HPE DSI 311
Introduction to Machine Learning

Spring 2023

Instructor: Ioannis Konstantinidis

Overview

Divide and Conquer

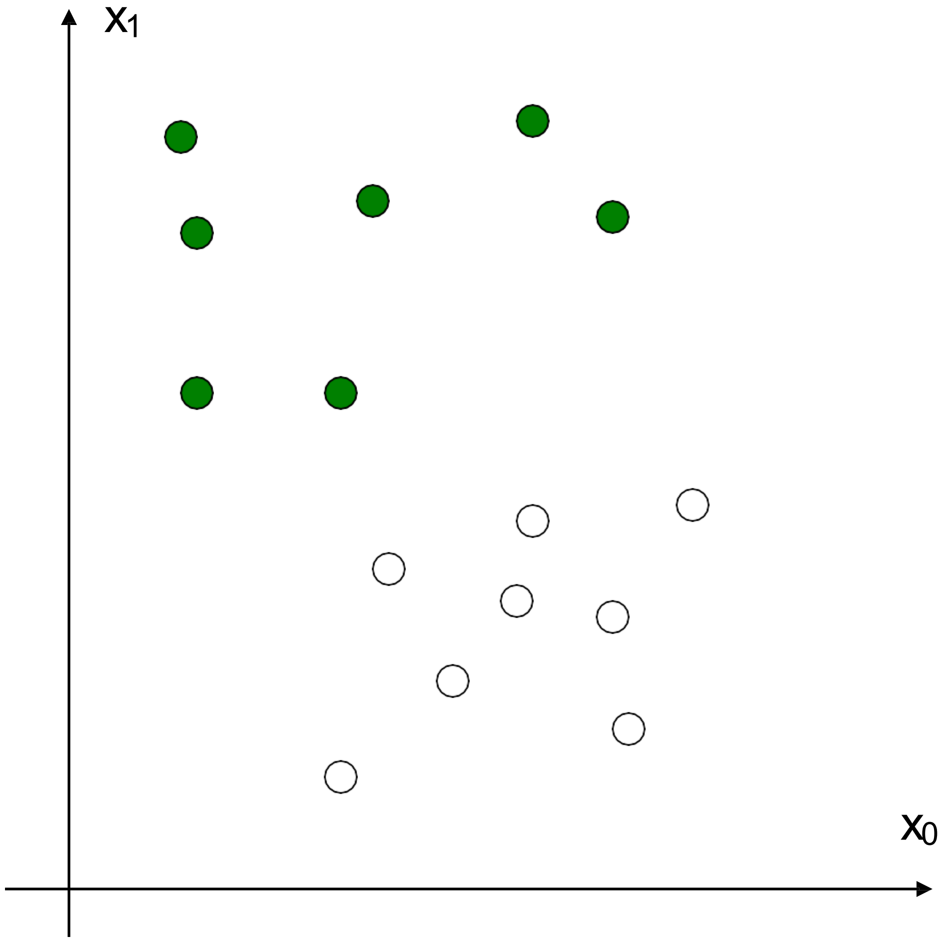
- Decision Trees

Strength in Unity

- Random Forests
- Gradient Boosting

A familiar picture

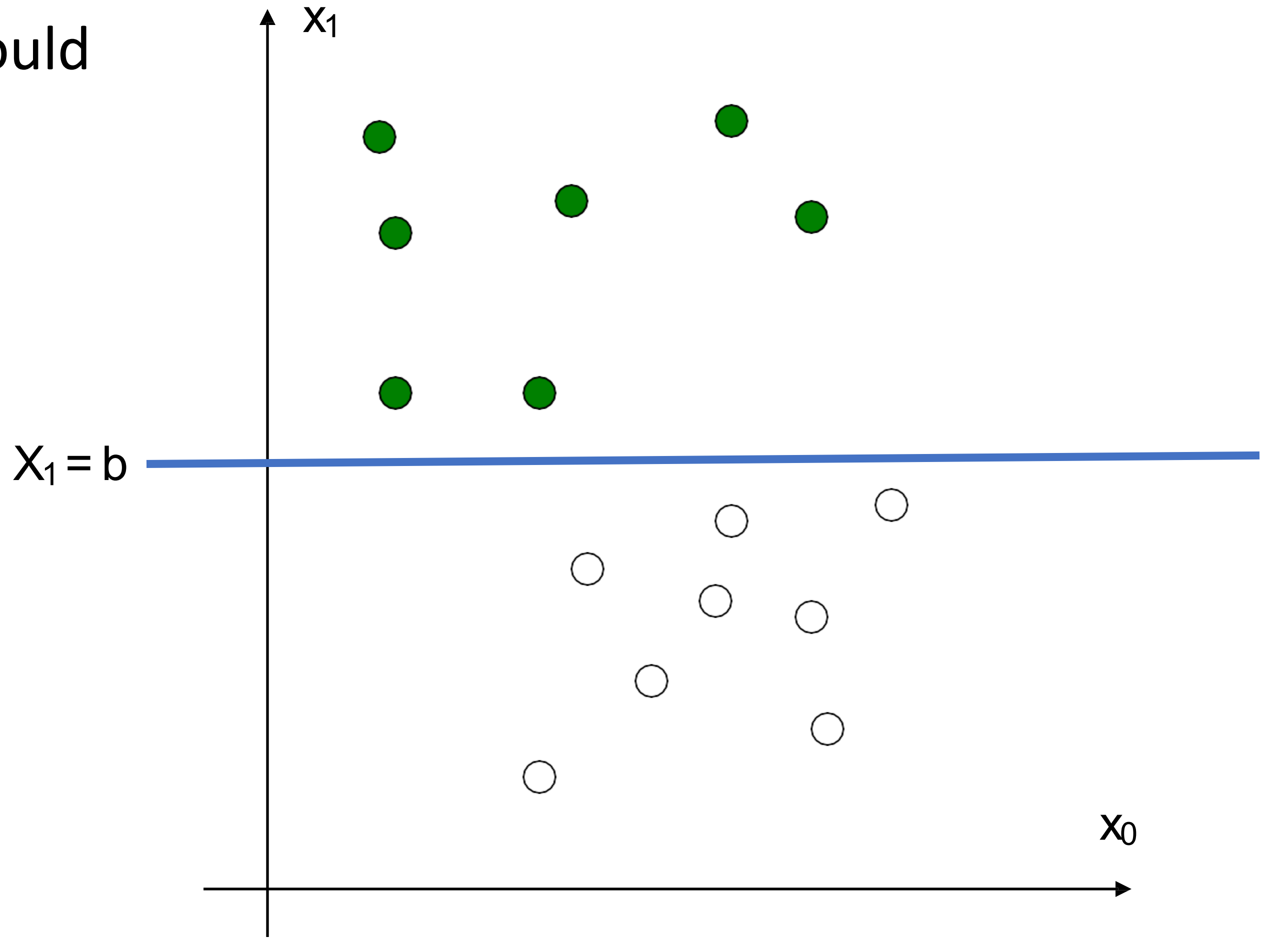
Based on **features** alone, how would you classify these points?



Decision points

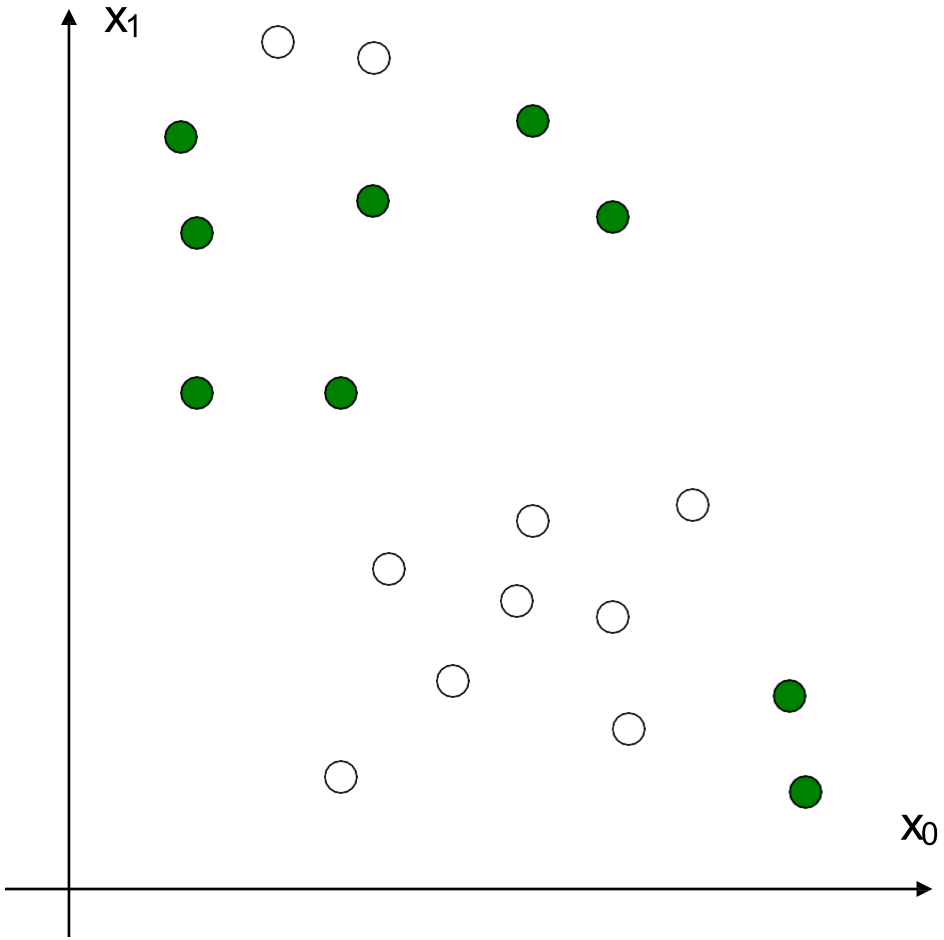
Based on **features** alone, how would you classify these points?

- If $X_1 > b$, then green
- If $X_1 < b$, then white



A slight complication

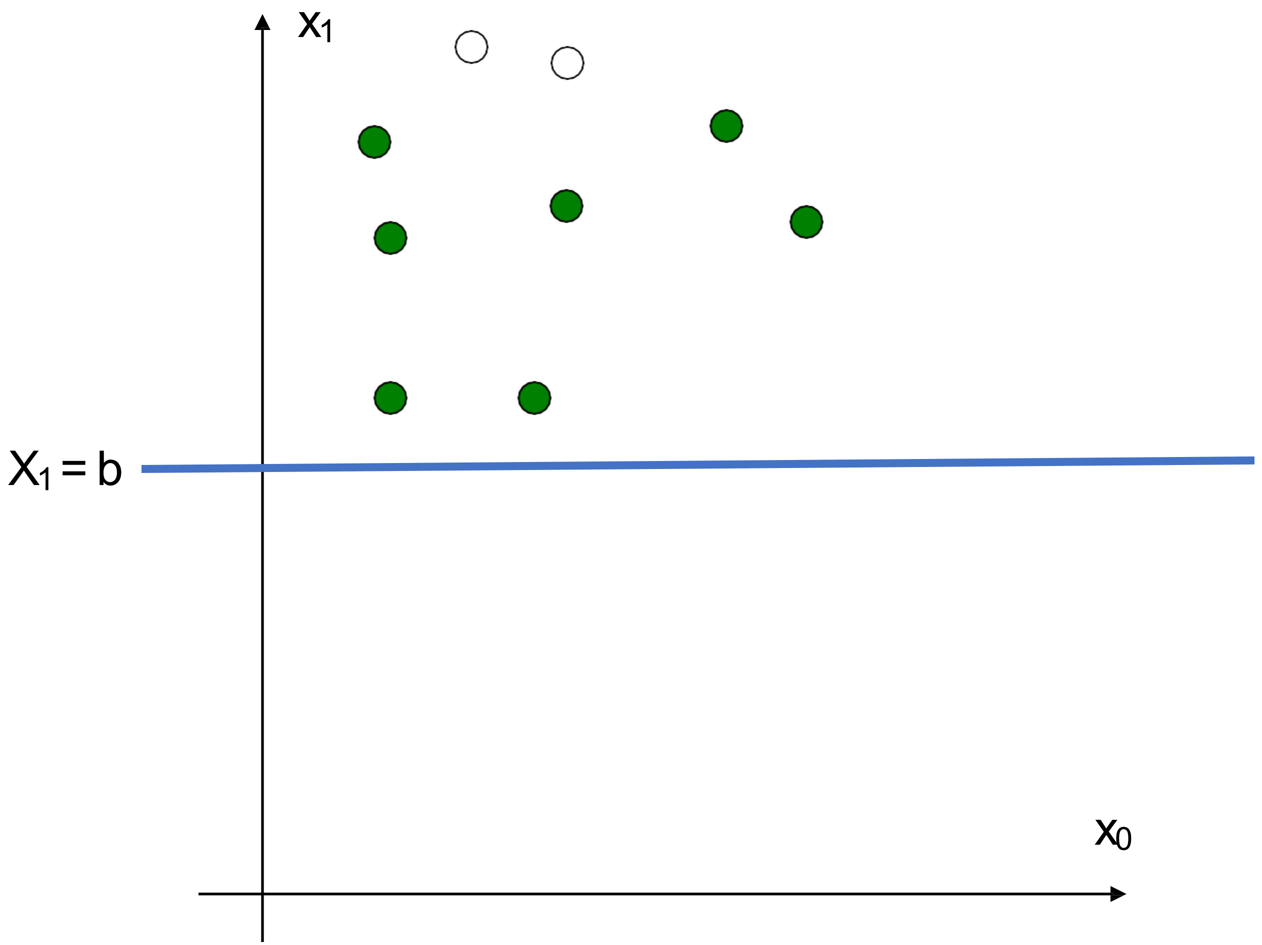
Based on **features** alone, how would you classify these points?



Recursion: Divide

Step 1 - Divide:

- If $X_1 > b$, then go to Step 2
- If $X_1 < b$, then go to Step 3



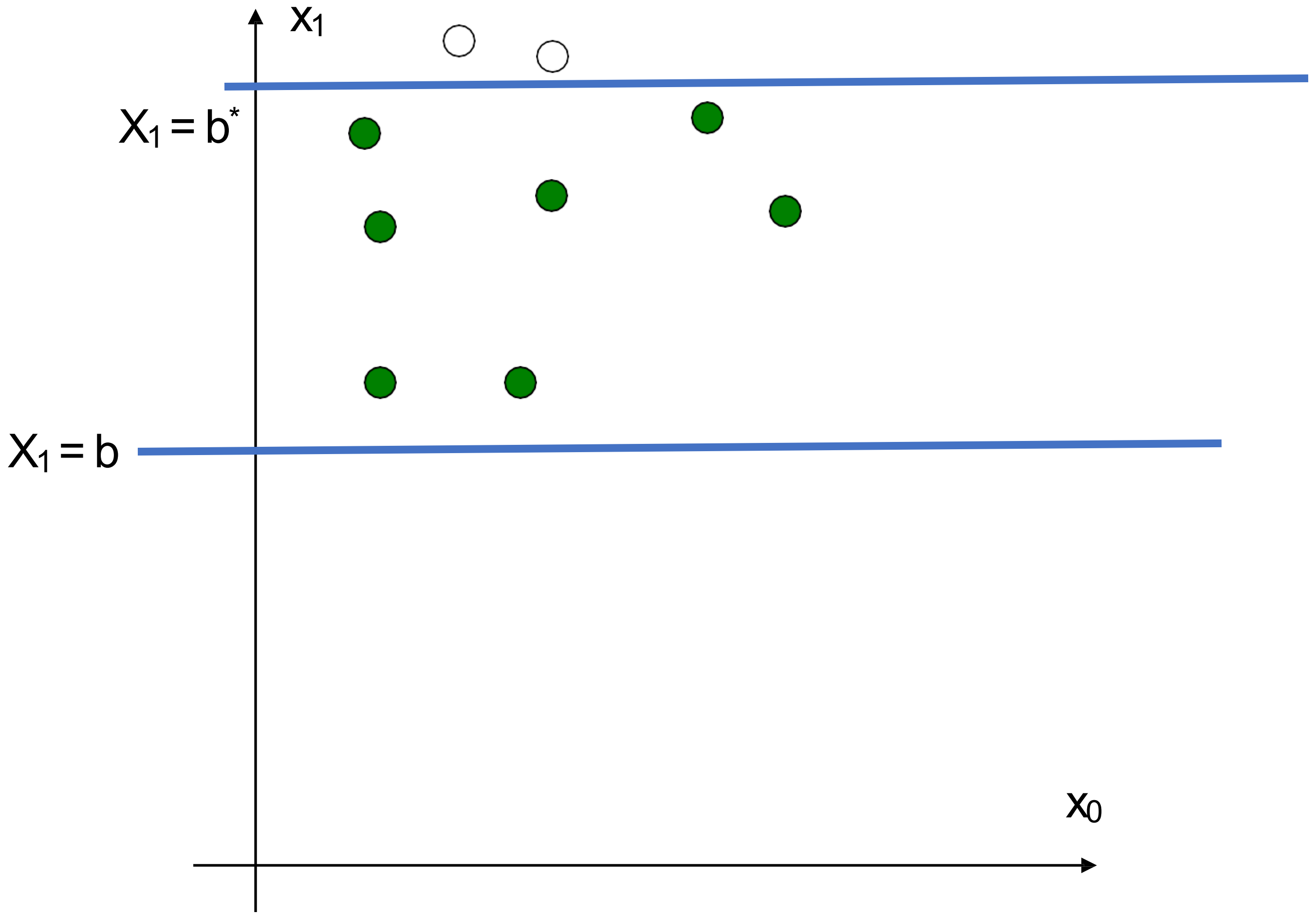
Recursion: Divide and Conquer

Step 1 – Divide:

- If $X_1 > b$, then go to Step 2
- If $X_1 < b$, then go to Step 3

Step 2

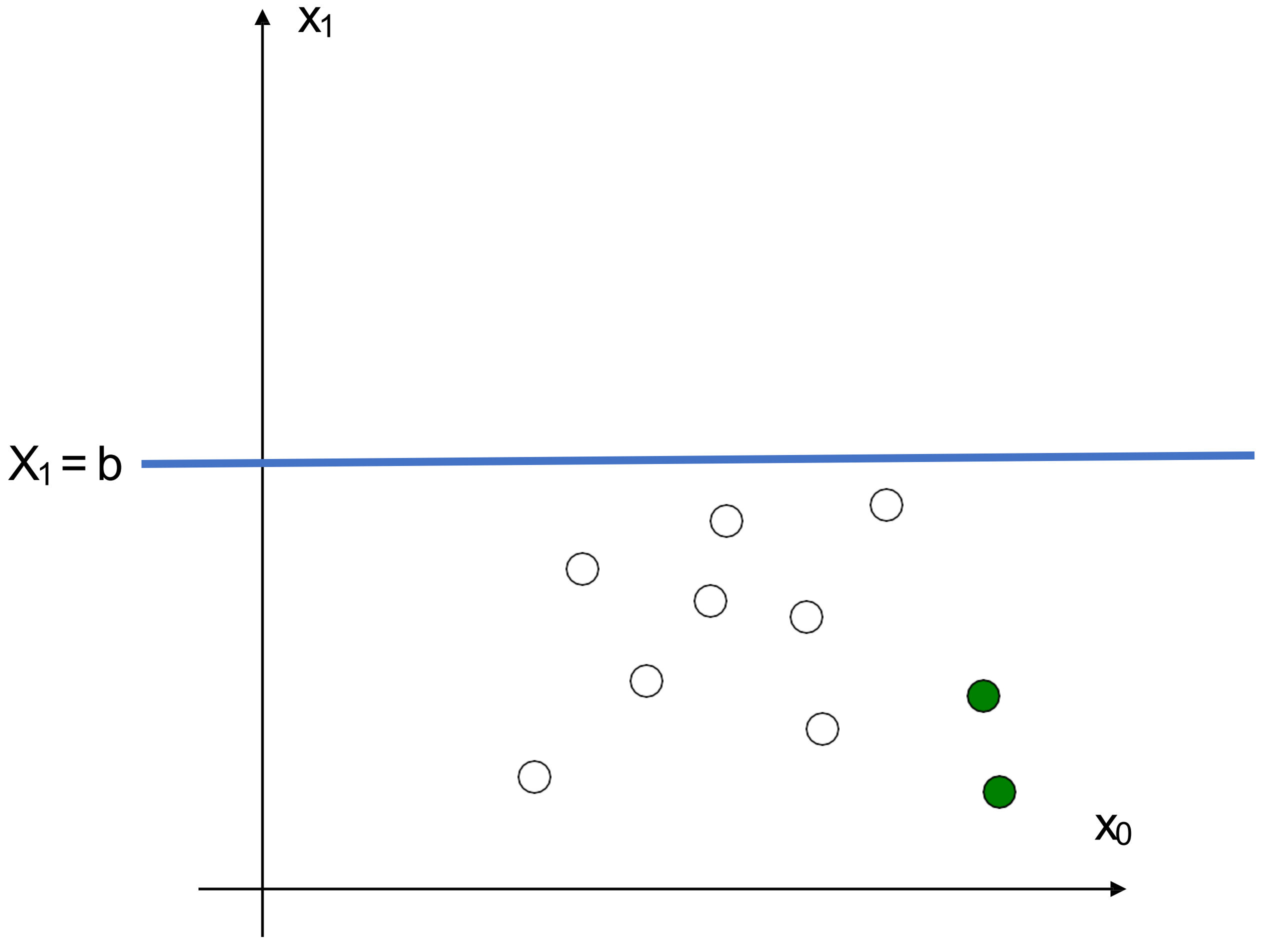
- If $X_1 > b^*$, then white
- If $X_1 < b^*$, then green



Recursion: Repeat as necessary

Step 1 - Divide:

- If $X_1 > b$, then go to Step 2
- If $X_1 < b$, then go to Step 3



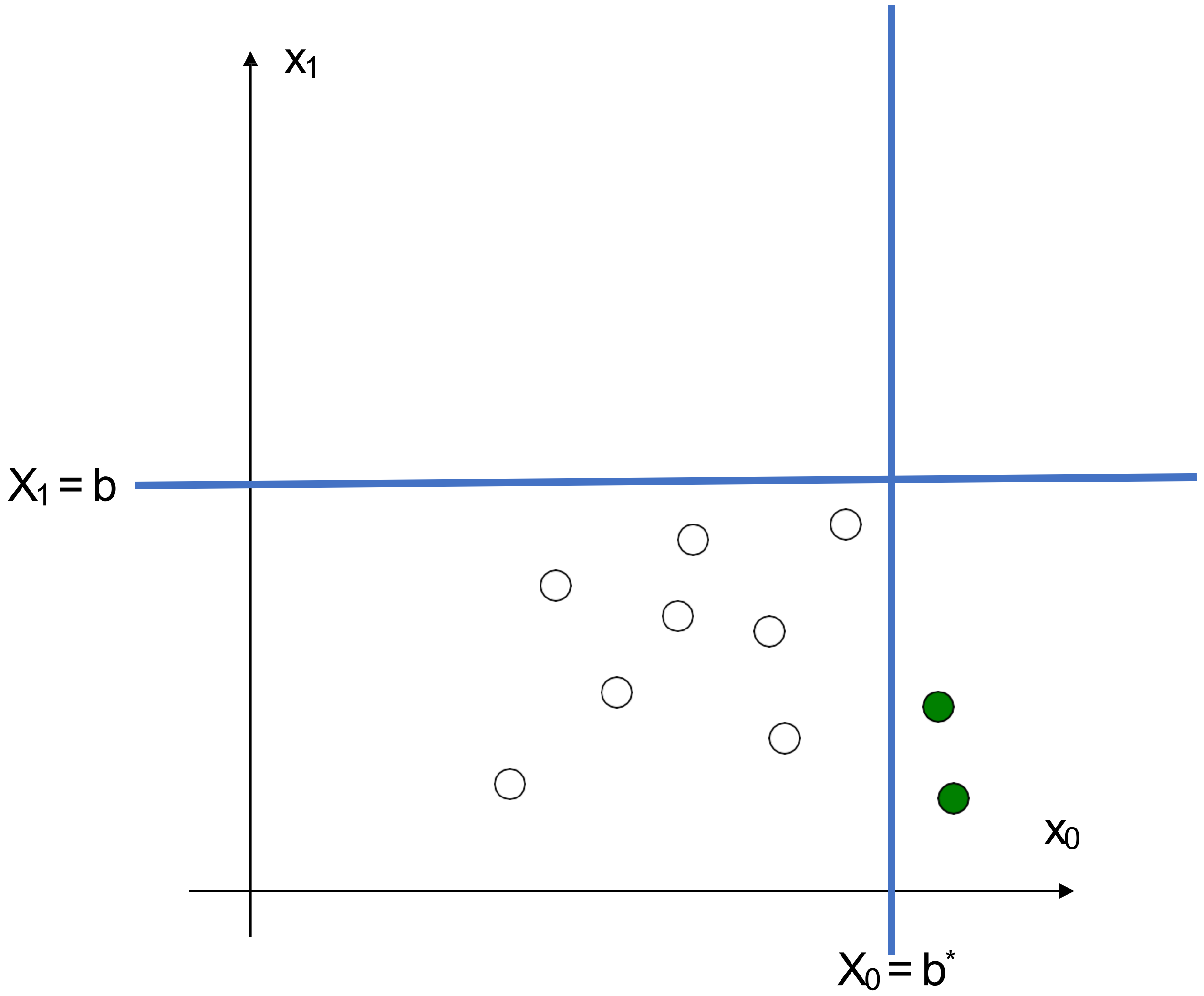
Recursion: Repeat as necessary

Step 1 – Divide:

- If $X_1 > b$, then go to Step 2
- If $X_1 < b$, then go to Step 3

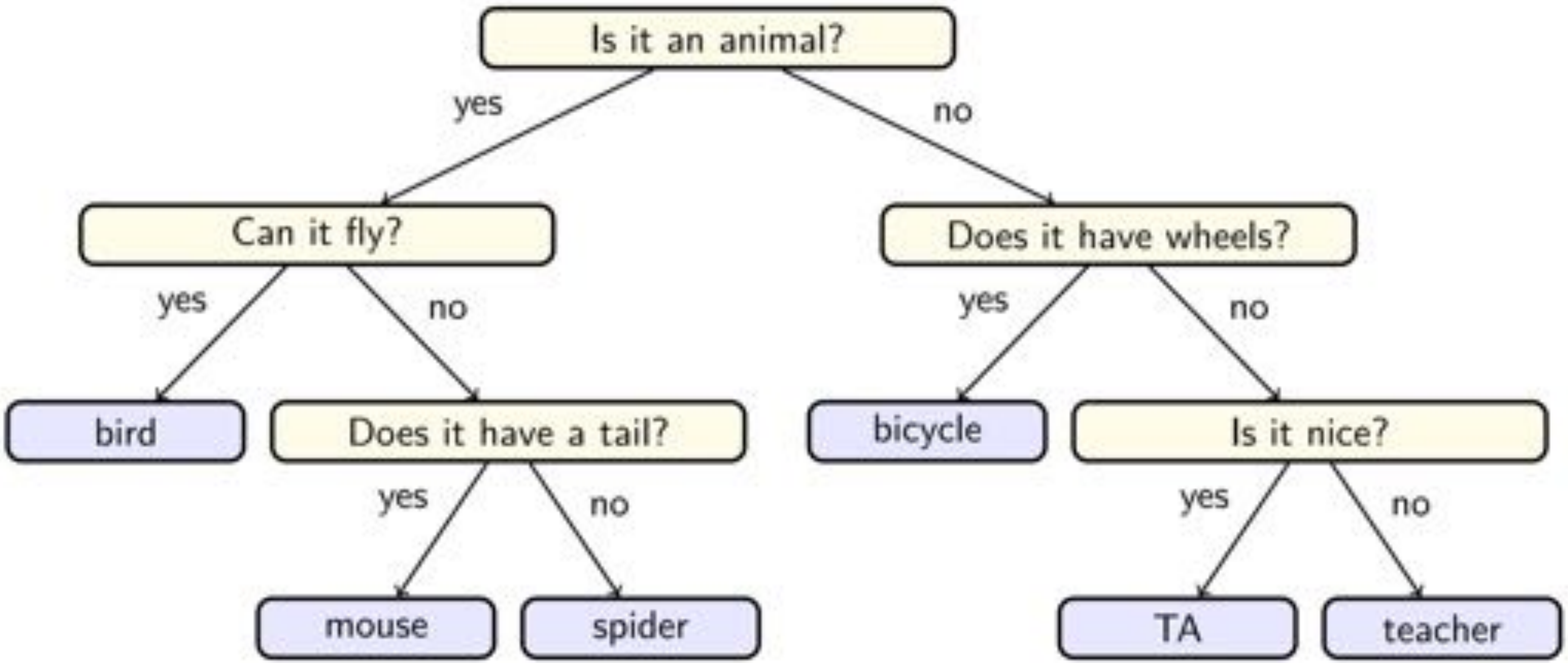
Step 3

- If $X_0 > b^*$, then green
- If $X_0 < b^*$, then white

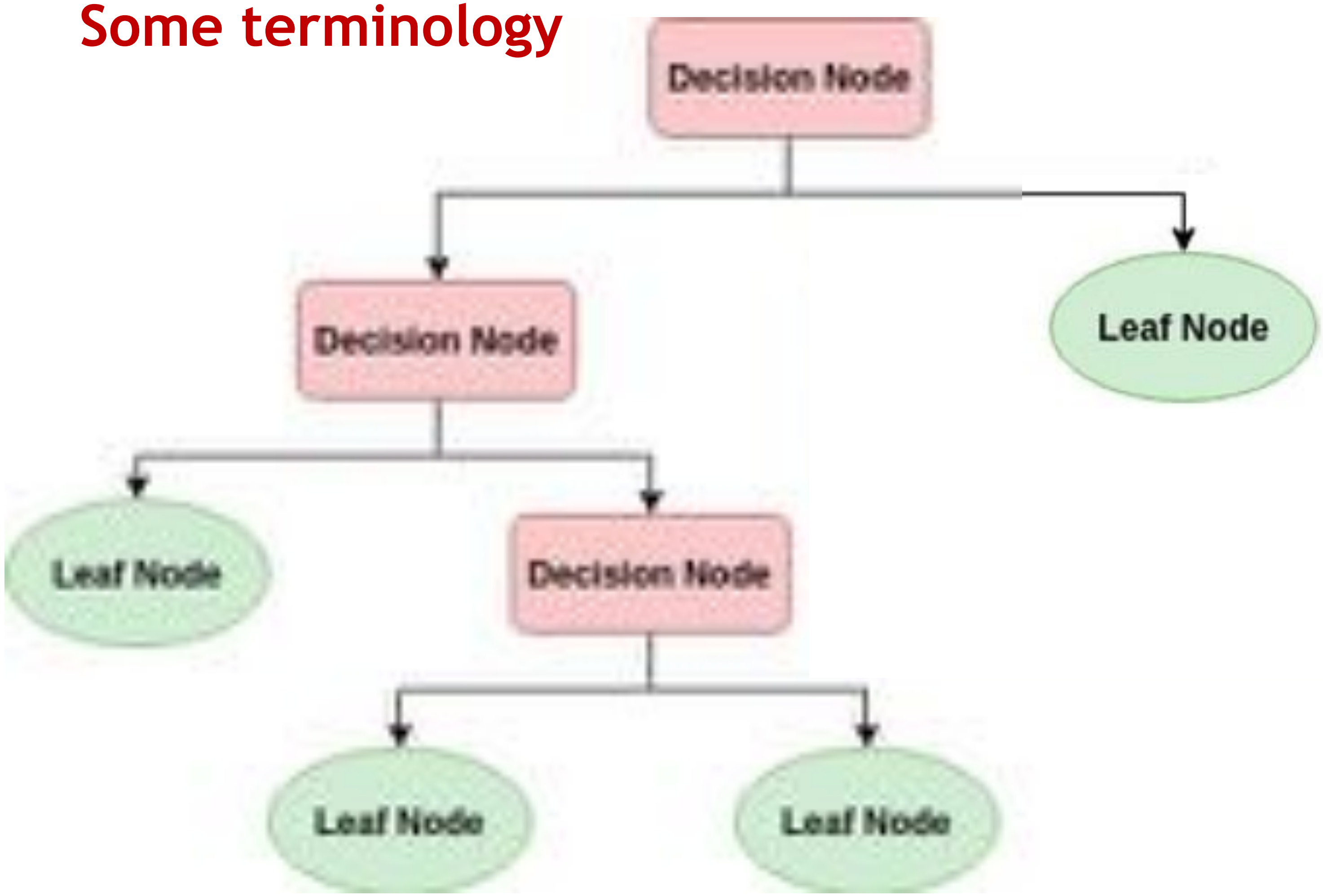


Decision Trees

Twenty Questions - AKA animal, vegetable, or mineral



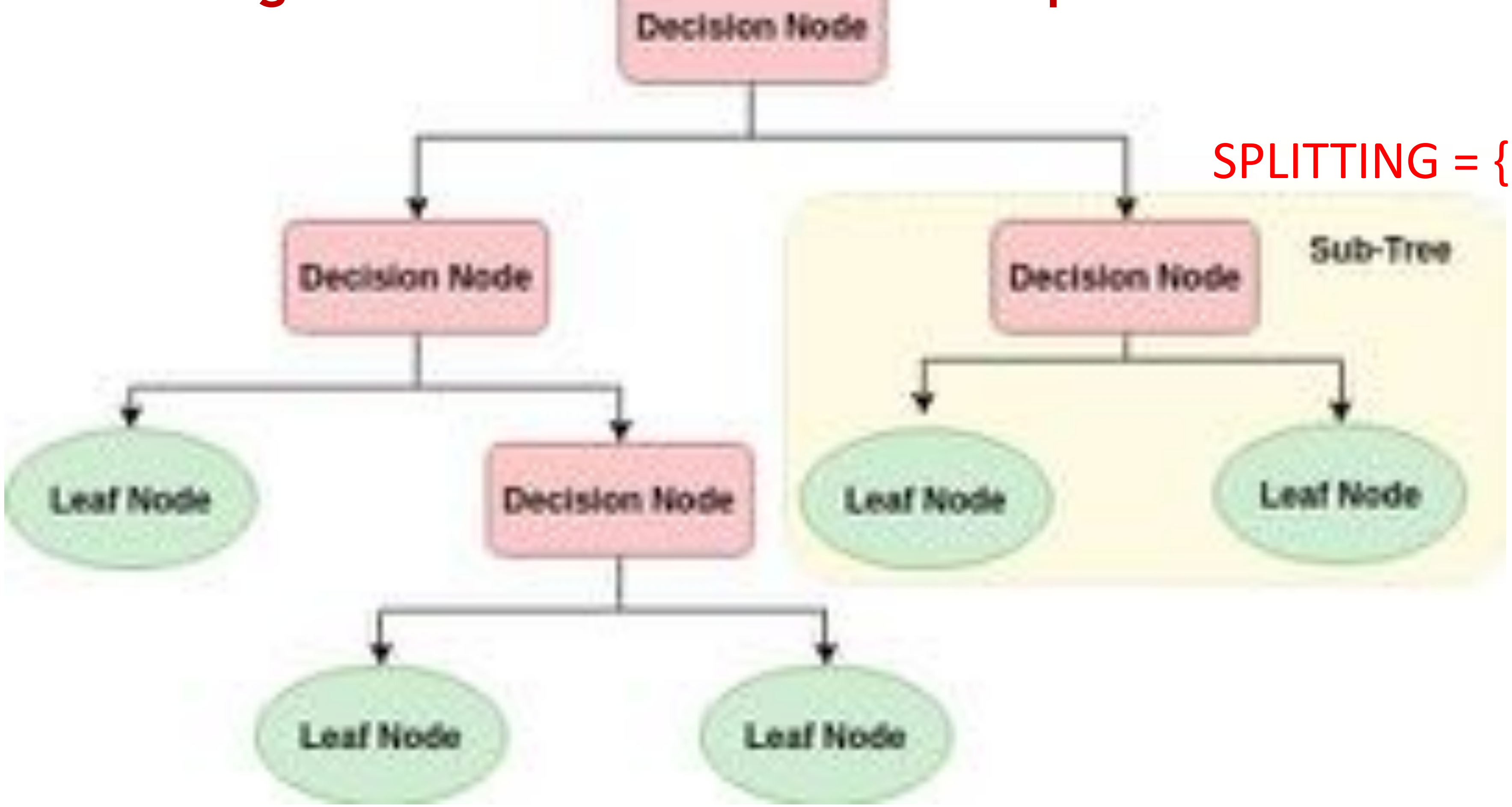
Some terminology



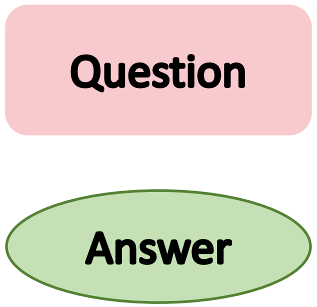
Question

Answer

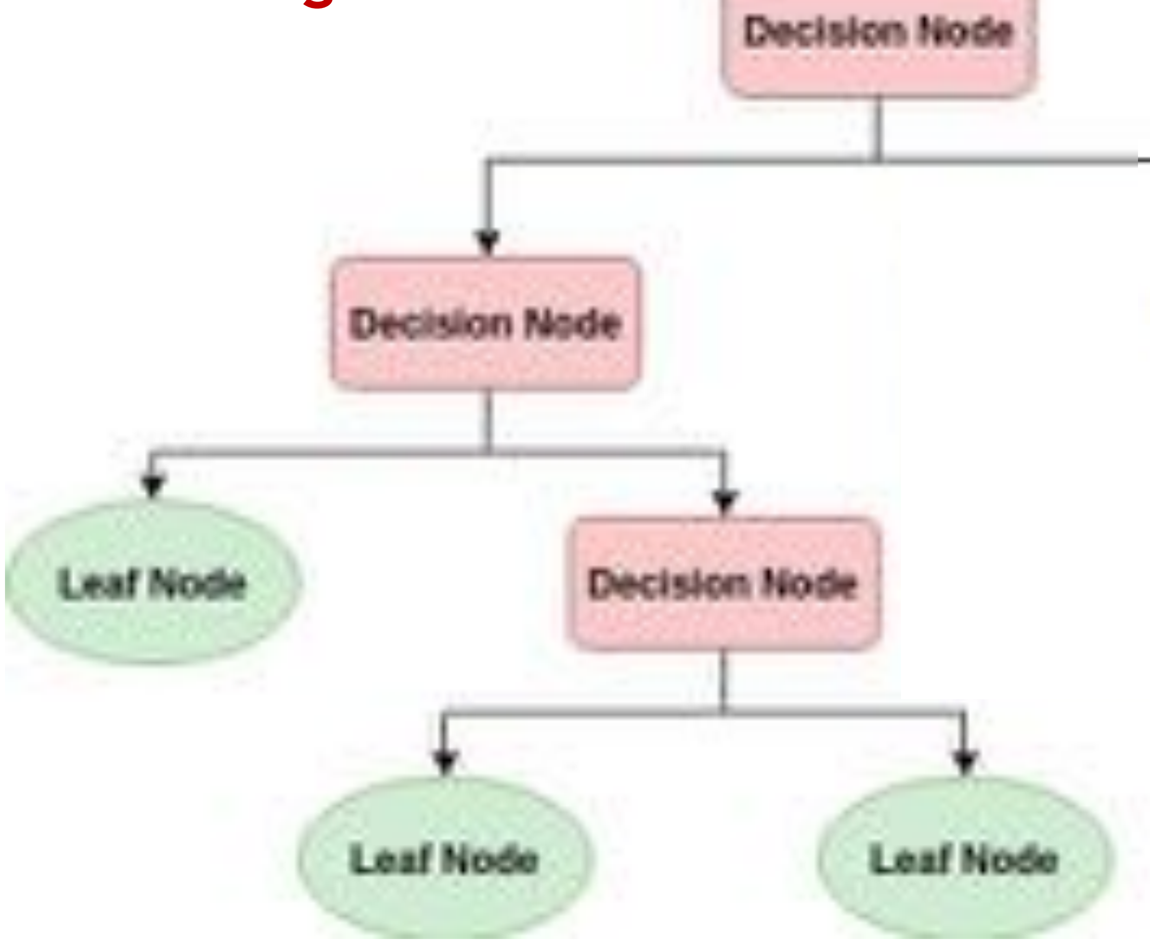
Growing trees is a recursive/iterative process



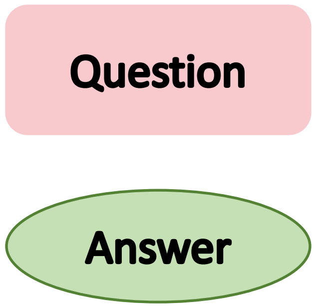
SPLITTING = {YES: go left
NO: go right



Growing trees is a recursive/iterative process



PRUNING
Collapse two child nodes into one parent node



Interactive visualization of the main idea

R2
D3

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

A visual
introduction to
machine learning

How do you choose the questions?

Picking a splitting rule

Candidate rules are chosen from the predictor variables
(the **max_features** option controls how many to consider at a time, and the **random_state** option controls ties)

```
For each candidate rule:
 Split the tree according to the rule
 For each leaf node (data subset):
     Compute how “impure” the leaf node is
 Compute the “average” impurity for all leaf nodes
Select the candidate rule that results in a split that is
“closest” to “average” purity
```

How do you choose the threshold?

Are the leaves pure?

We want the leaves, on average, to be **as close to** pure as possible (for high accuracy)

The **criterion** option determines what “impure” means

- Gini Impurity (CART)
- Entropy Decrease / Information Gain (ID3)
- Entropy Gain Ratio (C4.5)
- Chi-square (CHAID)

Are the leaves pure?

The **criterion** option determines what “impure” means

- Gini Impurity (CART):

  Similar to the Gini coefficient for income inequality

- Entropy Decrease / Information Gain (ID3):

  Entropy depends on the number of wrong labels per variable,

  so leaf=[5,5,0] is not the same as leaf=[5,5]

- Entropy Gain Ratio (C4.5):

  Normalizes entropy gain to account for # of labels

- Chi-square (CHAID):

  Allows more than yes/no (multiway) splits, so needs more data

Hands-on Example:

Model tuning

Is this all there is? Are we done?

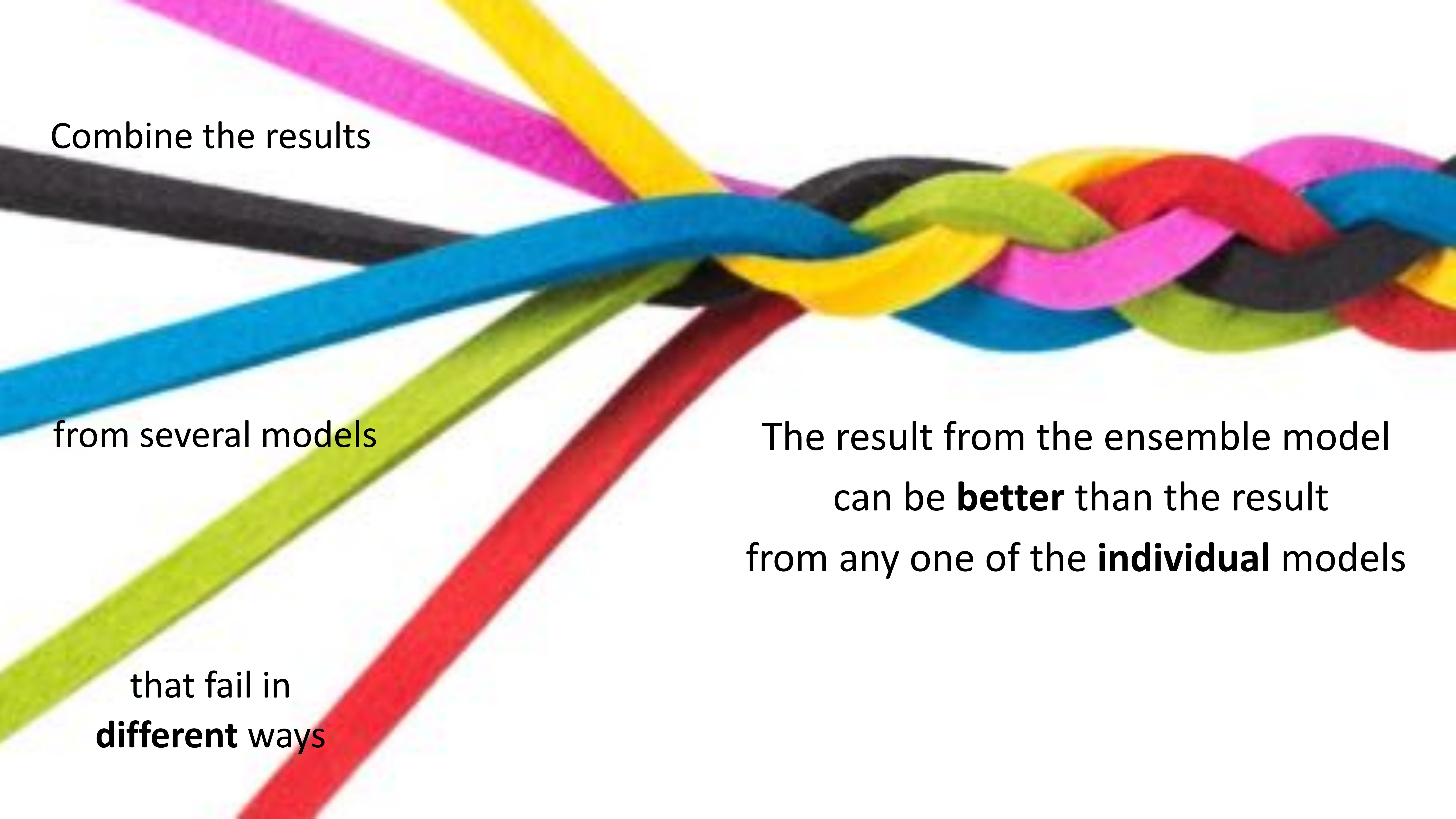
Potential Disadvantages of Decision Trees

- Imbalanced data sets can bias results

  If we have a majority class present, the top of the decision tree is likely to learn splits which separate out the majority class into pure groups at the expense of learning rules which separate the minority class

- Small changes to data points (noise) can lead to completely different branches/trees
- Overfitting

Ensemble Methods



Combine the results

from several models

that fail in
different ways

The result from the ensemble model
can be **better** than the result
from any one of the **individual** models

Ensemble Types

- Bagging (Bootstrap AGGregating)
  - Random Forest
  - Voting
- Boosting
  - Adaptive Boosting (AdaBoost)
  - Gradient Boosting (HistGradientBoosting, XGBoost)

Bagging methods: prediction by committee

- Bootstrap: Build several instances of an estimator (tree) on **random subsets** of the training set and features.
- Aggregate: **Average** over the individual predictions to form a combined prediction
- The randomness should yield estimators with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out in the aggregate.

Boosting methods: progressively learn from mistakes

- Train the first component estimator (tree) on the training set (X_i , y_i)
- Boost: Train a new component estimator to focus on **the mistakes** (X_i , $error_i$) of the boosted ensemble computed so far
- Gradient: Add the new component estimator to the boosted ensemble computed so far

  $$Boosted_{i+1} = Boosted_i + \gamma_i\, estimator_i$$

  (γ is computed via error gradient optimization techniques)
- Repeat

Complementary approaches

- Bagging methods usually work best with strong and complex models

  e.g., fully developed (tall) decision trees

- Boosting methods usually work best with weak models

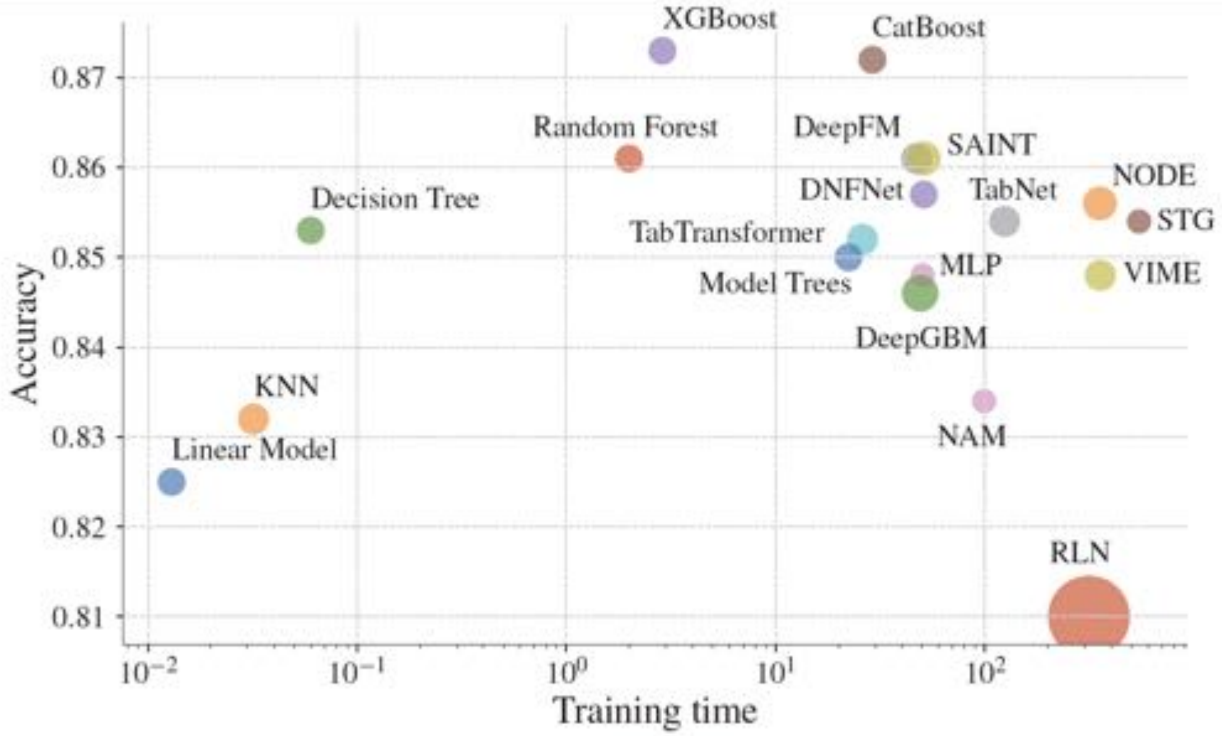
  e.g., shallow decision trees (stumps)

Hands-on Example: Ensemble

Hewlett Packard Enterprise
Data Science Institute
UNIVERSITY OF HOUSTON

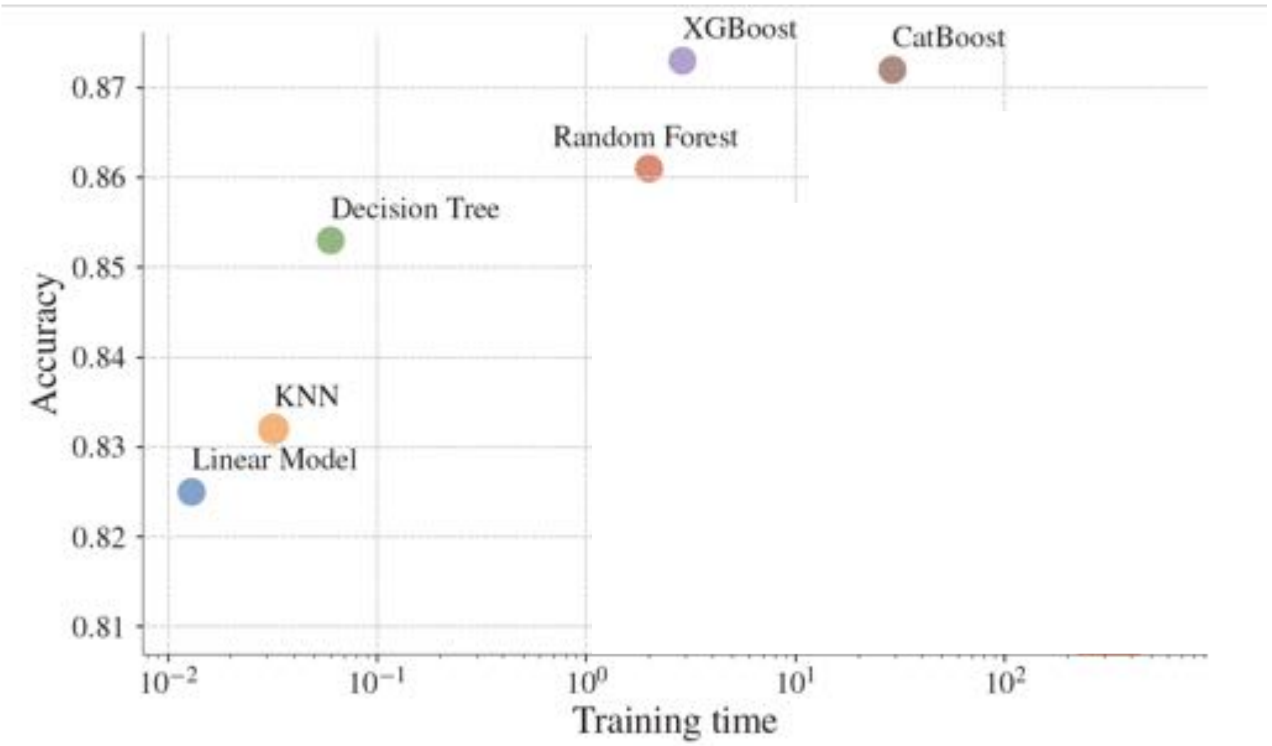
Deep Neural Networks and Tabular Data: A Survey

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug,
Martin Pawelczyk and Gjergji Kasneci

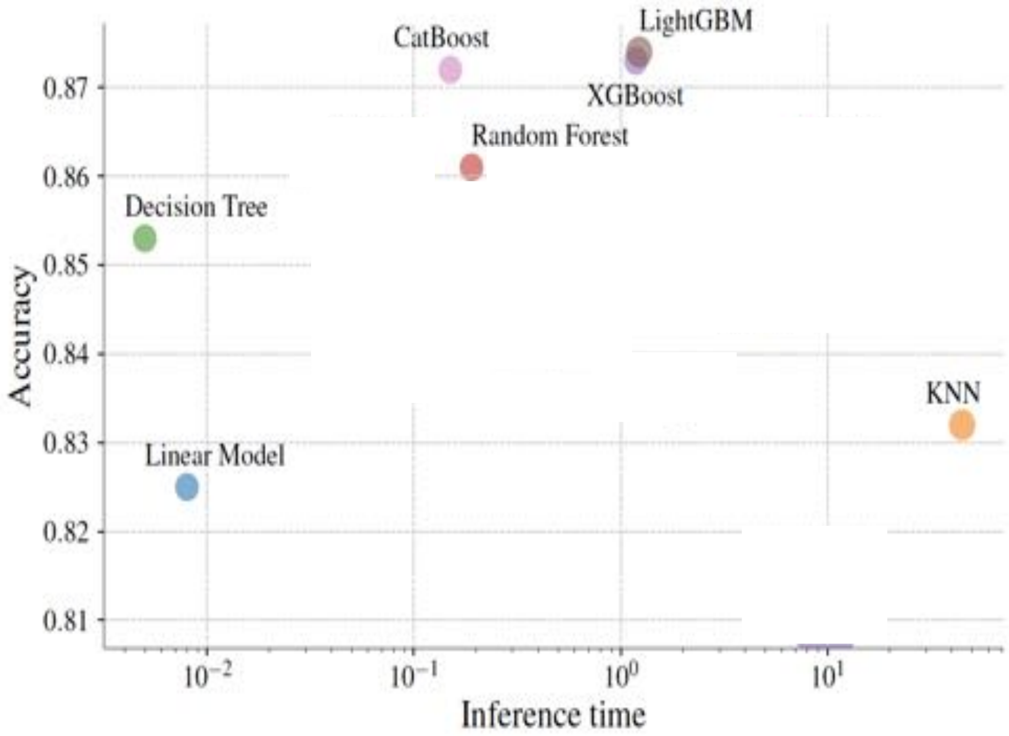


https://arxiv.org/pdf/2110.01889.pdf

Models we've seen so far: training time



Models we’ve seen so far: inference time



Training vs. inference time