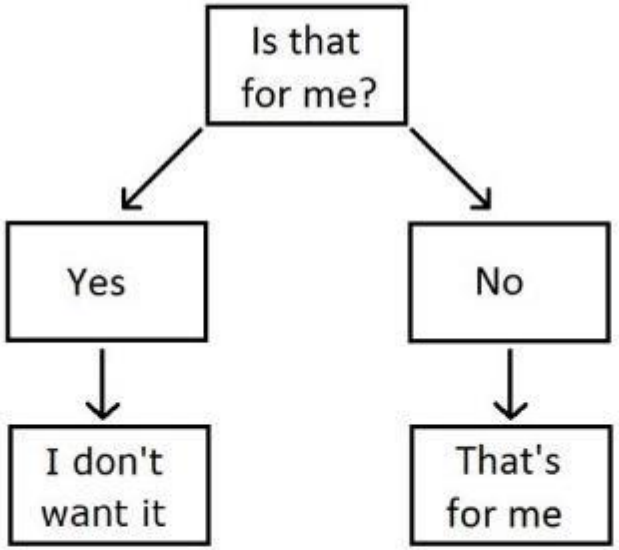


Rule induction with decision trees

Dr. Gonzalo Nápoles

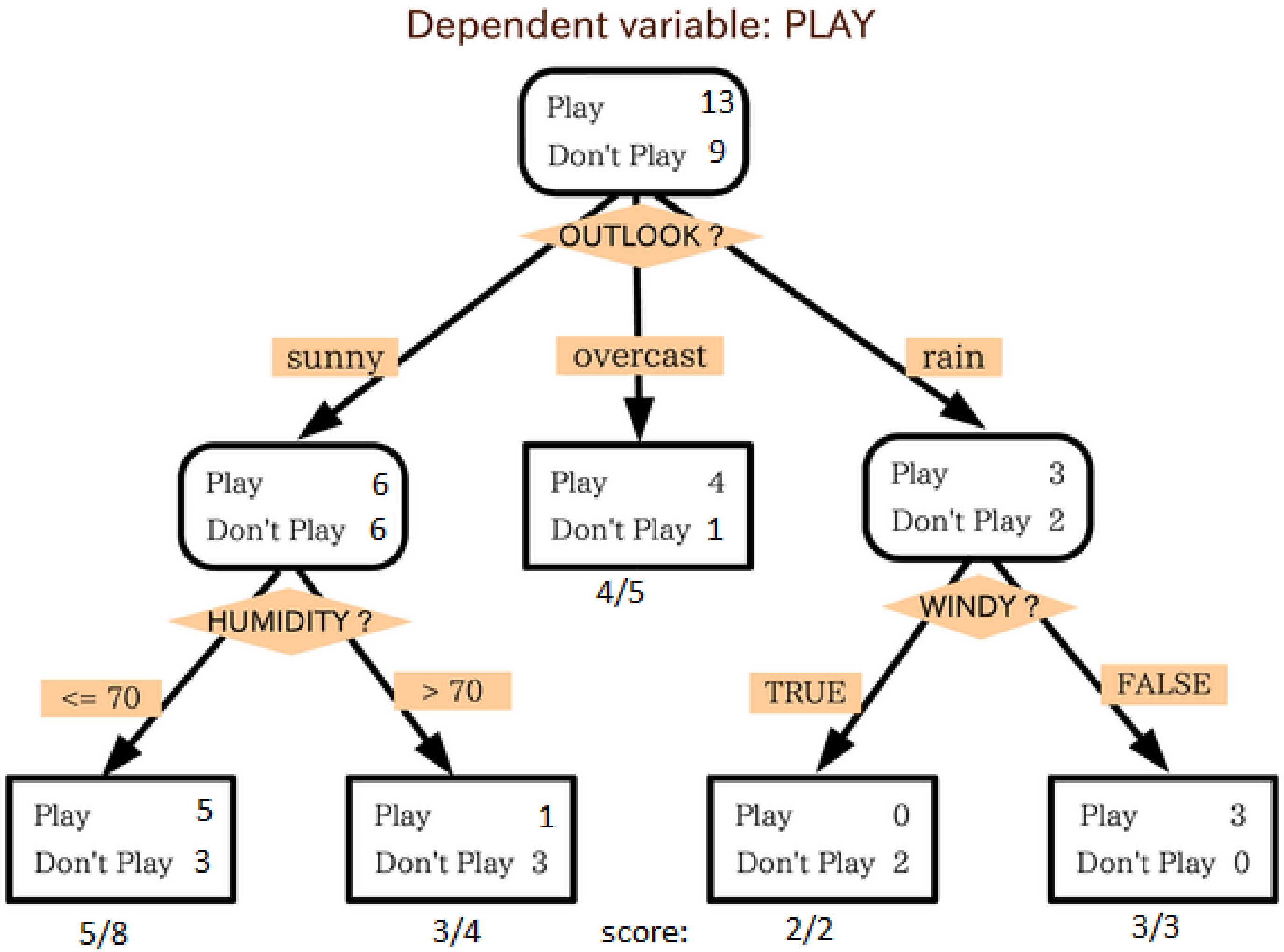
Supervised learning

- **The classification problem**
  - Given a set of labeled examples, build a model to determine the most appropriate decision class for a new instance.
  - The problem is supervised because the decision classes attached to training instances are known.

- **Potential applications**
  - Credit approval, direct marketing, fraud detection, medical diagnosis. In general, **classification models** can be used in any problem where inferring symbolic decisions is expected.

Supervised learning does not imply lack of automation.

The wheatear example



Decision trees

- An internal node denotes a test for a specific attribute, while a branch represents an outcome of the test.

  Example: temperature < 77.5

- A leaf node denotes a class label or class label distribution, which can be observed several times.

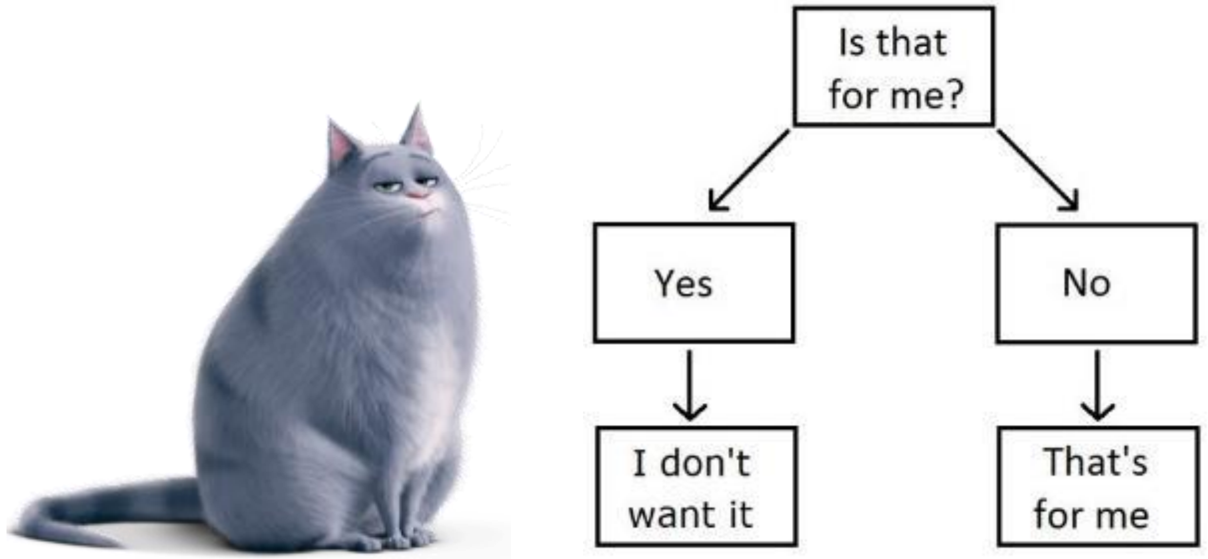
- At each node, one attribute is chosen to split the training set into distinct classes as much as possible.

A new case is classified by following a matching path to a leaf node.

Building decision trees

- Top-down tree construction
  - At the beginning, all training examples are at the root.
  - Partition the examples by choosing one attribute each time.
- Bottom-up tree pruning
  - Remove subtrees or branches, in a bottom-up manner, to improve the estimated accuracy on new cases.

• Construction step
• Optimization step



Which is the best attribute?

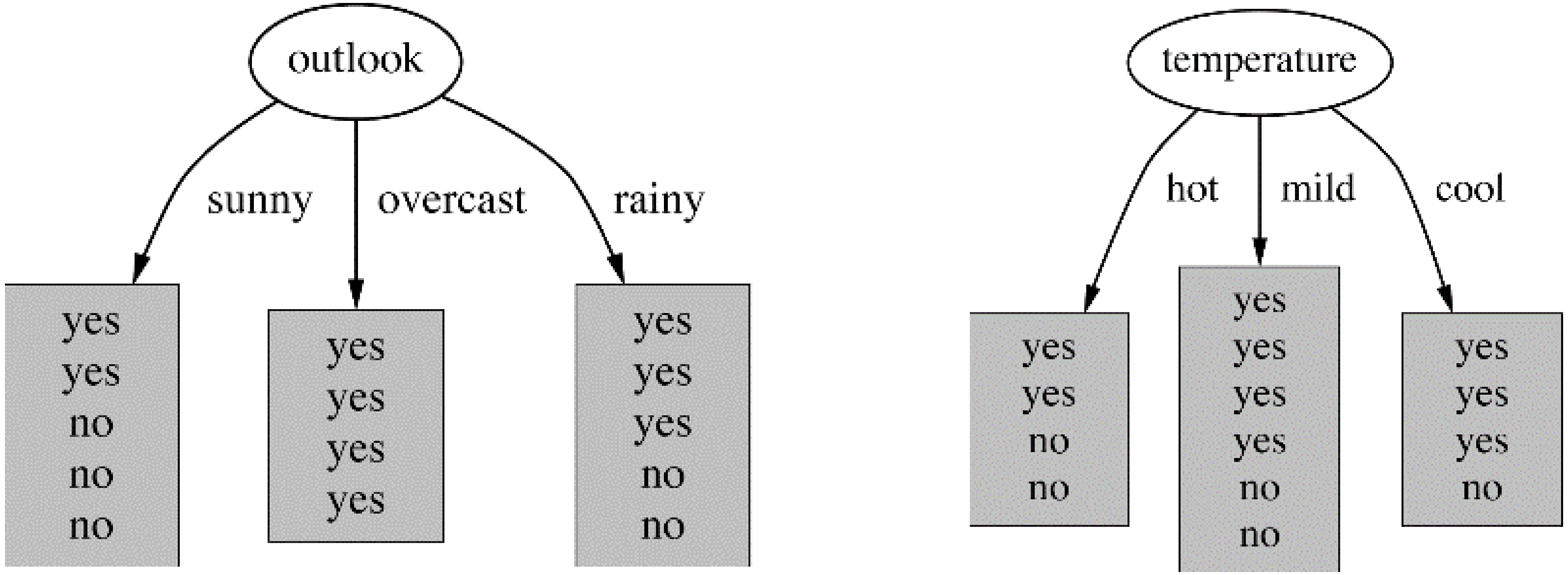
- At each node, available attributes are evaluated to separate the classes of the training examples.

- A quality function determines the goodness of each attribute being evaluated. Typical functions are:
  - information gain
  - information gain ratio
  - Gini index

The best attribute is the one which leads to the smallest tree.

Strategy: choose the attribute with highest information gain.

Which is the best attribute?



We can use the entropy to measure the amount of information attached to each attribute.

Which is the best attribute?

$$E(P) = -\sum_{i} p_i \log p_i$$

Given a probability distribution, the info required to predict an event is the distribution's entropy.

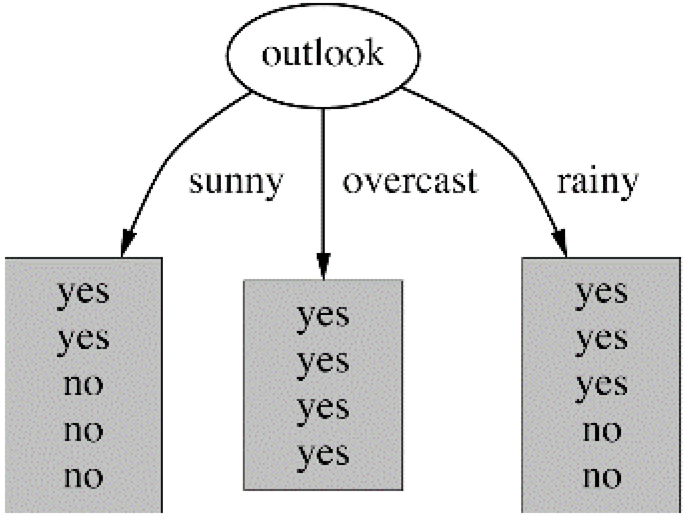
Why the Entropy measure?

- When the node is totally pure, the Entropy is zero
- When impurity is maximal, the Entropy is maximal
- Besides, the Entropy fulfils the *multistage property*

Which is the best attribute?

- $\text{info}(outlook \leftarrow \text{sunny}) = E(2/5\,,3/5) = 0.971$
- $\text{info}(outlook \leftarrow \text{overcast}) = E(4/4\,,0/4) = 0.0$
- $\text{info}(outlook \leftarrow \text{rainy}) = E(3/5\,,2/5) = 0.971$

$$\text{info}(outlook) = (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971 = 0.693$$



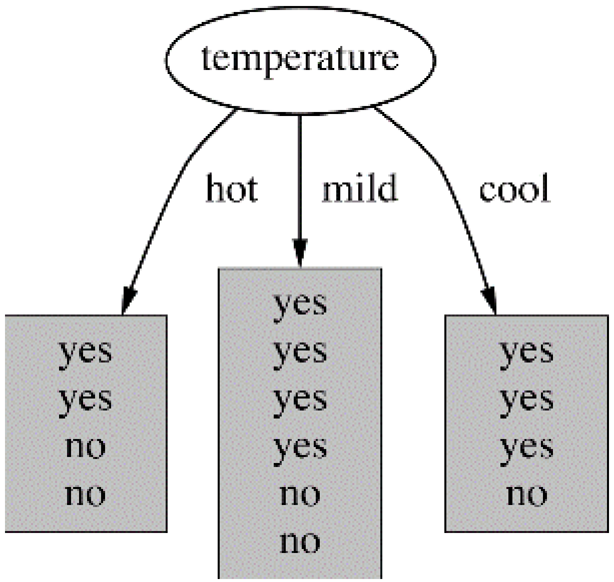
Which is the best attribute?

- $\text{info}(temperature \leftarrow \text{hot}) = E(2/4\,,2/4) = 1.0$
- $\text{info}(temperature \leftarrow \text{mild}) = E(4/6\,,2/6) = 0.92$
- $\text{info}(temperature \leftarrow \text{cool}) = E(3/4\,,1/4) = 0.81$

$$\text{info}(temperature) = (4/14) \times 1 +$$

$$(6/14) \times 0.92 + (4/14) \times 0.81$$

$$= 0.9114$$



Which is the best attribute?

- Once the entropy has been calculated for each attribute, we can compute the information gain.

$$\text{gain}(A) = \text{info}(root) - \text{info}(A)$$

Therefore,

$$\text{gain}(outlook) = \text{info}([9,5]) - \text{info}(outlook)$$

$$= 0.94 - 0.693 = 0.247 \quad ✓$$

$$\text{gain}(temperature) = \text{info}([9,5]) - \text{info}(temperature)$$

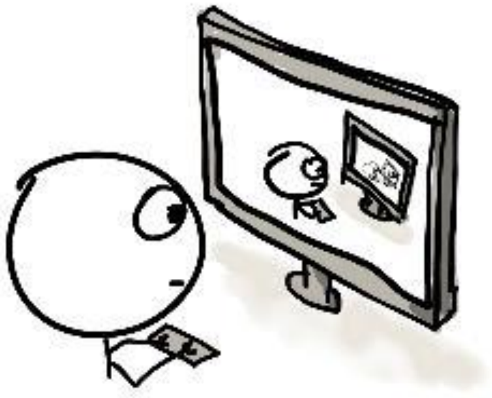
$$= 0.94 - 0.9114 = 0.029$$

Continue splitting recursively

- The tree construction procedure is performed in a recurrent fashion until a stopping criterion is satisfied.

- Not all leaves need to be pure (i.e. with the same decision). Sometimes identical instances lead to different classes; this situation is call inconsistency.

- The recursive construction process stops when the training set cannot be split any further.



Pseudocode

```
Function DT(Examples, Attributes, Target)
  Create a root node for the tree
  IF all examples belong to the same decision class
     return the root node as a leaf with that decision class
  END
  IF the attribute set is empty
     return the root node as a leaf with the most likely class
  END
```

Stopping criteria for the recursive construction procedure.

Pseudocode

```
Function DT(Examples, Attributes, Target)
  A ← best attribute in the attribute set
  FOREACH value V_i of A
     add a branch below the current node
     IF examples(V_i) THEN
       add a leaf node with the most likely class
     END
     DT(examples(V_i), A, Attributes – {A})
  END
```

Recursive construction procedure.

Further remarks

- The algorithm's performance is affected by attributes with a large number of values (extreme case: ID code).

- The partition induced by an attribute with a large number of values is more likely to be pure.

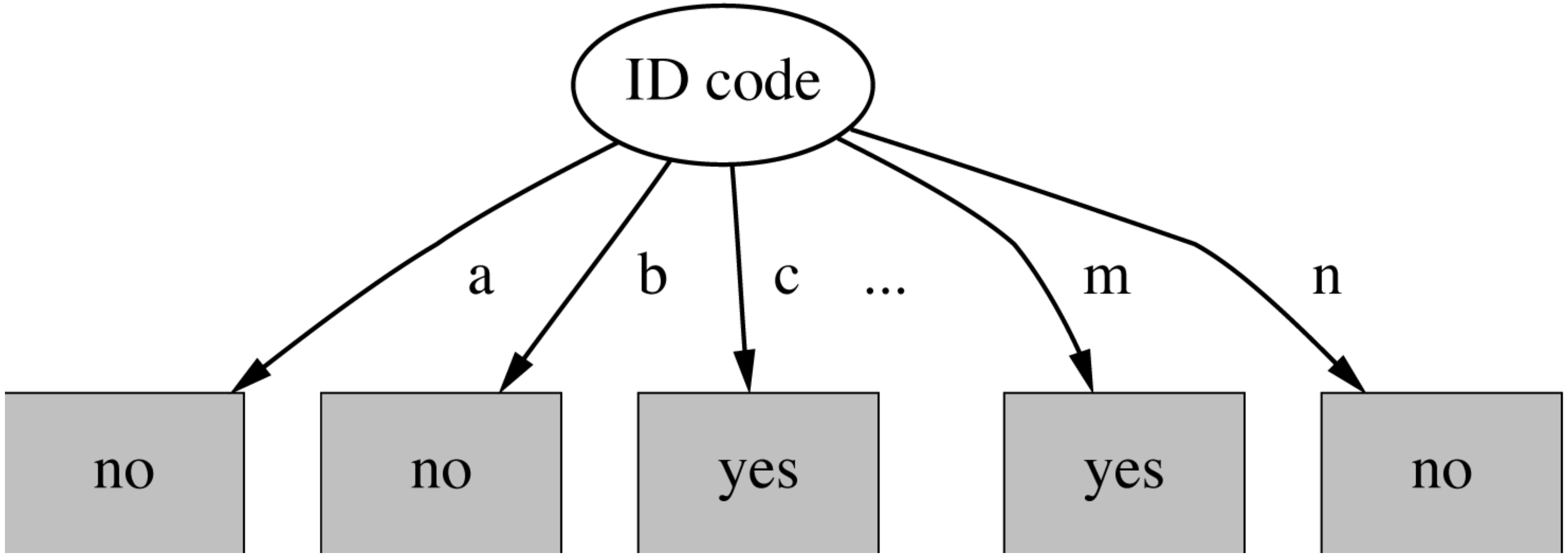
  - Therefore, the information gain measure is biased towards choosing attributes with a large number of values.

  - This behavior may result in **overfitting** (selection of an attribute that is non-optimal for solving the problem).

TILBURG UNIVERSITY

Further remarks

- The information gain for ID code is maximal since each leaf node contains a single case, this it's pure.



Alternatives

- The gain ratio is based on the information gain that reduces its bias on high-branch attributes.
- This measure takes the **number** and **size** of branches into account when choosing an attribute.
- This measure corrects the information gain by taking the intrinsic information of a split into account.

Intrinsic information

$$\text{split}(X, A) = -\sum_{v} \frac{P_v}{|X|} \log \frac{P_v}{|X|}$$

P_v is the number of instances in X such that $X_A = v$

What is different?

This measure considers the entropy of instances with regards to the target attribute.

Intrinsic information

$$\text{ratio}(X, A) = \frac{\text{gain}(A)}{\text{split}(X, A)}$$

$split(X, A)$ is used to normalize the information gain measure.

What is different?

The importance of an attribute decreases as intrinsic information gets larger!

Drawbacks of gain ratio

- It may overcompensate since it might choose an attribute just because its intrinsic information is very low.

- As alternative, we can do the following:

  - **Step#1**. Only consider those attributes with information gain greater than the average gain value.
  - **Step#2**. Compare preselected attributes according to the gain ration and select th one having maximal ratio.

Other thresholding heuristic may be adopted.

Another alternative: Gini index

$$\text{gini}(X, A) = 1 - \sum_j (p_j)^2$$

p_j is the relative frequency of class j at the current node.

Some features

The index will be maximum when classes are equally distributed, less interesting.

Another alternative: Gini index

$$\text{gini}_{split}(X, A) = \sum_{i=1}^{K} \frac{N_i}{N} \text{gini}(X, A)$$

When the node is split into K partitions (children).

The index is minimized, assuming that N_i and N are the number of instances on the child node and the current node, respectively.

Decision tree optimization – pruning

- The decision tree algorithm continues to grow a tree until it makes no errors over the set of training data.

- This fact makes ID3 prone to overfitting. In order to reduce overfitting, pruning is used:

  - **Postpruning**. take a fully-grown decision tree and discard unreliable parts, once the construction process is finished.
  - **Prepruning**. stop growing the tree when the information becomes unreliable. This strategy can stop too early!!

Decision tree optimization – pruning

- Pre-pruning is based on statistical significance test
  - Stop growing when there is no statistically significant association between any attribute and the class at a particular node.

- For example, the ID3 algorithm uses the chi-squared test in conjunction to the information gain measure:

  - As a result, only statistically “significant” attributes are allowed to be selected by the information gain procedure.

Decision tree optimization – pruning

- Post-pruning optimizes a full tree
  - **Problem**. some subtrees might be due to chance effects

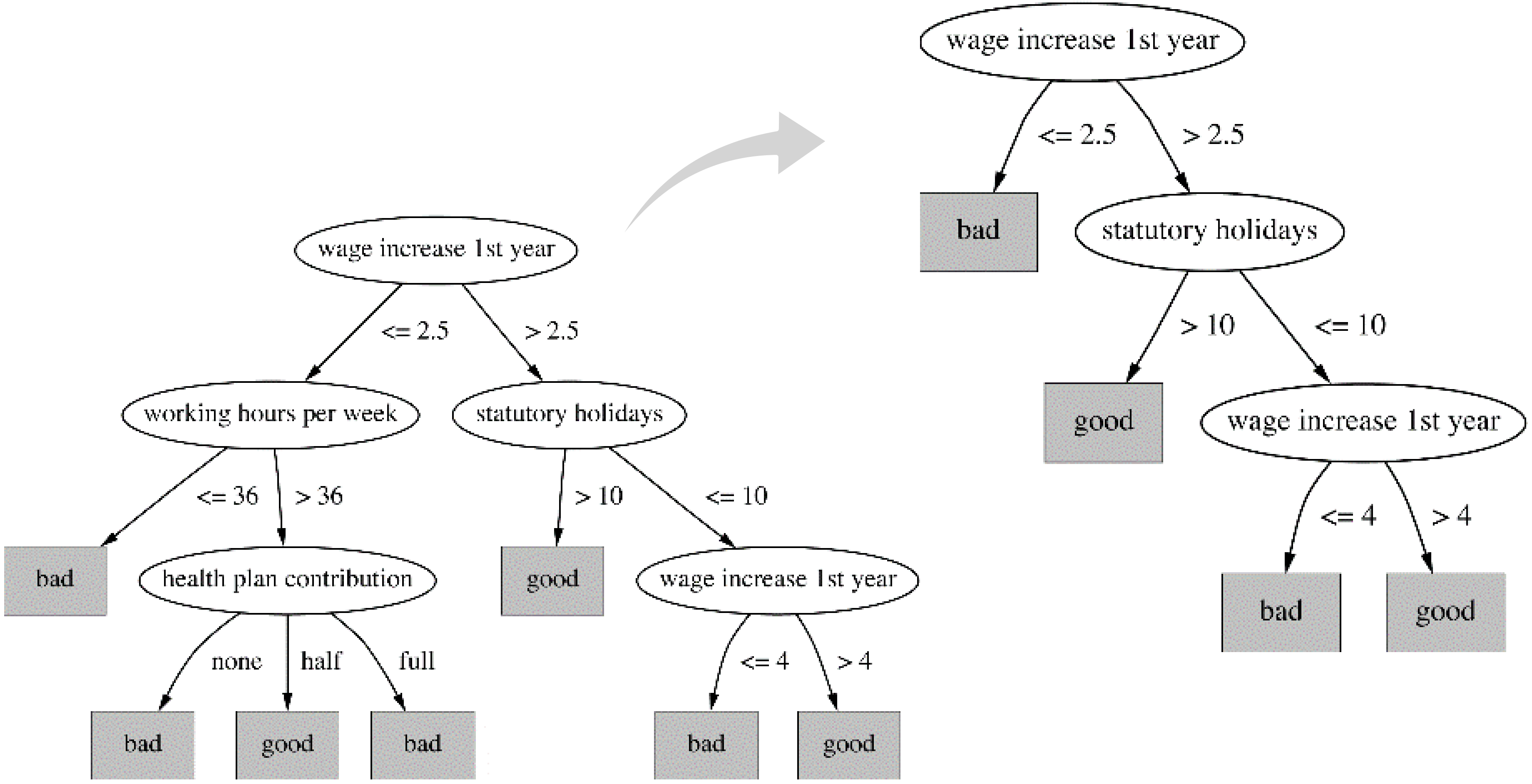
- Post-pruning is based on two main operations:

  - **Subtree replacement**. replaces the subtree with a single leaf.
  - **Subtree raising**. replaces the subtree with the child one.

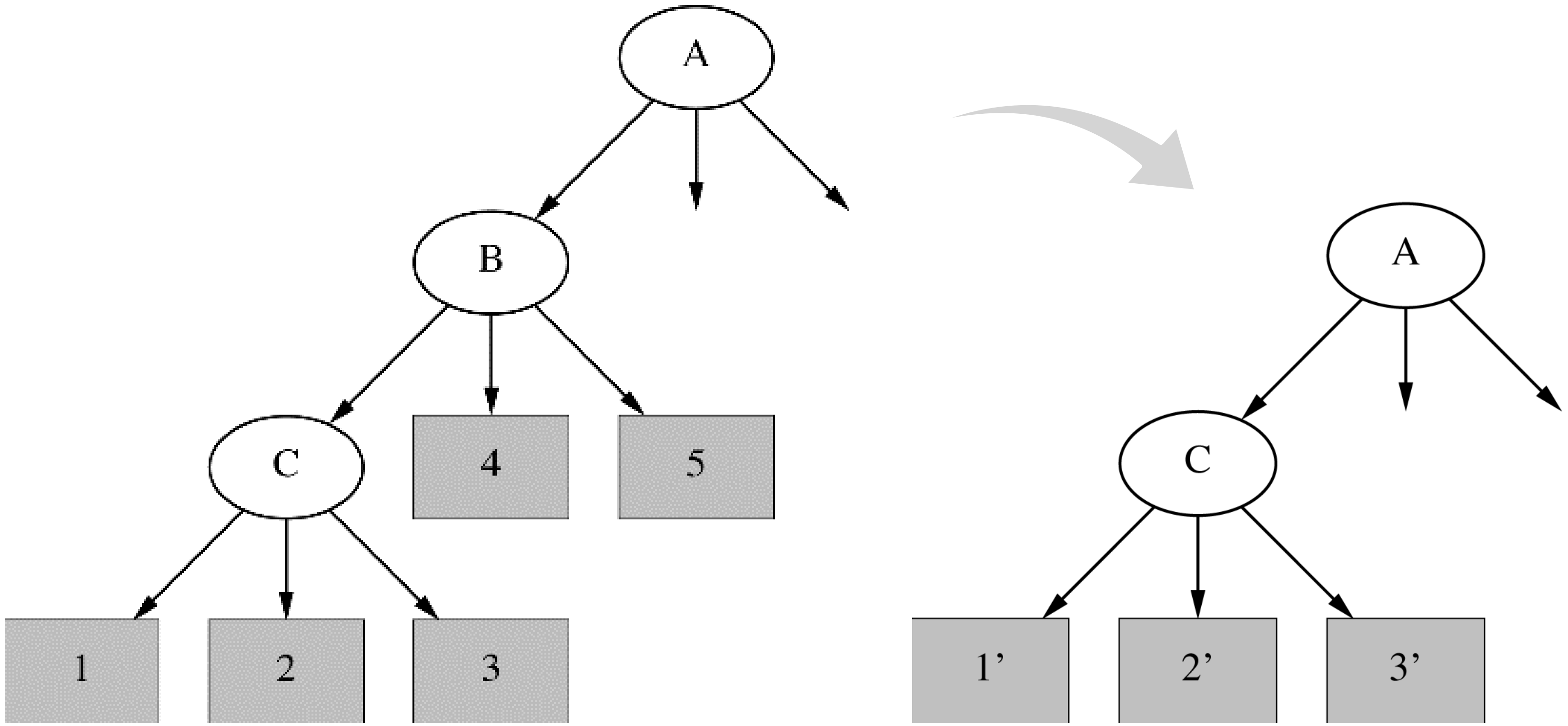
Pre-pruning faster than post-pruning.

Tilburg University

Decision tree optimization – pruning



Decision tree optimization – pruning



Delete node and redistribute the remaining instances

Decision tree optimization – pruning

- One approach to computing the error rates is to reserve a portion of the available dataset for validation. The validation set is not used during training.

- If the new error rate is grater than the error rate of a pruned version of the tree, pruning is performed.

- Reduced error pruning can reduce overfitting, but it reduces the amount of data available for training.

Statistical pruning

- The C4.5 algorithm uses statistical confidence estimates for pruning the tree, which uses the whole dataset.

$$\mu(E) = E + z\sqrt{\frac{E(E-1)}{N}}$$

Upper limit of the error confidence interval

where E is the error attached to the leaf, z is the z-score and N is the number of tested instances.