# Data Mining Tutorial

Mark A. Austin

University of Maryland

*austin@umd.edu*
*ENCE 688P, Spring Semester 2026*

February 13, 2026

# Overview

# Quick Review

# Artificial Intelligence (AI) and Machine Learning (ML)

Technical Implementation (2020, Google, Siemens, IBM)

- AI and ML will be deeply embedded in new software and algorithms.

Artificial Intelligence:

- Knowledge representation and reasoning with ontologies and rules. Semantic graphs. Executable event-based processing.

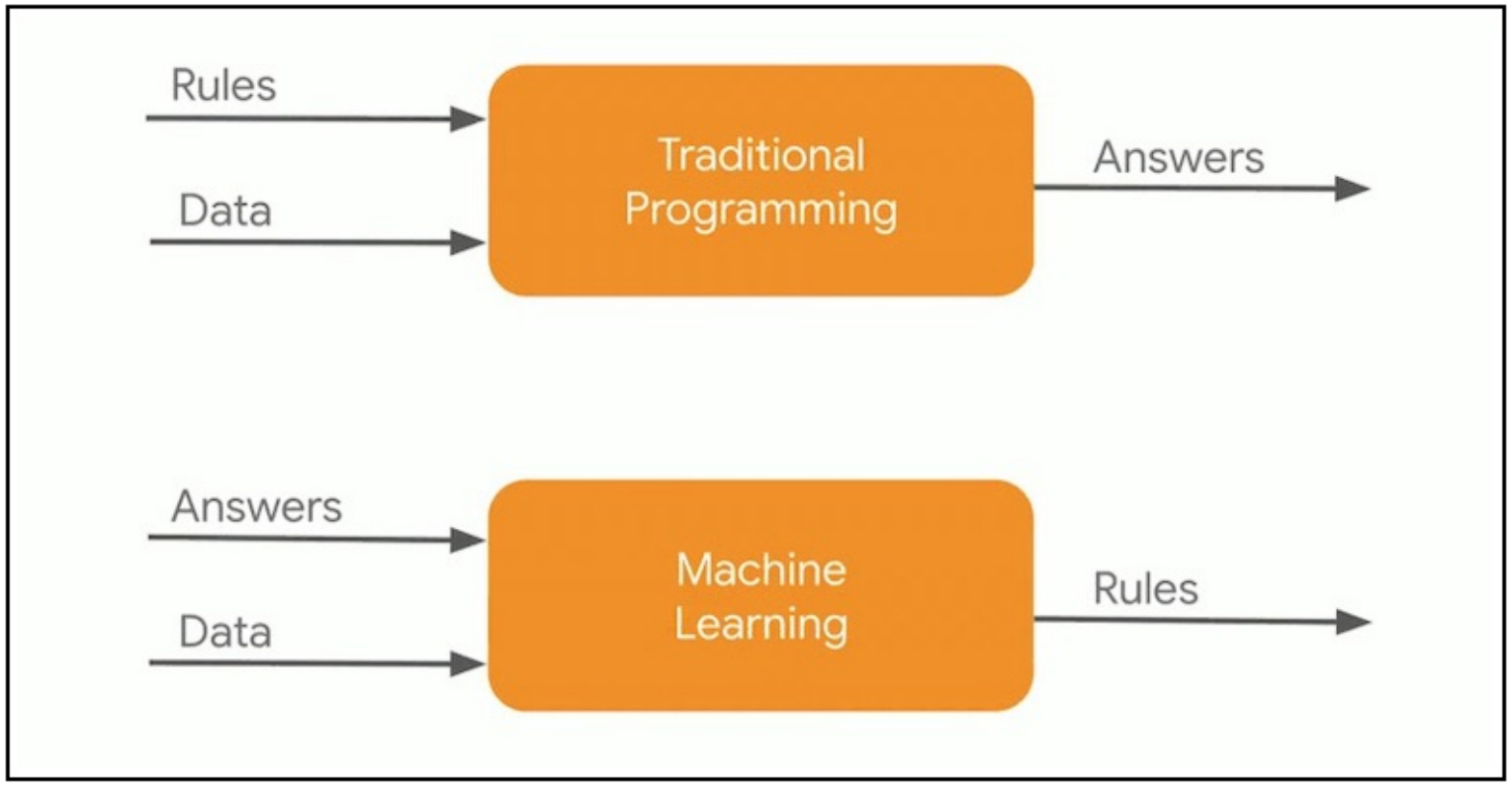
Machine Learning:

- Data mining.
- Modern neural networks. Input-to-output prediction.
- Identify objects, events, and anomalies.
- Learn structure and sequence. Remember stuff.

# Man and Machine (AI-ML View)

| **Man** | **AI-ML Machine** |
| --- | --- |
| • Good at formulating solutions to problems.<br>• Can work with incomplete data and information.<br>• Creative.<br>• Reasons logically, but very slow. Forgetful.<br>• Performance is static.<br>• Humans make the rules, then they break them. | • Manipulates Os and 1s.<br>• Can work with incomplete data and information.<br>• Creative.<br>• Fast logical reasoning.<br>• Performance doubles every 18-24 months.<br>• Data mining can discover the rules. |

# Traditional Programming vs AI-ML Workflow



# Introduction to Data Mining

# Numerous Definitions

## Data Mining

The field of data mining addresses the question of how to best use historical data to discover general regularities and improve future decisions (Mitchell, 1999).

## Data Mining

Data mining is the extraction of implicit, previously unknown, and potentially useful information – structural patterns – from data (Witten et al., 2017).

The process of discovering useful patterns from data must be automatic (or at least semi-automatic). Useful patterns allow us to reduce uncertainty and make nontrivial predictions on new data.

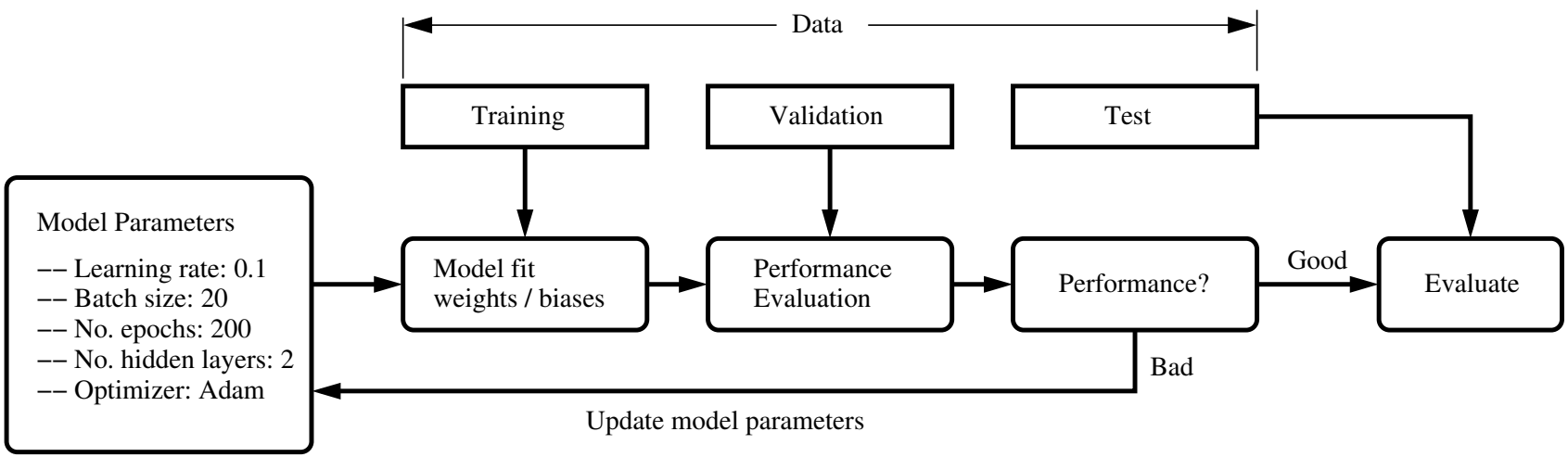
# Data Mining Techniques

## Working with Initial Dataset

- Data cleaning and curation
- Remove redundant features
- Identify input variables and output variable.

## Preprocessed Dataset:

- Data split: 80% for training, 20% for validation and testing.



# Data Mining Techniques

**Training Dataset**

- The sample of data used to fit the model.

**Validation Dataset**

- The sample of data used to provide an unbiased evaluation of the model fit on the training dataset while training the model parameters.

**Testing Dataset**

- The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

# Data Mining Techniques

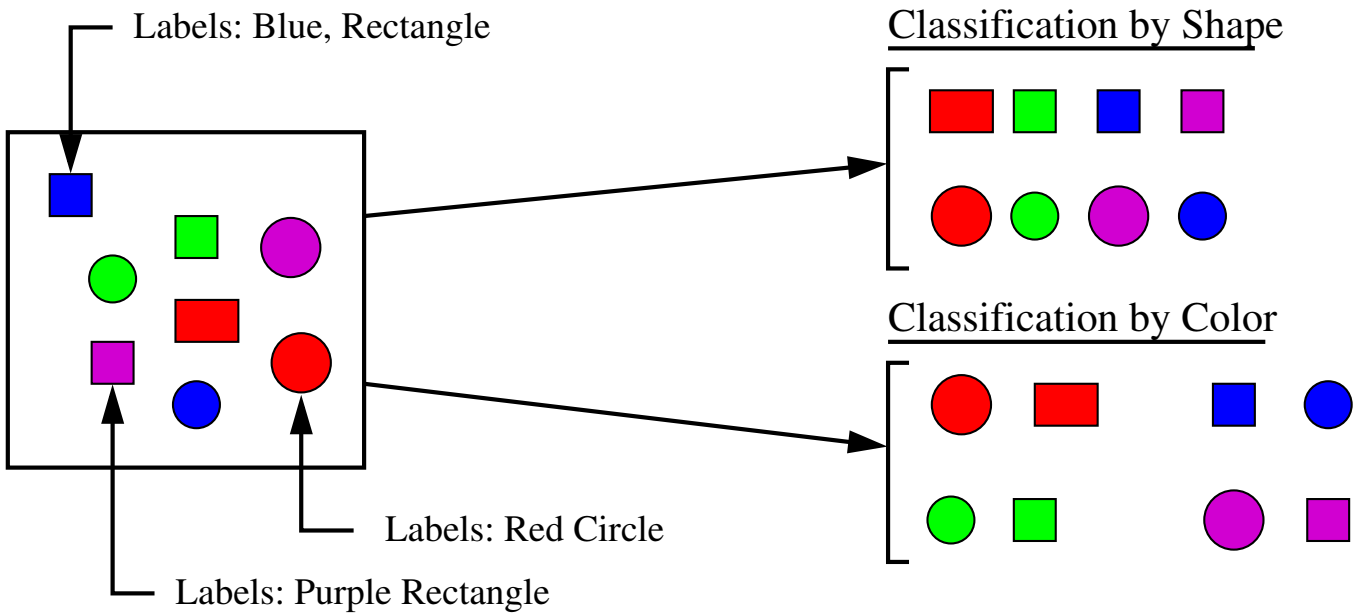
# Classification Analysis

# Data Mining Techniques

## Classification Analysis

Classification analysis learns a method for predicting the instance class from pre-labeled (classified) instances.

**Classification by Shape/Color** (Supervised Learning)



# Data Mining Techniques

## Classification Problem

- **Given** a set of $n$ attributes (ordinal or categorical), a set of $k$ classes, and a set of labeled training instances,

$$[(i_i, l_i), \cdots, (i_j, l_j)], \tag{1}$$

where $i = (v_1, v_2, \cdots, v_n)$ and $l \in (c_1, c_2, \cdots, c_k)$.

- **Goal** is to determine a classification rule – sequence of tests on the attributes – that predicts the class of any instance from the values of its attributes.

## Note

- This is a generalization of the concept learning problem since typically there are more than two (outcome) classes.
- Data will contain scatter; may have missing values.

# Decision Trees

# Decision Trees

## Decision Tree

A decision tree breaks decisions into a tree-like structure (hierarchical structure) of rules, making it easy to understand how predictions are made step by step.

## Binary Decision Tree

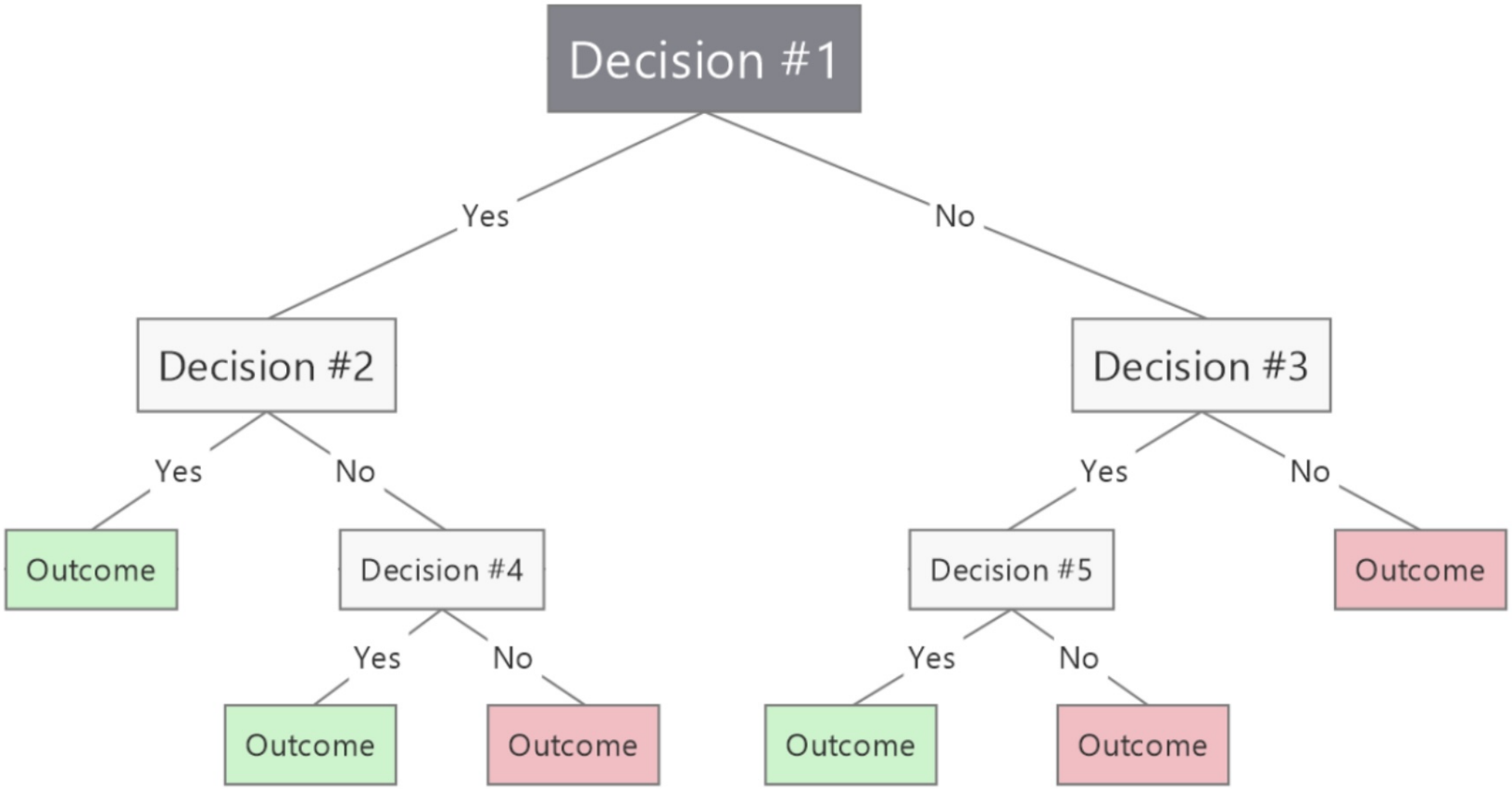
A binary decision tree is a tree-like structure that has exactly two branches representing outcomes for a given condition (typically, yes/no, true/false, and so forth).

## Regression Decision Tree

A regression decision tree is assembled by splitting attributes on continuous domains.

# Decision Trees

## Hierarchical Tree Structure:



# Decision Trees

**Hierarchical Tree Structure:**

- One root node plus internal nodes, edges and leaf nodes.
- Each internal node represents a test/condition on an attribute.
- Intermediate decisions split the data into subsets based on the outcomes of the test/condition
- Each branch represents the outcome of a test.
- Each leaf represents either a classification or prediction.

**Boolean Function Domains**

- Each attribute is binary valued (true or false; yes or no).

**Continuous Domains**

- Tests check if each attribute $a_i >$ value.

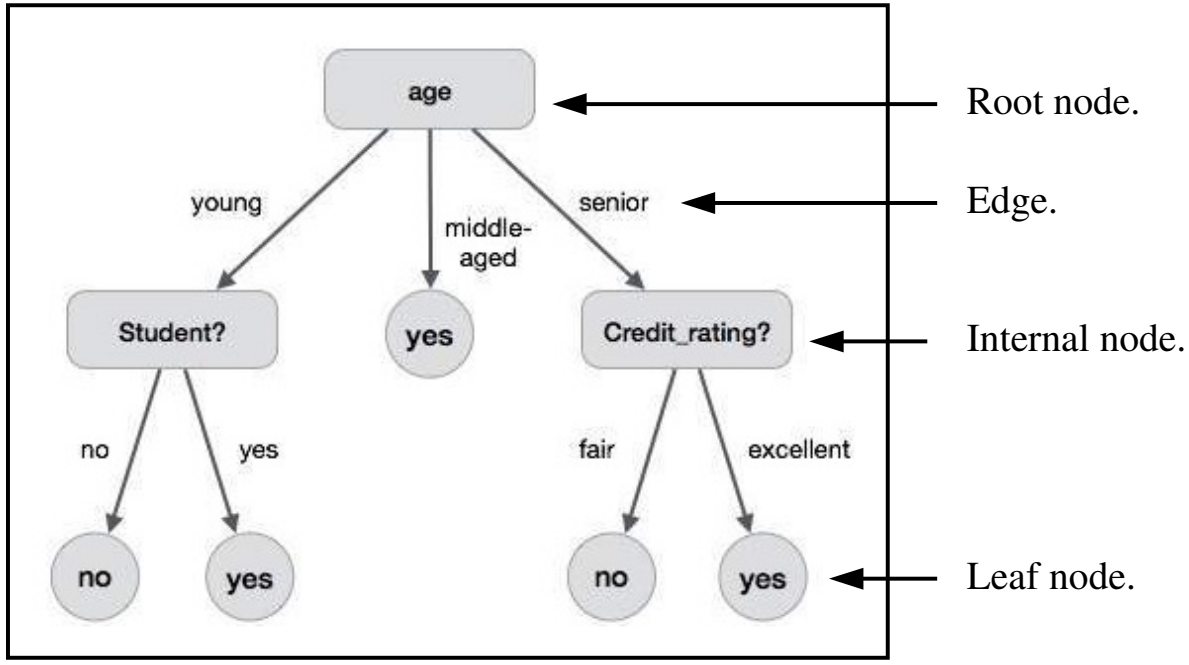
# Example 1: Decision Tree for Buying a Computer

**Sample Dataset.** Will customer buy a computer?

| ID | Age Group | Income | Student | Credit Rating | Buys Computer |
|---|---|---|---|---|---|
| 1 | young | high | no | fair | no |
| 2 | young | high | no | excellent | no |
| 3 | middle | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle | low | yes | excellent | yes |
| 8 | young | medium | no | fair | no |
| 9 | young | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | young | medium | yes | excellent | yes |
| 12 | middle | medium | no | excellent | yes |
| 13 | middle | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Example 1: Decision Tree for Buying a Computer

**Sample Decision Tree** (Split on Discrete Domain)




The pathways from root node to leaf node represent the rules of classification.

# Decision Trees

**Basic Questions:**

- How to choose the attribute (or value) to split on at each level of the tree?
- When should a node be declared a leaf?
- If a leaf is impure, how should it be labeled?
- If the tree is too large, how can it be pruned?

**Strategy for Tree Assembly:**

- When all of the data in a single node comes from the same class, can declare the node to be a leaf and stop splitting.
- When a group of data points have exactly the same attribute values, we cannot split any further. Declare the node to be a leaf, and output the class that is the majority.

# Decision Trees

**Data Mining Algorithms**

- Perceptron.
- Logistic Regression.
- Decision tree algorithms (C4.5, J48)
- Support Vector Machines (SVM).
- Random Forest.

**Data Mining Applications**

- Anomaly (Fraud) detection.
- Medical diagnosis.
- Industrial applications.

# Regression Decision Trees

## Regression Decision Trees

Regression decision trees are used for applications where the target variable is a continuous number such as performance, price, etc.

**Tree Model:**

- Comprised of piecewise-constant prediction surfaces – data points within a leaf get the same predicted numerical value.

**Data Split Procedure:**

- Assembly of the decision tree occurs by identification of the best feature(s) and split point(s) to divide the data in such a way that minimizes an error metric (MSE).
- Works with both continuous and categorical data.

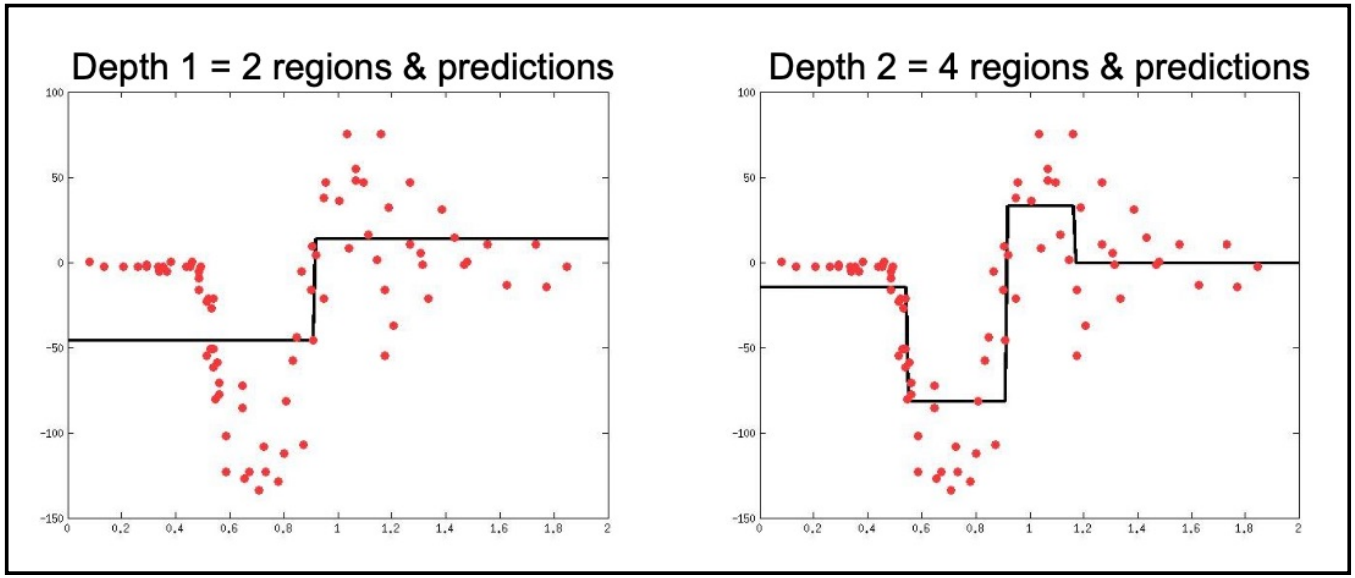
# Regression Decision Trees

**Prediction:**

- Start at the root and follow branches based on the feature values of the data until you reach a leaf node. Prediction is the average value of data at the leaf node.

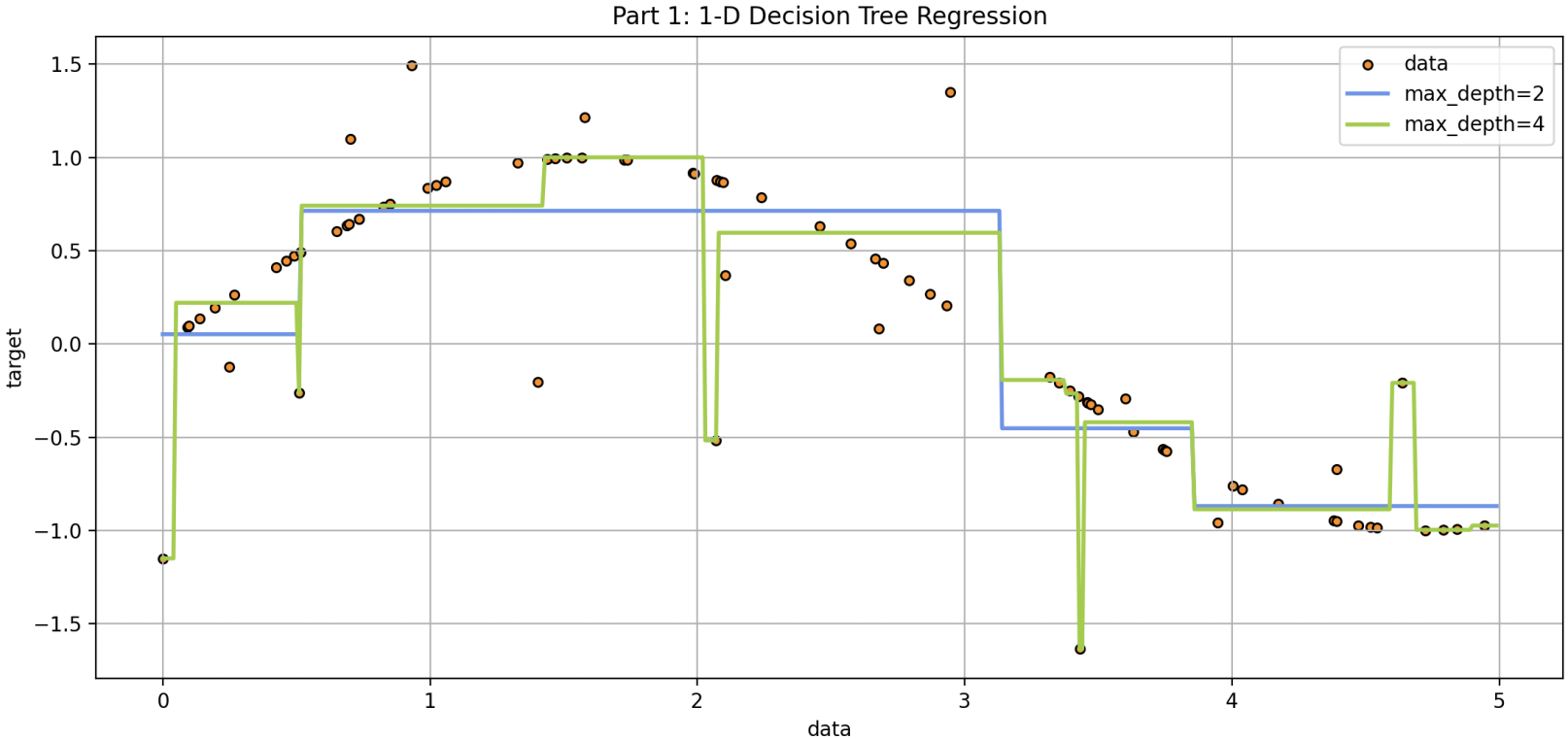
---

**Example:** Prediction of a Single Scalar Feature



# Regression Decision Trees

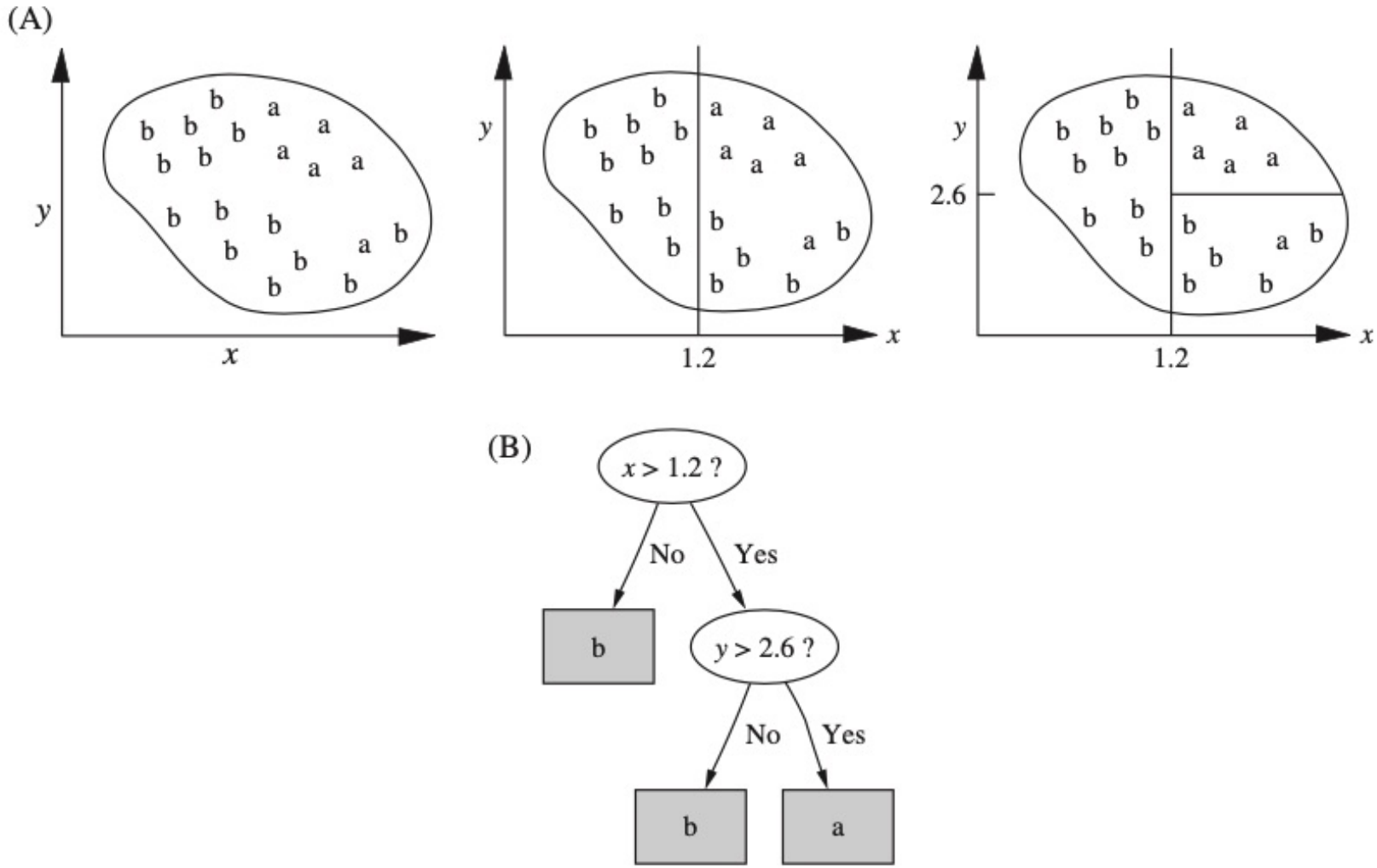
**Example:** One-Dimensional Regression (depth = 2 and 4)




**Source:** python-code-ai.d/regression/Test-Regression-Plot02.py

# Regression Decision Trees

## Splitting Data on Multiple Domains:

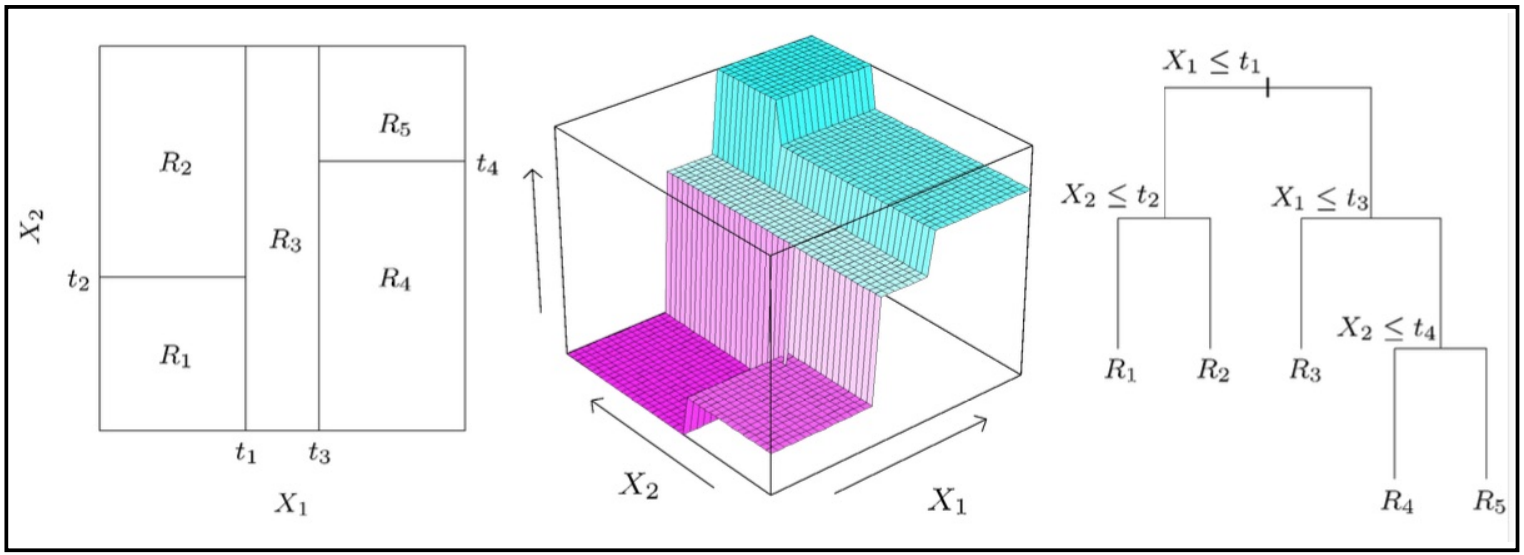


# Regression Decision Trees

**Example:** Two-Dimensional Regression:

- Each node splits tree according to a single feature.
- Mean values of training data are predicted at leaf nodes.

## Prediction of Two-Dimensional Regions



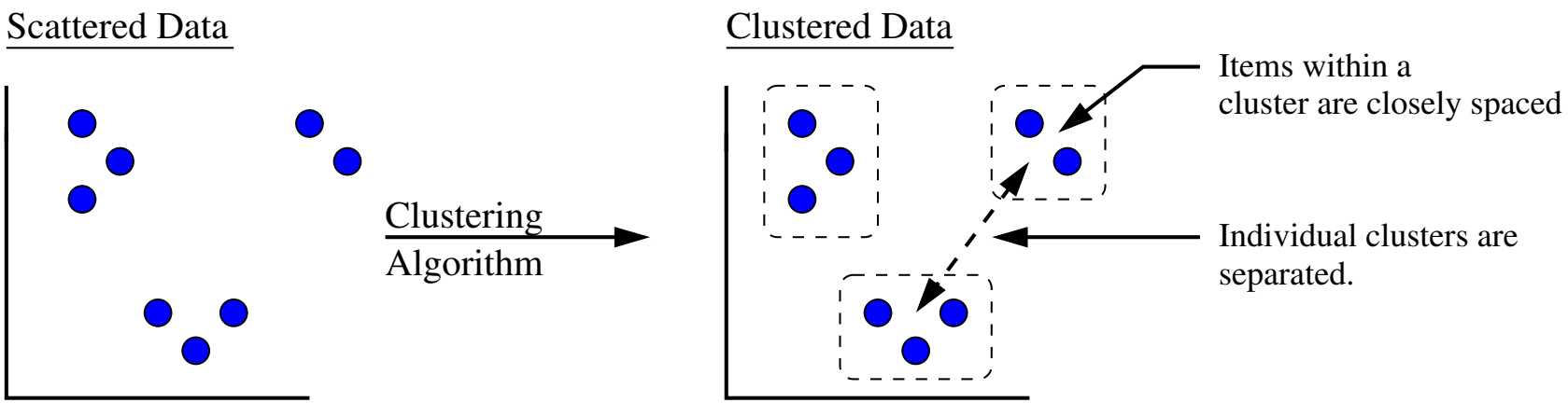
# Clustering

# Data Mining Techniques

## Clustering Problems

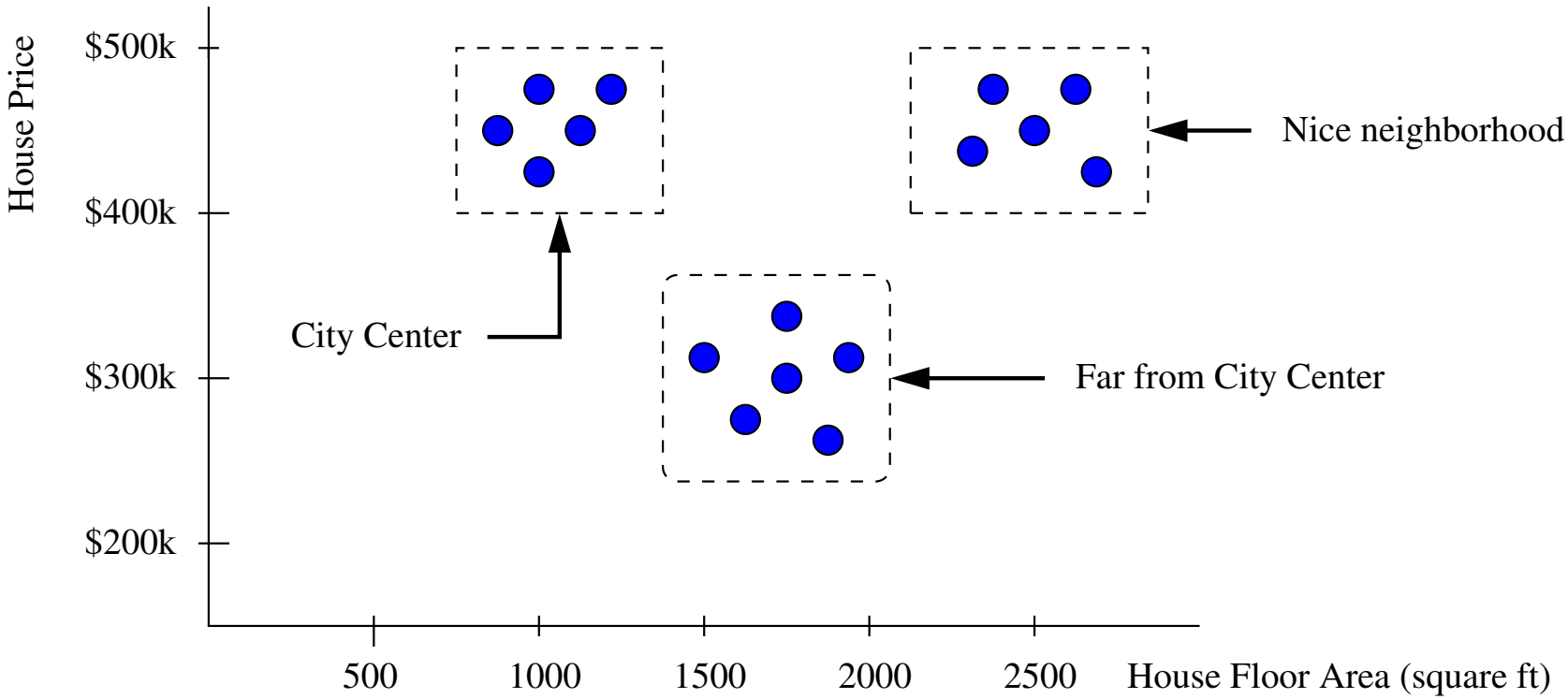
Clustering techniques apply when there is no class to be predicted, but when un-labeled instances need to be divided into common natural groups.

**Clustering Process** (Unsupervised Learning)



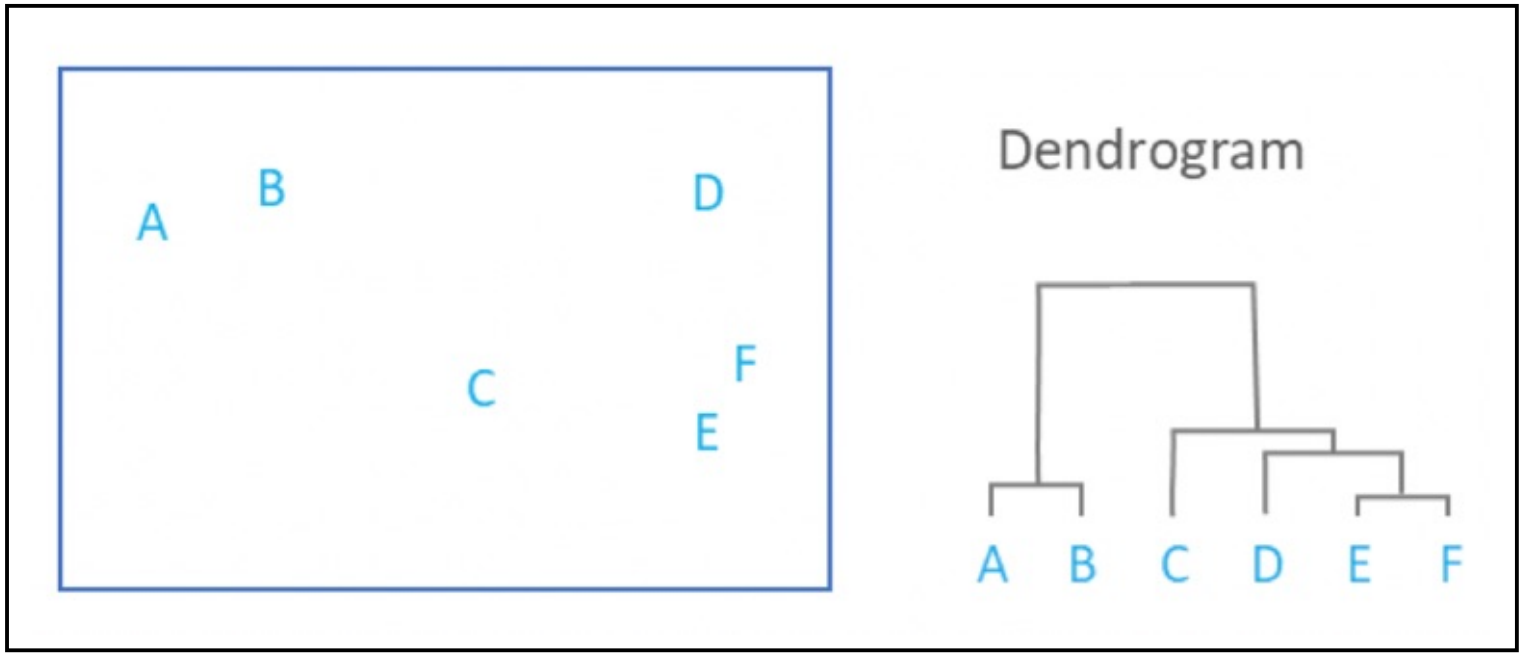
# Data Mining Techniques

**Example 1.** Clustering of House Prices and Floor Areas



# Data Mining Techniques

**Example 2.** Hierarchical Clustering and Dendrograms




## Dendrogram

A dendrogram is a branching (tree) diagram that represents relationships of similarity among groups of entities.

# Data Mining Techniques

**Algorithms**

- K-means clustering.
- Hierarchical clustering.

**Applications**

- Preprocessing step for many scientific applications.
- Natural language processing.
- Market segmentation.
- Netflix/movie recommendations.

# Association

# Relationships

# Data Mining Techniques

**Association**

Association is a data mining function that discovers the probability the co-occurrence of items (or patterns) in a collection of data.

## Association Rules

- Identify relationships between co-occurring items can be expressed as association rules (e.g., if X, then Y).

## Key Challenges

- How to identify useful correlations among all correlations?
- Correlation relationships are not the same as dependency relationships – *if X, then Y* does not *imply if Y, then X* !
- Historical data does not necessarily predict the future.

# Data Mining Techniques

**Goals of Predictive Analysis**

- For a customer who purchases product A, what other products will they purchase?
- Will coupons increase same-store sales?
- Will a reduced price mean higher sales?

**Retail Strategies**

- Put most frequently purchased item (e.g., milk) at the back of the store.
- Co-locate items that are bought together – can lead to increase in sales for both.

# Data Mining Techniques

**Example 1.** iPhone Color and Personality Traits.

| Phone Color | Personality Traits |
|---|---|
| Green | Fresh, harmonious, healthy, hopeful. |
| Blue | Confident, dependable, trustworthy. |
| Yellow | Happy, honorable, intelligent. |
| Pink | Compassionate, energetic, playful. |
| White | Balanced, calm, clean. |

Customers want to select an iPhone Color that correlates with their personality traits.

# Data Mining Techniques

**Example 2.** Urban Legend from early 1990s: Diapers and Beer

| ID | Items |
|---|---|
| 1 | {Bread, Milk} |
| 2 | {Bread, Diapers, Beer, Eggs} |
| 3 | {Milk, Diapers, Beer, Cola} |
| 4 | {Bread, Milk, Diapers, Beer} |
| 5 | {Bread, Milk, Diapers, Cola} |
| ... | ... |

market basket transactions

## Examples of Association Rules

- $\{Diapers\} \longrightarrow \{Beer\}$,
- $\{Milk, Bread\} \longrightarrow \{Eggs, Coke\}$,
- $\{Beer, Bread\} \longrightarrow \{Milk\}$.

# Data Mining Techniques

**Itemset** and **k-Itemset**

- A collection of one or more items (e.g., $\{Milk, Bread\}$.
- k-Itemset is an itemset containing k items.

**Support Count** $\sigma$

- Frequency of ocurrence of an itemset.
- Example: $\sigma(\{Milk, Bread, Diaper\}) = 2$.

**Support**

- Indicates how frequently the if/then relationship appears in the data.

**Association Rule**

- Expression of the form X $\longrightarrow$ Y, where X and Y are itemsets.

# Data Mining Techniques (Rule Evaluation Metrics)

**Support** (s)

- Fraction of transactions that contain both X and Y.
- Support(s) $= \frac{\sigma\{Milk, Diaper, Beer\}}{T} = 2/5 = 0.4.$

**Confidence** (c)

- Measures how often items in Y appear in transactions that contain X.
- Confidence(c) $= \frac{\{Milk, Diaper, Beer\}}{\{Milk, Diaper\}} = 2/3 = 0.67.$

**Data Mining for Association Rules**

Given a set of transactions $T$, find all rules having:

- Support(s) $\geq$ min support threshhold.
- Confidence(c) $\geq$ min confidence threshold.

# Data Mining Techniques (Brute-Force Enumeration)

## Brute-Force Enumeration

- Compute support and confidence for all possible association rules.
- Prune rules that do not meet min support/confidence thresholds.

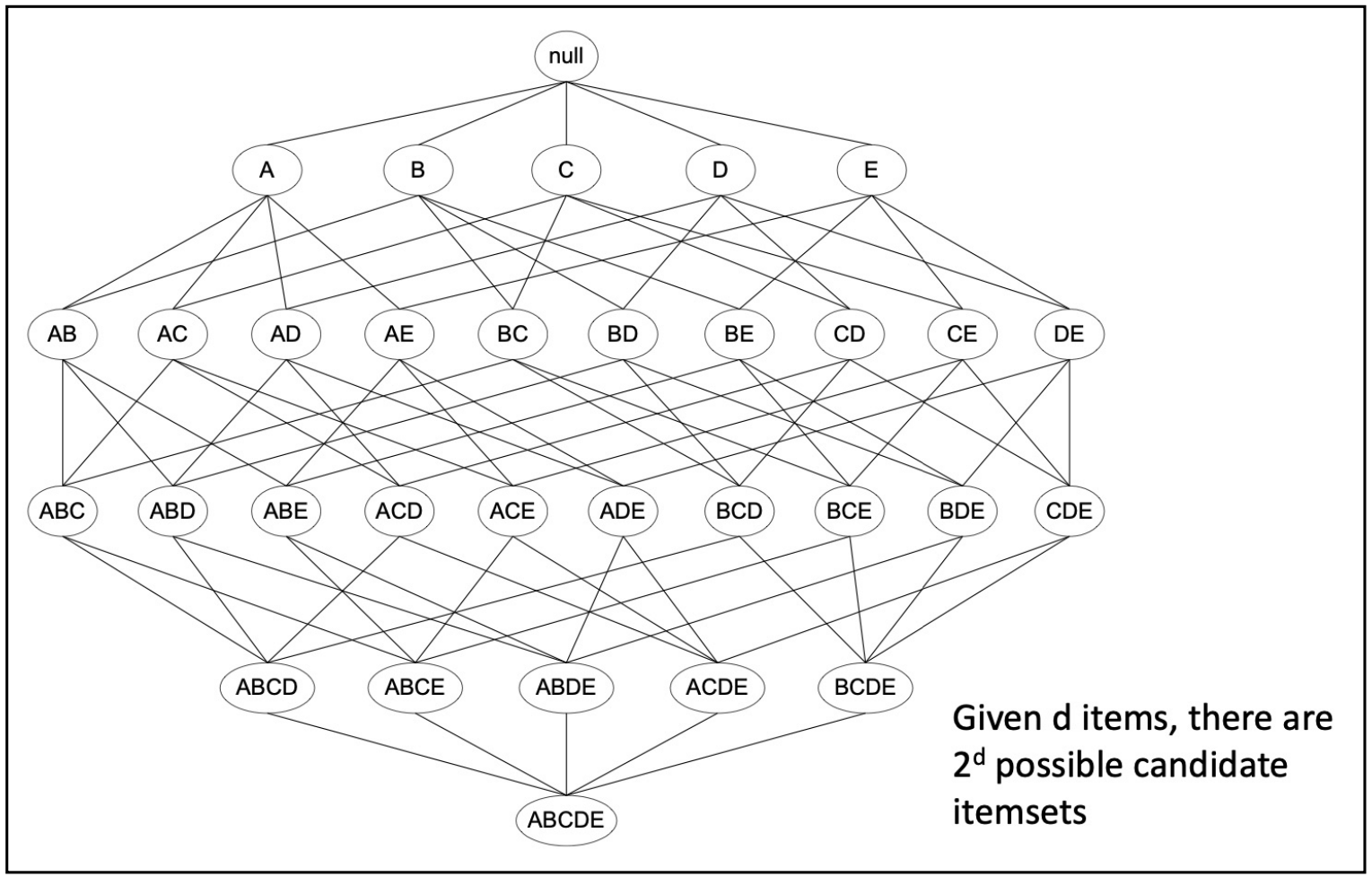
| TID | Items |
|---|---|
| 1 | Bread, Milk |
| 2 | Bread, Diaper, Beer, Eggs |
| 3 | Milk, Diaper, Beer, Coke |
| 4 | Bread, Milk, Diaper, Beer |
| 5 | Bread, Milk, Diaper, Coke |

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

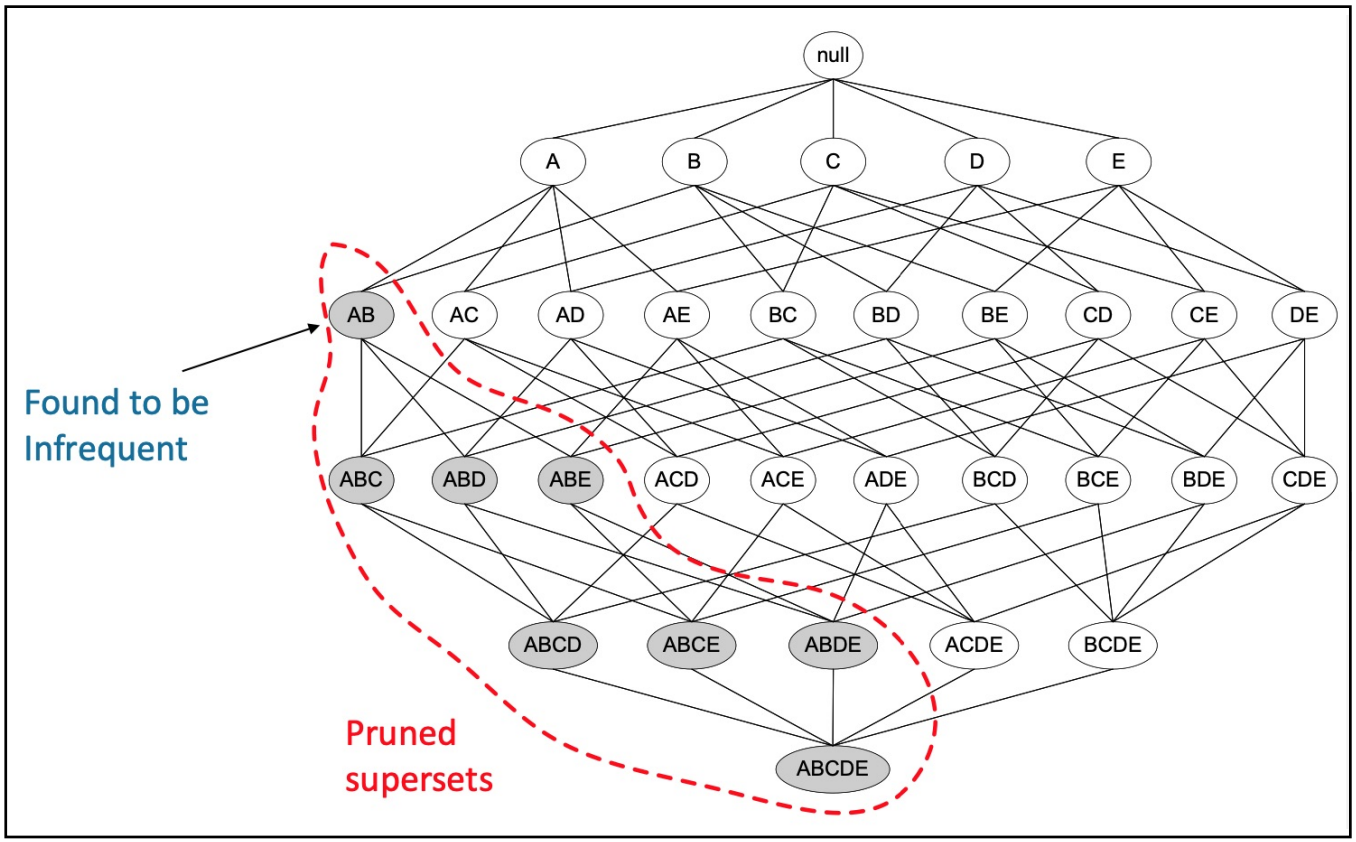
# Data Mining Techniques (Brute-Force Enumeration)

**Computational Complexity:** Given $d$ items, there are $2^d$ possible candidate itemsets.



# Data Mining Techniques (Brute-Force Enumeration)

Need strategies to reduce computational effort by systematically pruning the low scoring items from candidate space.



# Data Mining Techniques

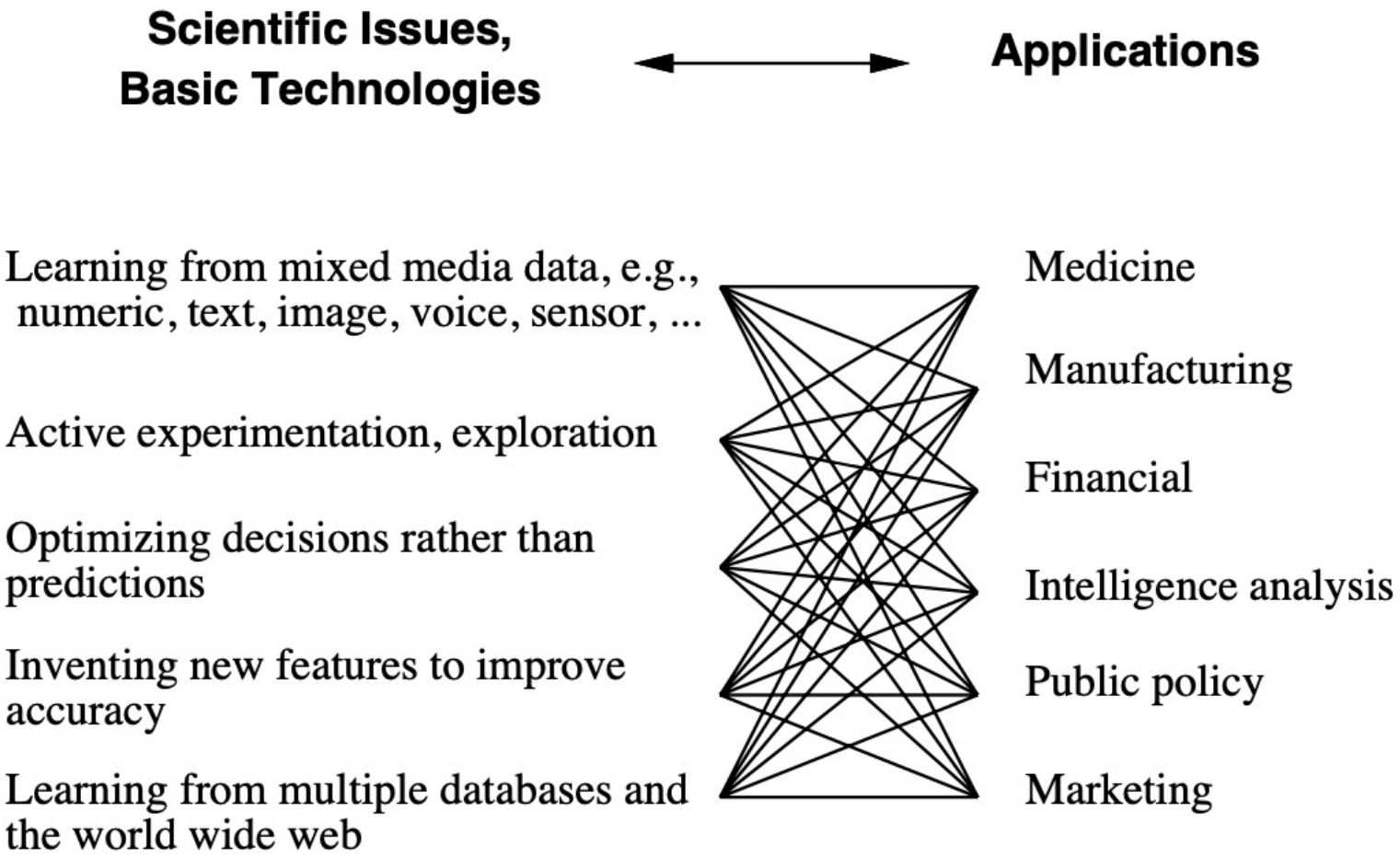
**Algorithms** (see Chapter 6 of Witten et al.)

- **Apriori**: Follows a generate-and-test methodology for finding frequent item sets, generating successively longer candidate item sets, and then scanning the item sets to see if they meet threshold limits.
- **Frequent Pattern Trees**: Begins by counting the number of times individual items – attribute-value pairs – occur in the dataset. This is a single pass. Then, a (sorted) tree structure is constructed with the goal of identifying large (frequent) item sets.

**Applications**

- Weather prediction,
- Medical diagnosis,
- Purchasing habits of retail customers.

# Scientific Research Enabling Applications




**Source:** Mitchell, 1999.

# Theoretical Considerations

Probability, Gini Impurity, Entropy, Information Gain

# Probability

**Definition of Probability:**

Simply put, probability is how likely something is to happen. Let the probabilities of $n$ possible outcomes $A_1$, $A_2$, $\cdots$, $A_n$, of an experiment be $p_1$, $p_2$, $\cdots$, $p_n$, respectively. The distribution:

$$P = (p_1, p_2, p_3, \cdots, p_n), \tag{2}$$

satisfies the constraints:

$$\sum_{i=1}^{n} p_i = 1, \tag{3}$$

and

$$p_1 \geq 0, p_2 \geq 0, \cdots, p_n \geq 0. \tag{4}$$

# Gini Impurity and Entropy

Decision tree procedures use two metrics, Gini Impurity and Entropy (next topic), to decide how to split data into branches. Both metrics determine how mixed or pure a dataset is, thereby helping to guide a model toward splits that create cleaner groups.

---

## Definition of Gini Impurity

Gini impurity measures how often a randomly chosen element of a set would be incorrectly labeled if it were labeled randomly and independently according to the distribution of labels in the set. Gini reaches its minimum (zero) when all cases in the node fall into a single target category.

# Gini Impurity

**Motivation:**

- Prevents random or uninformative splits that reduce predictive strength.
- Helps isolate class boundaries for better interpretability.
- Controls node quality and prevents over-growth of branches.
- Reduces classification ambiguity by separating noisy samples.
- Maintains model consistency across varied datasets.

# Gini Impurity

**Mathematical Formula:**

$$\text{Gini} = \sum_{i=1}^{N} p_i \left(1 - p_i\right) = 1 - \sum_{i=1}^{N} p_i^2 \tag{5}$$

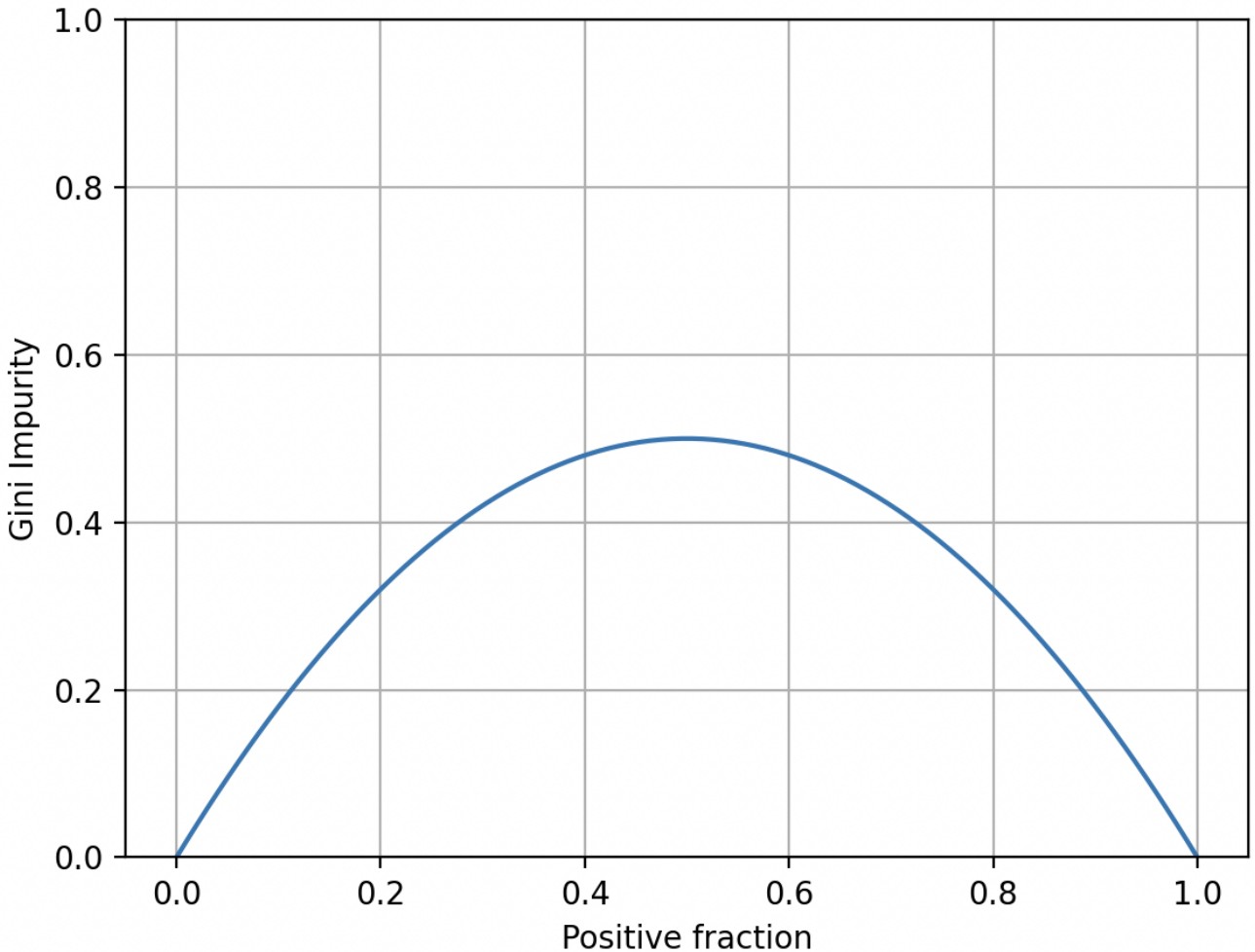
where $N$ is the number of classes and $p_i$ is the probability of class $i$.

**Properties:**

- Lower values indicate cleaner and more homogeneous nodes.
- Nodes become pure when all samples belong to one class.
- Slightly biased toward dominant classes during split selection.

# Gini Impurity

**Gini Impurity vs positive fraction $p$:**



# Entropy

## Definition of Entropy

As it relates to machine learning, entropy is is a measure of the randomness (disorder or uncertainty) of information being processed.

**Simple Example:** Tossing a Fair Coin (High Entropy):

- A fair coin has no affinity (or preference) for heads or tails.
- The outcome any number of tosses is difficult to predict because there no relationship between coin flipping and the outcome.

# Mathematical Models of Entropy

**Principle of Maximum Entropy (Jaynes, 1957)**

Given some partial information about a random variate, we should choose the probability distribution that is is consistent with the given information (e.g., boundary constraints), but otherwise has maximum entropy associated with it.

**Relationship of Entropy to Uncertainty and Probability**

- Every probability distribution has some uncertainty associated with it. Entropy provides a quantitative measure of this uncertainty.
- A principle goal of data mining models and algorithms is to reduce uncertainty.

# Measuring Uncertainty of a Probability Distribution

**Requirements for Measuring Uncertainty** (Kapur, 1989):

- It should be a function of $p_1$, $p_2$, $\cdots$, $p_n$, i.e.,

$$H = H_n(P) = H(p_1, p_2, \cdots, p_n). \tag{6}$$

- $H_n(P)$ should be a continuous and symmetric function.
- The maximum value of $H_n$ should increase as $n$ increases.
- It should be minimum (and possibly zero) when there is no uncertainty about the outcome. In other words, it should vanish when one of the outcomes is certain.

$$H_n(P) = 0 \text{ when } p_i = 1 \text{ and } p_j = 0, \ (j \neq i). \tag{7}$$

# Measuring Uncertainty of a Probability Distribution

- $H_n$ should be maximum when there is maximum uncertainty, which arises when the outcomes are equally likely, i.e.,

$$p_1 = p_2 = \cdots = p_n = \frac{1}{n}. \tag{8}$$

- For two independent probability distributions $P$ and $Q$,

$$\sum_{i=1}^{n} p_i = 1, \text{ and } \sum_{j=1}^{m} q_j = 1, \tag{9}$$

the uncertainty of the joint scheme $P \cup Q$ should be:

$$H_{m+n}(P \cup Q) = H_n(P) + H_m(Q). \tag{10}$$

If P and Q have outcomes $A_1$, $A_2$, $\cdots$, $A_n$ and $B_1$, $B_2$, $\cdots$, $r_n$, then the joint outcomes are $A_i B_j$ with probabilies $p_i q_j$.

# Mathematical Models of Entropy

## Shanon's Measure of Entropy

Shanon (1949) proposed the following measure:

$$H_n(P) = \sum_{i=1}^{n} p_i ln(\frac{1}{p_i}) = -\sum_{i=1}^{n} p_i ln(p_i). \quad (11)$$

Intial Observations:

- This function is continuous, symmetric, and convex.
- When one of the probabilities is 1, the others are zero. The entropy is zero and is a minimum value – no surprise.
- All of the commonly used probability distributions – uniform, normal, poisson, logarithmic – can be framed in terms of maximum entropy subject to constraints.

# Mathematical Models of Entropy

## Maximum Value of Entropy

We can use Lagrange's equations to find a maximum value, i.e.

$$-\sum_{i=1}^{n} p_i ln(p_i) - \lambda \left[ \sum_{i=1}^{n} p_i - 1 \right]. \tag{12}$$

This gives (uniform distribution):

$$p_1 = p_2 = \cdots = p_n = \frac{1}{n}. \tag{13}$$

The maximum value of $H_n$ is:

$$H_n = -\sum_{i=1}^{n} \frac{1}{n} ln(\frac{1}{n}) = ln(n) \rightarrow \text{ increases linearly with n.} \tag{14}$$

# Mathematical Models of Entropy

**Illustrative Example**

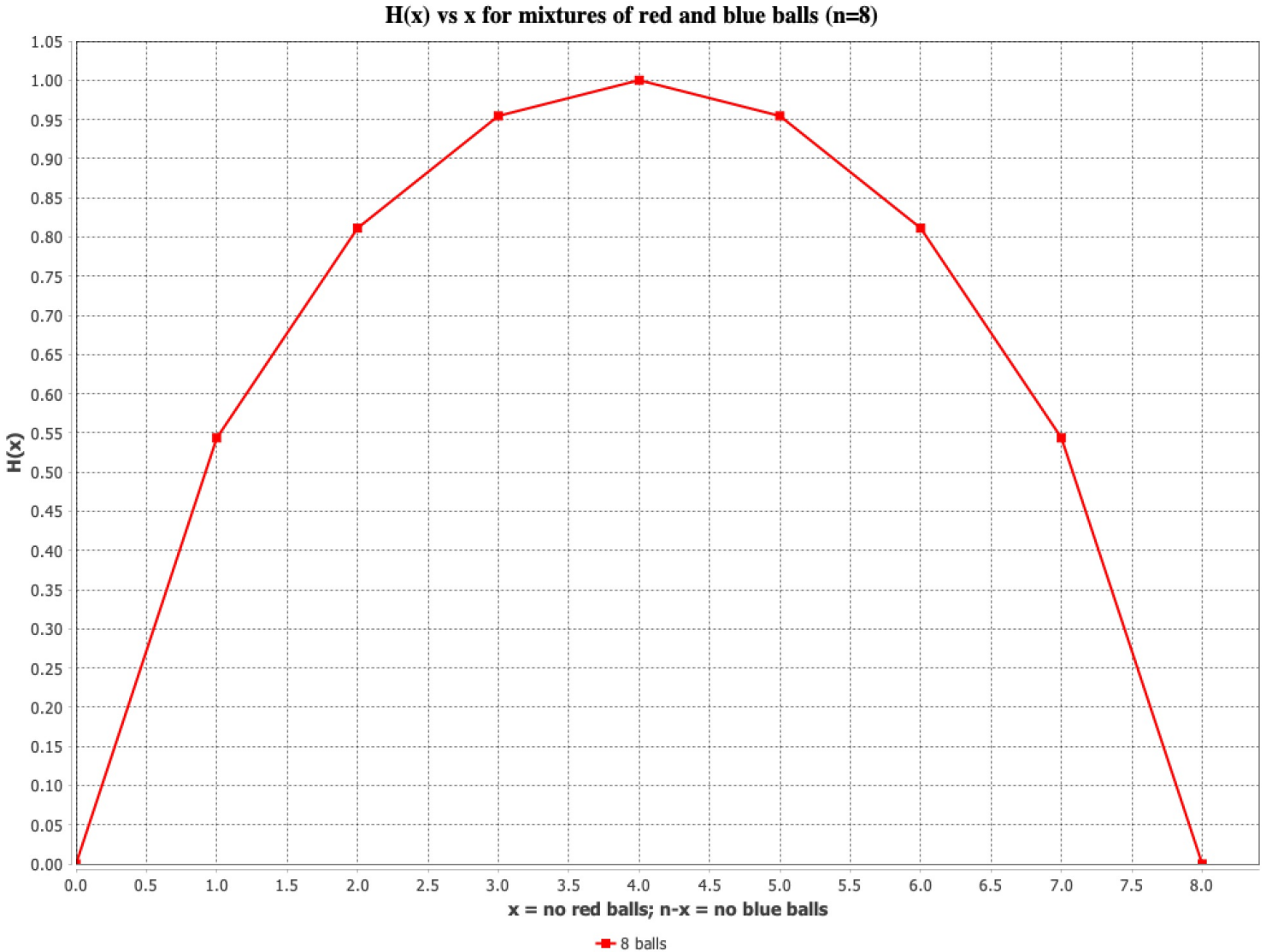
Suppose that an urn contains a mixture of red ($n_r$) red and blue ($n_b$) balls (i.e., $n = n_r + n_b$). The entropy is:

$$H_2(P) = -\left[\frac{n_r}{n}\right] \log_2 \left[\frac{n_r}{n}\right] - \left[\frac{n_b}{n}\right] \log_2 \left[\frac{n_b}{n}\right]. \tag{15}$$

**Sample Calculation.** Let $n_r = 2$, $n_b = 6$.

$$\begin{aligned} H_2(P) &= -\left[\frac{2}{8}\right] \log_2 \left[\frac{2}{8}\right] - \left[\frac{6}{8}\right] \log_2 \left[\frac{6}{8}\right] \\ &= \frac{1}{4} \cdot 2.0 + \frac{3}{4} \cdot 0.415 = 0.811 \end{aligned} \tag{16}$$

# Mathematical Models of Entropy



# Mathematical Models of Entropy

Key Points:

- Minimum values of entropy occur when the urn contains only red balls (i.e., x = 0) or only blue balls (i.e., x = 8). There is no disorder.
- The maximum value of entropy occurs when the urn system has maximum disorder – that is, four blue balls and four red balls.

$$H_2(P) = -\left[\frac{4}{8}\right]\log_2\left[\frac{4}{8}\right] - \left[\frac{4}{8}\right]\log_2\left[\frac{4}{8}\right] = 1.0 \qquad (17)$$

- Even higher levels of entropy (disorder) can be obtained by adding more colors to the urn, e.g., 2 blue balls, 2 green balls, 3 red balls, 1 purple ball. Now, P = $\left(\frac{1}{4}, \frac{1}{4}, \frac{3}{8}, \frac{1}{8}\right)$.

# Side-by-Side: Gini Impurity vs Entropy

**Split Behavior:**

- Gini impurity creates splits quickly, favoring dominant classes.
- Entropy produces more balanced node partitions

**Dataset Size:**

- Gini impurity is efficient for large, high-dimensional datasets.
- Entropy is useful for structured datasets with balanced classes.

**Sensitivity to Distribution:**

- Gini impurity is less sensitive to small probability differences.
- Entropy is more sensitive to subtle probability differences.

# Information Gain in Decision Trees

# Mathematical Framework

## Information Gain

The amount of information that is gained by knowing the value of an attribute. It equals the entropy of a distribution before a split minus the entropy of a distribution after a split.

$$IG(Y, X) = H(Y) - H(Y|X). \tag{18}$$

Here:

- Information gain $IG(X, Y)$ is the reduction of uncertainty about $Y$ given an additional piece of information $X$ about $Y$.
- $H(Y)$ is the entropy of $Y$ (before split).
- $H(Y|X)$ is the conditional entropy of $Y$ given the value of attribute $X$ (after split).

# Decision Trees

**Design of Data Partitions for Classification Tree:**

- Use information gain as measure for attribute selection.
- Pick atttribute split that maximizes information gain IG(Y,X), i.e.,

$$IG(D, A) = H(D) - \sum_{i=1}^{v} \frac{D_j}{D} H(D_j) \tag{19}$$

Here:

- $D$ is a prescribed data partition and A is an attribute.
- Split $D$ into $v$ partitions (or subsets) $\{D_1, D_2, \cdots, D_j\}$, where $D_j$ contains those tuples in $D$ that have outcome $a_j$ of A.

# Decision Trees

**Basic Algorithm** (This is a greedy algorithm) ...

- Decision tree is constructed in a top-down recursive divide-and-conquer manner.
- When the construction process begins, all training examples are at the root.
- Attributes are categorical – if continuous-valued they are discretized in advance.
- Need to design a sequence of selected attributes to partition dataset recursively.
- Test attributes are selected on basic of heuristic or statistical measure (e.g., information gain).

# Decision Trees

## Conditions for Stopping Partitioning

- Stop partitioning when all samples for a given node belong to the same class.

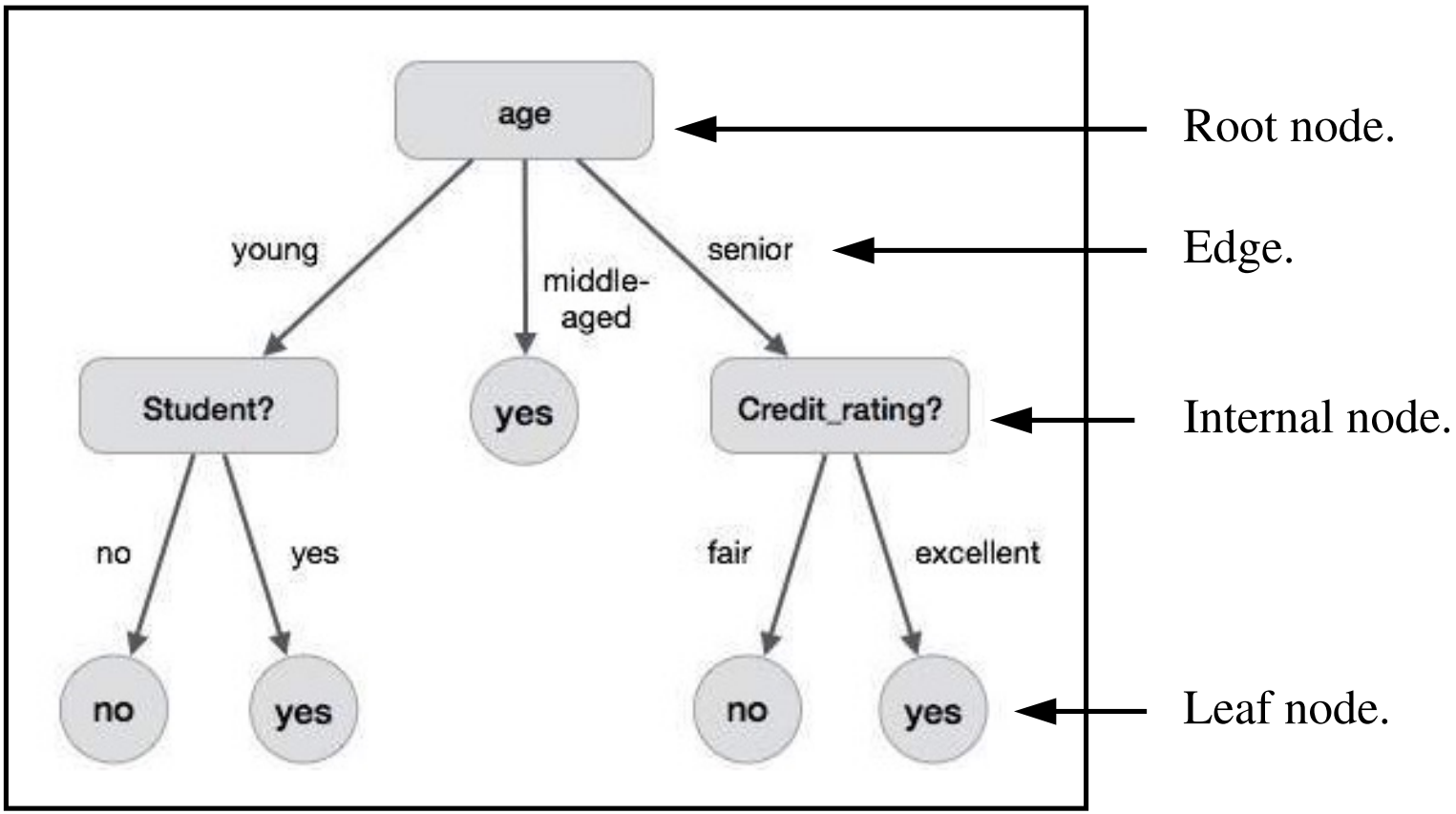
# Example 1 (Buy Computer)

**Initial Dataset.** Will customer buy a computer?

| ID | Age Group | Income | Student | Credit Rating | Buys Computer |
|---|---|---|---|---|---|
| 1 | young | high | no | fair | no |
| 2 | young | high | no | excellent | no |
| 3 | middle | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle | low | yes | excellent | yes |
| 8 | young | medium | no | fair | no |
| 9 | young | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | young | medium | yes | excellent | yes |
| 12 | middle | medium | no | excellent | yes |
| 13 | middle | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Example 1 (Buy Computer)

**Sample Decision Tree:** Initially split data based on age group.




Is this a good decision?

# Example 1 (Buy Computer)

## Entropy of Base Dataset

Purchase outcomes: $\{no = 5, yes = 9\}$.

$$H(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.94 \qquad (20)$$

**Partitioned Dataset.** Split dataset by age ...

| ID | Age Group | Income | Student | Credit Rating | Buys Computer |
|---|---|---|---|---|---|
| 1 | young | high | no | fair | no |
| 2 | young | high | no | excellent | no |
| 8 | young | medium | no | fair | no |
| 9 | young | low | yes | fair | yes |
| 11 | young | medium | yes | excellent | yes |

# Example 1 (Buy Computer)

**Partitioned Dataset.** Split dataset by age ...

| ID | Age Group | Income | Student | Credit Rating | Buys Computer |
|---|---|---|---|---|---|
| 3 | middle | high | no | fair | yes |
| 7 | middle | low | yes | excellent | yes |
| 12 | middle | medium | no | excellent | yes |
| 13 | middle | high | yes | fair | yes |

| ID | Age Group | Income | Student | Credit Rating | Buys Computer |
|---|---|---|---|---|---|
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 10 | senior | medium | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Example 1 (Buy Computer)

**Entropy of Partitioned Dataset.** Split by age group ...

$$\begin{aligned} IG(D, Age) &= H(D) - \sum_{v \in \{you, mid, sen\}} \frac{S_v}{S} H(S_v) \\ &= H(D) - \frac{5}{14} H(S_{you}) - \frac{4}{14} H(S_{mid}) - \frac{5}{14} H(S_{sen}) \\ &= 0.246. \end{aligned} \tag{21}$$

Remaining split options:

- IG(D,Income) $= 0.029$,
- IG(D,Student) $= 0.151$,
- IG(D,Credit Rating) $= 0.048$.

# Example 1 (Buy Computer)

**Conclusion**

- Attribute age has the highest information gain and therefore becomes the splitting attribute at the root node.

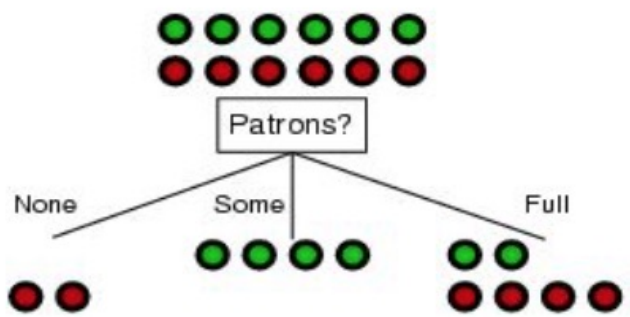
**Actions**

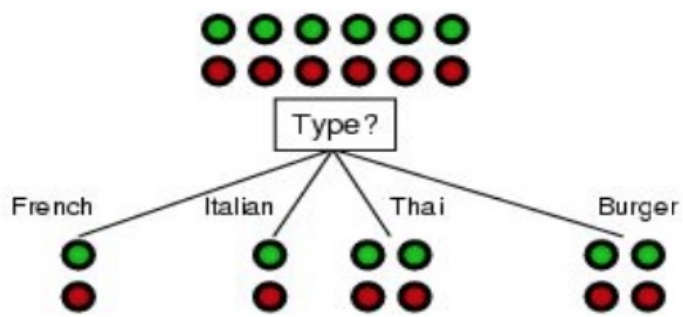
- Branches are grown for each outcome of age.
- Repeat process on lower-level nodes using split attributes of student and credit rating.

# Example 2 (Customer Wait for Table at Restaurant?)

**Customer Dataset** (Source: Russell and Norvig, 2010)

| Example | Attributes | | | | | | | | | | Target |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Alt* | *Bar* | *Fri* | *Hun* | *Pat* | *Price* | *Rain* | *Res* | *Type* | *Est* | *Wait* |
| $X_1$ | T | F | F | T | Some | $$$ | F | T | French | 0–10 | T |
| $X_2$ | T | F | F | T | Full | $ | F | F | Thai | 30–60 | F |
| $X_3$ | F | T | F | F | Some | $ | F | F | Burger | 0–10 | T |
| $X_4$ | T | F | T | T | Full | $ | F | F | Thai | 10–30 | T |
| $X_5$ | T | F | T | F | Full | $$$ | F | T | French | >60 | F |
| $X_6$ | F | T | F | T | Some | $$ | T | T | Italian | 0–10 | T |
| $X_7$ | F | T | F | F | None | $ | T | F | Burger | 0–10 | F |
| $X_8$ | F | F | F | T | Some | $$ | T | T | Thai | 0–10 | T |
| $X_9$ | F | T | T | F | Full | $ | T | F | Burger | >60 | F |
| $X_{10}$ | T | T | T | T | Full | $$$ | F | T | Italian | 10–30 | F |
| $X_{11}$ | F | F | F | F | None | $ | F | F | Thai | 0–10 | F |
| $X_{12}$ | T | T | T | T | Full | $ | F | F | Burger | 30–60 | T |



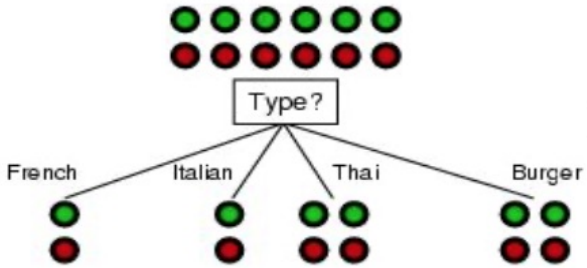
# Example 2 (Customer Wait for Table at Restaurant?)

**Dataset Attributes**

- **Alternate:** Is there a suitable alternate restaurant nearby?
- **Bar:** Does restaurant have comfortable bar area to wait in?
- **Fri/Sat:** True on Fridays and Saturdays.
- **Hungry:** True when customer is hungry.
- **Patrons:** How many people are in the restaurant? (none, some, and full).
- **Price:** The restaurant price range ($, $$ and $$$).
- **Raining:** Is it raining outside?
- **Reservation:** Did customer make a reservation?
- **Type:** Type of restaurant (French, Italian, Thai, or Burger).
- **WaitEstimate:** Wait time estimated by host (0-10 mins, 10-30, 30-60, or $>$ 60).

# Example 2 (Customer Wait for Table at Restaurant?)

## Split on Restaurant Type Attribute

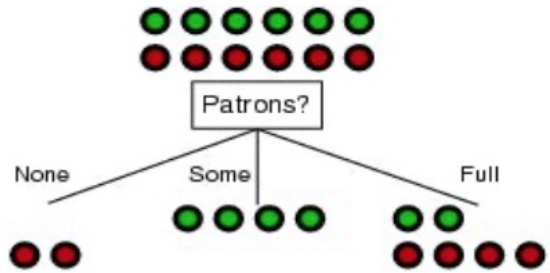



Root entropy: 0.5 * log(2) + 0.5 * log(2) = 1 bit

Leaf entropies: 2/12 * 1 + 2/12 * 1 + … = 1 bit

No reduction!

## Split on Patrons Attribute



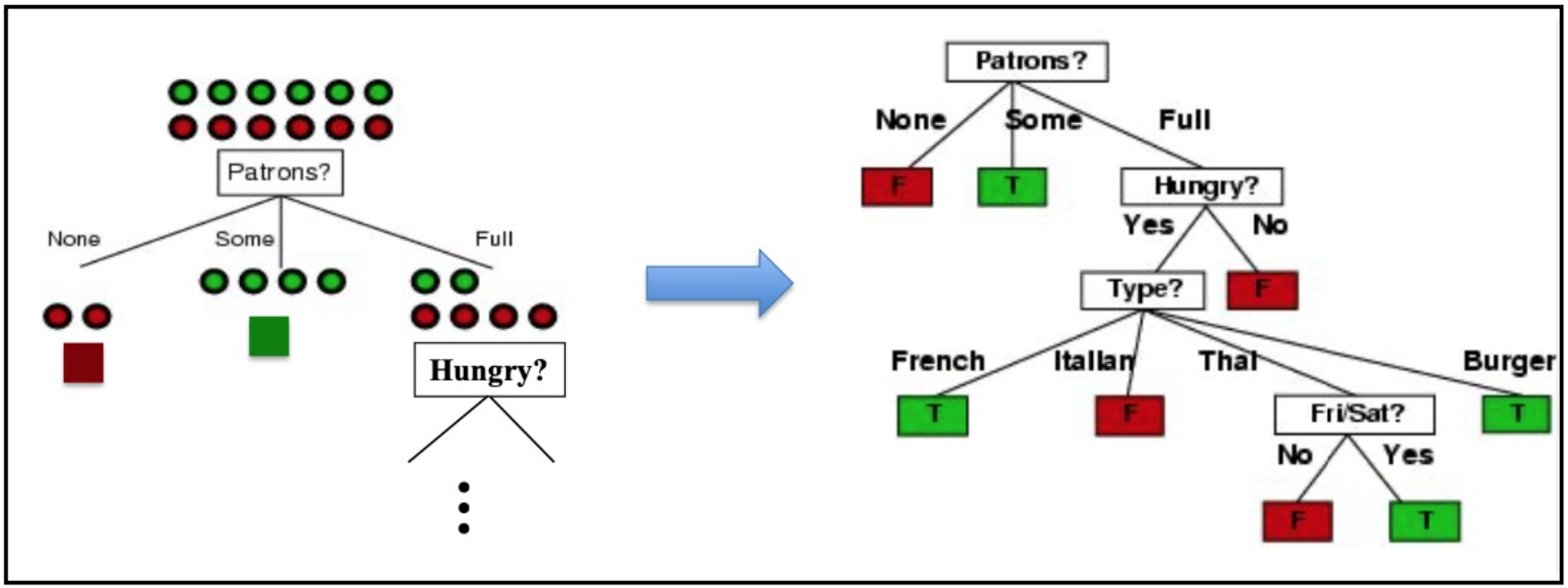

Root entropy: 0.5 * log(2) + 0.5 * log(2) = 1 bit

Leaf entropies: 2/12 * 0 + 4/12 * 0 + 6/12 * 0.9

Lower entropy after split!

# Example 2 (Customer Wait for Table at Restaurant?)

## Decision Tree Synthesis



# Classification with Decision Trees (Summary)

**Advantages**

- Decision trees are simple to understand and interpret.
- Requires only a small number of observations.
- Best and expected values can be determined for different scenarios.

**Disadvantages**

- Difficulties in handling data with missing values.
- Information gain criterion is biased in favor of attributes with more levels.
- Calculations become complex if values are uncertain or outcomes are linked.

# Ensemble Learning

# Ensemble Methods (General Idea)

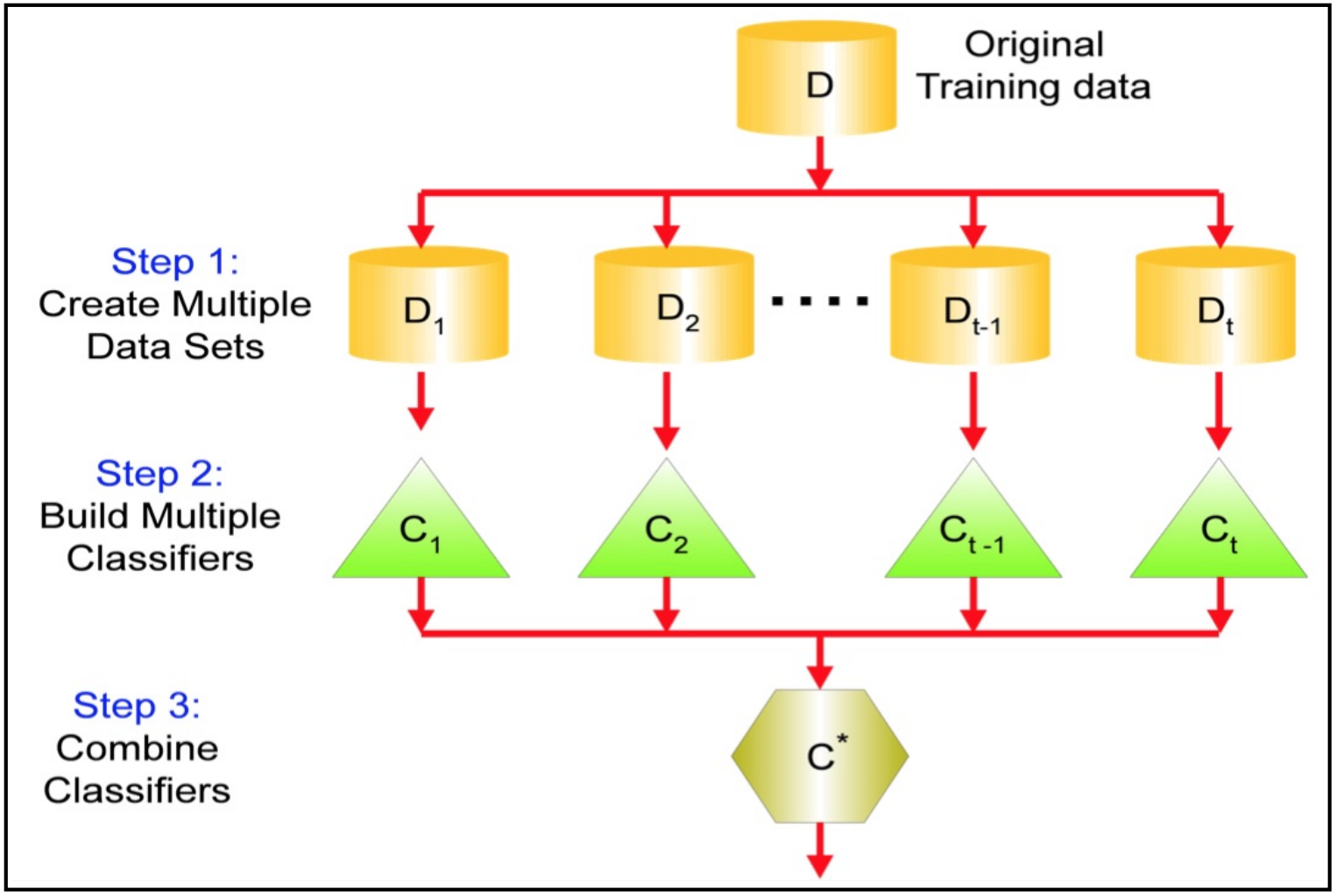
## Ensemble Methods

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any one constituent learning algorithm.

### Motivation and Approach

- Supervised learning algorithms search through a hypothesis space to find a hypothesis that will make good predictions.
- Even if the hypothesis space contains hypotheses that are well suited to a particular problem space, find a good hypothesis can still be very difficult.
- Ensembles combine hypotheses in the hope of finding a new one with superior predictive capabilities.

# Ensemble Learning (General Idea)



# Ensemble Learning (General Idea)

**Ensemble Learning**

- Combine predictions from multiple learning algorithms $\longrightarrow$ ensemble.
- Often leads to better predictive performance than a single learner.
- Works well then small differences in the training data produce very different classifiers (e.g., decision trees).

**Drawbacks**

- Increased computational effort.
- Reduced level of interpretability.

# Ensemble Learning (Why does it work?)

**Why does it work?**

- Assume classifiers $C_1, \cdots, C_k$ are independent, i.e.,

$$\text{correlation} \quad \sigma(C_1, C_2) = 0. \tag{22}$$

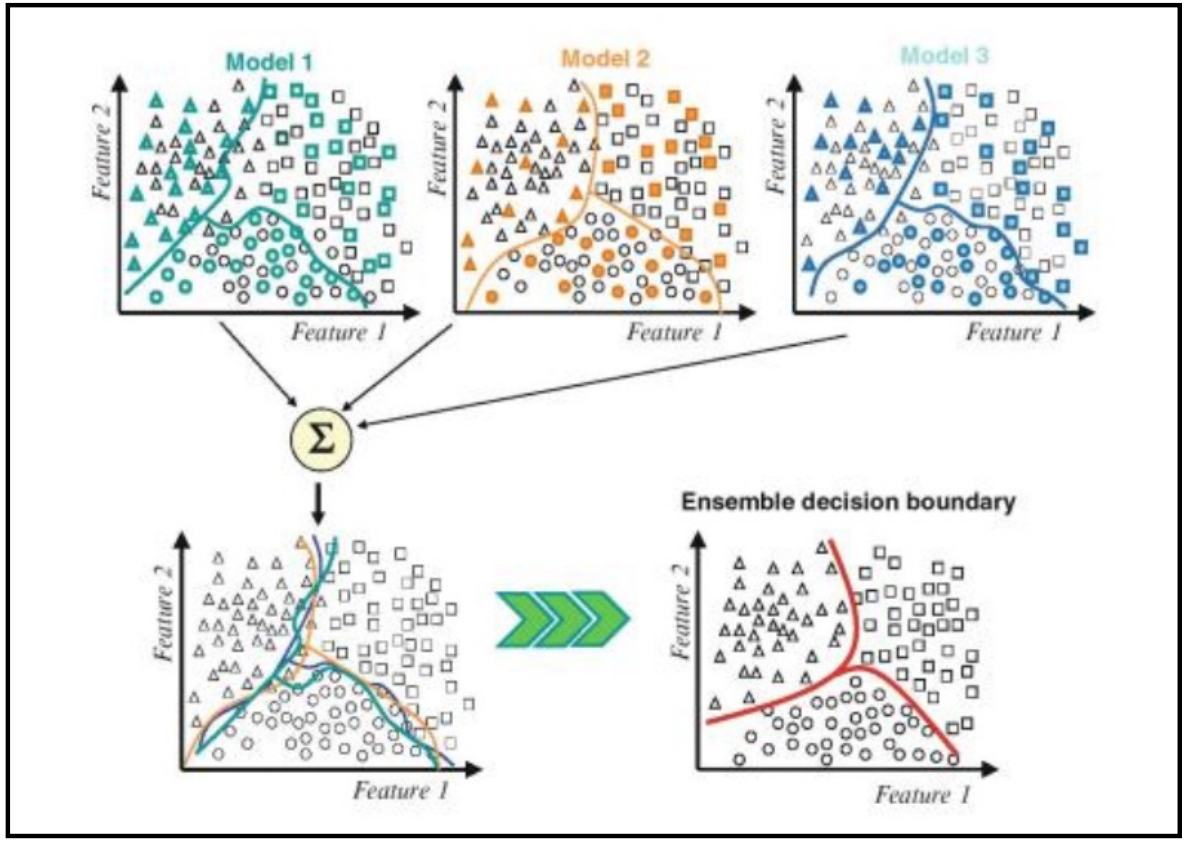
- Assume, for example, that there are 25 classifiers, each having an error rate $\eta = 0.35$.
- Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \eta^i (1-\eta)^{25-i} = 0.06. \tag{23}$$

which is much lower than any individual classifier.

# Ensemble Learning (Diversity in Prediction)

Use of ensemble methods can lead to improvements in prediction accuracy through reduction of variability.




Source: Zhang, et al, Ensemble Machine Learning, Springer, 2012.

# Ensemble Learning

**Constructing Ensembles:** Methods for obtaining sets of classifiers

- **Bagging.**
- **Random Forest.**
- **Cross-Validation.** Two key ideas: (1) instead of different classifiers, train same classifier on different data, (2) since training data is expensive, reuse data bu subsampling.

**Combining Classifiers:** Methods for combining different classifiers

- Stacking
- Bayesian Model Averaging
- Boosting
- AdaBoost

# Ensemble Techniques (Bagging)

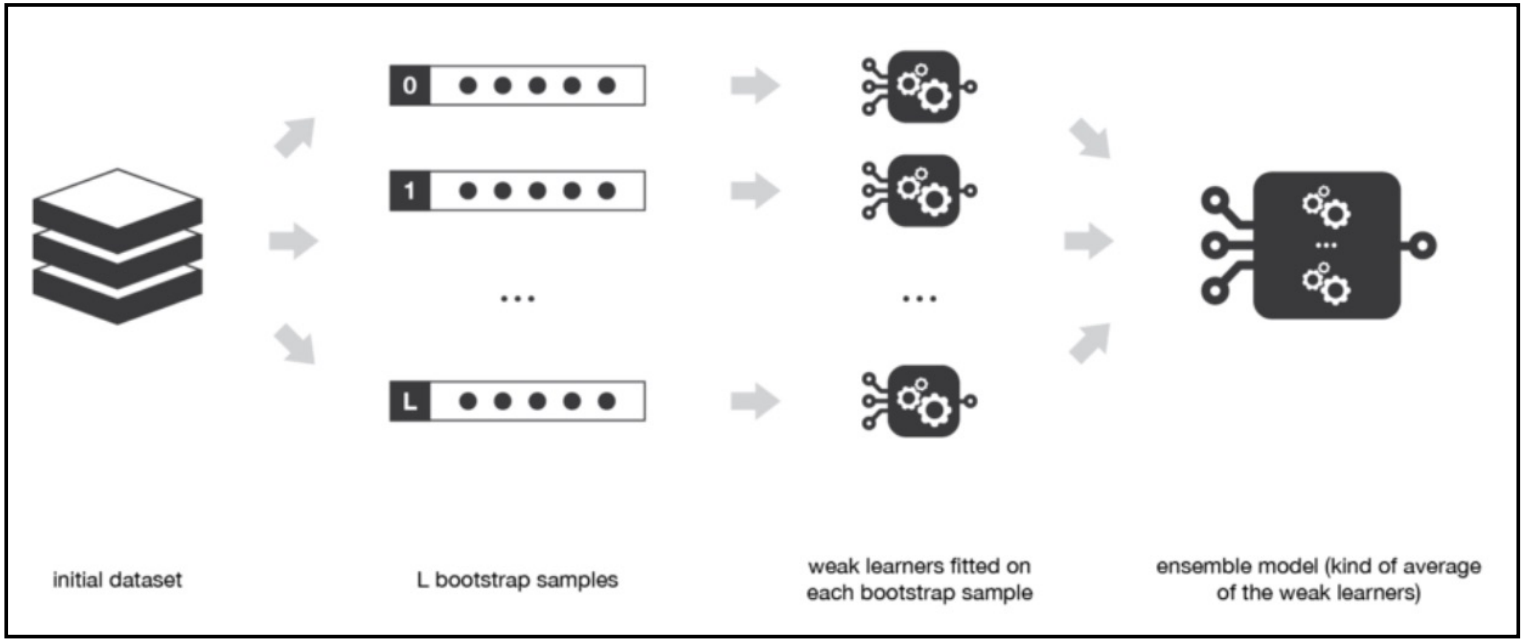
**Bagging** (Breiman, 1996). Bootstrapping on data.

- Create a data set by sampling data points with replacement.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Original Data | : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Bagging (Round 1): | | 7 | 2 | 9 | 7 | 3 | 2 | 1 | 1 | 4 | 5 |
| Bagging (Round 2): | | 6 | 10 | 4 | 2 | 10 | 3 | 8 | 9 | 7 | 4 |
| Bagging (Round 3): | | 4 | 6 | 8 | 2 | 5 | 1 | 6 | 3 | 1 | 9 |
| Bagging (Round 4): | | ..... | | | | | | | | | |
| Bagging (Round 5): | | ..... | | | | | | | | | |

- Create models based on the data sets.
- Generate more data sets and models.
- Make predictions by combining votes – Classification $\rightarrow$ majority vote; prediction $\rightarrow$ average.

# Ensemble Techniques (Bagging)



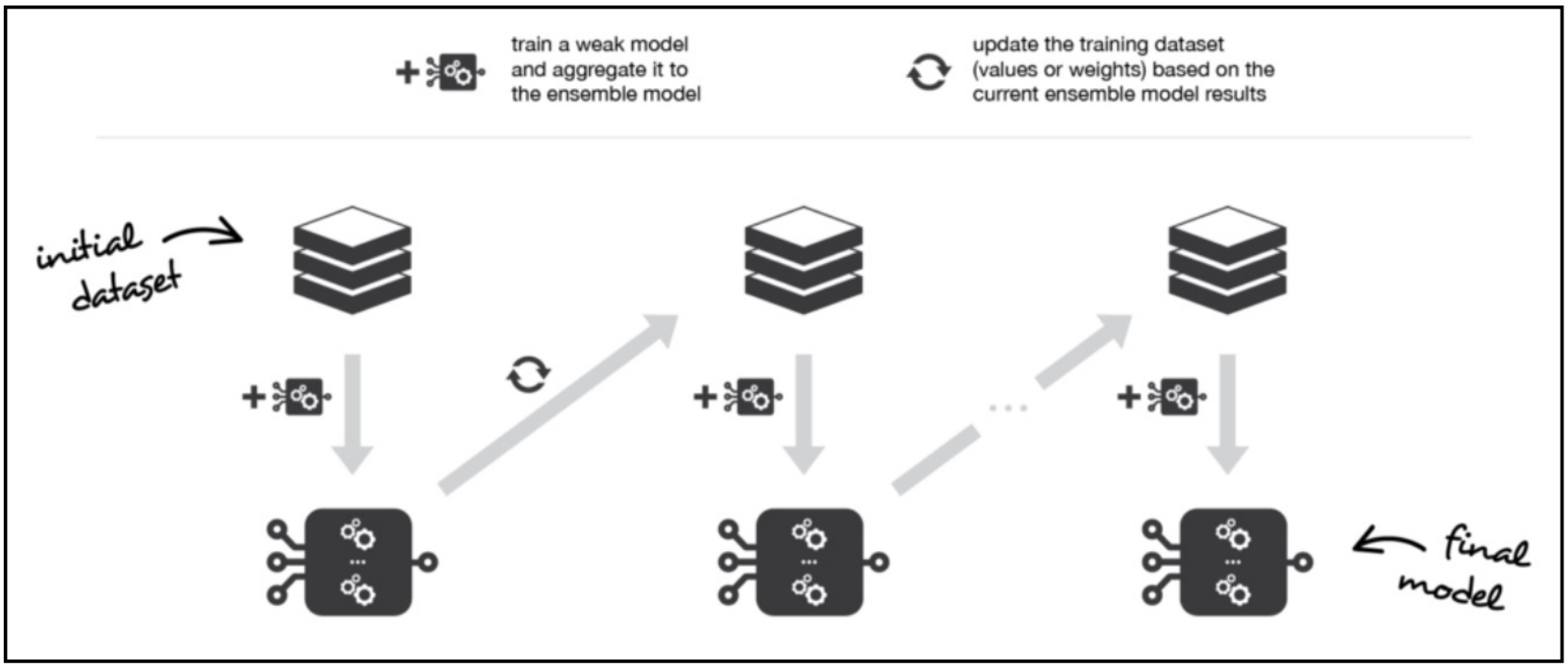

**Advantages/Disadvantages:**

- Helps when classifier is unstable (has high variance).
- Not helpful when classifier is stable and has large bias.

# Ensemble Techniques (Overview)

**Boosting** (Schapire, 1998). Recursively reweight data.

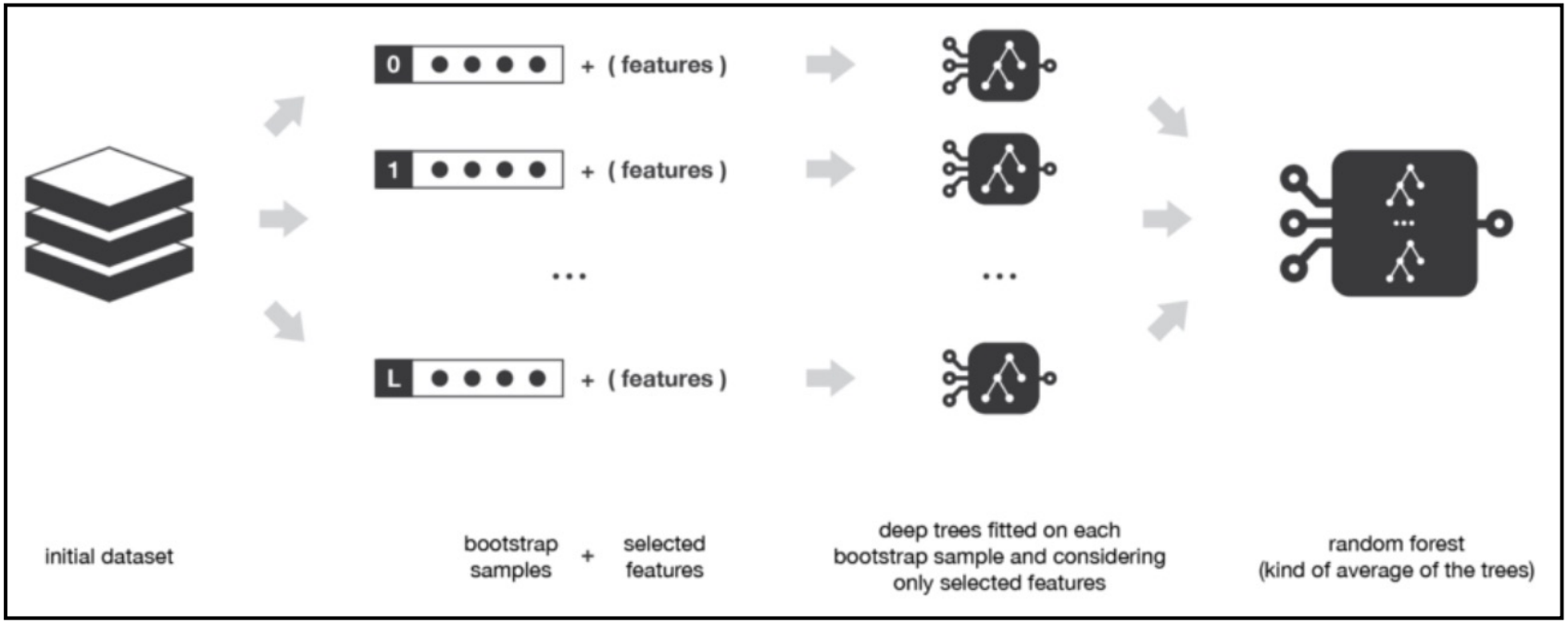
- Records wrongly classified will have their weights increased.
- Records correctly classified will have their weights decreased.



# Ensemble Techniques (Random Forest)

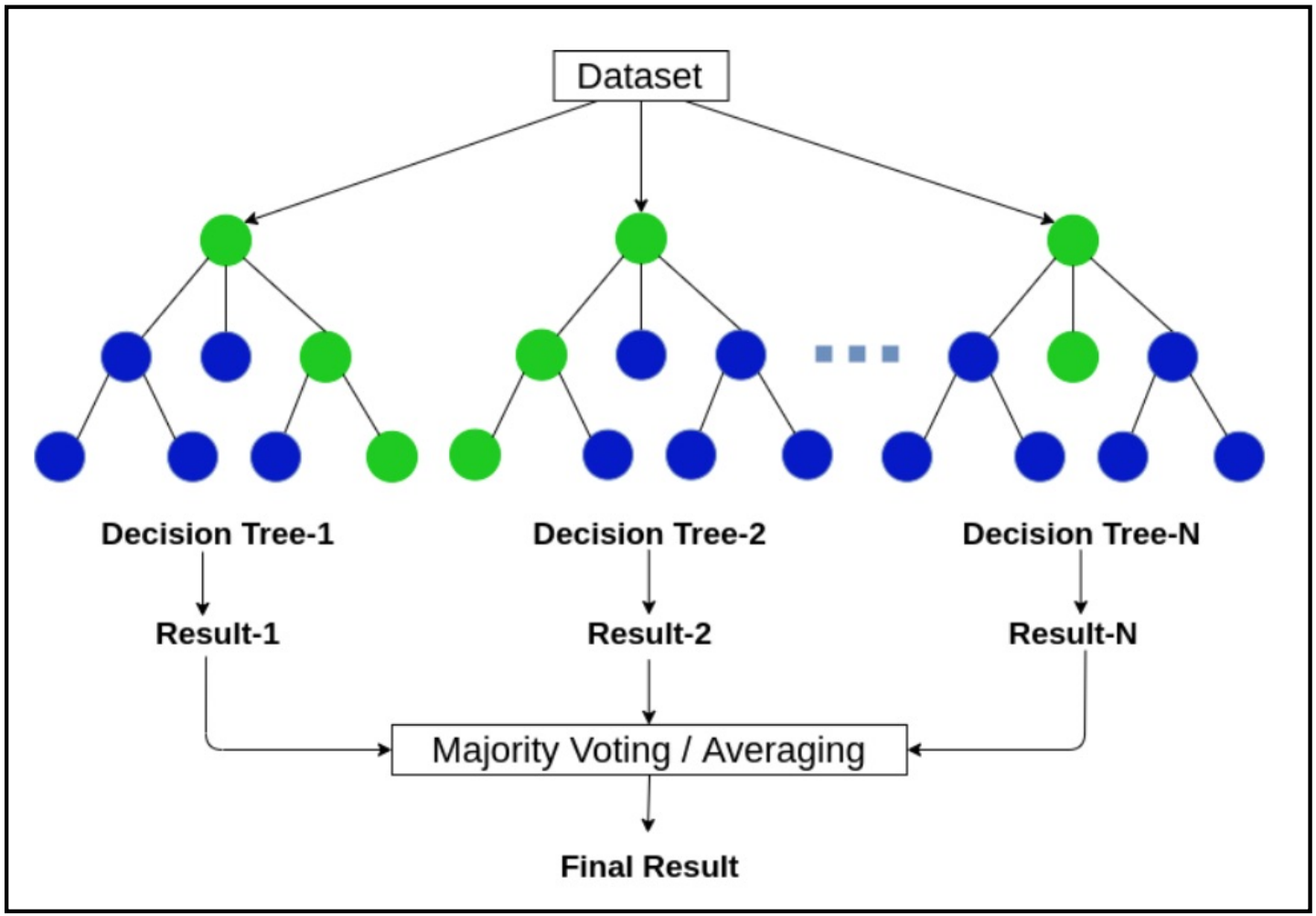
**Random Forest** (Breiman, 2001).

- Randomly pick features and data to generate diversity of classifiers (decision trees).



# Ensemble Techniques (Random Forest)

**Random Forest** (Breiman, 2001).



# Metrics of Evaluation

# Metrics of Evaluation

## Cross Validation Model

Cross validation is a method for assessing how the results of a data mining (statistical) analysis will generalize to an independent dataset. It is mainly used in predictive model applications.

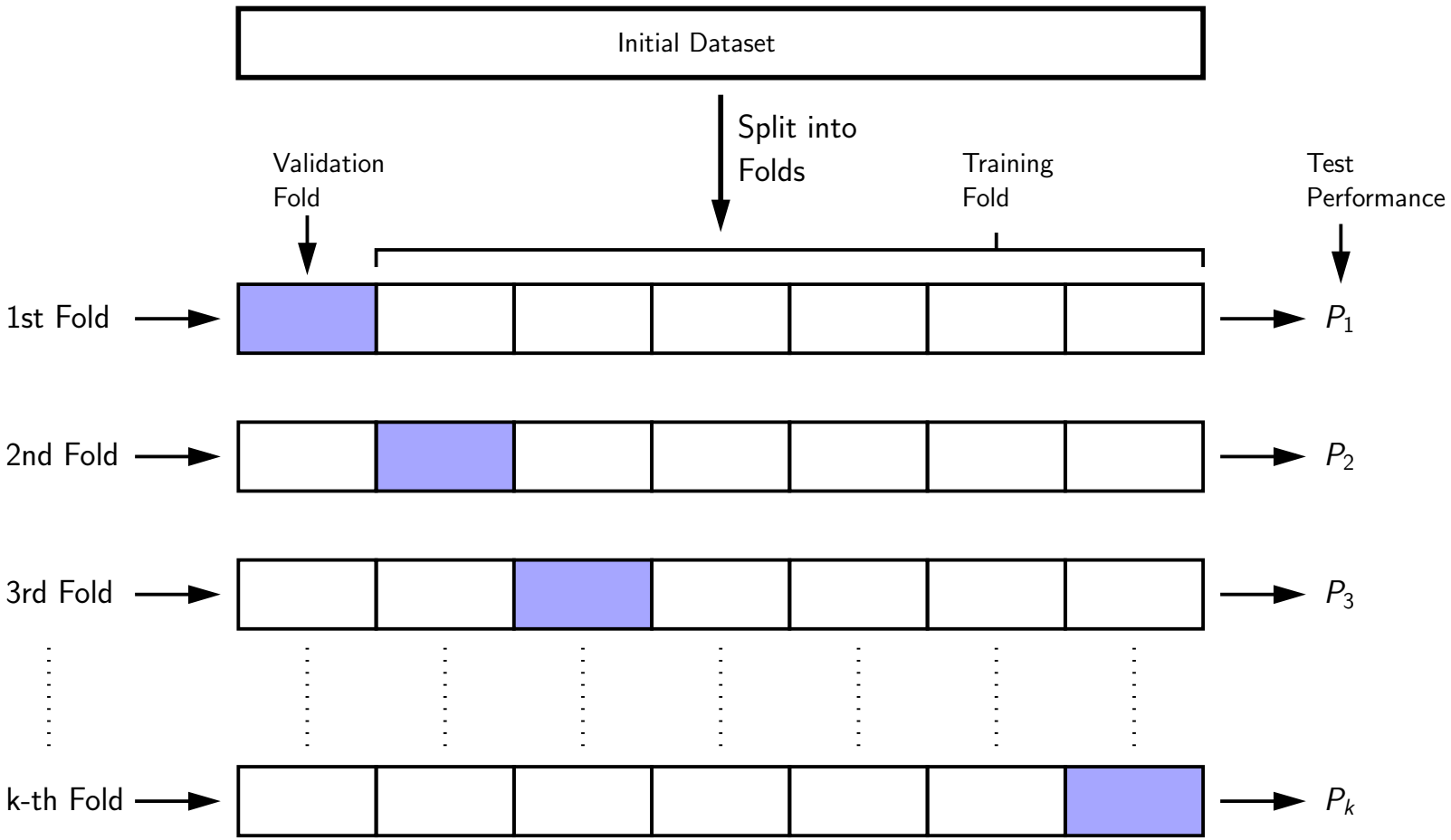
### K-Fold Cross Validation Method

- Divide the sample data into $k$ equal parts.
- Use $k-1$ parts for training and one for testing.
- Repeat the procedure $k$ times, rotating the test dataset.
- Compute metrics of performance across the iterations, i.e.,

$$\text{Performance} = \sum_{i=1}^{k} P_i. \tag{24}$$

# Metrics of Evaluation

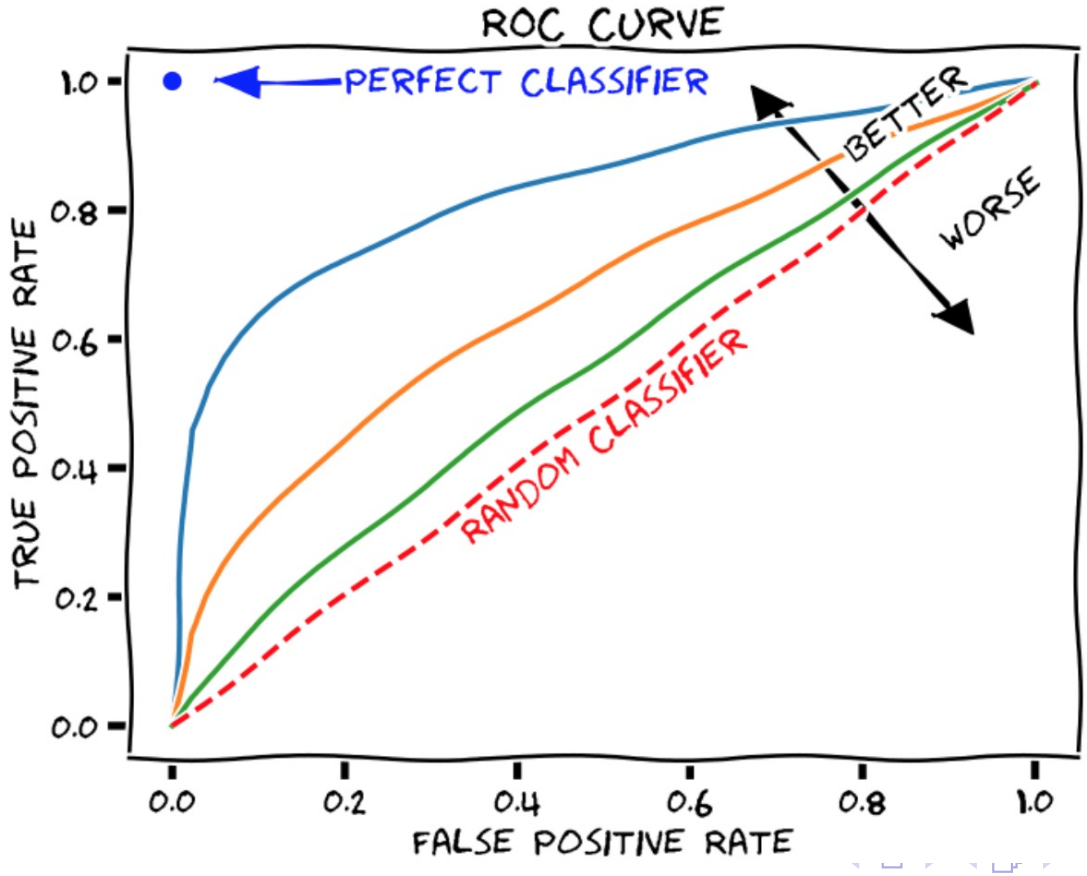
## K-Fold Cross Validation



# Metrics of Evaluation

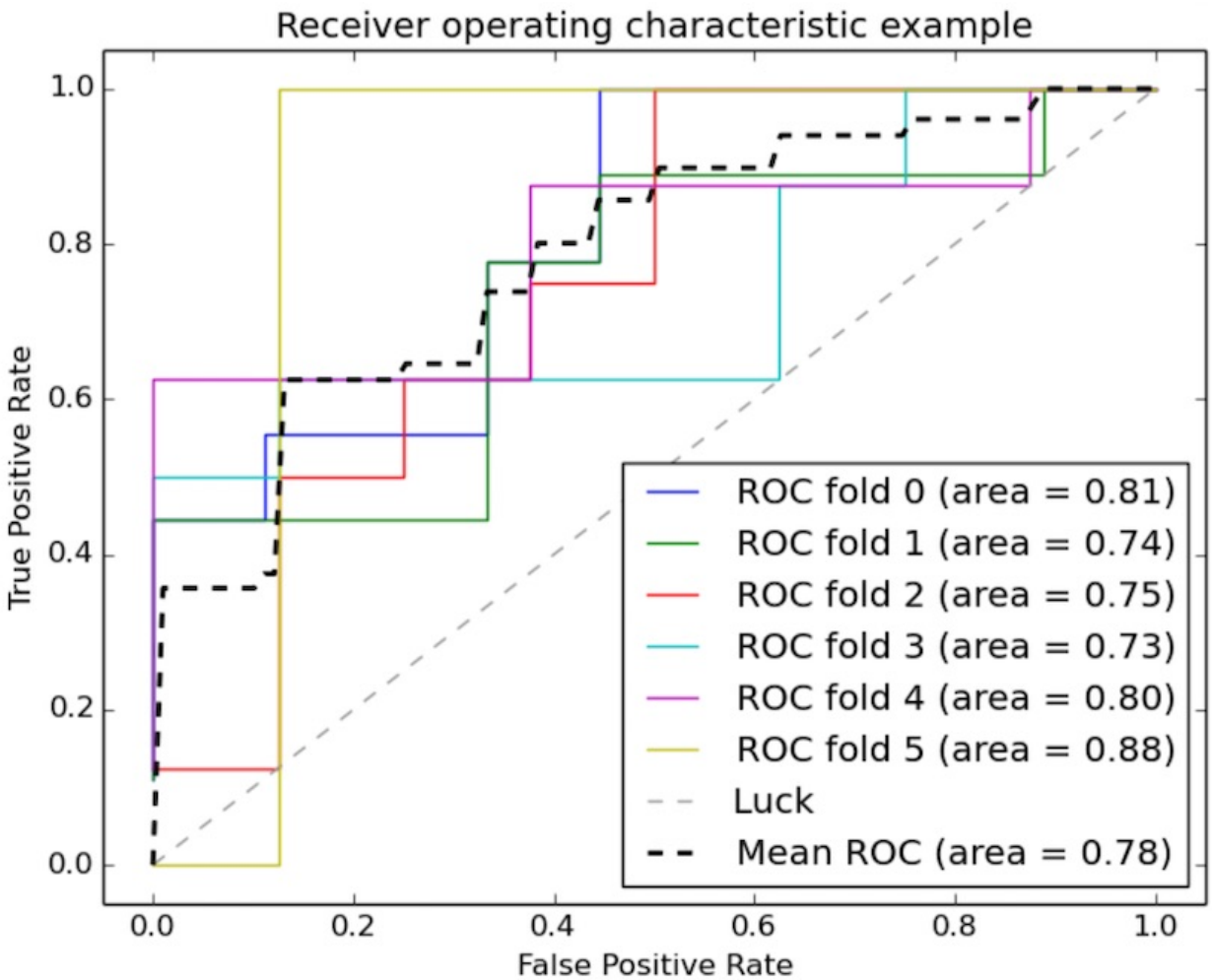
## Receiver Operating Curve

A receiver operating curve (ROC) illustrates diagnostic ability of a binary classifier as its discrimination threshold is varied.



# Metrics of Evaluation

## Typical ROC Curves



# References


- Jaynes E.T., Information Theory and Statistical Mechanics. II, Phys. Rev. 108, 171, October 1957.
- Kapur J.N., Maximum-Entropy Models in Science and Engineering, John Wiley and Sons, 1989.
- Mitchell T.M., Machine Learning and Data Mining, Communications of the ACM, Vol. 42., No. 11, November 1999.
- Russell S., and Norvig P., Artificial Intelligence: A Modern Approach (Third Edition), Prentice-Hall, 2010.
- Shanon C.E., and Weaver W., The Mathematical Theory of Communication, University of Illinois, Urbana, Chicago, 1949.
- Yeh I.C., Department of Information Management, Chung-Hua University, Hsin Chu, Taiwan 30067, R.O.C, Donated: August 2007.
- Witten I.H., Frank E., Hall M.A., and Pal C.J., Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2017.

# Appendix A

# Scikit-Learn (SkLearn) Library

# for Data Mining/ML

# SkLearn Library for Data Mining/ML

**Data Preprocessing**

1. StandardScaler()
2. MinMaxScaler()
3. RobustScaler()
4. Normalizer()
5. Binarizer()
6. LabelEncoder()
7. OneHotEncoder()
8. PolynomialFeatures()
9. SimpleImputer()
10. KNNImputer()

**Model Selection**

1. train_test_split()
2. cross_val_score()
3. GridSearchCV()
4. RandomizedSearchCV()
5. StratifiedKFold()
6. KFold()
7. RepeatedKFold()
8. RepeatedStratifiedKFold()
9. LeaveOneOut()
10. cross_val_predict()

**Classification Algorithms**

1. accuracy_score()
2. confusion_matrix()
3. classification_report()
4. roc_auc_score()
5. precision_score()
6. recall_score()
7. f1_score()
8. mean_squared_error()
9. mean_absolute_error()
10. r2_score()

**Classification Algorithms**

1. LogisticRegression()
2. KNeighborsClassifier()
3. DecisionTreeClassifier()
4. RandomForestClassifier()
5. SVC()
6. NaiveBayes()
7. GradientBoostingClassifier()
8. AdaBoostClassifier()
9. XGBClassifier()
10. LGBMClassifier()

**Regression Algorithms**

1. LinearRegression()
2. Ridge()
3. Lasso()
4. ElasticNet()
5. DecisionTreeRegressor()
6. RandomForestRegressor()
7. GradientBoostingRegressor()
8. SVR()
9. XGBRegressor()
10. LGBMRegressor()

# Appendix B

# Data Mining with Python

# Example 1. Will Customer Buy Computer?

**Sample Dataset.** Will customer buy a computer?

| ID | Age Group | Income | Student | Credit Rating | Buys Computer |
|---|---|---|---|---|---|
| 1 | young | high | no | fair | no |
| 2 | young | high | no | excellent | no |
| 3 | middle | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle | low | yes | excellent | yes |
| 8 | young | medium | no | fair | no |
| 9 | young | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | young | medium | yes | excellent | yes |
| 12 | middle | medium | no | excellent | yes |
| 13 | middle | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Example 1. Will Customer Buy Computer?

**Retail Dataset:** Information on 14 customers.

See: python-code-ai.d/data/retail/computer-sales.csv

**Data Attributes:**

- ID: Customer ID ...
- Age Group: young, medium, senior ...
- Income: low, medium, high ...
- Student: yes or no.
- Credit Rating: fair, excellent.
- Buys Computer: yes, no.

# Example 1. Will Customer Buy Computer?

## Load Data into Pandas:

```
dataFile = "../data/retail/computer-sales.csv";
df01 = pd.read_csv( dataFile )
```

## Change Column Names:

```
columns01 = { 'Age Group':'Age', 'Income':'Income', 'Student':'Student',
              'Credit Rating':'Credit', 'Buys Computer':'Buys' }

df01 = df01.rename( columns=columns01)
```

## Set Individual Column Types to Categorical:

```
df01['Age']     = df01['Age'].astype("category")
df01["Income"]  = df01["Income"].astype("category")
df01["Student"] = df01["Student"].astype("category")
df01["Credit"]  = df01["Credit"].astype("category")
df01["Buys"]    = df01["Buys"].astype("category")
```

# Example 1. Will Customer Buy Computer?

## Convert Categorical Columns to Numerical Data:

```
df01['Age']     = df01['Age'].cat.codes
df01['Income']  = df01['Income'].cat.codes
df01['Student'] = df01['Student'].cat.codes
df01['Credit']  = df01['Credit'].cat.codes
df01['Buys']    = df01['Buys'].cat.codes
```

## Transformed Dataset:

| | ID | Age | Income | Student | Credit | Buys |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 2 | 0 | 1 | 0 | 1 | 0 |
| 2 | 3 | 1 | 1 | 0 | 0 | 1 |
| 3 | 4 | 2 | 2 | 0 | 0 | 1 |
| 4 | 5 | 2 | 0 | 1 | 0 | 1 |
| 5 | 6 | 2 | 0 | 1 | 1 | 0 |
| 6 | 7 | 1 | 0 | 1 | 1 | 1 |
| 7 | 8 | 0 | 2 | 0 | 0 | 0 |
| 8 | 9 | 0 | 0 | 1 | 0 | 1 |
| 9 | 10 | 2 | 2 | 1 | 0 | 1 |
| 10 | 11 | 0 | 2 | 1 | 1 | 1 |
| 11 | 12 | 1 | 2 | 0 | 1 | 1 |
| 12 | 13 | 1 | 1 | 1 | 0 | 1 |
| 13 | 14 | 2 | 2 | 0 | 1 | 0 |

# Example 1. Will Customer Buy Computer?

## Assemble X and Y datasets:

```
X = df01 [['Age', 'Income', 'Student', 'Credit' ]]
Y = df01 ['Buys'].values.tolist()
```

## Data Input X: (14 rows)

| | Age | Income | Student | Credit |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 2 | 2 | 0 | 0 |
| 4 | 2 | 0 | 1 | 0 |
| 5 | 2 | 0 | 1 | 1 |
| 6 | 1 | 0 | 1 | 1 |
| 7 | 0 | 2 | 0 | 0 |
| 8 | 0 | 0 | 1 | 0 |
| 9 | 2 | 2 | 1 | 0 |
| 10 | 0 | 2 | 1 | 1 |
| 11 | 1 | 2 | 0 | 1 |
| 12 | 1 | 1 | 1 | 0 |
| 13 | 2 | 2 | 0 | 1 |

## Target Values: Y

```
[0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0]
```

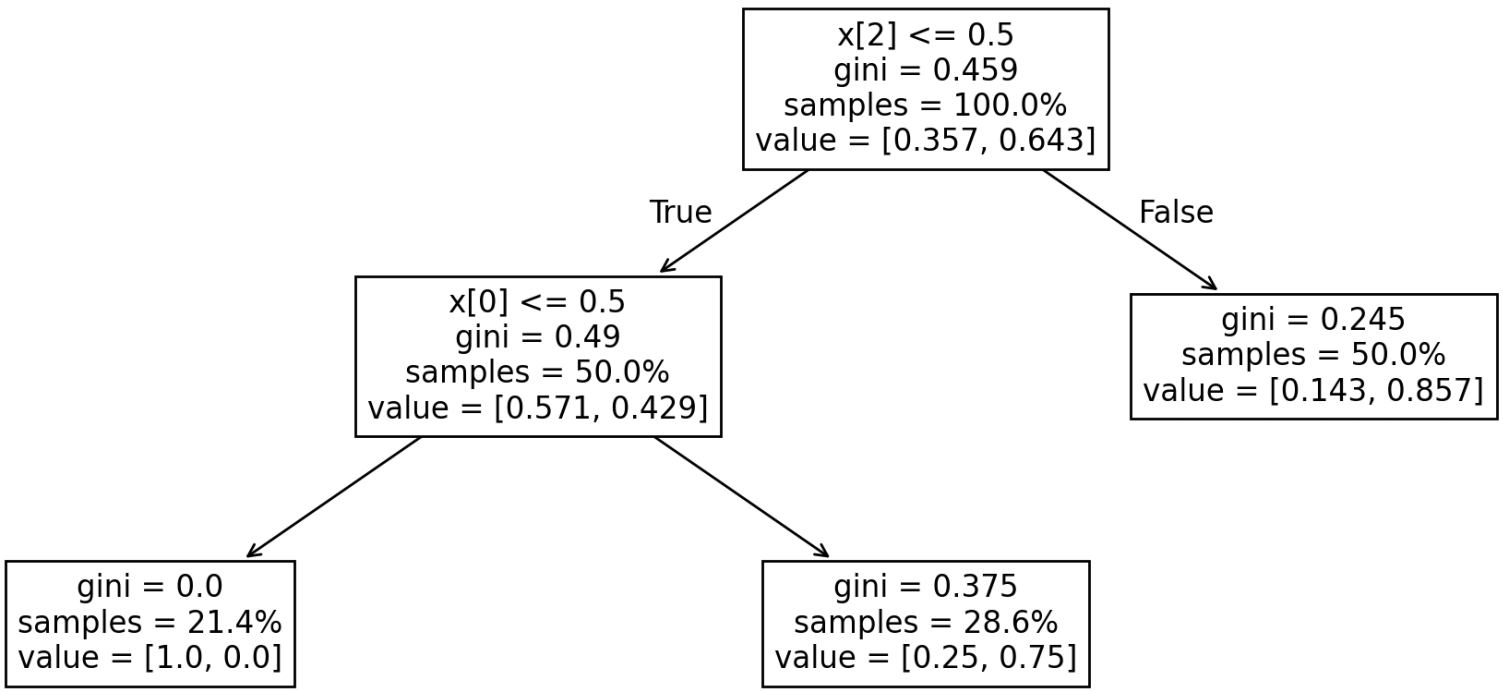
# Example 1. Will Customer Buy Computer?

## **Decision Tree:** (max_depth = 2)

```
Root (GINI impurity: 0.46; observations: {0: 5, 1: 9}: predicted: 1)

|--- Split rule: Student <= 0.5 (GINI impurity: 0.49; observations: {0: 4, 1: 3}; predictions: 0)

|------ Split rule: Age <= 0.5  (GINI impurity: 0.00; observations: {0: 3}; predictions: 0)
|------ Split rule: Age >  0.5  (GINI impurity: 0.38; observations: {1: 3, 0: 1}; | predictions: 1)

|--- Split rule: Student > 0.5  (GINI impurity: 0.24; observations: {1: 6, 0: 1}; predictions: 1)

|------ Split rule: Age <= 1.5  (GINI impurity: 0.00; observations: {1: 4}; | predictions: 1)
|------ Split rule: Age > 1.5   (GINI impurity: 0.44; observations: {1: 2, 0: 1}; predictions: 1)
```

# Example 1. Will Customer Buy Computer?

**Hierarchical Tree Structure:**



# Example 1. Will Customer Buy Computer?

**Use Decision Tree to make Predictions:**

| | Age | Income | Student | Credit | yhat |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 1 | 1 | 0 | 0 | 1 |
| 3 | 2 | 2 | 0 | 0 | 1 |
| 4 | 2 | 0 | 1 | 0 | 1 |
| 5 | 2 | 0 | 1 | 1 | 1 |
| 6 | 1 | 0 | 1 | 1 | 1 |
| 7 | 0 | 2 | 0 | 0 | 0 |
| 8 | 0 | 0 | 1 | 0 | 1 |
| 9 | 2 | 2 | 1 | 0 | 1 |
| 10 | 0 | 2 | 1 | 1 | 1 |
| 11 | 1 | 2 | 0 | 1 | 1 |
| 12 | 1 | 1 | 1 | 0 | 1 |
| 13 | 2 | 2 | 0 | 1 | 1 |

# Example 1. Will Customer Buy Computer?

**Summary:** How well does it work?

```
--- Sum( yvalues)     =    9.00 ...
--- Sum( yhatvalues) =   11.00 ...

--- Sum( abs(difference) ) =    2.00 ...

--- Compare columns: buys vs yhat ...

    self  other
5    0.0    1.0
13   0.0    1.0
```

# Example 2. Predict Casualties in Sinking of Titanic

**Problem Setup:** On April 12, 1912, the RMS Titanic (mail/passenger vessel) commenced her maiden voyage across the Atlantic, departing from Ireland and headed to New York.

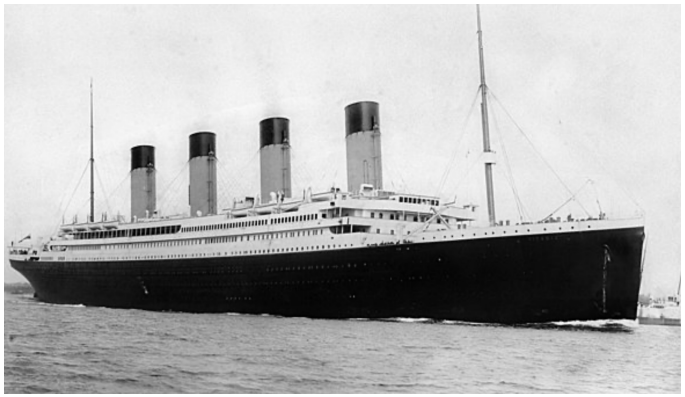
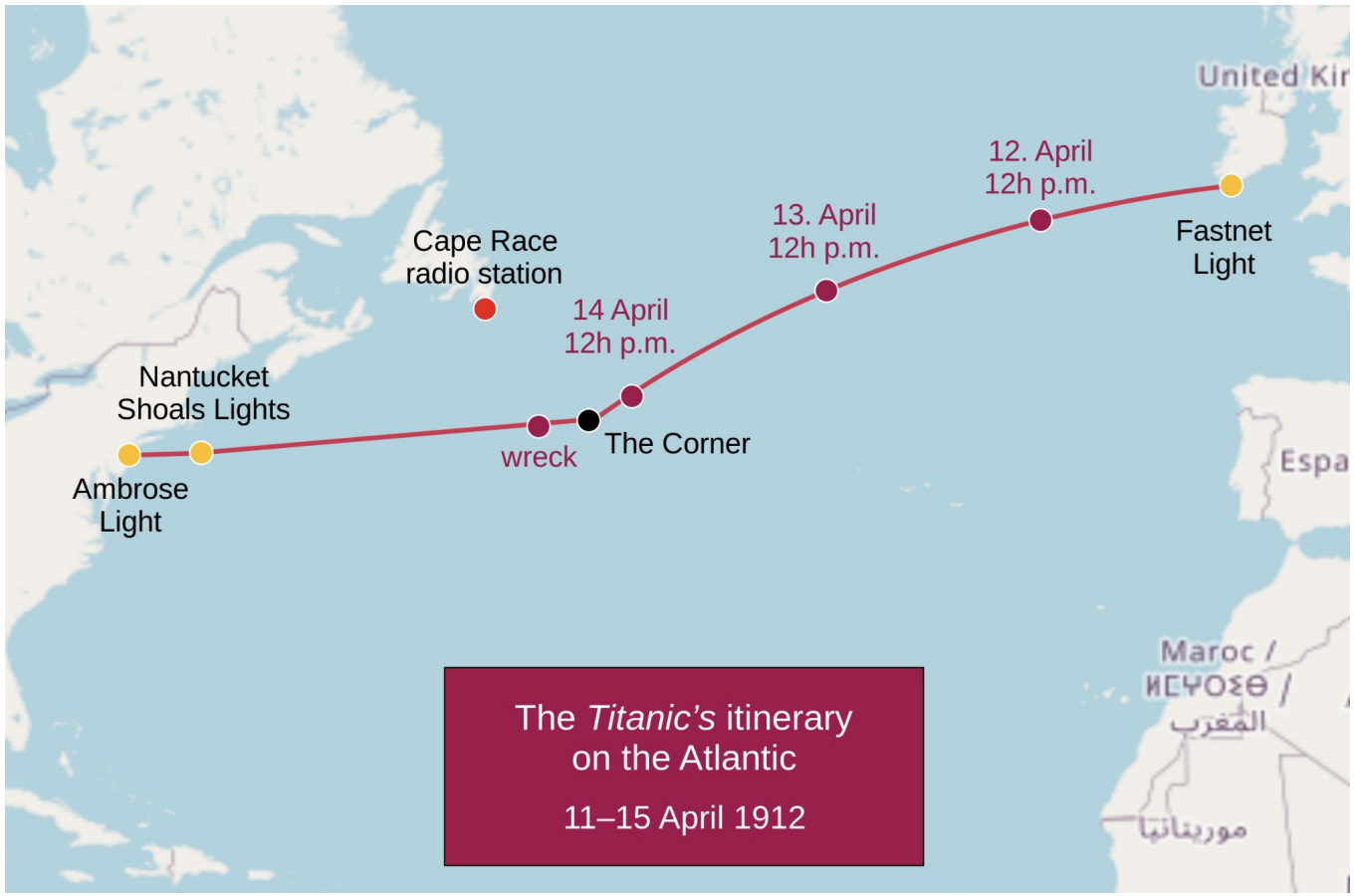


Figure: Titanic departing Southampton, April 10, 1912.

Three days later (April 15) the Titanic struck an iceberg and sank. Of the 2,224 passengers and crew aboard, approximately 1,500 died, making it the deadliest sinking of a ship at that time.

# Example 2. Predict Casualties in Sinking of Titanic

**Trans-Atlantic Route:** From Ireland to New York ...



# Example 2. Predict Casualties in Sinking of Titanic

**Raw Dataset:** Contains information on 891 passengers.

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S

....

884,0,2,"Banfield, Mr. Frederick James",male,28,0,0,C.A./SOTON 34068,10.5,,S
885,0,3,"Sutehall, Mr. Henry Jr",male,25,0,0,SOTON/OQ 392076,7.05,,S
886,0,3,"Rice, Mrs. William (Margaret Norton)",female,39,0,5,382652,29.125,,Q
887,0,2,"Montvila, Rev. Juozas",male,27,0,0,211536,13,,S
888,1,1,"Graham, Miss. Margaret Edith",female,19,0,0,112053,30,B42,S
889,0,3,"Johnston, Miss. Catherine Helen ""Carrie""",female,,1,2,W./C. 6607,23.45,,S
890,1,1,"Behr, Mr. Karl Howell",male,26,0,0,111369,30,C148,C
891,0,3,"Dooley, Mr. Patrick",male,32,0,0,370376,7.75,,Q
```

**Source:** python-code-ai.d/data/disaster/titanic-train.csv

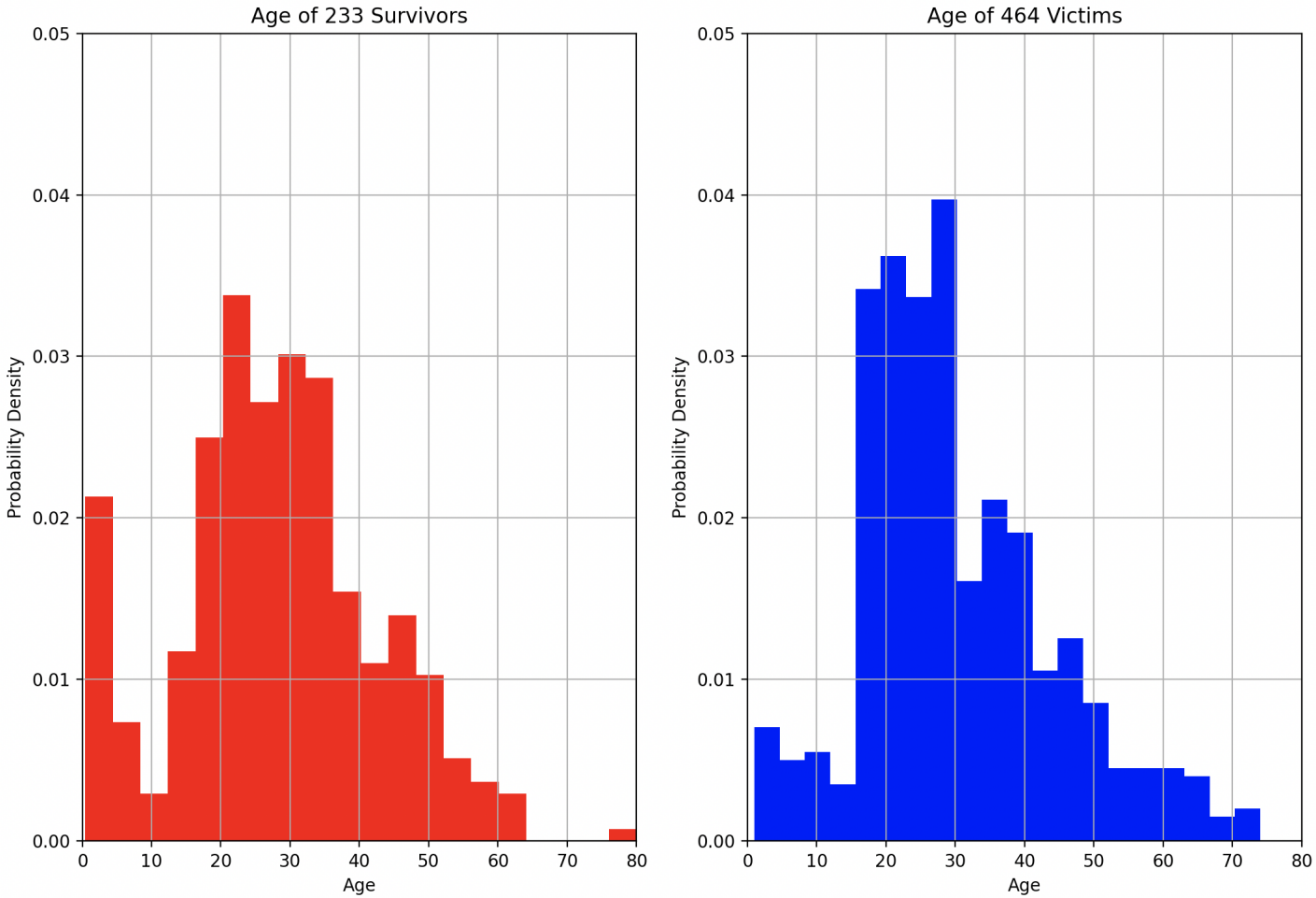
# Example 2. Predict Casualties in Sinking of Titanic

**Data Attributes:**

- Survived: 1 means passenger survived; 0 for victims.
- Pclass: 1, 2 and 3 for first, second and third class.
- Name: Master/miss first name, family name.
- Sex: male or female.
- Age: Covers the range 0 to 80.
- Siblings/Spouses Aboard
- Parents/Children Aboard
- Fare: First class (1) tickets are the most expensive.

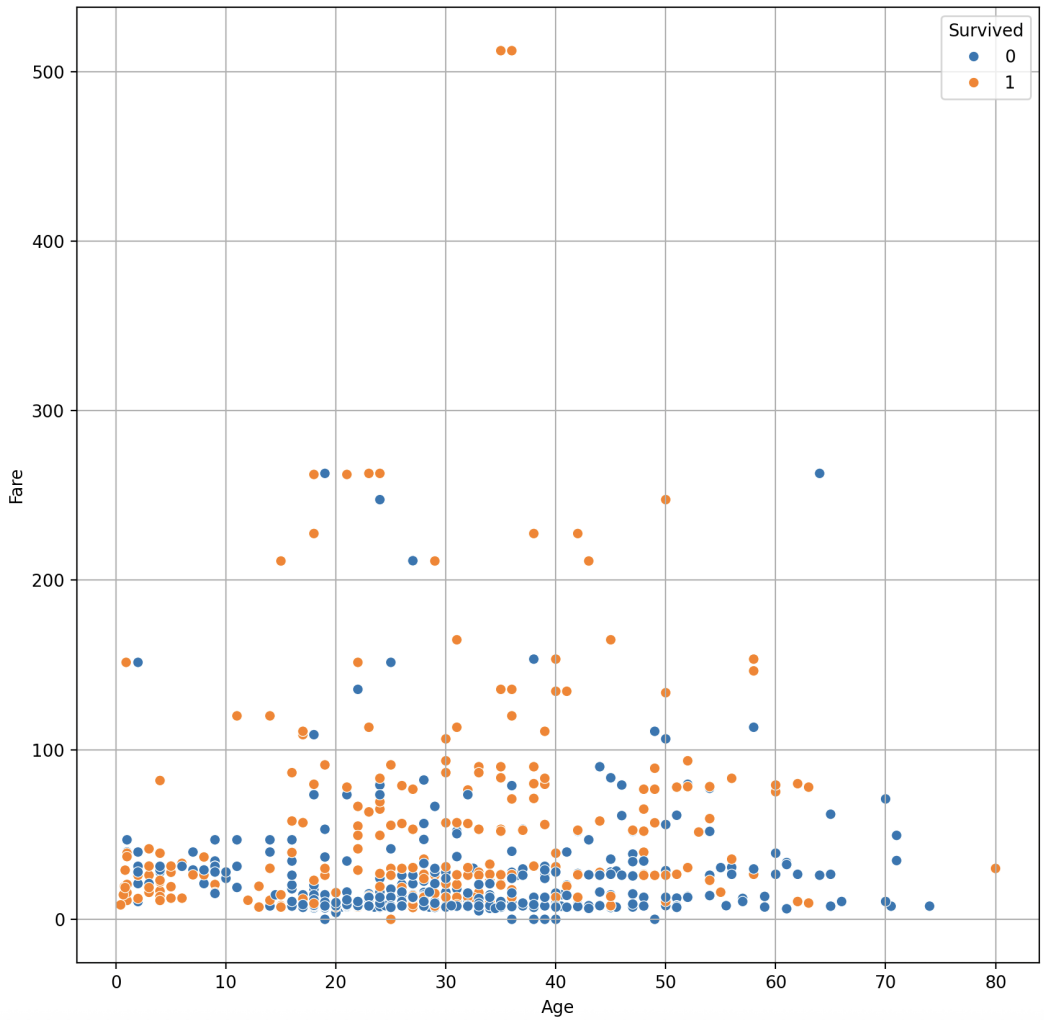
# Example 2. Predict Casualties in Sinking of Titanic

## Age Distribution of Survivors/Victims:



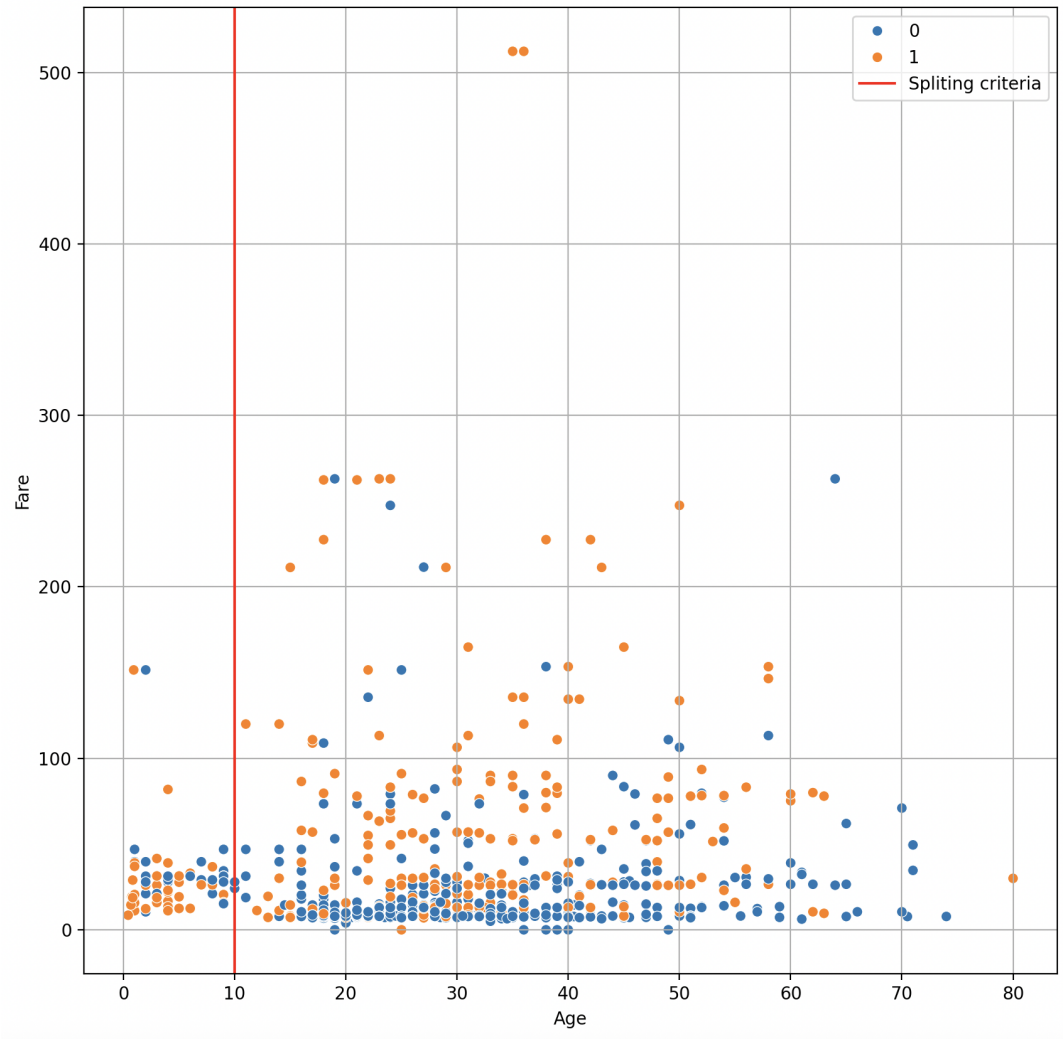
# Example 2. Predict Casualties in Sinking of Titanic

**Preliminary Investigation:** Plot fare vs passenger age.



# Example 2. Predict Casualties in Sinking of Titanic

**Preliminary Investigation:** Split data at age = 10.



# Example 2. Predict Casualties in Sinking of Titanic

## Read Training data into dataframe:

```
dataFile = "../data/disaster/titanic-train.csv";
d1 = pd.read_csv( dataFile )
```

## Filter Data on Age, Fare, Sex and Survived Passengers:

```
d2 = d1[['Age', 'Fare', 'Sex', 'Survived']].dropna()
```

## Manually Convert 'Sex' column to numerical data:

```
d2['Sex']      = d2['Sex'].astype('category')
d2['Sex_Code'] = d2['Sex'].cat.codes
```

## Decision Analysis:

```
X = d2 [['Age', 'Sex_Code', 'Fare']]
Y = d2 ['Survived'].values.tolist()
root = Node(Y, X, max_depth=3, min_samples_split=100)
```

# Example 2. Predict Casualties in Sinking of Titanic

## Data Input X: (714 rows)

| | Age | Sex_Code | Fare |
|---|---|---|---|
| 0 | 22.0 | 1 | 7.2500 |
| 1 | 38.0 | 0 | 71.2833 |
| 2 | 26.0 | 0 | 7.9250 |
| 3 | 35.0 | 0 | 53.1000 |
| 4 | 35.0 | 1 | 8.0500 |
| .. | ... | ... | ... |
| 885 | 39.0 | 0 | 29.1250 |
| 886 | 27.0 | 1 | 13.0000 |
| 887 | 19.0 | 0 | 30.0000 |
| 889 | 26.0 | 1 | 30.0000 |
| 890 | 32.0 | 1 | 7.7500 |

## Target Values: Y

```
[ 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, ....
  0, 0, 1, 1, 0 ]
```

## Basic Passenger Statistics: (Cleaned Data) ...

```
--- No passengers =  714 ...
--- No survivors  =  290 ...
--- No victims    =  424 ...

--- Min fare      =   0.00 dollars ...
--- Max fare      = 512.33 dollars ...
--- Average fare =  34.69 dollars ...

--- Min age       =   0.42 years ...
--- Max age       =  80.00 years ...
--- Average age   =  29.70 years ...
```

# Example 2. Predict Casualties in Sinking of Titanic

## **Decision Tree Output:** Max depth = 3

```
Root (GINI impurity: 0.48; observations {0: 424, 1: 290}; prediction: 0)

|-- Split rule: Sex_Code <= 0.5 (GINI impurity: 0.37; observations {1: 197, 0: 64}; prediction: 1)

|----- Split rule: Fare <= 48.2 (GINI impurity: 0.45; observations {1: 122, 0: 62} prediction: 1)
|-------- Split rule: Fare <= 10.481 (GINI impurity: 0.50; observations {1: 22, 0: 21}; prediction: 1
|-------- Split rule: Fare >  10.481 (GINI impurity: 0.41; observations {1: 100, 0: 41; prediction: 1

|----- Split rule: Fare > 48.2 (GINI impurity: 0.05; observations {1: 75, 0: 2}; prediction: 1)

|-- Split rule: Sex_Code > 0.5 (GINI impurity: 0.33; observations {0: 360, 1: 93}: prediction: 0)

|----- Split rule: Age <= 6.5 (GINI impurity: 0.44; observations {0: 8, 1: 16}; prediction: 1)

|----- Split rule: Age >  6.5 (GINI impurity: 0.29; observations {0: 352, 1: 77}; prediction: 0)
|-------- Split rule: Fare <= 26.269 (GINI impurity: 0.20; observations {0: 27, 1: 35}; prediction: 0)
|-------- Split rule: Fare >  26.269 (GINI impurity: 0.45; observations {0: 83, 1: 42}; prediction: 0)
```

Key takeaway:

- Women and children had the greatest chance of survival – because they got to go in the lifeboats.

# Example 2. Predict Casualties in Sinking of Titanic

## Use Decision Tree to make Predictions:

| | Age | Sex_Code | Fare | yhat |
|---|---|---|---|---|
| 0 | 22.0 | 1 | 7.2500 | 0 |
| 1 | 38.0 | 0 | 71.2833 | 1 |
| 2 | 26.0 | 0 | 7.9250 | 1 |
| 3 | 35.0 | 0 | 53.1000 | 1 |
| 4 | 35.0 | 1 | 8.0500 | 0 |
| .. | ... | ... | ... | ... |
| 885 | 39.0 | 0 | 29.1250 | 1 |
| 886 | 27.0 | 1 | 13.0000 | 0 |
| 887 | 19.0 | 0 | 30.0000 | 1 |
| 889 | 26.0 | 1 | 30.0000 | 0 |
| 890 | 32.0 | 1 | 7.7500 | 0 |

[714 rows x 4 columns]

## How well does it work?

```
--- Sum( yvalues)          =  290.00 ...
--- Sum( yhatvalues)       =  285.00 ...
--- Sum( abs(difference) ) =  149.00 ...
```

# Example 3. Predict Compressive Strength of Concrete

**Problem Statement:** Use regression tree analysis to predict the compressive strength of concrete (MPa), a nonlinear function of age (days) and ingredients.

---

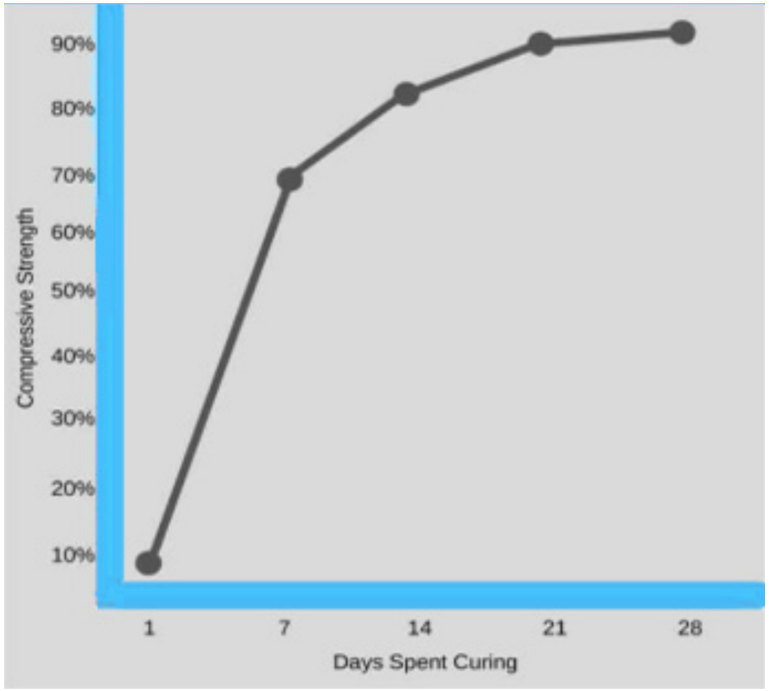
**Laboratory Dataset:**

- Number of instances (observations): 1030
- Number of Attributes: 9
- Attribute breakdown: 8 quantitative input variables, and 1 quantitative output variable

**Reference:** Yeh I.C., Department of Information Management, Chung-Hua University, Hsin Chu, Taiwan 30067, R.O.C, Donated: August 2007.

# Example 3. Predict Compressive Strength of Concrete

**From Google:** How long does it take concrete to cure?




**Rule of Thumb:** Wait 12-24 hrs to take foot traffic; $> 7$ days to take heavier traffic; full curing can take up to 28 days.

# Example 3. Predict Compressive Strength of Concrete

**Dataset:** 1030 entries:

| | Cement | Blast Furnace Slag | Fly Ash | ... | Fine Aggregate | Age | Strength |
|---|---|---|---|---|---|---|---|
| 0 | 540.0 | 0.0 | 0.0 | ... | 676.0 | 28 | 79.99 |
| 1 | 540.0 | 0.0 | 0.0 | ... | 676.0 | 28 | 61.89 |
| 2 | 332.5 | 142.5 | 0.0 | ... | 594.0 | 270 | 40.27 |
| 3 | 332.5 | 142.5 | 0.0 | ... | 594.0 | 365 | 41.05 |
| 4 | 198.6 | 132.4 | 0.0 | ... | 825.5 | 360 | 44.30 |
| 5 | 266.0 | 114.0 | 0.0 | ... | 670.0 | 90 | 47.03 |
| 6 | 380.0 | 95.0 | 0.0 | ... | 594.0 | 365 | 43.70 |
| 7 | 380.0 | 95.0 | 0.0 | ... | 594.0 | 28 | 36.45 |
| 8 | 266.0 | 114.0 | 0.0 | ... | 670.0 | 28 | 45.85 |
| 9 | 475.0 | 0.0 | 0.0 | ... | 594.0 | 28 | 39.29 |
| 10 | 198.6 | 132.4 | 0.0 | ... | 825.5 | 90 | 38.07 |
| 11 | 198.6 | 132.4 | 0.0 | ... | 825.5 | 28 | 28.02 |
| 12 | 427.5 | 47.5 | 0.0 | ... | 594.0 | 270 | 43.01 |
| 13 | 190.0 | 190.0 | 0.0 | ... | 670.0 | 90 | 42.33 |
| 14 | 304.0 | 76.0 | 0.0 | ... | 670.0 | 28 | 47.81 |
| 15 | 380.0 | 0.0 | 0.0 | ... | 670.0 | 90 | 52.91 |
| 16 | 139.6 | 209.4 | 0.0 | ... | 806.9 | 90 | 39.36 |
| 17 | 342.0 | 38.0 | 0.0 | ... | 670.0 | 365 | 56.14 |
| 18 | 380.0 | 95.0 | 0.0 | ... | 594.0 | 90 | 40.56 |
| 19 | 475.0 | 0.0 | 0.0 | ... | 594.0 | 180 | 42.62 |

....
(1030, 9)

**Source:** python-code-ai.d/data/materials/concrete-strength-data.csv

# Example 3. Predict Compressive Strength of Concrete

**Dataset Inputs:**

- Cement: kg in a m3 mixture
- Blast Furnace Slag: kg in a m3 mixture
- Fly Ash: kg in a m3 mixture
- Water: kg in a m3 mixture
- Superplasticizer: kg in a m3 mixture
- Coarse Aggregate: kg in a m3 mixture
- Fine Aggregate: kg in a m3 mixture
- Age: day (1–365)

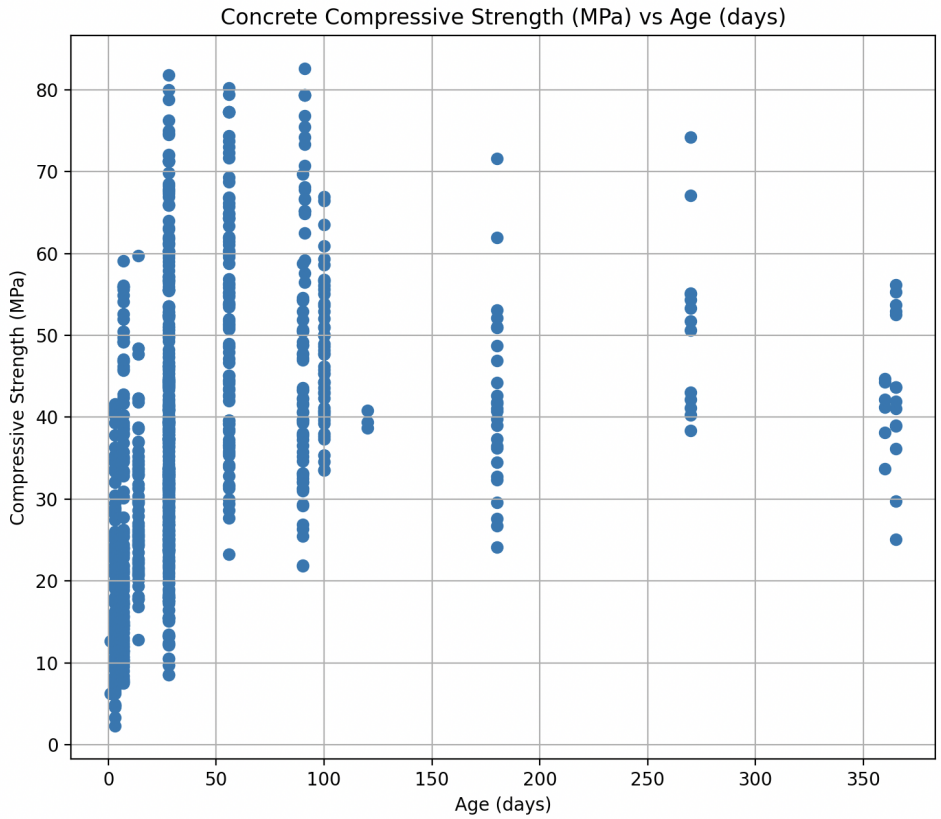
---

**Dataset Output**

- Compressive Strength: MPa.

# Example 3. Predict Compressive Strength of Concrete

**Scatter Chart:** Strength (MPa) vs Age (days) ...




**Statistics:** (MSE 278.81; observations 1030; prediction 35.818).

# Example 3. Predict Compressive Strength of Concrete

**Split Data:** 80% (824 items) for training; 20% (206) for testing.

**X_train shape:** (824, 8) ...

| | Cement | Blast Furnace Slag | Fly Ash | ... | Coarse Aggregate | Fine Aggregate | Age |
|---|---|---|---|---|---|---|---|
| 248 | 238.1 | 0.0 | 94.1 | ... | 949.9 | 847.0 | 100 |
| 756 | 540.0 | 0.0 | 0.0 | ... | 1125.0 | 613.0 | 270 |
| 144 | 475.0 | 118.8 | 0.0 | ... | 852.1 | 781.5 | 56 |
| .. | ... | ... | ... | ... | ... | ... | ... |
| 785 | 331.0 | 0.0 | 0.0 | ... | 978.0 | 825.0 | 7 |
| 749 | 500.0 | 0.0 | 0.0 | ... | 1125.0 | 613.0 | 14 |
| 898 | 336.0 | 0.0 | 0.0 | ... | 986.0 | 817.0 | 28 |

[824 rows x 8 columns]

**y_train shape:** (824,) ...

| | Strength |
|---|---|
| 248 | 44.30 |
| 756 | 74.17 |
| 144 | 72.30 |
| | ..... |
| 785 | 16.26 |
| 749 | 36.94 |
| 898 | 44.86 |

# Example 3. Predict Compressive Strength of Concrete

**Source:** python-code-ai.d/regression/Test-Regression-Materials02.py

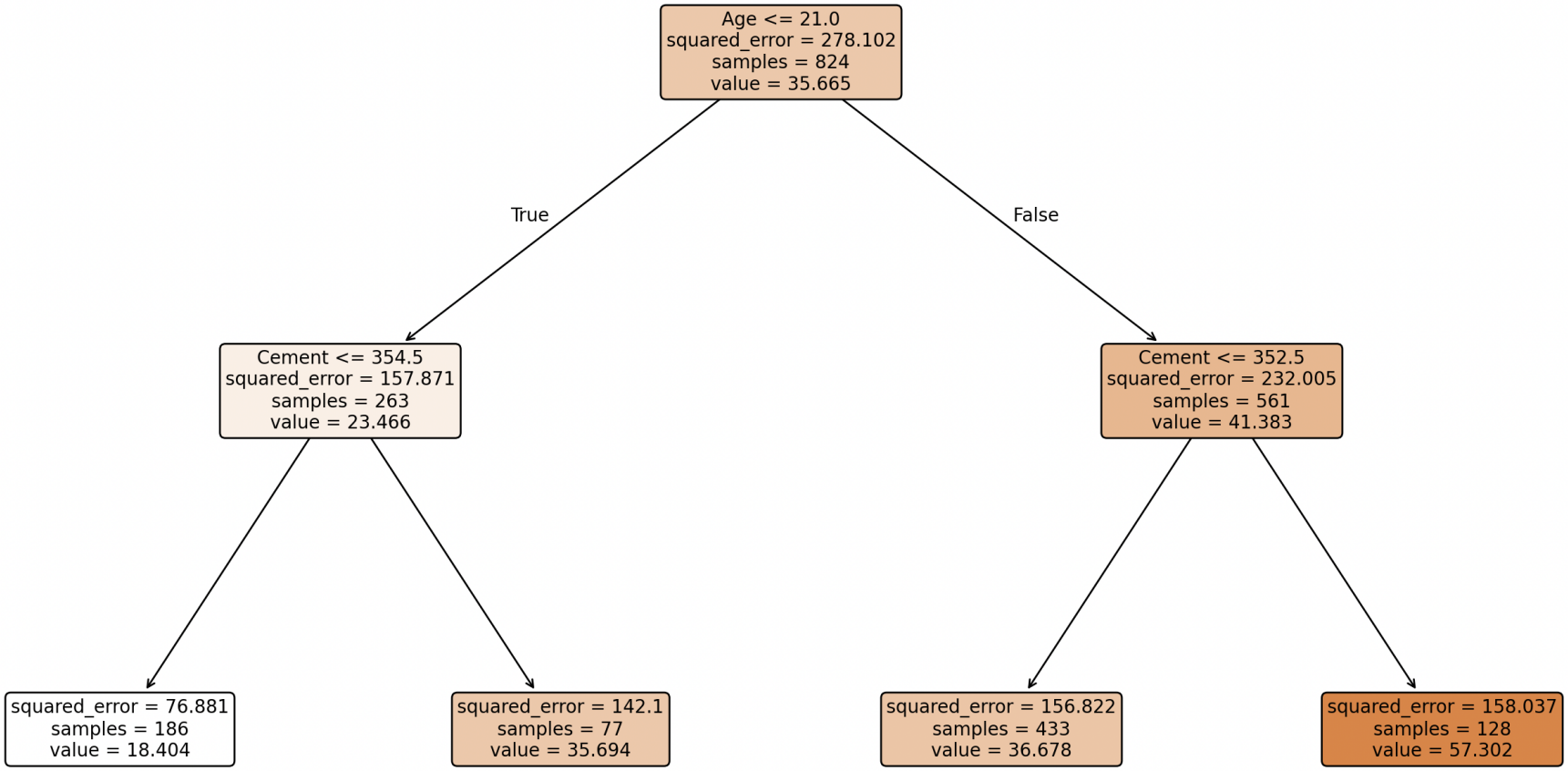
---

## Decision Tree Output: Max depth = 2

```
Root (MSE 278.10; observations 824; prediction 35.66)

|-- Split rule: Age <= 21.00 (MSE 157.8; observations 263; prediction 23.46)

|-------- Split rule: Cement <= 354.50 (MSE  76.8; observations 186; prediction 18.82)
|-------- Split rule: Cement >  354.50 (MSE 142.1; observations  77; prediction 36.12)

|-- Split rule: Age >  21.00 (MSE 232.0; observations 561; prediction 41.38)

|-------- Split rule: Cement <= 357.50 (MSE 156.8; observations 433; prediction 36.65)
|-------- Split rule: Cement >  357.50 (MSE 158.0; observations 128; prediction 57.43)
```

# Example 3. Predict Compressive Strength of Concrete

**Decision Tree:** Max depth = 2



# Example 3. Predict Compressive Strength of Concrete

**Summary:** 206 Predictions:

```
--- Index  Strength (MPa)  Prediction (MPa)  Error (MPa) ...
--- ================================================== ...
---   765           23.22             36.12        12.90 ...
---   773           37.42             57.43        20.01 ...
---   709           32.10             36.65         4.55 ...
---   661           10.39             18.82         8.43 ...
---   794           26.74             36.65         9.91 ...
---     1           61.89             57.43        -4.46 ...
---   820           67.11             57.43        -9.68 ...
---   748           33.21             36.12         2.91 ...

......

---   778           14.80             18.82         4.02 ...
---   640           30.14             36.12         5.98 ...
---   341           47.40             36.65       -10.75 ...
---   524           59.49             57.43        -2.06 ...
---   420           33.09             18.82       -14.27 ...
---   990           36.35             36.65         0.30 ...
--- ================================================== ...
--- Error Min Value: 0.21
--- Error Max Value: 36.34
--- Error Standard Deviation: 7.44
--- Mean Squared Error: 153.35
--- Root Mean Squared Error: 12.38
--- ================================================== ...
```

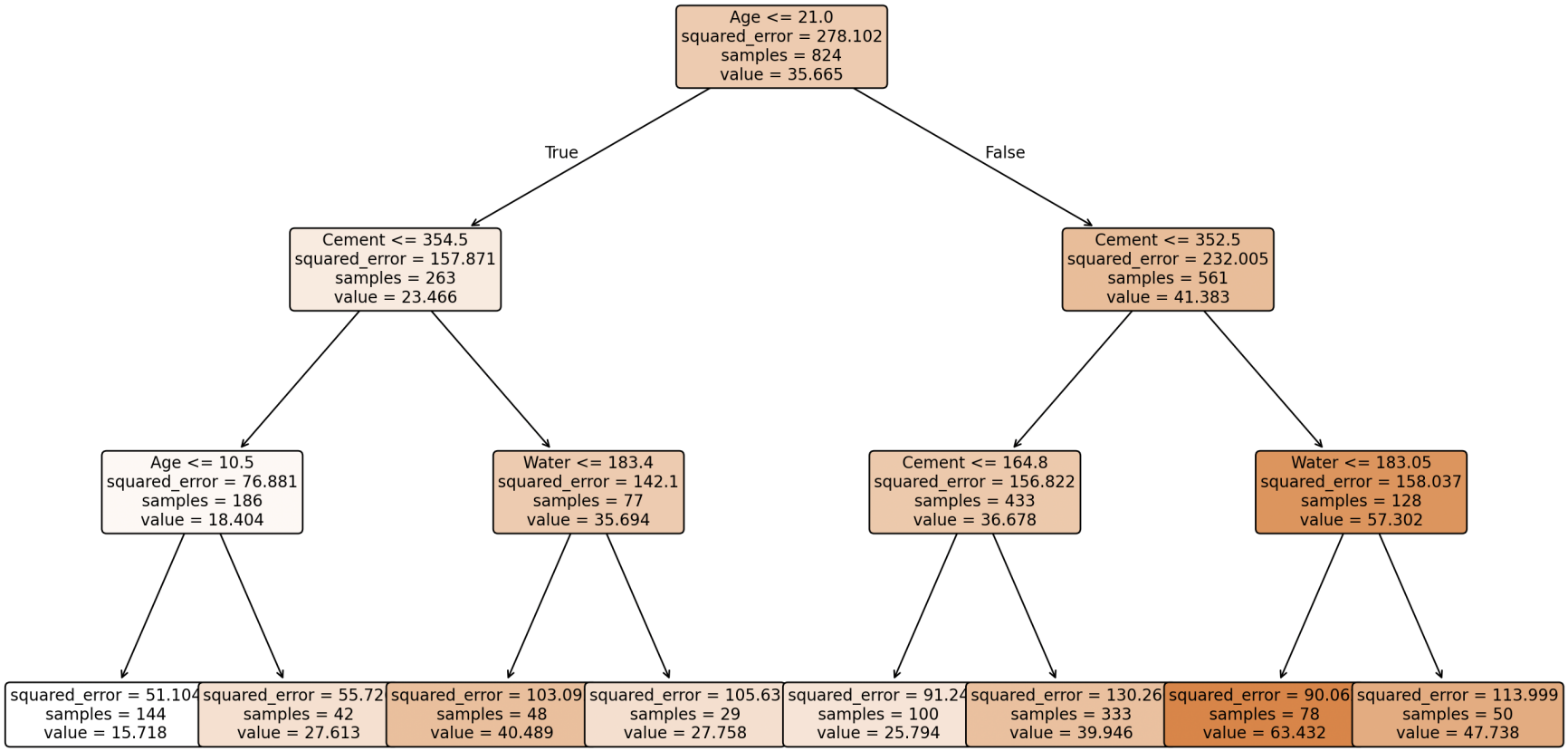
# Example 3. Predict Compressive Strength of Concrete

## **Decision Tree Output:** Max depth = 3

```
Root (MSE 278.10; observations 824; prediction 35.66)

|-- Split rule: Age <= 21.00 (MSE 157.87; observations 263; prediction 23.46)

|------ Split rule:  Cement <= 354.50 (MSE 76.88)

|---------- Split rule: Age <= 10.50 (MSE 51.10; observations 144; prediction 15.71)
|---------- Split rule: Age >  10.50 (MSE 55.72; observations  42; prediction 27.61)

|------ Split rule: Cement >  354.50 (MSE 142.1)

|---------- Split rule: Water <= 183.40 (MSE 103.0; observations  48; prediction 40.48)
|---------- Split rule: Water >  183.40 (MSE 105.6; observations  29; prediction 27.75)

|-- Split rule: Age >  21.00 (MSE 232.00; observations 561; prediction 41.38)

|------ Split rule:  Cement <= 352.50 (MSE 156.8)

|---------- Split rule: Cement <= 164.80 (MSE  91.2; observations 100; prediction 25.79)
|---------- Split rule: Cement >  164.80 (MSE 130.2; observations 333; prediction 39.94)

|------ Split rule: Cement >  352.50 (MSE 158.0)

|---------- Split rule: Water <= 183.05 (MSE  90.0; observations  78; prediction 63.43)
|---------- Split rule: Water >  183.05 (MSE 113.9; observations  50; prediction 47.73)
```

# Example 3. Predict Compressive Strength of Concrete

**Decision Tree:** Max depth = 3



# Example 3. Predict Compressive Strength of Concrete

**Summary:** 206 Predictions:

```
--- Index  Strength (MPa)  Prediction (MPa)  Error (MPa) ...
--- ================================================== ...
---   765           23.22             27.30         4.08 ...
---   773           37.42             47.28         9.86 ...
---   709           32.10             40.09         7.99 ...
---   661           10.39             15.84         5.45 ...
---   794           26.74             40.09        13.35 ...
---     1           61.89             64.45         2.56 ...
---   820           67.11             47.28       -19.83 ...

....

---   604           38.89             40.09         1.20 ...
---   778           14.80             15.84         1.04 ...
---   640           30.14             27.30        -2.84 ...
---   341           47.40             40.09        -7.31 ...
---   524           59.49             64.45         4.96 ...
---   420           33.09             27.53        -5.56 ...
---   990           36.35             26.20       -10.15 ...
--- ================================================== ...
--- Error Min Value: 0.03
--- Error Max Value: 32.90
--- Error Standard Deviation: 6.73
--- Mean Squared Error: 116.10
--- Root Mean Squared Error: 10.77
--- ================================================== ...
```

# Example 3. Summary

The data mining says:

- Age is the dominant predictor of concrete compressive strength, particularly during the first 21 days of curing.
- Concrete strength increases with levels of cement content (thank goodness).
- If you want to maximize concrete compressive strength, use lots of cement with moderate levels of water, and then wait at least 21 days.
- Too much water used in conjunction with high levels of cement content will result in a decrease of concrete strength.
- Organizing the data into a three level hierarchy (vs a two-level hierarchy) provides a modest improvement in reduction of uncertainty.

# Example 4: Predicting Weather in Seattle

**Problem Statement**

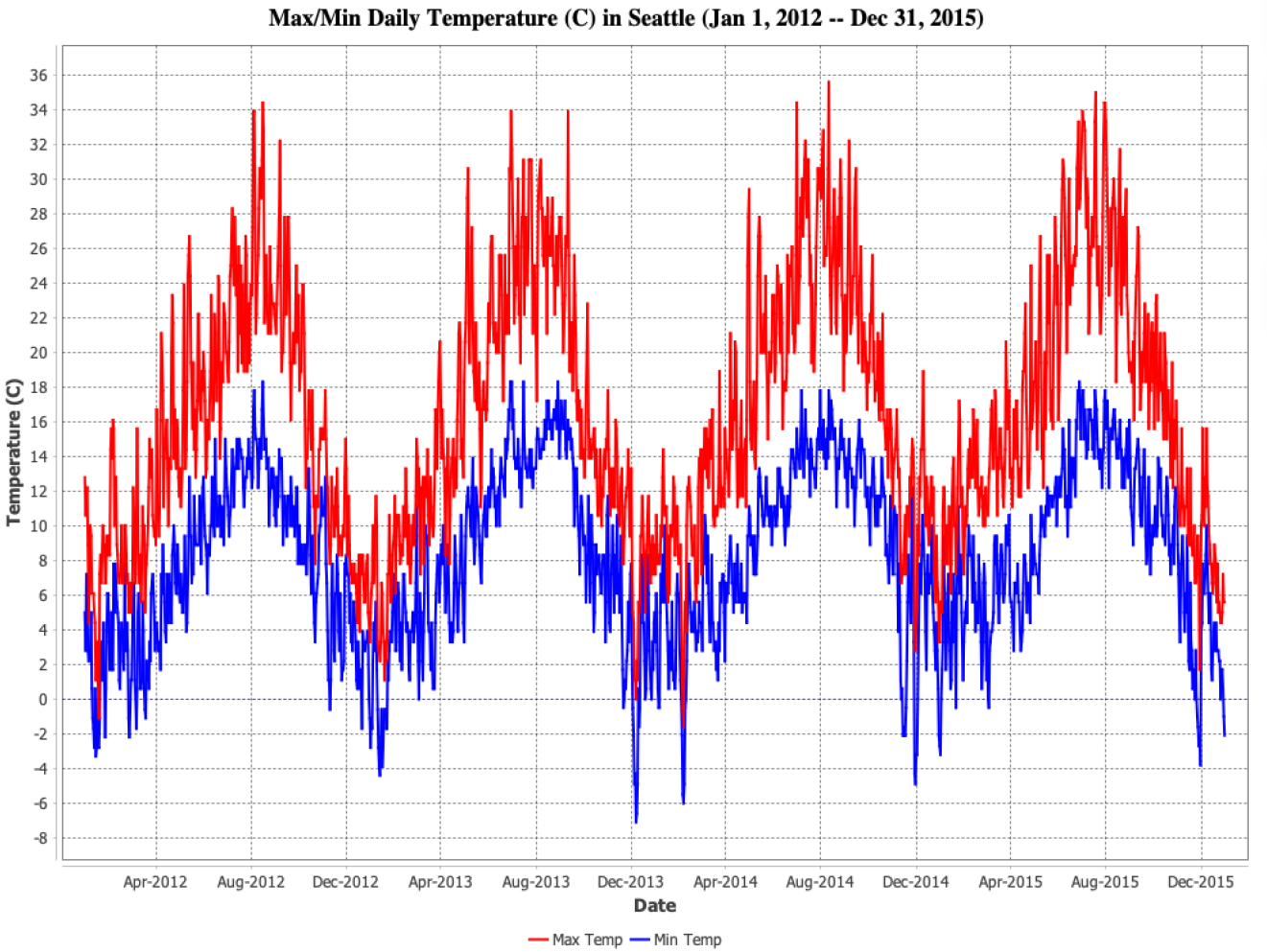
- Create regression model for prediction of weather conditions in Seattle, including max/min temperature and overall conditions (fog, rain, sun, drizzle, snow).

**Step-by-Step Procedure**

- Import and clean the dataset.
- Identify overall trends.
- Transform/expand the dataset.
- Uncover relationships (correlations) between variables.
- Train the machine learning model.

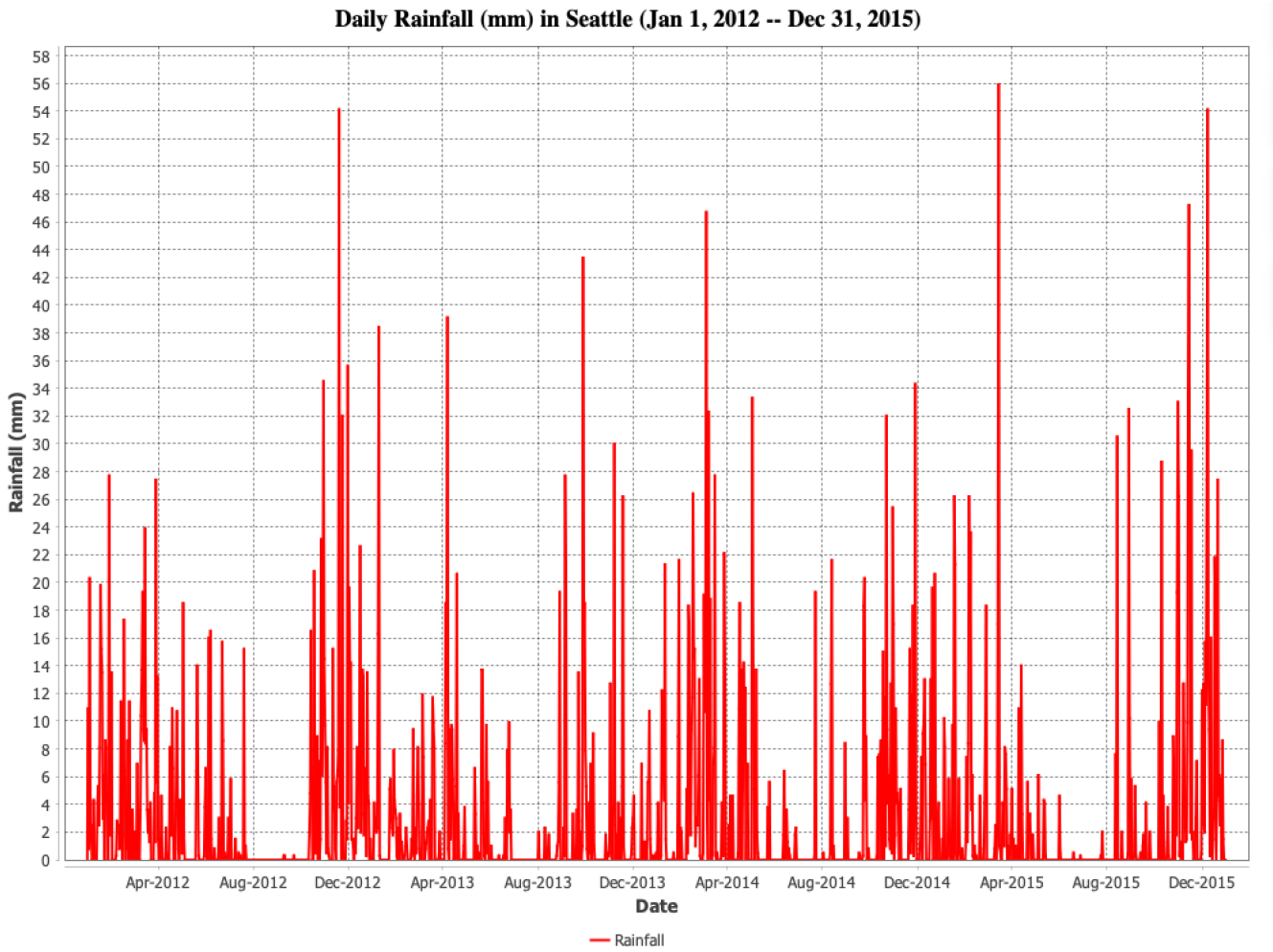
# Example 4: Predicting Weather in Seattle

**Time-History:** Max/min temperature in Seattle



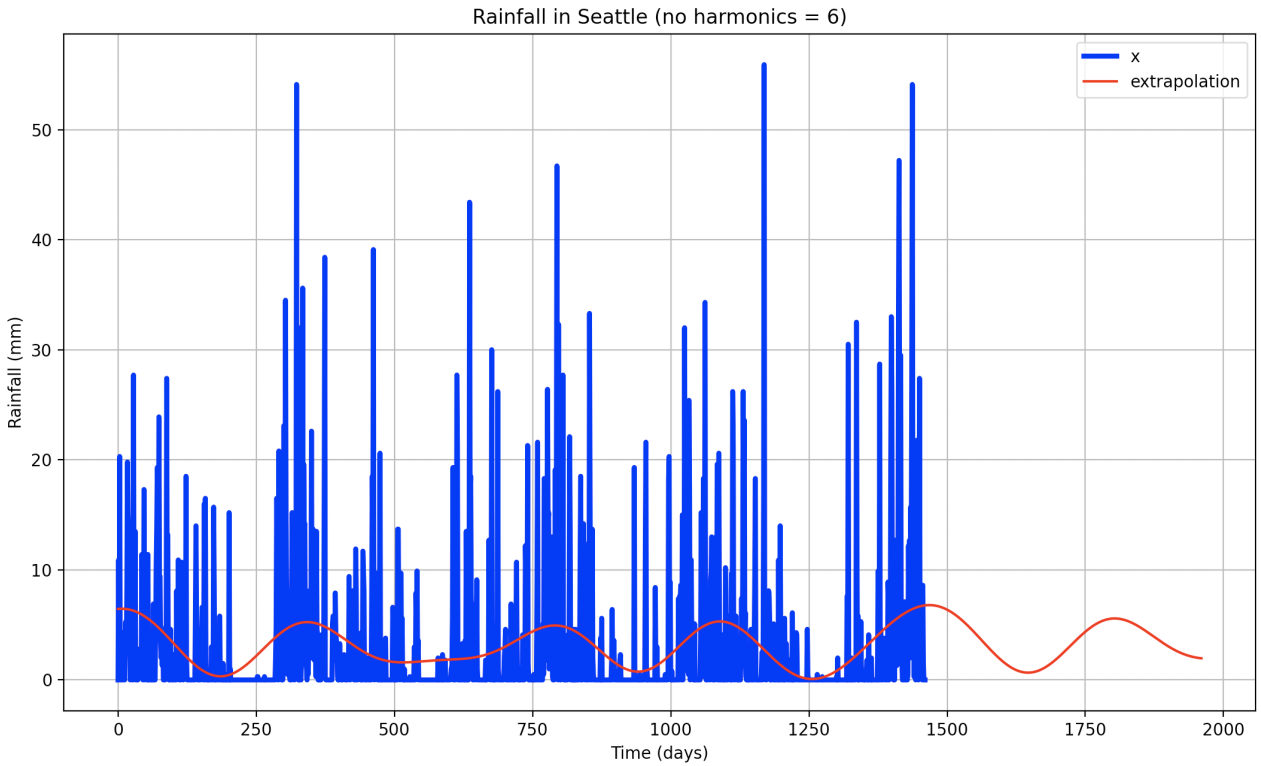
# Example 4: Predicting Weather in Seattle

**Time-History:** Rainfall in Seattle



# Example 4: Predicting Weather in Seattle

**Preliminary Analysis:** Extrapolated Fourier series (no harmonics = 6).




**Source Code:** See python-code.d/math/

# Example 4: Predicting Weather in Seattle

**Raw Datafile:** 1,461 data items ...

| | date | precipitation | temp_max | temp_min | wind | weather |
|---|---|---|---|---|---|---|
| 0 | 2012-01-01 | 0.0 | 12.8 | 5.0 | 4.7 | drizzle |
| 1 | 2012-01-02 | 10.9 | 10.6 | 2.8 | 4.5 | rain |
| 2 | 2012-01-03 | 0.8 | 11.7 | 7.2 | 2.3 | rain |
| 3 | 2012-01-04 | 20.3 | 12.2 | 5.6 | 4.7 | rain |
| 4 | 2012-01-05 | 1.3 | 8.9 | 2.8 | 6.1 | rain |
| ... | ... | ... | ... | ... | ... | ... |
| 1456 | 2015-12-27 | 8.6 | 4.4 | 1.7 | 2.9 | rain |
| 1457 | 2015-12-28 | 1.5 | 5.0 | 1.7 | 1.3 | rain |
| 1458 | 2015-12-29 | 0.0 | 7.2 | 0.6 | 2.6 | fog |
| 1459 | 2015-12-30 | 0.0 | 5.6 | -1.0 | 3.4 | sun |
| 1460 | 2015-12-31 | 0.0 | 5.6 | -2.1 | 3.5 | sun |

[1461 rows x 6 columns]

**Source:** python-code-ai.d/data/cities/seattle/seattle-weather.csv

# Example 4: Predicting Weather in Seattle

## Weather-to-Number Mapping

```
Dictionary: Weather to Number:
-------------------------------------------------
key: sun --> value: 1 ...
key: fog --> value: 2 ...
key: drizzle --> value: 3 ...
key: rain --> value: 4 ...
key: snow --> value: 5 ...
-------------------------------------------------
```

## Month-to-Number Mapping

```
Dictionary: Month to Number:
-------------------------------------------------
key: January --> value: 1 ...
key: February --> value: 2 ...
key: March --> value: 3 ...
key: April --> value: 4 ...
key: May --> value: 5 ...
key: June --> value: 6 ...
key: July --> value: 7 ...
key: August --> value: 8 ...
key: September --> value: 9 ...
key: October --> value: 10 ...
key: November --> value: 11 ...
key: December --> value: 12 ...
-------------------------------------------------
```

# Example 4: Predicting Weather in Seattle

## Data Preprocessing:

```
Analysis of Weather Category
----------------------------
4    641
1    640
2    101
3     53
5     26
Name: count, dtype: int64

Analysis of Month Category ...
----------------------------
1     124
3     124
5     124
7     124
8     124
10    124
12    124
4     120
6     120
9     120
11    120
2     113
```

# Example 4: Predicting Weather in Seattle

**Training Dataset:** 1,168 items.

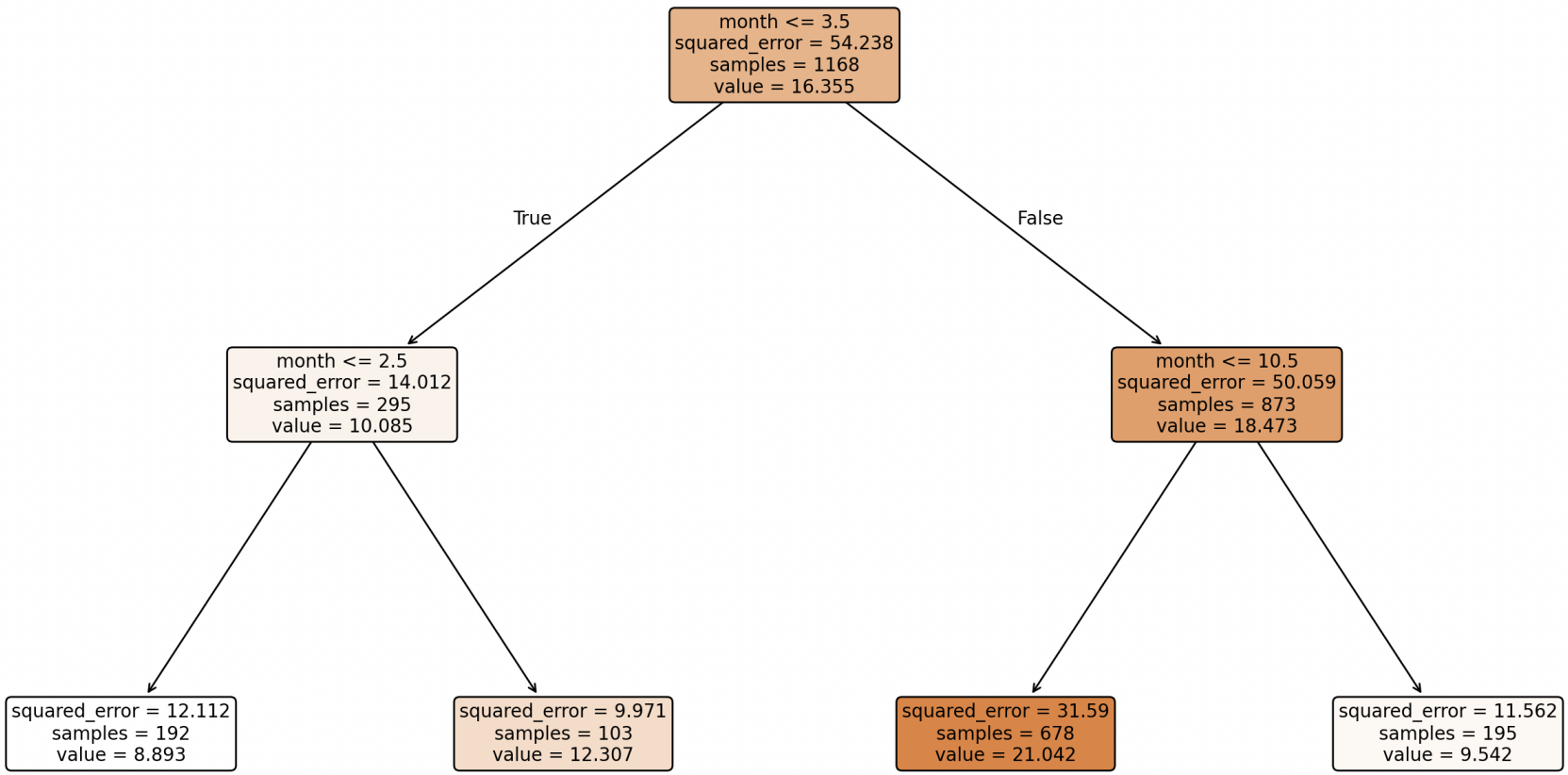
| Index | month | weather | precipitation | max temperature |
|---|---|---|---|---|
| 1043 | 11 | 4 | 5.1 | 13.3 |
| 1393 | 10 | 4 | 8.9 | 19.4 |
| 173 | 6 | 4 | 15.7 | 13.9 |
| ... | ... | ... | ... | .... |
| 257 | 9 | 1 | 0.0 | 26.1 |
| 507 | 5 | 4 | 13.7 | 11.1 |
| 119 | 4 | 4 | 4.3 | 15.6 |

**Test Dataset:** 293 items

| Index | month | weather | precipitation | max temperature |
|---|---|---|---|---|
| 1386 | 10 | 4 | 3.8 | 15.0 |
| 189 | 7 | 4 | 0.0 | 28.3 |
| 836 | 4 | 4 | 10.9 | 11.1 |
| ... | ... | ... | ... | .... |
| 8306 | 7 | 1 | 0.0 | 34.4 |
| 761 | 1 | 4 | 2.3 | 7.8 |
| 933 | 7 | 4 | 0.3 | 21.1 |

# Example 4: Predict Maximum Temperature in Seattle

**Decision Tree:** Max depth = 2



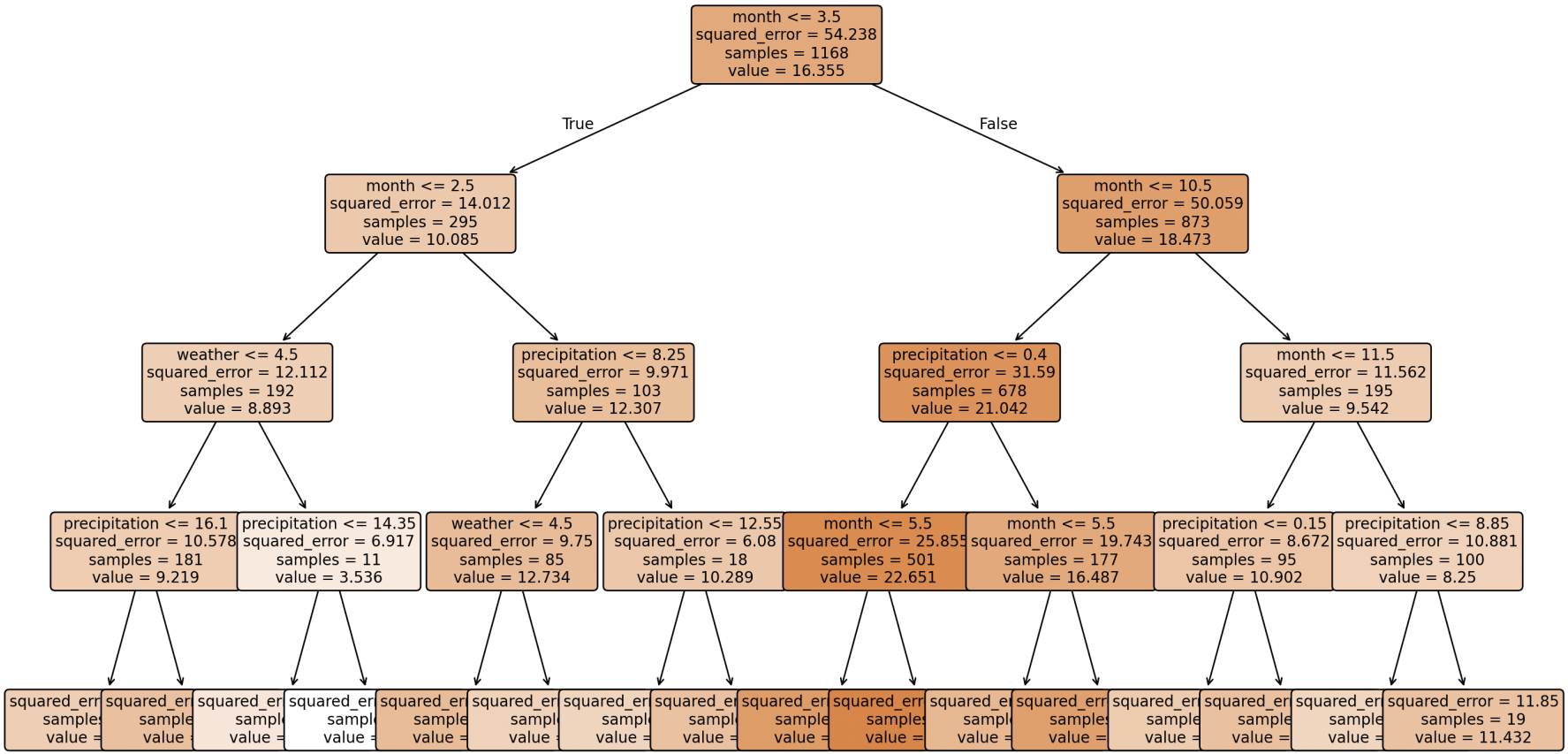
# Example 4: Predict Maximum Temperature in Seattle

**Summary:** 293 Predictions:

```
--- Index    Temp Max (C)    Prediction (C)    Error (C) ...
--- ================================================== ...

---  1386           15.00             21.20         6.20 ...
---   189           28.30             21.20        -7.10 ...
---   836           11.10             21.20        10.10 ...

....

---  1306           34.40             21.20       -13.20 ...
---   761            7.80              9.19         1.39 ...
---   933           21.10             21.20         0.10 ...

--- ================================================== ...
--- Error Min Value: 0.10
--- Error Max Value: 13.20
--- Error Standard Deviation: 2.92
--- Mean Squared Error: 24.74
--- Root Mean Squared Error: 4.97
--- ================================================== ...
```

# Example 4: Predict Maximum Temperature in Seattle

**Decision Tree:** Max depth = 4



# Example 4: Predict Maximum Temperature in Seattle

**Summary:** 293 Predictions:

```
--- Index    Temp Max (C)    Prediction (C)    Error (C) ...
--- ================================================== ...

---  1386           15.00             18.33         3.33 ...
---   189           28.30             23.42        -4.88 ...
---   836           11.10             13.74         2.64 ...

...

---  1306           34.40             23.42       -10.98 ...
---   761            7.80              9.32         1.52 ...
---   933           21.10             23.42         2.32 ...

--- ================================================== ...
--- Error Min Value: 0.03
--- Error Max Value: 11.72
--- Error Standard Deviation: 2.51
--- Mean Squared Error: 17.34
--- Root Mean Squared Error: 4.16
--- ================================================== ...
```

# Example 4: Predict Type of Weather in Seattle

**Training Dataset:** 1,168 items.

| Index | quarter | month | temp_max | temp_min | precipitation | wind | Weather |
|---|---|---|---|---|---|---|---|
| 1202 | 2 | 4 | 18.9 | 6.1 | 0.0 | 3.6 | 1 |
| 1116 | 1 | 1 | 7.2 | -0.5 | 0.0 | 1.3 | 2 |
| 1106 | 1 | 1 | 9.4 | 7.2 | 1.5 | 1.1 | 4 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 26 | 1 | 1 | 6.7 | -2.2 | 0.0 | 1.4 | 3 |
| 429 | 1 | 3 | 9.4 | 6.1 | 0.0 | 2.4 | 4 |
| 743 | 1 | 1 | 10.6 | 10.0 | 0.0 | 7.1 | 1 |

**Test Dataset:** 293 items

| | quarter | month | temp_max | temp_min | precipitation | wind | Weather |
|---|---|---|---|---|---|---|---|
| 1294 | 3 | 7 | 33.3 | 17.8 | 0.0 | 3.4 | 1 |
| 1407 | 4 | 11 | 11.1 | 7.8 | 6.6 | 1.8 | 4 |
| 562 | 3 | 7 | 31.1 | 18.3 | 0.0 | 4.1 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 587 | 3 | 8 | 25.6 | 15.0 | 2.3 | 2.9 | 4 |
| 412 | 1 | 2 | 11.1 | 3.9 | 0.0 | 5.6 | 4 |
| 1391 | 4 | 10 | 12.8 | 7.2 | 0.0 | 2.6 | 2 |

# Example 4: Predict Type of Weather in Seattle

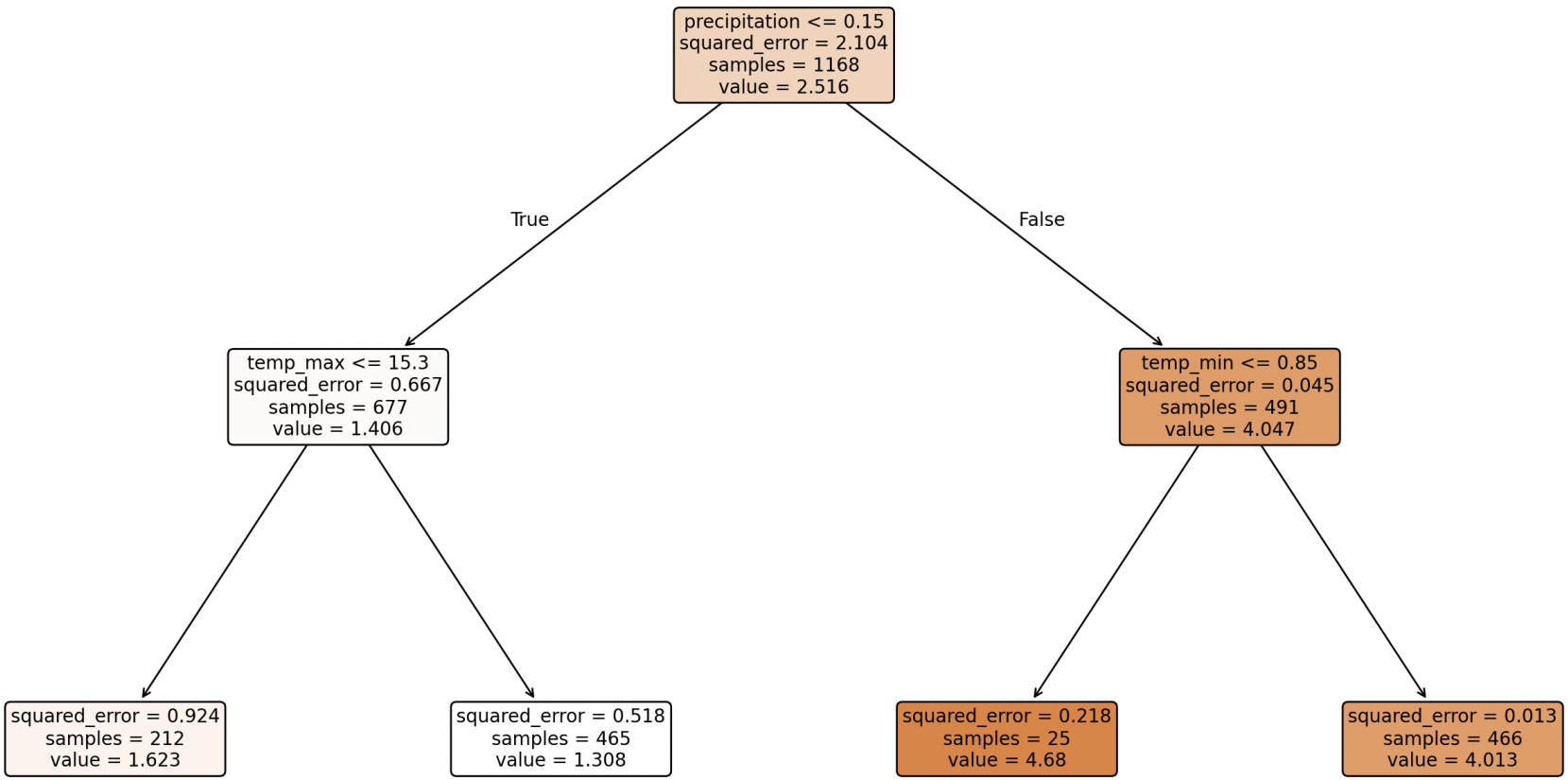
## Mappings:

```
Dictionary: Number to Weather:
------------------------------
key: 1 --> value: sun ...
key: 2 --> value: fog ...
key: 3 --> value: drizzle ...
key: 4 --> value: rain ...
key: 5 --> value: snow ...
------------------------------

Dictionary: Weather to Number:
------------------------------
key: sun --> value: 1 ...
key: fog --> value: 2 ...
key: drizzle --> value: 3 ...
key: rain --> value: 4 ...
key: snow --> value: 5 ...
------------------------------
```

```
Dictionary: Month to Number:
------------------------------
key: January --> value: 1 ...
key: February --> value: 2 ...
key: March --> value: 3 ...
key: April --> value: 4 ...
key: May --> value: 5 ...
key: June --> value: 6 ...
key: July --> value: 7 ...
key: August --> value: 8 ...
key: September --> value: 9 ...
key: October --> value: 10 ...
key: November --> value: 11 ...
key: December --> value: 12 ...
------------------------------
```

# Example 4: Predict Type of Weather in Seattle

**Decision Tree:** Max depth = 2



# Example 4: Predict Type of Weather in Seattle

**Summary:** 293 Predictions:

```
--- Index      Measurement (C)    Prediction (C)    Error (C) ...
--- ========================================================== ...
---  1294      1.00 --> sun       1.34 --> sun           0.34 ...
---  1407      4.00 --> rain      4.01 --> rain          0.01 ...
---   562      1.00 --> sun       1.34 --> sun           0.34 ...
---  1163      2.00 --> fog       1.34 --> sun          -0.66 ...
---   265      4.00 --> rain      4.01 --> rain          0.01 ...

....

---  1015      1.00 --> sun       1.34 --> sun           0.34 ...
---   197      4.00 --> rain      4.01 --> rain          0.01 ...
---   822      1.00 --> sun       1.34 --> sun           0.34 ...
---   587      4.00 --> rain      4.01 --> rain          0.01 ...
---   412      4.00 --> rain      1.34 --> sun          -2.66 ...
---  1391      2.00 --> fog       1.34 --> sun          -0.66 ...
--- ========================================================== ...
--- Error Min Value: 0.01
--- Error Max Value: 2.66
--- Error Standard Deviation: 0.48
--- Mean Squared Error: 0.34
--- Root Mean Squared Error: 0.58
--- ========================================================== ...
```

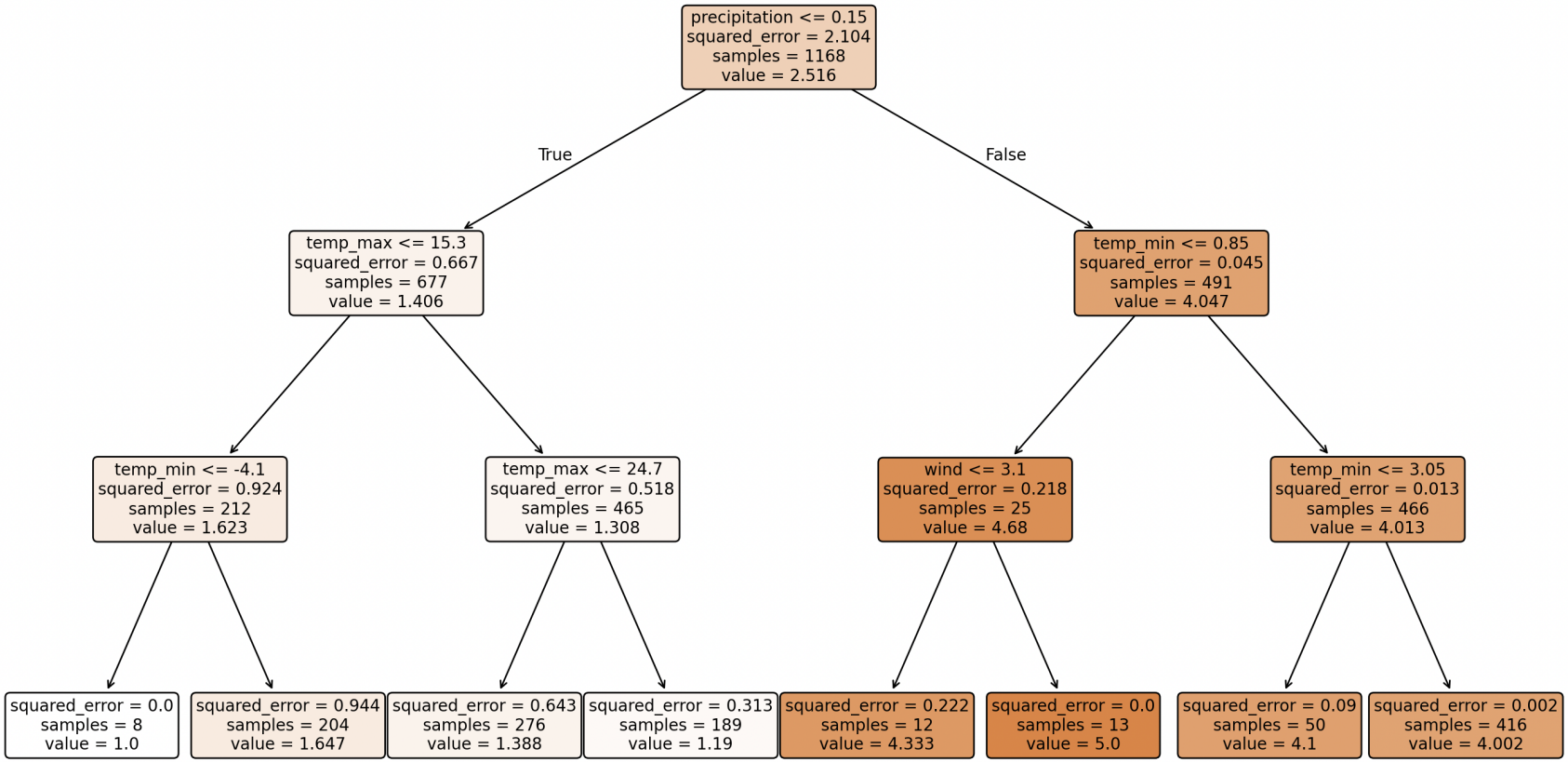
# Example 4: Predict Type of Weather in Seattle

## **Decision Tree Output:** Max depth = 3

```
Root (MSE 2.104; samples 1168; value 2.516)                                    Most likely weather
--------------------------------------------------------------------------------------------------
|-- Split rule: precipitation <= 0.15 (MSE 0.667; samples 667; value 1.406)

|----- Split rule: temp_max <= 15.3 (MSE  0.924; samples 212; value 1.623)

|---------- Split rule: temp_min <= -4.1 (MSE    0.0; samples   8; value 1.000)     <-- SUN !!
|---------- Split rule: temp_min  > -4.1 (MSE  0.944; samples 204; value 1.647)     <-- FOG !!

|----- Split rule: temp_max >  15.3 (MSE  0.518; samples 465; value 1.308)

|---------- Split rule: temp_max <= 24.7 (MSE  0.643; samples 276; value 1.388)
|---------- Split rule: temp_max >  24.7 (MSE  0.313; samples 189; value 1.190)     <-- SUN !!

|-- Split rule: precipitation > 0.15 (MSE 0.045; samples 491; value 4.047)    <-- RAIN/SNOW !!

|----- Split rule: temp_min <= 0.85 (MSE  0.218; samples 25; value 1.623)

|---------- Split rule: wind <= 3.10 (MSE  0.222; samples 12; value 4.33)
|---------- Split rule: wind  > 3.10 (MSE  0.000; samples 13; value 5.0)           <-- SNOW !!

|----- Split rule: temp_min >  0.85 (MSE  0.013; samples 466; value 4.013)

|---------- Split rule: temp_min <= 3.05 (MSE  0.09; samples  50; value 4.1)
|---------- Split rule: temp_min  > 3.05 (MSE 0.002; samples 416; value 4.0)       <-- RAIN !!
```

# Example 4: Predict Type of Weather in Seattle

**Decision Tree:** Max depth = 3



# Example 4: Predict Type of Weather in Seattle

**Summary:** 293 Predictions:

```
--- Index      Measurement (C)    Prediction (C)    Error (C) ...
--- ========================================================== ...
---  1294      1.00 --> sun       1.19 --> sun           0.19 ...
---  1407      4.00 --> rain      4.01 --> rain          0.01 ...
---   562      1.00 --> sun       1.19 --> sun           0.19 ...
---  1163      2.00 --> fog       1.40 --> sun          -0.60 ...
---   265      4.00 --> rain      4.01 --> rain          0.01 ...

....

---  1015      1.00 --> sun       1.40 --> sun           0.40 ...
---   197      4.00 --> rain      4.01 --> rain          0.01 ...
---   822      1.00 --> sun       1.40 --> sun           0.40 ...
---   587      4.00 --> rain      4.01 --> rain          0.01 ...
---   412      4.00 --> rain      1.40 --> sun          -2.60 ...
---  1391      2.00 --> fog       1.40 --> sun          -0.60 ...
--- ========================================================== ...
--- Error Min Value: 0.01
--- Error Max Value: 2.60
--- Error Standard Deviation: 0.49
--- Mean Squared Error: 0.35
--- Root Mean Squared Error: 0.59
--- ========================================================== ...
```

# Example 4: Summary

The data mining says:

- Weather conditions during the four year period are: sun (640 days), fog (101 days), drizzle (53 days), rain (641 days) and snow (26 days).
- Predictions of maximum temperature are dominated by season, whether or not their is precipitation, and the type of weather (particularly rain).
- Predictions of type of weather are affected by the presence/absense of preciptation, and the overall range of minimum/maximum tempertures.
- The three layer regression model does a remarkably good job at predicting sun and rain, but performs poorly on extreme values – only half of the predictions for snow are correct.

# Appendix C

# Data Mining with Java (Weka)

# Introduction to WEKA

## WEKA

WEKA (Waikato Environment for Knowledge Acquisition) is a workbench for data mining and machine learning.

### Software Download and Installation

- WEKA is written in Java, so it will run on both PCs and Macs.
- Download from: https://www.cs.waikato.ac.nz/weka/

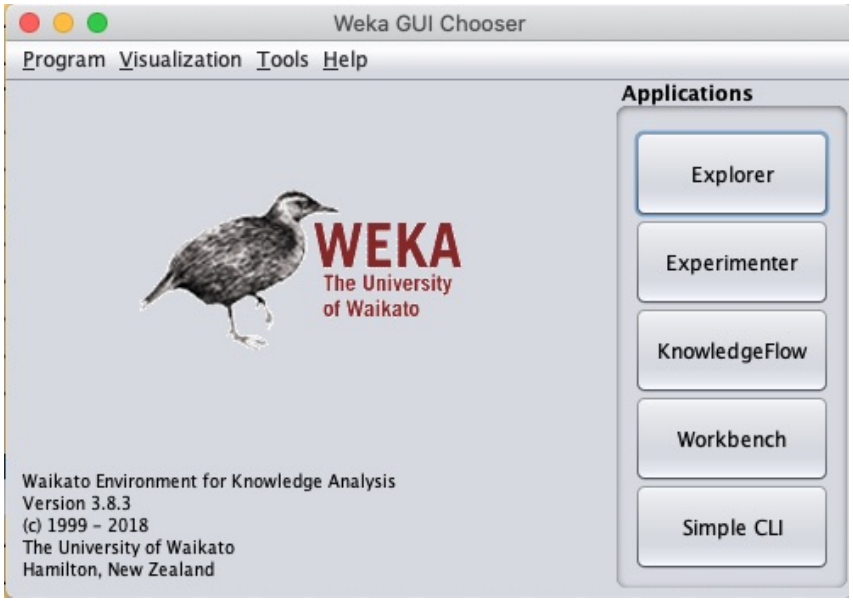
### Online Resources

- See class web page for evolving list of links to WEKA resources ...
- Videos learning machine learning with WEKA are available on YouTube.

# Getting Started

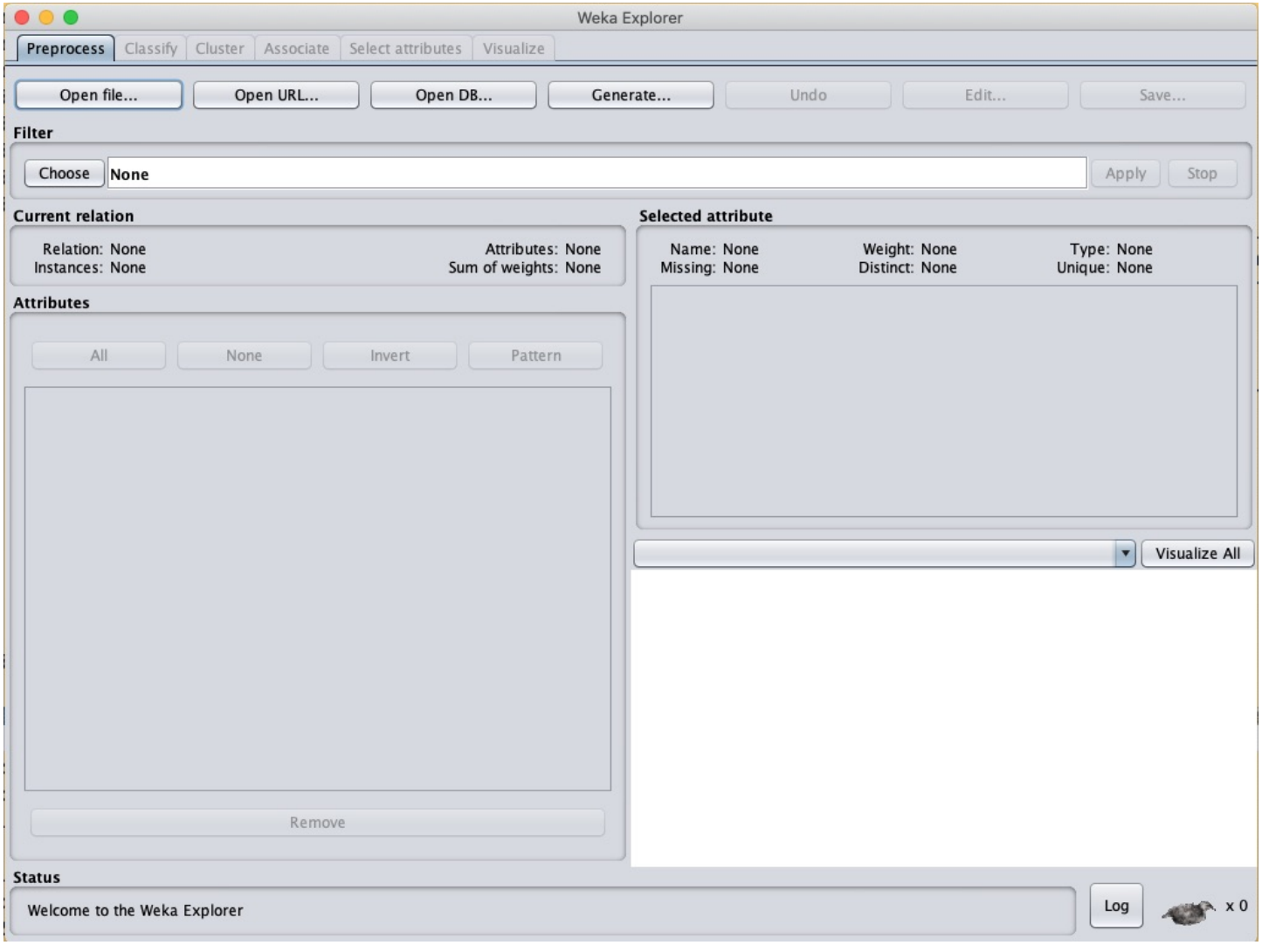
## From the Terminal Window

```
prompt >> java -jar weka.jar
```

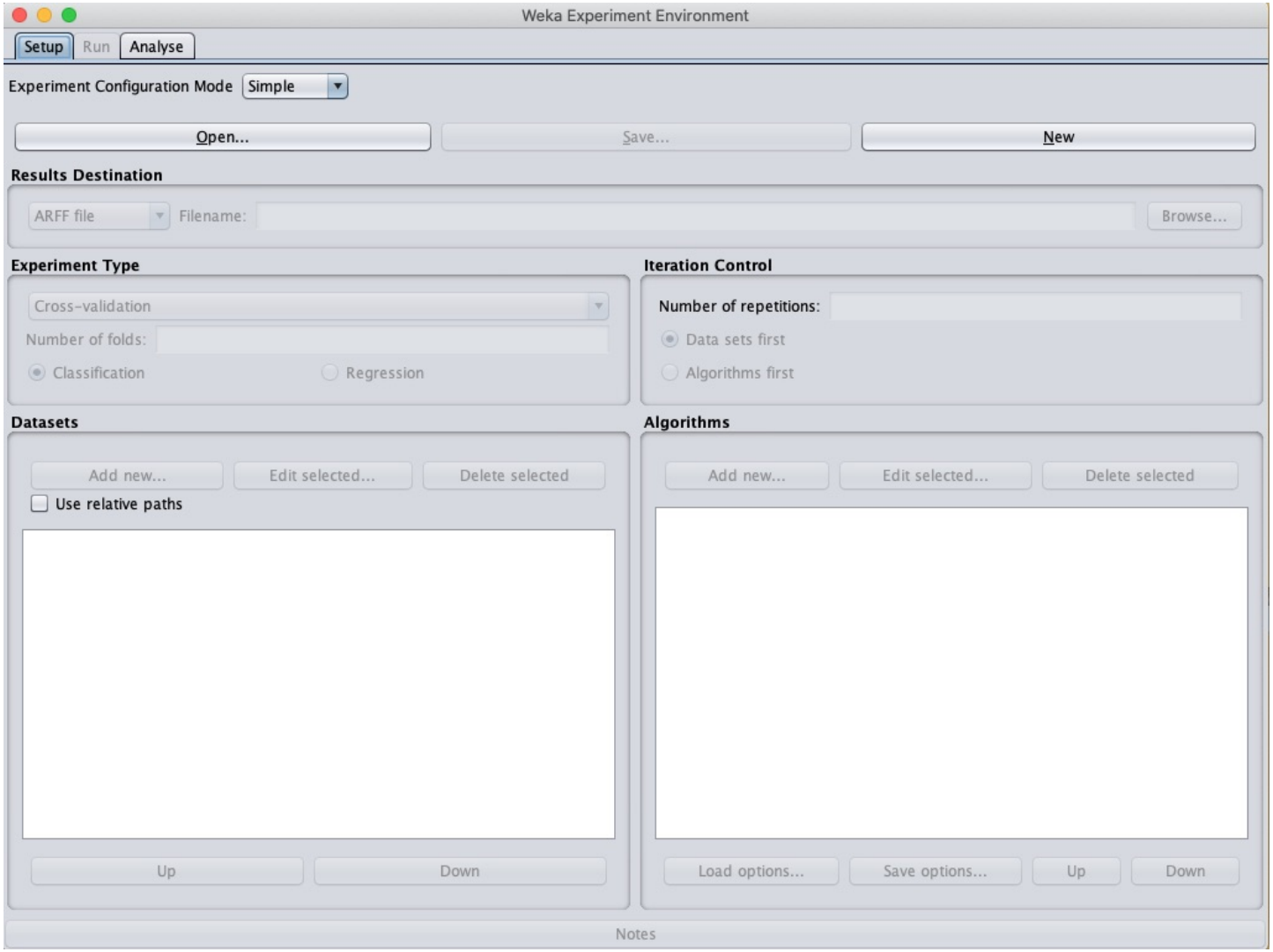



You can also write and run custom applications through the WEKA API.

# Weka GUI Explorer



# Weka GUI Experimenter



# Example 1. Will Customer Buy Computer?

## **Input datafile** (arff format)

```
1  % ============================================================
2  % ENCE 688P: Classification for Buy Computer?
3  % ============================================================
4
5  @relation 'computer'
6  @attribute id real
7  @attribute age { young, middle, senior}
8  @attribute income { low, medium, high}
9  @attribute student {yes, no}
10 @attribute credit { fair, excellent}
11 @attribute purchase { no, yes}
12
13 @data
14 1,young,high,no,fair,no
15 2,young,high,no,excellent,no
16 3,middle,high,no,fair,yes
17 4,senior,medium,no,fair,yes
18 5,senior,low,yes,fair,yes
19 6,senior,low,yes,excellent,no
20 7,middle,low,yes,excellent,yes
21 8,young,medium,no,fair,no
22 9,young,low,yes,fair,yes
23 10,senior,medium,yes,fair,yes
24 11,young,medium,yes,excellent,yes
25 12,middle,medium,no,excellent,yes
26 13,middle,high,yes,fair,yes
27 14,senior,medium,no,excellent,no
```

# Example 1. Will Customer Buy Computer?

**Java Source Code:**
java-code-ml-weka2018/src/ence688p/ClassificationTask.java

**Abbreviated Program Output** (J48 unpruned tree)

```
age = young
|   student = yes: yes (2.0)
|   student = no: no (3.0)
age = middle: yes (4.0)
age = senior
|   credit = fair: yes (3.0)
|   credit = excellent: no (2.0)

Number of Leaves  :  5
Size of the tree :  8
```

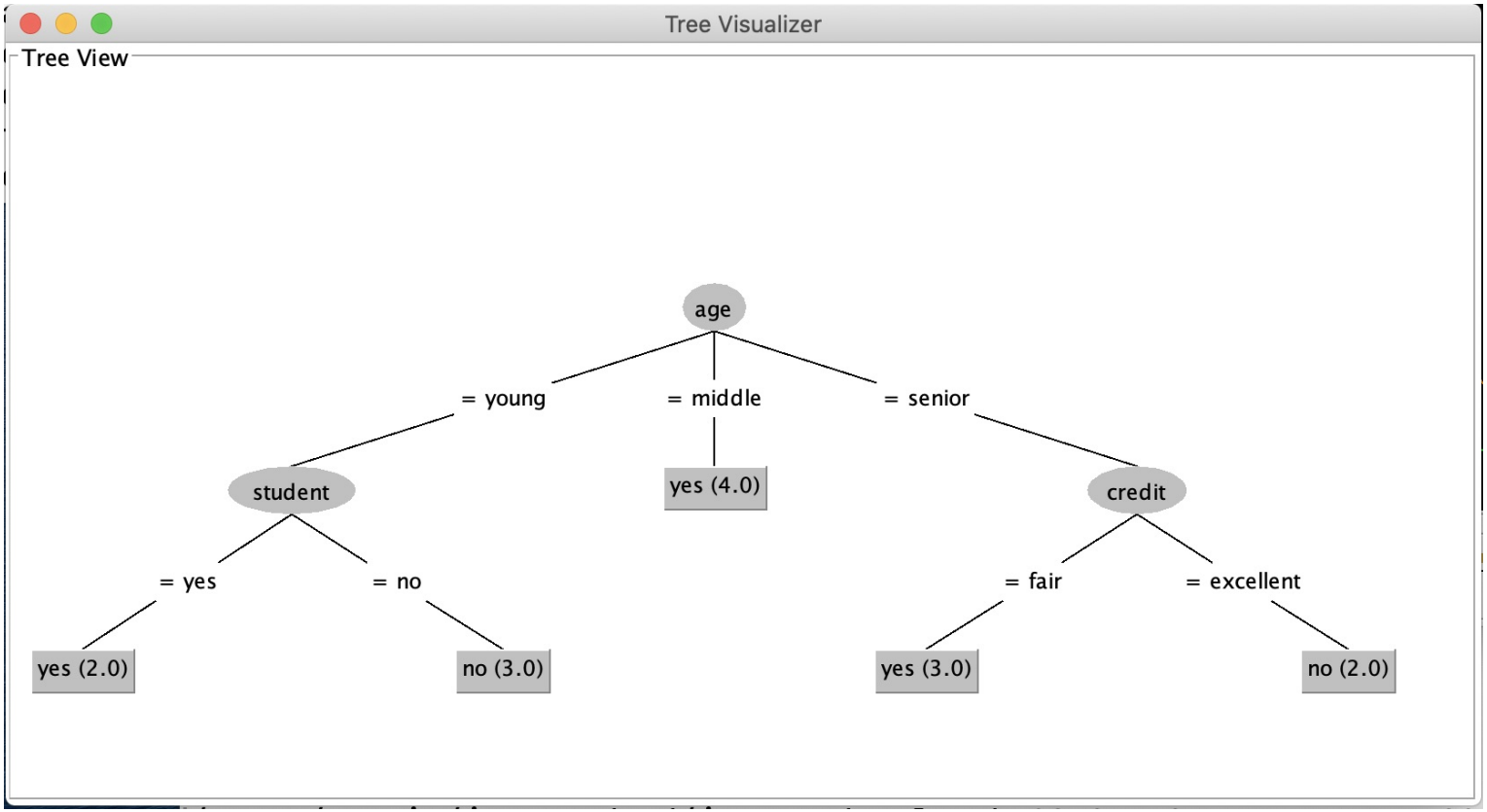
# Example 1. Will Customer Buy Computer?

## Classification Accuracy wrt Training Dataset

```
Correctly Classified Instances          14     100 %
Incorrectly Classified Instances         0       0 %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0 %
Root relative squared error              0 %
Total Number of Instances               14

=== Confusion Matrix ===

    a b   <-- classified as
    5 0 | a = no
    0 9 | b = yes
```

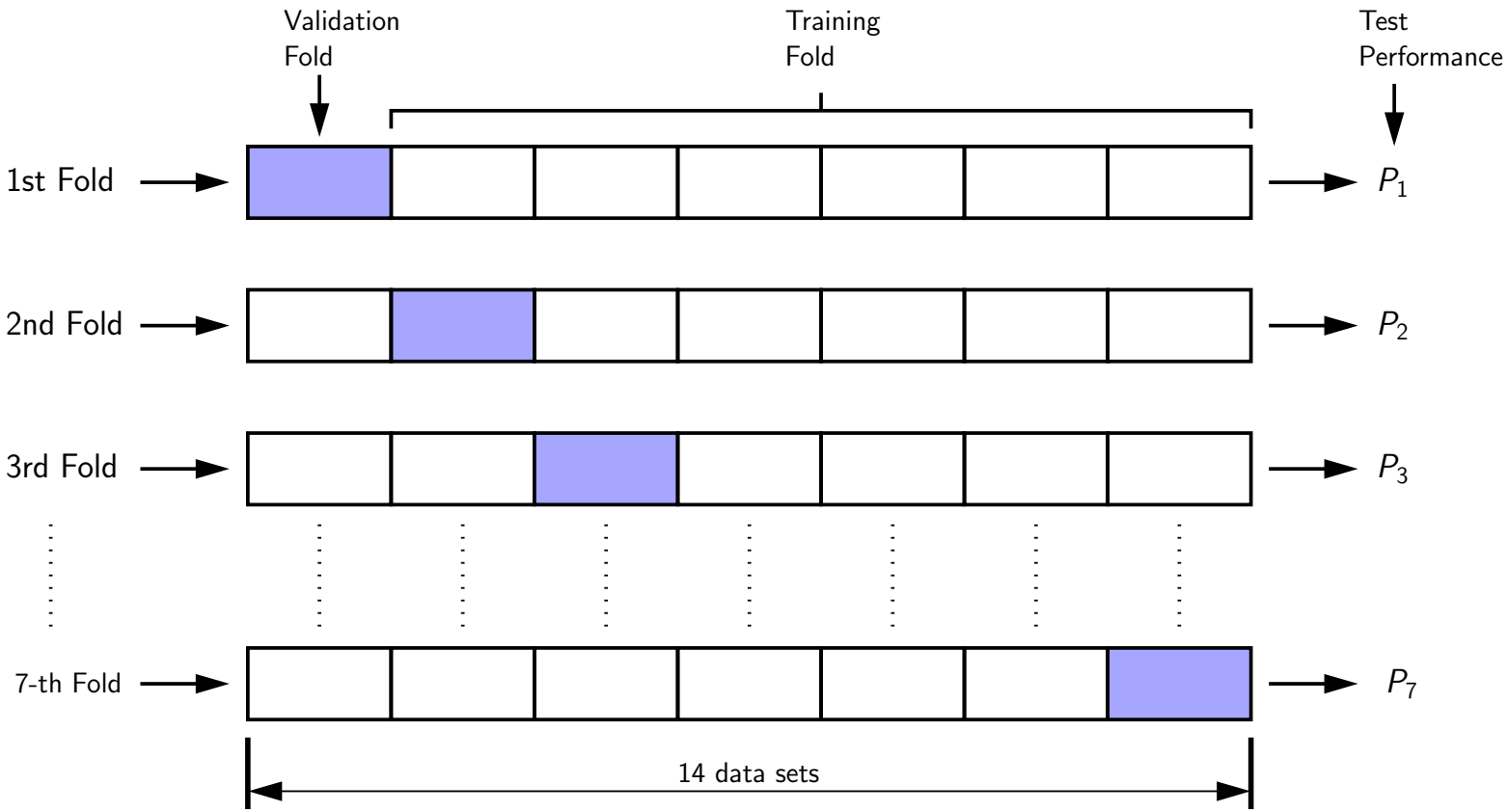
# Example 1. Will Customer Buy Computer?

## Classification Accuracy wrt Training Dataset



# Example 1. Will Customer Buy Computer?

**Cross Validation Model** (nofolds = 7)

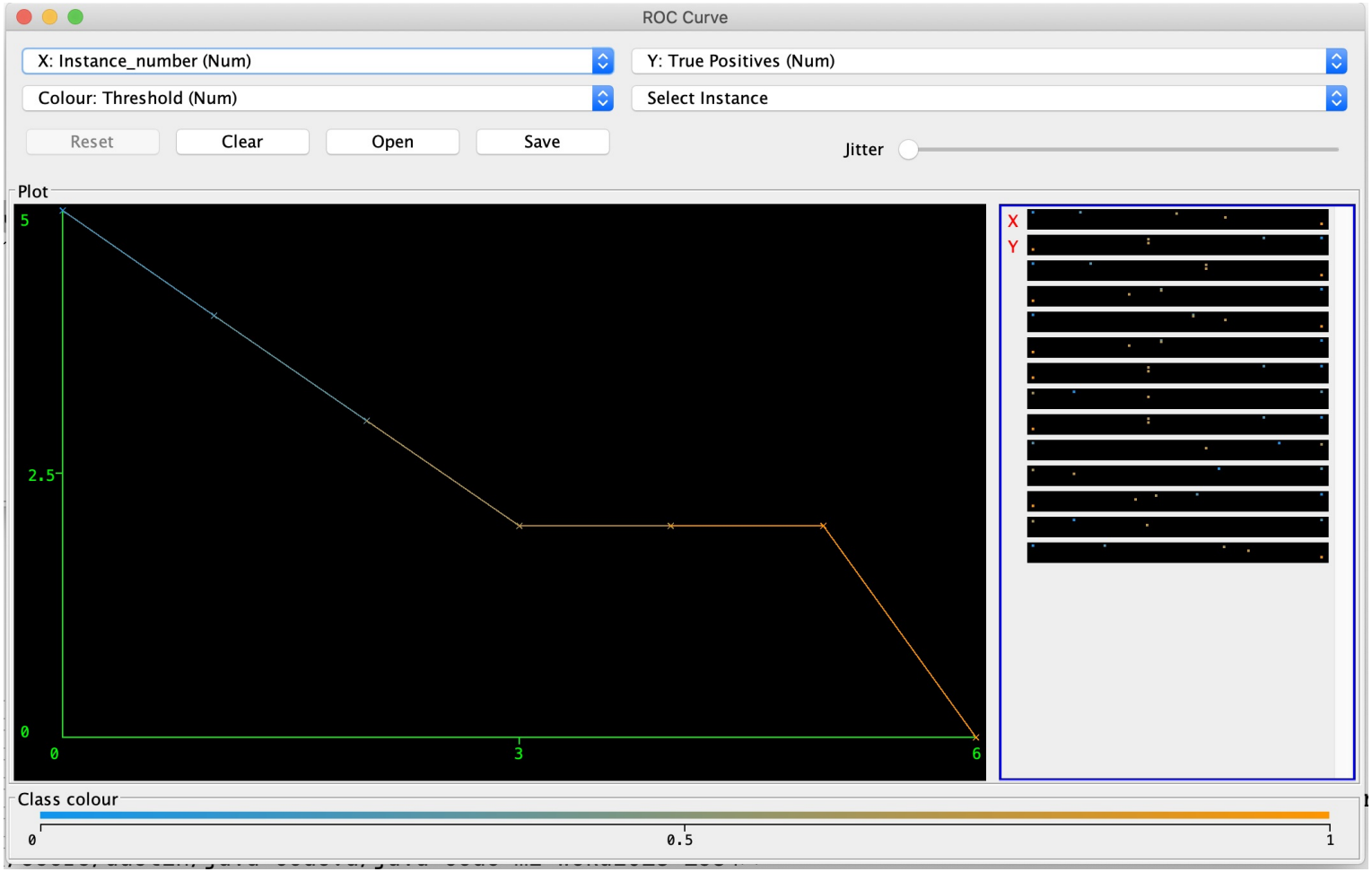


# Example 1. Will Customer Buy Computer?

**Cross Validation Model** (after classification) (nofolds = 7)

```
Correctly Classified Instances          10               71.4286 %
Incorrectly Classified Instances         4               28.5714 %
Kappa statistic                          0.3778
Mean absolute error                      0.2798
Root mean squared error                  0.4393
Relative absolute error                 58.3333 %
Root relative squared error             88.6322 %
Total Number of Instances               14

=== Confusion Matrix ===

    a b   <-- classified as
    3 2 | a = no
    2 7 | b = yes
```

# Example 1. Will Customer Buy Computer?



# Example 2. Milk, Diapers and Beer

## **Input datafile** (arff format)

```
1  % ================================================================
2  % ENCE 688P: Customer purchases at supermarket ..
3  %
4  % Mark Austin                                              March, 2021
5  % ================================================================
6
7  @relation 'supermarket'
8  @attribute id real
9  @attribute beer {t}
10 @attribute bread {t}
11 @attribute coke {t}
12 @attribute diapers {t}
13 @attribute eggs {t}
14 @attribute milk {t}
15
16 @data
17 1,?,t,?,?,?,t
18 2,t,t,?,t,t,?
19 3,t,?,t,t,?,t
20 4,t,t,?,t,?,t
21 5,?,t,t,t,?,t
```

# Example 2. Milk, Diapers and Beer

**Java Program Source Code** (Weka Code)

See: java-code-ml-weka2018/src/ence688p/Supermarket.java

**Abbreviated Program Output** (Print modified input file)

```
@relation supermarket-weka.filters.unsupervised.attribute.Remove-R1

@attribute beer {t}
... attributes removed ...
@attribute milk {t}

@data
?,t,?,?,?,t
t,t,?,t,t,?
t,?,t,t,?,t
t,t,?,t,?,t
?,t,t,t,?,t
```

# Example 2. Milk, Diapers and Beer

## **Abbreviated Program Output** (Apriori Model)

```
Size of set of large itemsets L(1): 6
Size of set of large itemsets L(2): 13
Size of set of large itemsets L(3): 12
Size of set of large itemsets L(4): 4

Best rules found:

 1. beer=t 3 ==> diapers=t 3 <conf:(1)> lift:(1.25) lev:(0.12) [0] conv:(0.6)
 2. coke=t 2 ==> diapers=t 2 <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.4)
 3. coke=t 2 ==> milk=t 2 <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:(0.4)
 4. beer=t bread=t 2 ==> diapers=t 2 <conf:(1)> lift:(1.25) lev:(0.08) [0] conv
 5. beer=t milk=t 2 ==> diapers=t 2 <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:
 6. coke=t milk=t 2 ==> diapers=t 2 <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:
 7. coke=t diapers=t 2 ==> milk=t 2 <conf:(1)> lift:(1.25) lev:(0.08) [0] conv:
 8. coke=t 2 ==> diapers=t milk=t 2 <conf:(1)> lift:(1.67) lev:(0.16) [0] conv:
 9. eggs=t 1 ==> beer=t 1 <conf:(1)> lift:(1.67) lev:(0.08) [0] conv:(0.4)
10. eggs=t 1 ==> bread=t 1 <conf:(1)> lift:(1.25) lev:(0.04) [0] conv:(0.2)
```

# Example 2. Milk, Diapers and Beer

## **Abbreviated Program Output** (FPGrowth Model)

```
FPGrowth found 38 rules (displaying top 10)

 1. [coke=t]: 2 ==> [milk=t]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 2. [beer=t]: 3 ==> [diapers=t]: 3 <conf:(1)> lift:(1.25) lev:(0.12) conv:(0.6)
 3. [coke=t]: 2 ==> [diapers=t]: 2 <conf:(1)> lift:(1.25) lev:(0.08) conv:(0.4)
 4. [eggs=t]: 1 ==> [diapers=t]: 1 <conf:(1)> lift:(1.25) lev:(0.04) conv:(0.2)
 5. [eggs=t]: 1 ==> [bread=t]: 1 <conf:(1)> lift:(1.25) lev:(0.04) conv:(0.2)
 6. [eggs=t]: 1 ==> [beer=t]: 1 <conf:(1)> lift:(1.67) lev:(0.08) conv:(0.4)
 7. [milk=t, beer=t]: 2 ==> [diapers=t]: 2 <conf:(1)> lift:(1.25) lev:(0.08) co
 8. [coke=t]: 2 ==> [milk=t, diapers=t]: 2 <conf:(1)> lift:(1.67) lev:(0.16) co
 9. [milk=t, coke=t]: 2 ==> [diapers=t]: 2 <conf:(1)> lift:(1.25) lev:(0.08) co
10. [diapers=t, coke=t]: 2 ==> [milk=t]: 2 <conf:(1)> lift:(1.25) lev:(0.08) co

--- ================================== ...
--- Finished !! ...
```