PADERBORN UNIVERSITY

Decision Trees & Rule Learning

Explainable Artificial Intelligence

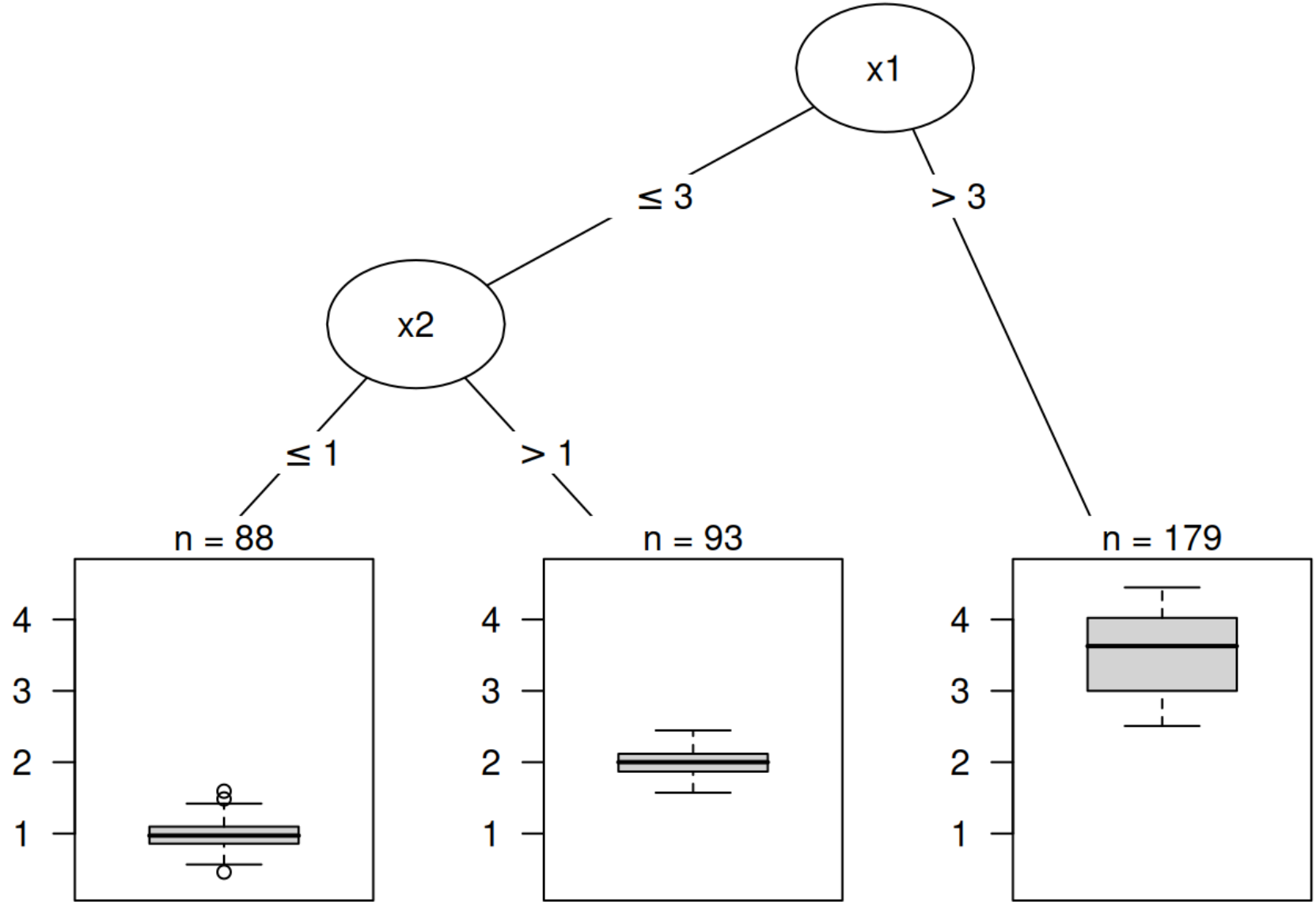
Dr. Stefan Heindorf

Outlook

- Decision trees
- Decision rules

Decision Trees

Decision tree



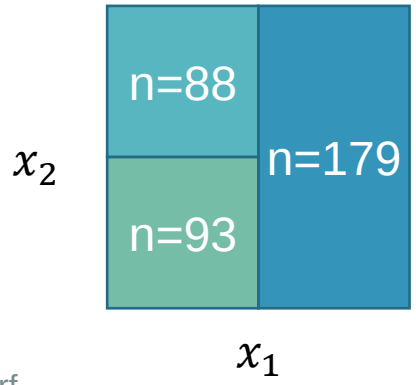
Decision tree

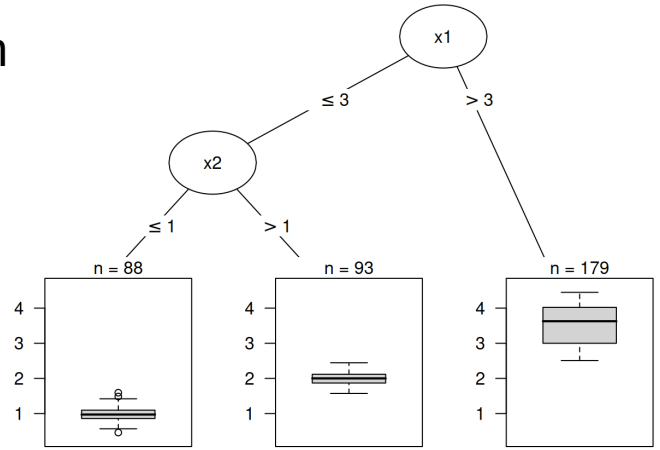
Linear regression and logistic regression models fail in situations

- where the relationship between features and outcome is nonlinear
- where features interact with each other

Decision tree

- Split the data multiple times according to certain cutoff values in the features
- Through splitting, different subsets of the dataset are created (with each instance belonging to one subset)
- Final subsets are called terminal or leaf nodes
- Intermediate subsets are called internal nodes or split nodes
- To predict the outcome in each leaf node, the average outcome of the training data in this node is used
- Trees can be used for classification and regression





Decision tree

Prediction

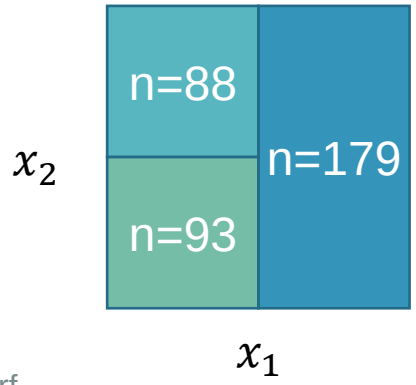
Relationship between the outcome y and features $\mathbf{x}$:

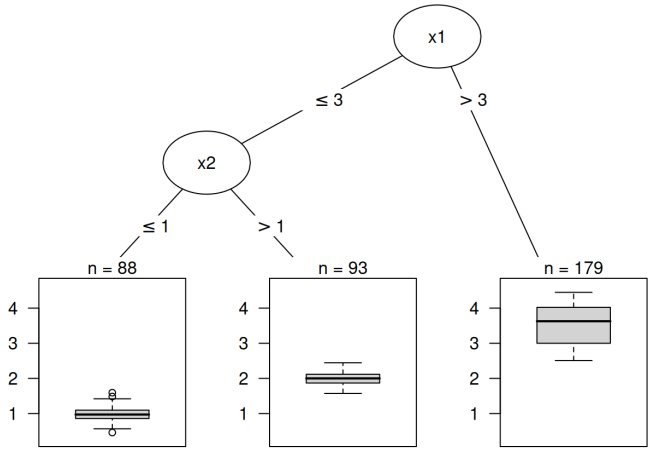
$$\hat{y} = \hat{f}(\mathbf{x}) = \sum_{m=1}^{M} c_m I\{\mathbf{x} \in R_m\}$$

R_m data subsets obtained by splitting
(each instance falls into exactly one leaf node (=subset R_m)

$I\{\mathbf{x} \in R_m\}$ identity function that returns 1 if $\mathbf{x}$ is in R_m and 0 otherwise

c_m predicted outcome for subset R_m:
(average of training instances in this subset)





Decision tree algorithm

Where do the subsets come from? General idea of splitting the dataset

1. For each feature j, determine best potential cut-off point s:
   - For regression task: determine cut-off point by minimizing the MSE of y
     (MSE tells us how much the y values in a node are spread around their mean value)
   - For classification task: determine cut-off point by minimizing Gini index
     (The Gini index tells us how "impure" a node is, e.g. if all classes have the same frequency, the node is impure, if only one class is present, it is maximally pure)
2. After the best cutoff per feature has been determined: select the feature for splitting that results in the best partition (in terms of the MSE or Gini index and adds this split to the tree)
3. Continue this search-and-split recursively in both new nodes until a stop criterion is reached, for example
   - Minimum number of instances that have to be in a node before the split
   - Minimum number of instances that have to be in a terminal node
   - Maximal depth of tree

Cut-off point for regression task

[Hastie et al. 2009, scikit-learn]

Regression task: determine cut-off point s by minimizing the MSE

$$N_m = |\mathbf{x}_i \in R_m|$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{y_i \in R_m} y_i$$

$$H(R_m) = \frac{1}{N_m} \sum_{y_i \in R_m} (y_i - \hat{c}_m)^2$$

Classification task: determine cut-off point s by minimizing Gini index

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{y_i \in R_m} I(y_i = k)$$

$$H(R_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Cut-off point for regression task

[Hastie et al. 2009, scikit-learn]

Given a region R_m and feature j, the region R_m is split into$R_m^{left}(s)$ and $R_m^{right}(s)$ with cut-off point s by minimizing

$$\min_s \frac{N_m^{left}}{N_m} H\left(R_m^{left}\right) + \frac{N_m^{right}}{N_m} H\left(R_m^{right}\right)$$

where

$$R_m^{left}(s) = \{X | X_j \leq s\}$$

$$R_m^{right}(s) = \{X | X_j > s\}$$

Example: Compute split values for s=0.15 and s=0.6

x	y
0	0
0.3	300
0.5	500
0.7	700
1.0	1000

Split s=0.15

Split s=0.6

Example: Compute split values for s=0.15 and s=0.6

x	y
0	0
0.3	300
0.5	500
0.7	700
1.0	1000

Split s=0.15

$$\hat{c}_1 = 0$$

$$\hat{c}_2 = \frac{300 + 500 + 700 + 1000}{4} = 625$$

$$\frac{1}{5} \cdot 0$$

$$\frac{4}{5} \cdot \frac{1}{4}\left((300 - 625)^2 + (500 - 625)^2 + (700 - 625)^2 + (1000 - 625)^2\right)$$

$$= 53500$$

Split s=0.6

$$\hat{c}_1 = \frac{0 + 300 + 500}{3} = \frac{800}{3}$$

$$\hat{c}_2 = \frac{700 + 1000}{2} = 850$$

$$\frac{3}{5} \cdot \frac{1}{3}\left(\left(0 - \frac{800}{3}\right)^2 + \left(300 - \frac{800}{3}\right)^2 + \left(500 - \frac{800}{3}\right)^2\right)$$

$$\frac{2}{5} \cdot \frac{1}{2}\left((700 - 850)^2 + (1000 - 850)^2\right)$$

$$\approx 34333$$

→ **Choose $s = 0.6$**

Alternative to Gini index: entropy

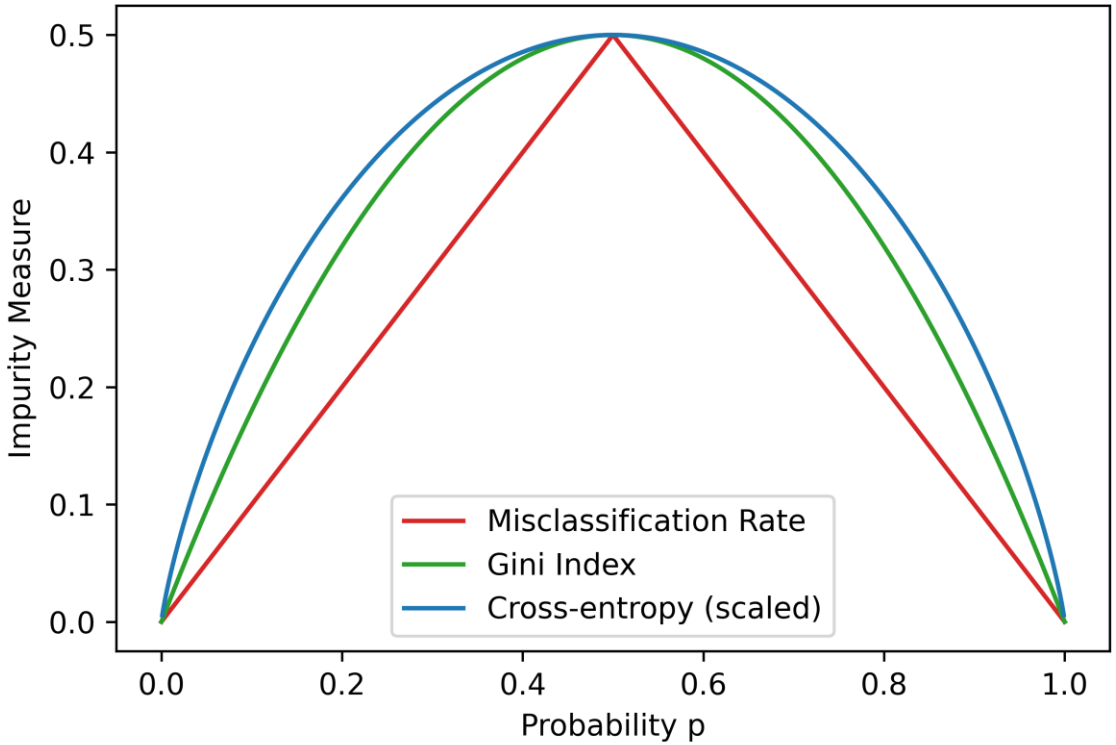
[Hastie et al. 2009, Raileanu et al. 2004]

Gini index

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Alternative: entropy

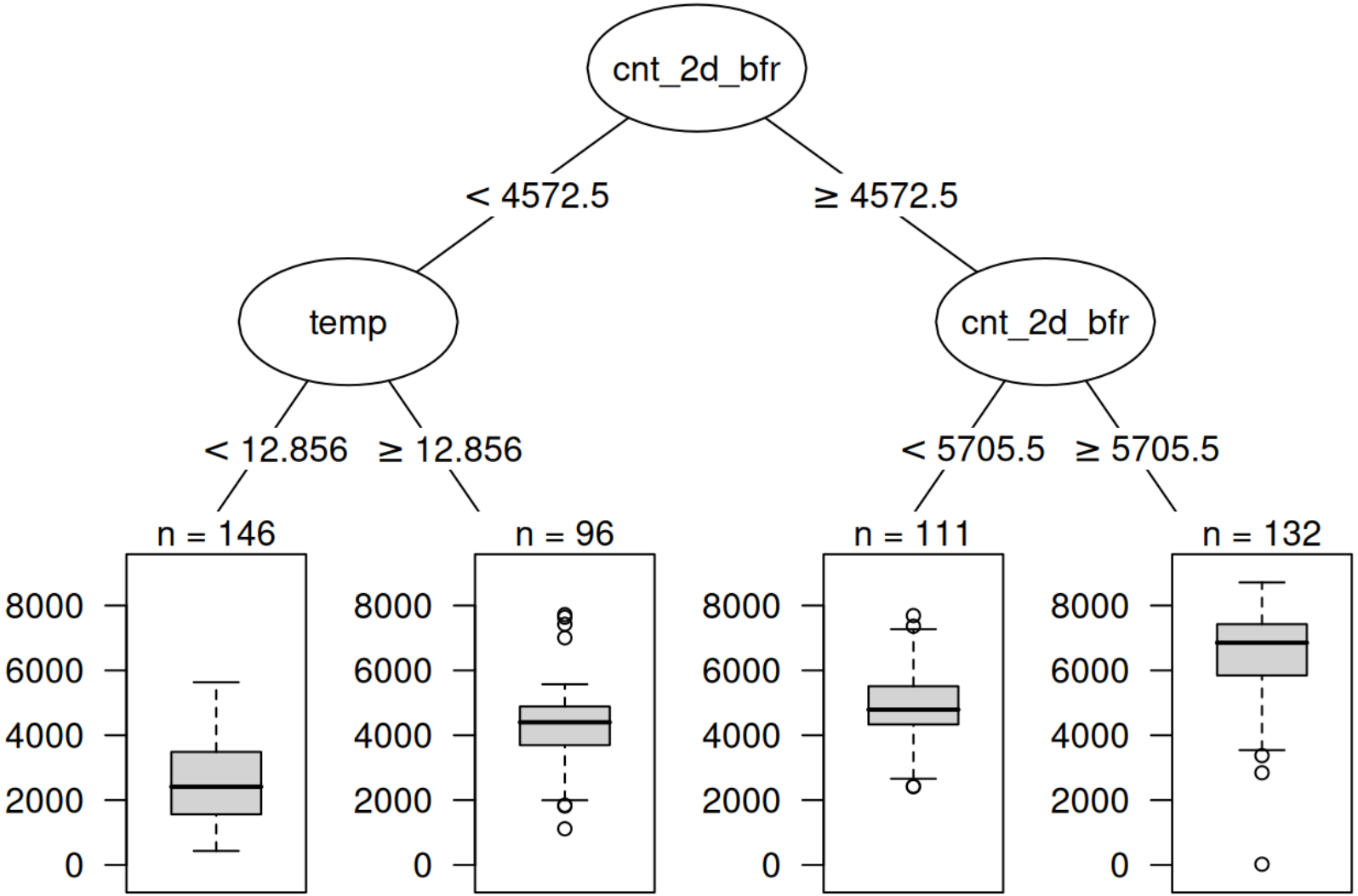
$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$



"We found that they [Gini index and cross entropy] disagree only in 2% of all cases, which explains why most previously published empirical results concluded that it is not possible to decide which one of the two tests performs better." (Raileanu et al. 2004)

Decision tree

Example: Regression tree fitted on the bike rental data



How to interpret the tree?

Interpretation of decision tree

Simulatability (human can easily simulate the decision-making process of model)

Simulatability

- Go from root to leaf (edges tell you which subset you are looking at)
- All edges are connected by ‘AND’
- Predicted outcome is the mean value of y of the instances in that node

First split performed with feature `cnt_2d_bfr`

- < 4572.5
- >= 4572.5

Second split performed with

- Temperature (`temp`)
  - If below 12.856: predict ~2000
  - If above 12.856: predict ~4000
- Previous bike count feature (`cnt_2d_bfr`)
  - If below 5705.5: predict ~5000
  - If above 5705.5: predict ~7000

Interpretation of decision tree

Feature importance (for global explanation of whole decision tree) [scikit-learn]

Reduction of impurity of a single split

- Measure how much the split has reduced the MSE or Gini index compared to the parent node ("impurity of parent minus sum of child impurities"):

$$\frac{N_m}{N}\left(H(R_m) - \left(\frac{N_m^{left}}{N_m}H(R_m^{left}) + \frac{N_m^{right}}{N_m}H(R_m^{right})\right)\right)$$

- Note the term $\frac{N_m}{N}$ where N_m denotes the number of nodes in a region and N the number of all nodes in the tree. The reduction of impurity is weighted according to this term.
- This reduction in impurity is also called information gain

Reduction of impurity of a feature

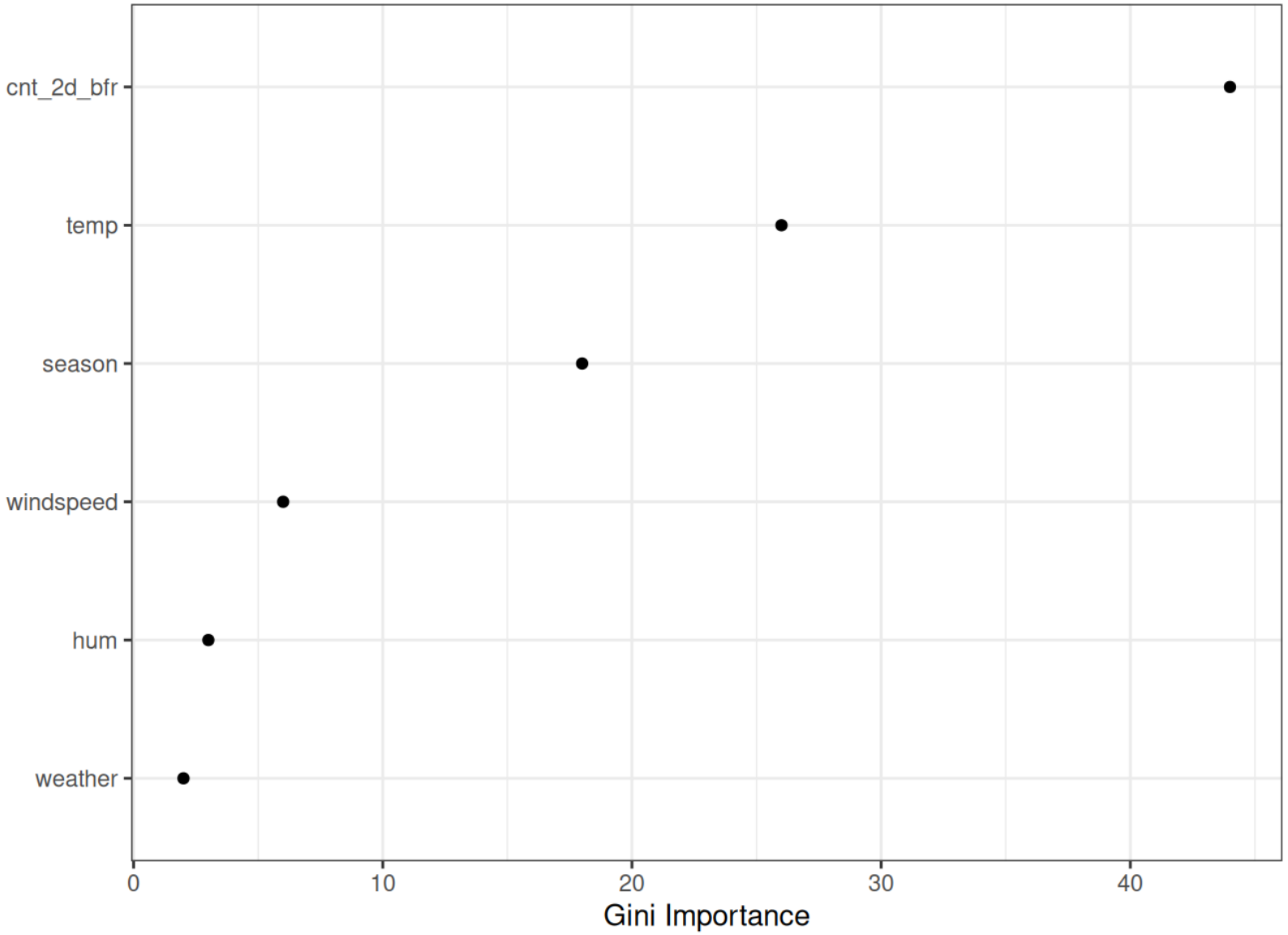
- Go through all the splits in the complete tree for which the feature was used
- Sum up their impurities

Feature importance

- For each feature, we know its reduction of impurity (see previous point)
- Normalize them such that they sum to 1 (= 100%)
- Those (normalized) reductions of impurities are the feature importances

Decision tree

Feature importance measured in terms of decrease in node impurity (MSE)



Decision tree

Tree decomposition (for local explanation of a single prediction)

- For the root node, we predict the mean of the whole training data $\bar{y}$
- Explain a prediction $\hat{f}(\mathbf{x})$ by the contributions added at each decision node $1, \ldots, D$

$$\hat{f}(\mathbf{x}) = \bar{y} + \sum_{d=1}^{D} \text{split. contrib}(d, \mathbf{x})$$

where

$$\text{split. contrib}(d, \mathbf{x}) \coloneqq \bar{y}_{child} - \bar{y}_{parent}$$

and $\bar{y}_{child}$, $\bar{y}_{parent}$ denote the average target value of all the instances in the region of the child node and the parent node of split d, respectively

- If a feature appears in multiple splits, combine their contributions

$$\hat{f}(\mathbf{x}) = \bar{y} + \sum_{d=1}^{D} \text{split. contrib}(d, \mathbf{x}) = \bar{y} + \sum_{j=1}^{p} \text{feat. contrib}(j, \mathbf{x})$$

Decision tree
Advantages

Tree structure is ideal for capturing interactions between features

Tree is easier to understand than linear regression
(data ends up in distinct groups, instead of a multi-dimensional hyperplane)

Tree structure has natural visualization: nodes and edges

Human-friendly explanations

- Contrastive: can compare the prediction of an instance with “what if”-scenarios (that are simply the other leaf nodes of the tree)
- Selective: if the tree is short, the resulting explanations are selective
- Truthfulness: depends on the predictive performance of the tree

No need to transform features (such as one-hot encoding of categorical data, logarithmic transformations, …)

Decision tree
Disadvantages

Trees fail to deal with linear relationships

- Linear relationship between an input feature and the outcome has to be approximated by splits (creating a step function)

This goes hand in hand with lack of smoothness

- Slight changes in input feature can have a big impact on the predicted outcome
- Example: predict the value of a house
  - 99 square meters: prediction 200 000 EUR
  - 100 square meters: prediction 200 000 EUR
  - 101 square meters: prediction 210 000 EUR

Trees are also quite unstable

- A few changes in the training dataset can create a completely different tree (this is because each split depends on the parent split)
- If different feature is selected as first split, the entire tree structure changes
- It does not create confidence in the model if the structure changes so easily

Decision trees are very interpretable – as long as they are short

- The number of terminal nodes increases exponentially with depth

Decision Rules

Decision rules
Introduction

Simple IF-THEN statement consisting of

- a condition (also called antecedent) and
- a prediction
- Example:
  - IF it rains today AND if it is April (condition)
  - THEN it will rain tomorrow (prediction)

Might be most interpretable prediction models

IF-THEN structure resembles natural language and the way we think, provided that

- the condition is built from intelligible features
- the length of the condition is short (small number of feature=value pairs combined with an AND)
- there are not too many rules

Decision rules

Example

Predict the value of a house (`low`, `medium` or `high`)

One rule: IF `size>100 AND garden=1` THEN `value=high`

- `size>100` is the first condition in the IF-part
- `garden=1` is the second condition in the IF-part
- The two conditions are connected with an ‘AND’ to create a new condition. Both must be true for the rule to apply.
- The predicted outcome (THEN-part) is `value=high`

A decision rule uses at least one `feature=value` statement in the condition (an exception is the default rule that has no explicit IF-part and that applies when no other rule applies)

Decision rules

Measuring the usefulness of a rule

Support (aka coverage)

- Percentage of instances to which the condition of a rule applies
- Example: `IF size=big AND location=good THEN value=high`
- Suppose 100 of 1000 houses are big and in a good location, then the support of the rule is 10% (the prediction [THEN-part] is not important for the calculation of support)

Accuracy (aka confidence)

- Measures of how accurate the rule is in predicting the correct class for the instances to which the condition of the rule applies
- Example: Out of the 100 houses, where the previous rule applies
  - 85 have `value=high`
  - 14 have `value=medium`
  - 1 has `value=low`
  - → accuracy of the rule is 85%

Usually there is a trade-off between accuracy and support

- Generally: more features in a condition → higher accuracy, less support

Decision rules

Interaction of multiple rules

A good classifier might require multiple rules

- Rules can overlap: How to handle contradictory explanations?
- No rule applies: What to do then?

Decision list (ordered)

- Use first rule that applies

Decision set (unordered)

- Ensure that rules are mutually exclusive or
- Use majority voting of the predictions
  (different rules can have different weights, e.g. based on the rule accuracies)

Default rule

- Both decision lists and sets can suffer from the problem that no rule applies

→ Use default rule (e.g., predict most frequent class of data points not covered by rules)

OneR

Learn rules from a single feature

Idea

- Select one feature that carries most information about the outcome of interest
- For each feature value, create a rule (multiple rules are created)

Algorithm

1. Discretize the continuous features by choosing appropriate intervals
2. For each feature
    1. Create a cross table between the feature values and the (categorical) outcome
    2. For each value of the feature, create a rule which predicts the most frequent class of the instances that have this particular feature value (can be read from the cross table)
    3. Calculate the total error of the rules for the feature
3. Select the feature with the smallest total error

Notes

- OneR always covers all instances.
- OneR is a decision tree with only one split

OneR

Example

location	size	pets	value
good	small	yes	high
good	big	no	high
good	big	no	high
bad	medium	no	medium
good	medium	only cats	medium
good	small	only cats	medium
bad	medium	yes	medium
bad	small	yes	low
bad	medium	yes	low
bad	small	no	low

OneR

Cross tables

	value=low	value=medium	value=high
location=bad	3	2	0
location=good	0	2	3

	value=low	value=medium	value=high
size=big	0	0	2
size=medium	1	3	0
size=small	2	1	1

	value=low	value=medium	value=high
pets=no	?	?	?
pets=only cats	?	?	?
pets=yes	?	?	?

OneR

Cross tables

	value=low	value=medium	value=high
location=bad	3	2	0
location=good	0	2	3

	value=low	value=medium	value=high
size=big	0	0	2
size=medium	1	3	0
size=small	2	1	1

	value=low	value=medium	value=high
pets=no	1	1	2
pets=only cats	0	2	0
pets=yes	2	1	1

OneR

Learned rules

	value=low	value=medium	value=high
size=big	0	0	2
size=medium	1	3	0
size=small	2	1	1

IF `size=small` THEN `value=low` (coverage: 40%, acc: 50%)

IF `size=medium` THEN `value=medium` (coverage: 40%, acc: 75%)

IF `size=big` THEN `???????` (coverage: `????` acc: `????`)

OneR

Learned rules

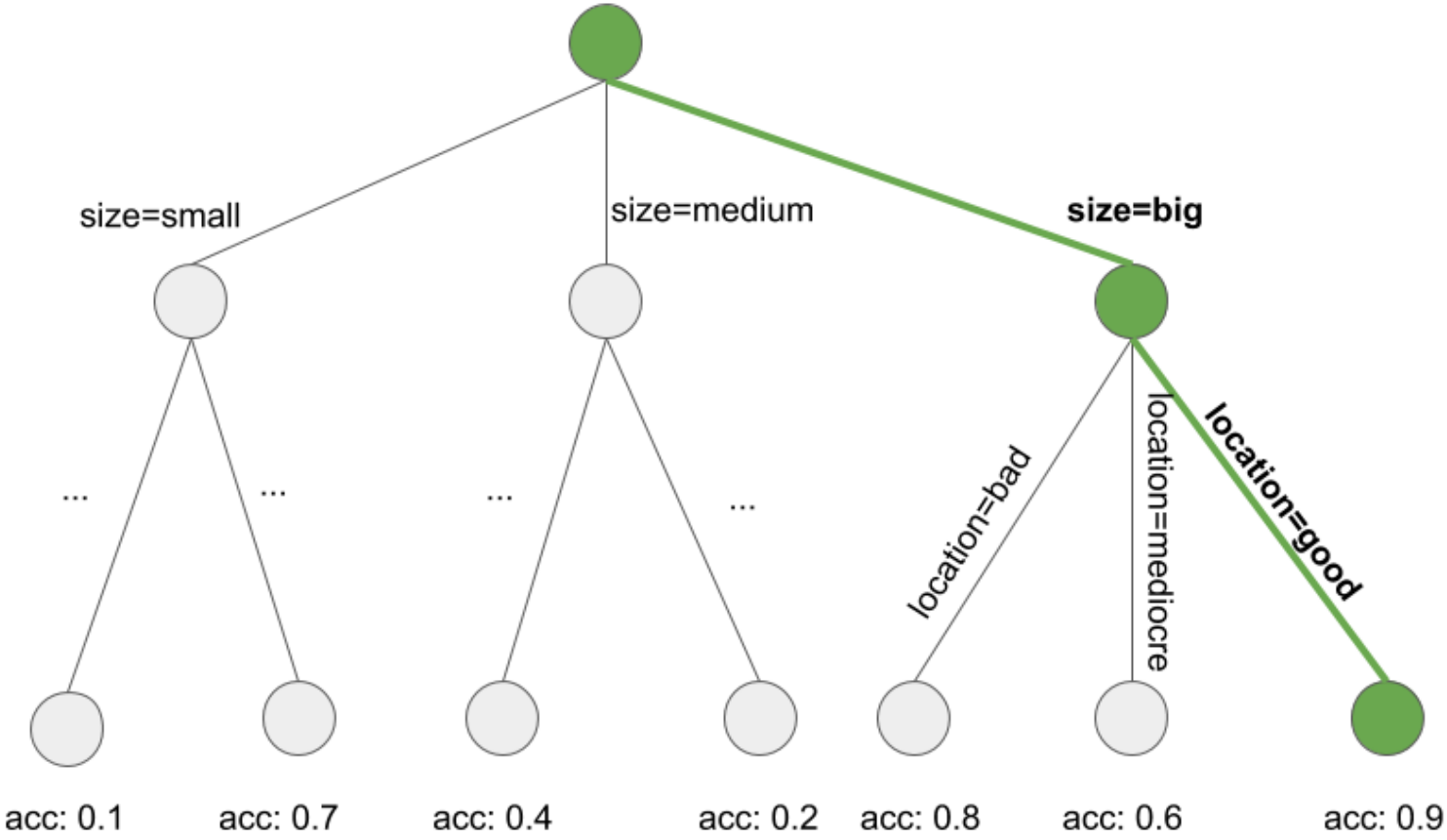
	value=low	value=medium	value=high
size=big	0	0	2
size=medium	1	3	0
size=small	2	1	1

IF `size=small` THEN `value=low` (coverage: 40%, acc: 50%)

IF `size=medium` THEN `value=medium` (coverage: 40%, acc: 75%)

IF `size=big` THEN `value=high` (coverage: 20%, acc: 100%)

Decision tree for rule learning



Learned rule: IF `location=good` AND `size=big` THEN `value=high`

Sequential covering

Idea: Repeatedly learn a single rule to create a decision list

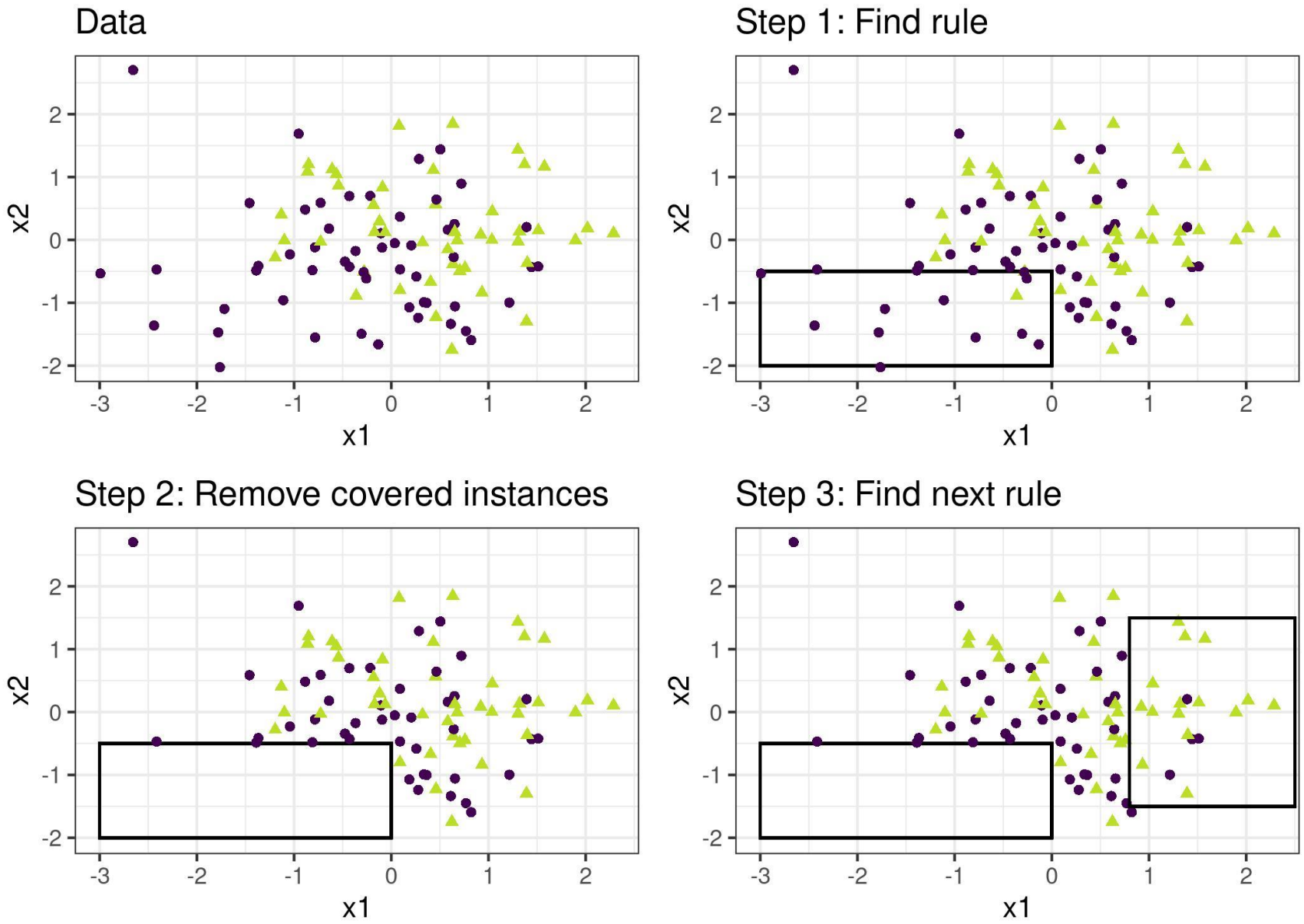
1. Find a good rule that applies to some of the data points
2. Remove all data points which are covered by the rule (a data point is covered when a condition applies, regardless of prediction)
3. Repeat rule-learning and removal of covered points with the remaining points until no more points are left or another stop condition is met. The result is a decision list.

Algorithm

1. Start with an empty list of rules (rlist)
2. Learn a rule r (e.g., with OneR or a tree learning algorithm)
3. While the list of rules is below a certain quality threshold:
    1. Add rule r to rlist
    2. Remove all data points covered by rule r
    3. Learn another rule on the remaining data
4. Return the decision list

Sequential covering

Example



Example: Bike rental dataset

RIPPER: Fast effective rule induction [Cohen et al, 1995]

Rules
(cnt_2d_bfr <= 5729) and (cnt_2d_bfr >= 4780) and (temp <= 26) => cnt=(4551, 5978.5]
(temp >= 14) and (cnt_2d_bfr <= 5515) and (hum <= 63) and (cnt_2d_bfr >= 3574) and (temp <= 27) => cnt=(4551, 5978.5]
(cnt_2d_bfr >= 3544) and (cnt_2d_bfr <= 3915) and (temp >= 17) and (temp <= 28) => cnt=(4551, 5978.5]
(cnt_2d_bfr <= 5336) and (cnt_2d_bfr >= 2914) and (weather = GOOD) and (temp >= 7) => cnt=(3193, 4551]
(cnt_2d_bfr <= 4833) and (cnt_2d_bfr >= 2169) and (hum <= 72) and (workday = Y) => cnt=(3193, 4551]
(cnt_2d_bfr <= 4097) and (season = WINTER) => cnt=[22, 3193]
(cnt_2d_bfr <= 4570) and (temp <= 13) => cnt=[22, 3193]
(hum >= 88) and (season = FALL) => cnt=[22, 3193]
(cnt_2d_bfr <= 3351) and (cnt_2d_bfr >= 2710) => cnt=[22, 3193]
=> cnt=(5978.5, 8714]

IF-THEN decision rules

Advantages

Rules are easy to interpret

- If the number of rules is small
- If the conditions of the rules are short
- If the rules are organized in a decision list or a non-overlapping decision set

Rules are compact

- In contrast to decision trees, which often suffer from replicated sub-trees

Rule predictions are fast

- Only a few binary statements need to be checked to determine which rules apply

Rules are robust

- Against monotonic transformations of the input features (because only the threshold in the conditions changes)
- Against outliers, since it only matters if a condition applies or not

Rules generate sparse models

- Sparse models: models with few features

IF-THEN decision rules

Disadvantages

Rules are less suitable for regression

- It is possible to divide a continuous target into intervals and turn it into a classification problem
- But: you loose information this way

Features need to be categorical

- Continuous numeric features must be categorized (into intervals)
- There are many ways to do it (fixed-size interval, quantiles, mutual information)
- But: you loose information this way

Rules are bad in describing linear relationships

- Decision trees suffer from the same problem
- Rules and decision trees only produce step-like prediction functions (changes in the prediction are always discrete steps and never smooth curves)
- This issue is related to the input of categorical features

Outlook: Other rule learning algorithms

Bayesian rule lists

- Identify frequent patterns in the dataset (e.g., via APRIORI, FP-Growth)
- Construct multiple rule lists from patterns (prefer short lists with short rules and high accuracy)
- Select best list (according to criteria based on Bayesian statistics)

RuleFit (combination of rule learning and linear models)

- Learn decision trees on data and convert them to rules
- Add the learned rules as additional features
- Train linear model (e.g., LASSO) on the combined features (original + new)

Genetic algorithm

- Rules are represented as trees (abstract syntax trees)
- Evolutionary operations are applied (selection, crossover, mutation)
- Best rule is selected according to fitness function

References

- **Cohen**, William W. "Fast effective rule induction." In Machine learning proceedings 1995, pp. 115-123. Morgan Kaufmann, **1995**.
- **Hastie**, Trevor, Robert Tibshirani, Jerome H. Friedman, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. New York: springer, **2009**. https://hastie.su.domains/ElemStatLearn/printings/ESLII_print12_toc.pdf
- **Raileanu**, Laura Elena, and Kilian Stoffel. "Theoretical comparison between the gini index and information gain criteria." *Annals of Mathematics and Artificial Intelligence* 41 (**2004**): 77-93.
- **Scikit-learn.** https://scikit-learn.org/stable/modules/tree.html#mathematical-formulation