CS2109S: Introduction to AI and Machine Learning

Lecture 4:
Intro to Machine Learning & Decision Trees

4 February 2025

DO <u>NOT</u> CLOSE YOUR POLLEVERYWHERE APP

There will be activities ahead

Overview

You are here

Week 1 | Week 4 | Week 7 | Week 10 | Week 13

“Classical” AI | “Classical” ML | “Modern” ML | Misc

Search Algorithms

- Uninformed search: BFS, DFS
- Local Search: Hill Climbing
- Informed search: A*
- Adversarial search: Minimax

"Classical" ML

- Decision Trees
- Linear/Logistic Regression
- Kernels and Support Vector Machines
- "Classical" Unsupervised Learning

"Modern" ML

- Neural Networks
- Deep Learning
- Sequential Data

Miscellaneous

- AI & Ethics

Applied CS2040S, CS1231
Python

Applied Linear Algebra, Calculus, Statistics & Probabilities
Numpy, Scikit-learn, PyTorch

Outline

- Machine Learning
  - Problems
  - Supervised Learning
- Decision Trees
  - Hypothesis class
  - Decision Tree Learning
  - Pruning
  - Data Preprocessing

Outline

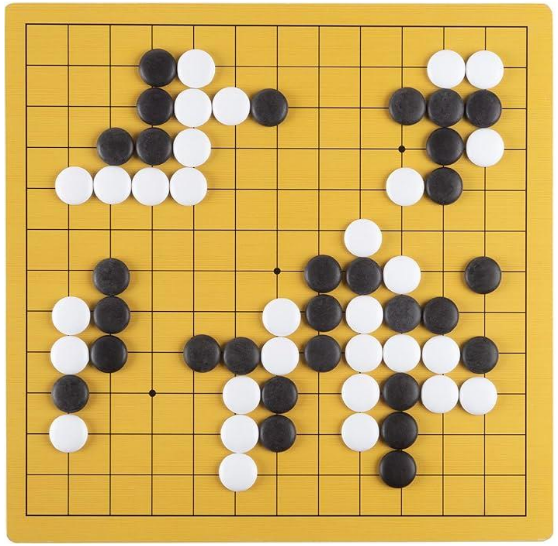
- **Machine Learning**
  - Problems
  - Supervised Learning
- Decision Trees
  - Hypothesis class
  - Decision Tree Learning
  - Pruning
  - Data Preprocessing

Problems

- Some problems are intractable to solve in general, meaning that no efficient solution exists for all cases.
  - Example: The game of Go, protein folding, and similar complex problems.

- Other problems are difficult to solve because formulating the rules in a way that a computer can understand and process is challenging.
  - Example: Recognizing numbers in an image, answering open-ended questions, and tasks requiring complex reasoning.

Problems – Intractable in General



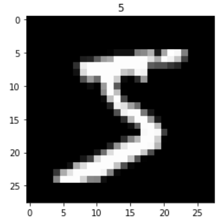
Credit: Amazon.sg

$b \approx 250, m \approx 150$ for "reasonable" games

Time complexity: $b^m = 250^{150} \approx 10^{360}$

Number of particles in the universe: $\sim 10^{80}$

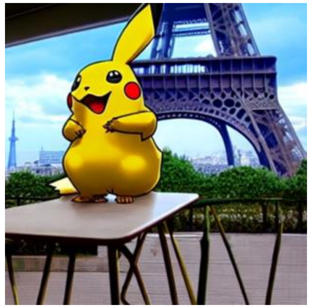
Problems – Difficult to Specify the Rules



→ Function → 5

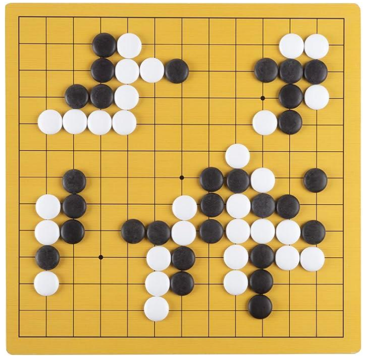
"What is AI?" → Function → "AI is …"

"Pikachu in the Eiffel tower" → Function →

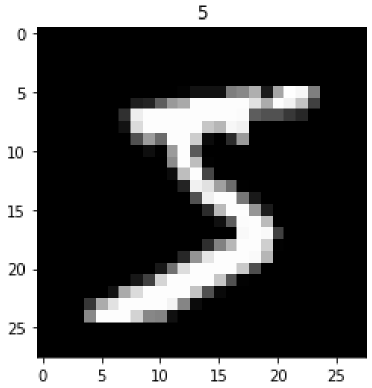


A New Paradigm: Learning Agent

Rather than solving the problem explicitly through search or by applying a set of rules, we can construct an agent that learns a function which could identify patterns in the data and makes decisions or provides solutions based on what it has learned.



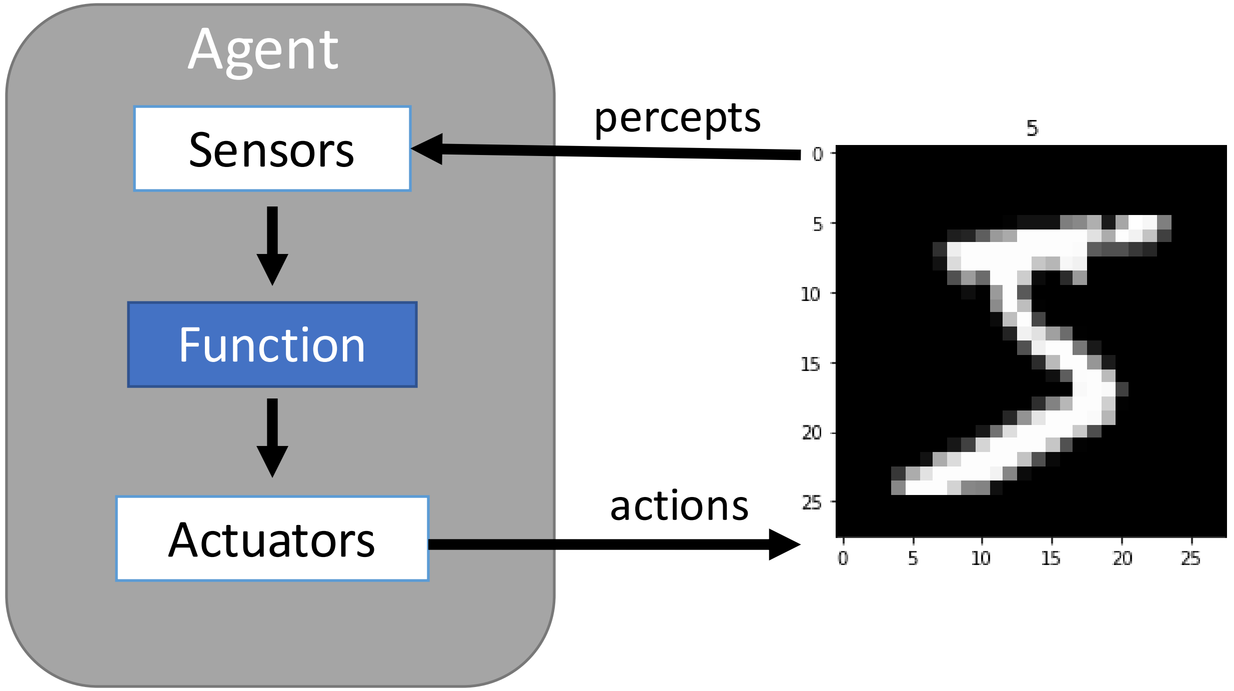
Credit: Amazon.sg



Designing an Agent

for problems where: The function is difficult to specify, or
Solutions are intractable to compute (in general)

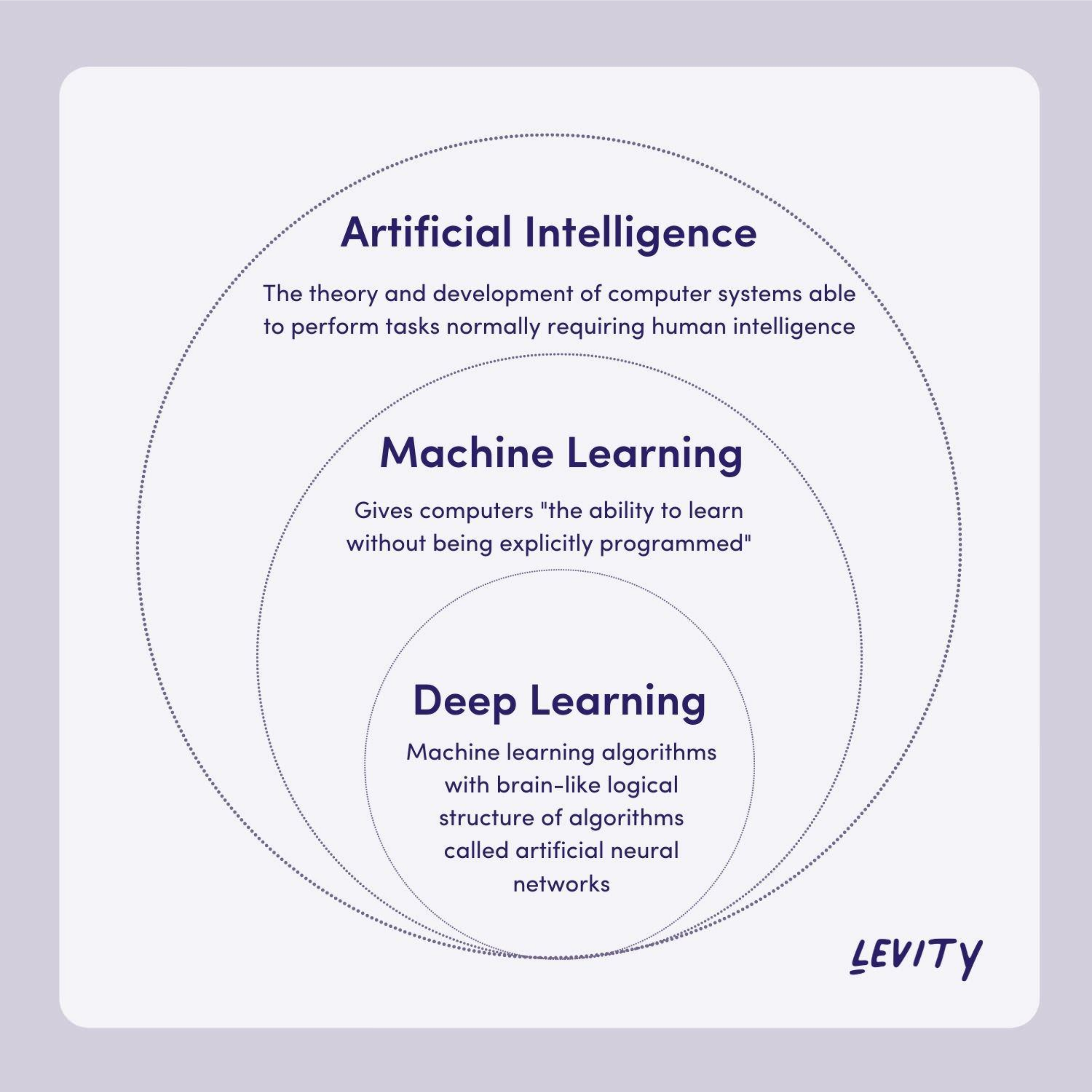
Learning Agent



Typically, episodic
But can also be sequential

Machine Learning

- **Machine learning (ML) is a subfield of AI that gives computers the ability to learn without being explicitly programmed.**
- It involves developing algorithms that can learn from and make predictions or decisions based on data.
- The goal is for a machine to improve its performance on a task over time by identifying patterns in the data it processes.

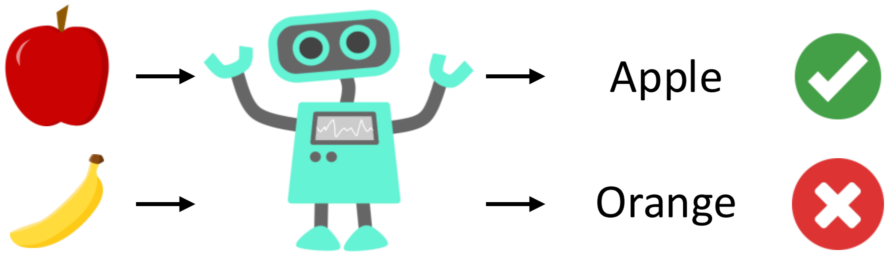


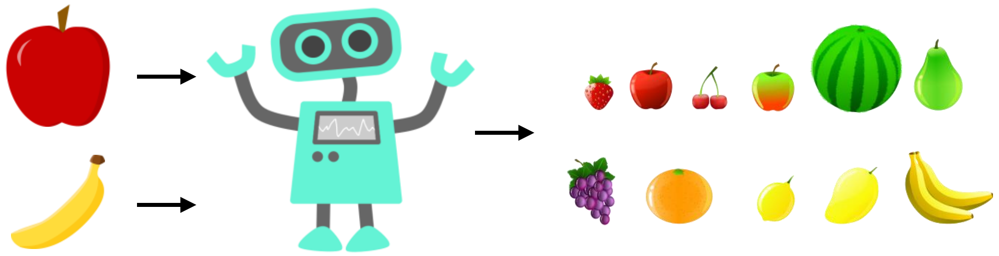
Credit: levity.ai

Machine Learning – Types

- **Supervised Learning**: A type of machine learning where an agent learns from labeled data (input-output pairs) to learn a mapping from inputs to outputs.
  - Example: classify whether an image contains a picture of a banana or apple
- **Unsupervised Learning**: A type of machine learning where an agent learns from unlabeled data (input only) and aims to find patterns or structure.
  - Example: group images of fruits into 2, 3, 4, … groups, based on features like color, shape, etc
- **Semi-supervised Learning**: A type of machine learning where an agent learns from labeled and unlabeled data.
- **Reinforcement Learning**: A type of machine learning where an agent learns to make decisions by interacting with an environment, receiving rewards or penalties based on its actions to maximize cumulative reward over time.

Machine Learning – Types





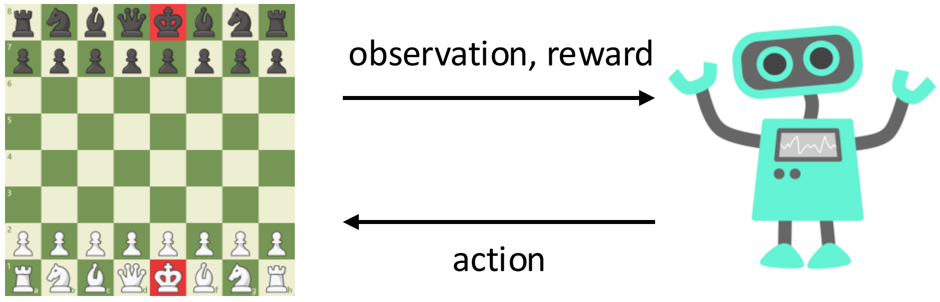
Supervised

"Teacher's feedback"

Semi/Weakly Supervised

Unsupervised

"No feedback"



Reinforcement

"Trial and error"

Supervised Learning

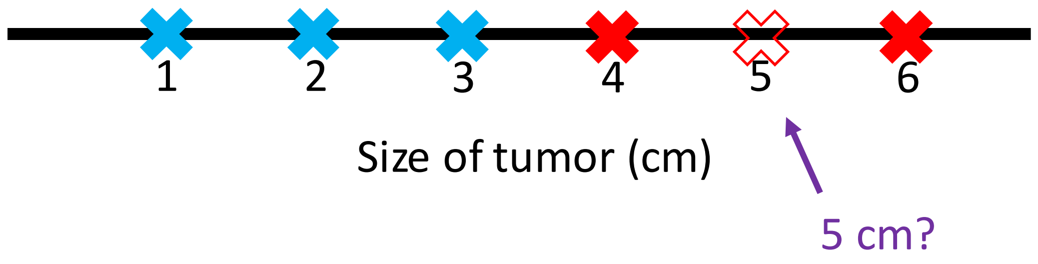
- Learns to map the inputs to the outputs in a dataset by minimizing the difference between its **predictions** and the provided **correct outputs/answers** (ground truth) using a learning algorithm.
  - This phase is known as the training phase.
  - The dataset is called the training set.
  - This results in a trained agent function, often called a model / hypothesis.
- Once learning is done, the model can predict the output for new, unseen data.
  - This phase is known as the testing / evaluation phase.
  - We can measure the performance of the model on a set of unseen data called test set.
  - This performance on unseen data measures the generalization of the model

Supervised Learning – Tasks

- **Classification**: A type of supervised learning where the goal is to predict a discrete label or category based on input features.
  - The output variable (target) is a **categorical value**, and the agent learns to assign an input to one of several predefined classes.
  - Example: spam detection, document categorization, patient classification
- **Regression**: A type of supervised learning where the goal is to predict a continuous numerical value based on input features.
  - The output variable (target) is a **real number**, and the agent learns to establish a mapping from input features to this continuous value.
  - Example: house price prediction, temperature prediction, sales forecasting

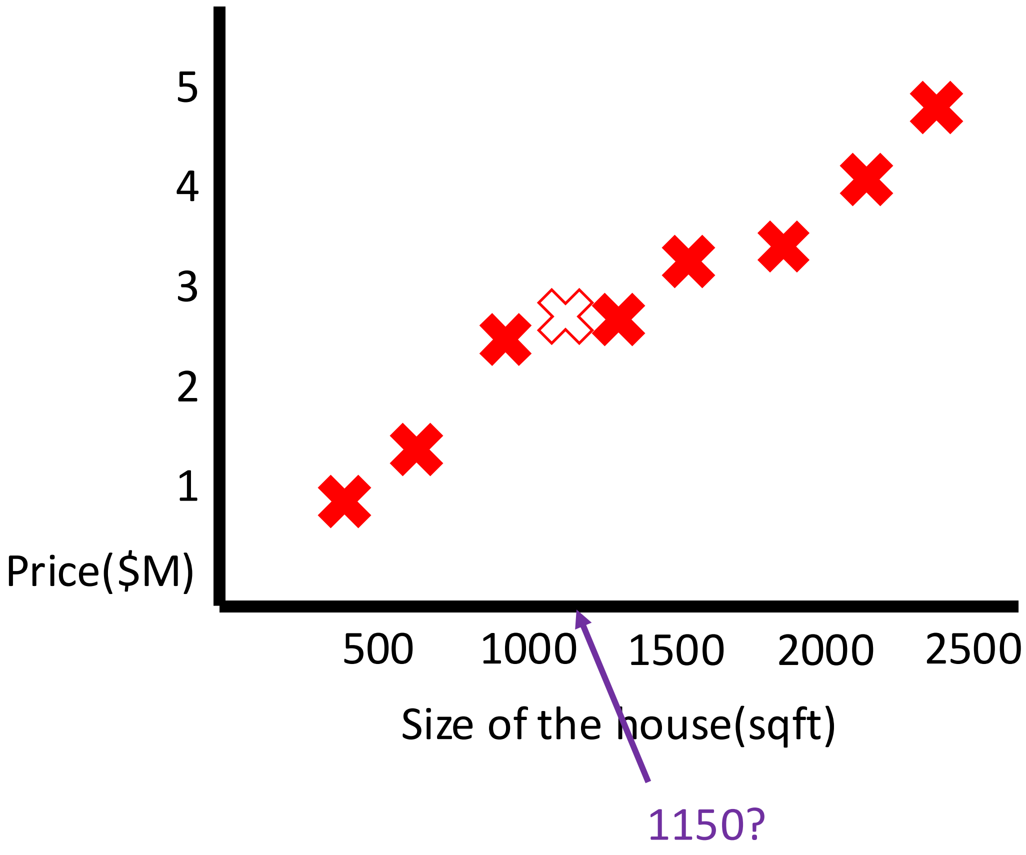
Supervised Learning – Classification

Cancer prediction: benign, malignant



Supervised Learning – Regression

Housing price prediction



Dataset

Dataset in supervised learning is represented as a set of pairs $(x^{(i)}, y^{(i)})$, where

- $x^{(i)} \in \mathbb{R}^d$ is the input vector (features) of the i-th data point of d features.
  - In general, $x^{(i)}$ can be multi-dimensional array of features / attributes
- $y^{(i)}$ is the label (target) for the i-th data point.
  - $y^{(i)} \in \mathbb{R}$ for regression
  - $y^{(i)} \in \{1,2, \dots, C\}$ for classification with C classes
- Dataset D with n examples can be represented as

$$D = \{\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \dots, (x^{(n)}, y^{(n)})\}$$

Dataset – True Data Generating Function

- We generally assume that there is an underlying true relationship between the input features x and the labels (outputs) y in the dataset: $y = f^*(x) + \epsilon$, where
  - $f^*(x)$ is the **true**, but unknown, function that generates the label from the input features.
  - ϵ is some noise or error term, which accounts for randomness or imperfections in the data generation process.
- The goal in supervised learning is to find a function that best approximates $f^*(x)$

Hypothesis Class

- A **hypothesis class** refers to the set of all possible models or functions that map from inputs to outputs $h: X \rightarrow Y$ and that can be learned by a learning algorithm
- Each element of the hypothesis class $h \in \mathcal{H}$ is a function called a **hypothesis** or **model**.
- We are interested in finding a hypothesis $h(x)$ that best approximates the true generating function $f^*(x)$
- Example (coming up later in this course):
  - Decision trees, (generalized) linear models, neural networks

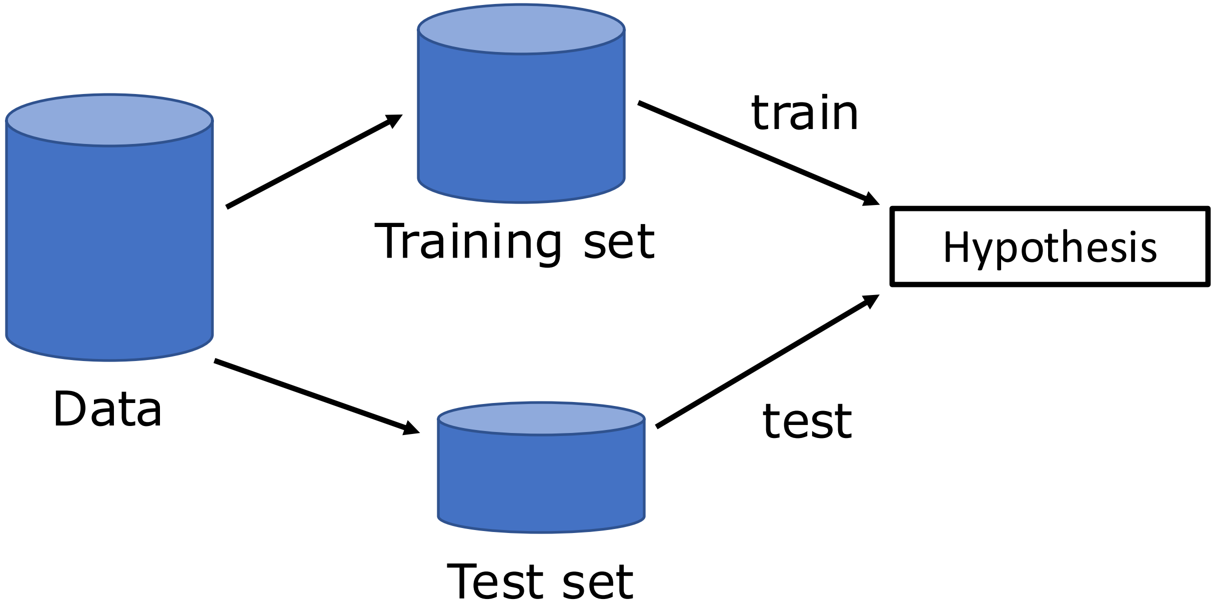
Learning Algorithms

- A **learning algorithm** A takes in a training set D_{train}, consisting of pairs $(x^{(i)}, y^{(i)})$, and seeks to find a function (model/hypothesis) h from a predefined hypothesis class $\mathcal{H}$ that approximates the true relationship between inputs and outputs.
  - $f^*(x) \approx h(x) = A(D_{train})$
  - Note: some learning algorithms "find" the hypothesis through construction.
- Example (coming up later in this course):
  - Decision tree learning, gradient descent

Performance Measure

How do we know that our model/hypothesis is good? i.e., $h(x) \approx f(x)$

Try the hypothesis on a new set of examples (**test set**)



Regression: Error

If the output of the hypothesis h is a **continuous** value, then we can measure its **error**. For an input x with a true output y, we can compute:

$$Absolute\ Error = |\hat{y} - y|$$

$$Squared\ Error = (\hat{y} - y)^2$$

Where $\hat{y} = h(x)$.

Regression: Mean Squared Error

For a set of n examples $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ we can compute the average (mean) **squared error** as follows.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

Where $\hat{y}^{(i)} = h(x^{(i)})$.

Regression: Mean Absolute Error

For a set of n examples $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$ we can compute the average (mean) **absolute error** as follows.

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}^{(i)} - y^{(i)}|$$

Where $\hat{y}^{(i)} = h(x^{(i)})$.

Classification: Correctness & Accuracy

Classification is correct when the prediction $\hat{y} = y$ (true label).

For a set of n examples$\left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)\right\}$ we can compute the average **correctness** (**accuracy**) as follows.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{\hat{y}^{(i)} = y^{(i)}}$$

Where $\hat{y}^{(i)} = h(x^{(i)})$.

Classification: Confusion Matrix

Ex.	Actual y	Predicted $\hat{y}$	
1	Cancer	Cancer	TP
2	Cancer	Cancer	
3	Cancer	Benign	FN
4	Cancer	Benign	
5	Cancer	Benign	
6	Benign	Benign	TN
7	Benign	Benign	
8	Benign	Benign	
9	Benign	Benign	
10	Benign	Cancer	FP

		Actual Label	
		Cancer	Benign
Predicted Label	Cancer	2 True Positive	1 False Positive
	Benign	3 False Negative	4 True Negative

“Predicted [Positive/Negative] and [True/False]”

FP: Type I error

FN: Type II error

Classification: Confusion Matrix

Predicted Label \ Actual Label	Cancer	Benign
Cancer	2 True Positive	1 False Positive
Benign	3 False Negative	4 True Negative

Accuracy = $\frac{\text{TP+TN}}{\text{TP+FN+FP+TN}}$

Precision

P = TP / (TP+FP)

How **precise** are the **positive predicted** instances?

Maximize this if <u>false positive (FP)</u> is very costly.

E.g., [email spam](), [satellite launch date]() prediction

Recall

R = TP / (TP+FN)

How many actual **positive instances** can be **recalled** (predicted)?

Maximize this if <u>false negative (FN)</u> is very dangerous.

E.g., [cancer prediction]() but not music recommendation

F1 Score

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Combination of both metrics (harmonic mean)

Supervised Learning (Illustrated)

Training

Training Set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$

Hypothesis Class H → Learning Algorithm A ← Performance Measure

Select $h(x) \approx f^*(x)$ in training set

Hypothesis $h(x)$

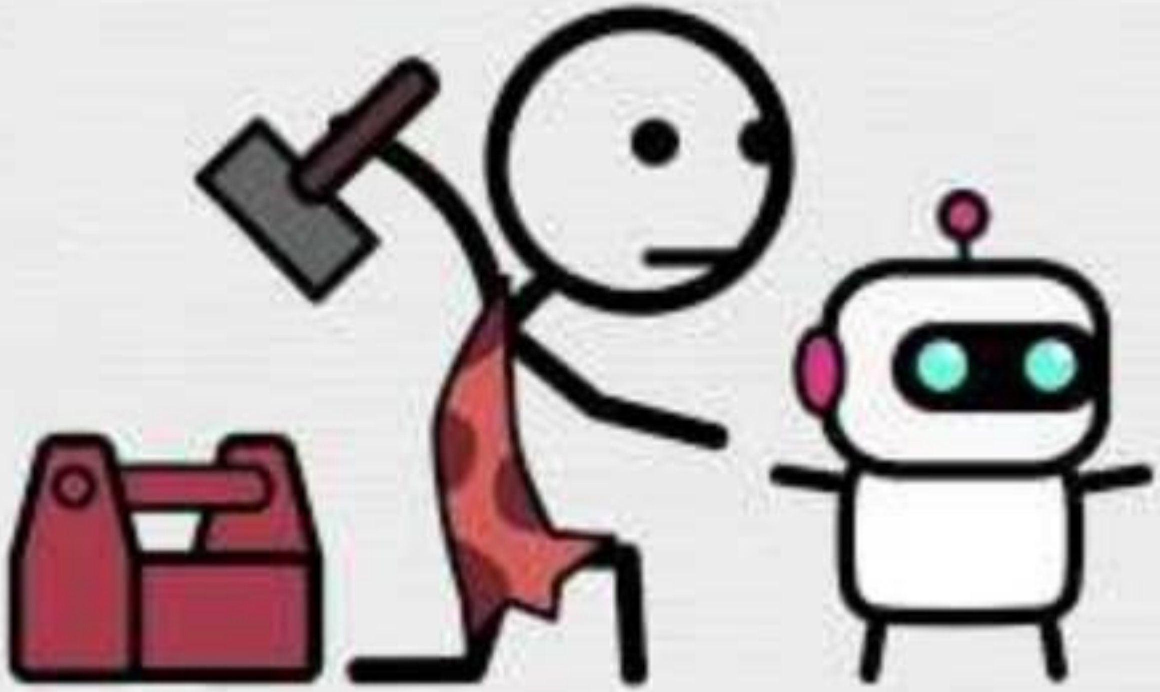
Testing

Test Set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$ → Performance Measure

Check $h(x) \approx f^*(x)$ in test set

(Generalization)

Break



Outline

- Machine Learning
  - Problems
  - Supervised Learning
- **Decision Trees**
  - Hypothesis class
  - Decision Tree Learning
  - Pruning
  - Data Preprocessing

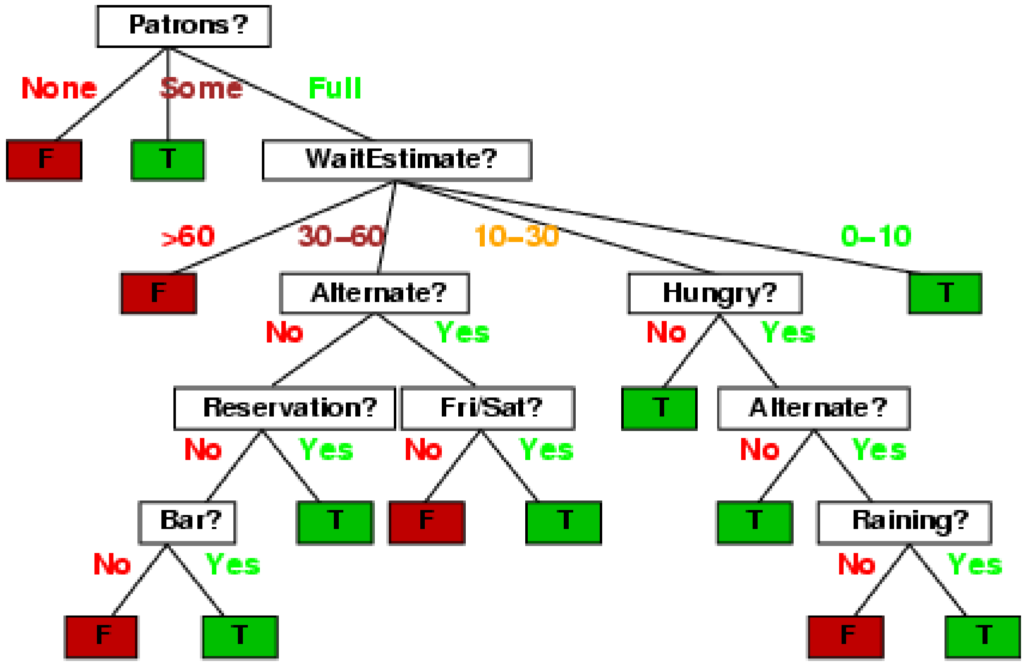
Decision Trees

- A decision tree represents a function that takes as input a vector of attribute values and returns a "decision"—a single output value.
  - Simplification: discrete attributes and Boolean output.

- Reach a decision by performing a sequence of tests starting from the root node until a leaf node is reached.

- Basically, a **nested if-else**

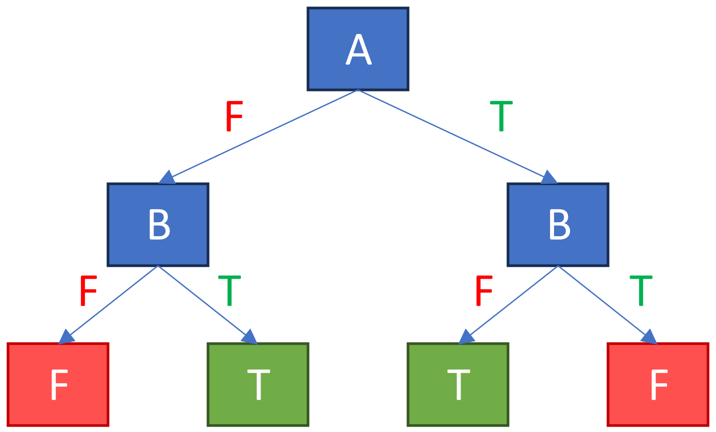
Example: Should we wait for a table?



Expressiveness

- Decision trees can express **any function** of the input attributes.
- Example: Boolean functions, each row → path from root to leaf

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



- There is a decision tree for any non-contradicting training set, but probably DT **won't generalize to new examples**.

The Size of the Hypothesis Class

How many distinct decision trees with n Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows

= 2^{2^n}

A	B		h(A,B)
F	F		T / F
F	T		T / F
T	F		T / F
T	T		T / F

With **6 Boolean attributes**, there are 18,446,744,073,709,551,616 trees

Finding a Good Decision Tree

Construct a decision tree by recursively **selecting the most informative attribute** to make decision.

Example: Should we wait for a table?

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time in minutes (0-10, 10-30, 30-60, >60)

Example: Should we wait for a table?

Collected data:

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

You are hungry, it's Friday, raining, …; **Should you wait?**

Choosing an Attribute

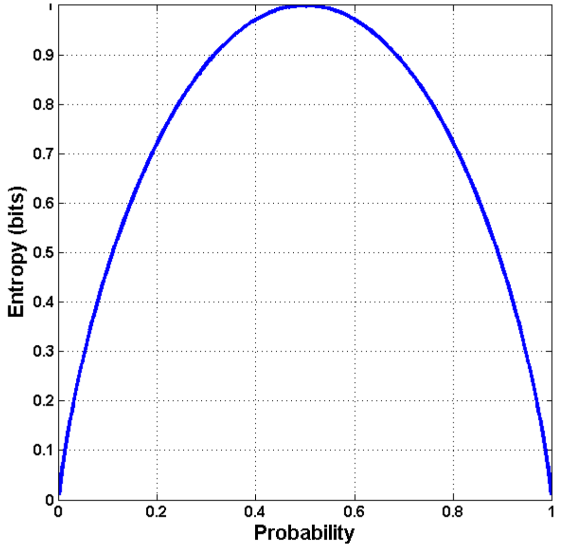
Which one is the most informative attribute in terms of making decision?

Ideally: we want to select an attribute that splits the examples into “all positive” or “all negative”

In general, how do we measure informativeness?

Background: Entropy

Maximum randomness,
Maximum entropy

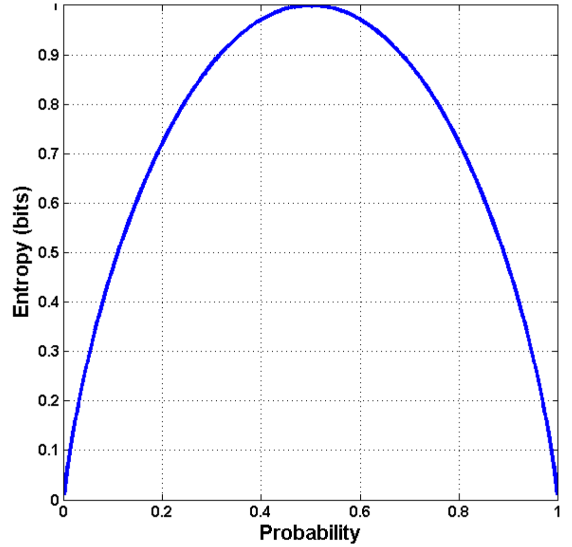


- Entropy is a measure of randomness
- Let $v_1, \ldots, v_k$ be outcomes and $P(v_1), \ldots, P(v_k)$ their probabilities
- Then, the entropy is defined as
  - $I\big(P(v_1), \ldots, P(v_k)\big) = -\sum_{i=1}^{k} P(v_i) \log_2 P(v_i)$

- Let a dataset contains p positive and n negative examples:
  - Probabilities $P(pos) = \frac{p}{p+n}$ and $P(neg) = \frac{n}{p+n}$
  - Entropy $I\big(P(pos), P(neg)\big) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$

Background: Entropy

Maximum randomness,
Maximum entropy



Which one has a higher entropy?

Information Gain

- Let attribute *A* have *v* distinct values.
- Attribute A divides the training set *E* into subsets $E_1, \ldots, E_v$ according to the value for *A*.
- In subset E_i, let there be p_i positive and n_i negative examples
  - Probability $\frac{p_i+n_i}{p+n}$ of being in this subset
  - Entropy $I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$.
- Average entropy after dividing, also called "remainder"
  - $remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$
- Information Gain (IG) or reduction in entropy

Expected reduction in entropy

  - $IG(A) = \boxed{I\left(\frac{p}{p+n}, \frac{n}{p+n}\right)} - \boxed{remainder(A)}$

Entropy before dividing *Average entropy after dividing*

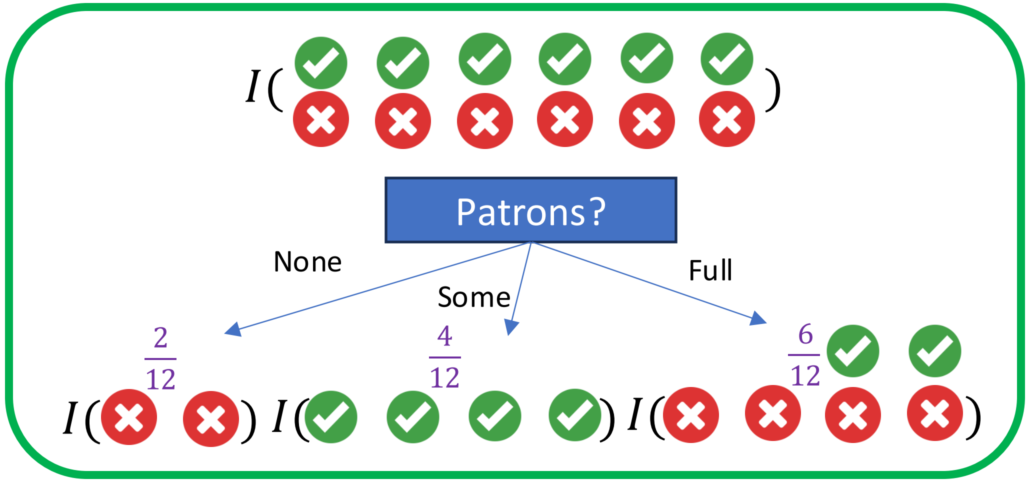
Information Gain

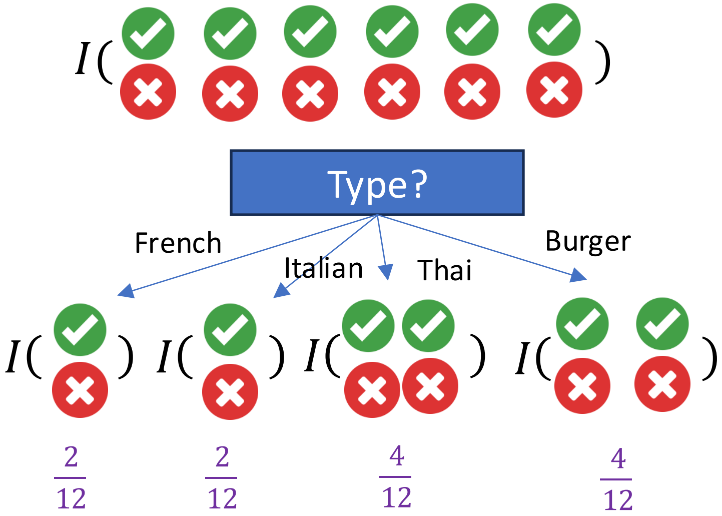
$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

$$IG(A) = \boxed{I\left(\frac{p}{p+n}, \frac{n}{p+n}\right)} - \boxed{remainder(A)}$$

Entropy of this node

Entropy of children nodes





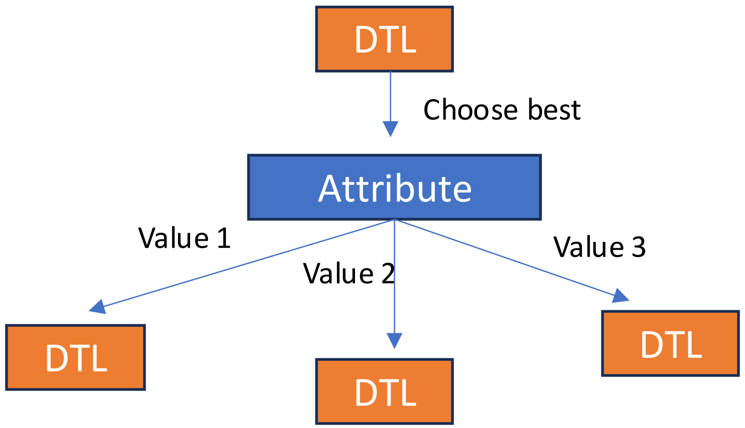
Decision Tree Learning – Key ideas

- Greedy, top-down, recursive algorithm

- Use Information Gain (or other measure) to choose the best attribute to divide the training set. In the code:
  best = **choose_attribute**(attributes, examples)

- Recursive: For each value of a chosen attribute, use DTL again on corresponding subset of examples

- Terminal conditions: return default value, or mode (i.e. majority).



Decision Tree Learning

with Information Gain

```
def DTL(examples, attributes, default):
    if examples is empty: return default
    if examples have the same classification:
        return classification
    if attributes is empty:
        return mode(examples)
    best = choose_attribute(attributes, examples)
    tree = a new decision tree with root best
    for each value v_i of best:
        examples_i = {rows in examples with best = v_i}
        subtree = DTL(examples_i, attributes – best, mode(examples))
        add a branch to tree with label v_i and subtree subtree
```

Terminal conditions

Find the best attribute from maximal information gain

Loop through all values of the attribute
Collect samples with this value
Run DTL on collected samples

Connect parent node with child node

Decision Tree Learning

with Information Gain

def **DTL**(examples, attributes, default):
  if examples is empty: return default
  if examples have the same classification:
    return classification
  if attributes is empty:
    return **mode**(examples)
  best = **choose_attribute**(attributes, examples)
  tree = a new decision tree with root best
  for each value v_i of best:
    $examples_i$ = {rows in examples with best = v_i}
    subtree = **DTL**($examples_i$, attributes – best, **mode**(examples))
    add a branch to tree with label v_i and subtree subtree

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

(2,4,6) (None, Some, Full)   (2,2,4,4) (French, Italian, Thai, Burger)

Patrons: (2,4,6)

2F   4T   2T, 4F

$$IG(Patrons) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right)\right] = 0.541\ bits$$

Type: (2,2,4,4)

T,F   T,F   2T,2F   2T,2 F

$$IG(Type) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right)\right] = 0 \text{ bits}$$

$IG(Alt), IG(Bar), IG(Fri), IG(Hun), \ldots$

Decision Tree Learning

with Information Gain

def **DTL**(examples, attributes, default):

  if examples is empty: return default

  if examples have the same classification:

    return classification

  if attributes is empty:

    return **mode**(examples)

  best = **choose_attribute**(attributes, examples)

  tree = a new decision tree with root best

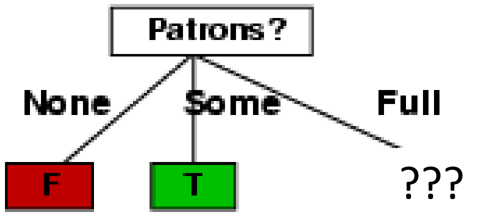
  for each value v_i of best:

    $examples_i$ = {rows in examples with best = v_i}

    subtree = **DTL**($examples_i$, attributes – best, **mode**(examples))

    add a branch to tree with label v_i and subtree subtree

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

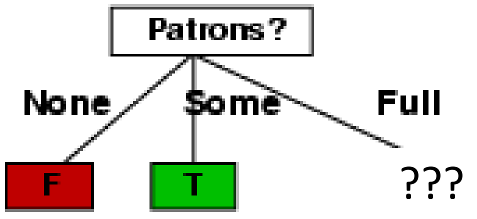


Decision Tree Learning

with Information Gain

```
def DTL(examples, attributes, default):
    if examples is empty: return default
    if examples have the same classification:
        return classification
    if attributes is empty:
        return mode(examples)
    best = choose_attribute(attributes, examples)
    tree = a new decision tree with root best
    for each value v_i of best:
        examples_i = {rows in examples with best = v_i}
        subtree = DTL(examples_i, attributes – best, mode(examples))
        add a branch to tree with label v_i and subtree subtree
```

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2	T	F	F	T	[redacted]	\$	F	F	Thai	30–60	F
X_3	[redacted]										
X_4	T	F	T	T	[redacted]	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	[redacted]	\$\$\$	F	T	French	>60	F
X_6	[redacted]										
X_7	[redacted]										
X_8	[redacted]										
X_9	F	T	T	F	[redacted]	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	[redacted]	\$\$\$	F	T	Italian	10–30	F
X_{11}	[redacted]										
X_{12}	T	T	T	T	[redacted]	\$	F	F	Burger	30–60	T



$IG(Alt), IG(Bar), IG(Fri), IG(Hun), \ldots$ except $IG(Pat)$

Decision Tree Learning

with Information Gain

def **DTL**(examples, attributes, default):

  if examples is empty: return default

  if examples have the same classification:

    return classification

  if attributes is empty:

    return **mode**(examples)

  best = **choose_attribute**(attributes, examples)

  tree = a new decision tree with root best

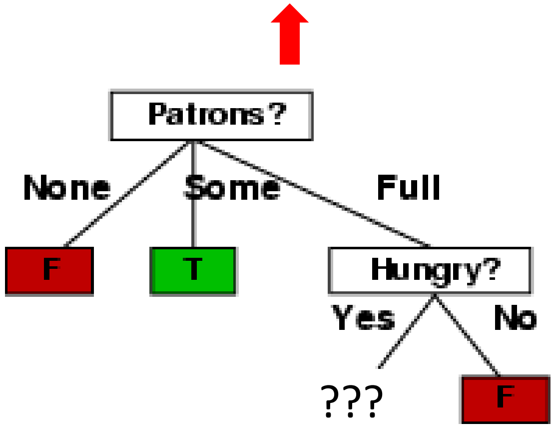
  for each value v_i of best:

    $examples_i$ = {rows in examples with best = v_i}

    subtree = **DTL**($examples_i$, attributes – best, **mode**(examples))

    add a branch to tree with label v_i and subtree subtree

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2	T	F	F	T	[redacted]	\$	F	F	Thai	30–60	F
X_3	[redacted]										
X_4	T	F	T	T	[redacted]	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	[redacted]	\$\$\$	F	T	French	>60	F
X_6	[redacted]										
X_7	[redacted]										
X_8	[redacted]										
X_9	F	T	T	F	[redacted]	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	[redacted]	\$\$\$	F	T	Italian	10–30	F
X_{11}	[redacted]										
X_{12}	T	T	T	T	[redacted]	\$	F	F	Burger	30–60	T

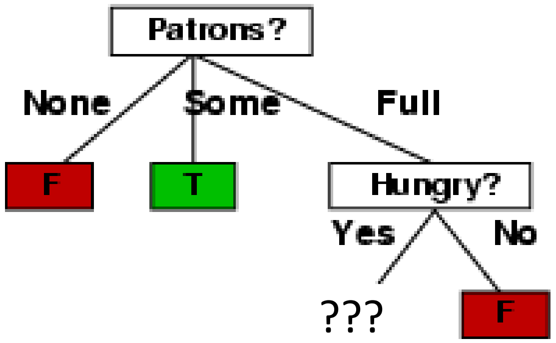


Decision Tree Learning

with Information Gain

```
def DTL(examples, attributes, default):
    if examples is empty: return default
    if examples have the same classification:
        return classification
    if attributes is empty:
        return mode(examples)
    best = choose_attribute(attributes, examples)
    tree = a new decision tree with root best
    for each value v_i of best:
        examples_i = {rows in examples with best = v_i}
        subtree = DTL(examples_i, attributes – best, mode(examples))
        add a branch to tree with label v_i and subtree subtree
```

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2	T	F	F	[redacted]	[redacted]	$	F	F	Thai	30–60	F
X_3	[redacted]										
X_4	T	F	T	[redacted]	[redacted]	$	F	F	Thai	10–30	T
X_5	[redacted]										
X_6	[redacted]										
X_7	[redacted]										
X_8	[redacted]										
X_9	[redacted]										
X_{10}	T	T	T	[redacted]	[redacted]	$$$	F	T	Italian	10–30	F
X_{11}	[redacted]										
X_{12}	T	T	T	[redacted]	[redacted]	$	F	F	Burger	30–60	T



$IG(Alt), IG(Bar), IG(Type), \ldots$ except $IG(Pat), IG(Hun)$

Decision Tree Learning

with Information Gain

def **DTL**(examples, attributes, default):

  if examples is empty: return default

  if examples have the same classification:

    return classification

  if attributes is empty:

    return **mode**(examples)

  best = **choose_attribute**(attributes, examples)

  tree = a new decision tree with root best

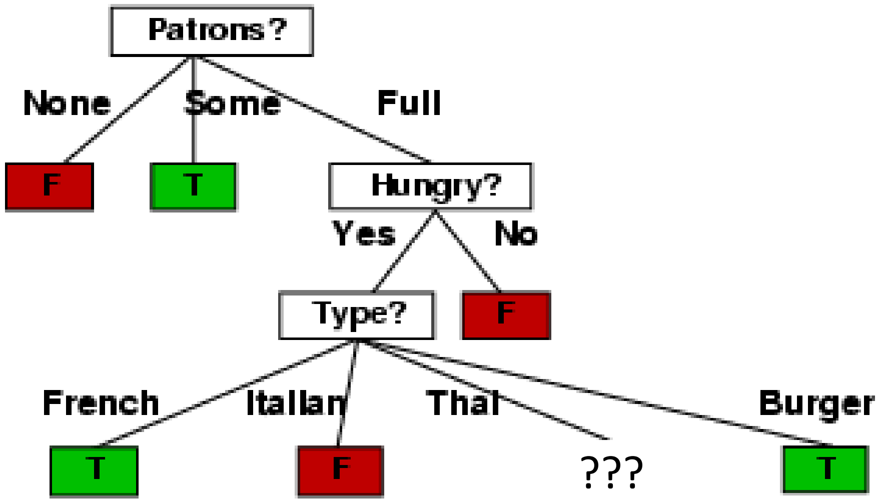
  for each value v_i of best:

    $examples_i$ = {rows in examples with best = v_i}

    subtree = **DTL**($examples_i$, attributes – best, **mode**(examples))

    add a branch to tree with label v_i and subtree subtree

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2	T	F	F	[redacted]		$	F	F	Thai	30–60	F
X_3	[redacted]										
X_4	T	F	T	[redacted]		$	F	F	Thai	10–30	T
X_5	[redacted]										
X_6											
X_7											
X_8											
X_9											
X_{10}	T	T	T	[redacted]		$$$	F	T	Italian	10–30	F
X_{11}	[redacted]										
X_{12}	T	T	T	[redacted]		$	F	F	Burger	30–60	T



There is no example for "French"

Since there is no mode (majority),

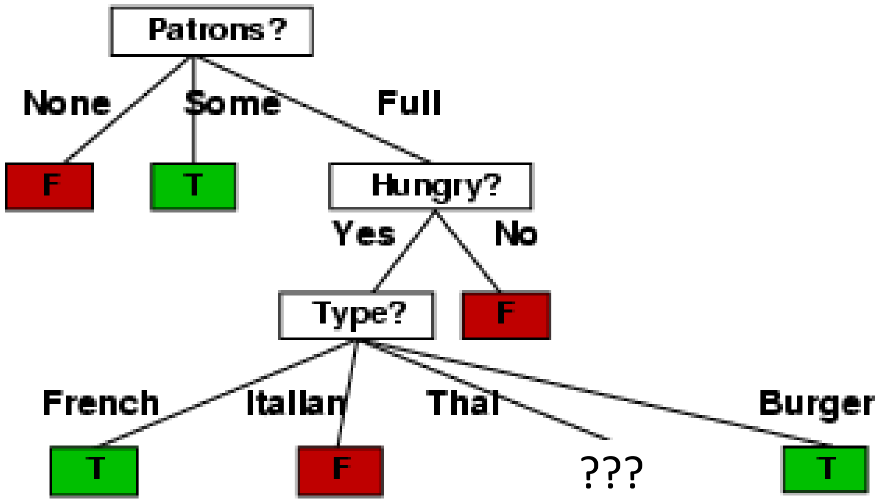
we select an arbitrary decision: "T"

Decision Tree Learning

with Information Gain

```
def DTL(examples, attributes, default):
    if examples is empty: return default
    if examples have the same classification:
        return classification
    if attributes is empty:
        return mode(examples)
    best = choose_attribute(attributes, examples)
    tree = a new decision tree with root best
    for each value v_i of best:
        examples_i = {rows in examples with best = v_i}
        subtree = DTL(examples_i, attributes – best, mode(examples))
        add a branch to tree with label v_i and subtree subtree
```

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2	T	F	F	[redacted]		$	F	F	[redacted]	30–60	F
X_3	[redacted]										
X_4	T	F	T	[redacted]		$	F	F	[redacted]	10–30	T
X_5	[redacted]										
X_6											
X_7											
X_8											
X_9											
X_{10}											
X_{11}											
X_{12}											



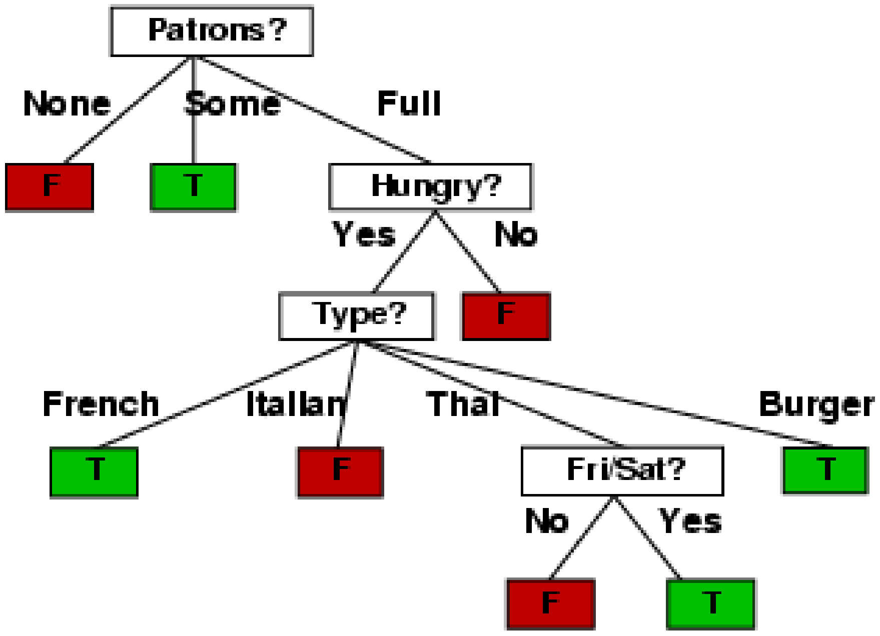
$IG(Alt), IG(Fri), \ldots$ except $IG(Pat), IG(Hun)\ , IG(Type)$

Decision Tree Learning

with Information Gain

```
def DTL(examples, attributes, default):
    if examples is empty: return default
    if examples have the same classification:
        return classification
    if attributes is empty:
        return mode(examples)
    best = choose_attribute(attributes, examples)
    tree = a new decision tree with root best
    for each value v_i of best:
        examples_i = {rows in examples with best = v_i}
        subtree = DTL(examples_i, attributes – best, mode(examples))
        add a branch to tree with label v_i and subtree subtree
```

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2	T	F	F	[redacted]		\$	F	F	[redacted]	30–60	F
X_3	[redacted]										
X_4	T	F	T	[redacted]		\$	F	F	[redacted]	10–30	T
X_5	[redacted]										
X_6											
X_7											
X_8											
X_9											
X_{10}											
X_{11}											
X_{12}											

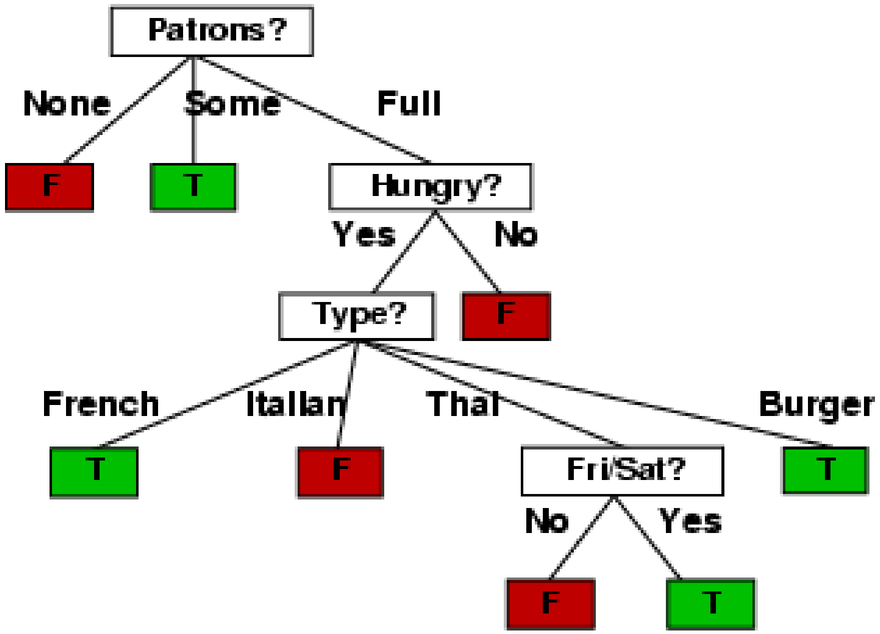


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	[redacted]										
X_2											
X_3											
X_4											
X_5											
X_6											
X_7											
X_8											
X_9											
X_{10}											
X_{11}											
X_{12}											

Decision Tree Learning

with Information Gain

def **DTL**(examples, attributes, default):
    if examples is empty: return default
    if examples have the same classification:
        return classification
    if attributes is empty:
        return **mode**(examples)
    best = **choose_attribute**(attributes, examples)
    tree = a new decision tree with root best
    for each value v_i of best:
        $examples_i$ = {rows in examples with best = v_i}
        subtree = **DTL**($examples_i$, attributes – best, **mode**(examples))
        add a branch to tree with label v_i and subtree subtree



Poll Everywhere

# Bad Keywords	Contains Link	Email Length	Spam (Target)
Many (10+)	Yes	Long (100+ words)	Yes
Few (1-5)	No	Short (up to 50 words)	No
Many (10+)	Yes	Long (100+ words)	Yes
Moderate (6-9)	No	Medium (51-99 words)	No
Few (1-5)	Yes	Medium (51-99 words)	Yes
Moderate (6-9)	No	Long (100+ words)	No

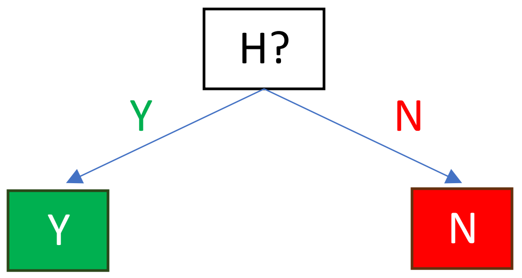
Which attribute should you choose for the first split in the decision tree to best reduce entropy and achieve the most informative split?

Poll Everywhere

# Bad Keywords	Contains Link	Email Length	Spam (Target)
Many (10+)	Yes	Long (100+ words)	Yes
Few (1-5)	No	Short (up to 50 words)	No
Many (10+)	Yes	Long (100+ words)	Yes
Moderate (6-9)	No	Medium (51-99 words)	No
Few (1-5)	Yes	Medium (51-99 words)	Yes
Moderate (6-9)	No	Long (100+ words)	No

Which attribute should you choose for the first split in the decision tree to best reduce entropy and achieve the most informative split?

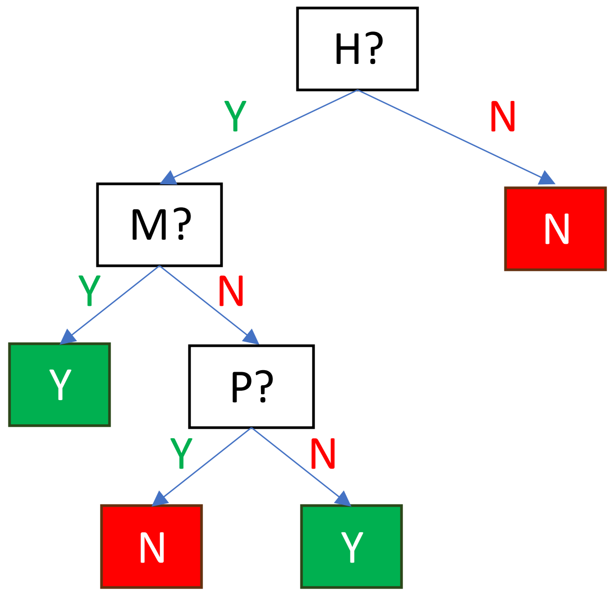
Overfitting



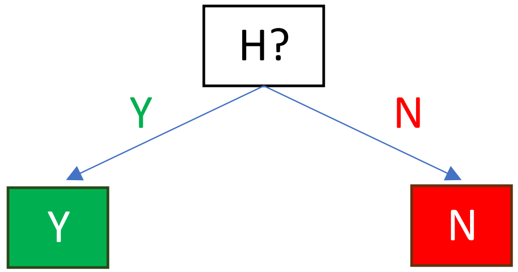
Decision Trees performance is perfect on training data, but worse on test data

- DT captures the data perfectly, **including the noise**

Hungry?	Moon aligns with the sun?	Problem set due?	Eat?
Yes	Yes	Yes	Yes
Yes	No	Yes	No
Yes	No	No	Yes
No	Yes	No	No
No	No	No	No



Occam's Razor



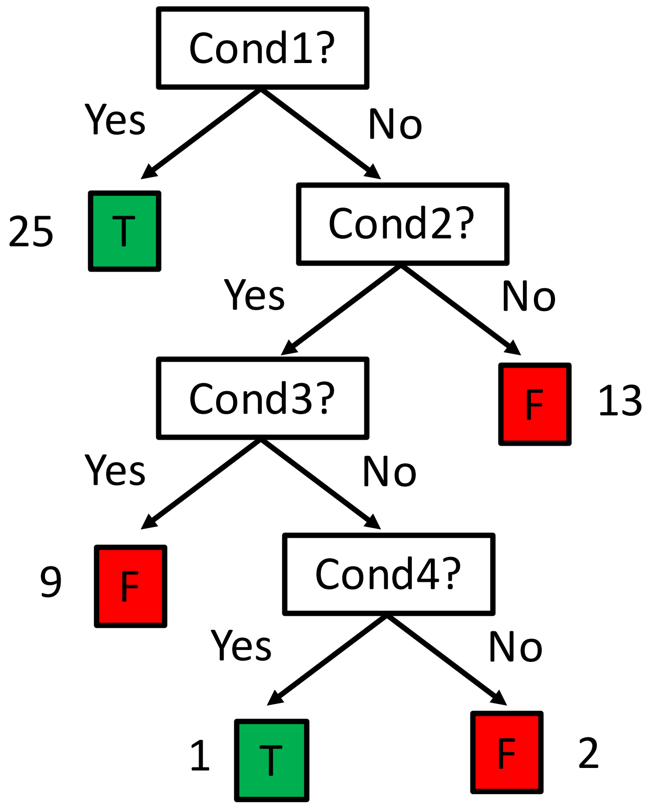
- Prefer short/simple hypotheses. Why?
- In favor:
  - **Short/simple** hypothesis that fits the data is **unlikely** to be **coincidence**
  - **Long/complex** hypothesis that fits the data **may be coincidence**
- Against:
  - Many ways to define small sets of hypotheses (e.g., trees with prime number of nodes that uses attribute beginning with "Z")
  - Different hypotheses representations may be used instead

Pruning

- Pruning is the process of **reducing the size of a decision tree by removing parts of the tree**. It prevent nodes from being split even when if fails to cleanly separate examples.
- Types:
  - **Min-sample leaf** pruning refers to setting a minimum threshold for the number of samples required to be in a leaf node. If a leaf node has fewer than this minimum number of samples, it may be pruned.
  - **Max depth** pruning involves limiting the maximum depth of the decision tree. The depth of a tree is the length of the longest path from the root node to a leaf node.
- Pruning can be done when tree is being created or on an existing tree.

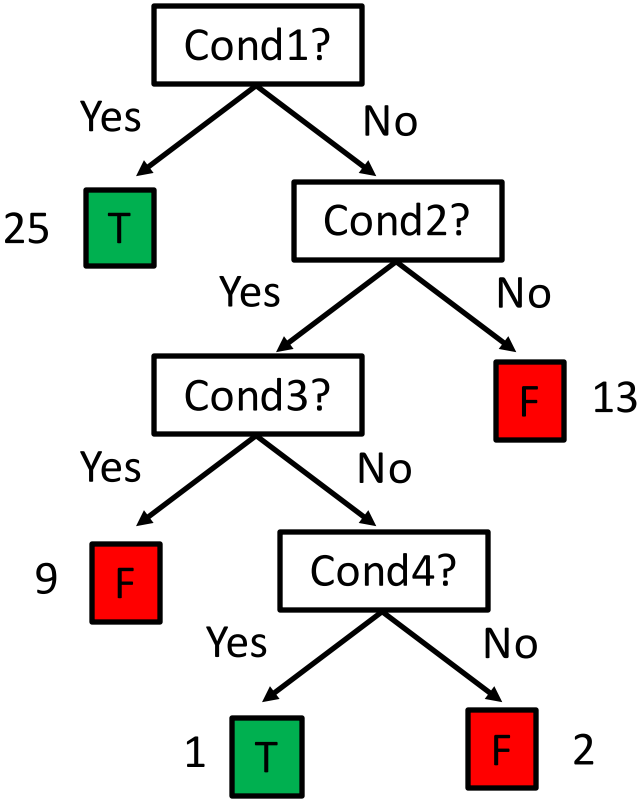
Pruning

Example: pruning an existing tree

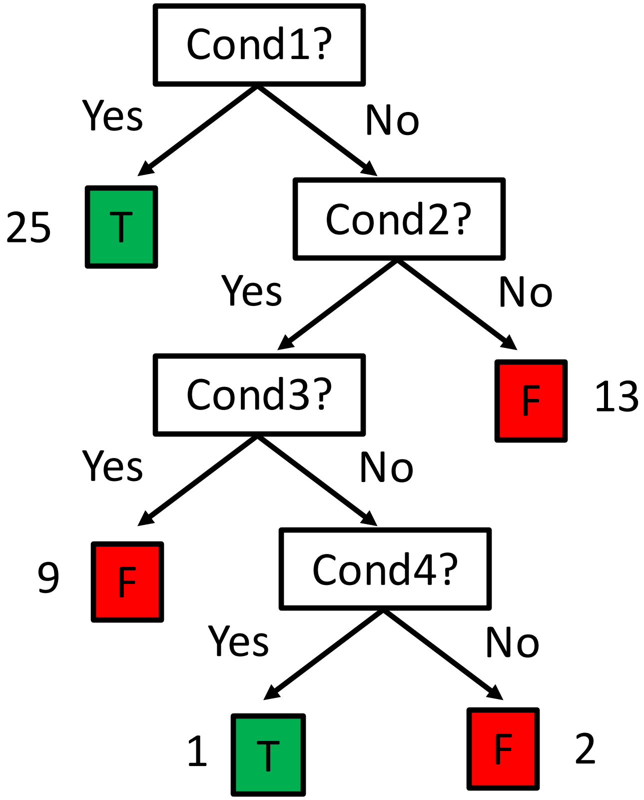


Pruning

Example: pruning an existing tree

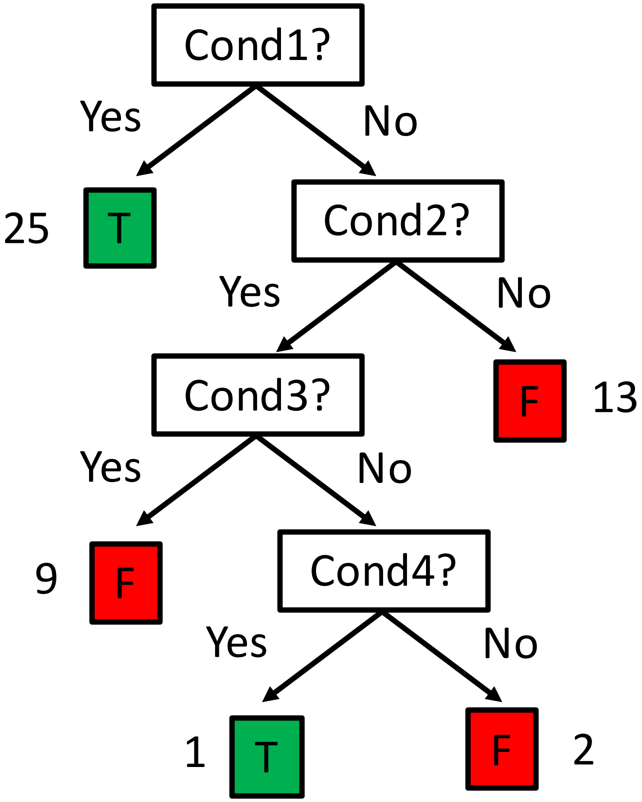


Min samples leaf = 5

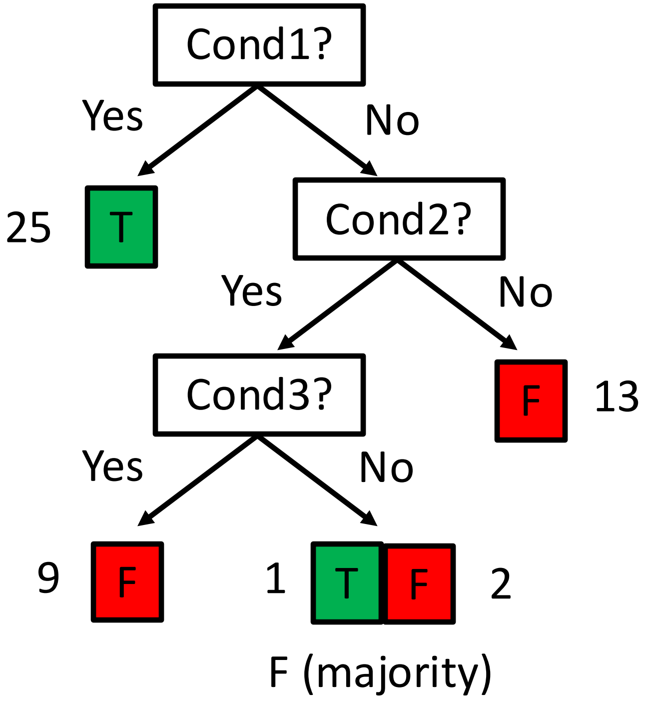


Pruning

Example: pruning an existing tree

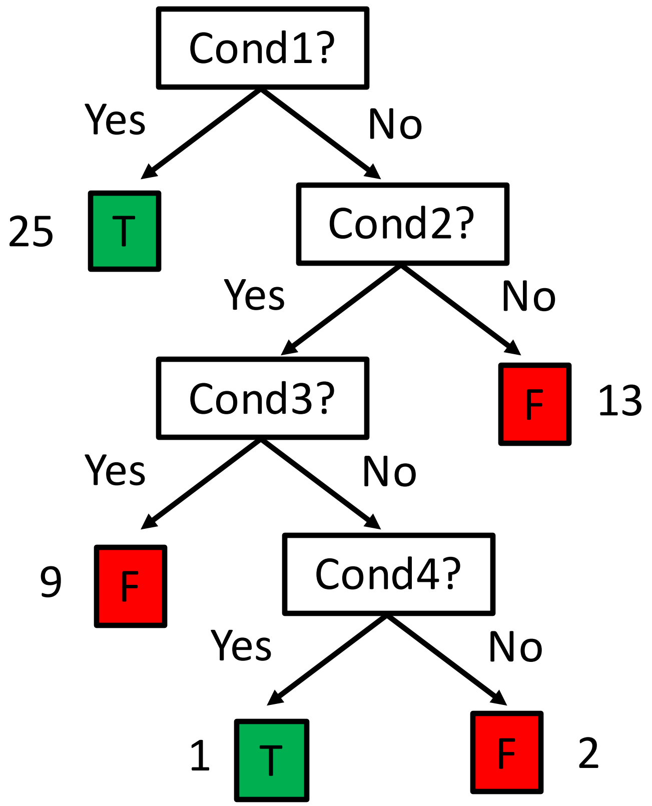


Min samples leaf = 5

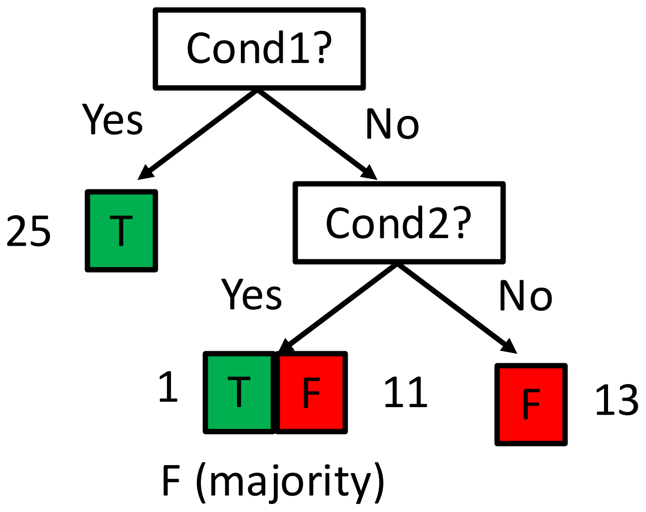


Pruning

Example: pruning an existing tree



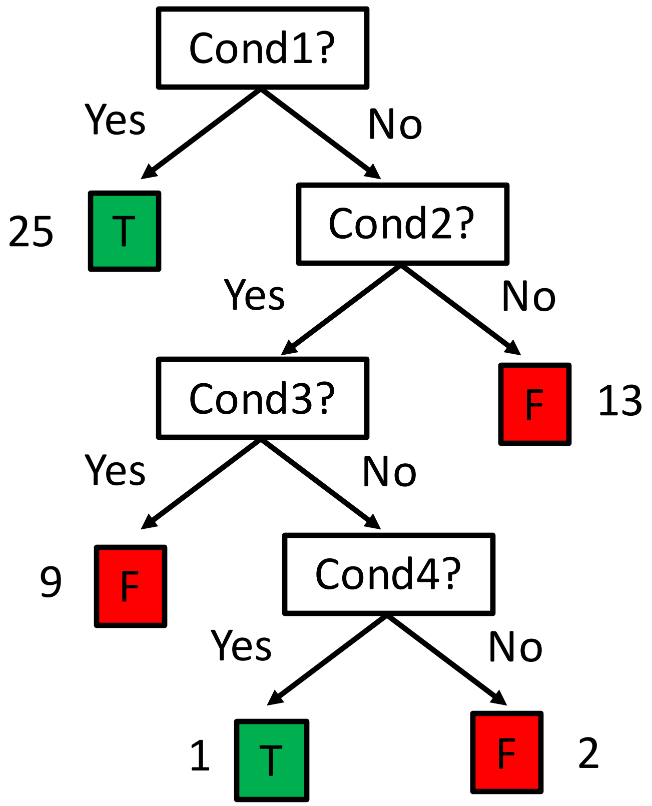
Min samples leaf = 5



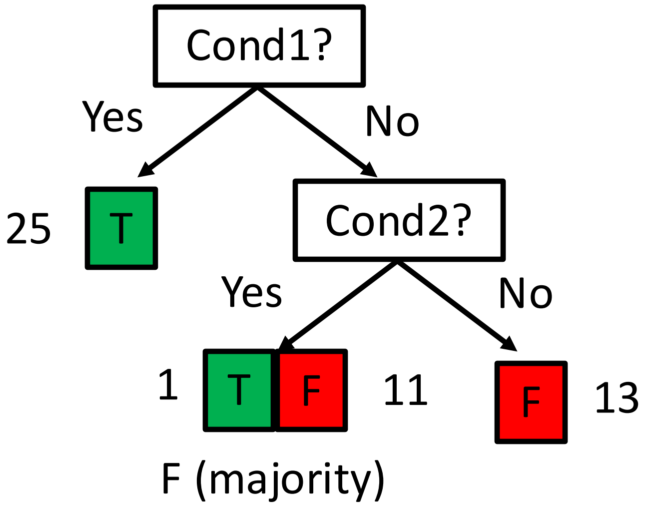
Done!

Pruning

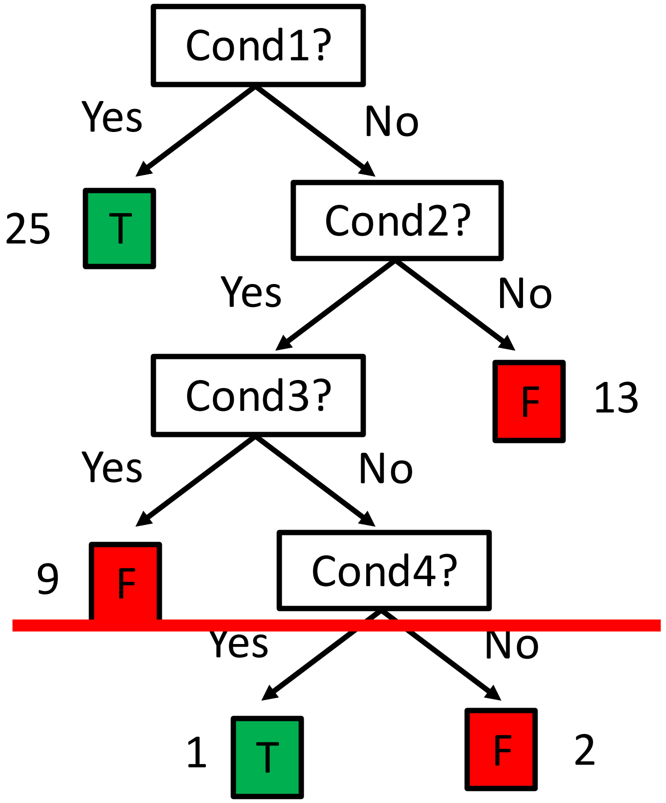
Example: pruning an existing tree



Min samples leaf = 5

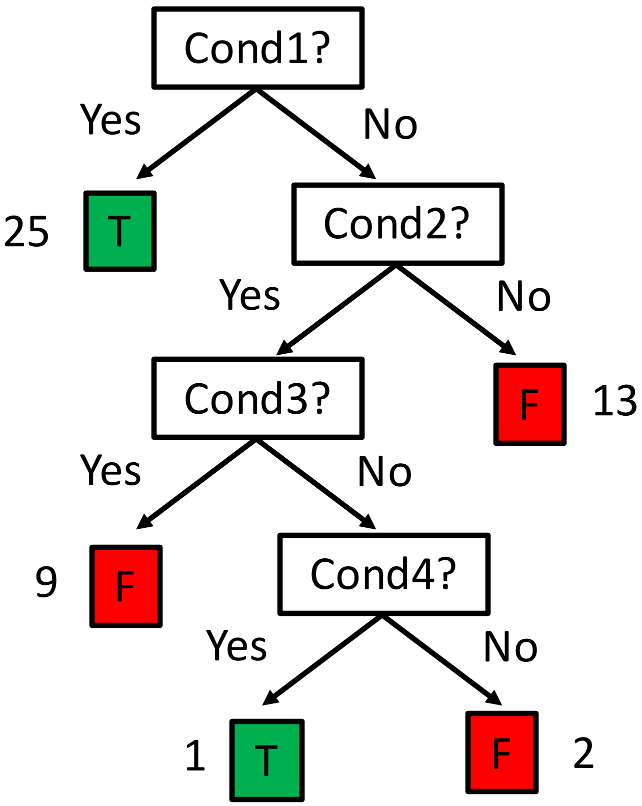


Max depth = 3

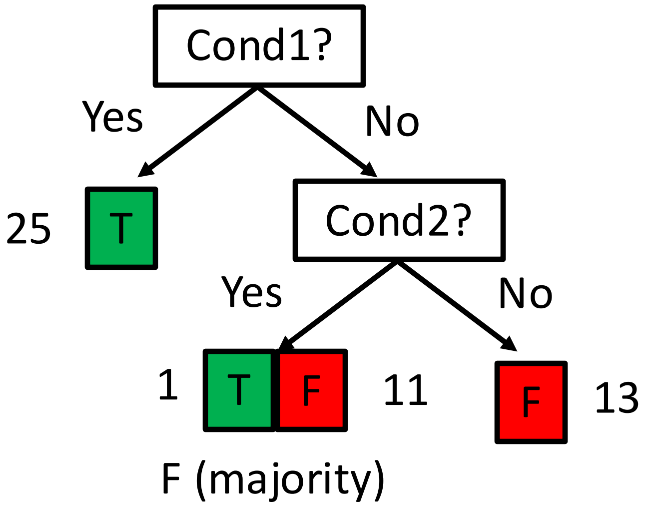


Pruning

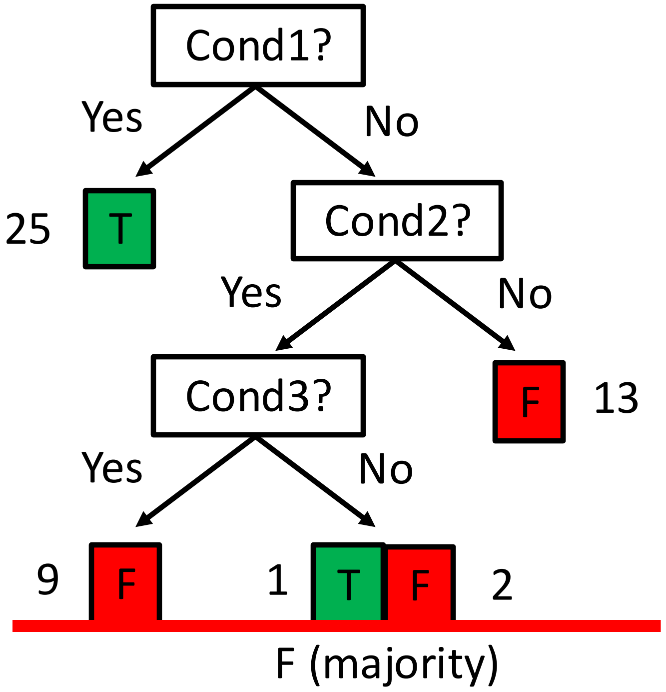
Example: pruning an existing tree



Min samples leaf = 5



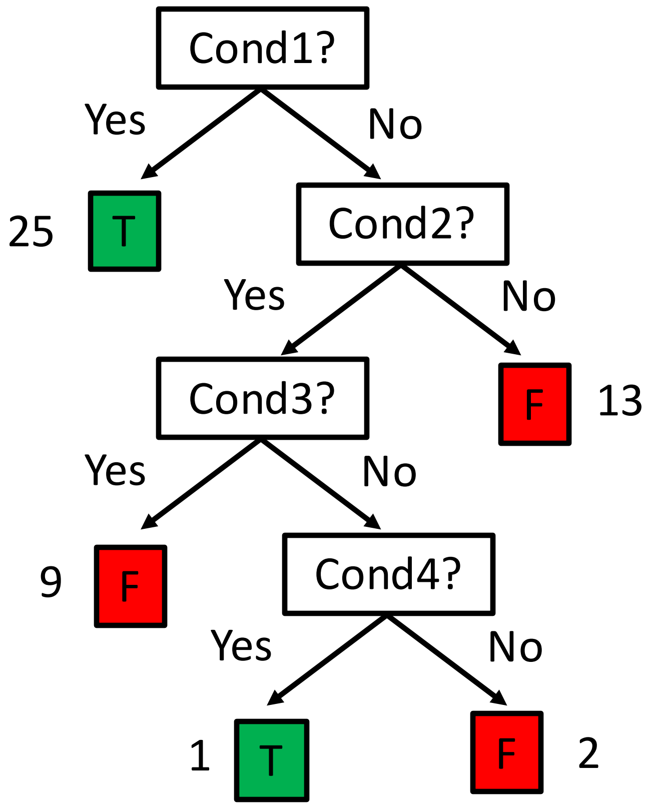
Max depth = 3



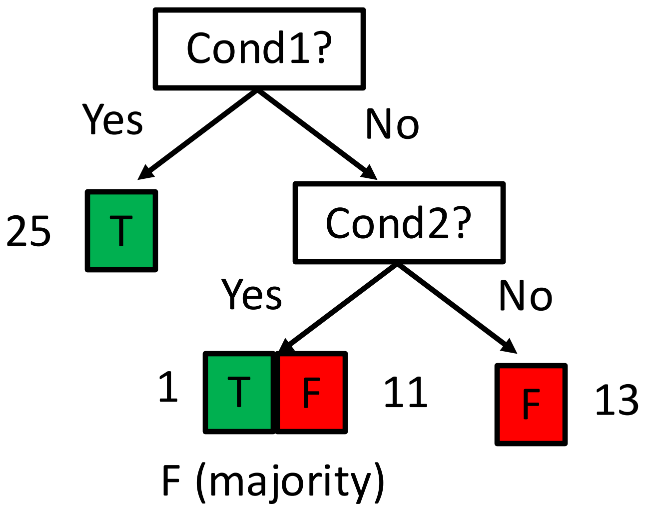
Done!

Pruning

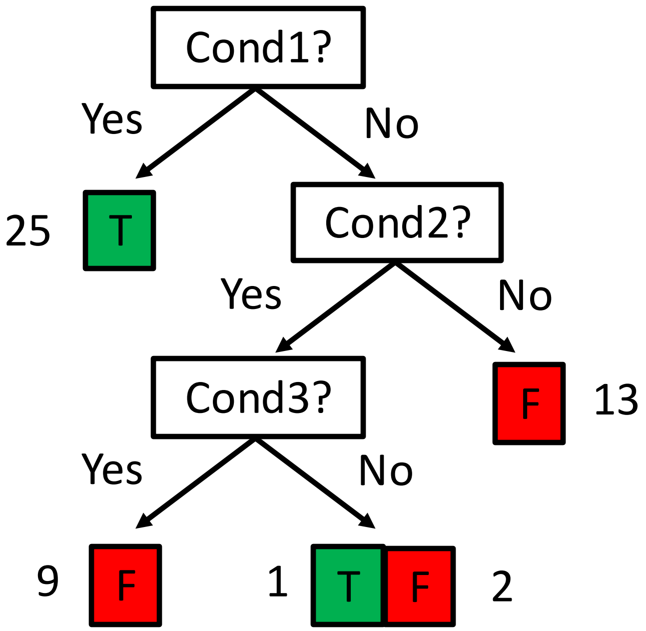
Example: pruning an existing tree



Min samples leaf = 5



Max depth = 3



The sample that is likely due to **noise** (1T) is ignored!

Results in a **smaller tree** which may have **higher accuracy**

Data Preprocessing

- **Continuous values**
  - Partition the values into a discrete set of intervals.
  - Example:
    - Estimated waiting time (minutes): 0-10, 10-30, 30-60, >60
    - Age (year): 0-12, 12-25, 25-40, 40-60, 60-80, >80
- **Missing values**
  - Assign the most common value of the attribute
  - Assign the most common value of the attribute with the same output
  - Assign probability to each possible value and sample
  - Drop the attribute
  - Drop the rows
  - …

Summary

- Machine Learning
  - Types: supervised, unsupervised, semi-supervised, reinforcement
  - Supervised Learning: dataset, hypothesis class, learning algorithms, performance measure
- Decision Trees
  - Hypothesis class
  - Decision Tree Learning (DTL): greedy, top-down, recursive algorithm
  - Entropy and Information Gain
  - Pruning: min-sample, max-depth

Coming Up Next Week

- Linear Regression

To Do

- **Lecture Training 4**
  - +250 EXP
  - +100 Early bird bonus
- **(Graded) Tutorial starts this week!**
- **Problem Set 2**
  - Will be released later today!