

CS60050

Machine Learning

Decision Trees

Somak Aditya

Assistant Professor
Department of CSE
August 16, 2023

Sudeshna Sarkar

Professor
CoE AI, Department of CSE



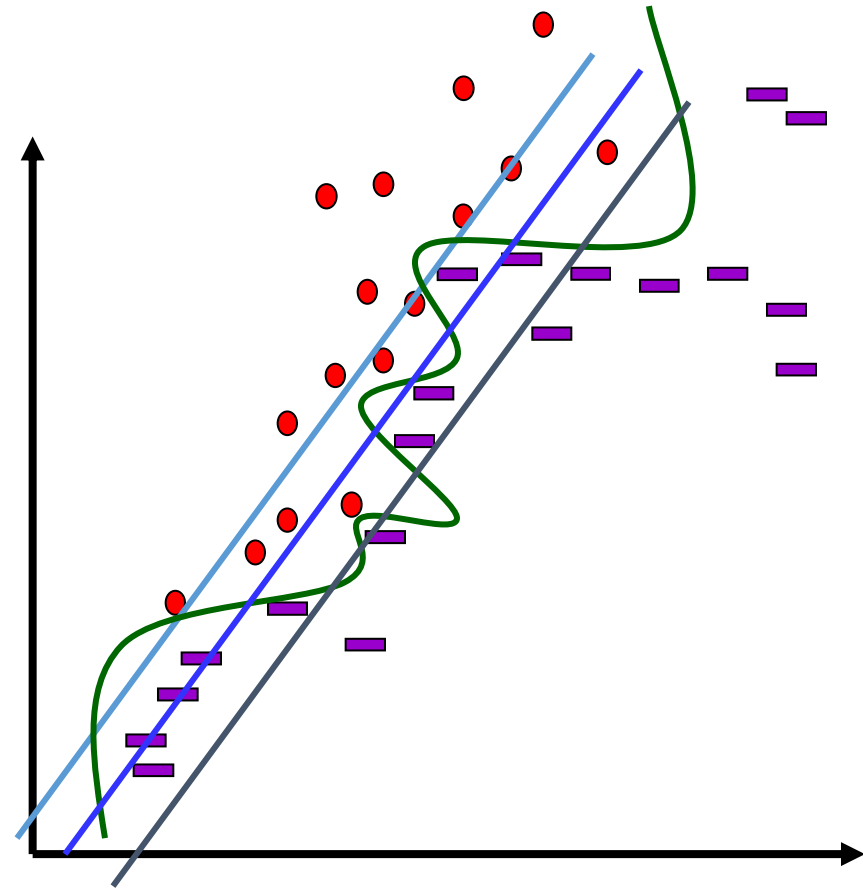
Recap

- Learning problem (classification)
Find a function that
best separates the data
- What function?

Linear:

$$Y = \text{sign}(\theta^T X)$$

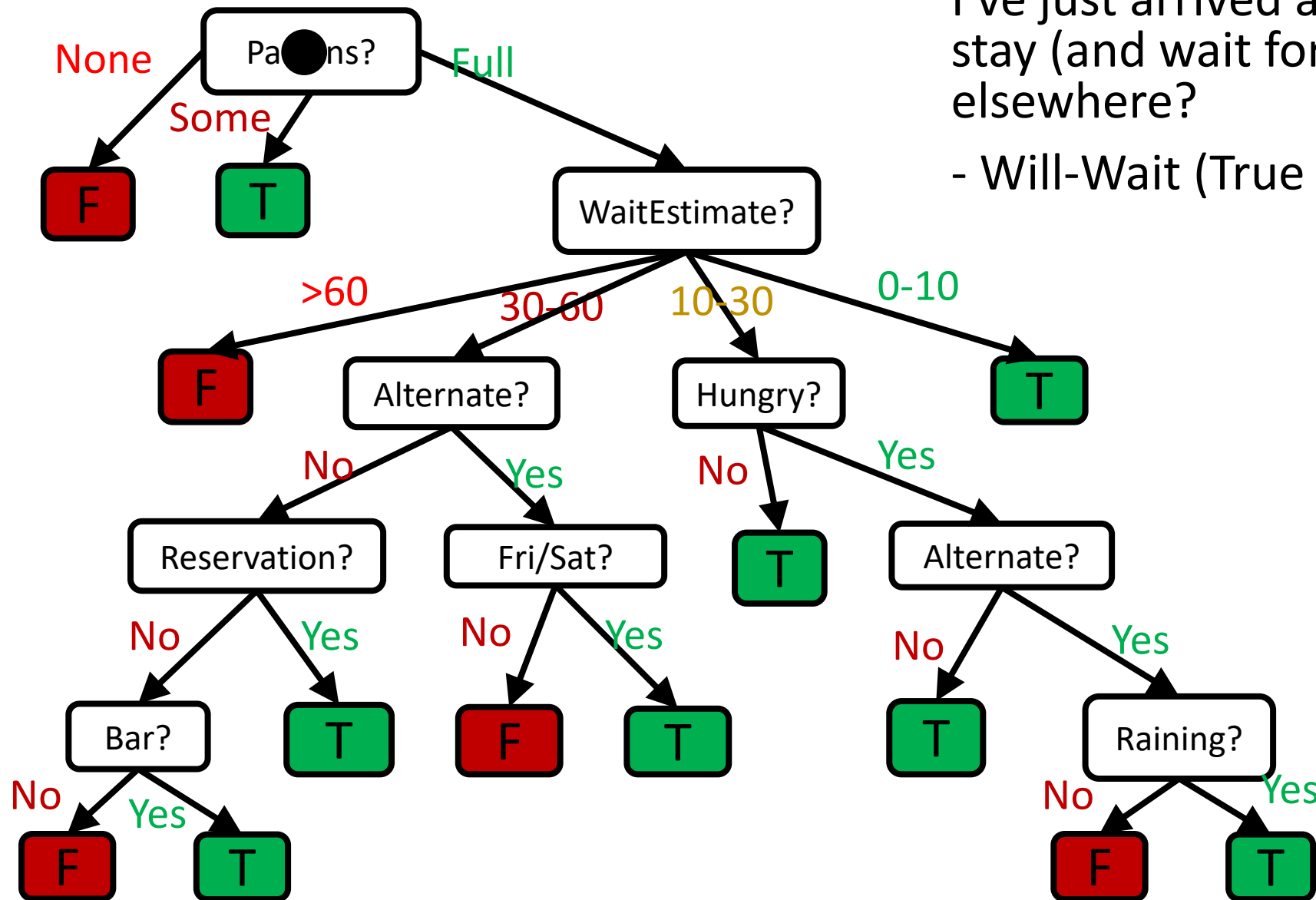
- Limitations of Linear Functions



Decision Trees

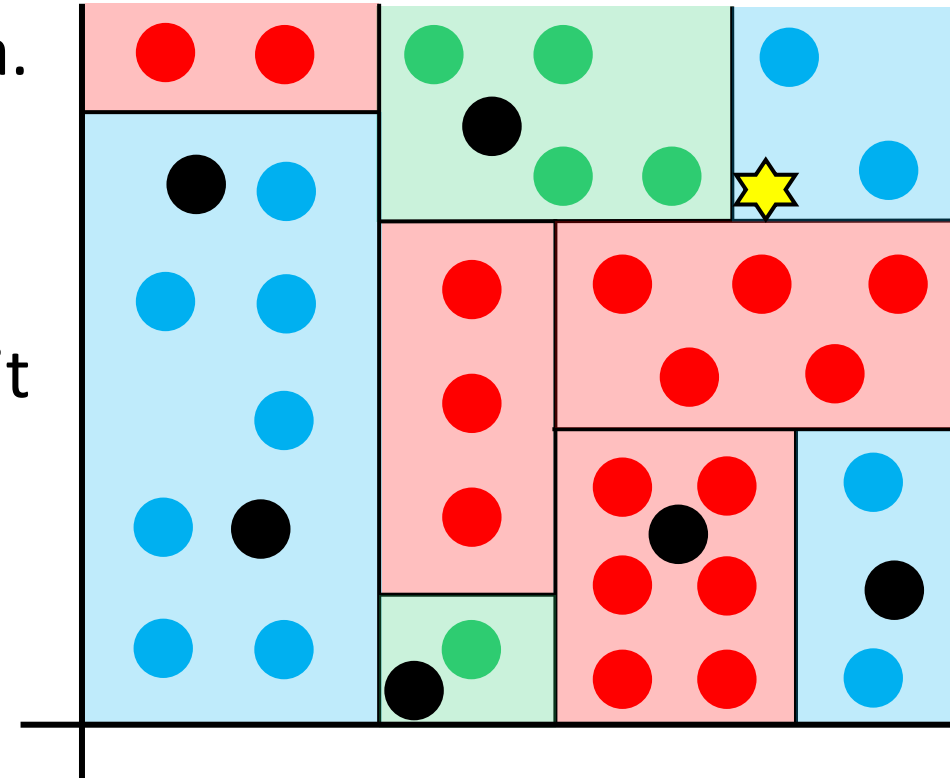
I've just arrived at a restaurant: should I stay (and wait for a table) or go elsewhere?

- Will-Wait (True or False?)

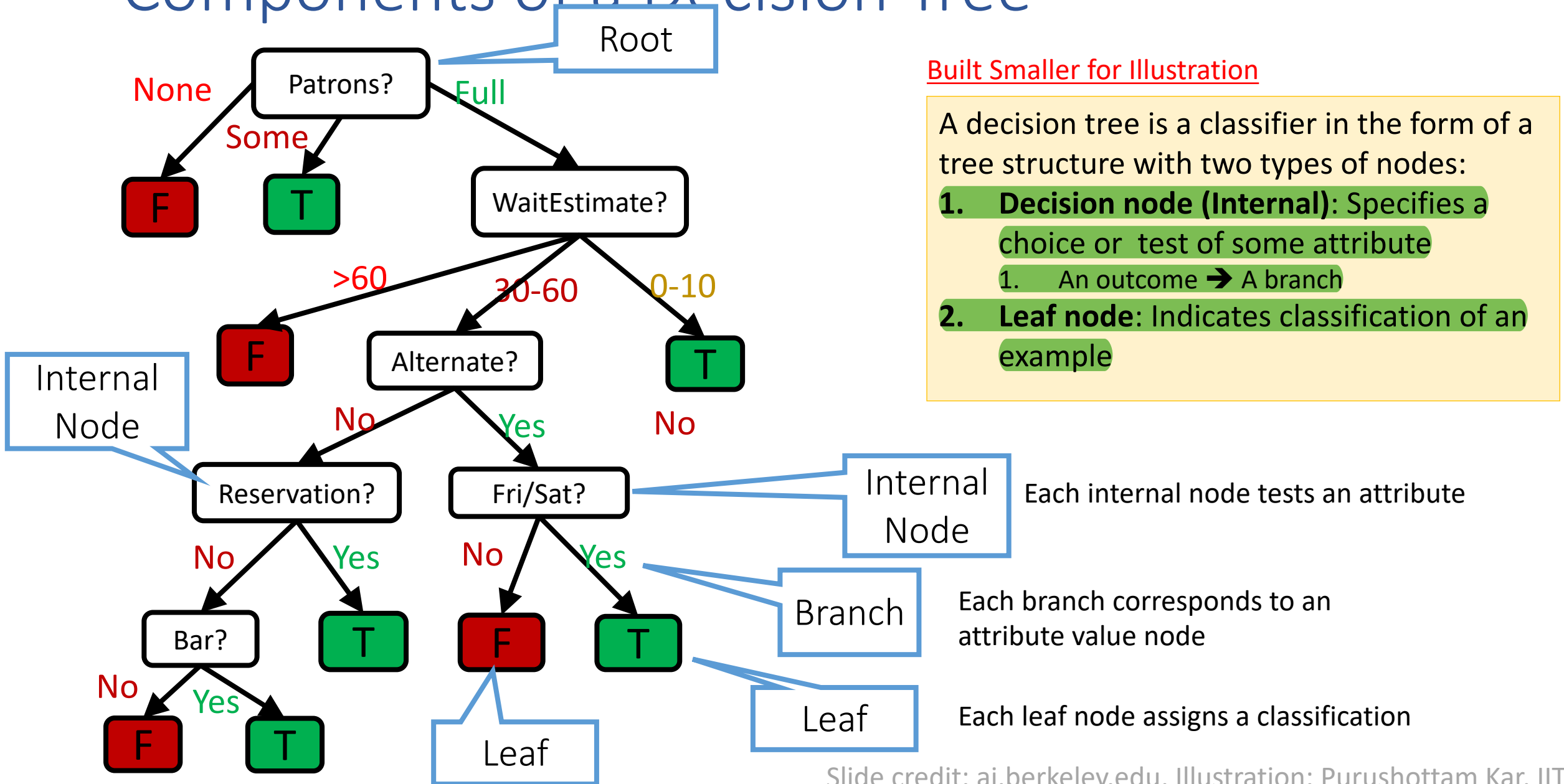


Decision Trees for Classification

- 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

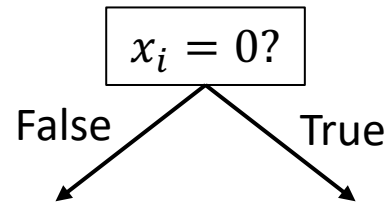


Components of a Decision Tree

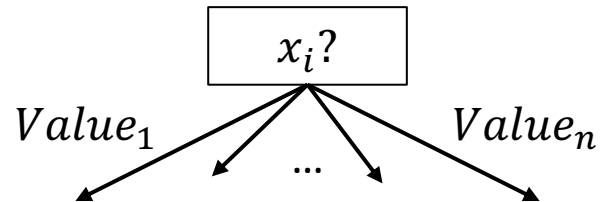


Type of Internal Tests

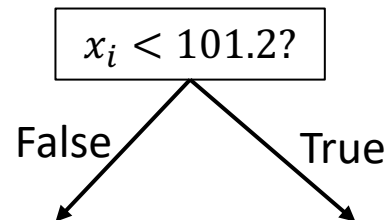
Binary Feature



Categorical Feature



Numeric Feature



Types of Leaves

Classification

$$y = POS$$

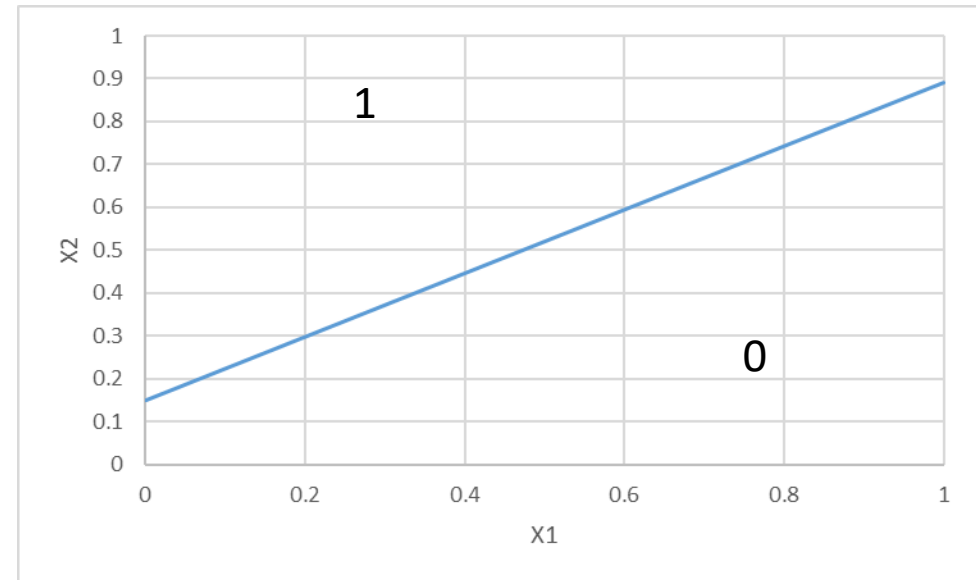
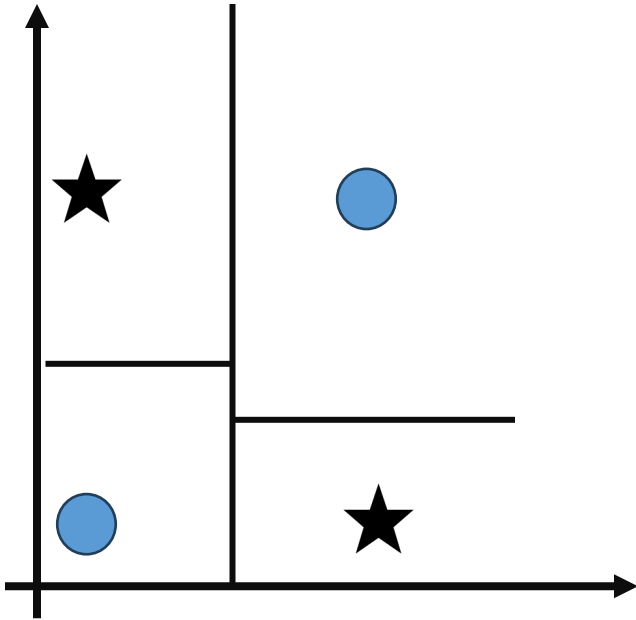
Regression

$$y = 5.2$$

Probability Estimate

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

Decision Trees vs Linear Models



Adding nodes (structure) allows more powerful representation

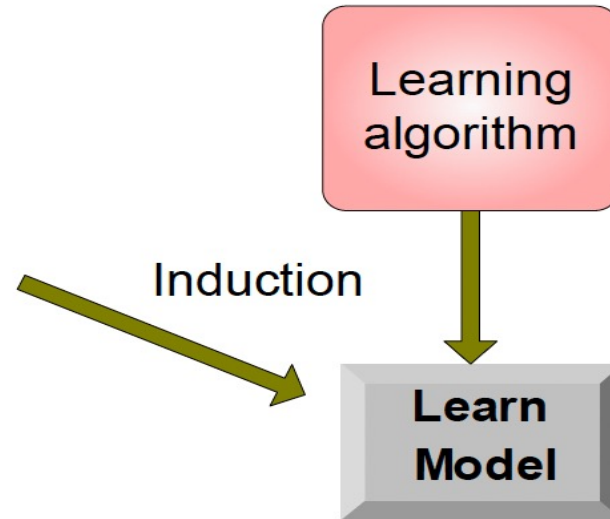
Classification in a Nutshell

| <i>Tid</i> | <i>Attrib1</i> | <i>Attrib2</i> | <i>Attrib3</i> | <i>Class</i> |
|------------|----------------|----------------|----------------|--------------|
| 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

| <i>Tid</i> | <i>Attrib1</i> | <i>Attrib2</i> | <i>Attrib3</i> | <i>Class</i> |
|------------|----------------|----------------|----------------|--------------|
| 11 | No | Small | 55K | ? |
| 12 | Yes | Medium | 80K | ? |
| 13 | Yes | Large | 110K | ? |
| 14 | No | Small | 95K | ? |
| 15 | No | Large | 67K | ? |

Test Set

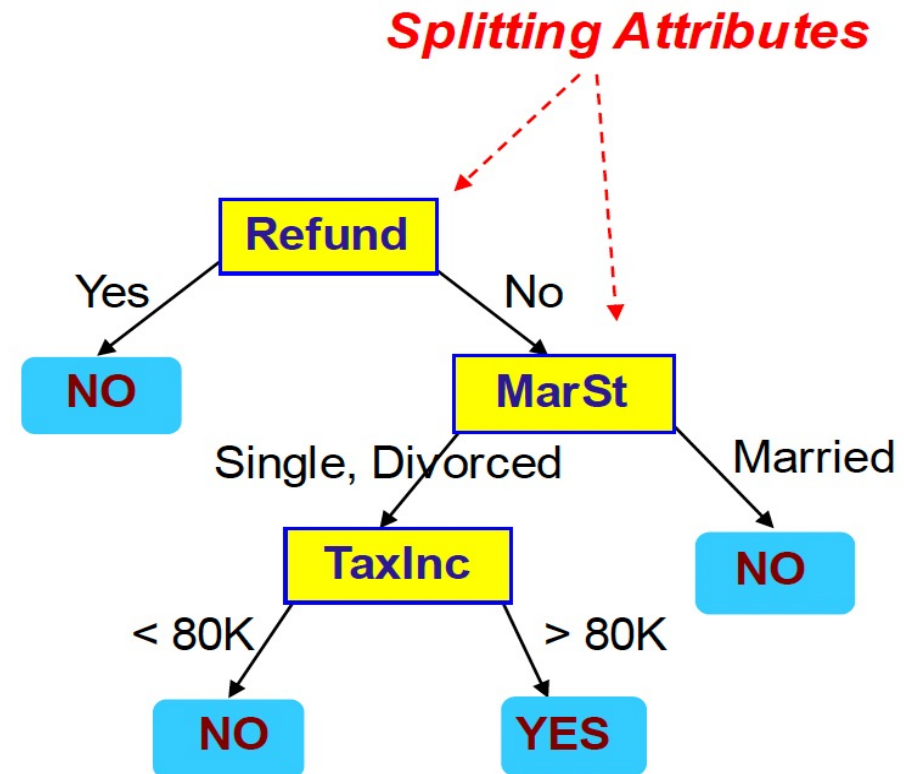


Learning Decision Tree from Data

| <i>Tid</i> | <i>Refund</i> | <i>Marital Status</i> | <i>Taxable Income</i> | <i>Cheat</i> |
|------------|---------------|-----------------------|-----------------------|--------------|
| 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 |

categorical
categorical
continuous
class

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 \leftarrow CHOOSE-BEST-ATTRIBUTE(*dataset*, *attributes*)

tree \leftarrow A new DT with root test *att*

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

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

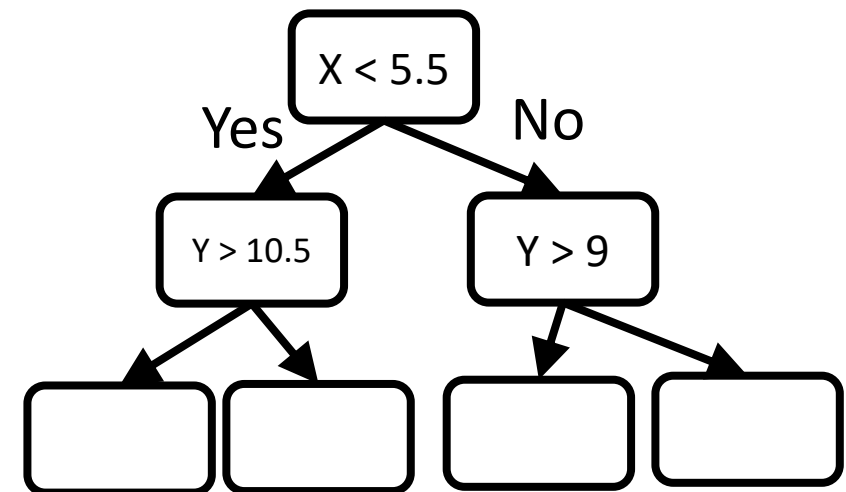
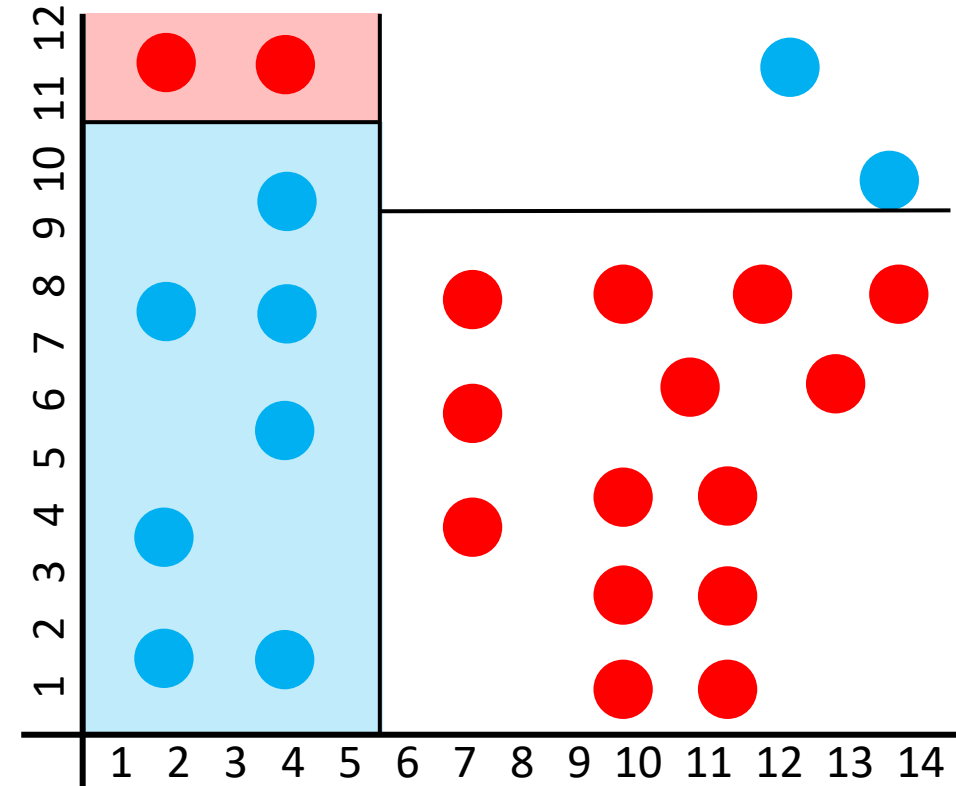
add branch to *tree* with value 1, subtree LeftNode

add branch to *tree* with value 0, subtree RightNode

generalize for multiple values

return *tree*

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 \leftarrow CHOOSE-BEST-ATTRIBUTE(*dataset*, *attributes*)

tree \leftarrow 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 \leftarrow CHOOSE-BEST-ATTRIBUTE(*dataset*, *attributes*)

tree \leftarrow A new DT with root test *att*

For each value v_j of attribute *att*:

$\text{ChildNode}_j = \text{BuildTree}(D(\text{att} = v_j), \text{attributes} - \{\text{att}\})$

 add branch to *tree* with value v_j , subtree ChildNode_j

return *tree*

Choices

1. When to stop

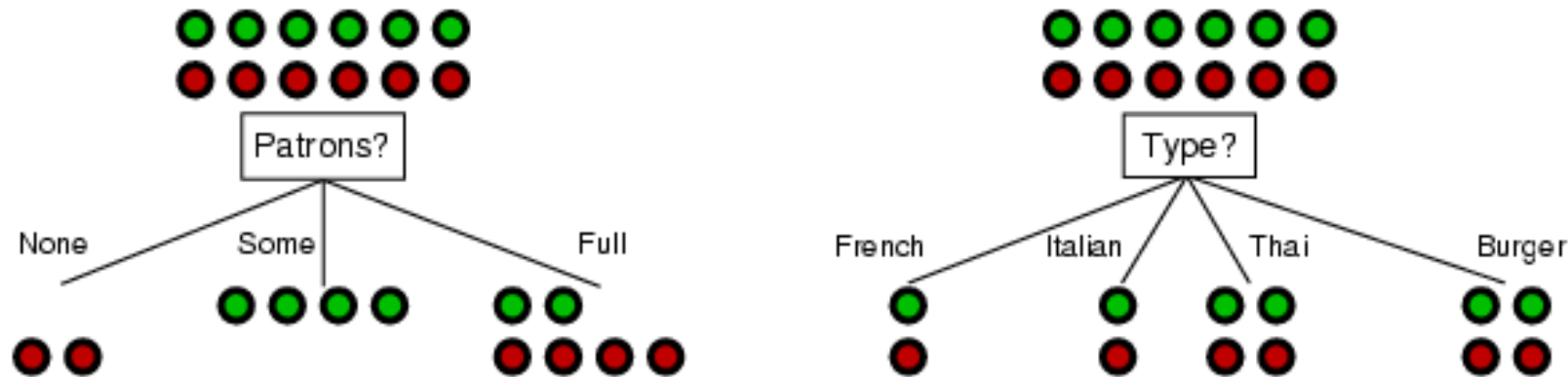
- no more input features
- all examples are classified the same
- too few examples to make an informative split

2. Which test to split on

- split gives smallest error.

Choosing an attribute

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

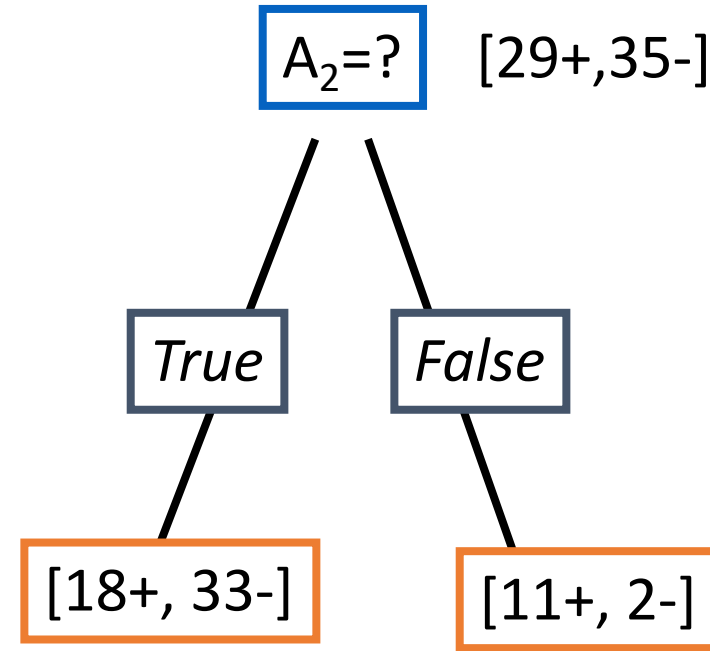
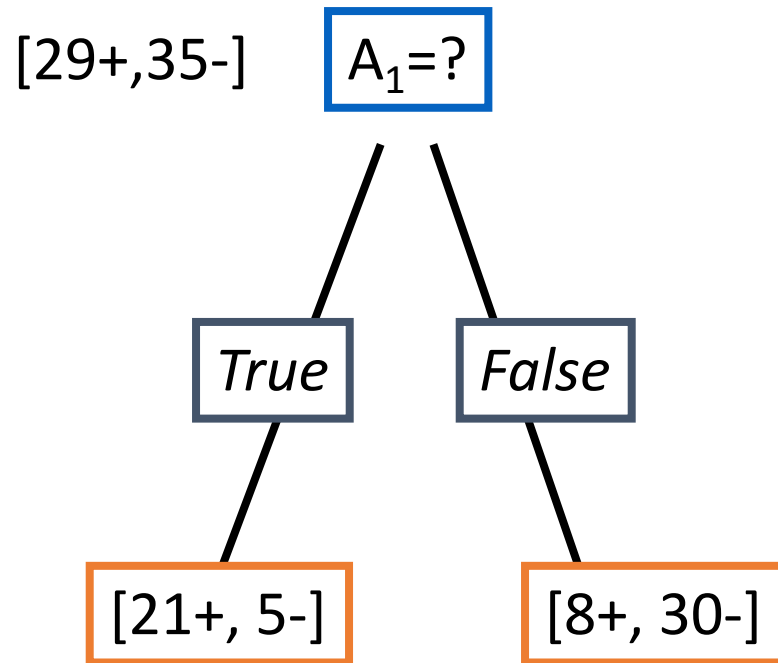


Which is better: *Patrons?* or *Type?*

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.

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 |

Non-homogeneous,
High degree of impurity

| |
|-------|
| C0: 9 |
| C1: 1 |

Homogeneous,
Low degree of impurity

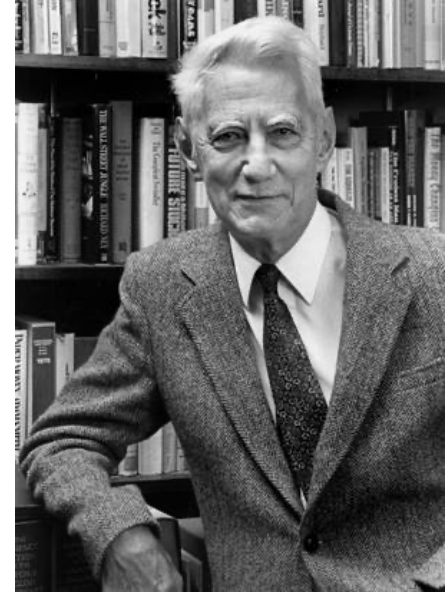
Information gain: measures how well a given attribute separates the training examples according to their target classification

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:

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



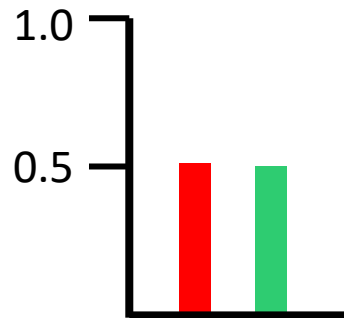
Claude Shannon

Entropy of a Distribution

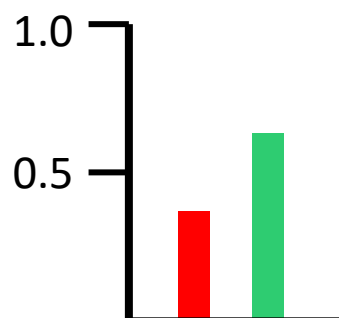
- Quantifies the amount of uncertainty associated with a specific probability distribution

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

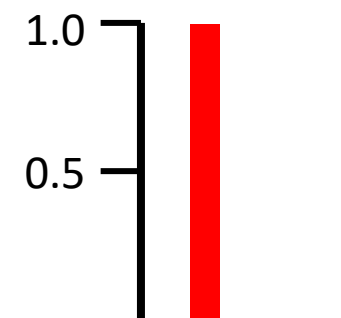
$$H(X) = - \sum_x p(X = x) \log_2 p(X = x)$$



$$\begin{aligned} H(P) &= -0.5 * (-1) + 0.5 \\ &\quad * (-1) \\ &= 1 \end{aligned}$$



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



$$\begin{aligned} H(P) &= -(1 * 1 + 0 * \log(0)) \\ &= 0 \end{aligned}$$

Entropy (Measure I)

A measure for
uncertainty
purity
information content

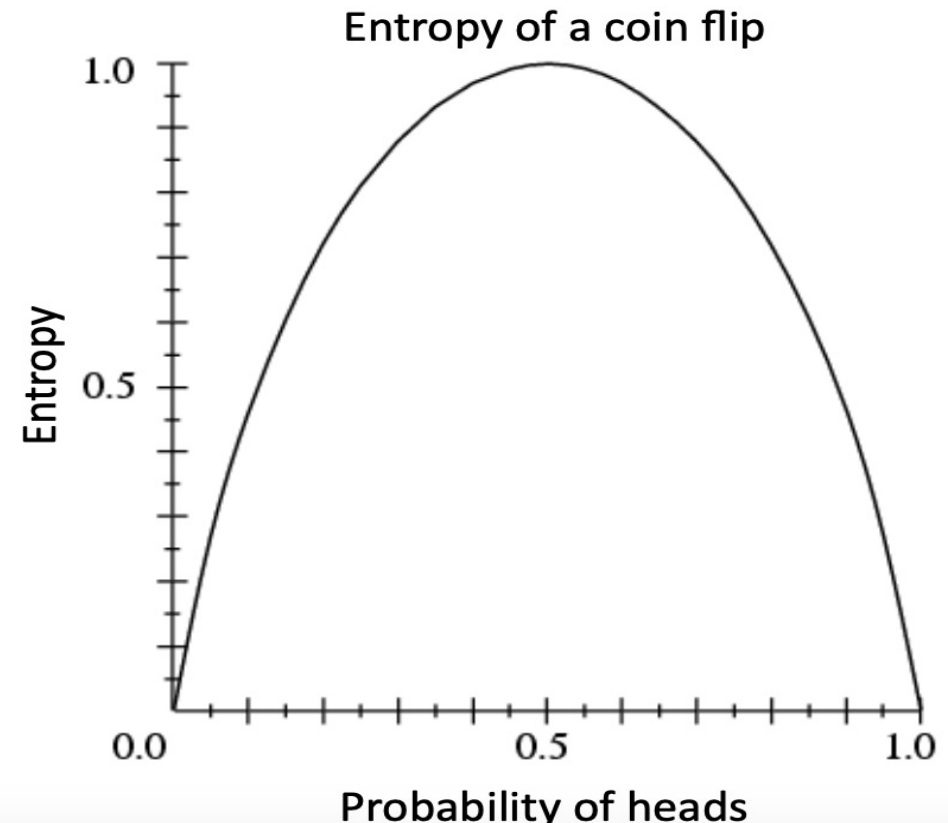
Entropy $H(Y)$ of a random variable Y

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

More uncertainty, more entropy!

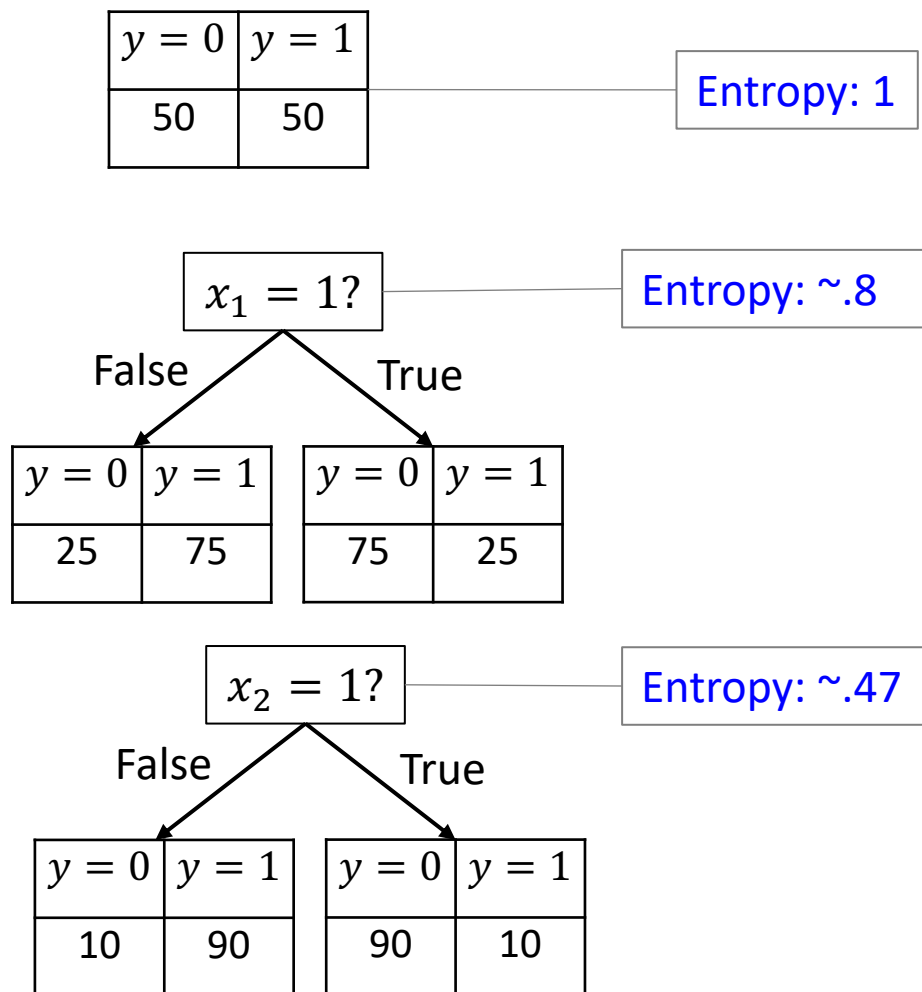
Information Theory interpretation:

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



Loss for Decision Trees

Should we stop splitting (in the first place)? Or continue?



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

Information Gain
 $\sim .2$

Information Gain
 $\sim .33$

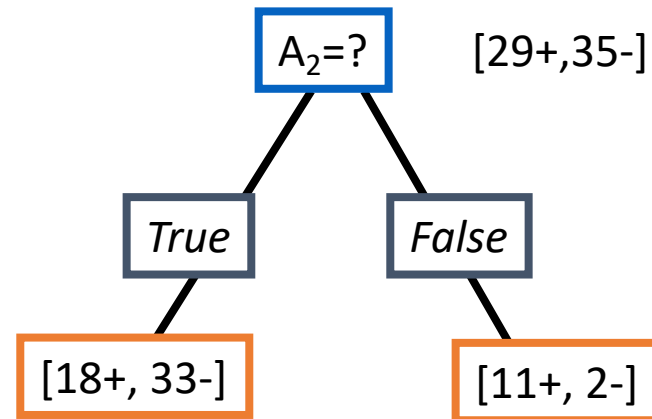
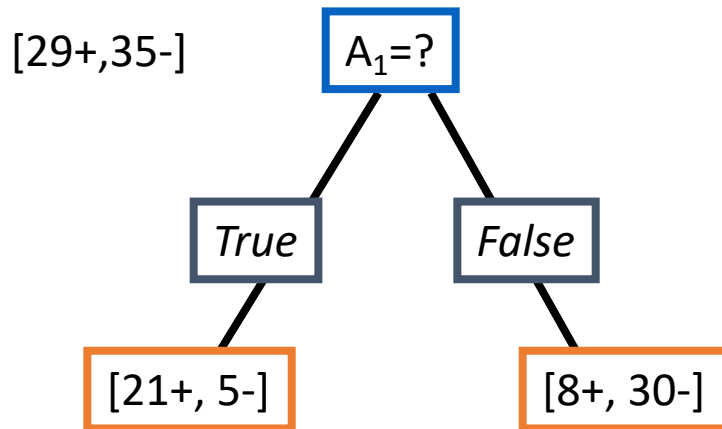
Information Gain – reduction in Entropy (loss) from a change to the model

Information Gain

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

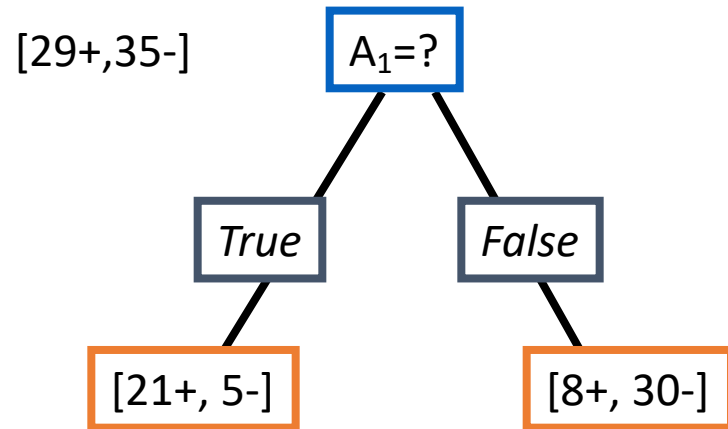
$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$$

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



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

Information Gain Computation

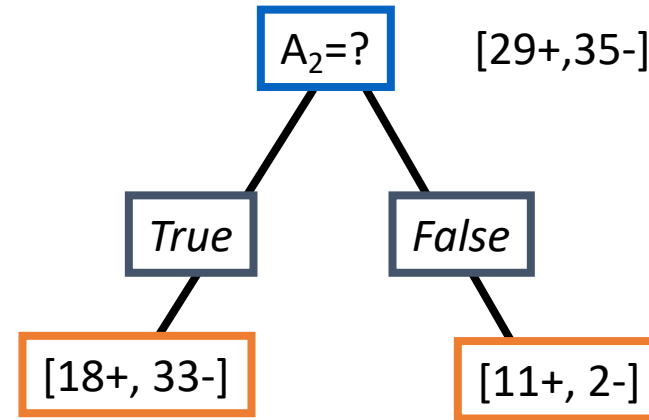


$$\text{Entropy}([21+,5-]) = 0.71$$

$$\text{Entropy}([8+,30-]) = 0.74$$

$$\text{Gain}(S, A_1) = \text{Entropy}(S)$$

$$\begin{aligned}
 & -\frac{26}{64} * \text{Entropy}([21+, 5 -]) \\
 & -\frac{38}{64} * \text{Entropy}([8+, 30-]) \\
 & = 0.27
 \end{aligned}$$



$$\text{Entropy}([18+,33-]) = 0.94$$

$$\text{Entropy}([8+,30-]) = 0.62$$

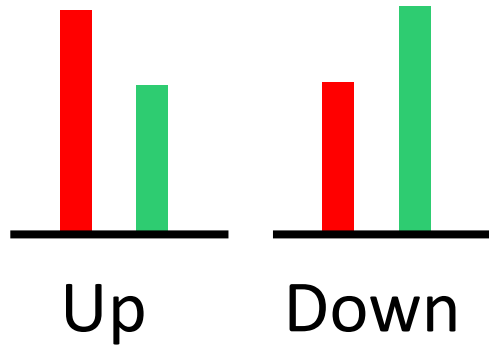
$$\text{Gain}(S, A_2)$$

$$\begin{aligned}
 & = \text{Entropy}(S) - \frac{51}{64} * \text{Entropy}([18+, 33-]) \\
 & - \frac{13}{64} * \text{Entropy}([11+, 2-]) \\
 & = 0.12
 \end{aligned}$$

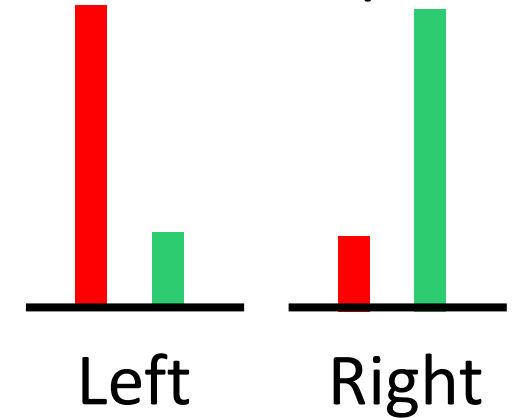
$$\begin{aligned}
 & \text{Entropy}([29+, 35 -]) \\
 & = -\frac{29}{64} \log_2 \frac{29}{64} - \frac{35}{64} \log_2 \frac{35}{64} \\
 & = 0.99
 \end{aligned}$$

An Illustration: DT with Real-Valued Features

“Best” (purest possible)
Horizontal Split



“Best” (purest possible)
Vertical Split



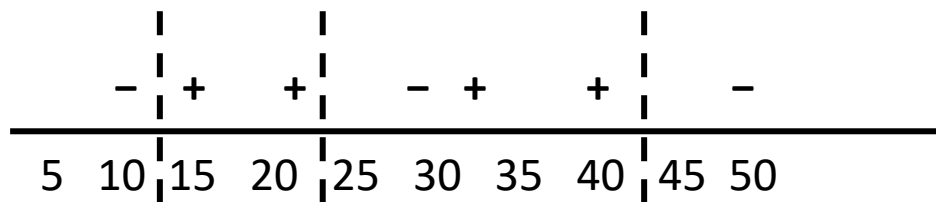
Between the best horizontal vs best vertical split, the vertical split is better (purer), hence we use this rule for the internal node

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.

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

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

| | |
|----|----------|
| C1 | 0 |
| C2 | 6 |

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

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

| | |
|----|----------|
| C1 | 1 |
| C2 | 5 |

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

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

| | |
|----|----------|
| C1 | 2 |
| C2 | 4 |

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

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

Measure of Impurity: Gini Index

Gini Index for a given node t :

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

$p(j | t)$ 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,

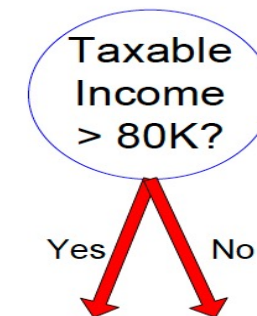
$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

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 $A \geq v$)
- Many choice.
- Each value v has a count matrix associated with it
 - Class counts with $A \geq v$
 - 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 | | | | | | | | | | | | | | | | | | | | | |
| | | 60 | | 70 | | 75 | | 85 | | 90 | | 95 | | 100 | | 120 | | 125 | | 220 | | | |
| Sorted Values | 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 | | <u>0.300</u> | | 0.343 | | 0.375 | | 0.400 | | 0.420 | |

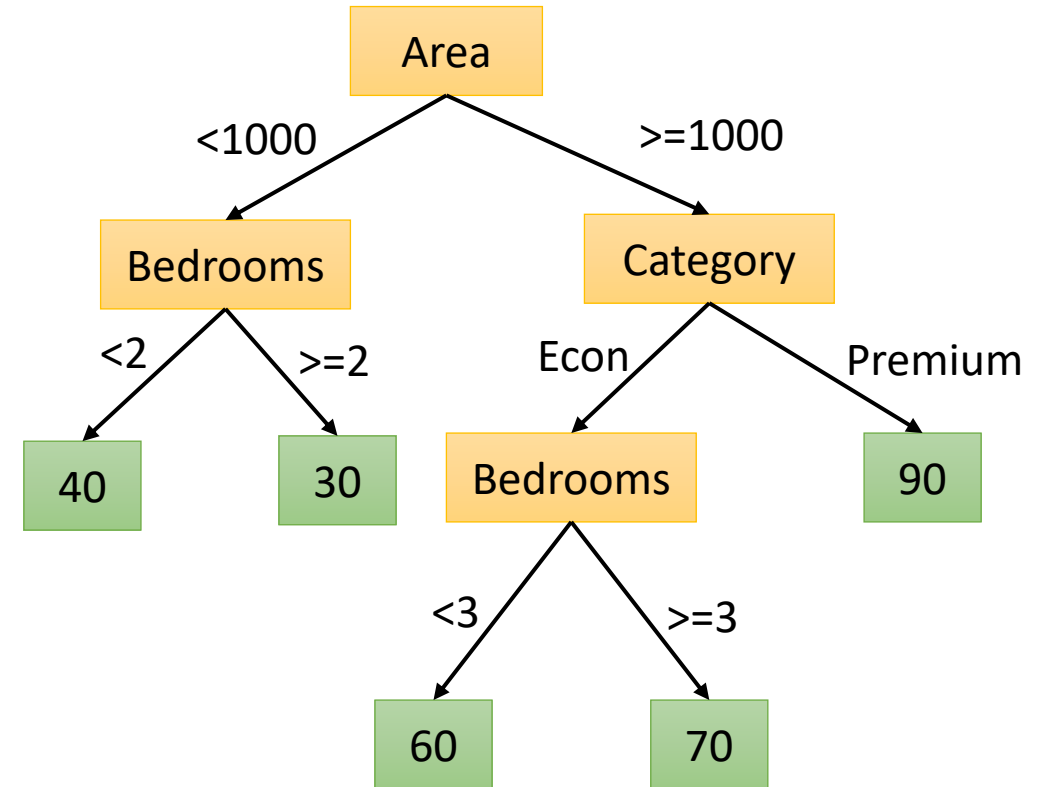
Continuous Attributes: Variance (Measure III)

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)