CLASSIFICATION AND REGRESSION TREES

Alejandro Ordonez

Assistant Professor - Department of Bioscience

Section for Ecoinformatics & Biodiversity

Center for Biodiversity Dynamics in a Changing World (BIOCHANGE)

CLASSIFICATION
WHAT WE WILL TALK ABOUT TODAY

- Classification as a regression – Using data to constrain the clustering
- Clarification and Regression trees (CART)– Use a single tree to generate perditions.
- Random Forests (RF)– Use many trees and bagging techniques to improve predictions.
- Boosted Regression Trees(RF)– Use many trees and boosting techniques to improve predictions.

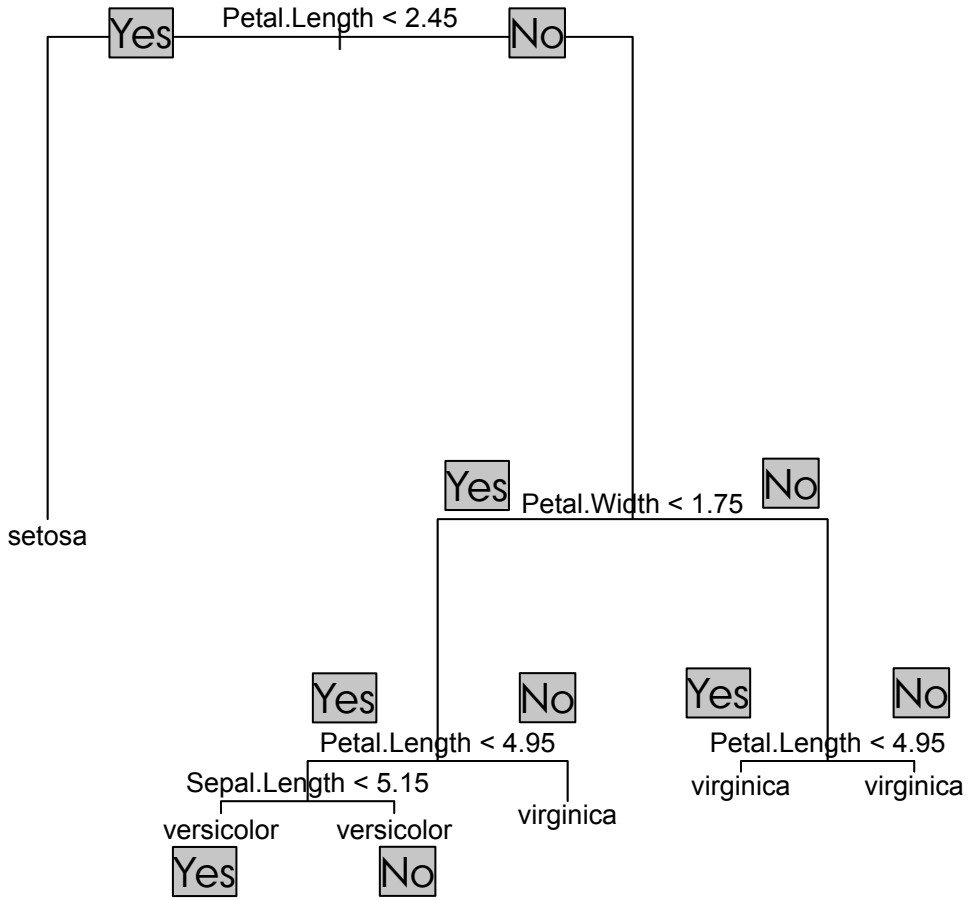
Any questions?

Ready to start?

TREE MODELS

These are...

computationally intensive methods based on the ***recursive partition of the response variable*** *based on the information contained in the predictors.*



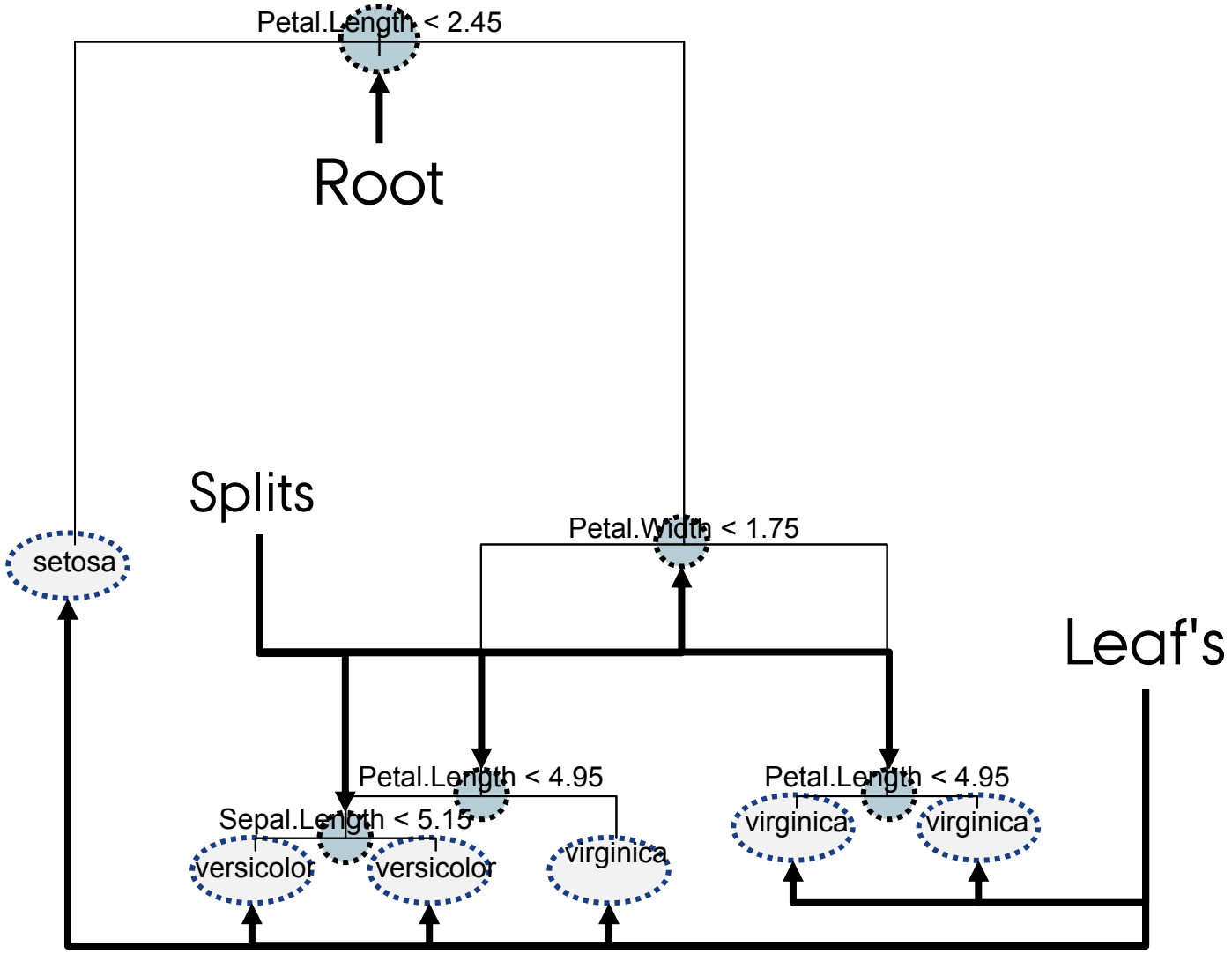
WHY USE TREE MODELS?

- They are simple to interpret
- Great for initial data inspection.
- Give a very clear picture of the structure of the data.
- Provide a highly intuitive insight into the kinds of interactions between variables.
- Require little to no data preparation.
- Great to handle non-linearities.

But they are black boxes

we know that a variable has an effect but not what is the effect

PARTS OF A TREE MODEL



So far so good?

Any questions?

Ready to move on?

(MULTIVARIATE) REGRESSION TREES

- Alternative non-linear /non-parametric models to describe how a response variable changes as a function of a set of continuous/categorical predictors.
- These approaches are particularly useful when the data has lots of features which interact in complicated, nonlinear ways.
- Regression trees are analogous to multiple regression models in the context that they use multiple predictors variables to generate a prediction.

(MULTIVARIATE) REGRESSION TREES

- The computation of a (M)RT consists in two procedures running together:
  - **Constrained partitioning of the data:**
    - Divining the response variables into yes or no conditions based on the predictors using a **binary recursive partitioning** approach.

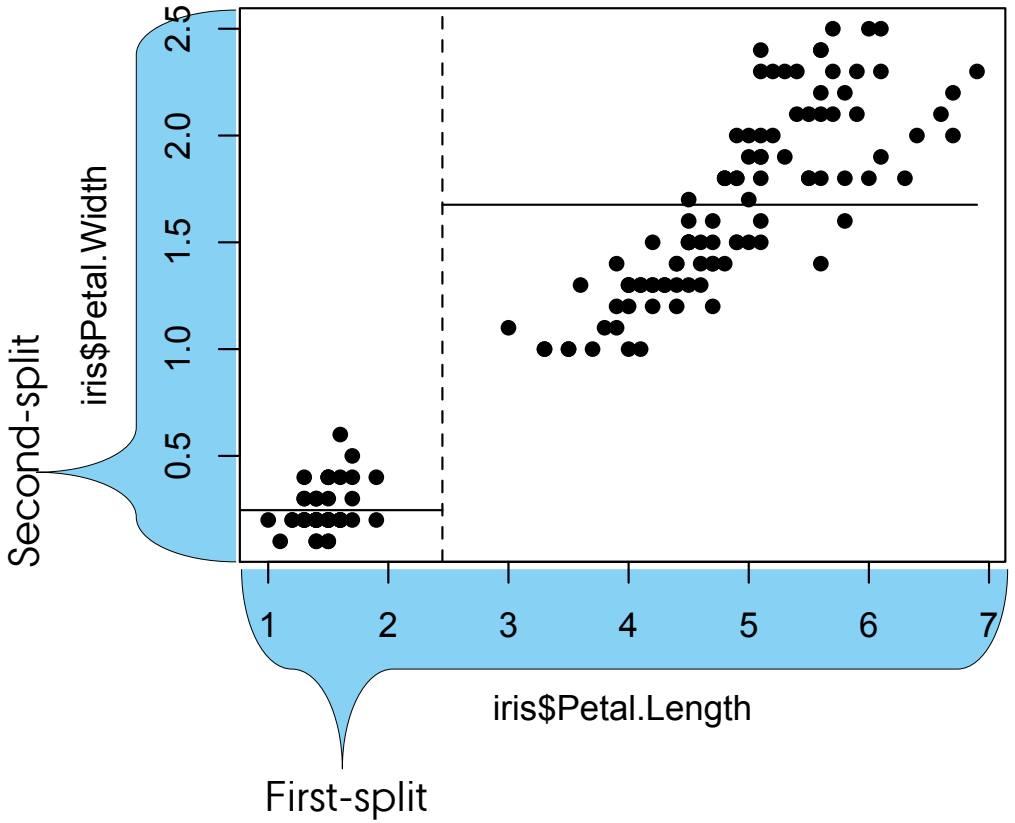
  - **Cross-validation of the results:**
    - Assess how good is the split using a subset of the dataset to assess the **deviance** or the **cross-validated relative error** as "quality" criteria.

(MULTIVARIATE) REGRESSION TREES

What is binary recursive partitioning?

Split in two and keep the solution that minimises the within group variability

This is done until all objects form their own group or until a preselected smallest number of objects per group is reached.



(MULTIVARIATE) REGRESSION TREES

Splits are based on a measure of "*purity*"

- *purity* = node contains observations primarily from a single class or similar values.

Deviance

$$D = \sum_{j} \left(y_i - \mu_{[j]}\right)^2$$

$\mu_{[j]}$= mean of the response var in node *j*

How much an observation differs from the group mean.

Gini index

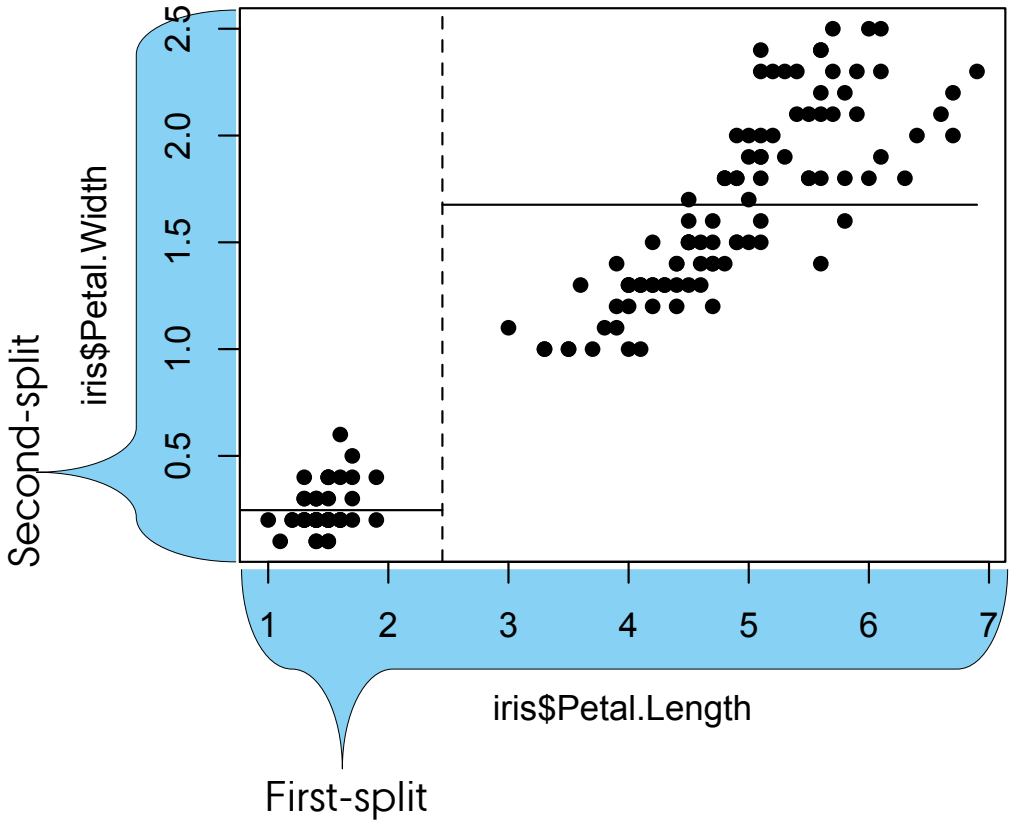
$$G = 1 - \sum_{j} (p_i)^2$$

p_i= proportion of class

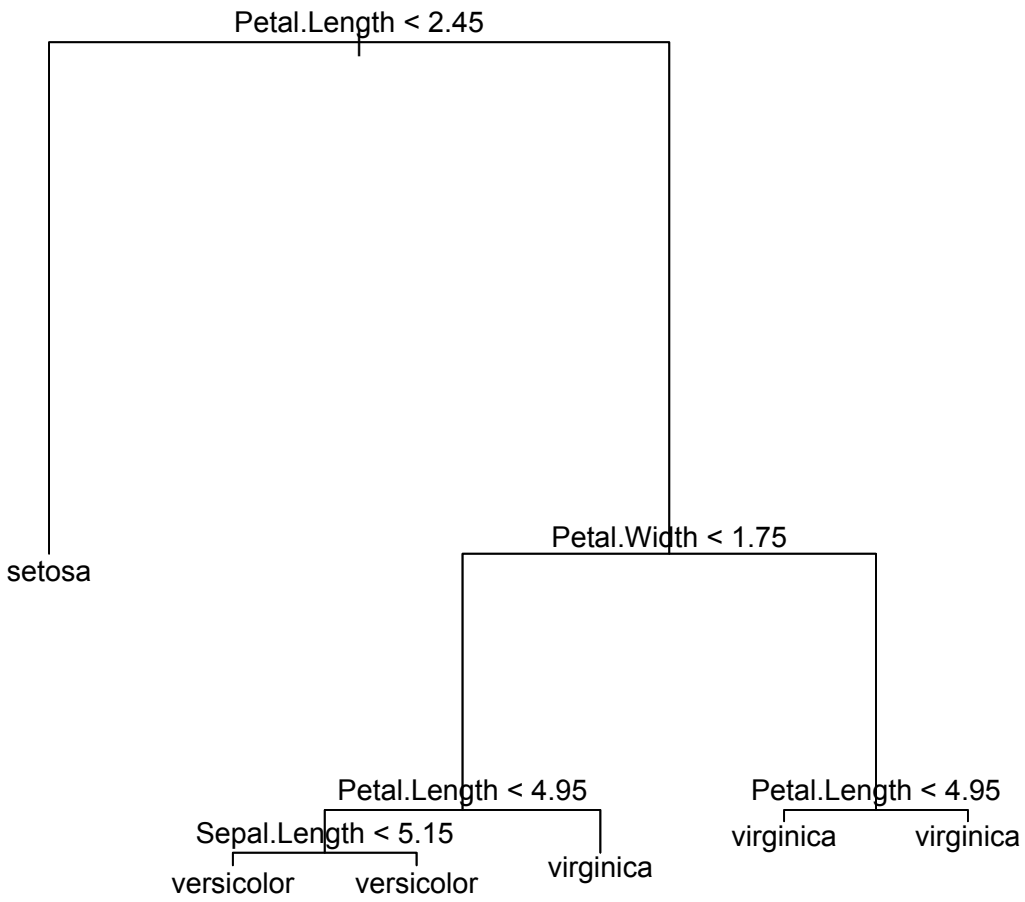
How good is a split in producing "pure" nodes.

(MULTIVARIATE) REGRESSION TREES

The split procedure takes place until nodes contain only one set of features, or the data are too sparse (fewer than six cases).



REGRESSION/CLASSIFICATION TREES



As for linear regression, **Regression trees produce a predictive model** that can be used in other datasets.

Like a regression we have two aims

- Simplicity (few nodes).
- Maximising the explained variance in the response variable.

REGRESSION/CLASSIFICATION TREES

As for any regression, you define the additive combination of predictors (X_1,...X_n) & the response , that can be **Categorical** *OR* **Continuous**

```
fit <- rpart(Y ~ X1 + X2 + X3 ,
             method="anova",
             data=DataIn)
```

REGRESSION/CLASSIFICATION TREES

Defines the type of data you have
"`class`" for a classification tree
"`anova`" for a regression tree
*you should **ALWAYS** specify this argument*

```
fit <- rpart(Y ~ X1 + X2 + X3 ,
             method="anova",
             data=DataIn)
```

classification tree = Categorical response
regression tree = Numerical response

So far so good?

Any questions?

Ready to move on?

REGRESSION/CLASSIFICATION TREES CROSS-VALIDATION

What is cross-validation? → a way to assess model fil by dividing the data into two sets

- **Training** → used to build a model (usually 70% of the observations)
- **Testing** → used to assess the "goodness-of-fit" of the regression tree (the remaining 30%))

n = 12
k = 3

Test Train

Data

1	2	3	4	5	6	7	8	9	10	11	12

REGRESSION/CLASSIFICATION TREES CROSS-VALIDATION

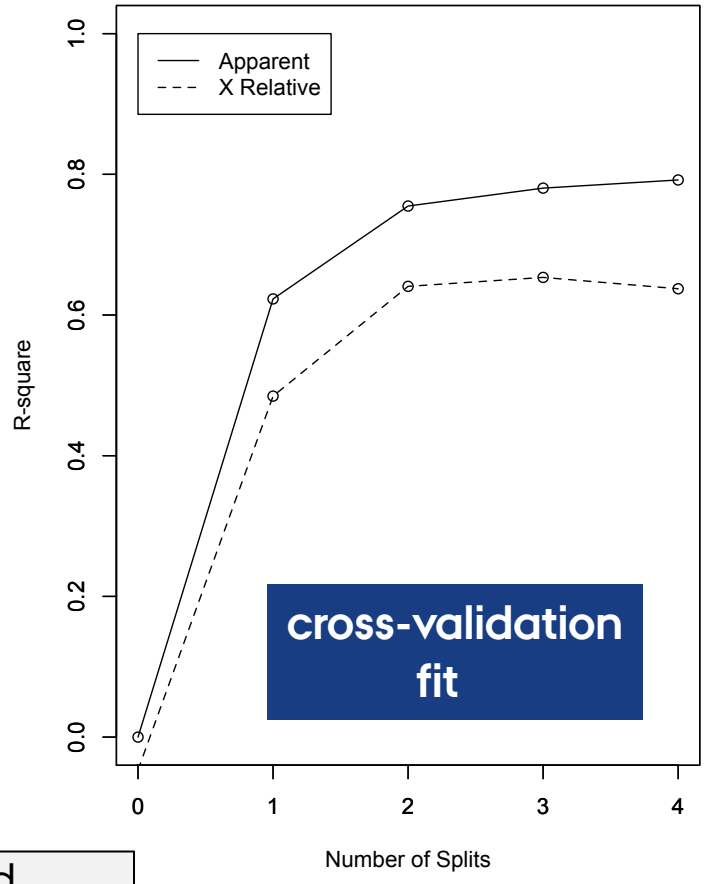
What is cross-validation? → a way to assess model fil by dividing the data into two sets

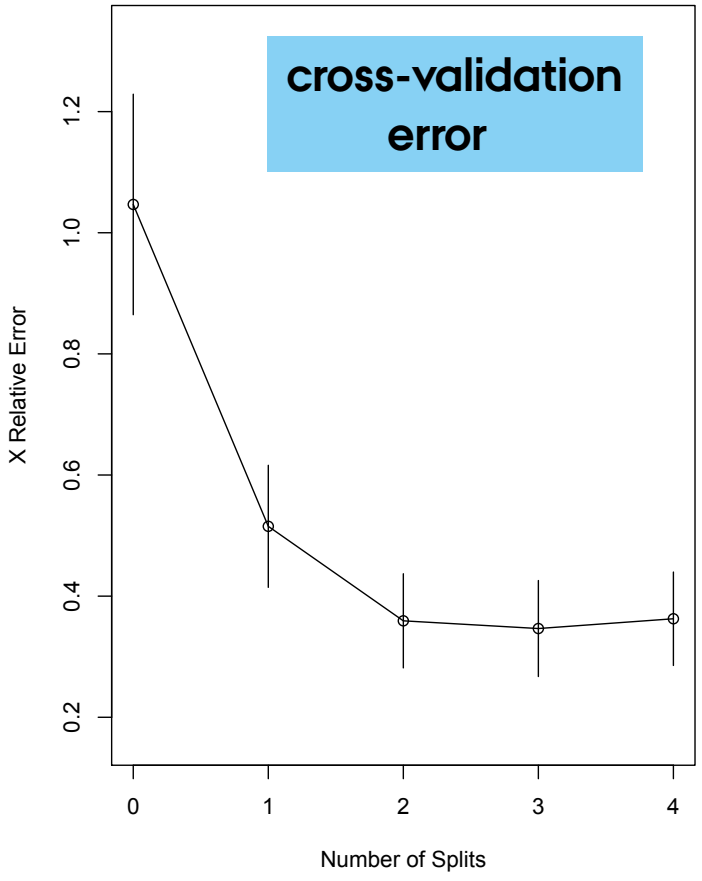
- **Training** → used to build a model (usually 70% of the observations)
- **Testing** → used to assess the "goodness-of-fit" of the regression tree (the remaining 30%))

The goal is to build multiple trees with different training datasets to assess model performance

REGRESSION/CLASSIFICATION TREES CROSS-VALIDATION

How good is your classification





How often you make an error

Apparent = Observed
Cross validated = x relative

So far so good?

Any questions?

Ready to move on?

REGRESSION/CLASSIFICATION TREES SIMPLIFICATION

Model simplification in the context of regression trees focuses on deciding how much of a model to retain (number of splits) by balancing:

- **Cross-validation values** → how good is my model
- **Complexity parameter** → cost of adding another variable/node to the model.

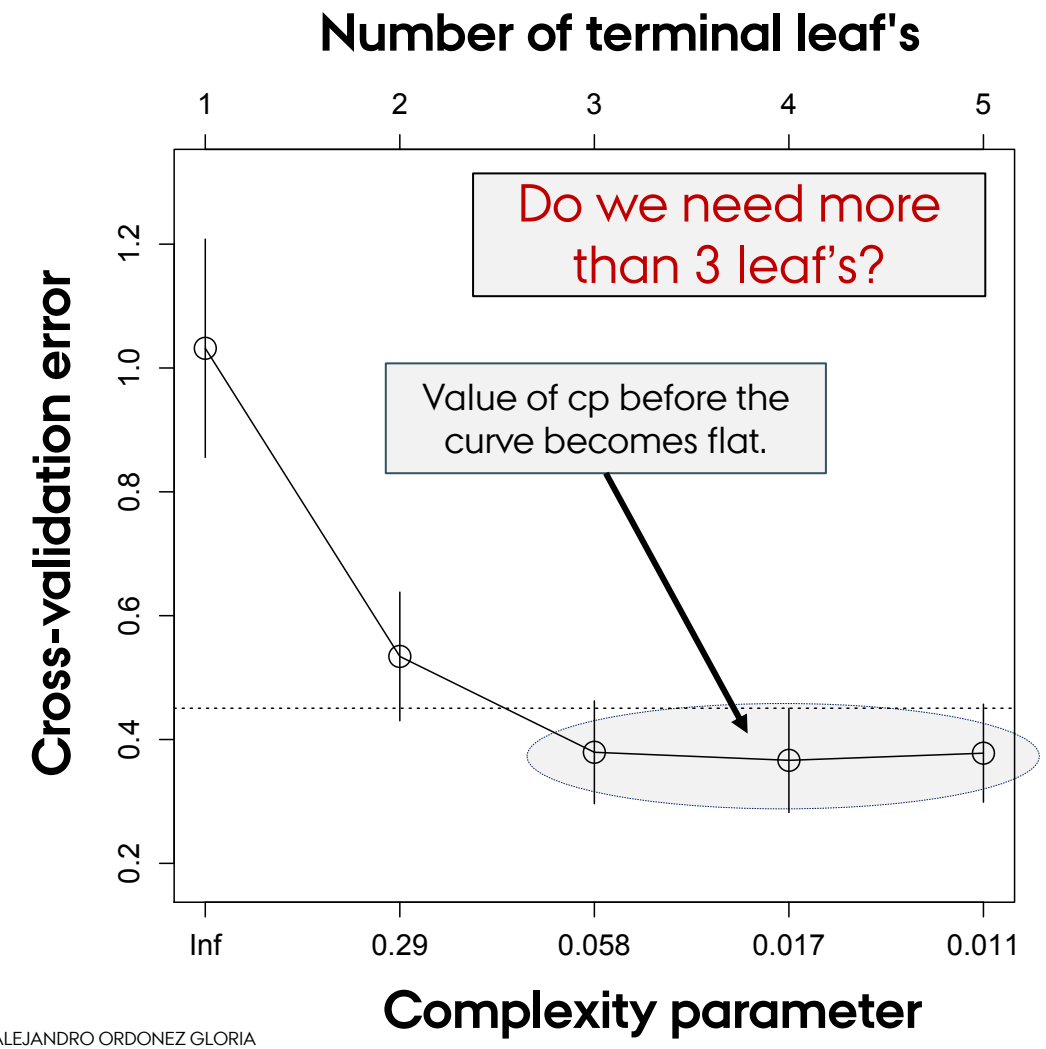
This is an analogue philosophy to stepwise selection where you add/remove variables and testing how much is gained/lost by doing so.

REGRESSION/CLASSIFICATION TREES SIMPLIFICATION

Based on a **cost-complexity measure**

The 1-SE rule

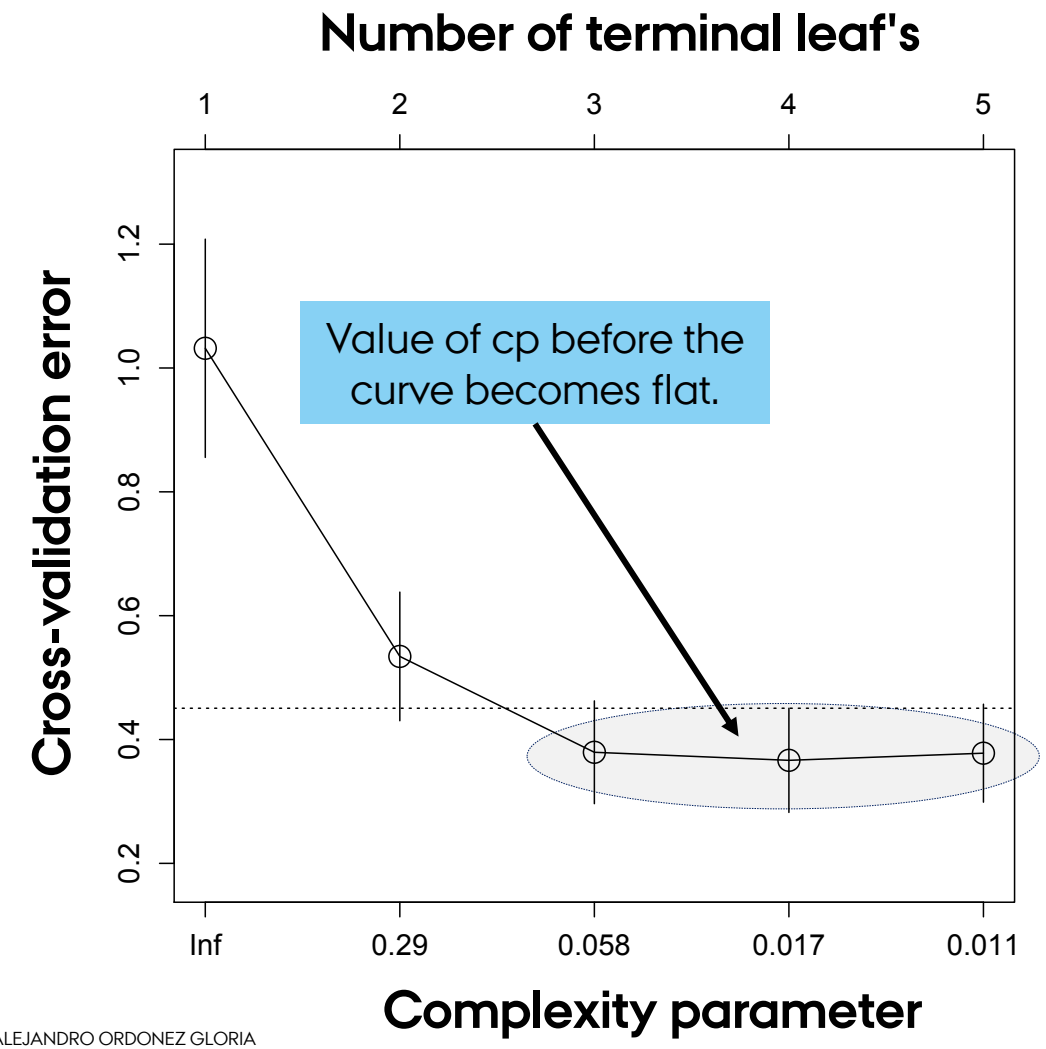
At which point adding a nodes does not improve the model.



REGRESSION/CLASSIFICATION TREES SIMPLIFICATION

Typically, you will want to select a Complexity parameter (hence tree size) that minimizes the cross-validated error.

Optimizes the trade-off between fit and explanatory power.



REGRESSION/CLASSIFICATION TREES SIMPLIFICATION

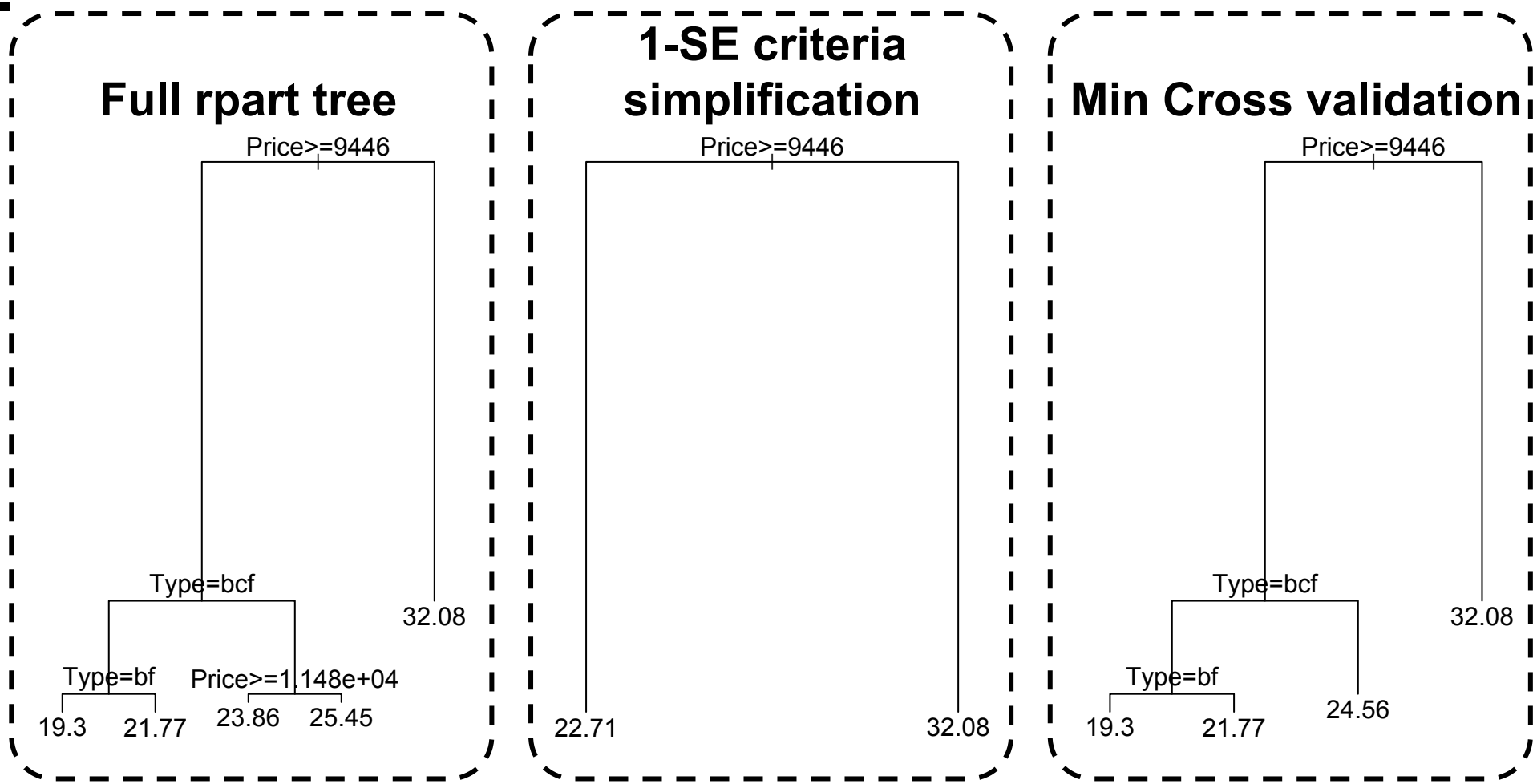
An alternative criteria is based on assessing the change in the **Cross-validation error**

```
printcp(fit)
Regression tree:
rpart(formula = Mileage ~ Price + Country +
Reliability + Type,
    data = cu.summary, method = "anova")
Variables actually used in tree construction:
[1] Price Type
Root node error: 1354.6/60 = 22.576
n=60 (57 observations deleted due to missingness)
        CP nsplit rel error  xerror     xstd
1 0.622885      0   1.00000 1.04671 0.182034
2 0.132061      1   0.37711 0.51528 0.100760
3 0.025441      2   0.24505 0.35924 0.077717
4 0.011604      3   0.21961 0.34653 0.079250
5 0.010000      4   0.20801 0.36266 0.077115
```

Cross-validated error

At this level the error reaches the minimum

REGRESSION/CLASSIFICATION TREES SIMPLIFICATION



So far so good?

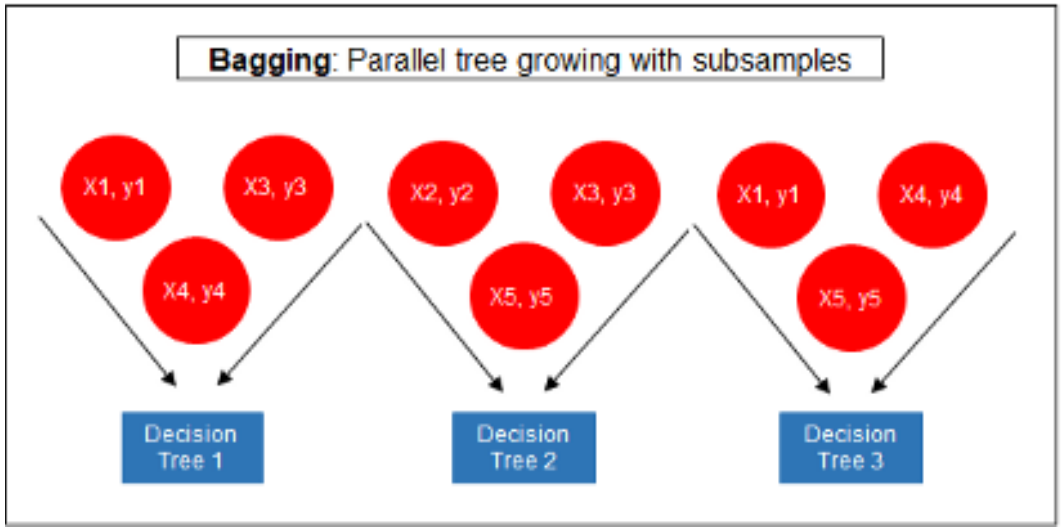
Any questions?

Ready to move on?

RADOM FORESTS
A GENERALIZED VIEW OF THE ALGORITHM

Radom Forests combine the strengths of two algorithms:

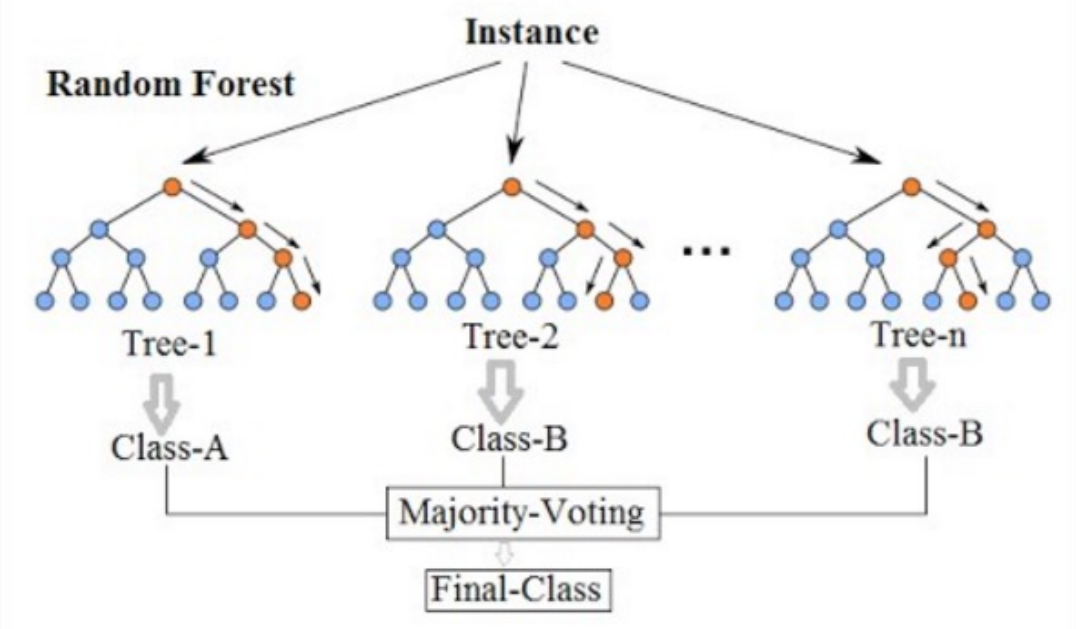
- **Regression trees:** models that relate a response to their predictors by recursive binary splits
- **Bagging**: an adaptive method for parallel tree growing with subsamples.



RANDOM FORESTS

An assembly of multiple regression trees

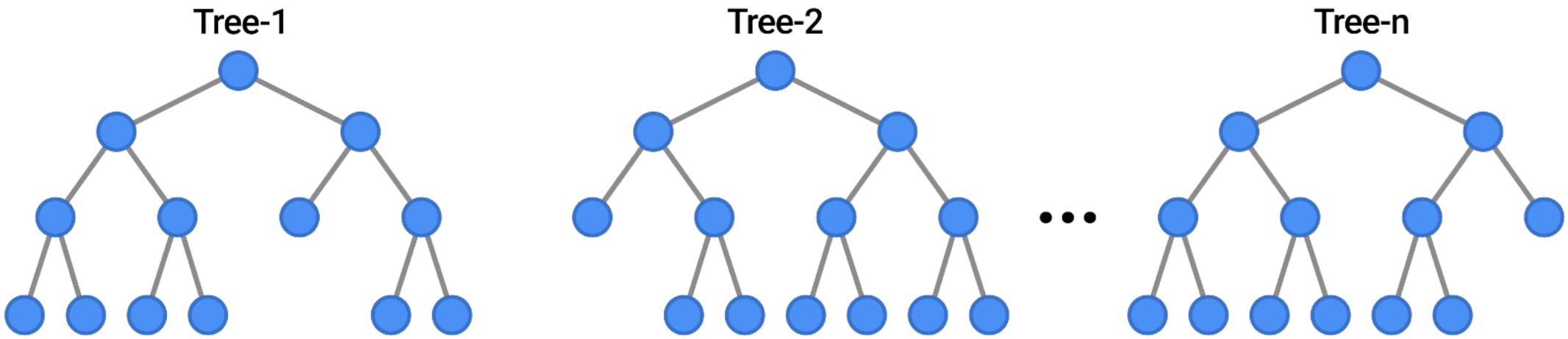
- Many trees are produced based on subsamples of the data [**this is bagging!**]
- Each case is classified using each tree in this new "forest" of trees
- A final predicted outcome is defined by combining the results across all the trees based on predefined criteria
  - **Average** in regression.
  - **Majority** vote in classification.



GOAL: Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features.

RANDOM FORESTS

EXAMPLES



RANDOM FORESTS
WHY (NOT) USE THESE?

Advantages

- Can be used with a variety of response types (binomial, gaussian, poisson).
- Stochastic, which improves predictive performance.
- It provides higher accuracy through cross validation.
- Robust to missing values and outliers.
- Can handle a large data sets with higher dimensionality.

Limitations

- Sensitive to differences in the prevalence of each evaluated class.
- Data and computational intensive:
  - You need many observations.
  - You need many trees.
- Random Forests will struggle to make accurate predictions when:
  - The data are very spars.
  - There are not clear cut boundaries.
- Sensitive to model parameters.

RANDOM FORESTS
WHAT DO I NEED TO SPECIFY IT?

RF have **five** important parameters that need to be specified by the user.

- **Number of trees to generate:** this defines on how many trees does my consensus prediction is based on.
- **How many predictors per tree**: define how many predictors are used in the classification of each randomly sampled dataset.
- **Sample size**: this defined how big of should be the sample of the original dataset.
- **Node size**: minimum number of samples left in a node before a split.
- **Maximum number of nodes**: Maximum depth of trees.

RANDOM FORESTS
WHAT DO I NEED TO SPECIFY IT?

The important parameters that need to be specified by the user.

- **Number of trees to generate:** this defines on how many trees does my consensus prediction is based on.
- **How many predictors per tree**: define how many predictors are used in the classification of each randomly sampled dataset.
- **Sample size**: this defined how big of should be the sample of the original dataset.

These are the set of parameters you will usually will like to start "adjusting" to optimise the prediction

RANDOM FORESTS IMPLEMENTING THESE IN R

Basic implementation via the `randomForest` package

```
randomForest(formula,
             data,
             ntree,
             mtry,
             importance)
```

- → a formula describing the model.
- → Data frame with all the data.
- → Number of random trees to generate.
- → Number of predictors used in each tree .
- → Assess the predictors importance.

Some advanced arguments

- `sampsize` → Size of the Random sample.
- `nodesize` → Minimum number of samples left in the final leaf node.
- `maxnodes` → Maximum depth of trees.
- `replace` → If samples should be bootstrapped with replacement.

RANDOM FORESTS – IMPORTANCE

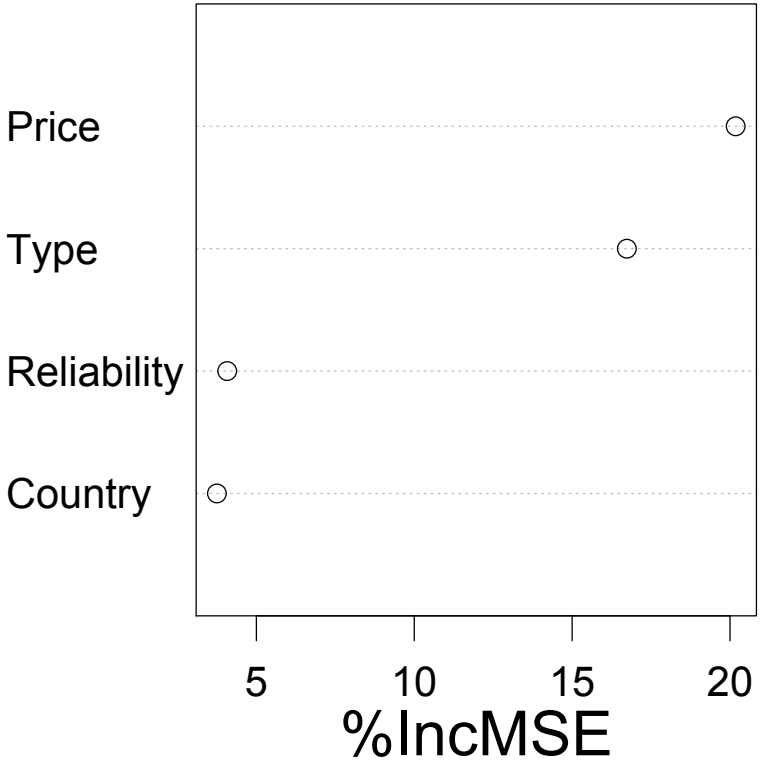
Implemented via the `randomForest` package

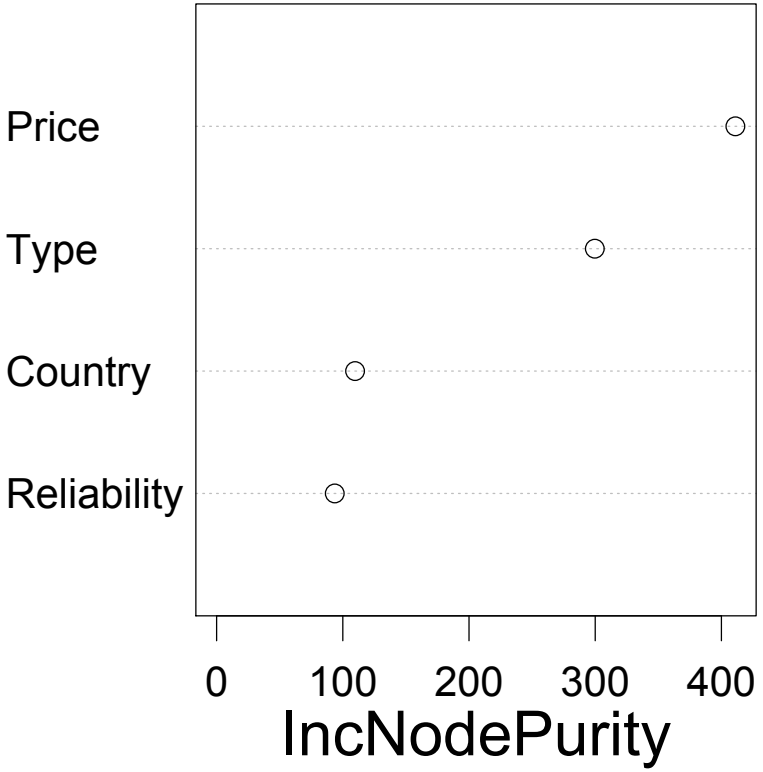
```
library(randomForest)
fit <- randomForest(Mileage~.,
         data= na.omit(cu.summary),
         importance=TRUE)
importance(fit) # importance of each predictor
```

```
               %IncMSE IncNodePurity
Price        20.179096     411.07061
Country       3.750334     109.71086
Reliability   4.076656      93.61005
Type         16.732901     299.68206
```

RANDOM FORESTS – IMPORTANCE

```
varImpPlot(fit)# plot the importance of each predictor
```





`%IncMS` mean squared error increase if a variable is removed from the predictors

`IncNodePurity` increase in node impurity (how well the trees split the data) averaged over all trees

So far so good?

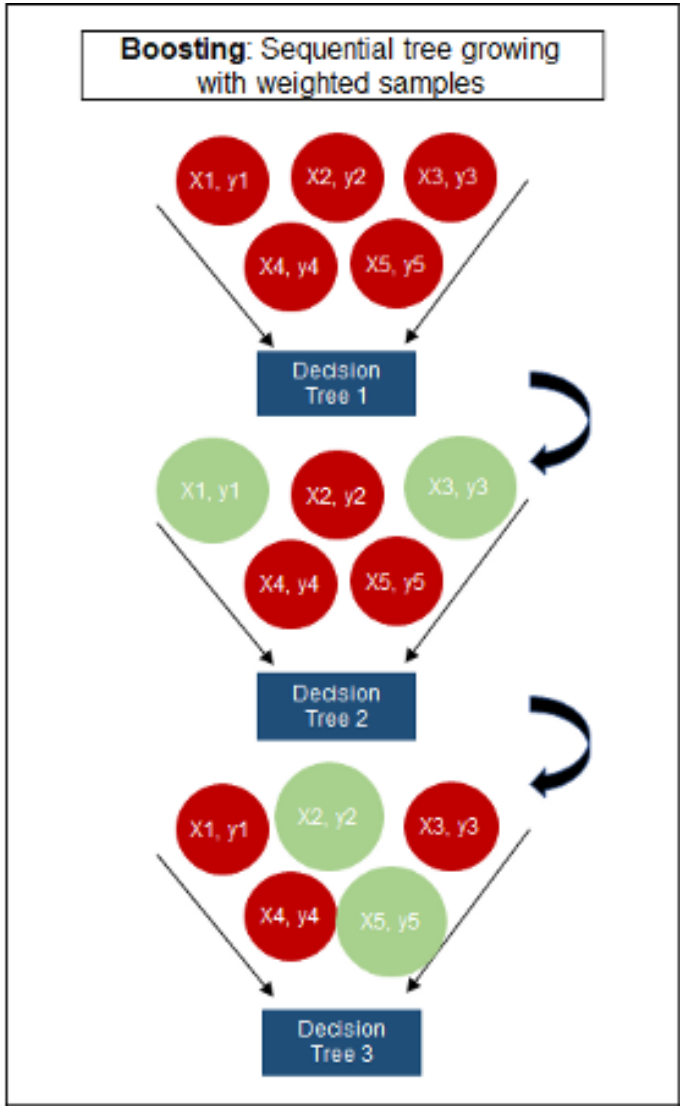
Any questions?

Ready to move on?

BOOSTED REGRESSION TREES (BRT)

These models combine the strengths of two algorithms:

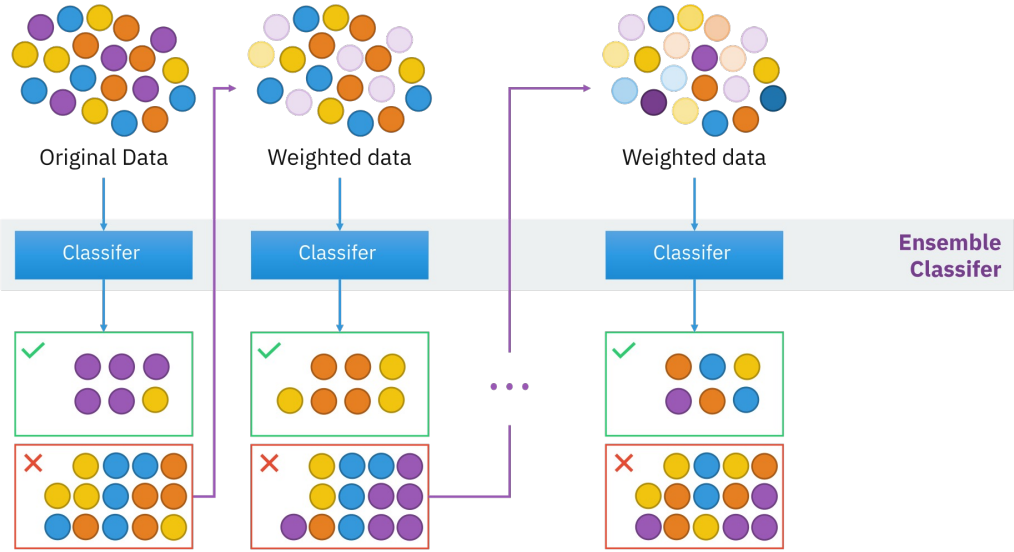
- **Regression trees:** models that relate a response to their predictors by recursive binary splits
- **Boosting**: an adaptive method for combining many simple models to give improved predictive performance.



BOOSTED REGRESSION TREES (BRT)

An assembly of **sequential** regression trees

- Many **SMALL** trees are produced based on subsamples of the data.
- But trees are built **sequentially** focusing on correcting past errors.
- A final predicted outcome is defined by weighting the results across all the trees.
  - Weighted average in regression.
  - Weighted classification.



GOAL: Build many small trees (week learners) that when combined make a good calcification by correcting past mistakes .

BOOSTED REGRESSION TREES (BRT)

What is the principle behind the approach?

- Repeatedly fit many decision trees to improve the accuracy of the model.
- The repetitions are based on random subsets (**with replacement**) have the same number of data points and are selected from the complete dataset.

BOOSTED REGRESSION TREES (BRT)

What is the difference with Random forest?

- In BRT, the random subsalting is weighted by the results of the trees build before.
  - ***This is boosting*** → subsequent models aim to correct the "error" of previous models.
- After the first tree is fitted the model will consider the error in the prediction of that tree to fit the next tree (upweight those observation wrongly classified before), and so on.

BOOSTED REGRESSION TREES (BRT) WHY (NOT) USE THESE?

Advantages

- Can be used with a variety of response types (binomial, gaussian, poisson).
- Stochastic, which improves predictive performance.
- The best fit is automatically detected by the algorithm.
- No need for data pre-processing
- Model represents the effect of each predictor after accounting for the effects of other predictors.
- Robust to missing values and outliers

Limitations

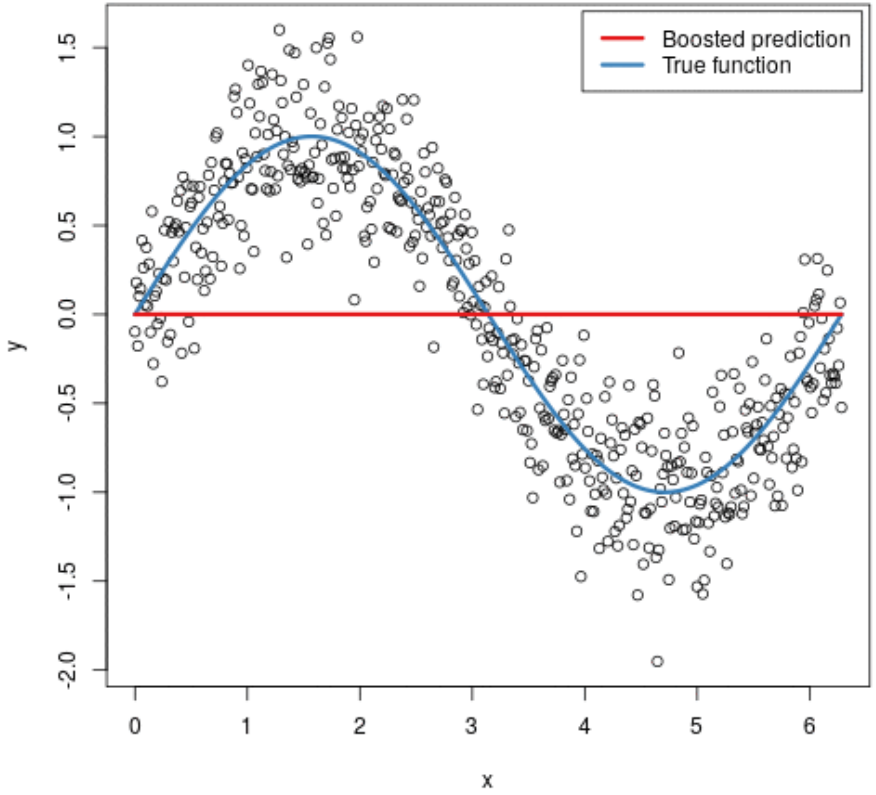
- Needs at least 2 predictor variables to run.
- Data and computational intensive:
    - you need many observations.
    - You need many trees.
- Sensitive to model specifications.
- Raw output are less interpretable.
- Can overemphasize outliers.

BOOSTED REGRESSION TREES (BRT)
A GENERALIZED VIEW OF THE ALGORITHM

The final BRT is simply a **stagewise additive model** of *n* individual regression trees

- You start with a poor model that is improved sequentially…
- and ends up with a series of trees that get get close to the true underlying pattern.

This is similar to basis functions in GAMs



BOOSTED REGRESSION TREES (BRT) WHAT DO I NEED TO SPECIFY IT?

BRT have three important parameters that need to be specified by the user.

- **Bag fraction**: this defined how big of should be the sample of the original dataset.
- **Tree complexity**: this controls the number of splits in each tree.
- **Learning rate**: this determines the contribution of each tree to the growing model.

BOOSTED REGRESSION TREES (BRT) WHAT DO I NEED TO SPECIFY IT?

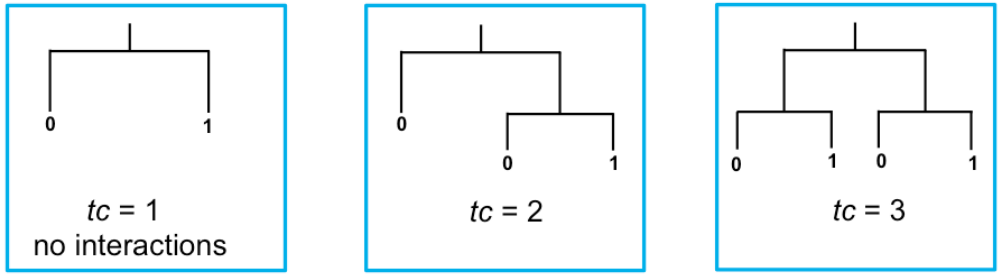
The important parameters that need to be specified by the user.

- **Tree complexity**: this controls the number of splits in each tree.
- **Learning rate**: this determines the contribution of each tree to the growing model.

These two parameters together determine the number of trees that is required for optimal prediction.

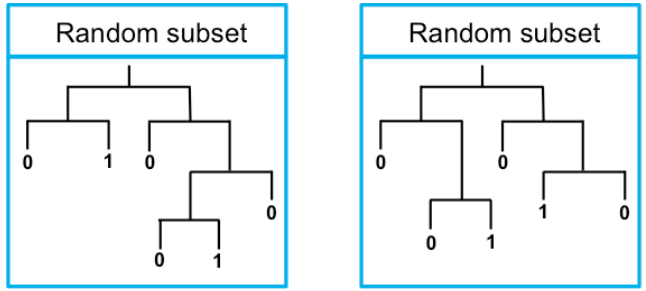
BOOSTED REGRESSION TREES (BRT) WHAT DO I NEED TO SPECIFY IT?

Tree complexity (*tc*)



How many branches per Small trees

Learning rate (*lr*)



lr: contribution to growing model
small value = many trees

How much information is gained by each subsequent tree

BOOSTED REGRESSION TREES (BRT) USING THE `GBM.STEP` FUNCTION

Basic implementation via the `gbm` package

```
gbm.step(data,
         gbm.x,
         gbm.y,
         family,
         tree.complexity,
         learning.rate,
         bag.fraction)
```

- → Data.frame with all the data
- → Indices or names of predictor variables
- → Indices or names of the response variable
- → Type of response [Bernoulli / Poisson / Gaussian]
- → number of splits in each tree
- → contribution of each tree to the growing mode
- → proportion of observations used

Some advanced arguments

- `n.folds` → Number of cross validations to be done – 10 by default
- `prev.stratify` → Balance the state of each predictor for each sate of the response. Only active for presence/absence data.
- `prev.stratify` → Balance the state of each predictor for each sate of the response.
- `n.trees` → Number trees to be used

BOOSTED REGRESSION TREES (BRT) VARIABLE IMPORTANCE

Once you have build the “best” model, you would like to know understand the variables that have the largest influence on the response variable.

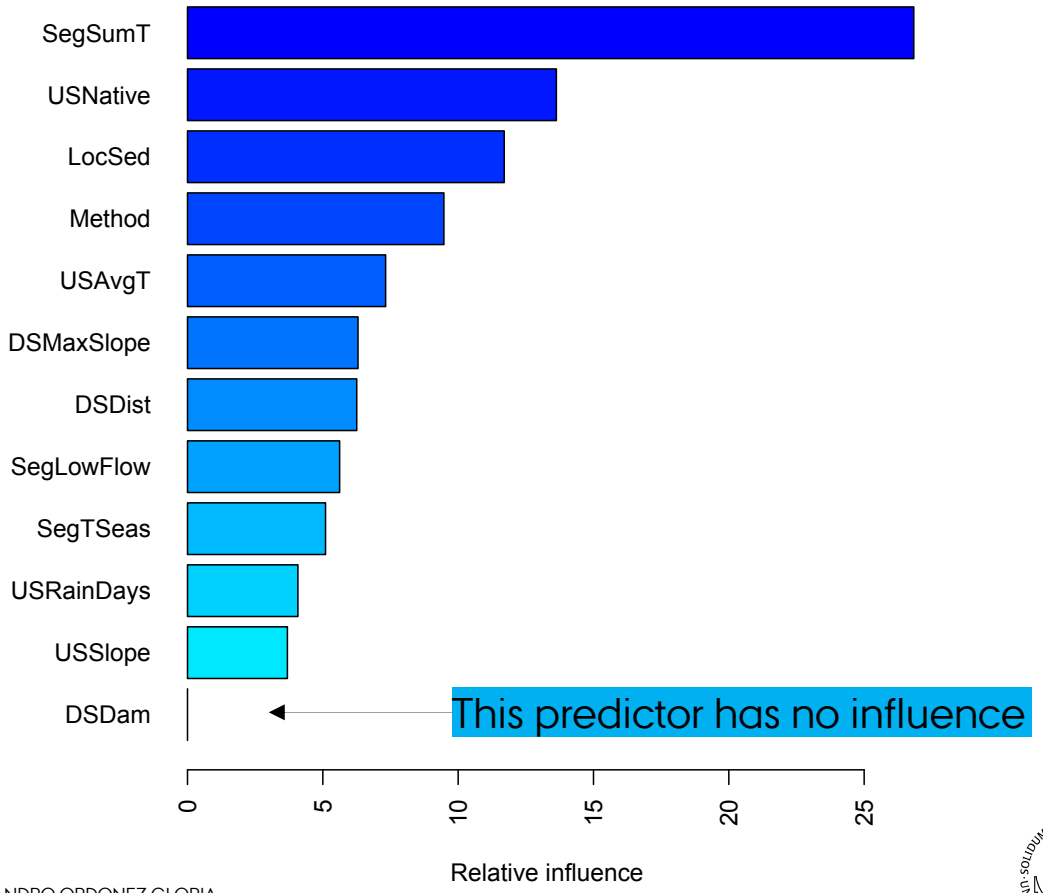
You can extract this by using either

- The `summary` function on a a `gbm.step` crated object [this also prints a plot of variance importance]

- The subscript `contributions` in a `gbm.step` crated object

BOOSTED REGRESSION TREES (BRT) VARIABLE IMPORTANCE

```
> summary(angaus.tc5.lr01,las=2)
                var   rel.inf
SegSumT       SegSumT 26.833196
USNative     USNative 13.628545
LocSed         LocSed 11.702216
Method         Method  9.472981
USAvgT         USAvgT  7.320455
DSMaxSlope DSMaxSlope  6.299540
DSDist         DSDist  6.252498
SegLowFlow SegLowFlow  5.620587
SegTSeas     SegTSeas  5.099794
USRainDays USRainDays  4.079246
USSlope       USSlope  3.690943
DSDam           DSDam  0.000000
```



So far so good?

Any questions?

Ready to finish?

IN SUMMARY…

- Classification and Regression trees (CART) provides tool to recursive partitioning of a **response variable** under the control of a set of **explanatory variables**.
  - Classification trees → a categorical response.
  - Regression trees → a quantitative response.
- The goal is to create sequential partitions that minimize a measure of purity of the classification

IN SUMMARY…

- The focus of is producing the best prediction (measured by node purity); and to optimize these CART can be combined with
  - **Bagging Techniques** → Random forests
  - **Boosting techniques** → Boosted regression trees

  … to provide a series of predictions that when combined result in a more accurate prediction than that of any individual tree.

AARHUS
UNIVERSITY