Tree Based Methods

Tree Based Methods

- The next few sections of the course will focus on tree based methods.
- There are 3 main methods:
    - Decision Trees
    - Random Forests
    - Boosted Trees

Tree Based Methods

- Each of these methods stems from the basic decision tree algorithm.
- We will cover each of these methods in their own section and then test your new skills with a project exercise after learning about all 3 method types.

Tree Based Methods

- Related Reading in ISLR
    - Chapter 8 covers tree-based methods.

Let’s get started!

Decision Trees

Theory and Intuition: History

Tree Based Methods

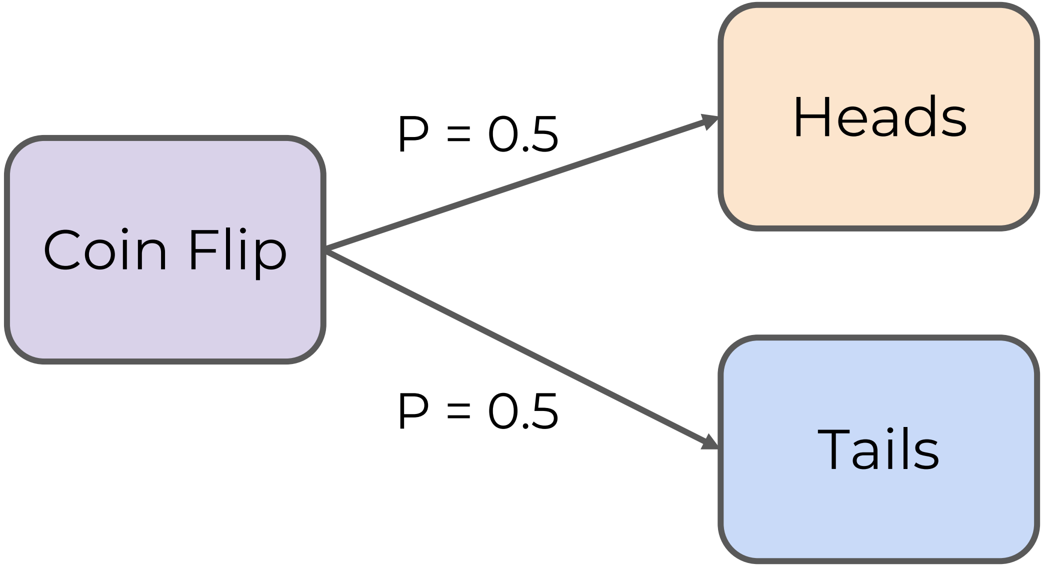
- While the use of basic decision trees for modeling choices and outcomes have been around for a very long time, statistical decision trees are a more recent development.
- Be careful to note the difference here!

Tree Based Methods

- The general term “decision tree” can refer to a flowchart mapping out outcomes:

Tree Based Methods

- The general term "decision tree" can refer to a flowchart mapping out outcomes:



Tree Based Methods

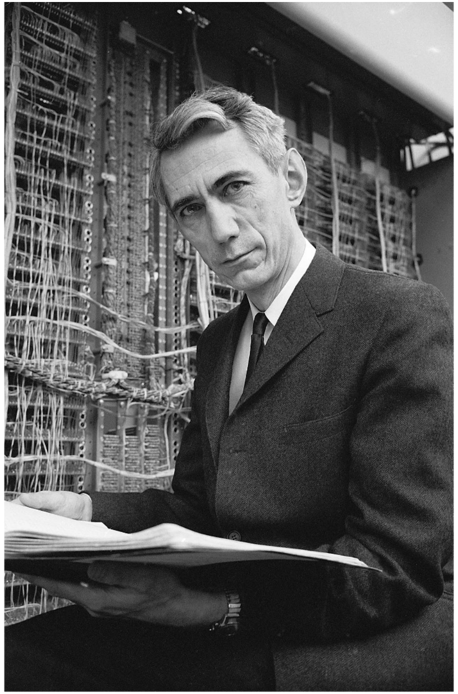
- Decision Tree Learning refers to the statistical modeling that uses a form of decision trees, where node splits are decided based on an information metric.
- Let’s dive deeper into the developments that lead to the ability to create predictions based on decision trees.

Tree Based Methods

- Fundamentally, decision trees and other tree based methods rely on the ability to **split** data based on **information** from features.
- This means we need a mathematical definition of **information** and the ability to measure it.

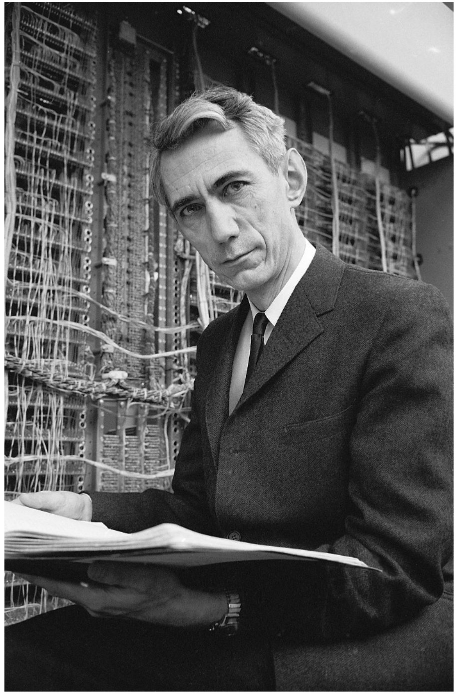
Tree Based Methods

- Claude Shannon is known as the “father of information theory”.
- Published “*A Mathematical Theory of Communication*” in 1948 in Bell System Technical Journal.



Tree Based Methods

- Later published as "*<u>The</u> Mathematical Theory of Communication*"
- Worked in many fields:
  - Circuit Design
  - Cryptography
  - Wearable Computers
  - Artificial Intelligence

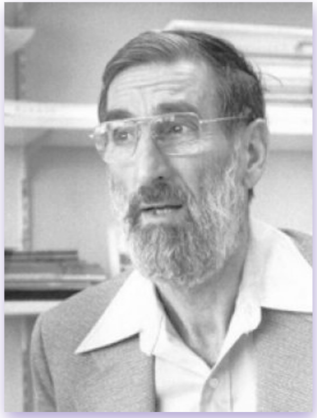


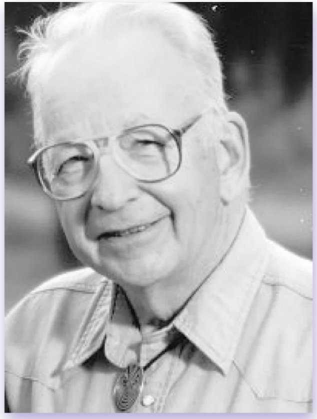
Tree Based Methods

- The ability to measure and define information will become more important as we learn the mathematics of how tree based methods are constructed.
- We will revisit this idea later on, for now, let’s move on to the development of decision trees.

Tree Based Methods

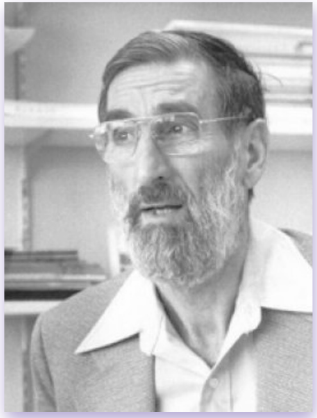
- 1963: First publication of regression tree algorithm by Morgan and Sonquist

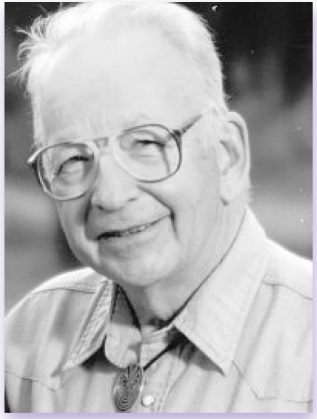




Tree Based Methods

- 1963: Morgan and Sonquist created piecewise-constant model with splits.



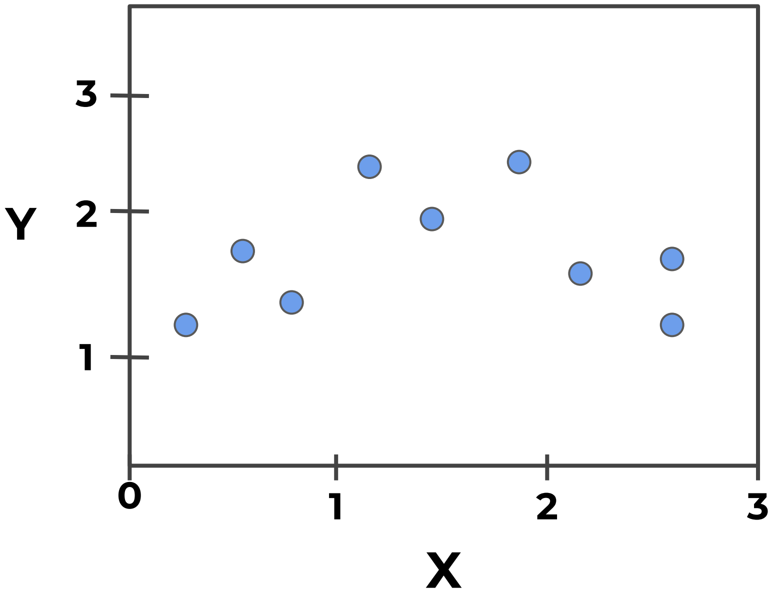


Tree Based Methods

- 1963: Piecewise-constant regression tree

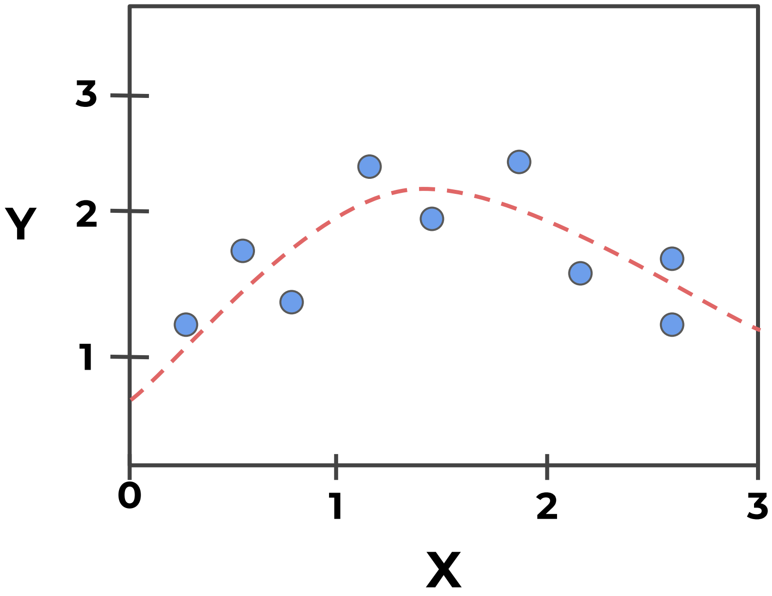
Tree Based Methods

- 1963: Piecewise-constant regression tree



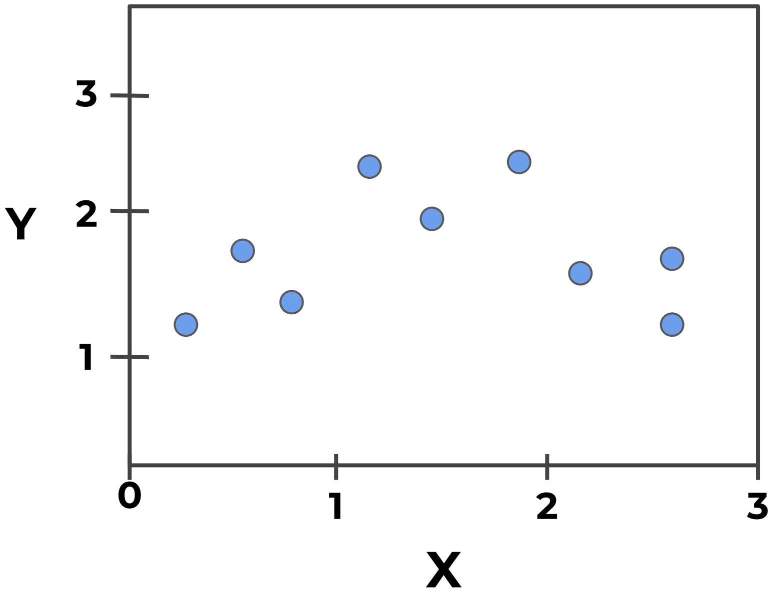
Tree Based Methods

- 1963: Piecewise-constant regression tree



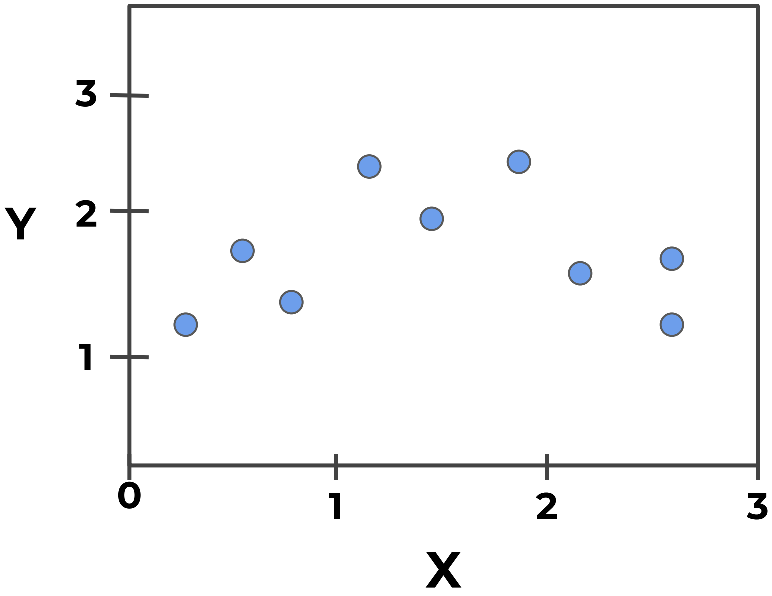
Tree Based Methods

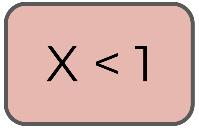
- 1963: Piecewise-constant regression tree



Tree Based Methods

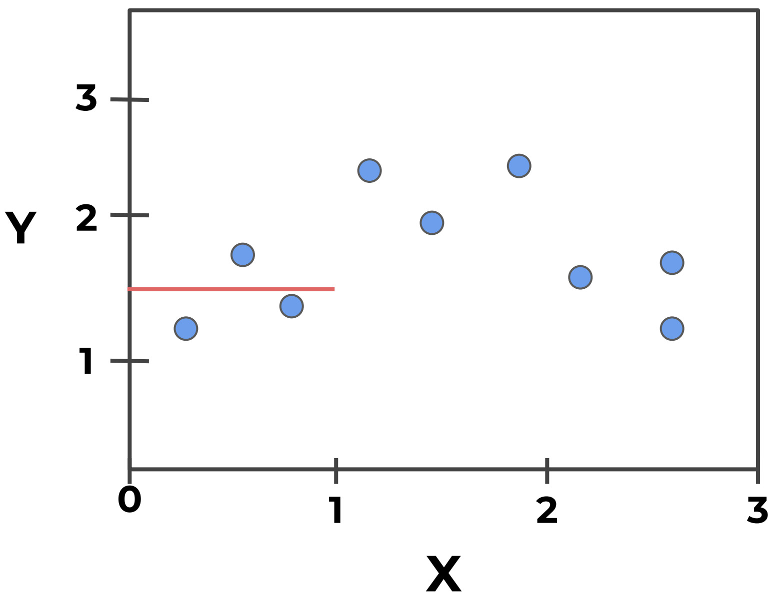
- 1963: Piecewise-constant regression tree

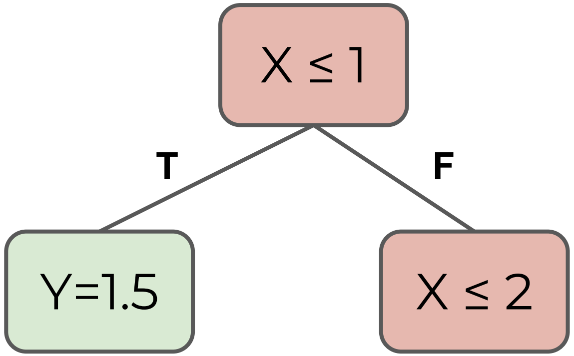




Tree Based Methods

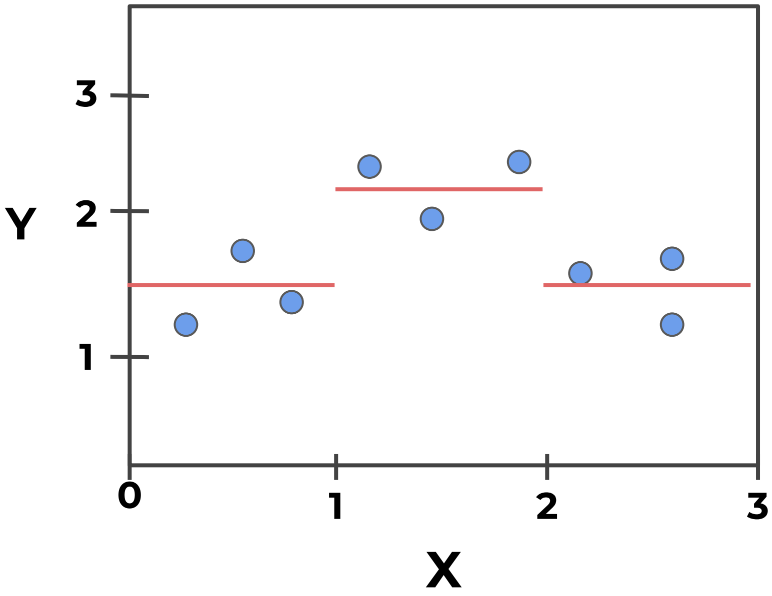
- 1963: Piecewise-constant regression tree

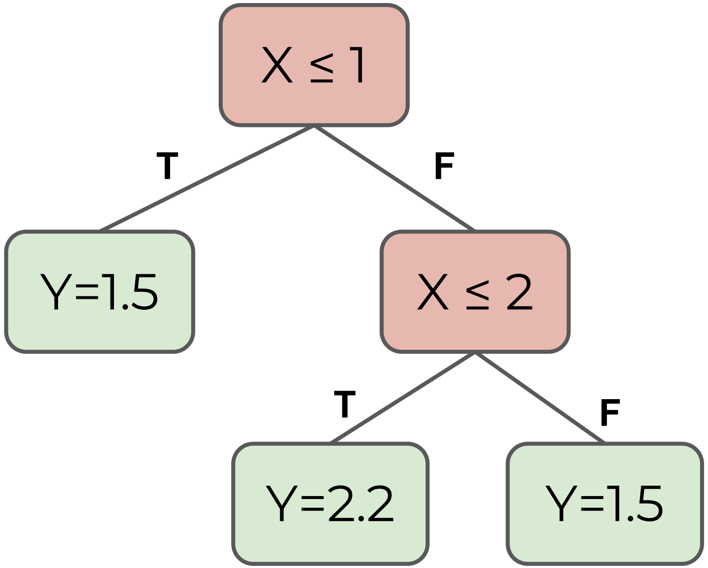




Tree Based Methods

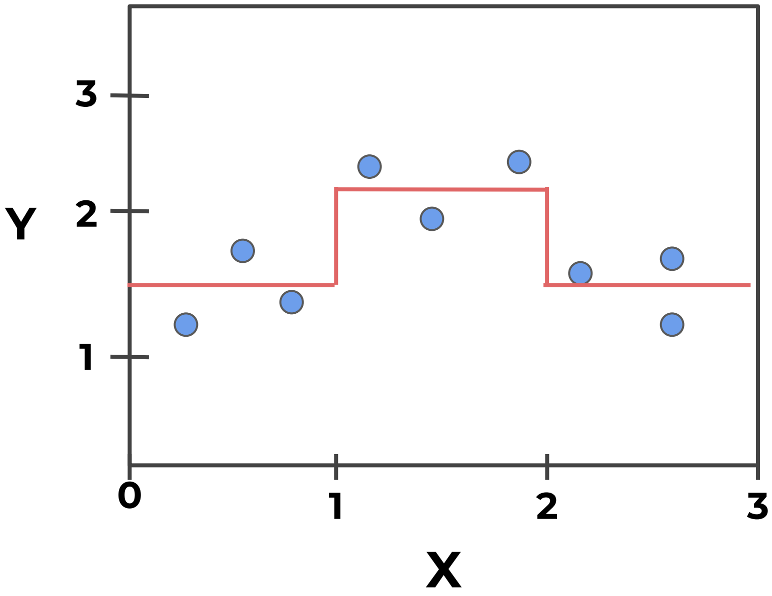
- 1963: Piecewise-constant regression tree

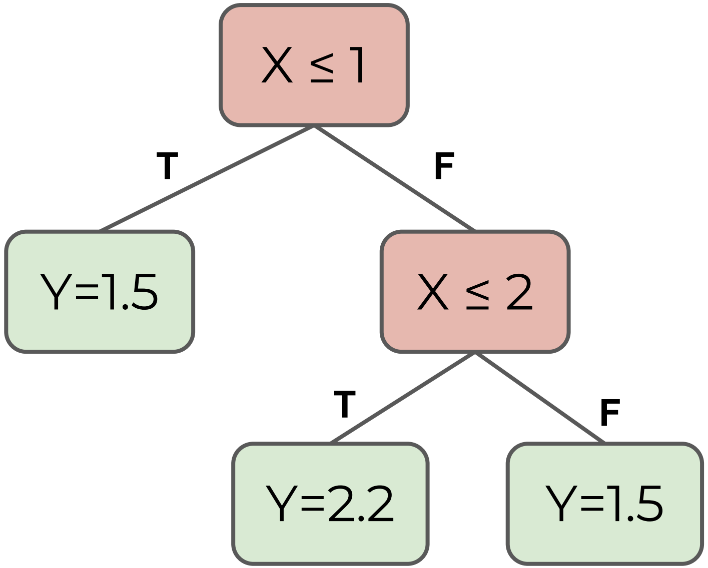




Tree Based Methods

- 1963: Piecewise-constant regression tree





Tree Based Methods

- In the 1963 paper, splits at each node **t** were decided based on **node impurity**, which was simply defined as an error metric:

$$\phi(t) = \sum_{i \in t} (y_i - \bar{y})^2$$

Tree Based Methods

- 1972: Robert Messenger and Lewis Mandell publish first classification tree algorithm with "*A model search technique for predictive nominal scale multivariate analysis.*"
- Split condition was named Theta Automatic Interaction Detection (THAID)

Tree Based Methods

- 1980: Gordon Kass publishes CHAID decision tree technique.
- Based on further work built on top of THAID algorithm from 1970s.
- CHAID: Chi-square automatic interaction detection .

Tree Based Methods

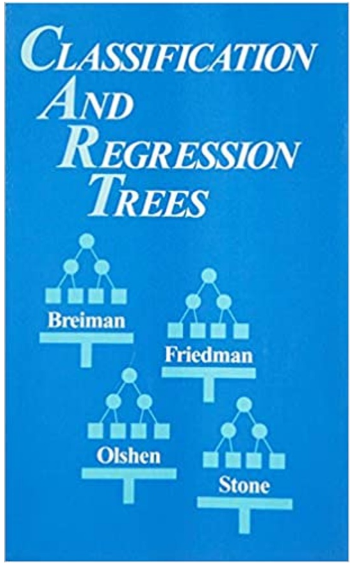
- 1970s: Leo Breiman and Charles Stone from Berkeley and Jerome Friedman and Richard Olshen from Stanford started developing the Classification and Regression tree (CART) based algorithms.

Tree Based Methods

- 1984: The CART book (Breiman et al.) is officially published, including a software implementation.
- CART was a huge leap forward in the practical usage of decision tree algorithm.
- CART based methods quickly became a standard (including scikit-learn!)

Tree Based Methods

- CART introduces many concepts:
  - Cross validation of Trees
  - Pruning Trees
  - Surrogate Splits
  - Variable Importance Scores
  - Search for Linear Splits



Tree Based Methods

- 1986: John Ross Quinlan developed ID3 decision tree algorithm based on the “gain ratio”.
- 1990s: Improved on ID3 with C4.5 (still very popular).
- 2000s: Released highly optimized commercial version C5.0 with various improvements.

Tree Based Methods

- Many of these improvements of basic decision trees were incorporated to other tree based methods such as random forests and gradient boosted trees.
- Let's move on to understanding the fundamental ideas behind a decision tree!

Decision Trees

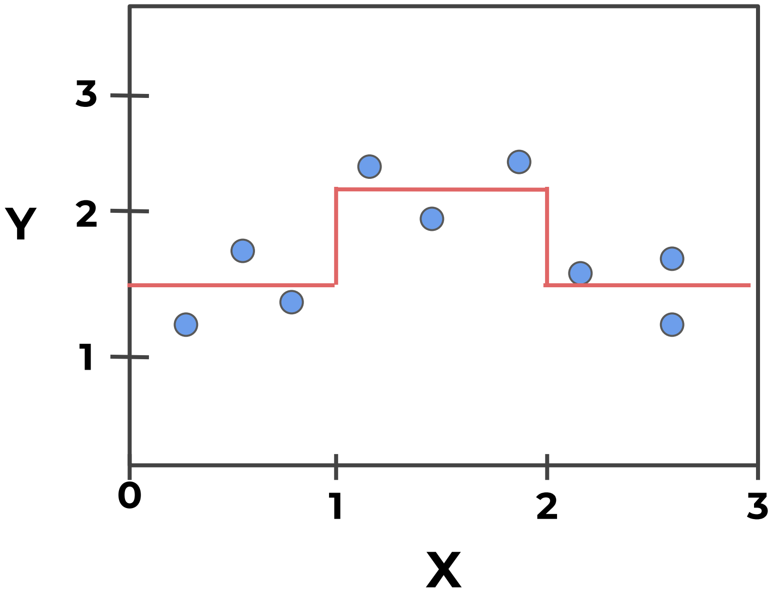
Theory and Intuition: Decision Tree Basics

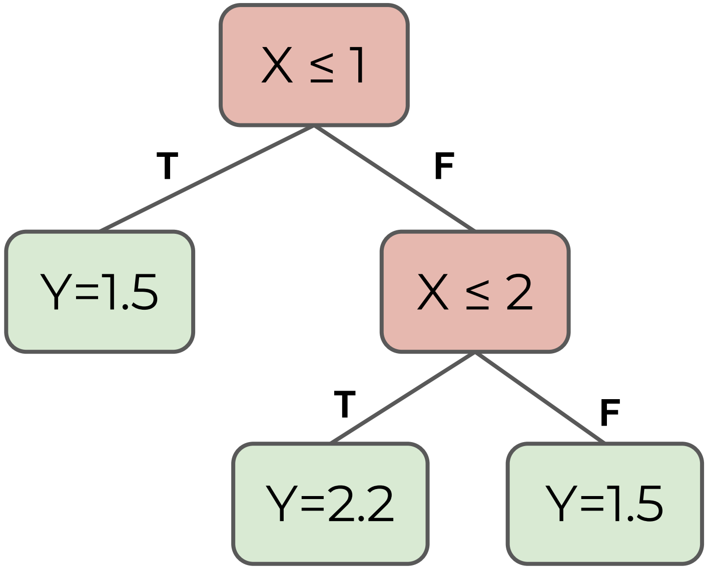
Decision Trees

- To begin understanding a decision tree, we first need to review some terminology about the decision tree components.

Decision Trees

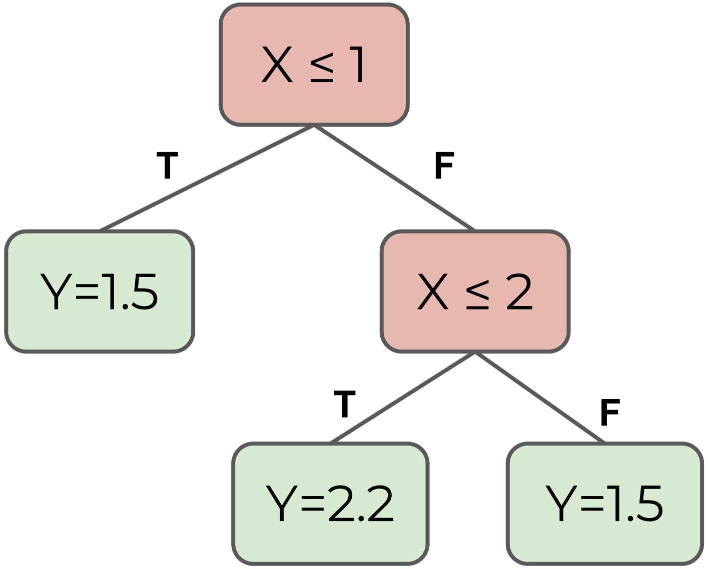
- Recall our simple regression tree:





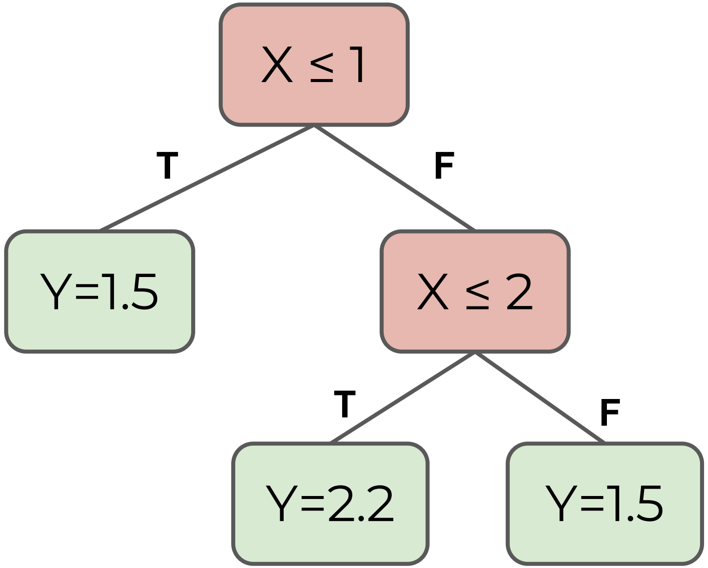
Decision Trees

- Recall our simple regression tree:



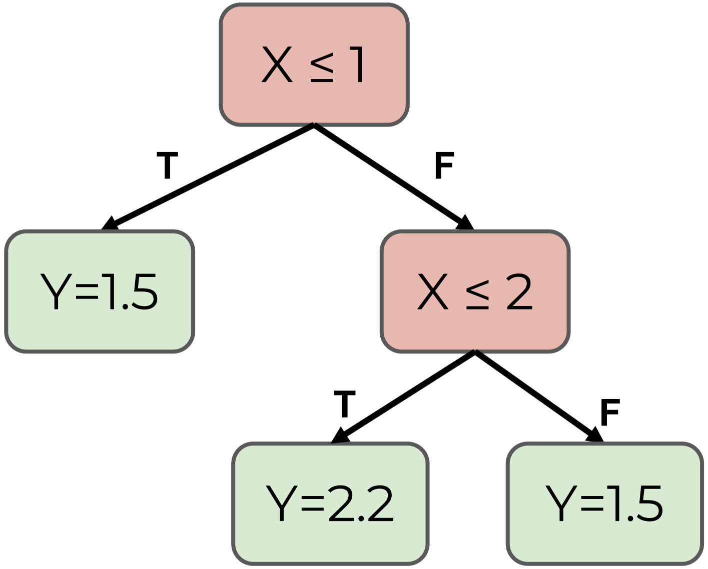
Decision Trees

- Splitting



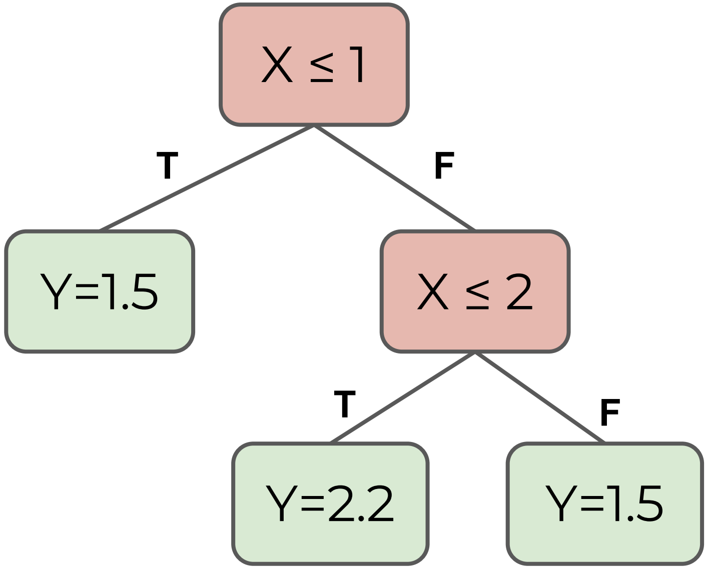
Decision Trees

- Splitting



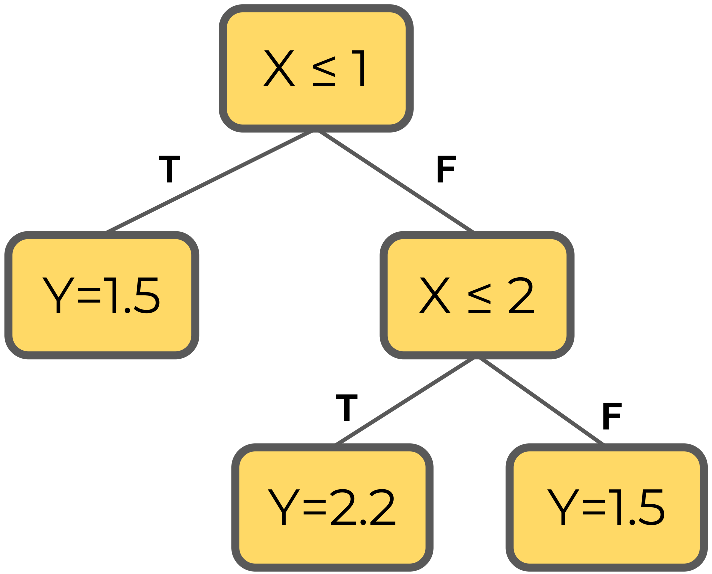
Decision Trees

- Nodes:



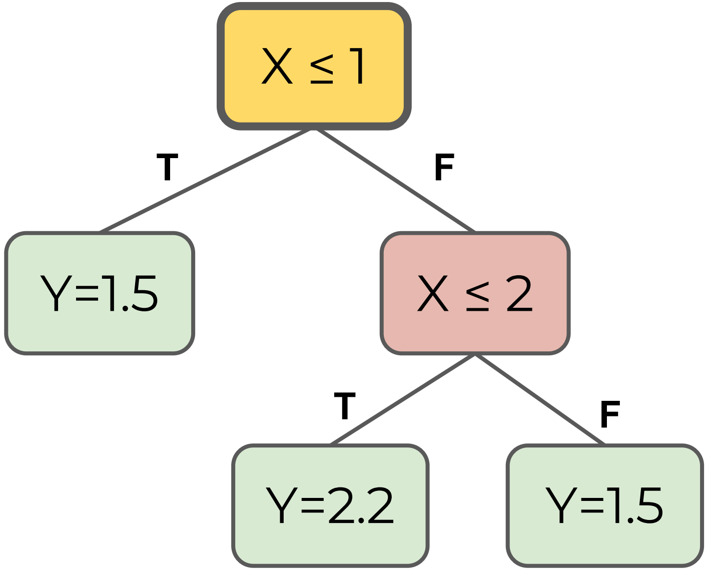
Decision Trees

- Nodes:



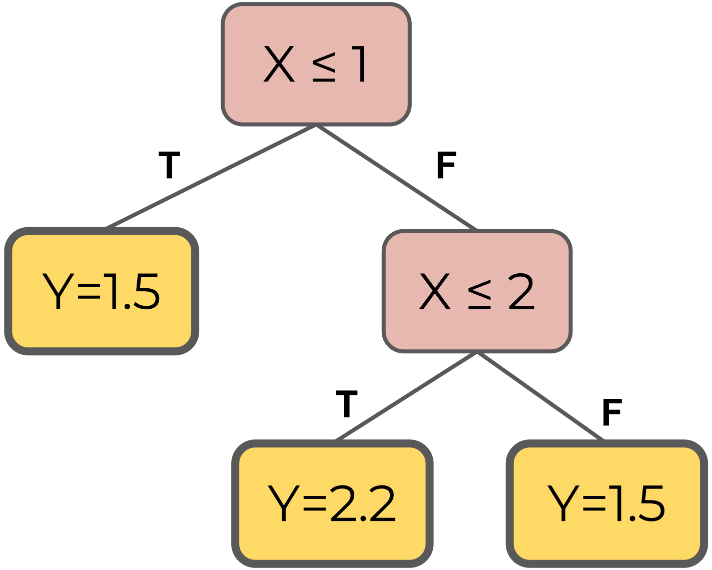
Decision Trees

- Root Node:



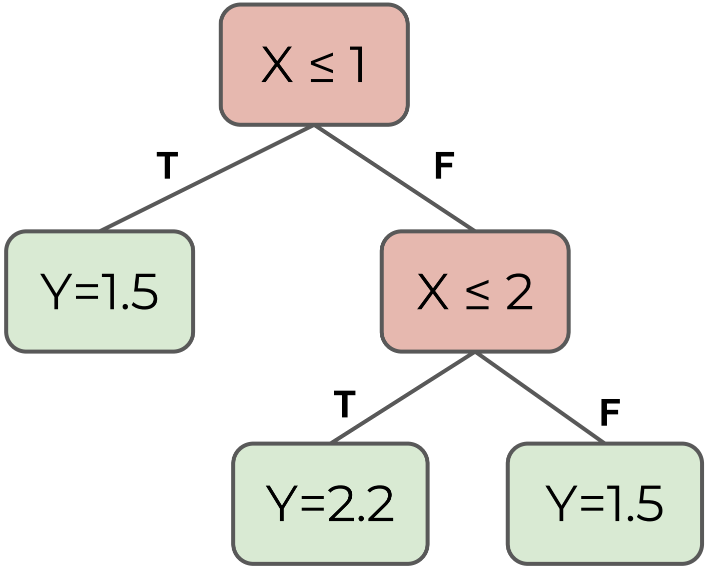
Decision Trees

- Leaf (Terminal) Nodes:



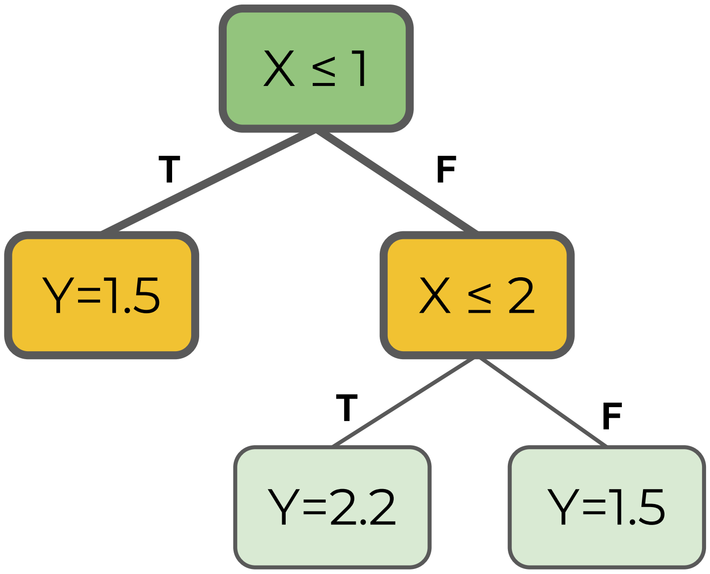
Decision Trees

- Parent and Children Nodes:



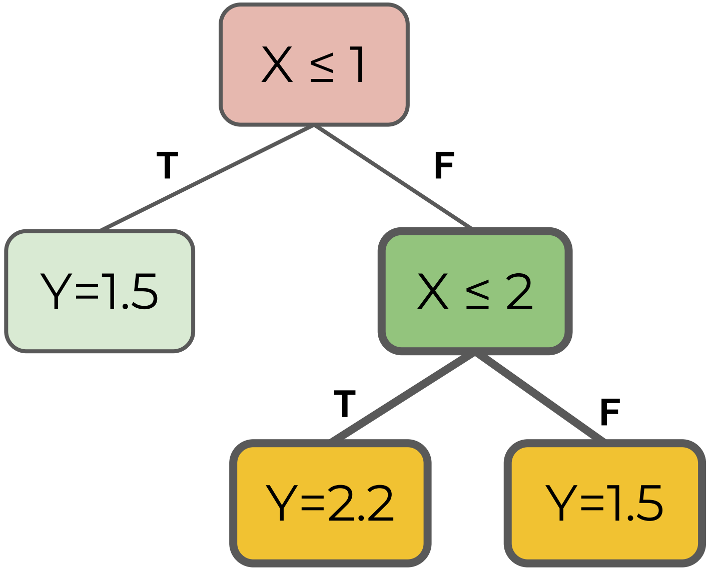
Decision Trees

- Parent and Children Nodes:



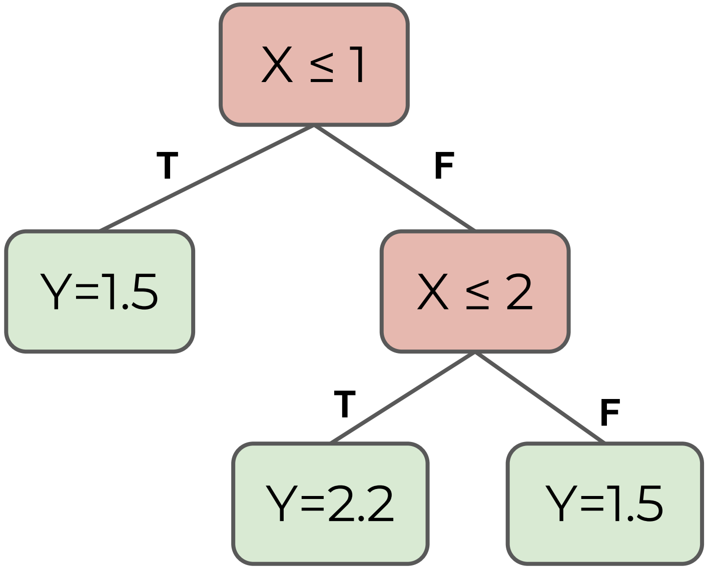
Decision Trees

- Parent and Children Nodes:



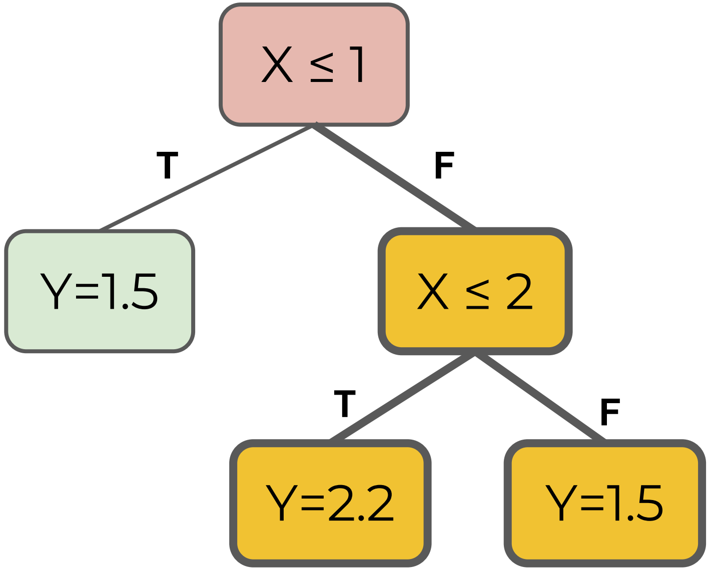
Decision Trees

- Tree Branches (Sub Trees):



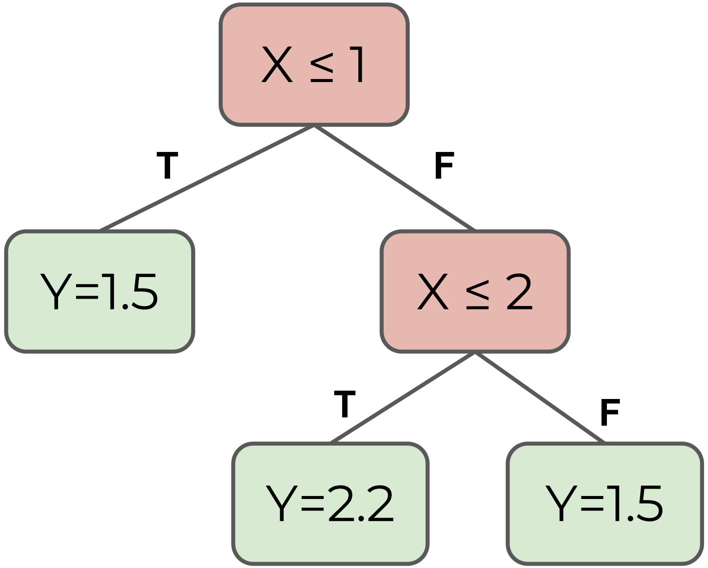
Decision Trees

- Tree Branches (Sub Trees):



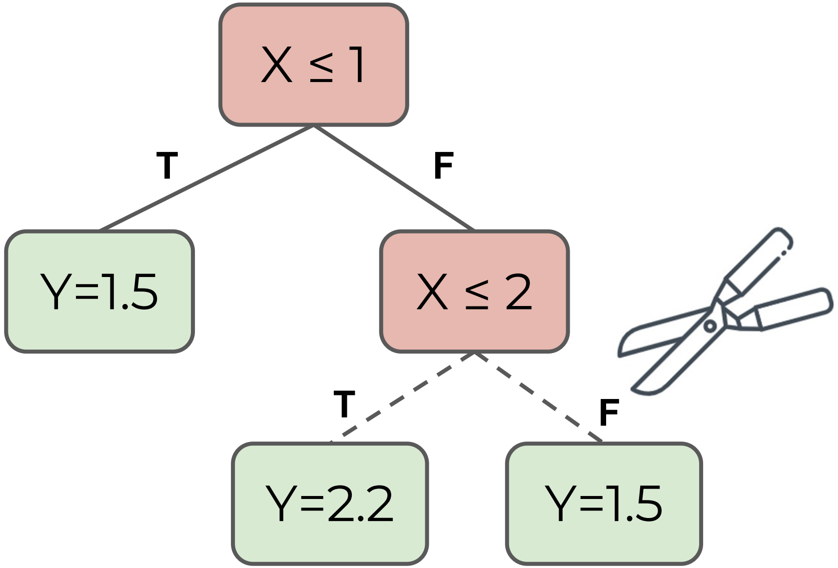
Decision Trees

- Pruning:



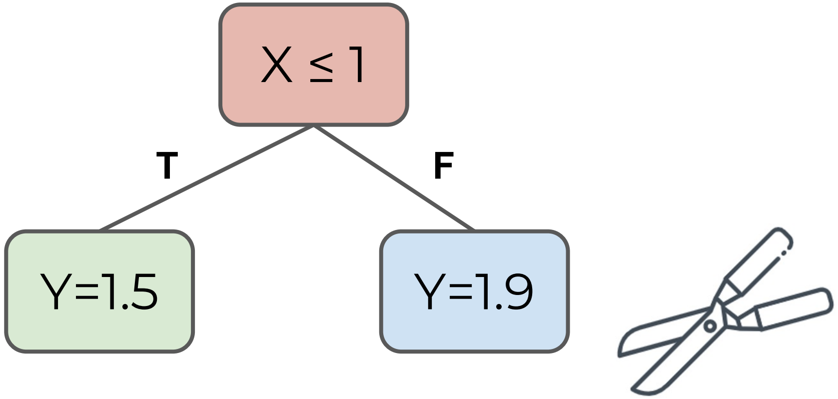
Decision Trees

- Pruning:



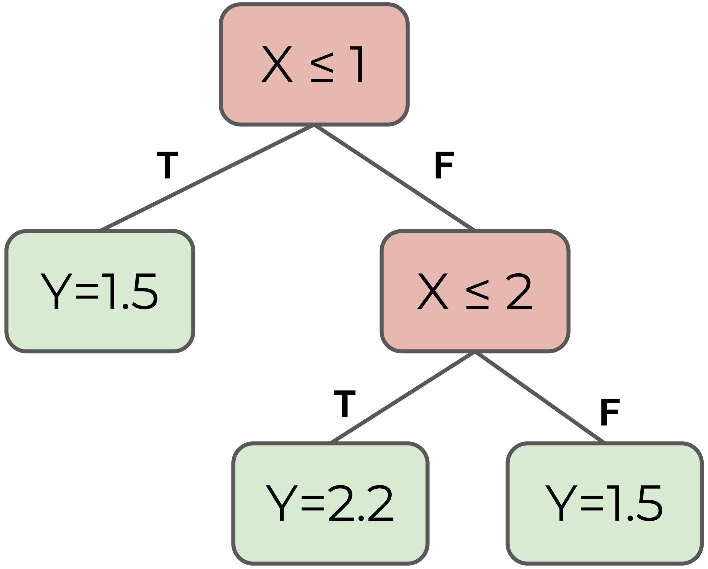
Decision Trees

- Pruning:



Decision Trees

- Let's now move on to constructing a tree!



Decision Trees

Theory and Intuition: Gini Impurity

Gini Impurity

- Before we explore how **splitting criterion** is used in constructing decision trees, let's explore the most common information measurement for decision trees, **gini impurity.**

Gini Impurity

- **Gini impurity** is a mathematical measurement of how “pure” the information in a data set is.
- In regards to classification, we can think of this as a measurement of class uniformity.
- Let’s see how this relates to the simplest case of two classes...

Gini Impurity

- Gini Impurity for Classification:
  - For a set of classes **C** for a given dataset **Q**:

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Gini Impurity

- Gini Impurity for Classification:
  - For a set of classes **C** for a given dataset **Q**, $\mathbf{p_c}$ is probability of class **c**.

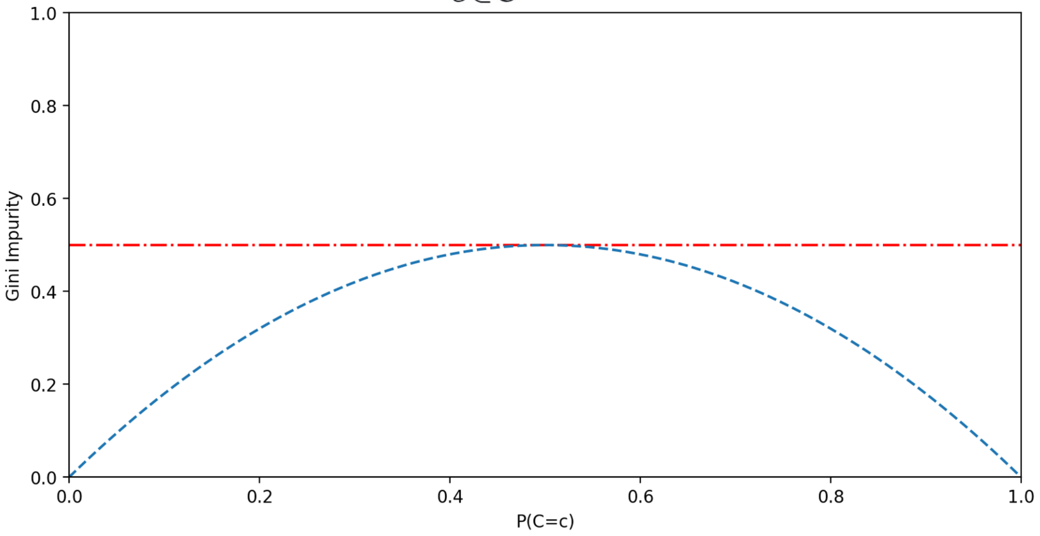
$$p_c = \frac{1}{N_Q} \sum_{x \in Q} \mathbb{1}(y_{class} = c)$$

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Gini Impurity

- Gini Impurity for Classification:

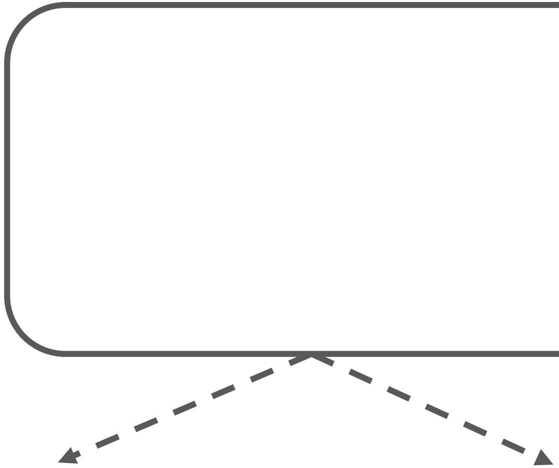
$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



Gini Impurity

- Gini Impurity for Classification:

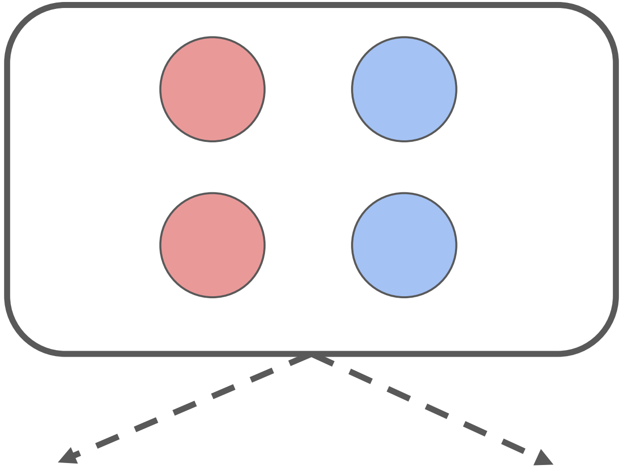
$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



Gini Impurity

- Gini Impurity for Classification:

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

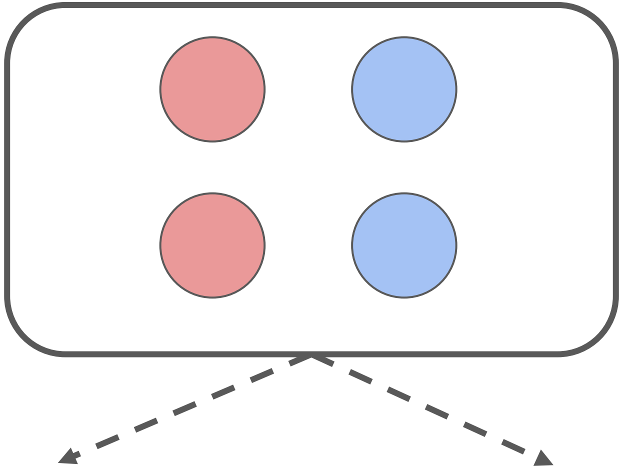


Gini Impurity

- Gini Impurity for Classification:

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

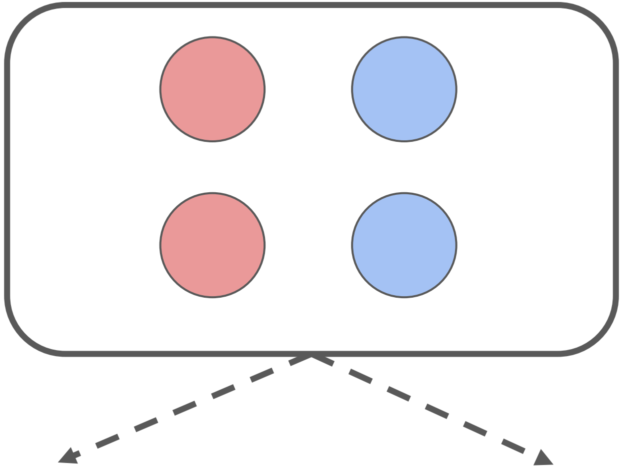
Class Red
(2/4)(1 - 2/4) = 0.25



Gini Impurity

- Gini Impurity for Classification:

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



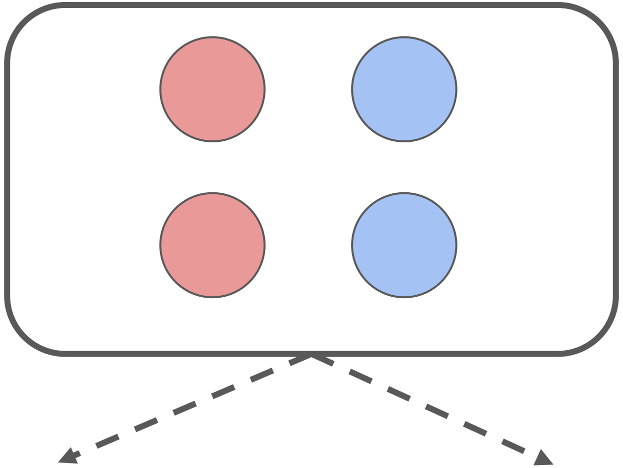
Class Red
(2/4)(1 - 2/4) = 0.25

Class Blue
(2/4)(1 - 2/4) = 0.25

Gini Impurity

- Gini Impurity for Classification:

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



Class Red
(2/4)(1 - 2/4) = 0.25

+

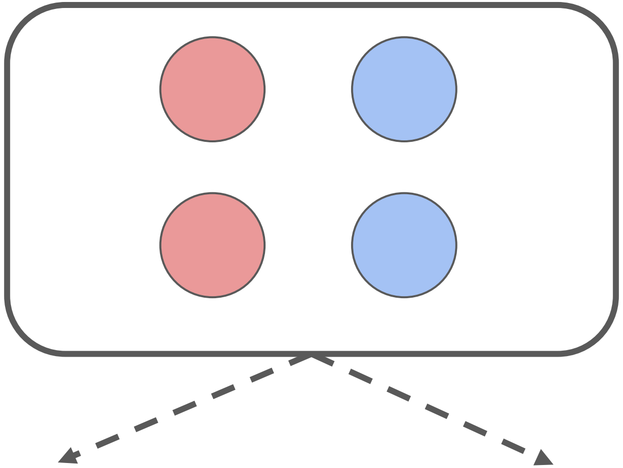
Class Blue
(2/4)(1 - 2/4) = 0.25

Gini Impurity
0.25 + 0.25 = 0.5

Gini Impurity

- “Maximum” Impurity Possible

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



Class Red
(2/4)(1 - 2/4) = 0.25

+

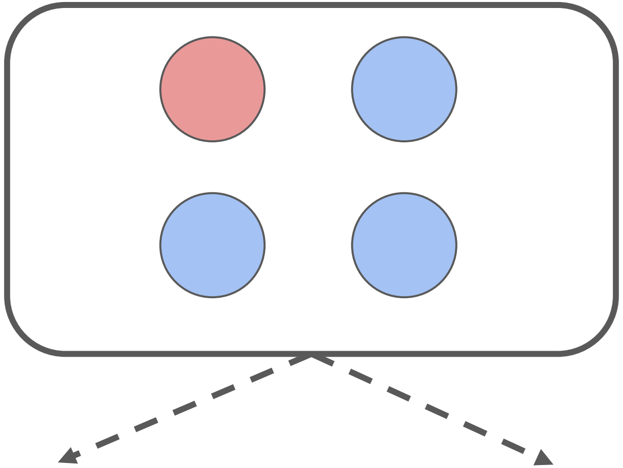
Class Blue
(2/4)(1 - 2/4) = 0.25

Gini Impurity
0.25 + 0.25 = 0.5

Gini Impurity

- Data is more “pure” (less impurity)

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



Class Red
(1/4)(1 - 1/4) = 0.1875

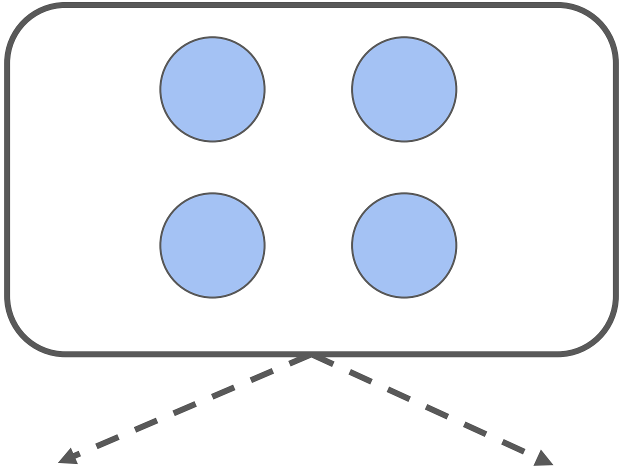
Class Blue
(3/4)(1 - 3/4) =0.1875

Gini Impurity
0.1875+0.1875 = 0.375

Gini Impurity

- Data is completely "pure" (no impurity)

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$



Class Red
(0/4)(1 - 0/4) = 0

Class Blue
(4/4)(1 - 4/4) = 0

Gini Impurity
0 + 0 = 0

Gini Impurity

- If the goal of a decision tree is to separate out classes, we can use **gini impurity** to decide on data split values.
- We want to **minimize** the gini impurity at leaf nodes.
- Minimized impurity at leaf nodes means we are separating classes effectively!

Gini Impurity

- In the next lecture we will construct a basic example of using gini impurity from a data set to calculate feature gini impurity.
- Afterwards, we'll explore splitting various feature types and deciding which feature should be the root node.

Decision Trees

Theory and Intuition: Gini Impurity in Trees

Decision Trees

- Let’s begin to understand how the ordering of nodes is decided and how splits are conducted within a tree.
- We’ll start by exploring how a decision tree is constructed from a training data set using **gini impurity**.

Decision Trees

- When first constructing a tree, we need to decide what feature will be used as the root node.
- We can use **gini impurity** to compare the **information** contained within features for the training data.
- Let’s explore this concept further…

Decision Trees

- Gini Impurity for Classification:
  - For a set of classes **C** for a given dataset **Q**:

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

- Gini Impurity for Classification:
  - For a set of classes **C** for a given dataset **Q**, $\mathbf{p_c}$ is probability of class **c**.

$$p_c = \frac{1}{N_Q} \sum_{x \in Q} \mathbb{1}(y_{class} = c)$$

$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

- Gini Impurity for Classification:
  - For a set of classes **C** for a given dataset **Q**, $\mathbf{p_c}$ is probability of class **c**.

$$p_c = \boxed{\frac{1}{N_Q} \sum_{x \in Q} \mathbb{1}(y_{class} = c)}$$

$$G(Q) = \sum_{c \in C} \boxed{p_c}(1 - \boxed{p_c})$$

Decision Trees

- Let's take a look at this data set:

X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No

Decision Trees

- Create a decision tree to predict spam.

X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No

Decision Trees

- Only one X feature to use for a node.

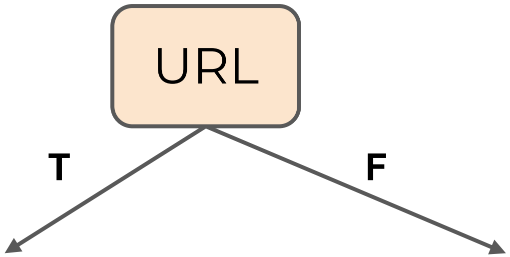
X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No

URL

Decision Trees

- Predict if email is spam if it contains a URL:

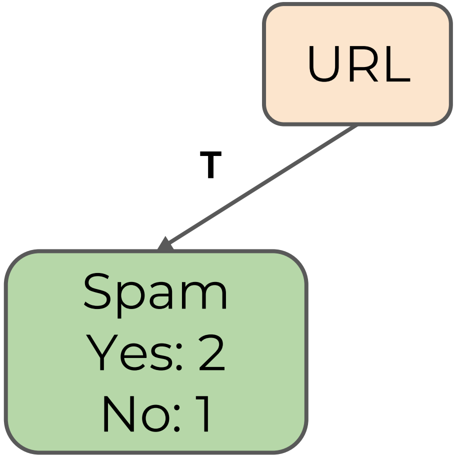
X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No



Decision Trees

- Predict if email is spam if it contains a URL:

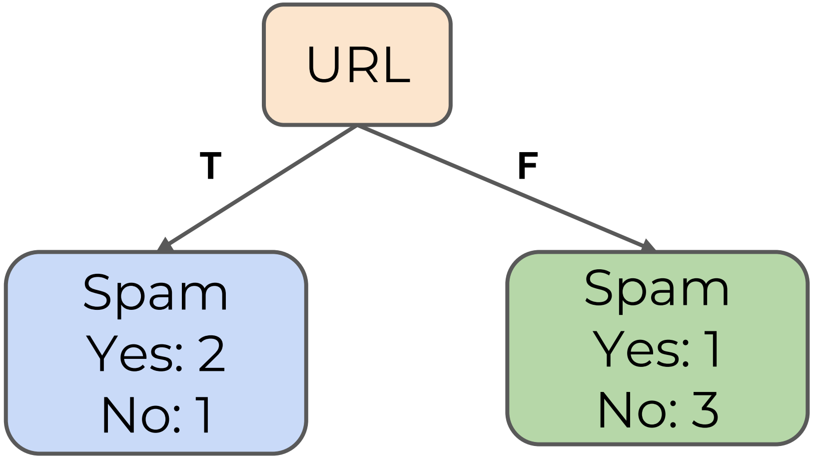
X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No



Decision Trees

- Predict if email is spam if it contains a URL:

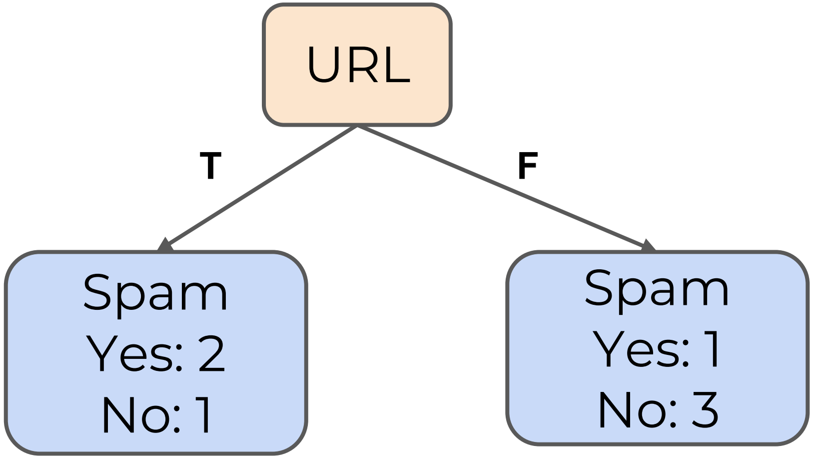
X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No



Decision Trees

- Predict if email is spam if it contains a URL:

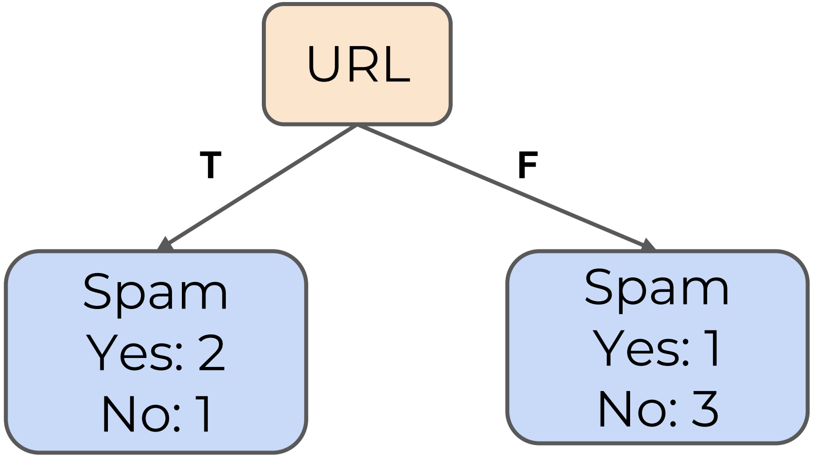
X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No



Decision Trees

- Recall the gini impurity formula:

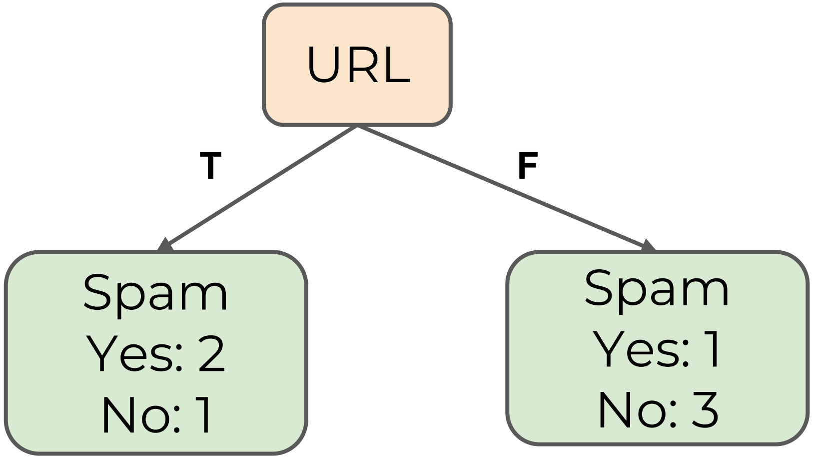
X - URL Link	Y-Spam
Yes	Yes
Yes	Yes
No	No
No	No
No	Yes
No	No
Yes	No



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

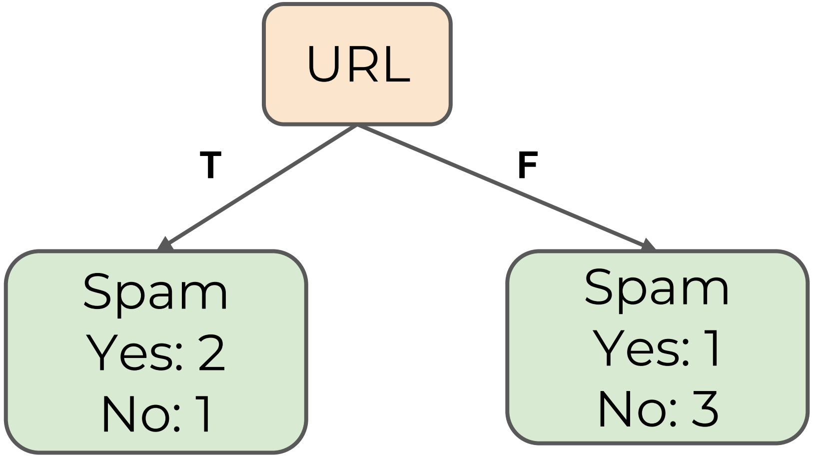
- Treat **Yes Spam** and **No Spam** as **C** classes:



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

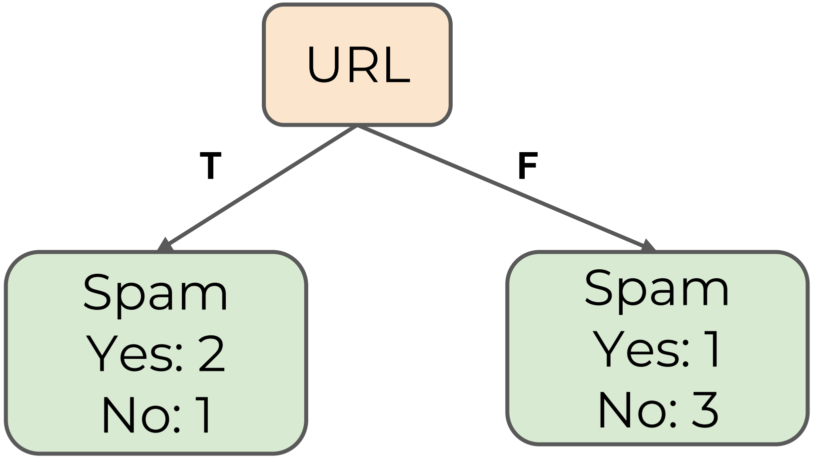
- Treat Yes Spam and No Spam as **c** classes:
- Left Leaf Node:
  - (⅔)(1-⅔)



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

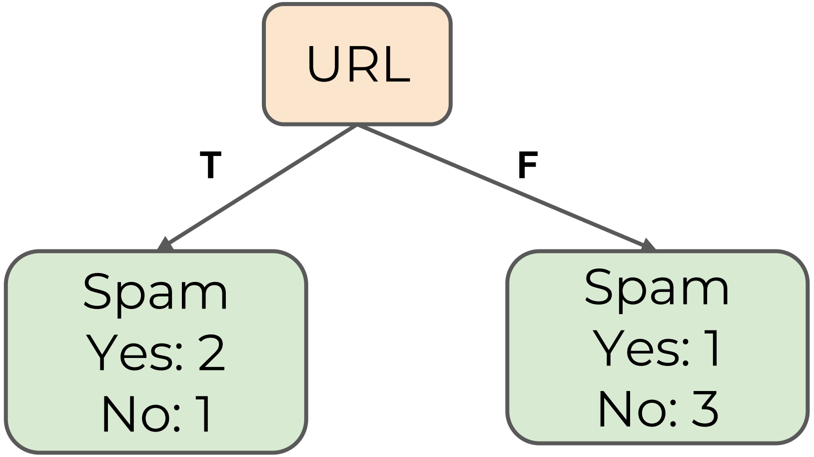
- Treat Yes Spam and No Spam as **c** classes:
- Left Leaf Node:
  - $(\frac{2}{3})(1-\frac{2}{3}) + (\frac{1}{3})(1-\frac{1}{3})$



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

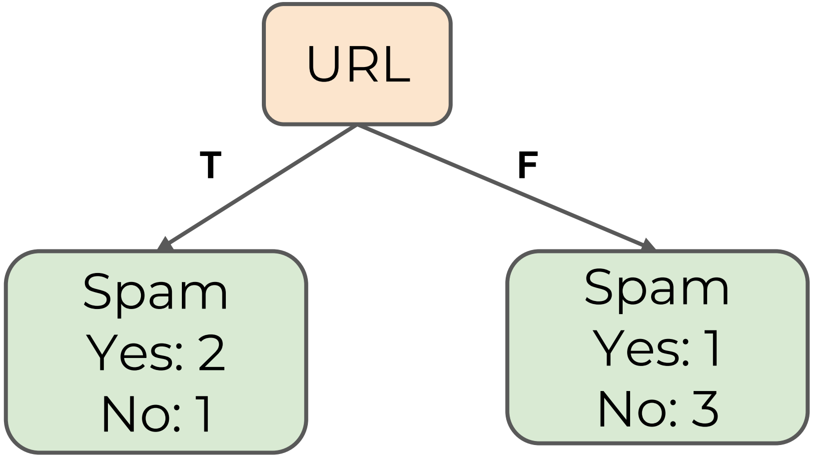
- Treat Yes Spam and No Spam as **c** classes:
- Left Leaf Node:
  - $(\frac{2}{3})(1-\frac{2}{3}) + (\frac{1}{3})(1-\frac{1}{3})$
  - Left Leaf Gini=0.44



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

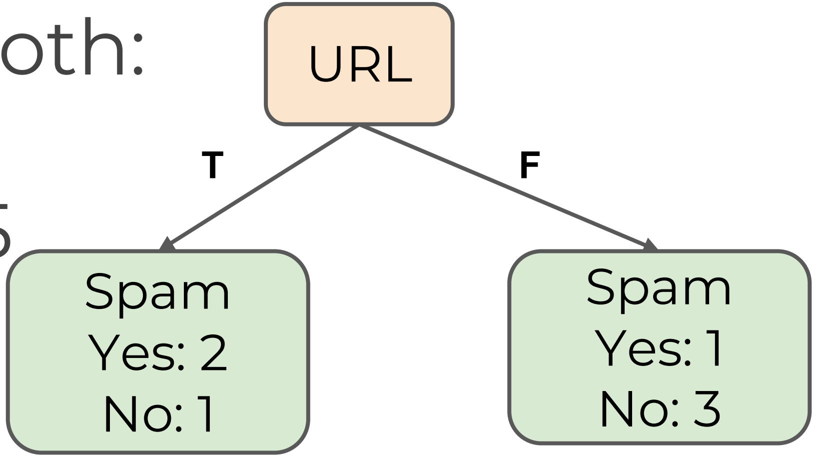
- Treat Yes Spam and No Spam as **c** classes:
- Left Leaf Node:
  - $(\frac{2}{3})(1-\frac{2}{3}) + (\frac{1}{3})(1-\frac{1}{3})$
  - Left Leaf Gini=0.44
- Right Leaf Node:
  - $(\frac{1}{4})(1-\frac{1}{4}) + (\frac{3}{4})(1-\frac{3}{4})$
  - Right Leaf Gini=0.375



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

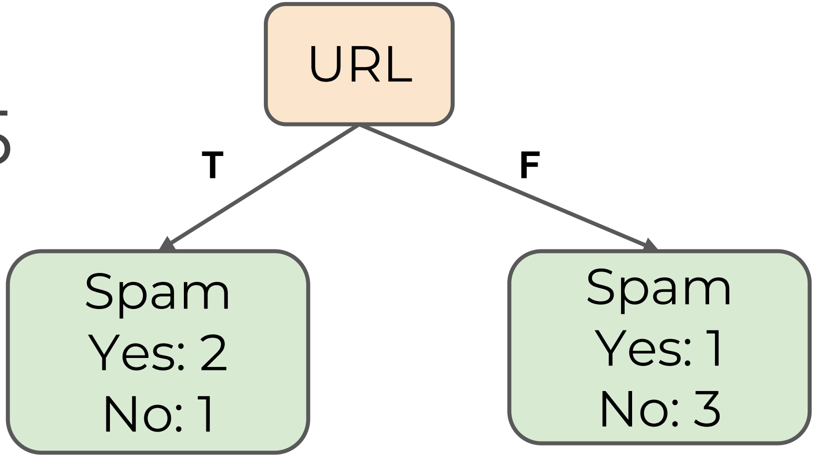
- Now calculate gini impurity of URL feature.
- Weighted Average of both:
  - Left Leaf Gini=0.44
  - Right Leaf Gini=0.375



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

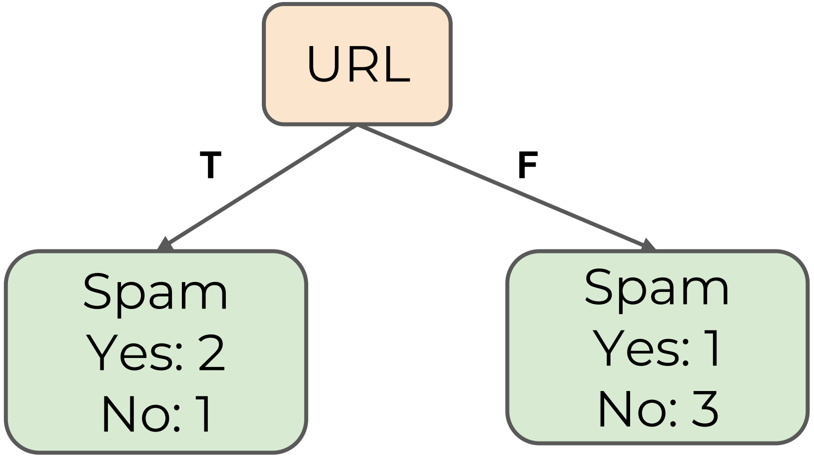
- Total Emails: (2+1) + (1+3) = 7
  - Left Leaf Gini=0.44
  - Right Leaf Gini=0.375



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

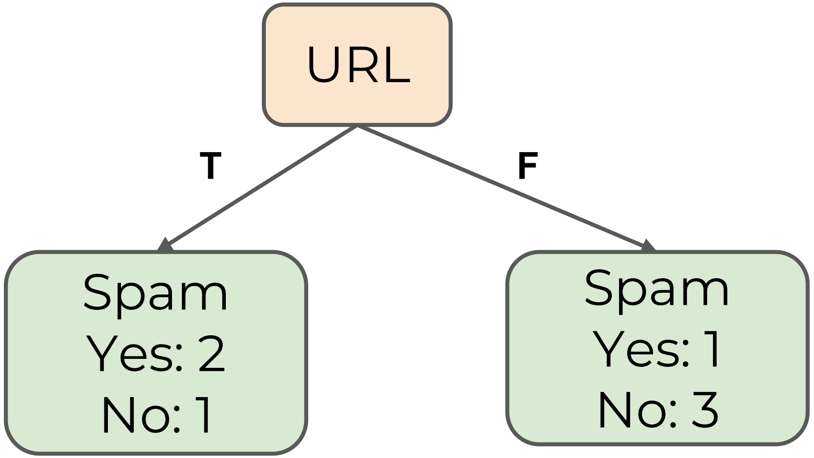
- Total Emails: (2+1) + (1+3) = 7
- Left Leaf Gini=0.44
- Right Leaf Gini=0.375
- Left Emails: 3
- Right Emails: 4



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

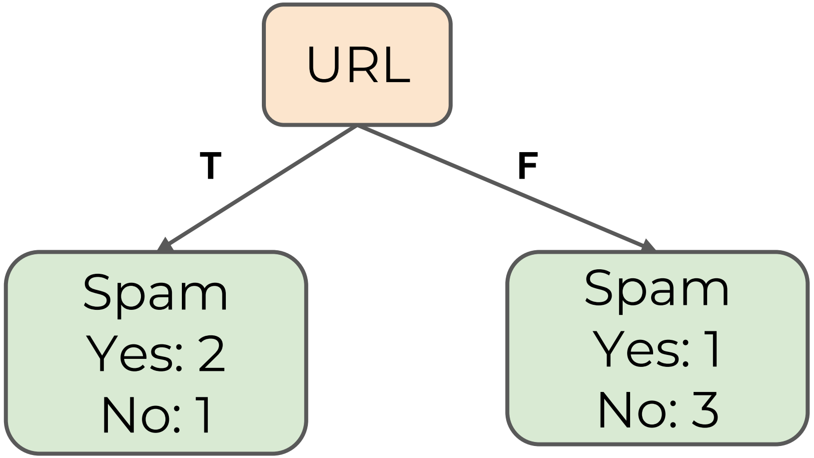
- Total Emails: (2+1) + (1+3) = 7
- Left Leaf Gini=0.44
- Right Leaf Gini=0.375
- Left Emails: 3
- Right Emails: 4
- (3/7)*0.44 + (4/7)*0.375



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

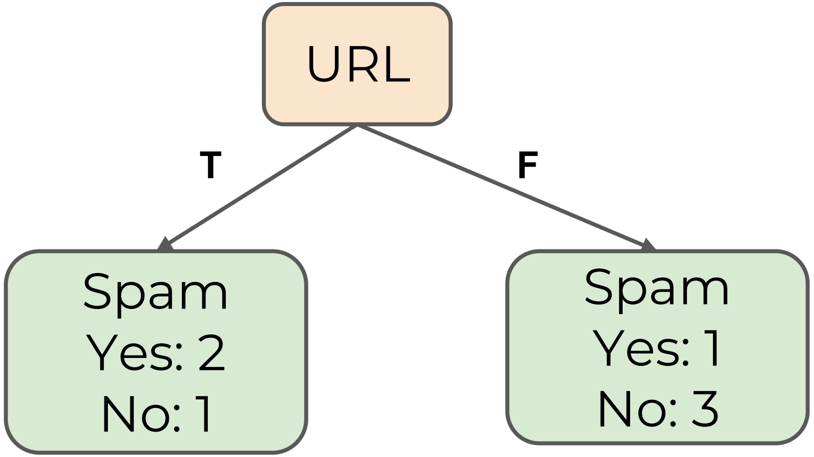
- Total Emails: (2+1) + (1+3) = 7
- Left Leaf Gini=0.44
- Right Leaf Gini=0.375
- Left Emails: 3
- Right Emails: 4
- (3/7)*0.44 + (4/7)*0.375
- Gini Impurity: 0.403



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

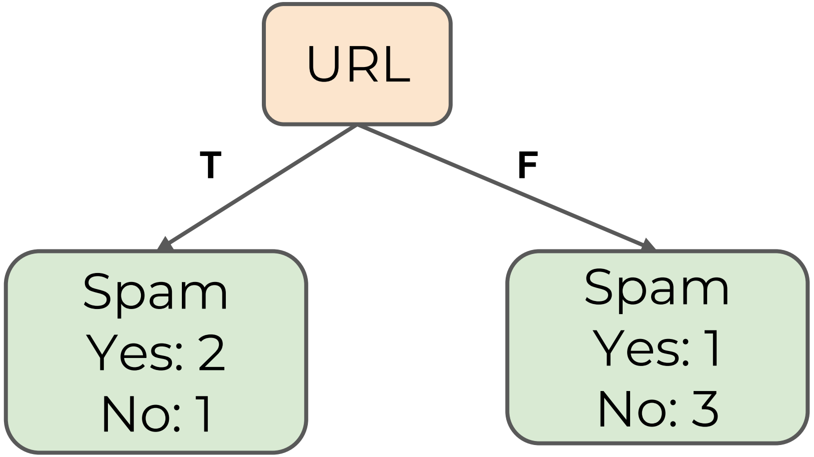
- Gini Impurity for URL feature: 0.403



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

- But what if we had multiple features?



$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

- We still have more issues to consider:
  - Multiple Features
  - Continuous Features
  - Multi-categorical Features
- We can incorporate the gini impurity to each of these issues to solve for best root nodes and best split parameters for leaves.

Decision Trees

Theory and Intuition: Gini Impurity Part Two

Decision Trees

- We explored how to calculate gini impurity for a binary categorical feature (only consisting of two categories).
- Now let's explore the following:
  - Continuous numeric features
  - Multi-categorical features (N>2)
  - Choosing a root node feature

Decision Trees

- Imagine a continuous feature:

X - Words in Email	Y-Spam
10	Yes
40	No
20	Yes
50	No
30	No

Decision Trees

- Let's calculate the feature gini impurity:

X - Words in Email	Y-Spam
10	Yes
40	No
20	Yes
50	No
30	No

Decision Trees

- First sort data:

X - Words in Email	Y-Spam
10	Yes
40	No
20	Yes
50	No
30	No

Decision Trees

- First sort data:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

Decision Trees

- Calculate potential split values for node:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

Decision Trees

- Calculate potential split values for node:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

Words ≤ N

Decision Trees

- Use averages between rows as values:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

15

25

35

45

Words ≤ N

Decision Trees

- Perform each potential split:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

15

25

35

45

Words ≤ 15

Decision Trees

- Calculate gini impurity for each split:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

15

Words ≤ 15

Decision Trees

- Calculate gini impurity for each split:

Words ≤ 15

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

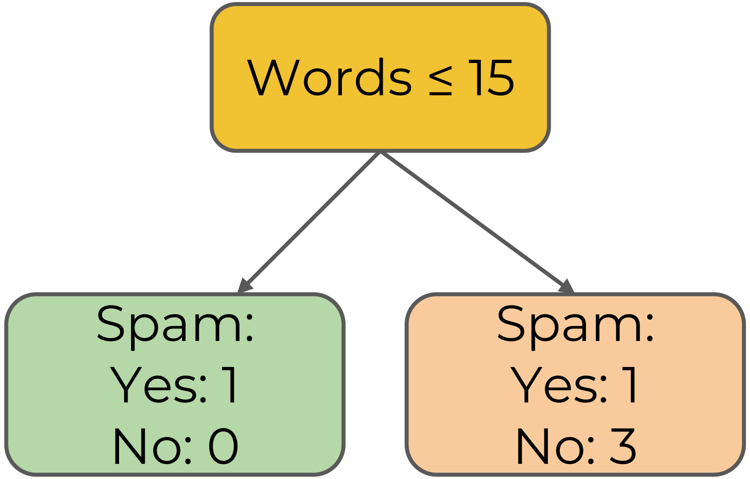
15

Decision Trees

- Calculate gini impurity for each split:

15

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No



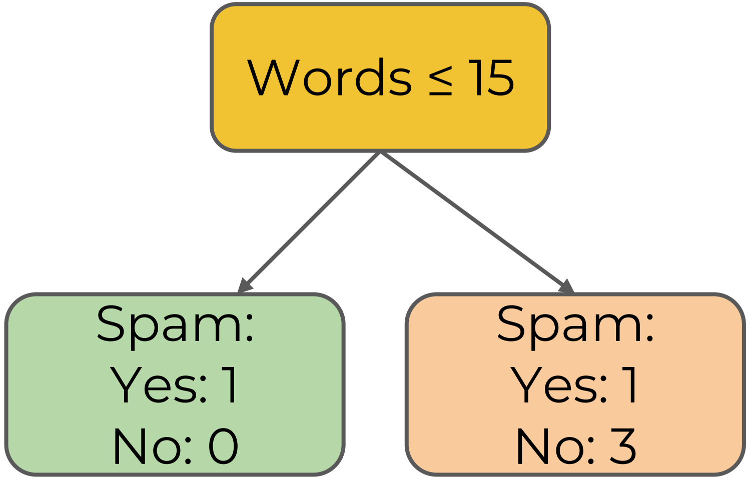
$$G(Q) = \sum_{c \in C} p_c(1 - p_c)$$

Decision Trees

- Calculate gini impurity for each split:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

15

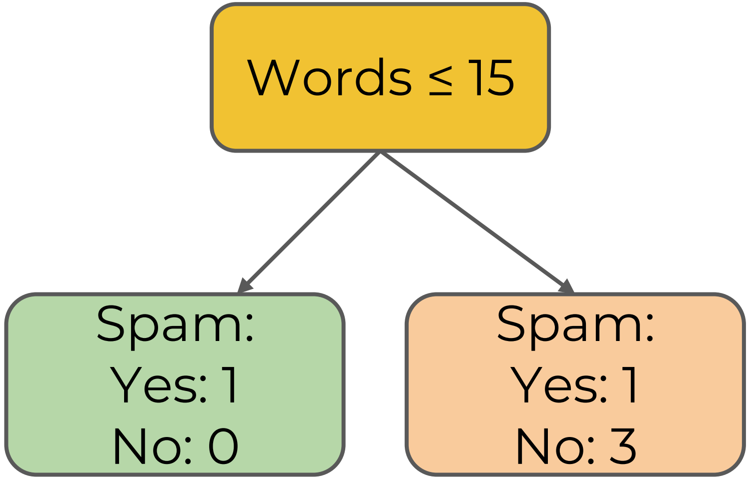


$G(Q) =$ $(\frac{1}{5})(0+0) + (\frac{4}{5})((\frac{1}{4})(1-\frac{1}{4})+(\frac{3}{4})(1-\frac{3}{4})$

Decision Trees

- Calculate gini impurity for each split:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No



$$G(Q) = (\tfrac{1}{5})(0+0) + (\tfrac{4}{5})((\tfrac{1}{4})(1-\tfrac{1}{4})+(\tfrac{3}{4})(1-\tfrac{3}{4})) = 0.3$$

Decision Trees

- Calculate gini impurity for each split:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

15

Gini=0.3

Decision Trees

- Repeat for all possible splits:

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

15 → Gini=0.3

25 → Gini=0

35 → Gini=0.26

45 → Gini=0.4

Decision Trees

- Choose lowest impurity split value

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

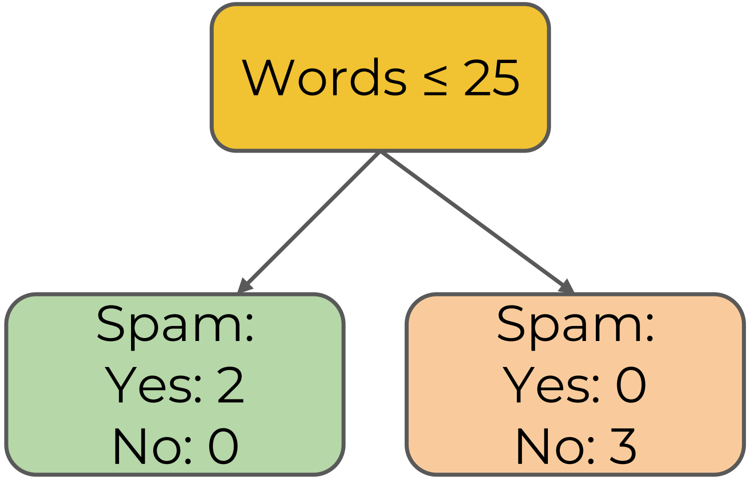
25 → Gini=0

Decision Trees

- Choose this as split value for node.

X - Words in Email	Y-Spam
10	Yes
20	Yes
30	No
40	No
50	No

25



$$G(Q) = 0$$

Decision Trees

- We have now calculated gini impurity for features that are:
  - Binary categories
  - Continuous numeric
- Finally, let’s explore calculating gini impurity for a feature that is multicategorical.

Decision Trees

- Multicategorical feature:

X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No

Decision Trees

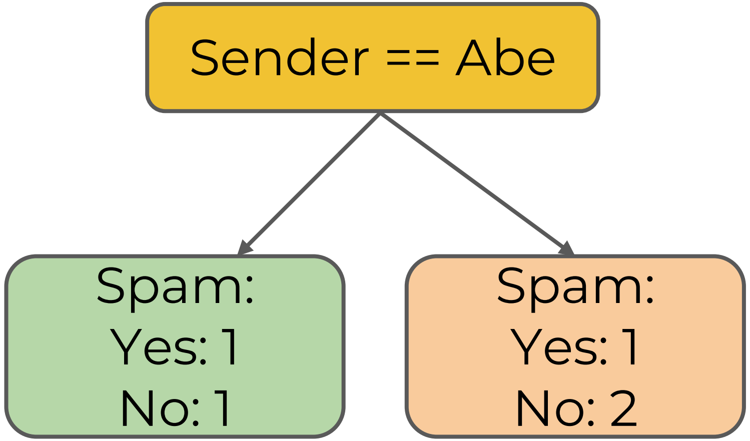
- Calculate gini impurity for all combinations:

X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No

Decision Trees

- Calculate gini impurity for all combinations:

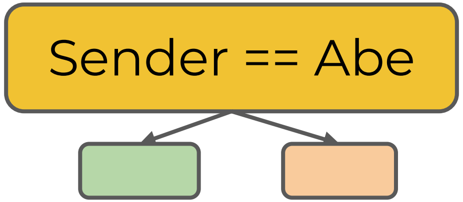
X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No



Decision Trees

- Calculate gini impurity for all combinations:

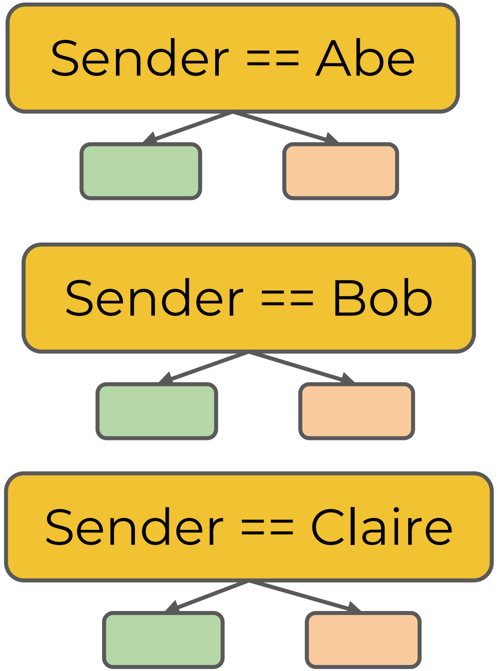
X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No



Decision Trees

- Calculate gini impurity for all combinations:

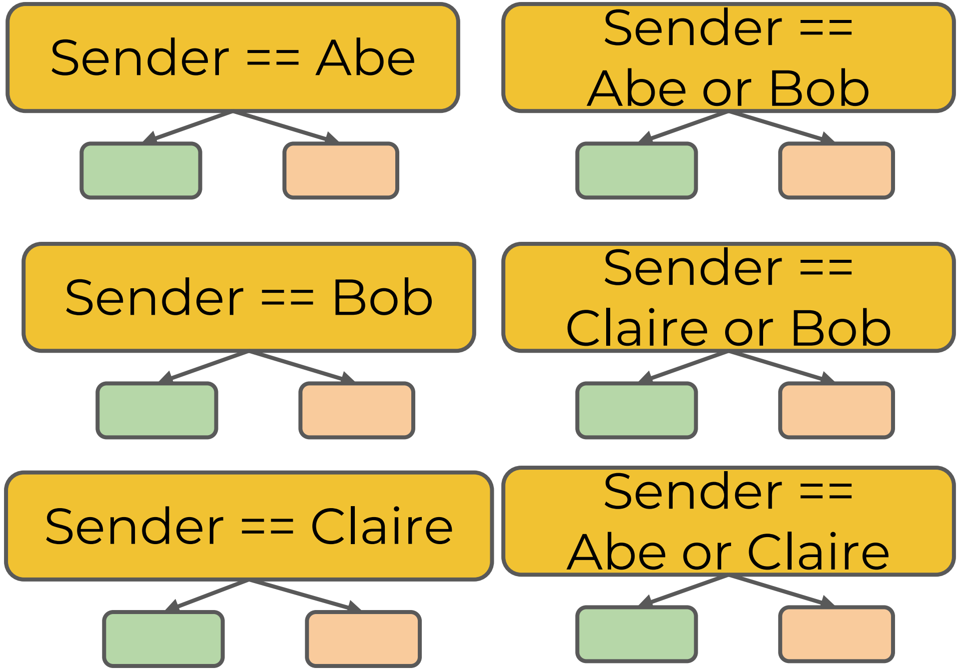
X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No



Decision Trees

- Calculate gini impurity for all combinations:

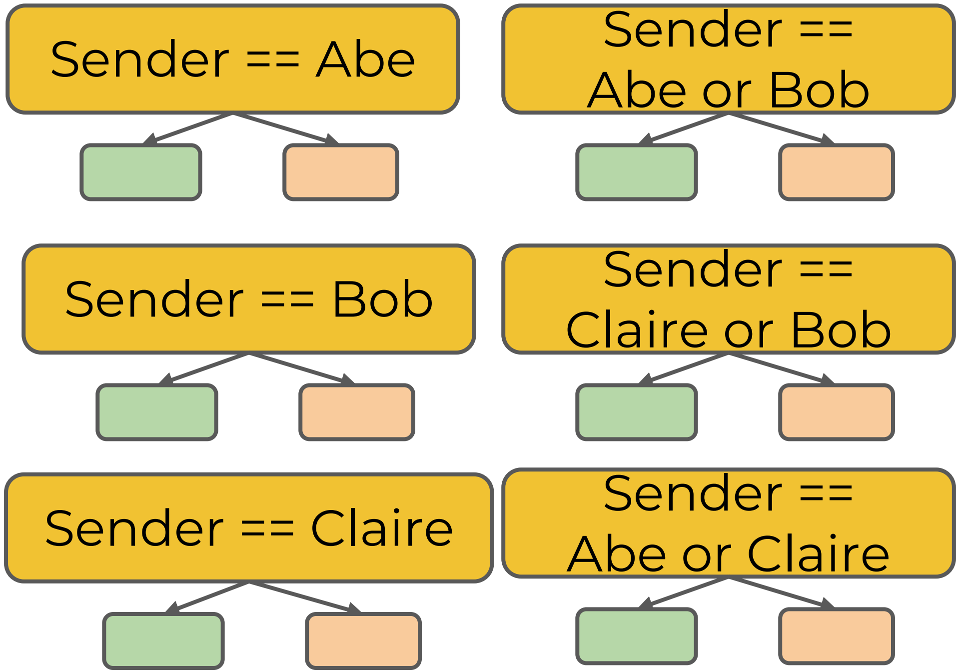
X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No



Decision Trees

- Choose lowest impurity split combination.

X - Sender	Y-Spam
Abe	Yes
Bob	Yes
Claire	No
Abe	No
Bob	No



Decision Trees

- Now we can split any type of feature.
- How does the decision tree decide on the root node of a multi-feature dataset?
- Calculate the gini impurity values of each feature and choose the lowest impurity value to split on first.

Decision Trees

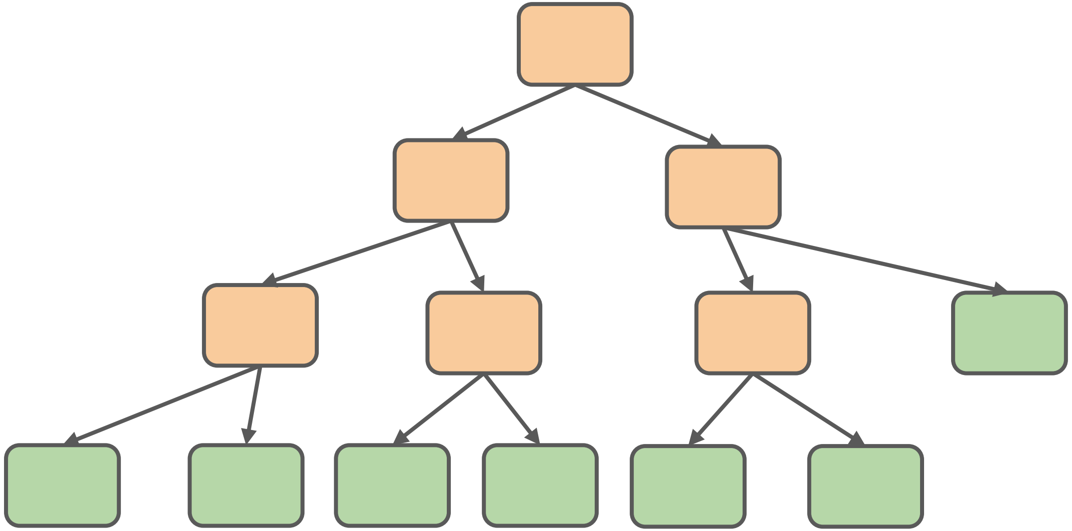
- By choosing the feature with the lowest resulting gini impurity in its leaf nodes, we are choosing the feature that best splits the data into “pure” classes.

Decision Trees

- We should also note, by using **gini impurity** as a measurement of the effectiveness of a node split, we can perform automatic feature selection by mandating an impurity threshold for an additional feature based split to occur.

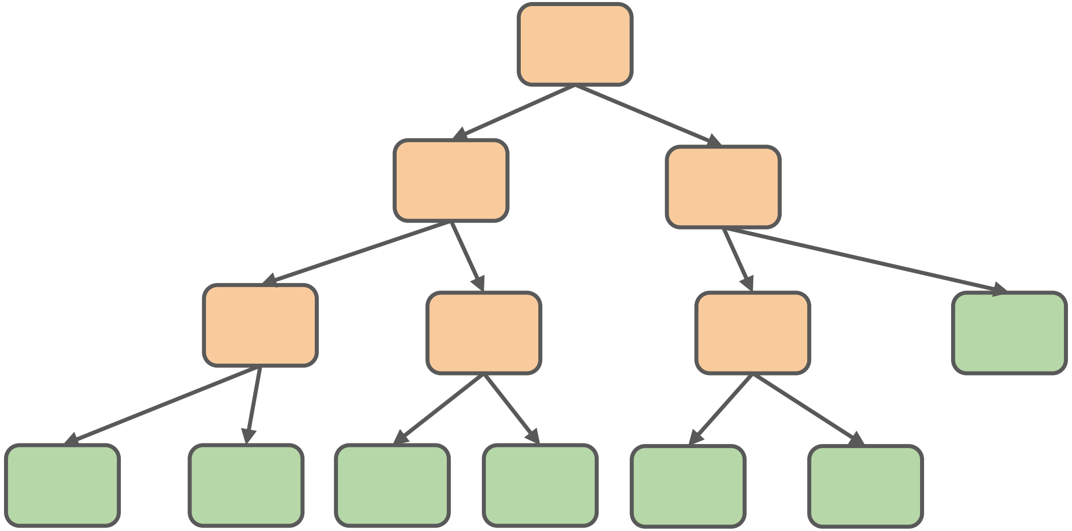
Decision Trees

- A large overfitted tree:



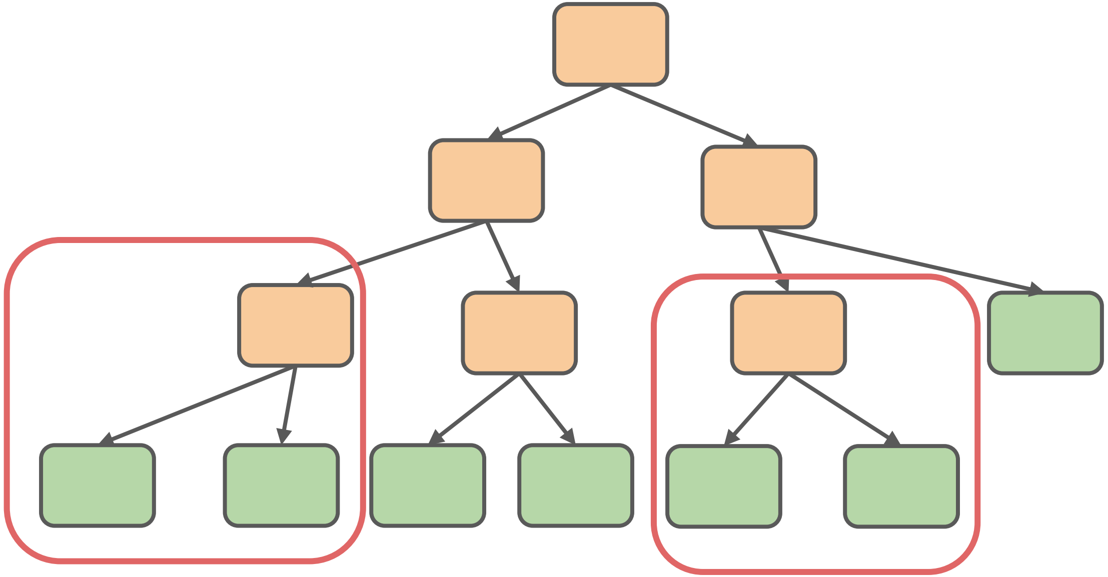
Decision Trees

- Add minimum gini impurity decrease



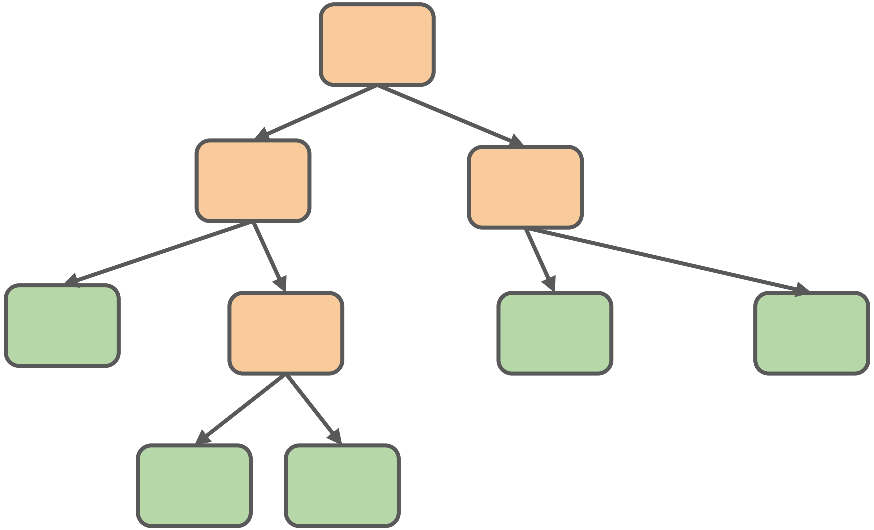
Decision Trees

- Add minimum gini impurity decrease



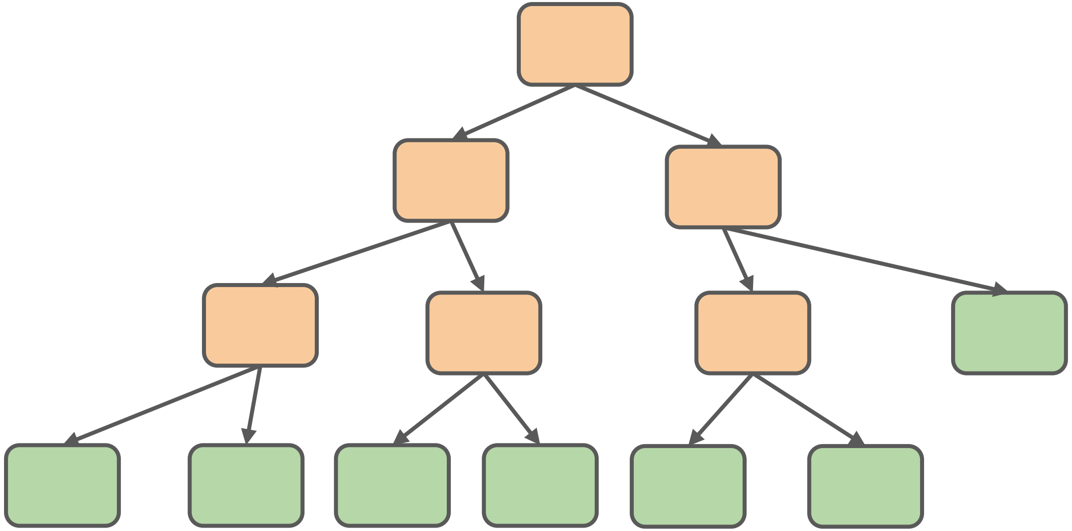
Decision Trees

- Add minimum gini impurity decrease



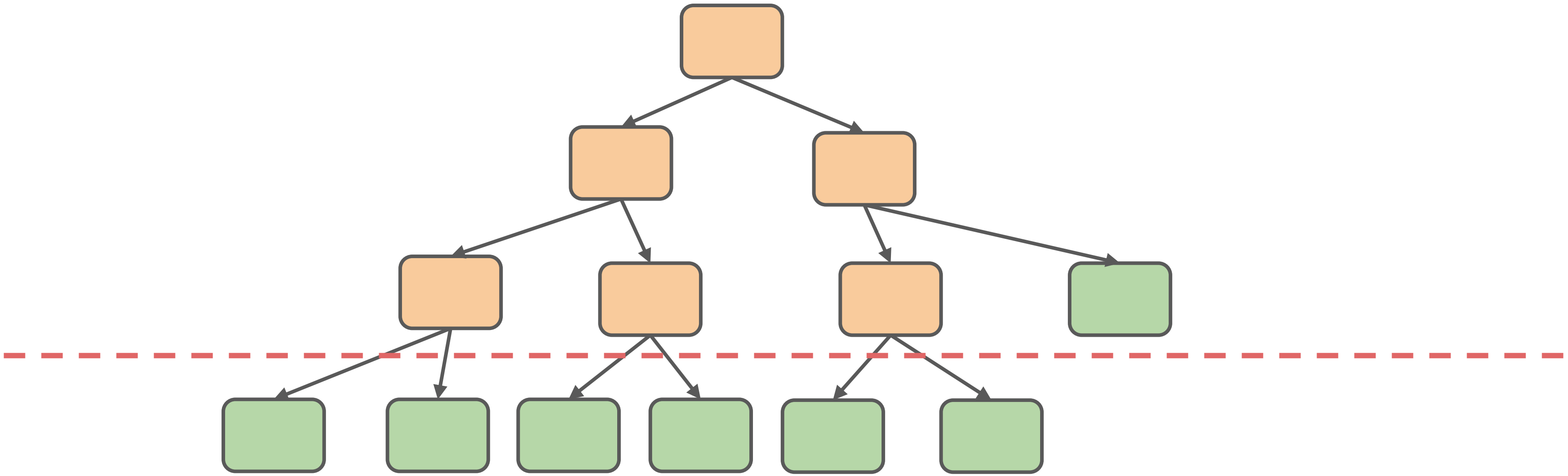
Decision Trees

- We can also mandate a max depth:



Decision Trees

- We can also mandate a max depth:



Decision Trees

- We can also mandate a max depth: