Bachelor's Thesis

Revisiting Sampling-Strategy-Machine-Learning Interplay: A Conceptual Replication

Marcel Niedballa

September 30, 2026

Advisors: Kallistos Weis
Tobias Dick

Reviewers: Prof. Dr. Sven Apel
Prof. Dr. Martina Maggio





UNIVERSITÄT DES SAARLANDES

Chair of Software Engineering ◇ Saarland Informatics Campus ◇ Saarland University

Marcel Niedballa: *Revisiting Sampling-Strategy-Machine-Learning Interplay: A Conceptual Replication,* © September 2026

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 für folgende Teilaufgaben: Textüberarbeitung/-korrektur, Überprüfung auf Fehler wie Grammatik und Rechtschreibung von Text, Brainstorming, Code-Vervollständigung und Identifizierung von Bugs und nicht zur Bearbeitung der in der Arbeit aufgeworfenen Fragestellungen zu Hilfe genommen habe. 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, check for mistakes such as grammar and spelling errors, brainstorming, code completion and identification of bugs, and not to address or formulate the main research questions of the thesis. 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, 30.09.26 _____
(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, 30.09.26 _____
(Datum Date) (Unterschrift Signature)

Abstract

Highly configurable software systems have large configuration spaces that grow exponentially with the number of options, making exhaustive performance analysis computationally challenging. Existing research in this field therefore relies on sampling strategies and machine-learning techniques to predict the performance of unseen configurations. Prior work identified significant interactions between these factors with respect to prediction quality. However, these findings were obtained under fixed experimental conditions, leaving the generalisability of the observed interplay unclear.

This thesis conducts a conceptual replication of the interplay between sampling strategies and machine-learning techniques under a modified experimental setting, combining the six machine-learning techniques of the original interplay study with six sampling strategies tailored to software configuration spaces and ten subject systems. Concretely, we evaluate these combinations in terms of prediction accuracy and stability, i.e. how much the prediction error varies across independently drawn samples. We additionally examine the sample size as a separate experimental factor and analyse its effect on prediction accuracy and stability.

Our results show that the pairing of machine-learning technique and sampling strategy primarily affects accuracy, but only once the sample is large enough. Techniques that do not depend on dense local training coverage, such as tree-based and regression techniques, achieve lower error, while stability remains largely unaffected by the pairing. Furthermore, more data consistently lowers the error, but the point at which additional data stops paying off depends on the learning technique. Stability also improves with more data, but without a technique-specific pattern. In conclusion, this thesis indicates that the reported interplay generalises for accuracy but not for stability.

Acknowledgments

To begin with, I would like to express my sincere gratitude to my advisors, Kallistos Weis and Tobias Dick, for their continuous support throughout this thesis and for always encouraging me to give my best. This work would not have been possible without their valuable feedback and guidance.
I would also like to thank Prof. Dr. Sven Apel and Prof. Dr. Martina Maggio for reviewing this thesis.
Special thanks go out to Nils Husung for his support whenever it was needed.
With love and deep appeciation, I thank Michael Hendrik Echternach, Marlon Felix Kraus, Corinna Jager and Lena Ramona Stegmann for their constant guidance, help and support throughout my academic journey, especially during the most challenging times.
Finally, I would like to thank everyone who supported me during this phase, in particular my colleagues at work.

Contents

List of Figures xi
List of Tables xii
1 Introduction 1
2 Background 3
2.1 Configurable Systems and Feature Models . . . 3
2.1.1 Feature Modelling . . . 4
2.2 Sampling Strategies . . . 5
2.3 Performance Prediction using Machine Learning . . . 6
2.3.1 Machine Learning Categories . . . 8
2.3.2 Prediction Evaluation . . . 9
3 Methodology 11
3.1 Research Questions . . . 11
3.2 Operationalisation . . . 12
4 Evaluation 19
4.1 Results . . . 19
4.1.1 RQ1: Combination of machine-learning techniques and sampling strategies . . . 19
4.1.2 RQ2: Impact of the sample size . . . 26
4.2 Discussion . . . 30
4.2.1 RQ1: Combination of machine-learning techniques and sampling strategies . . . 31
4.2.2 RQ2: Impact of the sample size . . . 33
4.3 Threats to Validity . . . 35
4.3.1 Threats to Internal Validity . . . 35
4.3.2 Threats to External Validity . . . 37
4.4 Future Work . . . 38
5 Related Work 39
5.1 Performance Prediction for Configurable Systems . . . 39
5.2 Sampling Strategies . . . 40
6 Conclusion 43
A Appendix 45
A.1 Additional Results for RQ1 . . . 45
A.1.1 Pairwise combination matrices at smaller reference sizes . . . 45
A.1.2 Aggregated boxplots at smaller reference sizes . . . 48
A.1.3 Per-system boxplots for prediction error at $t = 3$ . . . 49
A.1.4 Mean-rank matrices . . . 54
A.2 Hyperparameters . . . 55

B Statement on the Usage of Generative Digital Assistants 57

Bibliography 59

List of Figures

Figure 2.1 Feature model for a hypothetical mobile application 4
Figure 2.2 Workflow of machine-learning-based performance prediction 8
Figure 4.1 Pairwise combination comparison of learning techniques for $\overline{MRE}$ at $t = 3$ 20
Figure 4.2 Pairwise combination comparison of learning techniques for Var at $t = 3$ 21
Figure 4.3 Kendall's W at each reference sample size. 22
Figure 4.4 Per-combination distributions of $\overline{MRE}$ at $t = 3$ 23
Figure 4.5 Per-combination distributions of Var at $t = 3$ 24
Figure 4.6 Median $\overline{MRE}$ across reference sample sizes 25
Figure 4.7 Median Var across reference sample sizes 25
Figure 4.8 Evolution of $\overline{MRE}$ per subject system 27
Figure 4.9 Evolution of Var per subject system 28
Figure 4.10 Significant $\overline{MRE}$ reductions per proportion transition 29
Figure 4.11 Significant Var reductions per proportion transition 30
Figure A.1 Pairwise combination comparison for $\overline{MRE}$ at $t = 1$ 46
Figure A.2 Pairwise combination comparison for $\overline{MRE}$ at $t = 2$ 46
Figure A.3 Pairwise combination comparison for Var at $t = 1$ 47
Figure A.4 Pairwise combination comparison for Var at $t = 2$ 47
Figure A.5 Per-combination $\overline{MRE}$ distribution at $t = 1$ 48
Figure A.6 Per-combination $\overline{MRE}$ distribution at $t = 2$ 48
Figure A.7 Per-combination Var distribution at $t = 1$ 49
Figure A.8 Per-combination Var distribution at $t = 2$ 49
Figure A.9 Per-combination $\overline{MRE}$ distribution for 7z at $t = 3$ 50
Figure A.10 Per-combination $\overline{MRE}$ distribution for BerkeleyDBC at $t = 3$ 50
Figure A.11 Per-combination $\overline{MRE}$ distribution for dune at $t = 3$ 51
Figure A.12 Per-combination $\overline{MRE}$ distribution for HIPAcc at $t = 3$ 51
Figure A.13 Per-combination $\overline{MRE}$ distribution for lrzip at $t = 3$ 52
Figure A.14 Per-combination $\overline{MRE}$ distribution for JavaGC at $t = 3$ 52
Figure A.15 Per-combination $\overline{MRE}$ distribution for LLVM at $t = 3$ 53
Figure A.16 Per-combination $\overline{MRE}$ distribution for polly at $t = 3$ 53
Figure A.17 Per-combination $\overline{MRE}$ distribution for vp9 at $t = 3$ 54
Figure A.18 Per-combination $\overline{MRE}$ distribution for x264 at $t = 3$ 54
Figure A.19 Mean-rank matrix for $\overline{MRE}$ 55
Figure A.20 Mean-rank matrix for Var 55

List of Tables

Table 3.1 Overview of the sampling strategies including their respective abbreviations 13
Table 3.2 Overview of the machine-learning techniques including their respective abbreviations 13
Table 3.3 Overview of the subject systems 17
Table A.1 Applied hyperparameter values of this thesis. 56

1 Introduction

Modern software systems are typically developed as highly configurable systems that support different user requirements and usage scenarios [3, 41]. A configuration of such a system is defined by a specific selection of configuration options, i.e. switches that enable or disable functionalities of the system [4, 50]. Different configurations can influence how the system behaves with respect to non-functional properties, such as performance (e.g. execution time or throughput) or energy consumption [29, 54]. Understanding the effect of configuration options on these properties is therefore important for selecting configurations that meet given performance requirements and for reasoning about the system's behaviour. Because such properties are usually only observable by executing and measuring the system, performance analysis often requires measurements taken throughout various experimental runs [13, 54]. In principle, measuring all valid configurations would provide an overview of the system's overall performance behaviour. In this sense, constraints, such as mutual exclusions between configuration options, can reduce the number of valid configurations, but the remaining valid configuration space can still grow exponentially with the number of options, which in real-world system software can reach into the thousands [3, 28, 56]. As a result, exhaustive measurement of all valid configurations does not scale.

To address this challenge, researchers have proposed machine-learning-based approaches that learn a predictive model from a small, representative sample set of configurations. Such models capture the relationship between configuration options and non-functional properties, allowing the prediction of the performance of unseen configurations. Model quality then depends on two key factors. For one, *sampling strategies* determine which configurations enter the sample set and how well they represent the configuration space [2]. The *machine-learning techniques*, the second key factor, are then fit on the resulting samples. Neither of them alone determines the resulting prediction quality. Kaltenecker et al. [28] showed that both the sampling strategy and learning technique influence the prediction error, and that meaningful interactions exist between the two.

This study by Kaltenecker et al. [28], however, was conducted under specific experimental conditions. It emphasised numeric configuration options and sampling strategies derived from design of experiments [38, 43], combined with six machine-learning techniques. This raises the question whether the reported interplay effect between sampling strategies and learning techniques remains robust when the sampling strategies are replaced by those designed for configuration-space structure rather than for experimental design. A complementary setting is provided by Kaltenecker et al. [29], who introduced a diverse set of subject systems and sampling strategies specifically tailored to software configuration spaces, but

evaluated only a single machine-learning technique, leaving the interplay between sampling and learning unexplored in that context.

This thesis replicates the study of Kaltenecker et al. [28] under experimental conditions derived from [29]. Concretely, we use the six machine-learning techniques from [28] and six sampling strategies studied in [29] to train and evaluate predictive models across ten real-world configurable systems at three reference sample sizes with regard to prediction error and stability. This way, we test whether the observation that the choice of sampling strategy and learning technique together affect the prediction quality, rather than either factor alone, extends from that setting to configuration-space-tailored sampling. Beyond replicating this effect, we additionally examine how prediction error and stability change as the training-sample size grows by using six proportions of the valid configuration space.

Our replication shows that the combination effect between sampling strategy and learning technique expresses itself more strongly through prediction error than through stability. This asymmetry emerges only once the samples are large enough, where a clear ranking of combinations becomes visible. Specifically, techniques that do not depend on dense local coverage, such as tree-based and regression techniques, achieve a lower error. As the sample grows further, prediction error decreases for almost all combinations, but different learning techniques saturate at different points. Stability likewise improves with more data, but with no learning-technique-specific pattern.
Which sampling strategy is paired with which learning technique therefore matters, but only for sufficiently large samples and also only for the prediction error, not for its stability across experimental runs. The previously reported effect in [28] under the original setting hence generalises to sampling strategies tailored to configuration spaces for the accuracy, while leaving stability essentially unaffected by the pairing.

2

Background

This chapter introduces the fundamental concepts required for this thesis. It covers configurable software systems, feature modelling and sampling strategies. Finally, it presents machine-learning-based performance prediction and the evaluation of prediction accuracy and stability.

2.1 Configurable Systems and Feature Models

A *configurable software system* is characterised by a finite set of configuration options that parametrise the system's behaviour [32]. In this thesis, we focus on *Boolean configuration options*, each of which either enables or disables corresponding functionalities of the software system [4]. The selection of a set of such configuration options is also known as a *configuration*.

Definition 1 (Configuration)**.** Let $o \in \mathcal{O}$ be defined as a single configuration option from the set $\mathcal{O}$ of configuration options of the system. A *configuration* is a function that assigns each configuration option o a Boolean value, such that

$$c : \mathcal{O} \to \{0, 1\}.$$

If an option is selected, a 1 is assigned, while an assignment of 0 indicates that the option is deselected. The set of all configurations is called the *configuration space*, which is defined by $\mathcal{C} = \{0, 1\}^{\mathcal{O}}$.

Usually not every configuration is permissible. The selection of options is restricted by *constraints*, for example when two options are mutually exclusive or when one requires another. Every configuration c that adheres to these constraints is known as a *valid configuration*. We denote the set of all valid configurations as $\mathcal{V} = \{c \in \mathcal{C} \mid c \text{ satisfies all constraints}\}$ with $\mathcal{V} \subseteq \mathcal{C}$. Although constraints determine which configurations are valid, they do not capture how the selection of options affects the system's behaviour. The effect of one option may depend on which other options are selected. When the combined effect of two or more options differs from the sum of their individual effects, we call this an *interaction*. The two-option case is a *pairwise interaction* and all subsequent combinations of $n \geq 2$ options are called *n-way interactions*.

2.1.1 Feature Modelling

To effectively capture the variability and constraints of a configurable software system, *feature modelling* was introduced by Kang et al. [30]. In the terminology of feature models, a configuration option is called a feature. We use these two terms interchangeably. Features are arranged in a tree structure that represents the valid configuration space $\mathcal{V}$. A parent-child edge encodes an implication, requiring that the child can only be chosen when its parent is. Further, children may be grouped. An *Or Group* requires at least one of its members, while an *Alternative Group* requires exactly one when the parent is selected. Beyond this grouping, a feature is either *mandatory* and selected in every valid configuration or *optional*. Constraints that the tree cannot express are added as separate logical expressions over features. We refer to these constraints as *cross-tree constraints*. Figure 2.1 shows an example illustration of a feature model for a hypothetical mobile application.

Example 1 (Mobile Application)**.** In Figure 2.1, *Mobile Application* is the root feature and is contained in every configuration. *Platform* is a mandatory feature and forms an *Or Group* over *Android* and *IOS*, requiring that at least one platform is always selected. *Notification* on the other hand is optional and forms an *Alternative Group* over *Push-Notification* and *App-Notification*, such that whenever *Notification* is selected, exactly one of them must be chosen. The cross-tree constraint *Push-Notification* $\Rightarrow$ *Android* additionally imposes a constraint on the valid configuration space of our model. This means that whenever *Push-Notification* is selected, then *Android* must also be selected.

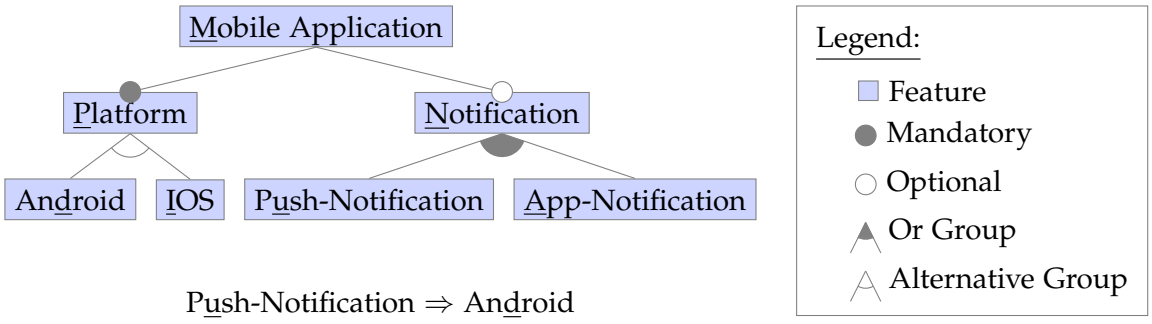


Figure 2.1: Feature model for a hypothetical mobile application

With regard to our previous definitions in this section, our feature model contains the following set of configuration options $\mathcal{O} = \{m, p, n, d, i, u, a\}$. Each feature is represented using its underlined letter in the model. A configuration is represented as a string of its selected features. Take for example the following valid configuration $c_1 = mpdnu$, which selects an *Android* (d) device using *Push-Notification* (u). The constraint forces d once u is chosen, showing an interaction that the separate subtrees rooted in *Platform* and *Notification* do not capture on their own.

2.2 Sampling Strategies

To effectively reason about large configuration spaces, *sampling strategies* are used to select a representative subset, also known as a *representative sample*, which not only contains relevant configuration options, but also interactions between these options, to accurately perform analysis and measurements [29]. A *sampling strategy* defines the criteria and distributions according to which configurations are selected from $\mathcal{V}$ and thus which parts of the configuration space the resulting sample represents [29]. Following the choice of the sampling strategy, the *size of the sample* plays an important role in determining how well the selected configurations represent the underlying configuration space. Intuitively, larger samples cover a greater part of the variability of the system, whereas smaller samples may leave out relevant constraints or interactions, resulting in a reduced representativeness.

Definition 2 (Sample Set)**.** A *sample set* $\mathcal{S} \subseteq \mathcal{V}$ is a subset of the valid configurations of a system. Its *sample size* is the number of configurations it contains, denoted as $|\mathcal{S}|$.

There are many different strategies, that select configurations according to different criteria. In this thesis, we consider the following six sampling strategies.

Random Sampling (`random`). Random sampling aims to draw configurations from $\mathcal{V}$ such that each valid configuration $c \in \mathcal{V}$ has the same probability of being selected. However, this is difficult in practice, because $\mathcal{V}$ is not enumerated, but rather defined through its constraints, so valid configurations have to be directly drawn from them. *Binary Decision Diagrams* [1] have therefore been proposed to represent $\mathcal{V}$ as a compact data structure from which valid configurations can be drawn [9, 37].

***t*-wise Sampling (`twise`).** t-wise sampling is a coverage-oriented strategy. It draws configurations to cover all valid combinations of t-many selected configuration options [29]. For $t = 1$, each option must be selected in at least one configuration of the sample; for $t = 2$, every valid pair of options must be selected together in at least one configuration; and analogously for higher values of t [36]. Although a larger t captures higher-degree interactions, t-wise sampling only covers a limited part of the configuration space and therefore omits interactions of a degree higher than t [29]. The sample size produced by this sampling strategy at a given t also serves as a *reference sample size* for other sampling strategies. This provides a comparable set of samples across strategies when reasoning about metrics derived from these samples.

Solver-based Sampling (`sbs`). Solver-based sampling explicitly accounts for constraints by utilising constraint solvers, such as SAT solvers, to generate valid configurations [25]. In this process, the solver is typically called repeatedly such that the first k solutions are selected during sampling [29], where k is the target sample size $|\mathcal{S}|$.

Randomised solver-based Sampling (`rsbs`). As the solver searches for new solutions located in the neighbourhood of already discovered ones, the resulting drawn configurations are often locally clustered, leading to skewed samples [29]. *Randomised solver-based sampling* addresses this by applying randomisation, varying the order in which options are considered or the order of solutions returned [23, 24]. Unlike *solver-based sampling*, different random seeds produce different samples.

Distance-based Sampling (`dbs`). This sampling strategy spreads the sample across the configuration space using a *distance metric* and probability distribution. Each valid configuration is assigned a *distance value*, here, this is the number of selected configuration options, which is equivalent to the *Hamming distance* to the empty configuration [18]. Configurations are then drawn so that their associated distance values follow a chosen probability distribution, such as the uniform distribution. Because valid configurations at very low or very high Hamming distances are typically far fewer than at intermediate ones, drawing uniformly over distance values tends to over-represent the extremes of the configuration space, compared to random sampling, whose samples reflect how the distance values are distributed in $\mathcal{V}$ [29].
Diversified distance-based Sampling (`ddbs`). Because many configurations may share the same distance value, the choice among them determines how well the sample covers the configuration space. *Diversified distance-based sampling* refines this selection. Among configurations with the same distance value, it prefers the one containing the least frequently selected option, such that diversity of the resulting sample is increased [29].

These sampling strategies can be grouped into two categories with respect to their reproducibility. We call *deterministic* those strategies that produce identical samples across repeated processes involving the same parameters and *stochastic* those that incorporate random choices and can therefore produce different samples across such repetitions. Under this definition, *solver-based sampling* and *t-wise sampling* are deterministic, while all other aforementioned strategies are stochastic.

2.3 Performance Prediction using Machine Learning

Machine learning refers to those computational techniques or algorithms that learn patterns from data in order to make predictions or decisions without being programmed explicitly for each task [5, 47]. In the context of configurable systems, machine learning can be used to learn the relationship between configuration choices and the resulting performance. Using sampling strategies on a feature model, we can generate representative *training data*, consisting of measured performance values for chosen configurations. The resulting data is then used by the learning technique to predict the performance of unmeasured configurations from a limited set of measured ones [29].
Performance, in this context, is a *non-functional property* of the system. A non-functional property describes how well a system performs, rather than what it does [8]. Performance can be measured for a given configuration, for example as its response time, throughput or memory usage [3, 58]. We describe it with a function over the valid configurations.

Definition 3 (Measured and Predicted Performance)**.** Let $c \in \mathcal{V}$ be a valid configuration. The *measured performance* is a function

$$\Pi : \mathcal{V} \to \mathbb{R}$$

where $\Pi(c)$ is the value measured for configuration c. A machine-learning technique approximates Π by the *predicted performance*

$$\hat{\Pi} : \mathcal{V} \to \mathbb{R}$$

where $\hat{\Pi}(c)$ is the value predicted for c.

A learning technique does not observe the measured performance Π directly; instead, it looks at values on a sampled subset of configurations. These configuration-performance pairs form the *training data*.

Definition 4 (Training Dataset)**.** Let $\mathcal{S} \subseteq \mathcal{V}$ be a sample set and Π be the measured performance. We define the training dataset as the set of all pairs of a sampled configuration and its measured performance, such that

$$\mathcal{D} = \{(c_i, \Pi(c_i)) \mid c_i \in \mathcal{S}\}$$

From the training dataset, a learning technique generates a *model*. This model is a function $\hat{\Pi}$ that approximates the measured performance Π and predicts the performance of configurations outside the sample, i.e. $\hat{\Pi}(c)$ for $c \notin \mathcal{S}$. The resulting structure of the model depends on the underlying learning technique [28]. For regression-based techniques, for example, it takes the form of an *influence model*, which represents performance as a linear combination of terms, where each term captures the influence of a configuration option or an interaction of options on performance, weighted by its effect [28].

Figure 2.2 illustrates how these components connect end-to-end, from sampling to prediction, exemplified on the mobile-application system.

Example 2 (Performance Prediction Workflow)**.** We base this example on the mobile-application system from Figure 2.1. Specifically, we consider three valid configurations $\{c_1, c_2, c_3\}$ that have been sampled from the system using an arbitrary sampling strategy. The provided training dataset

$$\mathcal{D} = \{(c_1, 5), (c_2, 8), (c_3, 9)\}$$

displays the respective configuration alongside their measured performance value $\Pi(c_i)$. During the *learning process*, a model $\hat{\Pi}$ is learned from this training dataset and then used to compute predicted performance values $\hat{\Pi}(c_j)$ of further, unseen configurations $\{c_4, c_5\}$. As shown in Figure 2.2, the predicted values differ from measured ones, a discrepancy that will be quantified formally by the relative error introduced in (Definition 5) below.

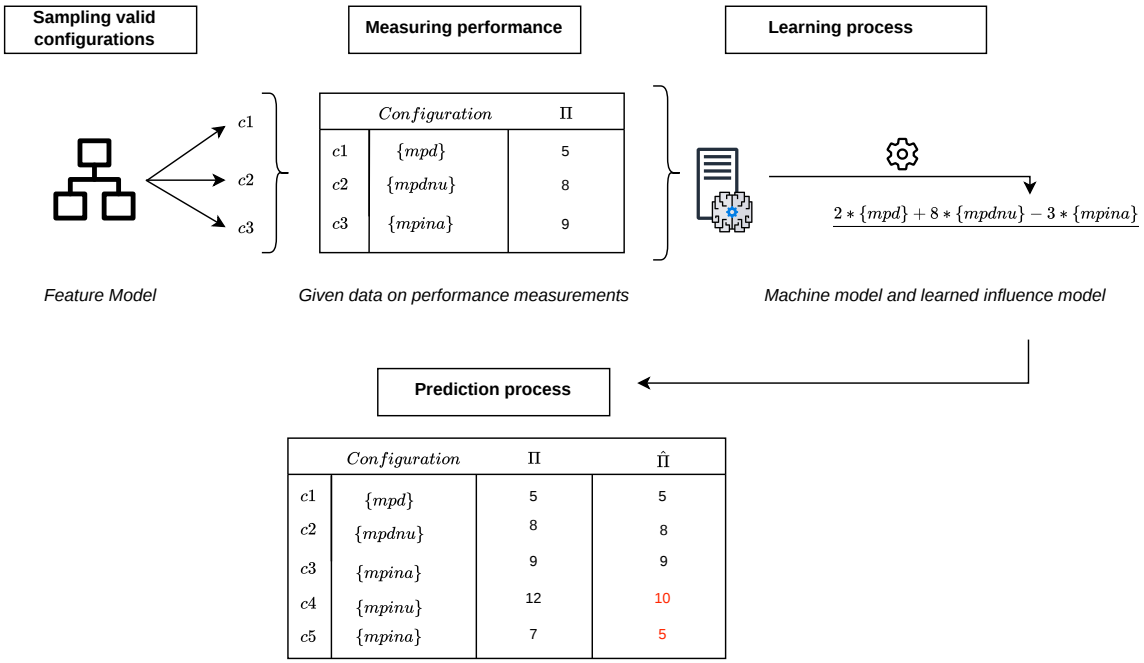


Figure 2.2: Workflow of machine-learning-based performance prediction

2.3.1 Machine Learning Categories

There are various categories into which machine learning models can be classified. One such category is *supervised learning models*, where models are trained on labelled data, such that input-output pairs are provided. The labels provide the target values against which the model is trained. We further classify them into subclasses.

Linear models. This category of models assumes a linear relationship between input variables and the target [27]. In this context, a linear relationship between configuration options and performance would be assumed. Multiple Regression (MR) predicts performance as a linear combination of the configuration options and their pairwise interactions, with coefficients estimated globally from the entire training sample [50]. This corresponds to the influence model introduced above Figure 2.2.

Tree-based learning models. In these models the predictions are built in a tree-like structure using a hierarchy of decision rules. Internal nodes test feature conditions whilst leaves output the predicted value. The tree splits the training configurations step by step, one option at a time and predicts the average performance of all training configurations that end up in the same leaf [19, 28]. A prediction therefore only relies on training configurations that match the predicted configuration in the few options tested along its path through the tree, not on training configurations that are similar to it overall.

Instance-based learning models. These models do not explicitly learn a parametric model during the training phase but instead compare stored data to new input data and thus analyse similarities. *k*-Nearest-Neighbours Regression (*k*NN) is encompassed in this model class [15]. Because a prediction is derived from the training data closest to the prediction point, *k*NN's accuracy consequently depends on how densely the training sample covers

the neighbourhood around each prediction point [20].
Kernel-based learning models. Kernel-based models predict the performance of a configuration by comparing it to the training configurations. A kernel-function measures how similar two configurations are, which allows the model to capture non-linear relationships between options and performance. A regularisation term is additionally applied to keep the model from being overly complex, which improves generalisation. This class includes Support Vector Regression (SVR) and Kernel-Ridge Regression (KRR) [21]. The two differ in which training configurations shape a prediction. KRR uses a ridge penalty with every training configuration contributing to each prediction. SVR instead ignores errors smaller than a threshold ϵ, so its predictions depend only on a subset of the training data, the support vectors. [21].
Ensemble-based learning models. This model class incorporates multiple learners (i.e. machine learning models) and combines their predicted values for higher accuracy [52]. Specifically, Random Forest (RF) belongs to this model class, as it combines multiple decision trees to improve prediction accuracy [22, 28].

2.3.2 Prediction Evaluation

The predicted performance generally differs from the measured performance. We can quantify this difference for a single configuration using the *relative error*.

Definition 5 (Prediction Error)**.** Let c_i be a configuration with measured performance $\Pi(c_i)$ and the predicted performance $\hat{\Pi}(c_i)$. The *relative error* for a single configuration is defined as the absolute difference between its measured and predicted performance, normalised by the measured performance, such that

$$RE(c_i) = \frac{|\Pi(c_i) - \hat{\Pi}(c_i)|}{|\Pi(c_i)|}.$$

To assess a technique beyond a single configuration, we average the relative error over an *evaluation set* $\mathcal{E} \subseteq \mathcal{V}$, which consists of the configurations not contained in the sample, $\mathcal{E} = \mathcal{V} \setminus \mathcal{S}$. As several sampling strategies are stochastic, the resulting predictions vary between runs. We therefore repeat the process over R runs and denote a single sample run with $r \in \{1, \ldots, R\}$, making R the highest run index. In our case, one run corresponds to a single sample generation, followed by training the learning technique on that sample and computing its *mean relative error* on the corresponding *evaluation set*. The *mean relative error* of a single run is defined as

$$MRE_r = \frac{1}{|\mathcal{E}|} \sum_{c_i \in \mathcal{E}} RE(c_i).$$

For deterministic strategies we set the maximal run index to $R = 1$, as the samples do not vary across runs. Consequently we can define the *prediction error* across all runs as the arithmetic mean of the per-run errors

$$\overline{MRE} = \frac{1}{R} \sum_{r=1}^{R} MRE_r$$

The closer the prediction error is to 0, the higher the accuracy of the learning technique's prediction [29].

Further, due to the imposed randomness of sampling strategies, it is also important to determine how prediction error varies when a combination of sampling strategy and technique is retrained on independently drawn samples. We measure the *stability* of such a combination as the variance of its MRE_r across the R runs,

$$Var = \frac{1}{(R-1)} \sum_{r=1}^{R} (MRE_r - \overline{MRE})^2$$

where a lower variance indicates more stable predictions across runs. By convention, we set $Var = 0$ for deterministic sampling strategies, as no variance is defined for $R = 1$. These strategies are then excluded from all stability analyses.

3

Methodology

This chapter describes the methodology used to answer our research questions. Section 3.1 motivates them. Section 3.2 describes the experimental setup and the applied statistical procedures to evaluate our research questions.

3.1 Research Questions

To effectively predict the performance of configurable software systems, both the choice of machine-learning technique and the sampling strategy play an important role. Choosing an appropriate sampling strategy affects how accurately a machine-learning technique can predict the performance [13, 29]. In particular, the chosen sampling strategy decides how well the underlying configuration space of the system is represented, which in turn influences the quality of the resulting performance model learned by the machine-learning technique. Because neither of these two components alone influences prediction quality, understanding how the two interact is essential.
Kaltenecker et al. [28] studied this interplay on ten subject systems with both binary and numeric configuration options. They combined binary sampling strategies with numeric ones from statistical design of experiments, such as Plackett-Burman designs [38, 43] and paired these combinations with six machine-learning techniques. It is unclear whether the reported findings still hold for binary-only subject systems and for sampling strategies, related to the structure of software configuration spaces, such as the solver-based and distance-based strategies, which have been studied by Kaltenecker et al. [29]. The latter two strategies were not part of the original study. We therefore replicate this interplay under the experimental conditions of [29].

Impact of the interplay between machine-learning techniques and sampling strategies on performance models. We assess each pair of sampling strategy and machine-learning technique along two metrics. The first is the *prediction accuracy*, which describes how closely the predicted performance approximates the measured one. A lower prediction error therefore indicates a higher prediction accuracy, whereas a higher prediction error indicates a less effective one. The second metric is the *stability* of the predictions, i.e. how much the prediction error varies when a combination is retrained on independently drawn samples (see Section 2.3.2). A higher variance therefore indicates lower stability, while a lower variance implies consistent prediction error across samples. With these two evaluation criteria, we state our first research question.

RQ1 (Combination of machine-learning techniques and sampling strategies): How does the interplay of machine-learning techniques and sampling strategies affect the prediction error and stability of performance models?

How prediction accuracy and stability are affected by the sample size. A key challenge in performance prediction in real-world configurable systems is the potentially large size of the configuration space [56]. This makes exhaustive analysis of the respective system computationally difficult. Existing approaches therefore rely on sampling strategies to obtain a representative subset of configurations [3, 17], which is then used as training data. This raises the question of how the size of the training data, i.e. the *sample size*, affects prediction quality. With small sample sizes, the machine-learning technique may lack sufficient data to capture the underlying relationships of the configurable system, resulting in lower prediction accuracy. Conversely, larger sample sizes may improve prediction accuracy, possibly with diminishing returns. Moreover, with small samples, the result depends more strongly on which configurations are drawn, such that sample size may affect stability. In addition to that, different machine-learning techniques may require different amounts of training data to achieve good performance. Therefore, it is essential to analyse how prediction accuracy and stability evolve with increasing sample sizes across different learning techniques, which leads us to our second research question.

RQ2 (Impact of the sample size): How does the sample size affect the prediction error and stability of machine-learning techniques when predicting the performance of configurable systems?

3.2 Operationalisation

In order to answer our research questions, we describe the experimental setup, operationalisation of the research questions and the statistical methods used to assess the results. We measure prediction error by the mean relative error $\overline{MRE}$ over all experimental runs (see Definition 5) and stability by its variance Var across runs (see Section 2.3.2).

Scope of the replication. This thesis conducts a conceptual replication of Kaltenecker et al. [28], who report the results of an empirical study by Grebhahn et al. [12]. That is, we keep the object of study, which is the analysis of the interplay between sampling strategies and machine-learning techniques with respect to prediction error and stability, but modify its experimental setting. This means that the six machine-learning techniques of the original study are preserved and combined with six binary sampling strategies from [29]. Two of these, `random` and `twise` sampling, were already part of the original work, while the remaining ones were not. Similarly, the original six subject systems are replaced with the ten binary-only subject system from [29]. Since [28] refers to [12] for a detailed description of the experimental setup, the details, that were not reported in [28], are taken from Grebhahn et al [12]. Beyond the replication, we additionally examine the sample size as a separate factor in our second research question.

Interplay of sampling strategies and machine-learning techniques. The two components that form the core of our evaluation are the sampling strategies and machine-learning techniques. Table 3.1 and Table 3.2 list the six sampling strategies and six machine-learning techniques together with their abbreviations introduced in Section 2.2 and Section 2.3.1, which we use throughout the remainder of this thesis.

Sampling strategy	**Abbreviation**
Random	`random`
t-wise ($t \in \{1, 2, 3\}$)	`twise`
Solver-based	`sbs`
Randomised solver-based	`rsbs`
Distance-based	`dbs`
Diversified distance-based	`ddbs`

Table 3.1: Overview of the sampling strategies including their respective abbreviations

Machine-learning technique	**Abbreviation**
Classification and Regression Trees	CART
k-Nearest-Neighbours Regression	kNN
Kernel-Ridge Regression	KRR
Multiple Regression	MR
Random Forest	RF
Support Vector Regression	SVR

Table 3.2: Overview of the machine-learning techniques including their respective abbreviations

Sample sizes. For the comparison in *RQ1*, all sampling strategies are required to produce samples of the same size, such that differences stem from the strategies themselves, instead of differing amounts of training data. Following Kaltenecker et al. [29], which defined three reference sample sizes by option-wise, pair-wise and three-wise sampling, we use t-wise sampling with $t \in \{1, 2, 3\}$ to determine three increasing reference sample sizes per subject system. This extends [28], in which only random sampling was matched to the sample size of t-wise sampling. Every other sampling strategy is then executed until it reaches the respective reference size. Besides that, only t-wise sampling itself guarantees the corresponding interaction coverage, while the other strategies produce samples of the same size. For *RQ2*, the effect of the sample size is isolated by omiting the interaction coverage. Instead, samples of fixed proportions $\pi \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$ of the valid configuration space $\mathcal{V}$ are drawn, such that one sample contains $\pi \cdot |\mathcal{V}|$ configurations. The chosen proportions cover small to moderate training sets and allow to analyse how the prediction error and stability evolve as the amount of training data increases.

Hyperparameters. Most machine-learning techniques have *hyperparameters* that influence their learning behaviour and, in turn, their prediction error [28, 44]. We tune them separately for each combination of subject system, learning technique and sample size on a random sample of the respective size, whose seed is not used in any evaluation run and select the hyperparameter values with the lowest mean relative error in a five-fold cross-validation. In contrast to the original study, which tuned them for every combination of sampling strategies [28], the same hyperparameters for all strategies of a given sample size are used. Otherwise, each strategy would then be evaluated with hyperparameters fitted to its own sample and the difference in prediction error between strategies may partly stem from tuning rather than from the samples themselves. Further, all combinations of hyperparameter values, except for SVR, whose value sets yield 288 combinations are evaluated. For this technique, random search is utilized, which evaluates 30 randomly chosen combinations to limit tuning time, similar to [12]. With a probability of 95 per cent, these include at least one of the ten per cent of combinations with the lowest error [7].
As a reference for the value sets, we use the values selected for the tuning runs of Grebhahn et al. [35], which Table A.1 compares to ours. The value sets of kNN, KRR and RF are reduced, as tuning with the full sets exceeded the 24-hour limit. Moreover, random seeds of CART and RF are fixed instead of tuned, as a seed that performs well on the tuning sample does not have to perform well on other samples. Finally, we approximate the stepwise regression of the original study by a Lasso regression on the options and their pairwise interactions. Its only hyperparameter, the regularisation strength, therefore has no counterpart in the original study.

Computational limit. Beyond the original study, a limit of 22,000 configurations on the training set for kNN, KRR and SVR on the subject systems JavaGC and VP9 is imposed, following the crossing of the 24-hour limit. We chose this value such that all proportions up to $\pi = 0.10$ remain unaffected on both systems (19,354 configurations for JavaGC and 21,600 for VP9). The cap does not affect *RQ1*, since the t-wise reference sample sizes remain well below it. In *RQ2*, however, the two largest proportions ($\pi \in \{0.15, 0.20\}$) exceed the cap on both systems. For these, the techniques are tuned and trained on a random subsample of 22,000 configurations. Since their training sample thus no longer correspond to the respective proportion, we exclude the corresponding 12 combinations of subject system, learning technique and proportion from the analysis. The remaining combinations are analysed as described in the *RQ2* evaluation paragraph below.

Variables and procedures. Each experiment is defined by a subject system, a sampling strategy, a machine-learning technique and a sample size. For a given combination of these factors, the sampling strategy selects a sample set $\mathcal{S} \subseteq \mathcal{V}$. Together with the measured performance of its configurations, $\mathcal{S}$ forms the training dataset $\mathcal{D} = \{(c, \Pi(c)) | c \in \mathcal{S}\}$ (see Definition 4), on which the respective machine-learning technique is trained. The trained model then predicts the performance $\hat{\Pi}(c)$ (see Definition 3) for every configuration c in the evaluation set $\mathcal{E} = \mathcal{V} \setminus \mathcal{S}$ of the respective run (see Section 2.3.2). Because the stochastic

sampling strategies (`random`, `rsbs`, `dbs` and `ddbs`) produce different samples in each run, we repeat each experiment over $R = 100$ runs, where R is the maximal run index. The learning techniques themselves use a fixed random seed, so that all variation between runs stems from the sampling. For each run, denoted by the run index $r \in \{1, \ldots, 100\}$, we compute the relative error of every configuration in $\mathcal{E}$ and aggregate it into the mean relative error MRE_r (see Definition 5). The mean of these R-many values yields the prediction error $\overline{MRE}$ and their variance, the stability Var (see Section 2.3.2). To ensure comparability, the run index r is used as the random seed and the same seed is used across all sampling strategies within the same run. For the deterministic strategies `twise` and `sbs`, a single run suffices, as they always produce identical samples. We therefore set $R = 1$ in that case.

Evaluation of RQ1. For *RQ1*, we fix a subject system and one of the three reference sample sizes induced by t-wise sampling and evaluate every combination of sampling strategy and machine-learning technique under this condition to obtain $\overline{MRE}$ and Var per combination. To assess whether the resulting differences between combinations in prediction error and stability are statistically significant, an additional non-parametric test is applied, which does not assume that the data follows a particular distribution, since performance metrics are not guaranteed to be normally distributed [45, 53]. As all combinations are evaluated on the same subject systems, first the *Friedman test* [42] in its *Iman-Davenport variant* [26] is applied, which is better applicable for moderate sample sizes. For each subject system, it ranks the combinations by their $\overline{MRE}$ and then checks whether some combinations consistently receive better ranks across the systems than others. A significant result indicates that the combinations are not interchangeable, but not which of them differ. Only in this case, we compare all pairs of combinations with the *two-sided Wilcoxon signed rank test* [55], paired by subject system. A significant pair indicates that one combination achieves consistently lower values than the other across systems. The resultng p-value is correcetd with the *Benjamini-Hochberg* procedure [6]. The *Holm-Bonferroni* correction is not appplied, since with only ten subject systems, even the smallest p-value the Wilcoxon test can produce would not remain significant after correction. The entire procedure is once applied to $\overline{MRE}$ and once to Var, excluding deterministic strategies for Var, since their variance is zero by construction (see Section 2.3.2). When visualising the pairwise comparisons of two learning techniques, their difference is deemed significant if at least two thirds of the comparisons with a shared sampling strategy are significant. Additionally, Kendall's coefficient of concordance W [31] is considered, which shows how strongly the subject systems agree on the ranking of the combinations, ranging from 0 (no agreement) to 1 (complete agreement). Following Schmidt [49], the results around 0.1 are interpreted as very weak, 0.3 as weak, 0.5 as moderate, 0.7 as strong and 0.9 as unusually strong agreement.

Evaluation of RQ2. Beyond the original study, *RQ2* examines the sample size as a separate factor. We fix the sampling strategy to random sampling and obtain $\overline{MRE}$ and Var for each combination of subject system, machine-learning technique and proportion $\pi \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$. Since the prediction error is expected to decrease with the sample size, possibly with diminishing returns, i.e. a monotonic but not necessarily linear relationship [10, 46], we therefore use non-parametric tests, similarly to *RQ1*. To

assess whether prediction error and stability change monotonically with the increasing sample size, *Spearman's rank correlation coefficient* ρ [51] is used across the six proportions for each combination of subject system and learning technique. A negative value of ρ implies that the error or variance decreases as the sample grows. Its p-value indicates how likely a correlation of this strength would occur by chance if the two were unrelated. Since each correlation is based on only six proportions, the exact p-values for each of the 60 combinations of subject system and learning technique is computed and corrected using the *Benjamini-Hochberg* procedure. For the capped combinations, only four proportions remain, which is too few for the correlation to become significant.
While Spearman's ρ captures a monotonic trend across the proportions, it does not identify at which specific step the prediction error stops improving meaningfully. To assess whether each proportion transition yields a significant reduction in prediction error, we apply a *one-sided Wilcoxon signed-rank test* [55] on the five consecutive transitions $0.01 \rightarrow 0.02$, $0.02 \rightarrow 0.05$, $0.05 \rightarrow 0.10$, $0.10 \rightarrow 0.15$, $0.15 \rightarrow 0.20$. For each combination of subject system and learning technique, we test whether the prediction error at the larger proportion is significantly lower than at the smaller one. It pairs the errors of the same run, as run r uses the same seed at both proportions. For stability, we use a one-sided *Brown-Forsythe* test [11]. At each of the two proportions, it computes the absolute deviations of the per-run errors MRE_r from the median at that proportion and checks whether these deviations are smaller at the larger proportion, i.e., whether the spread of MRE_r decreases. Using the median instead of the mean makes the test robust against non-normal distributions [11]. For both metrics, we correct the p-values over all transitions that are not affected by the cap using the *Benjamini-Hochberg* procedure.

Subject systems. For this thesis, a set of ten real-world configurable systems of Kaltenecker et al. [29], listed in Table 3.3 is considered. They stem from different domains, ranging from file compression and databases to compilers and video encoders and vary in their number of configuration options and valid configurations. Since the training data affects the learning behaviour and prediction quality [16], this diversity supports the generalisability of our findings across configurable systems. However, in all systems, the measured performance property is an execution time, such as compression, compilation or encoding time. For the experimental evaluation, we combine each subject system with the sample sizes, sampling strategies and machine-learning techniques defined for the respective research question. As measured performance values $\Pi(c)$, we reuse the measurements[1] conducted by Kaltenecker et al. [29].

1 https://github.com/ChristianKaltenecker/Distance-Based_Data/tree/master/SupplementaryWebsite/MeasuredPerformanceValues

Subject system	$\lvert\mathcal{V}\rvert$	#Options	Performance metric
7z	68 640	44	Compression Time
BerkeleyDBC	2 560	18	Response Time
dune	2 304	32	Solving Time
HIPAcc	13 485	54	Solving Time
JavaGC	193 536	39	Garbage collection time
LLVM	1 024	11	Compilation Time
lrzip	432	19	Compression Time
polly	60 000	40	Runtime
vp9	216 000	42	Encoding Time
x264	1 152	16	Encoding Time

Table 3.3: Overview of the subject systems, including the number of valid configurations $\lvert\mathcal{V}\rvert$, number of configuration options and the performance metric measured.

4

Evaluation

This chapter presents the results of our evaluation. Section 4.1 displays the findings for each research question, Section 4.2 interprets them in the context of the conceptual replication study, and Section 4.3 discusses the threats to the validity of our results. Section 4.4 outlines directions for future work.

4.1 Results

This section presents the results for both research questions. First, the interplay effect of sampling strategies and machine-learning techniques on prediction error and stability at three reference sample sizes is analysed. Following *RQ1*, prediction error and stability are examined in terms of how they evolve under fixed random sampling across six proportions of the investigated configuration spaces for *RQ2*.

4.1.1 RQ1: Combination of machine-learning techniques and sampling strategies

To answer the first research question, all 36 combinations of sampling strategy and learning technique at the three reference sample sizes are compared, each determined by t-wise sampling for $t \in \{1, 2, 3\}$. First we report whether and where the combinations differ, then how they rank and finally how the learning techniques evolve with reference size.

Differences between combinations. The combinations differ significantly in prediction error once the sample is large enough. At the smallest reference size $t = 1$, the Friedman test finds no significant difference ($p = 0.24$). At $t = 2$ and $t = 3$, it does ($p = 1.9 \cdot 10^{-12}$ and $p = 4 \cdot 10^{-41}$), which means that some combinations achieve consistently lower errors than others across subject systems. The pairwise Wilcoxon tests locate these differences in 135 of the 630 pairs of comparisons at $t = 2$ and in 301 pairs at $t = 3$. Regarding stability, the Friedman test is only significant at $t = 3$ ($p = 5 \cdot 10^{-4}$, compared to $p = 0.38$ at $t = 1$ and $p = 0.23$ at $t = 2$) though none of the 276 pairwise comparisons remains significant after the statistical correction. The combinations therefore differ in stability only weakly with no pair standing out significantly.
Figure 4.1 shows where the differences in prediction error at $t = 3$ lie. Each off-diagonal block compares the technique of its row to its column and each cell within a block shows their median difference in prediction error across the subject systems under one sampling strategy. Blue cells mean that the row technique achieves a lower error, while red cells

that it achieves a higher error and white cells that both perform equally. A yellow border marks a block in which at least two thirds of the cells are significant. They are denoted as *border threshold*. The diagonal blocks show the absolute median error of each technique per sampling strategy in greyscale, where lighter cells indicate lower prediction error. As each cell only compares two techniques under the same sampling strategy, the figure isolates the effect of the learning technique. The same layout is used for stability in Figure 4.2, where deterministic strategies are excluded, leaving four cells per block, so that three significant cells reach the border threshold.
Six pairs of techniques differ significantly and each of them sets CART, RF or MR against *k*NN or SVR. In all these pairs, CART, RF or MR show a lower median error under every sampling strategy. KRR lies in between. It shows lower errors than *k*NN and KRR, but higher errors than CART, RF and MR, although none of these differences reach the border threshold. This is confirmed by the diagonal blocks, where CART, RF and MR show the lightest cells across all sampling strategies, while *k*NN and SVR show the darkest ones. For KRR, mainly the cell of `dbs` stands out with the highest prediction error for this technique.

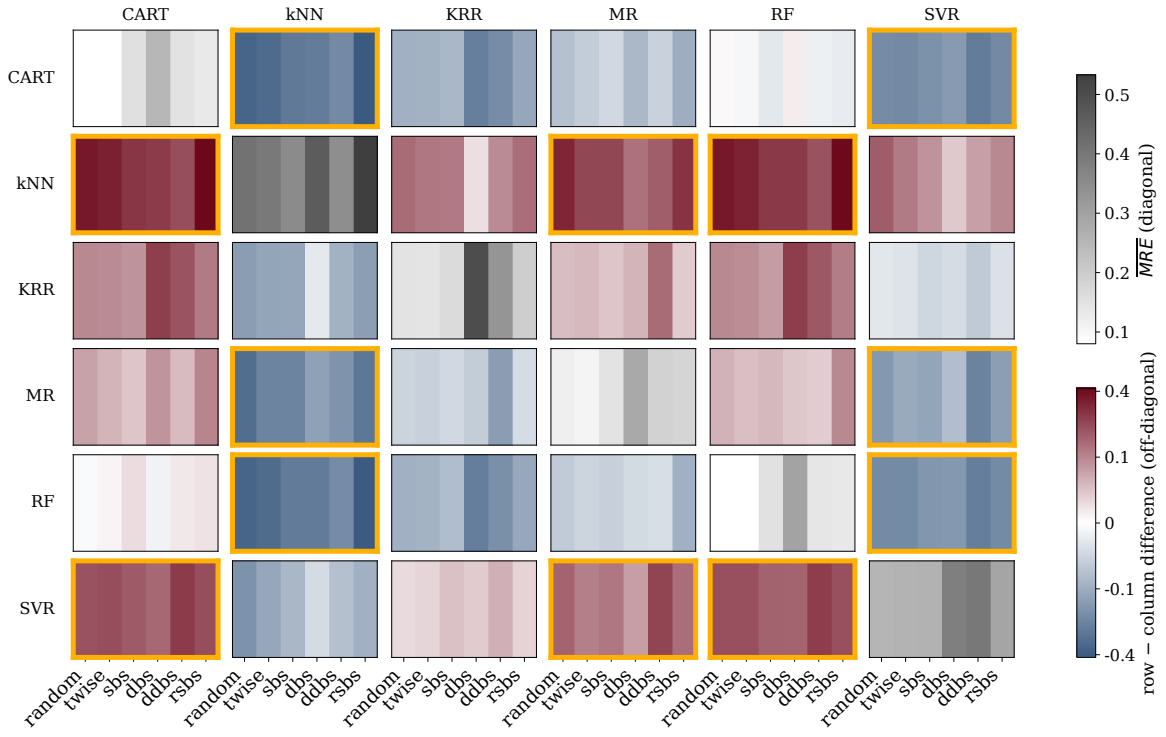


Figure 4.1: Pairwise combination comparison of learning techniques by median prediction error ($\overline{MRE}$) across the subject systems at $t = 3$, per sampling strategy. Off-diagonal blocks correspond to a pair of learning techniques (row, column), encoding the row-minus-column difference per shared sampling strategy on a blue-to-red scale (blue = row performs better, red = column performs better, white = equally stable). Diagonal blocks show absolute values in greyscale. Yellow-bordered blocks mark significant pairs.

At $t = 2$, only one pair differs significantly, with SVR achieving lower prediction error than *k*NN (Figure A.2). At $t = 1$, no pair differs significantly (Figure A.1).
For stability, no pair of learning techniques differs significantly at any reference size and

no block in Figure 4.2 reaches the border threshold. The largest differences occur in the row of *k*NN, whose variance is high with `random`, `rsbs` and low with `ddbs`. The row of RF, on the other hand, is predominantly blue, showing that this technique is more stable than others under most sampling strategies. The diagonal blocks show that the absolute variance depends on the sampling strategy. For *k*NN, it is the lowest with `ddbs` and highest with `rsbs`, while RF shows low variance under all four strategies. These differences, however, are not large or consistent enough to be pinpointed by pairwise tests.

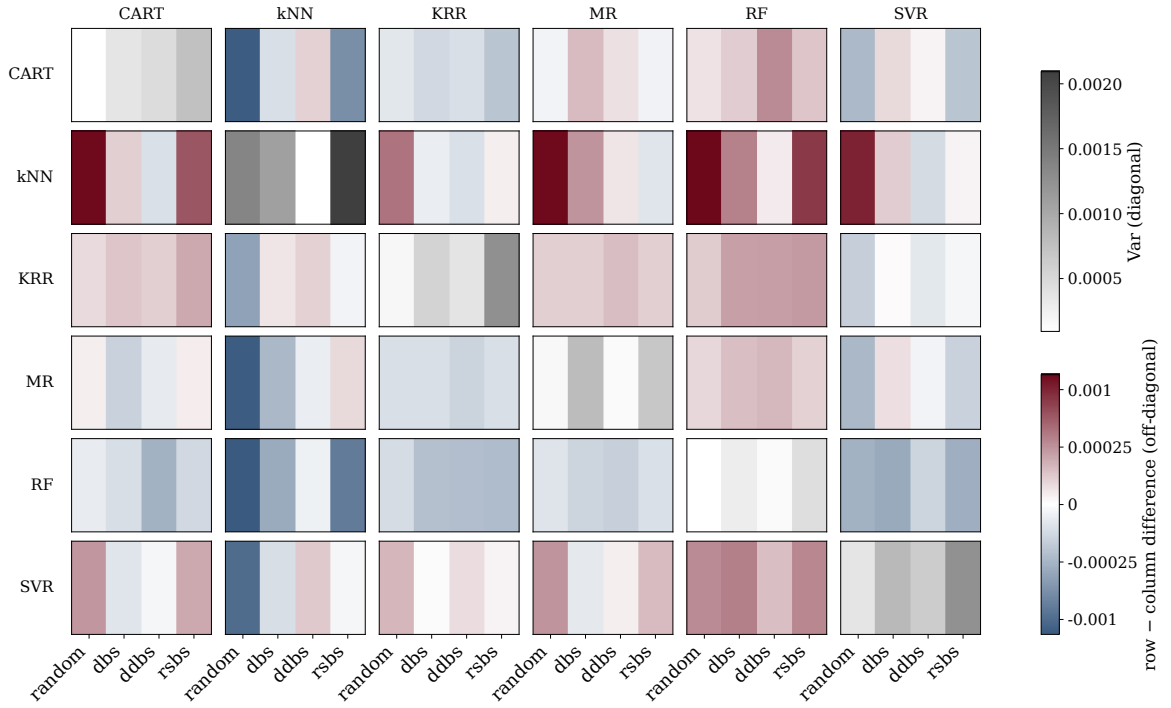


Figure 4.2: Pairwise combination comparison of the learning techniques by median stability (*Var*) across subject systems at $t = 3$, per stochastic sampling strategy. Off-diagonal blocks correspond to a pair of learning techniques (row, column), encoding the row-minus-column difference per shared sampling strategy on a blue-to-red scale (blue = row is more stable, red = column is more stable, white = equally stable). Diagonal blocks show absolute values in greyscale. Deterministic strategies are excluded, leaving only four cells per block. Yellow-bordered blocks mark significant differences.

Agreement across subject systems. Beyond identifying which combinations differ significantly, the question arises whether individual subject systems agree on how the combinations should be ordered. To compute this cross-system consensus, Kendall's coefficient of concordance W is used. Figure 4.3 shows Kendall's W across the three reference sample sizes for both prediction error and stability. For prediction error, the agreement grows from very weak at $t = 1$ ($W = 0.11$) to weak at $t = 2$ ($W = 0.32$) and moderate at $t = 3$ ($W = 0.58$). For stability, it remains very weak at $t = 1$ and $t = 2$ ($W = 0.11$ and $W = 0.12$) and only rises to $W = 0.21$ at $t = 3$. Overall, cross-system consensus on the best combinations for prediction error grows with reference size, whereas no comparable growth is visible for stability.

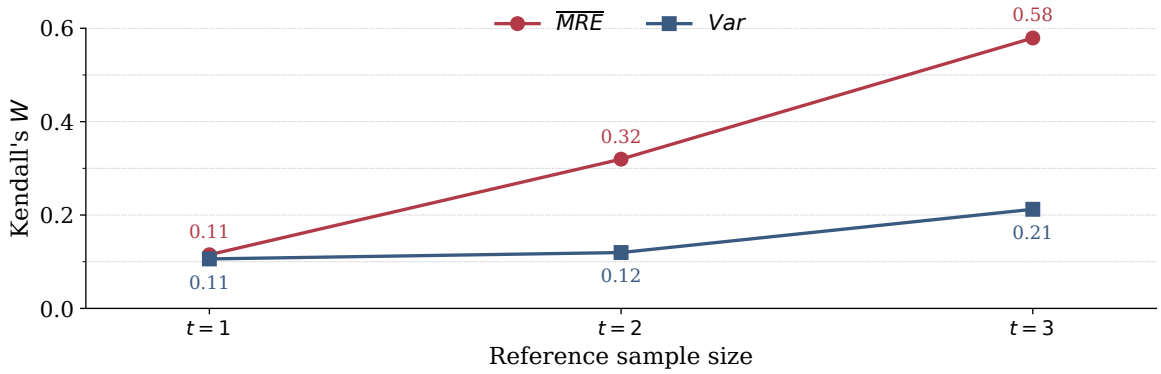


Figure 4.3: Kendall's coefficient of concordance W across subject systems for prediction error ($\overline{MRE}$) and stability (Var) at each reference sample size.

Ranking of combinations. While Kendall's W quantifies the cross-system consensus on rankings, it does not identify which combinations sit at the top or bottom of those rankings. Figure 4.4 orders the combinations at $t = 3$ by their median prediction error across subject systems. The best combinations pair `twise` or `random` with CART, RF or MR and reach median errors of about 0.08 with `twise` and RF performing best. At the other end, the medians reach up to about 0.53. kNN appears in four of the six worst combinations, while `dbs` appears among the worse combinations for most techniques. The distribution reveals a pattern: no combination with kNN reaches a median below 0.34, followed by combinations paired with `dbs`, which never reach a median below 0.25, regardless of the paired dimension. Furthermore, combinations considering kNN, `dbs`, KRR or SVR also show wider boxes, which means that their error varies more across the subject systems. At the smaller reference sizes, the leading combinations differ (see Section A.1.2). At $t = 1$, `sbs` dominates as a sampling strategy across techniques, although the combinations do not differ significantly at this size. At $t = 2$, `random` paired with CART or MR dominates the lower end of the prediction error. Only at $t = 3$ does RF emerge as the leading technique.

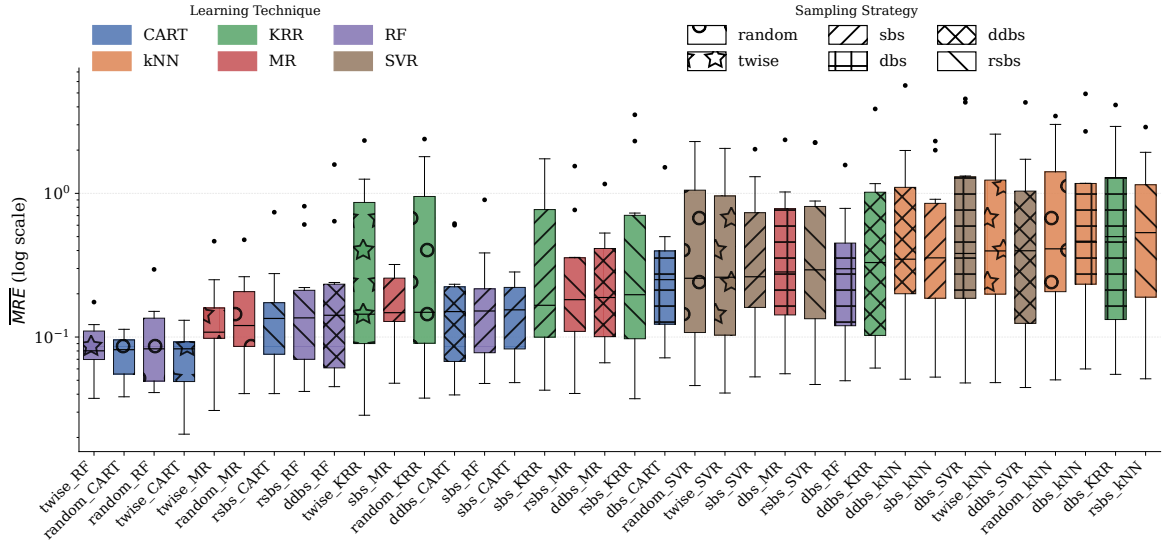


Figure 4.4: Distribution of prediction error ($\overline{MRE}$) across the ten subject systems for each combination of sampling strategy and learning technique at $t = 3$, in ascending order by median. Box colour encodes the learning technique and the hatch encodes the sampling strategy.

In Figure 4.5, the median variances range from about 10^{-4} to $2 \cdot 10^{-3}$ with the separate combination ranking being more mixed than for prediction error. `ddbs` with *k*NN is the most stable combination, while `rsbs` with *k*NN is the least stable one. `random` with CART or RF also clusters at the low-median end, in contrast to `rsbs` paired with *k*NN, KRR or SVR, which dominate the least stable end of the distribution. A pattern can be also seen here. The medians of RF vary least across the sampling strategies, while those of *k*NN vary most. Box widths are visibly wider for combinations involving *k*NN across all sampling strategies, indicating that stability itself varies substantially across systems for *k*NN-based combinations. At smaller reference sample sizes (see Section A.1.2), the best combinations differ substantially. At $t = 1$, `ddbs` paired with SVR ranks best.

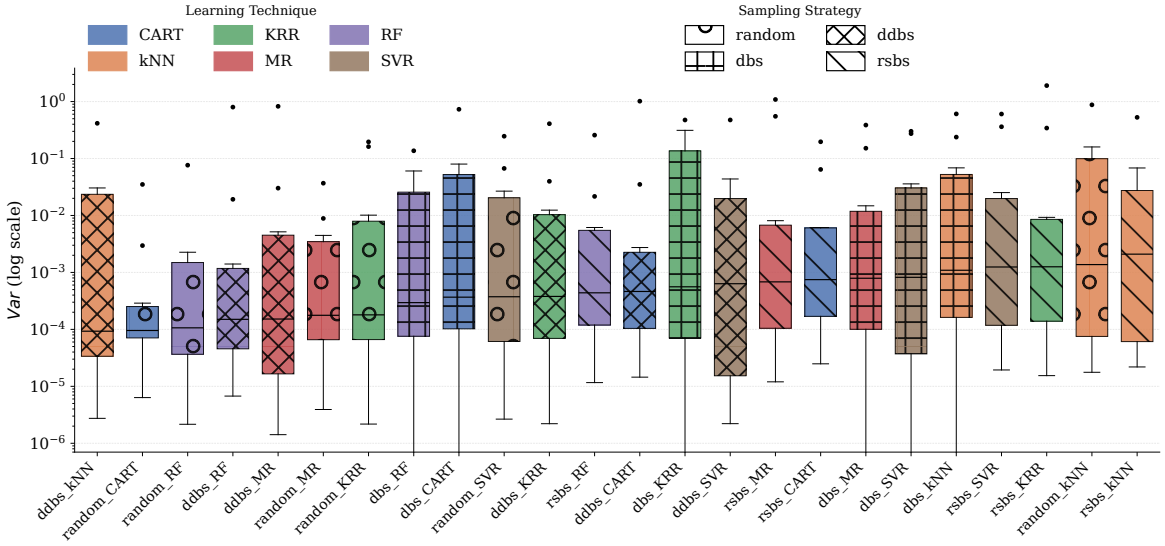


Figure 4.5: Distribution of stability (*Var*) across the ten subject systems for each combination of sampling strategy and learning technique at $t = 3$, in ascending order by median. Box colour encodes the learning technique and the hatch encodes the sampling strategy. Deterministic strategies are excluded.

Evolution across reference sizes. The analysis of the rankings captures the distributions across combinations at each reference sample size individually. To see how the techniques themselves evolve with reference sample size, we aggregate across sampling strategies and show the median per learning technique. Figure 4.6 and Figure 4.7 show how prediction error and stability evolve with rising reference sample size. All techniques improve in prediction error from $t = 1$ to $t = 3$. CART, MR and RF improve most, KRR improves mainly between $t = 2$ and $t = 3$, while *k*NN and SVR improve least and show the highest errors at $t = 3$. For stability, all techniques also become more stable, with *k*NN and SVR remaining the least stable at $t = 3$.

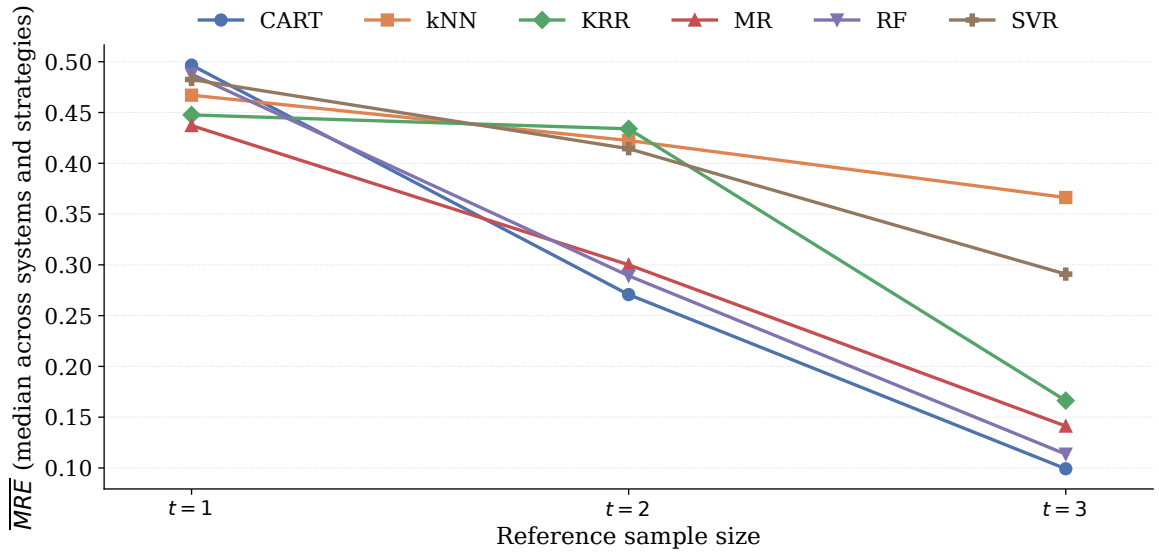


Figure 4.6: Median prediction error ($\overline{MRE}$) across the ten subject systems and sampling strategies per learning technique at $t \in \{1, 2, 3\}$. Techniques are distinguished by colour and marker style.

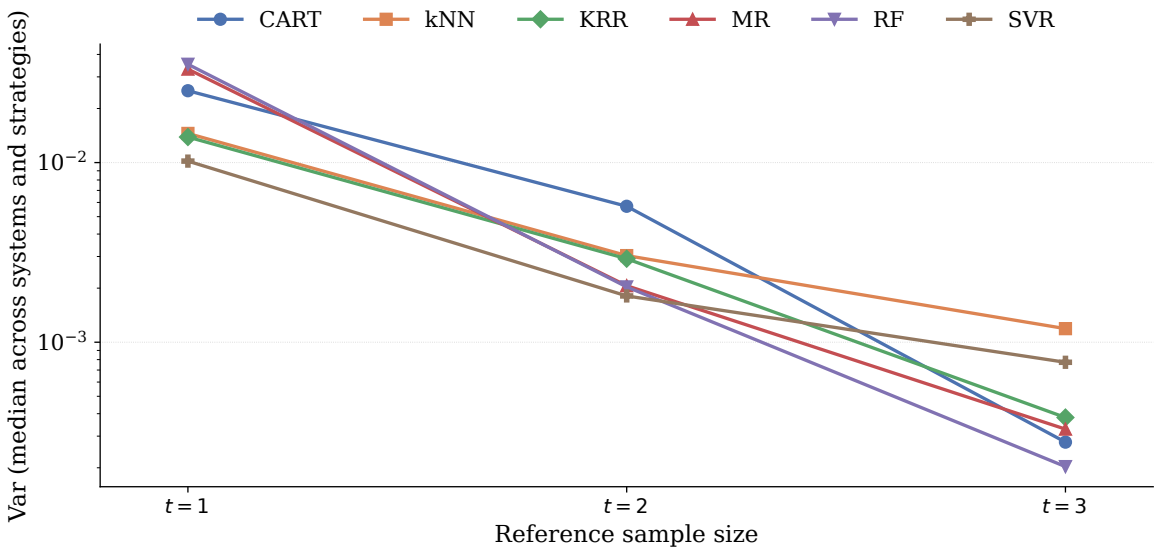


Figure 4.7: Median stability (*Var*) across the ten subject systems and sampling strategies per learning technique at $t \in \{1, 2, 3\}$. Techniques are distinguished by colour and marker style. Deterministic strategies are excluded from the aggregation.

Overall, at $t = 2$ and $t = 3$, the choice of the combination clearly affects prediction error with growing cross-system consensus on which combinations perform best. Combinations of `twise` or `random` with CART, RF or MR achieve the lowest errors, while *k*NN and `dbs` lead to the highest. Stability, in contrast, hardly depends on the combination and subject systems do not agree on a ranking, even at the largest reference sample size.

4.1.2 RQ2: Impact of the sample size

While *RQ1* focused on the interplay of sampling strategy and learning technique at three reference sample sizes derived from t-wise sampling, *RQ2* shifts the attention to sample size in detail.
For the second research question the sampling strategy is therefore fixed to `random` and vary the sample size across six proportions $\pi \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$ of the valid configuration space. First we report how the prediction error and stability evolve per subject system, followed by the inspection of the proportions, at which the metrics stop improving.

Evolution per subject system. The prediction error decreases with the sample size for almost all combinations of subject system and learning technique. Spearman's ρ confirms a significant monotonic decrease for 48 of the 60 combinations. Six of the remaining 12 are the capped combinations, which cannot become significant with only four proportions. The clearest exception is BerkeleyDBC with MR ($\rho = -0.66$), whose error stops decreasing at $\pi = 0.05$.
Figure 4.8 shows the evolution of prediction error across all six proportions. A technique-focused convergence pattern can be detected. For CART, RF and MR the per-system curves move closer together as the sample grows. With kNN, KRR and SVR, the gaps between systems largely persist. lrzip remains among the systems with the highest error for all techniques, closely followed by BerkeleyDBC for kNN and SVR specifically. With KRR, in contrast, the error of BerkeleyDBC drops from 3.5 to below 0.05. x264 and LLVM belong to the systems with the lowest errors for most techniques. With CART, the error of lrzip stays above 2 up to $\pi = 0.05$ and then drops to 0.29 at $\pi = 0.10$.
Alongside this convergence pattern, the curves reveal a saturation behaviour. For MR, 8 out of 10 per-system curves flatten between $\pi = 0.10$ and $\pi = 0.20$ with a decrease of less than 0.005 per step. CART and RF show partial saturation (six out of ten and five out of ten systems flat at the same span each) but a small subset continues to improve. For kNN, KRR and SVR, most per-system curves also show only a slight decrease past $\pi = 0.10$, with the exception of lrzip and BerkeleyDBC, which continue to decrease visibly.

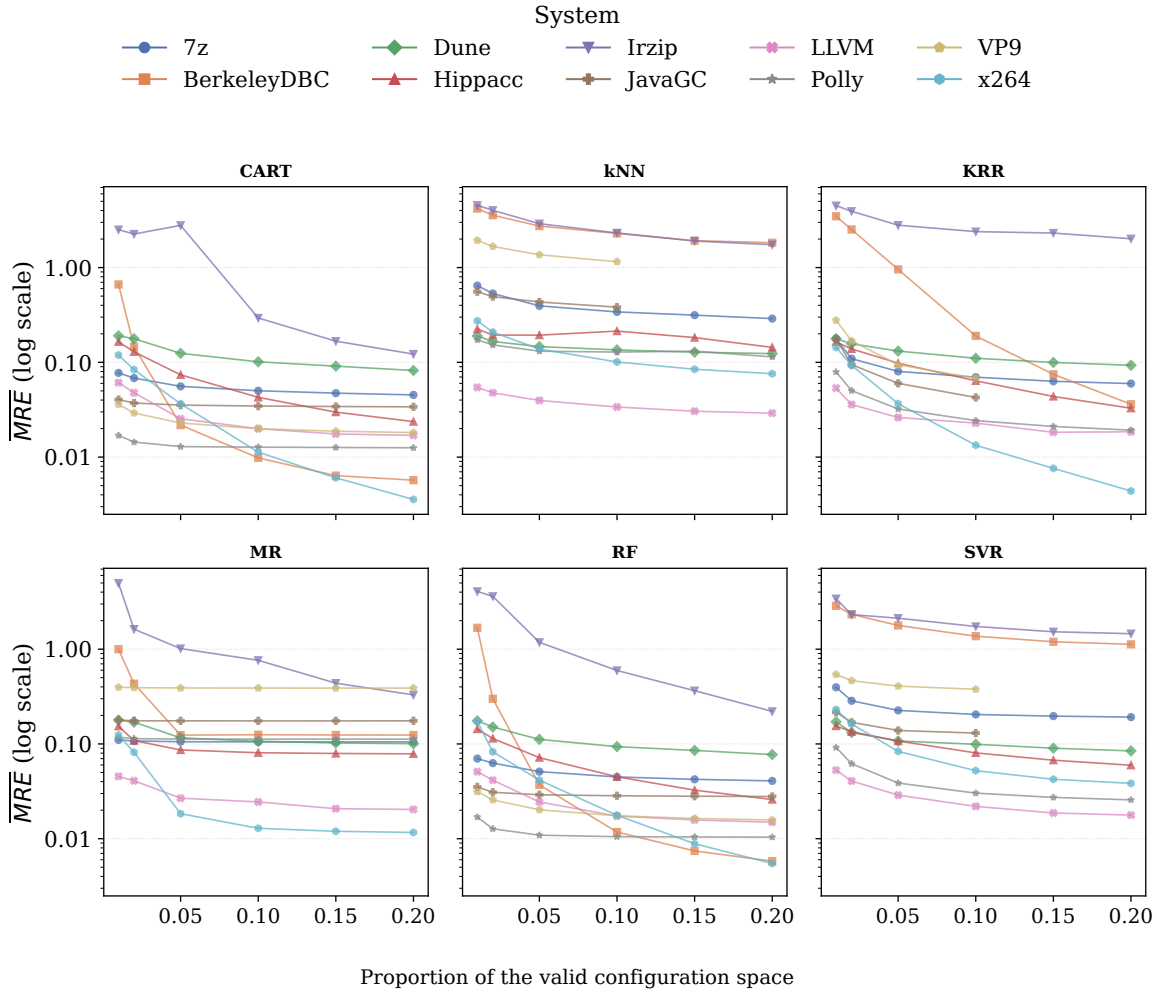


Figure 4.8: Evolution of prediction error ($\overline{MRE}$, log scale) across proportions of the valid configuration space for each subject system. Each block shows one learning technique and each colour and marker encodes one subject system.

Like the prediction error, the variance decreases with the sample size (Figure 4.9). Spearman's ρ also confirms a significant monotonic decrease for this dimension, with 54 out of 60 combinations. The remaining six combinations are exactly the capped combinations and cannot become significant. Unlike for prediction error, there is no exception among the combinations that are analysed over all six proportions.

As for prediction error, LRZIP has the highest variance for all techniques, followed by BERKELEYDBC having the second highest at the smaller proportions. For CART, RF and MR, both systems approach the others as the sample grows, such that the curves converge at larger proportions. Especially for CART and RF, BERKELEYDBC ends up below DUNE. Mirroring its prediction error, LRZIP stays above 1 up to $\pi = 0.05$ and then drops rapidly to about 0.04 at $\pi = 0.10$ for CART. With KRR, only BERKELEYDBC approaches the others. For kNN and SVR, both systems remain far above the rest. POLLY has the lowest variance for most techniques.

For most curves, the variance decreases fastest up to $\pi = 0.05$. Only a few curves rise at

some point, which is most clearly for LLVM with KRR, whose variance more than doubles from $\pi = 0.15$ to $\pi = 0.20$.

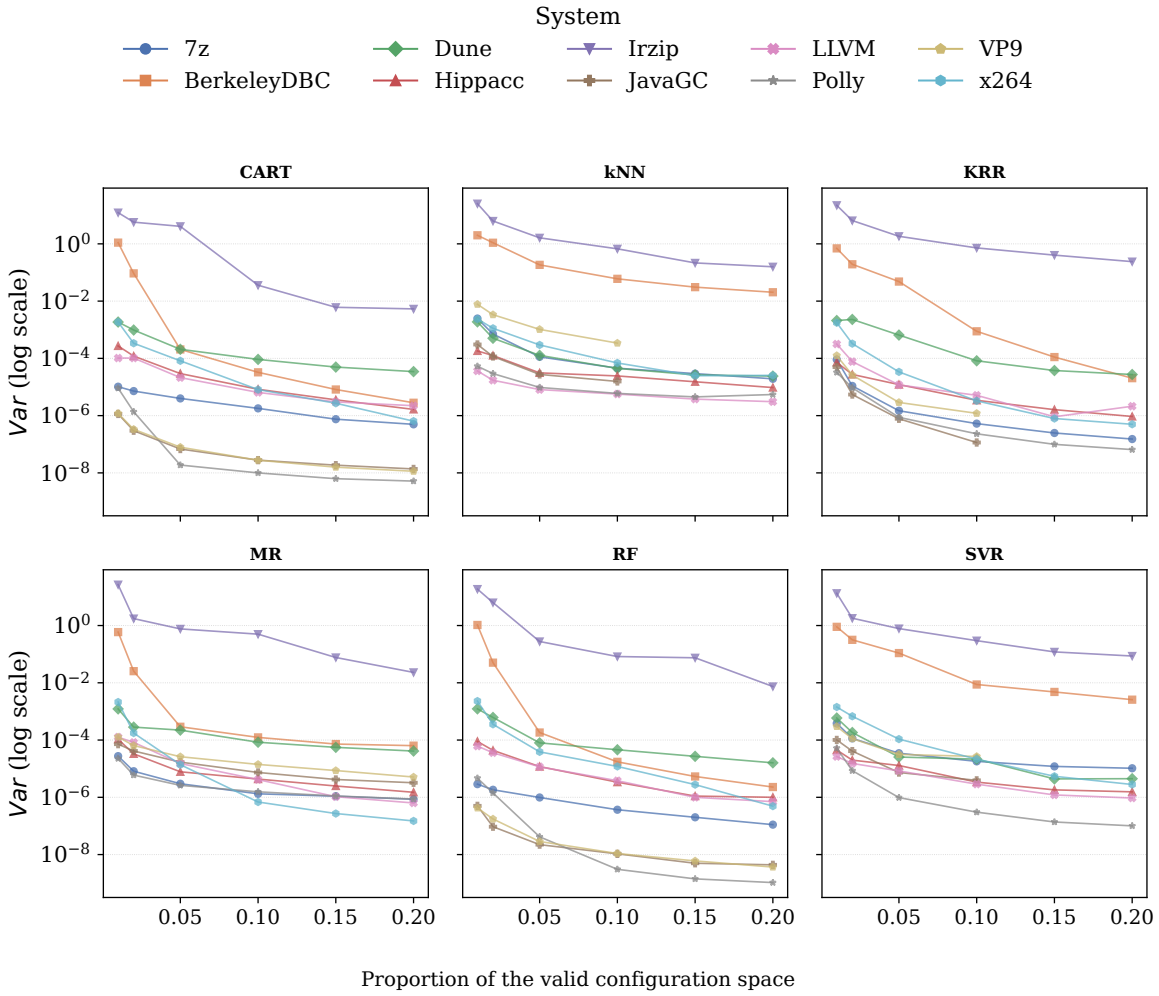


Figure 4.9: Evolution of stability (*Var*, log scale) across proportions of the valid configuration space for each subject system. Each block shows one learning technique and each colour and marker encodes one subject system.

Improvement per transition. While the curves show a global descent with a visible saturation pattern, they do not identify at which proportion transition further training data stops being statistically meaningful. Figure 4.10 shows at which transitions between proportions the error still decreases significantly, which is additionally illustrated by the shrinking value Δ. Three saturation patterns can be identified. MR saturates earliest. Only six out of ten systems improve at the $\pi = 0.05 \rightarrow 0.10$ transition with a median decrease of only $\Delta = -0.001$, dropping to five of ten at both later transitions. CART and RF still decrease significantly in at least nine out of ten systems at every transition, but their median decreases by approximately $\Delta = -0.001$ at the last transition, marking diminishing returns. *k*NN, KRR and SVR continue to benefit noticeably from more training data through the last transition. Specifically, *k*NN still reduces error by $\Delta = -0.021$ at $\pi = 0.15 \rightarrow 0.20$. This effect is roughly

twenty times larger than the decrease of CART and RF. KRR and SVR reduce error more moderately at this step ($\Delta \approx -0.005$).

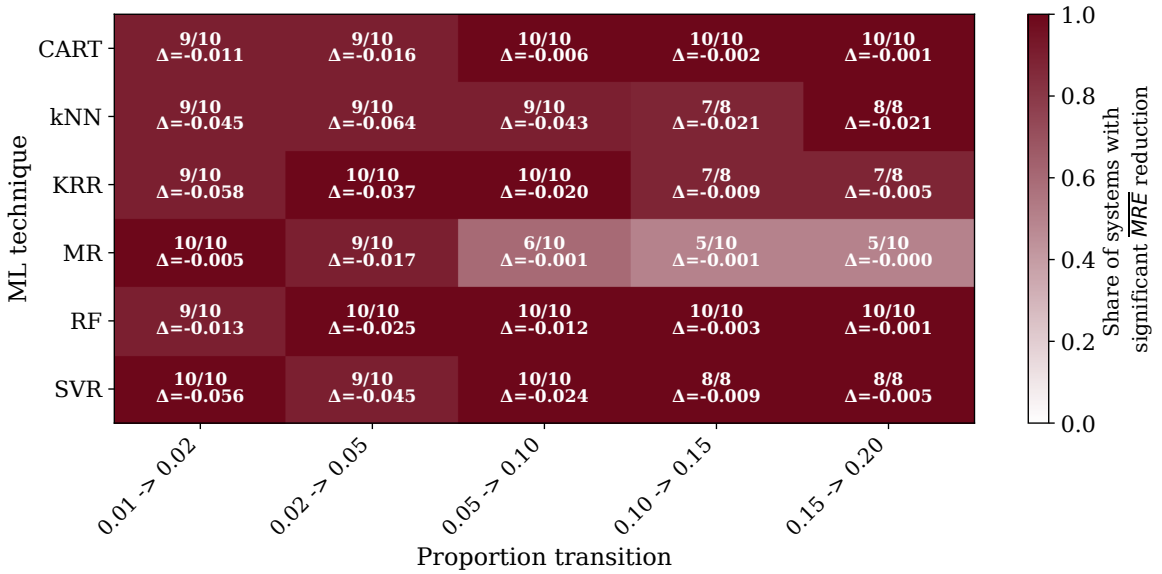


Figure 4.10: Number of subject systems whose prediction error ($\overline{MRE}$) decreases significantly at each proportion transition, per learning technique. Annotations in each cell show the number of significant systems out of the analysed ones and the median prediction error decrease Δ for that transition. Darker cells mark more systems improving, while lighter cells mark saturation.

Unlike prediction error, the stability improves in a similar way for all techniques. Figure 4.11 shows the same analysis for stability. Each cell states how many of the analysed subject systems show a significant decrease in variance at a transition, together with the median factor by which the variance changes. For example, $\times 0.35$ means that the variance at the larger proportion is 35 per cent of the variance at the smaller one. Overall, the variance decreases significantly in most transitions. Up to $\pi = 0.10$, most systems show a significant decrease at every transition. At each of these, the variance typically drops to between 15 and 45 per cent of its previous value. The largest decreases occur from $\pi = 0.02$ to $\pi = 0.05$, where KRR and RF reduce the variance to 15 per cent of its previous value. The only exception is CART at the first transition, where only five of the ten systems show a significant decrease.

From $\pi = 0.10$ on, the decrease in variance becomes smaller and occurs in fewer systems. At the last transition, the variance only drops to between 60 and 77 per cent of its previous value and only three to six systems per technique show a significant decrease. *k*NN and SVR improve least with three out of eight systems each. For CART, RF, *k*NN and SVR, this differs from the prediction error, which still decreases in almost all systems at this transition.

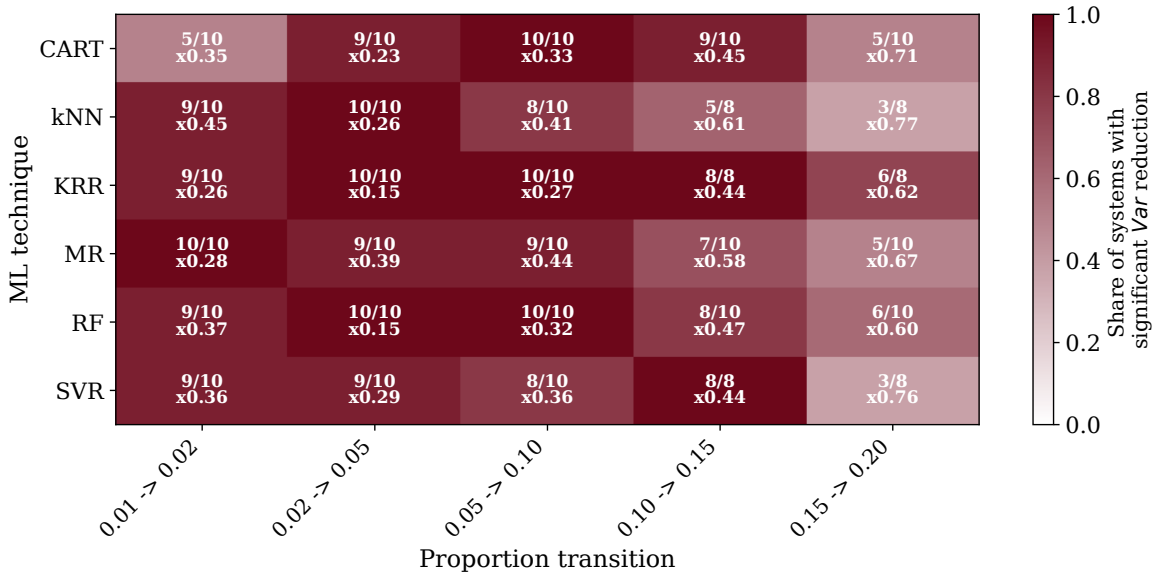


Figure 4.11: Number of subject systems whose variance (*Var*) decreases significantly at each proportion transition, per learning technique. Annotations in each cell show the number of significant systems out of the analysed ones and the median ratio across subject systems between the variance at the larger and at the smaller proportion. Darker cells mark more systems with a significant reduction in variance, while lighter cells mark saturation.

Overall, the results for RQ2 show that under random sampling, both prediction error and variance decrease with the sample size for almost all combinations. For prediction error, the point at which additional data leads to diminishing returns depends on the technique. MR saturates first, CART and RF follow with shrinking improvements, while *k*NN, KRR and SVR continue to improve throughout the tested range. For stability, the reductions fade at the largest proportions for all techniques equally.

4.2 Discussion

For this section, we discuss the results of our analysis in the context of our research questions. We interpret how the interplay between sampling strategies and machine-learning techniques affects prediction error and stability. Specifically, we examine why tree-based and regression techniques outperform kernel- and instance-based techniques. Next, the asymmetry between how sampling and technique choice affect prediction error and stability is inspected. Finally, we interpret to what extent the sample size affects the prediction error and stability, why the evolution with the training size is technique-specific and why some subject systems remain difficult to predict.

4.2.1 RQ1: Combination of machine-learning techniques and sampling strategies

Our results from *RQ1* reveal that at the largest reference sample size $t = 3$, tree-based and regression techniques systematically outperform *k*NN and SVR, while KRR lies in-between with regard to prediction error. The combinations involving *k*NN or `dbs` cluster at the worst end. As for stability, a much weaker effect is observed and this gap between prediction error and stability widens with the reference sample size. In the following, we discuss why these patterns emerge and situate our results in the context of the work by Kaltenecker et al. [28].

Interplay effect on prediction error (Learning techniques). As explained in Section 2.3.1, kernel- and instance-based techniques depend on the density of the training data around each prediction point. To ensure accuracy, the training sample has to cover the neighbourhood of the prediction point thoroughly. If that is not the case, the results degrade. In contrast, tree-based techniques only need enough training configurations on both sides of each split to estimate how a single option affects performance. They do not require dense neighbourhood coverage but instead tolerate sparse or even clustered training samples. MR arrives at similar robustness through a different route. It estimates coefficients globally from the entire training sample, such that local density around individual prediction points does not affect its prediction accuracy either. Although CART, RF and MR differ in their internal setup, they share the same insensitivity to local training density, which is what places all three techniques at the low-error end regardless of the sampling strategy they are paired with. This is reflected in Figure 4.1, where all six significant blocks set a tree-based or regression technique against a kernel- or instance-based one. CART, RF and MR all significantly outperform *k*NN and SVR across most sampling strategies. KRR sits in the middle, outperforming *k*NN and SVR but being beaten by CART, RF and MR, although none of these differences is significant. This placement of the technique is consistent with its setup, as KRR's regularisation penalises large weights, which may partially compensate for the sparse training coverage, explaining why it edges out SVR and *k*NN.

Interplay effect on prediction error (Sampling strategies). Beyond the pattern displayed by the learning techniques, a sampling-side pattern can be observed. As Figure 4.4 shows, no combination involving `dbs` reaches the low-error cluster. Because `dbs` draws configurations uniformly over distance values, which may over-represent the extremes of the configuration space, the resulting training configurations by `dbs` and evaluation configurations (drawn as $\mathcal{E} = \mathcal{V} \setminus \mathcal{S}$) end up occupying structurally different regions of $\mathcal{V}$. With regard to the subsequent learning step, *k*NN relies on training points being close to the prediction points, which becomes difficult when the reshaping of `dbs` leaves large parts of $\mathcal{E}$ far from the training configurations. That effect could be detected in Figure 4.4, where *k*NN dominates the worst-median end and also forms with `rsbs` the worst performing combination. KRR paired with `dbs` also sits at this end. The pattern is consistent with the internal setup of the techniques. As `dbs` may produces training sets that do not cover $\mathcal{V}$ evenly, *k*NN and KRR are affected most because of their dependence on local coverage.

Interplay effect on stability. Regarding the interplay effect on stability, a different pattern

than for prediction error is identified. Although our statistical tests detect rank differences for stability at the largest reference sample size, no pairwise comparison is deemed significant. In other words, the combinations do not differ in stability consistently enough across the subject systems for any single pair to be singled out. This suggests that this metric is shaped more by the properties of each subject system's configuration space than by the interplay between sampling strategy and learning technique. This is supported by two further observations. First, Kendall's W, where the cross-system consensus on the ranking for stability grows only weakly. Second, in $RQ2$, LRZIP and POLLY consistently occupy the extremes of variance across almost all techniques. Nevertheless, the stability of a technique still depends on the the choice of the sampling strategy, even though these differences are not consistent enough across the subject systems to be significant. One combination illustrates this clearly. As shown in Figure 4.5, kNN, which sits at the worst-median end for prediction error, becomes the best-performing technique when being paired with `ddbs` for stability. This might be due to the diversification of `ddbs`, which may constrain sample selection more strictly than `dbs`, producing more similar training sets across runs (see Section 2.2). At the same time, when the sampling strategy changes from `ddbs` to `random` or `rsbs`, the same kNN moves to the worst end of the stability ranking. This suggests that the run-to-run consistency of the paired sampling strategy affects the stability of kNN more than its accuracy, which remains poor regardless of the paired strategy. Thus, the stability of kNN also depends on how consistently the paired strategy draws its samples. Across the subject systems, however, no combination improves stability significantly.

Comparison with the original study. So far, we have interpreted the interplay pattern within our own results. Now, we situate these findings against the original study that is replicated. Kaltenecker et al. [28] reported that RF outperforms the other learning techniques in most cases, with CART and MR also performing well, but that no combination is clearly superior in all cases. Our results reflect both observations. At $t = 3$, `twise` and `random` with RF perform best, while the consensus across subject systems is moderate rather than strong. Our replication additionally extends the sampling-strategy side by including the distance-based family (`dbs`, `ddbs`) and randomised solver-based sampling (`rsbs`) from [29], which were not part of the original interplay study. This extension leads to two findings. First, `dbs` is grouped among the worst-performing strategies. This result could not have been observed in the original study, since `dbs` was not part of it. Second, Kaltenecker et al. [29] report that `ddbs` outperforms `twise`, `sbs` and `rsbs` for $t = 2$ and $t = 3$ and almost reaches the accuracy of random sampling. Our replication confirms that `random` performs better than `ddbs` when paired with MR (see Figure A.19). Unlike in the distance-based sampling study, however, `ddbs` does not outperform `twise` with MR. This difference may stem from our implementation of MR as a Lasso regression instead of the stepwise regression used in [29]. Our replication therefore connects the two prior studies as the dominance of combinations pairing `random` or `twise` with tree-based or regression techniques holds even when tested against `ddbs` introduced in [29].

RQ1 (Combination of machine-learning techniques and sampling strategies): At moderate to large reference sample sizes, the combination of sampling strategy and machine-learning technique affects prediction error strongly and stability weakly. CART, RF and MR achieve the lowest errors, while *k*NN and `dbs` lead to the highest errors. As in the original study, `random` or `twise` sampling with a tree-based or regression technique performs best, also compared to the sampling strategies added to our replication.

4.2.2 RQ2: Impact of the sample size

While *RQ1* examined three reference sample sizes derived from *t*-wise sampling, *RQ2* fixes random sampling and considers six proportions of the valid configuration space $\mathcal{V}$. Our results on the effect of the proportions $\pi \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$ suggest a technique-specific saturation pattern for prediction error, which mirrors the tree-based/regression and kernel-/instance-based split already observed in Section 4.2.1 and a uniform saturation pattern for stability. A system-level difficulty pattern that persists across all techniques can be also observed.

Saturation pattern on prediction error. As discussed in Section 4.2.1, the results reveal a split between tree-based/regression techniques and kernel-/instance-based ones. Under *RQ2*, this split can also be observed, when considering saturation. As Figure 4.8 and Figure 4.10 show, MR is the earliest technique to saturate, which is consistent with its characteristic of fitting global coefficients (see Section 2.3.1). Once enough training data covers the configuration space to identify each coefficient, further data adds no new information. CART and RF hit diminishing returns later, as their split structure exhausts more slowly. Specifically, their underlying tree representations continue to find new informative splits at every proportion in the tested range, but the reduction per transition shrinks to near zero by $\pi = 0.15 \rightarrow 0.20$, making the payoff smaller with every new split. In contrast to the early saturation timing, *k*NN, KRR and SVR keep improving as new data is added. Since their prediction accuracy depends on local coverage of the prediction points, there is not sufficient data until the training sample covers those neighbourhoods densely. At $\pi = 0.20$, that coverage is apparently not yet reached for most subject systems.

Convergence of error across subject systems. Beyond the technique-specific saturation pattern, the same split from Section 4.2.1 also shapes how prediction error behaves across subject systems at larger proportions (Figure 4.8). Tree-based and regression techniques compress the gap between best- and worst-predicted systems at $\pi = 0.20$, converging into a relatively narrow band. In contrast, kernel- and instance-based techniques do not show such convergence but instead maintain visible gaps. This especially applies to LRZIP, which stays separated from the main band for most techniques, and BERKELEYDBC, which stays separated under *k*NN and SVR. This is consistent with the pattern discussed in Section 4.2.1, where tree-based and regression techniques do not require dense local coverage to generalise, so that once enough data is available, they converge across systems. The remaining techniques do require it, however, which is visible at $\pi = 0.20$, at which point the training samples do not yet reach that density. The same split appears for stability. With CART,

MR and RF, the variance curves of LRZIP and BerkeleyDBC approach the other systems, whereas with *k*NN and SVR both remain far above the rest (see Figure 4.9).

System-level prediction difficulty. Within the convergence pattern, a subset of subject systems remains consistently at the top of the per-technique error. LRZIP remains among the systems with the highest error for all techniques, even at the largest analysed proportion $\pi = 0.20$, followed by BerkeleyDBC for *k*NN and SVR (Figure 4.8). At the other end, x264 and LLVM converge to the lowest errors across most techniques. Since these orderings persist across techniques, prediction difficulty appears to be a property of the subject systems themselves rather than of the learning technique. The same holds for stability, as LRZIP also shows the highest variance for all techniques. The subject systems that are hard to predict are thus also those whose predictions vary most across runs. Characterising what specifically makes these systems hard or easy to predict is beyond the scope of this thesis and remains a topic for future work.

Size effect on stability. Unlike prediction error, the variance saturates in a similar way for all techniques. Up to $\pi = 0.10$, it decreases significantly in most systems, whereas at the last transition only three to six systems per technique show a significant decrease (see Figure 4.11). A possible explanation is that with larger random samples the configuration space may be more completely represented, such that samples of different runs and therefore the resulting prediction models, differ less, regardless of the learning technique. This technique-independent pattern is consistent with *RQ1*, where stability hardly depended on the combination of sampling strategy and learning technique. For most techniques, however, prediction error and stability saturate at different points. At the last transition, the error of CART, RF, *k*NN and SVR still decreases significantly in all or almost all systems, while their variance does so in only three to six systems. For CART and RF, the remaining decrease is very small, whereas *k*NN reduces its error noticeably and SVR moderately. Only for MR and KRR do error and variance saturate at a similar point, with five of the ten systems for both metrics for MR and seven and six of eight systems for KRR. Additional data thus still makes most techniques more accurate, but hardly more stable.

Comparison to *RQ1*. Comparing these findings with those from Section 4.2.1 reveals that the same technique split from *RQ1* appears in both variations of the considered sample size. At a fixed reference sample size, it shows which combinations reach the low-error cluster, while under a more fine-grained set of proportions with random sampling, it shows which techniques saturate early and which continue to benefit from more data. The discussed insensitivity to local coverage for CART, RF and MR and the dependence on it for *k*NN, KRR and SVR is therefore not a sole conclusion from the experimental design of *RQ1*, but instead a robust property of the technique categories. This further strengthens the findings in Section 4.2.1, in that the best-performing combinations, pairing `random` or `twise` with CART, RF or MR, yield both low error at fixed sample sizes and early saturation as data grows.

RQ2 (Impact of the sample size): Under random sampling, prediction error and variance decrease with the sample size for almost all combinations. Tree-based and regression techniques reach diminishing returns in prediction error earlier than kernel- and instance-based ones, while improvements in stability fade at the largest proportions for all techniques alike. A few subject systems, such as LRZIP, remain difficult to predict across the entire range.

4.3 Threats to Validity

In this section, we discuss potential threats to the validity of our evaluation and analysis. Section 4.3.1 covers the accuracy and reliability of our experimental design and measurement processes, while Section 4.3.2 covers the generalisability of our results with regard to other configurable systems, sampling strategies and machine-learning techniques.

4.3.1 Threats to Internal Validity

Stochastic components. To properly evaluate prediction accuracy, we must ensure that factors influencing the results, such as stochastic components of sampling strategies, do not bias the outcomes. The sampling strategies `random`, `dbs`, `ddbs` and `rsbs` are stochastic and introduce variability across runs. This threat is primarily mitigated through the high number of repetitions. Each experiment is repeated over $R = 100$ runs, so that the prediction error $\overline{MRE}$ and the stability Var are aggregated over many independently sampled training sets rather than a single one (see Section 3.2). Learning techniques with random components, such as RF, use a fixed seed, so that all variation between runs stems from the sampling. As a consequence, our stability results do not capture the variability caused by the random choices of the learning techniques themselves. Within each run, the same seed is used across all sampling strategies to ensure comparability. Specifically for the stability analysis, the deterministic strategies (`twise` and `sbs`) are excluded by construction ($Var = 0$ trivially), so the findings apply only to the 24 combinations involving stochastic sampling strategies.

Hyperparameter tuning. Each applied machine-learning technique has specific hyperparameters that influence its learning behaviour and resulting predictions. The choice of such hyperparameters can significantly affect prediction performance and bias comparisons between techniques. To reduce this threat, the hyperparameters are only tuned once per subject system, learning technique and sample size combination on a random-sampled training set of that size. The resulting hyperparameters are then fixed and reused across all sampling strategies for that combination. Three threats emerge from this design. First, hyperparameters tuned on a random-sampled training set may be suboptimal for models later trained on samples produced by strategies with structurally different distributions, potentially biasing the ranking from *RQ1* against these strategies. Additionally considering the sampling strategy in our tuning process would remove this bias, but would multiply the number of tuning runs by the number of sampling strategies. Second, for SVR, only 30 of the 288 hyperparameter combinations are evaluated. Although these include one of

the best ten per cent of all combinations with a probability of more than 95 per cent (see Section 3.2), the selected hyperparameters may not correspond to the true grid optimum. Third, our value sets deviate from those of the original study, mainly because tuning with the full value sets exceeded our time limit. In particular, our value set for the regularisation strength of KRR does not contain the value selected in the original study, so that KRR may be more strongly regularised in our setting. Further differences are listed in Table A.1.

Implementation of MR. Regarding the applied learning techniques, considering the conceptual replication, MR's influence model is implemented as a linear combination of options and their pairwise interactions followed by Lasso regression, instead of the stepwise regression used in the original study. This may limit the comparability of MR with the original study, as discussed in Section 4.2.1. Due to computational infeasibility at higher degrees, MR is limited to pairwise interactions only. This means that MR cannot capture three-way or higher-order interactions, which may cause it to underperform on subject systems where such interactions dominate.

Measurement noise. The applied performance measurements may contain noise or inaccuracies, which can also influence the learned models and consequently prediction results. This threat is mitigated by using established datasets that were also used in prior work [29]. This ensures a consistent and reliable measurement basis across all experiments.

Fixed sampling strategy in RQ2. For *RQ2*, the sampling strategy is fixed to uniform random sampling to isolate the effect of sample size on prediction error and stability. Varying both sampling strategy and sample size simultaneously would introduce confounding effects, making it difficult to attribute observed changes solely to sample size. This design choice ensures a controlled analysis of the relationship between training data size and prediction performance, but our findings for *RQ2* may not transfer to other sampling strategies.

Computational limit. For the second research question, a 22,000 configuration limit on *k*NN, KRR and SVR for JavaGC and VP9 is imposed at proportions $\pi = 0.15$ and $\pi = 0.20$, resulting in 12 excluded system, technique and proportion combinations. The same limit is applied during hyperparameter tuning. Our conclusions about kernel- and instance-based techniques at large proportions on the two systems are therefore inferred from smaller proportions and the remaining subject systems. Since the saturation patterns for these techniques are consistent across the remaining systems at all tested proportions, we assume they extend to the missing combinations, though this cannot be verified.

Statistical analysis. Our conclusions depend on the choice and application of statistical tests. Non-parametric methods (Friedman with Iman-Davenport statistic, Wilcoxon signed rank, Spearman and Brown-Forsythe) with Benjamini-Hochberg correction are used, chosen for their robustness to the non-normal distribution of performance metrics. Alternative statistical applications could produce different rejection patterns. In particular, the 100 per-run values for $\overline{MRE}$ are aggregated into a single per-combination mean before applying the Friedman test, such that the effective sample size for the tests equals the number of systems, i.e. ten. This way, per-run noise is reduced.

Implementation correctness. Finally, our results depend on the correct implementation of the sampling strategies and the feature model processing. Errors in generating valid configurations, for example due to incorrect encoding of the feature model constraints, as well as mistakes in the sampling procedure, could distort the prediction results. To mitigate this, the sampling implementation is validated through unit tests covering configuration enumeration, interaction enumeration, coverage checking, the t-wise and random sampling algorithms. Additionally, the sampler performs runtime checks, such that every generated configuration is evaluated against the BDD.

4.3.2 Threats to External Validity

Subject systems. Our experimental setup considers a set of ten real-world configurable software systems inherited from Kaltenecker et al. [29]. Even though these systems stem from different domains and vary in the number of configuration options and valid configurations, they may not fully represent the diversity of all real-world systems. In particular, all of them contain only binary configuration options, whereas the subject systems of the original study also contain numeric ones, so our findings may not transfer to systems with numeric options. In addition, the measured performance property is an execution time for all systems, so our findings may not transfer to other non-functional properties, such as memory consumption or energy. We also reuse their measured performance values, so our findings apply to performance behaviour as observed under their workload and hardware.

Learning techniques. Furthermore, only a subset of machine-learning techniques is evaluated, which may limit the applicability of the results to other learning techniques. The six techniques are inherited from Kaltenecker et al. [28], which span the main categories of supervised regression, concretely tree-based (CART), ensemble-based (RF), linear (MR), instance-based (kNN) and kernel-based (KRR, SVR).

Sampling strategies. The selection of sampling strategies used to generate the training data for our models must also be considered in terms of generalisability. Six sampling strategies are also inherited from Kaltenecker et al. [29], spanning random, coverage-oriented (`twise`), solver-based (`sbs`, `rsbs`) and distance-based (`dbs`, `ddbs`) strategies. Our design does not include the strategies from statistical design of experiments, such as Plackett-Burman designs, which the original study used for numeric configuration options [28]. Since our subject systems contain only binary options, these strategies are not applicable in our setting. Because our replication combines the learning techniques from [28] with the sampling strategies and subject systems from [29], any discrepancy between our findings and either original study could reflect the change of the sampling strategies, the change of subject systems or their interaction.

Sample sizes. Finally, the definition of our sample sizes has to be considered. In our experimental setup, sample sizes are defined differently depending on the research question. For *RQ1*, they are derived from t-wise sampling at $t \in \{1, 2, 3\}$, so our findings are dependent on the interaction-coverage levels these sizes provide. For *RQ2*, sample sizes are

defined as proportions of the valid configuration space, so our findings for this research question are expressed in relative terms rather than absolute training-set sizes. While this allows for interaction-aware sampling and controlled analysis of scaling effects, alternative definitions may lead to different results and limit the generalisability of the findings.

4.4 Future Work

Some of the limitations discussed in Section 4.3 and some findings in Section 4.2 open up directions for future work. This section discusses three of them.

Sampler-specific hyperparameter tuning. In our study, we tune hyperparameters for each subject system, learning technique and sample size combination on a random sample of the respective size and reuse the resulting hyperparameters across all sampling strategies. Because different sampling strategies cover the configuration space differently, the ranking results from *RQ1* may be biased against the strategies whose distributions differ most from random sampling. An extension would be to consider sampling strategies as part of the tuning combinations. This would remove the sampling-side bias, but require substantially more computation. In addition, it would reveal whether the observed ranking is a property of the strategies themselves or part of the tuning-set choice.

Extending the technique and sampling categories. Our replication combines the sampling strategies from [29] with the machine-learning techniques from [28], which fixes the space over which our findings generalise. An extension would apply our design to subject systems with numeric configuration options, where the design-of-experiments sampling strategies from [28] become applicable, as well as neural-network techniques, which are also not covered by our design. This would reveal whether the tree-based/regression technique dominance and the sampling-side pattern generalise beyond current categories.

System-level difficulty. In Section 4.2.2 we identified LRZIP and BerkeleyDBC to be consistently hard to predict across most machine-learning techniques, indicating that prediction difficulty is a property of the subject systems rather than of the technique choice. However, this cannot be characterised further from the ten subject systems in our study. Extending the study to a broader and more diverse set of subject systems, such as embedded or distributed applications outside our domains, would widen the space over which system-level prediction difficulty can be observed. Combined with measurements of feature-model properties, for instance interaction-graph depth, this would allow tying e the prediction difficulty to concrete system characteristics.

5

Related Work

Research on performance prediction for highly configurable software systems has established machine-learning-based approaches as a method for modelling performance across large configuration spaces. Prior work has explored a wide range of learning techniques, including linear models [50], tree-based methods [48] and deep neural networks [14], as well as factors influencing the prediction quality. Since the combinatorial explosion of the configuration space makes an exhaustive evaluation of all valid configurations infeasible, a variety of sampling strategies have beem proposed [3]. These include random [2], interaction-aware [34] and constraint-aware approaches [25]. Previous studies also show that the prediction accuracy depends not only on a single technique in isolation, but on the interaction between the chosen sampling strategy and learning method, with no universally optimal combination [12].

5.1 Performance Prediction for Configurable Systems

Predicting the performance of configurable software systems has been widely studied as a means to approximate performance behaviour without exhaustively evaluating all configurations [12, 28, 50]. Machine-learning-based approaches learn the relationship between configuration options and observed performance values from a limited set of measurements.

Foundational machine-learning techniques. Early work by Siegmund et al. [50] introduced *performance-influence models*, which capture how individual configuration options and their interactions affect performance. These models can incorporate domain knowledge to reduce complexity and improve prediction quality. Guo et al. [13] proposed a data-efficient approach that combines multiple techniques from machine-learning and statistics to build, validate and select a prediction model from a small sample of measured configurations.

Factors influencing prediction quality. Mühlbauer et al. [39] studied the impact of workload characteristics (i.e. characteristics of the input fed to the configurable system) on performance prediction and showed that different workloads can significantly affect prediction accuracy. Building on this observation, further research has investigated trade-offs in performance prediction. Kolesnikov et al. [33] analysed the relationship between prediction accuracy, model complexity and computation time.

Effect of training-data size. The size of the training sample plays an important role in prediction quality, such that with larger sample sets a general improvement of performance up to a point of diminishing returns can be observed [46, 48]. Pereira et al. [2] replicated the study by Kaltenecker et al. [29] and examined how prediction error varies with sample size across several sampling strategies. Building on these observations, our work examines how prediction error and stability evolve across six proportions of the valid configuration space.

Interplay of sampling and learning. The interaction between data selection and learning technique has been analysed by Kaltenecker et al. [28]. They combined six machine-learning techniques with 13 binary and numeric sampling strategies, including strategies from design of experiments, across six subject systems and analysed the interplay of both prediction error and stability. The underlying empirical study by Grebhahn et al. [12] concludes that no universally optimal combination of sampling strategy and learning technique exists. Our work replicates this study under the configuration-space-tailored sampling strategies and subject systems of [29], using the hyperparameter values selected in the original study [12, 35] as the reference of our tuning procedure.

5.2 Sampling Strategies

Highly configurable software systems are often characterised by a large number of configuration options and their possible combinations [56], leading to a combinatorial explosion of the configuration space [3]. In the context of performance prediction, exhaustive evaluation of all configurations is therefore infeasible. As a result, learning-based approaches rely on sampling strategies to select a representative subset of configurations for training predictive models [28, 29]. The prediction quality is strongly influenced by which sampling strategy is chosen [12]. Systematic literature reviews and comparative studies identify sampling as a core aspect of performance modelling and analyse different strategies with the goal of obtaining representative sample sets [3, 12]. Different classes of sampling strategies have been proposed in this context.

Random sampling. Random sampling serves as a commonly used baseline and often achieves strong prediction accuracy in practice, sometimes outperforming more complex strategies such as t-wise sampling, as shown by Pereira et al. [2]. However, generating valid configurations to sample from can be challenging in the presence of complex constraints. This difficulty motivated the development of alternative approaches, including distance-based sampling by Kaltenecker et al. [29] discussed below.

Interaction-aware sampling. Interaction-aware sampling strategies, such as t-wise sampling, focus on covering feature interactions and presence conditions [34, 36]. They target combinations of options up to a fixed interaction degree t rather than uniformly over the entire configuration space. This can produce large samples, which can be inefficient for analysis at higher interaction degrees, as shown by Medeiros et al. [36]. Krieter et al. [34] shifted the notion of coverage from feature-model options to code-level presence conditions to make interaction-aware sampling more efficient.

Constraint-aware sampling. Constraint-aware approaches, based on SAT solving or Binary Decision Diagrams, explicitly account for constraints and enable the generation of valid configurations. Heradio et al. [25] proposed a statistical test to check whether SAT-based samplers produce uniform samples and found that, among four state-of-the-art samplers, only one is both scalable and uniform. Munoz et al. [40] extended constraint-aware sampling to feature models with numerical features via bit-blasting.

Diversity-driven and distance-based sampling. Diversity-driven strategies such as distance-based sampling aim to select configurations that are well distributed across the configuration space. Kaltenecker et al. [29] introduced two families of sampling strategies tailored to configuration spaces, namely distance-based and diversified distance-based sampling. Evaluating these strategies on ten real-world configurable systems, they demonstrated that diversified-distance-based sampling outperforms t-wise, solver-based and randomised solver-based sampling for medium to large sample sizes and almost reaches the accuracy of random sampling, without requiring an analysis of the whole configuration space. Our thesis adopts the six sampling strategies evaluated in that study, `random`, `twise`, `sbs`, `rsbs`, `dbs` and `ddbs`, together with the same subject systems.

Recent developments. Recent work further investigates properties of sampling strategies, such as the uniformity of random samplers [57]. On the sampling-infrastructure side, Dubslaff, Husung and Käfer [9] investigated BDD compilation techniques for feature models. These techniques provide the tooling foundation used in our implementation for representing the valid configuration space and drawing uniform random samples.

6

Conclusion

Prior work has shown that the choice of sampling strategy and the choice of machine-learning technique interact in non-trivial ways when predicting the performance of configurable software systems [28]. Whether this interplay holds under different experimental conditions has been an open question. This thesis addresses it through a conceptual replication.

We combined six sampling strategies, six machine-learning techniques and ten subject systems to investigate how the choice of combination affects prediction accuracy and stability. We used t-wise sampling at $t \in \{1, 2, 3\}$ to determine reference sample sizes for the analysis and evaluated the effect of sample size separately by drawing samples at proportions $\pi \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$ of each subject system's valid configuration space. For each combination, we compared the learning technique's prediction against measured performance values to compute prediction error and stability. To assess the results statistically, we applied the Iman-Davenport variant of the Friedman test, pairwise Wilcoxon tests with Benjamini-Hochberg correction and Kendall's coefficient of concordance W for the first research question. Spearman's rank correlation coefficient, Wilcoxon signed-rank tests and Brown-Forsythe tests were used for the results of the second research question.

The results of our first experiment revealed that the interplay affects prediction error strongly, but stability rather weakly. Specifically, tree-based and regression techniques achieved the lowest errors and significantly outperformed kNN and SVR, reflecting their tolerance for different training-data structures. Combinations involving kNN or `dbs` clustered at the worst end, as kNN depends on local training coverage, which `dbs` provides only unevenly. For stability, no pair of combinations differed significantly and subject systems hardly agreed on a ranking. These findings align with the original interplay study, which also placed tree-based and regression techniques at the top and extend it to the additional distance-based sampling strategies, that were not part of ist design.

For the second research question, prediction error decreases with the sample size for almost all combinations. However, the point at which further data provides little error reduction is technique-specific. MR saturates the earliest, consistent with its global estimation of coefficients, followed by tree-based techniques (CART, RF), which reach diminishing returns more gradually, while kernel- and instance-based techniques (kNN, KRR, SVR) keep benefiting even at our largest tested proportion. Stability also improves with the sample size, but unlike prediction error, its improvement fades at the largest proportions for all techniques alike. Hence, additional data can still make a technique more accurate without making it more stable. In addition, LRZIP remained hard to predict across all techniques, indicating that prediction difficulty is a property of the systems themselves, rather than of the learning technique choice.

One pattern runs through all our experiments. The internal structure of the learning technique, whether or not it depends on the local training coverage, shapes the interplay most visibly, determining both which combination performs best at a fixed sample size and how much data they need to saturate. The best-performing combinations consistently paired `random` or `twise` sampling with a tree-based or regression technique, while subject systems agreed only moderately on the exact ranking. The interplay reported in the original study therefore persists in our setting, with the learning technique shaping it more visibly than the sampling strategy.

A

Appendix

This chapter provides additional material for the evaluation and the experimental setup. We include further visualisations for *RQ1*, namely the pairwise combination comparison matrices, aggregated and per-system boxplots and mean-rank matrices of the combinations. Finally, we list the hyperparameter values used in this thesis.

A.1 Additional Results for RQ1

This section provides further visualisations at smaller reference sample sizes for the results of *RQ1*. First it shows the pairwise combination matrices at $t = 1$ and $t = 2$, then presents the per-combination distributions at the same sizes, followed by system-specific distributions for each of our ten considered subject systems at reference sample size $t = 3$ for the prediction error only. Finally, we report an alternative visualisation of our combination ranking as matrices at each reference sample size.

A.1.1 Pairwise combination matrices at smaller reference sizes

This subsection complements the results for the combination comparisons for smaller reference sample sizes $t = 2$ and $t = 1$ for both prediction error ($\overline{MRE}$) and stability (Var).

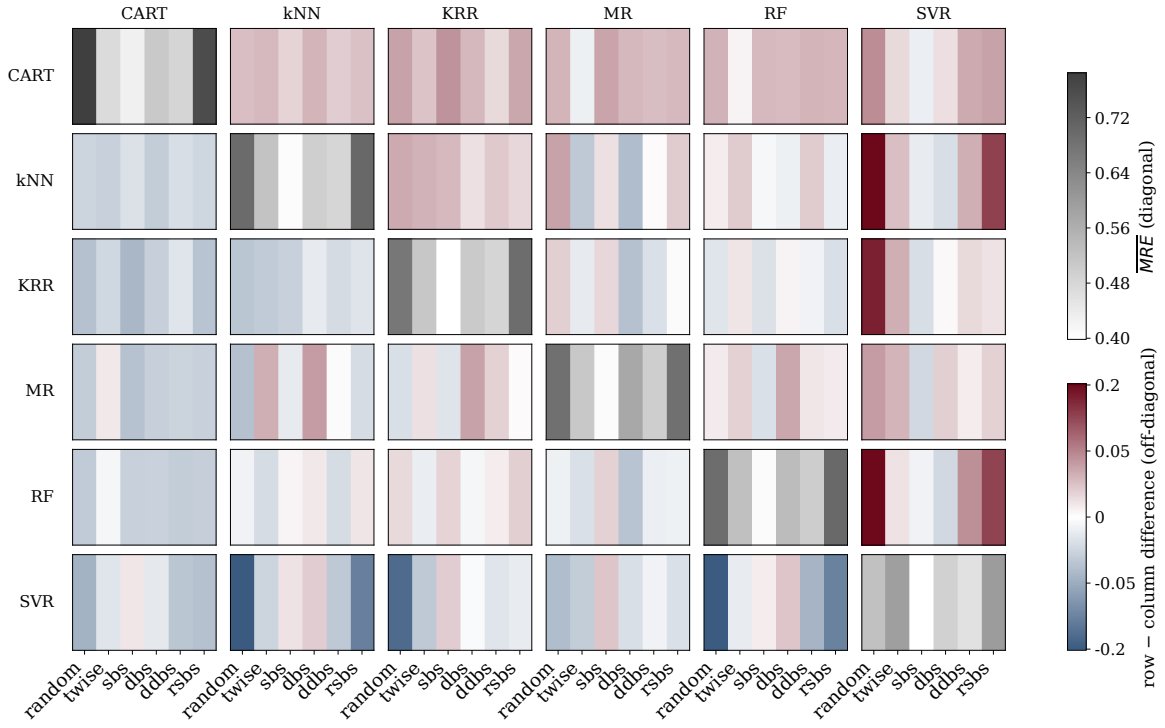


Figure A.1: Pairwise combination comparison for $\overline{MRE}$ at $t = 1$. Layout as in Figure 4.1.

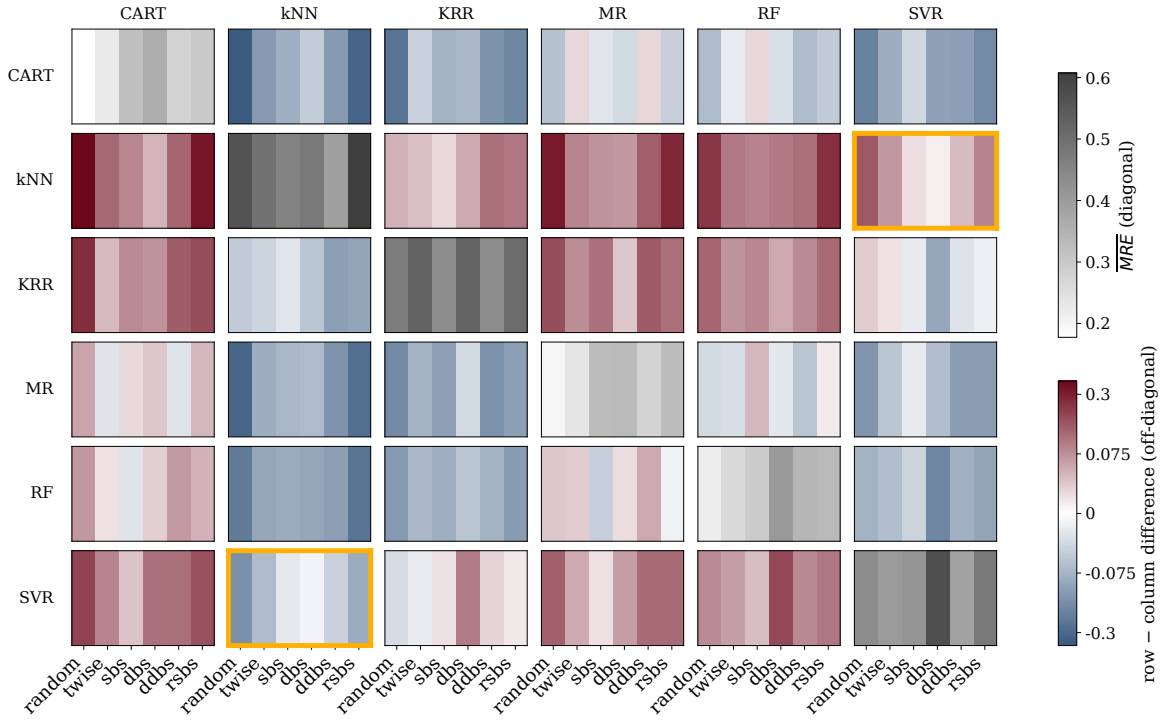


Figure A.2: Pairwise combination comparison for $\overline{MRE}$ at $t = 2$. Layout as in Figure 4.1.

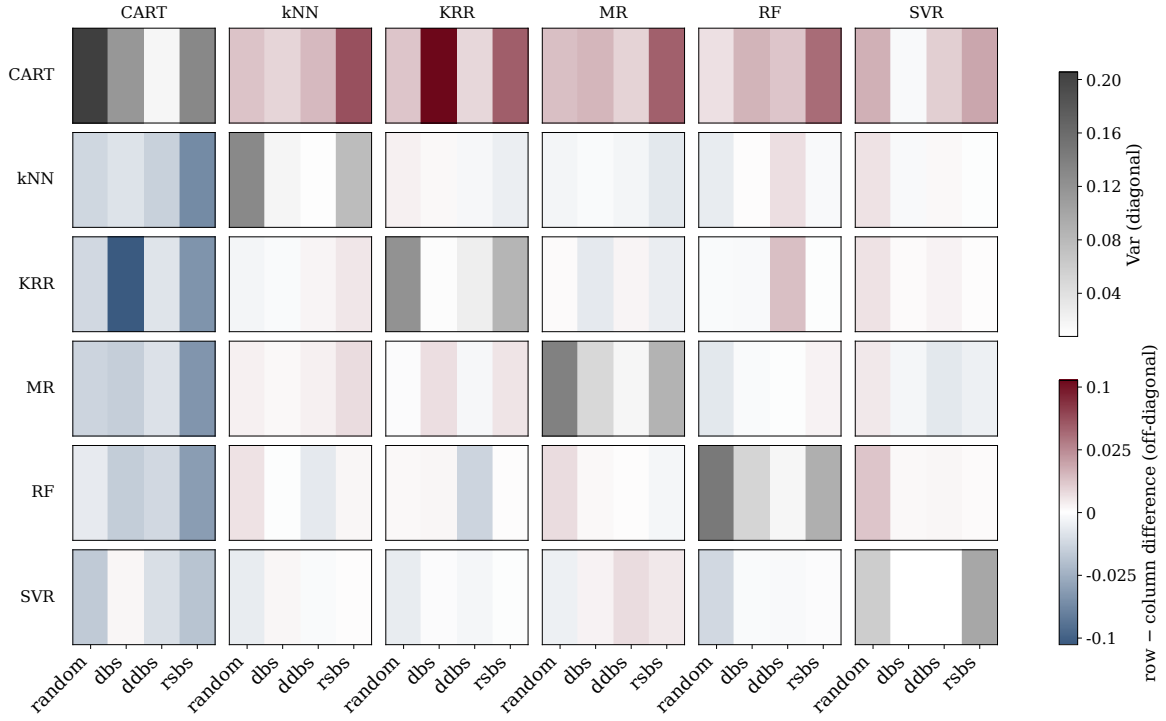


Figure A.3: Pairwise combination comparison for *Var* at $t = 1$. Layout as in Figure 4.2.

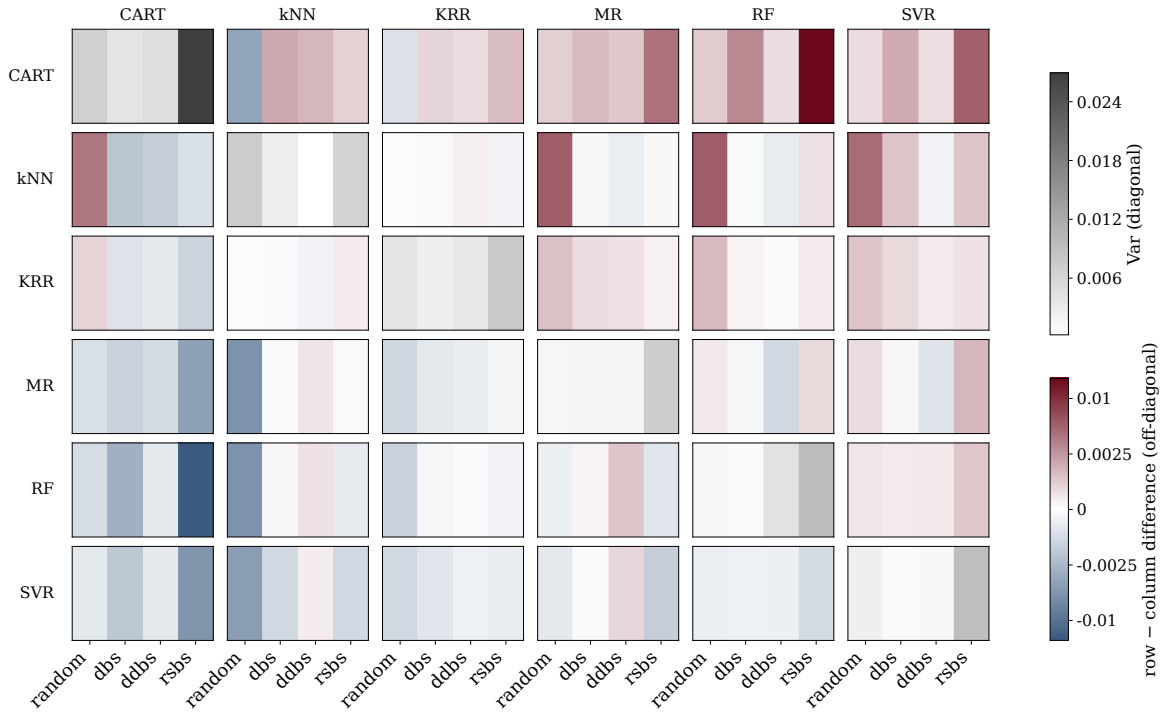


Figure A.4: Pairwise combination comparison for *Var* at $t = 2$. Layout as in Figure 4.2.

a.1.2 Aggregated boxplots at smaller reference sizes

This subsection complements the results for the per-combination distributions at smaller reference sample sizes $t = 2$ and $t = 1$ for both prediction error ($\overline{MRE}$) and stability (Var).

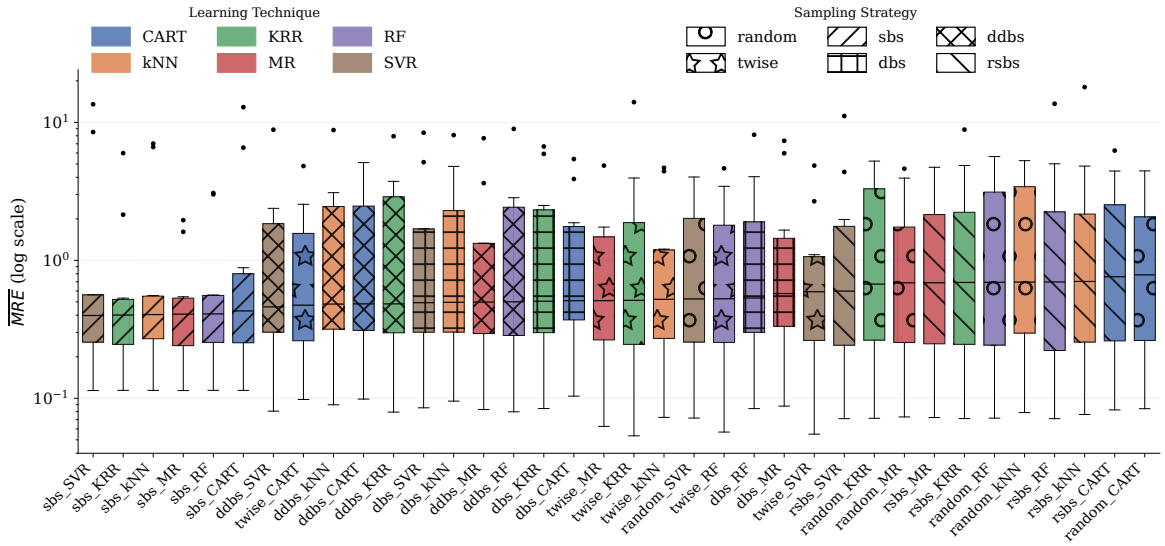


Figure A.5: Per-combination distribution of $\overline{MRE}$ across the ten subject systems at $t = 1$, ordered by median. Layout as in Figure 4.4.

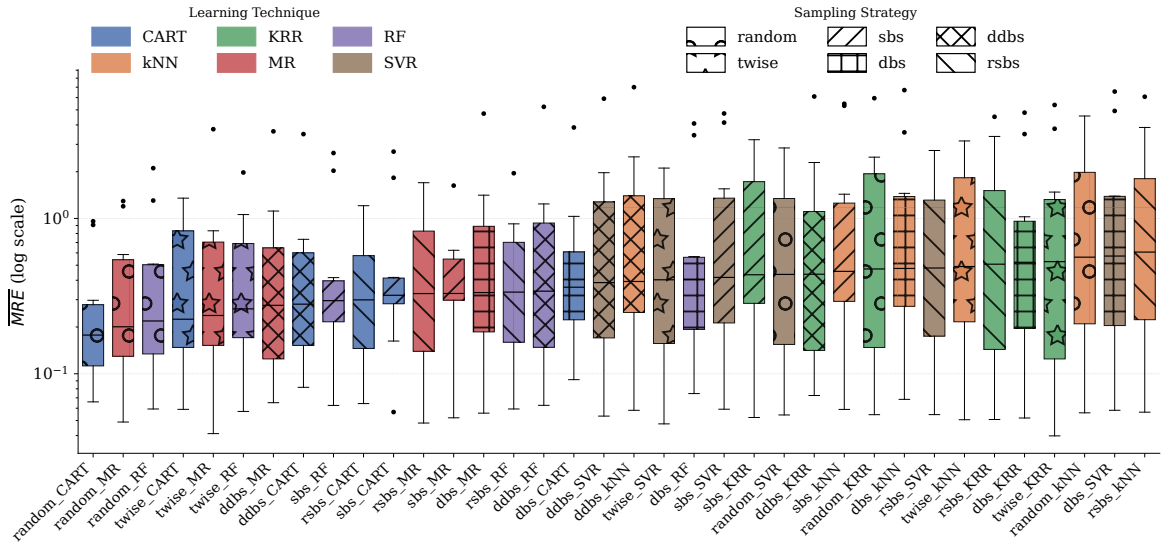


Figure A.6: Per-combination distribution of $\overline{MRE}$ across the ten subject systems at $t = 2$, ordered by median. Layout as in Figure 4.4.

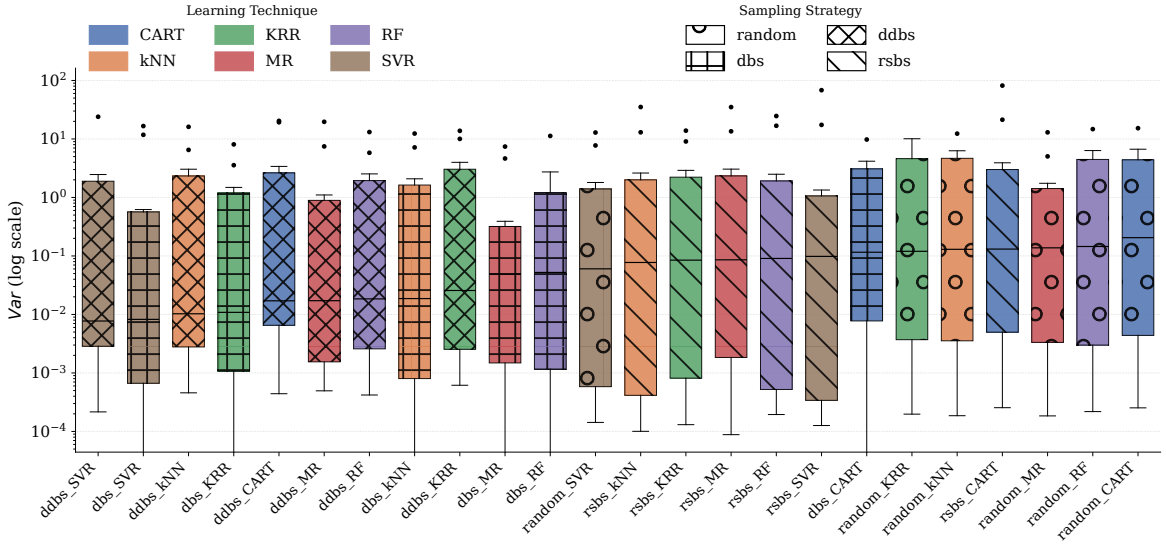


Figure A.7: Per-combination distribution of *Var* across the ten subject systems at $t = 1$, ordered by median. Layout as in Figure 4.5.

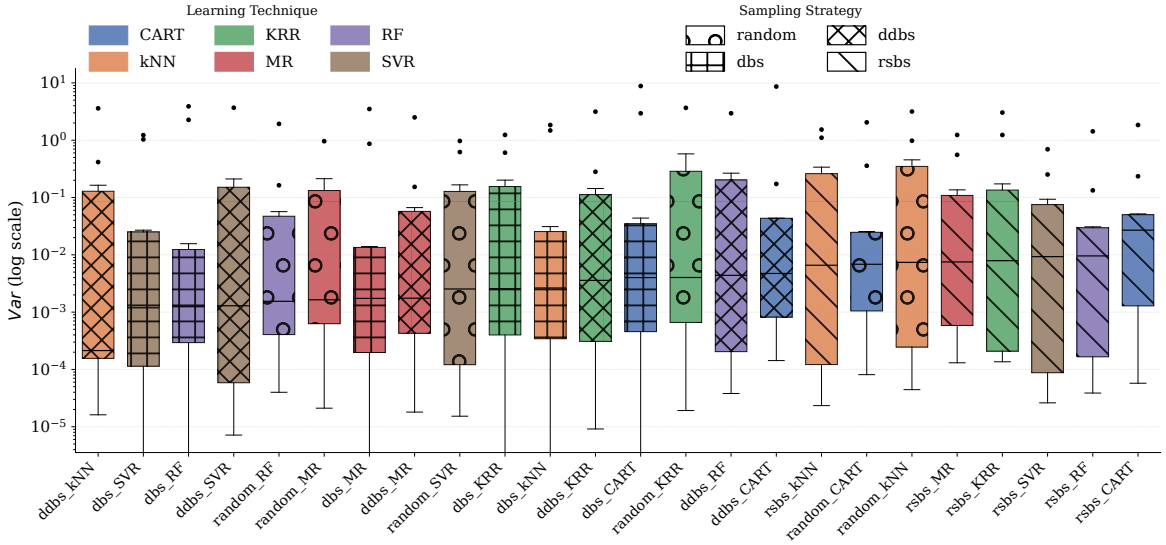


Figure A.8: Per-combination distribution of *Var* across the ten subject systems at $t = 2$, ordered by median. Layout as in Figure 4.5.

a.1.3 Per-system boxplots for prediction error at $t = 3$

Beyond the analysis of prediction error and stability aggregated across all subject systems, we also complement the upper results above with system-specific per-combination distributions. In these figures, we consider only the reference sample size $t = 3$ for the prediction error ($\overline{MRE}$).

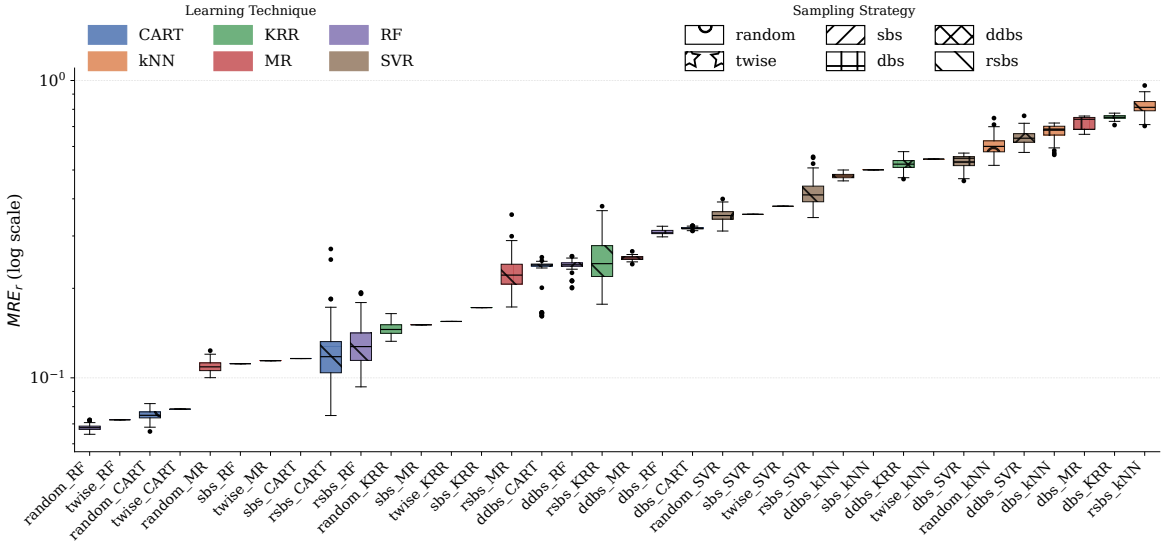


Figure A.9: Per-combination distribution of $\overline{MRE}$ over 100 seeds for 7z at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

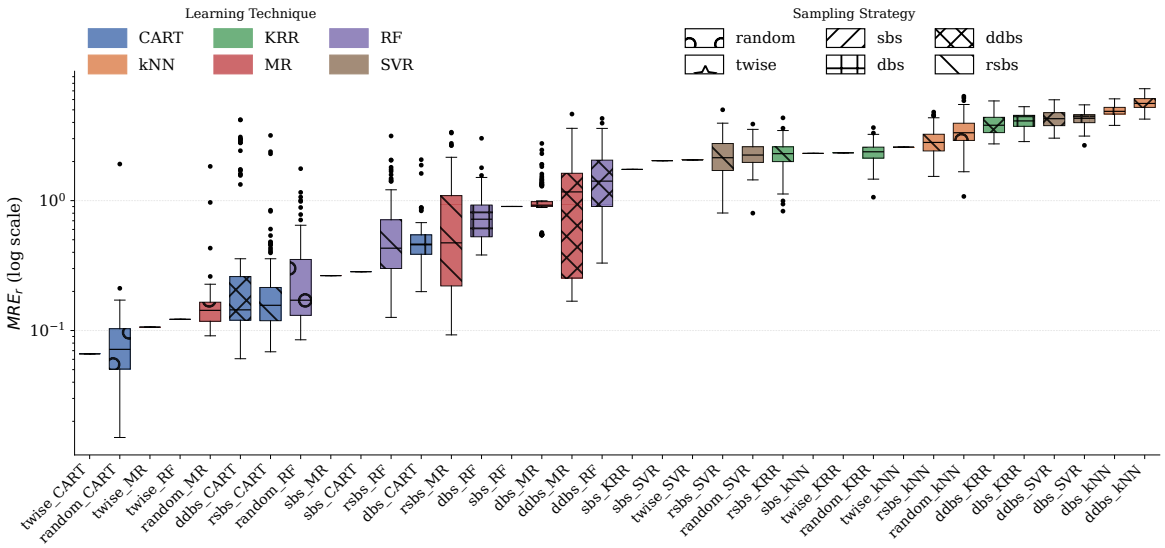


Figure A.10: Per-combination distribution of $\overline{MRE}$ over 100 seeds for BerkeleyDBC at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

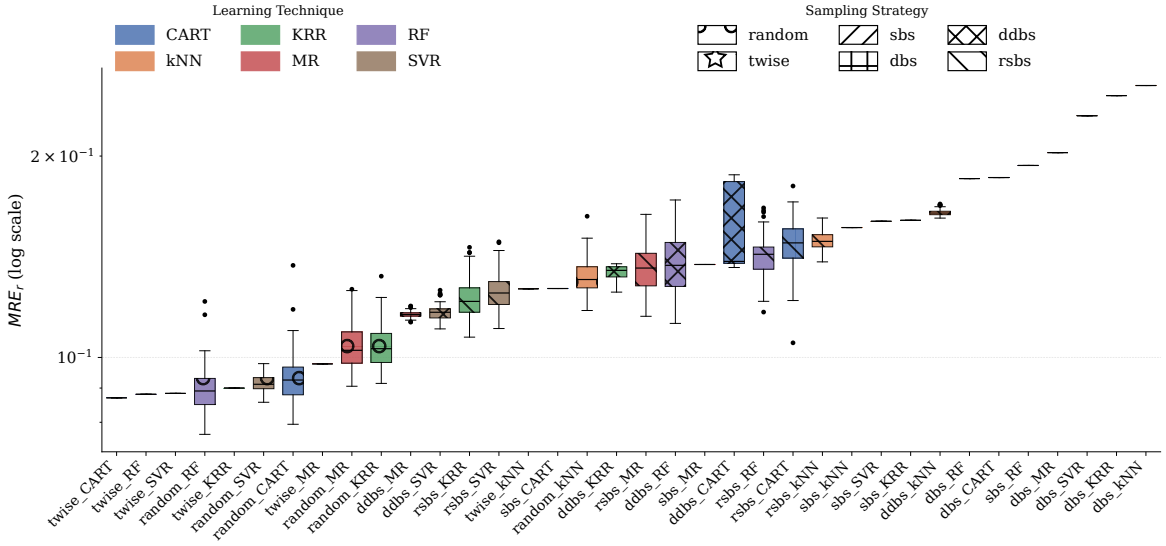


Figure A.11: Per-combination distribution of $\overline{MRE}$ over 100 seeds for DUNE at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

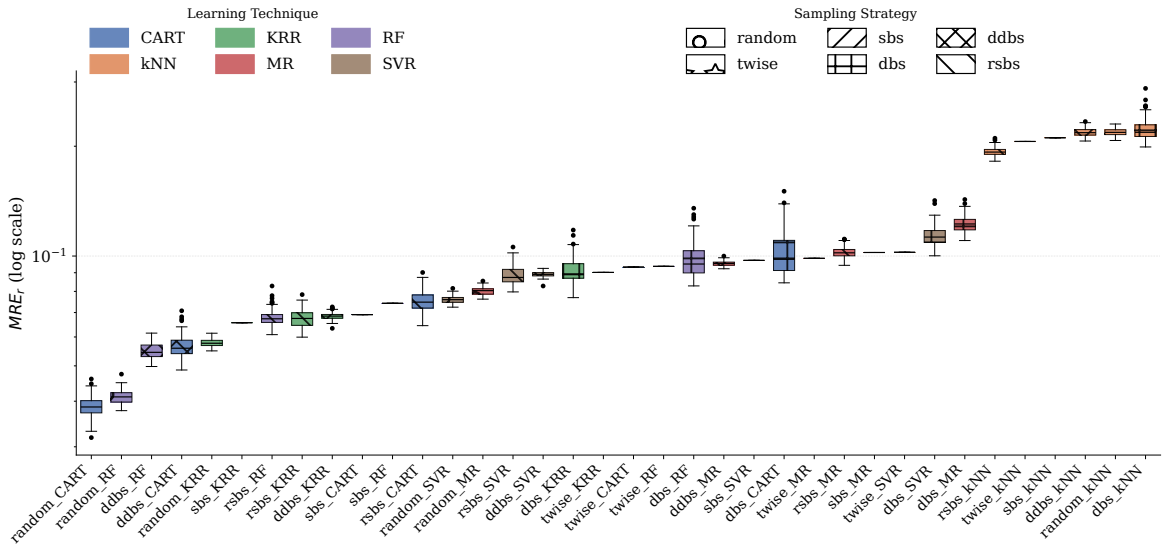


Figure A.12: Per-combination distribution of $\overline{MRE}$ over 100 seeds for HIPACC at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

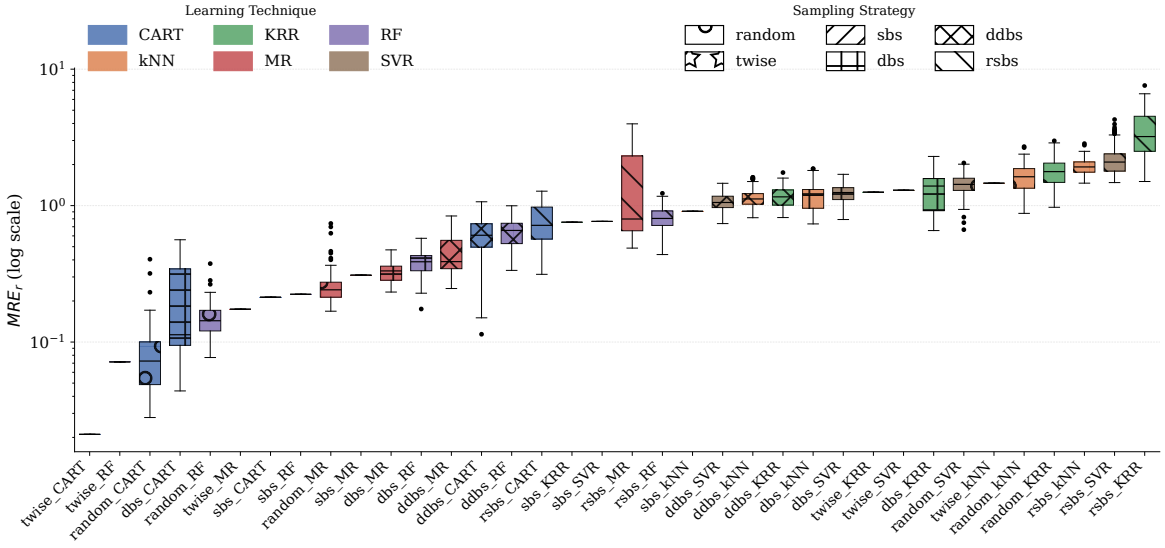


Figure A.13: Per-combination distribution of $\overline{MRE}$ over 100 seeds for LRZIP at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

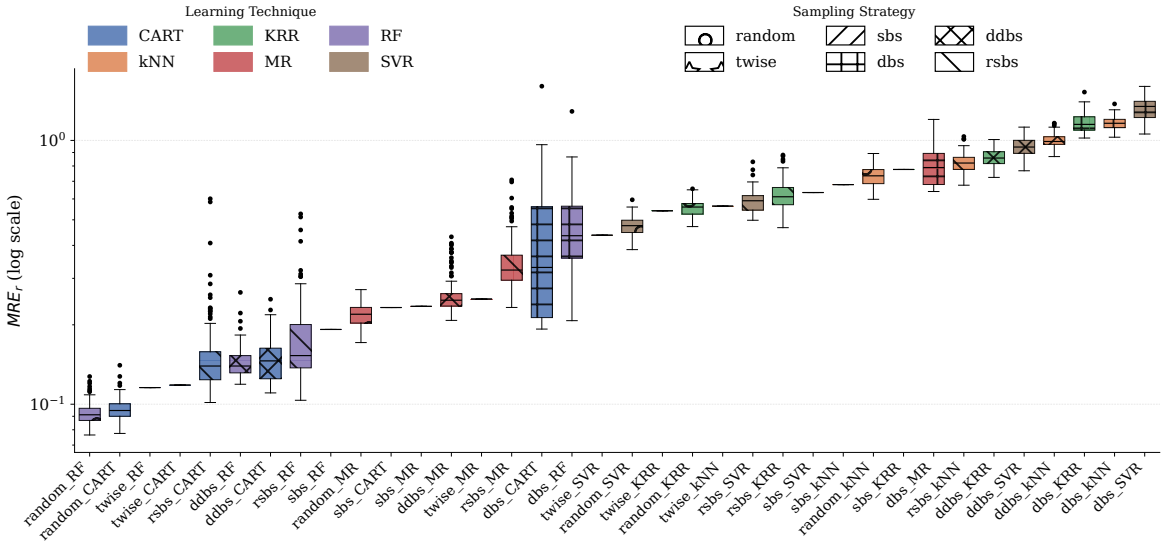


Figure A.14: Per-combination distribution of $\overline{MRE}$ over 100 seeds for JAVAGC at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

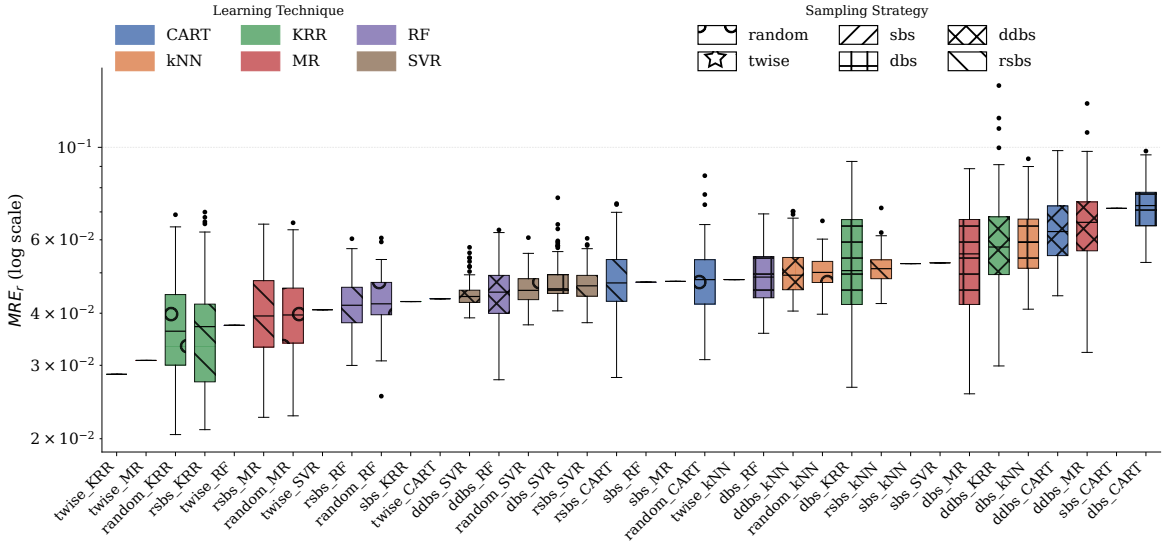


Figure A.15: Per-combination distribution of $\overline{MRE}$ over 100 seeds for LLVM at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

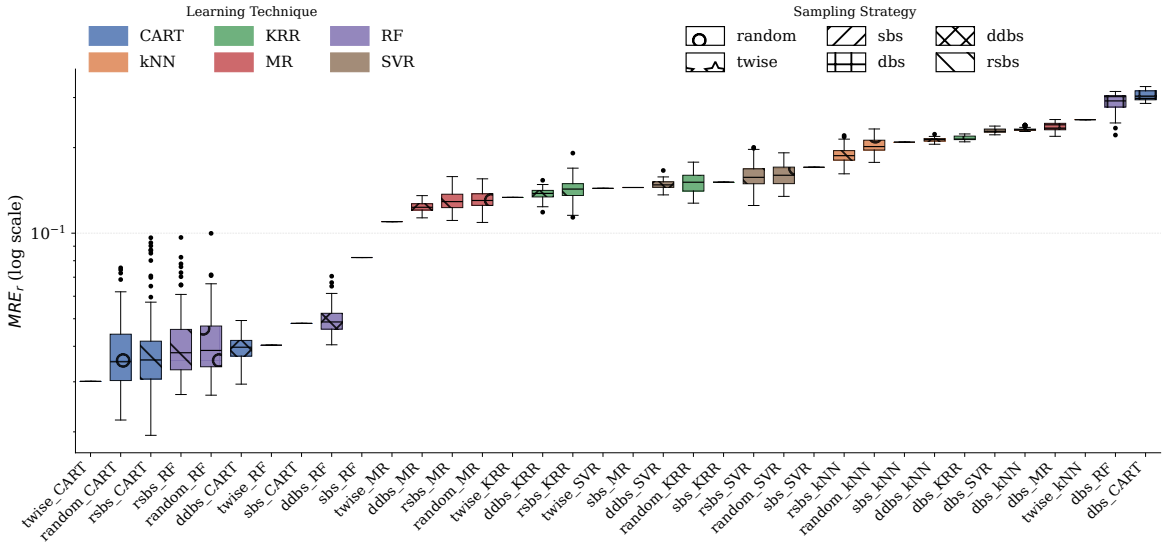


Figure A.16: Per-combination distribution of $\overline{MRE}$ over 100 seeds for POLLY at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

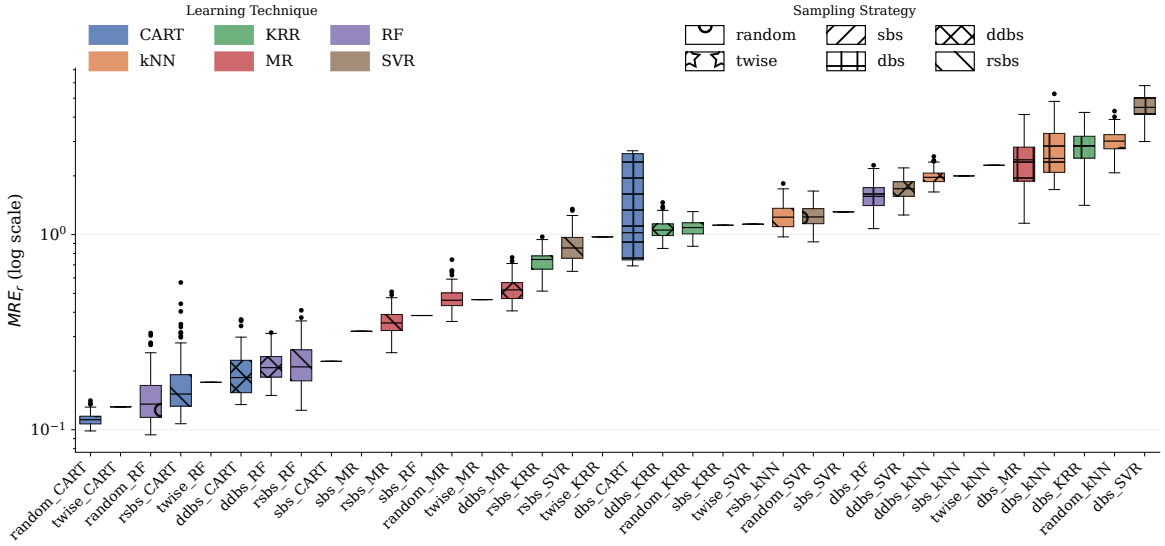


Figure A.17: Per-combination distribution of $\overline{MRE}$ over 100 seeds for VP9 at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

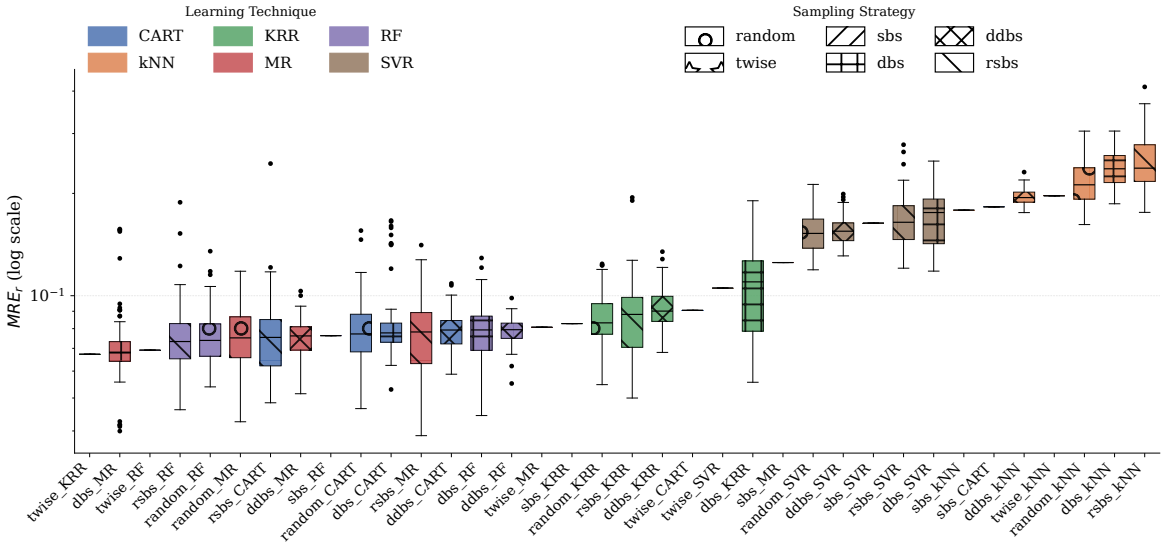


Figure A.18: Per-combination distribution of $\overline{MRE}$ over 100 seeds for x264 at $t = 3$, ordered by median. Deterministic strategies appear as flat markers (single seed). Layout as in Figure 4.4.

a.1.4 Mean-rank matrices

This final subsection shows an alternative ranking visualisation for each combination at all reference sample sizes $t = 1$, $t = 2$ and $t = 3$ for both prediction error ($\overline{MRE}$) and stability (Var). Each cell shows the mean rank of a combination across the ten subject systems, from

1 for the best combination to 36 for $\overline{MRE}$ and 24 for *Var*, on a white-blue scale (blue = lower rank/performs worse, white = higher rank/performs best).

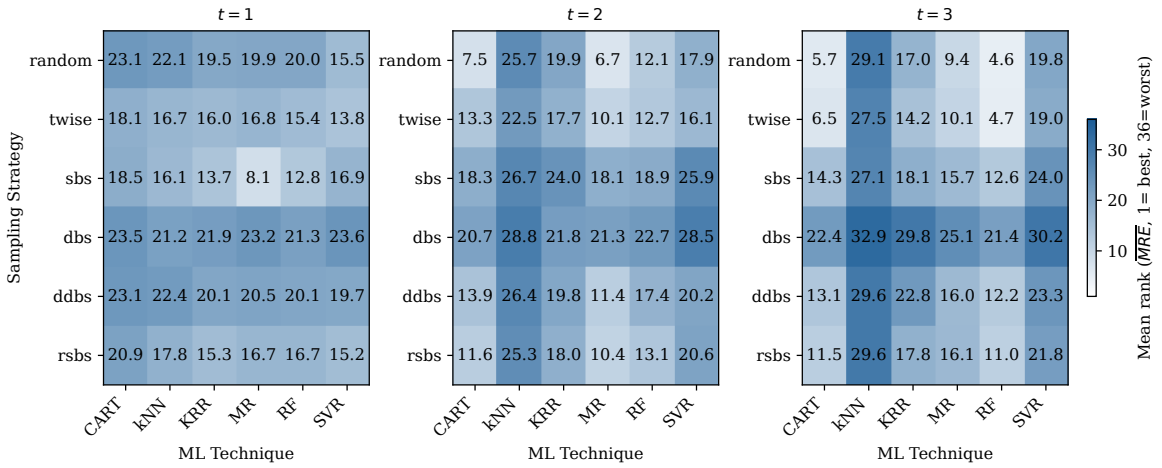


Figure A.19: Mean rank across the ten subject systems of each combination of sampling strategy and learning technique for $\overline{MRE}$ at each reference sample size. Rank 1 is the best and rank 36 is the worst.

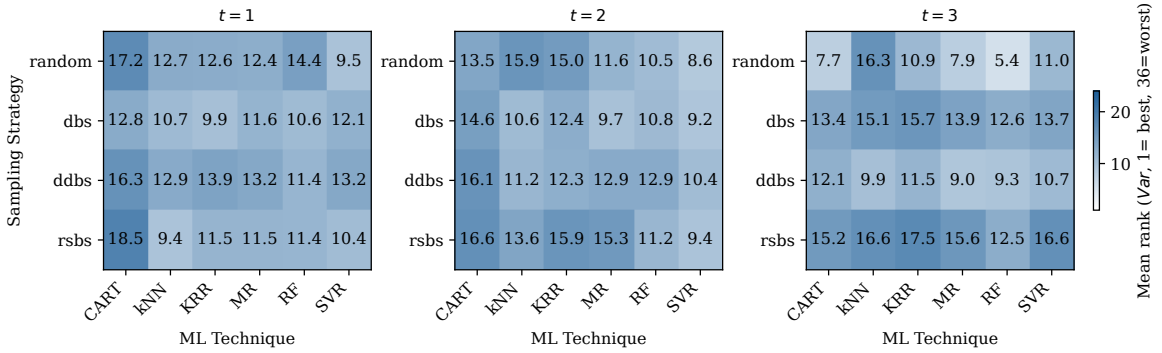


Figure A.20: Mean rank across the ten subject systems of each combination of sampling strategy and learning for *Var* at each reference sample size. Rank 1 is the best and rank 24 is the worst, since deterministic strategies are excluded by construction.

a.2 Hyperparameters

Table A.1 lists the hyperparameter values of the original study and of this thesis. The original values are taken from the supplementary website of the original study by Grebhahn et al. [12][1].

1 https://www.se.cs.uni-saarland.de/projects/splconqueror/expDesign/logFileOptimalParameters.html

Technique	Hyperparameter	Original value	Value for this thesis
CART	splitter	best, random	best, random
	max_features	all, log, sqrt	all, sqrt, log2
	min_samples_leaf	1, 2	1, 2
	random_state	1, 2, 3	0 fixed seed
*k*NN	n_neighbors	8, 9, . . . , 15	8, 10, 12, 15
	weights	distance, uniform	distance, uniform
	p	1, 2, 3	1, 2
	algorithm	auto	auto
KRR	alpha	10^{-6}	$10^{-4}, 10^{-2}, 10^{-1}$
	kernel	linear, poly, rbf	rbf
	gamma	0.01, 0.05, 0.1, 0.2	0.01, 0.05, 0.1, 0.2
	degree	1, 2, 3	–
MR	function types (3)	true, false	–
	lossFunction	absolute, relative	–
	minImprovementPerRound	0.1, 0.5, 1	–
	Lasso α	–	$10^{-3}, 10^{-2}, 10^{-1}, 1$
RF	n_estimators	10, 12, 15, 18, 20	10, 15, 20
	max_features	all, log2, sqrt	all, log2, sqrt
	min_samples_leaf	–	1, 2
	random_state	1, 2, 3	0 fixed seed
SVR	C	0.5, 1, 2, 5	0.5, 1, 2, 5
	ε	0.1, 0.2, 0.3	0.1, 0.2, 0.3
	coefo	0, 1, 2	0, 1, 2
	shrinking	true, false	true, false
	tol	0.05, 0.1, 0.2, 0.5	0.05, 0.1, 0.2, 0.3

Table A.1: Hyperparameter values of the original study and of this thesis. The original values are the union of all values selected in the log of the original study. *all* denotes using all options at each split.

B

Statement on the Usage of Generative Digital Assistants

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, ClaudeAI) exclusively for the following permitted sub-tasks: text rewriting/revision, check for textual mistakes such a grammar and spelling erros, brainstorming, code completion and finding of bugs, and not to address or formulate the main research questions of the thesis.
I am well acquainted with the potential risks, exhibited by the usage of AI tools and have utilized them with care and critical evaluation. No blind trust has been put into the results of these tools, instead they have been checked and only included if they aligned with my own understanding. Core ideas, structure and general conception of this thesis were set up independently without resorting to these tools.

Bibliography

[1] Sheldon B. Akers. "Binary Decision Diagrams." In: *IEEE Transactions on Computers* C-27.6 (1978), pp. 509–516. DOI: 10.1109/TC.1978.1675141.

[2] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. "Sampling Effect on Performance Prediction of Configurable Systems: A Case Study." In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. ICPE '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 277–288. DOI: 10.1145/3358960.3379137.

[3] 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 (2021), p. 111044. DOI: 10.1016/j.jss.2021.111044.

[4] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Berlin, Heidelberg: Springer, 2013. DOI: 10.1007/978-3-642-37521-7.

[5] Mariette Awad and Rahul Khanna. "Machine Learning." In: *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015, pp. 1–18. DOI: 10.1007/978-1-4302-5990-9_1.

[6] Yoav Benjamini and Yosef Hochberg. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 57.1 (1995), pp. 289–300. DOI: 10.2307/2346101. URL: https://www.researchgate.net/publication/221995234_Controlling_The_False_Discovery_Rate_-_A_Practical_And_Powerful_Approach_To_Multiple_Testing.

[7] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." In: *Journal of Machine Learning Research* 13 (2012), pp. 281–305. URL: https://jmlr.org/papers/v13/bergstra12a.html.

[8] Lawrence Chung and Julio Cesar Sampaio do Prado Leite. "On Non-Functional Requirements in Software Engineering." In: *Conceptual Modeling: Foundations and Applications – Essays in Honor of John Mylopoulos*. Ed. by Alexander Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. K. Yu. Vol. 5600. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 363–379. DOI: 10.1007/978-3-642-02463-4_19.

[9] Clemens Dubslaff, Nils Husung, and Nikolai Käfer. "Configuring BDD Compilation Techniques for Feature Models." In: *Proceedings of the 28th ACM International Systems and Software Product Line Conference*. SPLC '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 209–216. DOI: 10.1145/3646548.3676538.

[10] Rosa L. Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H. Ngo. "Predicting Sample Size Required for Classification Performance." In: *BMC Medical Informatics and Decision Making* 12 (Feb. 2012), p. 8. DOI: 10.1186/1472-6947-12-8.

[11] Joseph L. Gastwirth, Yulia R. Gel, and Weiwen Miao. "The Impact of Levene's Test of Equality of Variances on Statistical Theory and Practice." In: *Statistical Science* 24.3 (2009), pp. 343–360. DOI: 10.1214/09-STS301.

[12] Alexander Grebhahn, Norbert Siegmund, and Sven Apel. *Predicting Performance of Software Configurations: There is no Silver Bullet*. 2019. URL: https://doi.org/10.48550/arXiv.1911.12643.

[13] 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 (2018), pp. 1826–1867. DOI: 10.1007/s10664-017-9573-6.

[14] 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 '19. IEEE Press, 2019, pp. 1095–1106. DOI: 10.1109/ICSE.2019.00113.

[15] Rajib Kumar Halder, Mohammed Nasir Uddin, Md. Ashraf Uddin, Sunil Aryal, and Ansam Khraisat. "Enhancing K-Nearest Neighbor Algorithm: A Comprehensive Review and Performance Analysis of Modifications." In: *Journal of Big Data* 11 (2024). DOI: 10.1186/s40537-024-00973-y.

[16] Alon Halevy, Peter Norvig, and Fernando Pereira. "The Unreasonable Effectiveness of Data." In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12. DOI: 10.1109/MIS.2009.36.

[17] Axel Halin, Alexandre Nuttinck, Mathieu Acher, Xavier Devroey, Gilles Perrouin, and Benoit Baudry. "Test Them All, Is It Worth It? Assessing Configuration Sampling on the JHipster Web Development Stack." In: *Empirical Software Engineering* 24.2 (2019), pp. 674–717. DOI: 10.1007/s10664-018-9635-4.

[18] Richard W. Hamming. "Error Detecting and Error Correcting Codes." In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[19] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." In: Second. New York, NY: Springer, 2009. Chap. 9, pp. 295–336. DOI: 10.1007/978-0-387-84858-7_9.

[20] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." In: Second. New York, NY: Springer, 2009. Chap. 2, pp. 22–28. DOI: 10.1007/978-0-387-84858-7_2.

[21] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." In: Second. New York, NY: Springer, 2009. Chap. 12, pp. 434–436. DOI: 10.1007/978-0-387-84858-7_12.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." In: Second. New York, NY: Springer, 2009. Chap. 15, pp. 587–592. DOI: 10.1007/978-0-387-84858-7_15.

[23] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. "Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines." In: *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering*. Ed. by Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum. Vol. 1. ICSE '15. IEEE Computer Society, 2015, pp. 517–528. DOI: 10.1109/ICSE.2015.69.

[24] Christopher Henard, Mike Papadakis, Gilles Perrouin, Jacques Klein, Patrick Heymans, and Yves Le Traon. "Bypassing the Combinatorial Explosion: Using Similarity to Generate and Prioritize T-Wise Test Configurations for Software Product Lines." In: *IEEE Transactions on Software Engineering* 40.7 (2014), pp. 650–670. DOI: 10.1109/TSE.2014.2327020.

[25] Ruben Heradio, David Fernandez-Amoros, José A. Galindo, and David Benavides. "Uniform and Scalable SAT-Sampling for Configurable Systems." In: *Proceedings of the 24th ACM International Systems and Software Product Line Conference – Volume A*. SPLC '20. New York, NY, USA: Association for Computing Machinery, 2020, 17:1–17:11. DOI: 10.1145/3382025.3414951.

[26] Ronald L. Iman and James M. Davenport. "Approximations of the Critical Region of the Friedman Statistic." In: *Communications in Statistics – Theory and Methods* 9.6 (1980), pp. 571–595. DOI: 10.1080/03610928008827904.

[27] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. "Linear Regression." In: *An Introduction to Statistical Learning: with Applications in R*. New York, NY: Springer, 2013. Chap. 3, pp. 59–126. DOI: 10.1007/978-1-4614-7138-7_3.

[28] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. "The Interplay of Sampling and Machine Learning for Software Performance Prediction." In: *IEEE Software* 37.4 (July 2020), pp. 58–66. DOI: 10.1109/MS.2020.2987024.

[29] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. "Distance-based Sampling of Software Configuration Spaces." In: *Proceedings of the 41st International Conference on Software Engineering*. ICSE '19. IEEE Press, 2019, pp. 1084–1094. DOI: 10.1109/ICSE.2019.00112.

[30] Kyo Kang, Sholom Cohen, James Hess, William Novak, and A. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Tech. rep. CMU/SEI-90-TR-021. Software Engineering Institute, Carnegie Mellon University, 1990. URL: https://www.sei.cmu.edu/library/feature-oriented-domain-analysis-foda-feasibility-study/.

[31] Maurice G. Kendall and B. Babington Smith. "The Problem of *m* Rankings." In: *The Annals of Mathematical Statistics* 10.3 (1939), pp. 275–287. DOI: 10.1214/aoms/1177732186. URL: http://www.jstor.org/stable/2235668.

[32] Andy Kenner, Richard May, Jacob Krüger, Gunter Saake, and Thomas Leich. "Safety, Security, and Configurable Software Systems: A Systematic Mapping Study." In: *Proceedings of the 25th ACM International Systems and Software Product Line Conference – Volume A*. SPLC '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 148–159. DOI: 10.1145/3461001.3471147.

[33] 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.

[34] Sebastian Krieter, Thomas Thüm, Sandro Schulze, Sebastian Ruland, Malte Lochau, Gunter Saake, and Thomas Leich. *T-Wise Presence Condition Coverage and Sampling for Configurable Systems*. 2022. URL: https://arxiv.org/abs/2205.15180.

[35] *Log File of Optimal Hyperparameters*. Supplementary website of the study by Grebhahn et al. URL: https://www.se.cs.uni-saarland.de/projects/splconqueror/expDesign/logFileOptimalParameters.html.

[36] Flávio Medeiros, Christian Kästner, Márcio Ribeiro, Rohit Gheyi, and Sven Apel. "A Comparison of 10 Sampling Algorithms for Configurable Systems." In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 643–654. DOI: 10.1145/2884781.2884793.

[37] Marcilio Mendonca, Andrzej Wasowski, Krzysztof Czarnecki, and Donald Cowan. "Efficient Compilation Techniques for Large Scale Feature Models." In: *Proceedings of the 7th International Conference on Generative Programming and Component Engineering*. GPCE '08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 13–22. DOI: 10.1145/1449913.1449918.

[38] Douglas C. Montgomery. *Design and Analysis of Experiments*. 9th. New York: John Wiley & Sons, 2017. URL: https://www.researchgate.net/publication/362079778_Design_and_Analysis_of_Experiments_9th_Edition.

[39] Stefan Mühlbauer, Florian Sattler, Christian Kaltenecker, Johannes Dorn, Sven Apel, and Norbert Siegmund. "Analyzing the Impact of Workloads on Modeling the Performance of Configurable Software Systems." In: *Proceedings of the 45th International Conference on Software Engineering*. ICSE '23. IEEE Press, 2023, pp. 2085–2097. DOI: 10.1109/ICSE48619.2023.00176. URL: https://doi.org/10.1109/ICSE48619.2023.00176.

[40] Daniel-Jesus Munoz, Jeho Oh, Mónica Pinto, Lidia Fuentes, and Don Batory. "Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features." In: *Proceedings of the 23rd International Systems and Software Product Line Conference – Volume A*. SPLC '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 289–301. DOI: 10.1145/3336294.3336297.

[41] Raphael Pereira de Oliveira, Paulo Anselmo da Mota Silveira Neto, Qi Hong Chen, Eduardo Santana de Almeida, and Iftekhar Ahmed. "Different, Really! A Comparison of Highly-Configurable Systems and Single Systems." In: *Information and Software Technology* 152 (2022), p. 107035. DOI: 10.1016/j.infsof.2022.107035.

[42] Dulce G. Pereira, Anabela Afonso, and Fátima Melo Medeiros. "Overview of Friedman's Test and Post-hoc Analysis." In: *Communications in Statistics – Simulation and Computation* 44.10 (2015), pp. 2636–2653. DOI: 10.1080/03610918.2014.931971.

[43] Robin L. Plackett and J. P. Burman. "The Design of Optimum Multifactorial Experiments." In: *Biometrika* 33.4 (1946), pp. 305–325. DOI: 10.2307/2332195.

[44] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. "Tunability: Importance of Hyperparameters of Machine Learning Algorithms." In: *Journal of Machine Learning Research* 20.53 (2019), pp. 1–32. URL: https://jmlr.org/papers/v20/18-444.html.

[45] Oona Rainio, Jarmo Teuho, and Riku Klén. "Evaluation Metrics and Statistical Tests for Machine Learning." In: *Scientific Reports* 14 (2024), p. 6086. DOI: 10.1038/s41598-024-56706-x.

[46] Daniyal Rajput, Wei-Jen Wang, and Chun-Chuan Chen. "Evaluation of a Decided Sample Size in Machine Learning Applications." In: *BMC Bioinformatics* 24.1 (2023), p. 48. DOI: 10.1186/s12859-023-05156-9.

[47] Arthur L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers." In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[48] Atrisha Sarkar, Jianmei Guo, Norbert Siegmund, Sven Apel, and Krzysztof Czarnecki. "Cost-Efficient Sampling for Performance Prediction of Configurable Systems." In: *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*. ASE '15. IEEE, 2015, pp. 342–352. DOI: 10.1109/ASE.2015.45.

[49] Roy C. Schmidt. "Managing Delphi Surveys Using Nonparametric Statistical Techniques." In: *Decision Sciences* 28.3 (1997), pp. 763–774. DOI: 10.1111/j.1540-5915.1997.tb01330.x.

[50] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. "Performance-Influence Models for Highly Configurable Systems." In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, 2015, pp. 284–294. DOI: 10.1145/2786805.2786845.

[51] Charles Spearman. "The Proof and Measurement of Association between Two Things." In: *International Journal of Epidemiology* 39.5 (Oct. 2010), pp. 1137–1150. DOI: 10.1093/ije/dyq191.

[52] Tala Talaei Khoei and Naima Kaabouch. "Machine Learning: Models, Challenges, and Research Directions." In: *Future Internet* 15.10 (2023), p. 332. DOI: 10.3390/fi15100332.

[53] O. Taylan Turan, Marco Loog, and David M. J. Tax. "Generalization Performance Distributions along Learning Curves." In: *Pattern Recognition Letters* 201 (2026), pp. 29–36. DOI: 10.1016/j.patrec.2026.01.003.

[54] Max Weber, Christian Kaltenecker, Florian Sattler, Sven Apel, and Norbert Siegmund. "Twins or False Friends? A Study on Energy Consumption and Performance of Configurable Software." In: *Proceedings of the 45th International Conference on Software Engineering*. ICSE '23. IEEE Press, 2023, pp. 2098–2110. DOI: 10.1109/ICSE48619.2023.00177.

[55] Denise Welsch and Markus Neuhäuser. "Wilcoxon-Signed-Rank Test." In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Springer Berlin Heidelberg, 2025, pp. 2898–2900. DOI: 10.1007/978-3-662-69359-9_722.

[56] Tianyin Xu, Long Jin, Xuepeng Fan, Yuanyuan Zhou, Shankar Pasupathy, and Rukma Talwadker. "Hey, You Have Given Me Too Many Knobs!: Understanding and Dealing with Over-Designed Configuration in System Software." In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, 2015, pp. 307–319. DOI: 10.1145/2786805.2786852.

[57] Olivier Zeyen, Maxime Cordy, Martin Gubri, Gilles Perrouin, and Mathieu Acher. *Testing Uniform Random Samplers: Methods, Datasets and Protocols*. 2025. URL: https://arxiv.org/abs/2503.14079.

[58] Yutong Zhao, Lu Xiao, Chenhao Wei, Rick Kazman, and Ye Yang. *A Systematic Mapping Study on Architectural Approaches to Software Performance Analysis*. 2024. URL: https://arxiv.org/abs/2410.17372.