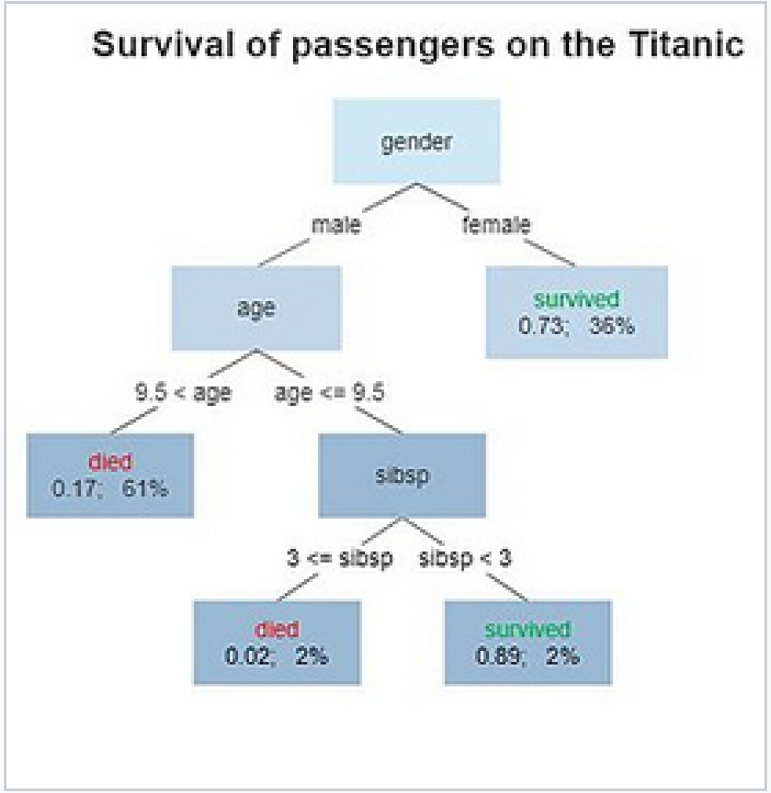
Decision Trees

UNIVERSITY OF CANBERRA

Prof Dharmendra Sharma

Unit Codes: 11482, 11512

Outline



A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf. Summarizing: Your chances of survival were good if you were (i) a female or (ii) a male younger than 9.5 years with strictly less than 3 siblings.

Decision Trees

- The Decision Tree creation was firstly presented by J. Ross Quinlan from the University of Sydney. It was called the Iterative Dichotomiser 3 (ID3) in his book *Machine Learning*, vol.1, no. 1, in 1975.
- A **decision tree** (DT) is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.
- It is one way to display an algorithm that only contains conditional control statements.
- DTs are commonly used in operations research, decision sciences specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

Function Approximation

Problem Setting

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \to \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \to \mathcal{Y}\}$

Input: Training examples of unknown target function f

$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

Output: Hypothesis $h \in H$ that best approximates f

Based on slide by Tom Mitchell

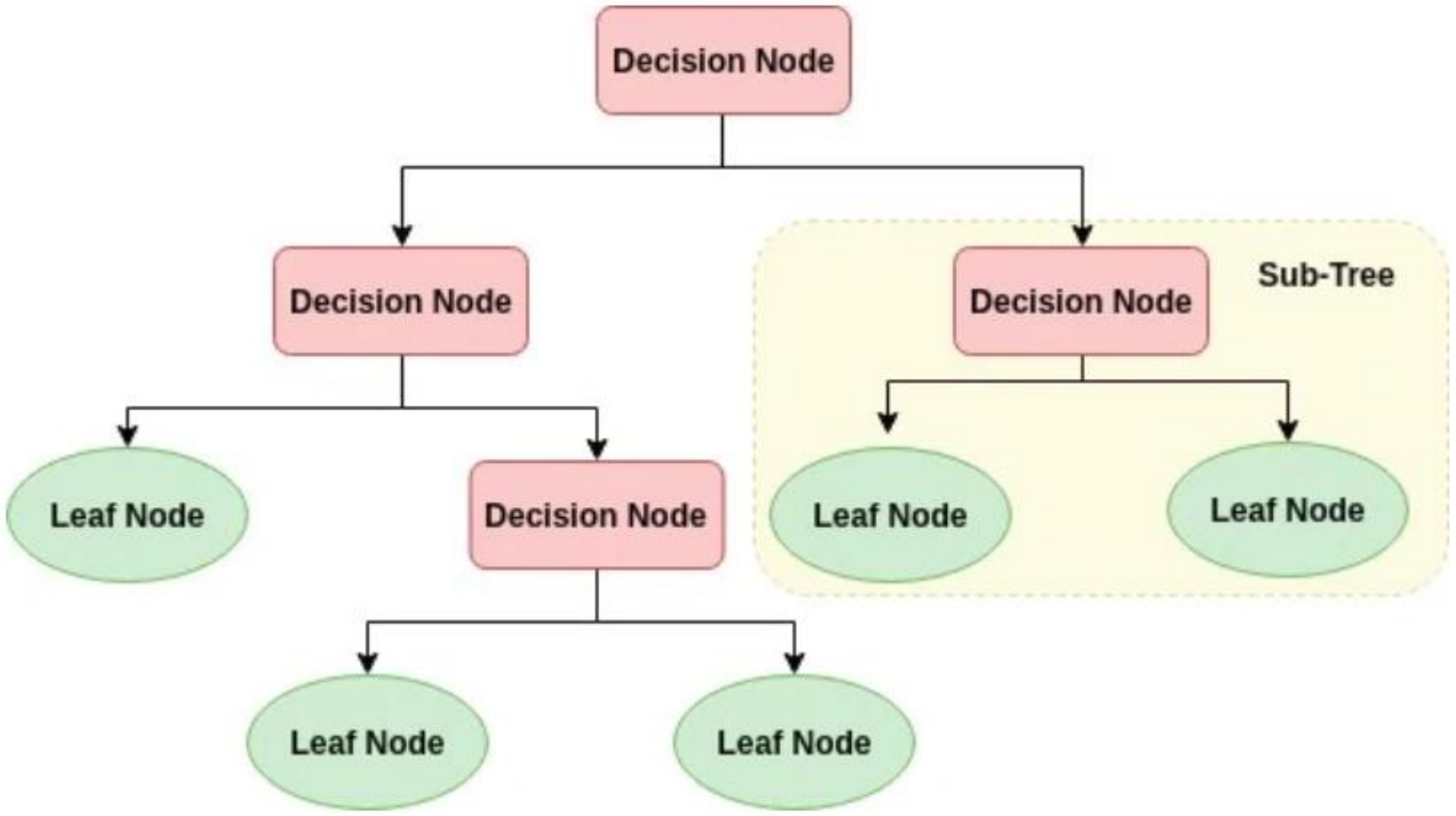
What is a Decision Tree?

- A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.
- The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.
- The tree is partitioned in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making.
- The tree provides visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

Decision Tree Elements

A decision tree consists of three types of nodes:

- Decision nodes – typically represented by squares
- Chance nodes – typically represented by circles
- End nodes – typically represented by triangles



Decision Tree Types (CART – classification & regression trees)

Decision tree has two main types:

- **Classification tree** analysis is when the predicted outcome is the class (discrete) to which the data belongs.

- **Regression tree** analysis is when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).

Decision Tree Rules

- Intuitively, we can also think of decision tree as nested "if-else" rules. And a rule is simply a conjunction of conditions. For example,

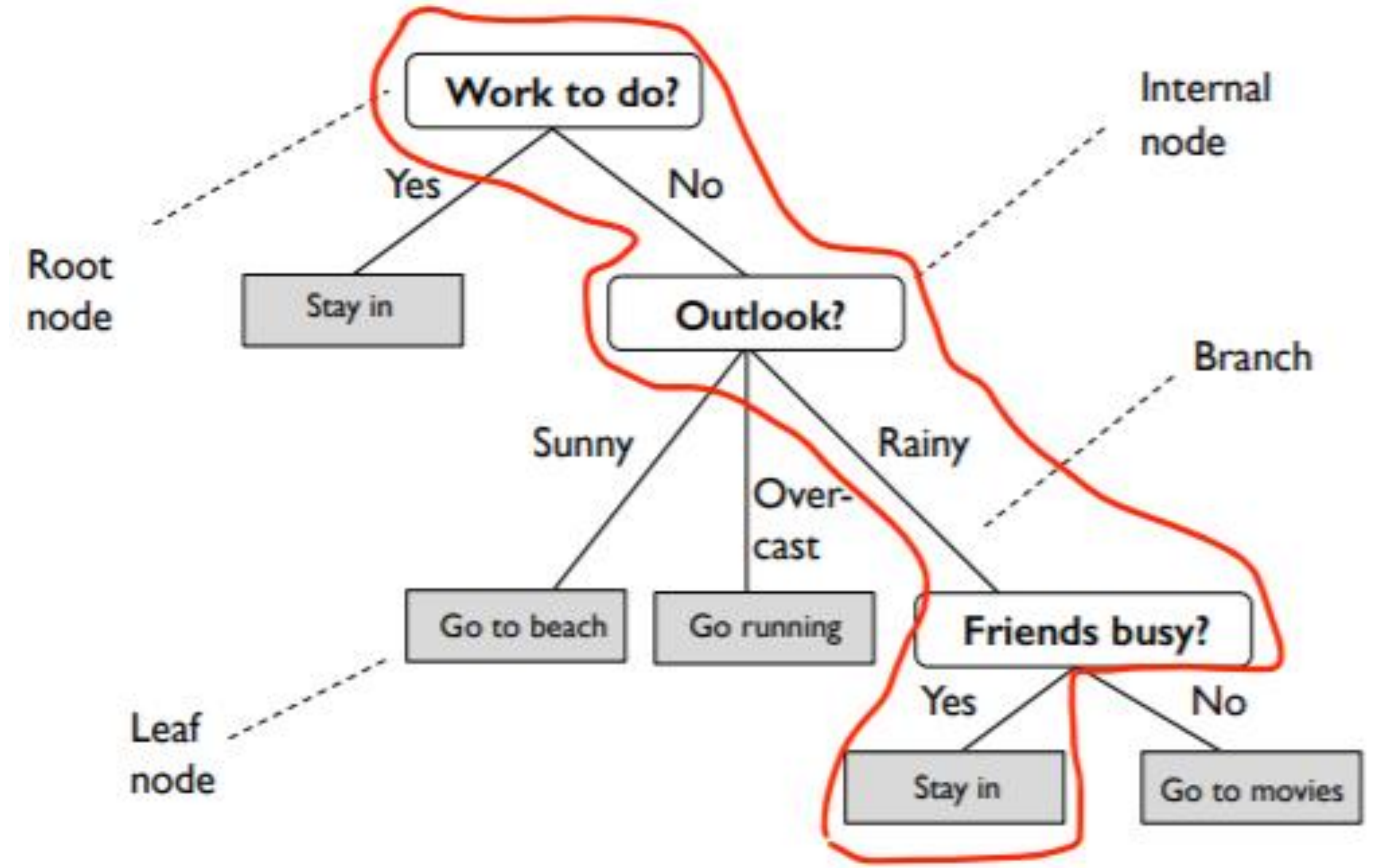
$$\text{Rule 1} = (\text{if } x = 1) \cap (\text{if } y = 2) \cap \ldots$$

- Multiple rules can then be joined into a set of rules, which can be applied to predict the target value of a training example or test instance. For example,

$$\text{Class 1} = \text{if } (\text{Rule 1=True}) \cup (\text{Rule 2=True}) \cup \ldots$$

- Each leaf node in a decision tree represents such a set of rules

Decision Tree Rules



A rule for a given leaf node (circled): (Work to do? = False) ∩ (Outlook? = Rainy) ∩ (Friends busy? = Yes)

Decision Tree Processes

Splitting

- The process of partitioning the data set into subsets.
- The dataset is split into two or more groups so that all the observations in the same group are most similar to each other (homogeneity) and the groups are significantly different from each other (heterogeneity).

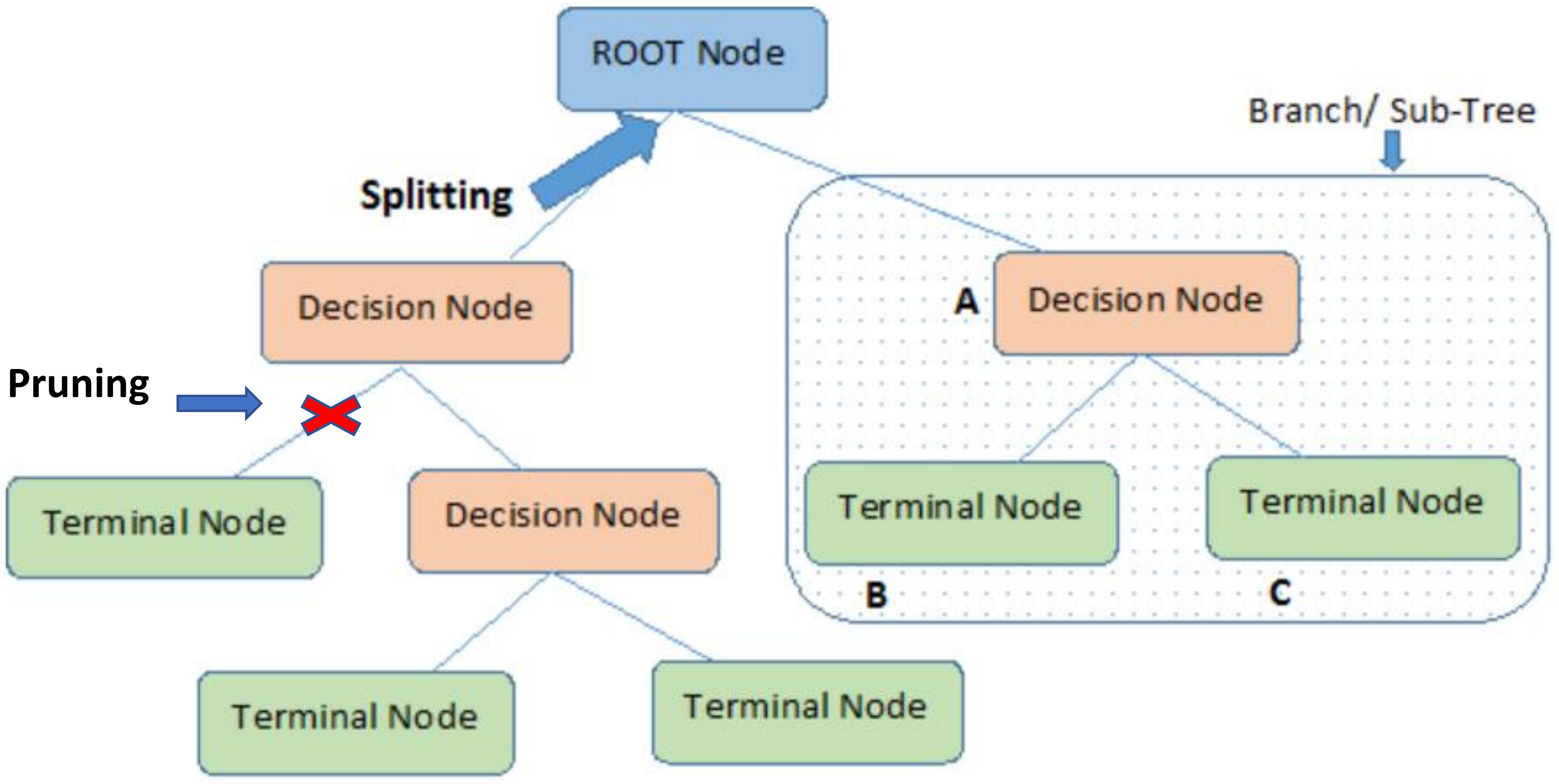
Pruning

- The process of reducing the size of the tree by removing sub-nodes of a decision node

Tree Selection

- The process of finding the smallest tree that fits the data. Usually this is the tree that yields the lowest error

Decision Tree Processes



Note:- A is parent node of B and C.

Sample Dataset

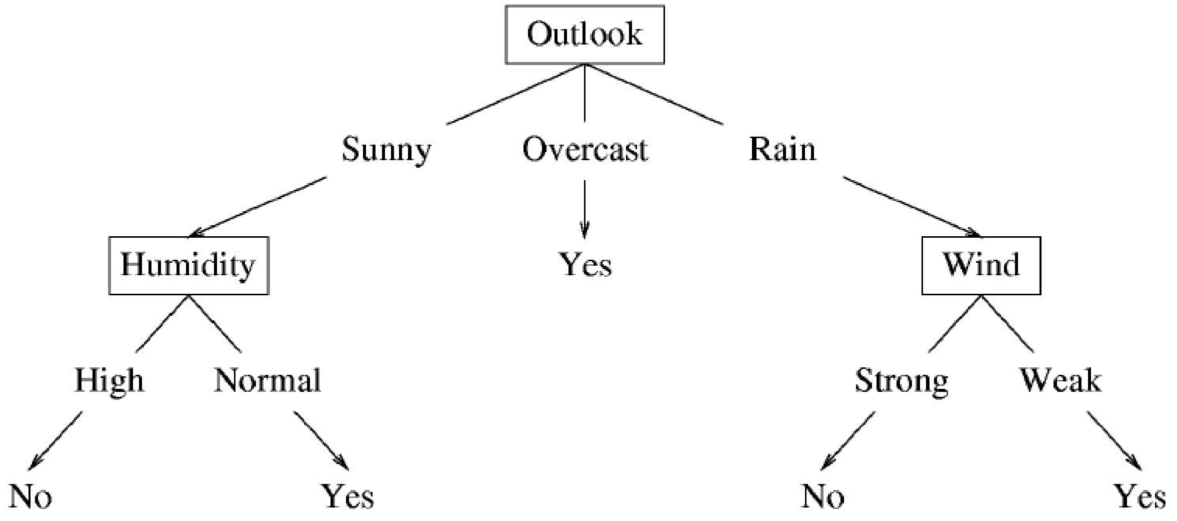
- Columns denote features X_i
- Rows denote labeled instances $\langle \boldsymbol{x}_i, y_i \rangle$
- Class label denotes whether a tennis game was played

$\langle \boldsymbol{x}_i, y_i \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision Tree

- A possible decision tree for the data:

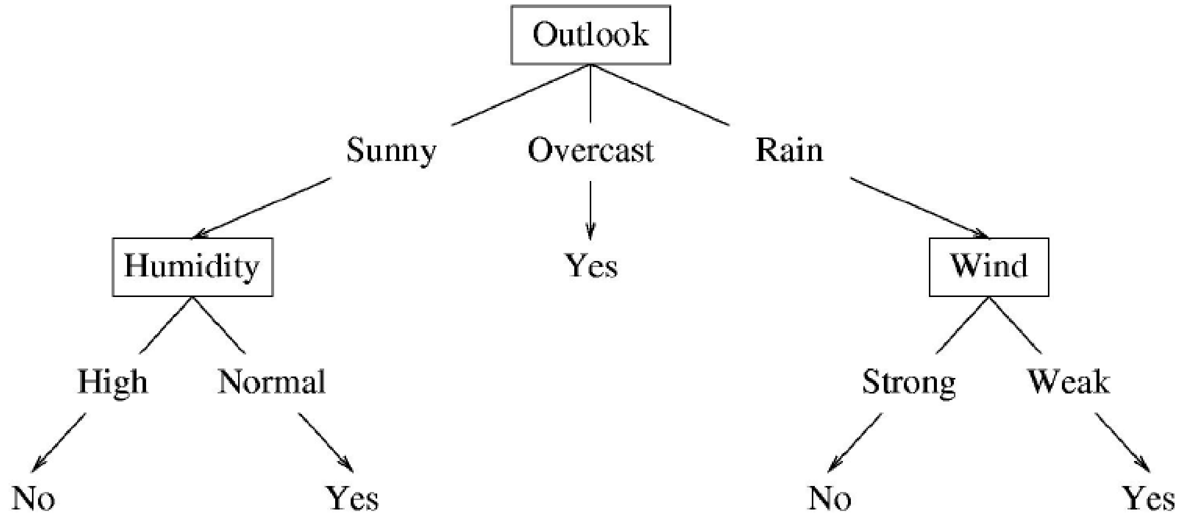


- Each internal node: test one attribute X_i
- Each branch from a node: selects one value for X_i
- Each leaf node: predict Y (or $p(Y \mid \boldsymbol{x} \in \text{leaf})$)

Based on slide by Tom Mitchell

Decision Tree

- A possible decision tree for the data:

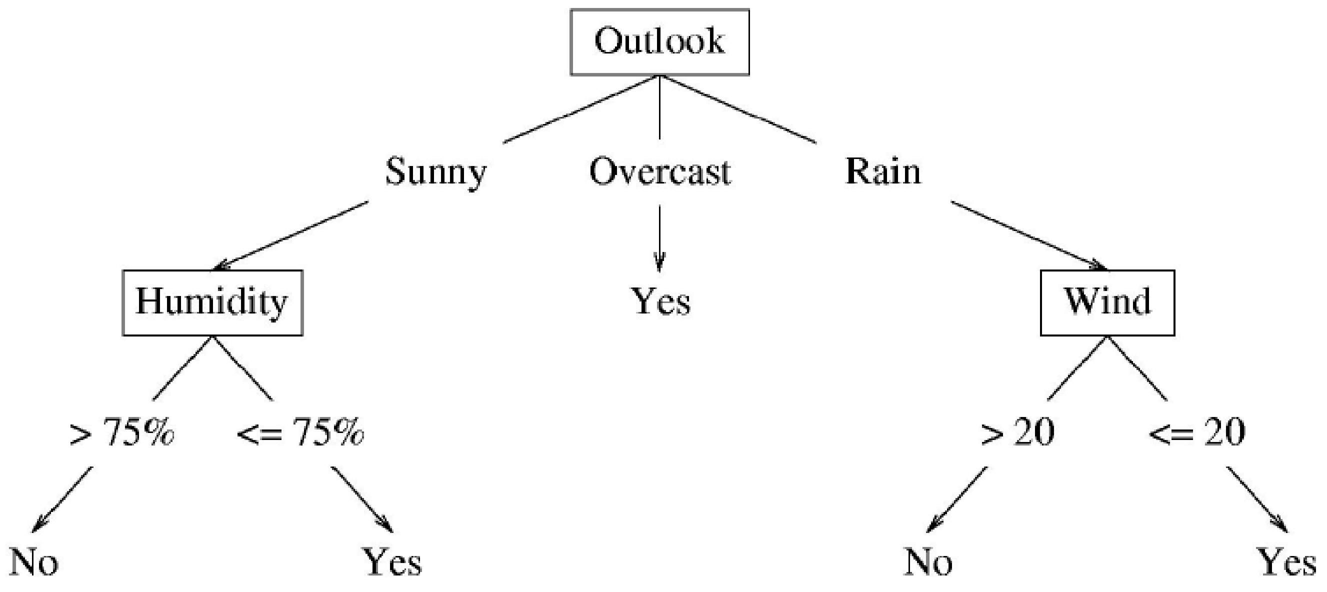


- What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ?

Based on slide by Tom Mitchell

Decision Tree

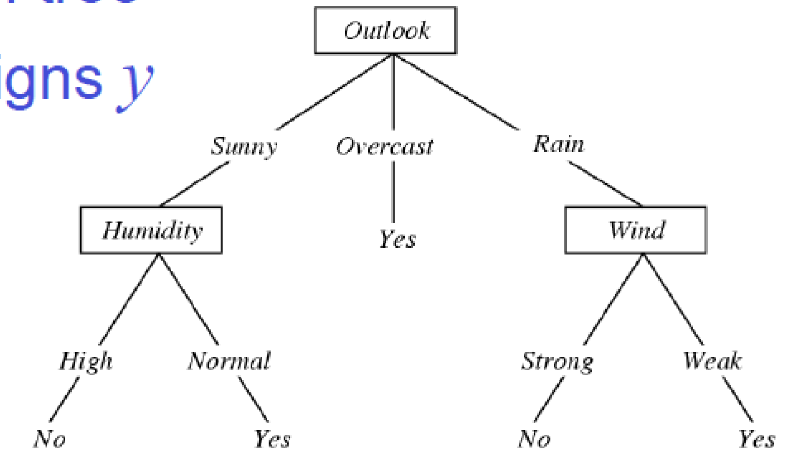
- If features are continuous, internal nodes can test the value of a feature against a threshold



Decision Tree Learning

Problem Setting:

- Set of possible instances X
  - each instance x in X is a feature vector
  - e.g., *<Humidity=low, Wind=weak, Outlook=rain, Temp=hot>*
- Unknown target function $f : X \rightarrow Y$
  - Y is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis h is a decision tree
  - trees sorts x to leaf, which assigns y



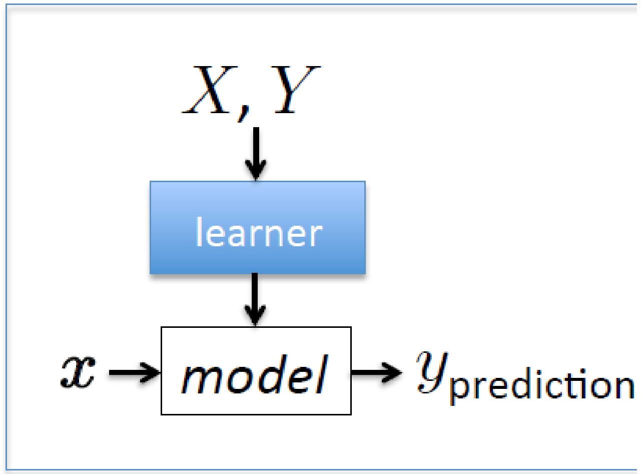
Stages: (batch based)

Given: labeled training data $X, Y = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^n$

- Assumes each $\boldsymbol{x}_i \sim \mathcal{D}(\mathcal{X})$ with $y_i = f_{target}(\boldsymbol{x}_i)$

Train the model:

model ← *classifier*.train(X, Y)

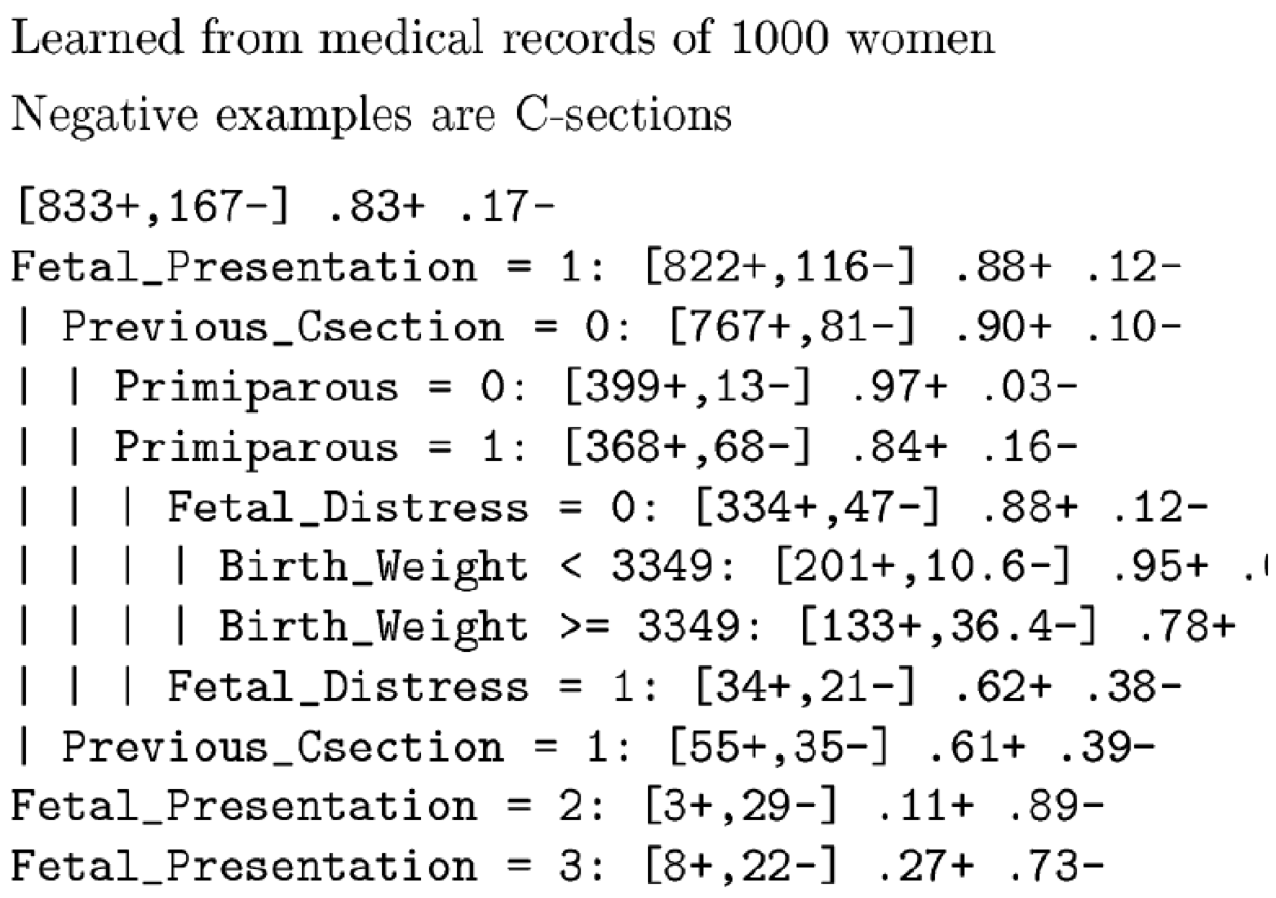


Apply the model to new data:

- Given: new unlabeled instance $x \sim \mathcal{D}(\mathcal{X})$

$y_{\text{prediction}}$ ← *model*.predict(x)

Example Application: A Tree to Predict Caesarean Section Risk



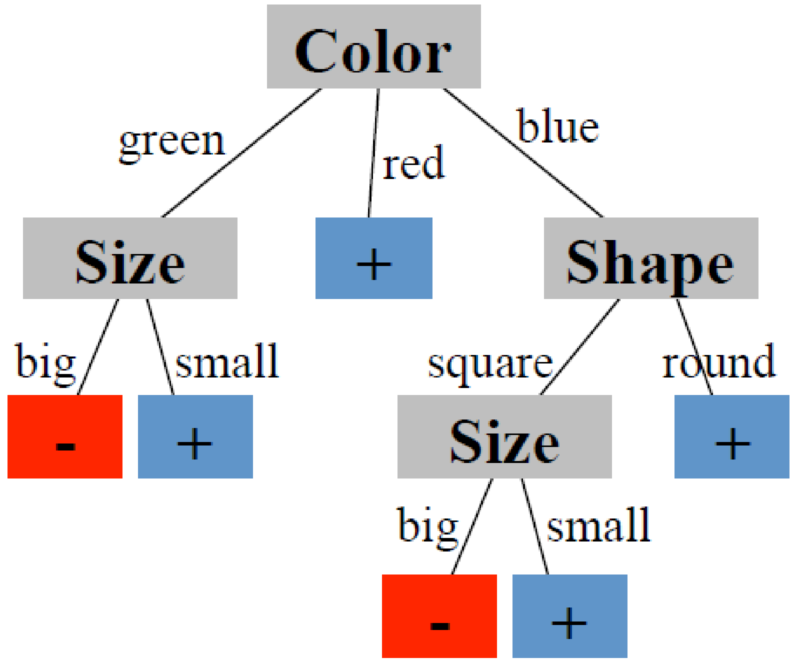
Learned from medical records of 1000 women

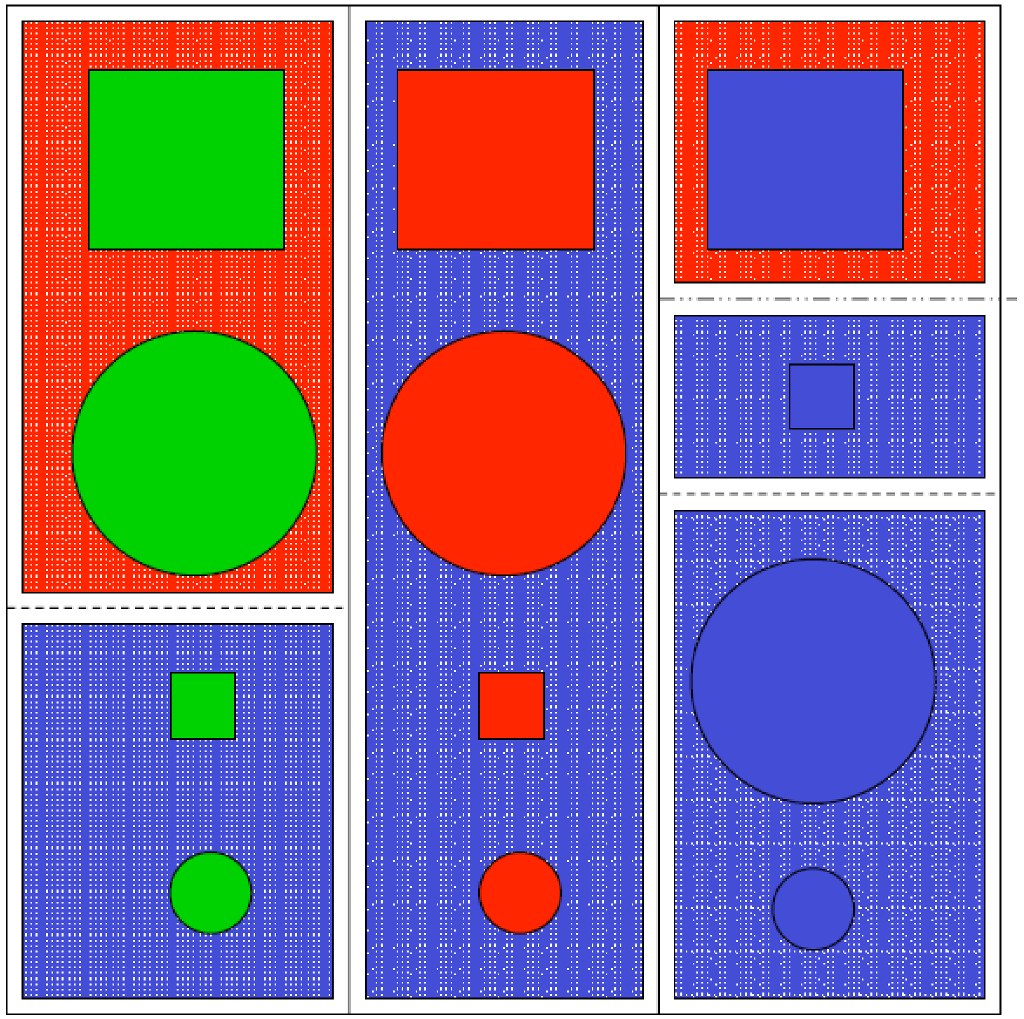
Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

Based on Example by Tom Mitchell

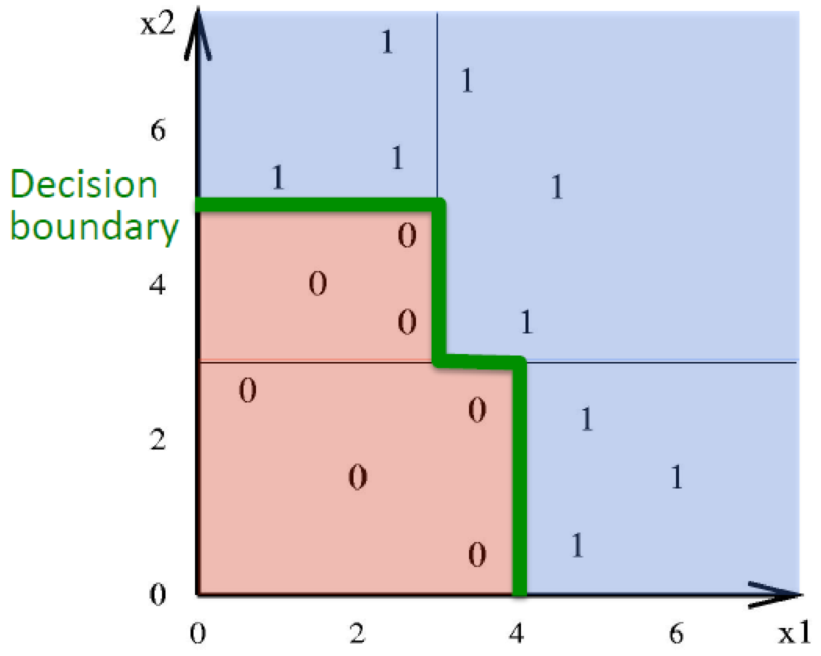
Decision Tree Induced Partition

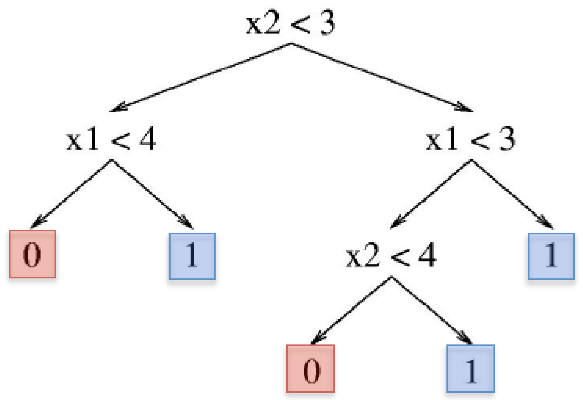




Decision Tree – Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label
  - or a probability distribution over labels



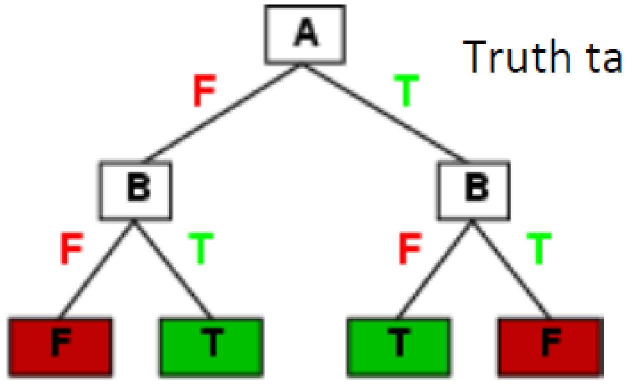


11

Expressiveness

- Decision trees can represent any boolean function of the input attributes

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Truth table row $\rightarrow$ path to leaf

- In the worst case, the tree will require exponentially many nodes

Preference bias: Ockham's Razor

- Principle stated by William of Ockham (1285-1347)
  - "*non sunt multiplicanda entia praeter necessitatem* "
  - entities are not to be multiplied beyond necessity
  - AKA Occam's Razor, Law of Economy, or Law of Parsimony

Idea: The simplest consistent explanation is the best

- Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  - Finding the provably smallest decision tree is NP-hard
  - ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

Basic Algorithm for Top-Down Induction of Decision Trees

[ID3, C4.5 by Quinlan]

```
node = root of decision tree
Main loop:
1.  A ← the "best" decision attribute for the next node.
2.  Assign A as decision attribute for node.
3.  For each value of A, create a new descendant of node.
4.  Sort training examples to leaf nodes.
5.  If training examples are perfectly classified, stop.
    Else, recurse over new leaf nodes.
```

How do we choose which attribute is best?

Choosing the Best Attribute

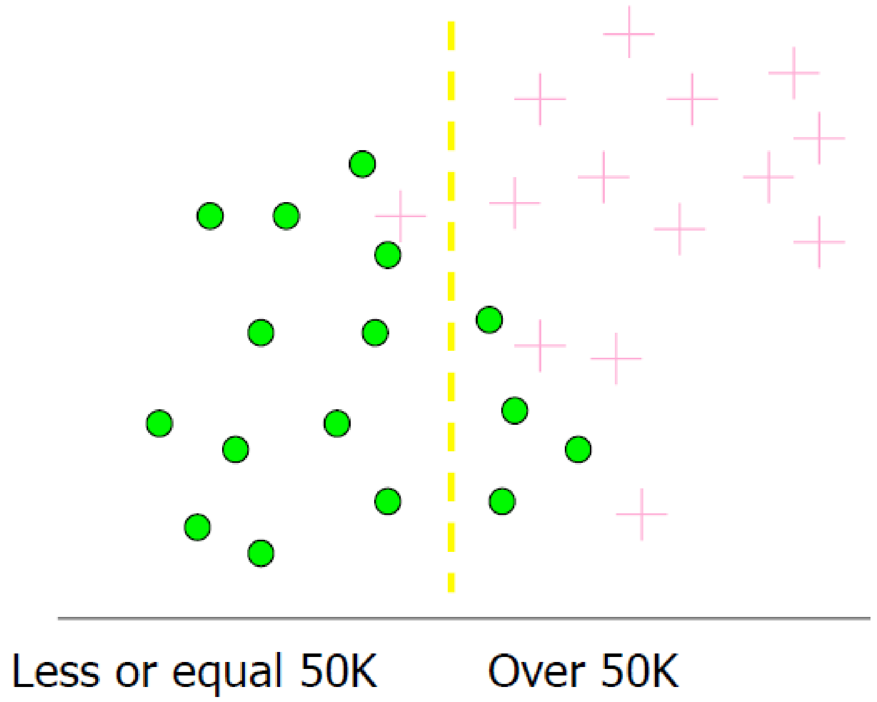
Key problem: choosing which attribute to split a given set of examples

- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values
  - **Most-Values:** Choose the attribute with the largest number of possible values
  - **Max-Gain:** Choose the attribute that has the largest expected *information gain*
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

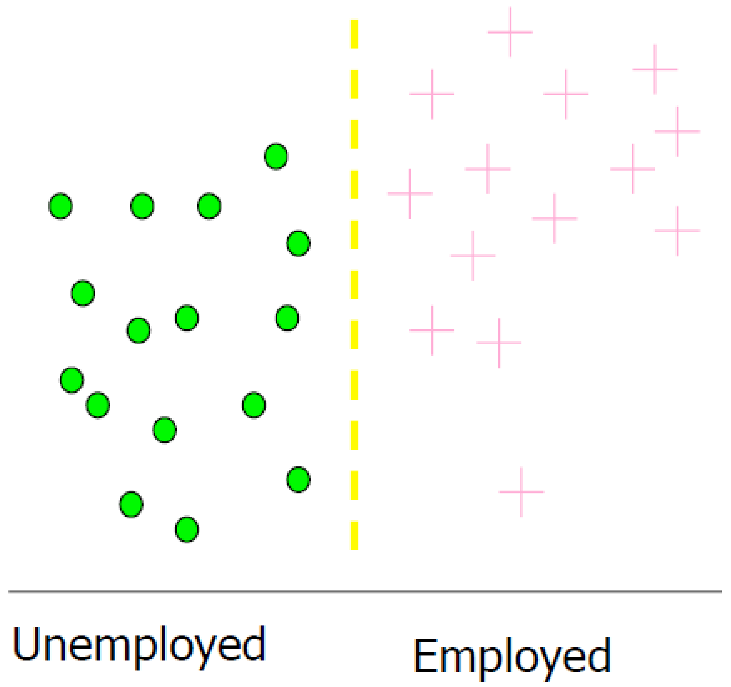
Information Gain

Which test is more informative?

Split over whether Balance exceeds 50K



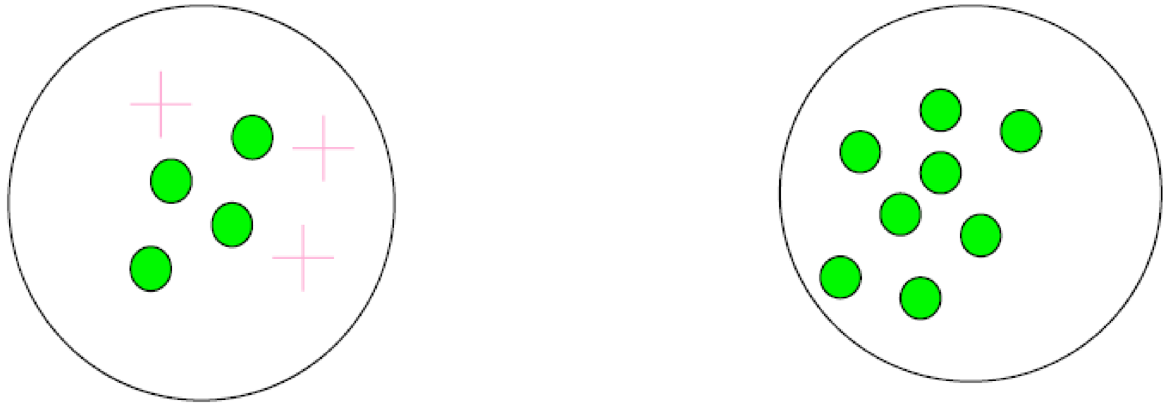
Split over whether applicant is employed



Information Gain

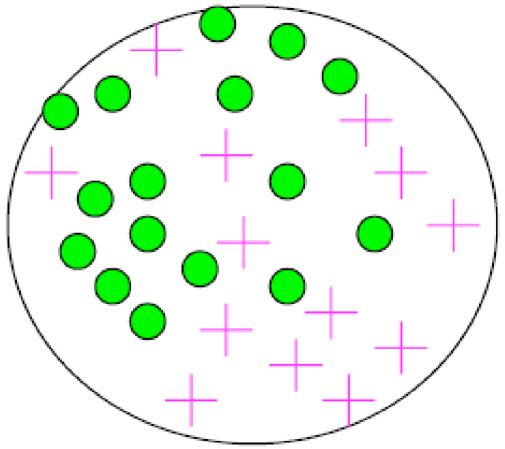
Impurity/Entropy (informal)

- Measures the level of **impurity** in a group of examples

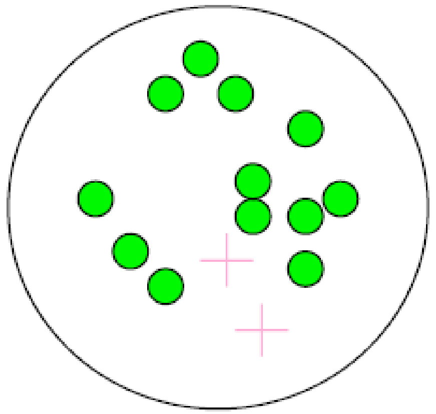


Impurity

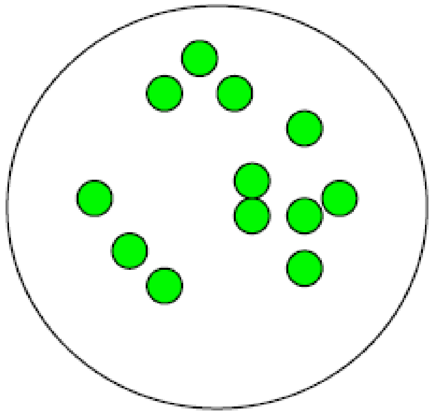
Very impure group



Less impure



Minimum impurity



Entropy – a way to measure impurity

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

# of possible values for X

$$\boxed{H(X) = -\sum_{i=1}^{n} P(X=i) \log_2 P(X=i)}$$

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

Why? Information theory:

- Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message $X=i$
- So, expected number of bits to code one random X is:

$$\sum_{i=1}^{n} P(X=i)(-\log_2 P(X=i))$$

Slide by Tom Mitchell

Entropy & Information Gain – for ID3 (C4.5)

- We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
- Information gain tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Information Gain = entropy(parent) – [average entropy(children)]

How well does it work?

ID3, C4.5 DT Classifiers

Many case studies have shown that decision trees are at least as accurate as human experts.

- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

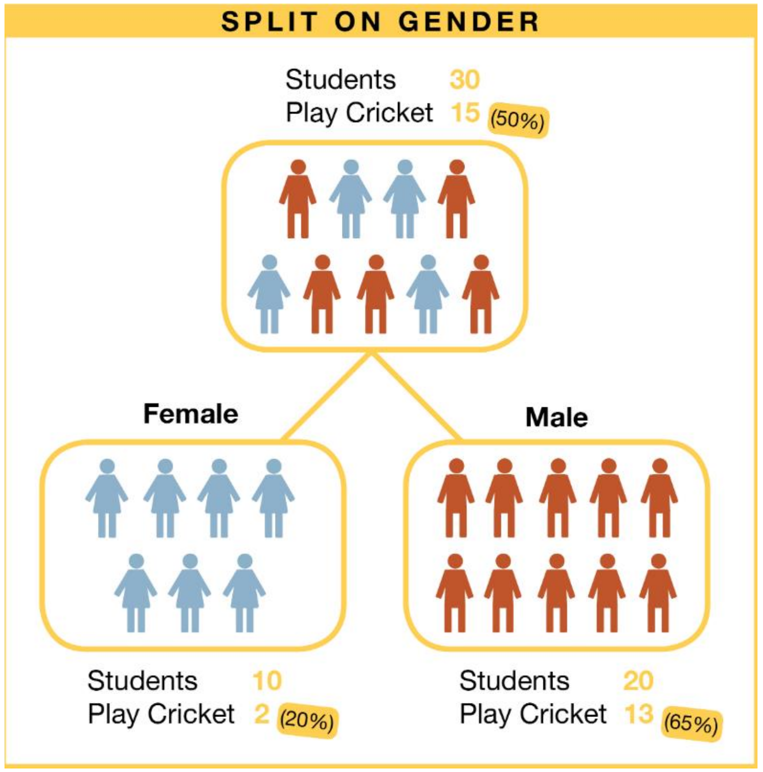
Based on Slide from M. desJardins & T. Finin

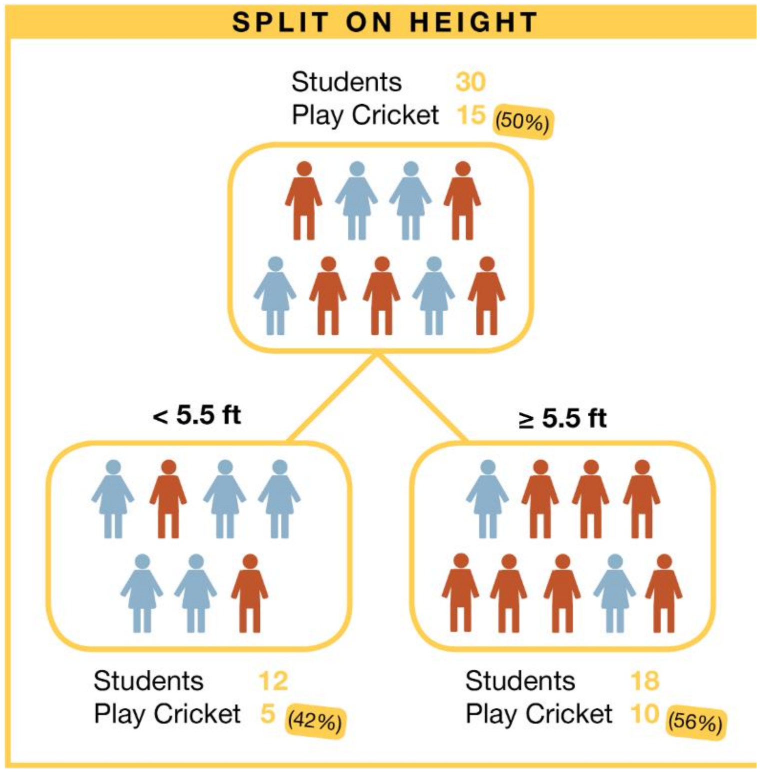
Splitting Example

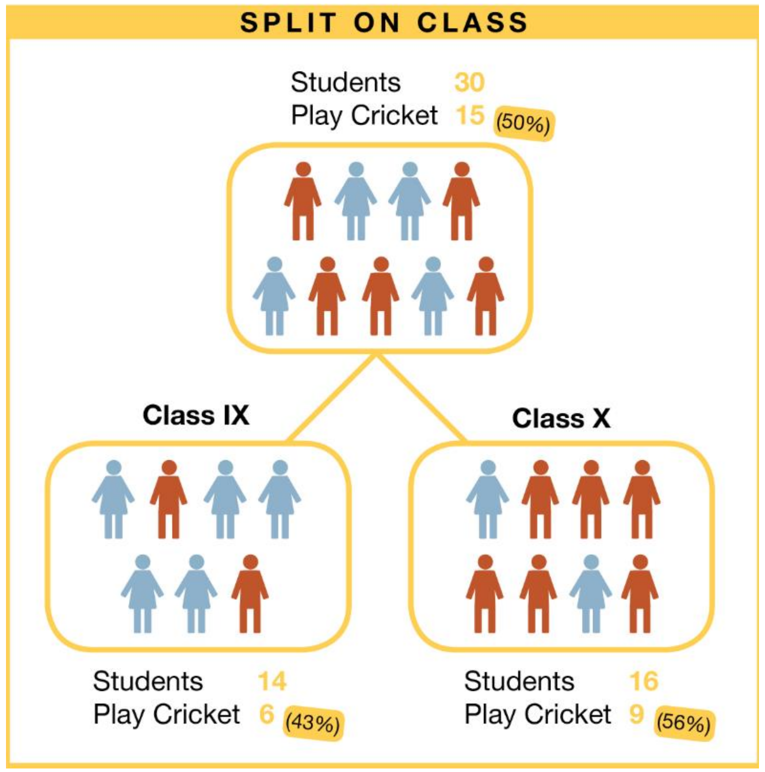
A class of 30 students. Three attributes:

- Gender
- Height
- Class

A predictive model





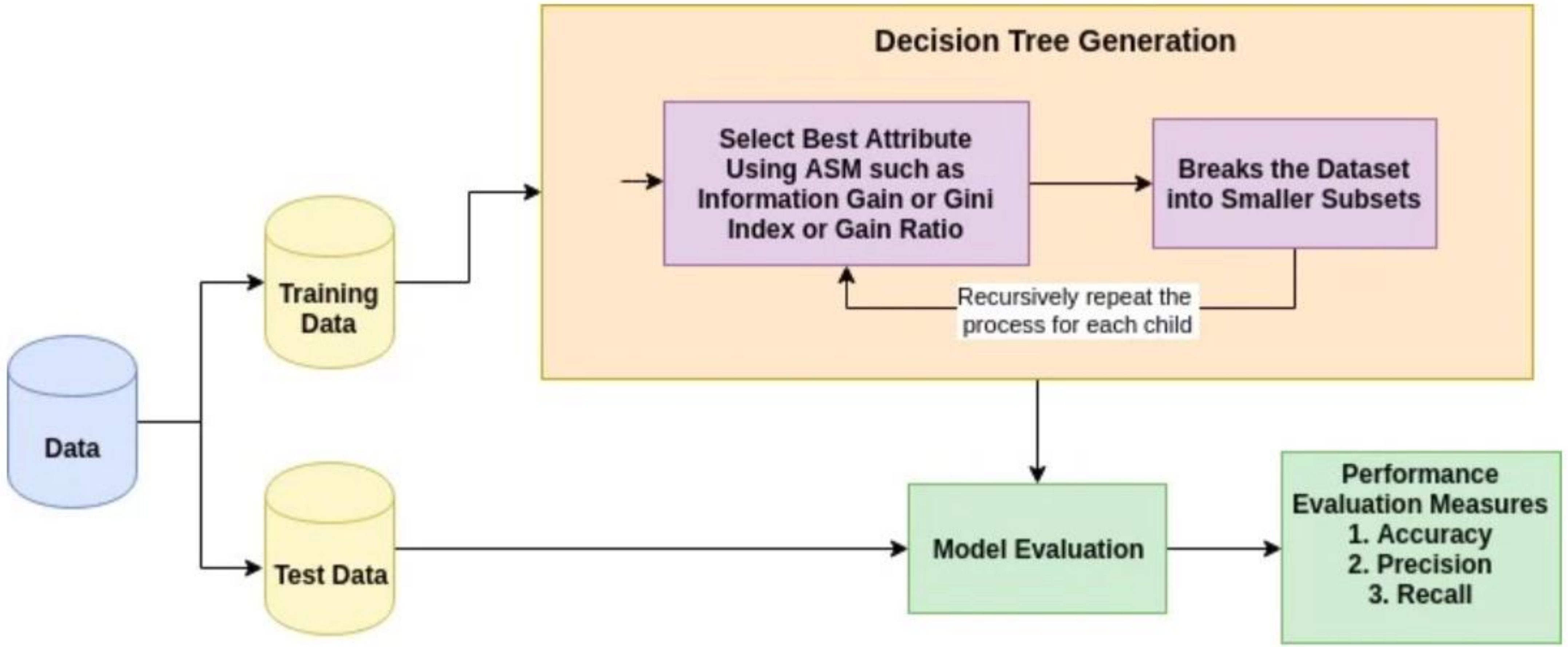


How does a Decision Tree work?

The basic idea behind any decision tree algorithm is as follows:

- Select the best attribute using Attribute Selection Measures (ASM) to split the records.
- Make that attribute a decision node and breaks the dataset into smaller subsets.
- Starts tree building by repeating this process recursively for each child until one of the condition will match:
    - All the tuples belong to the same attribute value.
    - There are no more remaining attributes.
    - There are no more instances.

DT process


Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Attribute Selection Measures

- Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner.
- It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node.
- ASM provides a rank to each feature(or attribute) by explaining the given dataset.
- Best score attribute will be selected as a splitting attribute
- Most popular selection measures currently are Information Gain, Gain Ratio, and Gini Index.

Information Gain

- Shannon invented the concept of entropy, which measures the impurity of the input set
- Information gain is the decrease in entropy.
- Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.
- The first decision tree (ID3) algorithm uses information gain.

Information Gain

Information gain from X to Y is given by

$$IG(Y, X) = E(Y) - E(Y|X)$$

Where an entropy E is computed by

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

With p_i is simply the frequency probability of an element/class *i* in the data.

An example

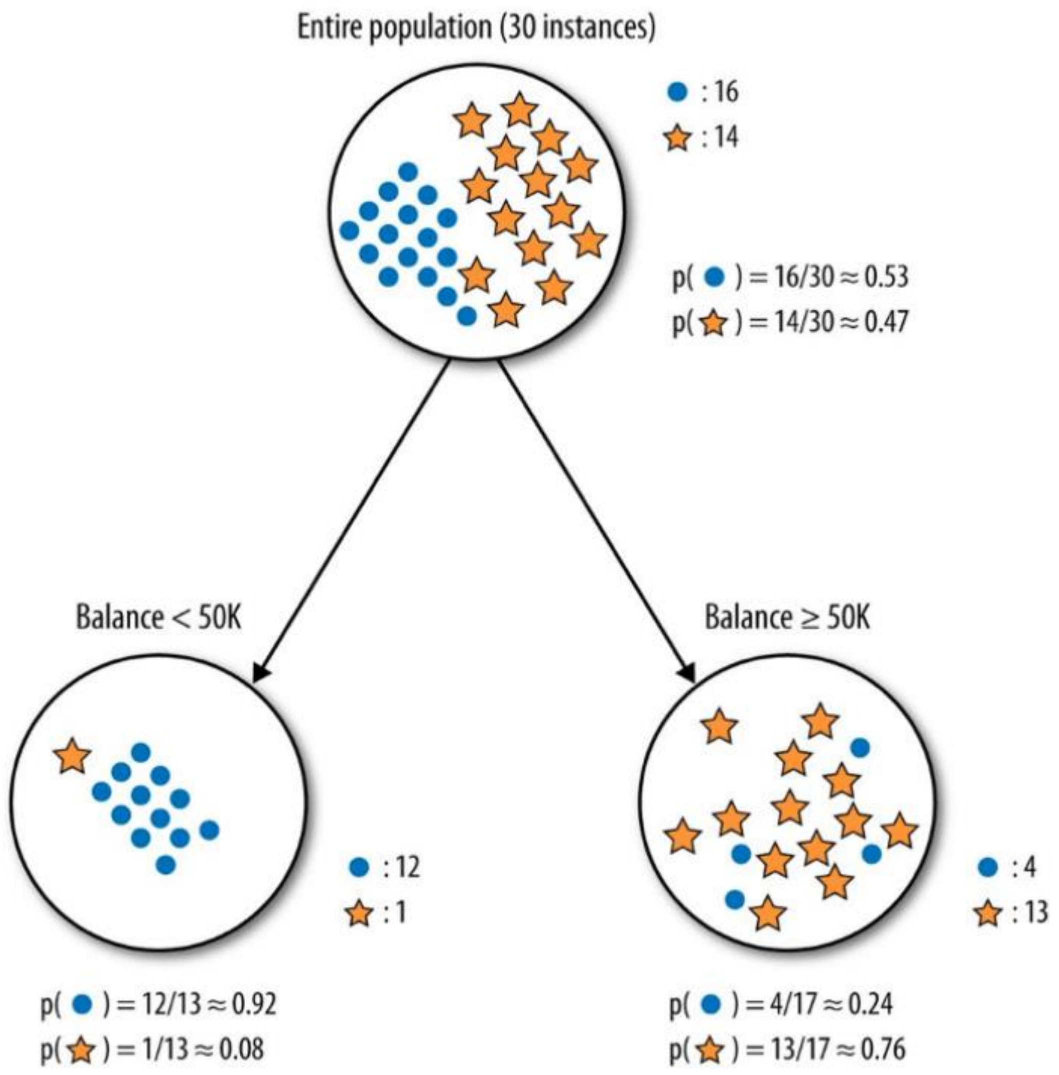
Consider an example where we are building a decision tree to predict whether a loan given to a person would result in a write-off or not.

Our entire population consists of 30 instances. 16 belong to the write-off class and the other 14 belong to the non-write-off class.

We have two features

- "Balance" that can take on two values -> "< 50K" or ">50K"
- "Residence" that can take on three values -> "OWN", "RENT" or "OTHER".

Balance Feature



$$E(Parent) = -\frac{16}{30}\log_2\left(\frac{16}{30}\right) - \frac{14}{30}\log_2\left(\frac{14}{30}\right) \approx 0.99$$

$$E(Balance < 50K) = -\frac{12}{13}\log_2\left(\frac{12}{13}\right) - \frac{1}{13}\log_2\left(\frac{1}{13}\right) \approx 0.39$$

$$E(Balance > 50K) = -\frac{4}{17}\log_2\left(\frac{4}{17}\right) - \frac{13}{17}\log_2\left(\frac{13}{17}\right) \approx 0.79$$

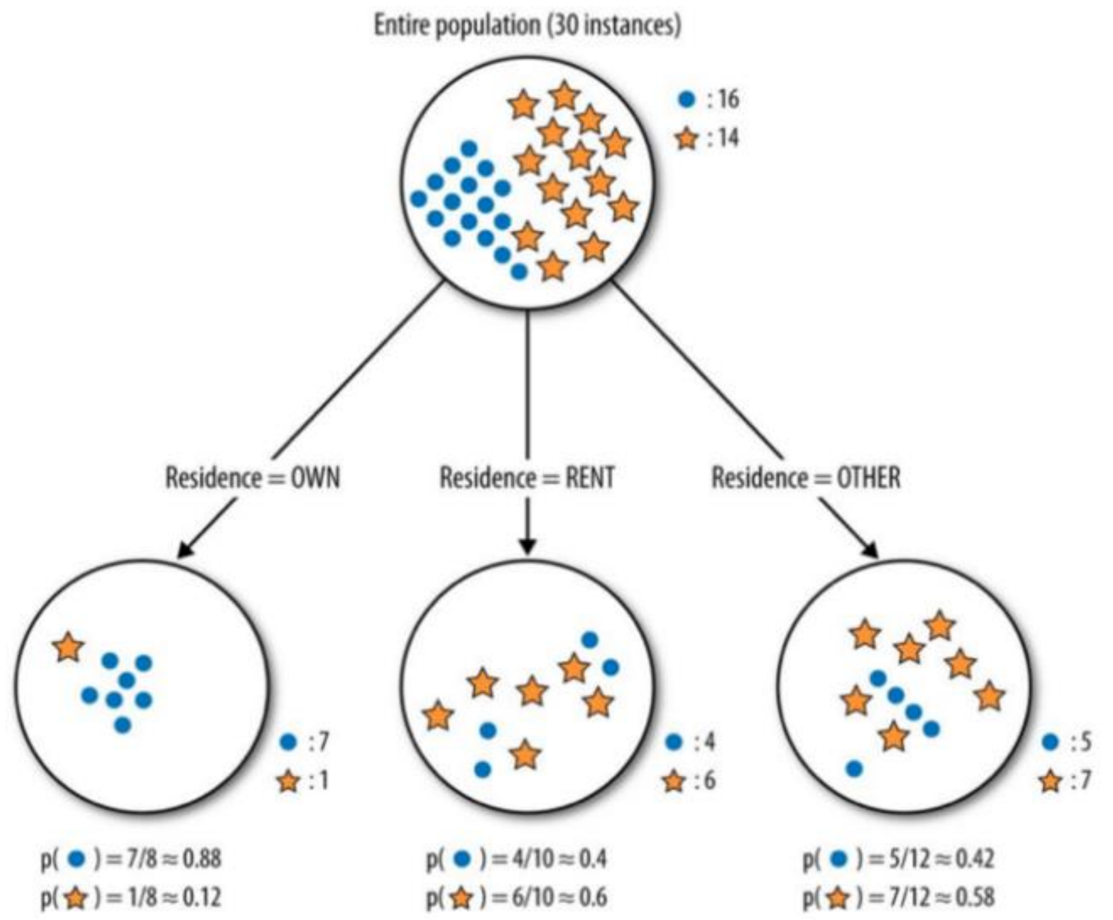
Weighted Average of entropy for each node:

$$E(Balance) = \frac{13}{30}\times 0.39 + \frac{17}{30}\times 0.79$$
$$= 0.62$$

Information Gain:

$$IG(Parent, Balance) = E(Parent) - E(Balance)$$
$$= 0.99 - 0.62$$
$$= 0.37$$

Residence Feature



$$E(Residence = OWN) = -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

$$E(Residence = RENT) = -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \approx 0.97$$

$$E(Residence = OTHER) = -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \approx 0.98$$

Weighted Average of entropies for each node:

$$E(Residence) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

Information Gain:

$$\begin{aligned} IG(Parent, Residence) &= E(Parent) - E(Residence) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

Balance provides more information about our target variable than Residence: 0.37 > 0.13

Gini Index - CART

- A decision tree algorithm CART uses the Gini method to create split points.
- Gini index works with categorical target variable “Success” or “Failure”.
- It performs only Binary splits
- Higher the value of Gini provides higher the homogeneity.

Steps to Calculate Gini for a split

- Calculate Gini for sub-nodes, using formula sum of square of probability for success and failure (p^2+q^2).
- Calculate Gini for split using weighted Gini score of each node of that split

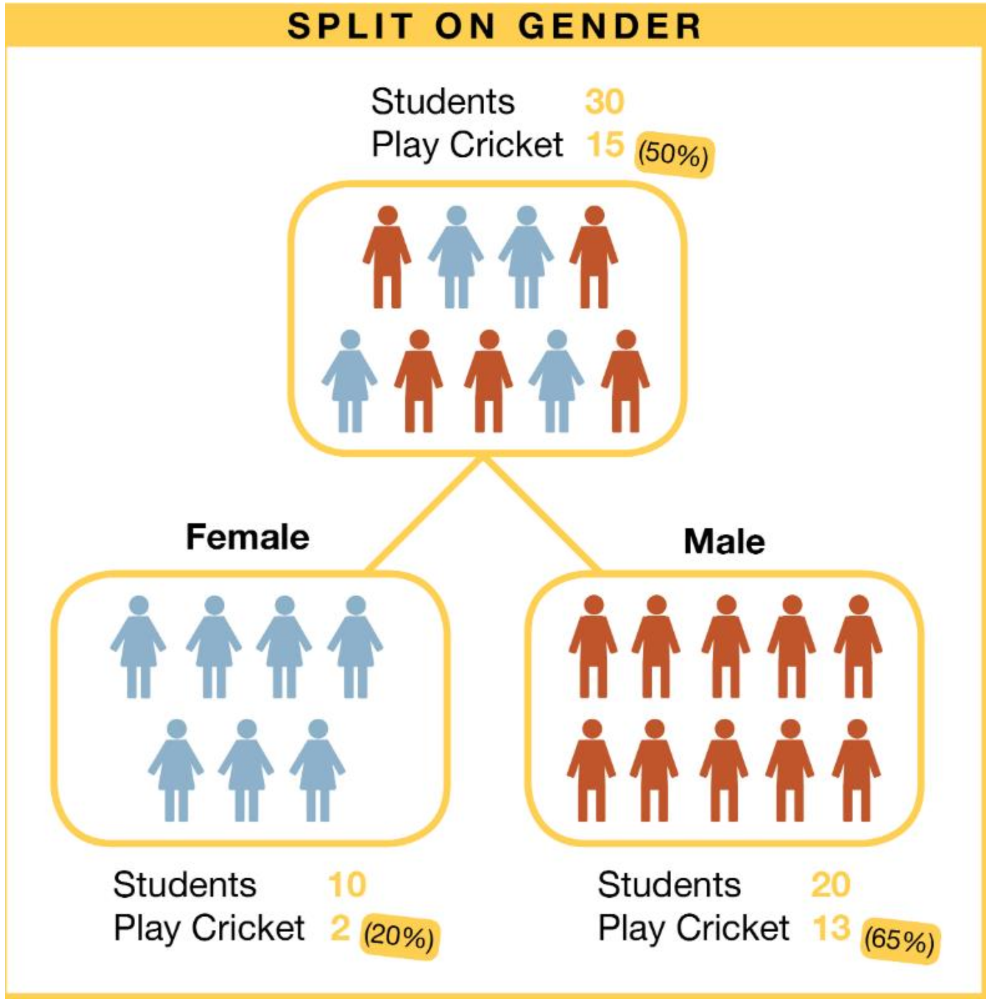
Properties of Gini index

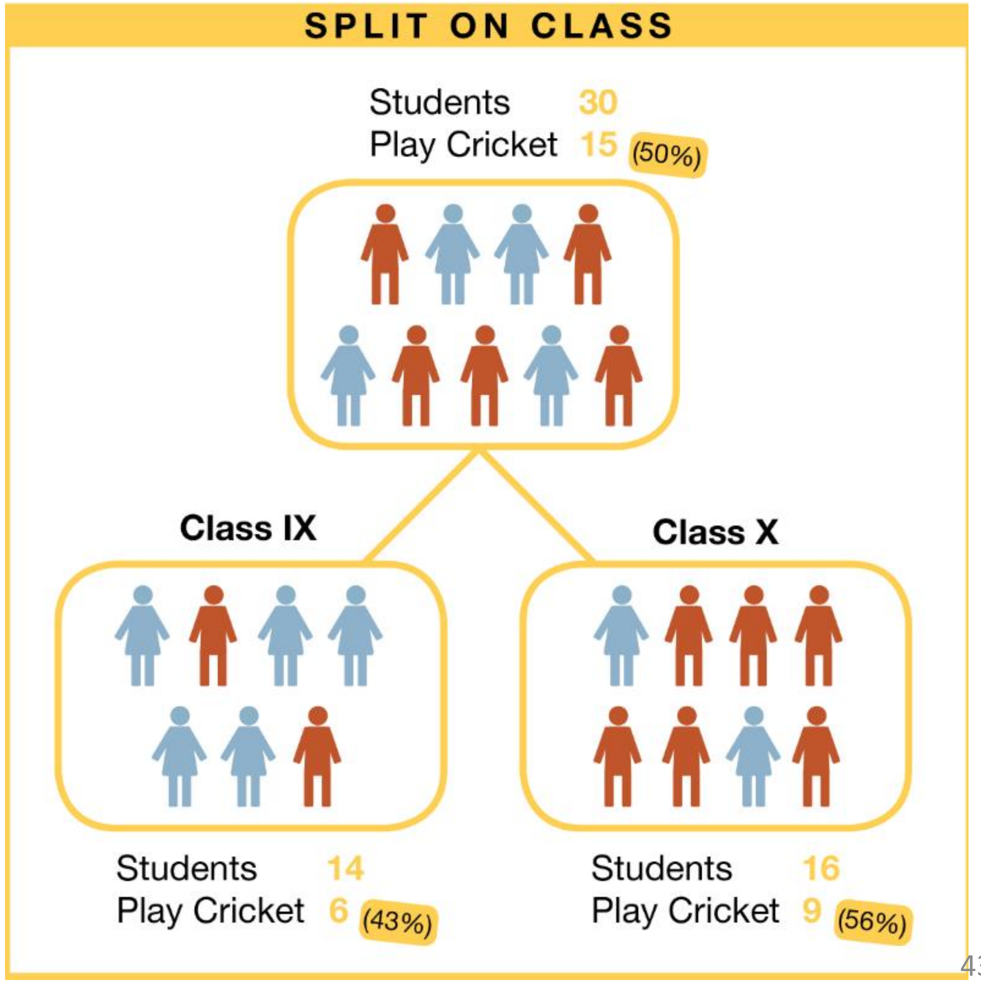
- The **Gini index varies between values 0 and 1**,
- 0 expresses the purity of classification, i.e. all the elements belong to a specified class or only one class exists there.
- 1 indicates the random distribution of elements across various classes.
- The value of 0.5 of the Gini Index shows an equal distribution of elements over some classes.

An Example

A class of 30 students. Three attributes:

- Gender
- Class





Example (cont)

Split on Gender:

1. Gini for sub-node Female = (0.2)*(0.2)+(0.8)*(0.8)=0.68
2. Gini for sub-node Male = (0.65)*(0.65)+(0.35)*(0.35)=0.55
3. Weighted Gini for Split Gender = (10/30)*0.68+(20/30)*0.55 = **0.59**

Similar for Split on Class:

1. Gini for sub-node Class IX = (0.43)*(0.43)+(0.57)*(0.57)=0.51
2. Gini for sub-node Class X = (0.56)*(0.56)+(0.44)*(0.44)=0.51
3. Weighted Gini for Split Class = (14/30)*0.51+(16/30)*0.51 = **0.51**

Gini score for *Split on Gender* is higher than *Split on Class,*

So, the dataset split will take place on Gender.

Gini Index vs Information Gain

- The Gini Index facilitates the bigger distributions so easy to implement whereas the Information Gain favors lesser distributions having small count with multiple specific values.

- The method of the Gini Index is used by CART algorithms, in contrast - Information Gain is used in ID3.

- Gini index operates on the categorical target variables in terms of “success” or “failure” and performs only binary split, in opposite to that Information Gain computes the difference between entropy before and after the split and indicates the impurity in classes of elements.

Optimizing Decision Tree Performance

Parameters used in Decision Tree with Sklearn

- criterion : optional (default=”gini”) or Choose attribute selection measure: This parameter allows us to use the different-different attribute selection measure. Supported criteria are “gini” for the Gini index and “entropy” for the information gain.

- splitter : string, optional (default=”best”) or Split Strategy: This parameter allows us to choose the split strategy. Supported strategies are “best” to choose the best split and “random” to choose the best random split.

- max_depth : int or None, optional (default=None) or Maximum Depth of a Tree: The maximum depth of the tree. If None, then nodes are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting.

Computational Complexity

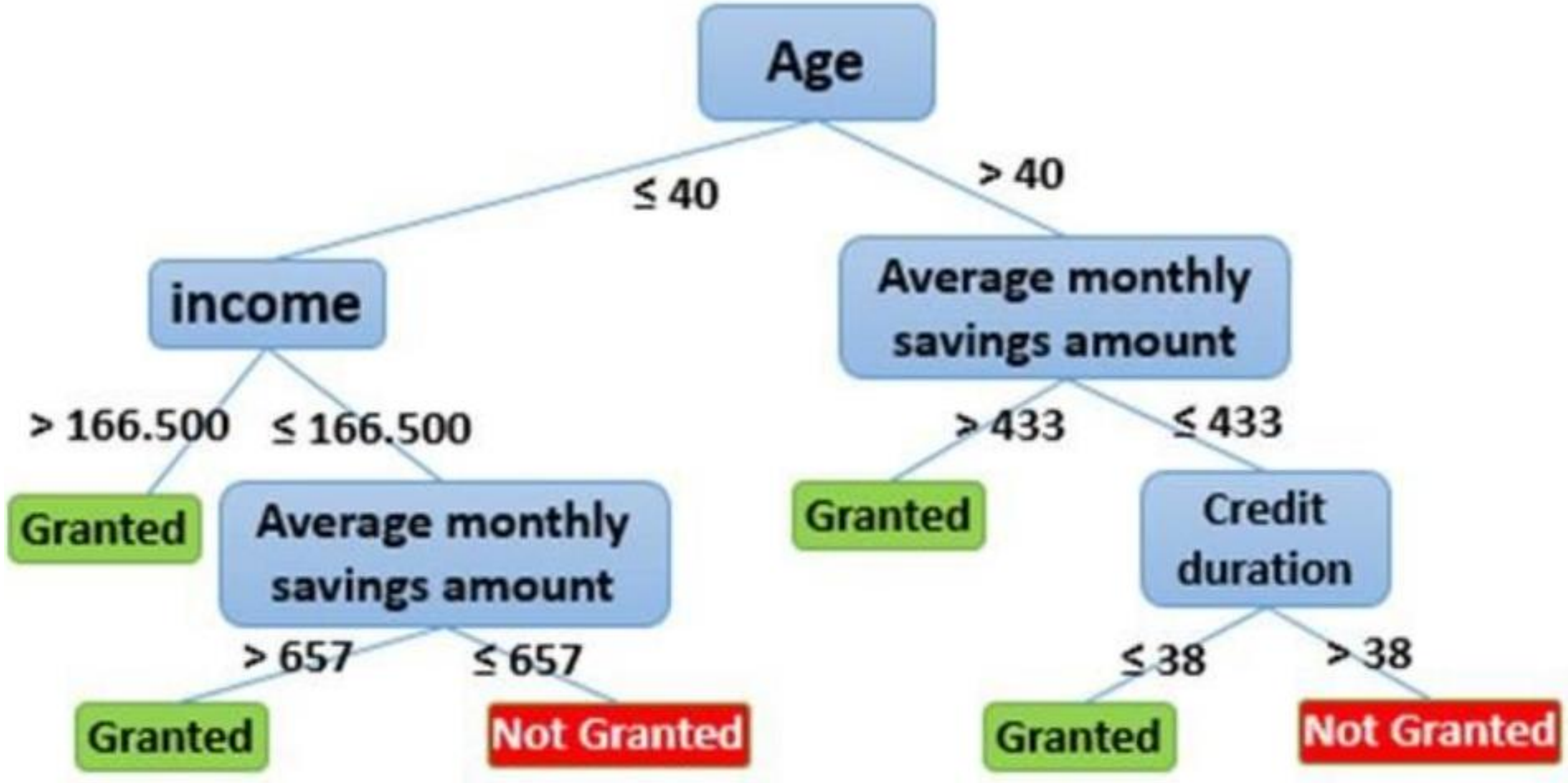
Assume *n* is numbers of instances and *m* is numbers of features

Complexity: $mn^2 log(n)$

Applications of Decision Tree

- Biomedical Engineering (decision trees for identifying features to be used in implantable devices)
- Financial analysis (Customer Satisfaction with a product or service).
- Astronomy (classify galaxies)
- System Control
- Manufacturing and Production (Quality control, Semiconductor manufacturing, etc)
- Medicine (diagnosis, cardiology, psychiatry)
- Physics (Particle detection)

Application in Finance: A case of Bank



Advantages of Decision Tree

- Decision trees are easy to interpret and visualize.
- It can easily capture Non-linear patterns.
- It requires fewer data preprocessing from the user, for example, there is no need to normalize columns.
- It can be used for feature engineering such as predicting missing values, suitable for variable selection.
- The decision tree has no assumptions about distribution because of the non-parametric nature of the algorithm.

Disadvantages

- Sensitive to noisy data. It can overfit noisy data.
- The small variation (or variance) in data can result in the different decision tree. This can be reduced by *bagging* and *boosting* algorithms.
- Decision trees are biased with imbalance dataset, so it is recommended that the dataset is balanced out before creating the decision tree.
- Large trees can be difficult to interpret and the decisions they make may seem counter intuitive.