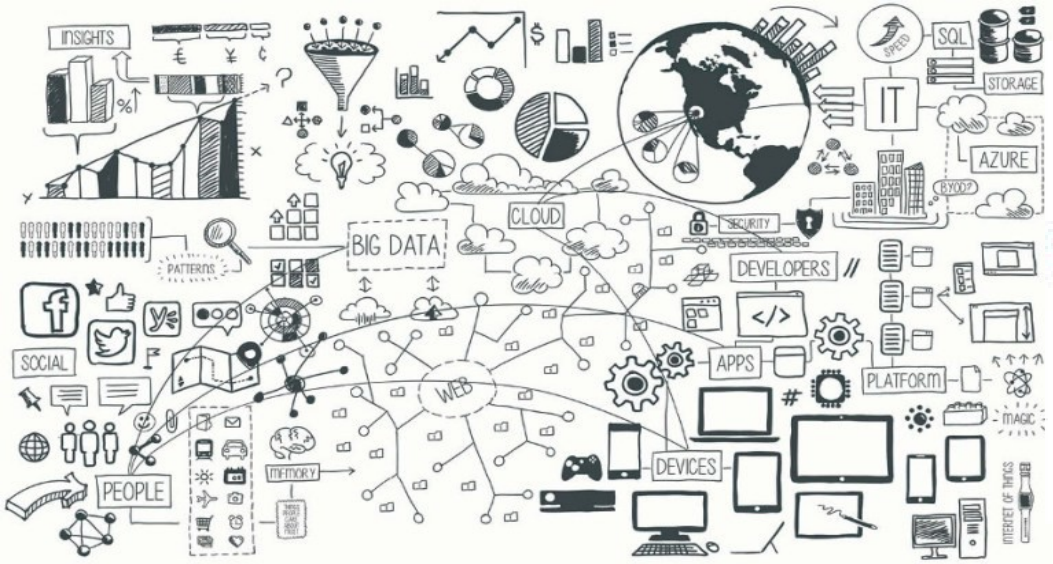
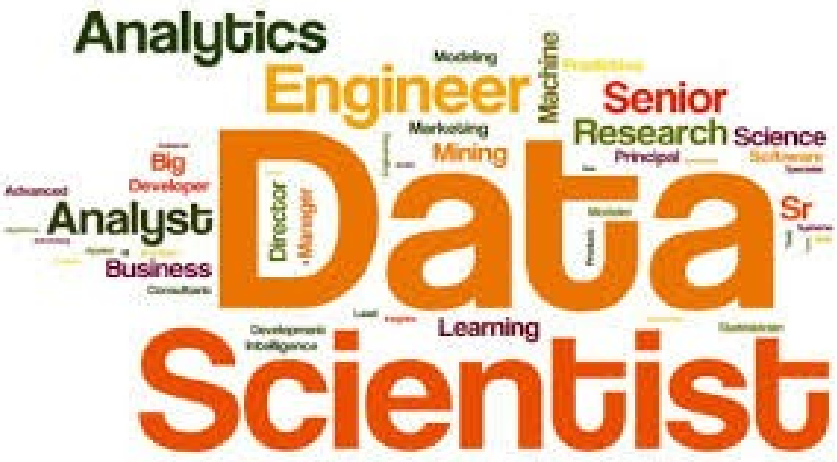
# Data Mining (Minería de Datos)

# Regression Trees

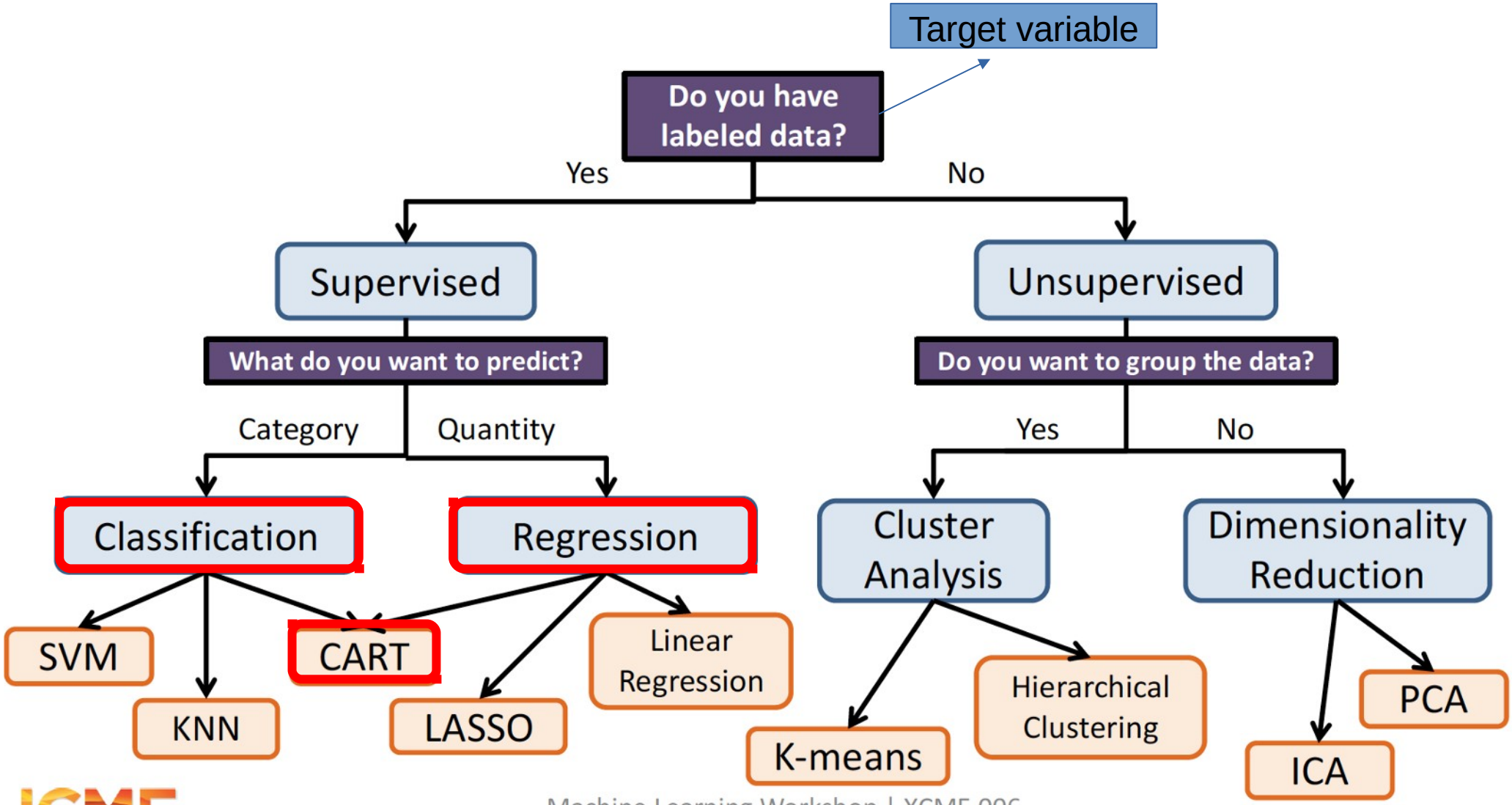


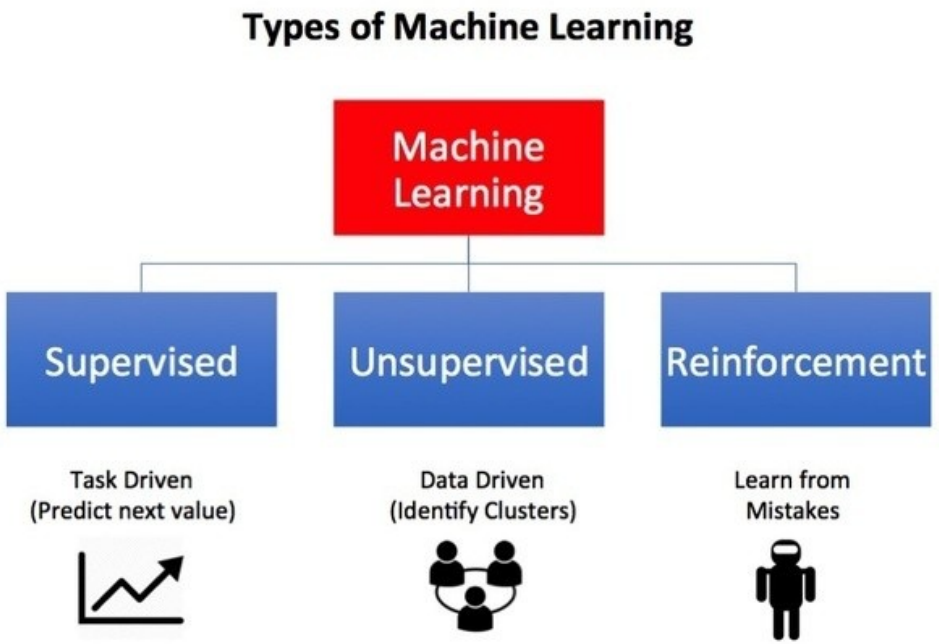




**Sixto Herrera, Rodrigo Manzanas**

**Grupo de Meteorología**

**Univ. de Cantabria – CSIC**
**MACC / IFCA**


Target variable
Do you have labeled data?
Yes
No
Supervised
Unsupervised
What do you want to predict?
Do you want to group the data?
Category
Quantity
Yes
No
Classification
Regression
Cluster Analysis
Dimensionality Reduction
SVM
KNN
CART
LASSO
Linear Regression
K-means
Hierarchical Clustering
ICA
PCA




NOTA: Las líneas de código R en esta presentación se muestran sobre un fondo gris

| | | |
|---|---|---|
| Nov | 3 | Presentación, introducción y perspectiva histórica |
| | 5 | Paradigmas, problemas canonicos y data challenges |
| | 10 | Reglas de asociación |
| | 12 | Practica: Reglas de asociación |
| | 17 | Evaluación, sobrejuste y crossvalidacion |
| | 19 | Practica: Crossvalidacion |
| | 24 | Árboles de clasificación y decisión |
| | 26 | Practica: Árboles de clasificación |
| | | **T01. Datos discretos** |
| Dic | 1 | Técnicas de vecinos cercano (k-NN) |
| | 3 | Práctica: Vecinos cercanos |
| | 10 | Comparación de Técnicas de Clasificación. |
| | 15 | Reducción de dimensión no lineal |
| | | **T02. Clasificación** |
| | 16 | **Árboles de clasificación y regresión (CART)** |
| | 18 | Práctica: Árboles de clasificación y regresion (CART) |
| | 22 | Practica: El paquete CARET |
| | | **T03. Prediccion** |
| Ene | 12 | Ensembles: Bagging and Boosting |
| | 14 | Random Forests y Gradient boosting |
| | 15 | Técnicas de agrupamiento |
| | 21 | Técnicas de agrupamiento |
| | 25 | Predicción Condicionada |
| | 27 | Sesión de refuerzo/repaso. |
| | 28 | Examen |

## Classification Trees

**Aim:**
To classify a **categorical** target variable (R factor) based on a set of **categorical or continuous** predictors.

## Regression Trees

**Aim:**
To predict a **continuous** target variable based on a set of **categorical or continuous** predictors.

**Structure:**
- Each **node** corresponds to a test on an **attribute**
- Each **branch** corresponds to an **attribute value**
- Each **leaf** (terminal node) represents a **final set of observations**
- Each **path** is a conjunction of attribute values

**Key Points:**
- Due to their intuitive representation, they are **easy to assimilate** by humans
- They can be constructed **relatively fast** as compared to other methods
- In general, they work **fine**

# Tree construction

There are several algorithms to build up the tree. However, the idea of all of them is the same: evaluate attribute according to its **power of separation**. For **Classification Trees** we have seen:

## ID3 → Information Gain (IG) & Entropy (H)

$$IG(X) = H(X) - H(X|Y)$$

$$\begin{cases} H(X) = -\sum_{X} p(x)\, log_2(p(x)) \\ H(X|Y) = -\sum_{X}\sum_{Y} p(x,y)\, log_2(p(x|y)) \end{cases}$$

## C4.5 → Information Gain (IG) & Gain Ratio (GR)

$$GR = -\frac{IG}{Info}$$

$$Info = -\sum_{i} \frac{|p_i|}{N}\, log_2 \frac{|p_i|}{N}$$

# Tree construction

There are several algorithms to build up the tree. However, the idea of all of them is the same: evaluate attribute according to its **power of separation**. For **Classification Trees** we have seen:

## ID3 → Information Gain (IG) & Entropy (H)

$$IG(X) = H(X) - H(X|Y)$$

$$\begin{cases} H(X) = -\sum_{X} p(x)\, log_2(p(x)) \\ H(X|Y) = -\sum_{X}\sum_{Y} p(x,y)\, log_2(p(x|y)) \end{cases}$$

## C4.5 → Information Gain (IG) & Gain Ratio (GR)

$$GR = -\frac{IG}{Info}$$

$$Info = -\sum_{i} \frac{|p_i|}{N}\, log_2 \frac{|p_i|}{N}$$

**What is the measure considered for Regression Trees?**

## Tree construction

There are several algorithms to build up the tree. However, the idea of all of them is the same: evaluate attribute according to its **power of separation**. For **Classification Trees** we have seen:

### ID3 → Information Gain (IG) & Entropy (H)

$$IG(X) = H(X) - H(X|Y)$$

$$\begin{cases} H(X) = -\sum_{X} p(x)\, log_2(p(x)) \\ H(X|Y) = -\sum_{X}\sum_{Y} p(x,y)\, log_2(p(x|y)) \end{cases}$$

### C4.5 → Information Gain (IG) & Gain Ratio (GR)

$$GR = -\frac{IG}{Info}$$

$$Info = -\sum_{i} \frac{|p_i|}{N}\, log_2 \frac{|p_i|}{N}$$

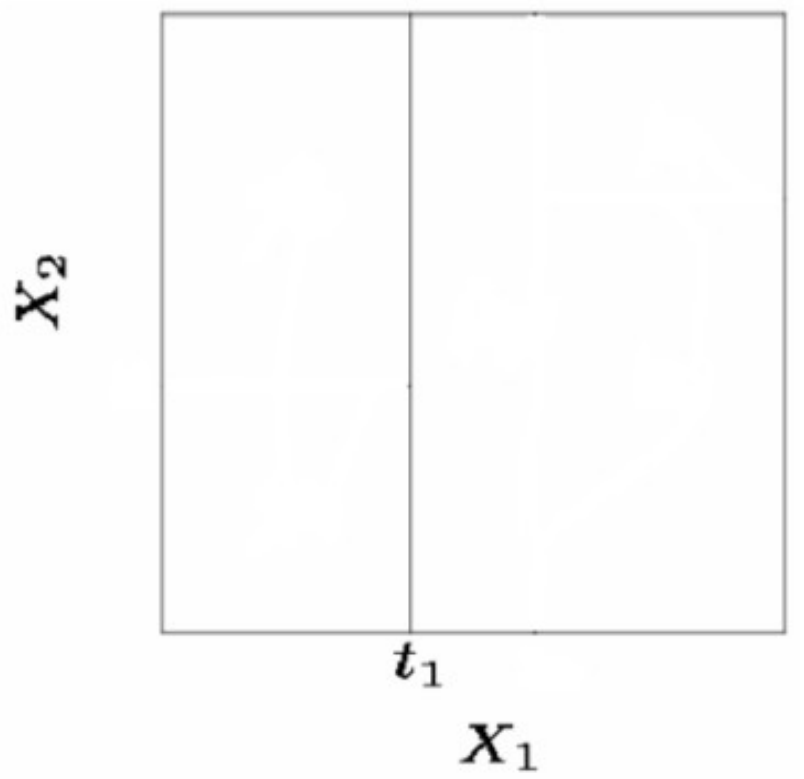
**What is the measure considered for Regression Trees?**

**Residual Sum of Squares (RSS)** ⟶

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

In CART, we divide the predictor space by iteratively splitting one of the ***X*** variables into two regions.

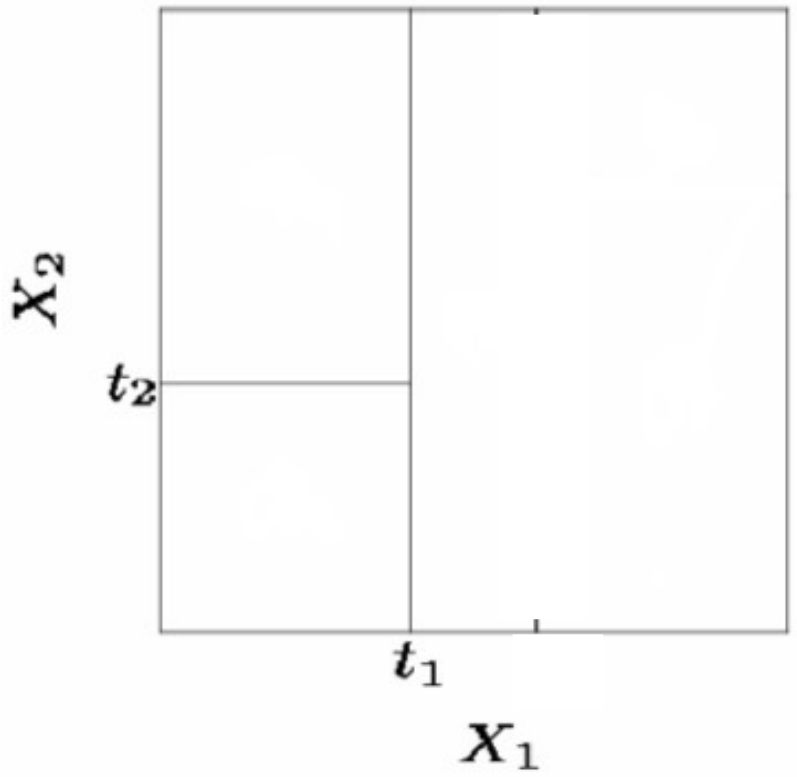
First split on: $X_1 = t_1$



In CART, we divide the predictor space by iteratively splitting one of the $X$ variables into two regions.

First split on: $X_1=t_1$

If $X_1<t_1$ split on: $X_2=t_2$

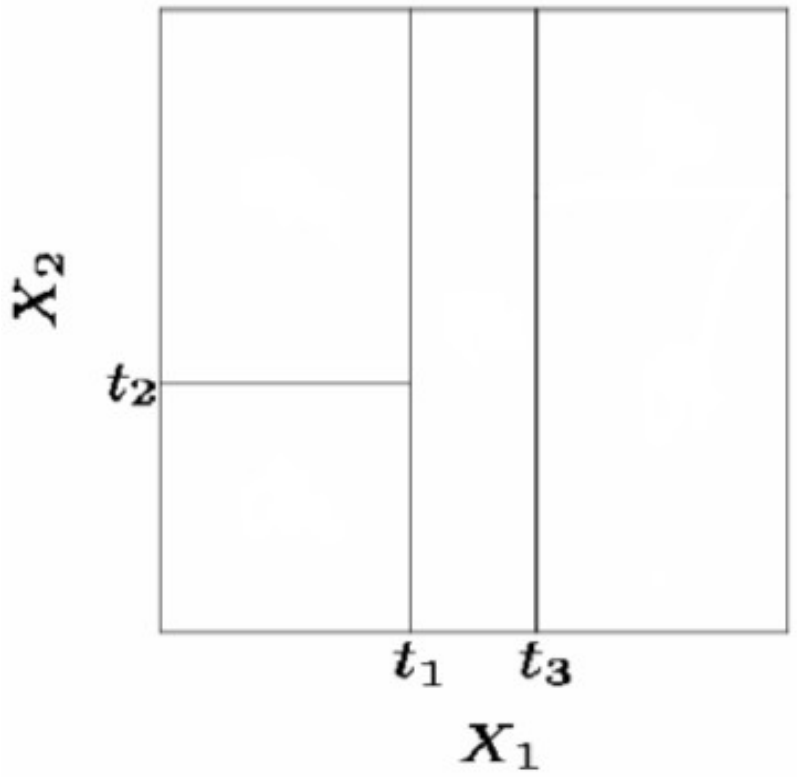


In CART, we divide the predictor space by iteratively splitting one of the X variables into two regions.

First split on: $X_1=t_1$

If $X_1<t_1$ split on: $X_2=t_2$

If $X_1>=t_1$ split on: $X_1=t_3$



In CART, we divide the predictor space by iteratively splitting one of the X variables into two regions.

First split on: $X_1=t_1$

If $X_1<t_1$ split on: $X_2=t_2$

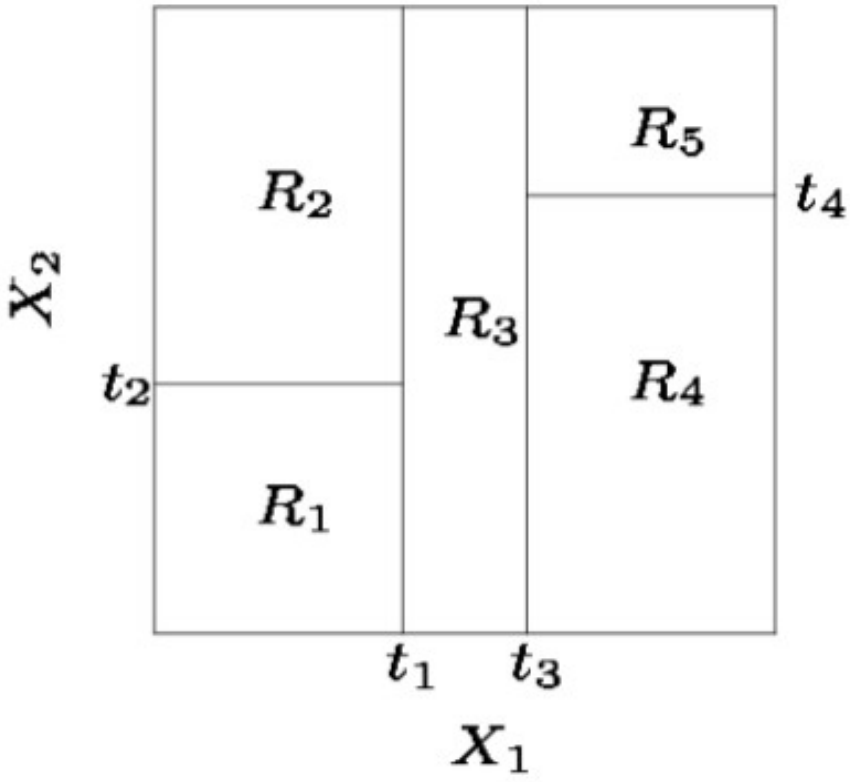
If $X_1>=t_1$ split on: $X_1=t_3$

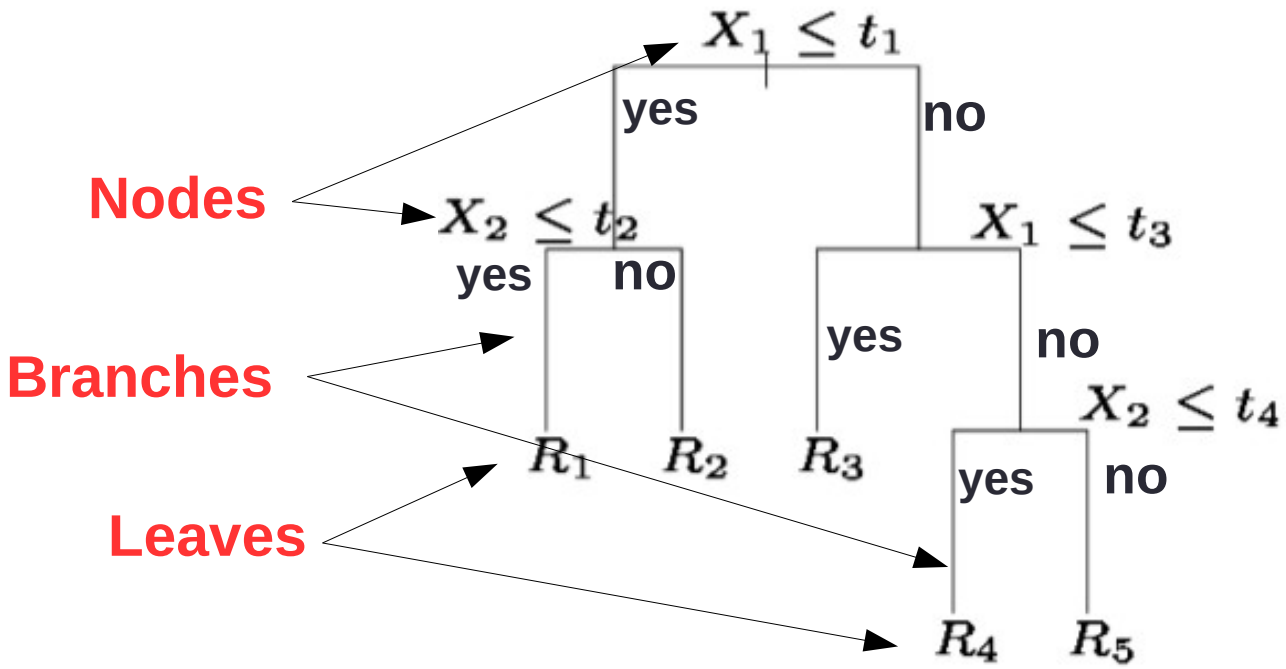
If $X_1>=t_3$ split on: $X_2=t_4$

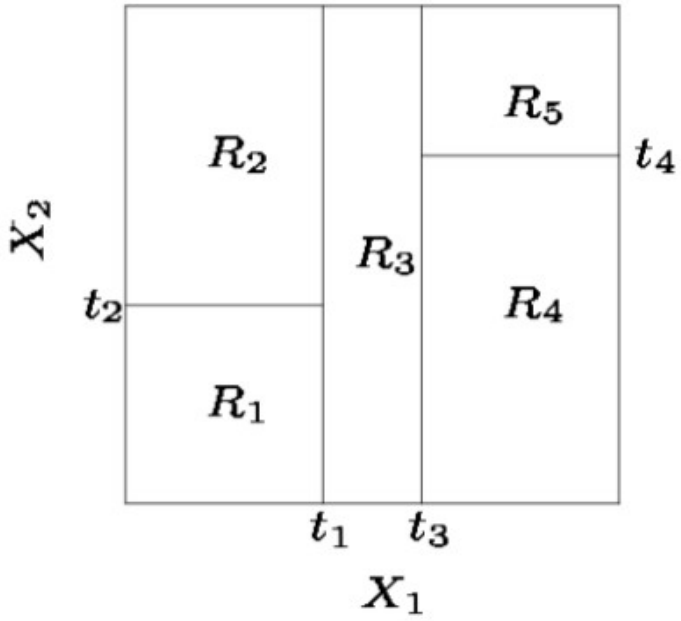
…

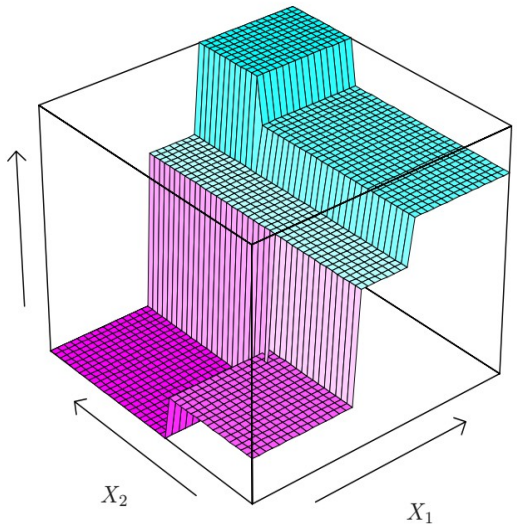
until:

- Error reduction too small vs. complexity
- Max. depth (or other restriction/s) is/are reached



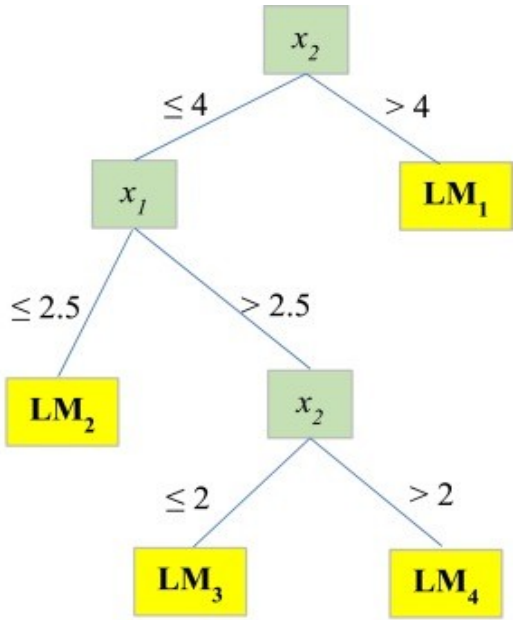

Nodes
Branches
Leaves
$X_1 \leq t_1$
yes
no
$X_2 \leq t_2$
$X_1 \leq t_3$
yes
no
yes
no
$X_2 \leq t_4$
yes
no
$R_1$
$R_2$
$R_3$
$R_4$
$R_5$

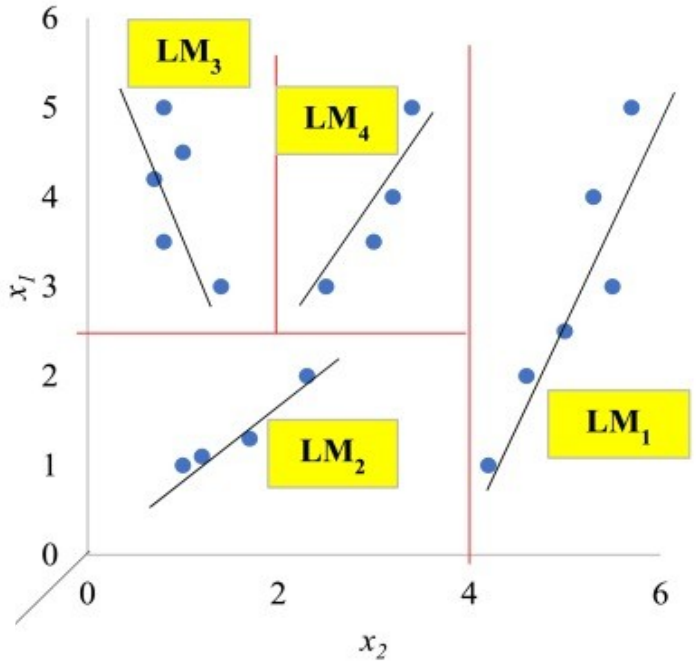

$R_5$
$R_2$
$t_4$
$X_2$
$R_3$
$t_2$
$R_4$
$R_1$
$t_1$
$t_3$
$X_1$


$X_2$
$X_1$

## Types of regression trees:

- **CART:** Output corresponds to the **mean** value within the leaf
- **M5, M5':** Output corresponds to a **linear regression** model of the instances that reach the leaf




M5 trees are included in caret through M5 and cubist methods:

```
library(caret)
if (!require(Cubist)) install.packages("Cubist")
? models
autoTree <- train(form = medv~., data= Boston, subset = indTrain, method = "cubist")
summary(autoTree)
```

# Regularization: Cost Complexity Pruning

Tree score $\longrightarrow$ $$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

**The objective is to  select a tree that leads to the lowest test error rate**

# Regularization: Cost Complexity Pruning

Tree score → $\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$ ← **Number of terminal nodes**

**Controls a trade-off between the tree's complexity and its fit to the training data. The most complex tree (0 error) would correspond to $\alpha$=0**

For each value of the regularization parameter $\alpha$ exists a tree that minimizes the expression above

↓

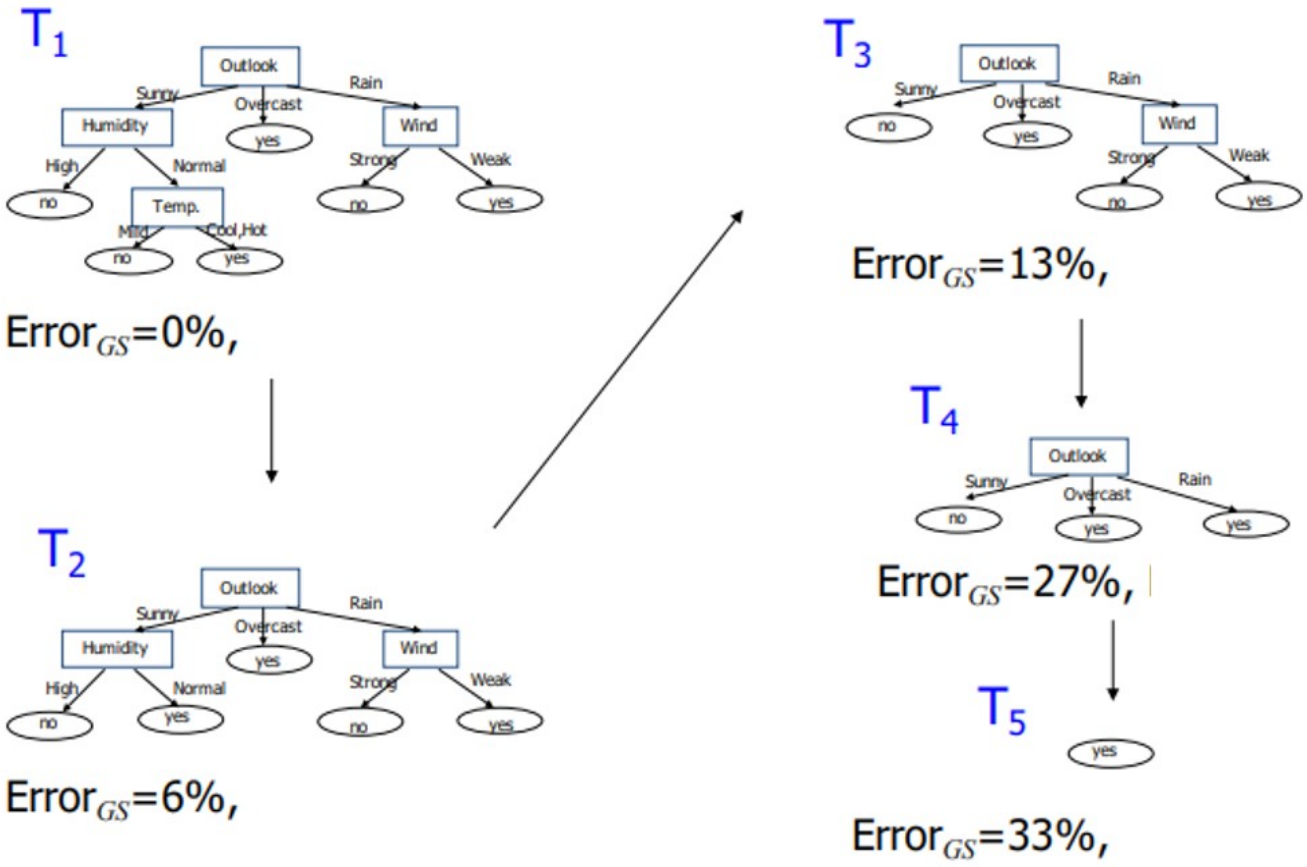
Cross-validation

**The objective is to select a tree that leads to the lowest test error rate**

# Regularization: Cost Complexity Pruning

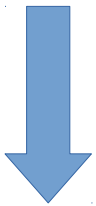
**Post-pruning:** allow the tree to overfit and then, once finalized, remove the less useful nodes. In general, this is the preferred option.
*Procedure:*
Compute a sequence of trees {T1, T2, …} where T1 is the complete tree. T2 is obtained by removing from T1 the node that less increases the error. This process is guided based on some **cost-complexity** criterion

The question is: **where to stop?** In practice, it is usual to split the learning dataset into two subsets: a **training sample** for growing the tree and a **validation sample** for evaluating its generalization error (e.g. hold-out **cross-validation**).



- **Pros of CART:**
  - Trees are very easy to explain to people (probably even easier than linear regression).
  - Trees can be plotted graphically, and are easily interpreted even by non-experts.
  - Trees can easily handle qualitative (e.g. discrete) predictors without the need to create dummy variables.
  - In general, they work fine for both classification and regression problems.
- **Cons of CART:**
  - Trees don't have the same prediction accuracy as some of the more complicated approaches that we see in this course.

***By aggregating many CARTs, the predictive performance of individual trees can be substantially improved***

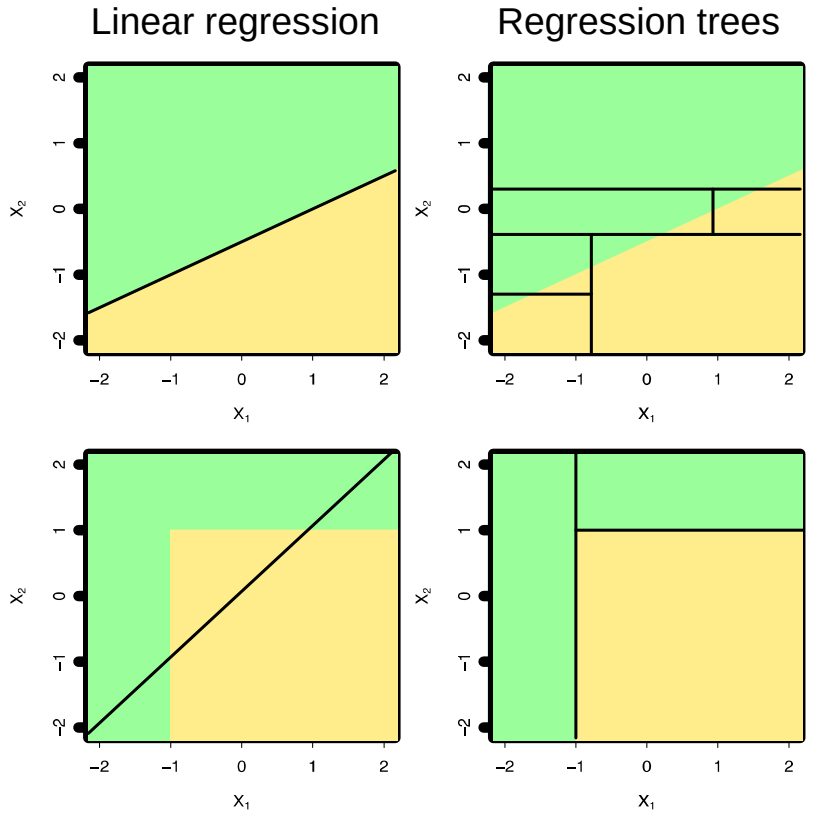


***Ensemble methods (e.g. random forests)***

# Linear Regression or Regression Trees: Which is better?

On the one hand, if the relationship between the predictors and response is linear, then classical linear models such as linear regression would outperform regression trees

On the other hand, if the relationship between the predictors and the response is non-linear, then decision trees would outperform classical linear models



```
library(MASS)
?Boston
```

# Housing values in the suburbs of Boston

**Description:**

The Boston data frame has 506 rows and 14 columns.

**Format:**

This data frame contains the following columns:

**crim** - per capita crime rate by town.
**zn** - proportion of residential land zoned for lots over 25000 sq.ft.
**indus** - proportion of non-retail business acres per town.
**chas** - Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
**nox** - nitrogen oxides concentration (parts per 10 million).
**rm** - average number of rooms per dwelling.
**age** - proportion of owner-occupied units built prior to 1940.
**dis** - weighted mean of distances to five Boston employment centres.
**rad** - index of accessibility to radial highways.
**tax** - full-value property-tax rate per 10000 $.
**ptratio** - pupil-teacher ratio by town.
**black** - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
**lstat** - lower status of the population (percent).
**medv** - median value of owner-occupied homes in 1000s $.

## Ejercicios prácticos (dataset *Boston*):

1) Divide el dataset en dos mitades aleatorias, una para training y otra para test

2) Nuestra variable objetivo será *medv* (precio de las viviendas). Construye un árbol usando la función *tree*, con los parámetros que vienen por defecto. Represéntalo. ¿Cuántas hojas tiene? ¿Cuántos predictores entran en juego? ¿Cuáles de ellos con los más importantes?

3) Construye otro árbol más sencillo que sólo utilice los dos predictores con mayor poder explicativo. Represéntalo. ¿Cuántas hojas tiene el árbol? Representa también la partición llevada a cabo en el espacio de predictores en este caso (función *partition.tree*)

4) Utilizando los parámetros adecuados en la función *tree*, construye ahora el árbol completo. Represéntalo. ¿Cuántas hojas tiene? ¿Cuántos predictores entran en juego?

5) Utiliza el árbol por defecto y el completo para predecir el precio de las viviendas tanto en el train como en el test. Valida estas predicciones en función del RMSE. ¿Qué conclusión obtienes?

6) Echa un vistazo a la variabilidad de las predicciones generadas con el árbol por defecto y con el completo. ¿Qué se observa?

7) Aplica una cross-validación con 5 folds sobre el árbol completo para hallar el número de hojas óptimo. A continuación, construye dicho árbol

8) Utiliza el árbol óptimo para predecir en el test. Valida estas predicciones en función del RMSE y el ratio de varianzas. ¿Qué podrías decir al comparar con los resultados que obtuviste para el árbol por defecto?

9) Entrena un nuevo árbol utilizando el método *cubist* de *caret* (con los parámetros por defecto) y utílizalo para predecir en el test. Analiza estas predicciones, ¿qué se puede decir con respecto a las mismas?