INNOPOLIS UNIVERSITY

Machine Learning

Prof. Adil Khan

Today's Objectives

1. A quick recap of what we have learned so far
2. Things to Know About ML: *model prediction vs. model inference*
3. Decision Trees:
   - What are decision trees? What's the motivation behind them? How are they different from other ML models? How do we learn them from data? How do we use them for both classification and regression problems?
4. Generalizing decision tree learning algorithm
   - What is information gain? How can we measure it?
   - Classification loss vs. Entropy to measure information gain

Recap

1. Machine Learning (Task, Expereince, Performance)
2. Predictors and Respone (Functional Relationship Estimation)
3. Regression and Classification
4. Linear Models
5. Non-linear Models
6. Representation Matters
7. Learning Representations

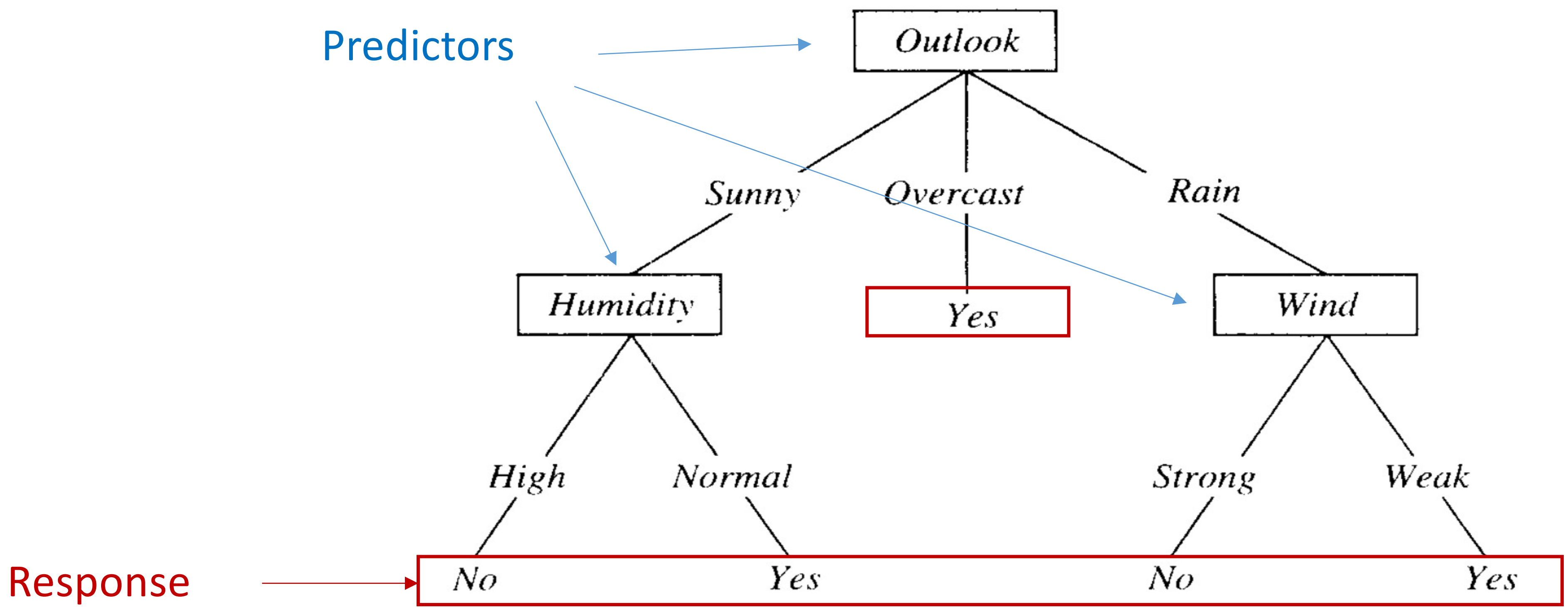
Decision Trees

Accurate Predictions and Good Inference

- Imagine a company asks you to build a ML model for one of their business problems
- Your model is required to make decisions (predictions)
  - It must be accurate (predictions)
  - It must also be understandable (inference: why those decisions)

Thus, you must pick a model that make a good compromise between accurate predictions and great inference

And That's What DT's are Good At!



A Decision Tree for a Binary Response "Walk"

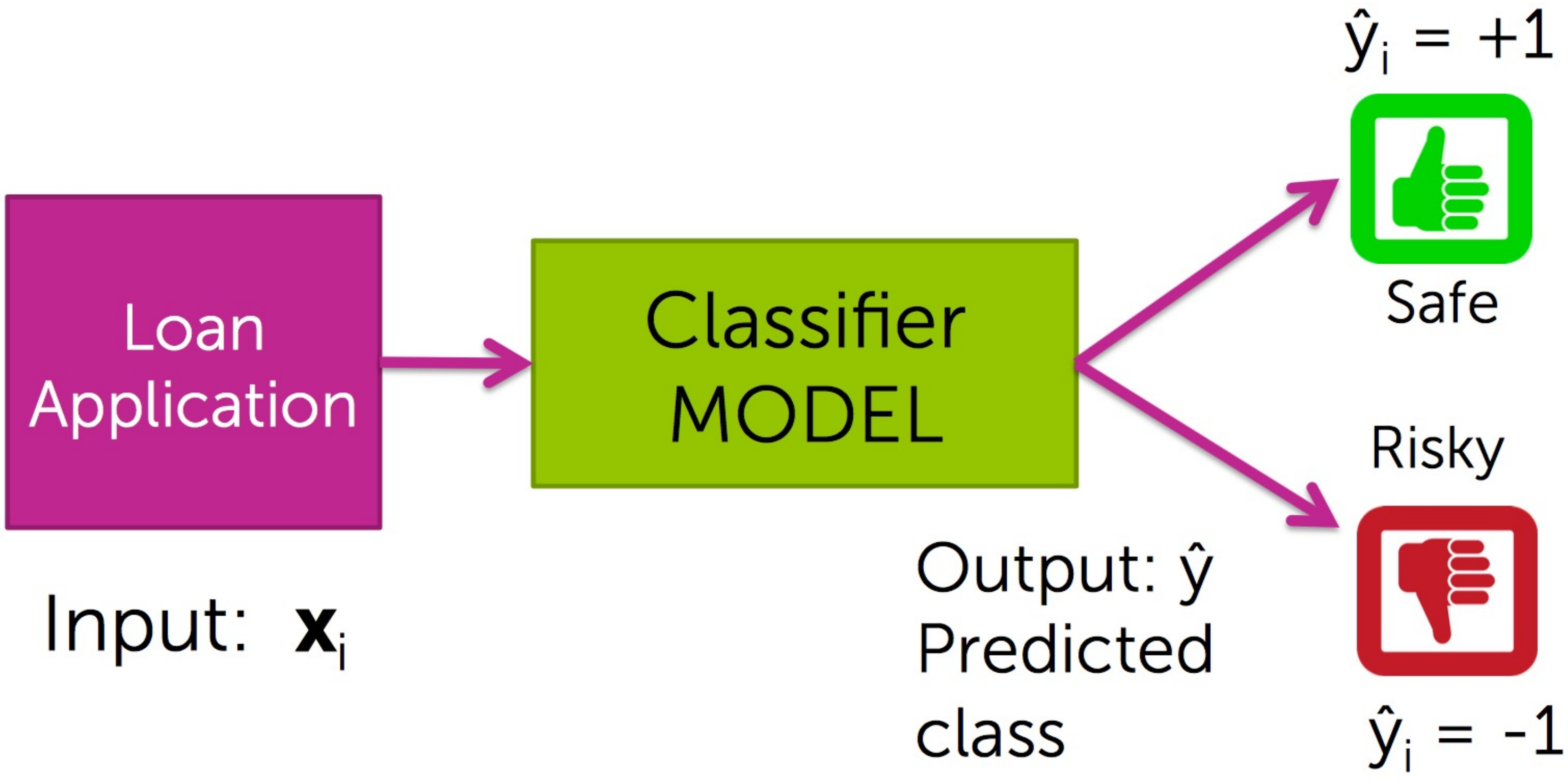
Example: Loan Application Decision System

- Loan applications

- Intelligent loan application review system

- Outcomes: Safe, Risky

The Prediction Problem



Our aim is to solve this problem using a Decision Tree

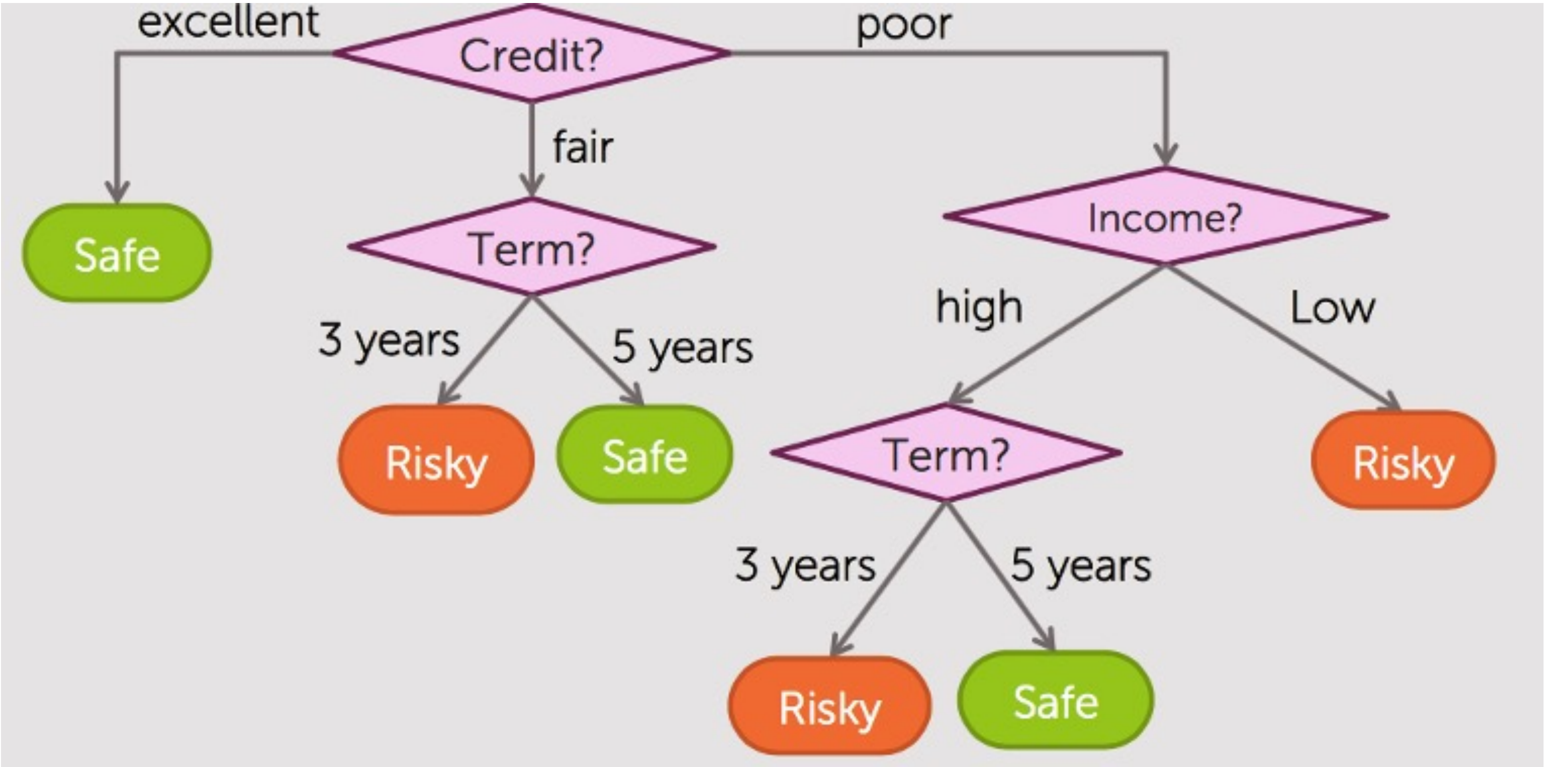
Factors that Such a System Might Consider

1. Credit history of the applicant

2. Term of the loan application

3. Income of the applicant

T(X) could look like this

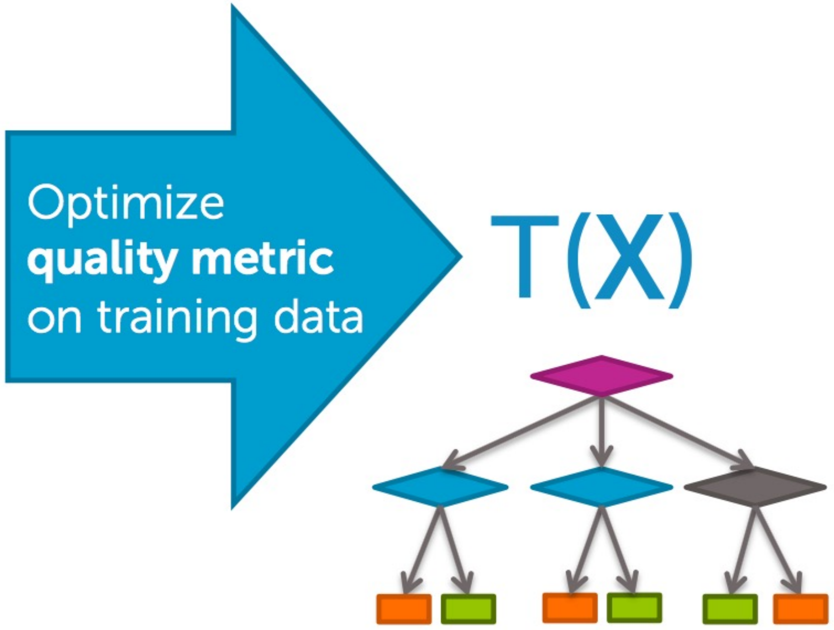


The question is *how will we learn it*?

Goal of DT Learning

Training data: N observations ($\mathbf{x}_i$,y_i)

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe



The Quality Metric We Will Use

- Classification Error

$$\text{Error} = \frac{\#\ \text{incorrect predictions}}{\#\ \text{examples}}$$

- Best possible value : 0.0
- Worst possible value: 1.0

Learning Objective

- Find the tree with the lowest classification error
- Trivial Solution: Construct a tree which has one path to a leaf for each example
- But this approach would simply memorize training data and will not generalize well to unseen test data

Learning Objective (2)

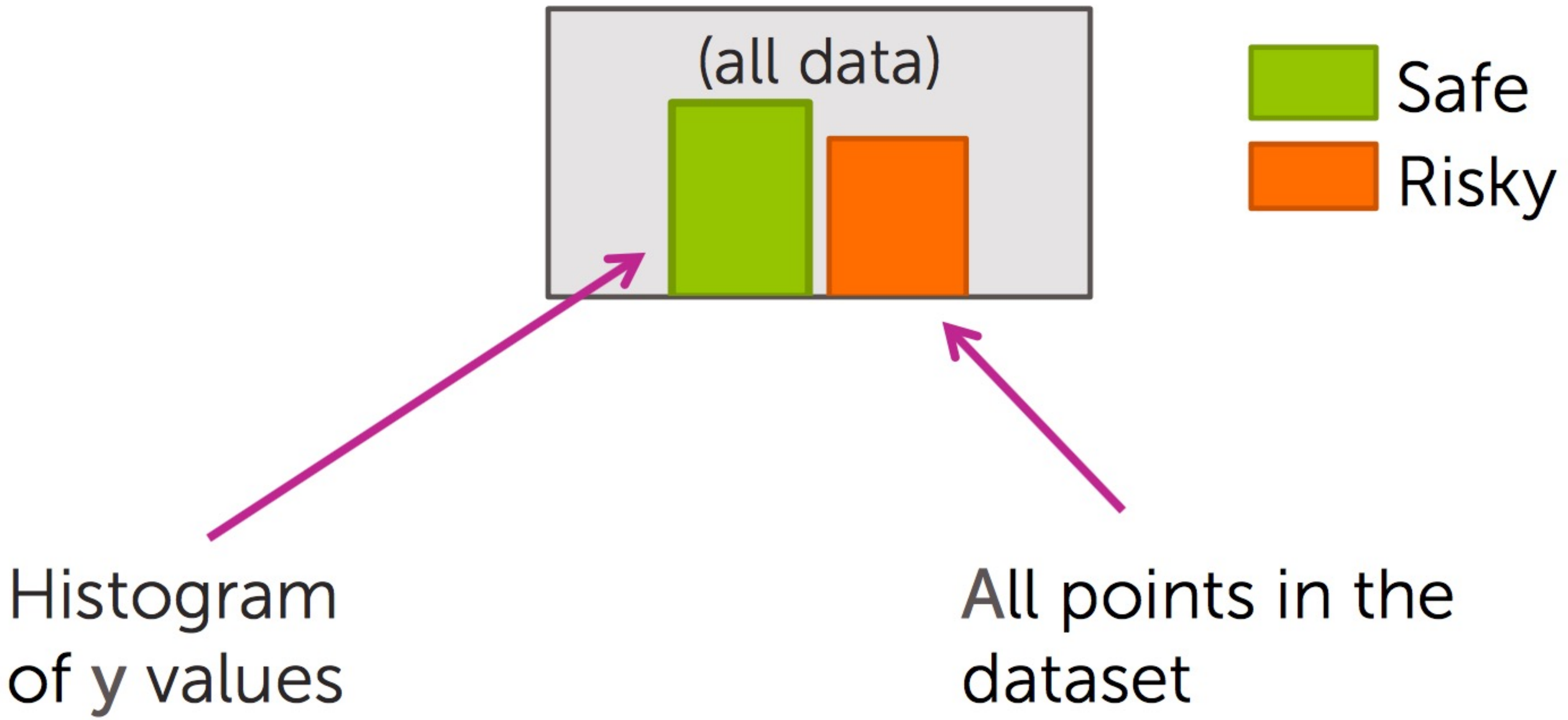
- Find the tree
  - that minimizes the expected number of tests required to identify the unknown object
- Which is an NP complete (no efficient solution)
  - Laurent Hyafil, Ronald L. Rivest, Constructing optimal binary decision trees is NP-complete, Information Processing Letters, Volume 5, Issue 1, 1976, Pages 15-17.

So How do we find an optimal tree?

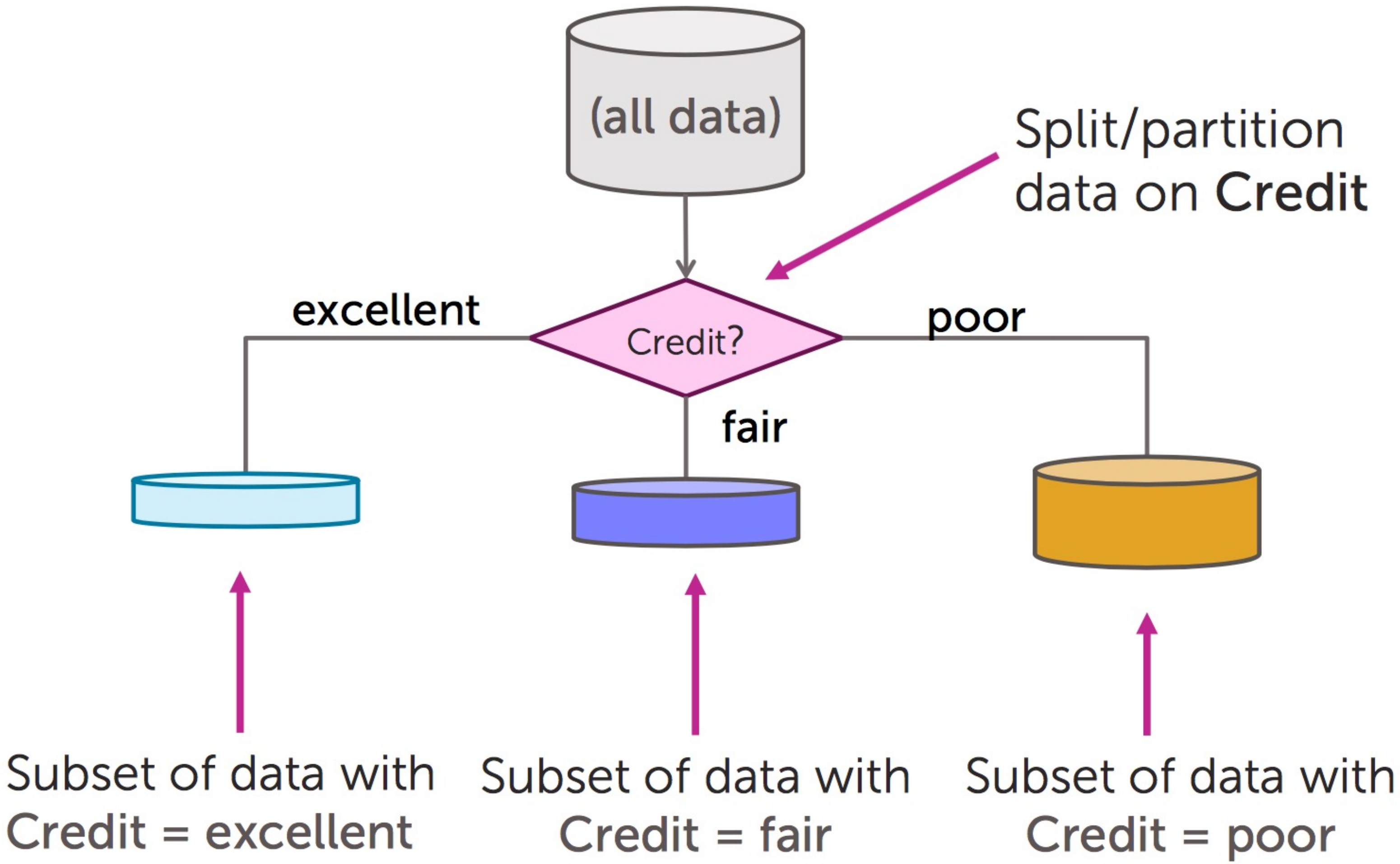
- We resort to a simple approch

Decision Tree Learning

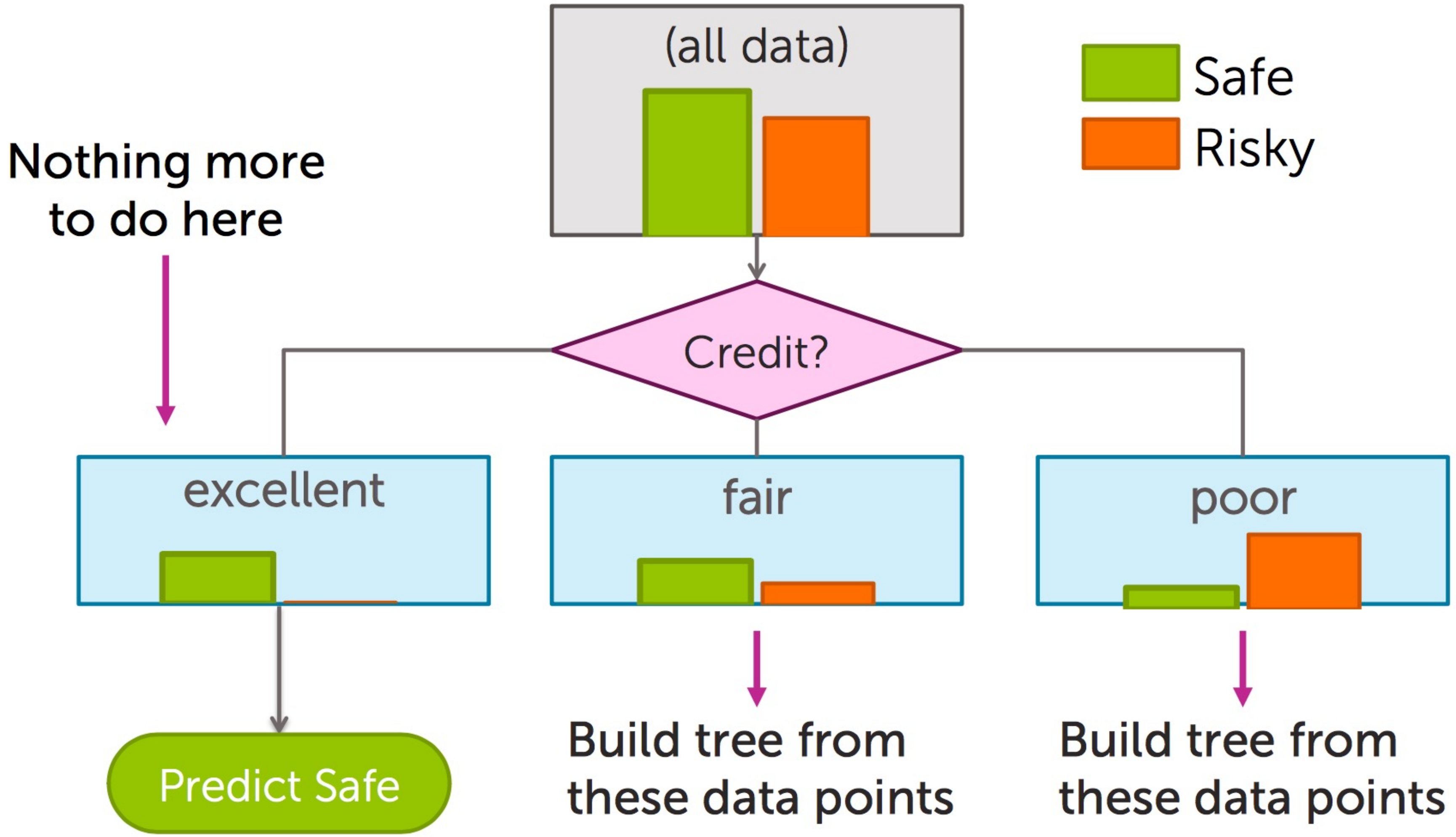
Step 1: Start with all Data on the Root Node



Step 2: Split on a “feature”



Step 3: Make Predictions OR Step 4: Recur



Decision Tree Learning Algorithm

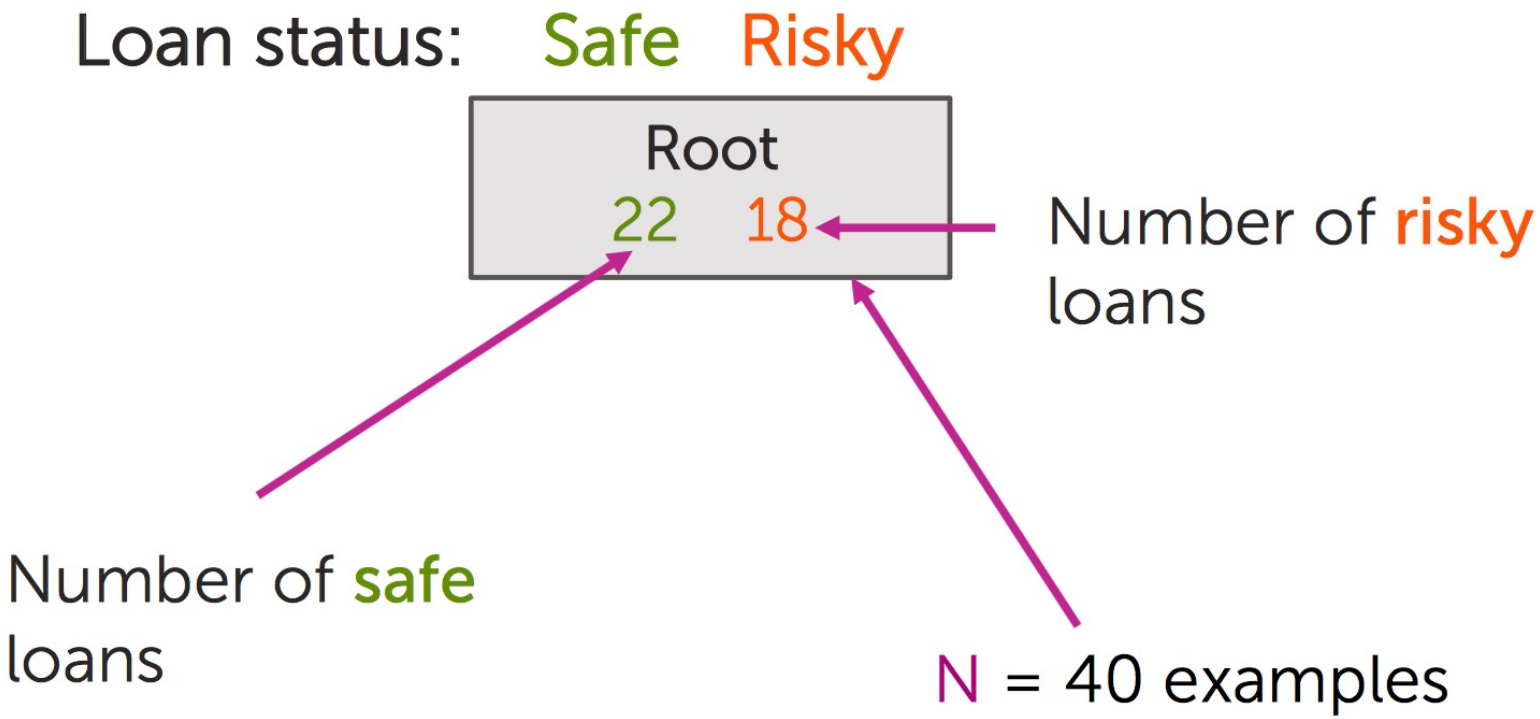
- **Step 1:** start with an empty tree
- **Step 2:** *select* a *feature* to *split* data
- For each split of the tree
  - **Step 3:** If nothing more to do, make predictions
  - **Step 4:** Otherwise, go to Step 2

Need to Fix Two Problems!

- Problem 1: Selecting which feature to make a split
- Problem 2: When to stop recursion

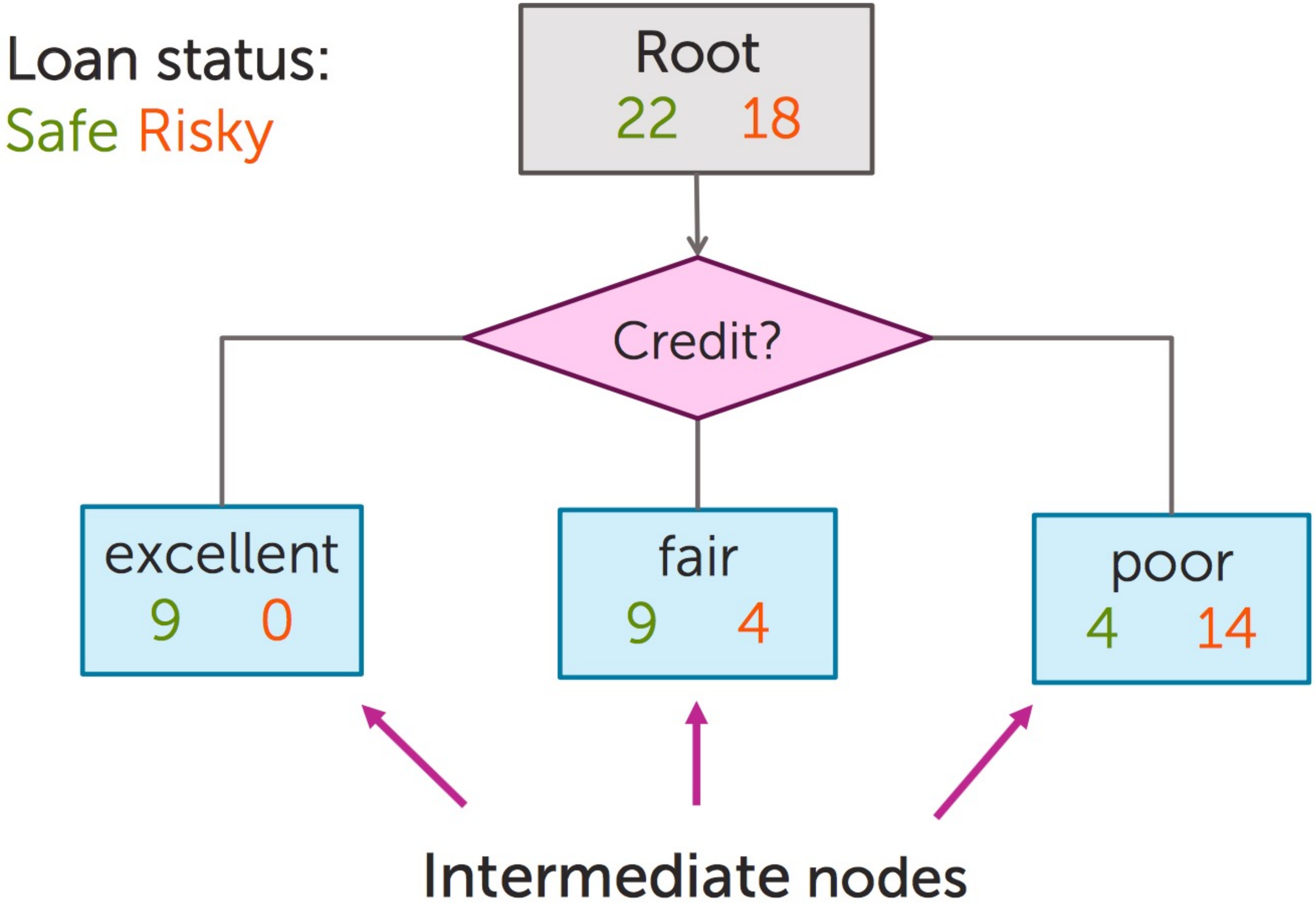
Choosing a Feature to Split

- Root Node for our example



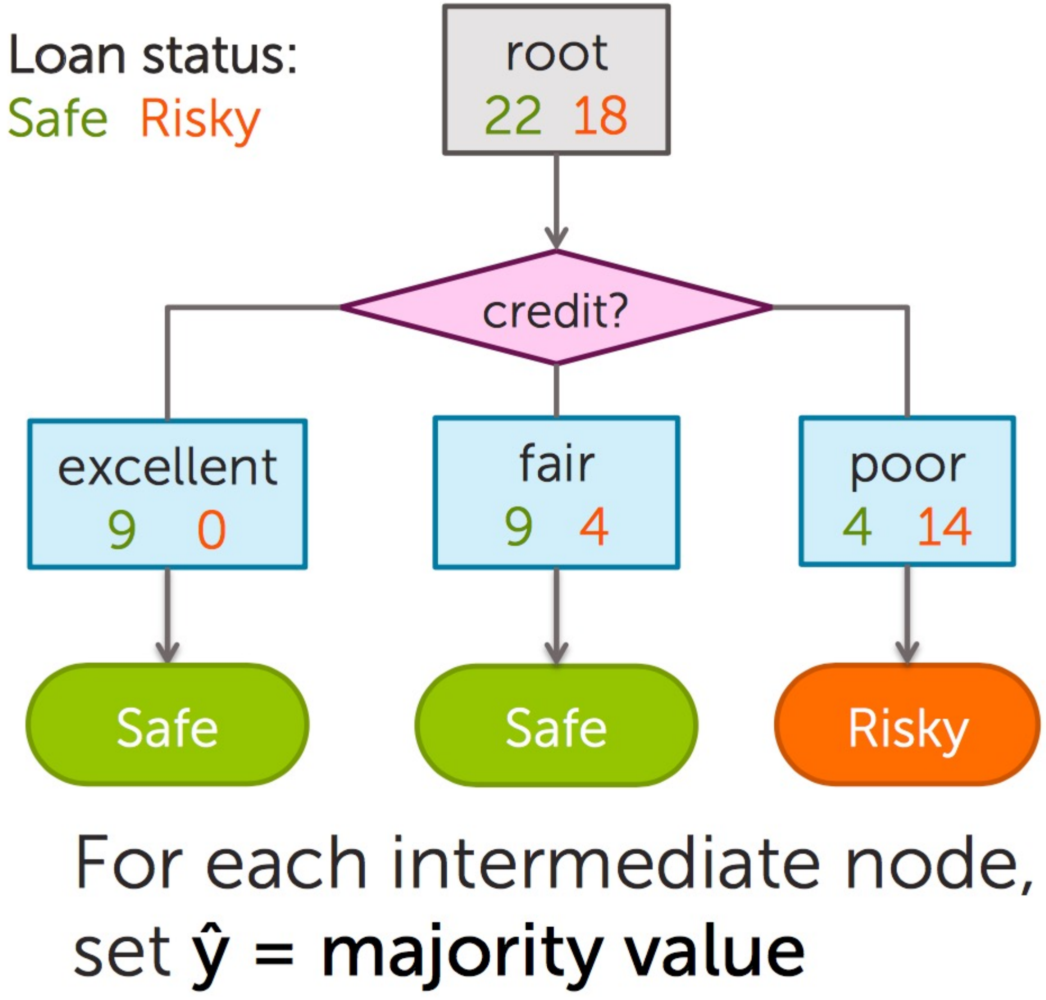
Choosing a Feature to Split (2)

- Split on "credit"



Choosing a Feature to Split (3)

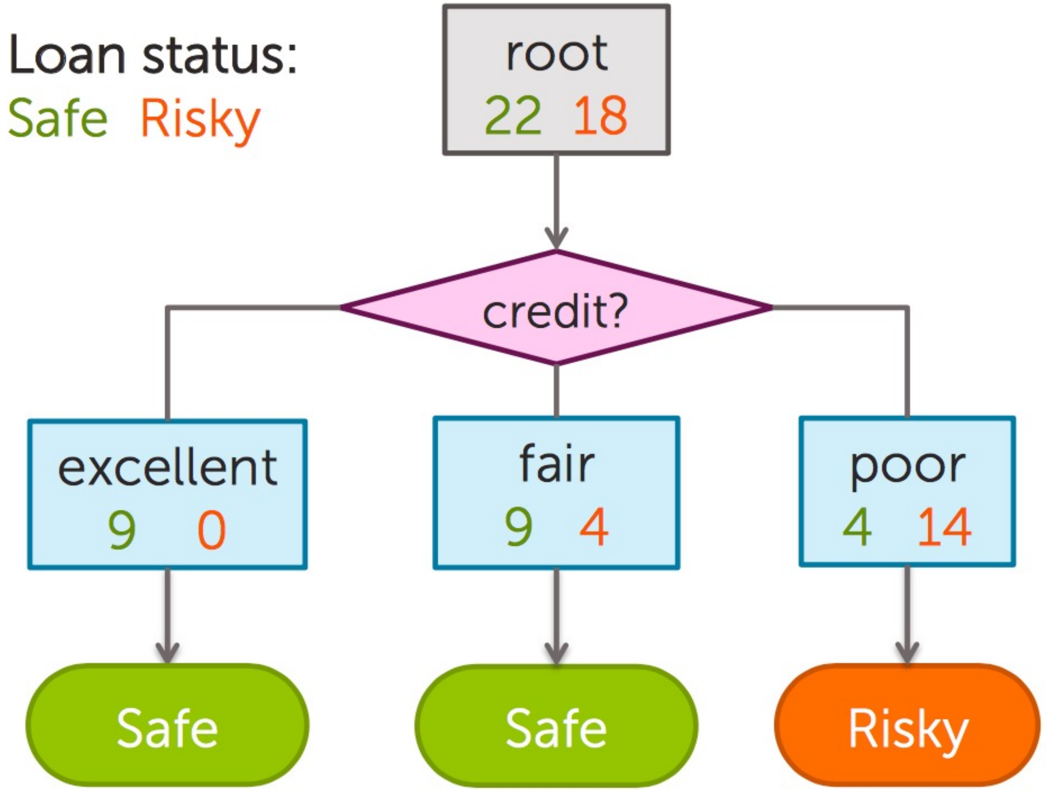
- Make predictions



Choosing a Feature to Split (4)

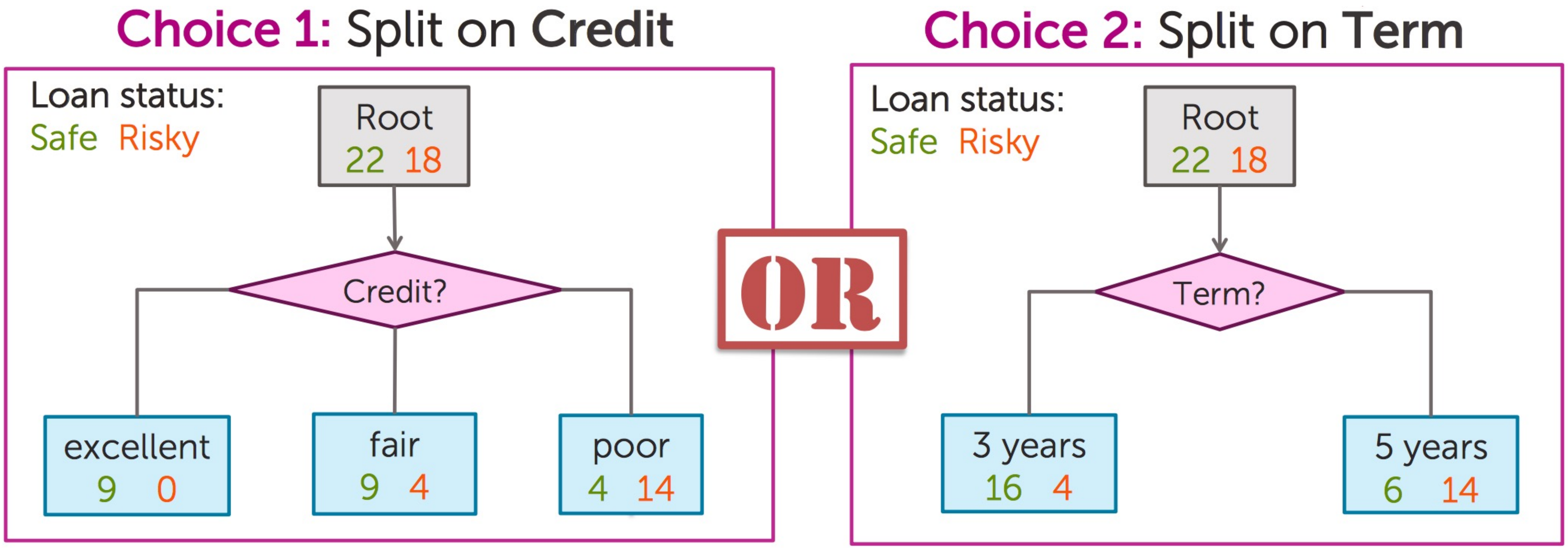
- Measure error

$$\text{Error} = \frac{\#\ \text{mistakes}}{\#\ \text{data points}}$$



For each intermediate node, set $\hat{y}$ = **majority value**

Comparing Splits

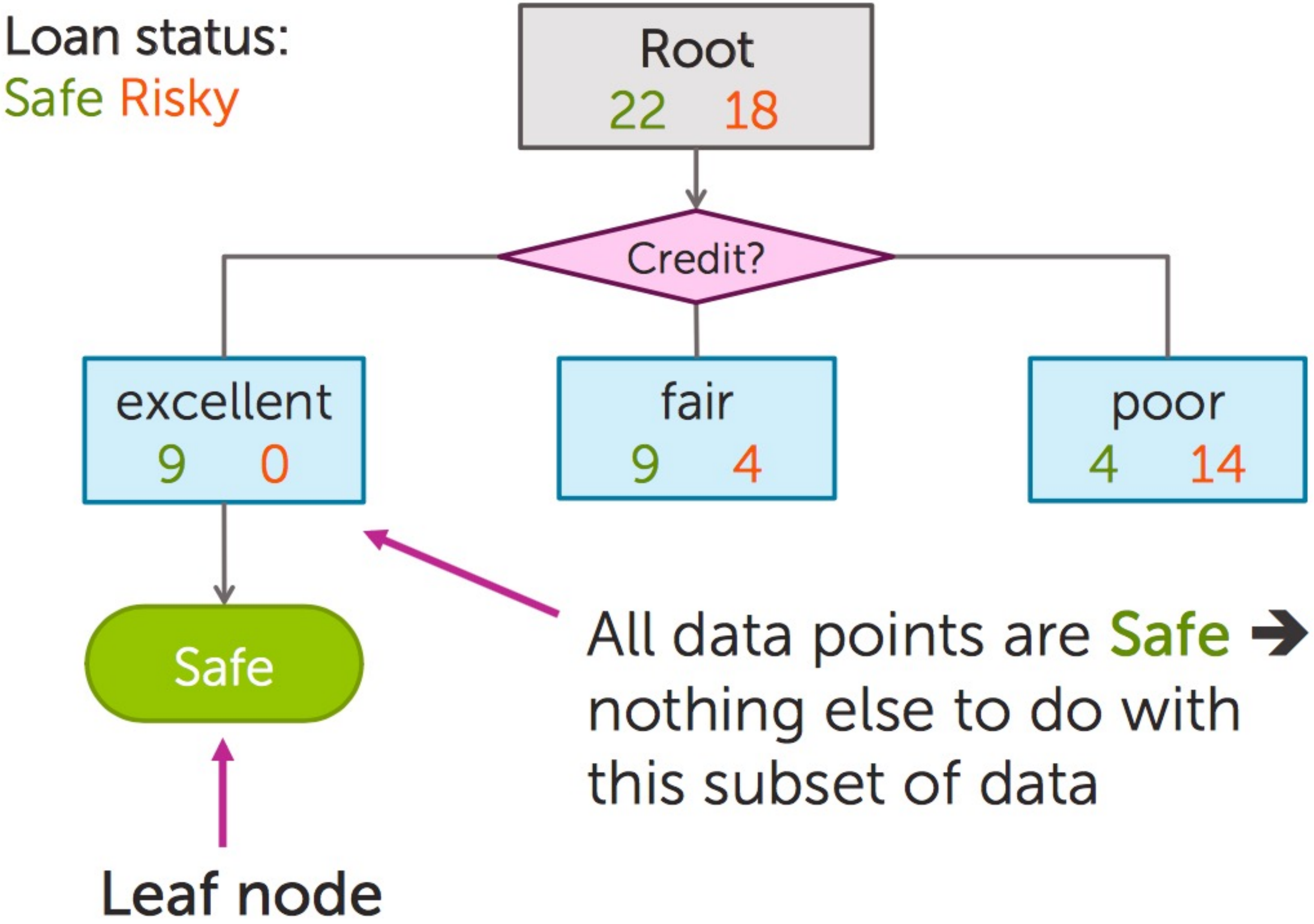


Thus we will favor “Credit”
This is called “Learning a Decision Stump”

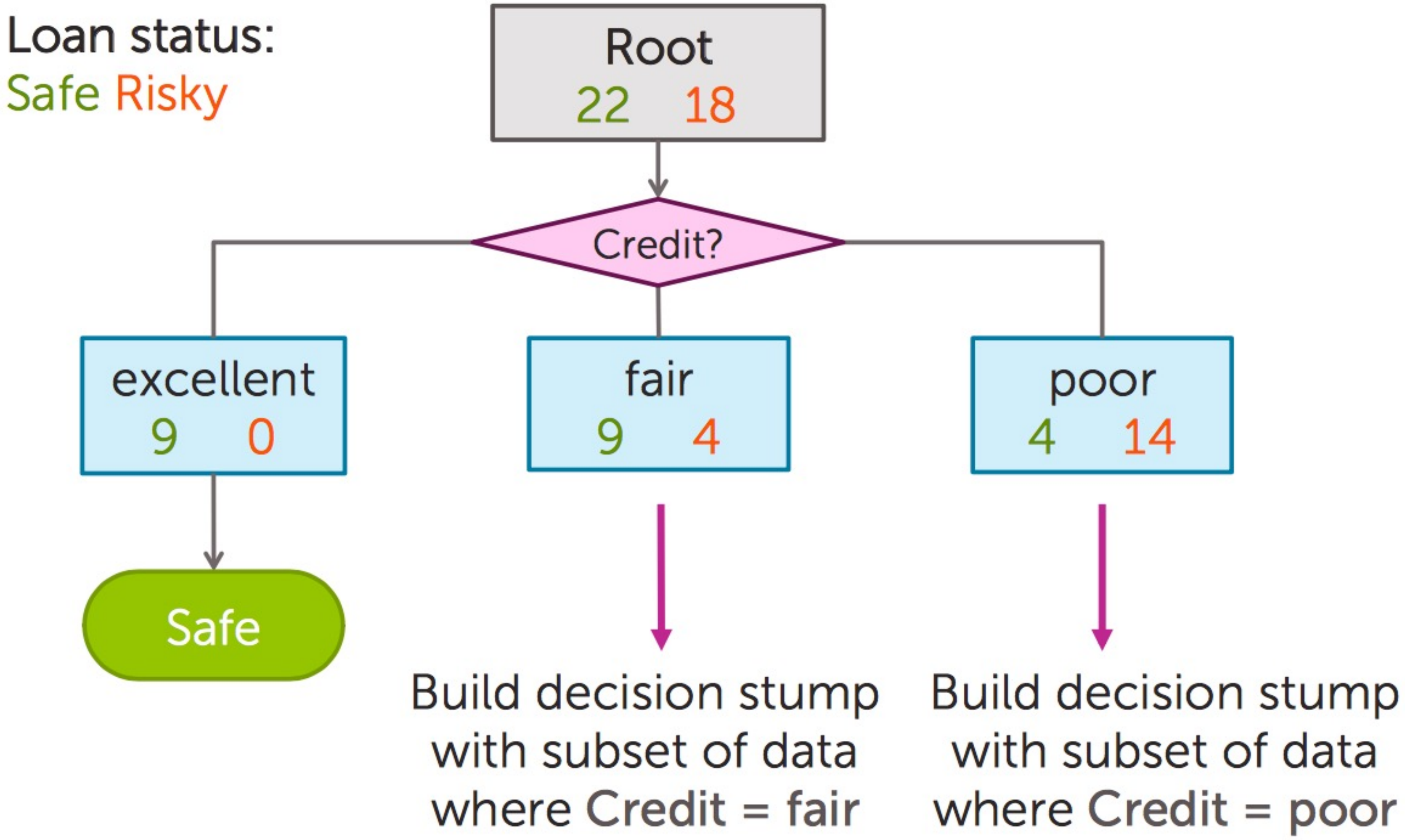
Decision Stump Learning

- Given a subset of data (a node in a tree)
- For each feature x_i
  - Split data of the node according to feature x_i
  - Compute classification error of the split
- Choose feature to split that has the lowest classification error

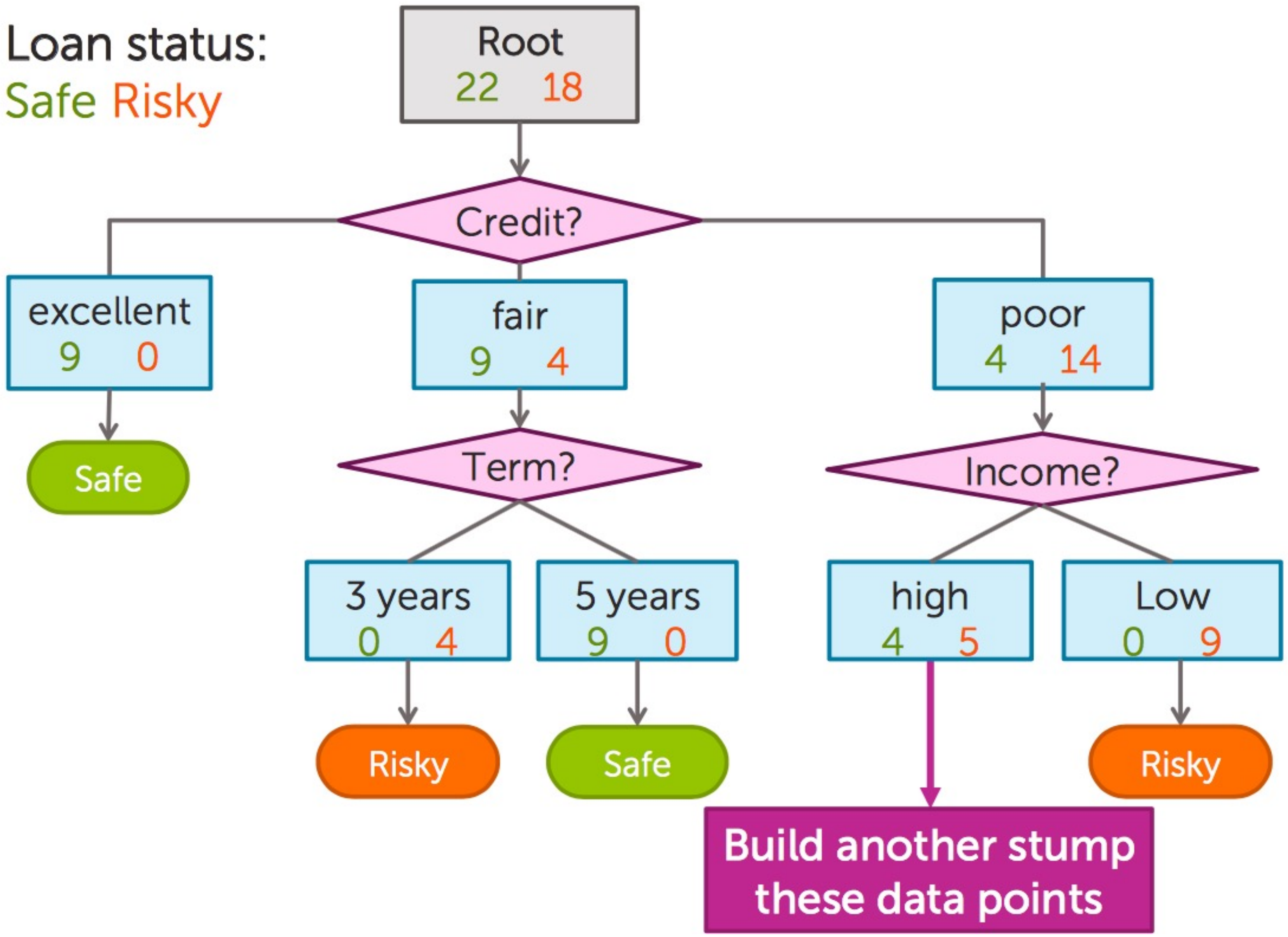
After Learning a Decision Stump



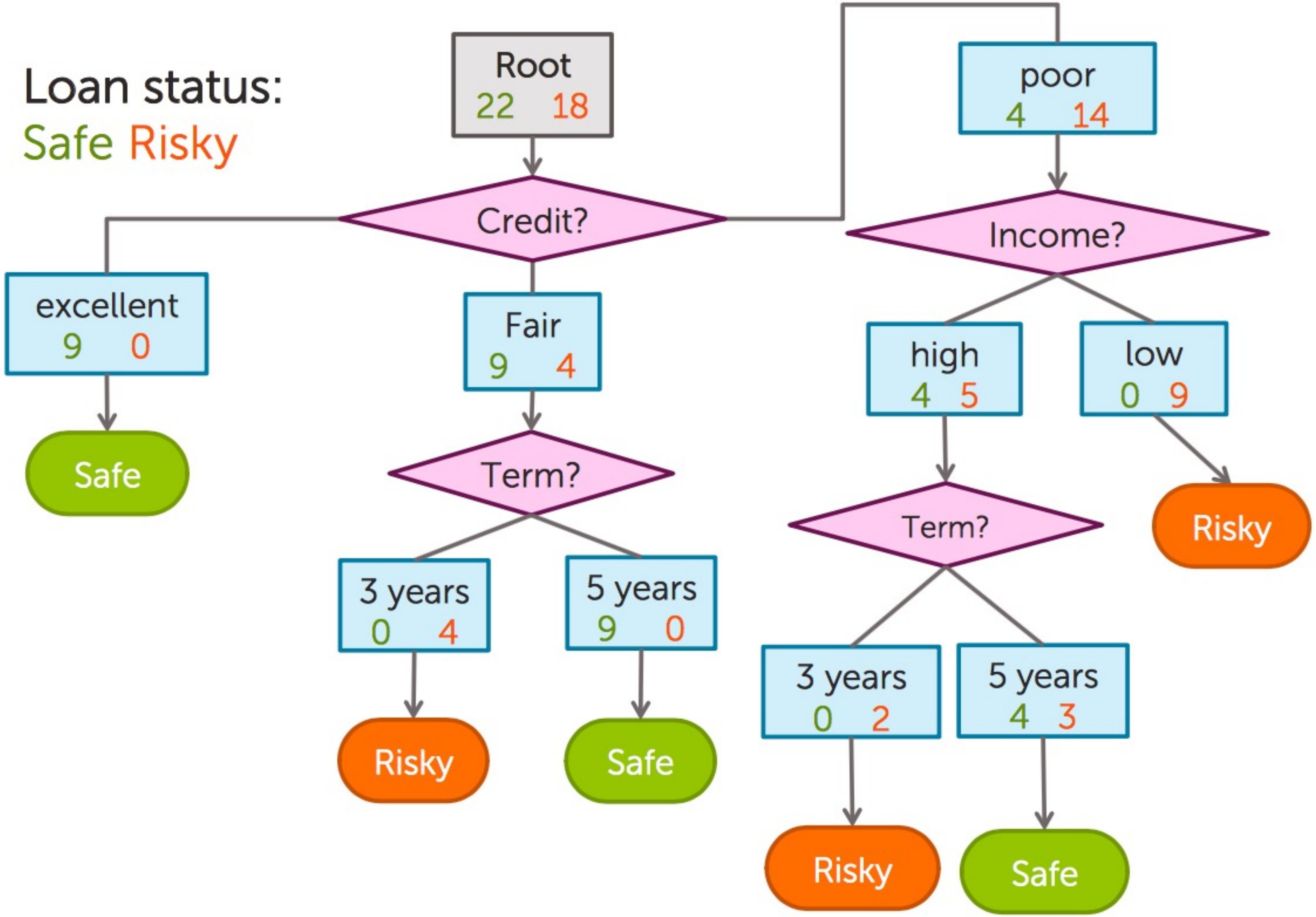
After Learning a Decision Stump (2)



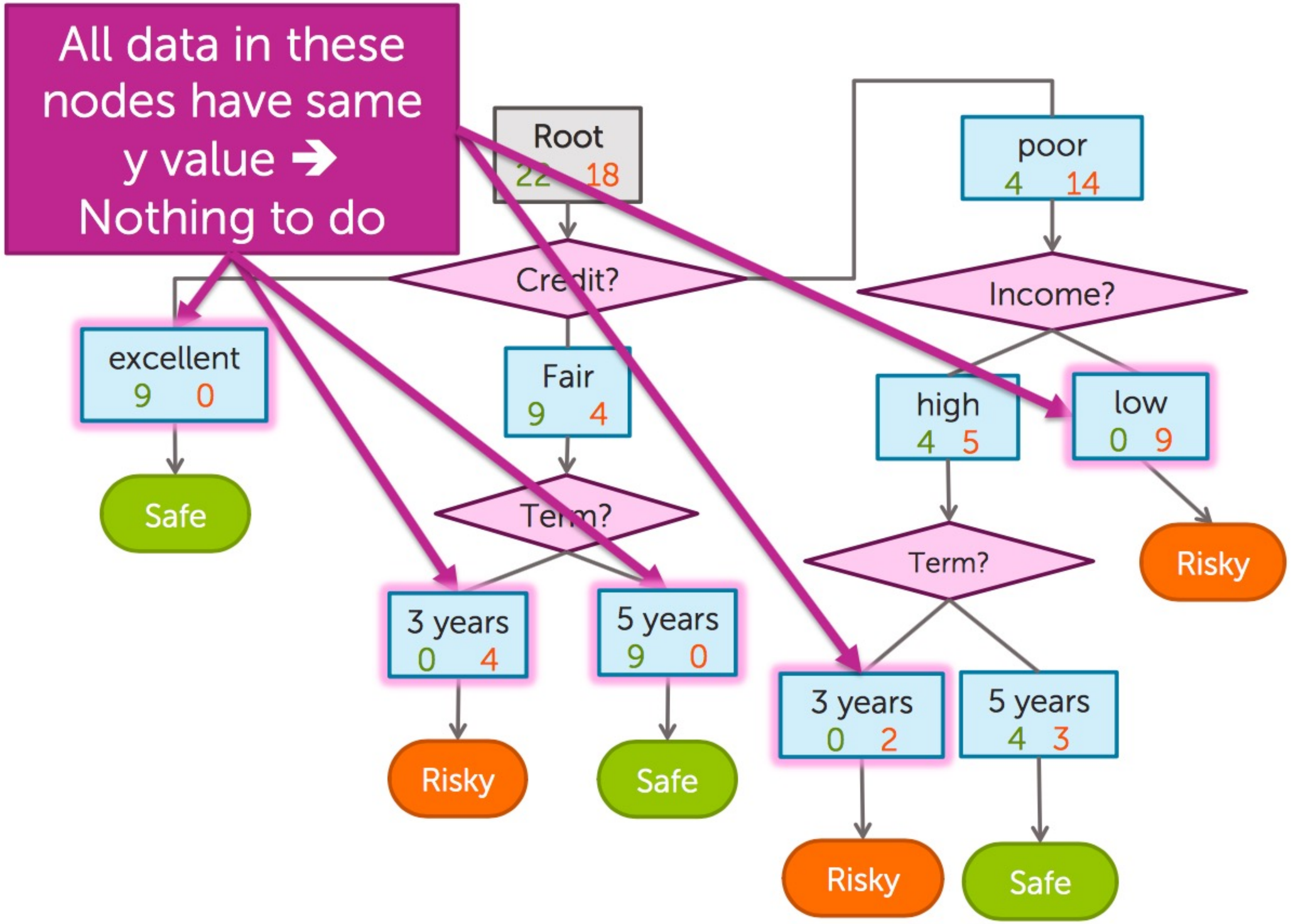
Second Level for our Example



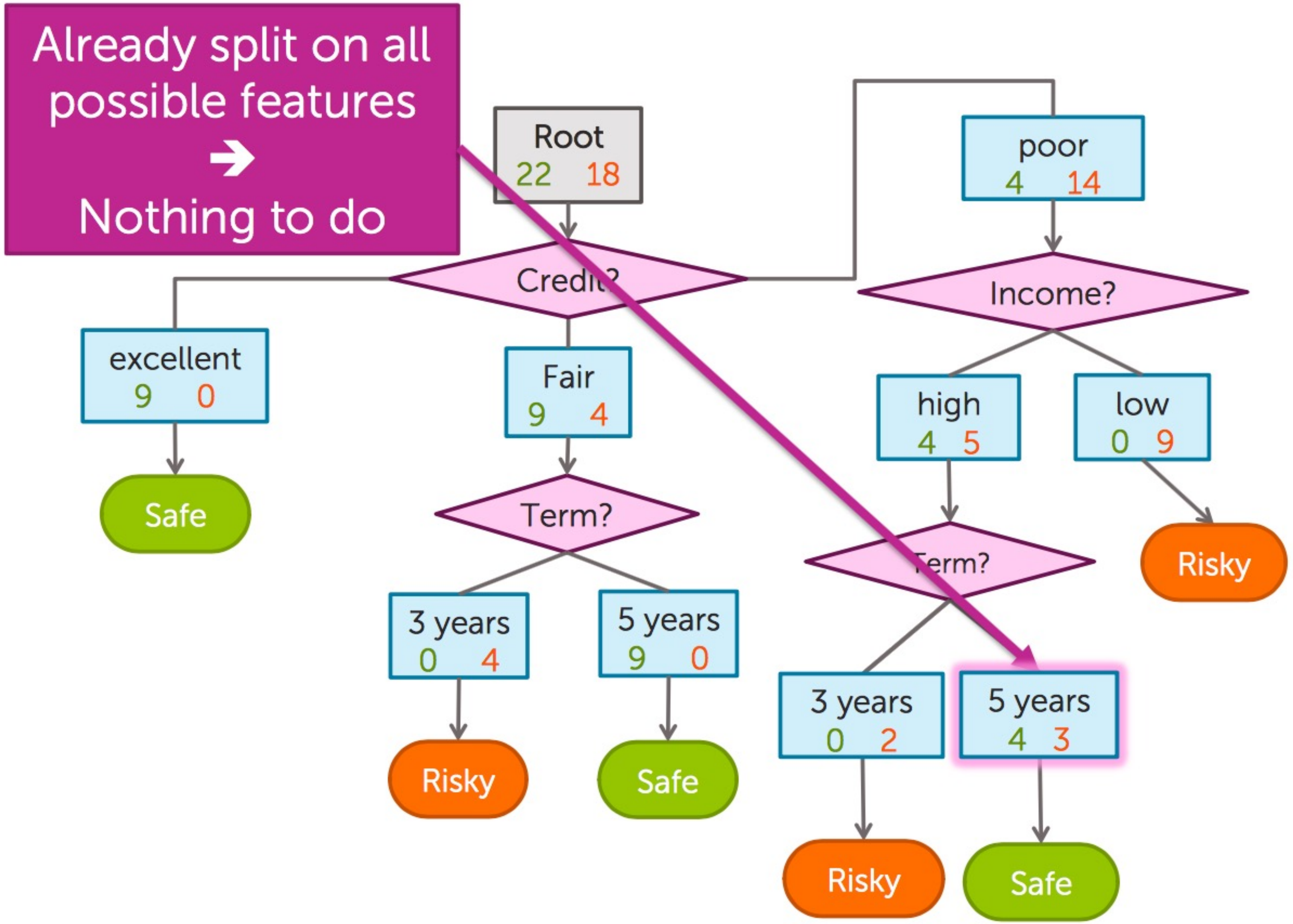
Third Level for our Example



Stopping Condition 1: All data in a node agrees on y



Stopping Condition 2: Already split on all features



Simple Greedy Decision Tree Learning Algorithm

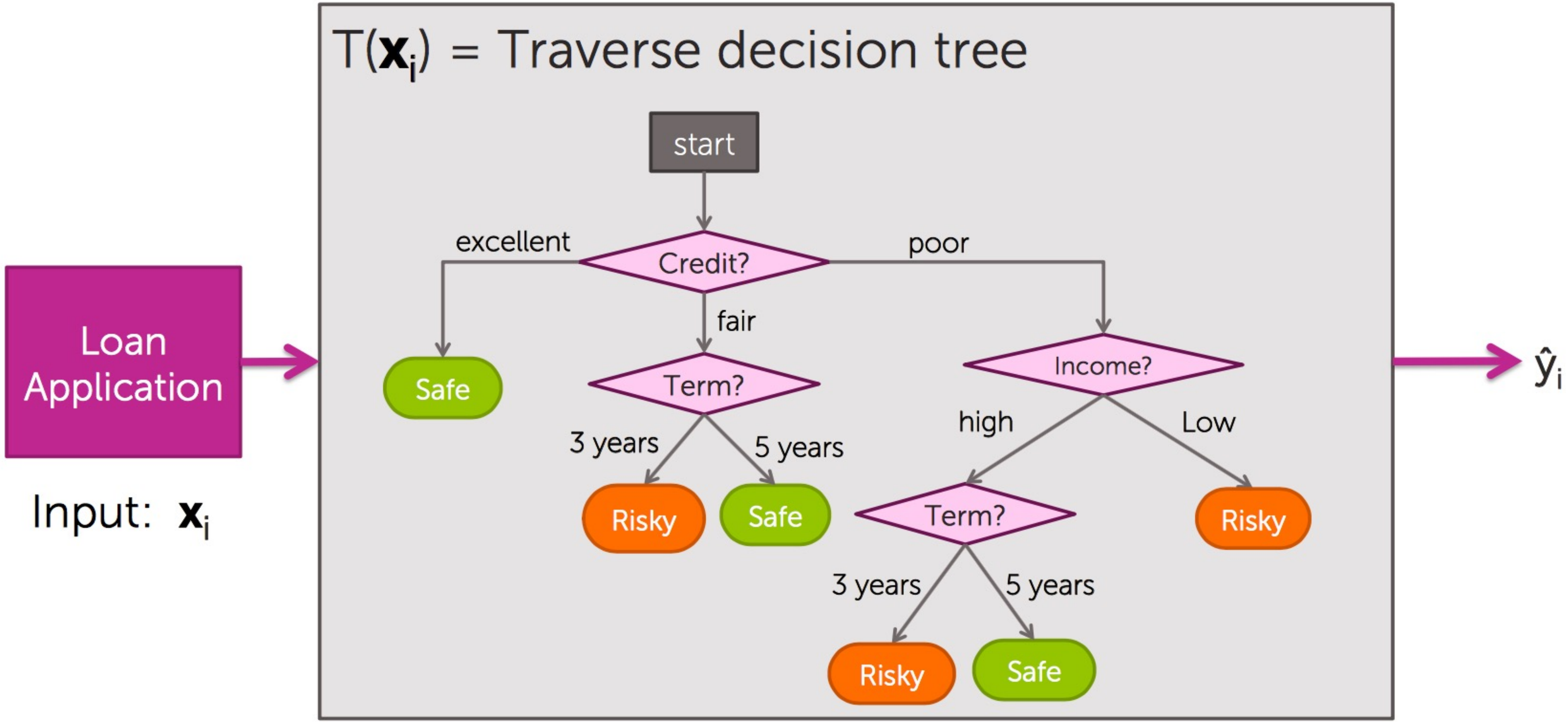
- Step 1: Start with an empty tree
- Step 2: Select a feature to split data
- For each split of the tree:
  - Step 3: If nothing more to, make predictions
  - Step 4: Otherwise, go to Step 2 & continue (recurse) on this split

Pick feature split leading to lowest classification error

Stopping conditions 1 & 2

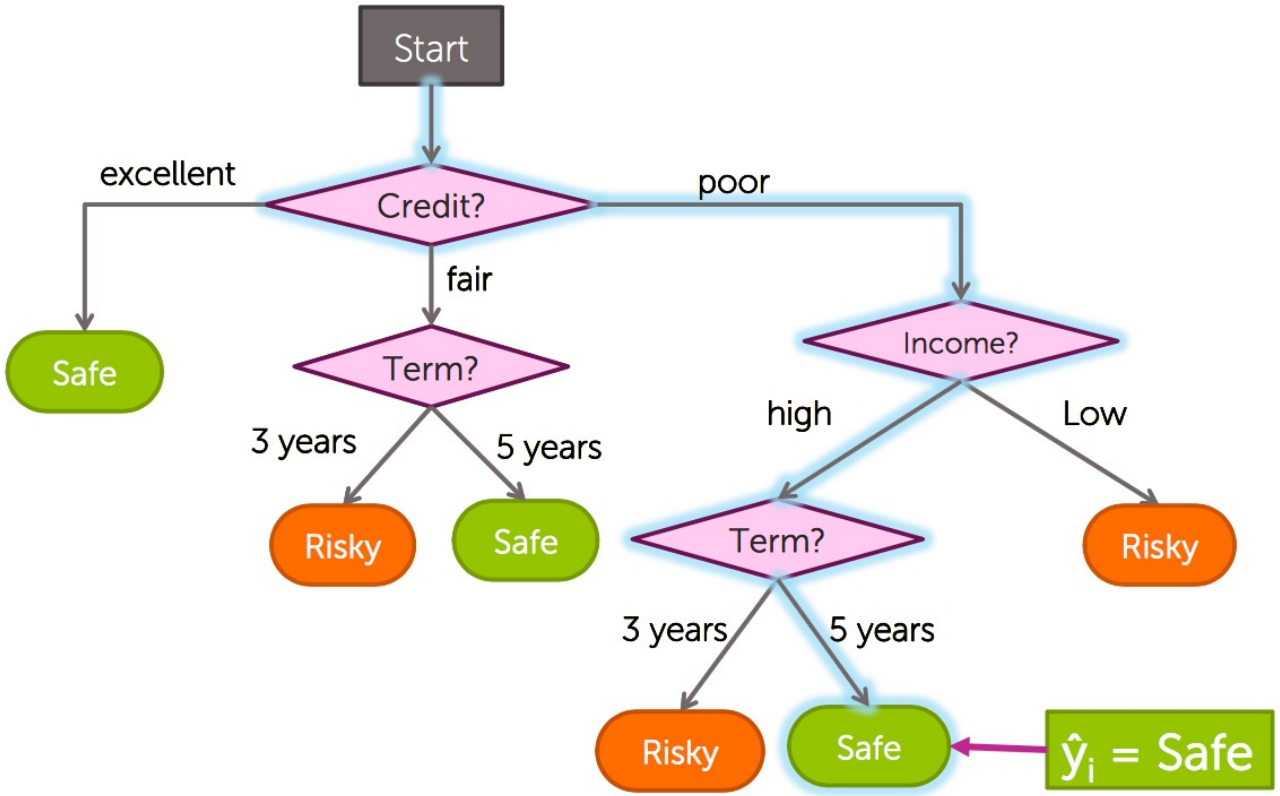
Recursion

Predictions with a Learned DT



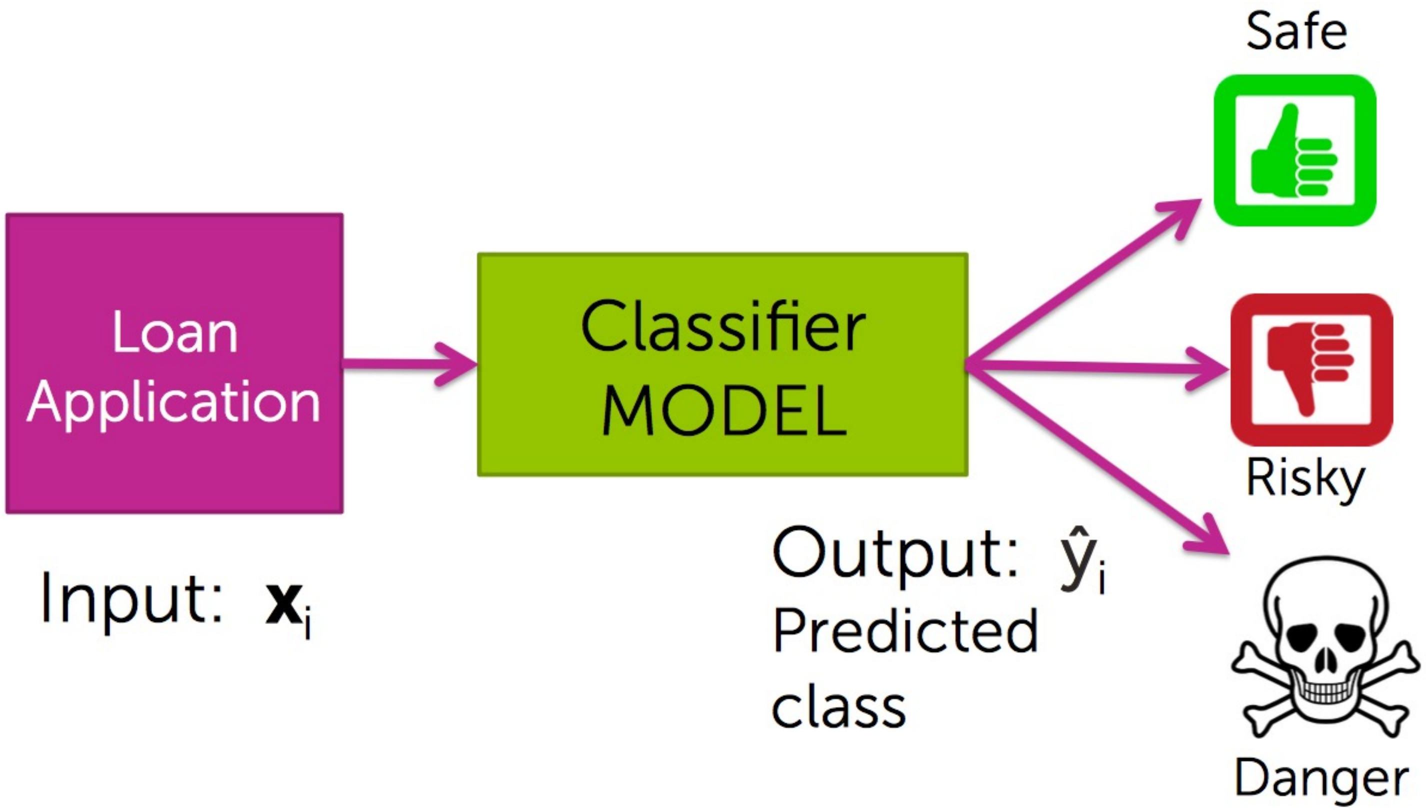
Predictions with a Learned DT (2)

$\mathbf{x}_i$ = (Credit = poor, Income = high, Term = 5 years)



DT for Multi-class Problems

- Can be learned the same way, without any modifications



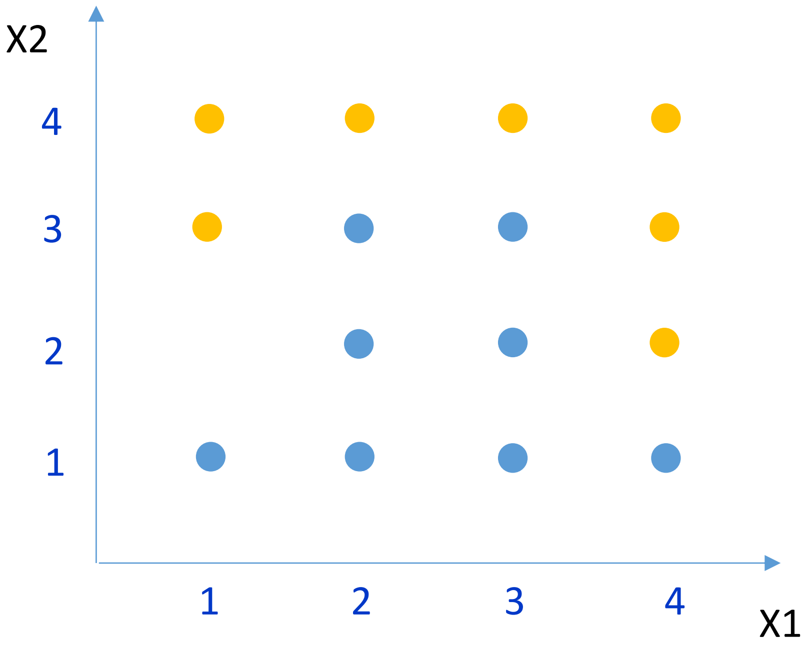
Decision Tree Class Boundaries

DT Decision Boundaries

- DT’s divide the input space into axis-parallel rectangles and label each rectangle with one of the given classes (majority class)

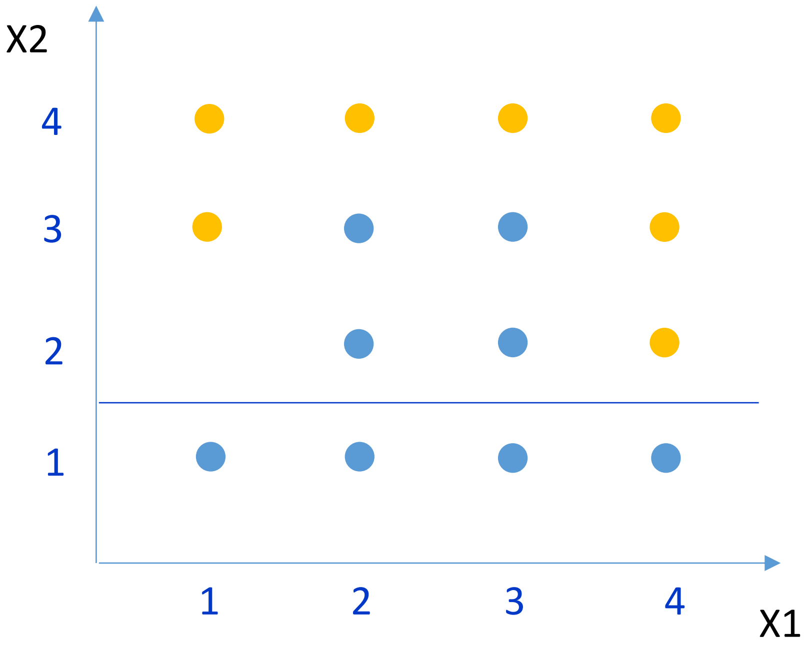
DT Decision Boundaries (2)

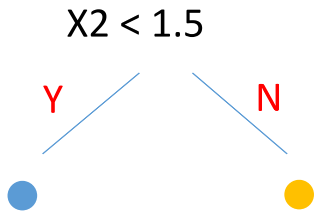
- DT’s divide the input space into axis-parallel rectangles and label each rectangle with one of the given classes **(majority class)**



DT Decision Boundaries (3)

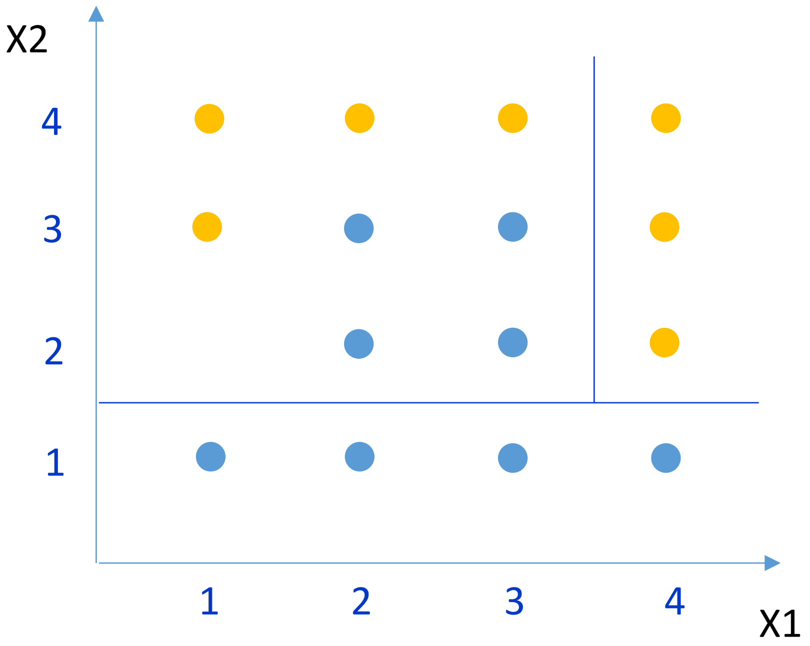
- DT’s divide the input space into axis-parallel rectangles and label each rectangle with one of the given classes **(majority class)**

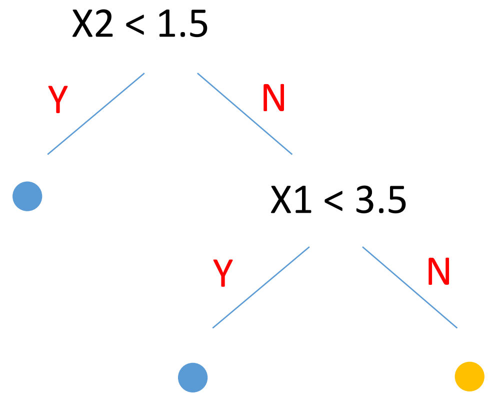




DT Decision Boundaries (4)

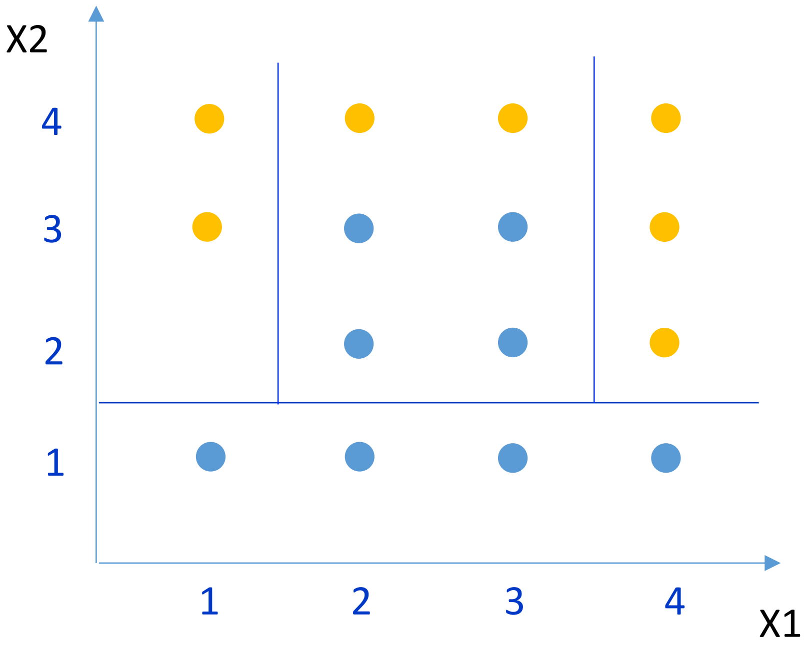
- DT's divide the input space into axis-parallel rectangles and label each rectangle with one of the given classes **(majority class)**

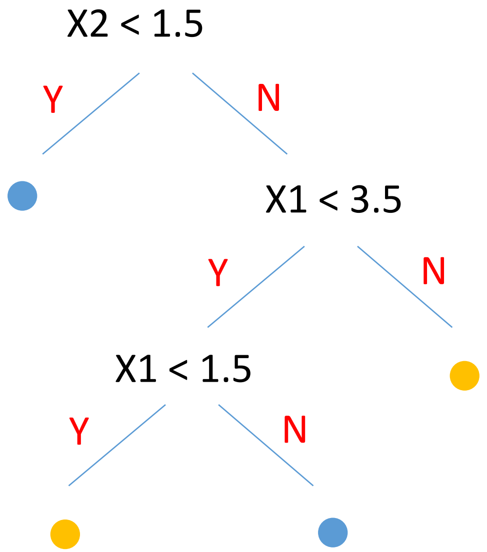




DT Decision Boundaries (5)

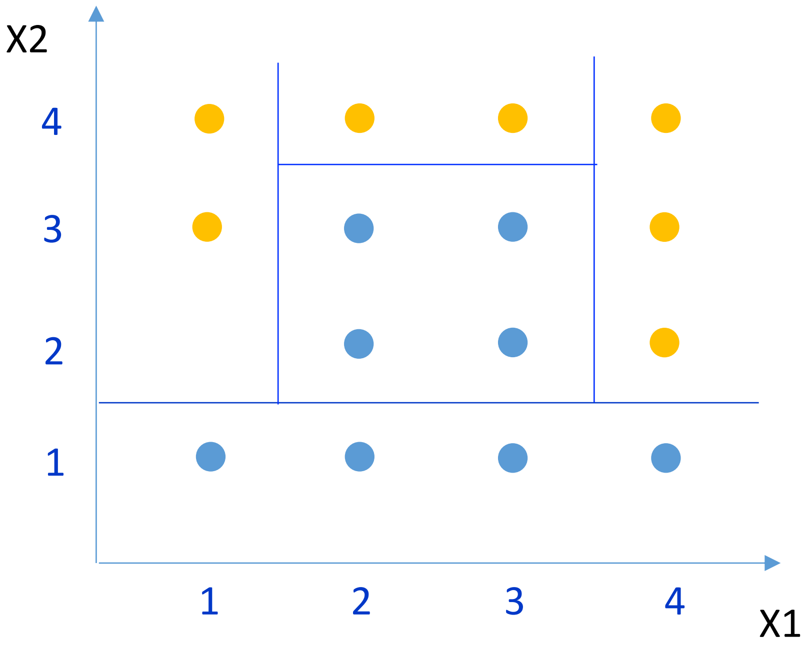
- DT's divide the input space into axis-parallel rectangles and label each rectangle with one of the given classes **(majority class)**



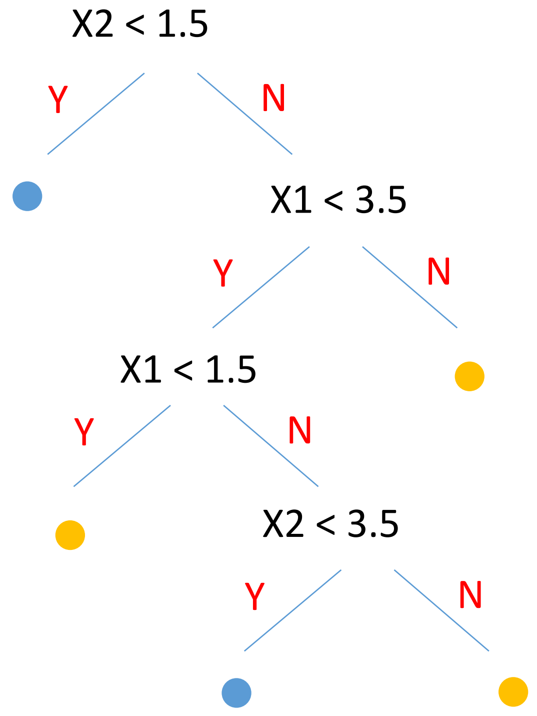


DT Decision Boundaries (6)

- DT's divide the input space into axis-parallel rectangles and label each rectangle with one of the given classes **(majority class)**

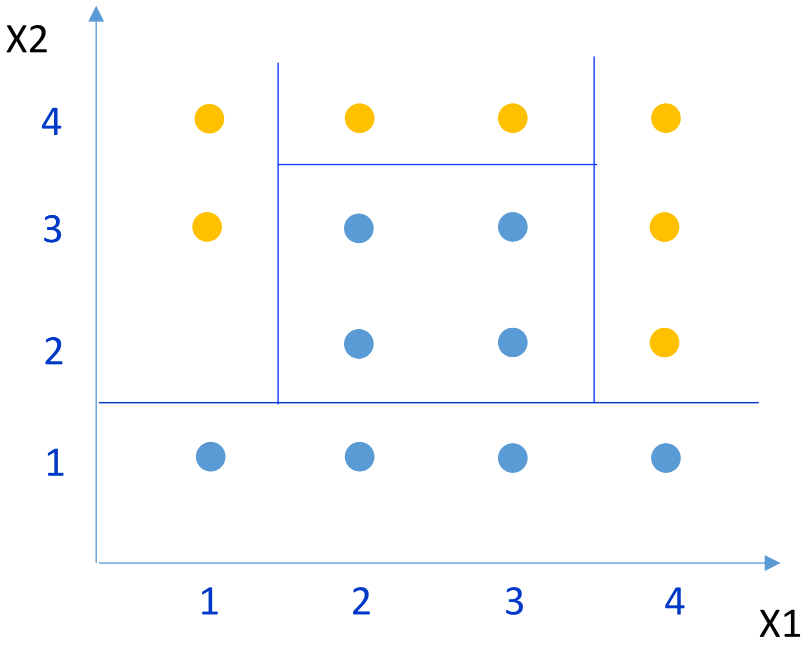


$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

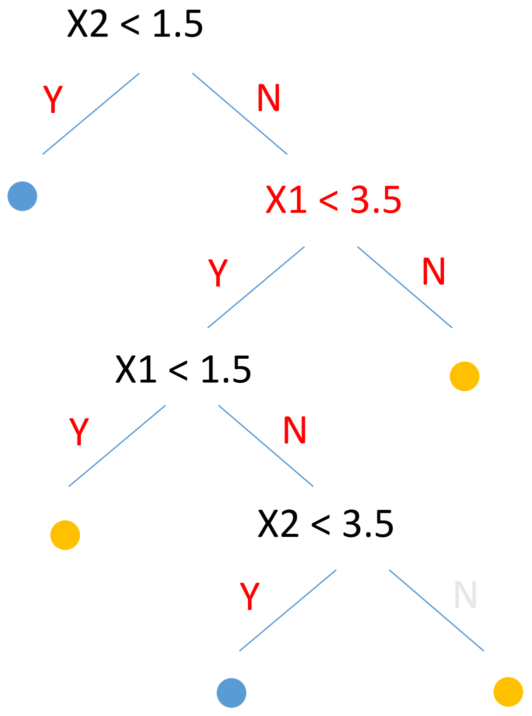


DTs as Regression Trees

- Exactly the same thing, except that the label of a given sample is ***the average response*** of a the rectangle to which it belongs



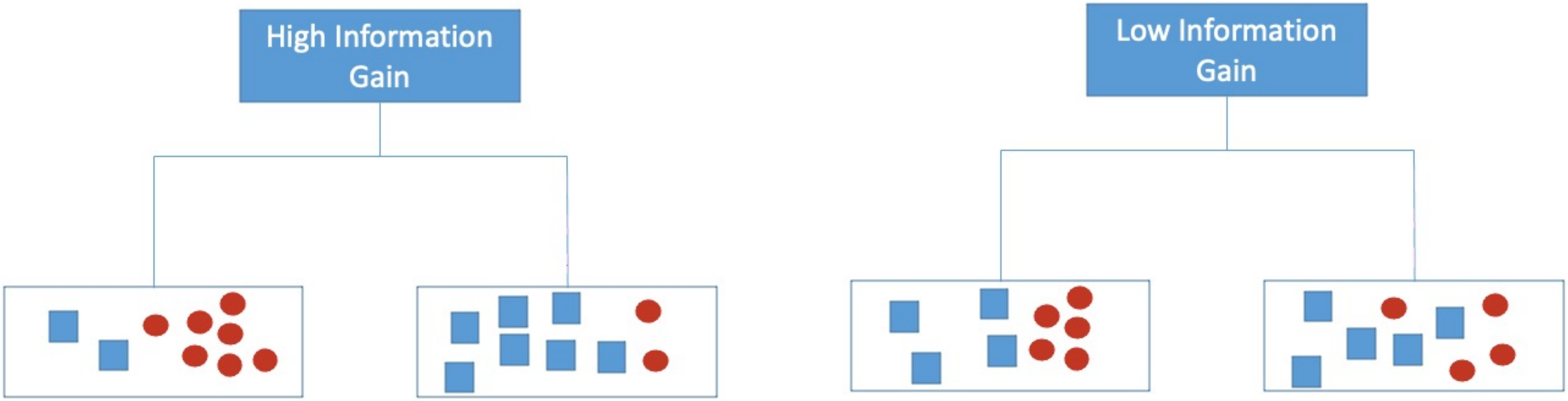
$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$



Information Gain

Information Gain

- How well a given feature (or predictor) separates training data according to their target classification



Formally

$$IG(D_p, x_i) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

- IG: Information Gain
- x_i : feature to perform the split
- N_p: number of samples in the parent node
- N_{left} : number of samples in the left child node
- N_{right} : number of samples in the right child node
- I: impurity
- D_p: training subset of the parent node
- D_{left} : training subset of the left child node
- D_right: training subset of the right child node

Revisting the DT Learning Algorithm

- **Step 1:** start at the root node (with an empty tree)
- **Step 2:** *Split* the parent node using feature x_i to maximize the information gain
- **Step 3:** Assign training samples to the new child nodes
- **Step 4:** Stop if leave nodes are pure, or a stopping criteria has met, otherwise repeat step 2 and 3 for each child node.

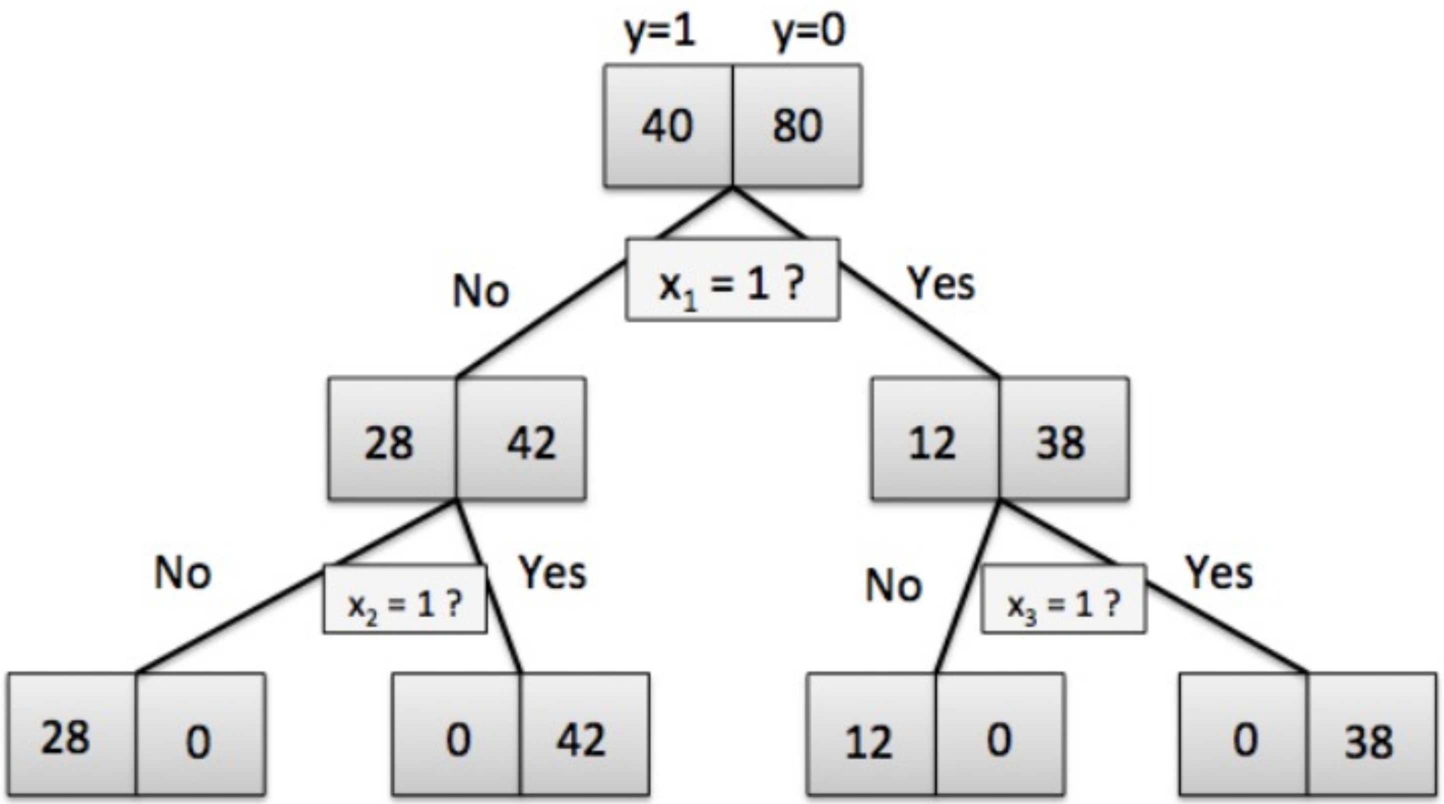
Revisiting the Stopping Rules

1. The leaf nodes are pure
2. No more features to split
3. A maximal node depth has reached
4. *Splitting a node does not lead to information gain*

Important!

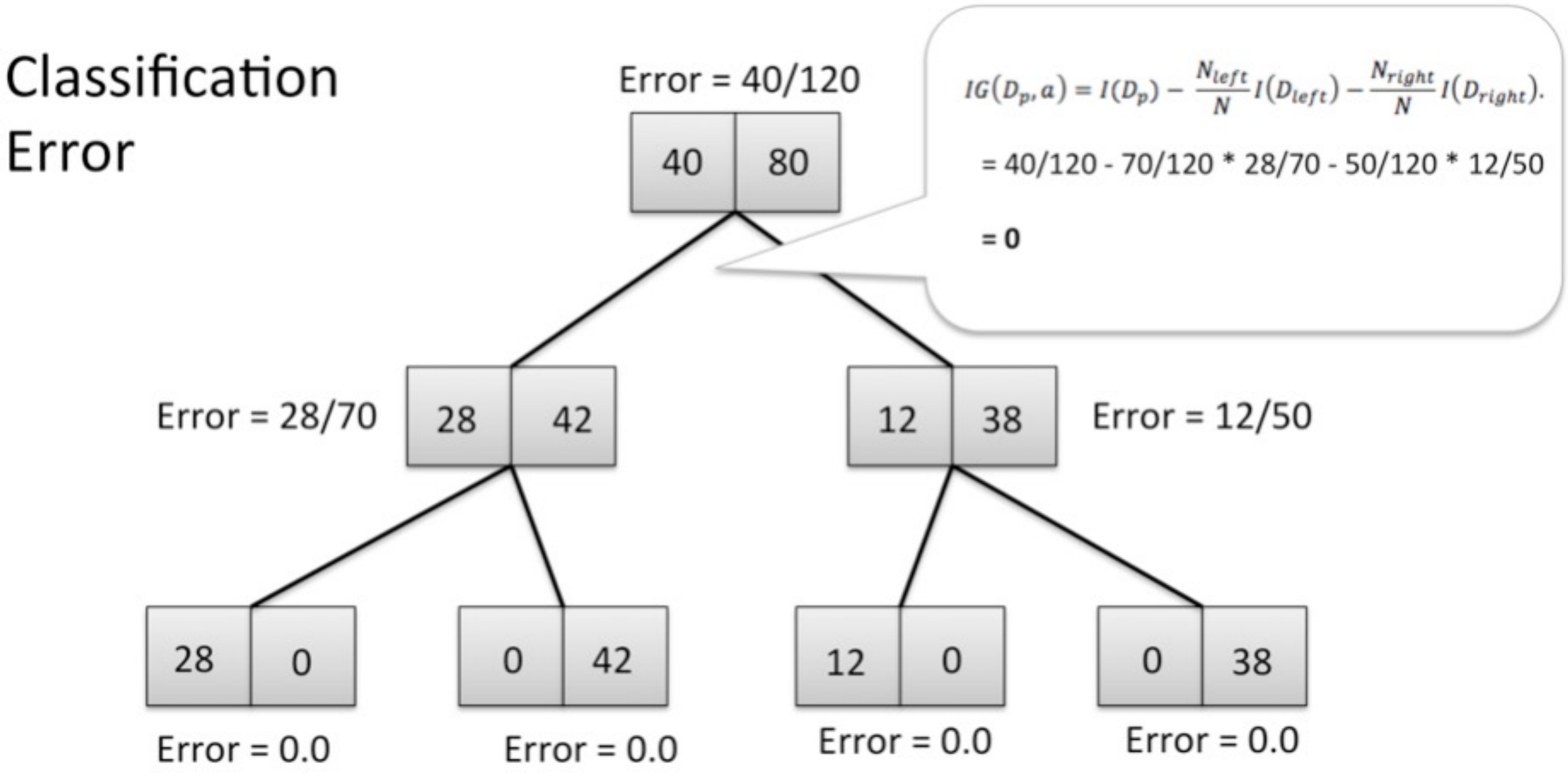
- So far, we have used the classification error as a way to measure information gain

An Interesting Example



https://sebastianraschka.com/faq/docs/decisiontree-error-vs-entropy.html

An Interesting Example



https://sebastianraschka.com/faq/docs/decisiontree-error-vs-entropy.html

Entropy: A Better Way to Measure Node Impurity

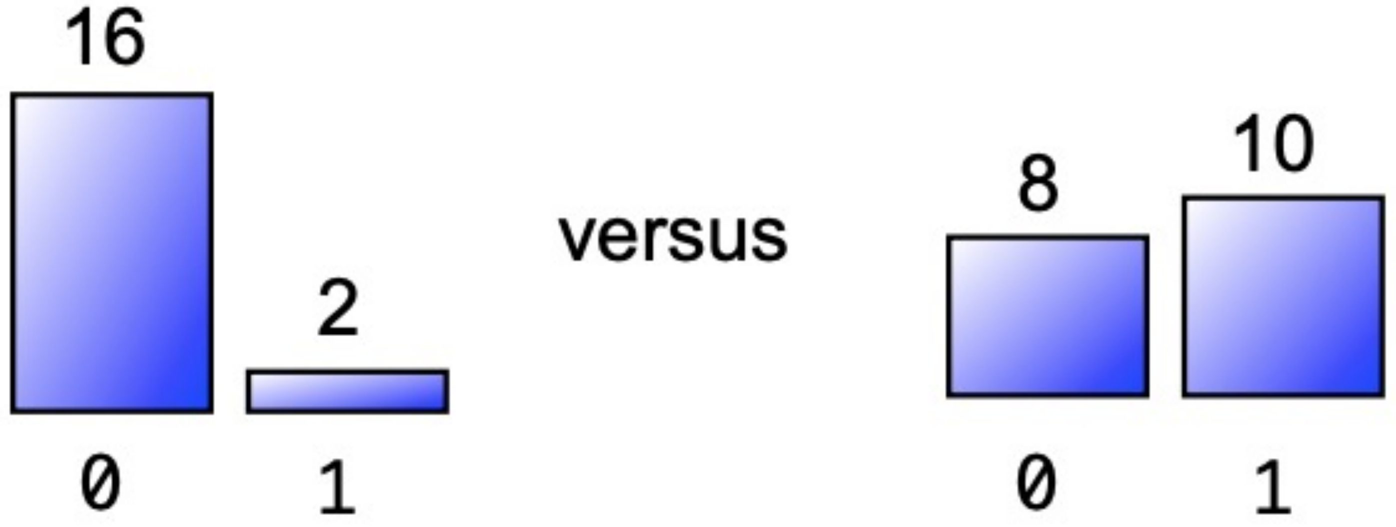
- The entropy of a discrete random variable is a number that quantifies the uncertainty inherent in its possible outcomes.

- Consider flipping two different coins

Flipping Two Different Coins

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

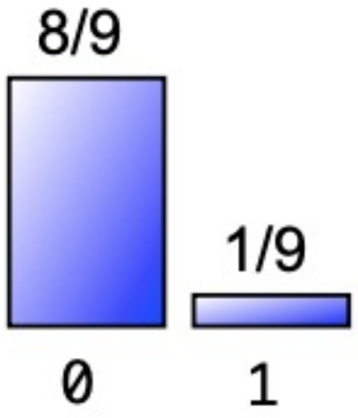
Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?

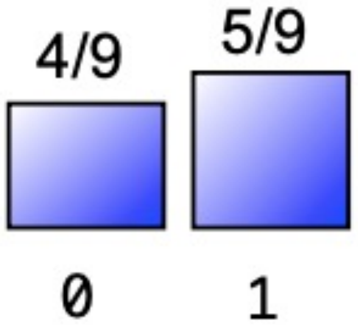


Quantifying Uncertainity

- The entropy of a coin with probability p of heads is given by

$$-p\log_2(p) - (1-p)\log_2(1-p)$$





$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

Entropy: A Better Way to Measure Node Impurity

$$I(t) = -\sum_{i=1}^{C} p(i|t)\, log_2\, p(i|t)$$

t: a given node

C: total number of classes

$p(i|t)$: the proportion of samples that belong to class i at node t

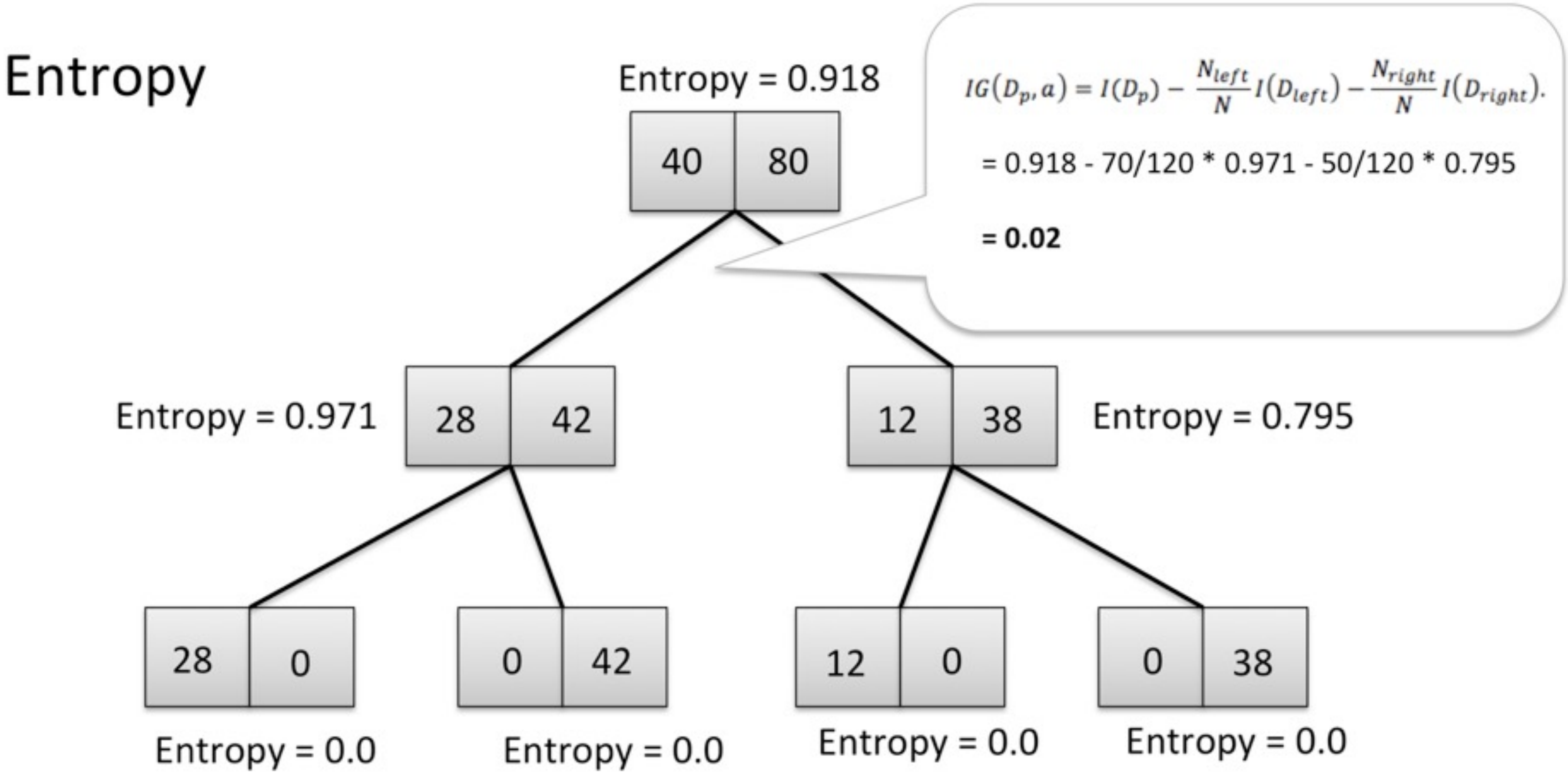
Measuring Entropy of a Node

40	80

$$I(t) = -\sum_{i=1}^{2} p(i|t)\, log_2\, p(i|t) \qquad = -\frac{40}{120} log_2\left(\frac{40}{120}\right) - \frac{80}{120} log_2\left(\frac{80}{120}\right)$$

$$= 0.918$$

Back to our Example but with Entropy



https://sebastianraschka.com/faq/docs/decisiontree-error-vs-entropy.html

Gini Index

- Another measure for impurity
- You must read about it yourselves!

Once Again, Revisting the Tree Learning Algorithm

- **Step 1:** start at the root node (with an empty tree)
- **Step 2:** *Split* the parent node using feature x_i to maximize the information gain (using Entropy or Gini)
- **Step 3:** Assign training samples to the new child nodes
- **Step 4:** Stop if leave nodes are pure, or a stopping criteria has met, otherwise repeat step 2 and 3 for each child node.

Example of DTs in Software Analysis

4 IEEE TRANSACTIONS ON RELIABILITY, VOL. 49, NO. 1, MARCH 2000

Classification-Tree Models of Software-Quality Over Multiple Releases

Taghi M. Khoshgoftaar, *Member, IEEE*, Edward B. Allen, *Member, IEEE*, Wendell D. Jones, and John P. Hudepohl

This week’s reading!!

Summary

1. What are decision trees? When should we prefer them?
2. How do we learn decision trees?
   - Learning decision stumps
   - Greedy decision tree learning algorithm
   - DT Classification boundaries
   - Regression Trees
3. Generalizing DT learning
   - Information gain
   - Classification vs Entropy for measuring information gain
   - Gini index