Classification and Regression Trees

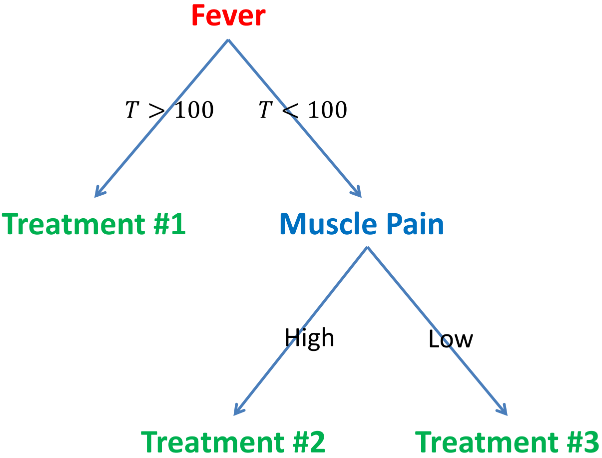
Mihaela van der Schaar

Department of Engineering Science
University of Oxford

March 1, 2017

Many decisions are tree structures

- Medical treatment



Overview

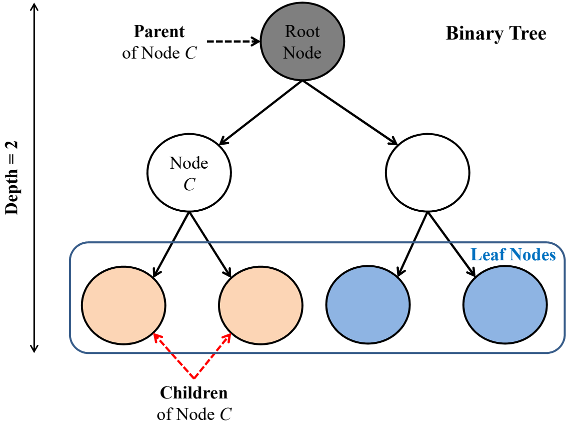
A Decision Tree is a hierarchically organized structure, with each node splitting the data space into pieces based on value of a feature.

- Equivalent to a partition of $\mathbb{R}^d$ into K disjoint feature subspaces $\{\mathcal{R}_1, ..., \mathcal{R}_K\}$, where each $\mathcal{R}_j \subset \mathbb{R}^d$
- On each feature subspace $\mathcal{R}_j$, the same decision/prediction is made for all $x \in \mathcal{R}_j$.

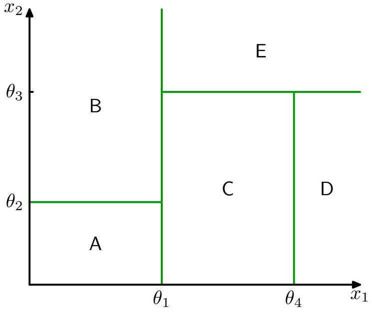
Terminology

- **Parent** of a node c is the immediate predecessor node.
- **Children** of a node c are the immediate successors of c, equivalently nodes which have c as a parent.
- **Root node** is the top node of the tree; the only node without parents.
- **Leaf nodes** are nodes which do not have children.
- **A** K**-ary tree** is a tree where each node (except for leaf nodes) has K children. Usually working with binary trees ($K = 2$).
- **Depth** of a tree is the maximal length of a path from the root node to a leaf node.

Terminology

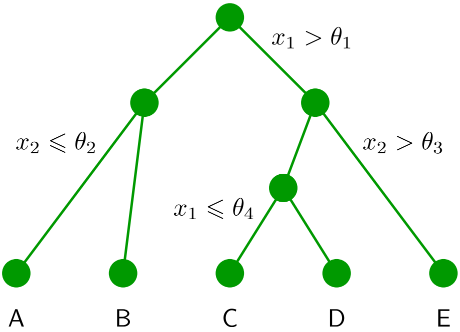


The leaves of a tree partition the feature space



Learning a tree model

- **Three things to learn**
  - The structure of the tree.
  - The threshold values (θ_i)
  - The values for the leafs (A,B,...)



Overview - Classification Tree

- **Classification Tree:**
  - Given the dataset $\mathcal{D} = (x_1, y_1), ..., (x_n, y_n)$ where $x_i \in \mathbb{R}, y_i \in \mathcal{Y} = \{1, ..., m\}$
  - minimize the misclassification error in each leaf
  - the estimated probability of each class k in region $\mathcal{R}_j$ is simply:

$$\hat{\beta}_{jk} = \frac{\sum_i \mathbb{I}(y_i = k) \cdot \mathbb{I}(x_i \in \mathcal{R}_j)}{\sum_i \mathbb{I}(x_i \in \mathcal{R}_j)}$$

  - This is the frequency in which label k occurs in the leaf $\mathcal{R}_j$
  - These estimates can be regularized as well.

Example

- Decide whether to wait for a table at a restaurant, based on the following attributes (Example from Russell and Norvig, AIMA)
  - Alternate: is there an alternative restaurant nearby?
  - Bar: is there a comfortable bar area to wait in?
  - Fri/Sat: is today Friday or Saturday?
  - Hungry: are we hungry?
  - Patrons: number of people in the restaurant (None, Some, Full)
  - Price: price range ($, $$, $$$)
  - Raining: is it raining outside?
  - Reservation: have we made a reservation?
  - Type: kind of restaurant (French, Italian, Thai, Burger)
  - Wait Estimate: estimated waiting time (0-10, 10-30, 30-60, >60)

A tree model for deciding where to eat

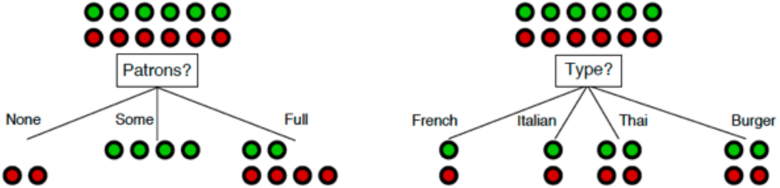
- Examples described by **attributes values** (Binary, discrete, continuous)

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of examples is positive (T) or negative (F)

First decision: at the root of the tree

- **Which attribute to split?**



- **patrons?** is a better choice - gives **information about the classification**
- **Idea:** use **information gain** to choose which attribute to split

Information gain

- A chosen attribute A divides the training set E into subsets $E_1, \dots, E_v$ according to their values for A, where A has v distinct values.

$$\text{remainder}(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} \times I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- Information Gain (IG) or reduction from the attribute test:

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder}(A)$$

- Choose the attribute with the largest $IG(A)$

How to measure information gain?

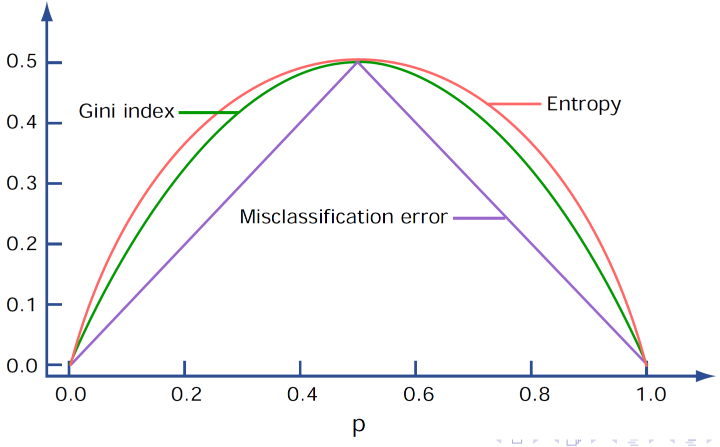
- **Idea: Gaining information reduces uncertainty**
- Use to entropy to measure uncertainty
  - If a random variable X has K different values, $(a_1, ..., a_K)$ its entropy is given by

$$H[X] = -\sum_{k=1}^{K} \mathcal{P}(X = a_k) \times \log \mathcal{P}(X = a_k)$$

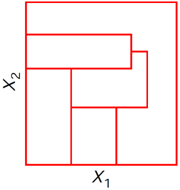
- Different measures of uncertainty:
  - **Misclassification error:** $1 - \max\{\mathcal{P}(X = a_k)\}$
  - **GINI Index:** $\sum_{k=1}^{K} 2\mathcal{P}(X = a_k)(1 - \mathcal{P}(X = a_k))$
- **C4.5 Tree:** Classification tree uses entropy to measure uncertainty.
- **CART:** Classification tree uses Gini index to measure uncertainty.

Learning trees

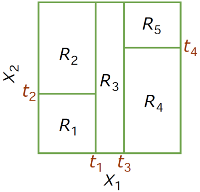
- Uncertainty measurement for two-class classification (as a function of the proportional p in class 1.)



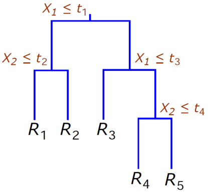
Partitions and CART



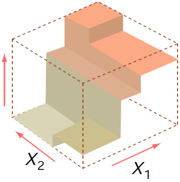
(a) General partition that cannot be obtained from recursive binary splitting.



(b) Partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data.



(c) Tree corresponding to the partition in the top right panel.

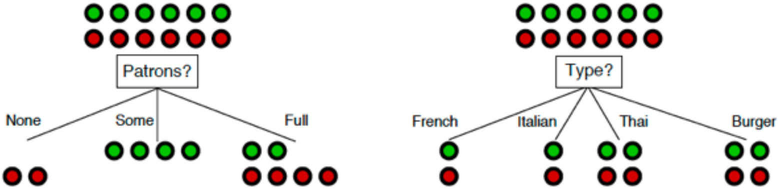


(d) A perspective plot of the prediction surface.

Algorithm for Classification Trees

- Start with $\mathcal{R}_1 = \mathbb{R}^d$
- For each feature $j = 1, ..., d$, for each value $v \in \mathbb{R}$ that we can split on:
  - Split the data set:
  $$I_< = \{i : x_{ij} < v\} \text{ and } I_> = \{i : x_{ij} \geq v\}$$
  - Estimate the parameters:
  $$\beta_< = \frac{\sum_i \mathbb{I}(y_i = 1) \cdot \mathbb{I}(x_i \in I_<)}{\sum_i \mathbb{I}(x_i \in I_<)} \text{ and } \beta_> = \frac{\sum_i \mathbb{I}(y_i = 1) \cdot \mathbb{I}(x_i \in I_>)}{\sum_i \mathbb{I}(x_i \in I_>)}$$
  - Quality of split is measured by the weighted sum of the uncertainty:
  $$\frac{|I_<|}{|I_<| + |I_>|} \times I(\beta_<, 1 - \beta_<) + \frac{|I_>|}{|I_<| + |I_>|} \times I(\beta_>, 1 - \beta_>)$$
- Choose split with minimal weighted sum of the uncertainty.
- Recurse on both children, with $(x_i, y_i)_{i \in I_<}$ and $(x_i, y_i)_{i \in I_>}$.

Which attribute to split?



- **Patron vs. Type?**
  - By choosing **Patron**, we end up with a partition (3 branches) with smaller entropy, i.e. smaller uncertainty **(0.45 bit)**
  - By choosing **Type**, we end up with uncertainty of **1 bit**.
  - Thus, we choose **Patron** over **Type**.

Uncertainty if we go with "Patron"

- **For "None" branch**

$$-(\frac{0}{0+2}\log\frac{0}{0+2}+\frac{2}{0+2}\log\frac{2}{0+2})=0$$

- **For "Some" branch**

$$-(\frac{4}{4+0}\log\frac{4}{4+0}+\frac{0}{4+0}\log\frac{0}{4+0})=0$$

- **For "Full" branch**

$$-(\frac{2}{2+4}\log\frac{2}{2+4}+\frac{4}{2+4}\log\frac{4}{2+4})\approx 0.9$$

- **For choosing "Patrons"**
- weighted average of each branch: this quantity is called **conditional entropy**

$$\frac{2}{12}\times 0+\frac{4}{12}\times 0+\frac{6}{12}\times 0.9=0.45$$

Conditional entropy

- **Definition**. Given two random variables X and Y

$$H[Y|X] = \sum_k \mathcal{P}(X = a_k) \times H[Y|X = a_k]$$

- **In our example**
  - X: the attribute to be split
  - Y: Wait or not
- **Relation to information gain**

$$\text{Gain} = H[Y] - H[Y|X]$$

  - When $H[Y]$ is fixed, we need only to compare conditional entropy.

Conditional entropy for "Type"

- **For "French" and "Italian" branch**

$$-(\frac{1}{1+1}\log\frac{1}{1+1}+\frac{1}{1+1}\log\frac{1}{1+1})=1$$

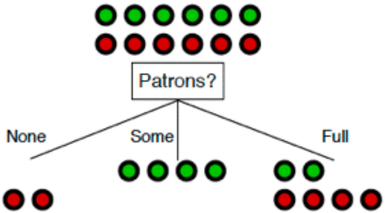
- **For "Thai" and "Burger" branch**

$$-(\frac{2}{2+2}\log\frac{2}{2+2}+\frac{2}{2+2}\log\frac{2}{2+2})=1$$

- **For choosing "Type"**
- weighted average of each branch (**conditional entropy**)

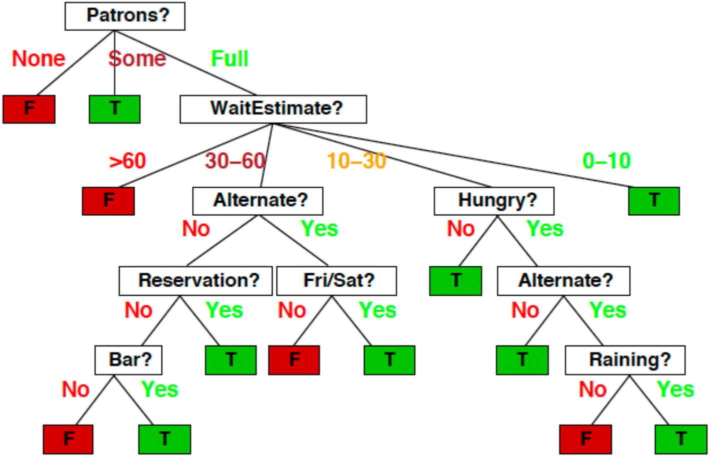
$$\frac{2}{12}\times 1+\frac{2}{12}\times 1+\frac{4}{12}\times 1+\frac{4}{12}\times 1=1$$

Next split?



- Do we split on **"None"** or **"Some"**?
- No, we do not
    - The decision is **deterministic**, as seen from the training data
- We will look only at the 6 instances with **Patrons** == **Full**

Greedily we build the tree and get this

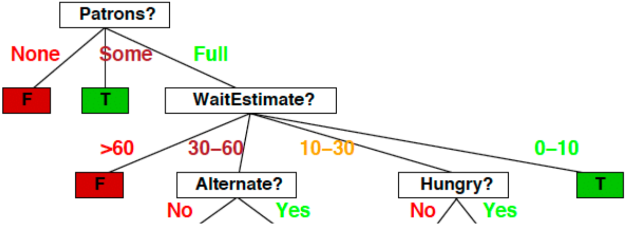


What is the optimal Tree Depth?

- We need to be careful to pick an appropriate **tree depth**.
- If the tree is too **deep**, we can **overfit**.
- If the tree is too **shallow**, we **underfit**
- **Max depth** is a hyper-parameter that should be tuned by the data.
- Alternative strategy is to create a very deep tree, and then to **prune** it.

Control the size of the tree

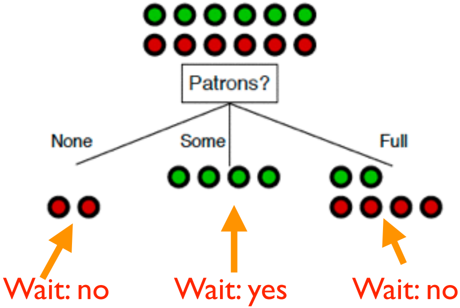
- **We would prune to have a smaller one**.



- If we stop here, **not all training samples** would be classified **correctly**.
- More importantly, how do we classify a **new instance**?
- We label the leaves of this smaller tree with **the majority of training samples' labels**.

Example

- **We stop after the root (first node).**



Computational Consideration

- **Numerical Features**
  - We could split on any **feature**, with any **threshold**.
  - However, for a given feature, the only split points we need to consider are the n **values** in the **training data** for this feature.
  - If we sort each feature by these n values, we can **quickly compute** our uncertainty metric of interest (cross entropy or others)
  - **This takes** $\mathcal{O}(dn \log n)$ **time**
- **Categorical Features**
  - Assuming q distinct categories, there are $2^{q-1} - 1$ **possible binary partitions** we can consider.
  - However, things simplify in the case of binary classification (or regression), and we can find the optimal split (for cross entropy and Gini) by only considering $q - 1$ **possible splits**

Summary of learning classification trees

- **Advantages**
  - Easily interpretable by human (as long as the tree is not too big)
  - Computationally efficient
  - Handles both numerical and categorical data
  - It is parametric thus compact: unlike Nearest Neighborhood Classification, we do not have to carry our training instances around
  - Building block for various ensemble methods (more on this later)
- **Disadvantages**
  - Heuristic training techniques
  - Finding partition of space that minimizes empirical error is NP-hard.
  - We resort to greedy approaches with limited theoretical underpinning.

Overview - Regression Tree

- **Regression Tree:**
  - Given the dataset $\mathcal{D} = (x_1, y_1), ..., (x_n, y_n)$ where $x_i \in \mathbb{R}, y_i \in \mathcal{Y} = \mathbb{R}$
  - minimize the error (e.g. square loss error) in each leaf
  - the parameterized function is

$$\hat{f}(x) = \sum_{j=1}^{K} \beta_j \cdot \mathbb{I}(x \in \mathcal{R}_j)$$

  - Using squared loss, optimal parameters are:

$$\hat{\beta}_j = \frac{\sum_i y_i \cdot \mathbb{I}(x_i \in \mathcal{R}_j)}{\sum_i \mathbb{I}(x_i \in \mathcal{R}_j)}$$

which is sample mean.

Assign the prediction for each leaf

For each leaf, we need to assign the prediction y which minimizes the loss for the regression problem.

- **Regression Tree:**

$$\hat{y} = \arg\min_{y \in \mathbb{R}} \sum_{i \in \mathcal{R}_k} (y - y_i)^2 = \frac{1}{\sum \mathbb{I}(x_i \in \mathcal{R}_k)} \cdot \sum \mathbb{I}(x_i \in \mathcal{R}_k) \cdot y_i$$

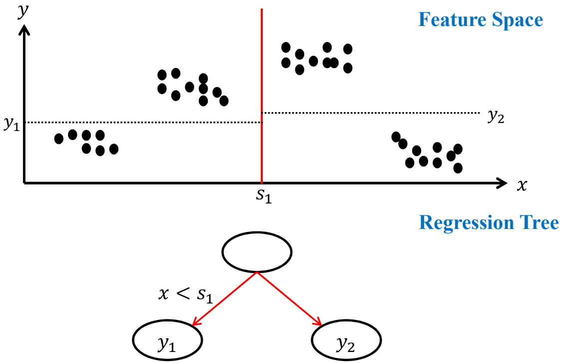
Growing the Tree: Overview

- Ideally, would like to find partition that achieves minimal risk: lowest mean-squared error for regression problem.
- Number of potential partitions is too large to search exhaustively.
- **Greedy** search heuristics for a good partition:
  - Start at root.
  - Determine the best feature and value to split.
  - Recurse on children of node.
  - Stop at some point (with heuristic pruning rules).

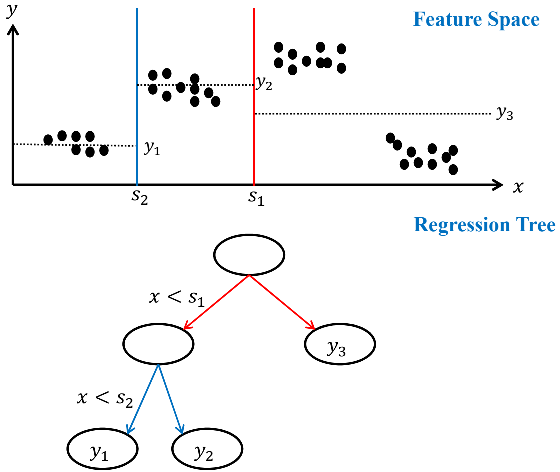
Algorithm for Regression Trees

- Start with $\mathcal{R}_1 = \mathbb{R}^d$
- For each feature $j = 1, ..., d$, for each value $v \in \mathbb{R}$ that we can split on:
  - Split the data set:
  $$I_< = \{i : x_{ij} < v\} \text{ and } I_> = \{i : x_{ij} \geq v\}$$
  - Estimate parameters:
  $$\beta_< = \frac{\sum_{i \in I_<} y_i}{|I_<|} \text{ and } \beta_> = \frac{\sum_{i \in I_>} y_i}{|I_>|}$$
  - Quality of split is measured by the squared loss:
  $$\sum_{i \in I_<} (y_i - \beta_<)^2 + \sum_{i \in I_>} (y_i - \beta_>)^2$$
- Choose split with minimal loss.
- Recurse on both children, with $(x_i, y_i)_{i \in I_<}$ and $(x_i, y_i)_{i \in I_>}$.

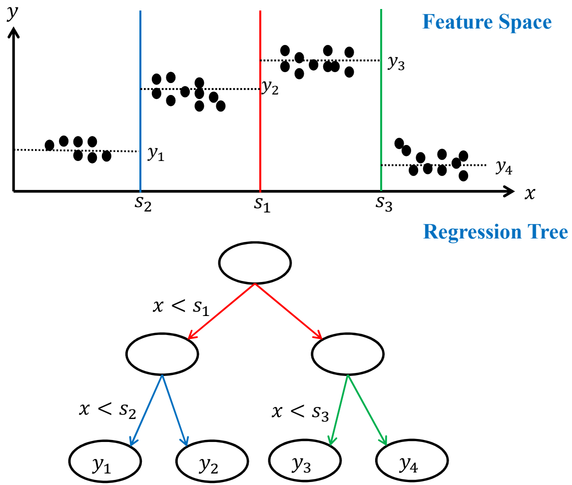
Example of Regression Trees



Example of Regression Trees



Example of Regression Trees



Model Complexity

- When should tree growing be stopped?
- Will need to control complexity to prevent over-fitting, and in general find optimal tree size with best predictive performance.
- Consider a regularized objective

$$\text{Traing Error}(Tree) + C \times \text{Size}(Tree)$$

  - Grow the tree from scratch and stop once the criterion objective starts to increase.
  - First grow the full tree and prune nodes (starting at leaves), until the objective starts to increase.
- Second option is preferred as the choice of tree is less sensitive to **wrong** choices of split points and variables to split on in the first stages of tree fitting.
- Use cross validation to determine optimal C.

Pruning Rule

- Stop when one instance in each leaf (regression problem)
- Stop when all the instance in each leaf have the same label (classification problem)
- Stop when the number of leaves is less than the threshold
- Stop when the leaf's error is less than the threshold
- Stop when the number of instances in each leaf is less than the threshold
- Stop when the p-value between two divided leafs is larger than the certain threshold (e.g. 0.05 or 0.01) based on chosen statistical tests.

Overview

- Decision Tree can be applied in **general applications** (e.g. medical applications/recommendation systems).
- We focus on **medical applications**.

1. **Neurosurgery**
   - **Aim:** To recommend certain type of neurosurgery to patients who have higher probability to exhibit the clinical improvement.
2. **Heart Transplant**
   - **Aim:** To match the best available heart to the patient whose survival probability is maximized.

Pruning Rule: Stop when the p-value between two divided leaves is larger than the certain threshold (0.05) based on the student t-test.

Student t-test for the pruning rule of decision tree

- Student t-test is a statistical hypothesis test used to determine whether two sets of data are statistically different from each other.
- Usually, it is used to test the null hypothesis that the means of two populations are equal.
- Therefore, when pruning decision trees, the Student t-test is used to determine whether the two resulting partitions (the populations of different leaves) have statistically different means.
- It is usually determined by the p-value (less than 0.05 or 0.01) computed by the Student t-test.

Details of Student t-test

- For the unpaired datasets and unequal variance setting,
- Mean of group A:

$$\bar{y}_A = \frac{1}{N_A} \sum y_i \cdot \mathbb{I}(X_i \in A)$$

- Variance of group A:

$$\bar{s}_A^2 = \frac{1}{N_A - 1} \sum (y_i - \bar{y}_A)^2 \cdot \mathbb{I}(X_i \in A)$$

.

- Therefore, t-value can be computed as follow

$$t = \frac{\bar{y}_A - \bar{y}_B}{\sqrt{\frac{\bar{s}_A^2}{N_A} + \frac{\bar{s}_B^2}{N_B}}}$$

- Based on the computed t-value, the p-value is computed as.

$$\text{p-value} = \mathcal{P}(Z > |t|) \text{ where } Z \sim \mathcal{N}(0, 1)$$

Neurosurgery: Block Diagram

Patient information

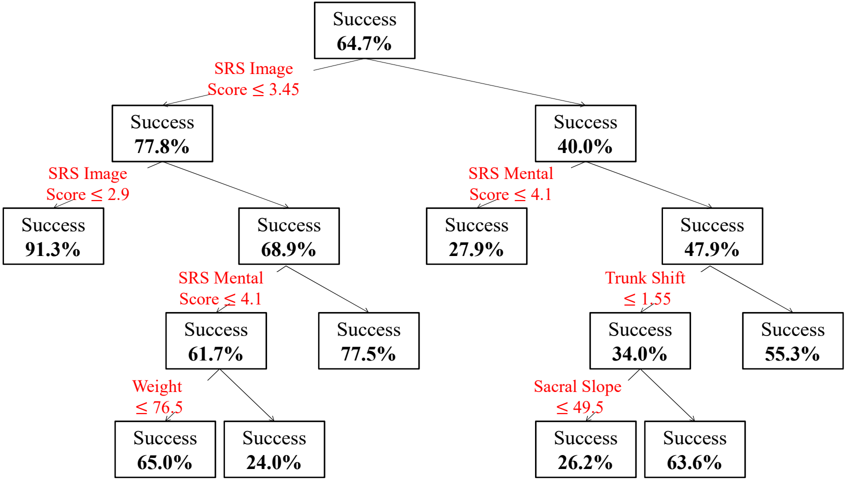
Predictive model (Decision Tree)

Prediction for the success probabilty of **Neurosurgery**

Recommend **Neurosurgery** only for certain patient groups

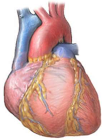
Type	Explanation	Note
Patient	1,449 patients with neurosurgery	2 year follow-up
Feature	91 Features (61 Continuous / 30 Binary)	
Label	MCID 1: 938 Patients (64.7%) MCID 0: 511 Patients (35.3%)	

Neurosurgery: Achieved Decision Tree



Heart Transplant: Block Diagram

Patients / Donors Information



→ **Predictive model (Decision Tree)** → Prediction for the success of **heart transplant**

↓

Optimally **match** patients and donors to minimize mortality

Type	Explanation	Note
Patient	56,716 patients (heart transplant patients)	follow-up until they died
Feature	141 Features (84 Continuous / 57 Binary)	From 1986 to 2015
Label	Dead: 16,986 Patients (29.95%) Alive: 39,730 Patients (70.05%)	

Heart Transplant: Achieved Decision Tree