AI Strategy and Digital transformation
5. Tree-based models

Piotr Wójcik
University of Warsaw (Poland)
pwojcik@wne.uw.edu.pl

January 2025

Decision trees

- Decision trees are a tool used in the **regression** tasks (continuous dependent variable) and in the **classification** tasks (discrete dependent variable)
- Their algorithm is based on the division (segmentation) of the dataset into non-overlapping sets (regions, segments) based on the values of explanatory variables (predictors)
- The attractiveness of decision trees is due to the fact that they give easy-to-interpret results
- These results can be directly translated into decision rules and applied to a data set.
- The results of the division can be presented in the form of an easily interpretable **tree**

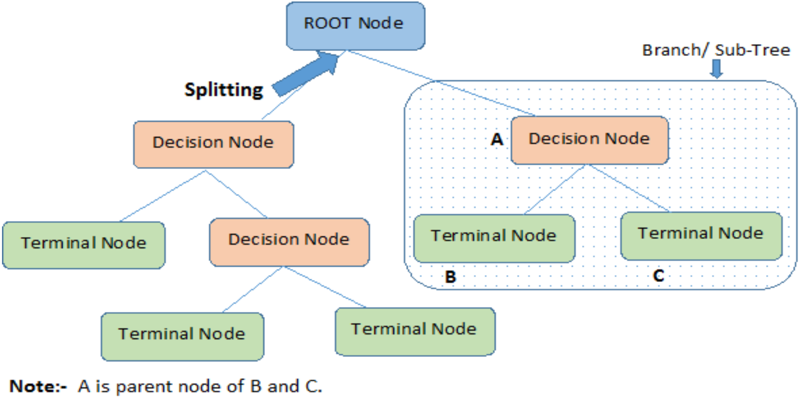
Tree construction

- the "divide et impera" rule is applied
- a tree is constructed by dividing a data set into two or more subsets of observations with respect to the values of one of the explanatory variables
- each subset is then further subdivided according to the same algorithm
- the process is repeated for subsequent subsets until some **stopping criterion** is adopted
- this recursive division creates a tree structure – in graphical form it is usually presented with "leaves down".

Terminology

- full set of data is called a **root**
- subsets create **branches**
- subsets at the lowest level that are not further divided are called **leaves** or **terminal nodes**
- any subset in the tree (including a root and a leaf) can be called a **node**

Decision tree – example



Source: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

Advantages of decision trees

- an intuitive, clear way of visualization
- may represent very complex relationships if they can be described as a series of simple conditions
- can use continuous and discrete variables
- high efficiency (speed) of algorithms
- easy to transform into a set of decision rules
- decision rules are unambiguous – it does not matter in which order they are used
- require less data cleaning compared to other predictive models
- they can easily take into account qualitative variables without the need to construct the necessary dummy variables
- are not sensitive to outliers or missing data

Disadvantages of decision trees

- only one attribute tested at each stage
- sometimes the trees are excessively complex, and the resulting complex rules
- end nodes often have a small number of observations, which can lead to random divisions
- less suitable for regression problems
- the method is exposed to the problem of **overfitting**

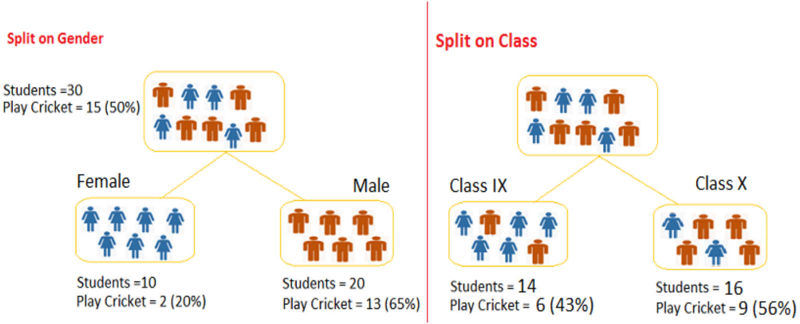
Split algorithms in decision trees

- the dataset is divided into more homogeneous parts in relation to the target variable
- at each stage, all possible divisions are analyzed according to all potential explanatory variables
- the division that gives the most homogeneous subsets is finally chosen
- the two most common recurrent (binary) splitting algorithms are:
  - **CART** (**C**lassification **A**nd **R**egression **T**ree) – by Leo Breinman
  - **C5.0** – developed by Ross Quinlan
- they differ in the measure used to assess which division is best

CART – Gini index

- CART algorithm uses a measure of node homogeneity – Gini index (**Gini purity**)
- for a binary explained variable it is calculated as $p^2 + q^2$, where p and q are the probabilities of event and non-event in the node, respectively
- **the more even** distribution in the node (close to 50%/50%), the **lower the value** of the Gini index
- its **high value** indicates that the observations for a given node have mostly the same value of the dependent variable (most of them come from the same class)
- Gini index for the considered division is calculated as the average Gini index for the resultant nodes weighted by the size of these nodes

CART – Gini index – example



Source: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

CART – Gini index – example, cont'd

Division with respect to **gender**:

- Gini for the node **Female** $= 0.2^2 + 0.8^2 = 0.68$
- Gini for the node **Male** $= 0.65^2 + 0.35^2 = 0.55$
- weighted Gini for **division with respect to gender** $= \frac{10}{30} * 0.68 + \frac{20}{30} * 0.55 = 0.59$

Podział wg klasy (**class**):

- Gini for the node **Class IX** $= 0.43^2 + 0.57^2 = 0.51$
- Gini for the node **Class X** $= 0.56^2 + 0.44^2 = 0.51$
- weighted Gini for **division with respect to class** $= \frac{14}{30} * 0.51 + \frac{16}{30} * 0.51 = 0.51$

Gini for division with respect to **gender** is higher than for **class**, therefore **gender** will be used to divide data on this stage.

The problem of overfitting decision trees

- **decision trees** for a given leaf predict the value of the target variable occurring more often in it (**mode**)
- decision and regression trees are very vulnerable to the problem of **overfitting**
- one can obtain **100% accuracy** of prediction on the training set – if the leaves contain single observations
- we can prevent this by setting the stopping criteria

Possible stopping criteria

- minimum size of a node to consider its division
- minimum leaf size
- maximum tree depth – number of levels (divisions)
- maximum number of leaves – related to the depth of the tree – for a depth of n with binary splits you can get a maximum of 2^n leaves

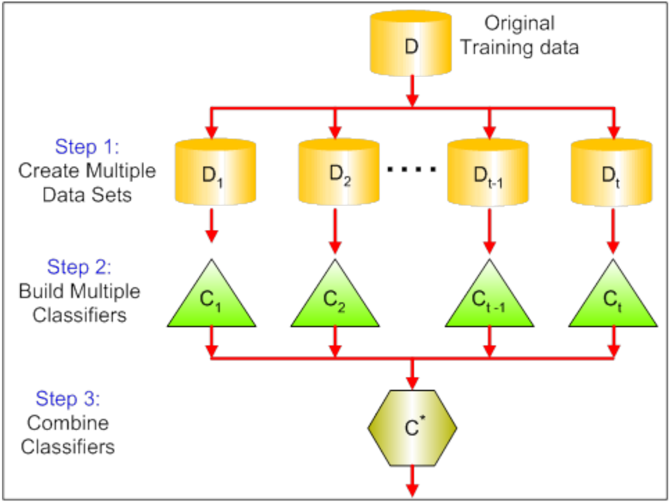
Decision trees – extensions

- decision trees usually have **worse predictions** compared to more advanced tools
- this disadvantage loses its importance in relation to the possibilities offered by extensions of tree models: **bagging**, **random forests** and **boosting**
- they are based on the construction of **many trees** and averaging the obtained results
- this approach allows one to significantly improve the accuracy of prediction
- similar extensions can also be used for other types of predictive models

What is bagging?

- **bagging** (*boostrap averaging*) is a technique used to reduce the variance of forecasts
- it consists in **multiple estimation of the same model** on different subsamples
- subsamples are created using **random sampling with replacement** of observations from the original set (so-called **bootstrap samples**)
- on each subsample **the same type of model is estimated**
- Forecasts are generated from each obtained model
- the final forecast from the model is created by **combining the results of all classifiers** – for **classification** using **majority voting**

What is bagging?



Random forests

- random forest is an extended version of bagging
- as in the *bagging procedure*, we build models using many *bootstrap subsamples* of the training set
- however during the construction of a single tree **at each division** only a **random subsample of predictors** is considered from the full set of predictors
- so the best predictor chosen for the division comes from a narrow set of variables
- at the next division in this set again the **new random subset** of predictors is considered

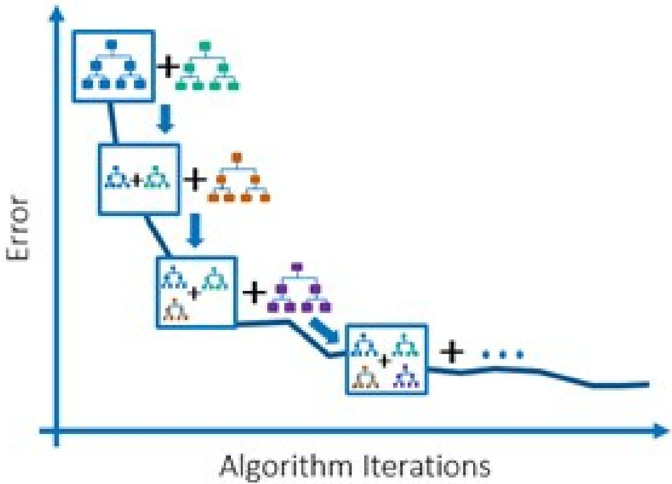
What is boosting?

- The **boosting** procedure works similarly. The difference is that trees grow in a **sequential** manner based on **information from previous trees**
- *Boosting* is often presented as a tool to transform a set of so-called *weak learners* into one **strong learner**
- **weak learner** is a relatively simple model whose classification error is slightly less than 0.5
- **strong learner** is one whose classification error is close to 0

Boosting vs. other methods of supporting trees

- Unlike the estimation of a single tree with a high degree of complexity (which is subject to **overfitting**) *boosting approach* is based on **slow learning**
- after estimating a certain decision tree, we build **a new tree on the residuals** of the previous model
- then we add a new tree to the existing one and in effect update the residuals
- each of these trees can be relatively small and can consist of only a few leaves (parameter number of divisions – d, tree depth)
- selected boosting algorithms: adaboost, gradient boosting machine (gbm), xgboost, catboost, lightGBM

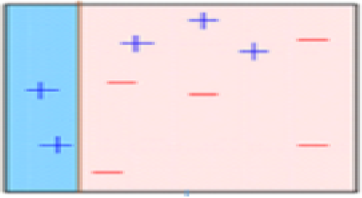
Boosting – visualization



Adaboost

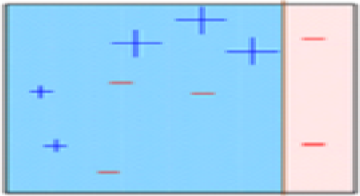
- one of the boosting algorithms is **AdaBoost** (from **Ada**ptive **Boost**ing)
- AdaBoost generates a **sequence of weak learners** assigning in each iteration **weights to individual observations**
- Observations **incorrectly classified** in k-th iteration receive **higher weights** in $(k+1)$-th iteration, and observations classified correctly have lower weights
- This means that observations which are **difficult to classify** receive **increasing weights** in such iterative process, until the algorithm classifies them correctly
- In this way, during each iteration, the algorithm "learns" all the features present in the data set, focusing on observations that are difficult to classify properly
- CAUTION! Adaboost is **very sensitive to outliers**!

Adaboost, **stage 1** – ilustration



- at the beginning all observations have **equal weights**
- we divide into (+) and (-), in this case it will be a vertical line
- three pluses (+) were incorrectly classified as minuses (-)
- and these three observations will get higher weights in the next iteration

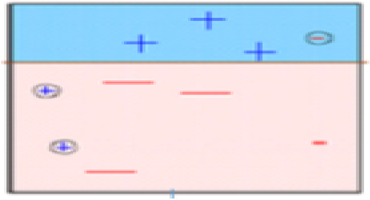
Adaboost, **stage 2** – ilustration



- sizes (weights) of three badly classified observations are greater
- the second division, classifies them correctly, but the consequence on the left is incorrect qualification of three minuses (-) as one plus (+)
- the effect is to assign higher weights to these three incorrectly classified observations in the next iteration

{Source: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/}

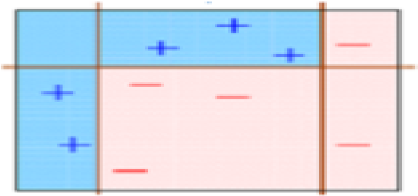
Adaboost, **stage 3** – ilustration



- three minuses (-) received higher weights
- division, this time a horizontal line, classifies them correctly
- the side effect is an incorrect classification of observations with relatively lower weights

{Source: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/}

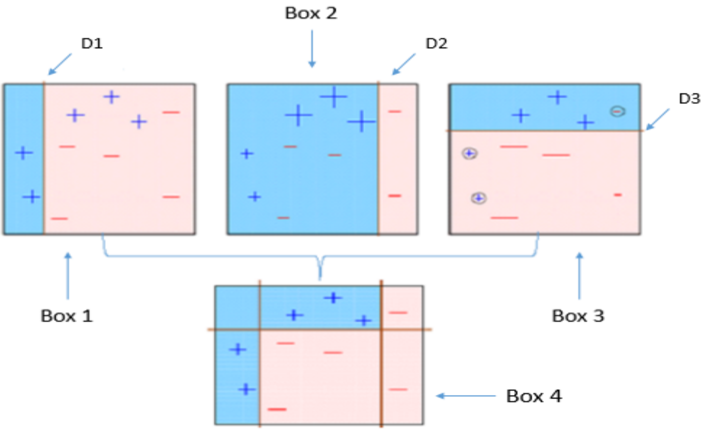
Adaboost, **stage 4** – ilustration



- in the final stage we combine the results of weak classifiers obtained in the previous stages
- we get one strong rule that classifies observations much more effectively

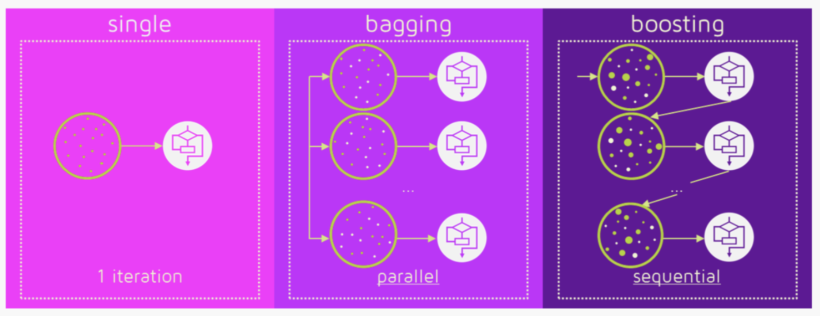
{Source: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/}

Adaboost – summary



{Source: https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/}

Tree-based models – visual summary



Tree-based models – intuitive explanation

Imagine you are a hiring manager interviewing several candidates with excellent qualifications.

- **Decision Tree**: you apply a set of criteria such as education level, number of years of experience, interview performance, etc.
- **Bagging**: instead of a single interviewer, there is an interview panel where each interviewer has a vote. It involves combining inputs from all interviewers for the final decision through a democratic voting process.
- **Random Forest**: bagging-based algorithm with a key difference where only a **subset of features is selected at random**. In other words, every interviewer will only test the interviewee on certain randomly selected qualifications.
- **Boosting**: alternative approach where each interviewer **adjusts the evaluation** criteria based on feedback from the previous interviewer.

Tree-based models – practical exercises in python

Thank you for your attention