Introduction to Machine Learning and Data Mining
Decision Trees

adapted and extended by Dmitry Ignatov ♢
author: Sergei Nikolenko

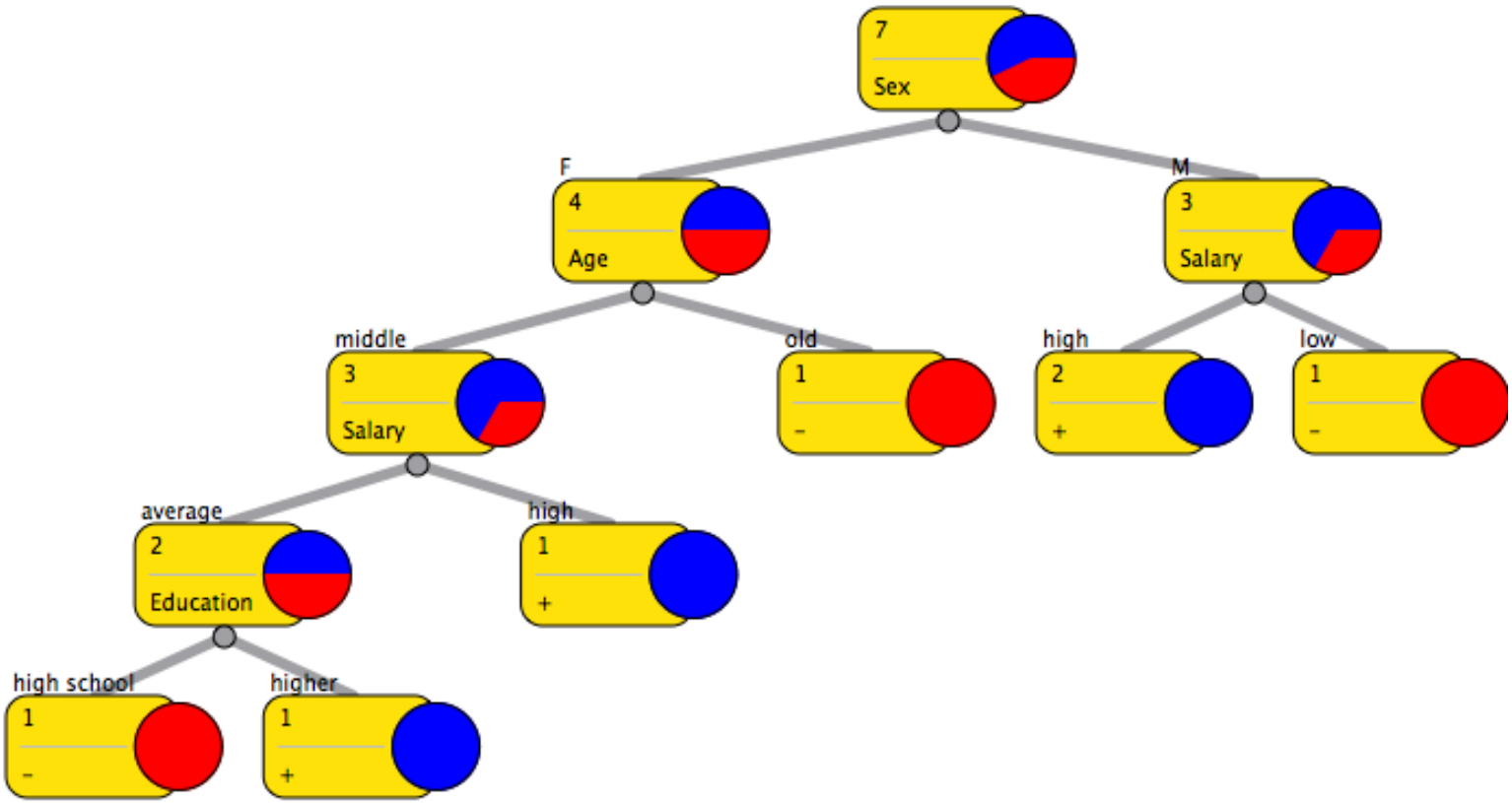
♢HSE
Computer Science Faculty
Dept. of Data Analysis and Artificial Intelligence

2019

Outline

Decision trees for credit scoring

Toy example



Example

Problem: Will «Zenit» win the next match?
Attributes:

- Is the competitor team higher in the tournament table;
- Is it a home match?
- Some of the team leaders is missing;
- Is it rainy?

We know only several outcomes and would like to predict the outcome of the next match with a new combination of the attributes' values.

Problem statement

Main task:

- Data classification
- Approximation of a given Boolean function

I.e. we have *partially* defined function f and would like to infer its values for unobserved examples.

Problem statement

Data:

- Attributes (function variables)
- Traning examples ($f(0,0,0,1)$, $f(0,0,1,1)$, $f(0,1,1,0)$, $f(0,1,1,1)$)

Required:

- To extrapolate function for other attributes' values (e.g. to find $f(0,0,0,0)$)
- To do that in optimal way

Decision Tree

Decision tree is a labeled tree:

- Nodes (not leafs): attributes
- Leafs: values of the target function
- Edges: a value of the attribute in the node incident to the outgoing edge

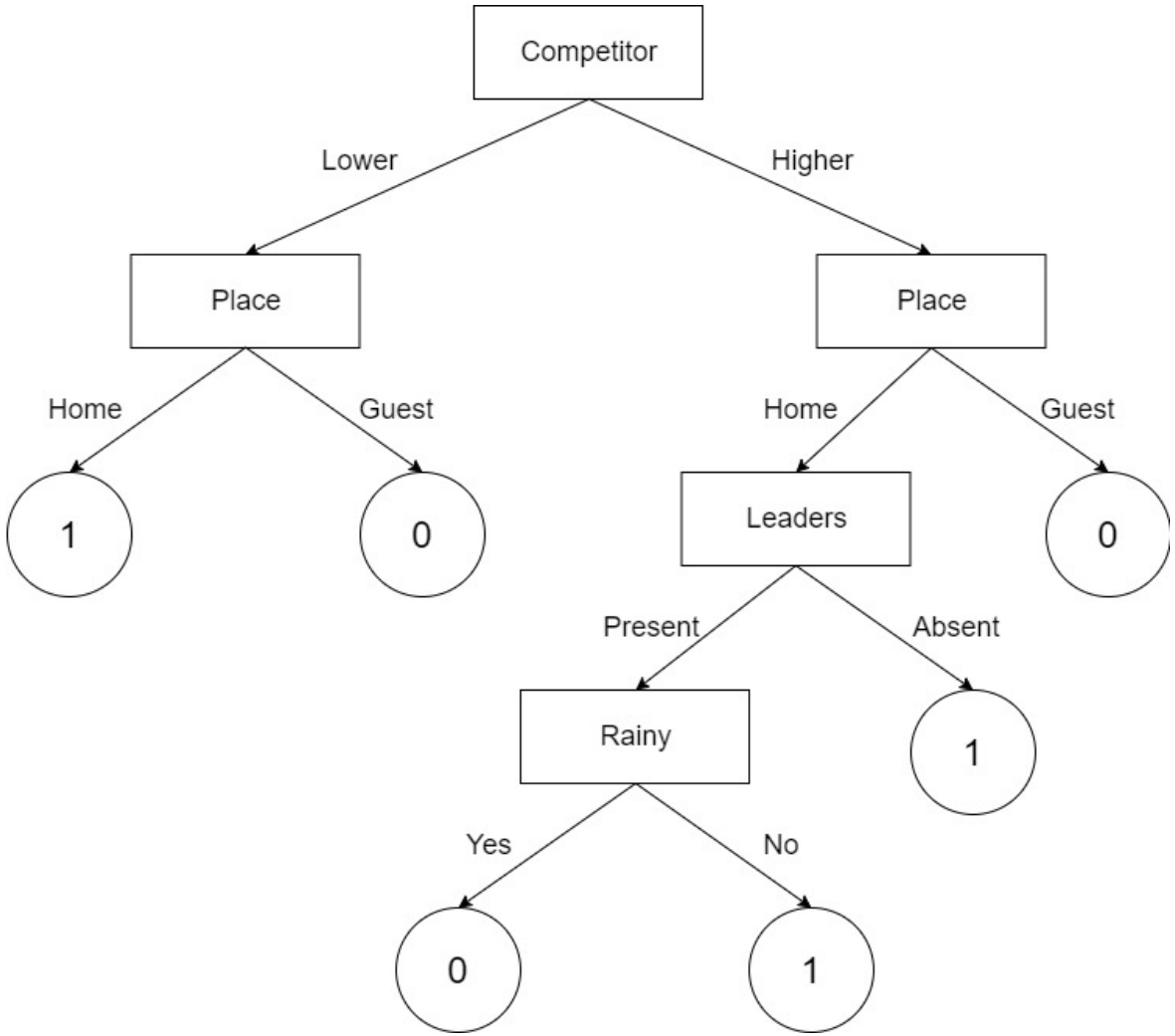
To classify new example, one needs to go down through the tree to the corresponding leaf and return its associated value.

Initial data

Table: How «Zenit» plays.

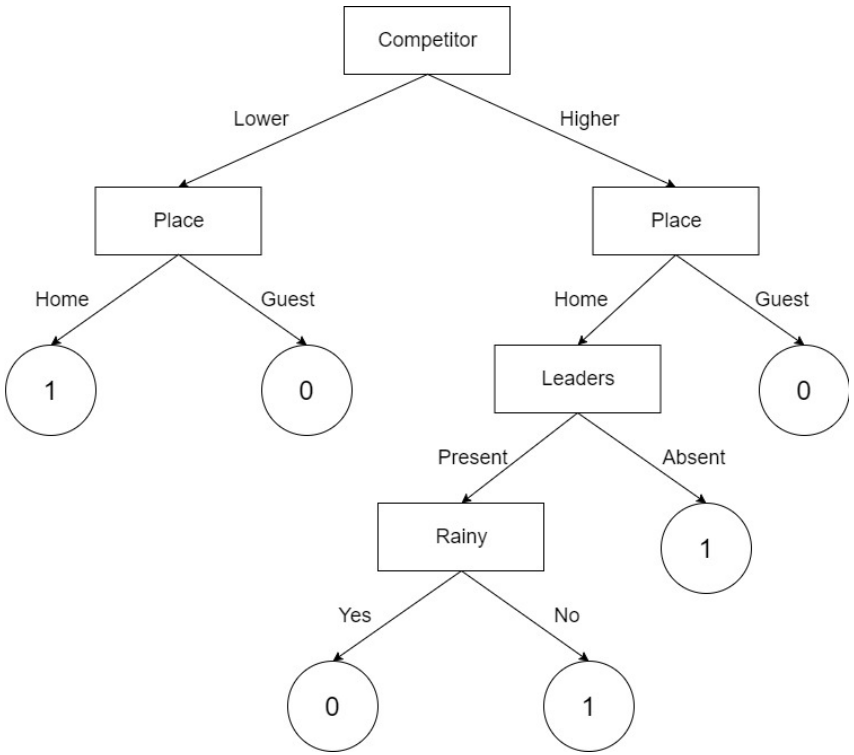
Competitor	Place	Leaders	Rainy	Victory
Higher	Home	Present	Yes	No
Higher	Home	Present	No	Yes
Higher	Home	Absent	No	Yes
Lower	Home	Absent	No	Yes
Lower	Guest	Absent	No	No
Lower	Home	Absent	Yes	Yes
Higher	Guest	Present	Yes	No
Lower	Guest	Present	No	*???*

The tree


Competitor
Lower
Higher
Place
Place
Home
Guest
Home
Guest
1
0
Leaders
0
Present
Absent
Rainy
1
Yes
No
0
1

How to use the tree

How to classify:

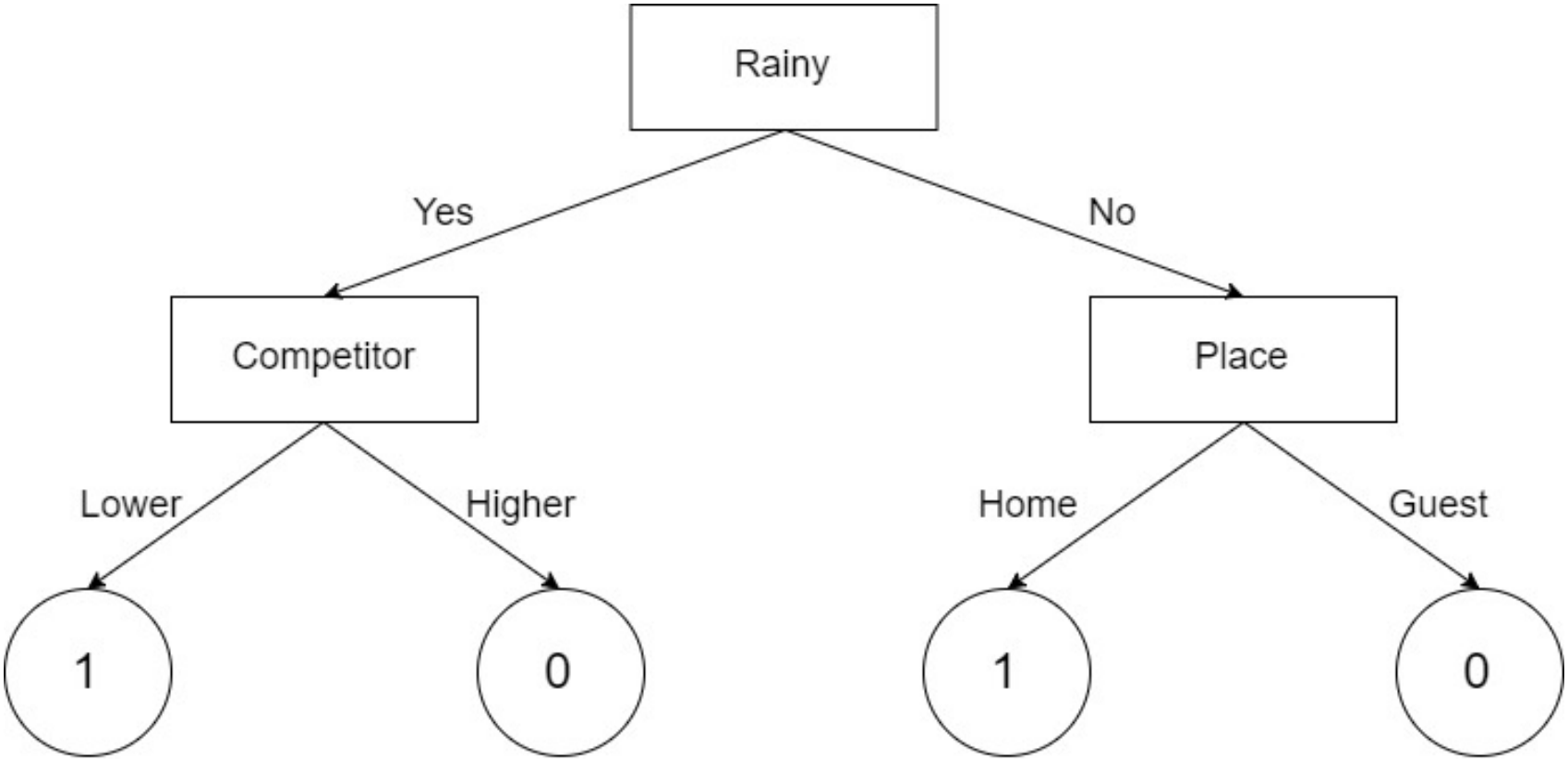


Competitor = Lower
Place = Guest
Leaders = Present
Rain = No
Victory = ???

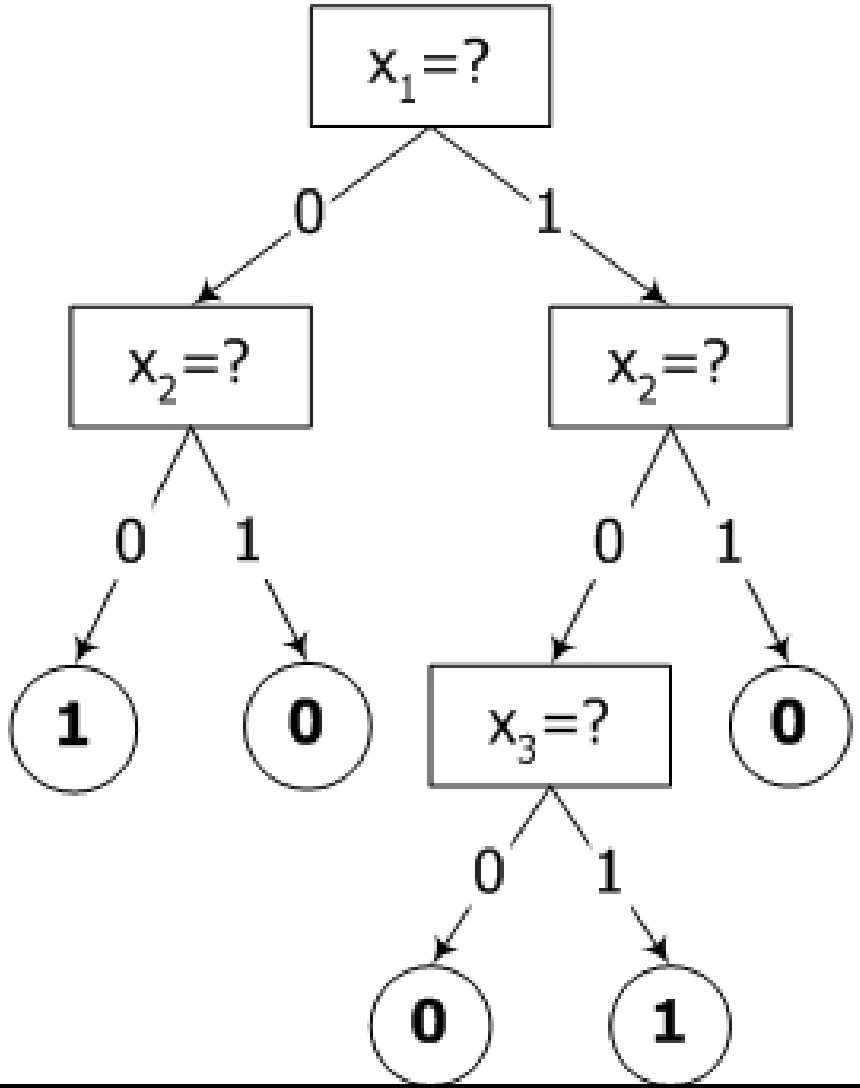
Go down to the tree, select matching attributes, and obtain the result: according to our tree «Zenit» should lose.

Optimal tree

That is a rather big tree, but there is smaller one for the same data:



Decision Trees and Boolean Functions



A decision tree induces a Boolean function as DNF.
For example, the tree in the figure results in:

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2x_3.$$

Exercises

Exercise. Draw decision trees for the following functions:

1. $x \vee (y \wedge \bar{z})$;
2. $(x \wedge \bar{y}) \vee (y \wedge \bar{z} \wedge t)$;
3. $(x \vee y) \wedge (\bar{y} \vee z)$.

Algorithm

How to build a decision tree:

- Select the next attribute A and put it in the root
- For all its values a_i:
  - Keep only those training examples where the value of A is a_i
  - Recursively build the tree for this new child node
- Output the resulting tree

Algorithm

How to build a decision tree:

- Select the next attribute A and put it in the root
- For all its values a_i:
  - Keep only those training examples where the value of A is a_i
  - Recursively build the tree for this new child node
- Output the resulting tree

Main problem:

- How to select a new attribute?

Entropy

Определение

Let S is n-element set where m of it elements has a property A. Then entropy of S w.r.t. A:

$$H(S, A) = -\frac{m}{n} \log_2 \frac{m}{n} - \frac{n-m}{n} \log_2 \frac{n-m}{n}.$$

Entropy depends on the subset split balance. More "equal" subsets imply higher entropy values.

Entropy

If property A is not binary (with a different values) and each of them takes place in m_i cases, then

$$H(S, A) = -\sum_{i=1}^{a} \frac{m_i}{n} \log \frac{m_i}{n}.$$

Entropy is an average number of bits to encode attribute A for an element of set S. If the probability of one of the values of A is $1/2$, then the entropy is 1 and we need one bit; if the values of A are not equiprobable, then we can encode a sequence of elements of S more efficiently.

Entropy: example

In our example, 7 matches of «Zenit», they lost three and won four of them. So, the entropy is

$$H(S, \text{Victory}) = -\frac{4}{7}\log_2\frac{4}{7} - \frac{3}{7}\log_2\frac{3}{7} \approx 0.9852.$$

Information Gain

An attribute for classification should be chosen such that after classification the entropy (w.r.t. the target function) is small as possible.

Определение

Suppose that a set S of elements with the target attribute Q is described by an attribute A with a possible values. Then information gain is defined as

$$\mathrm{Gain}(S, A) = H(S, Q) - \sum_{i=1}^{a} \frac{|S_i|}{|S|} H(S_i, Q),$$

where S_i is a set of elements of S such that A takes the value a_i.

Information gain: example

Now, let us calculate information gain for various attributes:

$$\begin{aligned}\mathrm{Gain}(S, \text{Competitor}) = H(S, \text{Victory}) - \frac{4}{7}H(S_{\text{Higher}}, \text{Victory}) - \\ - \frac{3}{7}H(S_{\text{Lower}}, \text{Victory}) \approx \\ \approx 0.9852 - \frac{4}{7}\left(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2}\right) - \\ - \frac{3}{7}\left(-\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3}\right) \approx 0.0202.\end{aligned}$$

The attribute is not good at all...

Information gain: example

$$\mathrm{Gain}(S, \text{Place}) \approx 0.4696.$$
$$\mathrm{Gain}(S, \text{Leader}) \approx 0.1281.$$
$$\mathrm{Gain}(S, \text{Rain}) \approx 0.1281.$$

IG suggests first classification split by place where match is played.

Exercise. The tree (check it) will be of depth 3. How to modify the attribute choice to obtain a tree of depth 2 with less number of nodes?

Outline

ID3 Algorithm

$ID3(S, \mathcal{A}, Q)$

- Build the tree root.
- If Q is true for all objects from S, then label the root by 1 and quit.
- If Q is false for all objects from S, then label the root by 0 and quit.
- If $\mathcal{A} = \emptyset$, then:
  - if Q is valid for half of greater part of S, then put 1 in the root and quit;
  - if Q is not valid for the majority of examples from S, then put 0 in the root and quit.

- Choose $A \in \mathcal{A}$ for which $\mathrm{Gain}(S, A)$ is maximal.
- Put the label A in the root.
- For every value a of A:
  - add new child to the root and label the outgoing edge by a;
  - if there are no cases in S, where A takes value a (i.e. $|S_a| = 0$), then this child according to major part of S for values of Q;
  - otherwise execute $ID3(S_a, \mathcal{A} \setminus \{A\}, Q)$ and add its result as a subtree with the root in this child.

The drawback of IG criterion

Problem: IG selects attributes with the greatest number of values. For example, let the dates of mathces are available. Information Gain:

$$\mathrm{Gain}(S, \mathrm{Date}) = H(S, \mathrm{Victory}) - \\ - \sum_{i=1}^{n} \frac{1}{n} H(S_{\mathrm{Date=i}}, \mathrm{Victory}) = H(S, \mathrm{Victory}).$$

Since there is only one case in each branch, their entropy is equal to 0.
The information gain is maximal, but the obtained tree is impossible to use.

Gain Ratio

Gain Ratio takes into account not only amount of information to record the result, but it counts amount of information to split the dataset by the current attribute.
Adjustment:

$$\mathrm{SplitInfo}(S, A) = -\sum_{i=1}^{a} \frac{|S_a|}{|S|} \log_2 \frac{|S_a|}{|S|},$$

Criterion maximization

$$\mathrm{GainRatio}(S, A) = \frac{\mathrm{Gain}(S, A)}{\mathrm{SplitInfo}(S, A)}.$$

Gain Ratio: example

For the attribute Data we have

$$\mathrm{SplitInfo}(S, \mathrm{Date}) = -\sum_{i=1}^{7} \frac{1}{7} \log_2 \frac{1}{7} \approx 2.80735\ldots,$$

and

$$\mathrm{GainRatio}(S, \mathrm{Date}) = \frac{\mathrm{Gain}(S, \mathrm{Data})}{\mathrm{SplitInfo}(S, \mathrm{Date})} \approx 0.350935\ldots$$

But for the attribute Place, we have

$$\mathrm{SplitInfo}(S, \mathrm{Place}) = -\frac{5}{7} \log_2 \frac{5}{7} - \frac{2}{7} \log_2 \frac{2}{7} \approx 0.86312\ldots,$$

and

$$\mathrm{GainRatio}(S, \mathrm{Place}) = \frac{\mathrm{Gain}(S, \mathrm{Place})}{\mathrm{SplitInfo}(S, \mathrm{Place})} \approx 0.5452\ldots$$

Gini impurity

For a set of examples S and an attribute A with a values, we have

$$\mathrm{Gini}(S, A) = 1 - \sum_{i=1}^{a} \left(\frac{|S_i|}{|S|} \right)^2 .$$

For a target attribute Q with q values its gain is as follows:

$$\mathrm{GiniPurityGain}(S, A, Q) = \mathrm{Gini}(S, Q) - \sum_{j=1}^{a} \frac{|S_j|}{|S|} \mathrm{Gini}(S_j, Q).$$

Gini Index

- There is a misconception that Gini impurity is the same as Gini coefficient in Economics, which is in fact related to AUC.
- In 1912 Corrado Gini has proposed the coefficient as measure of economical inequality of people.
- If one builds an income distribution curve for some population, then Gini coefficient is greater when greater part of income is concentrated in hands of a smaller fraction of the population.
- In its turn Gini impurity describes the expected error of random labeling according to the target class distribution.

Overfitting

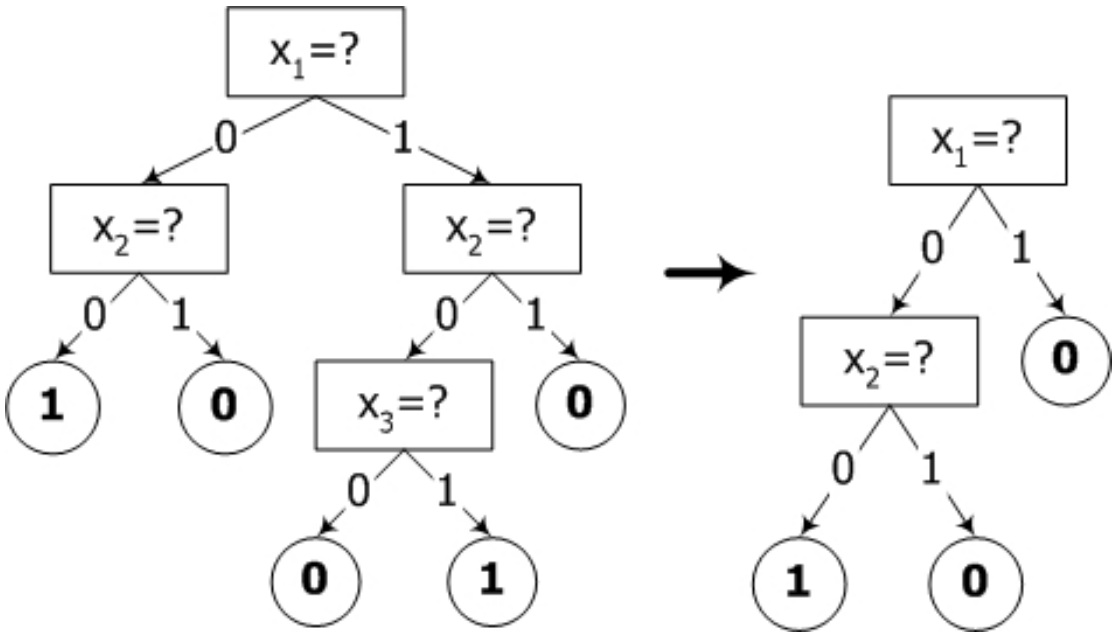
- ID3 is consistent with *all* input examples
- However, a part of data might be «noisy» or or contain errors
- That's why the tree is getting bigger and works worse

Overfitting: example

- Let us suppose «Zenit» wins at home in 90% cases and this event is independent of any other attribute that we use.
- Let in our historical data there is one home defeat.
- ID3 takes into account all the «causes» and will predict in future that «Zenit» should win in analogous cases
- However, in fact, «Zenit» wins with probability 90%

Pruning

We need to cut off redundant branches. Usually, a subtree is replaced but a node with the most frequent value in the subtree.



Which branches to prune?

Pruning: general algorithm

- Let us build the tree on the part of our dataset.
- Then test it on the remaining part.
- For each node:
  - prune the branch in that node
  - if the pruned tree has better accuracy on the test examples, then cut off that branch, otherwise continue to test other nodes.

Outline

Decision trees for numeric attributes

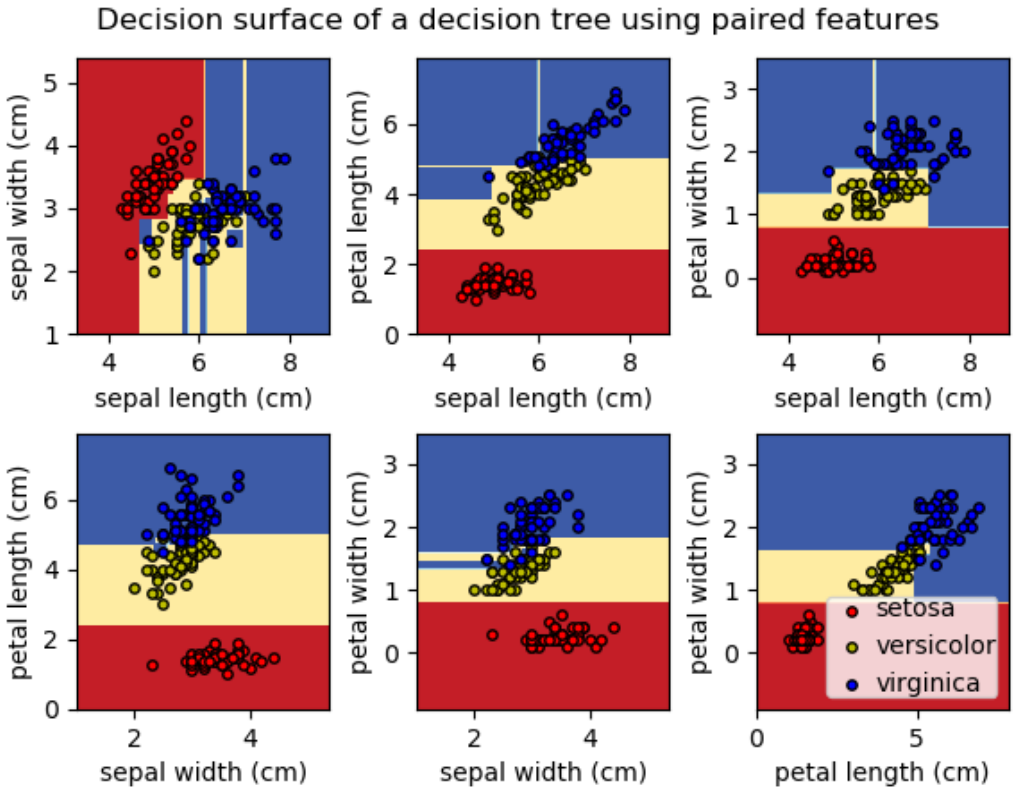


Figure: Source: https://scikit-learn.org/

Decision trees for numeric attributes
C4.5 Algorithm

R.Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993

- Extension of ID3 for numeric attributes (with missing values).
- Paper by R. Quinlan'a
- Is available in Weka as J48.
- Implementation of C5.0 by its author
- Decision trees in scikit-learn
- Chapter 19, Decision Tree Classifier, p. 481 in Zaki & Meira
- Paper from Deductor's developers (in Russian)

Regression Trees

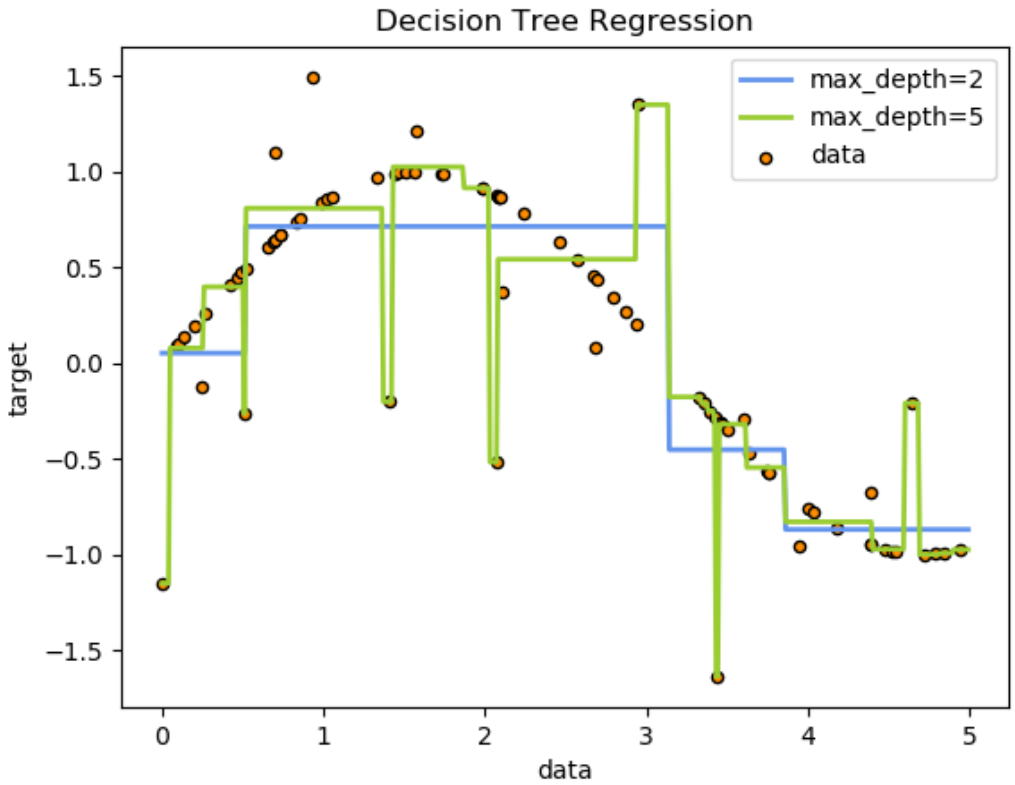


Figure: Iris example. Source: https://scikit-learn.org/

Regression Trees
CART – Classification and Regression Trees

Leo Breiman, J. H. Friedman, R. A. Olshen, C. J. Stone: Classification and Regression Trees.

Wadsworth 1984

- A general name for classification and regression based on decision tree like approach
- Decision trees are implemented in Orange, scikit-learn and many other packages
- Paper of Deductor's developers (in Russian)

Ensembles of trees

- Random Forests
  Leo Breiman: Random Forests. Machine Learning 45(1): 5-32 (2001)
- Gradient Boosting
  Jerome H. Friedman Greedy function approximation: A gradient boosting machine. Ann. Statist. 29(5): 1189-1232. (2001)
- "Gradient boosting machines, a tutorial" by Alexey Natekin
- XGBoost implementation (tutorial)

Oblivious decision trees and Matrixnet

- Oblivious Decision Trees: at each level they test the same attribute (with the same threshold for numeric attributes)
- Langley, P., & Sage, S. (1994). Oblivious decision trees and abstract cases. In Working notes of the AAAI-94 workshop on case-based reasoning (pp. 113-117)
- Matrixnet by Yandex:
  - Video 1
  - Video 2
  - Slides in Russian
- CatBoost

Just for fun

http://dilbert.com

Questions and contacts

www.hse.ru/staff/dima

Thank you!

dmitrii.ignatov[at]gmail.com

dignatov[at]hse.ru