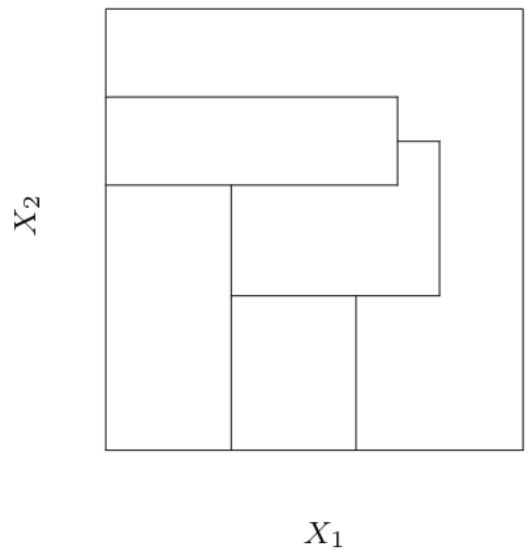
# Decision Trees

Decision tress can be applied to both **regression** and **classification** problems, which makes it very flexible and interpretable.

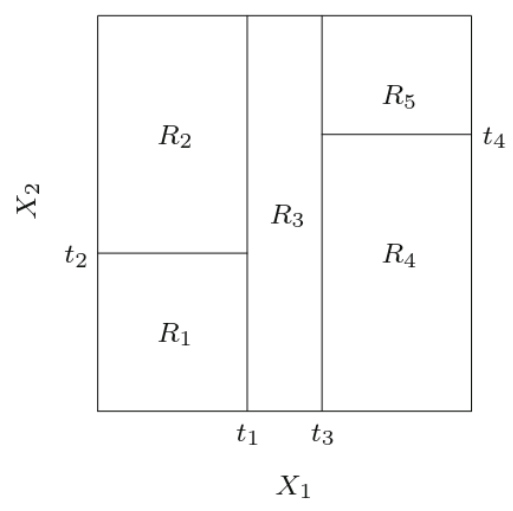
## CART

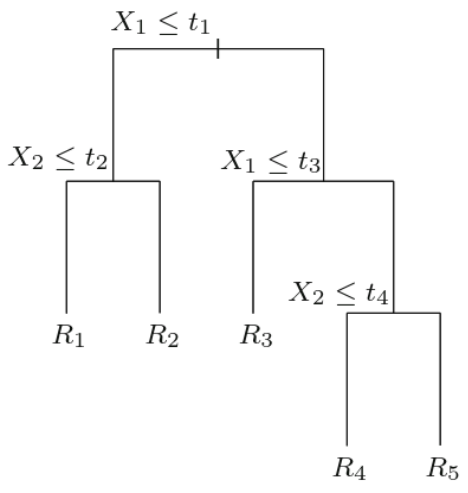
Stands for **Classification and Regression Trees** which is an (algorithm/method) for tree based methods.

Considering a response $Y$ and predictor $X_1, X_2$




By restricting it to **recursive binary partitions** by splitting the region into two and model the response by the mean of $Y$




# Regression Trees

To construct a **regression tress**, the algorithm needs to automatically decide on splitting variables(feature) and splitting point $s$ also what **shape** the tree should have.

$$y = \sum_{m=1}^{M} c_m I(x \in R_m)$$

- This models the response as a constant $c_m$ in each region $R_m$.
  Using the Residual Sum of Squares :

$$\sum_{i=1}(y_i - \sum^{M} c_m I(x \in R_m))^2$$

For one **Region** we get :

$$\mathcal{L}(c) = \sum^{n} (y_i - c)^2$$

Deriving w.r.t. $c$ :

$$\frac{d\,\mathcal{L}}{d\,c} = \sum^{n} 2(c - y_i)$$

Setting it to zero results in:

$$\hat{c} = \frac{1}{N_m} \sum_{i=1}^{n} y_i \equiv \hat{c}_m = \text{ave}(y_i | x_i \in R_m)$$

**Note**:

- $\sum_{i \in R_m} c = N_m . c$
- The constant $\hat{c}$ represent the mean of $\bar{y}$ on that region $m$
- $\text{avg}(y_i | x_i \in R_m)$ means the average of $y_i$ given that $x_i$ is in the region $m$

To find the best binary partition in terms of minimum sum of squares is computationally infeasible. Hence **regression trees** use a $\text{greedy algorithm}$.
At a given node we consider all possible splits $(j, s)$ by :

- Consider all features $p$ of $X$ given by $X_j$
  - For every possible threshold $s$
  - We evaluate the $R_1(j, s) = \{X | X_j \leq s\}$ and $R_2(j, s) = X | X_j > s$
- Repeat for each **feature** resulting in pairs $(j, s) \rightarrow (\text{feature,split point})$
  **ex** : $(age, 50), (age, 40), (height, 170), (height, 167) \dots$

Then we seek the **splitting variable** $j$ and **split point** $s$ that (minimize/solves) this :

$$\min_{j,s} \left[ \min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

The inner minimization is solved by:

$$\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s)) \text{ and } \hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s))$$

The $\text{greedy algorithm}$ Summary :

- Select splits which are pairs $(j, s)$
- Calculate the constant for the split $c_1, c_2$
- Evaluate the split by calculating the Residual Sum of Squares
- Choose the split $(j, s)$ that yields the smallest $\text{RSS}$

The question now is how large should we grow the tree? , very large tree might overfit the data easily while small might not learn the data and the underline structure.

## Tree Pruning

The size of tree is a **tuning parameter** which correspond to the model complexity, the **greedy** strategy is to grow a large tree $T_0$ and then stop growing after the minimum **node size** is reached($4$ observations per region).

The large tree $T_0$ is pruned using $\text{cost-complexity prunning}$ :

- The number of Observations in a region $m$ is denoted :

$$N_m = \#\{x_i \in R_m\},$$

- The constant for region $m$ is denoted :

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

- The **MSE** for region $m$ is denoted :

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$$

The **Cost complexity criterion** :

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

With :

- $|T|$ the number of terminal nodes(leafs)
- $\alpha$ penalty parameter

The idea is to find for each $\alpha$ a subtree $T_\alpha \subseteq T_0$ that minimize $C_\alpha(T)$,$\alpha \geq 0$ results in a tradeoff between the tree size and it's **goodness of fit**.

- Large values results in smaller trees $T_\alpha$
- $\alpha = 0$ results in $T_0$

## Weakest Link Pruning

To find $T_\alpha$ that minimize $C_\alpha(T)$ we use **weakest link pruning**:

First the starting point for the pruning is not $T_0$ , but rather $T_1 = T(0)$ which is the smallest subtree of $T_0$ that satisfy:

$$R(T_1) = R(T_0)$$

With $R(T) = \sum^{|T|} N_m Q_m(T)$

To Obtain $T_1$ First, we look at $T_0$ the biggest tree and for any **two terminal nodes(leafs)** from the same parent node, if we sum the error rate and it's the same as their parent node, we prune off these two terminal nodes :

$$R(t) = T(t_L) + R(t_R)$$

- Parent node $t$
- Two terminal nodes $t_L$ and $t_R$

This process is applied recursively. which results in a pruned $T_0$ while having the same error rate.

```
T_0:          T₁ (after pruning):
    A               A
   / \             / \
  B   C           B   C
 / \
D   E
```

- Since $R(B) = R(D) + R(E)$
- Making a prediction using the region $B$ or in $D$ and $E$ will results in the same error rate
- The **child nodes** doesn't provide any improvements over their **parents**

The **weakest link** method not only find the next $\alpha$ which results in different optimal subtree, but find that optimal subtree