Lecture 9: ML3 -- Decision trees and boost methods

Wengang Mao (Marine Technology)
Department of Mechanics and Maritime Sciences,
Chalmers University of Technology,
Goteborg, Sweden

Contents of this lecture

- **Decision trees (CART) for classification/regression**
- **Example of Decision Tree Algorithm**
- **Pros and Cons of CART**
- **Random forest (Ensemble techniques)**
    - Bagging/Boosting

Decision tree (CART) definition

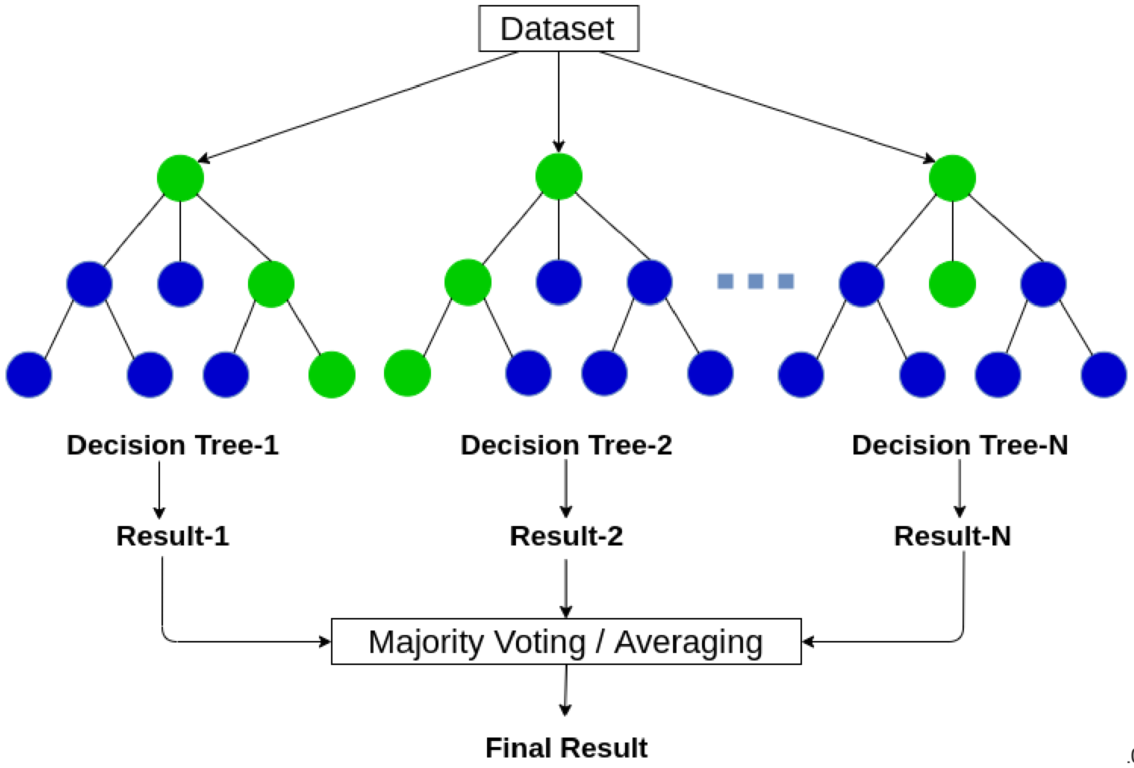
CART: Classification and Regression Trees

- ***Target:*** **represented by Leaves/Nodes**
- ***Features:*** **represented by Conjunctions/Branches**
- ***Location of leaves*: decided by Logical check >, =, or <**

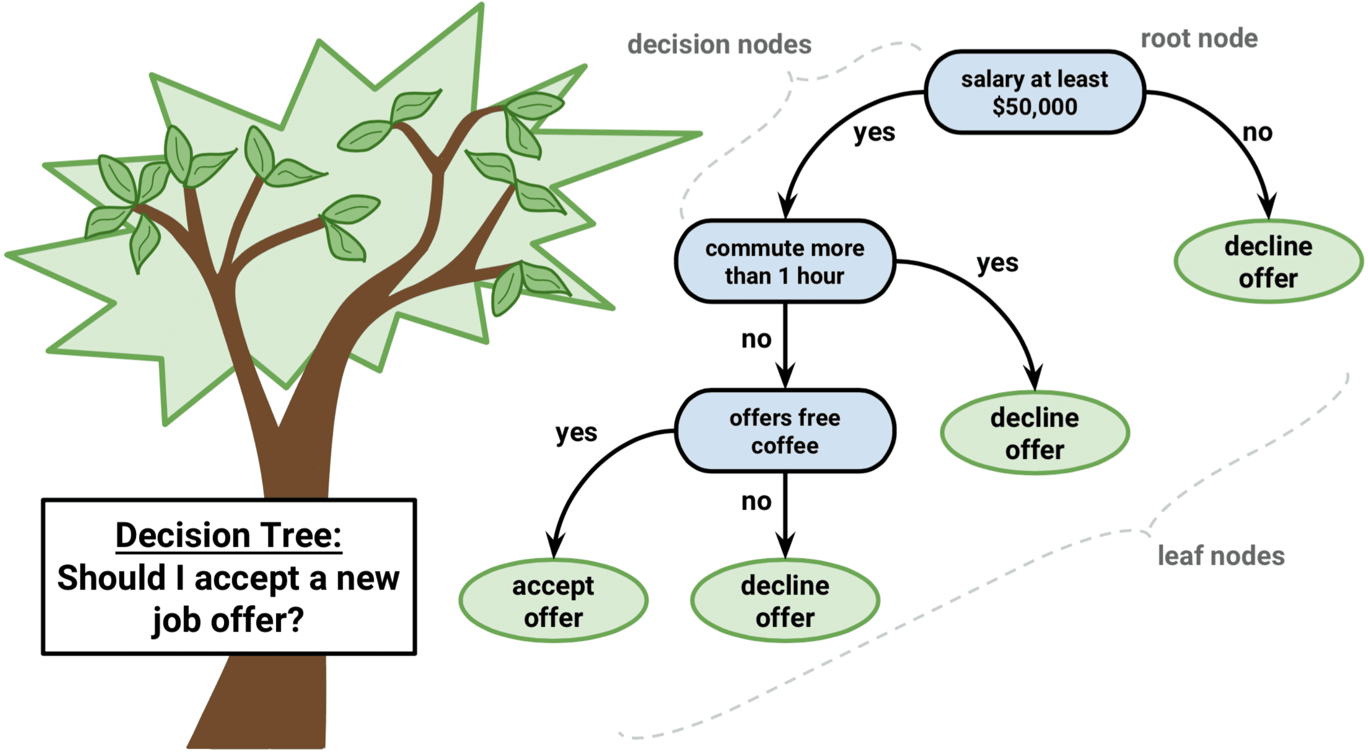
The key is how to define *Conjunctions/Nodes* that lead to Leaves, and *# of conjunctions/branches* in a tree or several trees

Decision trees for ML

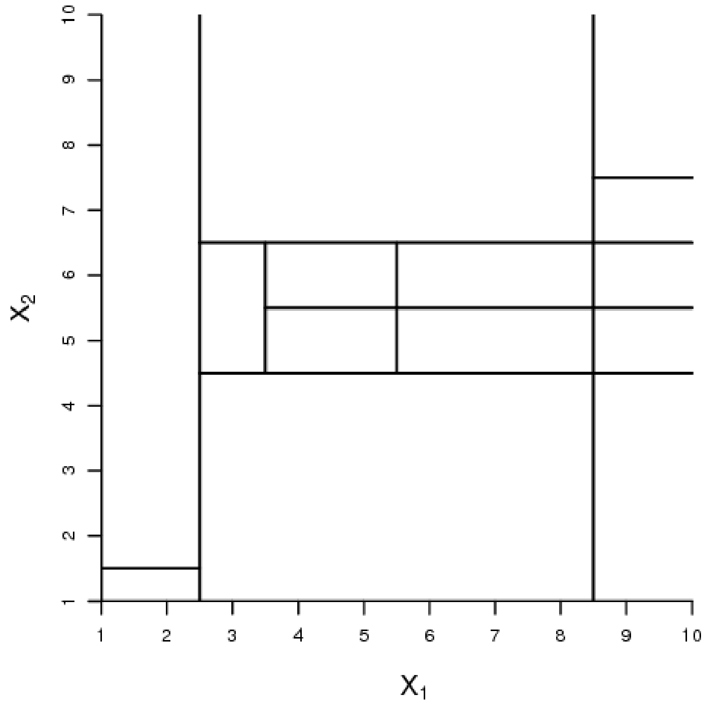
- Dataset
- Target variable
- Features
- Modelling of final results

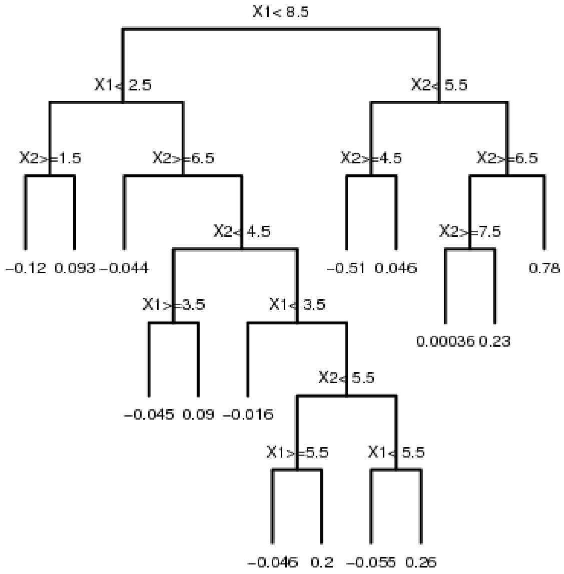


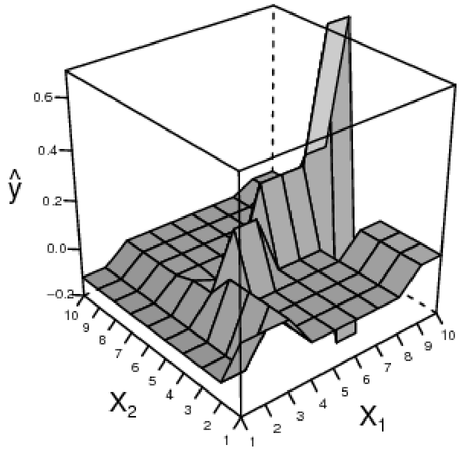
Decision trees for classification



Decision trees for regression







CART Summary

- The method of CART is to describe the target Y in terms of different features (X_1, X_2,…) associated with the target.
- Not all features should be used for conjunctions/branches
- The data spread into different leaves of a branch is often associated with a probability
- Some unused features can be used in the tree end to further model the target Y, i.e., the leaves are model classes $f(X_j, X_{j+1},\ldots)$

An example of DT algorithm

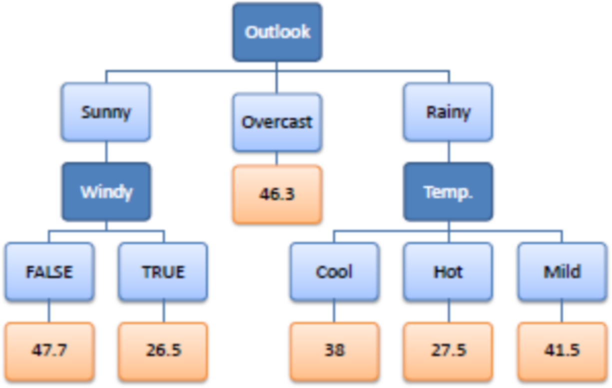
Regression DT: a complete example (1)

Decision Tree Algorithm

- Top-down greedy search
- Optimal formulation of branches and nodes by
- Gain with ***Standard Deviation Reduction***

Predictors | Target

Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



Regression DT: a complete example (2)

Decision Tree Algorithm: Standard Deviation Reduction

(1) Standard deviation for one attribute/node (only target)

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

$$Count = n = 14$$

$$Average = \bar{x} = \frac{\sum x}{n} = 39.8$$

$$Standard\ Deviation = S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} = 9.32$$

$$Coeffeicient\ of\ Variation = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

S: for tree building (branching).
CV: used to decide when to stop branching.
Avg: is the value in the leaf nodes.

Regression DT: a complete example (3)

Decision Tree Algorithm: Standard Deviation Reduction

(2) Standard deviation for two attributes (target and one predictor/feature):

$$S(T,X)=\sum_{c\in X}P(c)S(c)$$

$$SDR(T,X)=S(T)-S(T,X)$$

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

S(Hours, Outlook) = **P**(Sunny)***S**(Sunny) + **P**(Overcast)***S**(Overcast) + **P**(Rainy)***S**(Rainy)

= (4/14)*3.49 + (5/14)*7.78 + (5/14)*10.87

= 7.66

Regression DT: a complete example (4)

Decision Tree Algorithm: Standard Deviation Reduction

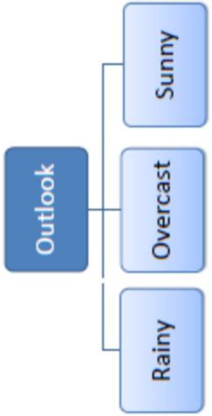
(3) The attribute/feature with the **largest standard deviation** reduction is chosen for the decision node

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		



Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

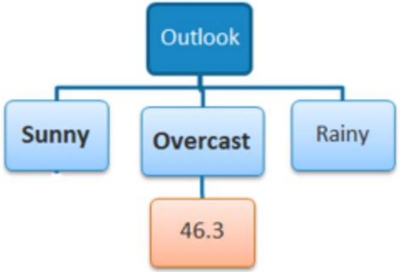
Regression DT: a complete example (5)

Decision Tree Algorithm: Standard Deviation Reduction

(4) Further splitting/branching: check if the CV is e.g., less than 10%

(5) Then we further look at each branch that require further splitting as steps (2-3)

		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5



Outlook - Sunny

Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Cool	Normal	TRUE	23
Mild	Normal	FALSE	46
Mild	High	TRUE	30
			S = 10.87
			AVG = 39.2
			CV = 28%

		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

SDR = 10.87-((2/5)*14.5 + (3/5)*7.32) = 0.678

		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

SDR = 10.87-((2/5)*7.5 + (3/5)*12.5) = 0.370

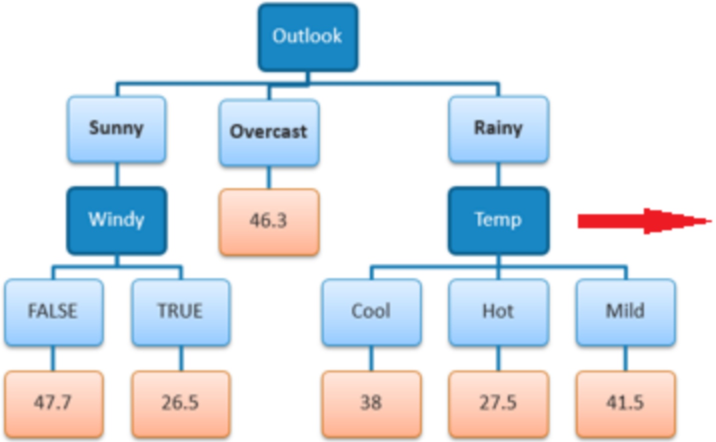
		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

SDR = 10.87-((3/5)*3.09 + (2/5)*3.5) = 7.62

Regression DT: a complete example (6)

Decision Tree Algorithm: Standard Deviation Reduction

(6) Finally, when there are only a few data left in a branch, we can stop the construction of a tree branch. The values of the leaves are taken as the average value of all the final categorized data.



Temp	Hours Played
Cool	38
Hot	25
Hot	30
Mild	35
Mild	48

CART PROS

- **Simple to understand and interpret.**
- **Requires little data preparation.**
- **Comparing to ANN**
  - Uses a white box or open-box model
  - Able to handle both numerical and categorical data
- **Comparing to statistical approaches**
  - no assumptions of the training data or prediction residuals;
  - no distributional, independence, or constant variance assumptions

CART CONS

- **Very non-robust:**

  A small change in data can lead to a large change of trees

- **Not guarantee global optimal results:**

  Learning algorithms based on heuristics such as the greedy algorithm where locally optimal decisions are made at each node.

- **Overfitting:**

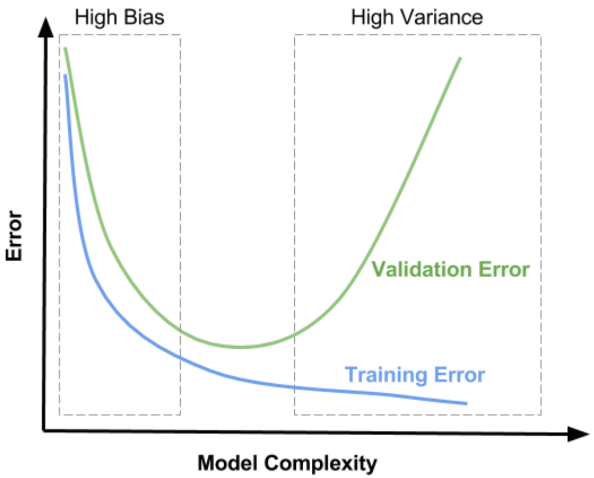
  Decision-tree learners can create over-complex trees that do not generalize well from the training data

- **Biased predictor selection:**

  Features with different numbers of levels, information gain in decision trees is biased in favour of attributes with more levels.

CART Ensemble techniques:

- Bagging methods: Boostrap/Random forecast
- Boosting techniques: AdaBoost/GradientBoost



CART ensemble methods

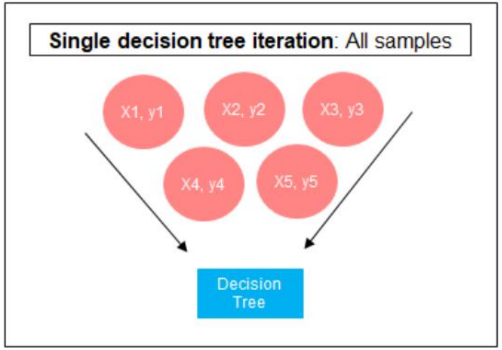
***Ensemble* methods: construct more than one decision tree to boost predictions:**

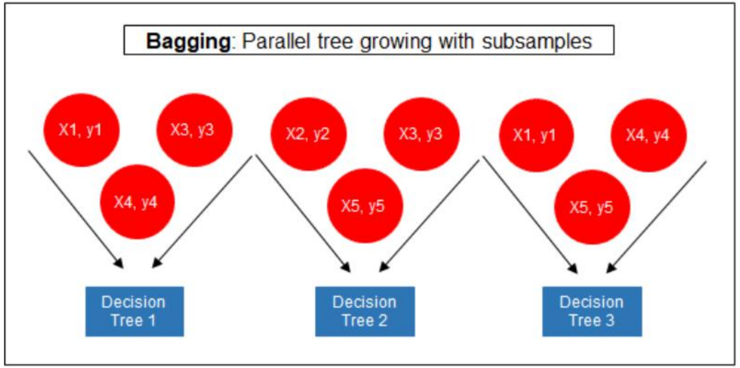
- **Bagging: Bootstrap/RF:** Builds multiple decision trees by resampling training data with replacement, and voting the trees for a consensus prediction.
- **Boosting trees** Incrementally building an ensemble by training each new instance to emphasize the training instances previously mis-modelled: AdaBoost, GBoost.
- **Rotation forest** – Every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.

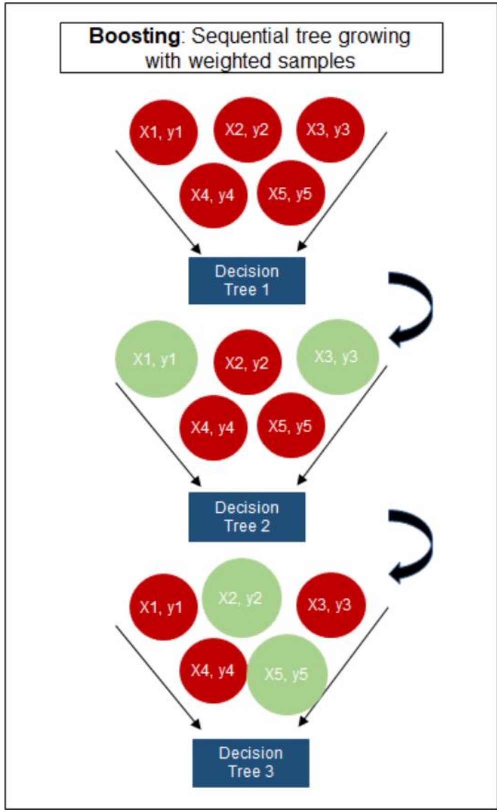
Tree based ensemble ML

One tree VS ensemble

- **Bagging**
  - Bootstrap resampling
  - Random forest
- **Boosting**
  - AdaBoost
  - XGBoost







Ensemble CART: Bagging

Bagging on the other hand refers to non-sequential learning (also called *bootstrapping*).

- For T rounds, a random subset of samples is drawn (with replacement) from the training sample.
- Each of these draws are independent of the previous round’s draw but have the same distribution. These randomly selected samples are then used to grow a decision tree (weak learner).
- *The most popular class* (or *average prediction value* in case of regression problems) is then chosen as the final prediction value.

Ensemble CART: Random forest (1)

The **pseudo code** for random forests: for t in *T* rounds (with *T* being the number of trees grown):

1. Draw a random sample *s* with replacement from the training set

2. Repeat the following steps recursively until the tree's prediction does not further improve:

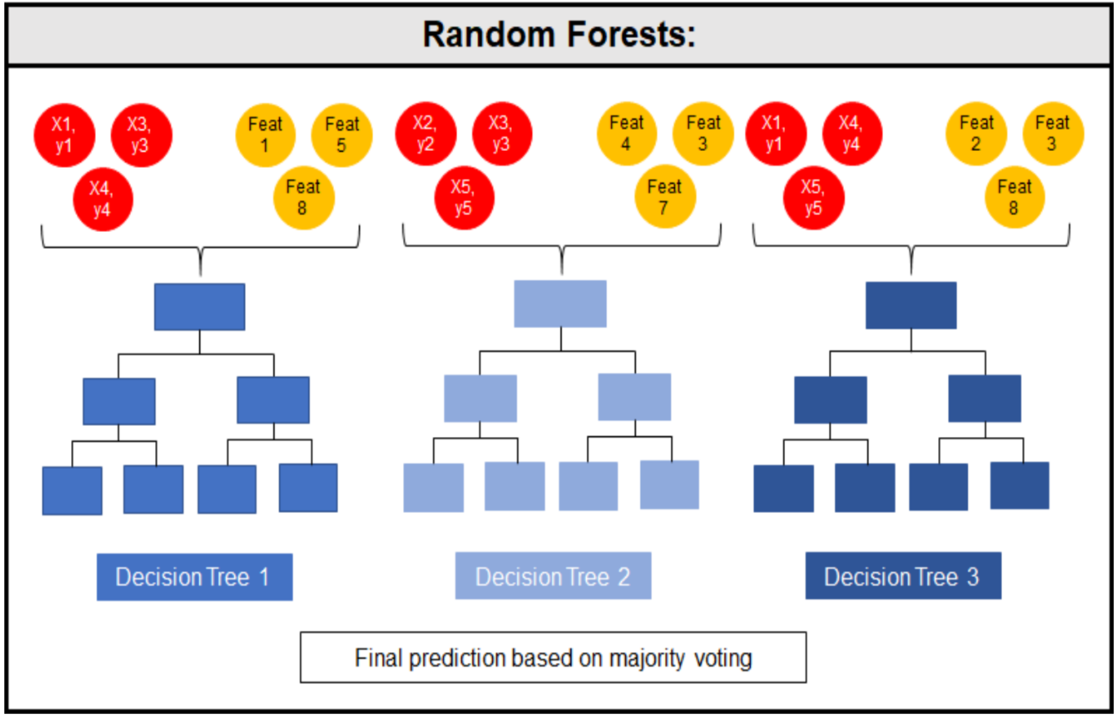
   2.1. Randomly choose *f* number of features from all available features *F*
   2.2. Choose the feature with the most information gain
   2.3. This feature is used to split the current node of the tree on

Output: majority voting or average.

Ensemble CART: Random forest (2)

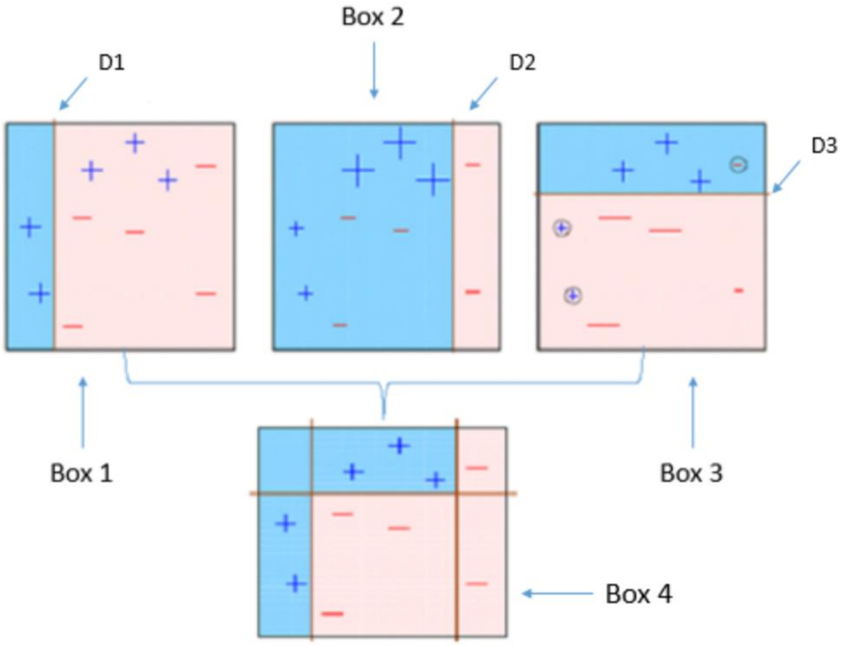
The random forests algorithm:

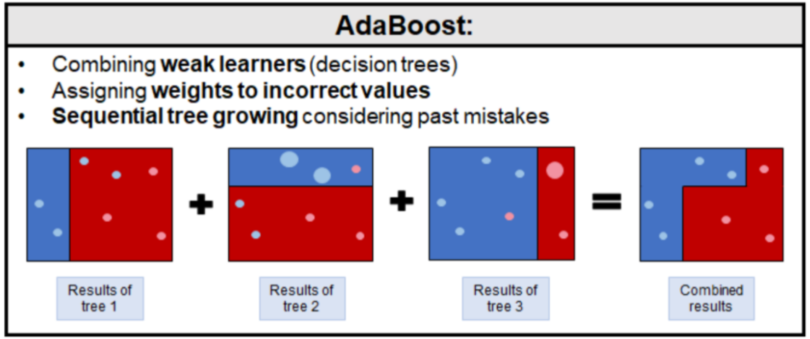
- A bagging approach.
- Bootstrapping the data by randomly choosing subsamples for each iteration of growing trees.



Ensemble CART: AdaBoost (1)

The AdaBoost algorithm is part of the family of boosting algorithms and was first introduced by Freund & Schapire in1996.





Ensemble CART: AdaBoost (2)

The **pseudo code** of the ***AdaBoost*** algorithm for a classification problem:

For *t* in *T* rounds (with *T* being the number of trees grown):

1. Calculate distribution *p* by normalizing the weight vector *w* (the initial weights in *w* for the first round are *1/N*, where *N* represents the number of labeled examples)
2. Grow a weak learner (decision tree) using the distribution *p*; return hypothesis *h* with prediction values for each example
3. Calculate error term ε of *h*
4. Assign β with $\varepsilon/(1- \varepsilon)$
5. Update the weight vector to $w = w^{*}\beta$ so that predictions with poor performance will have higher a weight and predictions with better performance will have a lower weight

- ***Output:*** final result -- a weighted majority vote of all *T* weak learners

Random Forest VS Boost trees

Difference between Random Forest and Boost trees:

- Random forests choose only a random subset of features to be included in each tree, while the former includes all features for all trees.
- Random forests reduce overfitting by combining many weak learners that underfit (only utilize a subset of all training samples).

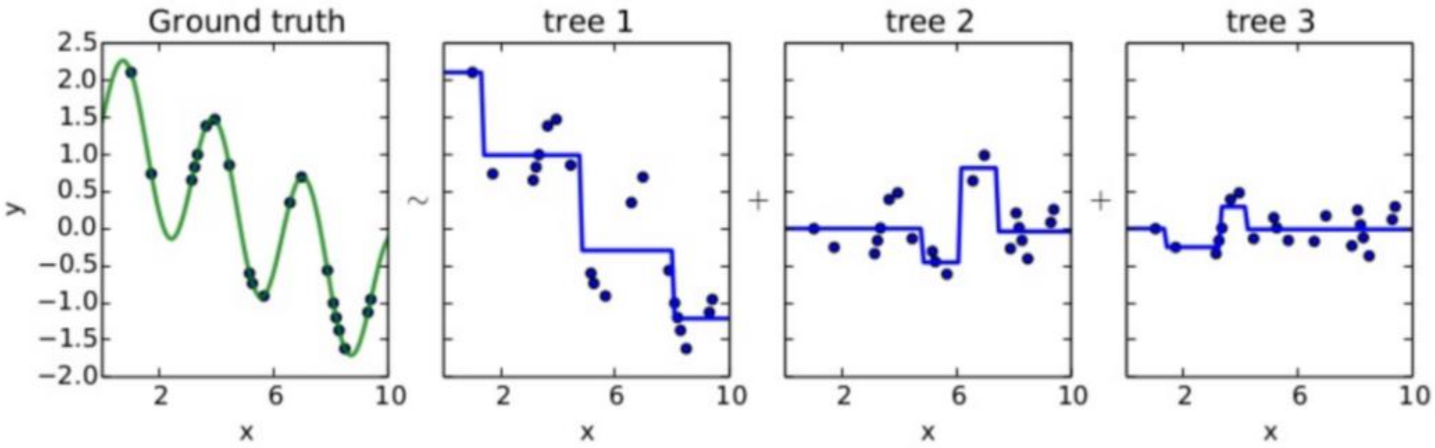
The **advantages** and **disadvantage** of RF over Boost trees:

- Less affected by noise and it generalizes better reducing variance
- More hyperparameter tuning necessary because of a higher number of relevant parameters.
- RF introduces randomness into the training and testing data which is not suitable for all data sets.

Gradient Boosting Machine (GBM)

Like Adaboost: sequentially adding predictors

Difference: fits new predictor/target to residual errors from previous predictor

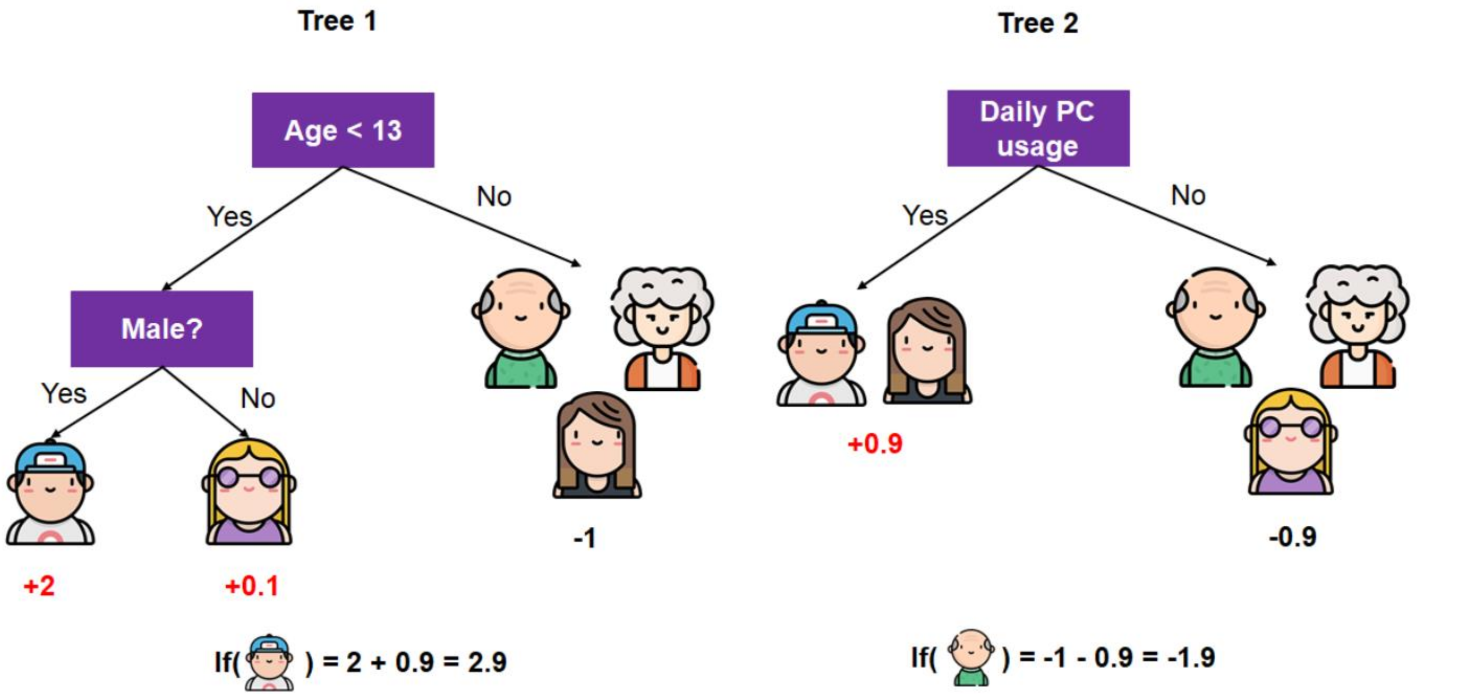


Gradient boosting method (GBM)

GBM algorithms can be implemented by the following steps:

- Let x denote feature, y denote target/predictor
- Fit a model (decision tree) to data: $F_1(x) \xrightarrow{model} y$
- Calculate the residuals/errors from above model: $\Delta y_1 = y - F_1(x)$
- Fit a next model (DT) to residuals: $h_1(x) \xrightarrow{model} \Delta y_1$
- Create a new model to original data *y*: $F_2(x) = F_1(x) + h_1(x)$
- Calculate the residuals/errors from above model: $\Delta y_2 = y - F_2(x)$
- Repeat the above steps until reaching to expectation.

Gradient Boosting: XGBoost (1)



Gradient Boosting: XGBoost (2)

XGBoost: focus on good computational speed and model performance:

- **Parallelization** of tree construction using all CPU cores.
- **Distributed Computing** for training very large models using a cluster of machines.
- **Out-of-Core Computing** for very large datasets.
- **Cache Optimization** of data and algorithm to make the best use of hardware.

AdaBoost VS XGBoost

Pros of XGBoost over AdaBoost

- AdaBoost has only a few hyperparameters that need to be tuned to improve model performance.
- Easy to understand and to visualize.
- AdaBoost performs worse when irrelevant features are included.
- It is not optimized for speed, being much slower than XGBoost.

AdaBoost is **best used** in a dataset with low noise when

- Computational complexity or timeliness of results is not a concern
- There are not enough resources for broader hyperparameter tuning due to lack of time and knowledge of the user.

CHALMERS

UNIVERSITY OF TECHNOLOGY