SIAD M2 DS – 2024/2025

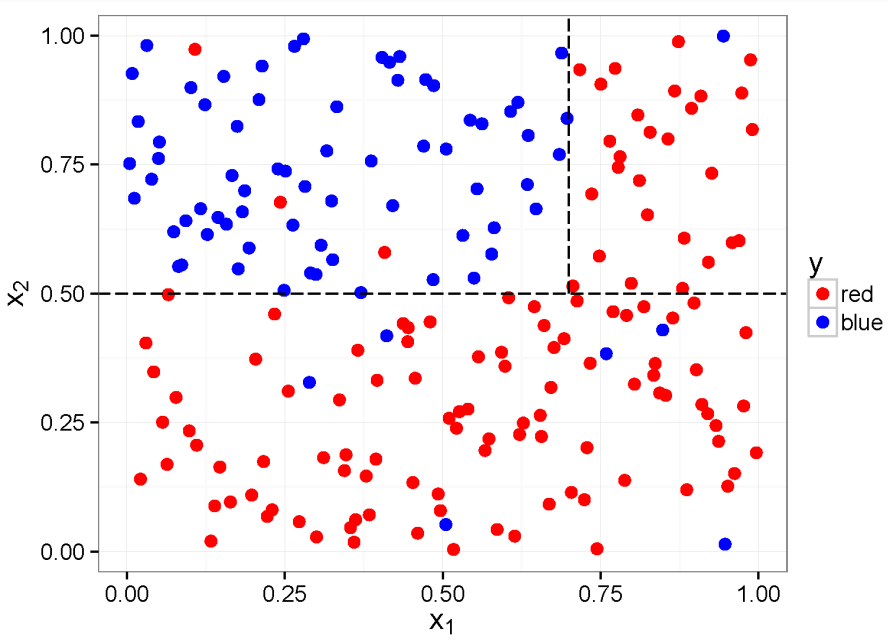
Machine Learning – Tree-based Methods

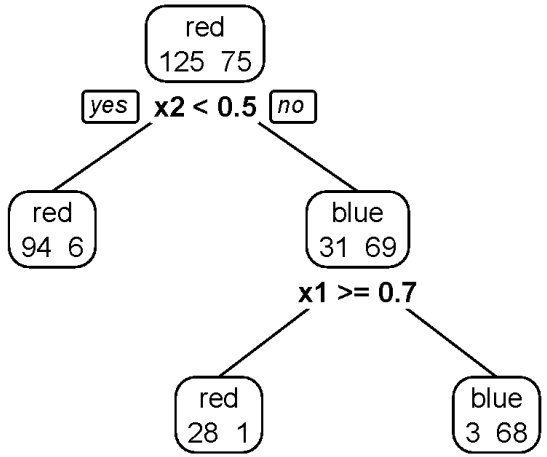
Nadarajen Veerapen

Faculté des sciences économiques, sociales et des territoires – Université de Lille

Classification Trees

Decision Trees





Divide and conquer:

1. Recursively partition the data
2. Learn a simple model for each of the partitions

In particular, and to make it simple:

1. A binary split (yes/no) induces partitions parallel to the axes
2. A greedy selection of the partition that maximizes the purity of the nodes in the tree
3. A prediction that is the same for all observations inside a partition

Decision Tree – Simplified Algorithm

```
function buildTree(data d)
        create a node t
        if the end condition is reached for t then
                assign a model to t
        else
                find a partition p of d that maximizes purity
                partition d into d_left and d_right according to p
                t_left = buildTree(d_left)
                t_right = buildTree(d_right)
        endif
        return t
```

Impurity Measures for a Node

If p is the proportion of observations of the other class in node t, then impurity can be computed in different ways, including:

Misclassification Error

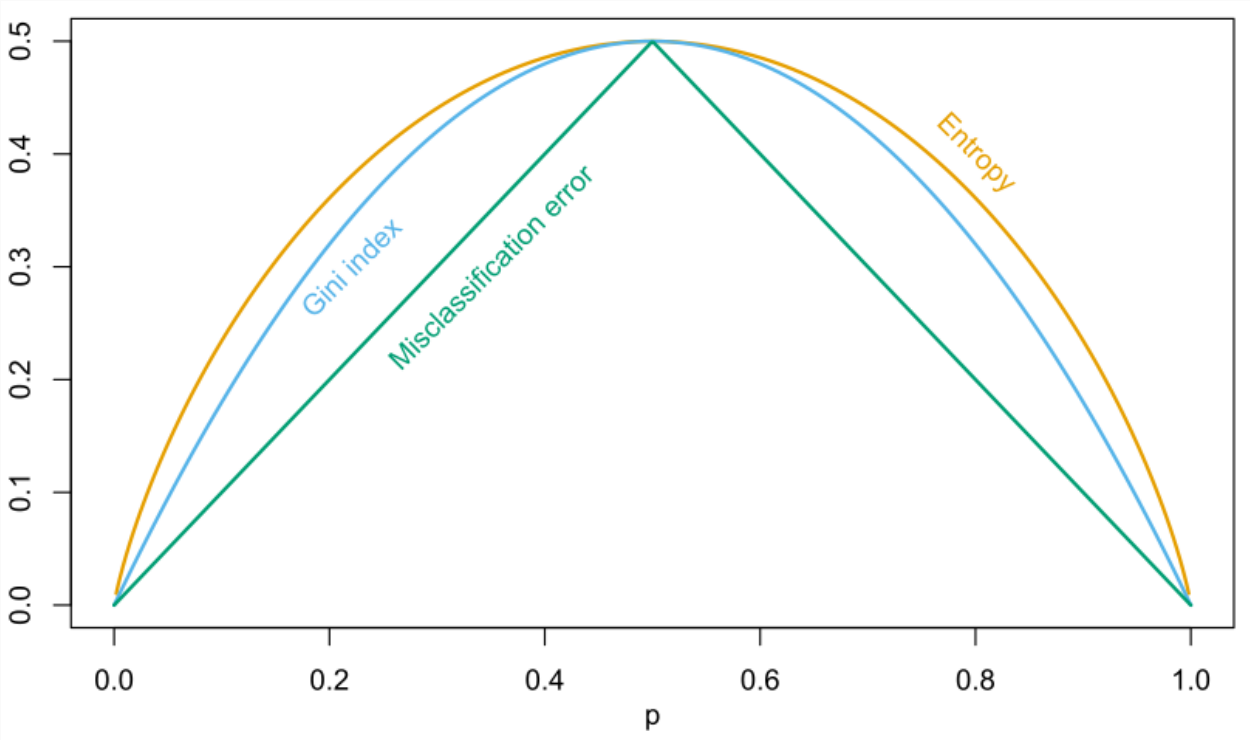
$$i(t) = p - \max(p, 1-p)$$

Gini Index

$$i(t) = 2p(1-p)$$

Entropy

$$i(t) = \frac{-p\log(p) - (1-p)\log(1-p)}{2\log(2)}$$



Values for measures in the binary case, image from Hastie et al. 2009

Algorithms for building Decision Trees

ID3 (Iterative Dichotomiser 3) dates back to 1986. This algorithm generates a **non-binary tree** in a **greedy** fashion where, for each node, the **categorical variable** that maximizes information gain (entropy) is chosen. The tree is generated until it reaches its maximum size. Then a **pruning** step is applied in order to increase the tree's capacity to **generalize** on unseen data.

C4.5 is the successor to **ID3**. It allows not only **categorical variables** but also **continuous variables** (by transforming them into a set of discrete intervals).

CART (**Classification** and **Regression** Trees) is similar to C4.5 but also allows for a continuous target (for regression). CART builds binary trees using the **Gini Index** as measure of impurity.

There are other algorithms to build decision trees.

Scikit-learn only implements the CART algorithm and does not allow categorical variables. These need to be transformed before being used.

Some considerations for Decision Trees

Decision trees have a tendency to **overfit** on datasets with a **large number of variables**. It may therefore be useful to **select the most promising variables first**.

Decreasing the **depth** of the tree, **increasing** the minimum number of **observations to partition a node** or increasing the minimum number of **observations per leaf** may help to **reduce overfitting**.

Decision trees are affected by **instability**. On real-life data, very little is needed for one variable to be chosen over another one when creating a node. This choice, especially at the beginning of building the tree, can greatly influence the rest of the construction. The consequence is that learning methods based on decision trees have a **large variance**.

Regression Trees

Regression Trees

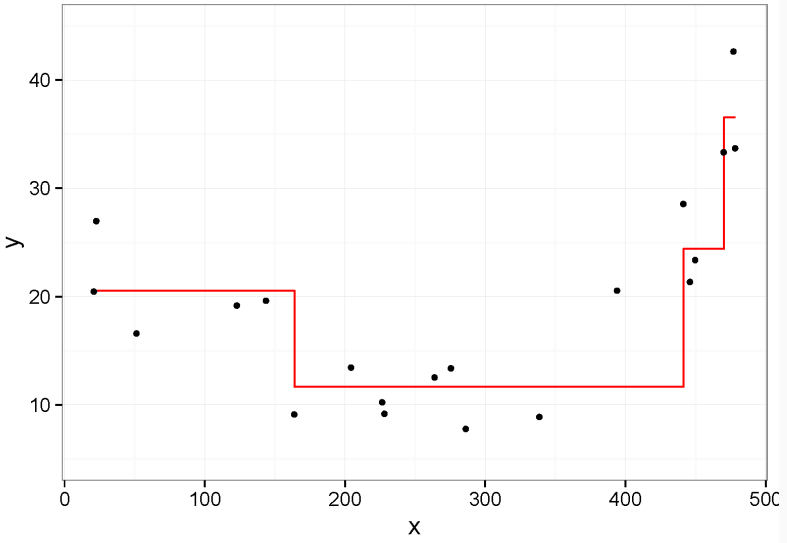
By changing the function computing impurity and the model used in a node, decision trees can also be used for regression.

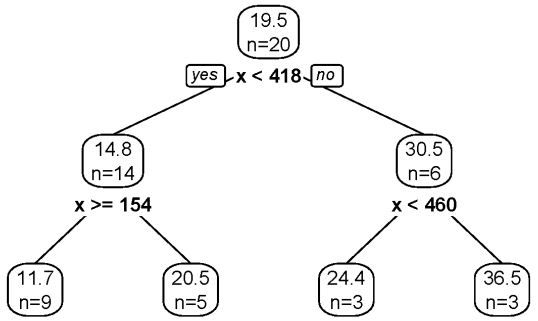
Classification

- y is a categorical variable
- The prediction corresponds to the majority class in the node.
- Entropy or Gini Index is used.

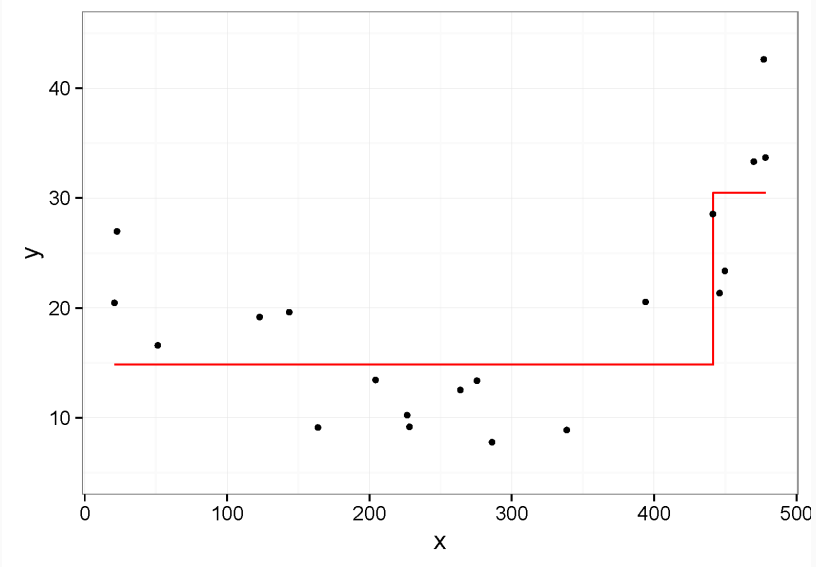
Regression

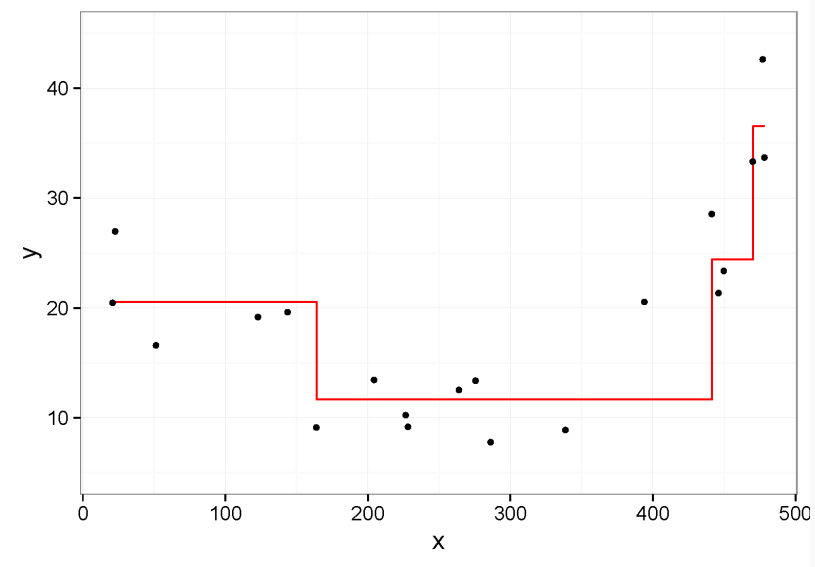
- y is a continuous variable
- The prediction corresponds to the mean of the observations in the node.
- Mean squared error (MSE) is used.

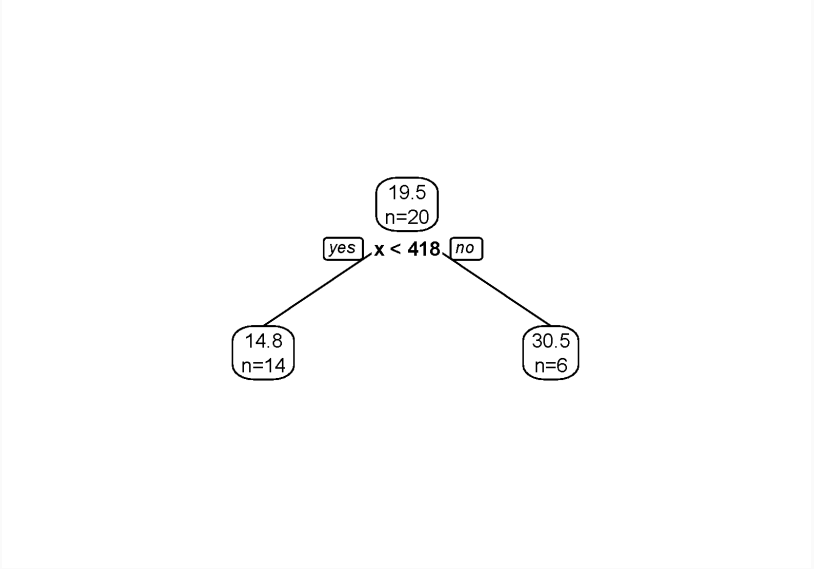


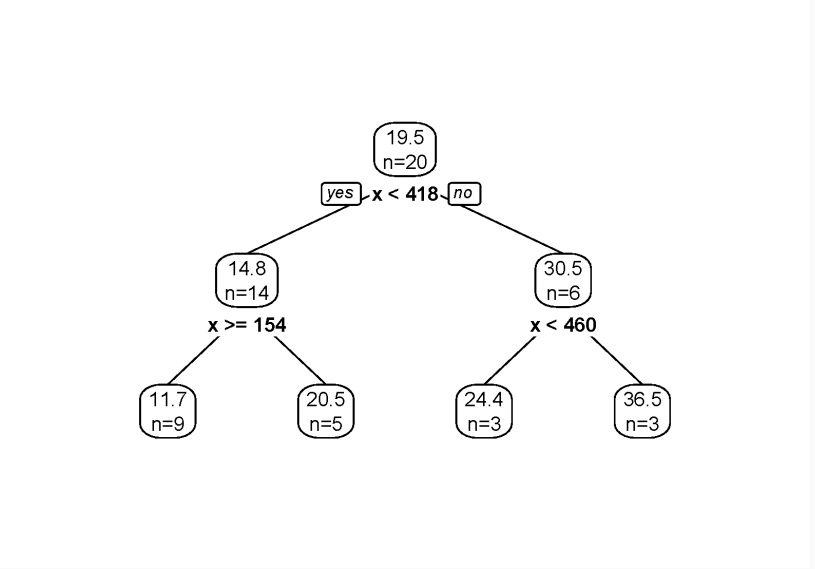


Effect on Tree Depth in Regression

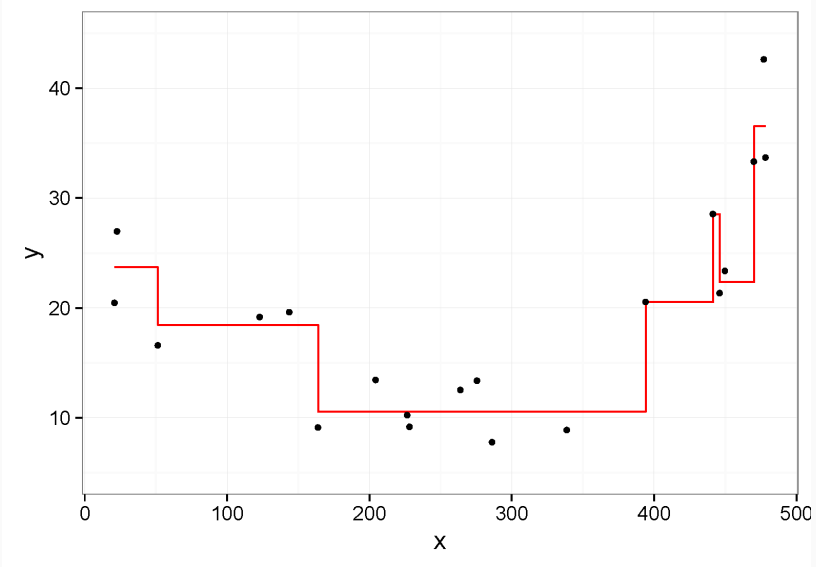


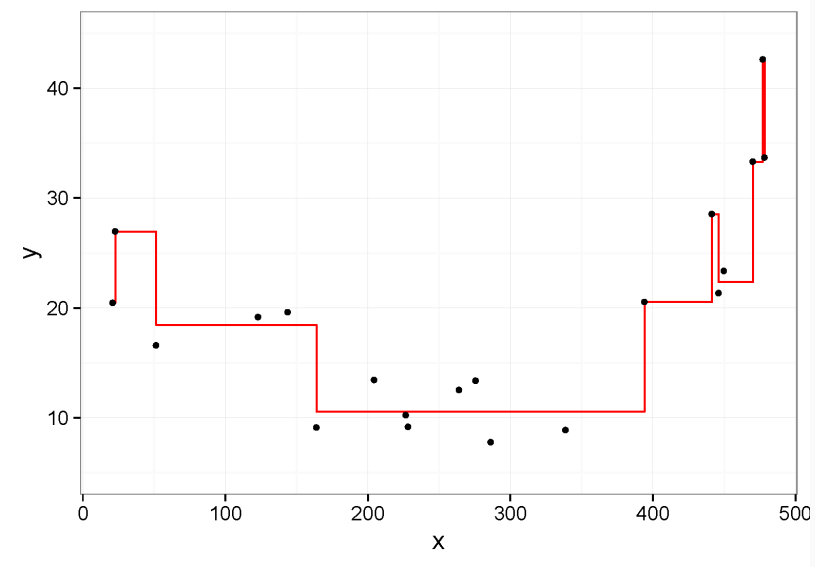


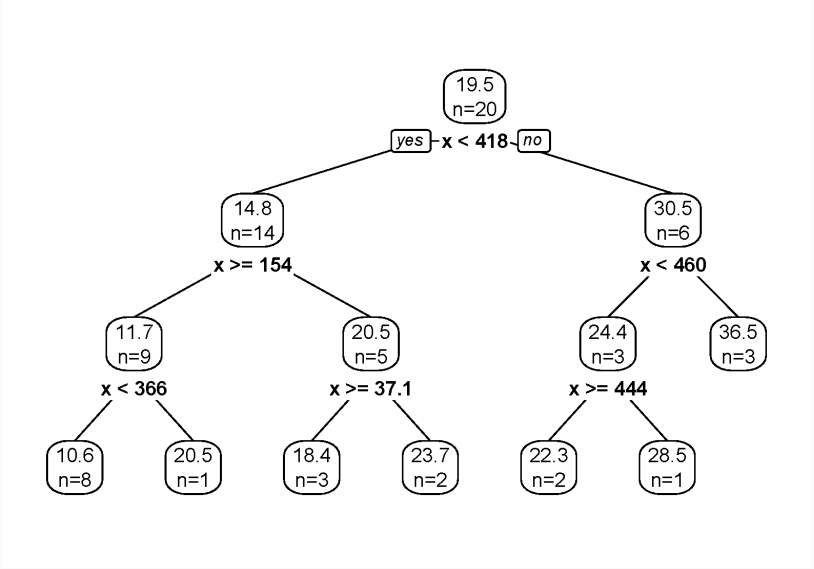


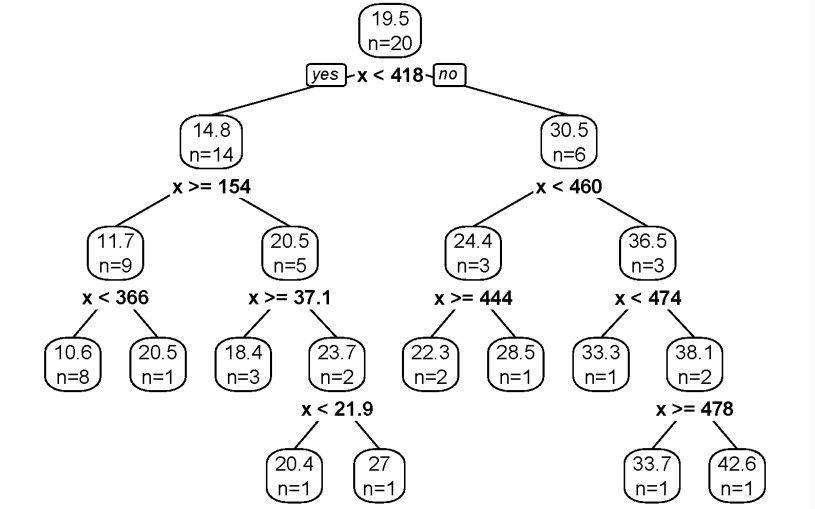


Effect on Tree Depth in Regression









Ensemble Methods

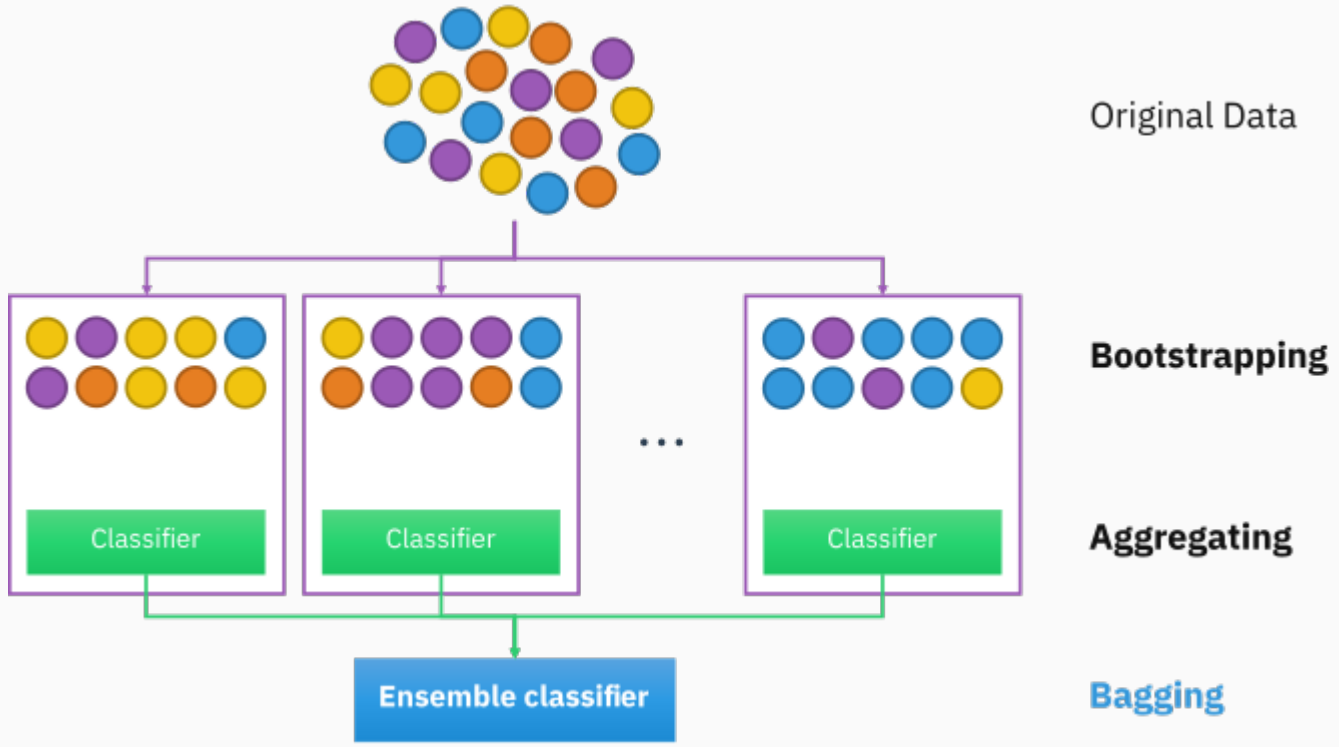
Ensemble Methods - Bagging

Ensemble methods combine the predictions of multiple models to improve the generalization capacity when compared to a single model.

Bagging (Bootstrap Aggregating)

Build multiple models *independently*, then average their predictions. The same kind of model is used each time, but with different training data.

Example: Random Forest



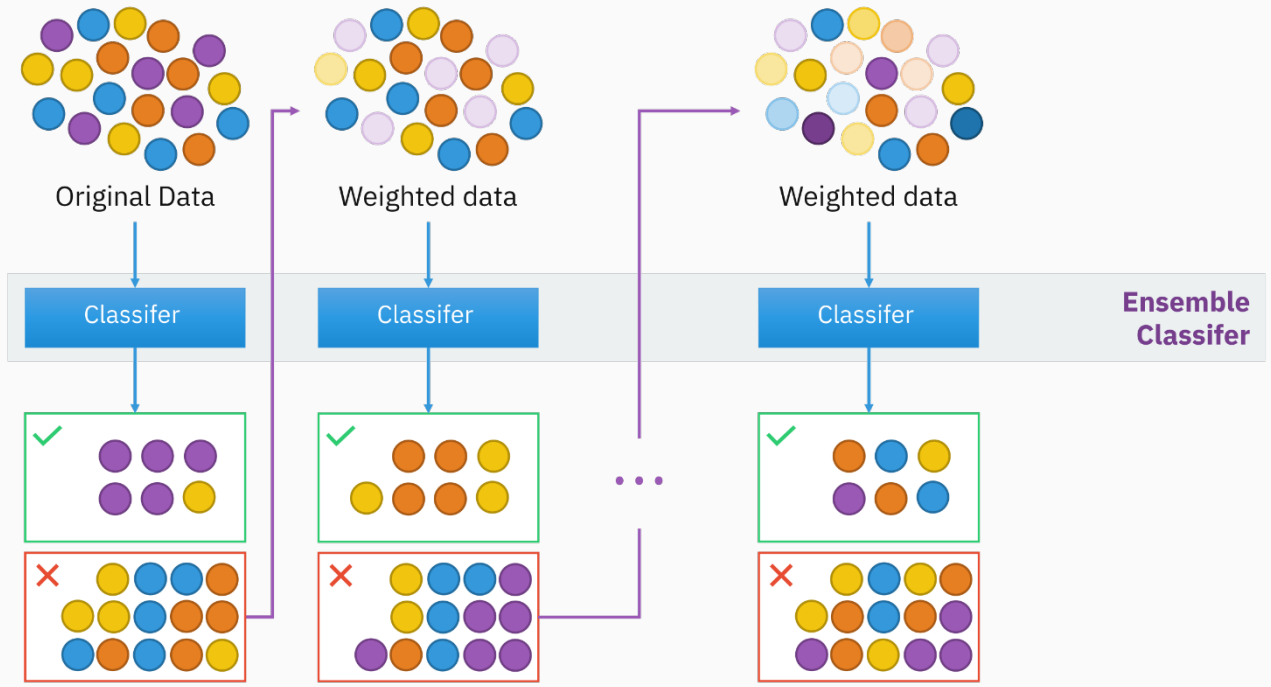
Ensemble Methods - Boosting

Boosting

Build multiple simple models (*weak learners*) *sequentially*, trying to reduce the mistakes of the previous ones.

AdaBoost (Adaptive Boosting) – At each step assign different weights to the training data. Initially, all the training samples have the same weight. In the following steps, the weights of the samples with incorrect predictions are increased, whereas the weights of the correctly predicted ones are decreased.

Gradient Boosting – At each step, the new model is built in order to minimize the mistakes of all the existing models and the new one, based on a loss function.



Random Forests

Ensemble methods combine several machine learning models in order to create more powerful models. Among successful ensemble models, we have random forests.

A decision tree by itself has a tendency to overfit on the training set. A **random forest** is an **ensemble of decision trees** where each tree is slightly different from the others.

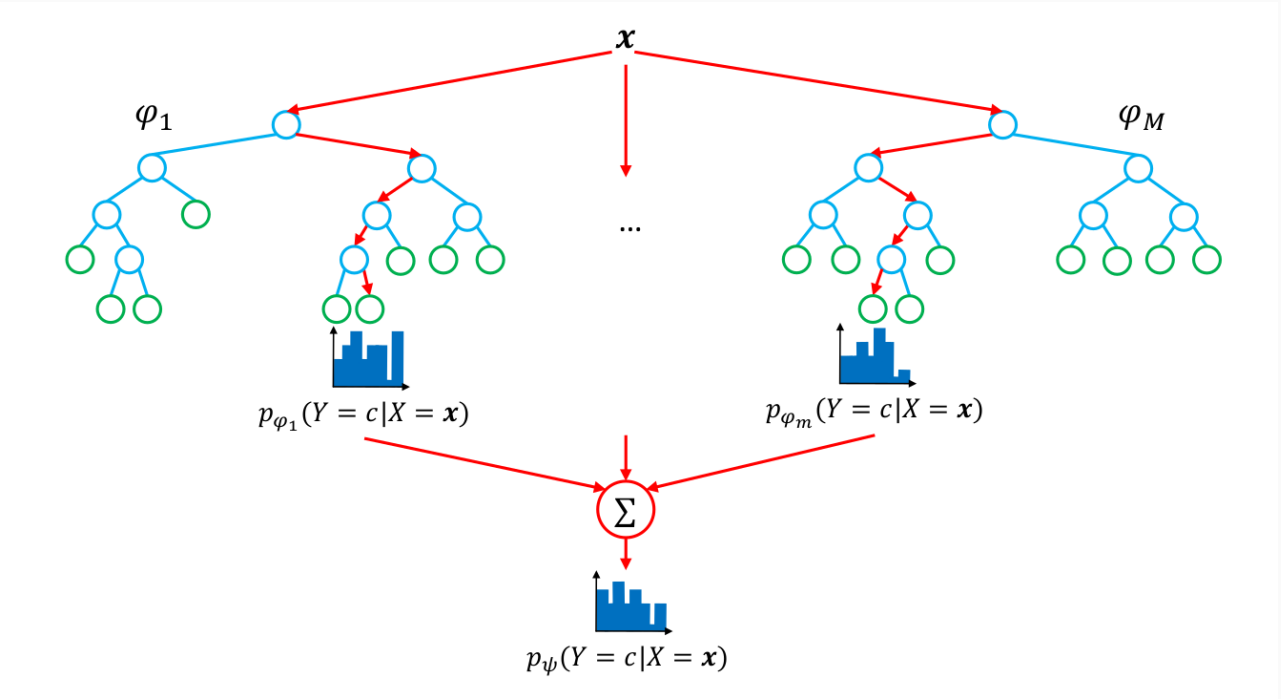
The basic idea is that

- Each tree will likely overfit with respect to part of the data
- Each tree will overfit in a different way
- Overfitting can be minimized by considering the predictions across all trees in the ensemble

Random Forests

We talk about **random** forests because randomness is injected into the construction process of each tree, by selecting a subset of:

- The observations used for learning
- The features considered when building a node



Building a random forest

- Set the number of trees in the ensemble/forest.
- A different training set is created for each tree through a random draw with remplacement from the initial training set. As many observations are drawn as there are in the initial training set.
- When creating each node of a tree, only a subset of features are considered. This subset is changed for each node.
- To make a prediction, each tree first makes its own prediction. These are combined, using the mean for regression and the mean of class probabilities, choosing the best class for classification.

The Tree against the Forest

Decision Tree

+ Flexible – can be applied in multiple contexts, and heterogeneous features
+ Training and prediction are very fast
+ Easy to visualize and interprete
- Lower accuracy
- Usually requires hyperparameter tuning

Random Forest

+ As flexible
+ Reasonably fast, easily parallelizable
+ Generally higher accuracy
+ Does not require much hyperparameter tuning
- Not very interpretatble