EEN1085/EEN1083

Data analysis and machine learning I

Ali Intizar

Ollscoil Chathair Bhaile Átha Cliath
Dublin City University

Decisions Trees

Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

Outline

- Decision Trees
  - Top-down greedy search
  - CART
  - Advantages and limitations of tree models

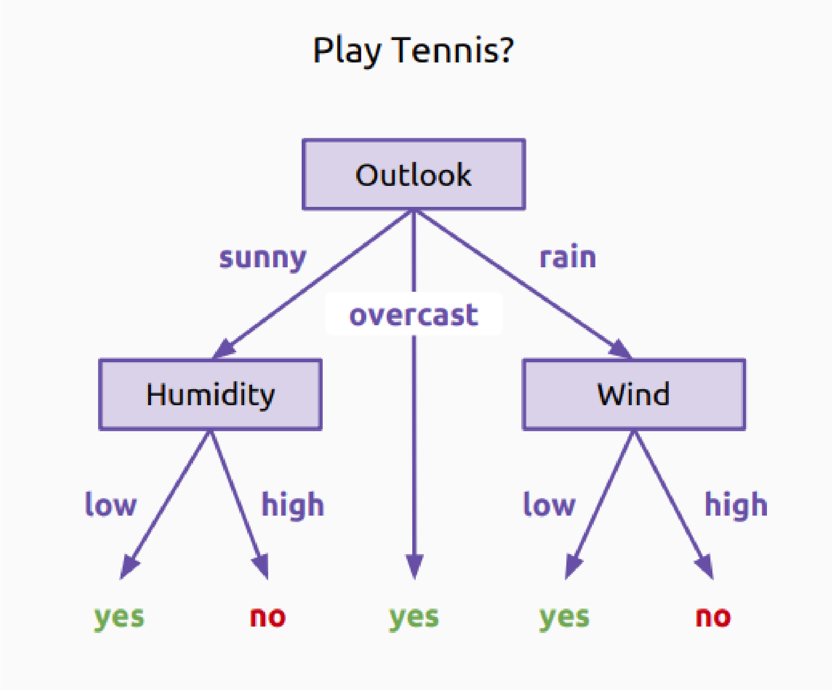
Decision Trees

Decision Trees

Hierarchical models for prediction.

Terminology:

- **Nodes** represent attributes of data (predictors, features, variates)
- **Branches** connect to other nodes based on value of attributes
- **Leaf** nodes assign outputs (e.g. classification)



Decision Trees

Classification trees: target variable is discrete and categorical. E.g. $y \in \{\text{red}, \text{green}, \text{blue}\}$

Regression trees: target variable is a real number. $y \in \mathrm{R}^D$

Fitting classification tree models

Given a training dataset, we would like an algorithm to produce a decision tree model that predicts a target variable from a set of attributes.

For now, lets simplify the problem a little:

- Assume ordinal attributes (e.g. $\mathbf{x} \in \mathrm{R}^D$)
- Assume binary classification $y \in \{0, 1\}$
- Consider only **binary** trees.

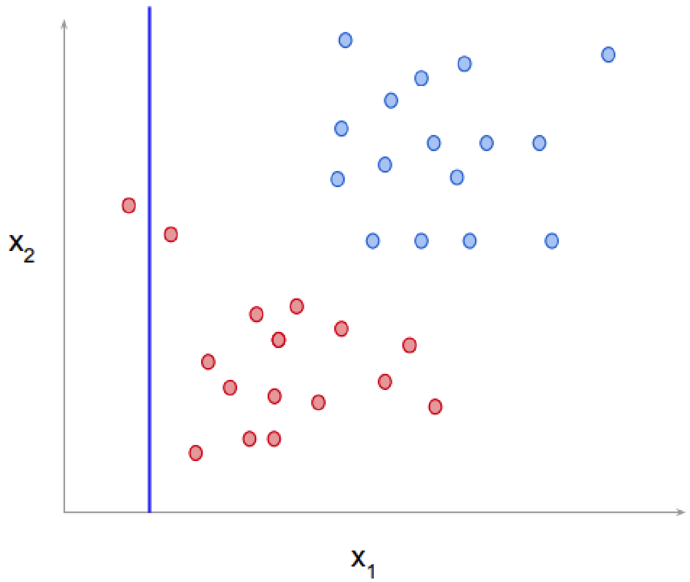
Idea: At each node in our tree we can split the data based on one of the attributes. Build the tree by recursively splitting the data.

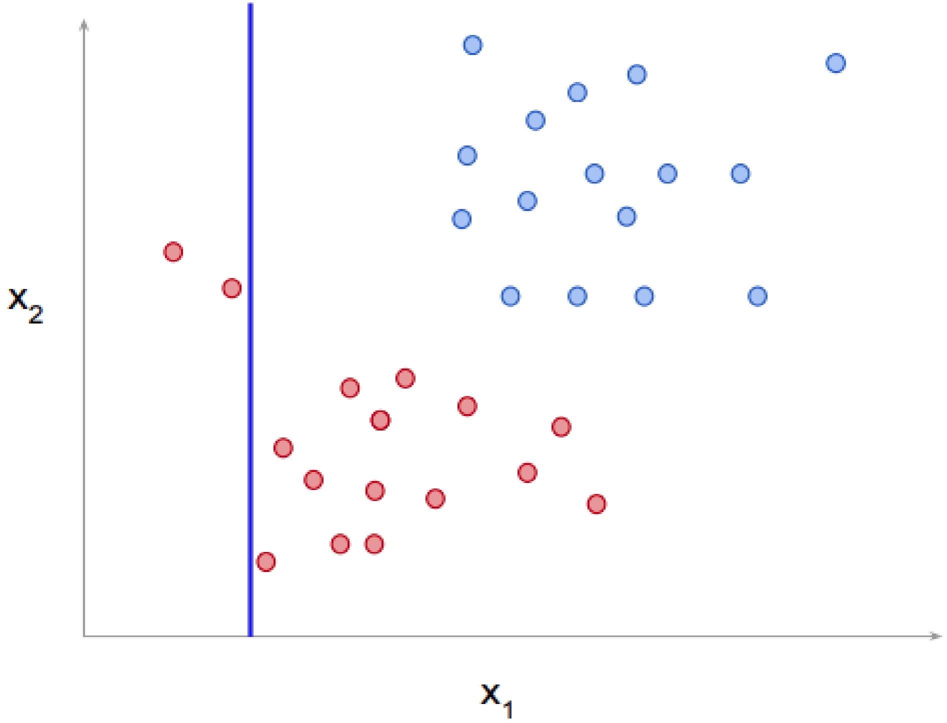
Top-down greedy search

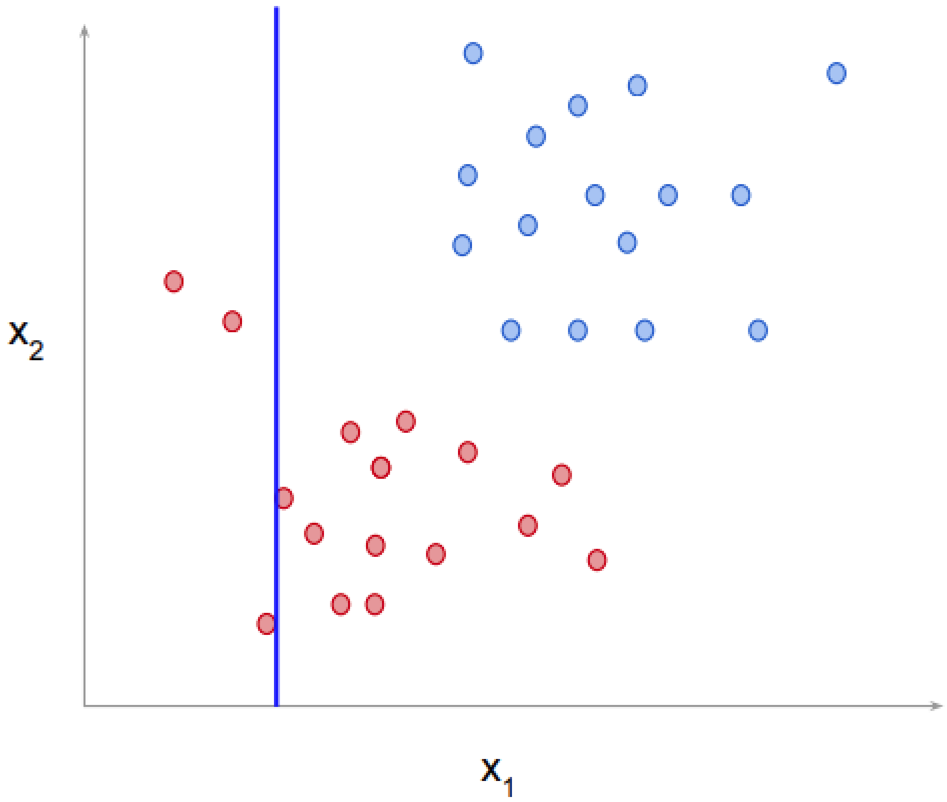
Starting with the root node, recursively:

1. Find the attribute (dimension) and split point (threshold) that maximize some **measure of quality** (e.g. accuracy)
2. If the **stopping criteria** is not met:
   - Create a tree node to split on this attribute.
   - Perform the same procedure recursively for the left and right children.

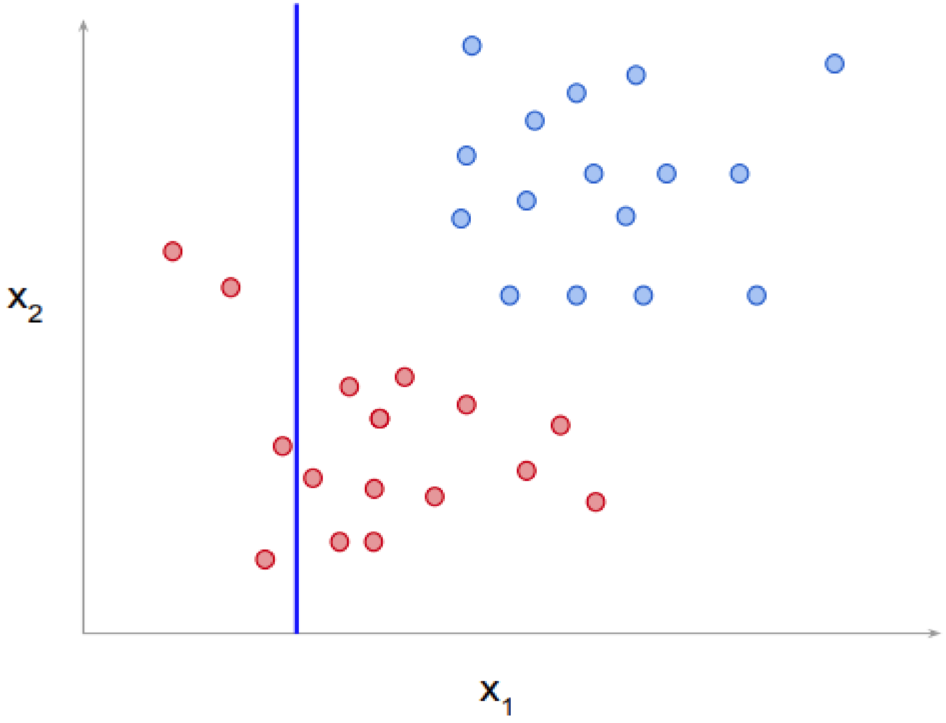
Example...

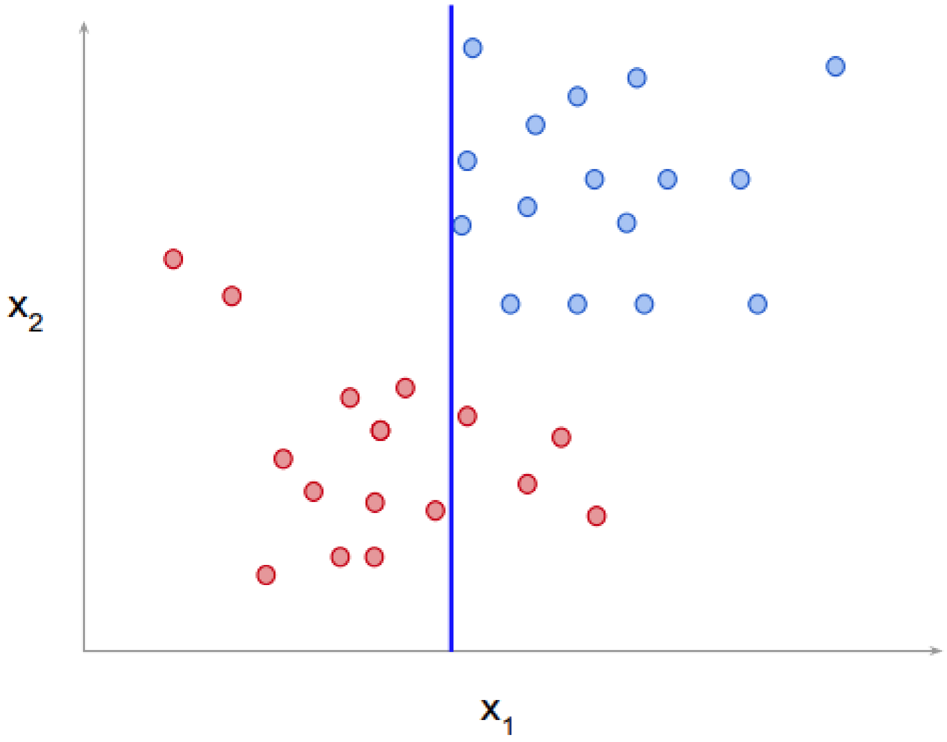

x_2
x_1


x_2
x_1

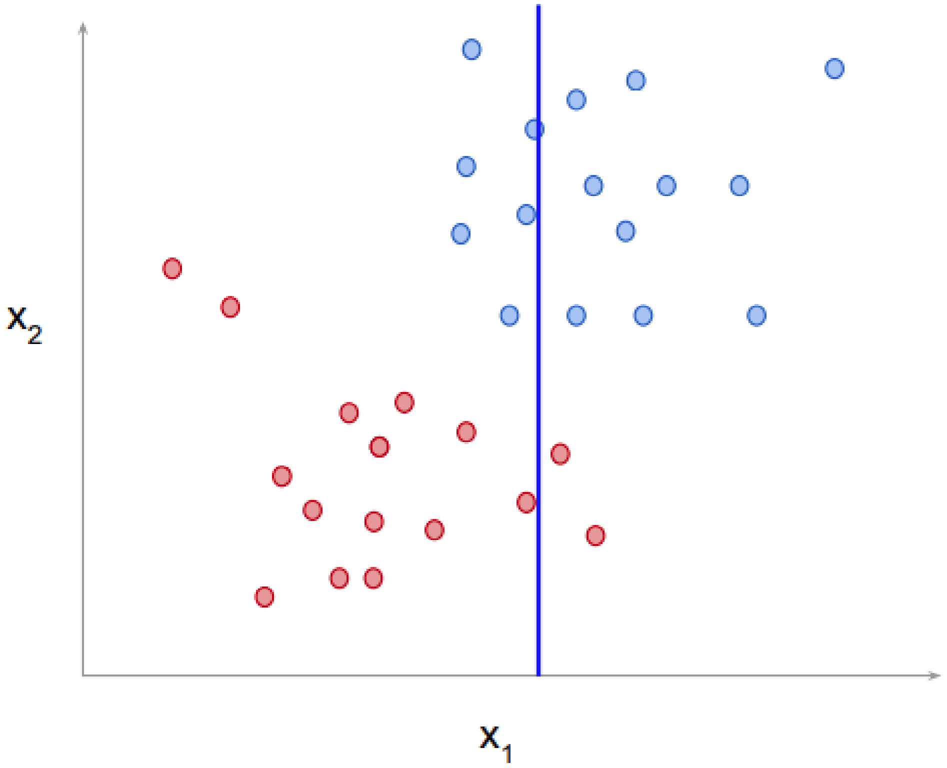


x_2
x_1

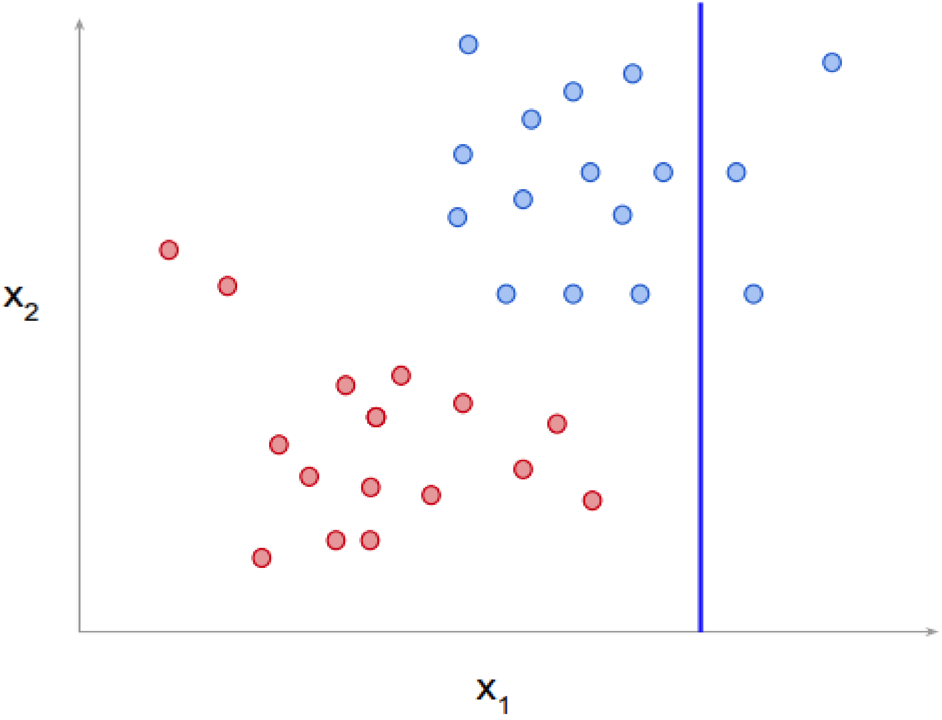

x_2
x_1



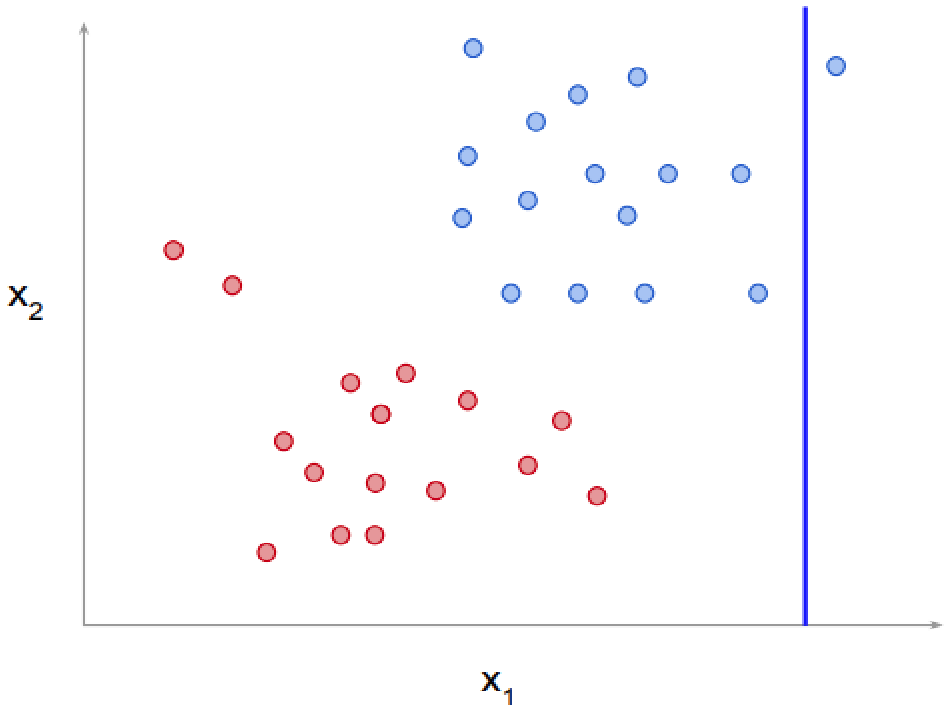
x_2
x_1

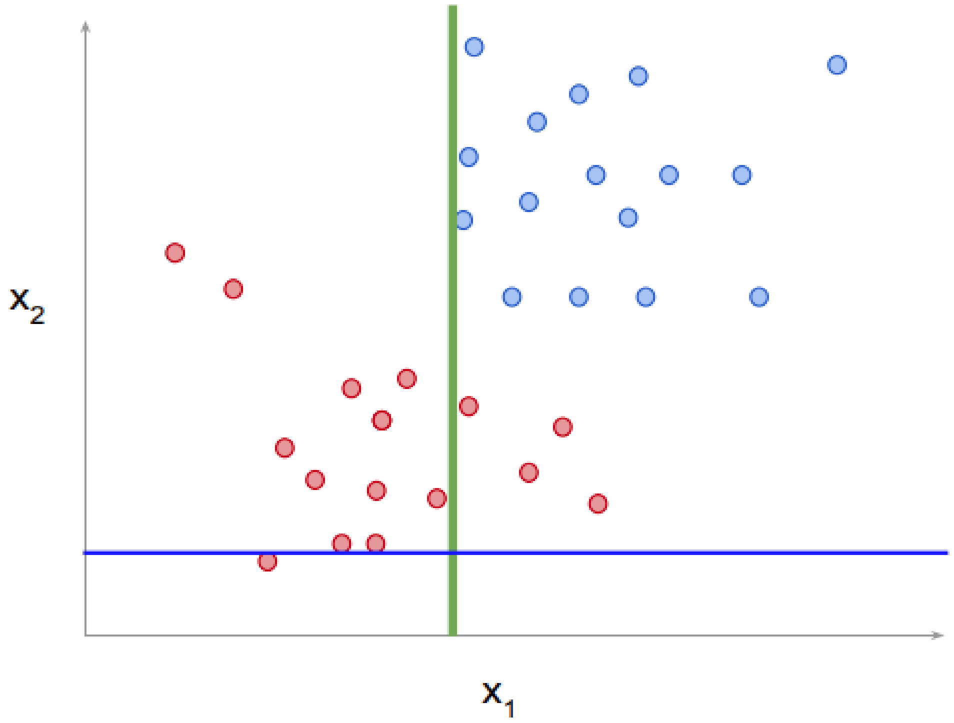


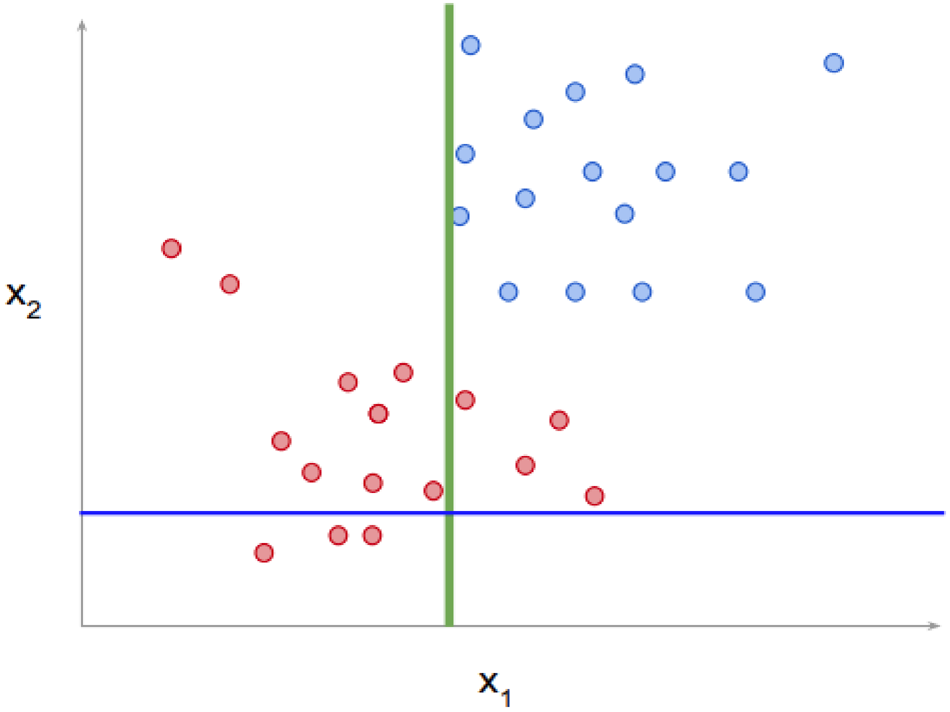
x2
x1



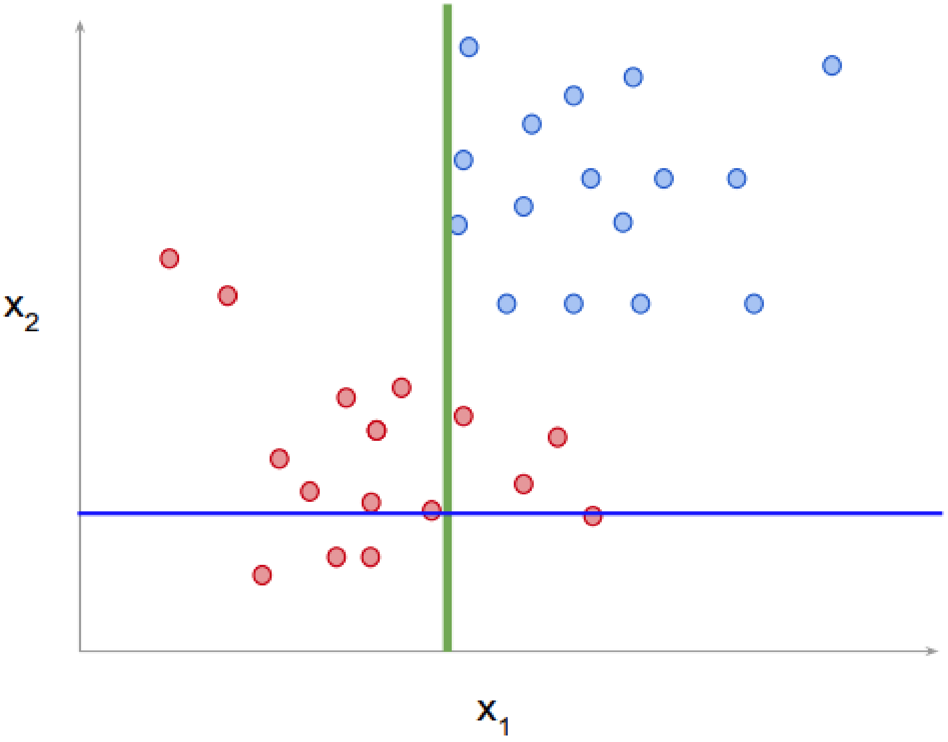
x2
x1


x_2
x_1


x_2
x_1

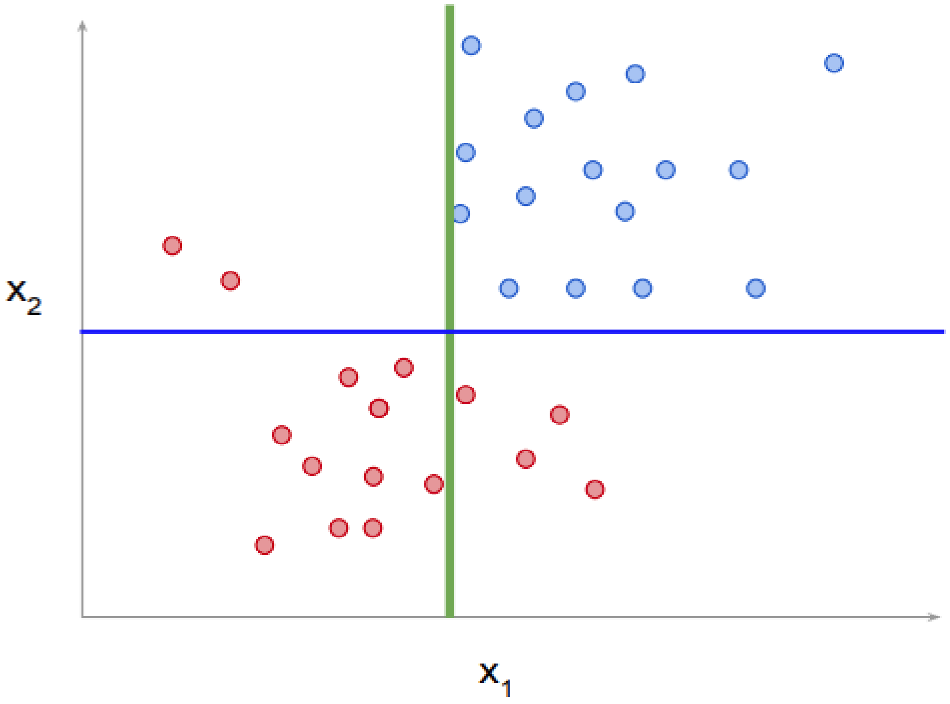


x2
x1

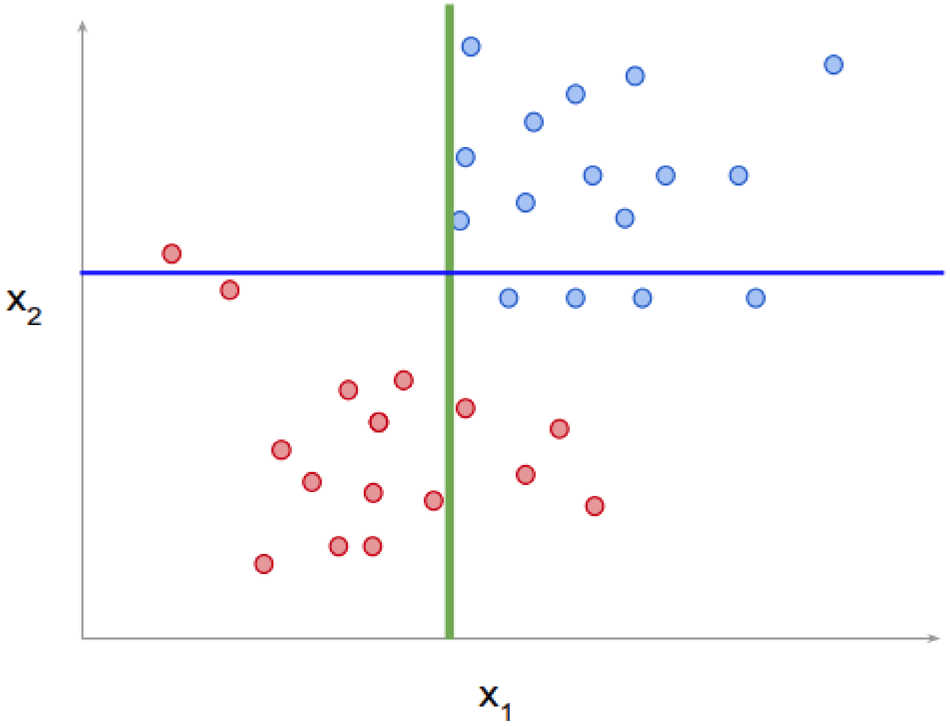


x_2
x_1

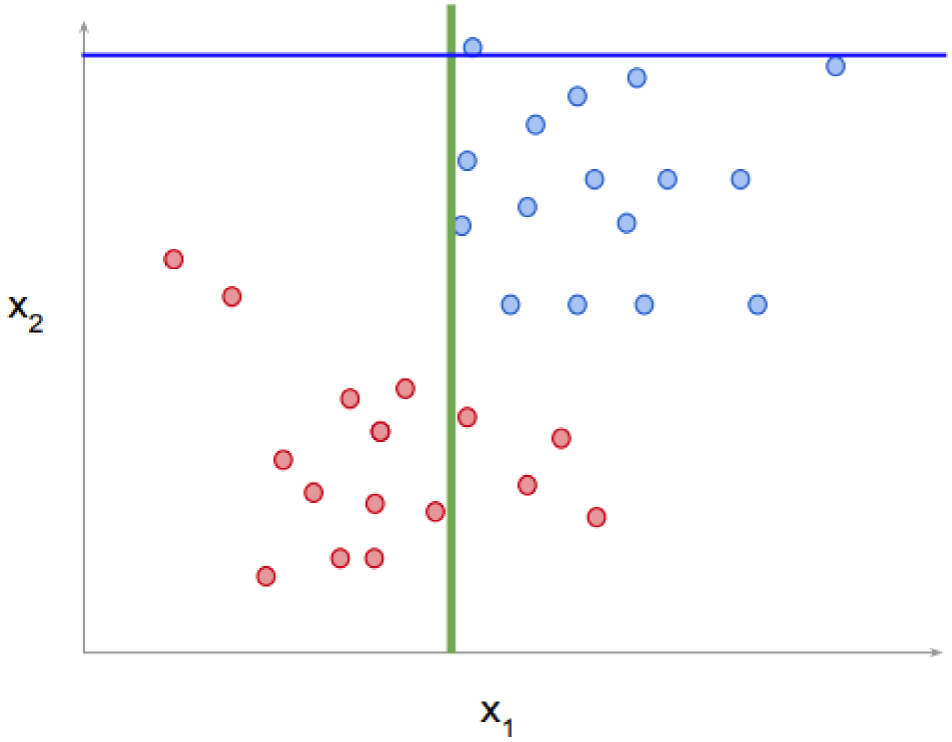
• • •

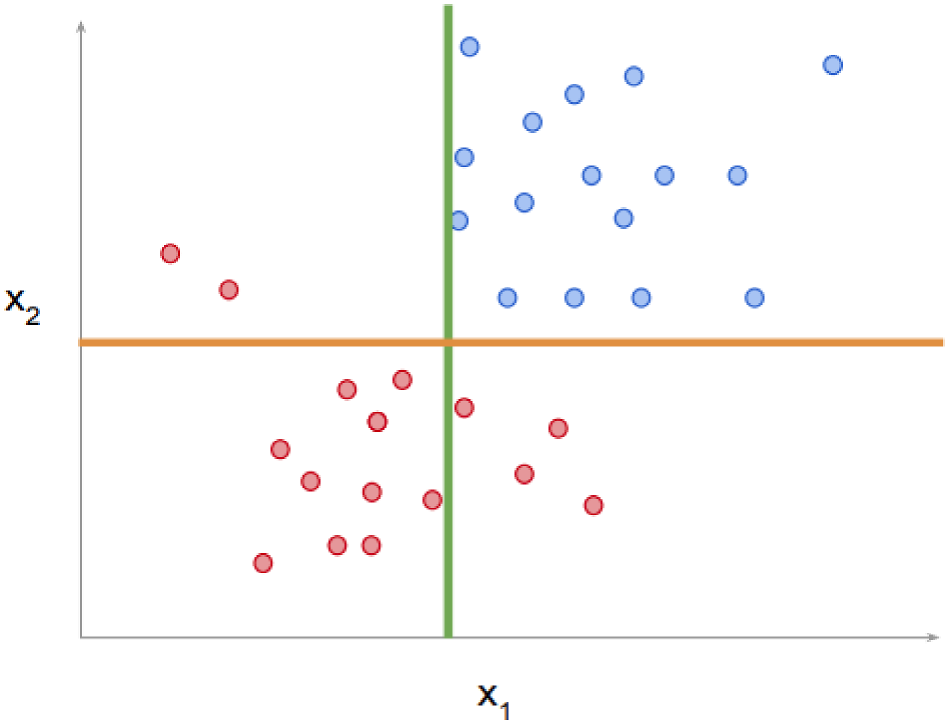


x_2
x_1

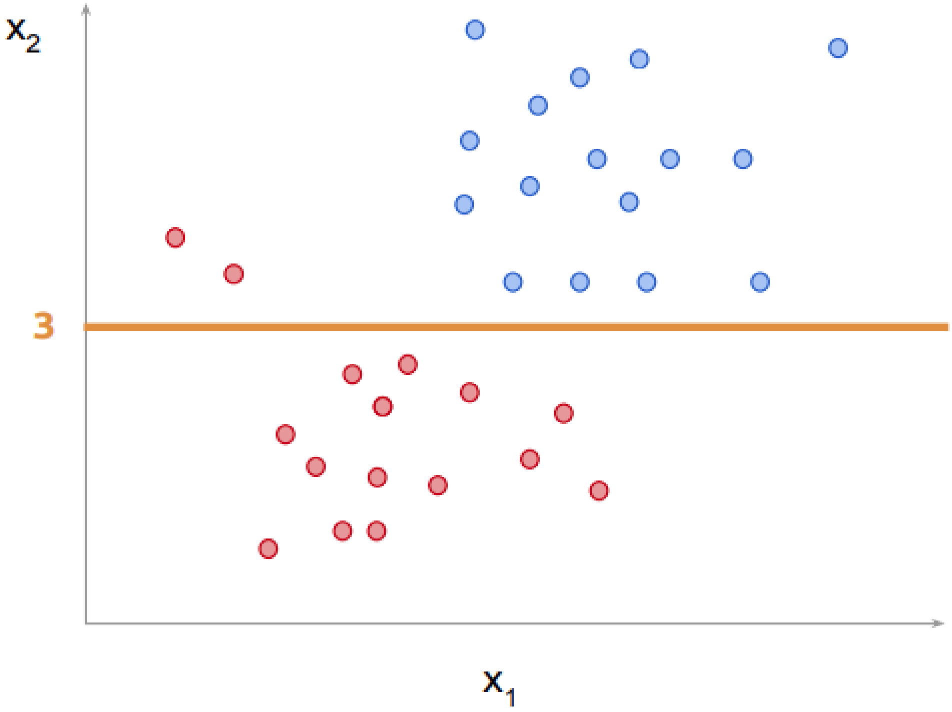

x1
x2

• • •

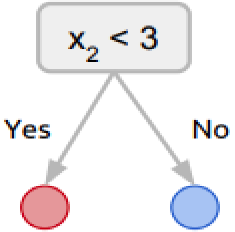

x_2
x_1



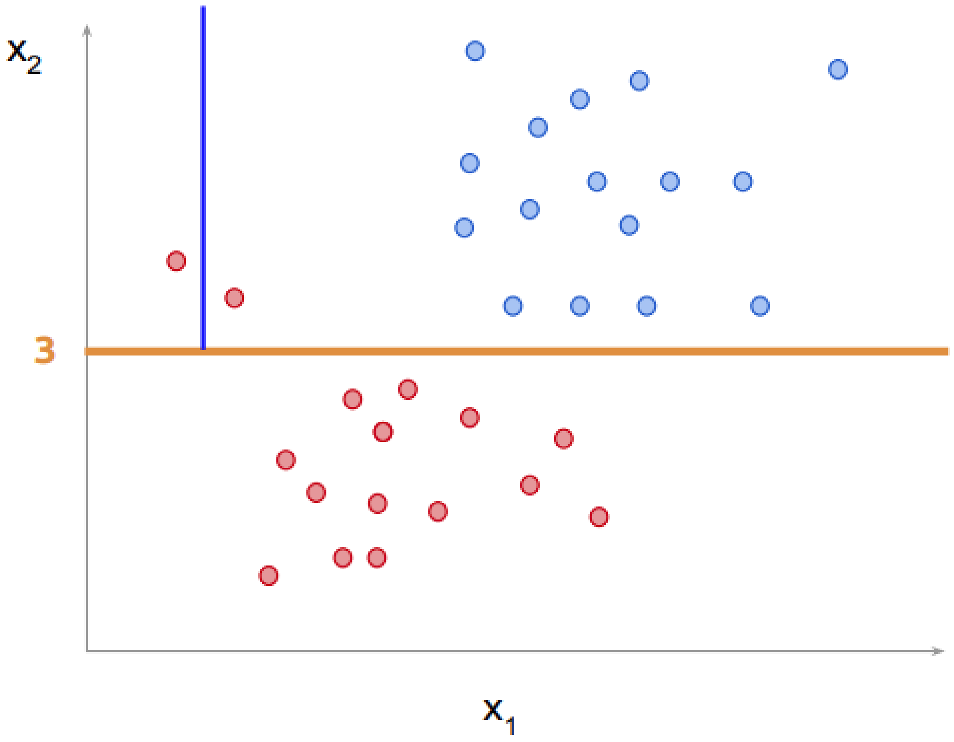
x2
x1

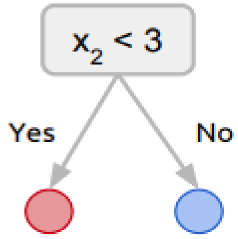


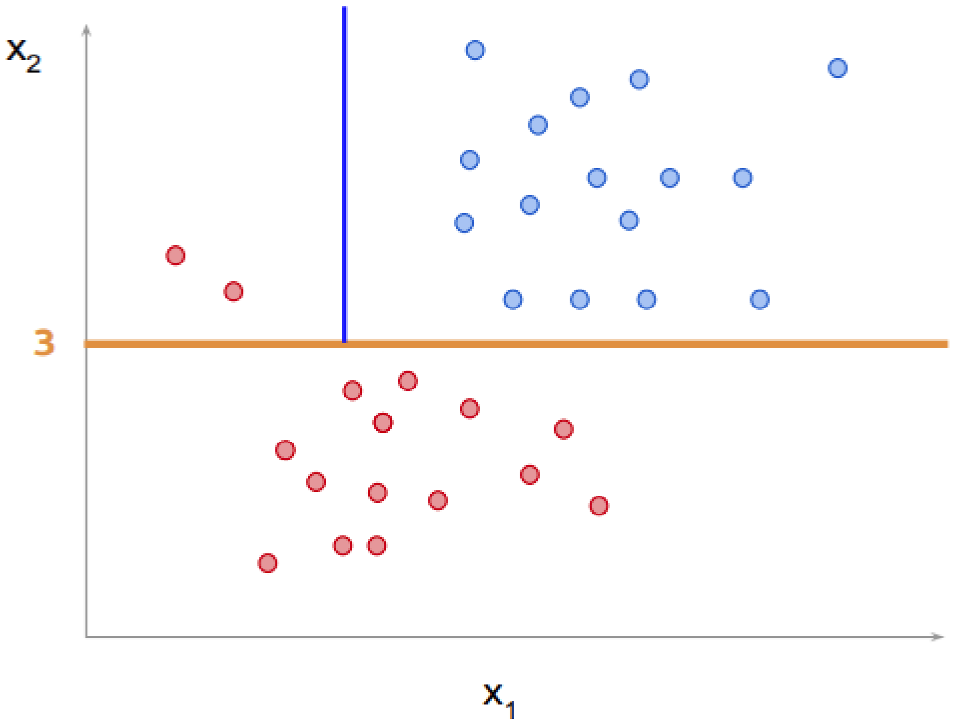
x_2
3
x_1



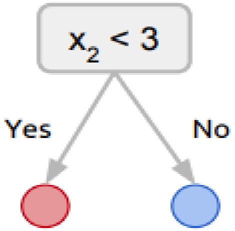
$x_2 < 3$
Yes
No


x2
3
x1

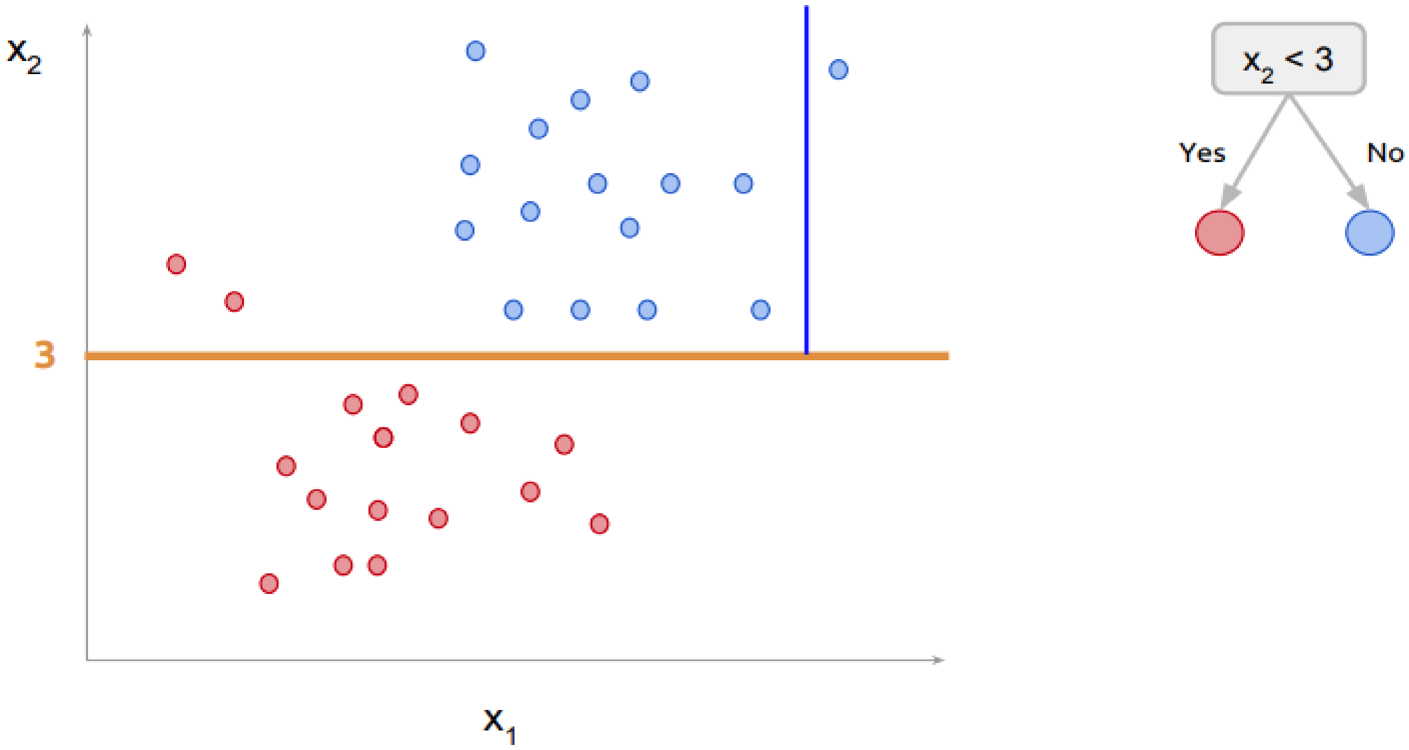

x2 < 3
Yes
No



x_2
3
x_1



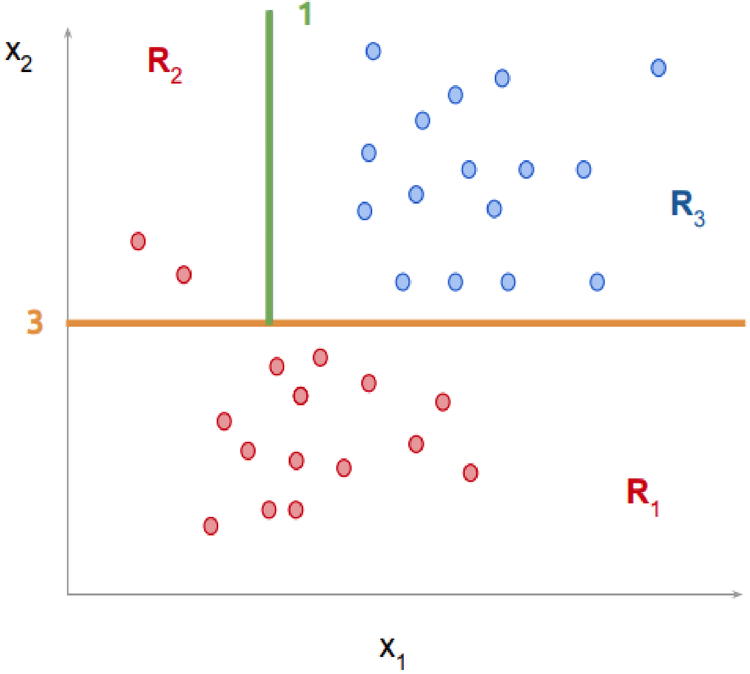
$x_2 < 3$
Yes
No

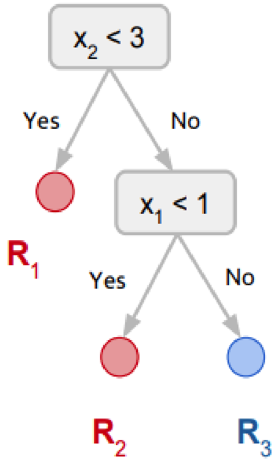


x2
3
x1
x2 < 3
Yes
No

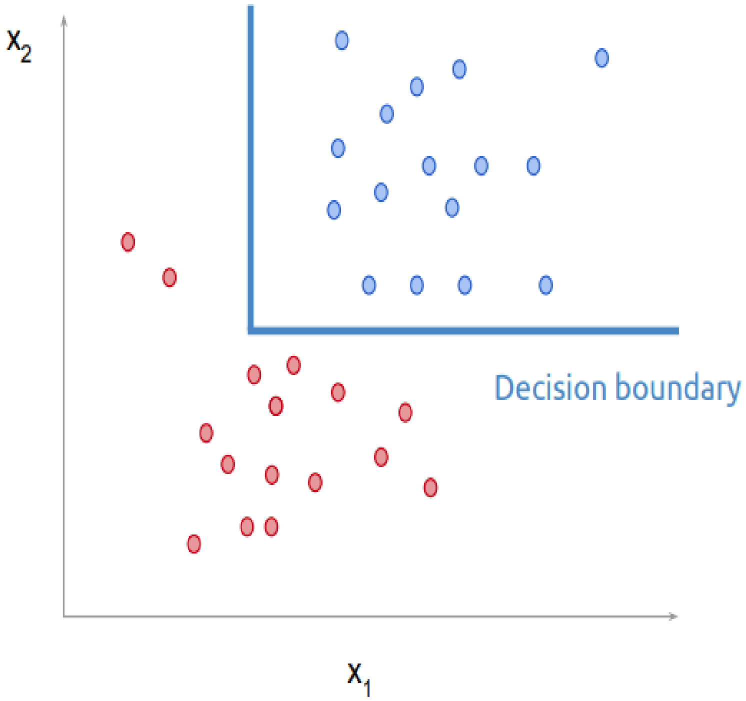
$$f(x) = \sum_{m=1}^{M} c_m [x \in R_m]$$

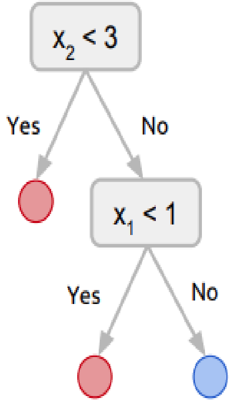
Most common category in the region





$$f(x) = \sum_{m=1}^{M} c_m [x \in R_m]$$



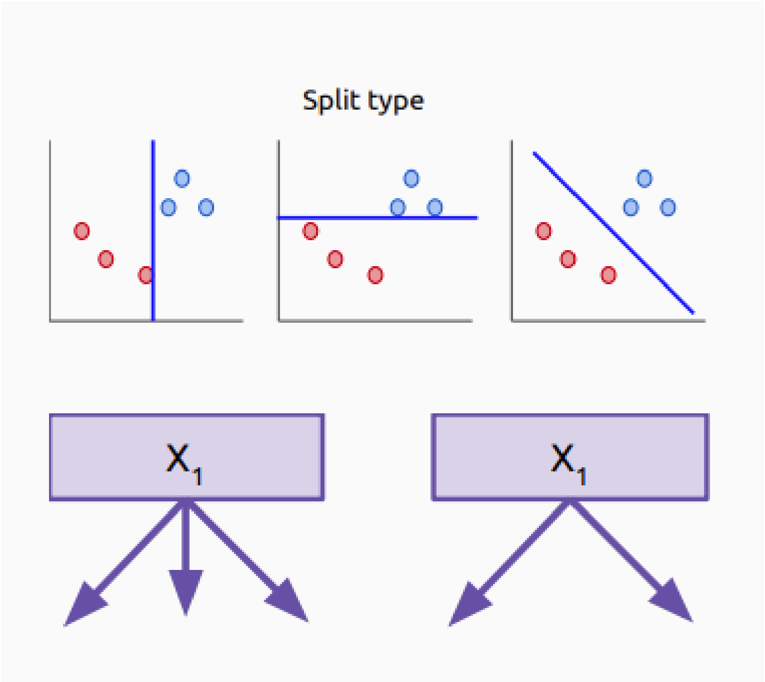


Decision tree algorithms

Many algorithms proposed for fitting (growing) decision trees

Variations:

- **Split type**: axes aligned (univariate) or linear
- **Branches per split**: binary or multinomial
- **Splitting criteria**: e.g. misclassification error, entropy, Gini index
- **Stopping criteria**
- Handling missing values



CART

- Classification and regression trees

- **Classification trees**: Predict a categorical output.
- Prediction for a region is the category that occurs most frequently in the region.

- **Regression trees**: Predict a real valued output.
- Instead of voting inside a region for the correct category, just average all y values in the region.

CART

- CART generates a **binary tree**.

- Top-down greedy splits nodes to minimize **impurity**:
  - Classification: Impurity of a node measured by impurity metric like the Gini index.
  - Regression: Impurity of a node measured by mean squared error.

- **Stopping criteria**: grow tree out to maximum size and prune it back. Use cross validation to find the optimal tree.

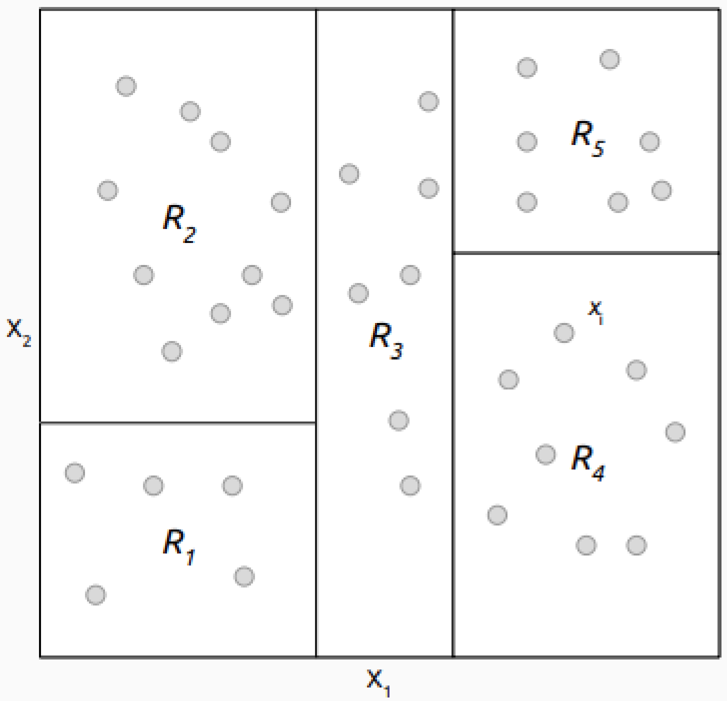
CART for regression

Notation:

- Training data: $\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathrm{R}^D,\ y_i \in \mathrm{R}\}_{i=1}^N$
- Features: X_j
- Regions: $\{R_1, \ldots, R_M\}$
- Region size: $N_m = |R_m|$

Prediction for region R_m is the average y value in the region:

$$c_m = \frac{1}{N_m} \sum_{i=1}^{N} y_i \mathbb{1}(\mathbf{x}_i \in R_m)$$



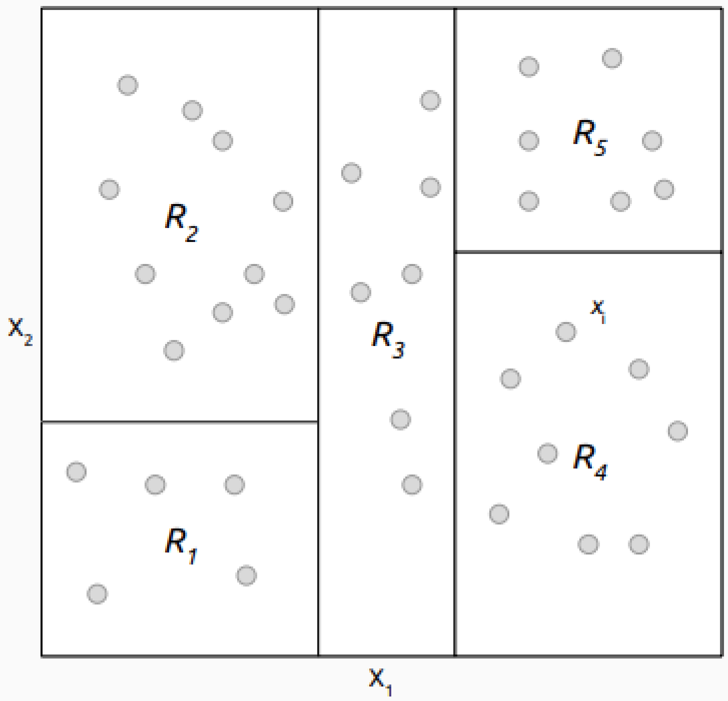
CART for regression

The decision function is:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{1}(\mathbf{x} \in R_m)$$

Define the **impurity** of region R_m as:

$$E_m = \sum_{x_i \in R_m} (y_i - c_m)^2$$



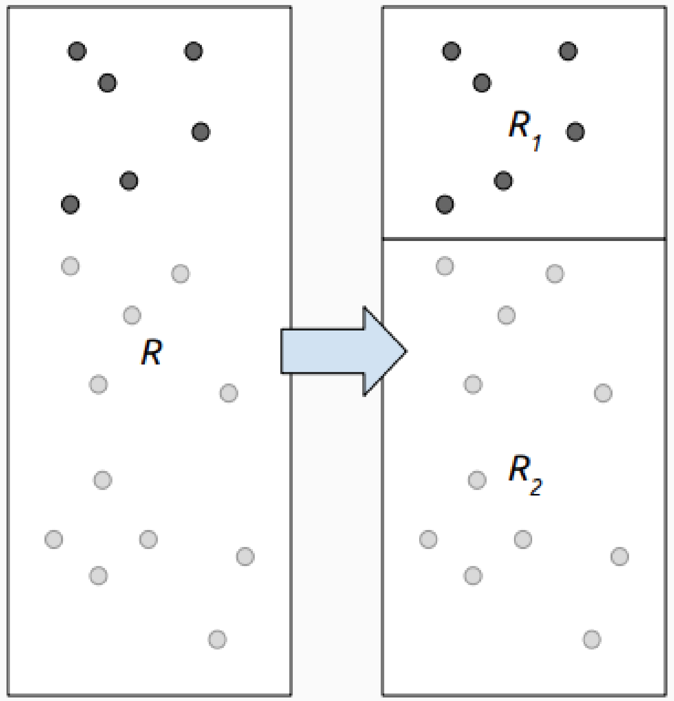
CART for regression

Greedy top-down algorithm.

Starting with a single region, recursively split region R into two subregions R_1 and R_2.

Split: Search all features X_j and split points s to *minimize the impurity* of the resulting regions:

$$L(j, s) = E_1 + E_2 = \sum_{x_i \in R_1} (y_i - c_1)^2 + \sum_{x_i \in R_2} (y_i - c_2)^2$$



CART for regression

Stopping criteria

Several strategies can be used:

1. Only split a node if the decrease in impurity exceeds a threshold.
2. Stop when each region contains only one point (grow tree to maximum depth).
3. Stop when each region contains fewer than K points.
4. Stop when the tree reaches a certain maximum depth.
5. Grow tree out to maximum depth and prune it back.

(1) is too shortsighted: poor split now could lead to great ones later. (2) can lead to overfitting. (3-4) are commonly used. (5) requires a **pruning algorithm**.

CART for regression

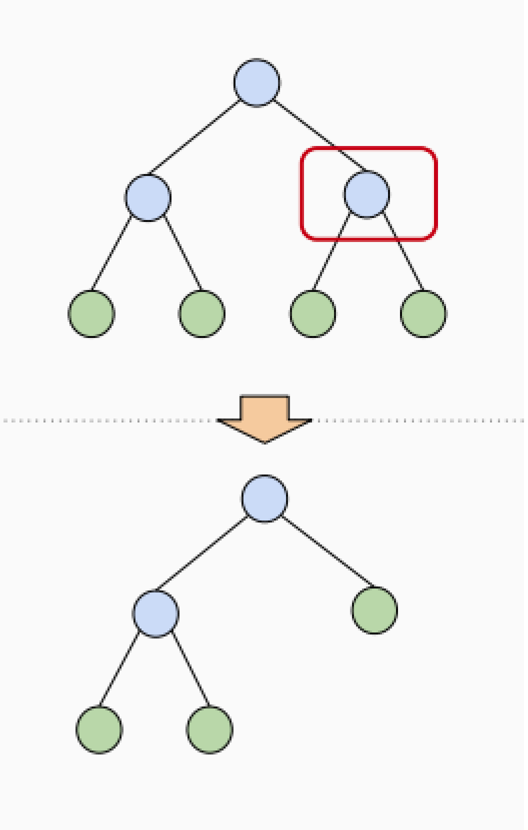
Cost complexity pruning

Idea: try to simultaneously minimize impurity (cost) and tree complexity.

Let T be the set of terminal (leaf) nodes in the tree. The *cost complexity criteria* trades off impurity for tree size $|T|$:

$$C_\alpha(T) = \sum_{m=1}^{|T|} E_m + \alpha|T|$$

Can find subtree T_α that minimizes cost complexity criteria for a given α by **weakest link pruning**.

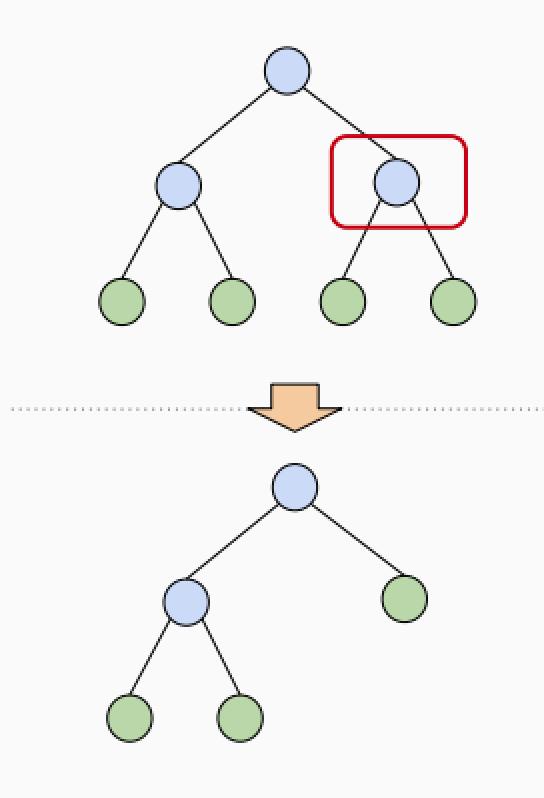


CART for regression

Weakest link pruning

1. Successively collapse internal nodes that produces the smallest increase in impurity until only the root node remains.
2. Compute C_α on each successive subtree and choose one with minimum value.

Need to choose hyperparameter α. Can be done using a validation set or by *cross validation*.

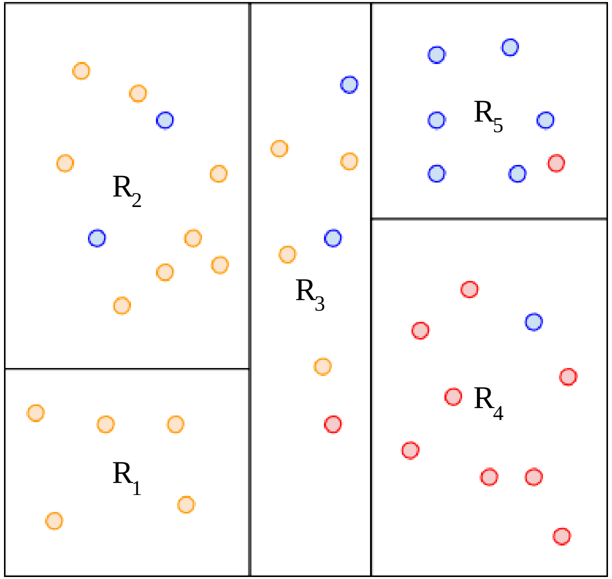


CART for classification

We can use almost the same algorithm for classification as we used for regression.

Two changes:

1. **Prediction function** (c_m): the prediction for a region is done by voting instead of averaging.
2. **Impurity measure** (E_m): Mean squared error is not appropriate for categorical variables.



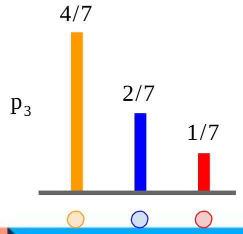
CART for classification

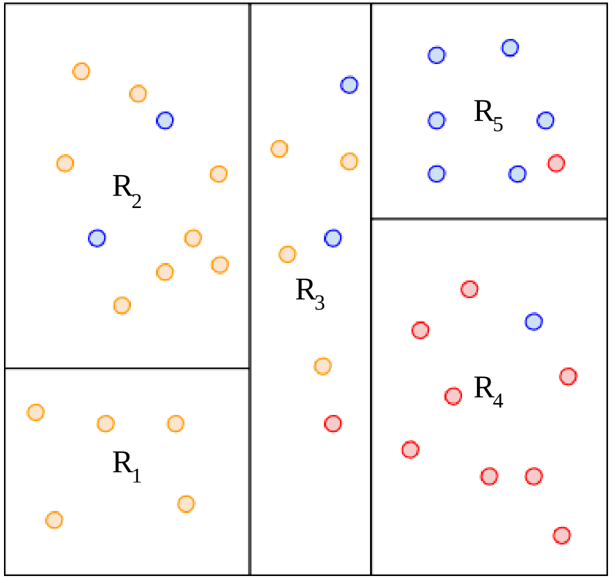
Region composition

Let $\hat{p}_{mk}$ be the proportion of y values in region R_m that take the value k:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbb{1}(y_i = k)$$

I.e. $\hat{p}_{mk}$ is a normalized **histogram** of the values of y in region R_m. E.g. region R_3, $\hat{p}_{3k}$ looks like:

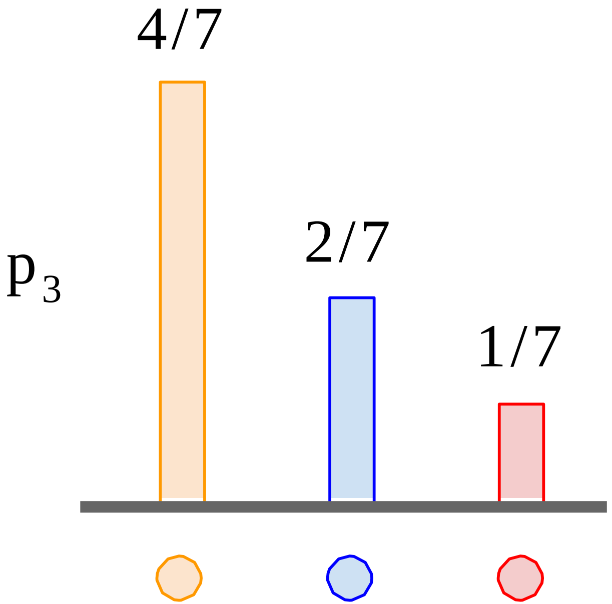




CART for classification

Prediction for region: y value with the most votes.

$$c_m = \max_k \hat{p}_{mk}$$



Region impurity Three commonly used metrics:

1. Misclassification error.
2. Entropy.
3. Gini index.

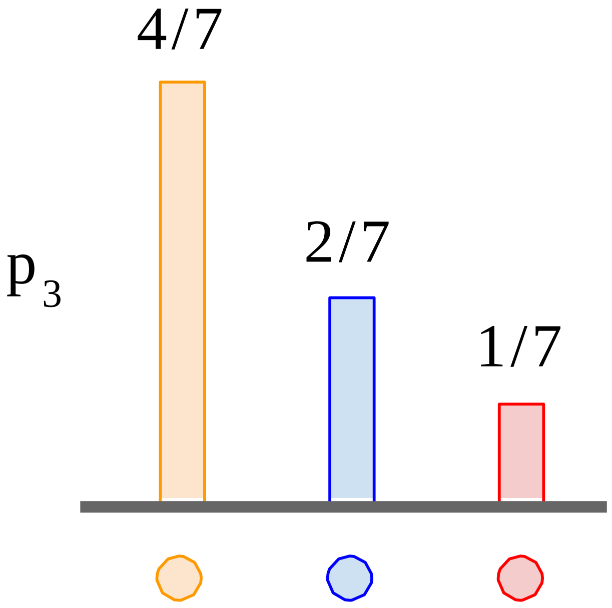
CART for classification

Misclassification error: number of misclassified elements in region R_m:

$$E_m = 1.0 - c_m$$

Entropy: information theoretical measure of *disorder* of a region.

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log_2 \hat{p}_{mk}$$

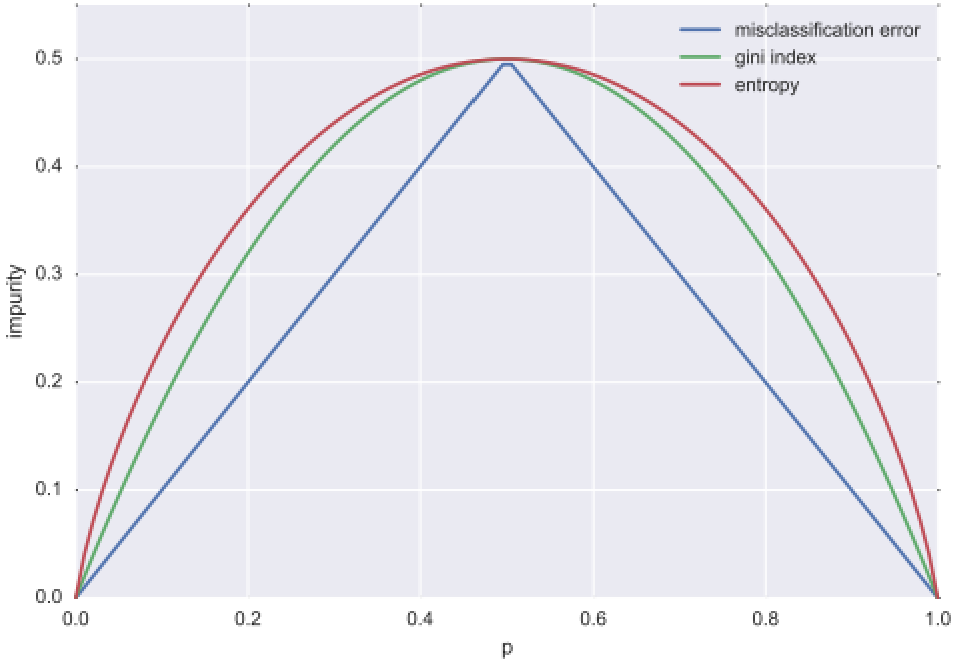


How many bits you would need to encode random draws from the region. Pure region has entropy 0.

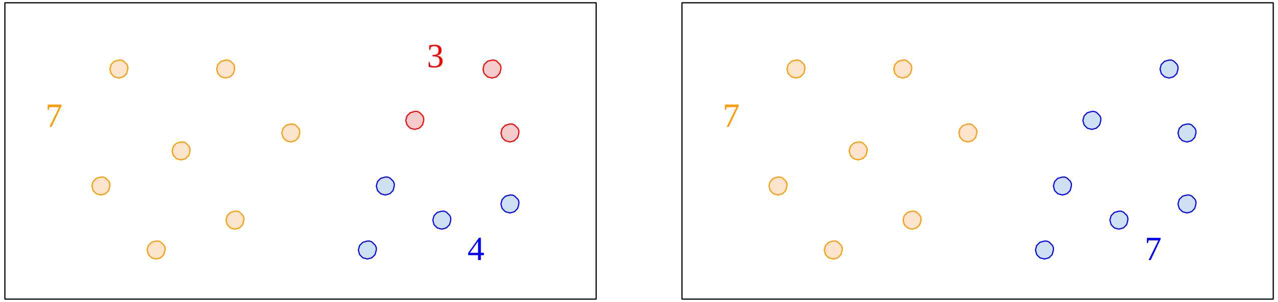
CART for classification

Gini index: most commonly used metric. Approximates entropy but is faster to compute and more numerically stable.

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$



Why use Entropy or Gini over misclassification rate?



$$E_m = 0.5 \qquad E_m = 0.5$$

$$H_m = 1.49 \qquad H_m = 1.0$$

$$G_m = 0.62 \qquad G_m = 0.5$$

Other issues

Categorical variables: encode categorical variables with K possible values using a one-hot encoding $\rightarrow K$ binary variables.

Why binary splits? Multiway splits fragment the data too quickly, leaving less data next level down. Series of binary splits can achieve same as multiway ones.

Missing values: Can use usual techniques like imputing. Can also only introduce new categorical variables for when a value is missing.

Linear combination splits: Instead of using axis aligned splits, we could use linear predictors $\mathbf{w}^T \mathbf{x} \leq s$. Empirically, this doesn't work much better and it is more difficult to optimize.

Summary of CART

	Regression	Classification
Prediction for region R_m	$c_m = \frac{1}{N_m}\sum_{i=1}^{N} y_i \mathbb{1}(\mathbf{x}_i \in R_m)$	$c_m = \max_k \hat{p}_{mk}$
Decision function	$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{1}(\mathbf{x} \in R_m)$	$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{1}(\mathbf{x} \in R_m)$
Impurity measures	SE: $E_m = \sum_{x_i \in R_m} (y_i - c_m)^2$	Gini: $G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
Search	Greedy top down	
Stopping	Depth threshold, region size threshold, pruning	

Advantages of decision trees

Fast to train: Splitting the root node requires searching through D dimensions and N possible values. However, if each split roughly divides data in half, then N reduces logarithmically with successive splits.

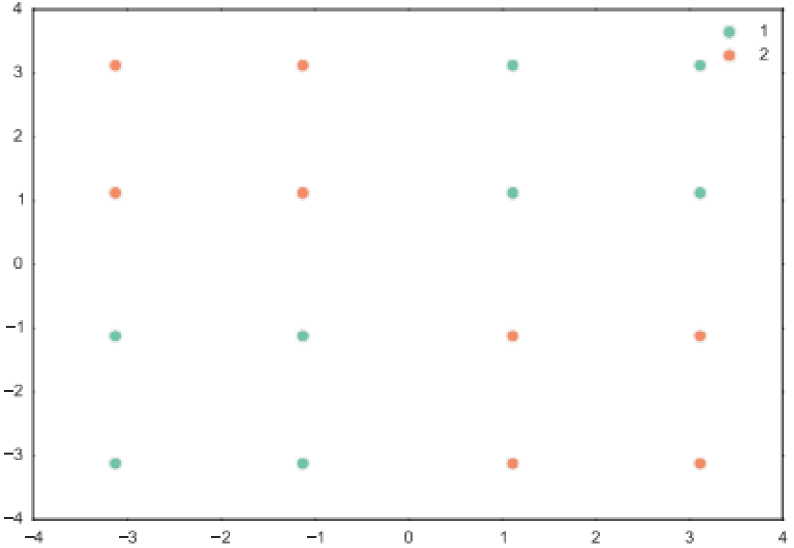
Fast to predict: Prediction simply involves traversing the tree from the root to a leaf node, following the rules as you descent the tree. Each rule is a comparison with a single feature.

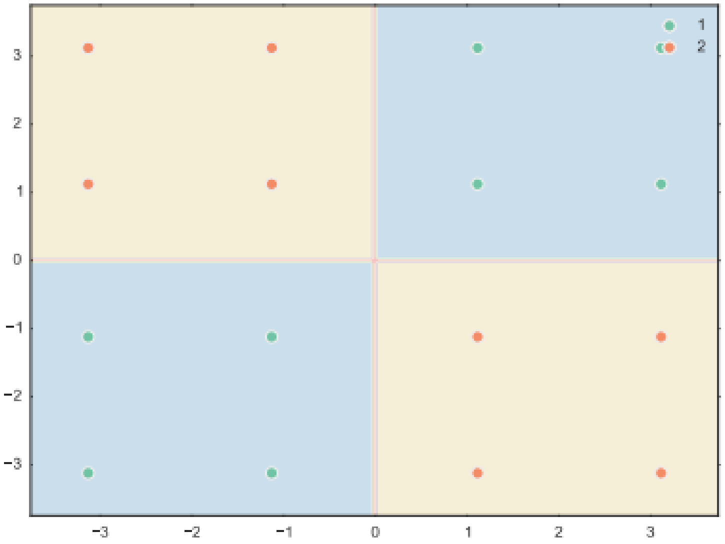
Interpretable: Easy to visualize and interpret model. Produces a successive set of rules. Features near root of tree can be though of as being more predictive.

Limitations of decision trees

- Difficulty in breaking symmetry
- Difficulty producing smooth/simple decision boundaries
- Difficulty in capturing additive structure
- High variance

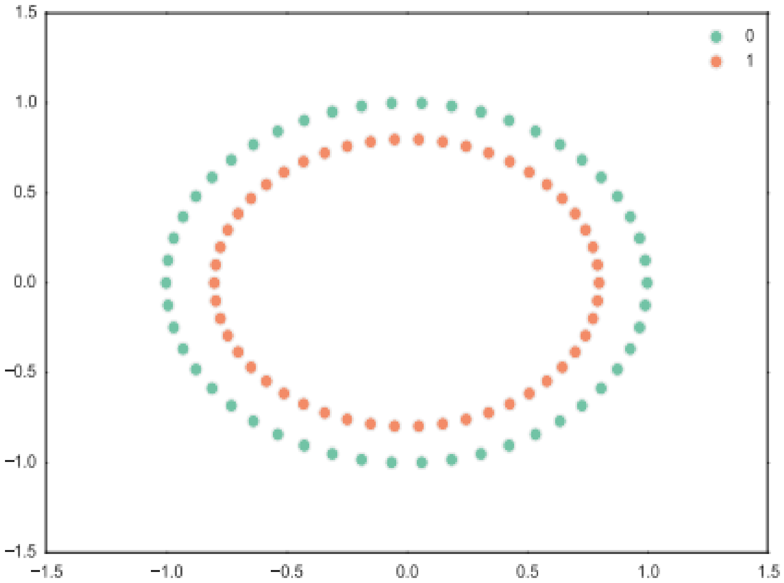
Limitations of decision trees: symmetry breaking

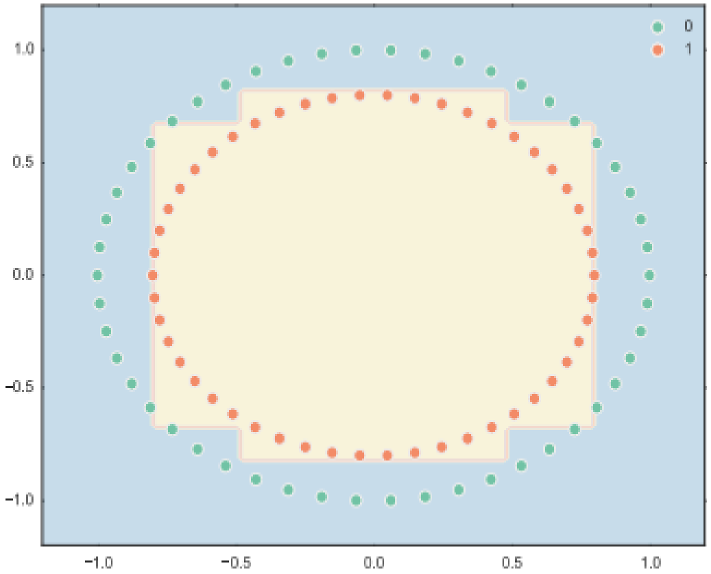




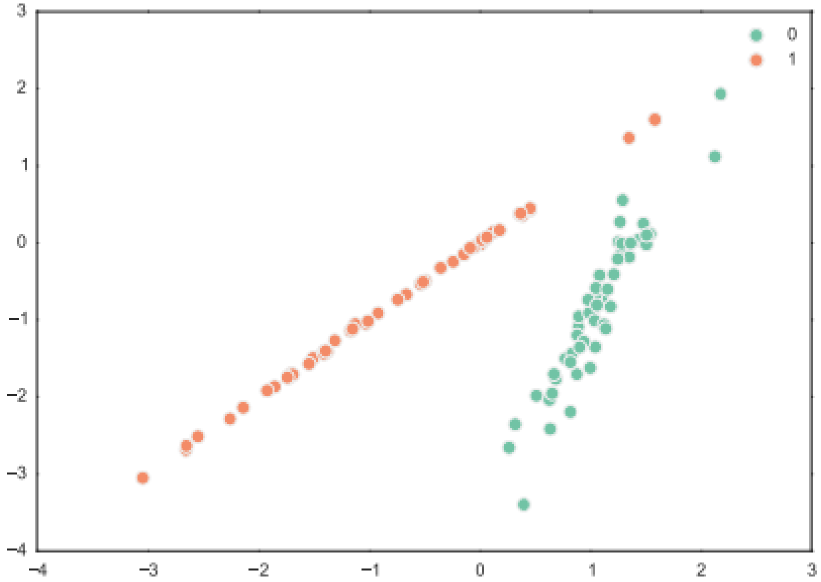
The above is also a good example of why rules to stop splitting if impurity reduction is too small are shortsighted.

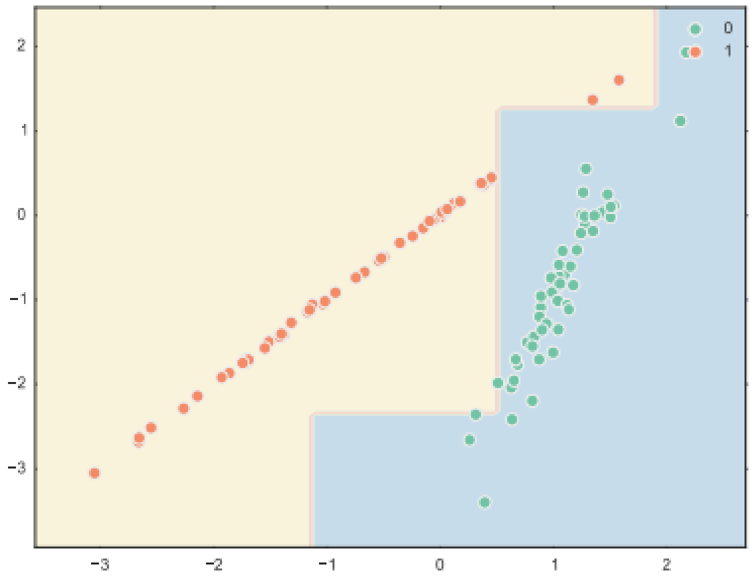
Limitations of decision trees: complex decision boundaries





Limitations of decision trees: complex decision boundaries





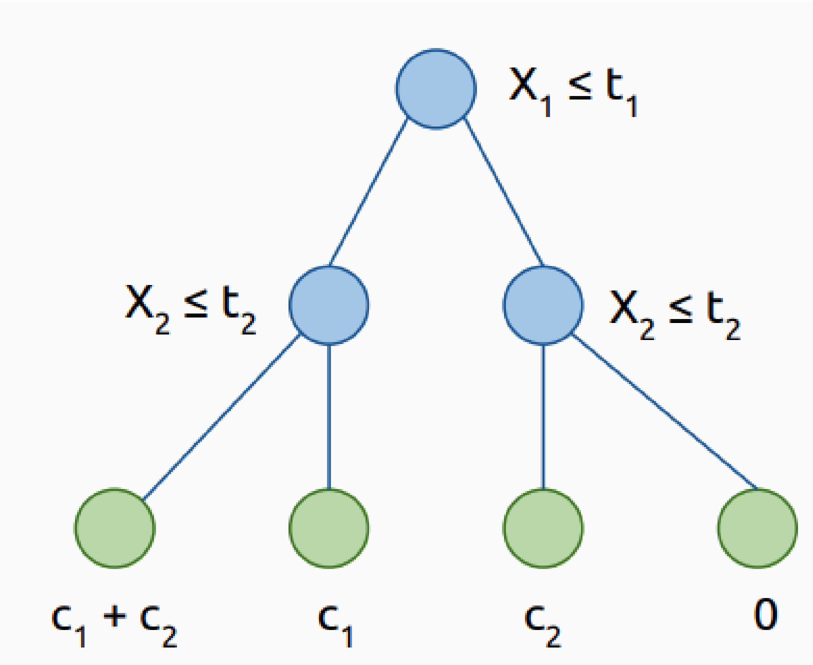
Limitations of decision trees: additive structure

$$y = c_1 \mathbb{1}(X_1 < t_1) + c_2 \mathbb{1}(X_2 < t_2) + c_3 \mathbb{1}(X_3 < t_t) + \ldots + \epsilon$$

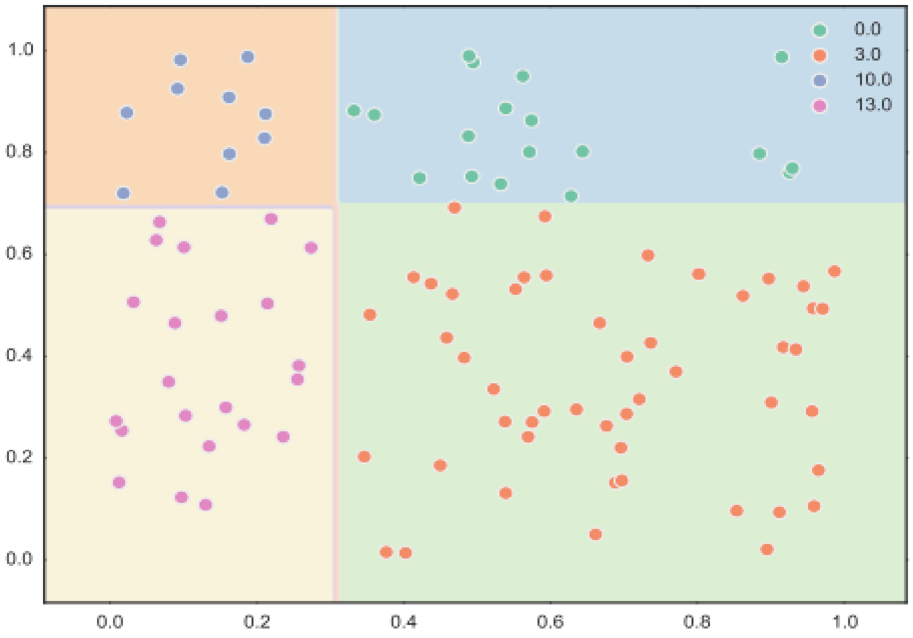
Assume tree first splits near t_1. Then:

- Need 2 splits to capture $c_2 \mathbb{1}(X_2 < t_2)$
- Need 4 splits to capture $c_3 \mathbb{1}(X_3 < t_3)$
- Need 8 splits to capture $c_4 \mathbb{1}(X_4 < t_4)$
- ...

Need **exponentially more parameters** to capture such structure than a classifier that could model the additive structure directly.



$$y = c_1 \mathbb{1}(X_1 < t_1) + c_2 \mathbb{1}(X_2 < t_2) + \epsilon$$



Limitations of decision trees: high variance

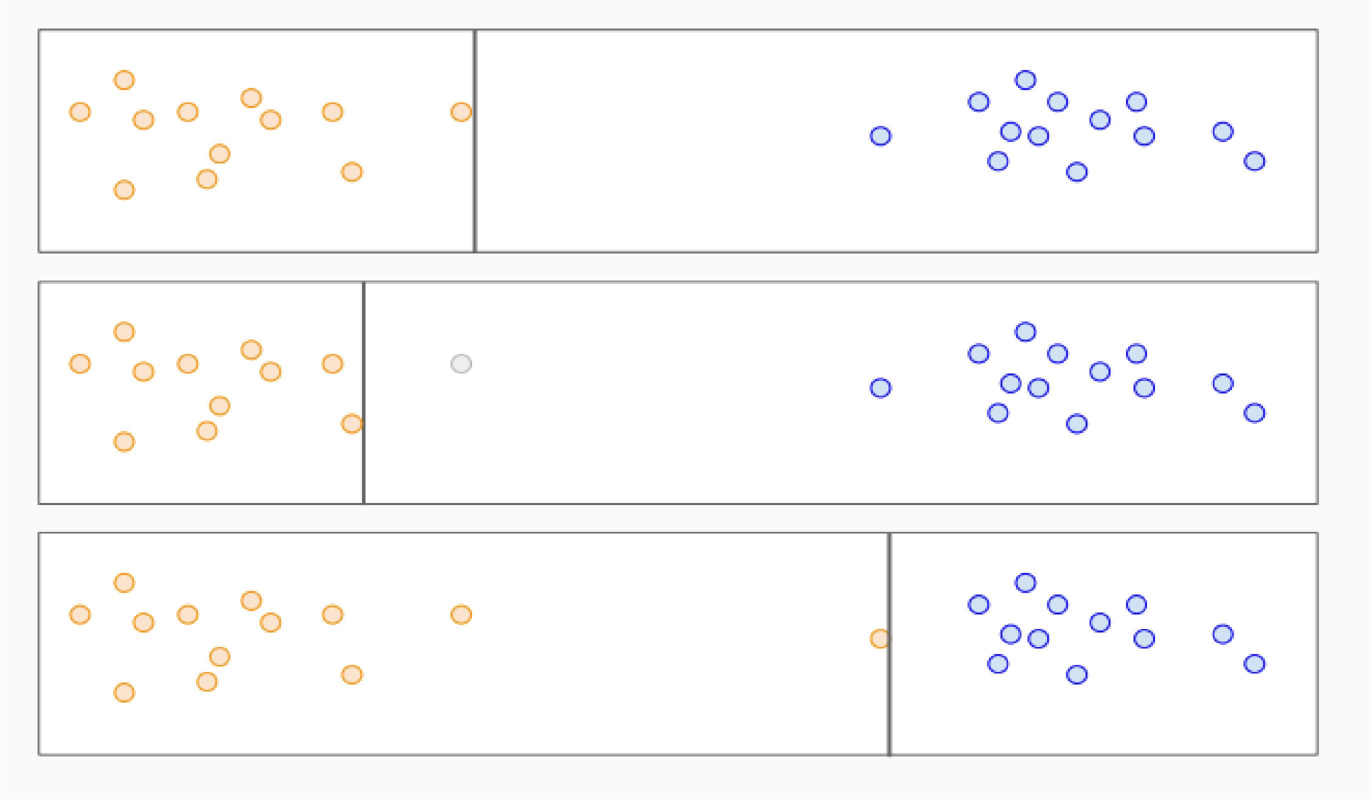
- The main drawback of using decision tree models is **high variance**.

- **Small changes in the data can produce very different trees**. These trees can make very different predictions on data outside of the training set.

- Makes trees very **sensitive to noise**. Trees are **not a robust classifier**.

- Also affects **interpretability** of the model. If small changes to the data cause very different trees, how interpretable are the rules in a given tree?

Limitations of decision trees: high variance



Further reading

The elements of statistical learning:

- Classification and regression trees: Section 9.2

Other resources

Jeff Miller's (mathematicalmonk) on CART: ML 2.1 :
https://www.youtube.com/playlist?list=PLD0F06AA0D2E8FFBA

Nando de Freitas' lectures

- Decision Trees