Submitted by
Martin Dallinger

Submitted at
LIT Secure and Correct Systems Lab, Secure Systems Group

Supervisor
Univ.-Prof. DI DI Dr. Stefan Rass

April 2025

Fuzzy Rule-Based Regression for Explainable AI

Bachelor's Thesis
to obtain the academic degree of
Bachelor of Science
in the Bachelor's Program
Artificial Intelligence

JOHANNES KEPLER UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Statutory Declaration

I hereby declare under oath, that the submitted thesis has been written solely by me without any third-party assistance. Only the declared sources and/or resources have been used. Sources for all literal, paraphrased and cited quotes have been accurately credited. In particular, I have clearly and unambiguously explained or credited all usage of generative AI.

The submitted document here present is identical to the electronically submitted document.

I am aware that the violation of this regulation will lead to failure of the thesis.

Signiert von:	Martin Franz Dallinger
Datum:	07.04.2025 20:59:42

TRUST

Dieses Dokument ist digital signiert!
Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.

Prüfinformation: Informationen zur Prüfung der elektronischen Signatur finden Sie unter: www.a-trust.at/pdf

Martin Dallinger
Name

Signature

Linz, 07.04.2025
Place, Date

Abstract

Modern AI systems are only as fair as the data they learn from. If a dataset contains biases, subtle or undisguised, the resulting models often reflect and reinforce them. This work introduces an open-source prototype to detect such biases at the data level. This enables bias-aware decisions regarding further data processing or model design before harmful data properties propagate into machine learning models. Utilizing rule-based fuzzy inference and extensive filtering built around Lasso regression and association rule learning techniques, the tool systematically extracts interpretable patterns that highlight potential unfairness or inconsistencies in datasets.

In contrast to classical fairness metrics, which quantify disparities and do not explain their cause, or classical fuzzy rule-based methods, which do not focus on fairness and limit the rule-bases, this approach aims to discover and quantify all possible bias-inducing rules. The system exhaustively generates and carefully filters all potential rules up to a given complexity threshold. Statistical significance testing ensures that the remaining rules are not merely artifacts but genuinely meaningful in explaining the dataset. This rigorous filtering allows the approach to also detect subtle, indirect biases that conventional methods are likely to overlook.

Experiments on synthetic and real-world datasets validate the method and reveal gender-based salary biases, socio-economic impacts on housing prices as well as other structured phenomena hidden in data. Compared to classical rule-based strategies, the method in this work favors inclusiveness over brevity, allowing critical, albeit subtle, patterns of biases to be discovered.

Rather than assuming the dataset represents an absolute truth, our approach focuses on identifying potential errors and biases by extracting if-then rules that reveal the underlying mechanisms behind these distortions. This work offers a tool for researchers, auditors and machine learning practitioners looking to understand and mitigate biases at their source — before they become embedded in models that influence real-world decisions.

Acknowledgements

I would like to express sincere appreciation to Univ.-Prof. Priv.-Doz. DDI Dr. Rass for his helpful support and insightful guidance. Both his academic and practical expertise in the field have certainly steered this research in a meaningful direction. Moreover, I would like to thank Lisa Krimbacher for the extensive shared proofreading sessions, which have greatly improved the coherence and clarity of this work.

Also note that the large language model GPT 4o (OpenAI) was occasionally utilized as an auxiliary resource to improve comprehensibility and to enhance certain formulations. Its role was restricted to improving linguistic coherence and precision. This helped in ensuring that complex ideas were conveyed with the intended nuance to readers largely unfamiliar with the covered topics. The intellectual development, critical analysis and results outlined in this work, as well as those published in [1], are entirely the product of independent research and reasoning.

Contents

1 Introduction and Motivation

Ensuring fairness in AI-driven decision-making starts with understanding the data from which models learn. If a dataset contains undesired properties, whether explicit or subtle, the resulting models are likely to maintain and even amplify these properties. This thesis presents a method for systematically uncovering such patterns in datasets and serves as a technical extension, describing the concrete implementation in our previous publication [1], which focuses on the general methodology.

While fuzzy rule-based explanations with regression have been explored in previous research, for instance by Cichy et al. [2, 3], our goal is to enhance the robustness of these techniques and to improve the accessibility. To achieve this, we propose an open-source prototype under the GPLv3 license [4], designed to be both user-friendly and executable directly in a web browser[1] [5]. For client-server use cases and improved performance, a Node.js-based (API) implementation is also available. In this work, we explicitly describe commit `b338d80` of [5], leaving the repository up to future changes, such as (but not limited to) those outlined in Section 6.2.

Nonetheless, it is important to emphasize that the method proposed in [1] is more general and requires a concrete way of modelling a rule as a mathematical basis function for the use in regression. Therefore, this work represents one concrete implementation that builds upon the well-established Fuzzy Inference Systems (FIS) described in Section 2.1.

As Mehrabi et al. [6] define it, *bias* in Machine Learning (ML) commonly stems from unfair or undesired properties inherent in datasets. In decision-making scenarios, this can lead to discrimination against certain groups. Unwanted biases manifest in various forms, and ML methods may inadvertently perpetuate them [7]. Some of these effects remain subjects of ongoing research and can be challenging to detect [6, 8, 9]. This issue is part of the broader field of eXplainable Artificial Intelligence (XAI), which focuses on providing explanations and interpretability for datasets, models and their outputs. The described method is applied at the preprocessing stage of a ML project, as it focuses on the data and is independent of the model application, which will be performed once the dataset can be assumed not to contain harmful biases.

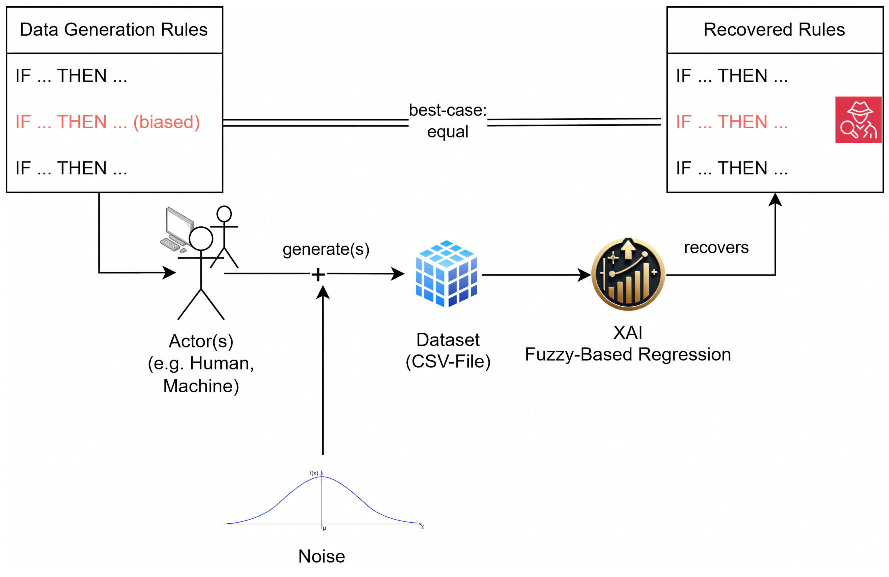
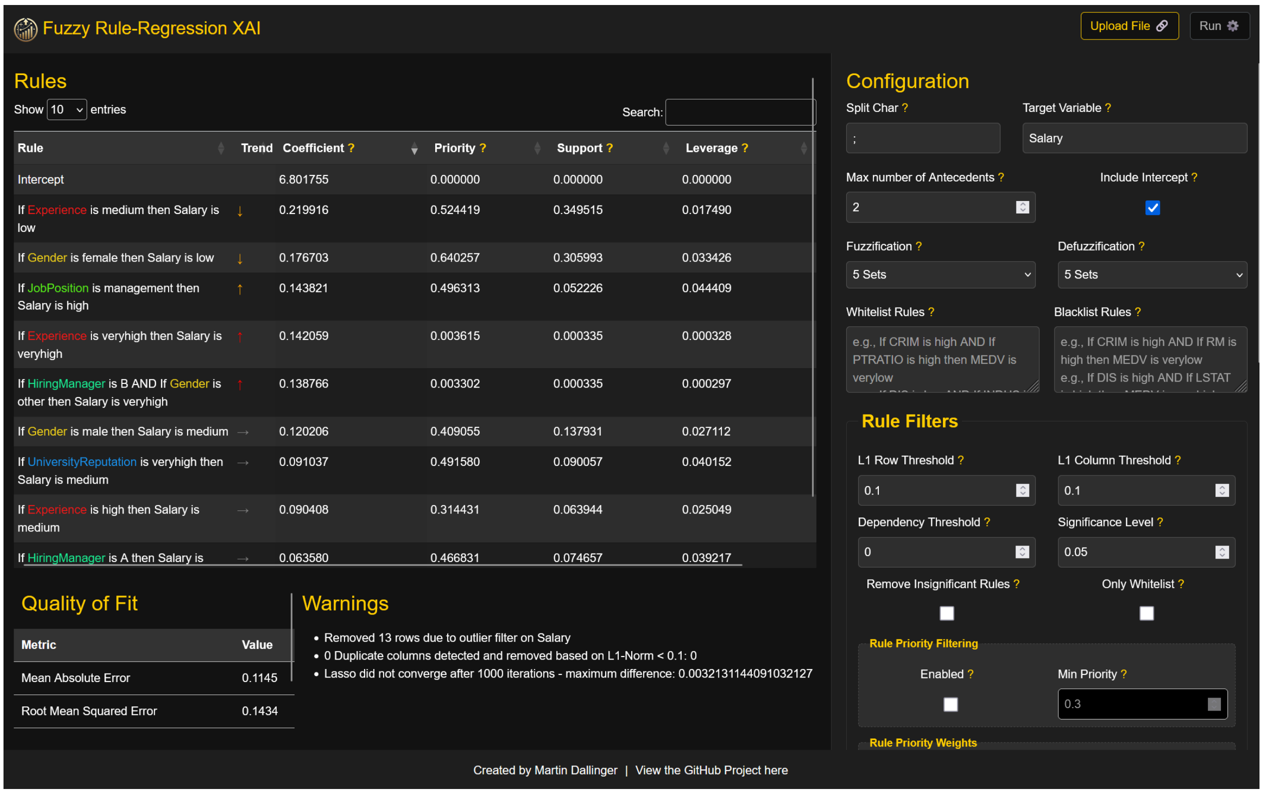


Figure 1.1: Recovering data generation rules expressed in natural language

[1]`https://xai.martin-dallinger.me`

The prototypical process when using this method is displayed in Figure 1.1. One aims to analyze datasets with numeric target variables by modeling rules that may have generated them and assessing their respective statistical significance. This enables the discovery of unfair patterns or inconsistencies in data generation processes, which is the main objective of this work and presents a different challenge from merely explaining individual records. For instance, in a loan approval setting, a loosely defined policy might give bank staff a degree of discretion, possibly leading some to favor certain customer groups. Such decision-making tendencies could be identified with this tool if the collected data reflects these influences.

An exemplary analysis in the web application is presented in Figure 1.2 with the “biased salaries” dataset from Section 1.4.2. The configuration parameters for the software are described in Section 7.2 (Appendix) and we will go through them step by step when we get into the details of the method. It is important to note that when the tool recovers the rules that correspond to the data generation process, one should not think that their objective is to explain the data as an absolute truth. Instead, their objective is to show the hidden discrepancies and biases.



Rules

Show 10 entries

Search:

Rule	Trend	Coefficient ?	Priority ?	Support ?	Leverage ?
Intercept		6.801755	0.000000	0.000000	0.000000
If Experience is medium then Salary is low	↓	0.219916	0.524419	0.349515	0.017490
If Gender is female then Salary is low	↓	0.176703	0.640257	0.305993	0.033426
If JobPosition is management then Salary is high	↑	0.143821	0.496313	0.052226	0.044409
If Experience is veryhigh then Salary is veryhigh	↑	0.142059	0.003615	0.000335	0.000328
If HiringManager is B AND If Gender is other then Salary is veryhigh	↑	0.138766	0.003302	0.000335	0.000297
If Gender is male then Salary is medium	→	0.120206	0.409055	0.137931	0.027112
If UniversityReputation is veryhigh then Salary is medium	→	0.091037	0.491580	0.090057	0.040152
If Experience is high then Salary is medium	→	0.090408	0.314431	0.063944	0.025049
If HiringManager is A then Salary is	→	0.063580	0.466831	0.074657	0.039217

Quality of Fit

Metric	Value
Mean Absolute Error	0.1145
Root Mean Squared Error	0.1434

Warnings

- Removed 13 rows due to outlier filter on Salary
- 0 Duplicate columns detected and removed based on L1-Norm < 0.1: 0
- Lasso did not converge after 1000 iterations - maximum difference: 0.0032131144091032127

Configuration

Split Char ?	;
Target Variable ?	Salary
Max number of Antecedents ?	2
Include Intercept ?	✓
Fuzzification ?	5 Sets
Defuzzification ?	5 Sets
Whitelist Rules ?	e.g., If CRIM is high AND If PTRATIO is high then MEDV is verylow
Blacklist Rules ?	e.g., If CRIM is high AND If RM is high then MEDV is verylow e.g., If DIS is high AND If LSTAT

Rule Filters

L1 Row Threshold ?	0.1
L1 Column Threshold ?	0.1
Dependency Threshold ?	0
Significance Level ?	0.05
Remove Insignificant Rules ?	☐
Only Whitelist ?	☐

Rule Priority Filtering

Enabled ?	☐
Min Priority ?	0.3

Rule Priority Weights

Created by Martin Dallinger | View the GitHub Project here

Figure 1.2: Image of the application with results from the biased salaries dataset (Section 1.4.2)

1.1 Related Work

Conventional fairness metrics (e.g., disparate impact or equality of opportunity [10]) quantify differences in outcomes — such as error rates or decision thresholds — across various groups defined by protected attributes or by comparing well- and underrepresented groups. However, these statistical measures do not explain the cause of biases. XAI techniques and rule-based systems have thus been leveraged to extract interpretable conditions under which biases emerge [11].

Some of these rule-based systems try to extract logical rules from data for describing relationships between the inputs and the outputs [12], thereby viewing the data as “ground truth”. One classical approach that infers rules is the Wang-Mendel procedure [13], originally developed for FIS. This method automatically generates a relatively compact set of fuzzy if-then rules from numerical data, optionally combined with expert-defined rules, to construct a knowledge base for inference. While the Wang-Mendel approach is a prominent procedure for learning interpretable rule-based models, it does not focus explicitly on bias detection, as the primary objective is function approximation. Although this strategy is efficient, a main shortcoming of this and similar methods is that they might be overly data-specific in the sense that they commonly only consider single dataset entries in their ruleset creation. Therefore, unwanted dataset properties (like biases) and other data generation rules, where no single entry of the dataset directly represents the bias but a more abstract, holistic view is required, are likely to remain hidden [1]. This issue is further compounded by the fact that unknown biases can render the assumption of data as "ground truth" inappropriate. Moreover, not all rule inference methods provide a quantification for the importance of a rule to the extent of employing statistical significance tests for determining whether specific rules indeed have an effect in the context of the entire dataset.

Other approaches, such as Fuzzy Rule Learning through Evolution for Regression (FRULER) [14] or other genetic algorithm-based methods as proposed by Cintra et al. [15], iteratively build rule bases, partially with advanced forms of Takagi-Sugeno-Kang (TSK) systems. In contrast, the method presented in this work explicitly targets bias detection by considering a broad spectrum of all possible rules, rather than limiting itself to a set of rules that appear directly in individual data records and strategically evolving this set. Through multiple filtering techniques — like Least Absolute Shrinkage and Selection Operator (Lasso) regression for sparsity, and statistical testing for rule significance — this procedure, which is based our previous work [1], performs more than the simple discovery of patterns. Rather, it enables one to quantitatively assess which rules are biasing the outcome and to find statistically significant inconsistencies in a dataset. Therefore, the approach outlined in this thesis differs from previous work conducted in the realm of rule inference.

Although fitting all possible rules up to a predefined complexity threshold is computationally demanding, it provides a formal guarantee that all potential rules are considered, which may uncover hidden biases or other interesting patterns in the data. Moreover, whenever two different rules have the same mathematical representation on the entire dataset, our method lists both variations. Consequently, more runtime-efficient rule-generation methods, such as Wang-Mendel, may fail to detect certain indirect biases that this approach can reveal.

Beyond fuzzy systems, other rule-based bias detection methods have been proposed [16, 17, 18]. A prominent approach is to mine classification or association rules from decision-making datasets to identify discriminatory patterns. Pedreschi et al. [17] formalize discrimination discovery by extracting classification rules that unveil "contexts of unlawful discrimination", quantifying the burden on protected groups via extended lift measures. This rule-based method also isolates particular combinations of attributes, which are associated with biased decisions, but tends to output long rule sets and does not assess them for statistical significance, leading to challenges in interpretability. Additionally, some paradigms are focused on biased model decisions and do not directly consider unfair patterns on datasets. This thesis highlights bias in raw data by listing weighted rules, independent of the applied AI model.

Meanwhile, model-agnostic XAI techniques have been leveraged to audit bias from a different perspective. Feature attribution methods such as SHAP [19] and LIME [20] highlight which fields most influence a model's predictions for different groups, potentially exposing indirect bias. However, these are post-hoc evaluations: They operate on trained models and offer little guidance in terms of dataset bias analysis before model selection.

Briefly summarized, existing work mostly focuses on efficient rule generation (Wang-Mendel), discrimination discovery in the decision outcome (rule mining) and post-hoc feature importance (SHAP / LIME), while this method rethinks these ideas in a **preprocessing bias detection framework** with a focus on completeness and formal statistical backing. It achieves this goal by generating a vast amount of rules and filtering the majority of them with heuristics from different areas. This enables revealing and quantifying biased or contradictory dataset patterns prior to model training.

1.2 Notation and Symbols

To maintain consistency and avoid confusion, we adopt terminology and variable names (listed in Table 1.1) similar to those used in our original work [1]. Let us agree to use the following conventions:

- Scalars are represented by lowercase letters.
- Vectors are denoted by lowercase boldface letters. Nonetheless, when accessing the i-th element of a vector $\mathbf{v}$ the result is a scalar and also the symbol denoting the vector is not printed in bold font, i.e., denoted v_i.
- Sets are indicated by uppercase letters.
- Matrices are represented by uppercase boldface letters.

For instance, consider a matrix $\mathbf{M}$. We write $\mathbf{M} = (m_{ij})$, to make its ij-th entry explicit, where the first index (i) always represents the row and the second index (j) refers to the column. In addition:

- $\mathbf{m}_{i,\cdot}$ denotes the i-th row (vector) of $\mathbf{M}$,
- $\mathbf{m}_{\cdot,j}$ denotes the j-th column (vector) of $\mathbf{M}$,
- An index with a minus sign indicates exclusion. For instance, $\mathbf{M}_{\cdot,-i}$ is the matrix $\mathbf{M}$ with its i-th column removed. Therefore, $\mathbf{M}_{\cdot,-i}$ can be represented as follows:

$$\mathbf{M}_{\cdot,-i} = (\mathbf{m}_{\cdot,1}, \dots, \mathbf{m}_{\cdot,i-1}, \mathbf{m}_{\cdot,i+1}, \dots, \mathbf{m}_{\cdot,n}) \quad \text{when } \mathbf{M} \text{ has } n \text{ columns.}$$

Further note that, in this work, the set $\mathbb{N}$ is defined as the set of all strictly positive integers (excluding 0) and that $\mathbb{R}_+$ comprises all non-negative real numbers, which in turn includes zero.

The provided *data*, supplied in the form of a Comma-Separated Values (CSV)-file, consists of a set of *records*, where rows (excluding the header) and columns (including the target variable) are mapped into the data matrix $\mathbf{D}$. Each row in the CSV-file represents a record, while the header contains the names of both the *fields* (independent variables) and the strictly numerical target (response) variable. For example, in a loan approval dataset, each record might represent an applicant, with fields such as the applicant's age or gender, while the target variable could indicate the approved loan amount.

We can quickly observe that $\mathbf{D} \in \mathcal{T}^{n \times (k+1)}$, where n is the number of records and k is the number of fields; an additional column represents the target variable. Here, the symbol $\mathcal{T}$ is still an abstract type, as the data may contain textual information for categorical variables as well, which we later also map to numbers. Thus, after all preprocessing steps have been applied, $\mathcal{T} = \mathbb{R}$.

To fulfill the assumptions of the later statistical techniques, the target variable vector $\mathbf{y} \in \mathbb{R}^n$, containing the values of the target variable for all (n) records, is normalized using the z-score method [21] before being stored in $\mathbf{D}$.

The k independent variables of the i-th record are described by the i-th row $\mathbf{x}_{i,\cdot} \in \mathbb{R}^k$ of the matrix $\mathbf{X} = (x_{ij}) \in \mathbb{R}^{n \times k}$. Together with the corresponding target variable the i-th row of $\mathbf{D}$ is constructed as follows: $\mathbf{d}_{i,\cdot} = (x_{i1}, x_{i2}, \cdots, x_{ik}, y_i)$.

Table 1.1: List of symbols in alphabetical order (based on Rass et al. [1])

Symbol	Meaning
$b_j : \mathbb{R}^k \to \mathbb{R}$	single basis function that embodies the j-th if-then rule providing a logic-based description of a pattern that we seek to uncover; as stated in [1], the general method is not restricted to FIS but this implementation uses them
$\mathbf{D} \in \mathbb{R}^{n \times (k+1)}$	data matrix containing the records in its rows; the i-th row ($\mathbf{d}_{i,\cdot}$) containing the independent variables $\mathbf{x}_{i,\cdot}$ and a single response variable (label) y_i, which are concatenated to $\mathbf{d}_{i,\cdot} = (x_{i1}, x_{i2}, \cdots, x_{ik}, y_i)$
$\mathbf{G} \in \mathbb{R}^{n \times m}$	design matrix introduced in Section 2.2 and visualized in Equation (2.6) containing all rule outputs for all records in numerical form
$k \in \mathbb{N}$	number of fields (independent variables)
$m \in \mathbb{N}$	number of basis functions (rules), including the constant intercept
$n \in \mathbb{N}$	number of records (rows) in $\mathbf{D}$
$t_{\max} \in \mathbb{N}$	maximum iterations of Lasso Coordinate Descent (Algorithm 1)
$\alpha \in (0, 1)$	significance level used for statistical testing; the null hypothesis $H_0 : \beta_i = 0$ is used to test whether the i-th basis function has a significant contribution
$\boldsymbol{\beta} \in \mathbb{R}^m$	vector of coefficients; β_i is the Lasso coefficient which is multiplied by the i-th basis function (rule)
$\epsilon \in \mathbb{R}_+$	convergence threshold for the Lasso solution in Algorithm 1
$\kappa \in \mathbb{N}$	number of antecedents (if-parts) that are combined at most in the rule construction process (Section 2); also referred to as the system parameter “Antecedent-Combinations” (Parameter 4 in Figure 7.12)
$\lambda \in \mathbb{R}_+$	Lasso regularization parameter, introduced in Section 4

1.3 Overview of the Workflow

Our approach involves trying to fit all possible “fuzzy” if-then rules to approximate the dataset while accounting for noise, followed by statistical significance testing of these rules. The workflow is compactly illustrated in Figure 1.3, showing how the software iteratively generates and refines a ruleset. In the following sections, we will briefly introduce the fundamental concepts that form the building blocks of this method.

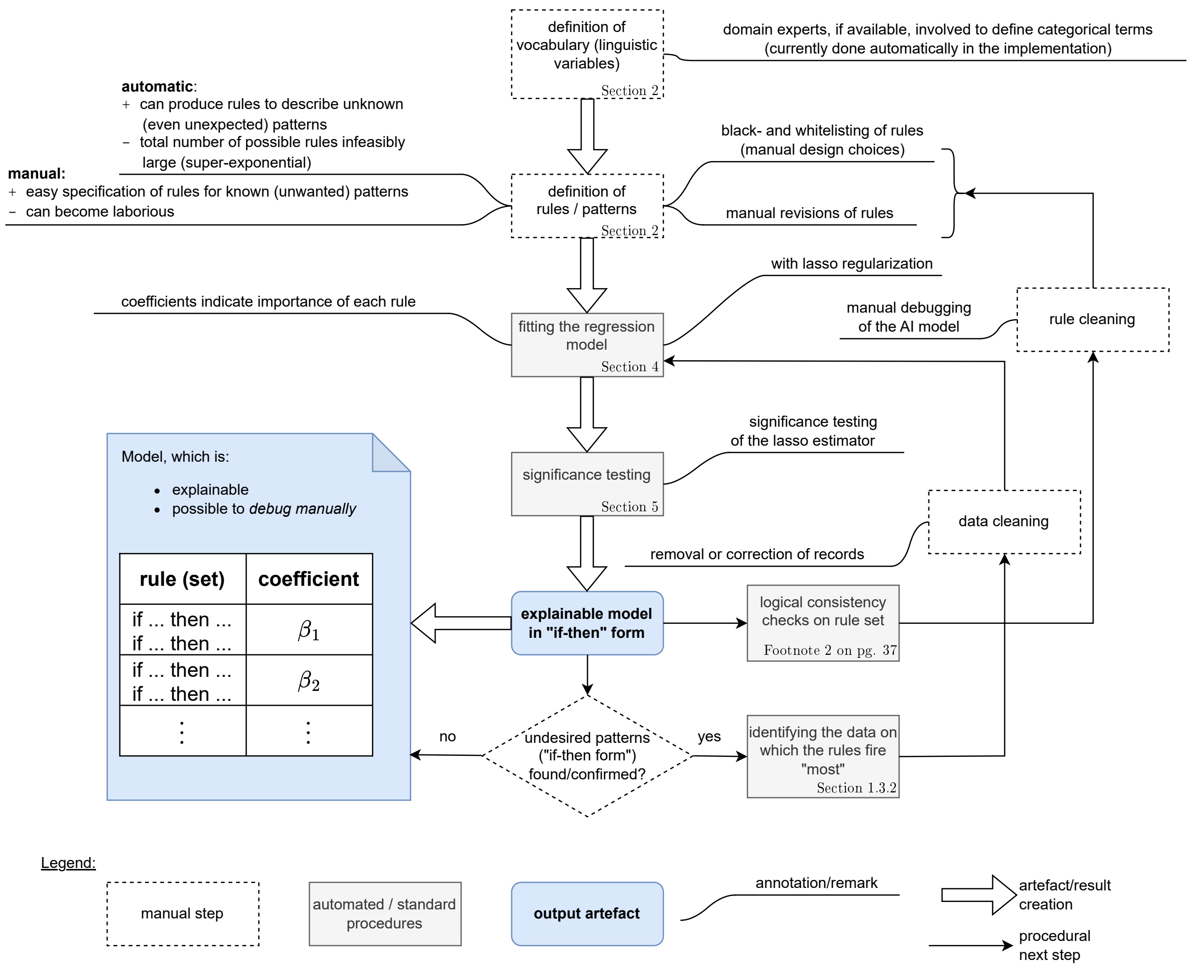

definition of vocabulary (linguistic variables)
Section 2
domain experts, if available, involved to define categorical terms (currently done automatically in the implementation)
automatic:
+ can produce rules to describe unknown (even unexpected) patterns
– total number of possible rules infeasibly large (super-exponential)
manual:
+ easy specification of rules for known (unwanted) patterns
– can become laborious
definition of rules / patterns
Section 2
black- and whitelisting of rules (manual design choices)
manual revisions of rules
with lasso regularization
coefficients indicate importance of each rule
fitting the regression model
Section 4
manual debugging of the AI model
rule cleaning
significance testing of the lasso estimator
significance testing
Section 5
Model, which is:
• explainable
• possible to debug manually
rule (set)
coefficient
if ... then ...
if ... then ...
β_1
if ... then ...
if ... then ...
β_2
⋮
⋮
data cleaning
removal or correction of records
explainable model in "if-then" form
logical consistency checks on rule set
Footnote 2 on pg. 37
no
undesired patterns ("if-then form") found/confirmed?
yes
identifying the data on which the rules fire "most"
Section 1.3.2
Legend:
manual step
automated / standard procedures
output artefact
annotation/remark
artefact/result creation
procedural next step

Figure 1.3: Underlying XAI pipeline of the framework (adapted from Rass et al. [1])

1.3.1 Finding and Filtering Rules to Explain the Data

The phrase "finding rules" may be misleading, as the tool does not actively search for them, but rather generates every potential rule up to a predefined combination threshold of "if-parts". For instance, a threshold value of three would still generate the rule "`If` A `AND` B `AND` C `Then` D" but no rules with more complex conditions. After generating all possibilities, the rule's fit to the data is jointly evaluated. Details on how a rule is mathematically processed are shown in Section 2.

Each mathematical representation of a rule acts as a basis function. Applying basic linear regression models to these basis functions across all records allows for an initial analysis. However, due to stability problems, multicollinearity issues and the need to filter rules, different regression approaches, including standard linear regression, Ridge regression and Lasso regression have been evaluated. Of these, the most promising results are obtained with Lasso regression, since it promotes sparsity, meaning it selects a small, yet meaningful, subset of rules. Additionally, as claimed in [22], the restriction on the optimization method to supply at least as many data points as basis functions is relaxed when choosing Lasso, offering greater flexibility in rule selection. The optimization details are discussed in Section 4.

If requested by the user, a computationally expensive statistical test is applied to the rules. This process, detailed in Section 5, tests the null hypothesis that the regression coefficient of a rule is zero, implying the rule has no effect. If the null hypothesis cannot be rejected, the rule may be considered irrelevant. However, as we cannot establish a strict lower bound on the power of the test, this serves only as a heuristic guideline. Notably, this limitation does not affect rules which reject the null hypothesis — these indeed likely play an essential role in explaining the data.

1.3.2 Analysis of the Results

Empirical investigations have shown that manually sorting rules by their normalized coefficients, reviewing them and checking their significance is an effective technique for sufficiently small datasets, as demonstrated in Section 6.1. However, as the number of the columns in the CSV-file increases, making the rule sets more complex, it may be more practical to apply automated techniques for examining the resulting statistically significant rules with a stringent threshold on the significance level α. Existing approaches such as model-based diagnosis [23] or conflict set computation [24], also highlighted in [1], could be viable but may have to be adapted, since the application of fuzzy rules does not permit a crisp (Boolean) interpretation of the natural language rules; they should instead be viewed as correlations [25].

Furthermore, the influence of a rule on a specific data point may be expressed in terms of relative contributions from its coefficients as highlighted in [1]. Specifically, one can compute the magnitude (absolute value) of the weighted basis function for a given record i relative to the sum of all weighted basis function magnitudes of a record as it is shown in Equation (1.1) (adapted from [1]). Therefore, the entry in the i-th row and the j-th column of the matrix $\mathbf{C} = (c_{ij}) \in \mathbb{R}^{n\times m}$ provides a heuristic for determining the extent to which the j-th rule contributes to the i-th record's value of the target variable.

$$c_{ij} = \frac{|\beta_j \cdot b_j(\mathbf{x}_{i,\cdot})|}{\sum_{k=1}^{m} |\beta_k \cdot b_k(\mathbf{x}_{i,\cdot})|} \in [0,1] \tag{1.1}$$

1.4 Examples of Biased Datasets

To ensure the functionality of the project, both synthetic and popular real-world data will be analyzed in Section 6.1. Let us briefly look over the datasets, from which we will try to extract the generating rules. Note that the first two datasets (sections 1.4.1 and 1.4.2) have been designed and generated in [1].

1.4.1 Simple Noise Dataset

This dataset is created to obtain a simple robustness verification of the rule extraction process. It comprises 3000 records with one target variable, “target_generator”, uniformly distributed in the range $[0, 5]$. Additionally, four random fields, “RANDOM1” to “RANDOM4”, are drawn independently from a uniform distribution ranging over the interval $[-3, 2]$, introducing noise with a non-zero mean to test the project's ability to distinguish meaningful patterns from pure uncorrelated noise. The final target variable, “target”, is yielded by the field “target_generator” without additional transformations, keeping the experiment as simple as possible. Hence, this dataset assesses whether the rule extraction process correctly identifies dependencies in the fields while disregarding random noise.

1.4.2 Biased Salaries Dataset

The biased salaries dataset [26] is designed to evaluate how well the methodology uncovers systematic and implicit biases in realistic employment data. It comprises 3000 records with the attributes years of experience, hiring manager, job position, university reputation, Grade Point Average (GPA) and gender. In the dataset, gender plays a vital role in the construction of the salary. Specifically, female employees have a reduced likelihood of being assigned to high-paying managerial roles and are on average assigned less experience. Additionally, one hiring manager exhibits preferential treatment toward non-binary employees. Base salaries are determined by a weighted linear combination of experience, university reputation, GPA, job position, additive Gaussian noise with variance $\sigma^2 = 1500\,\text{USD}$ and managerial influence, with biases deliberately embedded to test the method's ability to detect unfair patterns. Hence, there is no explicit rule in the data generation process claiming that female candidates have a lower salary, but it should still be detected by the software, as this is implicitly caused.

1.4.3 Boston Housing Dataset

To also examine the results on a non-synthetic dataset, the Boston housing dataset [27] seems to be a promising candidate, as it is a widely studied real-world dataset that includes socio-economic and environmental factors as well as housing prices [28, 29, 30]. It consists of 506 records with fields such as crime rate, proximity to employment centers, nitric oxide concentration and the proportion of lower-status individuals in the neighborhood. The target variable represents the median house price in thousands of dollars. While the original intention of the dataset was to analyze how air quality correlates with housing prices, as highlighted by the title of the original paper by Harrison et al. [27], prior analysis has revealed additional insights. Multiple research projects [29, 31] have shown correlations between socio-economic attributes and housing prices, making it a suitable benchmark for evaluating the method's capability to detect potential real-world biases.

Furthermore, racial demographics and neighborhood status indicators have been linked to systematic disparities in property values. Racial segregation is a well-documented phenomenon in cities of the U.S. [32]. Surprisingly, the analysis in [30] suggests a very slight bias in favor of areas with higher deviations of black residents from the average after controlling for confounders such as crime rates. In Section 6.1.3, we will examine if our tool supports this finding.

2 Rule Generation and Primitive Regression

The approach is to generate all possible if-then rules subject to a user-defined constraint on their complexity — by limiting the maximum number of combined conditions — since without such restrictions one could construct arbitrarily complex formulas with n variables. The generated rules exclusively use the `AND` operation in the if-part, which, as we will discover later in this Section, does not negatively impact the expressiveness. After the said rule generation, filtering, (re-)weighting and statistical tests are applied to refine the ruleset, retaining only those rules with meaningful relevance in the data. Figure 2.4 illustrates the entire process described in the remaining chapter, also emphasizing the combinatorial explosion when the system parameter κ for the maximum number of combined antecedents, "Antecedent-Combinations" (Parameter 4 in Figure 7.12), is increased.

Initially, the columns of a CSV-file are classified into numerical columns, categorical columns and a single target column. The primary distinction between numerical and categorical fields lies in their membership treatment: Categorical fields exhibit Boolean membership, whereas numerical ones involve a degree of membership since a fuzzy-based approach is implemented (cf., Section 2.1).

As shown in Figure 2.4, all possible rules with certain criteria are generated by default. A rule consists of two components:

- The *antecedents* represent the if-part of the rule. In some publications, the entire if-part is referred to as one "big" antecedent — we do not use this convention here. Instead, since in this work we only apply `AND` operations between conditions, we call each `AND`-ed part "antecedent" and say that a rule has multiple antecedents, which is another commonly accepted practice.
- The *consequent* is the then-part of the rule, representing the predicted outcome, when the antecedents are satisfied.

When it comes to generating rules, an important aspect to consider is avoiding antecedent combinations that are impossible to fulfill. For instance, demanding a field to take on two distinct categorical values simultaneously is inherently contradictory. However, in the fuzzy case with numerical fields, such conditions can still yield valid rules, as further discussed in Section 2.1. This distinction is best illustrated with an example: A person may be both medium height and tall at the intersection of fuzzy categories, as natural language descriptions are inherently imprecise. Still, one cannot be both tall and short at the same time. Therefore, only overlapping fuzzy sets, such as "medium" and "tall", should be considered in rule generation to ensure meaningful interpretations. These rules covering overlapping fuzzy sets are not generated in the original publication [1] but are incorporated into the software during this work for completeness, hence the results in Section 6 likely deviate from [1].

At this stage, rule generation remains a straightforward combinatorial task, easily implemented using nested loops with basic validity checks preventing impossible rules. Since the runtime and memory complexity of this approach is concerning, future revisions of the method may consider employing more involved approaches. One possible technique for optimizing the rule generation might be an advanced utilization of constrained Orthogonal Arrays (OAs) [33]. While we will not explore the details of OAs in this work, one should note that they offer a systematic method to efficiently cover large input spaces with certain constraints, such as combinatorial (e.g., pairwise) coverage for all value combinations of all fields. Essentially, one may choose to have a careful trade-off between expressiveness in terms of different basis functions and computational complexity. In this work, we deliberately choose to sacrifice computation time for completeness. Thus, assuming a sufficiently large value for the system parameter κ, we have a proper theoretical basis for discovering all possible unwanted patterns.

Moreover, note that users may disable automatic rule generation entirely and instead supply their own rules in the whitelist.

To refine the set of rules used in the regression, users can specify a whitelist and a blacklist:

- Whitelisted rules are (almost) always included in the regression, even if their number of antecedents exceeds the system parameter κ. Nonetheless, some filtering methods proposed in Section 3 may remove a whitelisted rule before the regression process starts.

- Blacklisted rules are excluded from rule generation.

Note that it is crucial to ensure that the chosen ruleset provides a valid foundation for adequately explaining the data. If this requirement is not met, or the Lasso regression is not properly configured, the statistical "best fit" may yield misleading conclusions due to a lack of explanatory flexibility.

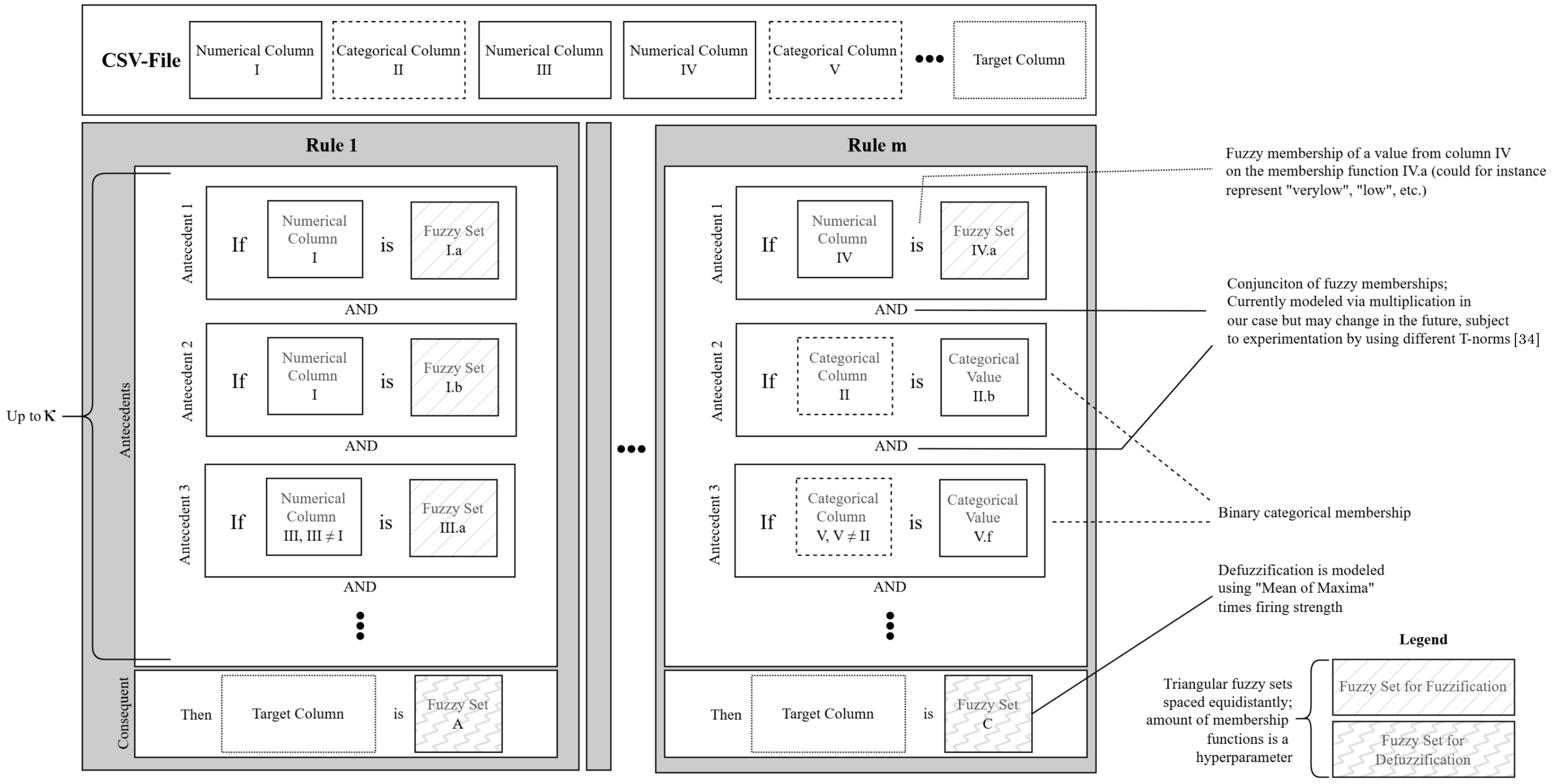

CSV-File
Numerical Column I
Categorical Column II
Numerical Column III
Numerical Column IV
Categorical Column V
Target Column
Rule 1
Rule m
Up to K
Antecedents
Antecedent 1
Antecedent 2
Antecedent 3
If Numerical Column I is Fuzzy Set I.a
AND
If Numerical Column I is Fuzzy Set I.b
AND
If Numerical Column III, III ≠ I is Fuzzy Set III.a
AND
Consequent
Then Target Column is Fuzzy Set A
If Numerical Column IV is Fuzzy Set IV.a
AND
If Categorical Column II is Categorical Value II.b
AND
If Categorical Column V, V ≠ II is Categorical Value V.f
AND
Then Target Column is Fuzzy Set C
Fuzzy membership of a value from column IV on the membership function IV.a (could for instance represent "verylow", "low", etc.)
Conjunciton of fuzzy memberships; Currently modeled via multiplication in our case but may change in the future, subject to experimentation by using different T-norms [34]
Binary categorical membership
Defuzzification is modeled using "Mean of Maxima" times firing strength
Legend
Triangular fuzzy sets spaced equidistantly; amount of membership functions is a hyperparameter
Fuzzy Set for Fuzzification
Fuzzy Set for Defuzzification

Figure 2.4: Rule generation process visualized with annotations for evaluation

2.1 (Mamdani) Fuzzy Inference Systems

To fully understand how rules operate, we need to examine the process of transforming a natural language rule into a mathematical basis function suitable for the regression discussed in Section 4. This concrete implementation uses Mamdani FIS to express the basis functions, since they are well-established and straightforward to encode. Note that this Section builds on our previous work [35], where more detailed descriptions can be found.

The foundation of FIS lies in *fuzzy sets*, proposed in 1965 by L.A. Zadeh [36]. By the default mathematical set membership we refer to binary membership (an element is either in a set or not), but fuzzy sets have an accompanying Membership Function (MF), which expresses the degree of membership of a domain element ranging from 0 to 1. For a fuzzy set X over $\mathbb{R}$, the MF $\mu_X : \mathbb{R} \to [0,1]$ could be defined as $\mu_X(x) = \min(x^2, 1)$. Of course, this exemplary μ_X was merely chosen for demonstration purposes because it matches typical domains and the co-domain $[0,1]$. Note that such a MF can be arbitrarily complicated and serves as a generalization of Boolean membership.

A degree of membership is useful because natural language is inherently imprecise (e.g., the meaning of the word “small” varies by context) and real-world problems often involve incomplete information [37, 38].

Moreover, we can apply logical operations to fuzzy sets, which are predominantly utilized for computing the rule's antecedent. Consider two values $y, z \in \mathbb{R}$ from the supplied CSV-file and let Y and Z be the corresponding fuzzy sets of the antecedent (e.g., “low”). Moreover, the values y and z are fuzzified with the respective MF μ_Y and μ_Z. One can express the most basic logical operations as follows:

- **NOT** $\mu_{\neg Y}$: As μ_Y takes values in $[0,1]$, a common way to define negation is simply $1-\mu_Y(y)$. Nonetheless, as highlighted by Dhiman et al. [39], even for this simple operation multiple techniques exist. For instance, the Yager complement ($\mu_{\neg Y} = \big(1 - (\mu_Y(y))^{\delta}\big)^{\frac{1}{\delta}}$) [40] is a flexible extension of the above notion of fuzzy negation via a hyperparameter $\delta \in (0,1]$.

- **OR** $\mu_{Y \cup Z}$: One commonly takes the maximum of the membership degrees: $\max(\mu_Y(y), \mu_Z(z))$. Nonetheless, there are also multiple ways to approach this operation. For instance, a common alternative is using the so-called algebraic sum approach $\mu_{Y \cup Z} = \mu_Y(y) + \mu_Z(z) - \mu_Y(y) \cdot \mu_Z(z)$ [41], which is similar to the probabilistic sum for the union of independent events.

- **AND** $\mu_{Y \cap Z}$: Two widespread ways to encode this are $\min(\mu_Y(y), \mu_Z(z))$ and $\mu_Y(y) \cdot \mu_Z(z)$. These operations can be viewed as special cases of the Frank t-norms [42], a parameterized family of fuzzy conjunctions that generalize the intersection operation by bridging between different t-norms. K. Wang [43] claims that the latter one (the product) is in many cases more desirable because of better smoothness conditions (due to differentiability).

To encode the if-then rules, we do not use implications, contrary to the way one would typically encode rules in traditional Boolean logic. Instead, we use FIS, which are regularly used to encode fuzzy rules [44]. The two most established methods are Mamdani [45] and TSK [46], whose process is outlined in Figure 2.5. The term *linguistic variables* in the image describes that there is an intermediate representation in terms of separate fuzzy sets which is also inherently interpretable. In the last step of the inference, the Mamdani system transforms the intermediate representation to a crisp output value, while the TSK system applies a weighted sum aggregation based on how well the corresponding antecedents match. Without going into further detail regarding the TSK systems, note that there are many different degrees of freedom in defining those systems. The Mamdani systems are simpler to construct as they do not require choosing polynomials for fuzzy rule outputs, hence the design choice is taken in favor of them.

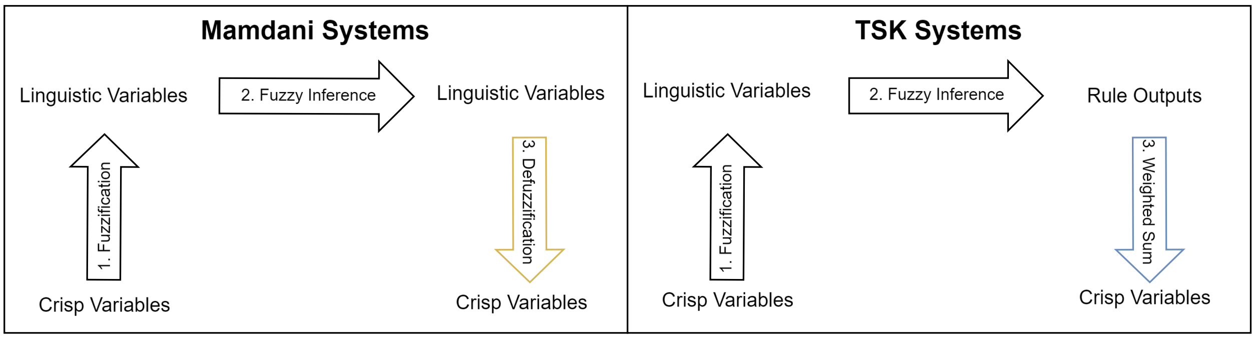


Figure 2.5: A high-level comparison between Mamdani and TSK systems (taken from [35])

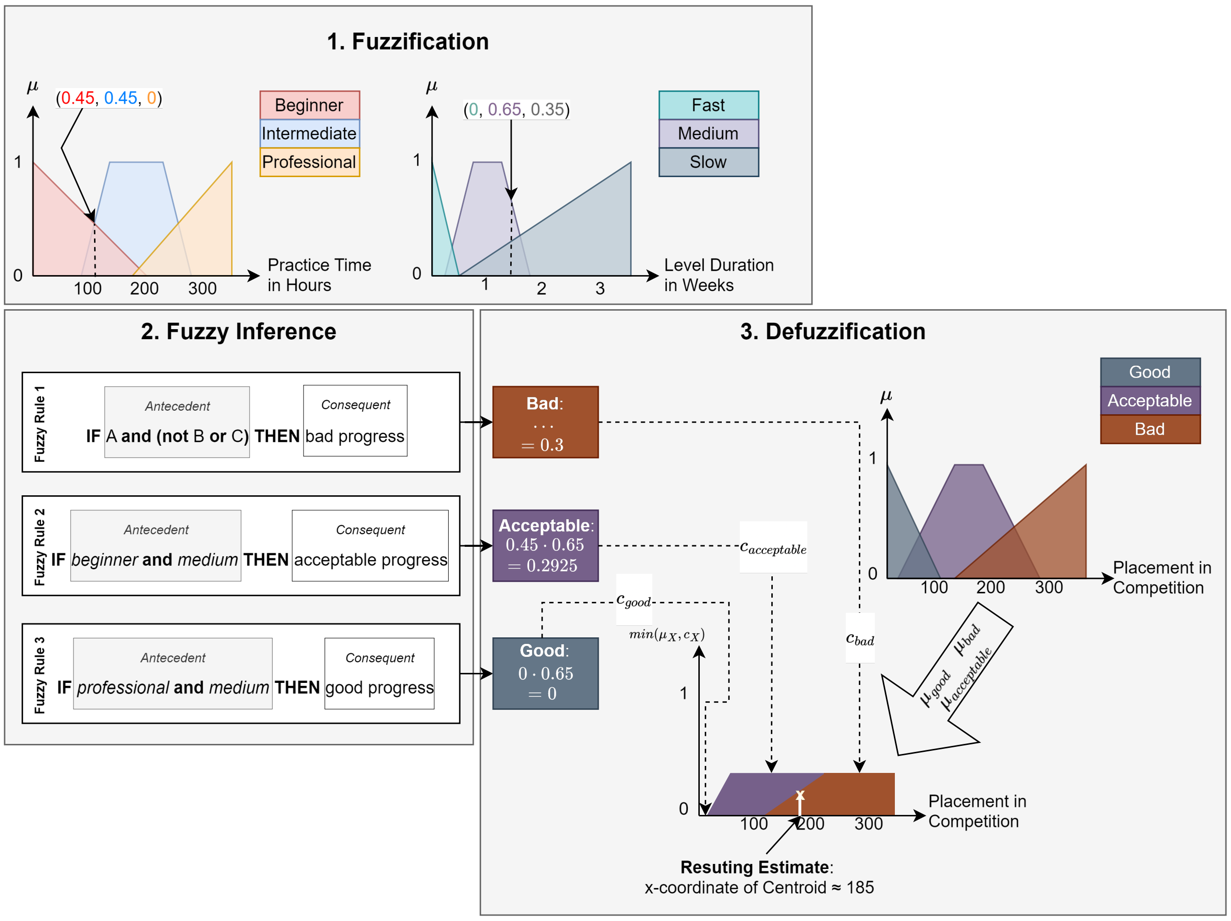


Figure 2.6: Exemplary Mamdani FIS employing COG [47] defuzzification (taken from [35])

To generally illustrate Mamdani FIS, a small exemplary inference system that models the estimate for a placement in a (hacking) competition is depicted in Figure 2.6. First, for the fuzzification step, one can observe that there are three manually defined fuzzy sets for each field. In this example, they are simple geometric shapes but, of course, one may choose more sophisticated MFs. The software, however, also only uses equidistantly spaced triangles for simplicity in the fuzzification and in the later defuzzification stage, as can be observed in Figure 2.7. Users can choose if 3, 5, 6 or 7 triangles should be used — separately for fuzzification and the defuzzification — with the respective fuzzy sets listed in Table 2.2. The arrows in the fuzzification section of Figure 2.6 indicate one concrete record, where the "practice time" is roughly 110 hours and the "level duration" is 1.5 weeks. One can simply read off the highlighted membership values of the six fuzzy sets.

While the described software implementation uses simple triangular MFs as a basic prototype, alternative non-linear functions, such as Gaussian or trapezoidal, could easily be incorporated to capture more nuanced or complex relationships within the data, if needed. Nonetheless, we will see in Section 6.1 that this simple implementation suffices.

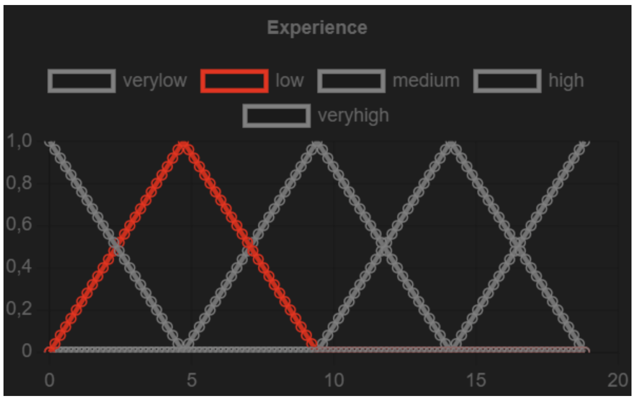


Figure 2.7: Fuzzy sets of the field "Experience" from the biased salaries dataset (Section 1.4.2)

Table 2.2: Names of the fuzzy sets for different granularities used in (de-)fuzzification

Number of sets	Respective names of fuzzy sets
3	low, medium, high
5	verylow, low, medium, high, veryhigh
6	verylow, low, mediumlow, mediumhigh, high, veryhigh
7	verylow, low, mediumlow, medium, mediumhigh, high, veryhigh

Moreover, Figure 2.6 compactly sketches the Mamdani fuzzy inference stage, in which for every consequent at most one natural language rule can be chosen to be evaluated on the fuzzy sets from the fuzzification. Since, in the general case, the if-part can be arbitrarily complex and does not only consist of `AND` operations, the entire if-part is referred to as the antecedent in the image. Our software applies vast simplification assumptions to this step:

- Only one rule is used for fuzzy inference. Hence, there is a FIS for every rule, which we later regress over.
- No logical connectives are allowed except for `AND`. The logical connective `OR` follows from the addition of two rules (FIS) in the regression. For the logical connective `NOT`, the most formally acceptable way to encode this would be using complement MFs ($1 - \text{membership}_A(x)$). In future versions, one can consider adding such rules to the regression process. Nonetheless, the results in Section 6.1 are already promising without this adaptation, since the regression can, for instance, attempt to express negated rules by both upscaling all other consequents with the same antecedents and by using a negative coefficient for the negated rule.

As regards the last step, the defuzzification, Figure 2.6 employs the well-known Center of Gravity (COG) [47] approach. To understand this method, one must first recognize that fuzzy sets for defuzzification also have corresponding MFs. In the shown example, these functions define the possible placement categories: “bad”, “acceptable” and “good”. Examining Figure 2.6 closely reveals the link between fuzzy inference and these defuzzification sets: Fuzzy inference determines the set membership of the respective linguistic categories, denoted as c_X for the fuzzy set X. Thus, one may see c_X as the “activation” of the rule with the linguistic output X. Subsequently, the fuzzy sets for defuzzification assist in converting this intermediate representation into a numerical prediction — i.e., a placement in the exemplary context.

To obtain the crisp output, in Figure 2.6 the computation $\min(\mu_X(x), c_X)$ is applied for each output fuzzy set X, where μ represents the defuzzification MF of X. Essentially, this limits the defuzzification function's mass according to the value of the consequent c_X. Finally, since COG is employed in the image, the centroid of the resulting shape is computed, and its position within the domain determines the final estimate.

In the software implementation, the last step of the defuzzification process is slightly different: For simplicity, the Mean of Maxima (MOM) [47] method is used instead of COG. This approach selects the set of domain values that attain the maximum of the capped defuzzification function values and calculates their arithmetic mean, which is identical to Equation (2.4) when skipping the multiplication with c_X.

However, with this method the fact that our FIS only contain one rule, meaning that at most one of the equidistantly spaced defuzzification MFs (triangles, denoted μ_X in Equation 2.2) remains nonzero in each FIS is problematic. This method only works for the outermost (non-symmetric) right-angled triangles, meaning the triangles that are formed by the abscissa and the outermost linear functions that are connecting the horizontal axis and the top-left (on the other side the top-right) corner seen in Figure 2.7 respectively. In case the chosen consequent does not correspond to one of these right-angled triangles, and its value is strictly greater than zero, the result will always be the center of the chosen triangle, regardless of the exact consequent and antecedent values. This occurs since removing the tip of a symmetric triangle and applying MOM consistently yields its center, independent of the capping level. Hence, we simply scale the rule's firing strength with the MOM, capturing the notion of multiplying with the respective activation:

$$\mathrm{defuz}(x) = \min(\mu_X(x), c_X) \tag{2.2}$$

$$\mathbf{m} = \arg\max_x \big(\mathrm{defuz}(x)\big) \tag{2.3}$$

$$\mathrm{result} = c_X \cdot \underbrace{\frac{\sum_i m_i}{\dim(\mathbf{m})}}_{\text{MOM}} \tag{2.4}$$

Here, $\mathbf{m} = (m_i)$ is the vector of all domain values where $\mathrm{defuz}(x)$ attains its maximum, and $\dim(\mathbf{m})$ denotes the number of elements in $\mathbf{m}$. The formulation in Equation (2.4) ensures that the output is always appropriately weighted by the rule's firing strength while still maintaining the simplicity of the MOM approach.

2.2 (Issues with) Unregularized Regression

The computations in the FIS and Equation (2.4) transform each rule into a *basis function*, effectively creating new fields for the regression while the original inputs are not used to regress over. In other words, we have performed a feature transformation using fuzzy rules expressed in natural language. Therefore, we can leverage these functions to perform a weighted linear combination of the fuzzy rules that survive the filtering from Section 3 as shown in Equation (2.5), where $\hat{y}_i$ is the predicted target variable for the i-th record and $\hat{\boldsymbol{\beta}}$ is the vector containing the estimated coefficients.

$$\hat{y}_i = \hat{\beta}_1 \, b_1(\mathbf{x}_{i,.}) + \hat{\beta}_2 \, b_2(\mathbf{x}_{i,.}) + \cdots + \hat{\beta}_m \, b_m(\mathbf{x}_{i,.}), \quad i = 1, \ldots, n \tag{2.5}$$

Applying the standard Ordinary Least Squares (OLS) [48] procedure for determining $\hat{\boldsymbol{\beta}}$, we first construct the *design matrix*, where each row represents a record and each column corresponds to the basis function of a specific rule. Let us denote this matrix, also visualized in Equation (2.6), as $\mathbf{G}$ from now on and let $b_i : \mathbb{R}^k \to \mathbb{R}$ be the basis function of the i-th rule, which always takes a full record $\mathbf{x}_{i,.}$, but only requires a subset of the entries from the vector to compute the antecedent activations.

$$\mathbf{G} = \begin{bmatrix} b_1(\mathbf{x}_{1,.}) & b_2(\mathbf{x}_{1,.}) & \ldots & b_m(\mathbf{x}_{1,.}) \\ b_1(\mathbf{x}_{2,.}) & b_2(\mathbf{x}_{2,.}) & \ldots & b_m(\mathbf{x}_{2,.}) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(\mathbf{x}_{n,.}) & b_2(\mathbf{x}_{n,.}) & \ldots & b_m(\mathbf{x}_{n,.}) \end{bmatrix} \tag{2.6}$$

Moreover, note that $b_1(\mathbf{x}) = 1 \ \forall \mathbf{x} \in \mathbb{R}^k$, if the default option to use the intercept is active. Otherwise, b_1 is the first generated rule which has not been removed by filtering.

The values within $\mathbf{G}$ are inherently zero-centered, since the target variable is standardized immediately after the CSV-file is imported. The option to use an intercept may therefore be carefully considered as the basis functions try to approximate the target variable and the z-score normalization removed the effort of predicting the mean, which is neutralized. Furthermore, the values in $\mathbf{G}$ therefore also have a useful numerical interpretation for debuggability, as every strong deviation from zero reflects a strong "opinion" of a rule.

However, the standard OLS formulation for linear regression requires the design matrix to be invertible. A simple example illustrates why this condition cannot hold if too many fuzzy sets are chosen for defuzzification:

- `If` A `is` high `Then` B `is` high
- `If` A `is` high `Then` B `is` veryhigh

Both of these rules are generated automatically by the procedure shown in Figure 2.4. However, they are effectively multiples of each other due to Equations (2.2) to (2.4), since the corresponding MFs, and thus the respective maxima, are merely shifted versions of one another.

As more precisely discussed in Section 4, Lasso regression is better equipped than OLS estimation to handle such multicollinearities and the filtering mechanisms introduced in Section 3 significantly mitigate this issue. Nonetheless, it is generally advisable to limit the number of fuzzy sets, as this hyperparameter presents a trade-off: Increasing the number of fuzzy sets enables finer-grained predictions and a closer fit to the data, but it also increases the risk of overfitting, because more predictors are available. Since the OLS estimator is prone to overfitting — especially as the number of estimators increases [49] — careful selection of the number of fuzzy sets may be a crucial factor.

3 Filtering Mechanisms

To remove unpromising options and to thereby obtain an improved basis for the later-applied regression, we examine the rule filtering mechanisms that complement the existing white- and blacklist approaches. It is worth noting that the filtering methods described in the following sections fully align with the techniques presented in [1] but are described in greater detail in this thesis. Moreover, they can be extended with additional techniques and certain simplifying assumptions may improve the results of the solutions found by the current implementation. In addition, users may disable any filtering mechanism using corresponding flags for the Application Programming Interface (API) or the web interface.

The omitted rules can be found in the “Warnings” section of the web interface or the API output. Since this output is hierarchical, it allows users to see which constraints caused the removal of a given rule, providing complete traceability for debugging purposes.

3.1 Outlier Filtering

This first filtering method is the only one that operates directly on the records and not on the rules. Outliers are known to disproportionately influence the result of linear regression models [50].

Therefore, it is strongly recommended to:

- Apply outlier removal on all columns of the CSV-file before using the method described in this thesis.
- Alternatively, use the built-in outlier-filtering options in the application, as shown in Figure 3.8, where a lower and an upper bound can be specified.
- Otherwise, one can also choose to set an Interquartile Range (IQR) multiplier value u in the application, where all records with values outside the interval $[q_{25} - u \cdot \mathrm{IQR}, q_{75} + u \cdot \mathrm{IQR}]$ will be dropped. The variable q_o denotes the o-th quantile (in percent) of the data, sorted by the respective field.

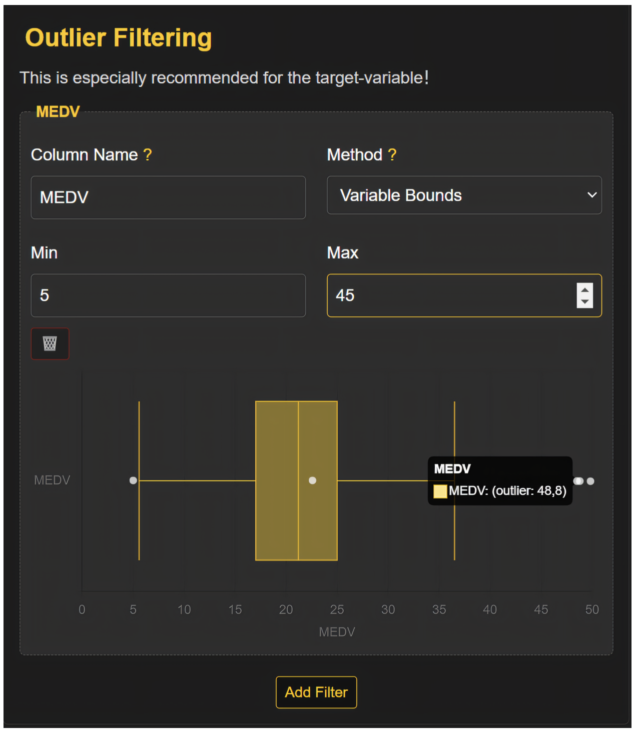
3.2 Priority Computation and Filtering via Rule Mining Metrics

Rule mining has been thoroughly researched in the AI and Computer Science literature as shown in a survey by Solanki et al. [51]. In a moment, we will employ two well-known metrics to compute a priority score for each rule. Before introducing them formally, let us first discretize the data, solely for priority computation. For this discretization, the fuzzy sets are used to assign each record to the class (formerly: fuzzy set) with the highest membership. In the case of ties, the discretization of the record is duplicated, and both possibilities are represented. The categorical fields are not modified, as they already are partitioned into discrete classes. Furthermore, note that the discretization of $\mathbf{D}$ is deleted from memory once the priority scores have been computed.

The priority score of each rule r is a weighted linear combination of the following metrics, where the user supplies the weight vector $\mathbf{w}$ for the linear combination, as presented in Equation (3.7).

$$\mathrm{priority}(r) = w_1 \cdot \mathrm{support}(r) + w_2 \cdot \mathrm{leverage}(r) + w_3 \cdot \frac{1}{|\mathrm{antecedents}(r)|} + w_4 \cdot \mathrm{whitelist}(r) \tag{3.7}$$

- The *support* [52] measures how often a rule holds true. It is defined as the percentage of discretized records where all of the rule’s antecedents align with their discretization and the consequent matches the discretized target class of a record.


Outlier Filtering

This is especially recommended for the target-variable!

MEDV

Column Name ?: MEDV

Method ?: Variable Bounds

Min: 5

Max: 45

MEDV

MEDV: (outlier: 48,8)

Figure 3.8: Outlier filtering in the web application

- The *leverage* [53] quantifies how much a rule deviates from statistical independence. It measures the difference between the amount of observed co-occurrences of antecedents (A) and consequents (C), denoted as support($A \to C$), and their expected number of co-occurrences if they were independent. This allows the computation to be expressed similarly to [54]:

$$\text{leverage}(A \to C) = \text{support}(A \to C) - \text{support}(A) \cdot \text{support}(C) \tag{3.8}$$

  Leverage values above zero indicate that a rule's antecedents and consequents co-occur more frequently than expected by chance; the further the value is above zero, the stronger the indication of a potentially interesting pattern according to this heuristic.

- The inverse of the number of antecedents is used as a heuristic for simplicity. This summand is added as in ML simpler rules are often desired, so employing a complexity-penalty might be beneficial.

- The *whitelist flag* allows for prioritized retention of user-specified rules. If a rule is included in the whitelist, the respective value is set to one; otherwise, it remains zero.

A user can define a minimum priority value (Parameter 18 in Figure 7.12) after initially observing the distribution of the rule priorities. This serves as a threshold for further processing the rules, i.e., rules with a priority below this threshold will not be included in the subsequent analysis. Note that the priority may take on a negative value, since one can find rules such that leverage(r) < 0.

Once the priority value for each rule has been determined, the columns (basis functions) of $\mathbf{G}$ that survive the filtering are sorted in descending priority order.

3.3 Threshold-Based Filtering

Let $\mathbf{g}_{\cdot,a}, \mathbf{g}_{\cdot,b}$ be two arbitrary but fixed columns and similarly let $\mathbf{g}_{e,\cdot}, \mathbf{g}_{f,\cdot}$ be two arbitrary but fixed rows of the design matrix $\mathbf{G}$ (from Section 2.2). If the Manhattan norm between two columns or rows is below the respective user-specified threshold, denoted $\psi_c \in \mathbb{R}$ for columns and $\psi_r \in \mathbb{R}$ for rows, i.e.,

$$\|\mathbf{g}_{\cdot,a} - \mathbf{g}_{\cdot,b}\|_1 < \psi_c \quad \text{or} \quad \|\mathbf{g}_{e,\cdot} - \mathbf{g}_{f,\cdot}\|_1 < \psi_r,$$

then a row or a column is removed. In the case of row-based (record-based) filtering, the row appearing first is always retained. For filtering column-wise, the rule with the lower priority will be filtered out. Nonetheless, the removed rules are explicitly printed as “secondary rules” for their primary counterpart, indicating an alternative natural language interpretation (rule) of identical basis functions for the dataset. In the case of ties, the rule generated later will be removed.

Note that both ψ_c and ψ_r can simply be set to any negative value to fully disable the respective filter, as the Manhattan norm cannot output a negative number.

3.4 Sparsity From Lasso Regularization

Since the Lasso objective is applied in the regression step (see Section 4), sparsity is inherently encouraged, commonly leading many rule coefficients to be precisely zero. A rule which has a coefficient of zero has no meaningful effect in the regression solution and therefore may safely be ignored.

Thus, after the regression step, the software discards all rule coefficients with a magnitude smaller than a user-specified parameter (Parameter 31 in Figure 7.12) — by default set to 10^{-8}, a value chosen as a practical trade-off between computational efficiency and completeness — and logs these discarded rules in a warning list included in the output. This saves processing power, as the removed rules are not considered during the computationally expensive statistical testing described in Section 5.

3.5 Statistically Insignificant Rules

Additionally, users can opt to retain only statistically significant rules, which can reject the null hypothesis that the coefficient of the i-th rule is zero ($H_0 : \beta_i = 0$) with a user-specified significance level α. Since computing p-values can be time-intensive, it is recommended to fine-tune all other filtering options beforehand. Moreover, this approach has further limitations, outlined in Section 5.2.

After this step, one may activate the configuration option to re-compute the Lasso coefficients once all statistically insignificant rules and all other rules affected by the preceding filters have been removed. It should be noted that this option is not enabled by default, and both outputs (with and without using this flag) should not explain the data incorrectly but might yield slightly different results, very likely in the magnitudes of the coefficients.

4 Rule Selection via Lasso Optimization

The Lasso estimator, first proposed by R. Tibshirani [55], extends the notion of the popular OLS estimator and applies a penalty to nonzero coefficients based on their magnitudes. Concretely, the ℓ_1 penalty ($\lambda\|\boldsymbol{\beta}\|_1$) in Equation (4.9) promotes sparsity and $\lambda \in [0, \infty)$ controls the strength of this regularization. This means that many coefficients are forced towards zero, which is commonly used to counter overfitting [56, 57]. In the context of this application, it means that numerous rules are eliminated.

Unlike the Gauss-Markov theorem [58] for OLS, no extensive guarantees apply here and Lasso purposefully introduces a bias [57]. This is a suitable decision since the goal of the method is to identify the few most important rules and to weigh them appropriately.

Recall the design-matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$ from Section 2.2, where n is the number of records and m represents the number of rules, including the intercept. The Lasso estimator is commonly used in its Lagrangian form, as introduced by R. Tibshirani [55] and presented in Equation (4.9).

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^m} \{\|\mathbf{y} - \mathbf{G}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1\} \tag{4.9}$$

In this formulation, the vector $\mathbf{y} \in \mathbb{R}^n$ contains the target values and $\lambda > 0$ is a user-selectable hyperparameter to balance between the fidelity to the data and the sparsity of the solution.

As highlighted by van de Geer et al. [22], even when the number of basis functions m exceeds the observations n, the Lasso estimator remains well-behaved under suitable conditions. Additionally, the authors show that the Karush-Kuhn-Tucker (KKT) conditions [59] can be inverted to yield a de-sparsified Lasso estimator that is asymptotically normal (see Section 5 for details). This makes Lasso particularly appealing for the present application, since it does not only provide a reduced, weighted ruleset but also enables the application of appropriate statistical tests [22].

To first obtain a Lasso solution, we use the convexity property of Equation (4.9) [60], which is given as both summands are convex. Thus, if the first derivative (wherever it exists) is decreasing to zero, we can assume to move towards one of the optima. We will show in Section 4.2 that strict convexity in the use case of this application usually does not hold because both summands in the said Equation (4.9) are not (necessarily) strictly convex. Hence, there are most likely multiple "optimal" solutions according to the optimization criterion stated in Equation (4.9).

4.1 Regression Solution via Coordinate Descent

The regression problem of early versions of the software, which use the OLS estimator, is easier to solve, as a closed-form solution exists. Matrix decompositions, such as the fast Cholesky decomposition or the more stable QR decomposition, allow for obtaining a quick solution to the said problem compared to Lasso. However, this estimator is not entirely suitable to this large-scale regression setting because of multiple issues, which are outlined in Section 2.2.

Note that the Lasso estimator uses an optimization criterion containing the ℓ_1 penalty and thus the absolute value function is applied, as shown in Equation (4.9). Since the absolute value function is not differentiable at zero, there is no analytical solution for obtaining the Lasso estimate. One can instead derive an iterative "coordinate descent" approach [61] to find a suitable solution by cyclically updating one concrete coefficient $\hat{\beta}_i$ at a time while keeping the others fixed. Recall that the columns of $\mathbf{G}$ are sorted by descending priority value, the computation of which is outlined in Section 3.2.

Hence, higher-prioritized rules are optimized first. Later, lower-prioritized rules, like multicollinear basis functions, will more likely be set to zero in every iteration. This property can be observed in Equation (4.12), which we will derive shortly. Notably, Lasso can cope (comparably) well with multicollinearity, whereas OLS often fails to yield a solution.

Inspired by Friedman et al. [62], we will now derive the update rules for the coordinate descent algorithm. When updating the j-th coefficient ($\hat{\beta}_j$), we want to isolate its effect on the loss. It is straightforward to separate the prediction into two parts

$$\mathbf{G}\hat{\boldsymbol{\beta}} = \mathbf{g}_{\cdot,j}\hat{\beta}_j + \sum_{\substack{k=1\\k\neq j}}^{m} \mathbf{g}_{\cdot,k}\hat{\beta}_k \tag{4.10}$$

where $\mathbf{g}_{\cdot,j}$ denotes the j-th column of $\mathbf{G}$, representing one concrete rule (basis function).

Next, let us define the *partial residual* (or "partial fit") as the residual vector obtained by excluding the contribution of the j-th predictor, which we store in the j-th row of the matrix $\mathbf{R} \in \mathbb{R}^{m\times n}$, denoted as $\mathbf{r}_{j,\cdot}$:

$$\mathbf{r}_{j,\cdot} = (\mathbf{y} - \sum_{\substack{k=1\\k\neq j}}^{m} \mathbf{g}_{\cdot,k}\hat{\beta}_k)^\top \tag{4.11}$$

By subtracting the contribution of all other predictors, $\mathbf{r}_{j,\cdot}$ isolates the part of $\mathbf{y}$ that $\hat{\beta}_j$ can explain by scaling $\mathbf{g}_{\cdot,j}$. This reduction transforms the multivariate optimization into a simpler one-dimensional problem:

$$\hat{\beta}_j = \arg\min_{\beta_j\in\mathbb{R}} \left\{ \|\mathbf{r}_{j,\cdot}^\top - \mathbf{g}_{\cdot,j}\beta_j\|_2^2 + \lambda|\beta_j| \right\}. \tag{4.12}$$

To write out the update more explicitly, let us define two helper variables in Equation (4.13), where $p_j \in \mathbb{R}$ represents the correlation of the j-th predictor with the current residual and $z_j \in \mathbb{R}$ is the squared Euclidean norm of the j-th column in $\mathbf{G} = (g_{ij})$, which is later used for normalization.

$$p_j = \sum_{i=1}^{n} g_{ij} r_{ji} = \mathbf{r}_{j,\cdot}\mathbf{g}_{\cdot,j} \qquad z_j = \sum_{i=1}^{n} g_{ij}^2 \tag{4.13}$$

By multiplying out the quadratic term in the univariate optimization problem from Equation (4.12) and ignoring the term $\sum_{i=1}^{n}(r_{ji})^2$, which is constant with respect to β_j, the optimization problem becomes:

$$\hat{\beta}_j = \arg\min_{\beta_j\in\mathbb{R}} \left\{ z_j\beta_j^2 - 2p_j\beta_j + \lambda|\beta_j| \right\}. \tag{4.14}$$

From here, to solve the minimization task, one may apply the subdifferential for the minimization problem in Equation (4.14) and set it to zero. Since Lasso is a convex optimization problem, this yields a global minimum. A detailed proof is provided by Friedman et al. [63]. It is not outlined here as the length of the respective proof with the subdifferential would go beyond the scope of this thesis. Skipping ahead, the solution is obtained by using the *soft-thresholding operator* $S(a, \gamma) : (\mathbb{R} \times \mathbb{R}_+) \to \mathbb{R}$, described in Equation (4.15), where $\mathbb{R}_+$ denotes all non-negative real numbers. Following the approach of Friedman et al. [63], this operator provides a solution for the sub-differential, therefore enabling the computation of the optimal $\hat{\beta}_j$ at a given iteration, as stated in Equation (4.16).

$$S(a, \gamma) = \begin{cases} a - \gamma & \text{if } a > \gamma, \\ 0 & \text{if } |a| \leq \gamma, \\ a + \gamma & \text{if } a < -\gamma. \end{cases} \tag{4.15}$$

$$\hat{\beta}_j = \frac{1}{z_j} S(p_j, \lambda/2). \tag{4.16}$$

Apart from its formal proof, the invocation of the soft-thresholding operator in Equation (4.16) has an intuitive interpretation. Regularization is applied by shrinking the magnitude of p_j by $\frac{\lambda}{2}$ and sparsity is enforced by setting coefficients to precisely zero where $|p_j| < \frac{\lambda}{2}$ holds. This means that predictors with weak effects are eliminated, with the threshold depending on the size of the regularization hyperparameter λ, as indicated in [55]. A suitable value of λ balances prediction accuracy and sparsity. In the context of this application, empirical results (cf., Section 6.1) have shown that selecting λ within the interval $[0.01, 100]$ provides an effective compromise.

Algorithm 1 summarizes the results of our above derivations and represents the pseudocode of the implementation in the software, incorporating two crucial hyperparameters: The variable $t_{\max} \in \mathbb{N}$ sets the maximum number of iterations, while $\epsilon \in \mathbb{R}_+$ serves as a convergence threshold. Convergence is assumed when, at iteration $t \in \mathbb{N}$, all coefficients $\boldsymbol{\beta}^{(t)}$ are sufficiently close to those of the previous iteration $\boldsymbol{\beta}^{(t-1)}$. These hyperparameters significantly impact the solution, as described in more detail in the Section 4.2.

4.2 Ambiguity of Lasso Optima and "Best" Rule Sets

Upon examining Equation (4.9), one can determine that the first summand may not be strictly convex, depending on the choice of $\mathbf{G}$, and the second summand is never strictly convex:

- The quadratic loss ($\|\mathbf{y} - \mathbf{G}\boldsymbol{\beta}\|_2^2$) is strictly convex if only if the columns of $\mathbf{G}$ are linearly independent. After the filtering, some redundant basis functions commonly remain that can be expressed as a linear combination of the remaining basis functions. Thus, $\mathbf{G}$ is expected to only be positive semi-definite instead of being strictly positive definite. This means that a nonzero vector $\boldsymbol{\delta} \in \mathbb{R}^m$ exists such that $\mathbf{G}\boldsymbol{\delta} = 0$ holds, which can be used to find an alternative optimal solution [65].

- The ℓ_1-penalty ($\lambda\|\boldsymbol{\beta}\|_1$) is convex but not strictly convex. Its diamond-shaped level sets contain linear segments. This linearity violates the definition of strict convexity as we can choose a convex combination on such a level set and note that the strict inequality, which would be necessary for strict convexity, is only an equality in that case.

Prior research by R. Tibshirani [65] leads to the conclusion that the Lasso estimator is not unique for sufficiently high correlation between the predictors, causing uncountably many optimal solutions. Hence, the proposed method mostly has multiple possible optima, as many columns of $\mathbf{G}$ are likely

Algorithm 1 Lasso Coordinate Descent (inspired by [64])

```
    Input:
      Design matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$
      Response vector $\mathbf{y} \in \mathbb{R}^n$
      Regularization parameter $\lambda \in [0, \infty)$
      Maximum iterations $t_{\max} \in \mathbb{N}$
      Convergence tolerance $\epsilon \in \mathbb{R}_+$
    Output: Approximation of $\hat{\boldsymbol{\beta}}$ from Equation (4.9)
 1: $\boldsymbol{\beta}^{(0)} = 0$
 2: $t = 1$
 3: repeat
 4:     $\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)}$
 5:     for $j = 1$ to $m$ do
 6:         Compute the partial residual of the $j$-th predictor:
                $\mathbf{r}_{j,\cdot} = \Big(\mathbf{y} - \sum_{k=1}^{j-1} \mathbf{g}_{\cdot,k}\beta_k^{(t+1)} - \sum_{k=j+1}^{m} \mathbf{g}_{\cdot,k}\beta_k^{(t)}\Big)^\top$
 7:         Compute:
                $p_j = \mathbf{r}_{j,\cdot}\mathbf{g}_{\cdot,j} \qquad z_j = \sum_{i=1}^{n} g_{ij}^2$
 8:         Update coefficient:
                $\beta_j^{(t+1)} = \frac{1}{z_j} S(p_j, \lambda/2)$
 9:     end for
10:     $t = t + 1$
11: until $t \geq t_{\max}$ or $\max_j |\beta_j^{(t)} - \beta_j^{(t-1)}| < \epsilon$
12: Return: $\boldsymbol{\beta}^{(t)}$
```

correlated, which means one can choose from a subspace of possible values for $\boldsymbol{\delta}$ to obtain a continuum of solutions. Nonetheless, the software is deterministic, in the sense that there is no randomness introduced in any of the used algorithms. Still, it is worth acknowledging that even small changes, especially in the selected value for the regularization λ, the maximum amount of iterations $t_{\max}$ or the convergence threshold ϵ, can sometimes have large effects on the output coefficients and therefore the interpretation of the rules. An idea for vastly reducing this subspace of optimal solutions would be to include the rule priorities from Section 3.2 in the optimization criterion shown in Equation (4.9). However, this is left up to future exploration.

Reducing multicollinearity by filtering (cf., Section 3) reduces the correlation in the columns of $\mathbf{G}$ and thereby also inhibits the range of ambiguous optimal solutions. However, when applied too restrictively, this may remove important rules that would have matched the data generation process. Thus, we can observe a trade-off between ambiguity and completeness.

Similarly to the justification by our previous work [1], prior research in the realm of XAI also supports the notion that reasoning (and thus explanations) can be structured in multiple, equally valid ways [66]. A helpful analogy can be drawn from mathematics, where a proof can vary in length, strategy and level of abstraction, among other aspects. Fundamentally, one could derive results directly from axioms instead of common theorems, yielding a more intricate but equivalent explanation [1].

Furthermore, one must critically analyze the meaning of rules with negative coefficients, which most likely also occur in the output of the Lasso regression. We interpret this as if the rule's output negatively correlates with the data [1]. A different interpretation for this phenomenon is a negation of the consequent which strengthens all complement sets. For instance, if one uses three fuzzy sets ("low", "medium" and "high") for the defuzzification and the statistically significant rule "`If` targetgenerator `is` veryhigh `Then` target `is` low" appears with a negative coefficient, then one may claim that a very high value for the field "targetgenerator" causes the target variable to be anything but low, i.e., either medium or high. This also impacts the construction of all other rules, as the regression coefficients influence each other. Thus, if no rules with negative coefficients are allowed, many rules (especially the ones with a non-zero coefficient) will likely be assigned different coefficients.

4.2.1 Non-Negative Soft-Thresholding

If one wants to avoid negative coefficients for interpretability reasons, there are multiple ways to perform Lasso optimization while still forcing the coefficients to be non-negative, as shown by K. Hagiwara [67]. One simple method to perform this change is to update the soft-thresholding operator from Equation (4.15) to only yield non-negative outputs, as demonstrated in Equation (4.17).

$$S_{+}(a, \gamma) = \begin{cases} a - \gamma & \text{if } a > \gamma, \\ 0 & \text{else} \end{cases} \tag{4.17}$$

Let us from now on refer to this modified model as Non-Negative Soft-Thresholding (NNST). The software is adapted such that users can simply set the option "only allow positive coefficients" to enable NNST, as this was not originally included in [1]. For evaluating the tool's performance in Section 6.1, both implementations are considered, though the standard soft-thresholding with negative outputs will be referred to as the "default" model to maintain comparability to [1].

Note that the proposed NNST model does not formally align with the statistical test proposed in Section 5, since the hard constraint of setting negative coefficients to zero alters the KKT conditions, demanding a different derivation of the debiasing corrections. Therefore, the asymptotic normality assumed in the statistical test is not formally guaranteed. This means the results in Section 6.1 should be interpreted cautiously regarding the statistical test for the NNST model. Nonetheless, based on empirical evaluations one can discover that the dropoff in the respective performance metrics (R^2 and Mean Squared Error (MSE)) caused by the statistical filtering (cf., Section 3.5) on the default and the NNST model is nearly identical. Therefore, one may assume that the impact of this modification on the overall conclusions is likely minimal.

4.2.2 Recommendations for Lasso Hyperparameter-Selections

Based on extensive testing on the three datasets outlined in Section 1.4, we arrived at the following conclusions:

- The hyperparameter λ should be large enough to prevent overfitting by eliminating many basis functions, but small enough to avoid underfitting (such that, for instance, the MSE does not get too large). As outlined in Section 5, choosing λ as described at the start of Section 5 is a beneficial choice for the statistical test, but may not be ideal for balancing under- and overfitting.

- Interestingly, the empirical evaluation indicates that setting the convergence tolerance (ϵ) and the iteration limit ($t_{\max}$) such that the algorithm stops early leads to simpler rules being favored over more complex ones (cf., Section 6.1). For example, in the biased salaries dataset (Section 1.4.2), using $\epsilon = 10^{-1}$, instead of the more common and still numerically stable $\epsilon = 10^{-6}$, results in a simpler model while also guiding the optimization in the right direction due to convexity. Further investigation is certainly needed to formally prove that this is indeed a general phenomenon (instead of merely being a fortunate error pattern) and to quantify *how early* one should stop the optimization in the best case, though this is beyond the scope of this work.

5 Statistical Testing

Van de Geer et al. [22] (Theorem 2.2) demonstrate that inverting the KKT conditions [59] for Lasso to construct an asymptotically normal, de-sparsified (de-biased) Lasso estimator, yields the Equations (5.20) to (5.22). These hold under suitable restrictions [22], especially for the use case of this work, provided that λ in Algorithm 1 is chosen appropriately. In particular, the authors [22] prescribe that one should assign λ such that the inequality in (5.19) holds, where $f \in \mathbb{R}_+$ serves as an upper bound for the maximum diagonal element of $\mathbf{G}$:

$$\sigma_\varepsilon^2 = \frac{\|\mathbf{y} - \mathbf{G}\hat{\boldsymbol{\beta}}\|_2^2}{n - s} \tag{5.18}$$

$$\lambda \geq 2\, f\, \sigma_\varepsilon \sqrt{\frac{2\log(m)}{n}} \tag{5.19}$$

The number of nonzero coefficients $s \in \mathbb{N}$ in $\hat{\boldsymbol{\beta}}$ is used to estimate the degrees of freedom for the variance of the unobserved noise (errors) from the regression setting, denoted σ_ε^2 in Equation (5.18). While the variable σ_ε may be estimated via the residuals as shown in Equation (5.18), it should be noted that this simple procedure comes at the cost of typically under-estimating the true standard deviation of errors [68] which may be counteracted with more involved techniques like scaled Lasso in future implementations [69, 22]. For the practical use in the implementation, it suffices to know that the last entry in the "Warnings" section of the output always shows if λ was chosen sufficiently large for the statistical test, which is never violated in all evaluations performed in this document. Hence, we may indeed make use of the following statements from van de Geer et al. [22] in our application context:

$$\sqrt{n}(\hat{\mathbf{b}} - \boldsymbol{\beta}_{H0}) = \mathbf{w} + \boldsymbol{\xi} \tag{5.20}$$

$$\mathbf{w}|\mathbf{G} \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{\Theta}) \tag{5.21}$$

$$\|\boldsymbol{\xi}\|_\infty = o_{\mathbb{P}}(1) \tag{5.22}$$

In the above expressions, $\hat{\mathbf{b}} \in \mathbb{R}^m$ corresponds to the de-sparsified estimator (where the bias introduced by Lasso is counteracted), $\boldsymbol{\beta}_{H0} \in \mathbb{R}^m$ denotes the true parameter vector (typically specified under the null hypothesis, thus the subscript), $\mathbf{0}$ is a column vector of dimensionality m filled with 0 at all indices and $\boldsymbol{\Theta} \in \mathbb{R}^{m\times m}$ represents the precision matrix, which serves as the theoretical inverse of the covariance matrix of $\mathbf{G}$. For the error vector $\boldsymbol{\xi} \in \mathbb{R}^m$ it can be shown that the maximum value is of the order $o_{\mathbb{P}}(1)$, and it therefore vanishes asymptotically with an increasing number of records n. More precisely, the subscript $\mathbb{P}$ used by van de Geer et al. [22] in $o_{\mathbb{P}}(1)$ indicates that the convergence property is expressed in probability by [22], i.e., if one has a sequence of random variables $X_n = o_{\mathbb{P}}(1)$ this means that for every $\eta \in \mathbb{R}_+\backslash\{0\}$ the statement $\lim_{n\to\infty} \mathbb{P}(|X_n| > \eta) = 0$ holds [70].

Since directly inverting the sample covariance matrix is generally infeasible due to its high dimensionality and potential multicollinearity, regularized approaches, such as nodewise regression (cf., Section 5.1), are commonly used to obtain a numerically stable estimate of $\boldsymbol{\Theta}$ [71, 72]. The concrete meaning of $\mathbf{w} \in \mathbb{R}^m$ is not important for the following explanations. Let us restrict ourselves to note that $\mathbf{w}$ is asymptotically normal when the design matrix is given, thus the left-hand side of Equation (5.20) approximately follows this property.

By restricting the Equations (5.20) to (5.22) to the j-th coefficient of $\hat{\mathbf{b}}$ and $\boldsymbol{\beta}_{H0}$ and by neglecting $\boldsymbol{\xi}$, we can observe that the deviation of the de-biased estimator from the null hypothesis is indeed approximately normally distributed, as shown in Expression (5.23) in which $\boldsymbol{\Theta} = (\theta_{ij})$. Adjusting this formulation to obtain a test statistic for an appropriate z-test brings us to Equation (5.24).

$$\sqrt{n}\,\left(\hat{b}_j - \beta_{H0_j}\right) \sim \mathcal{N}(0, \sigma^2\,\theta_{jj}) \tag{5.23}$$

$$z_j = \frac{\hat{b}_j - \beta_{H0_j}}{\sqrt{\sigma^2\,\theta_{jj}/n}} \;\sim\; \mathcal{N}(0,1) \tag{5.24}$$

This result enables us to perform statistical inference. Concretely, it provides the basis for constructing confidence intervals and for conducting hypothesis tests on individual rules. In other words, it is the statistical grounding used for showing that potentially biased patterns are indeed likely to appear in the data. Note that the method remains conceptually simple, as it relies solely on comparisons with the standard normal distribution, i.e., basic z-tests. However, this simplicity comes at the expense of conservatism, as discussed in greater detail in Section 5.2.

The construction of the de-sparsified estimator $\hat{\mathbf{b}}$ demands the approximation of uncorrelatedness in the columns of $\mathbf{G}$ [22]. In Line 3 of Algorithm 2, the de-sparsification step therefore employs the precision matrix — formally introduced in Section 5.1 and estimated via nodewise Lasso — to account for the bias introduced by the ℓ_1 penalty. This adjustment ensures that the resulting estimator $\hat{\mathbf{b}}$ is asymptotically normal [22], allowing for the direct application of standard z-testing procedures. However, nodewise Lasso is computationally expensive since it requires fitting multiple models, as further elaborated in Section 5.1.

5.1 De-biased and Nodewise Lasso

In high-dimensional settings, an estimate of the precision matrix $\hat{\boldsymbol{\Theta}} \approx \boldsymbol{\Sigma}^{-1}$ is often required [71, 72], where $\boldsymbol{\Sigma} = \frac{1}{n}\mathbf{G}^\top\mathbf{G}$ is the (sample) covariance matrix of the design matrix $\mathbf{G}$. This comes from the fact that a direct inversion of $\boldsymbol{\Sigma}$ is generally infeasible when $m > n$, since it is not full rank.

Nodewise Lasso, popularized by Callot et al. [71], offers a stable, sparse approximation of $\boldsymbol{\Sigma}^{-1}$. The idea behind Algorithm 3 is to estimate to which degree the j-th basis function depends on all the other basis functions. By trying to predict the j-th column via regressing over all other columns denoted as $(\mathbf{G}_{.,-j})$, with the negative index being a compact notation for the inclusion of all columns except the j-th, one captures the conditional relationships of the basis functions and can later detect partial correlations. This concept works since the entries of the inverse covariance (precision) matrix are related to these partial correlations, as indicated by Meinshausen et al. [73].

For the said regression in Line 3 of Algorithm 3, we again use the Lasso objective and optimize for the estimated coefficient vector $\hat{\mathbf{e}}_{j,.} \in \mathbb{R}^{m-1}$. The matrix $\hat{\mathbf{E}} = (\hat{e}_{ij}) \in \mathbb{R}^{m\times(m-1)}$ stores the respective coefficient vector $\hat{\mathbf{e}}_{j,.}$ in the j-th row. Therefore, we interpret $\hat{\mathbf{e}}_{j,.}$ as a row vector. The penalty $(\lambda\|\hat{\mathbf{e}}_{j,.}\|_1)$ forces many of the entries in $\hat{\mathbf{e}}_{j,.}$ to be exactly zero if they do not contribute sufficiently to predicting $\mathbf{g}_{.,j}$. This represents the assumption that most variables are conditionally independent in high dimensions [73]. Moreover, this regularization helps mitigate issues with multicollinearity by promoting sparsity.

Note that the regularization parameter in Algorithm 3 could have been chosen differently compared to the regularization parameter in Equation (4.9). However, van de Geer et al. [22] also recommend

Algorithm 2 De-biased Lasso Statistical Testing (inspired by [22])

```
    Input:
      Estimated coefficient vector $\hat{\boldsymbol{\beta}} \in \mathbb{R}^m$
      Design matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$
      Response vector $\mathbf{y} \in \mathbb{R}^n$
      Significance level $\alpha \in (0,1)$
    Output: $z$-statistics $\mathbf{z} \in \mathbb{R}^m$ and $p$-values $\mathbf{p} \in [0,1]^m$
 1: Determine the target covariance matrix:
        $$\boldsymbol{\Sigma} = \frac{1}{n}\mathbf{G}^\top \mathbf{G}$$
 2: Obtain the approximated debiasing matrix $\hat{\boldsymbol{\Theta}} \in \mathbb{R}^{m \times m}$ via nodewise Lasso regression on $\mathbf{G}$ as
    described in Section 5.1, particularly in Algorithm 3.
 3: Derive the de-biased (de-sparsified) estimator:
        $$\hat{\mathbf{b}} = \hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\Theta}}\,\frac{\mathbf{G}^\top\big(\mathbf{y} - \mathbf{G}\hat{\boldsymbol{\beta}}\big)}{n}$$
 4: Compute the asymptotic variance matrix:
        $$\hat{\boldsymbol{\Omega}} = (\hat{\omega}_{ij}) = \hat{\boldsymbol{\Theta}}\,\boldsymbol{\Sigma}\,\hat{\boldsymbol{\Theta}}^T$$
 5: for $j = 1$ to $m$ do
 6:     Extract the standard error of the de-biased estimator:
            $$\mathrm{SE}(\hat{b}_j) = \sqrt{\hat{\omega}_{jj}}$$
 7:     Determine the $z$-statistic for $H_0 : \beta_j = 0$:
            $$z_j = \frac{\hat{b}_j}{\mathrm{SE}(\hat{b}_j)}$$
 8:     Calculate the two-sided p-value:
            $$p_j = 2\Big(1 - \Phi_z\big(|z_j|\big)\Big),$$
        where $\Phi_z$ denotes the standard normal Cumulative Distribution Function (CDF) and $|\cdot|$ repre-
        sents the absolute value function.
 9: end for
10: Return: $\mathbf{z}, \mathbf{p}$
```

that the regularization in nodewise regression should also be chosen as claimed in Equation (5.19), so it is simply assigned to precisely the same value in this thesis.

In Line 4 of Algorithm 3, the residual variance $\tau_j^2 \in \mathbb{R}_+$ is computed to quantify the variability in $\mathbf{g}_{\cdot,j}$ that remains unexplained by the other basis functions. This measure is later used to normalize the coefficients when constructing the estimated precision matrix $\hat{\boldsymbol{\Theta}}$.

In the theory of Gaussian graphical models [74, 75], which has inspired this procedure, the diagonal elements of $\hat{\boldsymbol{\Theta}}$, denoted as $\hat{\theta}_{jj}$, are inversely related to the residual variances τ_j^2. For the off-diagonal elements, $\hat{\theta}_{jk}$, the above Lasso regression procedure with the respective vector $\hat{\mathbf{e}}_{j,\cdot}$ uncovers the partial correlations of the j-th column of $\mathbf{G}$ with all others. Normalizing with the inverse residual variances (denoted as $\mathbf{D}^{-1}$ in Algorithm 3) appropriately scales each regression result. An equivalent, more explicit formulation, compared to the last computation of Algorithm 3, can be phrased as follows:

$$\hat{\theta}_{jj} = \frac{1}{\tau_j^2}, \quad \hat{\theta}_{jk} = -\frac{\hat{e}_{jk}}{\tau_j^2} \quad \text{for } k \neq j. \tag{5.25}$$

Algorithm 3 Nodewise Lasso Regression for Estimating the Precision Matrix (inspired by [73])

```
    Input:
      Design matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$
      Regularization parameter $\lambda \in [0, \infty)$
    Output: Estimated precision matrix $\hat{\mathbf{\Theta}} \in \mathbb{R}^{m \times m}$
1:  for $j = 1$ to $m$ do
2:      Define the response vector $\mathbf{g}_{\cdot,j}$ and the predictor matrix $\mathbf{G}_{\cdot,-j}$ (having column $j$ removed).
3:      Solve the Lasso problem for the $j$-th column using Algorithm 1:
            $\hat{\mathbf{e}}_{j,\cdot} = \left(\arg\min_{\boldsymbol{\gamma} \in \mathbb{R}^{m-1}} \left\{ \|\mathbf{g}_{\cdot,j} - \mathbf{G}_{\cdot,-j}\boldsymbol{\gamma}\|_2^2 + \lambda\|\boldsymbol{\gamma}\|_1 \right\}\right)^\top$
4:      Compute the residual variance:
            $\tau_j^2 = \|\mathbf{g}_{\cdot,j} - \mathbf{G}_{\cdot,-j}\hat{\mathbf{e}}_{j,\cdot}^\top\|_2^2$
5:  end for
6:  Construct the estimator for the precision matrix (more compact formulation of Equation (5.25)):
        $\hat{\mathbf{\Theta}} = \mathbf{D}^{-1}(\mathbf{I} - \hat{\mathbf{\Gamma}}),$
    where $\mathbf{D} = \mathrm{diag}(\tau_1^2, \tau_2^2, \ldots, \tau_m^2)$ is a diagonal matrix containing the residual variances, $\mathbf{I}$ is the
    identity matrix of the same dimension and $\hat{\mathbf{\Gamma}} \in \mathbb{R}^{m \times m}$ is the matrix whose $j$-th row contains the
    estimated coefficients $\hat{\mathbf{e}}_{j,\cdot}$ with a zero appended at the $j$-th position.
7:  Return: $\hat{\mathbf{\Theta}}$
```

5.2 Challenges and Considerations

Although the method demonstrated in this work provides a simple approach to test rules for statistical significance, it is not without limitations. A main hurdle for practical applicability is the runtime complexity. Concretely, a majority of the compute time can be attributed to nodewise Lasso, as the second step of Algorithm 3 requires multiple executions of Algorithm 1, which, depending on the convergence hyperparameters $t_{\max}$ and ϵ, is commonly computationally expensive.

Moreover, the statistical testing procedure, as outlined in Algorithm 2, only imposes boundaries on the significance level without addressing the power of the statistical test. From the applied z-testing procedure with the null hypothesis $H_0 : \beta_i = 0$ we have a theoretical basis for concluding that $\beta_i \neq 0$ when H_0 can be rejected for the i-th coefficient, up to a certain significance level α, expressing the probability of false rejections. Nonetheless, not rejecting H_0 does not formally imply $\beta_i = 0$. Instead, we interpret it as a heuristic (hint) that β_i could be zero. This limitation exists since bounding the power of a statistical test is out of scope for this thesis, especially when the underlying likelihoods remain unknown.

Additionally, this work does not optimize the conservatism of the test by taking into account correlations between individual predictors. Especially in the case where only few rules are removed by filtering, this lack of optimization is likely problematic since numerous statistical tests are applied and each has a nonzero probability of wrongly rejecting the respective null hypothesis. A certainly overly conservative but straightforward solution to this issue would be lowering the significance level, similar to the Bonferroni-Holm method [76]. Therefore, in future revisions, the implementation could make use of efficient adjustments, for instance via the joint maximum of test statistics, as highlighted by van de Geer et al. [22], or by employing group testing and other simulation-based corrections, as suggested by P. Bühlmann [77].

6 Verification and Concluding Thoughts

In this section, we validate the effectiveness of our approach in identifying known biases within synthetic datasets, as well as its potential to uncover novel insights from a real-world dataset.

Note that the resulting rule sets and coefficients are heavily dependent on the chosen hyperparameters, at least in terms of explanation compactness. The justification for this phenomenon is demonstrated in [1] and was further elaborated in Section 4.2.

6.1 Evaluation

We evaluate the software and its results by applying it to the three datasets introduced in Section 1.4 and by examining the outputs in detail. Although these datasets are also analyzed in [1], our evaluation differs in the following aspects:

1. In this work, the default model is compared to the NNST model, which is not employed in [1].
2. Additionally, the constraint that fuzzy variables can only appear once in an antecedent is removed (i.e., a superset of potential rules is fitted compared to [1]).

Because of the latter, the exemplary rule "`If` height `is` medium `AND` height `is` high `Then` ..." may also be generated. Indeed, if "medium" and "high" are neighboring fuzzy sets, a record can have a non-zero membership in both fuzzy sets simultaneously and thus fulfills this condition.

Moreover, while the statistical test for the NNST model does not have the same formal justification as that of the default model, its use within the scope of this thesis is considered being acceptable due to the reasoning provided at the end of Section 4.2.1.

6.1.1 Simple Noise Dataset

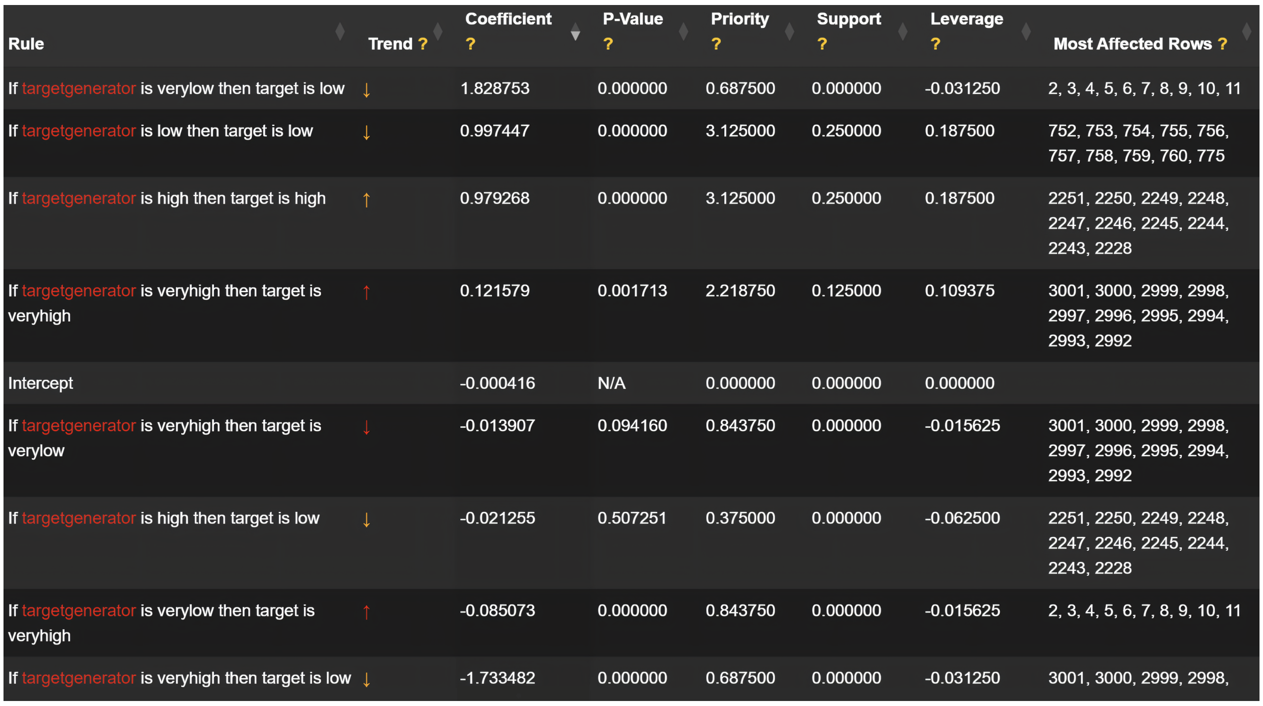
The dataset described in Section 1.4.1 serves as a sanity check, in the sense that it is intentionally simple, enabling us to verify if the method can reliably recover the most basic predefined rules and ignore uncorrelated noise. Indeed, our approach successfully explains this dataset across various hyperparameter configurations, though some minor peculiarities arise. Let us examine a concrete experiment by fixing the hyperparameters as follows:

- Regularization: $\lambda = 10$, $t_{\max} = 1000$, $\epsilon = 0.0001$
- No rule priority filtering is applied.
- Rules with precisely one antecedent are generated.
- Five MFs for fuzzification and defuzzification are used.

Rule priority filtering is not applied in this case, as the primary objective is to benchmark the rule generation and fitting components on this dataset. Thus, allowing advanced filtering may be perceived as "cheating", as it would allow the regression to select only correct rules on this simple dataset.

Figure 6.9 shows the full output from the default model, including all rules found using the provided hyperparameter settings and their respective (positive and negative) coefficients. Most importantly, the (uncorrelated) randomly generated columns are ignored by the model and all rules with positive coefficients are indeed reasonable statements. For many hyperparameter configurations, some slightly imprecise rules, like “`If` targetgenerator `is` verylow `Then` target `is` low”, can be found. This small peculiarity likely occurs due to the limited number of records. Moreover, the fuzzy sets where the MFs have non-zero densities at the outer borders of the domain, in this case “verylow” and “veryhigh”, intersect with their neighbor over all respective domain values where the membership degree is non-zero, as can be observed in Figure 2.7. Despite this slight imprecision, the outermost MFs remain essential for representing extreme values and outliers.

Subsequently, the experiment is repeated using the modified (NNST) model, where soft-thresholding with non-negative outputs is employed (Equation (4.17)) while all other hyperparameters are left untouched. As shown in Figure 6.10, the results reveal that the NNST model identified the rule “`If` targetgenerator `is` verylow `Then` target `is` verylow”, which was not found by the default model. Therefore, and by the fact that no “inverted” rules (with negative coefficients) are included, the explanatory capability of the NNST model on this simple dataset can indeed be judged as an improvement compared to the default model. Moreover, the resulting ruleset is more compact, which can be considered a comprehensibility enhancement.



Rule	Trend ?	Coefficient ?	P-Value ?	Priority ?	Support ?	Leverage ?	Most Affected Rows ?
If targetgenerator is verylow then target is low	↓	1.828753	0.000000	0.687500	0.000000	-0.031250	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
If targetgenerator is low then target is low	↓	0.997447	0.000000	3.125000	0.250000	0.187500	752, 753, 754, 755, 756, 757, 758, 759, 760, 775
If targetgenerator is high then target is high	↑	0.979268	0.000000	3.125000	0.250000	0.187500	2251, 2250, 2249, 2248, 2247, 2246, 2245, 2244, 2243, 2228
If targetgenerator is veryhigh then target is veryhigh	↑	0.121579	0.001713	2.218750	0.125000	0.109375	3001, 3000, 2999, 2998, 2997, 2996, 2995, 2994, 2993, 2992
Intercept		-0.000416	N/A	0.000000	0.000000	0.000000	
If targetgenerator is veryhigh then target is verylow	↓	-0.013907	0.094160	0.843750	0.000000	-0.015625	3001, 3000, 2999, 2998, 2997, 2996, 2995, 2994, 2993, 2992
If targetgenerator is high then target is low	↓	-0.021255	0.507251	0.375000	0.000000	-0.062500	2251, 2250, 2249, 2248, 2247, 2246, 2245, 2244, 2243, 2228
If targetgenerator is verylow then target is veryhigh	↑	-0.085073	0.000000	0.843750	0.000000	-0.015625	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
If targetgenerator is veryhigh then target is low	↓	-1.733482	0.000000	0.687500	0.000000	-0.031250	3001, 3000, 2999, 2998,

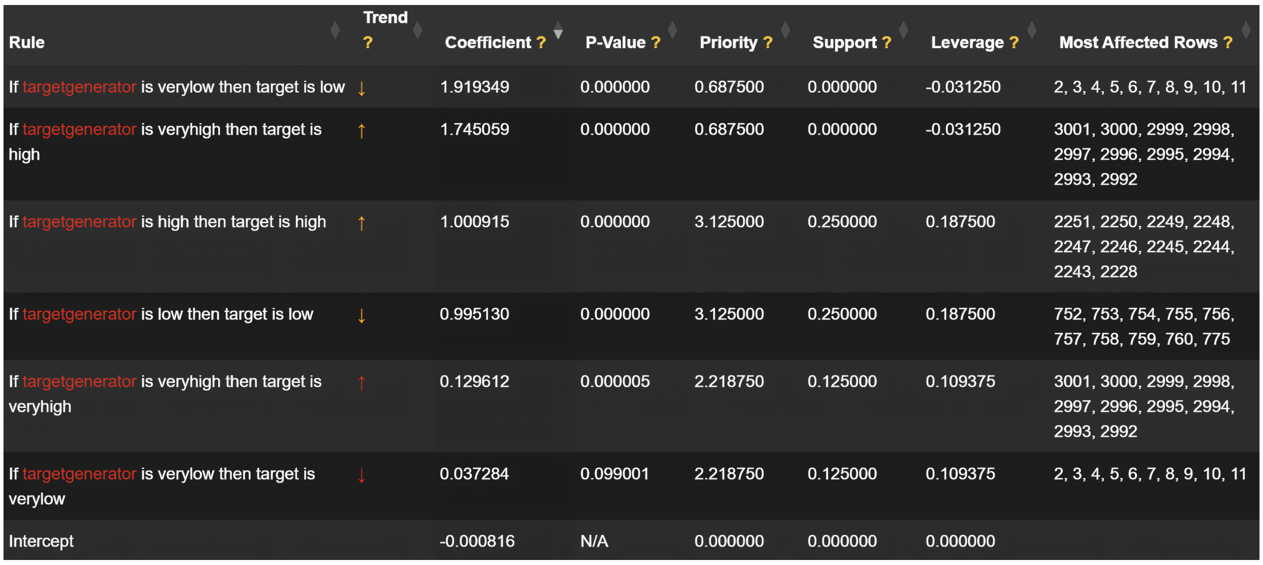
Figure 6.9: Default model’s rules obtained on the simple noise dataset (Section 1.4.1)

In both cases, the coefficient of determination (R^2) is 0.99. So, as expected, approximately all the variance is explained. Furthermore, this metric is still nearly perfect (approximately 0.99) after the filtering of the rules which could not reject the null hypothesis (cf., Section 3.5) is performed in both variants.

6.1.2 Biased Salaries Dataset

Since this dataset, described in Section 1.4.2, is much more complex and larger than the simple noise dataset from Section 1.4.1, the hyperparameters of the tool are selected to make use of more capabilities of the software, like rule priority filtering.

Therefore, the configuration settings are selected as follows:



Rule	Trend ?	Coefficient ?	P-Value ?	Priority ?	Support ?	Leverage ?	Most Affected Rows ?
If targetgenerator is verylow then target is low	↓	1.919349	0.000000	0.687500	0.000000	-0.031250	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
If targetgenerator is veryhigh then target is high	↑	1.745059	0.000000	0.687500	0.000000	-0.031250	3001, 3000, 2999, 2998, 2997, 2996, 2995, 2994, 2993, 2992
If targetgenerator is high then target is high	↑	1.000915	0.000000	3.125000	0.250000	0.187500	2251, 2250, 2249, 2248, 2247, 2246, 2245, 2244, 2243, 2228
If targetgenerator is low then target is low	↓	0.995130	0.000000	3.125000	0.250000	0.187500	752, 753, 754, 755, 756, 757, 758, 759, 760, 775
If targetgenerator is veryhigh then target is veryhigh	↑	0.129612	0.000005	2.218750	0.125000	0.109375	3001, 3000, 2999, 2998, 2997, 2996, 2995, 2994, 2993, 2992
If targetgenerator is verylow then target is verylow	↓	0.037284	0.099001	2.218750	0.125000	0.109375	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Intercept		-0.000816	N/A	0.000000	0.000000	0.000000	

Figure 6.10: NNST model's rules on the simple noise dataset (Section 1.4.1)

- Regularization: $\lambda = 50$, $t_{\max} = 5000$, $\epsilon = 0.0001$
- Rule priority filtering:

Min. priority: 0.01	Leverage weight: 10
Support weight: 1	Antecedents discount weight: 0

- All rules with up to two antecedents are generated.
- Five MFs for fuzzification and defuzzification are used.
- The target variable is filtered to remove records that do not fall into the range [25.000 USD, 85.000 USD].

For this dataset, the ambiguity highlighted in Section 4.2 comes into view. Notably, with the above hyperparameters, the rule "`If` gender `is` female `Then` salary `is` low" does not appear in the output of both the default and the NNST implementation. Instead, the extracted rules more directly show that low experience leads to a low salary. Since the dataset is explicitly designed to assign less experience to women [26], this direct explanation remains valid, even though it requires a more thorough investigation to discover the salary gap for women in the dataset.

As also mentioned in [1], such behavior is not observed for all hyperparameter settings. If we interrupt Lasso earlier but still apply strong regularization ($\lambda = 50, t_{\max} = 1000, \epsilon = 0.1$) and keep all other configuration options identical, then Table 6.3 shows that the model indeed discovers all injected biases from the dataset in the intended, more indirect form in both the default and the NNST variant. It is shown that even $H_0 : \beta_i = 0$ is rejected under the significance level $\alpha = 0.01$ for all desired rules. Note that a statistically significant rule (denoted with ✓✓) implies that the coefficient of the rule is not zero. In the case of Table 6.3, it also implies that the coefficients for the listed rules are strictly positive, as otherwise the meaning would be inverted (see Section 4.2).

Furthermore, the spurious influence included in the dataset — a uniform random field that does not help in explaining the data — is almost entirely ignored by both models. The remark "✗*" in Table 6.3 indicates that one or two rules (depending on the model) involving the uniform random field are present, suggesting occasional overfitting. These rules have low coefficients in both models and are not statistically significant, meaning that the null hypothesis that their coefficient is zero could not be rejected. For all experimented configurations, the p-values of these "wrong" rules consistently exceeded 0.5. Moreover, since these rules combined the random field with another antecedent, the reason for this effect to occur is likely the limited dataset size. Concretely, datasets with insufficiently many records randomly allow for the fact that the antecedent of a random field in conjunction with the MF of another field occasionally correlates with the target variable and therefore helps in explaining the data by chance. Most likely, different rule filtering approaches or using the method on datasets with more records would not exhibit this issue.

It is also noteworthy that the rule "`If` gender `is` female `Then` salary `is` low" ranked highest when sorted in descending order by the coefficient value on all four tested model variants shown in Table 6.4, yet it reflects an indirect effect. A model with $t_{\max} = 5000$ and $\epsilon = 0.0001$ captures this relationship in a less flexible manner than with $t_{\max} = 1000$ and $\epsilon = 0.1$. In both configurations, the model correctly identified that lower experience is associated with lower salary. However, reliably capturing the relationship between gender and experience with different hyperparameter settings would likely require modeling experience as a target variable and then incorporating the resulting rules, as discussed in more detail in Section 6.2.

Nonetheless, the more granular model ($t_{\max} = 5000$, $\epsilon = 0.0001$) also yields a rule suggesting that women with less experience tend to earn less — even if it does not opt for a more general (and sensational) formulation. However, considering that the income distribution between the genders in the dataset differs significantly — and that, among the top 15 percent most experienced individuals, women constitute less than 20 percent of the records, the more sensational formulation is certainly not faulty either.

Table 6.3: Discovered rules for the biased salaries dataset [26] (✓: $\beta_i > 10^{-8}$; ✓✓: p-value ≤ 0.01)

Rule or concept	**Default**	**NNST**
`If` gender `is` female `Then` salary `is` low	✓✓	✓✓
`If` job position `is` management `Then` salary `is` high	✓✓	✓✓
`If` hiring manager `is` B `and` gender `is` other `Then` salary is high	✓✓	✓✓
Influence of GPA	✓✓	✓✓
Influence of university reputation	✓✓	✓✓
Influence of experience	✓✓	✓✓
Influence of uncorrelated random field (should be false)	✗*	✗*
Influence of hiring manager A	✓✓	✓✓

For the updated configuration (with fewer iterations), the default model generated 87 rules in total, 33 of which reject the null hypothesis $H_0 : \beta_i = 0$ with the significance level $\alpha = 0.01$. However, the reduction for the NNST model, concretely, the number of omitted rules, for which H_0 could not be rejected, is considerably smaller compared to the default model. Interestingly, as shown in Table 6.4, both the full and the reduced default model explain the same amount of the target variable's variance as their NNST counterparts. Nonetheless, when only statistically significant rules are retained, the default model achieves this performance with a more compact rule set, whereas the NNST variant demands fewer rules on this dataset if significance-based filtering is not employed. Note that the statistical test is not explicitly adapted for the NNST model, so it is regarded only as a heuristic and these results should therefore be interpreted with caution (cf., Section 4.2.1).

Table 6.4: Model comparison with $\lambda = 50, t_{\max} = 1000, \epsilon = 0.1$ on the biased salaries dataset 1.4.2

Model description	**Number of rules**	$\mathbf{R^2}$
Default model	87	0.96
Default model (only statistically significant rules, $\alpha = 0.01$)	33	0.8
NNST model	73	0.96
NNST model (only statistically significant rules, $\alpha = 0.01$)	43	0.8

6.1.3 Boston Housing Dataset

The Boston housing dataset, previously introduced in Section 1.4.3, presents a regression problem on real-world data. To ensure a balanced approach between model complexity and generalization, hyperparameter tuning is conducted, especially focussing on the Lasso parameters. Thus, the following configuration is selected to optimize the trade-off between achieving a suitable coefficient of determination (R^2) and mitigating the risk of overfitting due to an excessive number of rules:

- Regularization: $\lambda = 20$, $t_{\max} = 1000$, $\epsilon = 0.001$
- Rule priority filtering:

Min. priority: 0.55	Leverage weight: 10
Support weight: 1	Antecedents discount weight: 1

- All rules up to two antecedents are generated.
- Five MFs for fuzzification and defuzzification are used.
- No records are filtered, since the authors of the dataset applied appropriate preprocessing and rescaling to the fields [28].

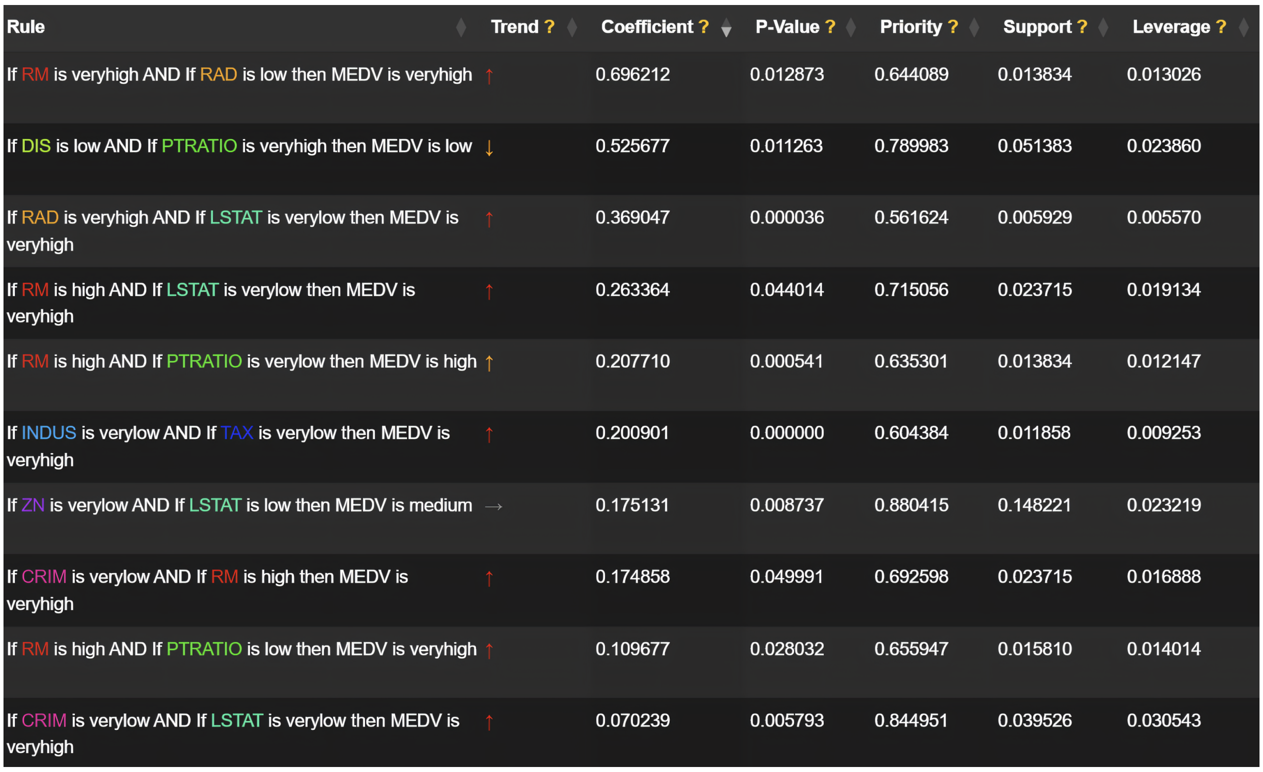
Both of the modelling techniques predict precisely the same top-three rules when sorting by the coefficients, which appear to be major impact factors in terms of driving the housing prices. The models show that districts with both high average room counts per household and poor accessibility to radial highways generally exhibit increased housing prices. And, neighborhoods with the quickest access to the radial highway and a low percentage of "low-status" residents similarly have high housing valuations. Conversely, areas with short weighted distances to employment centers and high pupil-to-teacher ratios tend to experience lower housing prices. All discovered rules, along with their corresponding p-values and priority values, are provided in Section 7.3 (Appendix), where the NNST results are listed in Table 7.6 and Table 7.7 shows the outputs for the default model.

A comparison of the model sizes and their respective explanatory power is provided in Table 6.5. As pessimistically expected, restricting the rule set to only retain statistically significant rules leads to a decline of the explained variance (R^2). The said reduction may be attributable to the many statistical tests conducted, which inherently increases the likelihood of at least one test being a false negative. This translates to the fact that at least one important rule for retaining the explanatory power could not reject the null hypothesis and is therefore erroneously removed.

Nevertheless, it is noteworthy that the NNST model retains a considerable portion of explained variance ($R^2 = 0.65$) with only ten rules, underscoring the efficiency of the demonstrated method. The reduced ruleset is visualized in Figure 6.11. Surprisingly, the reduction for the default model is now smaller compared to the NNST model. In Section 6.1.2, more precisely Table 6.4, one can observe the opposite effect for a different dataset, so there is likely no (strong) connection between the model type (default or NNST) and the number of statistically significant rules.

Table 6.5: Boston housing dataset model size and performance comparison

Model description	**Number of rules**	$\mathbf{R^2}$
Default model	55	0.88
Default model (only statistically significant rules, $\alpha = 0.05$)	18	0.74
NNST model	50	0.88
NNST model (only statistically significant rules, $\alpha = 0.05$)	10	0.65



Rule	Trend ?	Coefficient ?	P-Value ?	Priority ?	Support ?	Leverage ?
If RM is veryhigh AND If RAD is low then MEDV is veryhigh	↑	0.696212	0.012873	0.644089	0.013834	0.013026
If DIS is low AND If PTRATIO is veryhigh then MEDV is low	↓	0.525677	0.011263	0.789983	0.051383	0.023860
If RAD is veryhigh AND If LSTAT is verylow then MEDV is veryhigh	↑	0.369047	0.000036	0.561624	0.005929	0.005570
If RM is high AND If LSTAT is verylow then MEDV is veryhigh	↑	0.263364	0.044014	0.715056	0.023715	0.019134
If RM is high AND If PTRATIO is verylow then MEDV is high	↑	0.207710	0.000541	0.635301	0.013834	0.012147
If INDUS is verylow AND If TAX is verylow then MEDV is veryhigh	↑	0.200901	0.000000	0.604384	0.011858	0.009253
If ZN is verylow AND If LSTAT is low then MEDV is medium	→	0.175131	0.008737	0.880415	0.148221	0.023219
If CRIM is verylow AND If RM is high then MEDV is veryhigh	↑	0.174858	0.049991	0.692598	0.023715	0.016888
If RM is high AND If PTRATIO is low then MEDV is veryhigh	↑	0.109677	0.028032	0.655947	0.015810	0.014014
If CRIM is verylow AND If LSTAT is verylow then MEDV is veryhigh	↑	0.070239	0.005793	0.844951	0.039526	0.030543

Figure 6.11: NNST model's full significant ruleset for the Boston housing dataset [27] ($R^2 = 0.65$)

Moreover, the analysis partially aligns with the original work on the dataset [27] regarding the relationship between housing prices and concentration of nitric oxide in the respective district (`NOX`). The NNST model contains eight and the default model accommodates nine rules which have `NOX` as an antecedent before the application of statistical filtering (cf., Section 3.5). However, one should note that for the default model, precisely one of these rules is statistically significant with $\alpha = 0.05$, and the NNST model lists none of the rules containing `NOX` as statistically significant. In the analysis with the NNST models, other — perhaps more obvious — factors, like the number of rooms (`RM`) or the crime rate per capita (`CRIM`), show to be significant explanatory variables, as they are in (more) rules with lower p-values.

Another interesting point relates to the variable `B`, computed by the dataset authors as shown in Equation (6.26), where B_k is the percentage of Black residents in a district [27]. No conclusive influence of `B` can be found via the rules. Unfortunately, B_k is not explicitly given in the dataset. The racial disparities in this dataset from 1978 may have also correlated with other socio-economic factors, such as income or crime rates. Hence, rather than attributing influential power to `B`, the model could have assigned explanation weight to socio-economic factors correlated with `B`.

$$\texttt{B} = 1000(B_k - 0.63)^2 \tag{6.26}$$

As demonstrated in [1], alternative hyperparameter configurations for this dataset can yield conflicting rules, suggesting that there may be underlying influences not immediately observable. To systematically assess the rule consistency of the rule sets created in this thesis, a basic validation script[2] (also used in [1]) is employed here. This script systematically detects

- pairs of rules with identical antecedents but differing consequents and
- pairs of rules in which the set of antecedent assignments (if-parts) of one rule are a superset of the other rule, yet the consequent differs.

Nonetheless, unlike for the results obtained in [1], by the above criteria, no conflicting rules are discovered under the specified hyperparameters across all four models in Table 6.5 for the Boston housing dataset. With the entirety of the above analysis in mind, it may be concluded that the rules indeed confirm major relationships from socio-economic factors in Boston's (former) housing prices.

6.2 Future Work

As stated in [1], the software described in this thesis primarily serves as a proof of concept and there is great room for improvement and exploration. One possible area to focus on is incorporating (combinations of) more complex rules in the FIS and to thereby use more of their potential than the single-rule systems demonstrated herein. This may also aid in reducing the amount of necessary rules to generate, which causes the primary runtime- and space-complexity issue in the application. Although our minimal approach serves as a nicely performing proof of concept, more advanced rule sets combined with well-designed basis functions may ensure better efficiency and expressiveness.

Although using Lasso is a popular choice for promoting sparsity, there are a number of promising alternatives: Future work should consider other methods that may improve computational efficiency, some of which are outlined by Friedman et al. [64]. Moreover, there has been a vast amount of research surrounding different sparsity-inducing techniques as shown in Table 5.1 of [78]. Another worthwhile direction is enhancing the robustness against outliers — for instance, through regression techniques similar to Least Absolute Deviations (LAD) Lasso [79], albeit potentially at the cost of higher computational demands.

Additionally, advancements could focus on the interpretation of the resulting fuzzy rule sets themselves, where prior research has outlined that crisp interpretations are not appropriate [25]. The type-1 fuzzy sets used in this work offer simple means of associating linguistic labels (e.g., "low", "medium", "high") with underlying uncertain information, but they fundamentally rely on the meaning of these terms remaining static, even though they can greatly differ between contextual conditions and individuals. As also highlighted in [1], a natural improvement would be to extend type-1 fuzzy sets to type-2 [80] (or higher type) fuzzy sets that provide additional levels of uncertainty. Therefore, these models would be more flexible at representing variations in human interpretation and could provide more robust rule explanations. This, however, comes at a price: Type-2 fuzzy systems have more hyperparameters that users may need to specify — for instance, uncertainty bounds and additional aggregation strategies. This may therefore require more background knowledge or stronger

[2] [5, `/example_unveiling_biases/simple_contradiction_check.py`]

assumptions and increases computational complexity, leading to a system that could be harder to maintain, configure and interpret. Thus, a trade-off between expressiveness and practicality is required at this stage, which may be alleviated by future work.

Further aspects that can certainly be enhanced are the conservatism and the lack of power guarantees of the statistical testing methodology (cf., Section 5.2). Group-based testing methods, such as those highlighted in [22], might allow for improved estimates of rule significance. Building upon enhanced statistical tests, there may be vast potential in the integration of formal verification systems or actively evolving Chain-of-Thought (CoT) "reasoning models" [81, 82] to allow for a more compact representation of the method's resulting ruleset. Since the latter models operate on natural language, they can be supplied with (statistically significant) if-then rules resulting from the current method and are likely able to take their general-purpose world knowledge into consideration, which can help in avoiding a crisp interpretation of the rules, especially if additional background information on the fields and rules is provided.

A major restriction of the current implementation is that it only supports numerical target variables. Extending it to include categorical and multi-class classification problems would allow for more widespread adoption. Moreover, a potential area for future research lies in extending the approach to different modalities, like audio- or image-data. While this might initially sound far-fetched, fuzzy set theory has already been successfully employed in (medical) image processing [83] as well as audio classification [84]. These possibilities could also align with the concept of surrogate models, which are interpretable, but restricted to locally fitted approximations of complex models [85].

Once categorical target data can be modeled, a natural extension of the method would be to build a comprehensive explanation framework for complete datasets by running the model for every column as a target column. This may result in a graph-based representation in which nodes could correspond to fields and hyperedges could define learned rules, thereby representing their interactions. With appropriate postprocessing, such as causal influence models [86, 87] and advancements thereof which may be able to handle non-crisp rules, a holistic view of the relationships between the fields may be extracted from such a structure. This concept could be able to (fully) address ambiguity-related issues by representing all possible dependencies systematically, additionally reducing the differences in outputs across different hyperparameter settings.

6.3 Conclusion

Via a simple combination of FIS and a regularized regression model, we demonstrate a method that combines interpretability and statistical theory, allowing for both structured rule discovery and significance testing. Furthermore, we advance upon our previous developments [1] by fitting rules with overlapping MFs and enforcing non-negative rule coefficients. Thereby, the approach examines all feasible rule combinations up to a complexity threshold in a more interpretable manner, which ensures that hidden patterns (like biases) are not overlooked in favor of overly data-specific artifacts.

As demonstrated by empirical evaluations on both synthetic and real-world datasets, the method is able to detect and quantify both implicit and more direct biases in datasets with numerical target variables, such as gender-based salary disparities and socio-economic influences on housing prices. The results confirm that the described tool effectively isolates relevant decision-making factors while significantly downweighting or entirely discarding spurious correlations.

In spite of its advantages, the method has non-negligible limitations, including vast computational demands for big datasets and ambiguities in the formulation of the rulesets. Certainly, future implementations could improve computational efficiency without fully compromising generality, enhance the conservatism and runtime of the statistical testing process, and refine the postprocessing of weighted rule sets to reduce ambiguity. With this in mind, the method certainly has the potential for advancing XAI and for providing safer models by highlighting inconsistencies and disparities in datasets.

References

[1] S. Rass and M. Dallinger, “Statistically Testing Training Data for Unwanted Error Patterns using Rule-Oriented Regression,” 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2503.18497

[2] C. Cichy and S. Rass, “Fuzzy Expert Systems for Automated Data Quality Assessment and Improvement Processes,” in *Proceedings of the CEUR Workshop*, vol. 2751, 2020, pp. 7–11. [Online]. Available: https://ceur-ws.org/Vol-2751/short2.pdf

[3] ——, *A Fuzzy-Approximation-Approach to Explainable Information Quality Assessment.* International Business Information Management Association (IBIMA), 2019, pp. 3919–3931. [Online]. Available: https://www.researchgate.net/publication/337567351_A_Fuzzy-Approximation_Approach_to_Explainable_Data_Quality_Assessment

[4] “GNU General Public License Version 3,” Free Software Foundation. [Online]. Available: http://www.gnu.org/licenses/gpl.html

[5] M. Dallinger, “S0urC10ud/xai-fuzzy-regrules,” March 2025, [retrieved: 2025-03-01; commit b338d80]. [Online]. Available: https://github.com/S0urC10ud/xai-fuzzy-regrules

[6] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A Survey on Bias and Fairness in Machine Learning,” *ACM Comput. Surv.*, vol. 54, no. 6, July 2021. [Online]. Available: https://doi.org/10.1145/3457607

[7] Z. Chen, “Ethics and Discrimination in Artificial Intelligence-Enabled Recruitment Practices,” *Humanities and Social Sciences Communications*, vol. 10, no. 1, p. 567, 2023, published: 2023/09/13. [Online]. Available: https://doi.org/10.1057/s41599-023-02079-x

[8] A. Torralba and A. A. Efros, “Unbiased Look at Dataset Bias,” *CVPR 2011*, pp. 1521–1528, 2011. [Online]. Available: https://doi.org/10.1109/CVPR.2011.5995347

[9] V. Hofmann, P. R. Kalluri, D. Jurafsky, and S. King, “AI Generates Covertly Racist Decisions About People Based on Their Dialect,” *Nature*, vol. 633, no. 8028, pp. 147–154, September 2024, publisher: Nature Publishing Group, online: https://www.nature.com/articles/s41586-024-07856-5, DOI: 10.1038/s41586-024-07856-5 [retrieved: 2024-11-04].

[10] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning: Limitations and Opportunities.* MIT Press, 2023, [retrieved: 2025-03-31]. [Online]. Available: https://fairmlbook.org/

[11] B. Ghosh, “Interpretability and Fairness in Machine Learning: A Formal Methods Approach,” in *International Joint Conference on Artificial Intelligence*, 2023. [Online]. Available: https://doi.org/10.24963/ijcai.2023%2F816

[12] J. Han, J. Pei, and H. Tong, *Data Mining: Concepts and Techniques.* Morgan kaufmann, 2022. [Online]. Available: https://doi.org/10.1016/C2009-0-61819-5

[13] L.-X. Wang and J. Mendel, “Generating Fuzzy Rules by Learning from Examples,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992. [Online]. Available: https://doi.org/10.1109/21.199466

[14] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, “FRULER: Fuzzy Rule Learning through Evolution for Regression,” *Information Sciences*, vol. 354, pp. 1–18, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516301591

[15] M. Cintra, H. Camargo, and M. Monard, “Genetic Generation of Fuzzy Systems with Rule Extraction using Formal Concept Analysis,” *Information Sciences*, vol. 349-350, pp. 199–215, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516300858

[16] S. Caton and C. Haas, “Fairness in Machine Learning: A Survey,” *ACM Comput. Surv.*, vol. 56, no. 7, Apr. 2024. [Online]. Available: https://doi.org/10.1145/3616865

[17] S. Ruggieri, D. Pedreschi, and F. Turini, “Data Mining for Discrimination Discovery,” *TKDD*, vol. 4, 05 2010. [Online]. Available: http://dx.doi.org/10.1145/1754428.1754432

[18] L. Genga, L. Allodi, and N. Zannone, “Association Rule Mining Meets Regression Analysis: An Automated Approach to Unveil Systematic Biases in Decision-Making Processes,” *Journal of Cybersecurity and Privacy*, vol. 2, no. 1, pp. 191–219, 2022. [Online]. Available: https://www.mdpi.com/2624-800X/2/1/11

[19] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 4768–4777. [Online]. Available: https://dl.acm.org/doi/10.5555/3295222.3295230

[20] M. T. Ribeiro, S. Singh, and C. Guestrin, “"Why Should I Trust You?": Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

[21] W. Mendenhall and T. Sincich, *Statistics for Engineering and the Sciences*, 6th ed. Chapman and Hall/CRC, 2016. [Online]. Available: https://doi.org/10.1201/b19628

[22] S. v. d. Geer, P. Bühlmann, Y. Ritov, and R. Dezeure, “On Asymptotically Optimal Confidence Regions and Tests for High-Dimensional Models,” *The Annals of Statistics*, vol. 42, no. 3, pp. 1166–1202, June 2014, publisher: Institute of Mathematical Statistics. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-42/issue-3/On-asymptotically-optimal-confidence-regions-and-tests-for-high-dimensional/10.1214/14-AOS1221.full

[23] R. Reiter, “A Theory of Diagnosis from First Prinicples,” *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987, online: http://linkinghub.elsevier.com/retrieve/pii/0004370287900622 [retrieved: 2024-11-04].

[24] U. Junker, “QuickXPlain: Preferred Explanations and Relaxations for Over-Contrained Problems,” in *Proceedings of the 19th National Conference on Artifical Intelligence*, 2004, pp. 167–172. [Online]. Available: https://dl.acm.org/doi/10.5555/1597148.1597177

[25] J. M. Mendel and P. P. Bonissone, “Critical Thinking About Explainable AI (XAI) for Rule-Based Fuzzy Systems,” *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 12, pp. 3579–3593,

December 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9430516/

[26] M. Dallinger, "xai-fuzzy-regrules/example_unveiling_biases/biased_salaries at main · S0urC10ud/xai-fuzzy-regrules," March 2025. [Online]. Available: https://github.com/S0urC10ud/xai-fuzzy-regrules/tree/main/example_unveiling_biases/biased_salaries

[27] D. Harrison and D. L. Rubinfeld, "Hedonic Housing Prices and the Demand for Clean Air," *Journal of Environmental Economics and Management*, vol. 5, pp. 81–102, 1978. [Online]. Available: https://doi.org/10.1016/0095-0696%2878%2990006-2

[28] P. Perera, "The Boston Housing Dataset," 2018, online: https://kaggle.com/code/prasadperera/the-boston-housing-dataset [retrieved: 2025-02-19].

[29] A. Wang, "Factors Affected Housing Prices: Taking Boston as an Example," *Theoretical and Natural Science*, 2024. [Online]. Available: https://doi.org/10.54254/2753-8818/42/2024ch0213

[30] R. Binkyt.e-Sadauskien.e, K. Makhlouf, C. Pinz'on, S. Zhioua, and C. Palamidessi, "Causal Discovery for Fairness," in *AFCP*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2206.06685

[31] T. P. Pagano, R. B. Loureiro, F. V. N. Lisboa, R. M. Peixoto, G. A. S. Guimarães, G. O. R. Cruz, M. M. Araujo, L. L. Santos, M. A. S. Cruz, E. L. S. Oliveira, I. Winkler, and E. G. S. Nascimento, "Bias and Unfairness in Machine Learning Models: A Systematic Review on Datasets, Tools, Fairness Metrics, and Identification and Mitigation Methods," *Big Data and Cognitive Computing*, vol. 7, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2504-2289/7/1/15

[32] J. F. Kain and J. M. Quigley, "Housing Markets and Racial Discrimination: A Microeconomic Analysis," 1975. [Online]. Available: https://doi.org/10.2307/3146129

[33] G. Taguchi, *System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Costs*, ser. System of Experimental Design: Engineering Methods to Optimize Quality and Minimize Costs. UNIPUB/Kraus International Publications, 1987, no. Bd. 1. [Online]. Available: https://archive.org/details/systemofexperime0000tagu

[34] T. Ross, J. M. Booker, and W. J. Parkinson, *Fuzzy Logic and Probability Applications: Bridging the Gap*. ASA SIAM, 2002. [Online]. Available: https://dl.acm.org/doi/10.5555/778594

[35] M. Dallinger, "Utilizing Fuzzy Inference Systems for XAI," June 2024, Seminar Thesis, Supervised by S. Rass, [retrieved: 2025-03-01]. [Online]. Available: https://martin-dallinger.me/Martin%20Dallinger%20-%20Portfolio-Dateien/FuzzyRule-BasedRegressionForXAI.pdf

[36] L. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, June 1965. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S001999586590241X

[37] Mamdani, "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," *IEEE Transactions on Computers*, vol. C-26, no. 12, pp. 1182–1191, December 1977. [Online]. Available: http://ieeexplore.ieee.org/document/1674779/

[38] S. Y. C, J. Ebienazer, S. M, and S. S, "Fuzzy logic," *International Journal of Innovative Research in Information Security*, 2023. [Online]. Available: https://doi.org/10.26562/ijiris.2023.v0903.19

[39] N. Dhiman and M. Sharma, "Calculus of new intuitionistic fuzzy generator: In generated intuitionistic fuzzy sets and its applications in medical diagnosis," *Int. J. Adv. Appl. Sci*, vol. 7, pp. 125–130, 2020. [Online]. Available: https://doi.org/10.21833/ijaas.2020.10.014

[40] R. R. Yager, "On the measure of fuzziness and negation. ii. lattices," *Information and Control*, vol. 44, no. 3, pp. 236–260, 1980. [Online]. Available: https://doi.org/10.1016/S0019-9958(80)90156-4

[41] G. J. Klir and B. Yuan, "Fuzzy sets and fuzzy logic - theory and applications," 1995. [Online]. Available: https://dl.acm.org/doi/book/10.5555/202684

[42] E. P. Klement, R. Mesiar, and E. Pap, *Representations of T-norms.* Dordrecht: Springer Netherlands, 2000, pp. 121–140. [Online]. Available: https://doi.org/10.1007/978-94-015-9540-7_5

[43] K. Wang, "Computational Intelligence in Agile Manufacturing Engineering," in *Agile Manufacturing: The 21st Century Competitive Strategy.* Elsevier, 2001, pp. 297–315. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780080435671500164

[44] V. Ojha, A. Abraham, and V. Snášel, "Heuristic Design of Fuzzy Inference Systems: A Review of Three Decades of Research," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 845–864, 2019. [Online]. Available: https://doi.org/10.1016/j.engappai.2019.08.010

[45] E. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020737375800022

[46] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 1, pp. 116–132, January 1985. [Online]. Available: http://ieeexplore.ieee.org/document/6313399/

[47] J. J. Saade and H. B. Diab, "Defuzzification Methods and New Techniques for Fuzzy Controllers," *Iranian journal of electrical and computer engineering*, vol. 3, pp. 161–174, 2004, [retrieved: 2025-03-01]. [Online]. Available: https://www.sid.ir/FileServer/JE/89020040213.pdf

[48] M. C. Baddeley and D. V. Barrowclough, *An Introduction to Ordinary Least Squares.* Cambridge University Press, 2009, pp. 11–35. [Online]. Available: https://doi.org/10.1017/CBO9780511814839.003

[49] E. Massa, M. Jonker, K. Roes, and A. Coolen, "Correction of Overfitting Bias in Regression Models," 04 2022. [Online]. Available: http://dx.doi.org/10.48550/arXiv.2204.05827

[50] S.-W. Choi, "The Effect of Outliers on Regression Analysis: Regime Type and Foreign Direct Investment," *Quarterly Journal of Political Science*, vol. 4, pp. 153–165, 2009. [Online]. Available: https://doi.org/10.1561/100.00008021

[51] S. K. Solanki and J. T. Patel, "A Survey on Association Rule Mining," in *2015 Fifth International Conference on Advanced Computing and Communication Technologies*, 2015, pp. 212–216. [Online]. Available: https://doi.org/10.1109/ACCT.2015.69

[52] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser.

VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. [Online]. Available: https://dl.acm.org/doi/10.5555/645920.672836

[53] B. Bouchon-Meunier, G. Coletti, and R. R. Yager, *Modern Information Processing: From Theory to Applications.* USA: Elsevier Science Inc., 2011. [Online]. Available: https://doi.org/10.1016/B978-0-444-52075-3.X5000-6

[54] I. L. Ansorena, "Forecasting of Traffic Flows at Ferry Terminals. A Hybrid Model," *International Journal of Agile Systems and Management*, 2019. [Online]. Available: https://doi.org/10.1504/IJASM.2019.098707

[55] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of the royal statistical society series b-methodological*, vol. 58, pp. 267–288, 1996. [Online]. Available: https://doi.org/10.1111/J.2517-6161.1996.TB02080.X

[56] J. Ranstam and J. A. Cook, "LASSO Regression," *British Journal of Surgery*, vol. 105, no. 10, pp. 1348–1348, 08 2018. [Online]. Available: https://doi.org/10.1002/bjs.10895

[57] "Comparison of Lasso and Stepwise Regression in Psychological Data," vol. 20, pp. 121–143, Jun. 2024. [Online]. Available: https://doi.org/10.5964/meth.11523

[58] Y. Dodge, *The Concise Encyclopedia of Statistics, Gauss–Markov Theorem.* New York, NY: Springer New York, 2008, pp. 217–218. [Online]. Available: https://doi.org/10.1007/978-0-387-32833-1_159

[59] H. W. Kuhn and A. W. Tucker, "Nonlinear Programming," in *Traces and emergence of nonlinear programming.* Springer, 2013, pp. 247–258. [Online]. Available: https://doi.org/10.1007/978-3-0348-0439-4_11

[60] M. R. Osborne, B. Presnell, and B. A. Turlach, "On the lasso and its dual," *Journal of Computational and Graphical Statistics*, vol. 9, no. 2, pp. 319–337, 2000. [Online]. Available: https://doi.org/10.2307/1390657

[61] J. Nocedal and S. J. Wright, "Numerical Optimization," in *Fundamental Statistical Inference*, 2018. [Online]. Available: https://doi.org/10.1007/b98874

[62] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC2929880/

[63] J. H. Friedman, T. J. Hastie, H. Hofling, and R. Tibshirani, "Pathwise Coordinate Optimization," *The Annals of Applied Statistics*, vol. 1, pp. 302–332, 2007. [Online]. Available: https://doi.org/10.1214/07-AOAS131

[64] J. H. Friedman, T. J. Hastie, and R. Tibshirani, "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of statistical software*, vol. 33 1, pp. 1–22, 2010. [Online]. Available: https://doi.org/10.18637/JSS.V033.I01

[65] R. J. Tibshirani, "The Lasso Problem and Uniqueness," *Electronic Journal of Statistics*, vol. 7, no. none, pp. 1456 – 1490, 2013. [Online]. Available: https://doi.org/10.1214/13-EJS815

[66] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining Machine Learning Classifiers

Through Diverse Counterfactual Explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ser. FAT* '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 607–617. [Online]. Available: https://doi.org/10.1145/3351095.3372850

[67] K. Hagiwara, "Bridging between Soft and Hard Thresholding by Scaling," *IEICE Trans. Inf. Syst.*, vol. 105-D, pp. 1529–1536, 2021. [Online]. Available: https://doi.org/10.1587/transinf.2021EDP7223

[68] S. Reid, R. Tibshirani, and J. Friedman, "A Study of Error Variance Estimation in Lasso Regression," *Statistica Sinica*, vol. 26, no. 1, pp. 35–67, 2016. [Online]. Available: http://www.jstor.org/stable/24721190

[69] T. Sun and C.-H. Zhang, "Scaled Sparse Linear Regression," *Biometrika*, vol. 99, no. 4, pp. 879–898, 09 2012. [Online]. Available: https://doi.org/10.1093/biomet/ass043

[70] A. W. v. d. Vaart, *Asymptotic Statistics*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. [Online]. Available: https://doi.org/10.1017/CBO9780511802256

[71] L. Callot, M. Caner, A. Ö. Önder, and E. Ulaşan, "A Nodewise Regression Approach to Estimating Large Portfolios," *Journal of Business & Economic Statistics*, vol. 39, pp. 520 – 531, 2016. [Online]. Available: https://doi.org/10.1080/07350015.2019.1683018

[72] M. Caner, M. C. Medeiros, and G. F. R. Vasconcelos, "Residual Based Nodewise Regression in Factor Models with Ultra-High Dimensions: Analysis of Mean-Variance Portfolio Efficiency and Estimation of Out-of-Sample and Constrained Maximum Sharpe Ratios," Rio de Janeiro, 2021. [Online]. Available: https://hdl.handle.net/10419/249732

[73] N. Meinshausen and P. Bühlmann, "High-Dimensional Graphs and Variable Selection with the Lasso," *The Annals of Statistics*, vol. 34, no. 3, pp. 1436 – 1462, 2006. [Online]. Available: https://doi.org/10.1214/009053606000000281

[74] J. Whittaker, *Graphical Models in Applied Multivariate Statistics.* Wiley Publishing, 2009. [Online]. Available: https://doi.org/10.1007/BF02618478

[75] C. Uhler, "Geometry of Maximum Likelihood Estimation in Gaussian Graphical Models," *Annals of Statistics*, vol. 40, pp. 238–261, 2010. [Online]. Available: https://doi.org/10.1214/11-AOS957

[76] S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," *Scandinavian Journal of Statistics*, pp. 65–70, 1979. [Online]. Available: http://www.jstor.org/stable/4615733

[77] P. Bühlmann, "Statistical Significance in High-Dimensional Linear Models," *Bernoulli*, vol. 19, pp. 1212–1242, 2012. [Online]. Available: https://doi.org/10.3150/12-BEJSP11

[78] J. F. Lawless, "Statistics in Action: A Canadian Outlook," 2014. [Online]. Available: https://doi.org/10.1201/B16597

[79] G. L. Hansheng Wang and G. Jiang, "Robust Regression Shrinkage and Consistent Variable Selection Through the LAD-Lasso," *Journal of Business & Economic Statistics*, vol. 25, no. 3, pp. 347–355, 2007. [Online]. Available: https://doi.org/10.1198/073500106000000251

[80] J. Mendel and R. John, “Type-2 Fuzzy Sets Made Simple,” *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 117–127, April 2002. [Online]. Available: http://ieeexplore.ieee.org/document/995115/

[81] T. Zhong *et al.*, “Evaluation of OpenAI o1: Opportunities and Challenges of AGI,” *ArXiv*, vol. abs/2409.18486, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2409.18486

[82] DeepSeek-AI, D. Guo *et al.*, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” *ArXiv*, vol. abs/2501.12948, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2501.12948

[83] I. Bloch and A. Ralescu, *Fuzzy Sets Methods in Image Processing and Understanding.* Springer, 2023. [Online]. Available: https://doi.org/10.1007/978-3-031-19425-2

[84] M. Liu, C. Wan, and L. Wang, *A Fuzzy Logic Approach for Content-Based Audio Classification and Boolean Retrieval.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 135–156. [Online]. Available: https://doi.org/10.1007/978-3-540-39988-9_7

[85] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek, *Explainable AI Methods - A Brief Overview.* Cham: Springer International Publishing, 2022, pp. 13–38. [Online]. Available: https://doi.org/10.1007/978-3-031-04083-2_2

[86] J. Pearl, *Causality: Models, Reasoning and Inference*, 2nd ed. USA: Cambridge University Press, 2009. [Online]. Available: https://dl.acm.org/doi/book/10.5555/1642718

[87] A. Sharma and E. Kiciman, “DoWhy: An End-to-End Library for Causal Inference,” 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2011.04216

7 Appendix

This section serves as a complementary resource to the main parts of the thesis, providing supplementary details and further usage information for the software implementation. It includes an extensive list of acronyms as well as hyperparameters with their descriptions, default values and commented (inter-)dependencies. Additionally, Table 7.6 and Table 7.7 contain the complete rulesets from the analysis of the Boston Housing dataset (cf., Section 6.1.3), listing the relevant (statistical) metrics such as coefficients, p-values and priority scores for further analysis and interpretation.

7.1 Acronyms

API Application Programming Interface

CDF Cumulative Distribution Function

COG Center of Gravity

CoT Chain-of-Thought

CSV Comma-Separated Values

FIS Fuzzy Inference Systems

FRULER Fuzzy Rule Learning through Evolution for Regression

GPA Grade Point Average

IQR Interquartile Range

KKT Karush-Kuhn-Tucker

Lasso Least Absolute Shrinkage and Selection Operator

MF Membership Function

ML Machine Learning

MOM Mean of Maxima

MSE Mean Squared Error

NNST Non-Negative Soft-Thresholding

OA Orthogonal Array

OLS Ordinary Least Squares

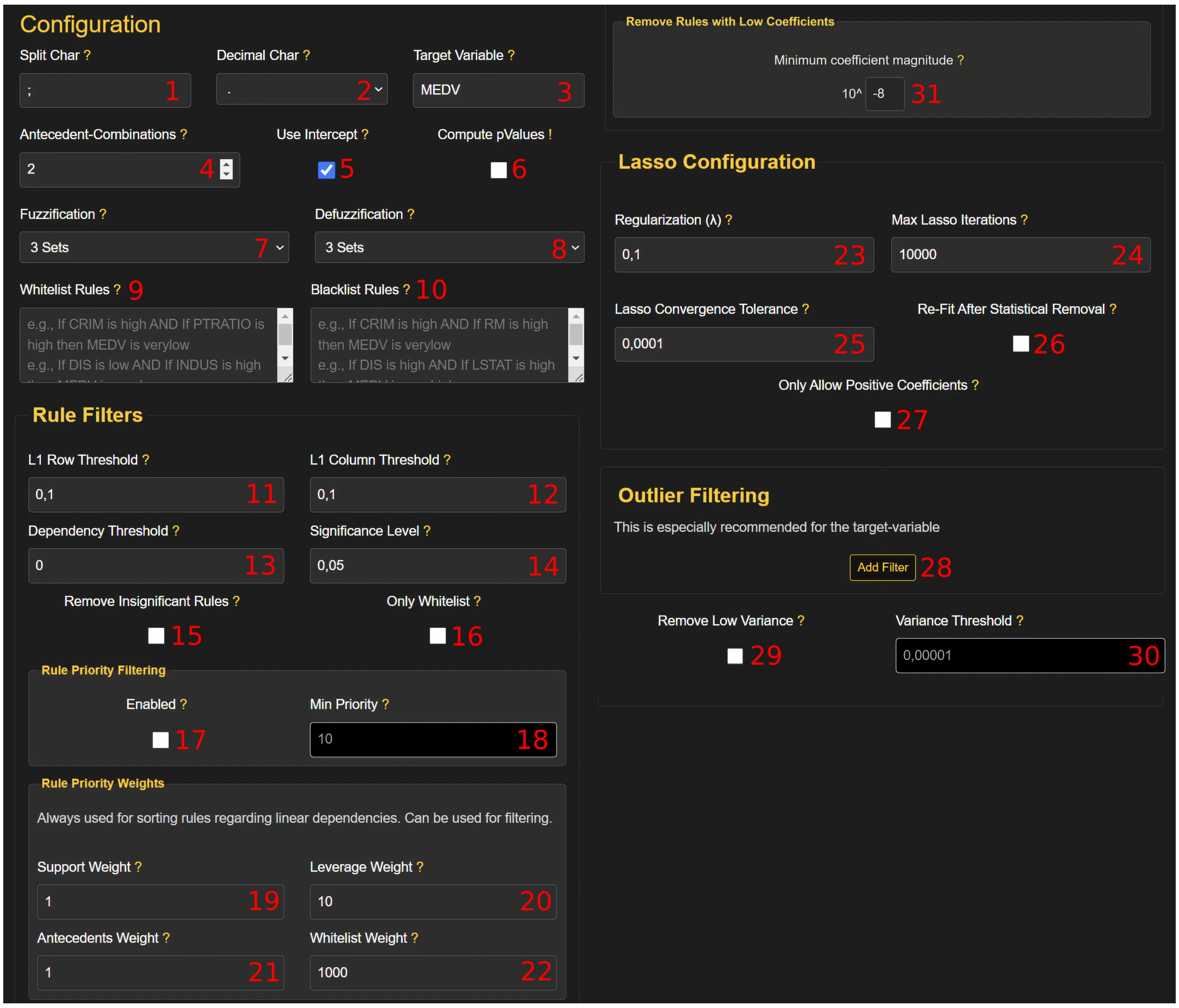
SSE Sum of Squared Errors

TSK Takagi-Sugeno-Kang

XAI eXplainable Artificial Intelligence

7.2 Usage of Hyperparameters

The following sections detail the hyperparameters configurable via the API and the web interface. Note that the digits in the subsequent enumeration refer to the red digits in Figure 7.12, which displays the configuration options available in the web interface.



Configuration

Split Char ? ; 1
Decimal Char ? . 2
Target Variable ? MEDV 3
Antecedent-Combinations ? 2 4
Use Intercept ? ☑ 5
Compute pValues ! ☐ 6
Fuzzification ? 3 Sets 7
Defuzzification ? 3 Sets 8
Whitelist Rules ? 9 e.g., If CRIM is high AND If PTRATIO is high then MEDV is verylow e.g., If DIS is low AND If INDUS is high
Blacklist Rules ? 10 e.g., If CRIM is high AND If RM is high then MEDV is verylow e.g., If DIS is high AND If LSTAT is high

Rule Filters

L1 Row Threshold ? 0,1 11
L1 Column Threshold ? 0,1 12
Dependency Threshold ? 0 13
Significance Level ? 0,05 14
Remove Insignificant Rules ? ☐ 15
Only Whitelist ? ☐ 16

Rule Priority Filtering

Enabled ? ☐ 17
Min Priority ? 10 18

Rule Priority Weights

Always used for sorting rules regarding linear dependencies. Can be used for filtering.

Support Weight ? 1 19
Leverage Weight ? 10 20
Antecedents Weight ? 1 21
Whitelist Weight ? 1000 22

Remove Rules with Low Coefficients

Minimum coefficient magnitude ? 10^ -8 31

Lasso Configuration

Regularization (λ) ? 0,1 23
Max Lasso Iterations ? 10000 24
Lasso Convergence Tolerance ? 0,0001 25
Re-Fit After Statistical Removal ? ☐ 26
Only Allow Positive Coefficients ? ☐ 27

Outlier Filtering

This is especially recommended for the target-variable

Add Filter 28

Remove Low Variance ? ☐ 29
Variance Threshold ? 0,00001 30

Figure 7.12: Configuration options in the web interface; red numbers reference the enumeration.

Basic Configuration

This section covers commonly used configuration settings for data ingestion and rule generation basics.

1. **Split Char**
   - API parameter name: `split_char`
   - Description: The character used to split columns in the CSV-file during data ingestion.
   - Default: ";"

2. **Decimal Char**
   - API parameter name: `decimal_point`
   - Description: Specifies the decimal separator for numerical values in the CSV-file.
   - Default: “`.`”
3. **Target Variable**
   - API parameter name: `target_var`
   - Description: The name of the target variable (column) in the CSV-file that the tool will attempt to explain.
   - Default: “Salary”
4. **Antecedent-Combinations** (κ)
   - API parameter name: `num_vars`
   - Description: This represents the number of if-parts (variables / antecedents) to combine via `AND` when generating rules. Increasing this number vastly increases the number of possible rule combinations.
   - Default: 2
   - Related Section: 2 (see discussion on combinatorial explosion of rule generation)
5. **Use Intercept**
   - API parameter name: `include_intercept`
   - Description: Determines whether an intercept term is included in the regression model (i.e., a basis function returning 1 for all inputs). Note that the target-data is z-score normalized upon data ingestion, so it may not be necessary.
   - Default: true
   - Related Section: 2.2
6. **Compute pValues**
   - API parameter name: `compute_pvalues`
   - Description: If true, the system computes p-values for each rule by fitting additional models for statistical testing. This process is computationally intensive as can be seen in Algorithm 2.
   - Default: true
   - Related Section: 5

7. **Fuzzification**
   - API parameter name: `numerical_fuzzification`
   - Description: An array of strings that defines the fuzzy sets (e.g., verylow, low, medium, high, veryhigh) used to convert numerical input variables into fuzzy membership values. In the web-interface, only a number can be selected, which relates to Table 2.2.
   - Default: three sets represented as [“low”, “medium”, “high”]
   - Related Section: 2.1
8. **Defuzzification**
   - API parameter name: `numerical_defuzzification`
   - Description: An array of strings specifying the fuzzy sets used during defuzzification to map fuzzy outputs back to crisp numerical predictions. In the web-interface, only a number can be selected, which relates to Table 2.2.
   - Default: [“low”, “medium”, “high”]
   - Related Section: 2.1
9. **Whitelist Rules**
   - API parameter name: `whitelist`
   - Description: A list of rules that are additionally proposed to the model, but may be filtered in Section 3.
   - Default: A record list (e.g., “`If` CRIM `is` high `AND If` PTRATIO `is` high `Then` MEDV `is` verylow”)
   - Related Section: 2
10. **Blacklist Rules**
    - API parameter name: `blacklist`
    - Description: A list of rules that are excluded after the automatic generation.
    - Default: A record list (e.g., “`If` CRIM `is` high `AND If` PTRATIO `is` high `Then` MEDV `is` verylow”)
    - Related Section: 2

Rule Filters

This section describes parameters used to filter the generated rules based on various criteria before or after the regression analysis.

11. **L1 Row Threshold** (ϕ_r)
    - API parameter name: `rule_filters.l1_row_threshold`
    - Description: The Manhattan norm threshold applied row-wise to detect and filter out duplicate or near-duplicate records in the design matrix before the regression starts.
    - Default: 0.1
    - Related Section: 3.3
12. **L1 Column Threshold** (ϕ_c)
    - API parameter name: `rule_filters.l1_column_threshold`
    - Description: The Manhattan norm threshold applied column-wise to identify and remove redundant basis functions (rules) before the regression starts.
    - Default: 0.1
    - Related Section: 3.3
13. **Dependency Threshold**
    - API parameter name: `rule_filters.dependency_threshold`
    - Description: The threshold for the norm of the residual from the Gram-Schmidt orthogonalization process. If a rule's residual vector norm falls below this value, it is considered linearly dependent on previously processed rules and can be removed. Setting this hyperparameter to 0 disables this check.
    - Default: 0
    - Note: This parameter is deprecated and not covered in this thesis, as experiments showed it was generally unnecessary after the introduction of Lasso regularization.
14. **Significance Level** (α)
    - API parameter name: `rule_filters.significance_level`
    - Description: The significance level (α) used with the null hypothesis $H_0 : \beta_i = 0$ to determine if the i-th rule's coefficient is statistically significant. The supplied value of the significance level is only relevant if the options “Compute pValues” (Parameter 6) and “Remove Insignificant Rules” (Parameter 15) are enabled.
    - Default: 0.05
    - Related Section: 5

15. **Remove Insignificant Rules**
    - API parameter name: `rule_filters.remove_insignificant_rules`
    - Description: If true (and if p-value computation is enabled, see Parameter 6), the tool automatically removes rules that fail the statistical significance test (i.e., those with p-values greater than or equal to the significance level α).
    - Default: false
    - Related Section: 3.5
16. **Only Whitelist**
    - API parameter name: `rule_filters.only_whitelist`
    - Description: When true, this option disables the automatic rule generation process, forcing the system to exclusively use the rules provided in the whitelist (Parameter 9).
    - Default: false
    - Related Section: Rule Generation in 2

Rule Priority Filtering: Parameters to enable and configure rule filtering based on a calculated priority score.

17. **Enabled**
    - API parameter name: `rule_filters.rule_priority_filtering.enabled`
    - Description: Enables the filtering of rules based on a computed priority score. This score is derived from metrics like support and leverage, the number of antecedents, whitelist status, and the corresponding weights (parameters 19-22).
    - Default: false
    - Related Section: 3.2
18. **Min Priority**
    - API parameter name: `rule_filters.rule_priority_filtering.min_priority`
    - Description: The minimum priority score a rule must achieve to be retained during the priority filtering process. Rules scoring lower than this threshold are removed. Note that priorities can be negative as the leverage may be negative.
    - Default: 0.04
    - Related Section: 3.2

Rule Priority Weights: These weights are always used for sorting rules, particularly relevant for the dependency threshold check (Parameter 13). If rule priority filtering (Parameter 17) is enabled, these weights determine the priority score used for filtering.

19. **Support Weight**

    – API parameter name: `rule_priority_weights.support_weight`

    – Description: The weight assigned to the rule's support metric (the proportion of records where the rule holds true in a discretized form of the data) when calculating its priority score.

    – Default: 1

    – Related Section: 3.2

20. **Leverage Weight**

    – API parameter name: `rule_priority_weights.leverage_weight`

    – Description: The weight assigned to the rule's leverage metric, which measures how much more often the antecedents and consequent appear together than expected by chance (deviation from statistical independence). Note that this metric can be negative.

    – Default: 10

    – Related Section: 3.2

21. **Antecedents Weight**

    – API parameter name: `rule_priority_weights.num_antecedents_weight`

    – Description: A weight applied to the inverse of the number of antecedents in a rule. This typically acts as a penalty for more complex rules (rules with more antecedents) when calculating priority.

    – Default: 1

    – Related Section: 3.2

22. **Whitelist Weight**

    – API parameter name: `rule_priority_weights.whitelist_boolean_weight`

    – Description: An additive weight applied to any rule present in the whitelist (Parameter 9). This boosts the priority of whitelisted rules, potentially preventing them from being filtered out by priority filtering if configured appropriately.

    – Default: 1000

    – Related Section: 3.2

Remove Rules With Low Coefficients: This configuration option has a higher parameter index since it was the last to be introduced during the final stages of refining this thesis.

31. **Minimum coefficient magnitude**
    - API parameter name: `rule_filters.coefficient_existence_threshold`
    - Description: Rules with a Lasso coefficient magnitude lower than this value will be ignored for statistical testing, as their coefficient is assumed to be effectively zero. This serves as a performance optimization by skipping tests for rules with negligible impact.
    - Default: 10^{-8}
    - Related Section: 3.4

Lasso Configuration

This section focuses on parameters controlling the Lasso regression process, including regularization strength and convergence criteria.

23. **Regularization** (λ)
    - API parameter name: `lasso.regularization`
    - Description: The regularization parameter (λ) for Lasso, used in Algorithm 1 and Algorithm 3. It controls the strength of the ℓ_1 penalty, managing the trade-off between model fit and coefficient sparsity. Higher values lead to more sparsity (more coefficients shrunk to zero).
    - Default: 1
    - Related Section: 4, especially Equation (4.9)
24. **Max Lasso Iterations** ($t_{\max}$)
    - API parameter name: `lasso.max_lasso_iterations`
    - Description: The maximum number of iterations ($t_{\max}$) allowed for the Lasso coordinate descent algorithm. The algorithm terminates upon convergence (determined using Parameter 25) or when this limit is reached, whichever happens first.
    - Default: 10^5
    - Related Algorithm: 1
25. **Lasso Convergence Tolerance** (ϵ)
    - API parameter name: `lasso.lasso_convergence_tolerance`
    - Description: The convergence threshold ϵ in Algorithm 1. The iterative Lasso algorithm is considered converged when the maximum absolute change in coefficients between successive iterations falls below this value.

- Default: 10^{-4}
- Related Algorithm: 1

26. **Re-Fit After Statistical Removal**
    - API parameter name: `re_fit_after_removing_insignificant_rules`
    - Description: If true (and if insignificant rules are removed via Parameter 15), the Lasso regression model is re-fitted using only the remaining significant rules. This updates the coefficients based on the reduced rule set.
    - Default: false
    - Related Section: 3.5
    - Note: Currently (commit `b338d80`), using a separate regularization parameter for the re-fitting step is not supported.

27. **Only Allow Positive Coefficients**
    - API parameter name: `lasso.only_allow_pos_coeff`
    - Description: If set to true, this option modifies the soft-thresholding operator within the Lasso algorithm to only permit non-negative coefficients (see Equation (4.17)), which is also referred to as the NNST model variant in this thesis. This enforces positive rule contributions, potentially improving interpretability in certain contexts.
    - Default: false
    - Related Section: 4.2.1

Outlier Filtering

Configuration for removing outliers from specific numerical columns prior to the analysis. This is particularly recommended for the target variable.

28. **Add Filter** (Mechanism)
    - API parameter name: `outlier_filtering`
    - Description: Provides configuration for outlier removal on specific numerical columns. This parameter accepts an object where keys are column names (e.g., “Salary”) and values specify the filtering method (e.g., “VariableBounds”, “IQR”) and its corresponding parameters (e.g., min/max bounds, IQR multiplier). Multiple columns can be configured independently.
    - Default: None (no filtering applied by default)
    - Related Section: 3.1
    - Example Usage: See the `Readme.md` file in the GitHub repository [5] for detailed examples like specifying bounds for “Salary” or using the IQR method for “TAX”.

Additional Configuration Options

The following configuration parameters offer additional options that are rarely needed, and are therefore listed at the end.

29. **Remove Low Variance**
    - API parameter name: `remove_low_variance`
    - Description: If true, numerical columns of the CSV-file with variance below the specified threshold (Parameter 30) are removed from the analysis rather than merely being mentioned in the "Warnings" section of the output.
    - Default: false
30. **Variance Threshold**
    - API parameter name: `variance_threshold`
    - Description: Fields with a variance below this threshold are flagged as low-variance. This helps in improving the rank of the design matrix.
    - Default: 10^{-5}
    - Related Section: 2.2

- **Return Contributions** (Not available in web interface)
  - API parameter name: `return_contributions`
  - Description: When set to true, the output includes the contribution matrix **C** from Section 1.3.2, which shows the normalized contribution (as originally conceptualized in [1]) of every rule for each record.
  - Default: false
  - Related Section: 1.3.2

7.3 Boston Housing Dataset Rules

The following two tables offer a complete list of the resulting rulesets obtained in Section 6.1.3. Table 7.6 lists all rules for the NNST model, where the p-values have to be interpreted more cautiously because the statistical test does not formally align with NNST (cf., Section 4.2.1). This does not influence the p-values in Table 7.7, which might, however, be harder to interpret because of rules with negative coefficients being included.

Table 7.6: Rules from the NNST model in Section 6.1.3 on the Boston housing dataset [27]

Rule Title	**Coefficient**	**p-value**	**Priority**
`If` RM `is` veryhigh `AND` RAD `is` low `Then` MEDV `is` veryhigh	0.7	0.013	0.64
`If` DIS `is` low `AND` PTRATIO `is` veryhigh `Then` MEDV `is` low	0.53	0.011	0.79

Rule Title	**Coefficient**	***p*-value**	**Priority**
If RAD is veryhigh AND LSTAT is verylow Then MEDV is veryhigh	0.37	< 0.001	0.56
If AGE is low AND DIS is low Then MEDV is medium	0.28	0.946	0.64
If RM is high AND LSTAT is verylow Then MEDV is veryhigh	0.26	0.044	0.72
If CRIM is medium Then MEDV is low	0.22	0.545	1.0
If RM is high AND PTRATIO is verylow Then MEDV is high	0.21	< 0.001	0.64
If LSTAT is high Then MEDV is low	0.2	0.456	1.01
If INDUS is verylow AND TAX is verylow Then MEDV is veryhigh	0.2	< 0.001	0.6
If CHAS is verylow AND RM is veryhigh Then MEDV is veryhigh	0.19	0.171	0.62
If NOX is high AND LSTAT is high Then MEDV is verylow	0.19	0.801	0.72
If ZN is verylow AND LSTAT is low Then MEDV is medium	0.18	0.009	0.88
If CRIM is verylow AND RM is high Then MEDV is veryhigh	0.17	0.05	0.69
If DIS is verylow AND LSTAT is verylow Then MEDV is veryhigh	0.16	0.554	0.68
If AGE is high AND B is low Then MEDV is low	0.15	0.305	0.57
If INDUS is high AND NOX is high Then MEDV is verylow	0.14	0.892	0.85
If CHAS is veryhigh AND NOX is medium Then MEDV is veryhigh	0.14	0.115	0.6
If DIS is verylow AND B is verylow Then MEDV is verylow	0.11	0.086	0.65
If RM is high AND PTRATIO is low Then MEDV is veryhigh	0.11	0.028	0.66
If RM is low AND RAD is low Then MEDV is low	0.11	0.064	0.66
If LSTAT is veryhigh Then MEDV is low	0.11	0.305	1.03
If TAX is veryhigh AND LSTAT is high Then MEDV is verylow	0.1	0.823	0.82
If AGE is veryhigh AND LSTAT is medium Then MEDV is verylow	0.1	0.445	0.63
If DIS is verylow AND LSTAT is high Then MEDV is verylow	0.1	0.832	0.82
If NOX is medium AND LSTAT is high Then MEDV is verylow	0.1	0.819	0.6
If CRIM is high Then MEDV is low	0.09	0.076	0.98
If CRIM is low AND NOX is high Then MEDV is verylow	0.08	0.886	0.78
If CHAS is veryhigh AND PTRATIO is high Then MEDV is veryhigh	0.07	0.186	0.55
If NOX is low AND RM is veryhigh Then MEDV is veryhigh	0.07	0.459	0.56
If CRIM is verylow AND LSTAT is verylow Then MEDV is veryhigh	0.07	0.006	0.84
If AGE is veryhigh AND LSTAT is high Then MEDV is verylow	0.06	0.724	0.77

Rule Title	**Coefficient**	p**-value**	**Priority**
If NOX is medium AND LSTAT is verylow Then MEDV is veryhigh	0.06	0.865	0.68
If LSTAT is verylow Then MEDV is veryhigh	0.06	0.333	1.34
If TAX is veryhigh AND LSTAT is medium Then MEDV is verylow	0.06	0.828	0.7
If RM is veryhigh AND DIS is low Then MEDV is veryhigh	0.05	0.272	0.56
If RM is high AND AGE is low Then MEDV is veryhigh	0.05	0.296	0.59
If INDUS is high AND LSTAT is verylow Then MEDV is veryhigh	0.05	0.935	0.64
If RM is high AND RAD is low Then MEDV is veryhigh	0.05	0.9	0.66
If CHAS is veryhigh AND B is veryhigh Then MEDV is medium	0.04	0.548	0.55
If CRIM is verylow AND B is veryhigh Then MEDV is high	0.04	0.254	0.71
If CHAS is verylow AND PTRATIO is high Then MEDV is low	0.04	0.207	1.1
If CHAS is verylow AND LSTAT is medium Then MEDV is verylow	0.04	0.11	0.65
If CRIM is medium Then MEDV is verylow	0.02	0.644	1.06
If NOX is medium AND PTRATIO is veryhigh Then MEDV is low	0.02	0.409	0.66
If CHAS is veryhigh AND RM is high Then MEDV is veryhigh	0.02	0.959	0.58
If INDUS is high AND B is low Then MEDV is verylow	0.01	0.607	0.57
If INDUS is high AND LSTAT is medium Then MEDV is verylow	0.01	0.431	0.67
If DIS is verylow AND LSTAT is medium Then MEDV is verylow	0.0	0.404	0.69
If AGE is low AND DIS is low Then MEDV is high	0.0	0.858	0.57

Table 7.7: Rules from the default model in Section 6.1.3 on the Boston housing dataset [27]

Rule Title	**Coefficient**	p**-value**	**Priority**
If RM is veryhigh AND RAD is low Then MEDV is veryhigh	0.68	0.003	0.64
If DIS is low AND PTRATIO is veryhigh Then MEDV is low	0.53	0.001	0.79
If RAD is veryhigh AND LSTAT is verylow Then MEDV is veryhigh	0.36	< 0.001	0.56
If AGE is low AND DIS is low Then MEDV is medium	0.26	0.023	0.64
If RM is high AND LSTAT is verylow Then MEDV is veryhigh	0.24	0.01	0.72
If INDUS is high AND NOX is high Then MEDV is verylow	0.22	0.371	0.85
If INDUS is verylow AND TAX is verylow Then MEDV is veryhigh	0.2	< 0.001	0.6
If DIS is low AND RAD is low Then MEDV is medium	0.19	0.003	0.81

Rule Title	**Coefficient**	***p*-value**	**Priority**
If CRIM is verylow AND RM is high Then MEDV is veryhigh	0.19	< 0.001	0.69
If CHAS is verylow AND RM is veryhigh Then MEDV is veryhigh	0.18	0.023	0.62
If AGE is veryhigh AND LSTAT is medium Then MEDV is low	0.18	0.166	1.33
If RM is high AND PTRATIO is verylow Then MEDV is high	0.16	0.012	0.64
If NOX is medium AND PTRATIO is veryhigh Then MEDV is low	0.15	0.17	0.66
If CHAS is veryhigh AND NOX is medium Then MEDV is veryhigh	0.13	0.101	0.6
If CRIM is medium Then MEDV is low	0.13	0.429	1.0
If NOX is medium AND LSTAT is verylow Then MEDV is veryhigh	0.11	0.023	0.68
If RM is high AND PTRATIO is low Then MEDV is veryhigh	0.11	0.005	0.66
If RM is low AND RAD is low Then MEDV is low	0.1	< 0.001	0.66
If CRIM is verylow AND B is veryhigh Then MEDV is high	0.1	0.157	0.71
If DIS is verylow AND LSTAT is verylow Then MEDV is veryhigh	0.1	0.514	0.68
If AGE is veryhigh AND LSTAT is high Then MEDV is verylow	0.09	0.249	0.77
If DIS is verylow AND LSTAT is high Then MEDV is verylow	0.09	0.931	0.82
If TAX is veryhigh AND LSTAT is high Then MEDV is verylow	0.08	0.524	0.82
If DIS is verylow AND B is verylow Then MEDV is verylow	0.08	0.134	0.65
If NOX is high AND LSTAT is high Then MEDV is verylow	0.07	0.427	0.72
If CRIM is verylow AND B is veryhigh Then MEDV is medium	0.07	0.411	1.34
If RM is high AND AGE is low Then MEDV is high	0.07	0.139	0.75
If NOX is low AND RM is veryhigh Then MEDV is veryhigh	0.06	0.076	0.56
If CHAS is veryhigh AND PTRATIO is high Then MEDV is veryhigh	0.06	0.046	0.55
If DIS is verylow AND LSTAT is medium Then MEDV is verylow	0.05	0.515	0.69
If RM is veryhigh AND DIS is low Then MEDV is veryhigh	0.05	0.261	0.56
If NOX is high AND DIS is verylow Then MEDV is verylow	0.05	0.3	0.86
If NOX is medium AND LSTAT is high Then MEDV is verylow	0.04	0.18	0.6
If RM is high AND RAD is low Then MEDV is high	0.04	0.979	0.79
If CRIM is medium AND TAX is veryhigh Then MEDV is verylow	0.03	0.264	0.56
If INDUS is high AND LSTAT is verylow Then MEDV is veryhigh	0.03	0.134	0.64

Rule Title	**Coefficient**	p**-value**	**Priority**
If CHAS is veryhigh AND RM is high Then MEDV is veryhigh	0.03	0.255	0.58
If CHAS is verylow AND PTRATIO is high Then MEDV is verylow	0.02	0.37	0.91
If LSTAT is high Then MEDV is low	0.02	0.204	1.01
If CHAS is veryhigh AND B is veryhigh Then MEDV is medium	0.01	0.994	0.55
If CRIM is low AND NOX is high Then MEDV is verylow	0.01	0.684	0.78
If RAD is verylow AND TAX is verylow Then MEDV is high	0.01	0.405	0.66
If CHAS is verylow AND PTRATIO is high Then MEDV is low	0.01	0.402	1.1
If AGE is high AND B is low Then MEDV is low	0.01	0.224	0.57
If CHAS is verylow AND LSTAT is medium Then MEDV is verylow	0.0	0.128	0.65
If RM is high AND AGE is low Then MEDV is veryhigh	0.0	0.092	0.59
If CRIM is verylow Then MEDV is high	0.0	0.846	1.14
If CRIM is verylow AND B is veryhigh Then MEDV is low	-0.0	0.387	0.58
If TAX is verylow Then MEDV is verylow	-0.01	0.906	0.91
If LSTAT is high Then MEDV is medium	-0.07	0.19	0.81
If LSTAT is verylow Then MEDV is verylow	-0.28	0.03	0.88
If DIS is verylow AND LSTAT is low Then MEDV is low	-0.29	< 0.001	0.67
If B is high AND LSTAT is low Then MEDV is low	-0.36	0.014	0.57
If RAD is veryhigh AND LSTAT is low Then MEDV is low	-0.37	< 0.001	0.69