

Bachelor's Thesis

Performance Modeling of Configurable Systems using Subgroup Discovery

Cag Öztopal

September 8, 2026

Advisor: Tobias Dick

Reviewers: Prof. Dr. Sven Apel
Prof. Dr. Jilles Vreeken



UNIVERSITÄT
DES
SAARLANDES

Erklärung Statement

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne die Beteiligung dritter Personen verfasst habe, und dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus Veröffentlichungen oder aus anderweitigen fremden Äußerungen entnommen wurden, sind als solche kenntlich gemacht. Insbesondere bestätige ich hiermit, dass ich bei der Erstellung der nachfolgenden Arbeit mittels künstlicher Intelligenz betriebene Software (z. B. ChatGPT) ausschließlich zur Textüberarbeitung/-korrektur und zur Code-Vervollständigung und nicht zur Bearbeitung der in der Arbeit aufgeworfenen Fragestellungen zu Hilfe genommen habe. Alle mittels künstlicher Intelligenz betriebenen Software (z. B. ChatGPT) generierten und/oder bearbeiteten Teile der Arbeit wurden kenntlich gemacht und als Hilfsmittel angegeben. Ich erkläre mich damit einverstanden, dass die Arbeit mittels eines Plagiatsprogrammes auf die Nutzung einer solchen Software überprüft wird. Mir ist bewusst, dass der Verstoß gegen diese Versicherung zum Nichtbestehen der Prüfung bis hin zum Verlust des Prüfungsanspruchs führen kann.

I hereby declare that I have written this thesis independently and without the involvement of third parties, and that I have used no sources or aids other than those indicated. All passages taken directly or indirectly from publications or other external sources have been identified as such. In particular, I confirm that I have used AI-based software (e.g., ChatGPT) exclusively for the following permitted sub-tasks: text rewriting/revision and code completion, and not to address or formulate the main research questions of the thesis. All parts of the thesis that were generated and/or edited using AI-based software (e.g., ChatGPT) have been disclosed and documented in accordance with the documentation requirements. I agree that the thesis may be checked using plagiarism detection software, including checks for the use of such software. I am aware that any violation of this declaration may result in failing the examination and lead to losing the right to be examined.

Saarbrücken, 9.9.2026 Sag Ötöpal
(Datum Date) (Unterschrift Signature)

Einverständniserklärung (optional) Declaration of Consent (optional)

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 9.9.26 Sag Ötöpal
(Datum Date) (Unterschrift Signature)

Abstract

Identifying performance-relevant options and interactions in configurable software is challenging because configuration spaces grow exponentially. Performance-influence models address this challenge by learning compact, interpretable models from a limited number of measurements. Existing approaches construct these models by heuristically generating and selecting candidate interaction terms.

In this thesis, we propose using subgroup discovery to generate interaction-related candidate terms for additive performance-influence models. To do so, we apply subgroup discovery to the signed residuals of a linear regression model and identify regions of the configuration space in which performance is systematically overpredicted or underpredicted.

We evaluate our approach on ten configurable software systems using both semi-synthetic and measured performance data. The discovered subgroup rules recover the underlying interaction structure only partially but provide useful alternative representations of interaction effects. On measured performance data, SD-guided feature selection significantly outperforms plain linear regression and remains competitive with two established baselines. Together, these findings support subgroup discovery as a promising mechanism for improving linear performance-influence models.

Contents

1	Introduction	1
2	Background	3
2.1	Configurable Systems	3
2.2	Performance-influence models	3
2.3	Regression	5
2.4	Subgroup Discovery	6
2.5	Subgroup Discovery in Our Setting	8
3	Methodology	9
3.1	Approach	9
3.2	Research Questions	11
3.3	Operationalization	11
3.4	Implementation Details and Hyperparameters	18
4	Evaluation	21
4.1	Results	21
4.1.1	RQ1: Alignment of Discovered Subgroup Rules and Recovery of Interactions	21
4.1.2	RQ2: Predictive Performance	33
4.2	Discussion	41
4.3	Threats to Validity	45
4.4	Future Work	47
4.4.1	Sampling Strategies	47
4.4.2	Model Simplification	48
5	Related Work	49
5.1	Performance Prediction for Configurable Systems	49
5.1.1	Interpretable regression models	49
5.1.2	Tree-based performance models	50
5.1.3	Other performance-prediction techniques	50
5.2	Subgroup Discovery and Rule-Based Model Analysis	51
5.3	Positioning of This Thesis	52
6	Concluding Remarks	53
A	Appendix	55
A.1	Supplementary RMSE Results for RQ2	55
	Statement on the Usage of Generative Digital Assistants	57
	Bibliography	59

List of Figures

Figure 4.1	Distributions of per run mean region alignment scores across datasets and interaction scenarios. Higher scores indicate closer correspondence between discovered subgroup rules and seeded interaction regions.	23
Figure 4.2	Distributions of per run mean region alignment scores across discovered-rule lengths and scenarios.	24
Figure 4.3	Distributions of per-run mean recovery scores for SD-GUIDED and STEPWISE across interaction scenarios.	25
Figure 4.4	Mean recovery by seeded interaction order, scenario, and method. Recovery is first averaged within each experimental run and interaction order, and then across datasets and seeds. Lines connect the interaction orders available within each scenario.	26
Figure 4.5	Mean recovery by training set support, interaction order, and method. Support is the percentage of training configurations in which a seeded interaction is active. Seeded interactions are grouped into the fixed support intervals shown on the horizontal axis. Recovery is first averaged within each experimental run and support interval, and then across datasets, scenarios, and seeds. Lines connect adjacent support intervals.	26
Figure 4.6	Median cumulative improvement in validation R^2 relative to the initial regression model across SD-GUIDED iterations. Each panel represents one dataset and summarizes 250 runs across five interaction scenarios and 50 seeds. For runs that terminate early, the final retained validation R^2 is carried forward when computing the median. Each curve is displayed only up to the first iteration at which it reaches its final plateau. Dashed vertical lines indicate the mean stopping iteration.	28
Figure 4.7	Median cumulative improvement in validation R^2 relative to the initial regression model across SD-GUIDED iterations. Each panel represents one interaction scenario and summarizes 500 runs across ten datasets and 50 seeds. For runs that terminate early, the final retained validation R^2 is carried forward when computing the median. Each curve is displayed only up to the first iteration at which it reaches its final plateau. Dashed vertical lines indicate the mean stopping iteration.	28

Figure 4.8	Distributions of held out test set R^2 scores for LINEAR, SD-GUIDED, and STEPWISE across datasets. Each panel represents one dataset and summarizes 250 runs per method across five interaction scenarios and 50 seeds. Boxes indicate the interquartile range, and the line inside each box marks the median. Potential outliers are shown as hollow circles.	31
Figure 4.9	Distributions of held out test set R^2 scores for LINEAR, SD-GUIDED, and STEPWISE across interaction scenarios. Each panel represents one scenario and summarizes 500 runs per method across ten datasets and 50 seeds. Boxes indicate the interquartile range, and the line inside each box marks the median. Potential outliers are shown as hollow circles.	32
Figure 4.10	Test set MAE distributions across 30 random seeds, shown separately for each dataset and method. Lower values indicate better predictive accuracy.	33
Figure 4.11	Test set R^2 distributions across 30 random seeds, shown separately for each dataset and method. Higher values indicate better predictive accuracy.	35
Figure 4.12	Median validation R^2 gain relative to the initial linear model over the SD-guided feature selection iterations, shown separately for each dataset and aggregated across 30 random seeds.	39
Figure 4.13	Distributions of run level mean seeded-interaction recovery for each scenario, comparing region-based Jaccard similarity with absolute MCC.	44
Figure 4.14	Distributions of run level mean rule alignment for each scenario, comparing region-based Jaccard similarity with absolute MCC. Runs without discovered rules are excluded.	44
Figure A.1	Distribution of test-set RMSE across datasets and prediction methods.	55

List of Tables

Table 2.1	Predicted execution time for all configurations under $\hat{\Pi}(c) = 100 + 27E + 10C - 5(E = 1 \wedge C = 1)$	4
Table 3.1	Configuration space characteristics of the real-world datasets before and after preprocessing.	12
Table 3.2	Semi-synthetic interaction scenarios.	15
Table 3.3	Default subgroup discovery settings used in the proposed approach.	18
Table 3.4	Default settings of the iterative feature-selection procedure.	19

Table 4.1	Percentage of the 50 semi-synthetic runs in which SD-guided discovered no rules, darker cells are worse.	22
Table 4.2	Median model size and composition across 250 semi-synthetic runs per dataset. Terms denotes the total number of regression terms, Rules the number of subgroup rules added by SD-guided, and Interactions the number of interaction terms retained by Stepwise.	30
Table 4.3	Test set MAE comparison. Each cell shows SD-guided's average percentage reduction in MAE relative to the method named above it across the same 30 data splits. Positive values favor SD-guided; parentheses show how often it achieved the lower MAE. Green indicates a majority of wins and red a majority of losses.	33
Table 4.4	Test set R^2 comparison. Each cell shows the average difference between SD-guided and the method named above it across the same 30 data splits. Positive values favor SD-guided; parentheses show how often it achieved the higher R^2 . Green indicates a majority of wins and red a majority of losses.	35
Table 4.5	Two-sided Wilcoxon signed-rank tests comparing SD-guided with each baseline across ten datasets. For each dataset, the performance difference was averaged over the same 30 data splits before testing. Cells show Holm-adjusted p -values; significant results ($p < 0.05$) are bold.	36
Table 4.6	Median model size and composition across 30 random seeds. Terms denote the total number of regression terms; Rules gives the number of subgroup rules added by SD-guided; and Interactions gives the number of interaction terms retained by Stepwise.	38
Table 4.7	Median runtime in seconds across 30 random seeds.	40
Table A.1	Test-set RMSE comparison. Each cell shows SD-guided's average percentage reduction in RMSE relative to the method named above it across the same 30 data splits. Positive values favor SD-guided; parentheses show how often it achieved the lower RMSE. Green indicates a majority of wins and red a majority of losses.	56

Introduction

Modern software systems offer many configuration options to adapt to different hardware, workloads, and user requirements. Such options range from feature toggles to numeric parameters. However, this flexibility increases variability in non functional properties such as performance or energy consumption: these properties can vary widely across configurations, and configuration options often interact; therefore, the effect of enabling two options together cannot be inferred from their individual effects alone [2, 24]. Moreover, because the configuration space grows exponentially with the number of options [24], exhaustively measuring all configurations is infeasible. Developers therefore need models that can predict the performance of unseen configurations from a limited number of measurements while also explaining which options and option interactions drive performance variation.

Prior work has explored diverse approaches for predicting performance under limited measurement budgets, including regression-tree models [7], transfer-learning techniques [11], approaches such as Fourier learning [28], and deep-learning-based models [9]. This thesis focuses on *performance-influence models* (PIMs) [23], i.e., typically linear models that represent performance as a sum of weighted terms for individual configuration options and selected interactions among options. Concretely, a PIM assigns weights to options (and some option combinations to account for influential interactions) and aggregates these weights to estimate performance of a previously unseen configuration. These interpretable models help developers identify influential options, explain why a configuration is fast or slow, debug performance issues, and optimize configurations under constraints. Siegmund et al. have shown that such models can be inferred with high accuracy from sparse measurements of the configuration space [23].

A key challenge in learning accurate PIMs is deciding which interaction terms to include. Because the number of possible option interactions grows exponentially with the number of configuration options, practical approaches cannot consider all possible terms explicitly. Existing approaches therefore rely on heuristic strategies to construct and select a manageable set of candidate interaction terms [23]. Regardless of how the configurations are collected, the learned model is constrained by the candidate terms made available to the selection procedure. If a relevant interaction is missing from this set, its effect may appear as systematic prediction errors for configurations that share particular option settings. This raises the question of whether such error patterns can reveal useful candidate terms for performance-influence models.

In this thesis, we explore *subgroup discovery* [1] as an alternative, error driven mechanism for generating subgroup-derived candidate model terms for performance-influence models. The approach is motivated by the observation that linear performance models with only main effects often underperform when relevant option interactions are present [23, 24].

Instead of enumerating interaction candidate terms upfront, we first fit a baseline regression model containing only single option terms and analyze its signed residuals on a separate validation set. If a relevant interaction is missing from the model, configurations in which this interaction is active may be systematically overpredicted or underpredicted. Systematic residual patterns may therefore reveal regions of the configuration space in which the current model fails to capture relevant performance behavior. These regions can provide useful candidates for model refinement, while still requiring validation because they may reflect true option interactions, but also noise, sampling artifacts, or correlated configuration options.

Subgroup discovery is suited for identifying such structures because it searches for interpretable regions of the data space that are exceptional with respect to a user-defined quality function and has been used to analyze where a trained model underperforms [17, 25, 26]. It returns logical descriptions of such regions, for example `ENCRYPTION = on  $\wedge$  COMPRESSION = off` in a database system. To capture both directions of systematic prediction error, we apply subgroup discovery once to the signed residuals and once to their negation, yielding subgroup descriptions for regions in which the model tends to underpredict or overpredict. Each discovered subgroup description is transformed into a binary subgroup-derived candidate term and temporarily added to the PIM. We then select the term that yields the larger improvement on the validation set and permanently add it to the model if the improvement exceeds a predefined threshold. This process is repeated iteratively until none of the candidate model terms provides sufficient validation improvement. After the feature set has been selected, the final model is retrained and evaluated on a held-out test set.

Background

In this section, we introduce the concepts required to understand our approach. We first define configurable systems and performance-influence models, building on and adapting the formalization of Siegmund et al. [23] to our setting. We then introduce subgroup discovery and the quality function used in our method.

2.1 Configurable Systems

A configurable system is characterized by a set of configuration options, denoted as $\mathcal{O} = \{o_1, o_2, \dots, o_n\}$. Each option $o_i \in \mathcal{O}$ takes values from a specific domain V_i . For binary options, $V_i = \{0, 1\}$, where 1 indicates that the option is enabled and 0 otherwise. For categorical options, V_i is a finite unordered set. For numeric options, the domain is a subset of the real numbers, $V_i \subseteq \mathbb{R}$.

A configuration $c \in V_1 \times V_2 \times \dots \times V_n$ is an n -tuple representing a complete assignment of options. In practice, not every tuple forms a valid configuration due to constraints (e.g., specifying a cache size is invalid if caching is disabled). We denote the set of all valid configurations as $\mathcal{C}$. For $c = (c_1, \dots, c_n) \in \mathcal{C}$, let $\pi_i : \mathcal{C} \rightarrow V_i$ denote the i -th coordinate projection, $\pi_i(c) = c_i$. For the sake of readability, we write $o_i(c) := \pi_i(c)$, and when c is clear from context we abbreviate $o_i(c)$ by o_i .

Example 2.1 (Example configuration). Consider a system with two binary configuration options, Encryption (E) and Compression (C), such that the option set is $\mathcal{O} = \{E, C\}$ with domains $V_E = V_C = \{0, 1\}$. The configuration space consists of four valid configurations: $\mathcal{C} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. For a given configuration $c = (0, 1)$, the encryption option evaluates to $E(c) = 0$. Following our established shorthand, this is abbreviated simply as $E = 0$ to indicate that encryption is disabled.

2.2 Performance-influence models

Performance-influence models (PIMs) quantify how individual configuration options and their interactions affect performance in an interpretable way. Formally a performance-influence model is a function from valid configurations to a performance measure: $\hat{\Pi} : \mathcal{C} \rightarrow \mathbb{R}$. The performance can be any measurable property such as energy consumption, execution time or memory consumption.

In general a PIM is represented by following formula:

$$\hat{\Pi}(c) = \beta_0 + \sum_{o_i \in \mathcal{O}} \phi_{o_i}(o_i) + \sum_{o_i \dots o_j \in \mathcal{O}} \Phi_{o_i \dots o_j}(o_i, \dots, o_j).$$

Here, β_0 is a constant baseline term, also called the intercept, that is independent of the chosen configuration. Each function ϕ_{o_i} models the contribution of a single option o_i based on its value. Each function $\Phi_{o_i \dots o_j}$ models the additional contribution of the interaction between options $o_i, \dots, o_j$ based on their joint values. The second sum ranges over the interaction terms included in the model.

This formulation is generalized: the option specific effects ϕ_{o_i} and interaction effects $\Phi_{o_i \dots o_j}$ may be non linear in the option values. In this work, however, we use a representation that is linear in its parameters. Concretely, we model performance as

$$\hat{\Pi}(c) = \beta_0 + \sum_{k=1} \beta_k t_k(c),$$

where each term $t_k : \mathcal{C} \rightarrow \mathbb{R}$ is an interpretable function of the configuration. Such terms may capture main effects of individual options as well as interactions between multiple options. For example, a singular option term may take the form $t_k(c) = o_i$, a conventional algebraic interaction term may take the form $t_k(c) = o_i o_j$, and a propositional interaction term may take the form $t_k(c) = P_k(c)$, where $P_k : \mathcal{C} \rightarrow \{0, 1\}$ is a Boolean valued function induced by a propositional condition over the configuration options. For a configuration c , such a term evaluates to 1 if c satisfies the corresponding condition, and to 0 otherwise.

For illustration, we reuse the example system from above with the binary options encryption (E) and compression (C), where $E, C \in \{0, 1\}$.

Example 2.2 (Example PIM).

$$\hat{\Pi}(c) = 100 + 27E + 10C - 5(E = 1 \wedge C = 1).$$

The constant term captures the baseline execution time. The terms $27E$ and $10C$ are main effects of enabling encryption and compression. Crucially, the interaction term $-5(E = 1 \wedge C = 1)$ models that enabling both options together has a smaller overhead than the sum of the individual effects, since compression reduces the amount of data that must be encrypted.

E	C	$E = 1 \wedge C = 1$	$\hat{\Pi}(c)$
0	0	0	100
0	1	0	110
1	0	0	127
1	1	1	132

Table 2.1: Predicted execution time for all configurations under $\hat{\Pi}(c) = 100 + 27E + 10C - 5(E = 1 \wedge C = 1)$.

A performance-influence model $\hat{\Pi}$ is learned from a set of measured configurations $\mathcal{C}_m \subseteq \mathcal{C}$ and their observed performance values $\Pi : \mathcal{C}_m \rightarrow \mathbb{R}$. Given a fixed set of terms (singular option terms and/or option interaction terms), the parameters can be learned using regression. The main challenge is selecting a small set of informative terms, because the number of possible option interactions grows exponentially with $|\mathcal{O}|$ [24]. Existing approaches therefore combine regression with term-selection strategies (e.g., stepwise selection) [23] to obtain accurate yet interpretable models.

2.3 Regression

Linear regression models a real valued target variable y as a linear combination of predictors $x_1, \dots, x_p$:

$$y = \beta_0 + \sum_{k=1}^p \beta_k x_k + \varepsilon,$$

where β_0 is the intercept, $\beta_1, \dots, \beta_p$ are coefficients, and ε is the error term. Given a set of observations, the coefficients are typically estimated by minimizing the sum of squared errors.

In our setting, each observation consists of a measured configuration $c \in \mathcal{C}_m$ and its observed performance values $\Pi(c)$. We fix a set of model terms $t_1, \dots, t_p$ and use their evaluations as predictors, i.e., $x_k := t_k(c)$. Thus, the regression model takes the form

$$\hat{\Pi}(c) = \beta_0 + \sum_{k=1}^p \beta_k t_k(c),$$

where the terms $t_k(c)$ may represent main effects i.e. the effects exhibited by individual configuration options, algebraic interaction terms, or propositional interaction terms of the form $t_k(c) = P_k(c)$. Learning $\hat{\Pi}$ therefore amounts to estimating the coefficients β_k such that $\hat{\Pi}(c)$ approximates $\Pi(c)$ on $\mathcal{C}_m$.

For $c \in \mathcal{C}_m$, we define the residual

$$r(c) = \hat{\Pi}(c) - \Pi(c)$$

Error Metrics: We assess predictive performance using three complementary metrics [10]:

The coefficient of determination (R^2) indicates how much of the variance in the target variable is explained by the model.

$$R^2 = 1 - \frac{\sum_{c \in \mathcal{C}_m} r(c)^2}{\sum_{c \in \mathcal{C}_m} (\Pi(c) - \mu)^2}.$$

The mean absolute error (MAE) measures the average absolute deviation between predictions and true values. Because it is expressed in the original unit of the target, it is easy to interpret as the typical prediction error.

$$\text{MAE} = \frac{1}{|\mathcal{C}_m|} \sum_{c \in \mathcal{C}_m} |r(c)|.$$

The root mean squared error (RMSE) measures the quadratic average of the prediction errors. By squaring deviations before averaging, it assigns greater weight to large errors and is therefore particularly informative when large mispredictions are undesirable.

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{C}_m|} \sum_{c \in \mathcal{C}_m} r(c)^2}.$$

Here, $\mu = \frac{1}{|\mathcal{C}_m|} \sum_{c \in \mathcal{C}_m} \Pi(c)$ denotes the mean of the true values.

2.4 Subgroup Discovery

Subgroup Discovery is a descriptive data mining technique for identifying subsets of observations in which a target variable exhibits unusual behavior relative to the full dataset [1]. We introduce the following definitions, based on Atzmüller [1], and adapt them to standard machine learning notation. Let $\mathcal{X}$ denote the space of predictor vectors and let Y be a real-valued target variable with domain $\mathcal{Y} \subseteq \mathbb{R}$. We consider a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n,$$

where each observation consists of a predictor vector $x_i \in \mathcal{X}$ and an observed target value $y_i \in \mathcal{Y}$. The goal of subgroup discovery is to find subsets of observations, called subgroups, that can be described by interpretable conditions on the predictor variables and are exceptional with respect to the target variable.

To describe such subsets formally, we introduce *selectors*. Let $x = (x_1, \dots, x_p) \in \mathcal{X}$ denote the predictor vector of an observation, where each component x_j takes values in a domain $\mathcal{X}_j$. A selector is a Boolean-valued function

$$s : \mathcal{X} \rightarrow \{0, 1\}$$

that evaluates whether an observation x_i satisfies a condition on one predictor x_j . We represent Boolean truth values numerically, with 1 denoting true and 0 denoting false. Depending on the type of predictor, a selector is defined in one of the following ways.

For a nominal or binary predictor x_j , the selector checks whether the j -th predictor of observation x_i is equal to a specific value v :

$$s(x_i) = \begin{cases} 1 & \text{if } x_{ij} = v, \\ 0 & \text{otherwise.} \end{cases} \quad (2.1)$$

For a numeric predictor, the selector checks whether the value of predictor x_j in observation x_i lies within a specified interval $[v_{\min}, v_{\max}] \subseteq \mathcal{X}_j$:

$$s(x_i) = \begin{cases} 1 & \text{if } x_{ij} \in [v_{\min}, v_{\max}], \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

Subgroup Descriptions: Individual selectors are combined to form a subgroup description d . This is achieved using logical conjunction. Thus, a description d belonging to a description language L is formulated as:

$$d(x_i) = s_1(x_i) \wedge s_2(x_i) \wedge \cdots \wedge s_k(x_i) \quad (2.3)$$

A subgroup $S_d \subseteq \mathcal{D}$ is formally defined as the extension of a description d , that is, the set of all observations in the dataset that satisfy the description:

$$S_d = \{(x_i, y_i) \in \mathcal{D} \mid d(x_i) = 1\} \quad (2.4)$$

The Quality Function: To quantify the “exceptionality” of a subgroup S_d with respect to the continuous target variable Y , Subgroup Discovery uses a quality function $q : L \rightarrow \mathbb{R}$. This function serves as an objective measure for ranking subgroup descriptions during the search process.

For a continuous target variable, the quality function can, for example, balance the size of the subgroup against the deviation of its target mean from the mean target value in the full dataset. A standard formulation is:

$$q(d) = |S_d|^\alpha \cdot (\mu_{S_d} - \mu_{\mathcal{D}}) \quad (2.5)$$

Where:

- $|S_d|$ is the number of observations in the subgroup.
- $\alpha \in [0, 1]$ is a weighting parameter, often set to $\alpha = 0.5$, that controls the trade-off between subgroup size and deviation magnitude.
- μ_{S_d} is the mean of the target variable Y over the subgroup S_d .
- $\mu_{\mathcal{D}}$ is the mean of the target variable Y over the full dataset $\mathcal{D}$.

The subgroup discovery task: Given a dataset

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n,$$

a target variable Y , a description language L , and a quality function $q : L \rightarrow \mathbb{R}$, the goal of the subgroup discovery task is to find a description $d \in L$ that induces a high-quality subgroup $S_d \subseteq \mathcal{D}$. Formally, we aim to identify

$$d^* = \arg \max_{d \in L} q(d).$$

The search strategy: Subgroup discovery can be performed using *exhaustive* or *heuristic* search strategies. Exhaustive methods, such as *branch-and-bound* search, systematically explore the description space L and can guarantee optimality under suitable pruning conditions. Heuristic methods, such as *beam search*, trade optimality for scalability. Since the description space L is typically too large to enumerate exhaustively, we rely on heuristic search strategies and specifically use beam search, a method that keeps only a bounded number of candidates per refinement step.

Beam search starts from the empty description $d_\emptyset$ and iteratively refines descriptions by adding one selector, i.e., $d' = d \wedge s$. Let $\text{len}(d)$ denote the number of selectors in d . At level ℓ , beam search maintains a beam $B_\ell \subseteq L$ of at most b descriptions (the *beam width*), where all $d \in B_\ell$ satisfy $\text{len}(d) = \ell$. For each $d \in B_\ell$, beam search generates candidate refinements, evaluates them using the quality function $q(\cdot)$, and selects the top- b candidates to form $B_{\ell+1}$. The search stops after reaching a maximum description length or when no valid refinements remain. Beam search returns the highest-quality description encountered during the search and thus provides an approximation of the optimum of q .

2.5 Subgroup Discovery in Our Setting

In our setting, each data point consists of a measured configuration $c_i \in \mathcal{C}_m$ and an associated numeric value $z_i \in \mathbb{R}$. Using the shorthand introduced earlier, we write the configuration as $x_i = c_i = (o_1, \dots, o_p)$, where o_j denotes the value of option j in configuration c_i . The corresponding target variable is $y_i = z_i$. For example, if the target is measured performance, then $z_i = \Pi(c_i)$. If the target is a model error, then $z_i = r(c_i)$ or $z_i = -r(c_i)$.

A subgroup description is therefore a conjunction of predicates over option values and identifies a subset of measured configurations. For binary options, a description may take the form $o_1 = 1 \wedge o_2 = 1$, which identifies configurations in which o_1 and o_2 are enabled simultaneously.

Methodology

We first introduce Subgroup-Discovery-guided (SD-guided) feature selection. This technique applies subgroup discovery on signed residuals to generate subgroup derived candidate terms, which may correspond to interaction-related regions of the configuration space. We then define the research questions that guide the evaluation and describe how the approach is measured against two primary goals: investigating whether and when the found descriptions correspond to known interaction structure and whether they improve predictive performance. Finally, we discuss the experimental setup, evaluation metrics, and the baselines we compare our method to.

3.1 Approach

Our goal is to predict the performance of a configurable system for an unseen configuration. To this end, we augment a base regression model with additional terms that aim to enable it to capture interaction effects. Specifically, we employ subgroup discovery as an iterative feature selection mechanism to identify informative, rule-based descriptors of the configuration space in which the model underperforms. These descriptions are added to the regression model as predictors whenever they yield a measurable improvement in predictive performance. Algorithm 1 outlines the pseudocode for this process, and the individual steps are explained in more detail afterwards.

Algorithm 1 SD-guided feature selection**Require:** Dataset $\mathcal{D}$, improvement threshold ε , maximum number of iterations T **Ensure:** Selected feature set F and final model M_{final}

```

1:  $(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}) \leftarrow \text{SPLIT}(\mathcal{D})$ 
2:  $F \leftarrow$  original configuration features
3:  $M \leftarrow \text{TRAINLINEARMODEL}(\mathcal{D}_{\text{train}}, F)$ 
4:  $s \leftarrow \text{EVALUATE}(M, \mathcal{D}_{\text{val}})$ 
5: for  $t = 1, \dots, T$  do
6:    $r \leftarrow \text{SIGNEDRESIDUALS}(M, \mathcal{D}_{\text{val}})$ 
7:    $d_+ \leftarrow \text{SUBGROUPDISCOVERY}(\mathcal{D}_{\text{val}}, r)$ 
8:    $d_- \leftarrow \text{SUBGROUPDISCOVERY}(\mathcal{D}_{\text{val}}, -r)$ 
9:    $H \leftarrow \{\text{INDICATORFEATURE}(d_+), \text{INDICATORFEATURE}(d_-)\}$ 
10:   $(h^*, M^*, s^*) \leftarrow \text{BESTCANDIDATE}(H, F, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})$ 
11:  if  $s^* - s < \varepsilon$  then
12:    break
13:  end if
14:   $F \leftarrow F \cup \{h^*\}$ 
15:   $M \leftarrow M^*$ 
16:   $s \leftarrow s^*$ 
17: end for
18:  $M_{\text{final}} \leftarrow \text{TRAINLINEARMODEL}(\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}}, F)$ 
19:  $\text{EVALUATE}(M_{\text{final}}, \mathcal{D}_{\text{test}})$ 

```

Given a dataset $\mathcal{D} = \{(c_i, \Pi(c_i))\}_{i=1}^n$ where $c_i \in \mathcal{C}_m$ denotes the configuration of the i -th observation and $\Pi(c_i)$ its measured performance, the prediction target is given by $y = \Pi$. For a fixed improvement threshold ε , we first split the data into a training set $\mathcal{D}_{\text{train}}$, a validation set $\mathcal{D}_{\text{val}}$ and a test set $\mathcal{D}_{\text{test}}$. The test set is held out and used only for the final evaluation.

On the training set, we first fit a linear regression model using only individual option terms derived from the original configuration options, e.g., without additional interaction or subgroup-derived indicator terms. Following prior work on performance-influence models [23], we use regular linear regression because it yields an explicit additive, easily readable model. We then use the fitted model to predict the performance of the configurations in the validation set $\mathcal{D}_{\text{val}}$. For each validation configuration $c_i \in \mathcal{D}_{\text{val}}$, we compute the signed residual $r_i = \hat{\Pi}(c_i) - \Pi(c_i)$. This yields the residual dataset $\mathcal{D}_r = \{(c_i, r_i)\}_{i=1}^{n_{\text{val}}}$, where n_{val} denotes the number of validation observations. Positive residuals correspond to configurations for which the current model overpredicts performance, whereas negative residuals correspond to configurations for which it underpredicts performance. To identify both types of directional prediction errors, we apply subgroup discovery twice: once on $\mathcal{D}_r$ and once on $\mathcal{D}_{-r} = \{(c_i, -r_i)\}_{i=1}^{n_{\text{val}}}$. In both cases, the signed residual value is used as the target variable. We use the mean-shift quality function $q(d) = |S_d|^\alpha \cdot (\mu_{S_d} - \mu_{\mathcal{D}})$, where S_d is the set of validation observations covered by subgroup description d , μ_{S_d} is the mean target value within the subgroup, $\mu_{\mathcal{D}}$ is the mean target value over the corresponding residual dataset, and α controls the trade-off between subgroup size and deviation magnitude.

This allows us to identify regions of the configuration space in which the current model exhibits systematic directional prediction errors. We treat such regions as potential signs of missing interaction structure, since they may correspond to parts of the configuration space

in which the model containing only individual configuration options cannot adequately capture the joint influence of multiple options.

After obtaining the highest-ranked subgroup descriptions from the positive and negative residual directions, we transform each description into a binary indicator feature. For a subgroup description d , the corresponding feature $d(c_i)$ evaluates to 1 if configuration c_i is covered by d and to 0 otherwise. Indicator features are rule based model terms rather than necessarily conventional algebraic interaction terms. We then temporarily augment the training data with each subgroup indicator feature, refit the regression model on the augmented feature space, and evaluate the resulting model on the validation set. The subgroup indicator feature that yields the larger validation improvement is selected. If this improvement is at least ε , we permanently add the corresponding indicator feature to the model. We then recompute the residuals on the validation set and repeat the process.

This procedure is iterated until either a maximum number of iterations is reached or no candidate feature improves the validation R^2 by at least the predefined threshold ε .

3.2 Research Questions

To evaluate the validity of our approach, we evaluate our model guided by two research questions. The first investigates whether subgroup discovery is capable of generating subgroup descriptions which correspond to meaningful option interactions. We consider an interaction meaningful when the combined effect of multiple options on performance is not additive with respect to the effects of the individual options. In answering this question, we also analyze under which conditions subgroup discovery is able to recover such interaction-related regions and compare the identified regions with those obtained using a standard stepwise feature-selection approach. To answer this question, we use a semi-synthetic data setup in which the ground truth model is known.

RQ1. Can subgroup discovery identify regions that correspond to performance relevant interactions among configuration options?

The second research question examines whether our approach generalizes to real-world configurable systems by evaluating its predictive performance on real-world performance data. We compare its performance against established baseline approaches, including stepwise feature selection and classification and regression trees (CART).

RQ2. To what extent does SD-guided feature selection improve predictive performance compared to plain linear regression and existing baselines?

3.3 Operationalization

Datasets We use the real-world configurable-system datasets introduced by Kaltenecker et al. [12]. Table 3.1 summarizes their domains, number of configuration options, and number

of measured configurations. The datasets are used in their encoded binary representation. Thus, categorical configuration choices are represented by separate binary features, and $|\mathcal{O}|$ denotes the number of binary configuration features used in our experiments. Because the datasets cover different domains and configuration-space sizes, they allow us to evaluate the approach across a diverse set of configurable systems.

Data preprocessing Before conducting the experiments, we parse each system’s feature model and retain only configuration features that also occur in the corresponding measurement data. XML configurations are converted into binary indicator features, and dataset-specific target columns are standardized to Performance and converted to numeric values. Empty CSV columns and configuration features that are constant across all measured configurations are removed. The resulting non-constant binary features are used for both RQ1 and RQ2. All preprocessing is performed before creating the train-validation-test splits.

Table 3.1: Configuration space characteristics of the real-world datasets before and after preprocessing.

System	Domain	$ \mathcal{O}_{\text{orig}} $	$ \mathcal{O}_{\text{used}} $	$ \mathcal{C}_m $
7z	File archive utility	44	39	68,640
BERKELEYDBC	Embedded database	18	16	2,560
DUNE	Multigrid solver	32	26	2,304
HIPACC	Image processing	54	50	13,485
JAVAGC	Garbage collector	39	32	193,536
LLVM	Compiler infrastructure	11	10	1,024
LRZIP	File archive utility	19	18	432
POLLY	Code optimizer	40	31	60,000
VP9	Video encoder	42	33	216,000
x264	Video encoder	16	13	1,152

Ground-Truth-Based Evaluation of Discovered Subgroups To assess RQ1, we require datasets with known ground-truth interaction structure. Such ground truth is typically unavailable for real-world performance measurements, because the true option interactions that influence the observed performance are unknown. We therefore construct semi-synthetic datasets based on the real datasets listed in Table 3.1. For each dataset, we retain the real configuration space, including the observed option structure and sampled configurations, but replace the measured performance values with values generated from a synthetic performance-influence model. Since the interaction terms of this generated PIM are known by construction, the resulting datasets allow us to evaluate how closely the subgroup descriptions discovered by our approach align with the seeded interactions.

Let $m \in \mathbb{N}$ denote the number of configuration options, and let $c = (o_1, \dots, o_m)$ denote a configuration. For each configuration c , we generate a synthetic performance value using a performance-influence model of the form

$$\Pi_{syn}(c) = \beta_0 + \sum_{i=1}^m \beta_i o_i + \sum_{\ell=1}^L \beta_{I_\ell} \prod_{i \in I_\ell} o_i + \varepsilon. \quad (3.1)$$

Here, $\beta_0 \in \mathbb{R}$ is an intercept term, β_i denotes the weight of configuration option o_i , L is the number of seeded interaction terms, $I_\ell \subseteq \{1, \dots, m\}$ denotes the index set of options participating in the ℓ -th seeded interaction, β_{I_ℓ} is the corresponding interaction weight, and ε is a Gaussian noise term, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

Let $\mathcal{C}_m$ denote the set of configurations of one of the datasets listed in Table 3.1. For the semi-synthetic evaluation, we use all configurations $c \in \mathcal{C}_m$ and generate their performance values using the data-generating PIM defined in Equation (3.1) instead of using the measured performance. This yields a new dataset $\mathcal{D} = \{(c, \Pi_{syn}(c)) \mid c \in \mathcal{C}_m\}$.

The seeded interactions in the synthetic data-generating model are algebraic interaction terms over binary configuration features, for example products of the form $\prod_{i \in I_\ell} o_i$. In contrast, the terms generated by our approach are subgroup-indicator features induced by rule-based descriptions. A subgroup description may refer to the same configuration options as a seeded interaction, but it does not necessarily represent the same algebraic term or cover the same region of the configuration space. For example, a seeded interaction $o_i o_j$ is active when $o_i = 1 \wedge o_j = 1$, whereas a subgroup description over the same variables could also describe a different region, such as $o_i = 0 \wedge o_j = 1$. For each interaction scenario, we predetermine the number of interactions of each order. To generate an interaction of order k , we randomly select k distinct binary configuration features. We retain the candidate only if all selected features are active together in at least five configurations; otherwise, we sample another candidate. Duplicate interactions are excluded.

We use this deliberately low minimum-support threshold to remove impossible or nearly unsupported interactions while retaining higher-order scenarios across all datasets. For example, in Dune, the maximum support of any four-way interaction is seven, meaning that a substantially higher threshold would make the four-way scenarios infeasible. Sampling is limited to 1000 attempts per required interaction. If the required number of valid interactions cannot be generated within this limit, data generation terminates with an error to make sure that the approach always terminates.

To keep the set of main effects, which describe the effects of individual configuration options, sparse and its size constant across interaction scenarios, we randomly select exactly five configuration options for each generated PIM and assign non-zero main effect coefficients only to these options. The coefficients of all remaining options are set to zero. We chose five as an empirically informed heuristic based on the RQ2 experiments with measured performance data. Across 300 Stepwise runs covering ten systems and 30 data splits, Stepwise retained an average of 3.24 singular configuration option terms. We therefore selected five as a moderately conservative value above the observed average. Holding this number constant ensures that differences between the semi-synthetic scenarios are primarily caused by their seeded interaction structures rather than by differences in the number of main effects.

For each of the five selected main effects, a sign is sampled uniformly from $\{-1, +1\}$, and its absolute magnitude is sampled uniformly from $[5, 15]$. Thus, each non-zero main effect coefficient satisfies

$$\beta_i \in [-15, -5] \cup [5, 15],$$

while all other main effect coefficients are set to zero.

Interaction coefficients are generated analogously, with their signs sampled uniformly and their absolute magnitudes sampled uniformly from $[5, 25]$. Therefore,

$$\beta_{I_\ell} \in [-25, -5] \cup [5, 25].$$

These distributions exclude non-zero coefficients close to zero.

Using the sampled coefficients, we first compute the noiseless performance of each configuration. Let $g(c)$ denote the noiseless performance of configuration c , excluding the intercept:

$$g(c) = \sum_{i=1}^m \beta_i o_i + \sum_{\ell=1}^L \beta_{I_\ell} \prod_{i \in I_\ell} o_i.$$

Here, β_i denotes the main effect coefficient of option o_i , while β_{I_ℓ} denotes the coefficient of the interaction involving the options indexed by I_ℓ .

To obtain a comparable relative noise level across systems and interaction scenarios, we scale the noise according to the variation in the generated noiseless performance values. Let σ_g denote the standard deviation of $g(c)$ across all configurations. We add independent Gaussian noise to each configuration:

$$\varepsilon_c \sim \mathcal{N}(0, (0.3\sigma_g)^2).$$

Thus, the noise standard deviation corresponds to 30% of the standard deviation of the noiseless performance values.

Finally, we add an intercept to prevent the generated performance values from becoming unrealistically small or negative. Let

$$y_{\min} = \min_{c \in \mathcal{C}_m} (g(c) + \varepsilon_c)$$

denote the smallest generated value before adding the intercept. We define

$$\beta_0 = \max\{0, 50 - y_{\min}\}.$$

Consequently, the generated performance values are shifted upwards only when their minimum would otherwise be below 50.

To keep the experimental design tractable, we hold the number of non-zero main effects, the coefficient distributions, and the relative noise level fixed across all interaction scenarios. The scenarios therefore differ in their prescribed number and order of seeded interactions, while the remaining data-generating choices remain constant.

For each of the ten datasets and five interaction scenarios, we generate semi-synthetic data using 50 random seeds, numbered 2001 to 2050. The random seed determines the selected configuration options and their coefficients, the seeded interaction terms and their coefficients, the Gaussian-noise realization, the train-validation-test partition, and

randomized operations such as tie-breaking in Stepwise. Each dataset is divided into training, validation, and test sets using a 60-20-20 split. All evaluated methods use the same partition for a given dataset, scenario, and seed. The training and validation sets are used for model construction and feature selection. After feature selection, each final model is refitted on the combined training and validation data. The test set remains held out and is used exclusively for final evaluation.

Across the five scenarios, the prescribed number and order of the seeded interactions vary, while the coefficient distribution windows and relative noise level remain fixed. Consequently, the scenarios isolate the influence of interaction structure without introducing coefficient magnitude or noise level as additional experimental factors.

Scenarios We evaluate the semi-synthetic data-generation process under several interaction scenarios. The considered interaction structures are summarized in Table 3.2. For each scenario, we generate synthetic performance values using the data-generating PIM defined in Equation 3.1.

Table 3.2: Semi-synthetic interaction scenarios.

Scenario	Pairwise interactions	Three-way interactions	Four-way interactions
S ₁	1	0	0
S ₂	2	1	0
S ₃	3	0	0
S ₄	1	1	1
S ₅	2	2	1

Previous work indicates that performance-relevant interactions are frequently pairwise, whereas interactions involving four or more options are less common and generally have a smaller effect on prediction accuracy [14, 24]. Our scenarios therefore give pairwise interactions the greatest overall representation while still including three- and four-way interactions.

The central purpose of the scenario design is to distinguish the effect of the number of interactions from the effect of their order. Three of the five scenarios—S₂, S₃, and S₄ contain exactly three seeded interactions but differ in their composition. S₂ contains two pairwise interactions and one three-way interaction, S₃ contains three pairwise interactions, and S₄ contains one pairwise, one three-way, and one four-way interaction. Comparing these three scenarios allows us to examine how the presence of higher-order interactions affects recovery while holding the total number of interactions constant.

S₁ and S₅ serve as reference cases with different interaction densities. S₁ contains a single pairwise interaction and represents the sparsest interaction structure considered. It allows us to examine whether an individual interaction can be recovered when no additional interaction effects are present. In contrast, S₅ contains five interactions: two pairwise, two three-way, and one four-way, and represents the densest and most compositionally varied scenario. Together, the five scenarios allow us to investigate how interaction recovery changes with both the number of interactions and their orders.

Notation and similarity measures. To assess similarity between sets, we use the Jaccard similarity [22], defined for two sets A and B as

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

For a discovered subgroup description $d \in \mathcal{R}$, let

$$\text{Ext}(d) = \{c \in \mathcal{C}_m \mid d(c) = 1\}$$

denote the set of sampled configurations that satisfy d , and are therefore covered by the subgroup description.

For any algebraic interaction term t , whether seeded in the ground-truth PIM or selected by Stepwise, we define its active region as

$$\text{Act}(t) = \left\{ c \in \mathcal{C}_m \mid \prod_{i \in t} o_i = 1 \right\}.$$

Thus, $\text{Act}(t)$ contains the sampled configurations in which all binary features participating in t are enabled. This definition is suitable for our setting because the interaction terms are generated over the encoded binary configuration features. Categorical configuration choices are represented by one-hot encoded binary features, while Boolean options are represented directly as binary features.

The region-based similarity between a discovered description d and an interaction term t is defined as

$$J_{\text{reg}}(d, t) = J(\text{Ext}(d), \text{Act}(t)) = \frac{|\text{Ext}(d) \cap \text{Act}(t)|}{|\text{Ext}(d) \cup \text{Act}(t)|}.$$

The region-based alignment quality of a discovered subgroup description d is then defined as its maximum region-based similarity to any seeded interaction:

$$Q_{\text{reg}}(d) = \max_{I_\ell \in \mathcal{I}} J_{\text{reg}}(d, I_\ell).$$

A high value of $Q_{\text{reg}}(d)$ indicates that the subgroup description covers a set of configurations that closely matches the active region of at least one seeded interaction.

Conversely, to assess the recovery of seeded interactions, we compute for each $I_\ell \in \mathcal{I}$ its maximum region-based similarity to any discovered subgroup description:

$$R_{\text{reg}}(I_\ell) = \max_{d \in \mathcal{R}} J_{\text{reg}}(d, I_\ell).$$

A high value of $R_{\text{reg}}(I_\ell)$ indicates that at least one discovered subgroup closely captures the active region of the seeded interaction I_ℓ .

We evaluate Stepwise analogously by computing, for each seeded interaction I_ℓ , the Jaccard similarity between its active region and the active region of every interaction term selected by Stepwise, and taking the maximum. If Stepwise selects no interaction term or SD-guided feature selection discovers no subgroup rule, the corresponding recovery score is zero.

Note on Numeric Configuration Options The region-based alignment metric is defined for the encoded binary feature representation used in our experiments. For raw numeric configuration options, the notion of an interaction being simply active or inactive is less direct. In such cases, the evaluation could be extended by comparing discovered subgroup regions with high-impact regions, i.e., configurations in which the numeric interaction contribution is particularly large.

Predictive Performance (RQ2) Evaluation To evaluate the predictive performance of SD-guided feature selection, we compare it against three baselines. First, we use plain linear regression without interaction or subgroup-derived terms. This baseline isolates the contribution of the additional terms constructed by the proposed approach.

Second, we use an interaction-selection baseline inspired by Siegmund et al. [23], which is conceptually the closest related method. This baseline constructs candidate interaction terms over the encoded binary configuration features and considers interactions up to order four. Candidate terms are selected according to their improvement on the validation set.

Third, we include a classification and regression tree (CART) as a nonlinear baseline. Tree-based models can represent interactions implicitly through sequences of splits without requiring explicit interaction terms.

For each dataset in Table 3.1, we repeat the evaluation using 30 random seeds, numbered 9001 to 9030. For each seed, the available measured configurations are randomly divided into three disjoint subsets: $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$. We use a 60-20-20 split, assigning 60% of the configurations to training, 20% to validation, and 20% to testing. All methods use the same partition for a given seed, enabling paired comparisons under identical data conditions.

SD-guided and Stepwise use the training set to fit candidate models and the validation set for feature selection. For SD-guided feature selection, the validation set additionally provides the residuals on which subgroup discovery operates. After feature selection, both methods are refitted on the combined training and validation data, $\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}}$, using their selected feature sets.

Plain linear regression and CART do not perform validation-based model selection under their fixed hyperparameter settings. They are therefore fitted directly on the combined training and validation data. All final models are evaluated on the held-out test set, which does not influence model fitting, feature selection, or hyperparameter selection.

The random seed determines the data partition, Stepwise tie-breaking, and the `random_state` used by CART. The same seed is used for all methods within a repetition. Repeating the evaluation across 30 seeds reduces the influence of individual data partitions and randomized decisions on the reported results. The complete implementation and hyperparameter settings are reported in Section 3.4.

To assess predictive performance, we use R^2 and MAE as our primary evaluation metrics, as introduced in Section 2.3. R^2 quantifies the proportion of variance in the observed performance values explained by the model, while MAE measures the average prediction error in the original unit of the target. We additionally report RMSE as a supplementary robustness measure because it assigns greater weight to large prediction errors. The complete RMSE results are provided in Appendix A.1.

Based on these metrics, we compare SD-guided feature selection with the selected baselines on each dataset and examine whether it yields consistent improvements in predictive accuracy across different systems. For each dataset and method, we summarize the test set performance across the 30 seeds using box plots that show the median, interquartile range, and potential outliers.

Because all methods use the same data partition for a given seed, we additionally conduct paired comparisons between SD-guided feature selection and each baseline. For each dataset, we report the mean paired performance difference across the 30 data splits and the number of splits in which SD-guided performs better than the respective baseline.

To assess whether the paired performance differences are consistent across systems, we follow the recommendations of Demšar [4] and apply two-sided Wilcoxon signed-rank tests. For each dataset, we first average the performance differences across the 30 paired data splits. We then test the resulting ten dataset-level differences using a significance level of $\alpha = 0.05$. Holm’s correction is applied to account for the three comparisons between SD-guided feature selection and the baselines within each metric.

In addition to predictive performance, we report model complexity and runtime. For the regression-based methods, model complexity is measured by the number of terms in the final model. We also examine the validation- R^2 improvement across the SD-guided feature-selection iterations. This convergence analysis allows us to assess how quickly the method obtains most of its final improvement and whether the maximum number of iterations could be reduced without substantially decreasing predictive performance.

3.4 Implementation Details and Hyperparameters

Table 3.3: Default subgroup discovery settings used in the proposed approach.

Parameter	Default value	Purpose
Search strategy	Beam search	Searches the subgroup description space heuristically
Beam width	25	Retains candidate descriptions per search level
Maximum description length	4	Limits the number of selectors per subgroup description
Quality function	Mean-shift quality	Scores subgroups by residual exceptionality
Quality-function parameter	$\alpha = 0.5$	Balances subgroup size and residual deviation

We use beam search as a heuristic search strategy because the space of possible subgroup descriptions is too large to enumerate exhaustively. The subgroup discovery settings are chosen as conservative defaults that balance search expressiveness, interpretability, and computational cost. The beam width is set to 25, which provides a moderate search breadth: larger beams retain more candidate subgroup descriptions at each search level but increase runtime, whereas smaller beams reduce runtime at the risk of pruning promising

descriptions too early. We set the maximum subgroup description length to four. This allows the search to detect higher-order residual patterns while keeping the search space bounded. At the same time, longer subgroup descriptions tend to define more specific regions of the configuration space and may therefore be more susceptible to capturing noise rather than systematic behavior. Prior work on performance-influence models indicates that higher-order interactions involving four or more configuration options have little effect on prediction error while increasing computational cost and model size [14]. Therefore, depth four serves as a conservative upper bound. It allows the discovery of potentially relevant higher-order structures, while the subsequent validation based selection step stops subgroup terms from being added unless they improve predictive performance. Finally, we set the subgroup-size weighting parameter to $\alpha = 0.5$ to balance residual deviation and subgroup coverage. This provides a compromise between discovering localized residual patterns and identifying subgroups large enough to support stable model improvement.

Table 3.4: Default settings of the iterative feature-selection procedure.

Setting	Default value	Purpose
Selection metric	Validation R^2	Measures whether a candidate feature improves validation performance
Maximum iterations	$T = 10$	Limits the number of subgroup-derived indicator features added to the model
Improvement threshold	$\varepsilon = 0.01$	Minimum validation improvement required to add a candidate feature

We limit the number of feature-selection iterations to $T = 10$. This bounds the number of subgroup-derived indicator features that can be added to the model. The limit constrains model growth and keeps the added rule based terms inspectable, reduces the risk of overfitting to the validation set, and avoids adding many subgroup features that describe similar parts of the configuration space. The choice of ten iterations is in line with prior empirical observations that, in many configurable systems, a small number of low-order model terms can already provide accurate predictions, while additional terms often yield only marginal improvements [14]. If no candidate feature improves validation performance by at least ε , the algorithm stops before reaching this limit. Since validation performance is measured using R^2 , we set the improvement threshold to $\varepsilon = 0.01$. This means that a subgroup-derived indicator feature is added only if it increases validation R^2 by at least 0.01, corresponding to an additional 1 percentage point of explained variance. This threshold filters out very small improvements that may reflect noise, while still allowing features with a measurable predictive benefit to be selected.

The reported settings define the default experimental configuration.

The approach is implemented in Python using ordinary least squares linear regression via `LinearRegression` from `scikit-learn`, and the `pysubgroup` library for subgroup discovery.

We use the stepwise learning implementation from the VaRA feature library as a baseline.¹ We adapt the implementation to the train-validation-test protocol used throughout our evaluation. Whereas the original implementation fits and evaluates candidate models on the

¹ [VaRA feature library, stepwise learning implementation.](#)

same data using MAPE, we fit candidate models on the training set and evaluate candidate additions on a separate validation set. This reduces optimistic feature selection and ensures that the test set remains reserved for final evaluation. We further replace MAPE with $1 - R^2$. This makes the selection objective consistent with the validation criterion used by SD-guided feature selection and provides a scale-independent criterion across systems with different performance ranges. Moreover, unlike MAPE, $1 - R^2$ does not involve division by the observed target and therefore remains well-defined when performance values are zero or close to zero. We consider interactions up to order four, allowing pairwise, three-way, and four-way interaction terms while bounding the combinatorial growth of the candidate space. The forward-selection procedure stops when the improvement in validation R^2 falls below 0.01, or when the validation loss $1 - R^2$ falls below 0.01. After forward selection, a backward-selection stage iteratively removes terms whose removal does not increase the validation loss. The final model is then fitted on the combined training and validation data using the selected terms.

We implement CART using scikit-learn’s `DecisionTreeRegressor`. We restrict the maximum tree depth to four to bound model complexity and to keep the maximum number of successive splits comparable to the maximum subgroup-description length used by SD-guided feature selection. We set `min_samples_split` to 4 and `min_samples_leaf` to 2, preventing the tree from creating splits or leaves based on very small numbers of configurations.

Because these hyperparameters are fixed, CART does not require validation-based model selection. The model is fitted on the combined training and validation data and evaluated on the held-out test set. Its `random_state` is set to the seed of the corresponding experimental run to ensure reproducibility.

Evaluation

This chapter evaluates the proposed SD-guided feature-selection approach, referred to as SD-GUIDED throughout this chapter. First, we examine whether its subgroup-derived features correspond to the seeded interaction structure in the semi-synthetic experiments and to what extent the seeded interactions are recovered. Second, we investigate whether SD-GUIDED improves predictive performance on measured performance data relative to plain linear regression and the other baseline methods.

For readability, we refer to plain linear regression as LINEAR and stepwise feature selection as STEPWISE throughout this chapter.

Section 4.1 presents the results for the two research questions. Section 4.2 discusses the implications of these findings, while Section 4.3 examines threats to the validity of the evaluation.

4.1 Results

This section presents the results for the two research questions. RQ1 examines the correspondence between the subgroup rules discovered by SD-GUIDED and the seeded interactions in the semi-synthetic experiments. RQ2 evaluates the predictive performance of SD-GUIDED on measured real world performance data relative to the baseline methods.

The semi-synthetic interaction scenarios used for RQ1 are defined in Table 3.2.

4.1.1 RQ1: Alignment of Discovered Subgroup Rules and Recovery of Interactions

To answer RQ1, we examine whether the subgroup rules identified by SD-guided correspond to the seeded interaction structure and how their recovery compares with Stepwise feature selection. We first analyze the alignment of the discovered rules with the ground truth, before reversing the perspective and examining how well the seeded interactions are recovered. We then investigate interaction order and support as factors that may affect recovery.

Table 4.1: Percentage of the 50 semi-synthetic runs in which SD-guided discovered no rules, darker cells are worse.

Dataset	S1	S2	S3	S4	S5
7z	18%	2%	2%	6%	2%
BerkeleyDBC	4%	0%	0%	0%	0%
Dune	6%	0%	0%	2%	0%
HIPAcc	44%	12%	6%	32%	12%
LRZIP	28%	12%	8%	12%	4%
JavaGC	6%	0%	0%	0%	0%
LLVM	12%	2%	2%	0%	0%
Polly	2%	2%	0%	0%	0%
VP9	6%	0%	0%	0%	0%
x264	12%	0%	0%	2%	0%

Alignment and runs without rules. We first evaluate how closely the regions described by the discovered subgroup rules correspond to the active regions of the seeded interactions. For each discovered rule, the alignment score measures its closest correspondence to any seeded interaction. A high score indicates that the rule describes a region similar to the active region of a known interaction, whereas a low score indicates that the two regions differ substantially. Alignment therefore assesses whether the discovered rules reflect the seeded interaction structure; it does not, however, determine whether every seeded interaction is recovered. Because an alignment score is defined only when at least one rule is discovered, runs without a discovered rule are excluded from this analysis.

Table 4.1 reports the percentage of the 50 semi-synthetic runs in which no rule was discovered for each dataset and scenario. S1, which contains a single pairwise interaction, has the highest or joint highest no rule rate for every dataset. This suggests that rule discovery is most difficult in the sparsest scenario.

The no rule rate also varies substantially between datasets. In S1, it ranges from 2% for Polly to 44% for HIPAcc. HIPAcc and LRZIP continue to have comparatively high no rule rates in the other scenarios. Overall, whether a rule is discovered appears to depend on both the interaction scenario and the underlying dataset.

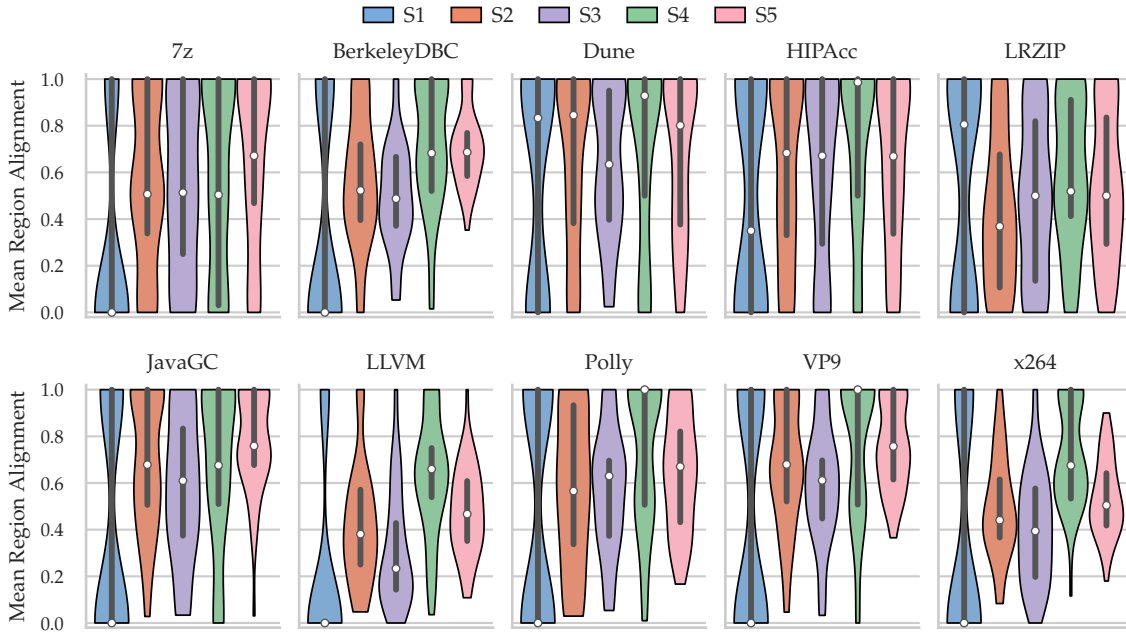


Figure 4.1: Distributions of per run mean region alignment scores across datasets and interaction scenarios. Higher scores indicate closer correspondence between discovered subgroup rules and seeded interaction regions.

Alignment across datasets. For each run, we averaged the region-alignment scores across all discovered subgroup rules. Figure 4.1 shows their distributions by dataset and scenario; higher scores indicate closer correspondence with the active regions of the seeded interactions, while violin width indicates the concentration of runs.

The S1 distributions, representing a single pairwise interaction, often concentrate near zero and, for several datasets, also near one. This indicates polarization between low and high mean alignment. Although S1 medians are generally low, Dune and LRZIP exhibit comparatively high alignment.

S2 and S3 both contain three seeded interactions but differ in their composition. S2 contains two pairwise interactions and one three-way interaction, whereas S3 contains three pairwise interactions. Both scenarios have more centrally distributed scores, and neither S2 nor S3 consistently yields higher alignment across datasets. Thus, replacing one pairwise interaction with a three-way interaction does not lead to a consistent change in alignment. Absolute alignment nevertheless varies by dataset: Dune is comparatively high in both scenarios, whereas LLVM and x264 have lower medians. LRZIP also has lower alignment in S2 than in S1. S4 and S5 both contain pairwise, three-way, and four-way interactions. S4 has one interaction of each order, while S5 has five interactions in total. S4 usually has the highest median alignment, especially for Dune, HIPAcc, Polly, and VP9. S5 also has relatively high alignment, but performs worse than S4 on several datasets. Overall, both scenarios include low-alignment runs, and alignment varies considerably across datasets rather than improving consistently with the number or order of seeded interactions.

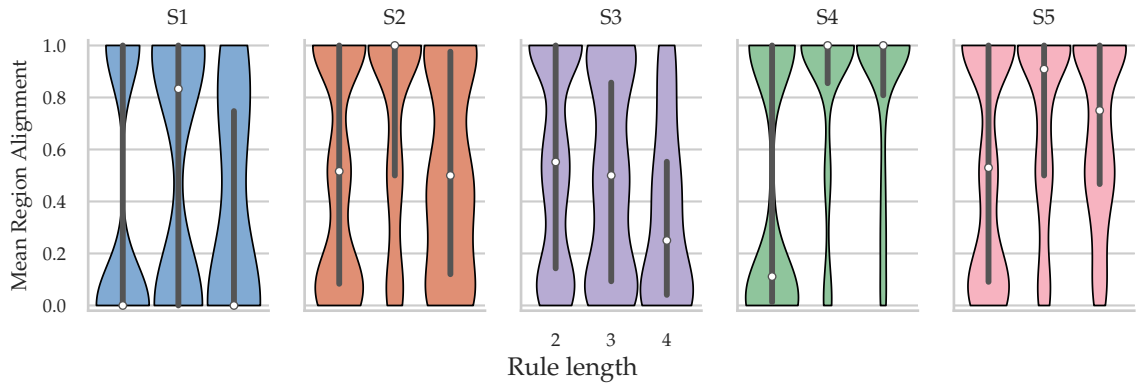


Figure 4.2: Distributions of per run mean region alignment scores across discovered-rule lengths and scenarios.

Alignment by rule length. We next examine whether alignment is related to the length of the discovered rules. Rule length is the number of selectors in a rule. For each dataset, scenario, run, and rule length, we calculate the mean alignment across all rules of that length. Only runs containing at least one rule of the corresponding length are included.

Figure 4.2 shows the resulting distributions across datasets and runs. Each panel represents one interaction scenario, while the horizontal axis indicates the number of selectors in a discovered rule.

Although S1 and S3 contain only pairwise interactions, they show different patterns. In S1, three selector rules have the highest median alignment. In S3, two selector rules have the highest median, and alignment decreases as rule length increases.

S2, S4, and S5 also contain higher-order interactions. Three selector rules have the highest median alignment in S2 and S5. In S4, both three and four selector rules have high median alignment, while two selector rules have a much lower median. However, four selector rules generally have lower alignment than three selector rules in the other scenarios. The wide distributions also show considerable variation between datasets and runs. Overall, there is no consistent relationship between rule length and alignment.

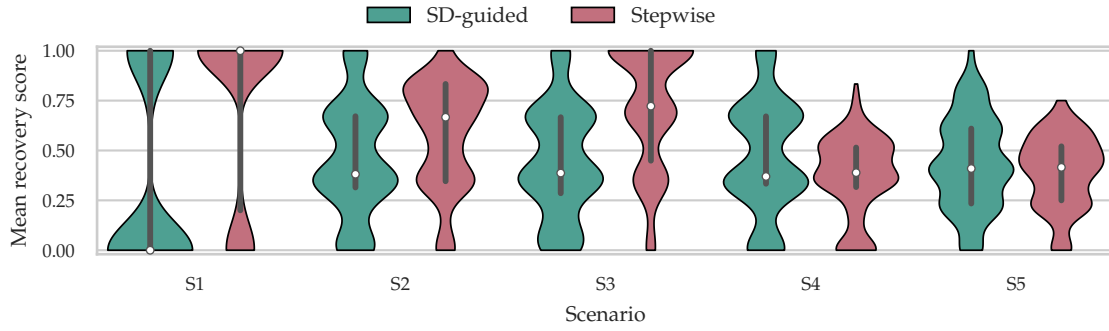


Figure 4.3: Distributions of per-run mean recovery scores for SD-GUIDED and STEPWISE across interaction scenarios.

Recovery across scenarios. In the previous paragraphs, we examined whether each discovered rule corresponds to a seeded interaction. Recovery considers the opposite perspective: for each seeded interaction, it asks whether the method selected a model term that represents it. This distinction is important because a method may discover a few well aligned rules while still missing many of the seeded interactions.

For each seeded interaction, we use the highest recovery score achieved by any model term selected by SD-GUIDED or STEPWISE. A high score means that at least one selected term closely represents the interaction. If the method selects no corresponding term, the interaction receives a recovery score of zero.

For each experimental run, we then calculate the mean recovery score across all seeded interactions. Figure 4.3 shows the distributions of these per-run mean recovery scores across datasets and seeds, grouped by scenario and method. The violin width, median point, and interquartile-range bar follow the same conventions as in Figure 4.1.

A clear difference between the scenarios is visible. In S1 and S3, which contain exclusively pairwise interactions, Stepwise achieves substantially higher median recovery than SD-GUIDED. STEPWISE also achieves higher recovery in S2, although the difference between the methods is smaller. In S4 and S5, which contain both pairwise and higher-order interactions, the recovery distributions of the two methods are more similar.

However, the scenario level results combine interactions of different orders. Because all scenarios contain pairwise interactions, and because the per-run mean gives equal weight to every seeded interaction, the overall recovery score is influenced by the interaction composition of each scenario. It therefore does not isolate how recovery changes with interaction order. We examine this distinction separately in Figure 4.4.

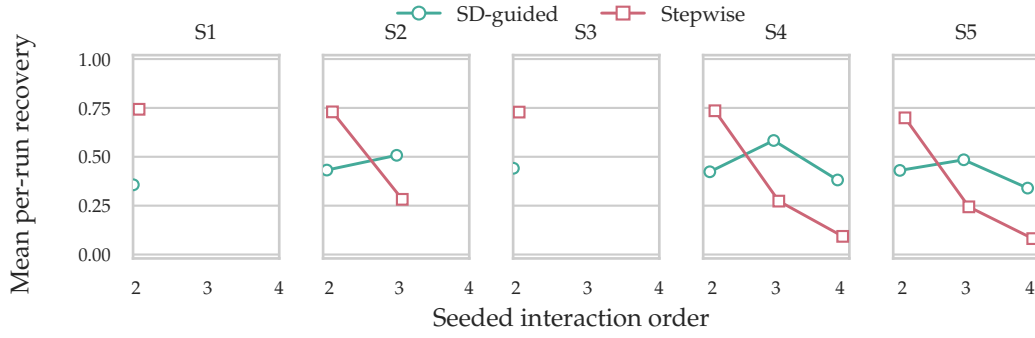


Figure 4.4: Mean recovery by seeded interaction order, scenario, and method. Recovery is first averaged within each experimental run and interaction order, and then across datasets and seeds. Lines connect the interaction orders available within each scenario.



Figure 4.5: Mean recovery by training set support, interaction order, and method. Support is the percentage of training configurations in which a seeded interaction is active. Seeded interactions are grouped into the fixed support intervals shown on the horizontal axis. Recovery is first averaged within each experimental run and support interval, and then across datasets, scenarios, and seeds. Lines connect adjacent support intervals.

Recovery by interaction order. Figure 4.4 reveals a clear difference between the two methods. STEPWISE achieves higher mean recovery for pairwise interactions in every scenario. For three- and four-way interactions, however, SD-GUIDED consistently achieves higher mean recovery than Stepwise.

STEPWISE recovery decreases as interaction order increases in all scenarios containing multiple interaction orders. SD-GUIDED does not exhibit the same monotonic pattern. Instead, three way interactions have the highest mean recovery for SD-GUIDED in each scenario in which they occur, while recovery decreases again for four-way interactions.

Recovery by support. Figure 4.5 shows a generally positive relationship between training set support and mean recovery. For pairwise interactions, Stepwise recovery increases consistently across the support intervals and exceeds SD-GUIDED recovery once support is

above 1%. SD-GUIDED recovery also initially increases with support, reaching its highest value in the 2.5-5% interval, but decreases again in the highest-support interval.

For three- and four-way interactions, both methods achieve progressively higher mean recovery as support increases. SD-GUIDED has higher mean recovery than Stepwise in every support interval for these interaction orders. Its largest increase occurs between the lowest support intervals, after which the improvement becomes more gradual. STEPWISE also improves with increasing support, but remains below SD-GUIDED throughout. Overall, the association between support and recovery is clearest for higher-order interactions and for pairwise interactions recovered by STEPWISE.

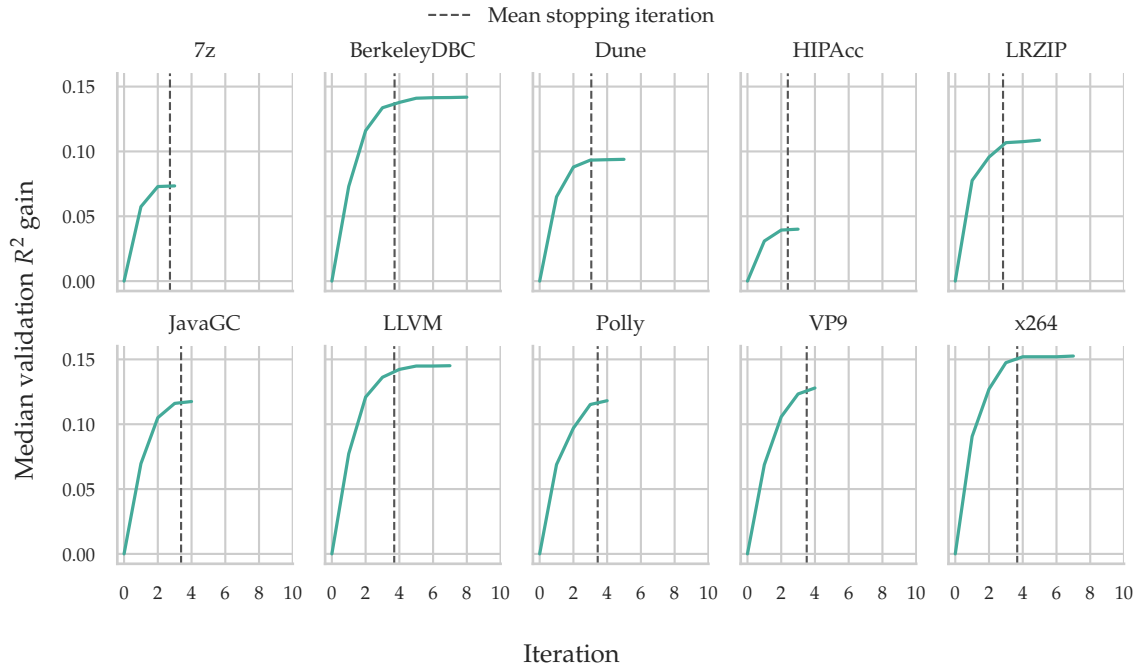


Figure 4.6: Median cumulative improvement in validation R^2 relative to the initial regression model across SD-GUIDED iterations. Each panel represents one dataset and summarizes 250 runs across five interaction scenarios and 50 seeds. For runs that terminate early, the final retained validation R^2 is carried forward when computing the median. Each curve is displayed only up to the first iteration at which it reaches its final plateau. Dashed vertical lines indicate the mean stopping iteration.

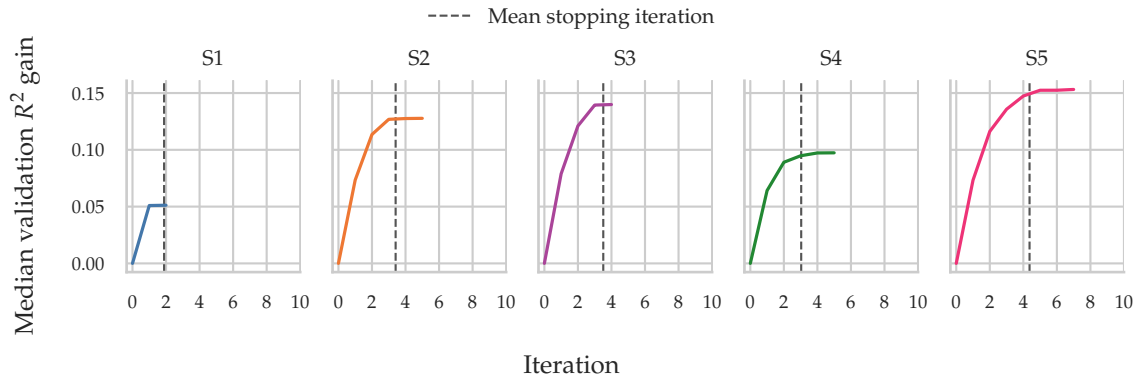


Figure 4.7: Median cumulative improvement in validation R^2 relative to the initial regression model across SD-GUIDED iterations. Each panel represents one interaction scenario and summarizes 500 runs across ten datasets and 50 seeds. For runs that terminate early, the final retained validation R^2 is carried forward when computing the median. Each curve is displayed only up to the first iteration at which it reaches its final plateau. Dashed vertical lines indicate the mean stopping iteration.

Convergence behavior. The dataset-specific convergence curves in Figure 4.6 have broadly similar shapes: most of the improvement occurs during the first few iterations, after which

the validation R^2 gain levels off. Nevertheless, the magnitude of the final improvement differs across datasets.

The scenario-level curves in Figure 4.7 reveal a clearer difference in convergence speed. S_1 , which contains one seeded interaction, stabilizes earliest. S_2 , S_3 , and S_4 each contain three seeded interactions and show similar convergence behavior despite differing in interaction order. S_5 , which contains five seeded interactions, requires the most iterations before its improvement levels off. Within the evaluated scenarios, convergence speed therefore appears to be more closely associated with the number of seeded interactions than with their order.

Table 4.2: Median model size and composition across 250 semi-synthetic runs per dataset. Terms denotes the total number of regression terms, Rules the number of subgroup rules added by SD-guided, and Interactions the number of interaction terms retained by Stepwise.

Dataset	Linear	SD-guided		Stepwise	
	Terms	Terms	Rules	Terms	Interactions
7z	39.0	41.0	2.0	7.0	1.0
BerkeleyDBC	16.0	19.0	3.0	9.0	3.0
Dune	26.0	28.0	2.0	7.0	2.0
HIPAcc	50.0	51.0	1.0	7.0	1.0
LRZIP	18.0	20.0	2.0	8.0	3.0
JavaGC	32.0	34.0	2.0	8.0	2.0
LLVM	10.0	13.0	3.0	9.0	4.0
Polly	31.0	33.5	2.5	9.0	2.0
VP9	33.0	36.0	3.0	8.0	2.0
x264	13.0	16.0	3.0	9.0	4.0

Model Complexity. Table 4.2 summarizes median model size and composition across the semi-synthetic experiments. SD-GUIDED extends the initial linear model only slightly: it retains all encoded configuration-option terms and adds only a few subgroup rules. The rule set therefore remains compact, although the full model can be large for datasets with many configuration options. STEPWISE produces substantially smaller models by retaining only a subset of the original configuration option terms and interactions. Most retained terms represent individual options, while interactions form only a small share. Thus, although both methods include only a few interaction terms, they differ substantially in how many original configuration option terms they preserve.

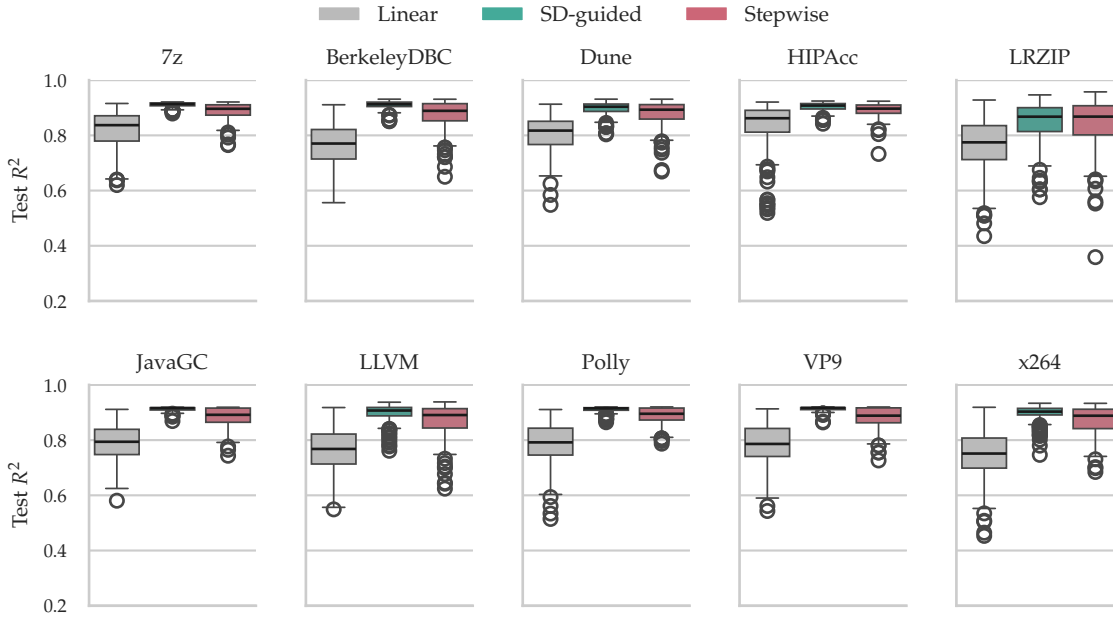


Figure 4.8: Distributions of held out test set R^2 scores for LINEAR, SD-GUIDED, and STEPWISE across datasets. Each panel represents one dataset and summarizes 250 runs per method across five interaction scenarios and 50 seeds. Boxes indicate the interquartile range, and the line inside each box marks the median. Potential outliers are shown as hollow circles.

Predictive performance. The previous analyses evaluated how accurately the selected model terms capture the seeded interactions. We now examine whether incomplete recovery is associated with lower predictive performance. We measure predictive performance using held-out test set R^2 , which does not depend on the scale and allows comparisons across semi-synthetic datasets with different performance scales.

Figure 4.8 shows that SD-GUIDED and STEPWISE generally achieve higher median test set R^2 than LINEAR across all datasets. The two feature-selection methods are broadly comparable, although SD-GUIDED has a slightly higher median in most dataset panels. LRZIP is the main exception, with both methods producing nearly identical median scores.

Figure 4.9 provides further context for the structural recovery results. In S1, SD-GUIDED and STEPWISE achieve nearly identical predictive performance despite the lower pairwise recovery of SD-GUIDED. In the remaining scenarios, SD-GUIDED is comparable to or better than STEPWISE, with the clearest differences occurring in S4 and S5.

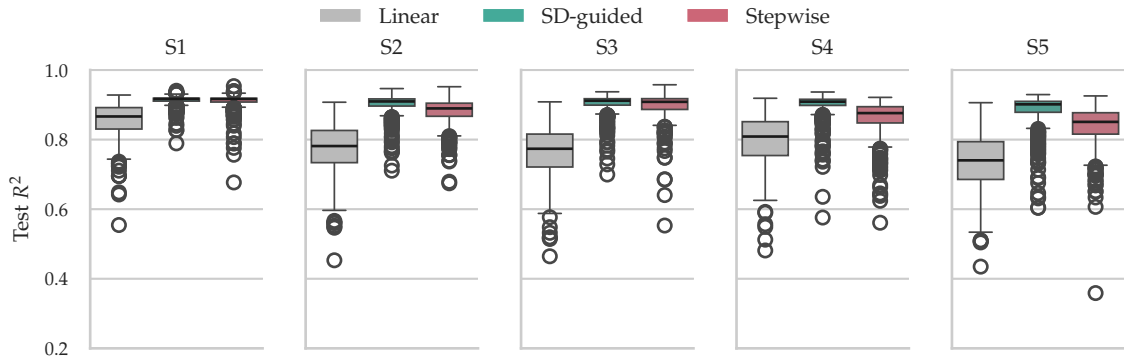


Figure 4.9: Distributions of held out test set R^2 scores for LINEAR, SD-GUIDED, and STEPWISE across interaction scenarios. Each panel represents one scenario and summarizes 500 runs per method across ten datasets and 50 seeds. Boxes indicate the interquartile range, and the line inside each box marks the median. Potential outliers are shown as hollow circles.

Summary. Taken together, these results answer RQ₁ by showing that the discovered subgroup rules correspond only partially to the exact seeded interaction structure. Alignment varies substantially across datasets and scenarios and exhibits no consistent relationship with rule length. The recovery analysis reveals complementary strengths: STEPWISE recovers pairwise interactions more effectively, whereas SD-GUIDED achieves higher recovery for three- and four-way interactions, particularly as their training-set support increases. Despite these differences in structural recovery, the methods achieve broadly comparable predictive performance. Exact recovery of the seeded interactions is therefore not necessary for discovering predictively useful terms.

4.1.2 RQ2: Predictive Performance

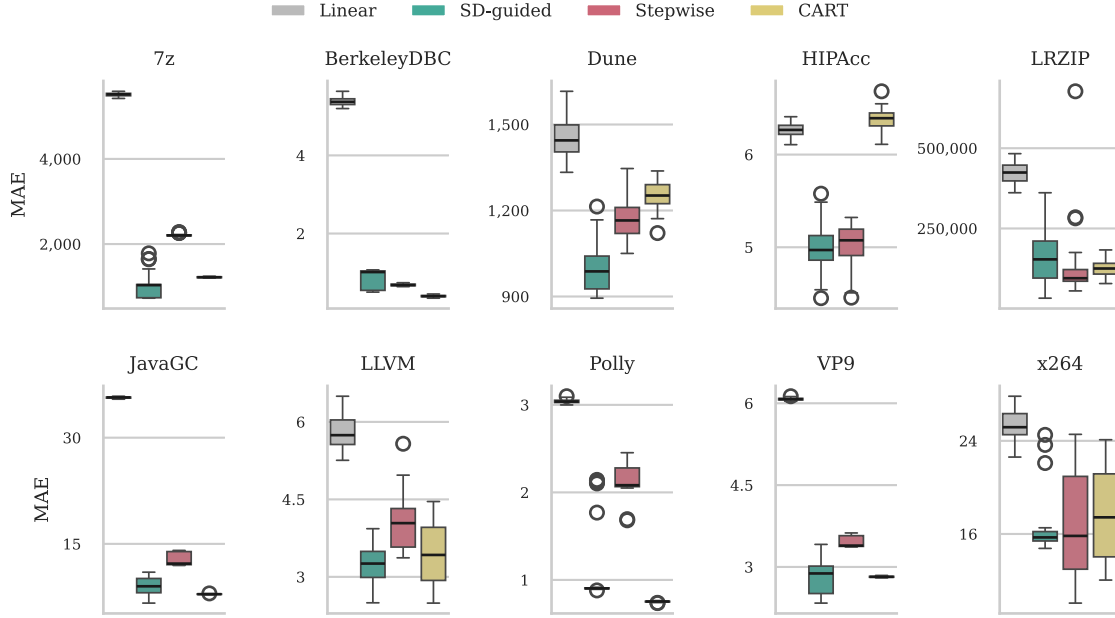


Figure 4.10: Test set MAE distributions across 30 random seeds, shown separately for each dataset and method. Lower values indicate better predictive accuracy.

Table 4.3: Test set MAE comparison. Each cell shows SD-guided’s average percentage reduction in MAE relative to the method named above it across the same 30 data splits. Positive values favor SD-guided; parentheses show how often it achieved the lower MAE. Green indicates a majority of wins and red a majority of losses.

Dataset	Linear	Stepwise	CART
7z	+81.4% (30/30)	+53.6% (30/30)	+16.0% (23/30)
BerkeleyDBC	+83.9% (30/30)	-26.4% (10/30)	-116.7% (0/30)
Dune	+31.5% (30/30)	+14.4% (28/30)	+20.3% (30/30)
HIPAcc	+20.1% (30/30)	+0.1% (15/30)	+21.5% (30/30)
JavaGC	+75.0% (30/30)	+29.8% (30/30)	-12.9% (4/30)
LLVM	+43.9% (30/30)	+18.8% (29/30)	+3.4% (17/30)
LRZIP	+61.0% (30/30)	-64.2% (8/30)	-32.9% (10/30)
Polly	+64.0% (30/30)	+48.4% (27/30)	-45.3% (0/30)
VP9	+53.3% (30/30)	+17.8% (30/30)	-0.5% (10/30)
x264	+35.2% (30/30)	-6.3% (15/30)	+3.3% (14/30)

To answer RQ2, we assess whether SD-guided feature selection improves predictive performance relative to plain linear regression and the established `STEPWISE` and `CART` baselines. We first compare predictive accuracy using R^2 and MAE, and then examine the statistical significance of performance differences, model complexity, convergence, and runtime. Supplementary results based on RMSE are reported in Appendix A.1 as they mostly mirror the MAE results.

MAE Results. Figure 4.10 presents the distribution of test-set MAE across the 30 data splits for each dataset and method, while Table 4.3 provides paired comparisons between `SD-GUIDED` and each baseline using runs with the same dataset and random seed. The table reports the average percentage reduction in MAE achieved by `SD-GUIDED` and, in parentheses, the number of paired runs in which it obtains the lower MAE. Positive values favor `SD-GUIDED`.

The clearest result is the consistent improvement over linear regression. `SD-GUIDED` achieves a lower median MAE on all ten datasets and a lower MAE in all 30 paired runs for every dataset. The mean reduction ranges from 20.1% to 83.9%. The improvement is therefore robust across data splits rather than being driven by a small number of favorable runs, indicating that the subgroup-derived features capture performance relationships that are not adequately represented by the initial linear model.

The comparison with `STEPWISE` is more dataset dependent. `SD-GUIDED` achieves the lower median MAE on eight datasets and wins the majority of paired runs on six. `BerkeleyDBC` and `LRZIP` clearly favor `STEPWISE`, while `HIPAcc` and `x264` are evenly split by win frequency. On `x264`, the average paired result nevertheless favors `STEPWISE`. Overall, the results do not indicate a consistent advantage for either feature selection strategy.

The comparison with `CART` is similarly heterogeneous. Each method achieves the lower median MAE on five datasets. In the paired comparison, `SD-GUIDED` wins the majority of runs on four datasets, including all 30 runs on `Dune` and `HIPAcc`, whereas `CART` wins the majority on the remaining six. The strongest advantages for `CART` occur on `BerkeleyDBC` and `Polly`. On `x264`, `SD-GUIDED` wins fewer runs but achieves a positive mean reduction, illustrating that win frequency and average effect magnitude need not agree. Thus, neither method shows a consistent advantage across systems.

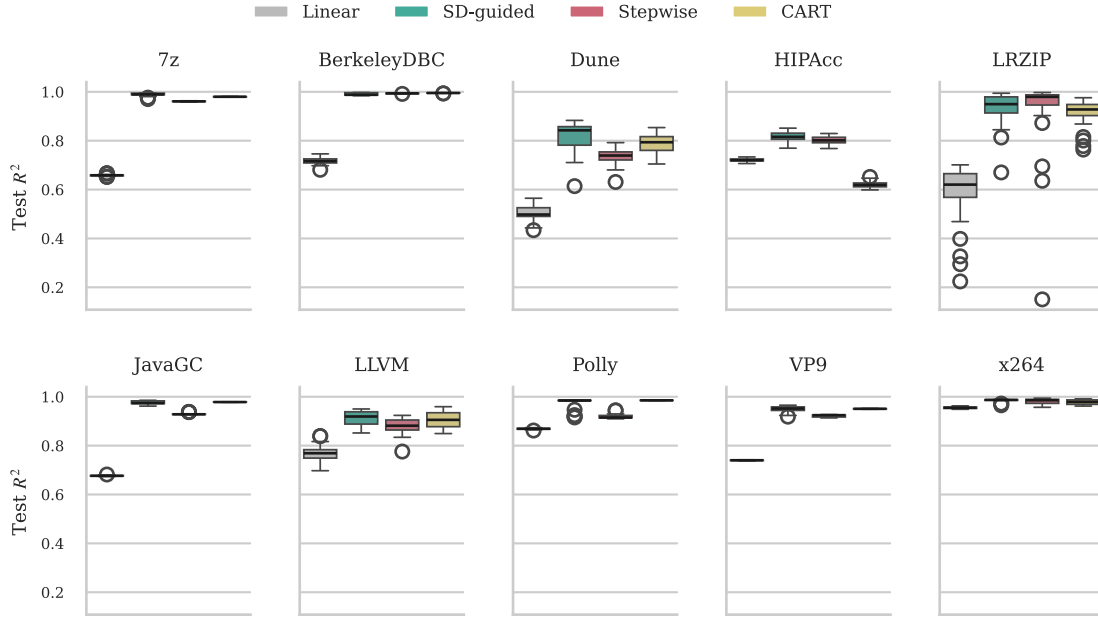


Figure 4.11: Test set R^2 distributions across 30 random seeds, shown separately for each dataset and method. Higher values indicate better predictive accuracy.

Table 4.4: Test set R^2 comparison. Each cell shows the average difference between SD-guided and the method named above it across the same 30 data splits. Positive values favor SD-guided; parentheses show how often it achieved the higher R^2 . Green indicates a majority of wins and red a majority of losses.

Dataset	Linear	Stepwise	CART
7z	+0.330 (30/30)	+0.028 (30/30)	+0.009 (27/30)
BerkeleyDBC	+0.273 (30/30)	-0.003 (10/30)	-0.005 (10/30)
Dune	+0.316 (30/30)	+0.083 (26/30)	+0.032 (22/30)
HIPAcc	+0.096 (30/30)	+0.015 (21/30)	+0.197 (30/30)
JavaGC	+0.299 (30/30)	+0.046 (30/30)	-0.003 (10/30)
LLVM	+0.143 (30/30)	+0.032 (28/30)	+0.006 (14/30)
LRZIP	+0.344 (30/30)	+0.007 (11/30)	+0.016 (20/30)
Polly	+0.106 (30/30)	+0.055 (30/30)	-0.011 (0/30)
VP9	+0.209 (30/30)	+0.026 (29/30)	-0.002 (18/30)
x264	+0.030 (30/30)	+0.003 (17/30)	+0.007 (21/30)

R^2 Results. Figure 4.11 presents the distribution of test-set R^2 across the 30 data splits for each dataset and method, while Table 4.4 provides paired comparisons between SD-GUIDED and each baseline using runs with the same dataset and random seed. The table reports the average difference in R^2 across the 30 paired runs and, in parentheses, the number of

runs in which SD-GUIDED achieves the higher R^2 . Positive values favor SD-GUIDED, whereas negative values favor the respective baseline.

The clearest result is again the consistent improvement over linear regression. SD-GUIDED achieves a higher median test-set R^2 than LINEAR on all ten datasets and a higher R^2 in every paired run on every dataset. The mean paired differences range from +0.030 to +0.344. Together with the MAE results, this shows that the improvement over the initial linear model is consistent across data splits and across both primary evaluation metrics.

The comparison with STEPWISE is more dataset dependent. SD-GUIDED achieves the higher median R^2 on eight datasets and wins the majority of paired runs on the same eight datasets. BerkeleyDBC and LRZIP are the exceptions by win frequency, with STEPWISE performing better more often. On LRZIP, however, SD-GUIDED has a positive mean paired difference despite winning fewer runs, indicating that its less frequent improvements are larger than its losses on average. Overall, the R^2 results favor SD-GUIDED more consistently than the MAE results, although the relative performance of the two feature selection methods remains dependent on the dataset.

The comparison with CART is similarly heterogeneous. SD-GUIDED achieves the higher median R^2 on seven datasets and wins the majority of paired runs on six, with its clearest advantage occurring on HIPAcc, where it wins all 30 runs. CART achieves the higher median on BerkeleyDBC, JavaGC, and Polly and is favored by win frequency on four datasets. As in the MAE comparison, some datasets, including LLVM and VP9, show that win frequency and average effect magnitude do not necessarily agree. Overall, the results do not indicate a consistent predictive advantage for either method across systems.

Table 4.5: Two-sided Wilcoxon signed-rank tests comparing SD-guided with each baseline across ten datasets. For each dataset, the performance difference was averaged over the same 30 data splits before testing. Cells show Holm-adjusted p -values; significant results ($p < 0.05$) are bold.

Metric	Linear	Stepwise	CART
R^2	0.006	0.012	0.160
MAE	0.006	0.750	0.750

Statistical Significance. Table 4.5 reports the results of the statistical analysis described in Section 3.3.

For each dataset, we average the paired percentage reductions in MAE and the paired differences in R^2 (SD-guided minus baseline) across the 30 seeds. For each metric and baseline, we apply a two-sided Wilcoxon signed-rank test to the ten dataset-level averages. Holm correction is applied across the three baseline comparisons within each metric. SD-GUIDED significantly outperforms LINEAR for both R^2 ($p_{\text{Holm}} = 0.006$) and MAE ($p_{\text{Holm}} = 0.006$). Compared with STEPWISE, the results are mixed: SD-GUIDED achieves a significant improvement in R^2 ($p_{\text{Holm}} = 0.012$), whereas the difference in MAE is not statistically significant ($p_{\text{Holm}} = 0.750$). Compared with CART, the differences are not statistically significant for either R^2 ($p_{\text{Holm}} = 0.160$) or MAE ($p_{\text{Holm}} = 0.750$).

Summary. Taken together, the results show that SD-GUIDED clearly and significantly outperforms LINEAR on both MAE and R^2 . Compared with STEPWISE, SD-GUIDED performs significantly better in terms of R^2 , but not MAE. Against CART, neither metric shows a statistically significant difference. The evidence therefore supports a clear improvement over the initial linear model, but not a general advantage over the other established baselines.

Table 4.6: Median model size and composition across 30 random seeds. Terms denote the total number of regression terms; Rules gives the number of subgroup rules added by SD-guided; and Interactions gives the number of interaction terms retained by Stepwise.

Dataset	Linear	SD-guided		Stepwise	
	Terms	Terms	Rules	Terms	Interactions
7z	39	43	4	7	3
BerkeleyDBC	16	18	2	4	3
Dune	26	30	4	11	4
HIPAcc	50	55	5	14	12
JavaGC	32	36	4	7	5
LLVM	10	12	2	9	5
LRZIP	18	21	3	8	6
Polly	31	32	1	5	3
VP9	33	37	4	9	6
x264	13	15	2	7	3

Model complexity.

Table 4.6 compares the structural complexity of the three regression-based methods. In addition to the total number of estimated model terms, the table reports the number of subgroup rules added by SD-GUIDED and the number of interaction terms retained by STEPWISE. CART is excluded because it is structurally too different to the other methods.

Two patterns emerge. First, SD-GUIDED extends the initial linear model only incrementally, adding between one and five subgroup rules while retaining all original encoded feature terms. Its predictive improvements therefore require only a small expansion of the initial model, although the resulting models remain substantially larger than the sparse STEPWISE models.

Second, the compactness of STEPWISE does not imply a relative sparsity of interaction terms. STEPWISE retains between three and twelve interaction terms, which constitute at least half of the displayed median model size for seven of the ten datasets. This pattern is most pronounced for HIPAcc, where 12 of the 14 retained terms are interactions. STEPWISE therefore achieves compactness primarily by disregarding configuration option terms while preserving a comparatively interaction heavy representation.

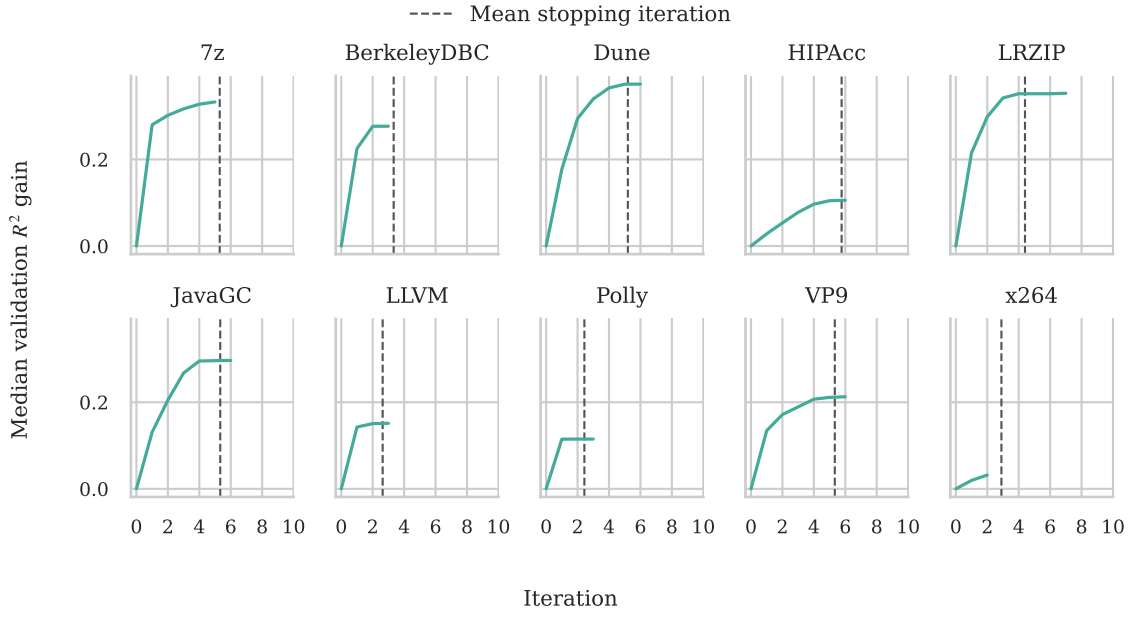


Figure 4.12: Median validation R^2 gain relative to the initial linear model over the SD-guided feature selection iterations, shown separately for each dataset and aggregated across 30 random seeds.

Convergence behavior.

Figure 4.12 shows the median validation R^2 gain over the initial model across the iterations of SD-GUIDED. Each curve ends at the beginning of its final plateau, while the dashed vertical line indicates the mean stopping iteration across runs. The stopping iteration includes the final unsuccessful iteration. Because the curves report the median whereas the stopping line reports the mean, a small number of late-stopping runs can shift the mean beyond the plateau of the median curve. Most of the improvement occurs within the first few iterations. Depending on the dataset, the median run reaches 90% of its final validation R^2 gain within one to four iterations. The final median gain varies substantially across datasets, ranging from approximately 0.03 for x264 to 0.37 for Dune, indicating that the benefit of the constructed terms is dataset-dependent. Overall, the mean stopping iteration remains well below the maximum of 10 iterations. Together with the concentration of improvements in the early iterations, this suggests that a lower iteration limit, such as five iterations, may be sufficient in many cases.

Table 4.7: Median runtime in seconds across 30 random seeds.

Dataset	Linear	SD-guided	Stepwise	CART
7z	0.089	17.513	8.347	0.055
BerkeleyDBC	0.020	2.542	0.515	0.007
Dune	0.019	6.391	5.416	0.007
HIPAcc	0.043	15.962	14.723	0.015
JavaGC	0.183	28.393	12.911	0.170
LLVM	0.021	1.367	1.683	0.006
LRZIP	0.018	3.616	1.243	0.006
Polly	0.059	6.253	3.389	0.050
VP9	0.221	33.399	35.801	0.207
x264	0.022	1.936	1.167	0.006

Runtime. Table 4.7 reports the median runtime across the 30 random seeds. SD-GUIDED and STEPWISE generally operate within the same order of magnitude, although SD-GUIDED is slower on eight of the ten datasets. The exceptions are LLVM and VP9, for which SD-GUIDED is slightly faster. Across all datasets, the median runtime of SD-GUIDED ranges from approximately 1.4 to 33.4 seconds, while that of Stepwise ranges from 0.5 to 35.8 seconds.

LINEAR and CART are substantially faster, remaining below 0.25 seconds for every dataset. This difference is expected because they fit a single model, whereas SD-GUIDED and STEPWISE perform iterative searches that repeatedly fit and evaluate candidate models. Nevertheless, the median runtime of SD-GUIDED remains below 34 seconds for every evaluated dataset, indicating that its additional computational cost is manageable at the scale considered in this evaluation.

4.2 Discussion

In this section, we first briefly summarize our main findings, then discuss their practical implications, and finally examine selected results in more detail.

Summary of findings. Our results for RQ₁ and RQ₂ show that subgroup derived rules can improve the predictive performance of a model containing only individual configuration option terms. At the same time, the modest recovery observed in RQ₁ shows that these improvements do not require the discovered rules to closely match the seeded interaction terms. This suggests that subgroup derived rules can capture relevant interaction effects in alternative forms, even when they do not reproduce the underlying interactions directly.

Comparison with Stepwise Feature Selection. The predictive comparison with stepwise feature selection (STEPWISE) does not establish that either feature selection strategy is generally superior. Although SD-guided feature selection (SD-GUIDED) performs better than STEPWISE in terms of R^2 , only the difference in R^2 is statistically significant, whereas the difference in MAE is not. One possible explanation is that R^2 is based on squared prediction errors and therefore gives larger errors more weight, while MAE averages the absolute prediction errors. Thus, reducing a few large errors can have a stronger effect on R^2 than on MAE. The recovery analysis nevertheless reveals complementary strengths. STEPWISE recovers pairwise interactions more effectively, whereas SD-GUIDED achieves higher recovery for three- and four-way interactions. This suggests that subgroup discovery’s wider beam is more robust when sparse higher-order interactions occur without their corresponding lower-order interaction terms. Despite producing substantially different model structures, the two methods remain broadly comparable in predictive performance. They therefore provide alternative representations of non-additive performance behavior, whose effectiveness depends on the dataset and the complexity of the underlying interactions.

The models learned by STEPWISE also differ structurally between the semi-synthetic and real-world experiments. On real-world data, STEPWISE selects many interaction terms, whereas on the semi-synthetic data it relies more heavily on individual configuration option terms. One possible reason is that our semi synthetic setup does not enforce a hierarchical interaction structure, meaning that higher-order interactions can occur without corresponding lower-order interactions being present. Since STEPWISE constructs higher-order interactions hierarchically, weak lower-order signals may prevent it from reaching some of these interactions. The predominance of individual option terms in the semi-synthetic models suggests that this lower-order signal was generally insufficient. This provides further support for the hierarchical interaction structure reported by Siegmund et al., who found that relevant performance interactions do usually not appear independently but form a hierarchy, with higher-order interactions building on corresponding lower-order interactions [23]. The interaction-heavy models learned from real-world data further support this finding.

Comparison with CART. Our comparison uses a CART model restricted to a maximum depth of four to keep its complexity and interpretability reasonably comparable to the regression-based methods. This constraint limits the number of partitions and conditional

relationships that the tree can represent and may therefore lead to underfitting on systems with more complex performance behavior. The comparison with CART is strongly dataset-dependent: SD-GUIDED performs better on some systems, most notably HIPAcc, whereas CART is more effective on others. Since the differences in MAE and R^2 are not statistically significant, our results do not support a general predictive advantage for either approach. The particularly large difference on HIPAcc may partly result from the depth restriction. Our results should therefore be interpreted as a comparison with an interpretability-constrained CART model rather than with a tree optimized solely for predictive accuracy. Allowing deeper trees might improve CART’s predictive performance, but would also reduce the comparability in terms of model complexity and interpretability.

Convergence and Model Complexity. The convergence analysis indicates that subgroup rules provide most of their predictive benefit after only a small number of iterations. In the semi-synthetic data, scenarios with fewer seeded interactions stabilize earlier than those with more interactions, suggesting that convergence depends more strongly on the number of interactions than on their order. Rapid stabilization is also evident in the real world datasets, where the median convergence curves reach 90% of their final validation R^2 gain within one to four iterations. The real world convergence curves therefore generally resemble those of the sparser semi-synthetic scenarios. This is consistent with previous findings that a small number of influential options and interactions can explain a large proportion of the performance variation in configurable systems [14, 23]. However, convergence alone cannot reveal the complete interaction structure. Further interactions may exist, but only a limited subset appears to provide substantial predictive improvement.

Because SD-GUIDED retains all original configuration-option terms, its models remain larger than those produced by STEPWISE. The rules therefore provide a relatively compact extension of the initial model, although that model may still contain redundant individual option terms. Removing highly collinear predictors before residual mining or eliminating unnecessary terms after rule construction could produce smaller models. Since these changes could also affect the residuals and the rules subsequently discovered, they require a separate evaluation and are discussed further in Section 4.4.

Practical Implications. SD-GUIDED may be particularly useful for systems in which sparse higher-order interactions are expected and a hierarchical interaction structure cannot be assumed, as in our semi-synthetic setting. In such cases, STEPWISE may fail to reach relevant higher-order interactions because the corresponding lower-order terms do not provide sufficient improvement to be selected first. Even though beam search also explores subgroup descriptions hierarchically, SD-GUIDED can recover higher-order interactions more effectively in practice. With a sufficiently wide beam, promising intermediate descriptions can remain in the search without requiring the corresponding lower-order interaction terms to be retained in the regression model. In contrast, STEPWISE must first select suitable lower-order interaction terms before it can construct higher-order ones. Our results therefore suggest that SD-GUIDED can be less restrictive in practice when higher-order structure is not supported by sufficiently strong lower-order effects.

Conversely, STEPWISE may be preferable when a compact model is required or when the selected interactions should be represented explicitly through conventional interaction terms. It may also be more suitable when prior empirical evidence or domain knowledge indicates

that pairwise interactions predominate. Although SD-GUIDED is slower than STEPWISE on most of the evaluated datasets, their runtimes remain within the same order of magnitude. This suggests that incorporating subgroup discovery is computationally feasible in the considered setting. Stricter stopping criteria could potentially reduce its runtime further.

The choice between SD-GUIDED and CART depends primarily on the intended use of the resulting model. CART may be preferable when predictive performance and computational efficiency are the primary objectives and a tree-based representation is acceptable. In contrast, SD-GUIDED provides an alternative when the additive structure of a performance-influence model should be retained and extended through explicit rule-based terms.

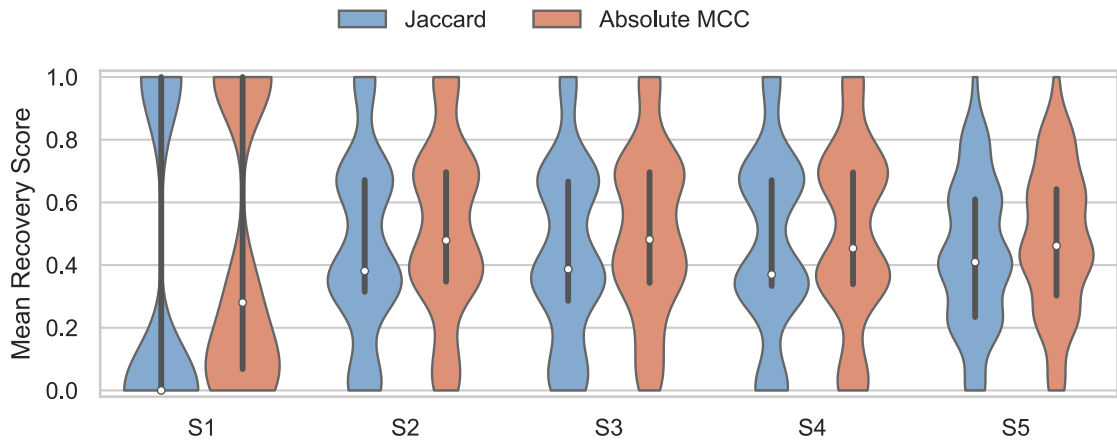


Figure 4.13: Distributions of run level mean seeded-interaction recovery for each scenario, comparing region-based Jaccard similarity with absolute MCC.

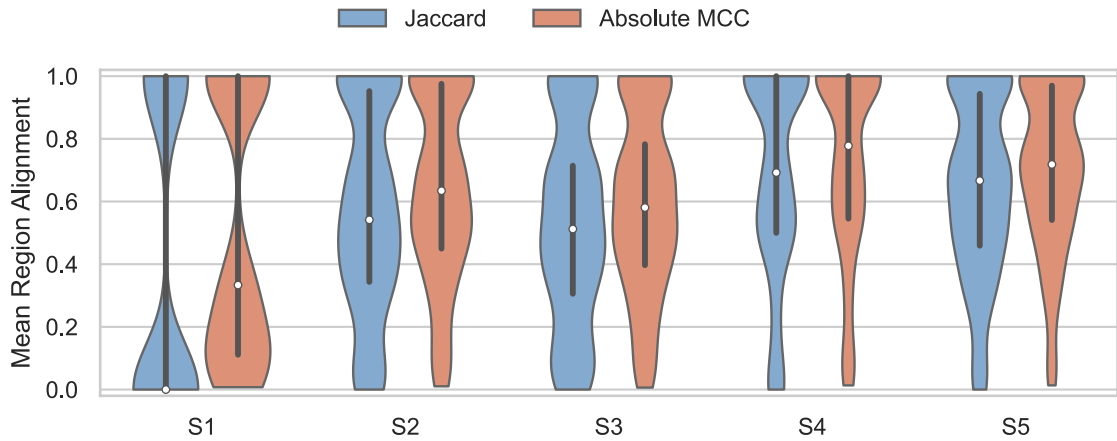


Figure 4.14: Distributions of run level mean rule alignment for each scenario, comparing region-based Jaccard similarity with absolute MCC. Runs without discovered rules are excluded.

Explaining Limited Structural Recovery. Despite the strong predictive performance, the recovery of the seeded interactions was more modest than expected. The alignment of the discovered subgroup rules with the seeded interaction regions was also generally low. Moreover, the alignment analysis only considers runs in which at least one subgroup rule was learned. Runs without any discovered rules are therefore not penalised, meaning that the reported alignment is, if anything, an optimistic estimate.

One possible explanation for the modest recovery was that subgroup discovery identifies regions that are complementary to the seeded interaction regions. Such regions can have a Jaccard similarity close to zero despite being strongly related. To investigate this possibility, we repeat the analysis using the Matthews correlation coefficient (MCC) [16]. MCC is commonly used in binary classification to measure agreement between predicted and observed class labels [3]. It ranges from -1 to 1 , where 1 represents perfect agreement, 0 represents no correlation, and -1 represents perfect disagreement. In our setting, the

binary labels indicate whether a configuration belongs to the region defined by a rule or interaction. MCC can therefore reveal complementary regions through negative scores, with -1 indicating that two regions are exact complements.

For each comparison, we use the absolute MCC when selecting the strongest match. This is relevant for prediction because a region and its exact complement encode the same binary partition of the data. Given an intercept, a model using the complementary indicator $1 - R$ can represent the same predictions as one using R by adjusting the intercept and reversing the coefficient. We therefore treat strong agreement and strong disagreement equally during matching, while retaining the signed MCC to distinguish direct matches from complementary ones.

As shown in Figures 4.13 and 4.14, using the absolute MCC produces only a small numerical increase in recovery and alignment. The overall distributions remain similar to those obtained with the Jaccard score. The analysis therefore provides little evidence that the modest Jaccard-based recovery is primarily caused by the discovery of globally complementary regions.

A further consideration is that the effect of an interaction can be explained by the regression model without being recovered as the same region. SD-GUIDED includes both the original configuration options and the discovered subgroup rules in the regression model. These predictors can be collinear, meaning that the same performance pattern can be represented by different combinations of model terms. For example, for binary options A and B , the interaction AB can also be written as $A - A(1 - B)$. The model may therefore represent a seeded interaction by combining an original option with a discovered rule, even if that rule alone has little overlap with the seeded interaction. This relates to the broader difficulty of attributing observed effects to particular features or interactions and isolating their individual contributions [5, 27]. Because Jaccard similarity and MCC compare each discovered rule with the seeded interaction in isolation, they do not capture such combined representations. Low region-based similarity may therefore indicate a different representation rather than a failure to capture the interaction related effect on performance.

This limitation also affects how the discovered rules should be interpreted. A subgroup rule identifies a region in which the initial model makes systematic prediction errors, but it does not necessarily identify the interaction responsible for those errors. The discovered rules should therefore primarily be treated as predictive representations rather than as direct evidence of the underlying interaction structure.

4.3 Threats to Validity

This section discusses potential threats to the validity of the proposed evaluation. We distinguish between internal validity, which concerns whether the observed results can be attributed to the proposed method and experimental design, and external validity, which concerns the extent to which the findings generalize beyond the datasets and settings considered in this thesis. For each category, we describe possible limitations and the steps taken to mitigate them.

Internal Validity One aspect of the semi-synthetic design that may affect the comparison is the absence of an enforced interaction hierarchy. Although both methods construct higher-order patterns incrementally, their treatment of intermediate candidates differs. Subgroup discovery uses a relatively wide beam that retains several lower-order descriptions for further refinement without requiring them to become part of the regression model. In contrast, *STEPWISE* can construct a higher-order interaction only after the corresponding lower-order term has provided sufficient predictive improvement to be selected and retained in the model. Although a higher-order interaction can induce an average performance signal for its lower-order subsets, this signal may be too weak for *STEPWISE* to select and retain the corresponding lower-order terms.

Another design choice concerns how many individual options are represented in the semi-synthetic PIMs. We fixed this number at five as an empirically informed heuristic based on the average number of options retained by *STEPWISE* in the RQ2 experiments. The purpose was to create a sparse main effect structure while avoiding an unrealistically small model. Keeping this number constant also prevents differences between the interaction scenarios from being caused by different numbers of configuration options. However, we did not evaluate alternative levels of main configuration option sparsity. Including more or fewer individual option terms could alter the residual patterns and the relative importance of the seeded interactions, potentially affecting their recovery.

Another internal threat concerns the hyperparameters of the proposed approach. The results may depend on choices such as the beam width, maximum description length, maximum number of iterations, and improvement threshold. During development, we tried different settings to choose suitable default values. However, these tests were not recorded as a systematic sensitivity analysis. We therefore report the final settings for reproducibility, but cannot determine how strongly different settings would affect the results.

A further consideration for internal validity is that the two methods use the validation set differently. *SD-GUIDED* uses it both to discover subgroup descriptions and to select which candidate features to retain, whereas *STEPWISE* uses it only to evaluate candidates. Because subgroup descriptions are mined from the signed validation residuals, the validation data play a larger role in the search procedure of *SD-GUIDED*. This difference should be considered when comparing the two methods. Following standard model evaluation practice [10], we evaluate each final model on a held out test set that is not used for training, subgroup discovery, or feature selection. We also limit the number of feature selection iterations, reducing repeated adaptation to the validation data. Evaluation across random splits then allows us to assess whether the selected subgroup-derived features generalize beyond the data used during model construction.

Finally, another threat arises from the random variation in the experimental setup that may affect the observed recovery and predictive performance. In particular, the train-validation-test partition and the sampled semi-synthetic PIM may favor one method by chance. To reduce this influence, we use 50 random seeds per semi-synthetic scenario in RQ1 and 30 random splits per real-world dataset in RQ2. All methods are evaluated on the same partition within each repetition, and the results are aggregated across repetitions. This reduces the likelihood that the conclusions are driven by a single favorable split or sampled interaction structure, although the reported results remain estimates based on the evaluated samples.

External Validity A threat to external validity concerns the generalizability of the results beyond the systems considered in this evaluation. Although the selected systems cover a broad range of domains, they are not necessarily representative of all configurable software systems. Other systems may have larger configuration spaces, different option types, stronger constraints between options, or more complex interaction structures. For example, since our experimental design relies on binary, one-hot-encoded configuration features, it would require adaptation for systems with numeric configuration options. The observed results may therefore not transfer directly to all configurable systems.

Another threat concerns the performance measurements themselves. Performance values may vary within and across hardware and software environments due to differences in processors, memory hierarchies, operating systems, compiler versions, background load, or measurement noise. Because all evaluated methods learn from these measurements, their models and selected features may capture environment-specific effects rather than stable configuration behavior. This applies both to subgroup descriptions discovered by SD-GUIDED and to interaction terms selected by the comparison methods. Although the use of the same measurements for all methods makes it unlikely that this issue systematically favors one method, the resulting models may not necessarily generalize to other hardware or software environments.

Finally, the semi-synthetic evaluation of RQ1 uses performance values generated from a synthetic PIM. This is useful because the true interactions are known. However, it assumes that performance follows a linear model with explicit interaction terms. This is an unrealistic assumption as real configurable systems are unlikely to behave like this. Therefore, the semi-synthetic results may be more favorable than results on real performance data. To reduce this threat, we add noise and vary the data-generating parameters described in Section 3.3, making the semi-synthetic setting less idealized.

4.4 Future Work

4.4.1 Sampling Strategies

The present evaluation uses the complete set of available measurements for each subject system and randomly partitions these observations into training, validation, and test sets. This setup allows us to evaluate the feature selection procedure without additionally considering the cost of collecting performance measurements. In practical applications, however, obtaining such extensive measurement data may be infeasible. Configuration spaces can contain a very large number of valid configurations, and compiling, executing, and benchmarking each configuration may require substantial computational effort.

Previous work has therefore proposed different strategies for selecting a smaller but informative subset of configurations. Random sampling provides a simple and generally applicable baseline. In contrast, t -wise sampling aims to include every valid combination of settings for each group of t options at least once. This may help represent interactions between options in the sample. Distance-based strategies [12] instead attempt to distribute measurements broadly across the configuration space. Since the suitability of a sampling

strategy depends on the application, subject system, property of interest, and available domain knowledge, no generally optimal sampling strategy has yet been established [20].

Future work should therefore evaluate SD-GUIDED feature selection under restricted measurement budgets and alternative configuration-sampling strategies. Random, t -wise, and distance-based sampling could be compared using equal measurement budgets, focusing on the predictive performance of the resulting PIMs on unseen configurations. This would show whether SD-GUIDED feature selection remains effective when only a limited number of performance measurements is available.

4.4.2 Model Simplification

The current approach retains all configuration option terms and adds a small number of subgroup features intended to capture interaction structure not represented by these options. However, prior work indicates that the performance of configurable systems can often be predicted accurately using a relatively small set of influential options and interactions [14]. Future work could therefore combine SD-guided feature construction with backward feature elimination, as used by Siegmund et al. [23], to remove terms that do not contribute sufficiently to predictive performance.

In our experiments, the procedure rarely reached the maximum of ten iterations. This suggests that the stopping criterion could be refined to terminate when an additional subgroup feature provides insufficient improvement on validation data relative to the increase in model complexity. Such a criterion could produce smaller models while retaining most of the predictive improvement.

The current approach also does not explicitly account for multicollinearity among the configuration options. Variance inflation factor (VIF) analysis could be used to identify highly collinear predictors [13]. Redundant options could then be considered for removal before fitting the regression model. The effect of this additional step on model size, predictive accuracy, and interpretability would need to be evaluated.

Related Work

This chapter discusses work related to our approach. We first review performance-prediction techniques for configurable systems, with an emphasis on approaches that learn interpretable models from sparse measurements. We then discuss subgroup discovery and related rule-based methods for analyzing predictive models. Finally, we position this thesis with respect to both lines of work.

5.1 Performance Prediction for Configurable Systems

Performance prediction for configurable software systems aims to infer the performance of unseen configurations from a limited number of measurements. This is necessary because the number of possible configurations grows rapidly with the number of configuration options, making exhaustive measurement infeasible. Prior work has therefore explored different model classes and sampling strategies to learn accurate predictors under restricted measurement budgets.

5.1.1 Interpretable regression models

Regression-based approaches are especially relevant to this thesis because they can yield interpretable models. In a linear model, coefficients can be read as estimates of how individual configuration options influence performance. For configurable systems, however, individual options alone are often insufficient: the impact of one option may depend on the state of another option, leading to feature interactions [24]. An accurate model must therefore include not only individual option effects, but also selected interaction terms.

Siegmund et al. [23] address this problem with *performance-influence models*. These models represent performance as a combination of influence terms for individual options and selected interactions. Their key advantage is that they combine prediction with explanation: the learned terms indicate which configuration choices are responsible for performance variation. Since the space of possible interactions is large, the approach relies on heuristics and stepwise regression to construct and select a manageable set of terms.

Kolesnikov et al. [14] study the practical trade-offs involved in learning such models. They analyze how prediction error, model size, and computation time change with increasing model complexity. Their results show that adding more complex interaction terms does not necessarily lead to proportional improvements in prediction accuracy. This observation is important for the present thesis: subgroup-derived terms may improve a model, but they also increase model complexity and therefore need to be selected carefully.

Kaltenecker et al. [13] study how the performance effects of configuration options change across software releases. They use performance-influence models and compare their coefficients between releases. Before making these comparisons, they account for multicollinearity. Configuration constraints can make one model term predictable from a combination of other terms. In that case, different coefficient values can produce the same predictions, making the individual coefficients difficult to compare reliably. The authors detect such dependencies using variance inflation factor analysis and remove redundant terms before comparing coefficients. This does not reduce predictive performance because the removed terms provide no independent information. Their work therefore highlights the difference between interpreting the predictions of a model and assigning a unique influence to each individual coefficient.

Together, these works establish performance-influence models as an interpretable approach to performance prediction. They also motivate the central problem addressed in this thesis: useful interaction terms are needed, but considering all possible interactions is infeasible. This thesis therefore investigates whether systematic residual patterns can be used to generate candidate terms more selectively.

5.1.2 Tree-based performance models

Classification and Regression Trees recursively partition an input space into regions with similar target values. Guo et al. [7] applied this idea to configurable software systems by using such splits to identify regions of the configuration space with similar performance behavior. Because interactions can be represented implicitly through paths in the tree, such models do not require interaction terms to be specified explicitly in advance.

Guo et al. [8] also propose DECART, a data-efficient approach that combines CART with automated resampling and hyperparameter tuning. Other work has also explored tree-based and related learning strategies for finding good configurations or improving performance prediction under limited measurements [7, 18, 19]. These approaches are attractive because they can model nonlinear behavior and interactions without manually constructing a set of candidate interaction terms.

The main difference lies in the structure of the resulting model. A regression tree explains a prediction through a sequence of splits, whereas a performance-influence model uses explicit weighted terms. Tree-based models are therefore useful prediction baselines, but they do not provide the same additive representation of options and interactions.

5.1.3 Other performance-prediction techniques

Beyond regression and tree-based models, prior work has explored several other prediction techniques. Zhang et al. [28] apply Fourier learning to configurable systems. They represent each configuration as a binary vector and view performance as a function from configurations to real-valued performance measurements. Their method approximates this function by estimating its Fourier coefficients from a sample of measured configurations.

Ha and Zhang [9] propose DeepPerf, a deep sparse neural network for performance prediction. Neural models can capture complex nonlinear dependencies between configuration

options and performance, but their predictions are generally less directly interpretable than those of performance-influence models.

Jamshidi et al. [11] study transfer learning for configurable systems. Instead of learning a performance model from scratch for every new environment, transfer-learning approaches reuse knowledge across settings such as different hardware platforms or workloads. This can reduce the number of measurements required in a target environment.

These techniques address the broader problem of accurate performance prediction. In contrast, this thesis investigates an alternative way of constructing performance-influence models.

5.2 Subgroup Discovery and Rule-Based Model Analysis

Subgroup discovery identifies interpretable regions of a data space whose target behavior differs from the overall population. In the context of model analysis, such regions can reveal where a predictive model behaves unusually, for example by producing systematically high or low errors. This makes subgroup discovery relevant to the problem considered in this thesis: residual patterns may indicate parts of the configuration space that are not yet captured by the current performance model.

Sutton et al. [26] use subgroup discovery to characterize domains of applicability of machine-learning models. Their approach identifies interpretable regions in which a model performs better than average. This demonstrates that subgroup descriptions can be used to explain where a learned model is reliable. For our setting, this idea is relevant because it treats model behavior itself as the object of analysis, rather than only the original prediction target.

Pimentel et al. [21] propose Error Distribution Rules for regression-model analysis. Their method searches for subgroups in which the distribution of prediction errors differs from the global error distribution. This residual-oriented view is close to the starting point of our approach: both methods use subgroup mining to identify systematic patterns in regression errors. The difference is in how the discovered patterns are used. Error Distribution Rules support model inspection, whereas this thesis uses residual-based subgroup descriptions as candidate features for refining a global performance-influence model.

Subbaswamy et al. [25] use subgroup-style rules to audit clinical AI models. Their framework identifies patient subgroups in which a model underperforms and describes those subgroups with simple rules. This line of work further illustrates the value of interpretable subgroup descriptions for debugging predictive models. Its goal, however, remains diagnostic: the discovered groups explain where a model fails. In this thesis, such explanations are not the final output, but a source of candidate model terms.

Exceptional Model Mining provides a broader framework for finding subgroups in which a local model differs from a global model [15]. Duivesteijn et al. [6] instantiate this idea for regression models by searching for subgroups whose locally fitted regression models differ substantially from the global regression model. Their interestingness criterion is based on Cook’s distance. This connects subgroup descriptions with regression behavior, but the emphasis is on discovering local model deviations rather than on modifying the original global model.

Recent work by Mishra et al. [17] introduces Exceptional Model Residual Mining, which is closely related to our approach because it also mines subgroups from model residuals. In addition, they study richer description languages for residual-based subgroup mining. Such description languages are an interesting direction for future work in the domain of configurable systems, because they may better capture nonlinear or interaction-related regions of the configuration space. The main difference to our work lies in how the discovered subgroups are used. While Mishra et al. focus on residual-based subgroup mining as a model-analysis task, we use the discovered subgroup descriptions as intermediate candidates in an iterative model-construction procedure.

5.3 Positioning of This Thesis

This thesis lies at the intersection of performance-influence modeling for configurable systems and subgroup-discovery-based model analysis. Performance-influence models provide the additive and interpretable model form used in this work, while subgroup discovery provides a way to describe regions of the data in which a model behaves unusually.

The main distinction from prior work is the intended use of subgroup discovery. Existing subgroup-based approaches mainly use discovered regions to analyze or debug predictive models. In this thesis, we investigate whether such regions can also serve as candidate terms in performance-influence models for configurable software systems.

To our knowledge, residual-based subgroup mining has not previously been studied as a mechanism for constructing candidate terms for performance-influence models of configurable software systems.

Concluding Remarks

Predicting the performance of configurable software systems is challenging because their configuration spaces are generally too large to measure exhaustively. Performance-influence models address this problem by learning the effects of configuration options and selected interactions from a limited number of measurements [23]. However, identifying suitable interaction terms remains difficult because the number of possible interactions grows rapidly with the number of configuration options [24].

This thesis proposes SD-guided feature selection as an error-driven mechanism for generating subgroup derived candidate terms for performance-influence models. The approach applies subgroup discovery to the signed residuals of a linear main-effect model. It then converts regions with systematic prediction errors into human-readable binary rule features and adds these features when they improve validation performance.

We evaluated the approach in two complementary settings. In the semi-synthetic setting, we used performance data with known interaction terms to examine both the alignment of individual subgroup-derived rules with seeded interaction regions and the recovery of seeded interactions by the selected model terms. In the real-world setting, we used measured performance data from ten configurable systems to assess whether subgroup-derived features improve predictive performance. We compared the proposed approach with plain linear regression, stepwise feature selection, and CART.

The semi-synthetic evaluation shows that the subgroup-derived rules only partly correspond to the seeded interactions. Some rules align closely with known interaction regions, but overall alignment and recovery were modest. SD-guided feature selection recovers three- and four-way interactions more effectively, whereas Stepwise feature selection performs better for pairwise interactions. This suggests that subgroup rules often provide alternative predictive representations rather than exact reconstructions of the original interaction terms.

On the measured performance data, SD-guided feature selection outperformed plain linear regression across all ten systems, with statistically significant improvements in both R^2 and MAE. This indicates that the subgroup-derived rules capture predictive effects missing from the initial linear model. SD-guided feature selection also remained competitive with stepwise feature selection and CART, although no method consistently performed best across all systems and metrics. Overall, the findings show that a small number of subgroup derived features can capture non linear performance behavior without reproducing the underlying interaction terms directly. The main contribution of this thesis is the integration of residual-guided subgroup discovery into iterative feature selection for linear performance-influence models. Unlike existing feature selection methods, SD-guided feature selection can capture higher-order interactions without first selecting their lower-order subsets, giving it an advantage in sparse higher-order scenarios.

Appendix

a.1 Supplementary RMSE Results for RQ2

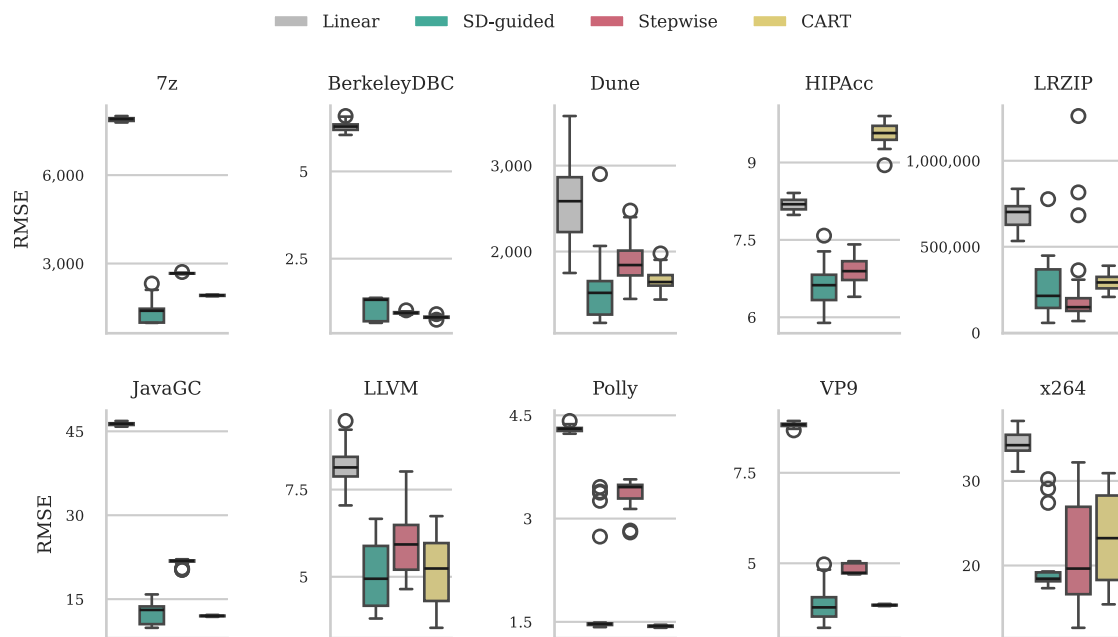


Figure A.1: Distribution of test-set RMSE across datasets and prediction methods.

Table A.1: Test-set RMSE comparison. Each cell shows SD-guided’s average percentage reduction in RMSE relative to the method named above it across the same 30 data splits. Positive values favor SD-guided; parentheses show how often it achieved the lower RMSE. Green indicates a majority of wins and red a majority of losses.

Dataset	Linear	Stepwise	CART
7z	+82.6% (30/30)	+48.4% (30/30)	+28.4% (27/30)
BerkeleyDBC	+82.1% (30/30)	-18.7% (10/30)	-36.4% (10/30)
Dune	+40.1% (30/30)	+17.5% (26/30)	+7.5% (22/30)
HIPAcc	+19.1% (30/30)	+3.7% (21/30)	+30.6% (30/30)
JavaGC	+72.8% (30/30)	+41.8% (30/30)	-5.0% (10/30)
LLVM	+39.0% (30/30)	+14.7% (28/30)	+2.7% (14/30)
LRZIP	+61.3% (30/30)	-72.8% (11/30)	+8.1% (20/30)
Polly	+59.1% (30/30)	+47.9% (30/30)	-22.5% (0/30)
VP9	+55.9% (30/30)	+19.3% (29/30)	-1.2% (18/30)
x264	+43.2% (30/30)	-0.0% (17/30)	+12.9% (21/30)

Statement on the Usage of Generative Digital Assistants

For this thesis, the following generative digital assistants were used: `CHATGPT`, using the GPT-5.5 and GPT-5.6 models, was used to explain code errors and issues related to dataset parsing, as well as to revise and format parts of the text. In particular, it was used to improve grammar, make the writing more formal, and refine the visual presentation of tables and figures. The underlying logic was implemented by the author. The tool was also used to provide general feedback on text. All suggestions were critically reviewed and adapted where necessary. All methodological choices, interpretations, and final wording decisions were made by the author.

For code completion, `GITHUB COPILOT` was additionally used with its default settings in `VISUAL STUDIO CODE`, primarily to complete repetitive code patterns and boilerplate code. All generated code was manually reviewed, tested where applicable, and integrated only after the author had understood its functionality and implications.

The potential risks associated with generative digital assistants were considered throughout, and these tools were used with caution, critical reflection, and responsibility.

Bibliography

- [1] Martin Atzmueller. “Subgroup discovery.” In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.1 (2015), pp. 35–49. DOI: [10.1002/widm.1144](https://doi.org/10.1002/widm.1144).
- [2] Muffy Calder, Mario Kolberg, Evan H. Magill, and Stephan Reiff-Marganiec. “Feature interaction: a critical review and considered forecast.” In: *Computer Networks* 41.1 (Jan. 2003), pp. 115–141. ISSN: 1389-1286. DOI: [10.1016/S1389-1286\(02\)00352-3](https://doi.org/10.1016/S1389-1286(02)00352-3). URL: <https://www.sciencedirect.com/science/article/pii/S1389128602003523>.
- [3] Davide Chicco and Giuseppe Jurman. “The Advantages of the Matthews Correlation Coefficient (MCC) over F1 Score and Accuracy in Binary Classification Evaluation.” In: *BMC Genomics* 21.1, 6 (2020). DOI: [10.1186/s12864-019-6413-7](https://doi.org/10.1186/s12864-019-6413-7).
- [4] Janez Demšar. “Statistical Comparisons of Classifiers over Multiple Data Sets.” In: *Journal of Machine Learning Research* 7 (2006), pp. 1–30. URL: <https://www.jmlr.org/papers/v7/demsar06a.html>.
- [5] Clemens Dubsloff, Kallistos Weis, Christel Baier, and Sven Apel. “Causality in Configurable Software Systems.” In: *Proceedings of the 44th International Conference on Software Engineering*. Association for Computing Machinery, 2022, pp. 325–337. DOI: [10.1145/3510003.3510200](https://doi.org/10.1145/3510003.3510200).
- [6] Wouter Duivesteyn, Ad J. Feelders, and Arno J. Knobbe. “Different Slopes for Different Folks: Mining for Exceptional Regression Models with Cook’s Distance.” In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’12)*. Beijing, China: ACM, 2012, pp. 868–876. DOI: [10.1145/2339530.2339668](https://doi.org/10.1145/2339530.2339668).
- [7] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrzej Wasowski. “Variability-aware performance prediction: A statistical learning approach.” In: *2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11–15, 2013*. Ed. by Ewen Denney, Tevfik Bultan, and Andreas Zeller. IEEE, 2013, pp. 301–311. DOI: [10.1109/ASE.2013.6693089](https://doi.org/10.1109/ASE.2013.6693089). URL: <https://doi.org/10.1109/ASE.2013.6693089>.
- [8] Jianmei Guo, Dingyu Yang, Norbert Siegmund, Sven Apel, Atrisha Sarkar, Pavel Valov, Krzysztof Czarnecki, Andrzej Wasowski, and Huiqun Yu. “Data-efficient performance learning for configurable systems.” In: *Empirical Software Engineering* 23.3 (June 2018), pp. 1826–1867. DOI: [10.1007/s10664-017-9573-6](https://doi.org/10.1007/s10664-017-9573-6). URL: <https://doi.org/10.1007/s10664-017-9573-6>.

- [9] Huong Ha and Hongyu Zhang. “DeepPerf: Performance Prediction for Configurable Software with Deep Sparse Neural Network.” In: *Proceedings of the 41st International Conference on Software Engineering (ICSE 2019)*. IEEE/ACM, 2019, pp. 1095–1106. DOI: [10.1109/ICSE.2019.00113](https://doi.org/10.1109/ICSE.2019.00113).
- [10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer Series in Statistics. Springer, 2009. DOI: [10.1007/978-0-387-84858-7](https://doi.org/10.1007/978-0-387-84858-7).
- [11] Pooyan Jamshidi, Norbert Siegmund, Miguel Velez, Christian Kästner, Akshay Patel, and Yuvraj Agarwal. “Transfer learning for performance modeling of configurable systems: an exploratory analysis.” In: *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017)*. IEEE Computer Society, 2017, pp. 497–508. DOI: [10.1109/ASE.2017.8115661](https://doi.org/10.1109/ASE.2017.8115661). URL: <https://doi.org/10.1109/ASE.2017.8115661>.
- [12] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. “Distance-Based Sampling of Software Configuration Spaces.” In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1084–1094. DOI: [10.1109/ICSE.2019.00112](https://doi.org/10.1109/ICSE.2019.00112). URL: <https://doi.org/10.1109/ICSE.2019.00112>.
- [13] Christian Kaltenecker, Stefan Mühlbauer, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. “Performance Evolution of Configurable Software Systems: An Empirical Study.” In: *Empirical Software Engineering* 28.6, 152 (2023). DOI: [10.1007/s10664-023-10338-3](https://doi.org/10.1007/s10664-023-10338-3). URL: <https://doi.org/10.1007/s10664-023-10338-3>.
- [14] Sergiy Kolesnikov, Norbert Siegmund, Christian Kästner, Alexander Grebhahn, and Sven Apel. “Tradeoffs in Modeling Performance of Highly Configurable Software Systems.” In: *Software and Systems Modeling* 18.3 (2019), pp. 2265–2283. DOI: [10.1007/s10270-018-0662-9](https://doi.org/10.1007/s10270-018-0662-9).
- [15] Dennis Leman, Ad Feelders, and Arno J. Knobbe. “Exceptional Model Mining.” In: *Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2008, Part II)*. 2008, pp. 1–16. DOI: [10.1007/978-3-540-87481-2_1](https://doi.org/10.1007/978-3-540-87481-2_1).
- [16] Brian W. Matthews. “Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme.” In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), pp. 442–451. DOI: [10.1016/0005-2795\(75\)90109-9](https://doi.org/10.1016/0005-2795(75)90109-9).
- [17] Aniket Mishra, Cristiana Cărbunaru, and Wouter Duivesteijn. “Exceptional Model Residual Mining, and Three Richer EMM Description Languages.” In: *Advances in Intelligent Data Analysis XXIV*. Vol. 16513. Lecture Notes in Computer Science. Springer, 2026, pp. 269–282. DOI: [10.1007/978-3-032-23833-7_20](https://doi.org/10.1007/978-3-032-23833-7_20).
- [18] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. “Faster Discovery of Faster System Configurations with Spectral Learning.” In: *CoRR abs/1701.08106* (2017). arXiv: [1701.08106 \[cs.SE\]](https://arxiv.org/abs/1701.08106). URL: <https://arxiv.org/abs/1701.08106>.
- [19] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. “Using Bad Learners to Find Good Configurations.” In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2017. ACM, 2017, pp. 257–267. DOI: [10.1145/3106237.3106238](https://doi.org/10.1145/3106237.3106238).

- [20] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. "Learning software configuration spaces: A systematic literature review." In: *Journal of Systems and Software* 182, 111044 (2021). DOI: [10.1016/j.jss.2021.111044](https://doi.org/10.1016/j.jss.2021.111044). URL: <https://doi.org/10.1016/j.jss.2021.111044>.
- [21] João Pimentel, Paulo J. Azevedo, and Luís Torgo. "Subgroup mining for performance analysis of regression models." In: *Expert Systems* 40.1, e13118 (2023). DOI: [10.1111/exsy.13118](https://doi.org/10.1111/exsy.13118).
- [22] Raimundo Real and Juan M. Vargas. "The Probabilistic Basis of Jaccard's Index of Similarity." In: *Systematic Biology* 45.3 (1996), pp. 380–385. DOI: [10.1093/sysbio/45.3.380](https://doi.org/10.1093/sysbio/45.3.380).
- [23] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. "Performance-influence models for highly configurable systems." In: *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 2015, pp. 284–294. DOI: [10.1145/2786805.2786845](https://doi.org/10.1145/2786805.2786845).
- [24] Norbert Siegmund, Sergiy S. Kolesnikov, Christian Kästner, Sven Apel, Don S. Batory, Marko Rosenmüller, and Gunter Saake. "Predicting performance via automated feature-interaction detection." In: *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*. Ed. by Martin Glinz, Gail C. Murphy, and Mauro Pezzè. IEEE Computer Society, 2012, pp. 167–177. DOI: [10.1109/ICSE.2012.6227196](https://doi.org/10.1109/ICSE.2012.6227196). URL: <https://doi.org/10.1109/ICSE.2012.6227196>.
- [25] Adarsh Subbaswamy, Berkman Sahiner, Nicholas Petrick, Vinay Pai, Roy Adams, Matthew C. Diamond, and Suchi Saria. "A data-driven framework for identifying patient subgroups on which an AI/machine learning model may underperform." In: *npj Digital Medicine* 7 (2024), p. 334. DOI: [10.1038/s41746-024-01275-6](https://doi.org/10.1038/s41746-024-01275-6). URL: <https://doi.org/10.1038/s41746-024-01275-6>.
- [26] Christopher Sutton, Mario Boley, Luca M. Ghiringhelli, Matthias Rupp, Jilles Vreeken, and Matthias Scheffler. "Identifying domains of applicability of machine learning models for materials science." In: *Nature Communications* 11, 4428 (2020). DOI: [10.1038/s41467-020-17112-9](https://doi.org/10.1038/s41467-020-17112-9).
- [27] Kallistos Weis, Leopoldo Teixeira, Clemens Dubslaff, and Sven Apel. "Blackbox Observability of Features and Feature Interactions." In: *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*. ACM, 2024, pp. 1120–1132. DOI: [10.1145/3691620.3695490](https://doi.org/10.1145/3691620.3695490).
- [28] Yi Zhang, Jianmei Guo, Eric Blais, and Krzysztof Czarnecki. "Performance Prediction of Configurable Software Systems by Fourier Learning." In: *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)*. IEEE, 2015, pp. 365–373. DOI: [10.1109/ASE.2015.15](https://doi.org/10.1109/ASE.2015.15).