Decision Trees

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya
Study Abroad

jordi.villa@uvic.cat

course 2023-2024

Index

Preliminary note

The material in these slides is strongly based on [1]. When other materials are used, they are cited accordingly.
Mathematical notation follows as good as it can a good practices proposal from the Beijing Academy of Artificial Intelligence.

What to expect?

In this session we will discuss:

- Decision trees
- Random Forests
- Boosting

Tree-based methods

- Simple, intuitive and powerfiul for both regression and classification
- The method divides a feature space X into smaller regions and fit a simple prediction function for each region.

  Regression eg, take the mean of the training responses associated with the training features that fall in the specific region

  Classification eg, take the majority vote among corresponding response variables.

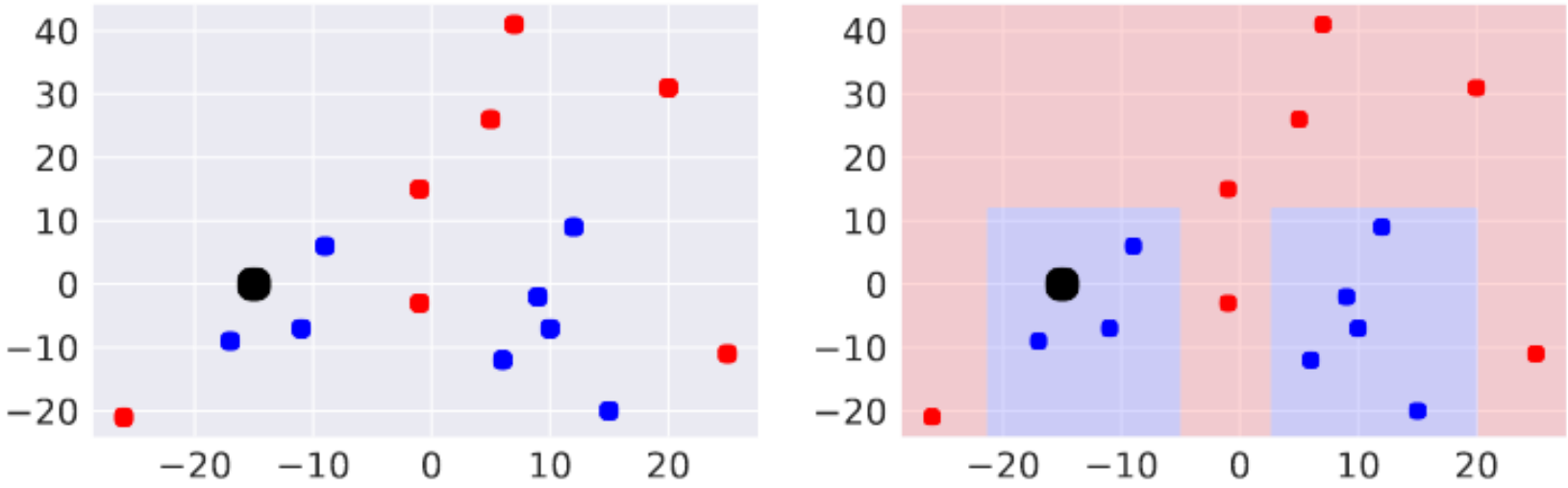
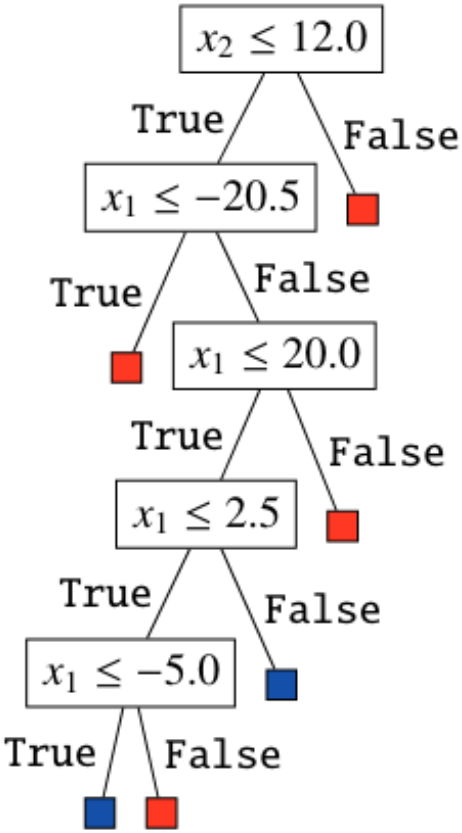


Figure 1: Left: training data and a new feature. Right: a partition of the feature space[1]

- In the example above, we cannot separate regions linearly, but we can create rectangles in the $\mathbb{R}^2$ space.
- The classifier g, thus takes a color "blue" or "red" according to where the new black dot goes.
- Both the classification procedure and the partitioning can be represented by a binary *decision tree*.

Partitioning of feature space X



The decision tree corresponding to Fig. 1[1]

- Each node ν corresponds to a region $\mathscr{R}_\nu$ of the feature space X.
- The root node is the featre space X itself.
- The final (undivided) leafs $w_1, w_2, \ldots$ (red/blue squares) form a partition of X, as they are disjoint and their union is X.
- Regional prediction functions g^w are associated with each leaf.

Decision

Let us take the input $\mathbf{x} = (x_1, x_2)^T$:

1. we start at the root node, which contains a condition $x_2 \leq 12.0$ that the input data satisifes.
2. We then proceed to the left child which contains the condition $x_2 \leq -20.5$, which our data does not satisfies, so we go the next right leaf and so on.

More generally, a binary tree $\mathbb{T}$ will partition the feature space into as many regions as leaf nodes, and the prediction function becomes:

$$g(\mathbf{x}) = \sum_{w \in \mathscr{W}} g^w(\mathbf{x}) \mathbb{1}\{\mathbf{x} \in \mathscr{R}_w\}$$

Algorithm 8.2.1: `Construct_Subtree`

Input: A node v and a subset of the training data: $\sigma \subseteq \tau$.

Output: A (sub) decision tree $\mathbb{T}_v$.

if termination criterion is met **then** `// v is a leaf node`

  Train a regional prediction function g^v using the training data σ.

else `// split the node`

  Find the best splitting rule s_v for node v.

  Create successors v_{T} and v_{F} of v.

  $\sigma_{\mathrm{T}} \leftarrow \{(\boldsymbol{x}, y) \in \sigma \ : \ s_v(\boldsymbol{x}) = \texttt{True}\}$

  $\sigma_{\mathrm{F}} \leftarrow \{(\boldsymbol{x}, y) \in \sigma \ : \ s_v(\boldsymbol{x}) = \texttt{False}\}$

  $\mathbb{T}_{v_{\mathrm{T}}} \leftarrow \texttt{Construct\_Subtree}\,(v_{\mathrm{T}}, \sigma_{\mathrm{T}})$ `// left branch`

  $\mathbb{T}_{v_{\mathrm{F}}} \leftarrow \texttt{Construct\_Subtree}\,(v_{\mathrm{F}}, \sigma_{\mathrm{F}})$ `// right branch`

return $\mathbb{T}_v$

FACULTAT DE CIÈNCIES, TECNOLOGIA I ENGINYERIES
UVIC | UVIC-UCC

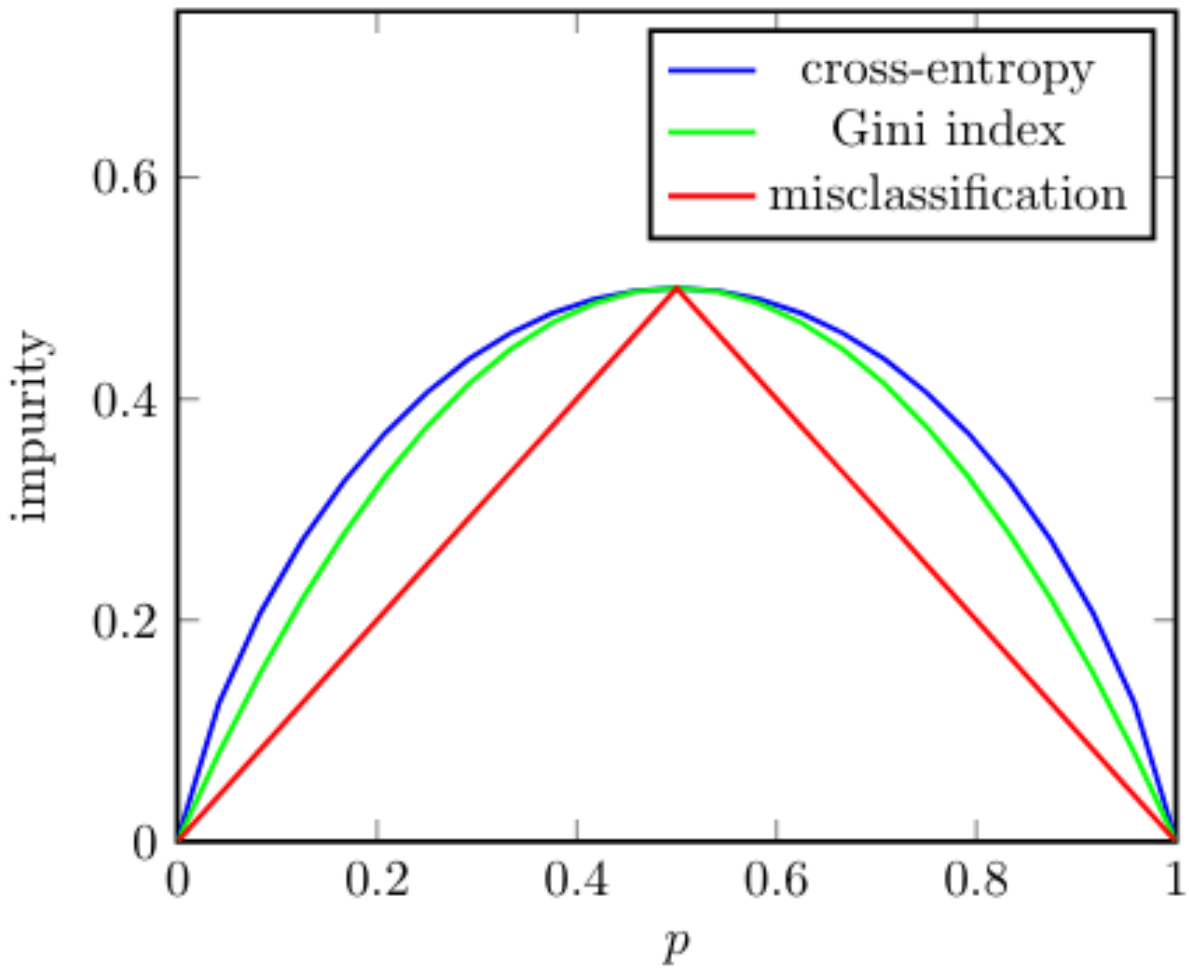


Figure 2: Entropy(normilized, divided by 2), Gini and missclassification impurities in binary classification.[1]

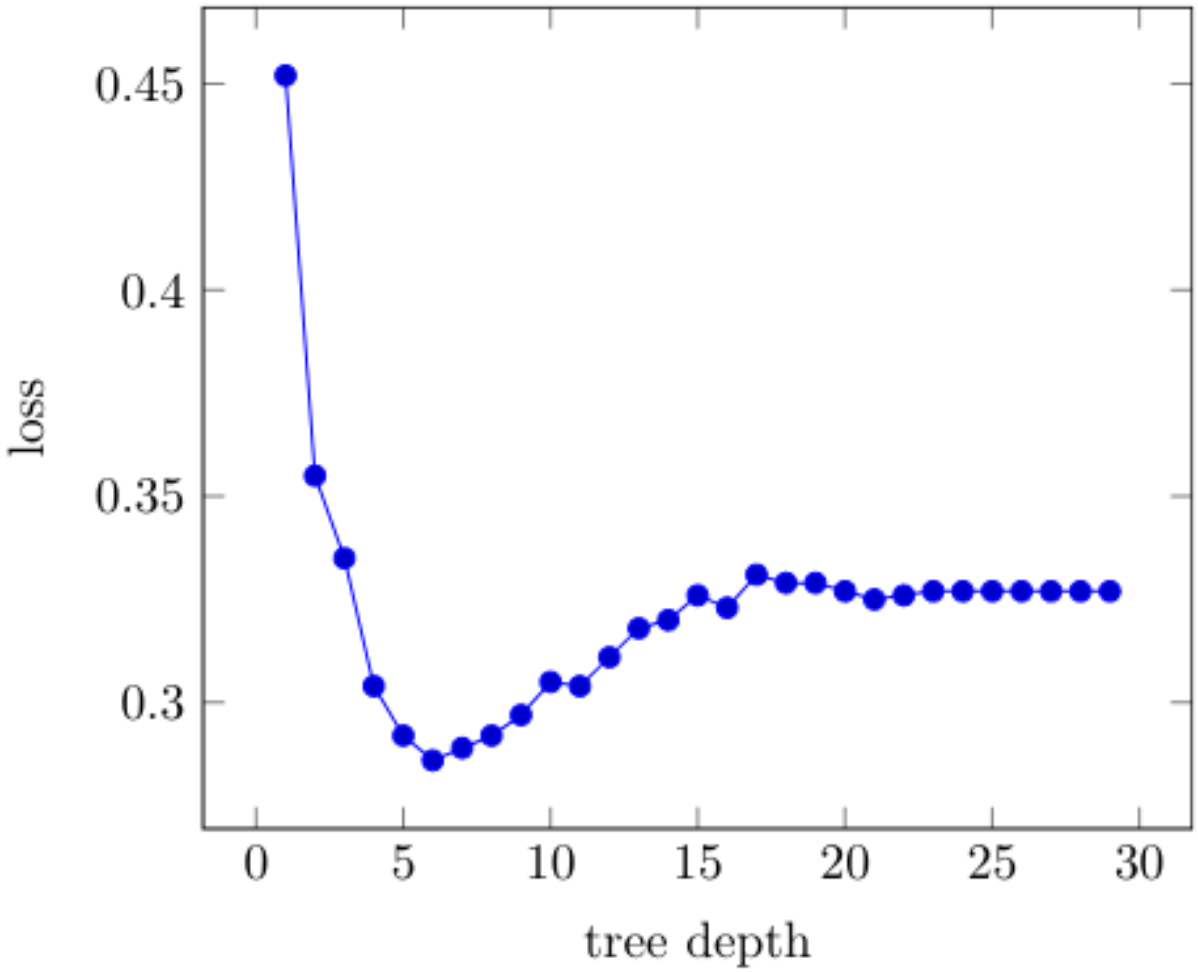


Figure 3: Ten-fold crossvalidation loss fnction as a function of the maximal tree depth in a classification problem.[1]

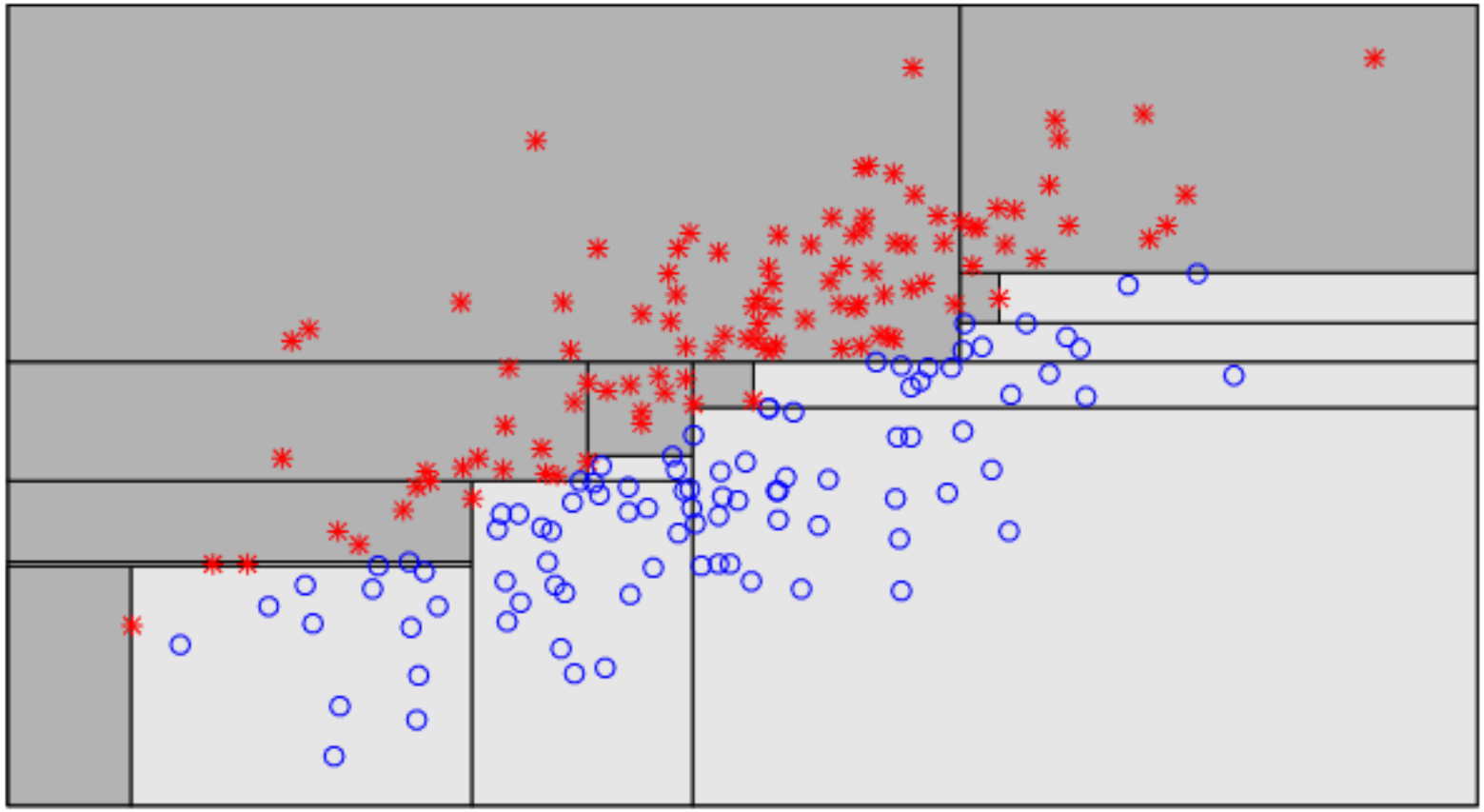


Figure 4: The two groups are separated, in decision tree, with a collection of rectangles, leading to an unnecessary classification procedure some times.[1]

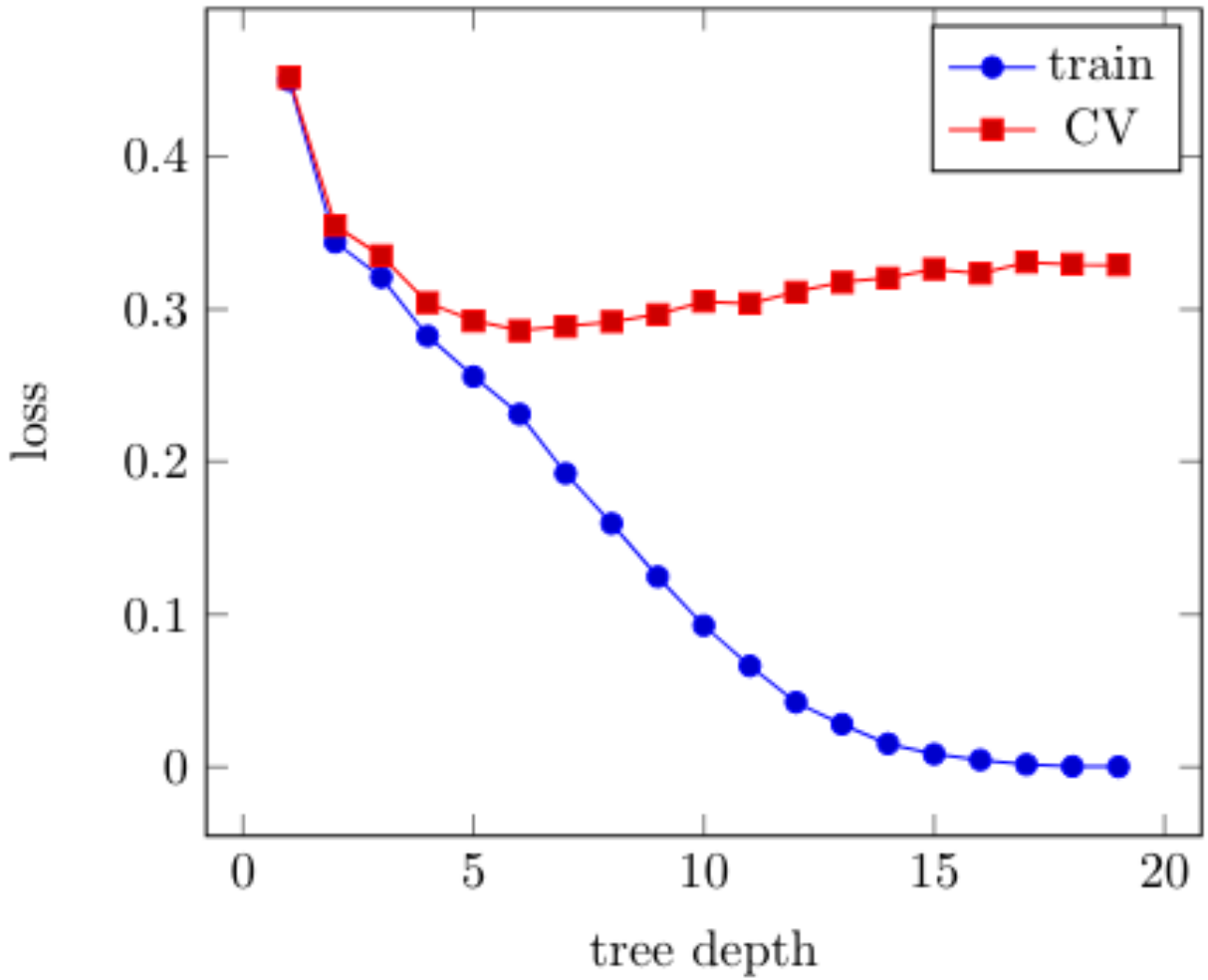


Figure 5: Cross-validation and training loss as a function of the tree depth dfor a binary classification problem.[1]

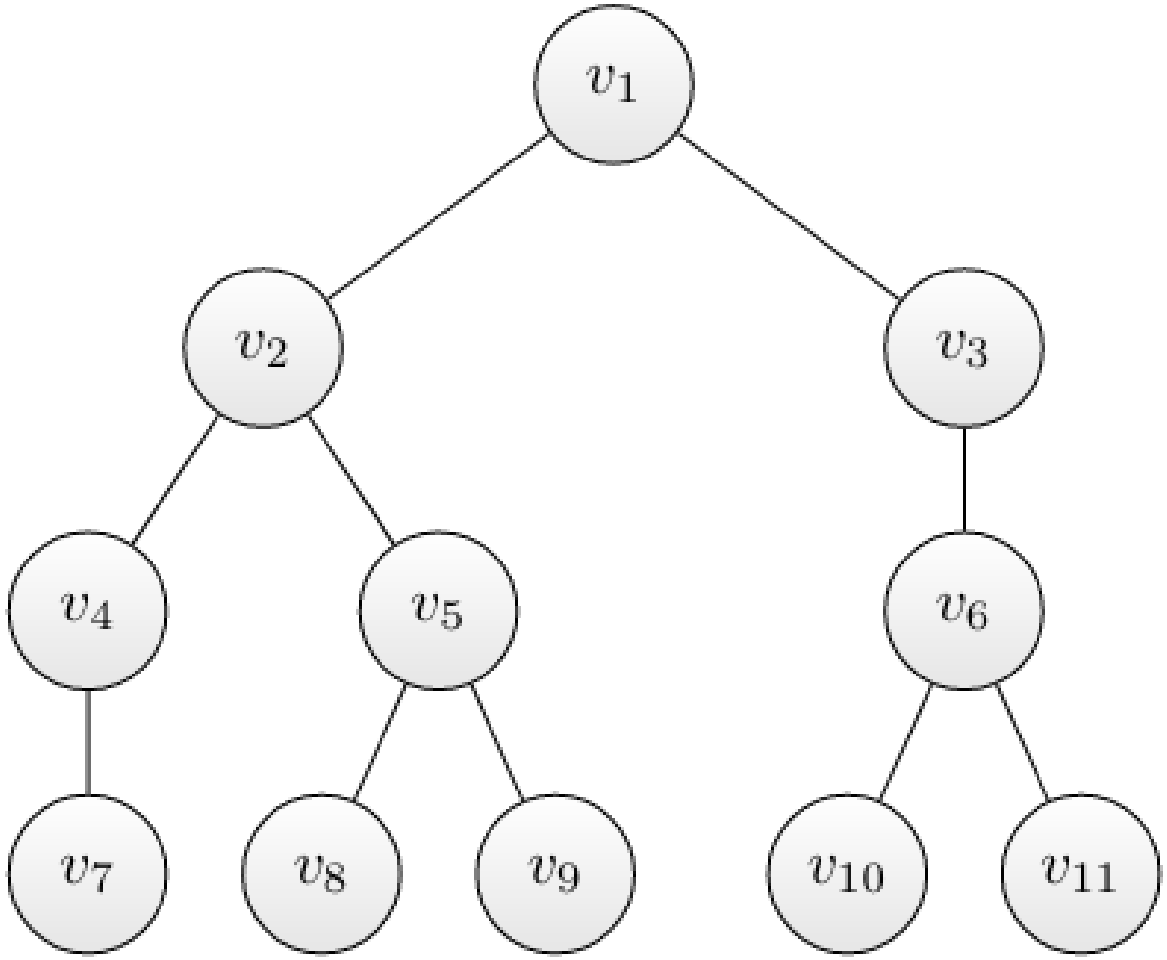


Figure 6: Different ancestors for different nodes.[1]

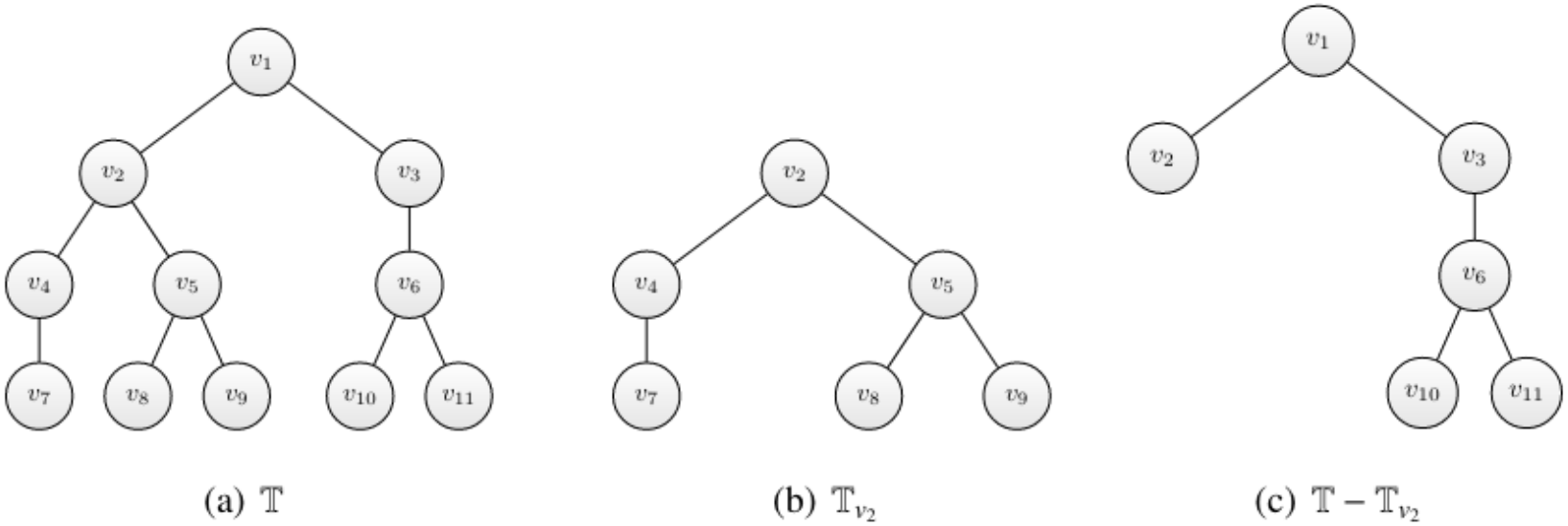


Figure 7: Pruning branches of the tree.[1]

Algorithm 8.5.1: Bootstrap Aggregation Sampling

Input: Training set $\tau = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ and resample size B.

Output: Bootstrapped data sets.

for $b = 1$ **to** B **do**
  $\mathcal{T}_b^* \leftarrow \emptyset$
  for $i = 1$ **to** n **do**
    Draw $U \sim \mathcal{U}(0, 1)$
    $I \leftarrow \lceil nU \rceil$ `// select random index`
    $\mathcal{T}_b^* \leftarrow \mathcal{T}_b^* \cup \{(\boldsymbol{x}_I, y_I)\}$.
return $\mathcal{T}_b^*, b = 1, \ldots, B$.

FACULTAT DE CIÈNCIES, TECNOLOGIA I ENGINYERIES
UVIC | UVIC-UCC

Algorithm 8.5.2: Out-of-Bag Loss Estimation

Input: The original data set $\tau = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, the bootstrapped data sets $\{\mathcal{T}_1^*, \ldots, \mathcal{T}_B^*\}$, and the trained predictors $\left\{g_{\mathcal{T}_1^*}, \ldots, g_{\mathcal{T}_B^*}\right\}$.

Output: Out-of-bag loss for the averaged model.

for $i = 1$ **to** n **do**
  $C_i \leftarrow \emptyset$ // Indices of predictors not depending on $(\boldsymbol{x}_i, y_i)$
  for $b = 1$ **to** B **do**
    if $(\boldsymbol{x}_i, y_i) \notin \mathcal{T}_b^*$ **then** $C_i \leftarrow C_i \cup \{b\}$
  $Y_i' \leftarrow |C_i|^{-1} \sum_{b \in C_i} g_{\mathcal{T}_b^*}(\boldsymbol{x}_i)$
  $L_i \leftarrow \text{Loss}\left(y_i, Y_i'\right)$
$L_{\text{OOB}} \leftarrow \frac{1}{n} \sum_{i=1}^{n} L_i$
return L_{OOB}.

Algorithm 8.6.1: Random Forest Construction

Input: Training set $\tau = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, the number of trees in the forest B, and the number $m \leqslant p$ of features to be included, where p is the total number of features in $\boldsymbol{x}$.

Output: Ensemble of trees.

1 Generate bootstrapped training sets $\{\mathcal{T}_1^*, \ldots, \mathcal{T}_B^*\}$ via Algorithm 8.5.1.

2 **for** $b = 1$ **to** B **do**

3 Train a decision tree $g_{\mathcal{T}_b^*}$ via Algorithm 8.2.1, where each split is performed using m randomly selected features out of p.

4 **return** $\{g_{\mathcal{T}_b^*}\}_{b=1}^{B}$.

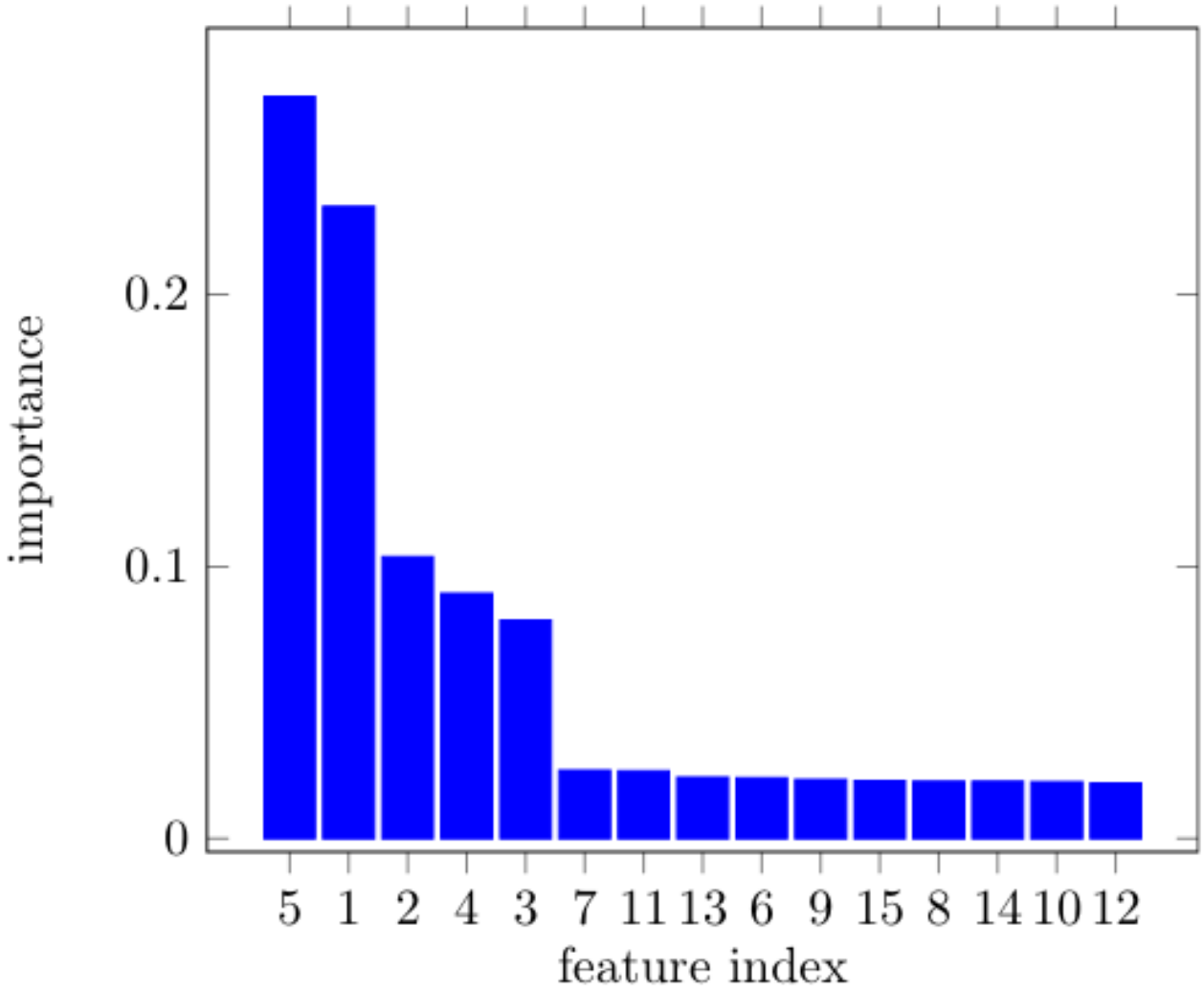


Figure 8: Importance measure for a 15-feature data set with only informative features x_1, x_2, x_3, x_4, x_5[1].

Algorithm 8.7.1: Regression Boosting with Squared-Error Loss

Input: Training set $\tau = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the number of boosting rounds B, and a shrinkage step-size parameter γ.

Output: Boosted prediction function.

1 Set $g_0(\boldsymbol{x}) \leftarrow n^{-1} \sum_{i=1}^n y_i$.

2 **for** $b = 1$ **to** B **do**

3 Set $e_i^{(b)} \leftarrow y_i - g_{b-1}(\boldsymbol{x}_i)$ for $i = 1, \ldots, n$, and let $\tau_b \leftarrow \left\{\left(\boldsymbol{x}_i, e_i^{(b)}\right)\right\}_{i=1}^n$.

4 Fit a prediction function h_b on the training data τ_b.

5 Set $g_b(\boldsymbol{x}) \leftarrow g_{b-1}(\boldsymbol{x}) + \gamma\, h_b(\boldsymbol{x})$.

6 **return** g_B.

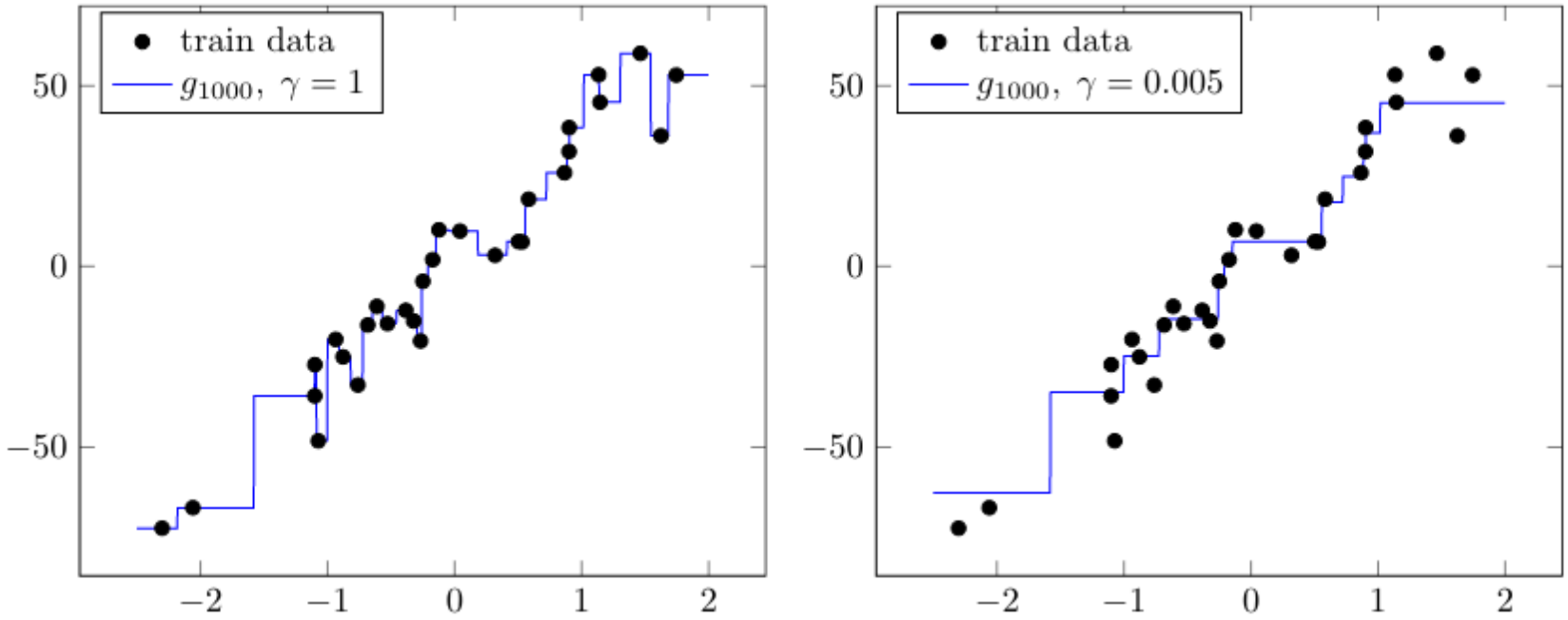


Figure 9: Fitted boosting regression model with two different values of γ. [1]

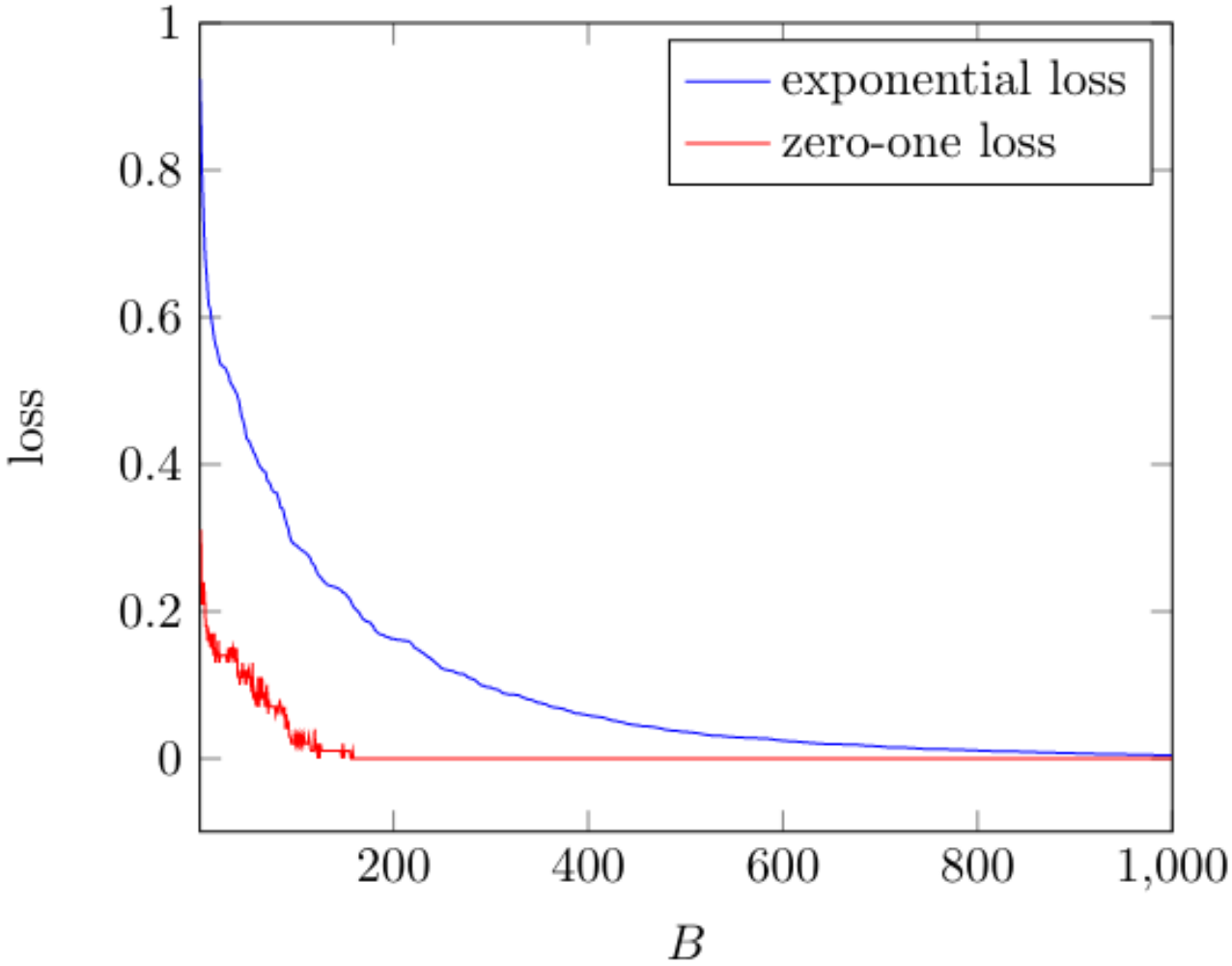


Figure 10: Exponential and zero-one training loss as a function of the number of boosting rounds B for a binary classification problem. [1]


Dirk P. Kroese, Zdravko Botev, Thomas Taimre, and Radislav: Vaisman.
Data Science and Machine Learning: Mathematical and Statistical Methods.
Machine Learning & Pattern Recognition. Chapman & Hall/CRC, 2020.