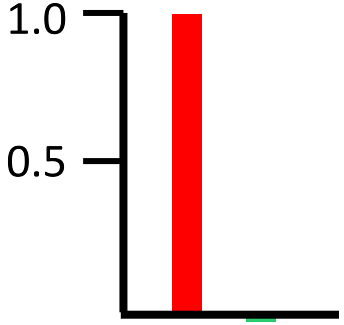
$$H(X) = -\sum_x p(X = x) \log_2 p(X = x)$$



$H(P)$
$= -.5 * (-1) \; + \; 0.5$
$* (-1)$
$= 1$



$H(P)$
$= -\left(\frac{2}{3} * \log\left(\frac{2}{3}\right) + \frac{1}{3} * \log\left(\frac{1}{3}\right)\right)$
= 0.92



$H(P)$
$= -(1 * 1 \; + \; 0 * \log(0))$
$= \; 0$

Entropy (Measure I)

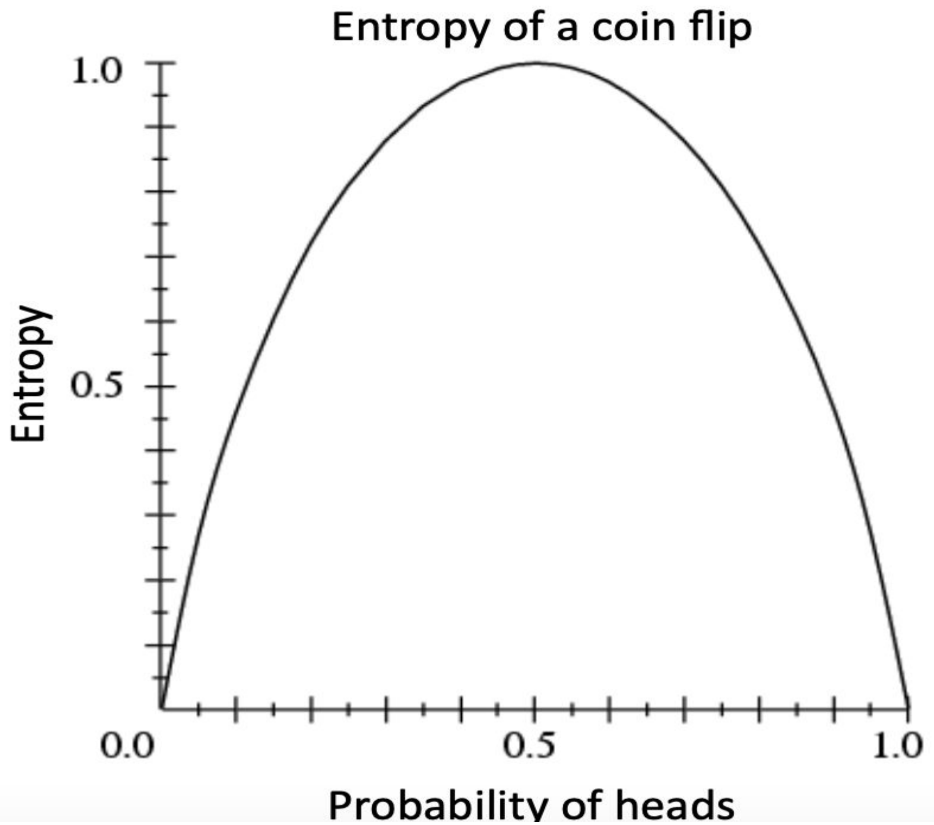
A measure for
- uncertainty
- purity
- information content

Entropy $H(Y)$ of a random variable Y

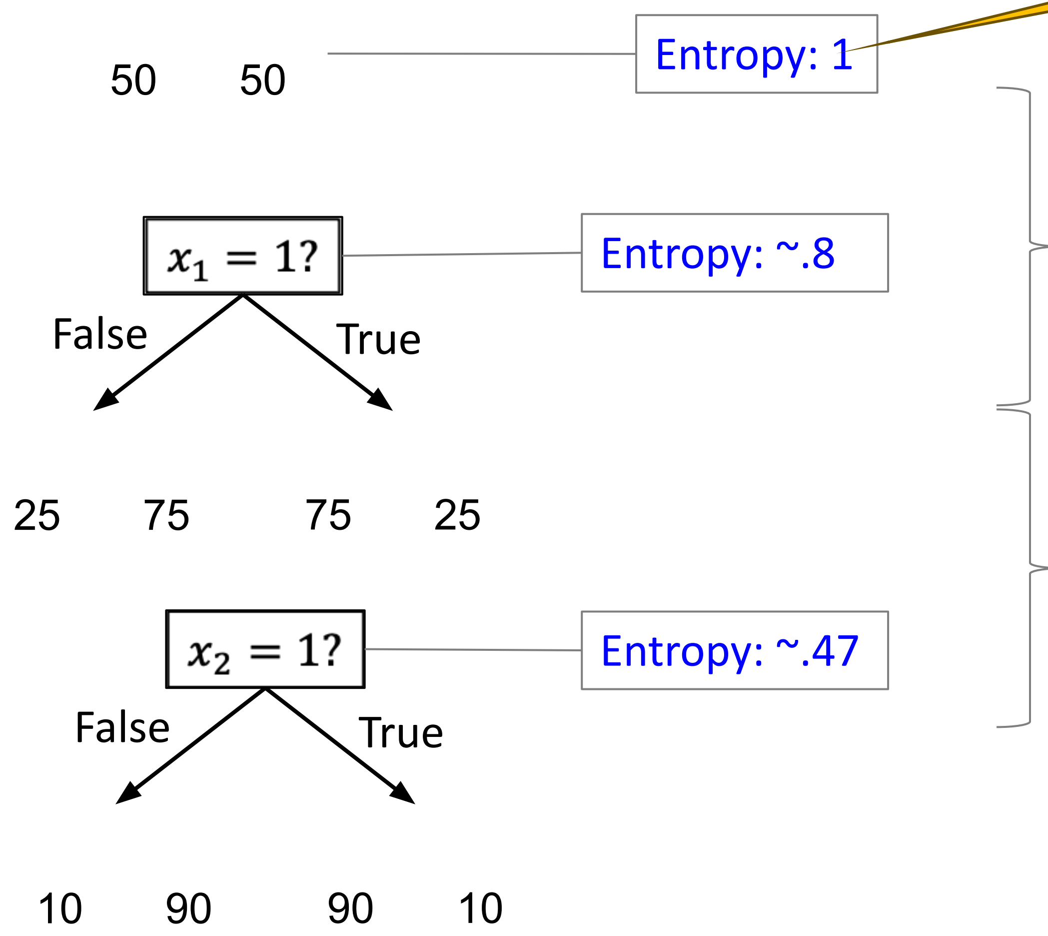
$$H(Y) = -\sum_{i=1}^{k} p(Y = y_i) \log_2 p(Y = y_i)$$

More uncertainty, more entropy!

Information Theory interpretation:
$H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)



Loss for Decision Trees



Information Gain
~.2

Information Gain wrt original
~.53

- $$Loss(S) = \frac{1}{n}\sum_{i=1}^{n} Entropy(Leaf(S_i))$$

Information Gain – reduction in Entropy (loss) from a change to the model

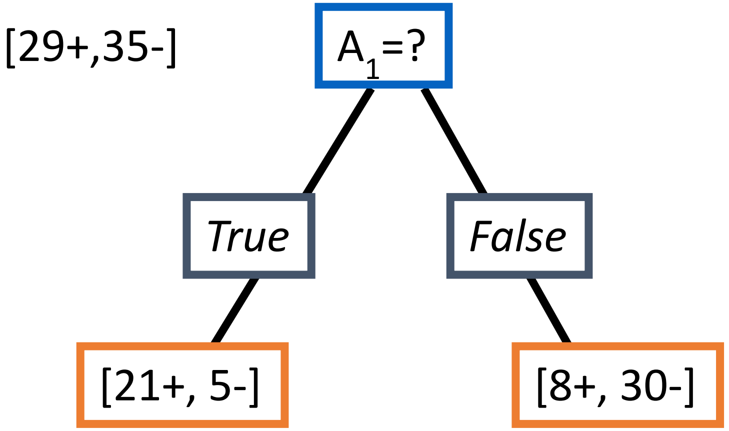
Information Gain

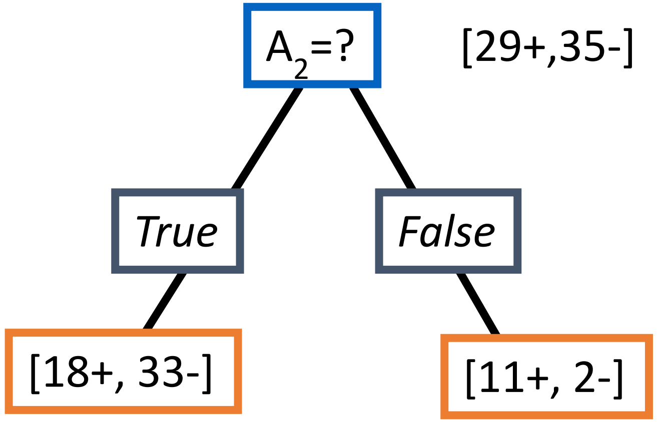
Gain(S,A): expected reduction in entropy due to splitting S on attribute A

$$Gain(S,A) = Entropy(S) - \underbrace{\sum_{v \in values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)}_{\text{Conditional Entropy}}$$

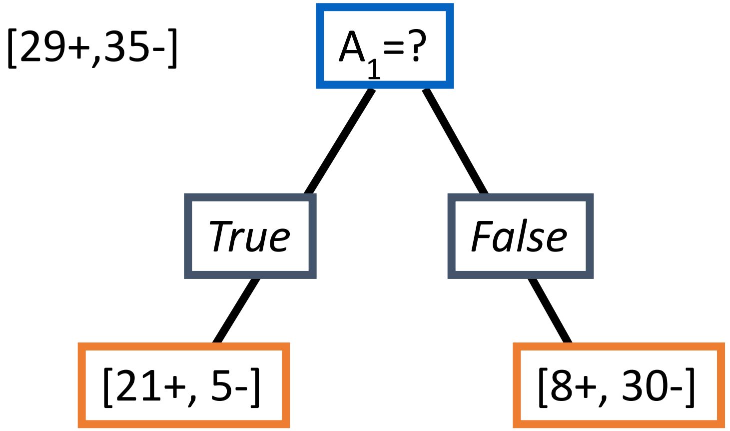
S_v is the subset of S for which attribute A has value v, and

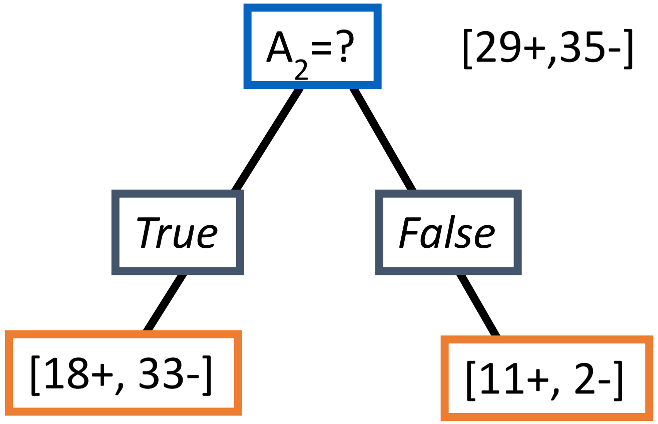
$$Entropy([29+, 35-]) = -\frac{29}{64}\log_2\frac{29}{64} - \frac{35}{64}\log_2\frac{35}{64} = 0.99$$





Information Gain Computation





$$Entropy([29+,35-]) = -\frac{29}{64}\log_2\frac{29}{64} - \frac{35}{64}\log_2\frac{35}{64} = 0.99$$

- Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

$$Gain(S,A_1) = Entropy(S) - \frac{26}{64} * Entropy([21+,5-]) - \frac{38}{64} * Entropy([8+,30-]) = 0.27$$

Entropy([18+,33-]) = 0.94

Entropy([8+,30-]) = 0.62

$$Gain(S,A_2) = Entropy(S) - \frac{51}{64} * Entropy([18+,33-]) - \frac{13}{64} * Entropy([11+,2-]) = 0.12$$

Conditional Entropy

Entropy Definition

$$H(Y) = \sum_y p(Y = y) \log_2 \frac{1}{p(Y=y)}$$

$$H(Y) = -\sum_y p(Y = y) \log_2 p(Y = y)$$

Conditional Entropy

Entropy after splitting on a particular feature

- Must consider expected value over both branches!

Mutual Information: $I(Y; X) = H(Y) - H(Y|X)$

Mutual Information

Let X be a random variable with $X \in \mathcal{X}$.
Let Y be a random variable with $Y \in \mathcal{Y}$.

Entropy: $H(Y) = -\sum_{y \in \mathcal{Y}} P(Y = y) \log_2 P(Y = y)$

Specific Conditional Entropy: $H(Y \mid X = x) = -\sum_{y \in \mathcal{Y}} P(Y = y \mid X = x) \log_2 P(Y = y \mid X = x)$

Conditional Entropy: $H(Y \mid X) = \sum_{x \in \mathcal{X}} P(X = x) H(Y \mid X = x)$

Mutual Information: $I(Y; X) = H(Y) - H(Y|X)$

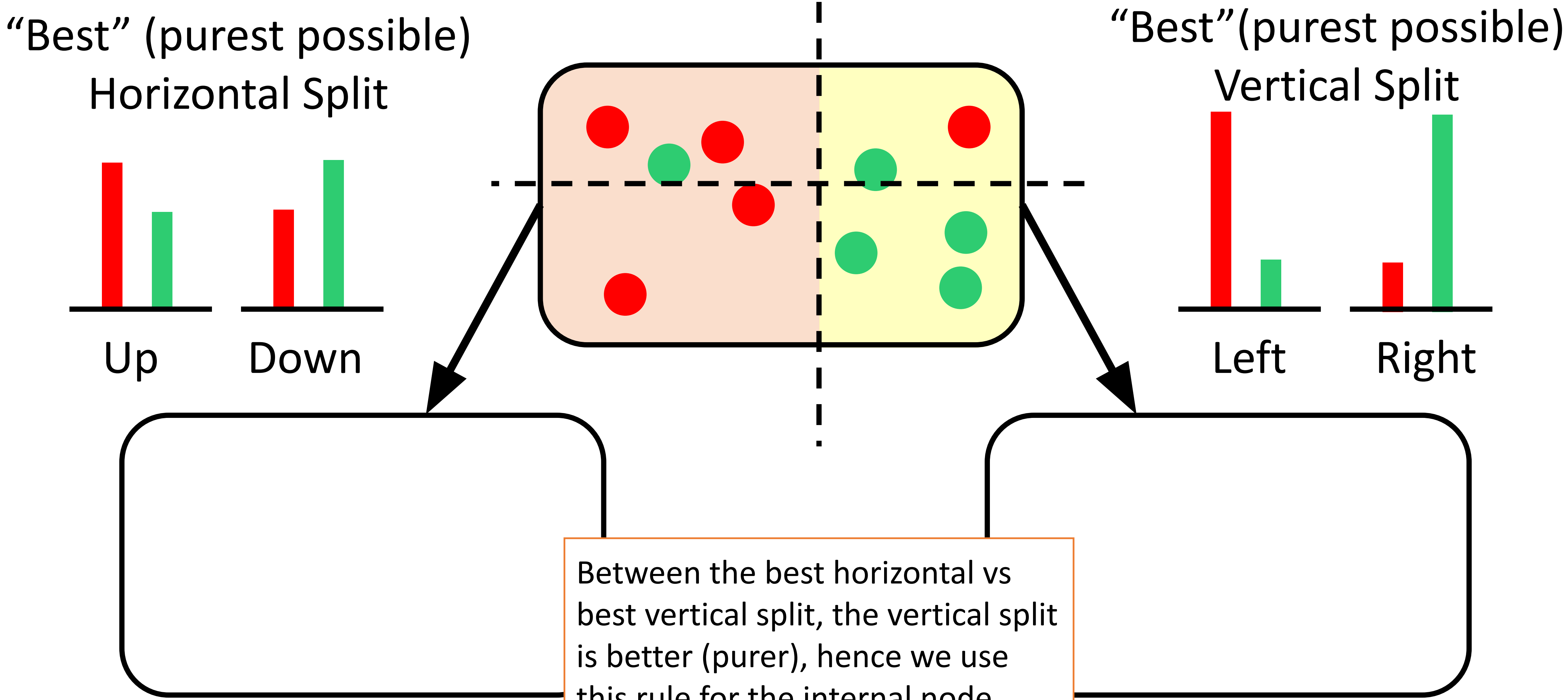
- **Entropy** measures the **expected # of bits** to code one random draw from X.
- For a decision tree, we want to **reduce the entropy of the random variable we are trying to predict**!

Conditional entropy is the expected value of specific conditional entropy $E_{P(X=x)}[H(Y \mid X = x)]$

Informally, we say that **mutual information** is a measure of the following: *If we know X, how much does this reduce our uncertainty about Y?*

Slide credit: CMU MLD Matt Gormley

An Illustration: DT with Real-Valued Features



Between the best horizontal vs best vertical split, the vertical split is better (purer), hence we use this rule for the internal node

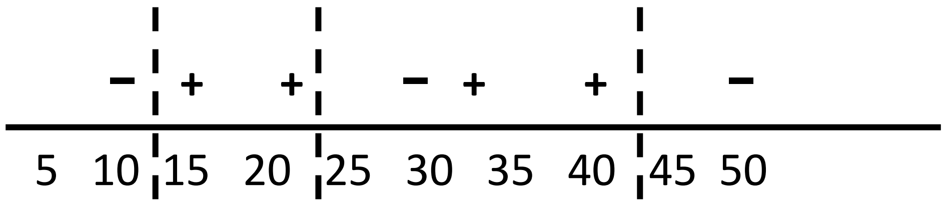
This illustration's credit: Purushottam Kar

DT with Real-Valued Features

Example:

- Length (L): 10 15 21 28 32 40 50
- Class: – + + – + + –
- Check thresholds:

  L < 12.5; L < 24.5; L < 45



How to find the split with the highest gain ?

For each continuous feature A:

- Sort examples according to the value of A
- For each ordered pair (x,y) with different labels
- Check the mid-point as a possible threshold.

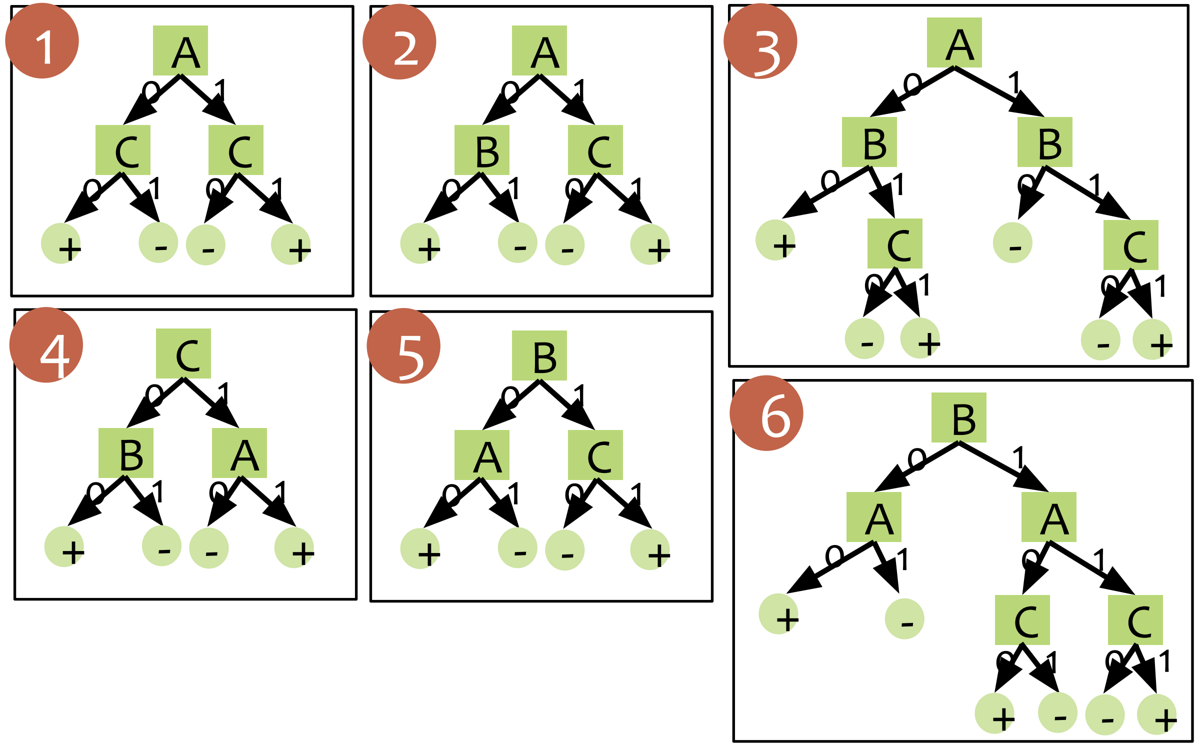
Decision Trees

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
2	5.9	3.0	5.1	1.8

- A few tools
- Majority vote:
- $\hat{y} = \underset{c}{\operatorname{argmax}} \frac{N_c}{N}$
- Classification error rate:
- $ErrorRate = \frac{1}{N} \sum_i \mathbb{I}\left(y^{(i)} \neq \hat{y}^{(i)}\right)$
- What fraction did we predict incorrectly
- Expected value
- $\mathbb{E}[f(X)] = \sum_{x \in \mathcal{X}} f(x)\, P(X = x)$ or $\mathbb{E}[f(X)] = \int_{\mathcal{X}} f(x)\, p(x)\, dx$
-

Poll 3

- Which of the following trees would be learned by the decision tree learning algorithm using "error rate" as the splitting criterion?
- (Assume ties are broken alphabetically.)



Dataset:

Output Y, Attributes A, B, C

Y	A	B	C
+	0	0	0
+	0	0	1
-	0	1	0
+	0	1	1
-	1	0	0
-	1	0	1
-	1	1	0
+	1	1	1

Slide credit: CMU MLD Matt Gormley

Gini Impurity: Decision Trees (Measure II)

- Gini impurity estimates the following
  - Choose an element randomly.
  - Label it using the distribution of labels on the set.
  - How often the element is incorrectly labeled?
- For a set of items with C classes, with relative frequency p_c for class c, the probability of choosing an item with label c is p_c, and the probability of misclassifying the item is

$$\sum_{k \neq c} p_k = 1 - p_c$$

- The Gini Index is computed by summing pairwise products of these probabilities for each class label:

$$GINI(p) = \sum_{c=1}^{C} p_c(1 - p_c) = 1 - \sum_{c=1}^{C} p_c^2$$

Examples of Computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	0
C2	6

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	1
C2	5

P(C1) = 1/6   P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	2
C2	4

P(C1) = 2/6   P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Measure of Impurity: Gini Index

Gini Index for a given node t :

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

$p(j \mid t)$ is the relative frequency of class j at node t

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information [n_c: number of classes]
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Splitting Based on GINI

Used in CART, SLIQ, SPRINT.

When a node *p* is split into *k* partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

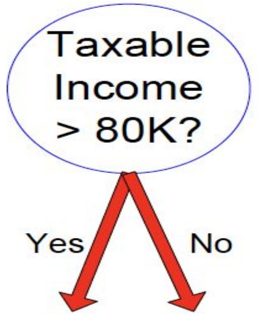
where, n_i = number of records at child i,

n = number of records at node p.

Continuous Attributes: Computing Gini Index

- Use binary decisions (A<v and A>=v)
- Many choice.
- Each value v has a count matrix associated with it
  - Class counts with A >= v
  - Class counts with A < v
- Method to choose best v
  - For each v, scan the dataset to get the count matrix. Compute Gini Index.
  - Choose the one with min. Gini Index.
  - **Inefficient. Repetition of work!**

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values	60	70	75	85	90	95	100	120	125	220

Split Positions	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		*0.300*		0.343		0.375		0.400		0.420	

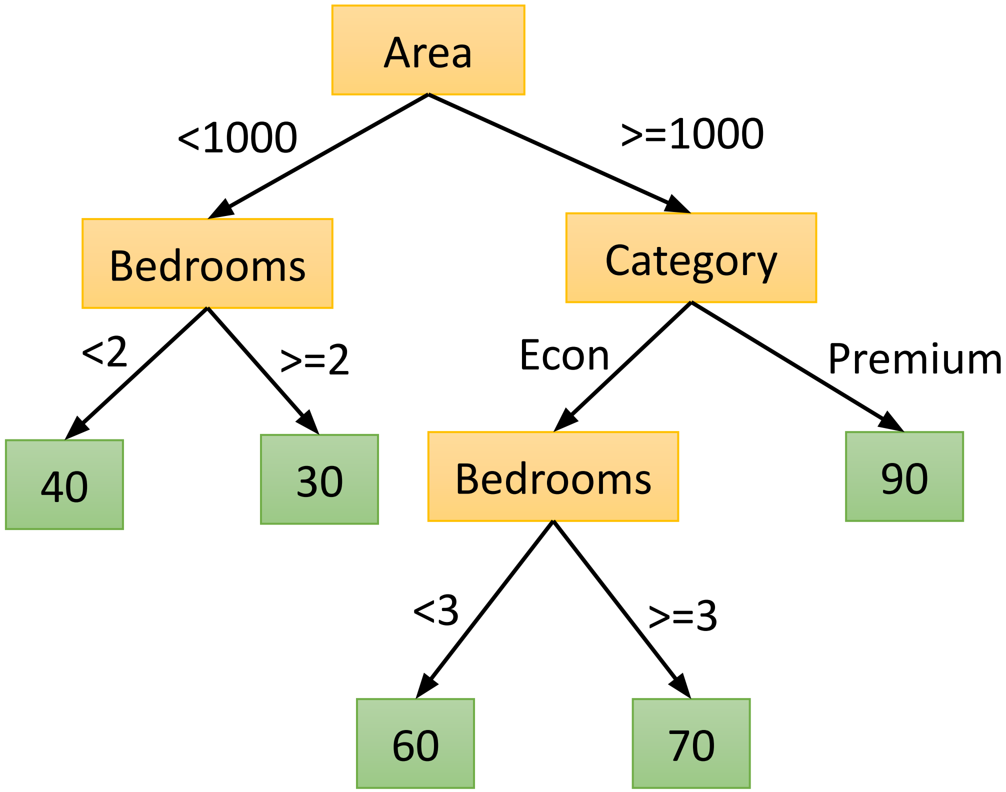
Continuous Attributes: Variance (Measure III)

Another Example of Splitting Criteria:

Variance Reduction

If a node is entirely homogeneous, then the variance is zero.

1. For each split, individually calculate the variance of each child node
2. Calculate the variance of each split as the weighted average variance of child nodes
3. Select the split with the lowest variance



Practical Issues

- Missing Values
- Attributes with different costs
  - Change information gain so that low cost attribute are preferred
- Dealing with features with different # of values

Experimental Machine Learning

- Machine Learning is an Experimental Field
- Basics:
  - Split your data into two (or three) sets:
    - Training data (often 70-90%)
    - Test data (often 10-20%)
    - Development data (10-20%)
- You need to report performance on test data, but you are not allowed to look at it.
  - You are allowed to look at the development data (and use it to tweak parameters)