Decision Trees

Introduction to Machine learning

Ellen Kartysheva, Alisa Alenicheva

Saint-Petersburg,
2022

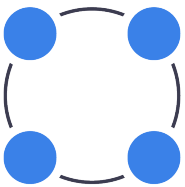


Table of Contents

Table of Contents

Decision tree

- Decision tree is a tree of applied decision rules
- Each internal node contains some test for a certain feature
- Each edge corresponds to a test outcome
- Each leaf contains a final prediction

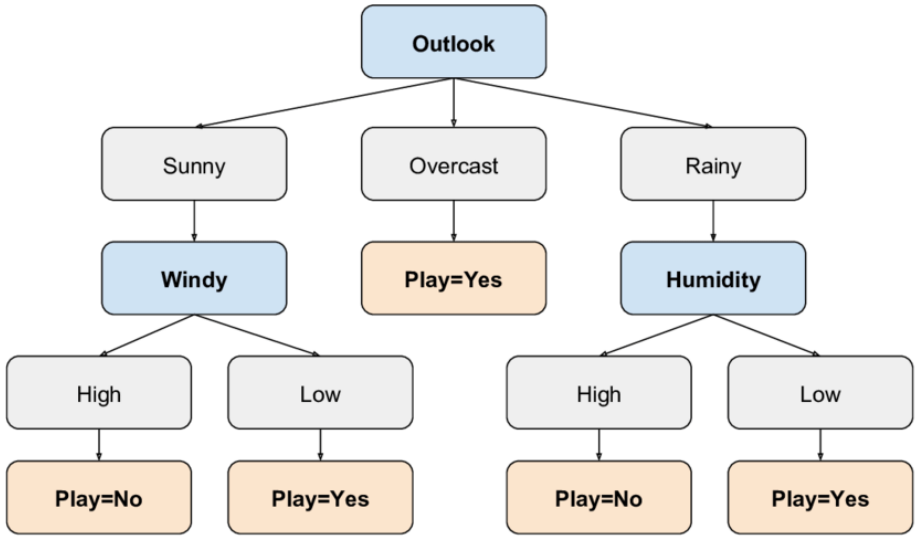


Figure: Decision tree for playing tennis

Divide-And-Conquer algorithm

A traditional way to learn a decision tree is to use some variant of Divide-And-Conquer algorithm.

1. Select a test for root node and create a branch for each possible outcome of the test.
2. Split instances into subsets for each branch according to the test.
3. Repeat recursively for each branch, using only instances that reach the branch.
4. Stop recursion for a branch if all its instances have the same class

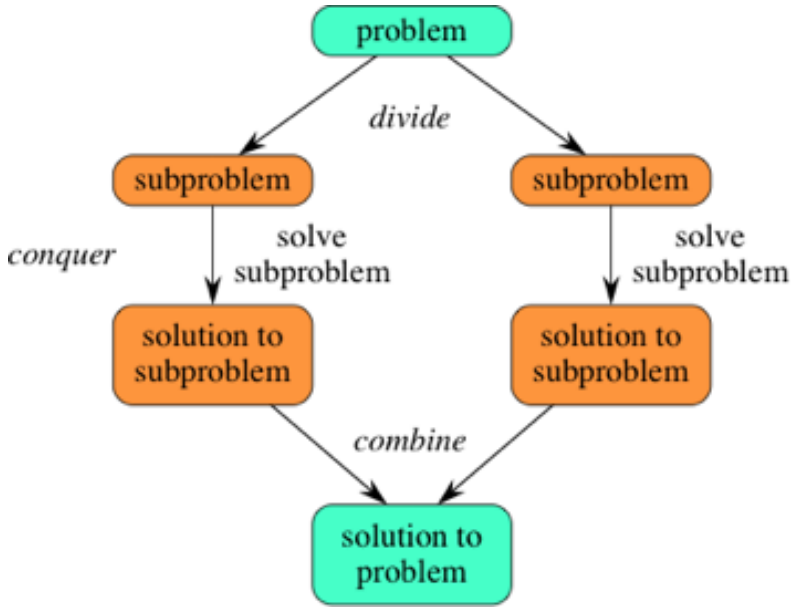


Figure: Scheme of Devide-And-Conquere algorithm

Table of Contents

Binary Decision Tree

- Decision tree is often a binary decision tree
- Each internal node u is a predicate $p_u : X \to \{0, 1\}$
- Each edge corresponds to 1 or 0 outcome of predicate p_u
- Each leaf contains a final prediction

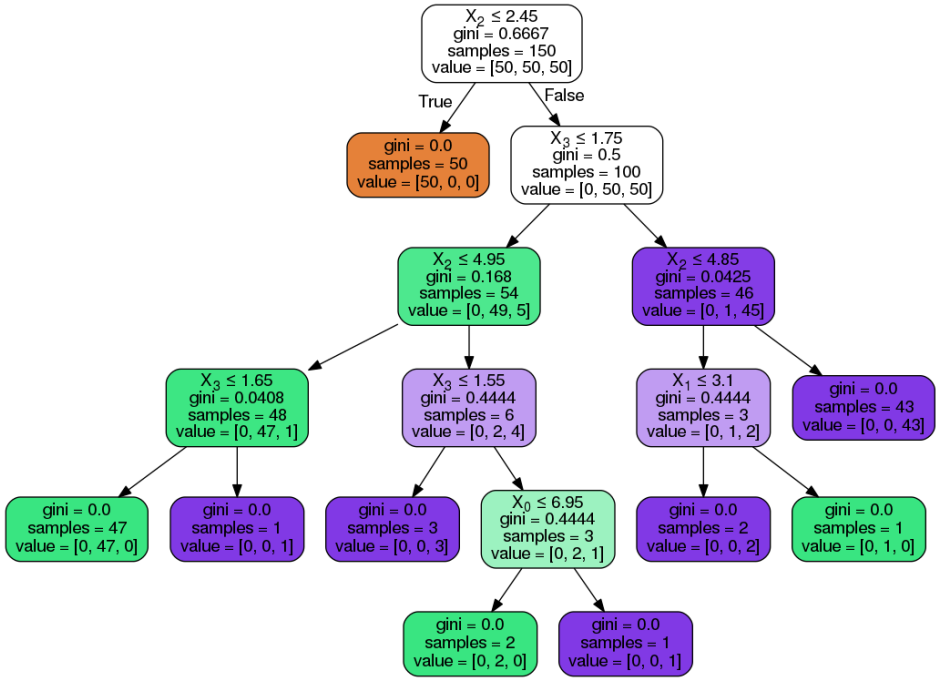


Figure: Binary decision tree

How to build and apply a decision tree?

Greedy learning of binary DT

Let's consider a binary classification problem. Input: training set $S = \{x_i, y_i\}_{i=1}^{n}$. Output: decision tree

1. Set $S_{node} = S$
2. Greedy split S_{node} into S_{right} and S_{left} optimizing a given loss $Q(S_{node}, j, t) \to min_{(j,t)}$:

$$S_{left} = \{(x_i^{(j)}, y_i) \in S_{node} | x^{(j)} \leq t\}$$

$$S_{right} = \{(x_i^{(j)}, y_i) \in S_{node} | x^{(j)} \leq t\}$$

3. Create internal node u corresponding to the predicate $1[x_i^{(j)} < t]$
4. If a stopping criterion is satisfied for u, declare it a leaf, setting some $c_u \in Y$ as leaf prediction
5. If not, repeat 2–4 for $S_{left}(j, t)$ and $S_{right}(j, t)$

Design choices for learning a decision tree classifier

The following parameters may be varied in the greedy algorithm:

- Type of predicate in internal nodes
- The loss function $Q(S_{node}, j, t)$
- The stopping criterion

What are the alternatives?

Table of Contents

Types of predicates

Types of predicates p_u in the internal nodes:

- $p_u = [x_i \geq b]$
- $p_u = [a \leq x_i \leq b]$
- $p_u = [\sum_j [a_j \leq x_i \leq b_j] \leq d]$
- $p_u = [\sum_j w_j x_j \geq w_0]$
- $p_u = [\rho(x, x_0) \leq w_0]$

How to choose the right split value?

Split Evaluation: Information Gain

What is a good attribute to split the data in the internal node u?

- The best attribute maximizes the "purity" of the split:

$$Q(S_{node}, j, t) = \frac{|S_{node}|}{|S_{total}|} Im(node) - \frac{|S_{right}|}{|S_{total}|} Im(right) - \frac{|S_{left}|}{|S_{total}|} Im(left)$$

- Maximize purity = minimize Impurity
- There are different alternatives for Impurity metric

Impurity metrics (1)

Classification:

- Misclassification Error: $Im(node) = 1 - max(p(y))$
- Entropy: $Im(node) = -\sum_{y \in Y} p(y) log_2(p(y))$. Information Gain (IG).
- Gini impurity: $Im(node) = \sum_{y \in Y} p(y)(1 - p(y))$. Gini Gain (GG).

Regression:
$\sum_{x_i \in X} \sum_{x_j \in X} \frac{1}{2}(y_i - y_j)^2$

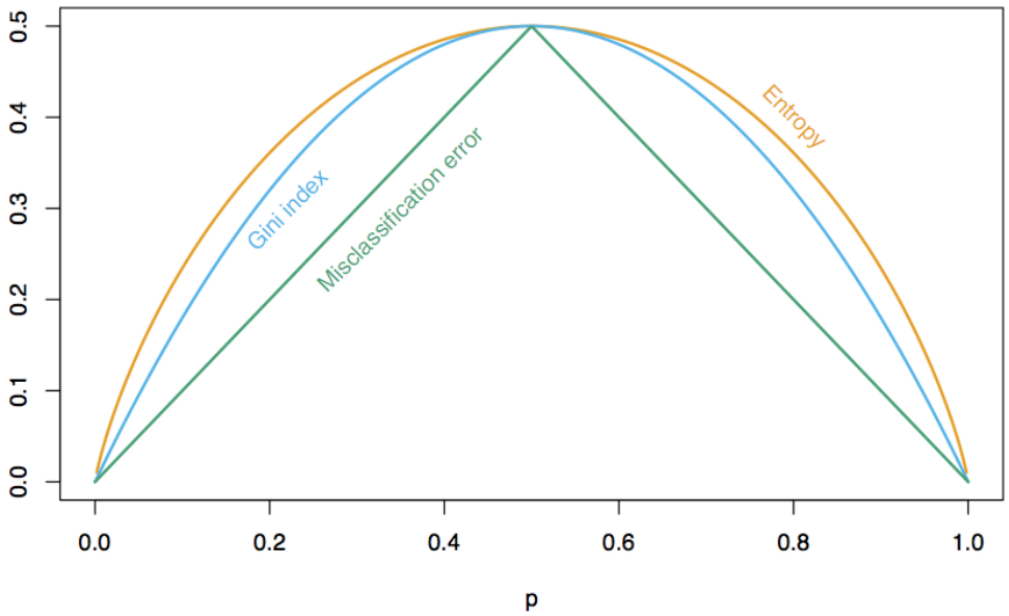


Figure: Impurity metrics for classification

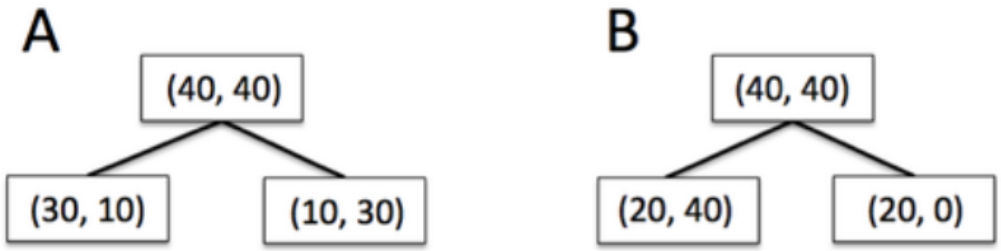
Impurity metrics (2)

- Entropy: subsets are more likely to be pure if there is a large number of different attribute values. Information gain is biased towards choosing attributes with a large number of values.
- Gain ratio takes number and size of branches into account when choosing an attribute attribute:

$$GainRatio = \frac{IG(S_{node}, j)}{SplitInfo(S_{node}, j)}$$

$$SplitInfo(S_{node}, j) = -\sum_{i \in values(j)} \frac{|S_i|}{|S_{node}|} log_2 \frac{|S_i|}{|S_{node}|}$$

Example of different impurity metrics



Misscalssification Error:

- $I(A) = 1 \cdot 0.5 - 0.5 \cdot 0.25 - 0.5 \cdot 0.25 = 0.25$
- $I(B) = 1 \cdot 0.5 - 0.75 \cdot 0.33 - 0.25 \cdot 0 = 0.25$

Entropy:

- $I(A) = 1 \cdot 1 - 0.5 \cdot 0.81 - 0.5 \cdot 0.81 = 0.19$
- $I(B) = 1 \cdot 1 - 0.75 \cdot 0.92 - 0.25 \cdot 0 = 0.31$

Gini:

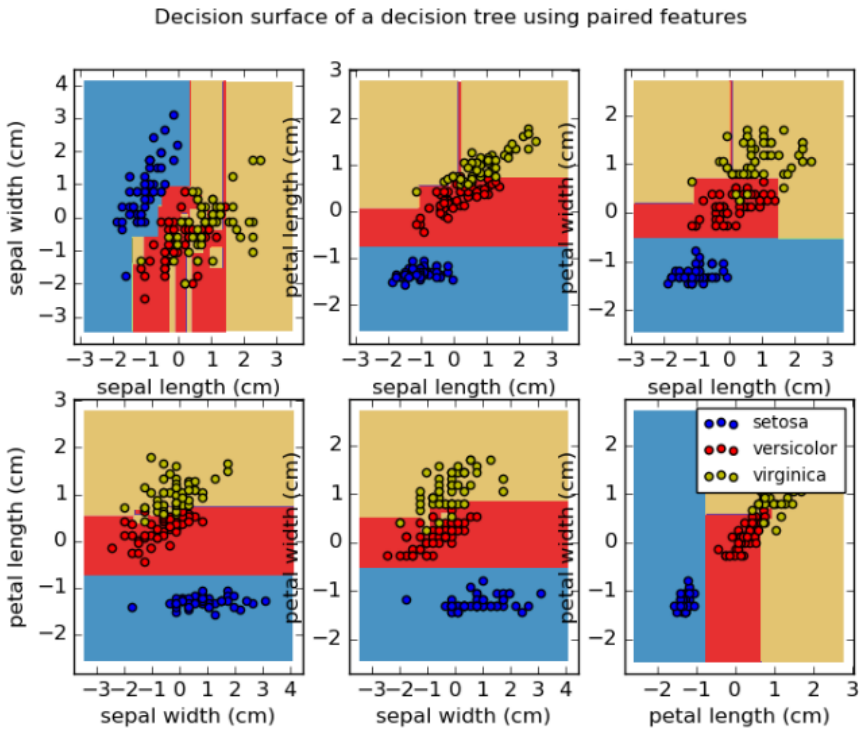
- $I(A) = 1 \cdot 0.5 - 0.5 \cdot 0.375 - 0.5 \cdot 0.375 = 0.125$
- $I(B) = 1 \cdot 0.5 - 0.75 \cdot 0.4 - 0.25 \cdot 0 = 0.16$

Table of Contents

Algorithms

- ID3 is one of the first algorithms. Allows only nominal variables, no missing values.
- C4.5 algorithm is an improvement of ID3. Allows numeric attributes and missing values. Uses error-based pruning.
- CART is very similar to C4.5, but is able to solve regression task.
  1. If all objects in the node belong to the same class, mark the leaf as this class and stop.
  2. Find the threshold with the best information gain. If no threshold yields information gain, mark the node with the majority class (or assign class probability) and stop
  3. Separate objects into children nodes by the threshold rule
  4. Call 1 for every new node

CART example



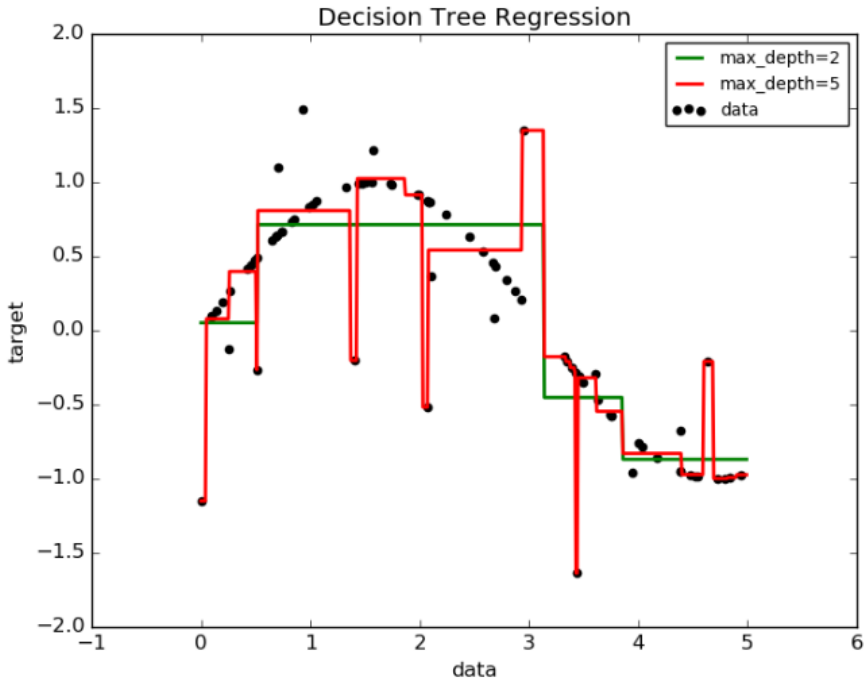


Figure: Example of classification and regression with DT

Table of Contents

Missing values

There are different strategies to deal with missing values:

- Ignore features with missing values (ID3)
- For nominal variables treat missing value as a separate category
- Use all children of the node (C4.5). Will discuss later in Fuzzy Decision Trees
- Use the majority branch

Regularization on Decision Tree

- Pruning is a name of regularization techniques for DTs
- General idea: the smaller the complexity of a concept, usually the better its generalization ability is
- Pre-pruning stops growing a branch when information becomes unreliable. Fast, but can stop too early.
  - Based on statistical significant test. Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node.
- Post-pruning prunes the tree after it's built
  - Minimum error pruning: remove nodes while validation error is not increasing.
  - Cost complexity pruning: remove nodes while validation error is not increasing by more than α. In other words, introduce a new, cumulative error function: $[error] + \alpha[size]$

Table of Contents

Advantages and Disadvantages

Pros:

- Interpretability
- Handling different types of data
- Handling missing values

Cons:

- Overfit super-easily
- Instability to noise

Table of Contents

Pareto efficiency

- The Pareto frontier is the set of all Pareto efficient allocations, conventionally shown graphically
- A classifier is Pareto efficient if there is no better classifier in terms of both precision and recall.

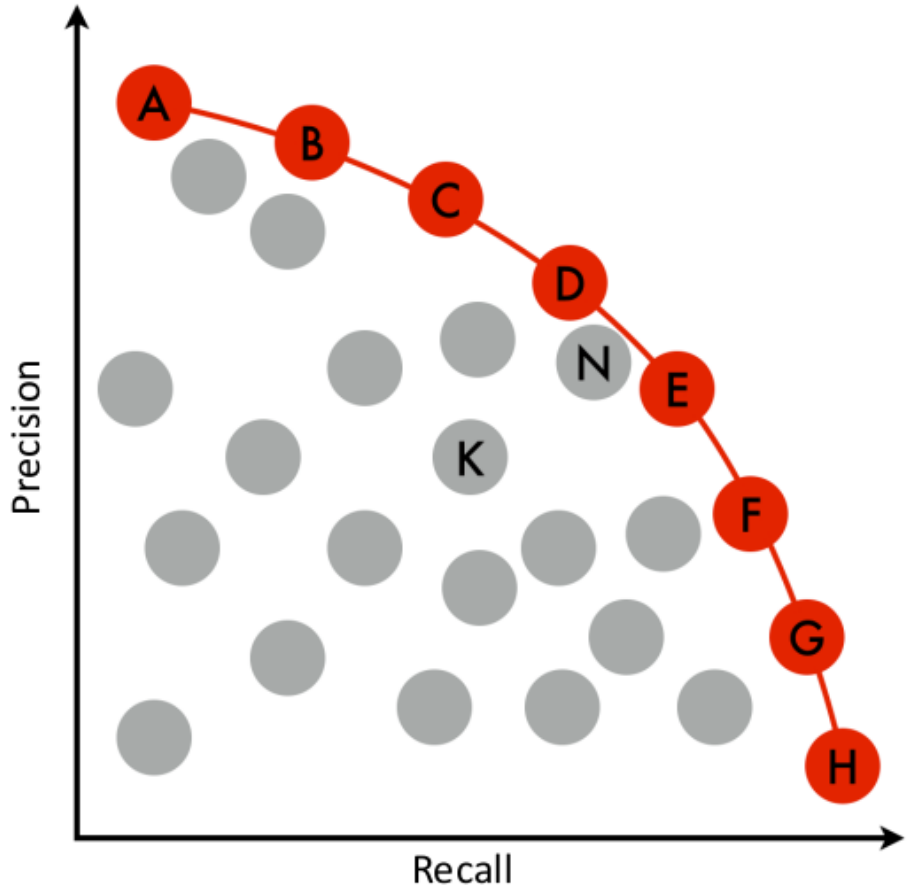


Figure: Pareto frontier

Precision-Recall curve

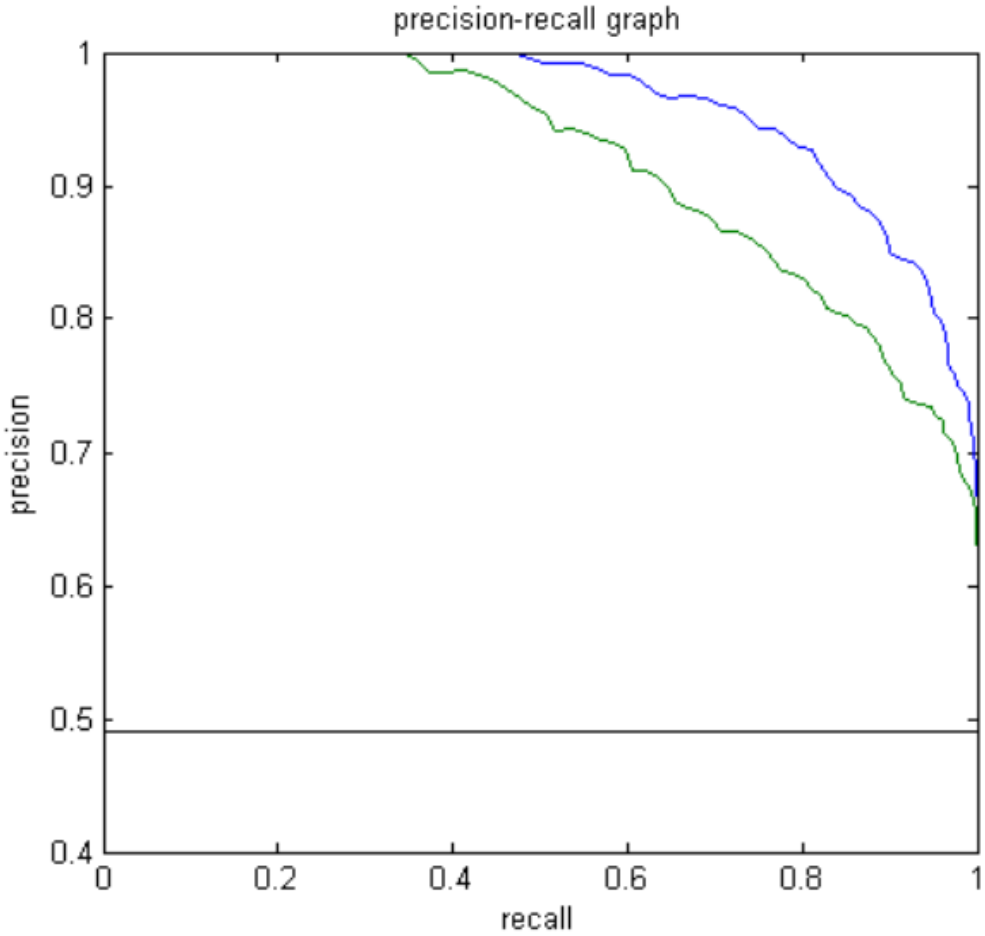


Figure: Precision-Recall curve

ROC curve

- Each point of the curve corresponds to some classifier
- Abscissa: FPR as a function of the threshold
- Abscissa: TPR as a function of the threshold
- (1 – FPR) is called specificity, TPR is called sensitivity
- AUC is an Area Under ROC-curve, used to characterize classification accuracy

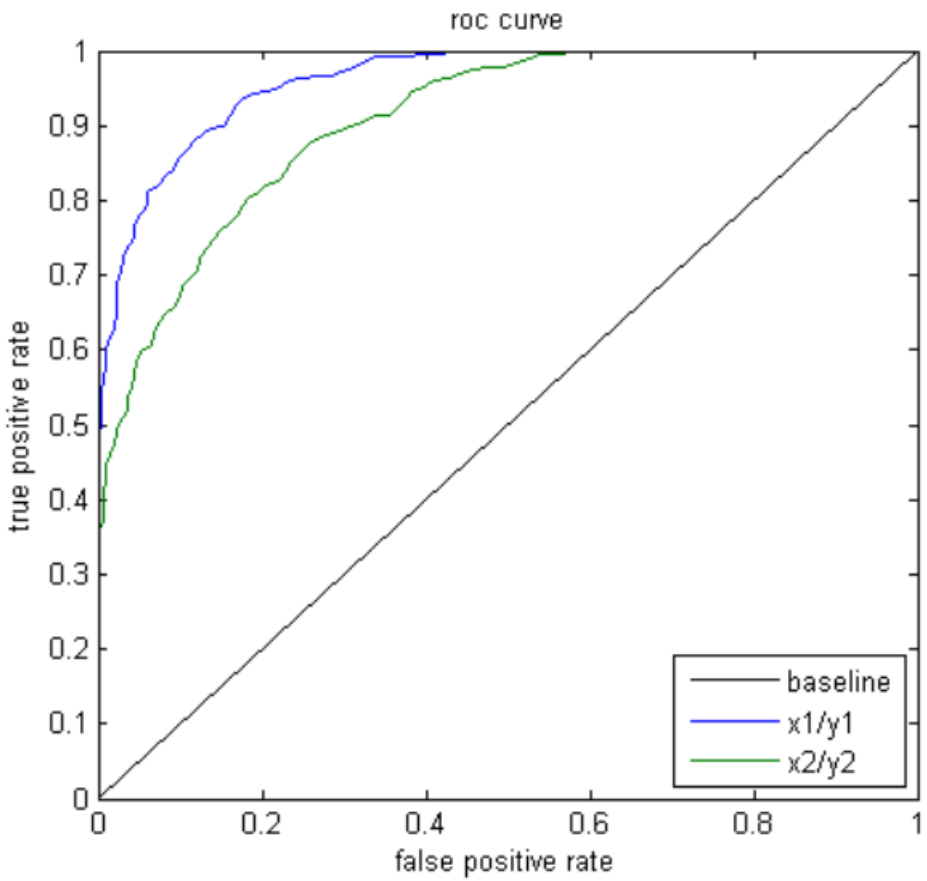


Figure: ROC curve

Table of Contents

Oblivious Decision Tree

Oblivious Decision Tree:

- There is the same attribute in each branch within one level
- Feature importance by design
- Even simpler interpretation

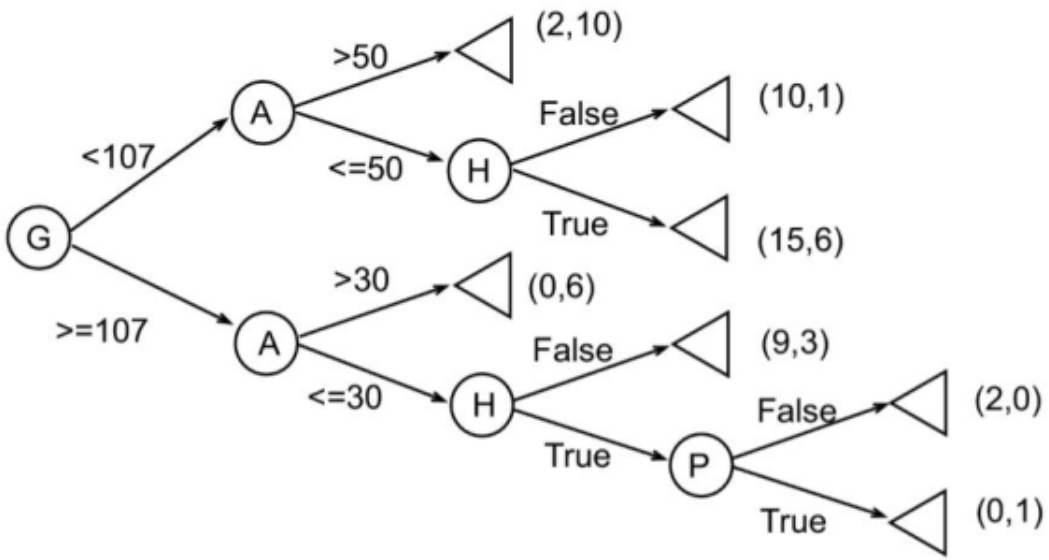


Figure: Oblivious DecisionTree

Fuzzy Decision Tree

Fuzzy Decision Tree:

- A way for handling fuzzy decision
- At each level we split data to a branch using some weight w according to our confidence
- Final result is measured by the sum of final confidences in the leaves
- Could be used for missing values

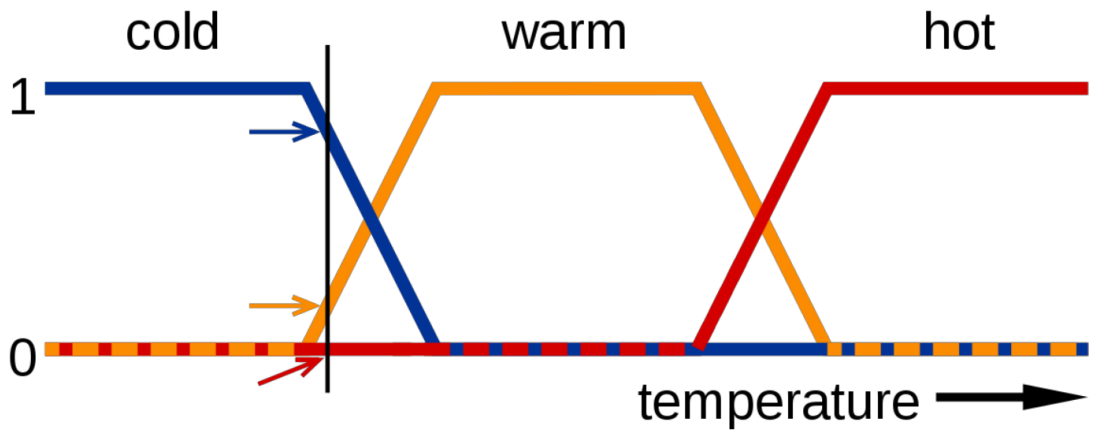


Figure: Fuzzy Decision Tree