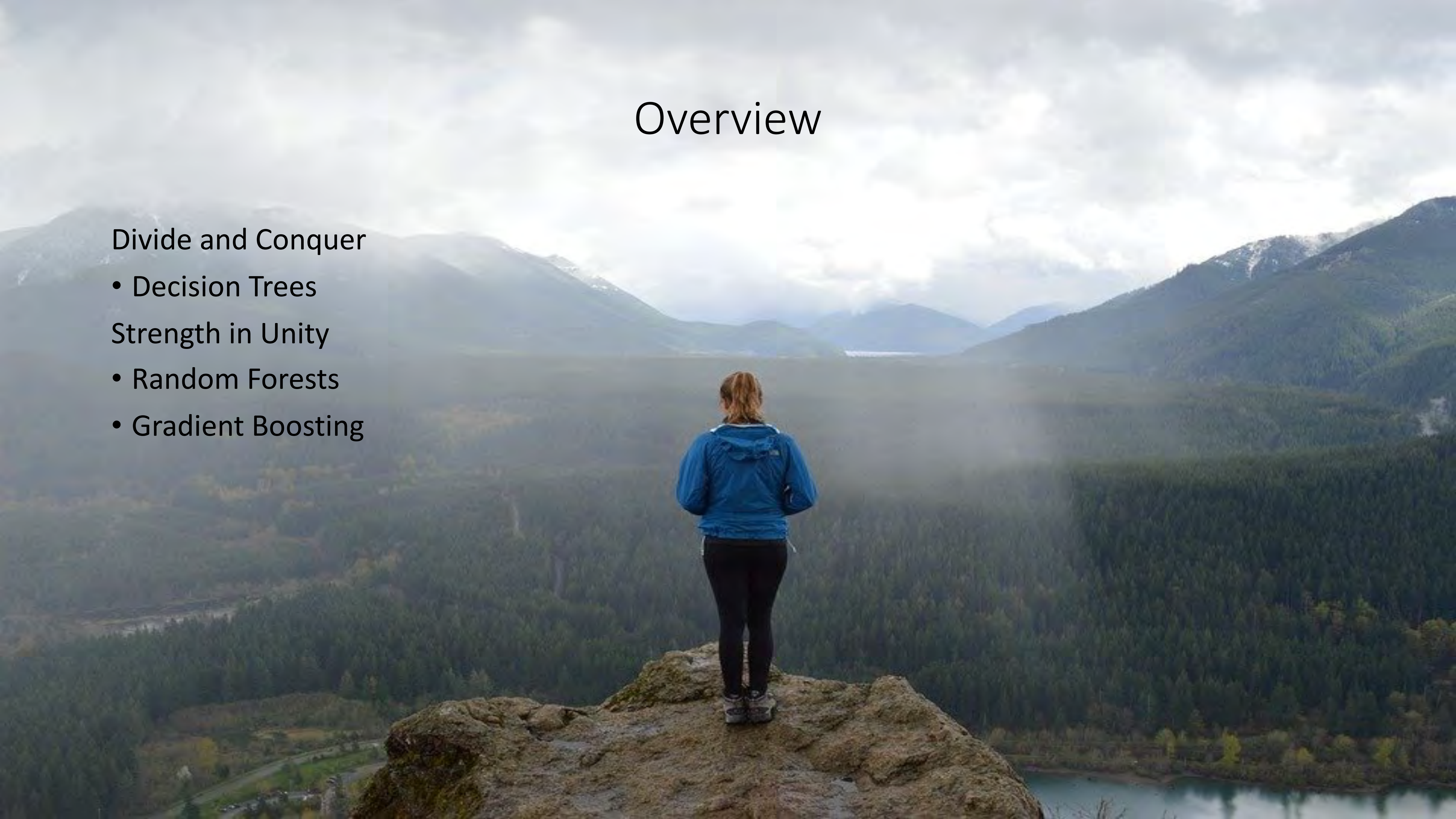
HPE DSI 311
Introduction to Machine Learning

Summer 2021

Instructor: Ioannis Konstantinidis

UNIVERSITY of HOUSTON
DIVISION OF RESEARCH
HEWLETT PACKARD ENTERPRISE DATA SCIENCE INSTITUTE

Overview



Divide and Conquer

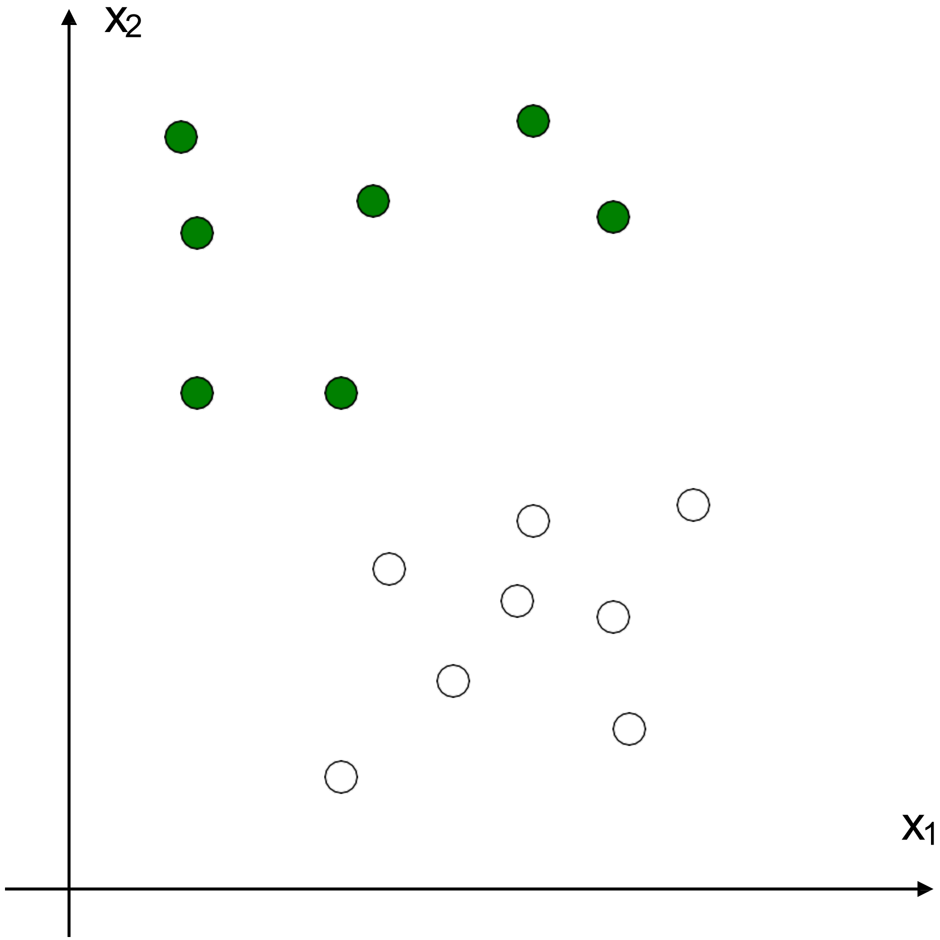
- Decision Trees

Strength in Unity

- Random Forests
- Gradient Boosting

A familiar picture

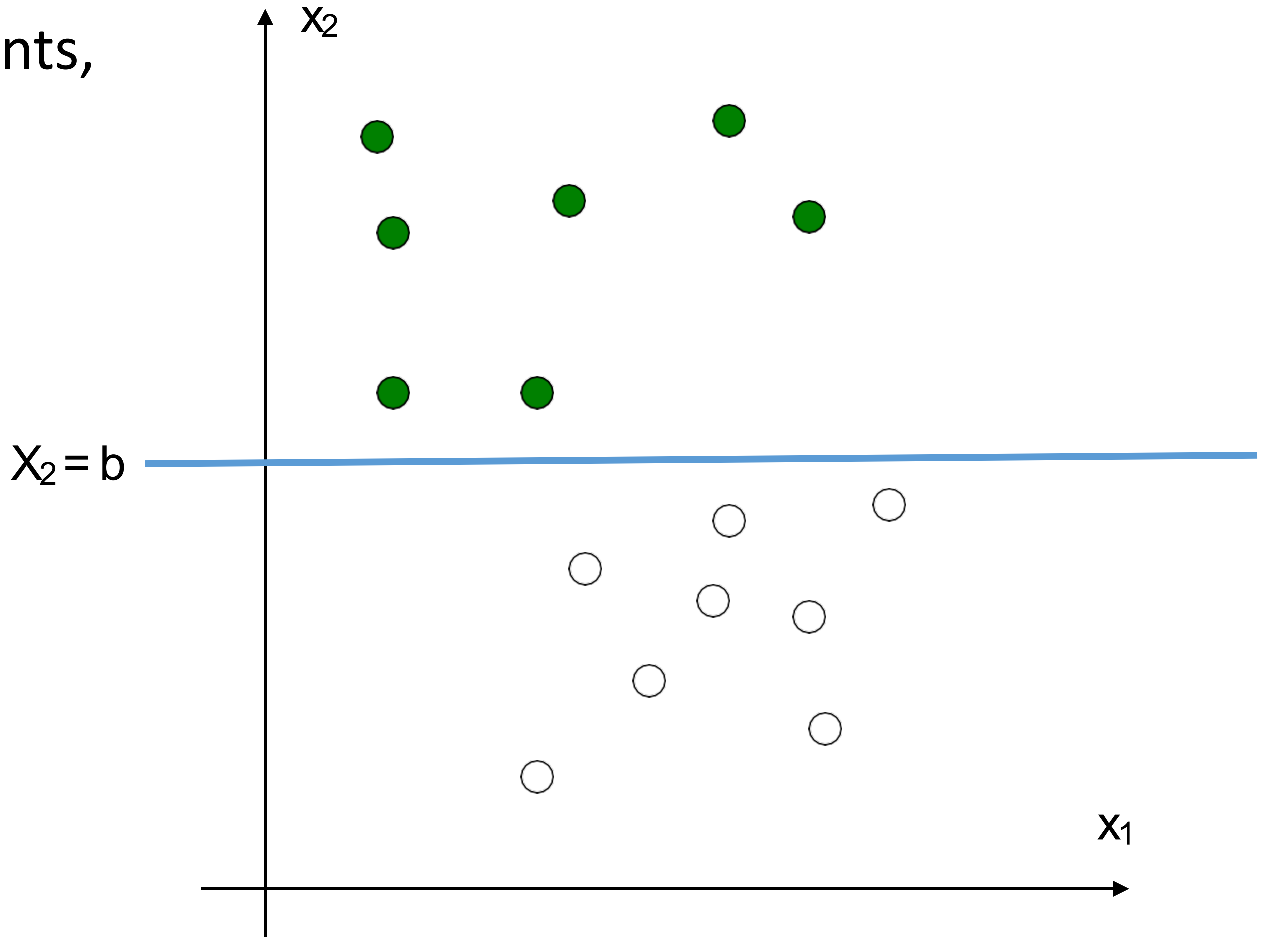
How would you classify these points, based on features alone?



Decision points

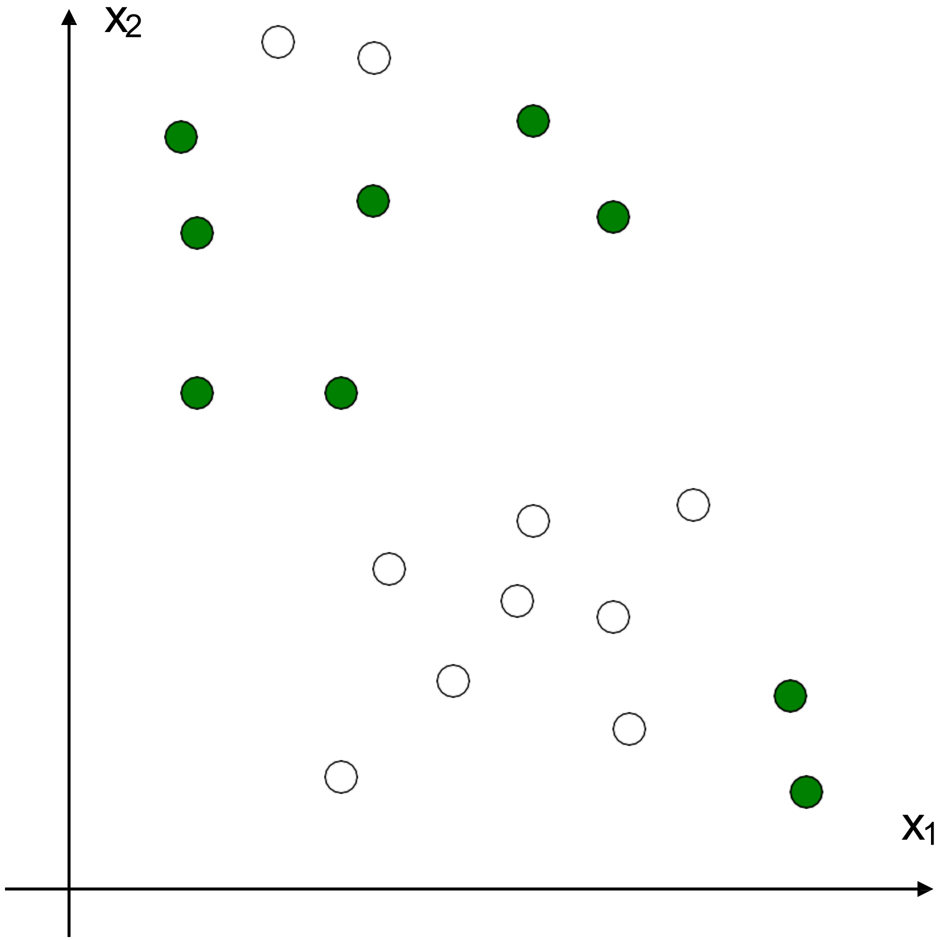
How would you classify these points, based on features alone?

- If $X_2 > b$, then green
- If $X_2 < b$, then white



A slight complication

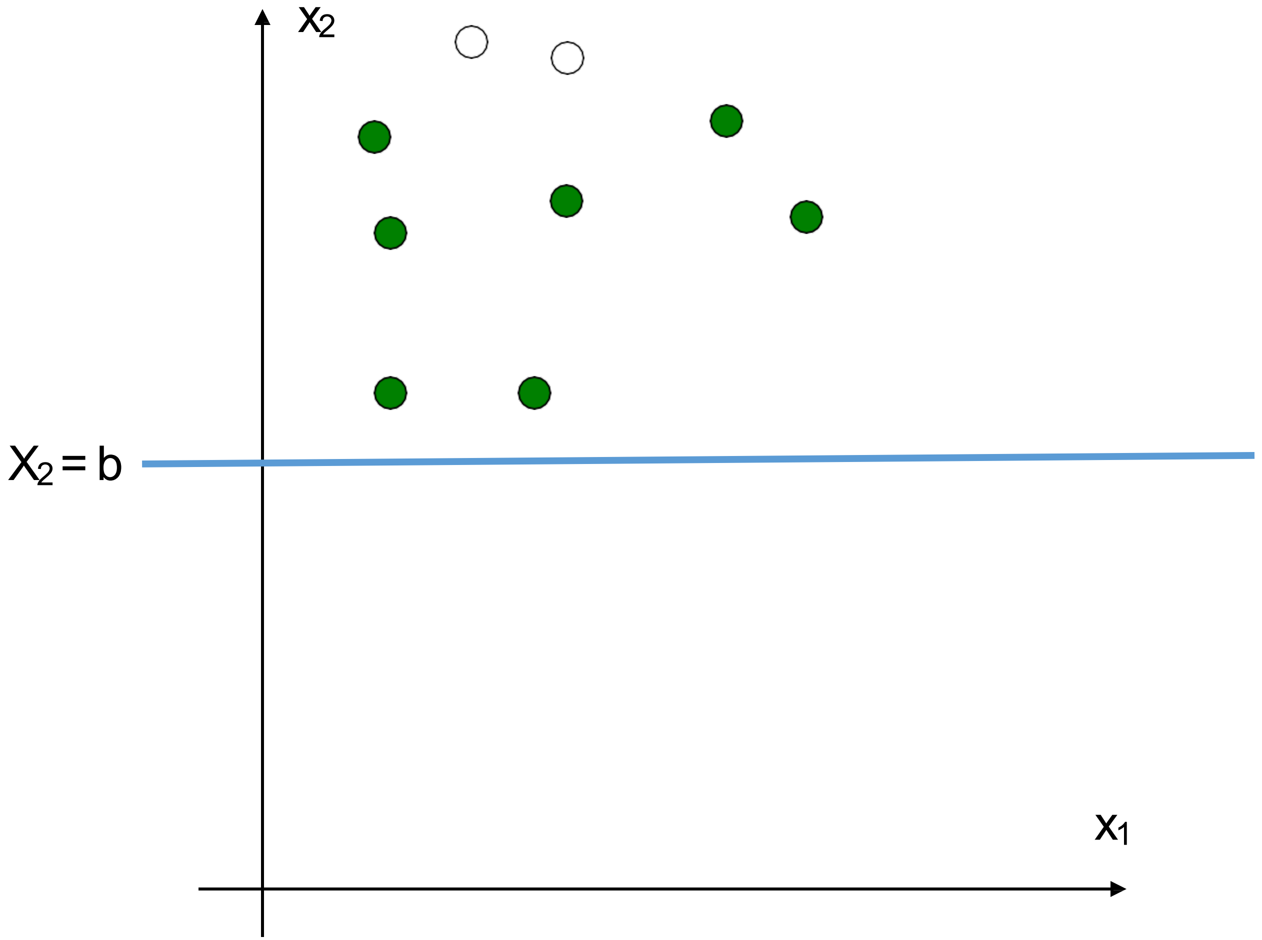
How would you classify these points, based on features alone?



Recursion: Divide

Step 1 - Divide:

- If $X_2 > b$, then go to Step 2
- If $X_2 < b$, then go to Step 3



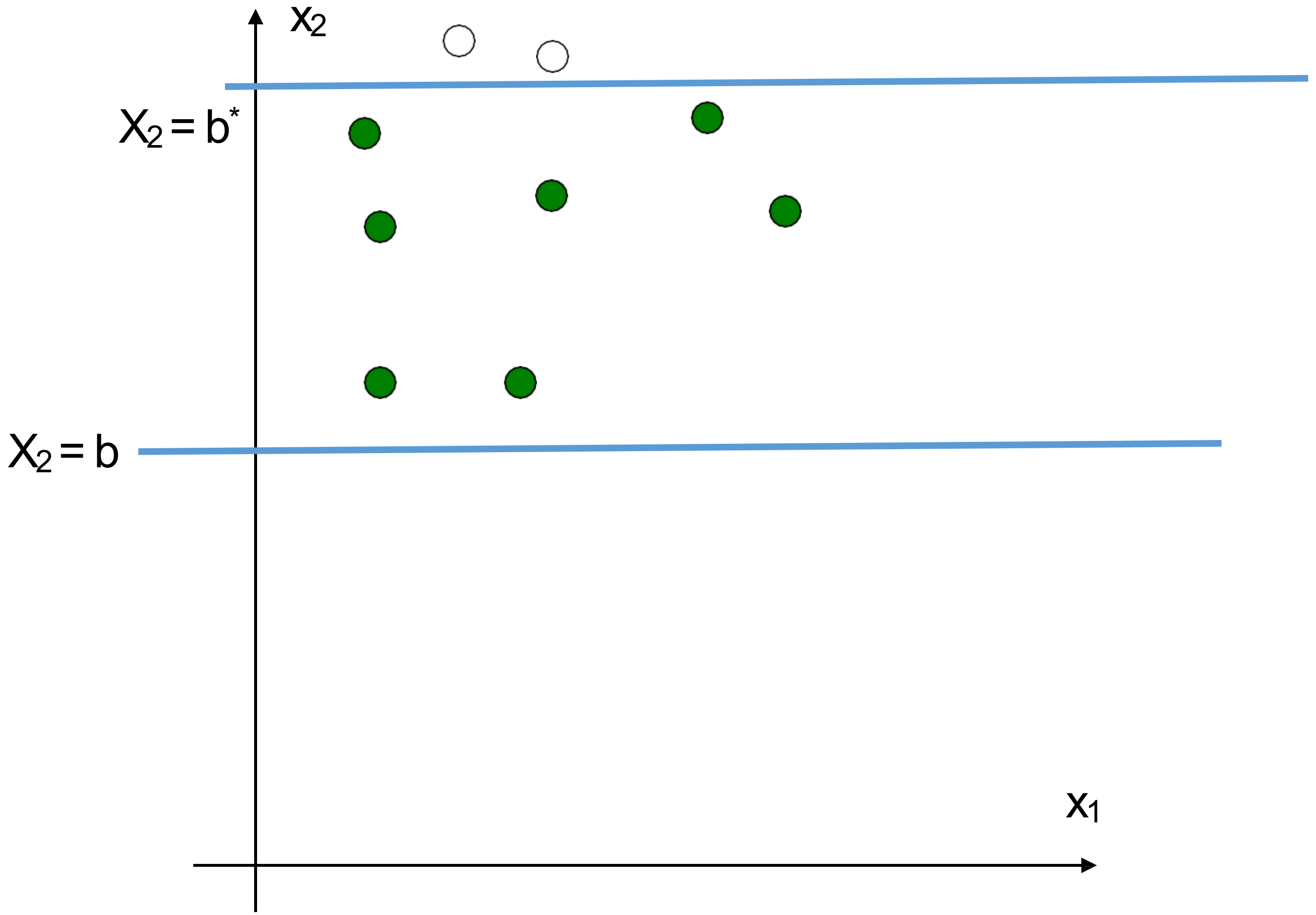
Recursion: Divide and Conquer

Step 1 – Divide:

- If $X_2 > b$, then go to Step 2
- If $X_2 < b$, then go to Step 3

Step 2

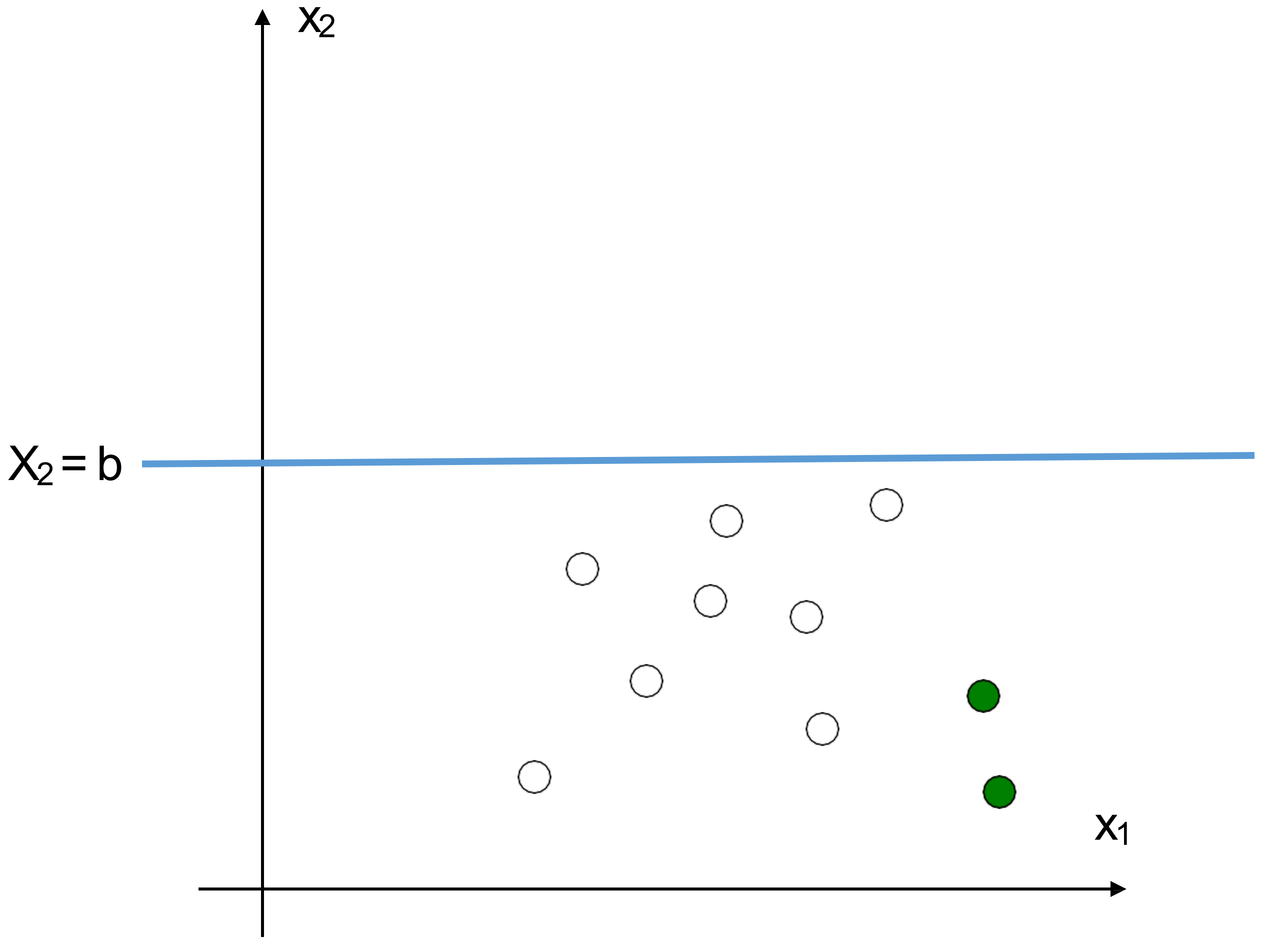
- If $X_2 > b^*$, then white
- If $X_2 < b^*$, then green



Recursion: Repeat as necessary

Step 1 - Divide:

- If $X_2 > b$, then go to Step 2
- If $X_2 < b$, then go to Step 3



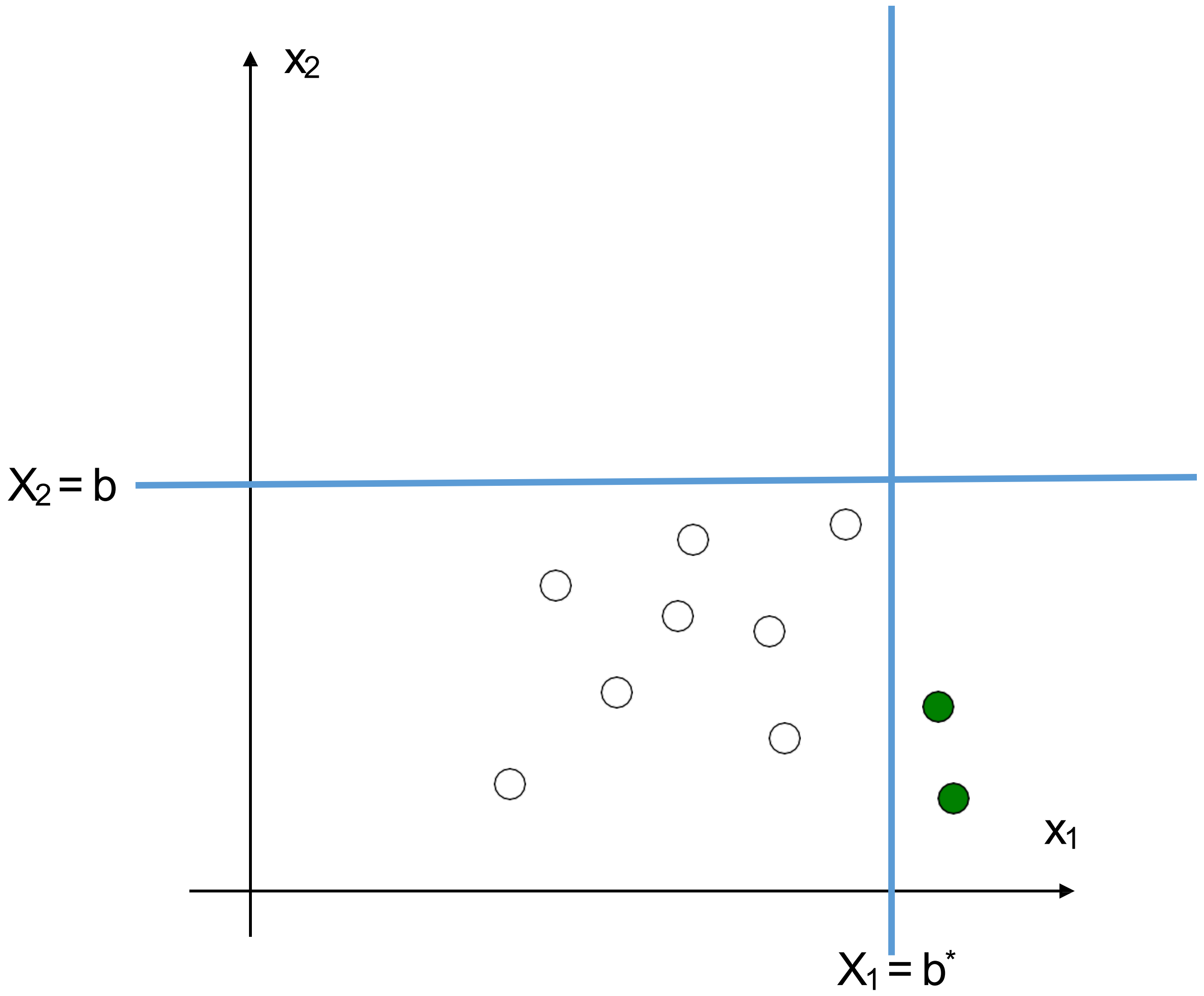
Recursion: Repeat as necessary

Step 1 – Divide:

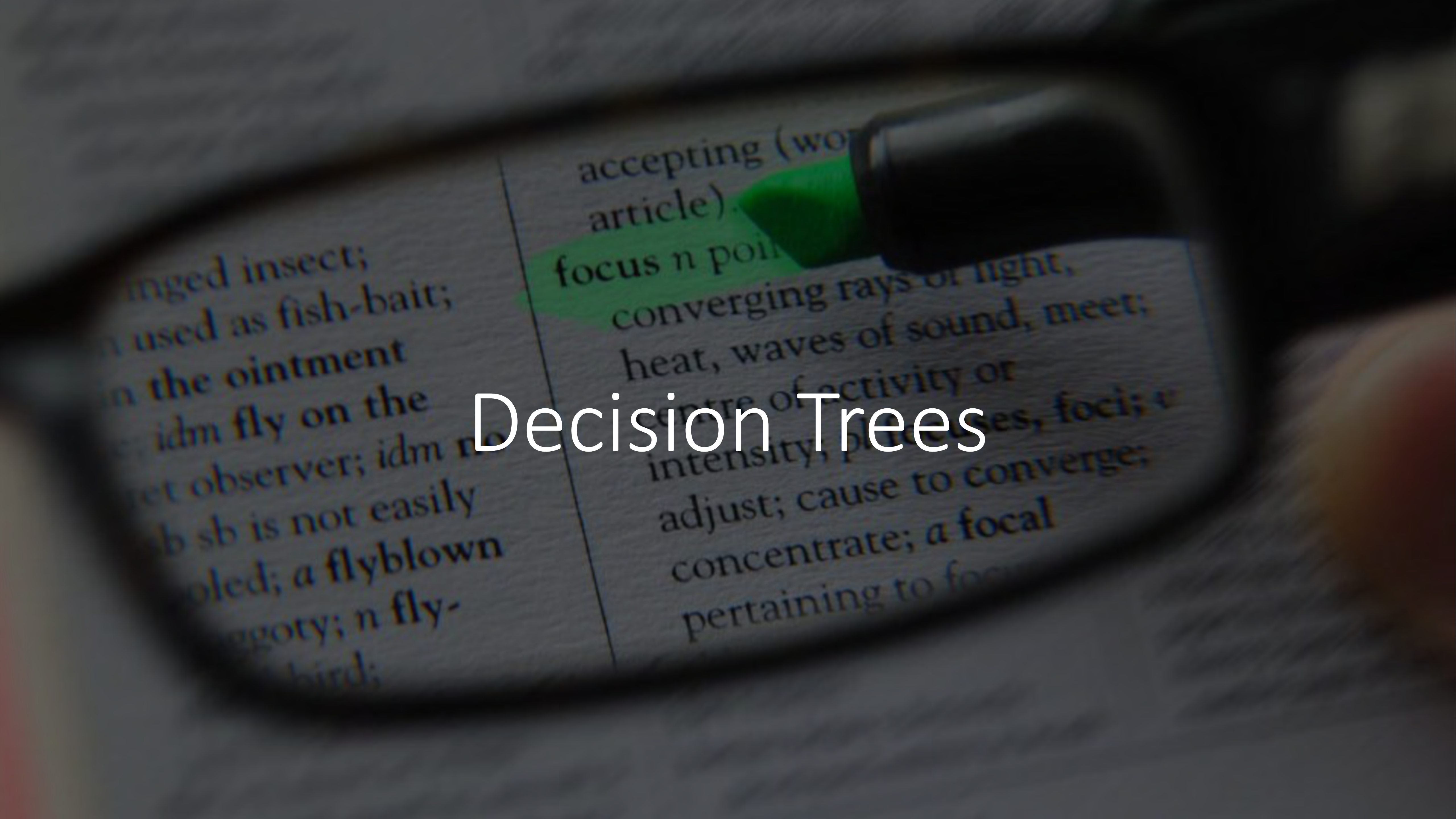
- If $X_2 > b$, then go to Step 2
- If $X_2 < b$, then go to Step 3

Step 3

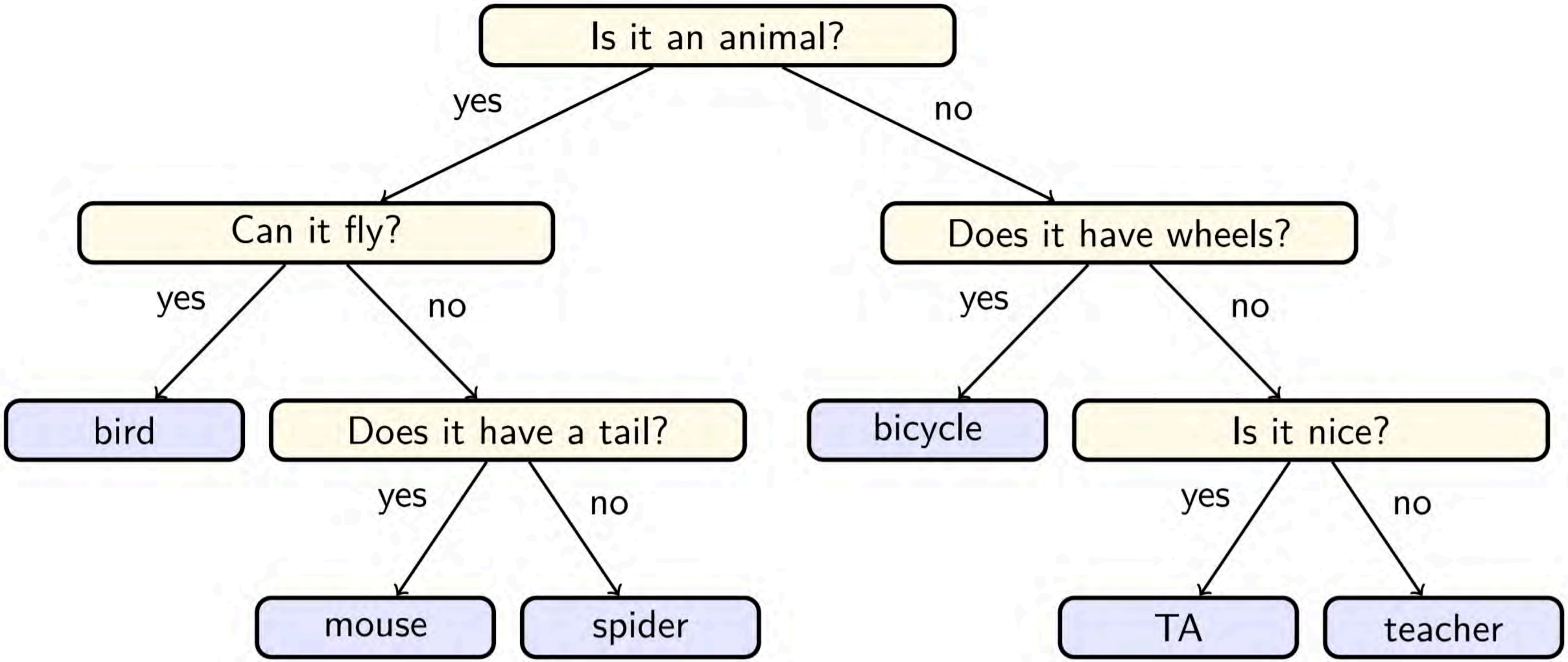
- If $X_1 > b^*$, then green
- If $X_1 < b^*$, then white



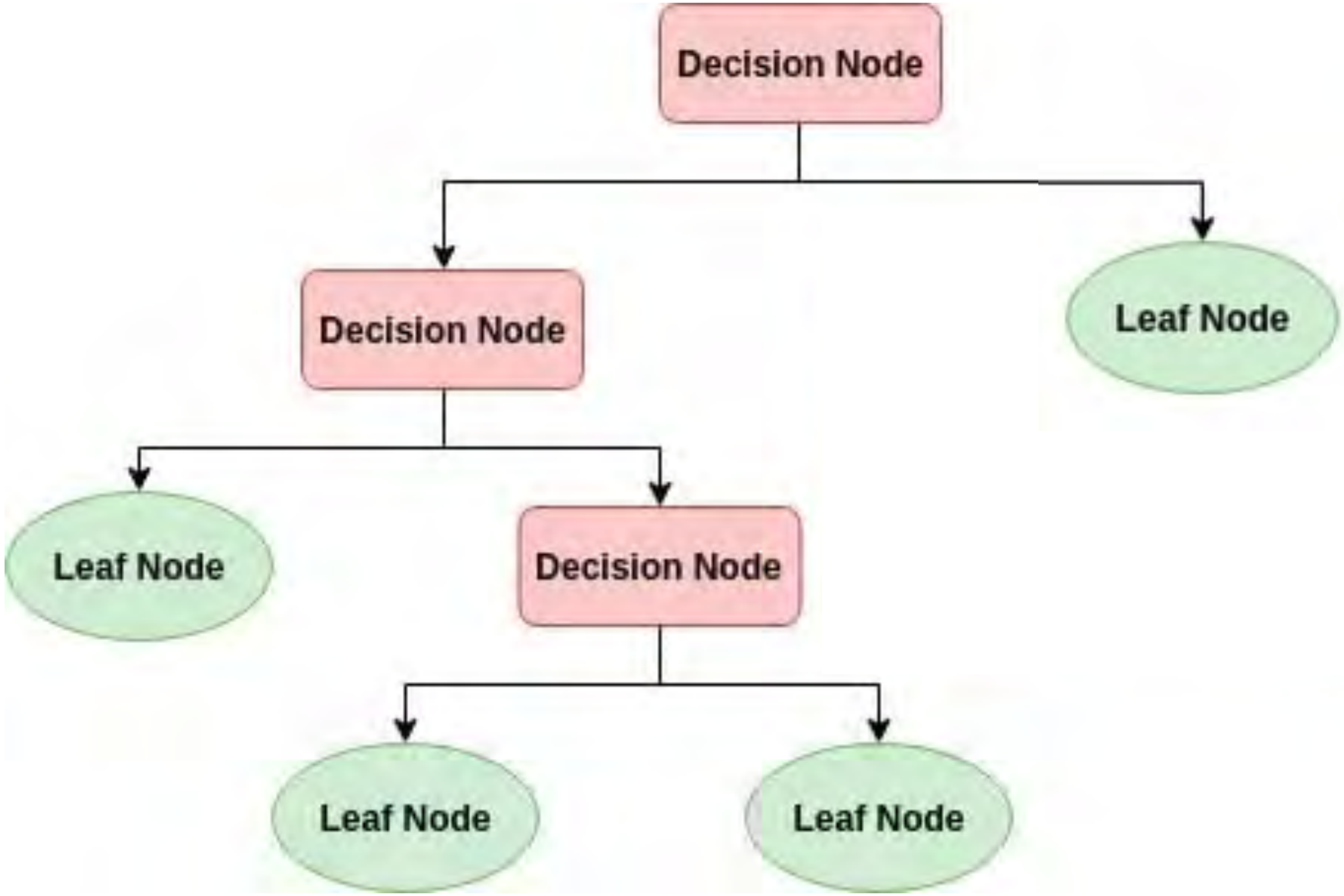
Decision Trees

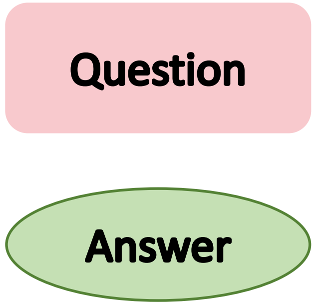


Twenty Questions - AKA animal, vegetable, or mineral

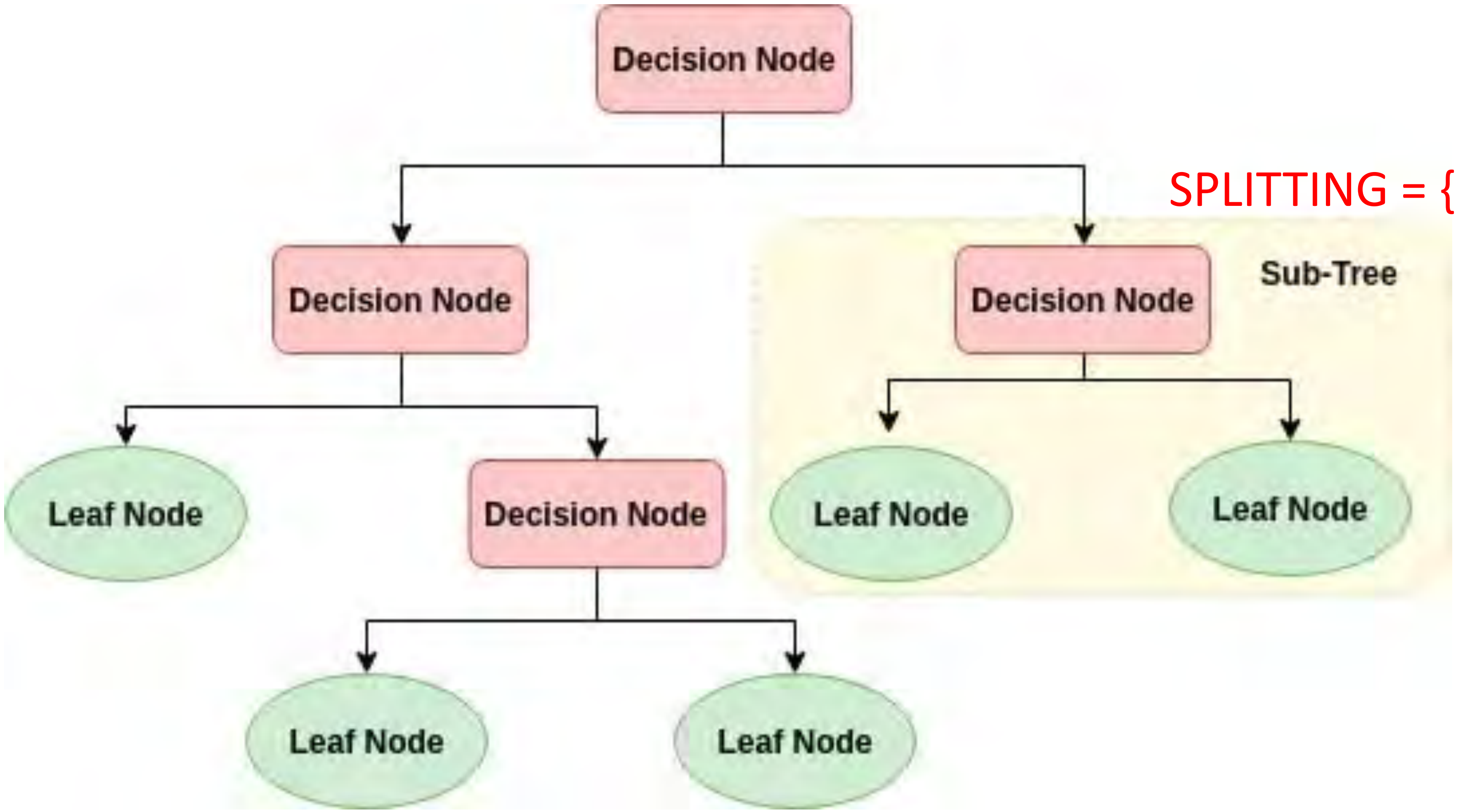


Some terminology

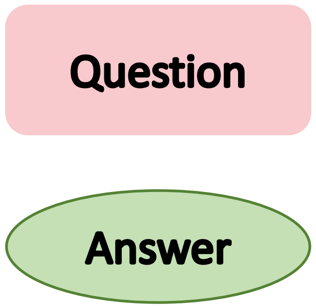




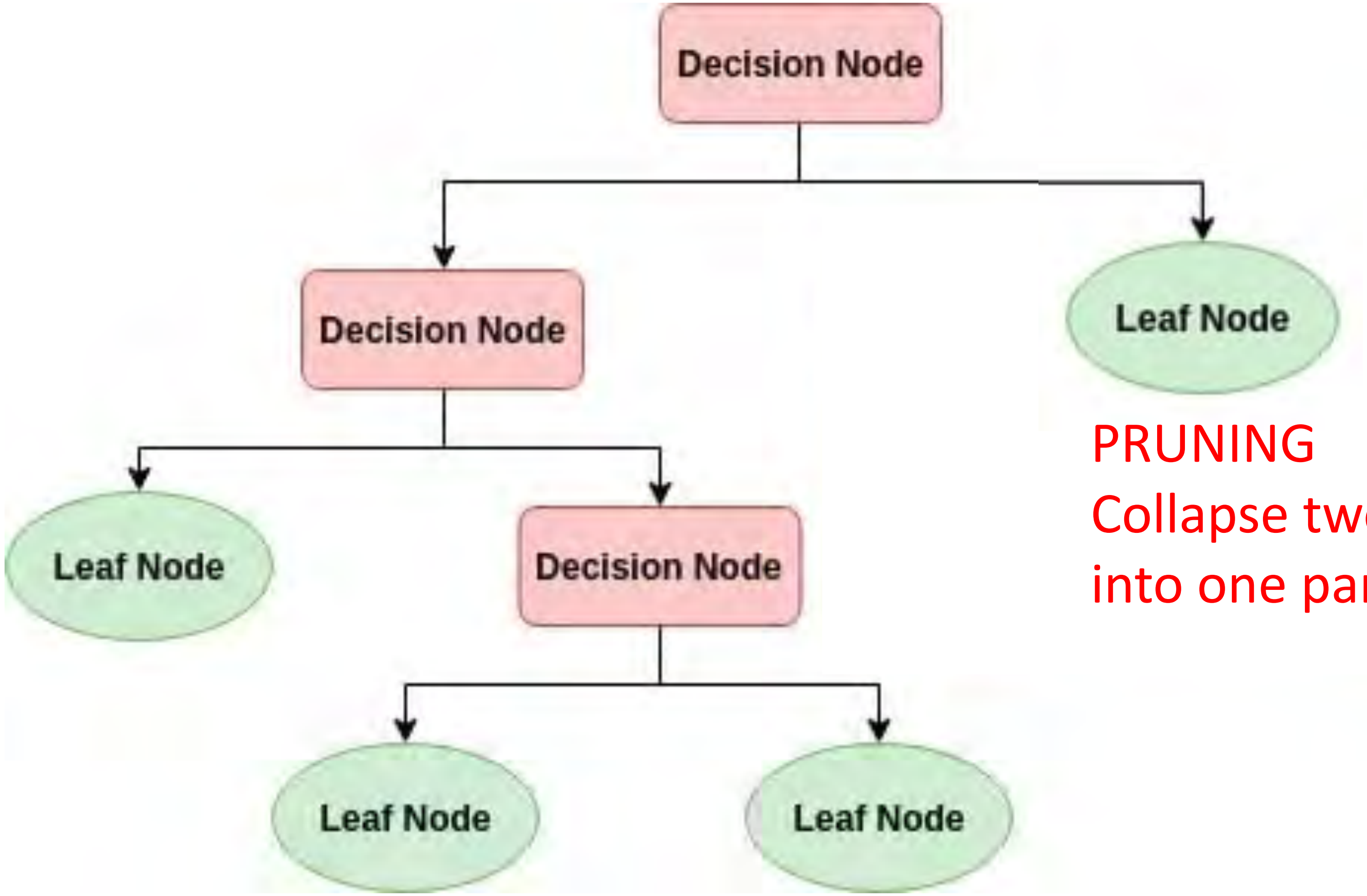
Growing trees is a recursive/iterative process



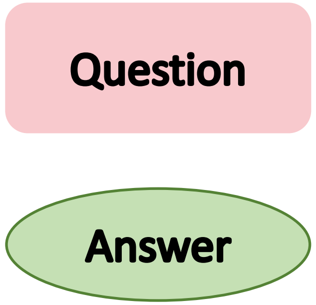
SPLITTING = $\begin{cases} \text{YES: go left} \\ \text{NO: go right} \end{cases}$



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

When should the iterations stop?

Are the leaves pure?

Ideally, we keep splitting until **all** the leaves are **pure**
This means that all the points in each leaf share the same label

If we stop before that, we want the leaves, on average, to be **as close to** pure as possible

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):

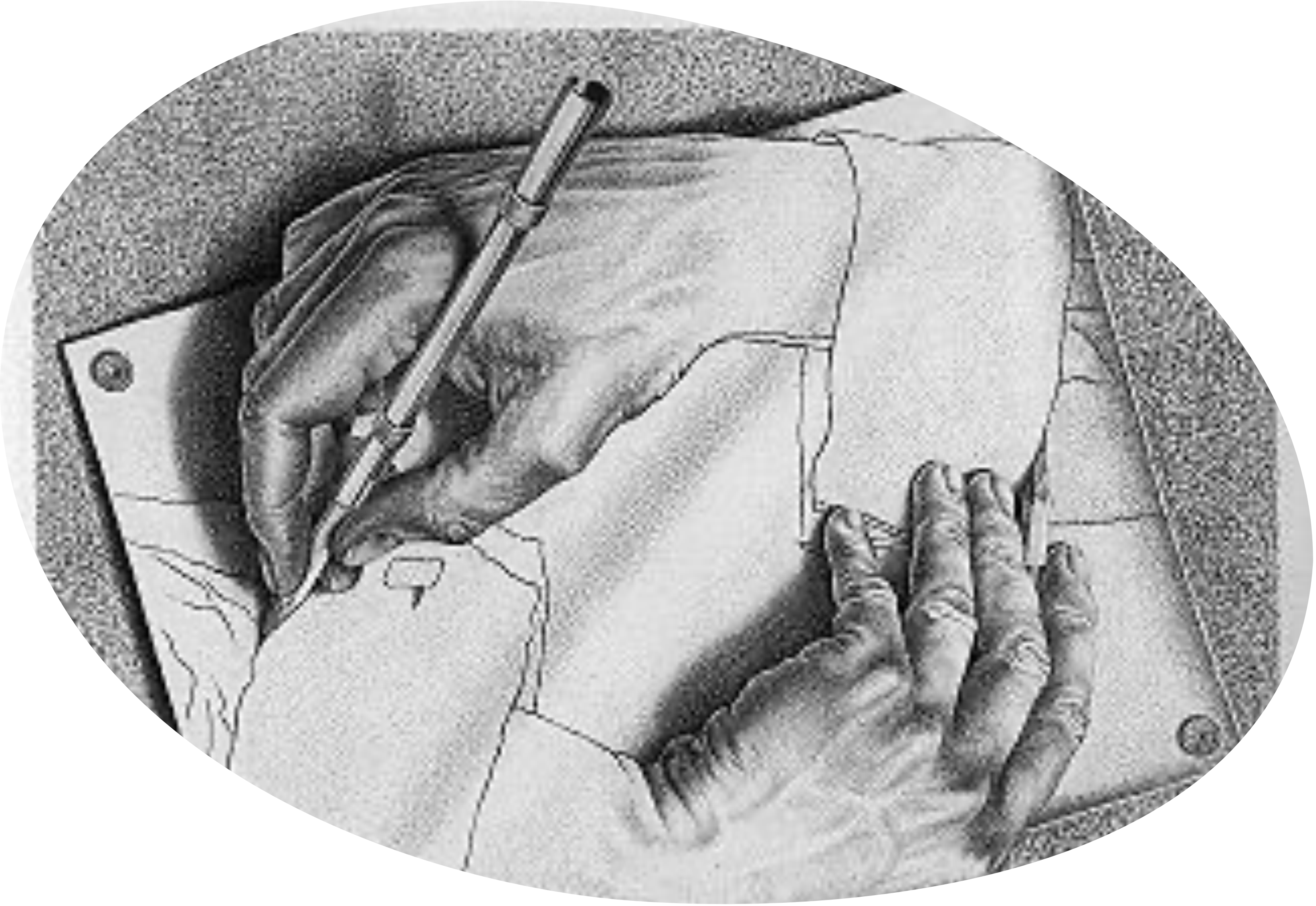
  Allows more than yes/no (multiway) splits, so needs more data

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
```



Hands-on Example:

Model tuning

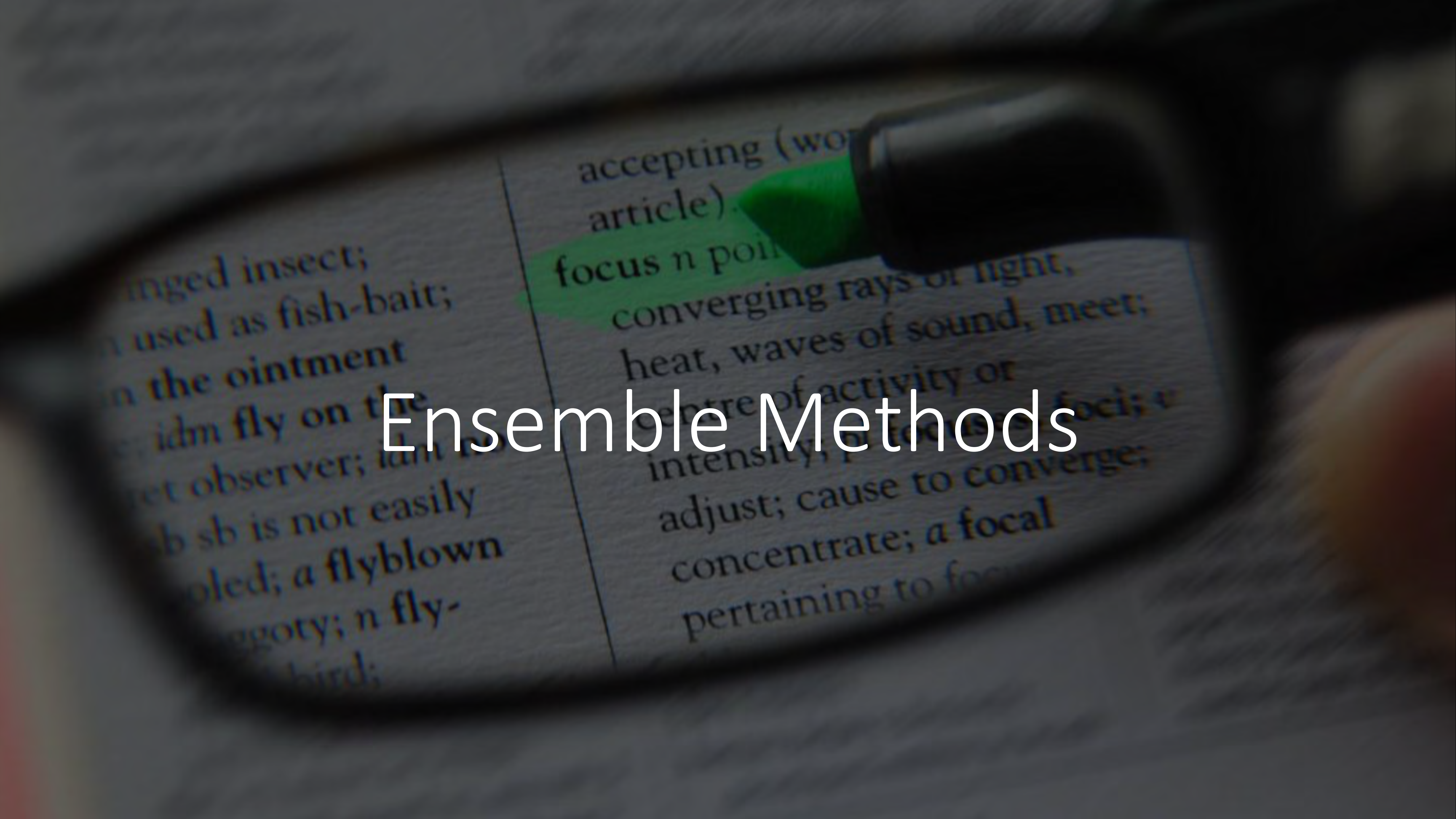
Is this all there is?

Are we done?

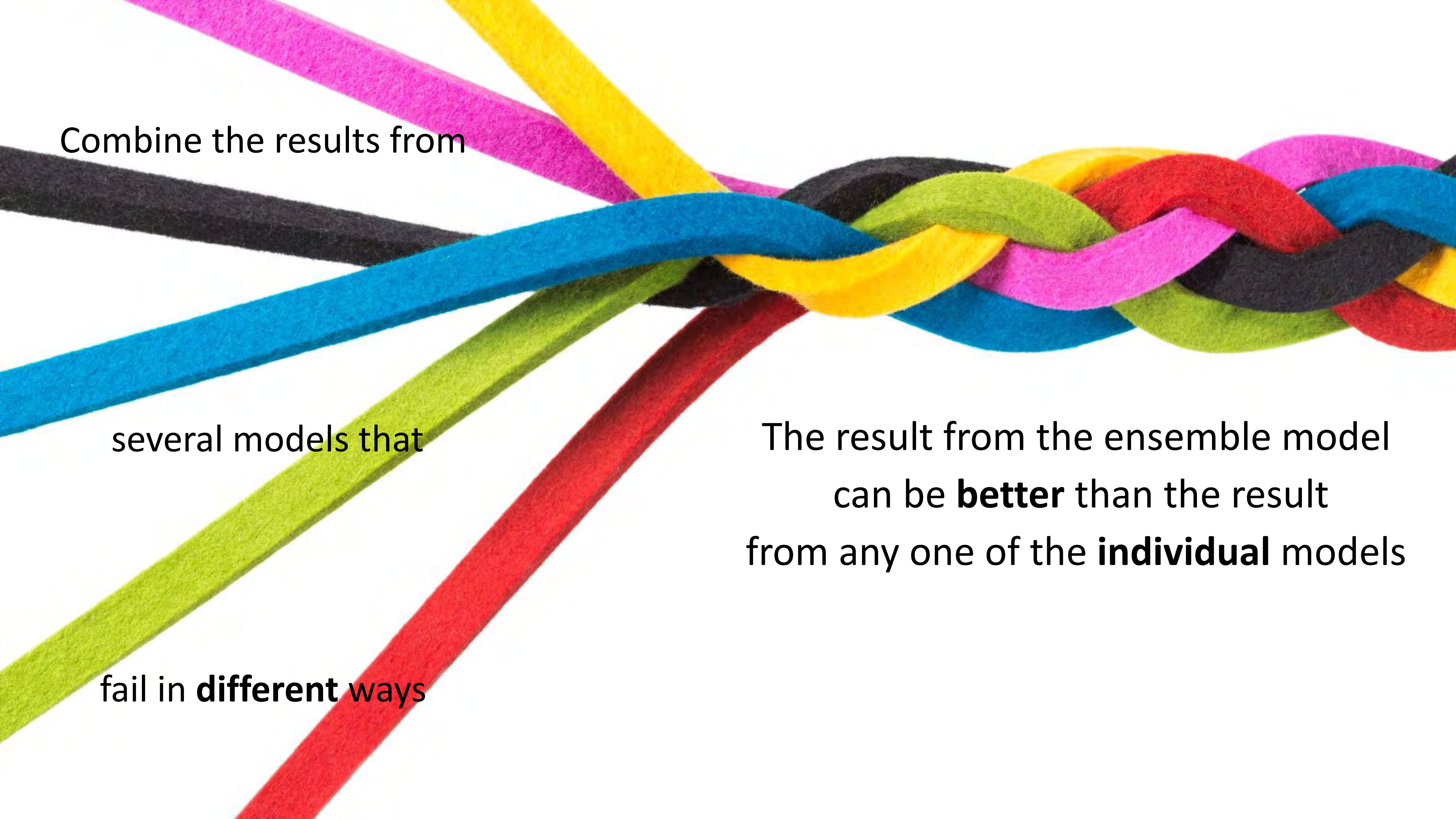
Potential Disadvantages of Decision Trees

- 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 (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: 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

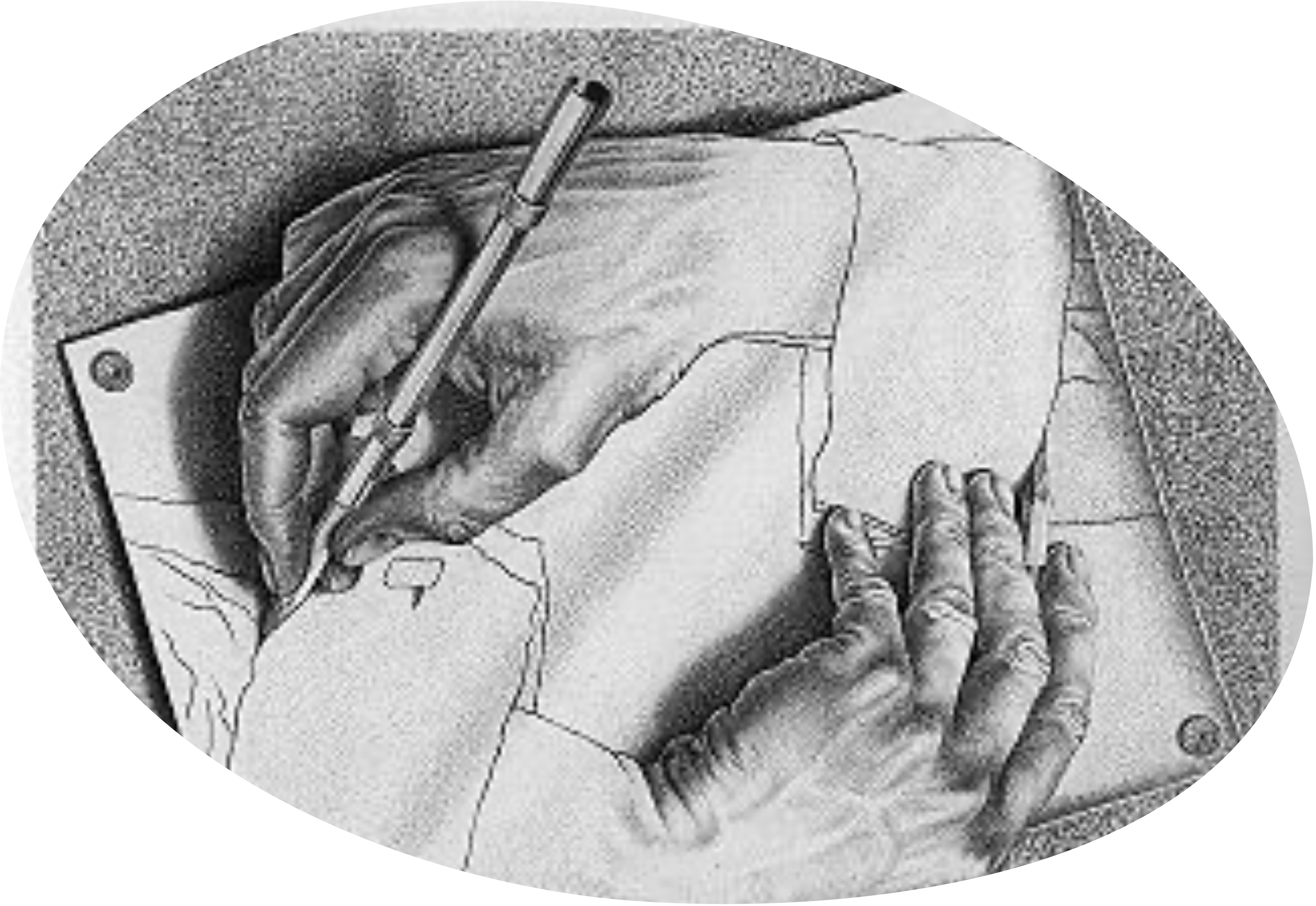
  e.g., fully developed (tall) decision trees

  players who consistently accumulate triple doubles

- Boosting methods usually work best with weak models

  e.g., shallow decision trees (stumps)

  players who are top of the league in one statistic



Hands-on Example: Ensemble