Synthetic Tabular Data Augmentation for Imbalanced Regression



António Pedro Correia da Silva Pinheiro
Master in Information Security
Department of Computer Science
Faculty of Sciences of the University of Porto
2025

Synthetic Tabular Data Augmentation for Imbalanced Regression

António Pedro Correia da Silva Pinheiro
Dissertation carried out as part of the Master in Information Security
Department of Computer Science
2025

Supervisor
Prof. Dr.ª Rita P. Ribeiro, Assistant Professor, University of Porto

“If you’re lucky enough to be different, never change.
And if you’re lucky enough to find something you love, and you have a shot at being good at it, don’t stop, don’t put it down.”

Taylor Swift

Acknowledgements

I would like to start by thanking Professor Rita Ribeiro for the guidance throughout the whole process and sharing her insights, knowledge and expertise with me, as well as the patience to navigate this project's waters with me.

Secondly, a very special thanks to my parents, who not only encouraged me to take on this challenge, for a second time, but also supported me throughout my whole life and especially in this moment, by always believing in me. Your example, tenacity, drive and perseverance have made me aim higher, and to set my own path in this world. To my sister, for always being a ray of sunshine, for making me laugh uncontrollably, and for pretending to understand what I am doing when she asks me for help with her computer. To my grandfather, who was always concerned if I was eating enough while working. And to my dog, Binky, for keeping me company throughout all those endless afternoons while I worked on this thesis, and never leaving my side by the desk.

I would also like to express my gratitude to Vinícius, for spending several hours on video calls with me while I was working on this thesis and listening to me vent so many times when something in my code didn't work and I didn't know where or what the error was. Thank you for pretending to understand a single thing I was saying. Your company and encouragement were fundamental to this whole process.

Finally, I would like to thank my closest friends, for making this journey lighter and more fun and for encouraging me to pursue my dreams, including this one.

As I finish this journey, I feel privileged to have had this opportunity, among all the many opportunities I have had throughout my life.

To all, I address my sincerest thank you.

Resumo

Abordar a questão de distribuições desequilibradas na variável-alvo em problemas de regressão continua a ser um desafio pouco explorado, particularmente em cenários de dados tabulares em que as regiões com poucos registos podem degradar a precisão da previsão. As abordagens ao nível dos dados, como a amostragem aleatória e as técnicas inspiradas no algoritmo SMOTE, tentaram resolver este problema adaptando as estratégias de classificação às tarefas de regressão. No entanto, e como resultado destas adaptações, estes métodos dependem muitas vezes de limites rígidos definidos manualmente sobre a variável objetivo, que pode resultar em reformulações arbitrárias e pouco intuitivas do problema original. Soluções mais recentes que utilizam modelos generativos, como *Generative Adversarial Networks* e *Variational Autoencoders*, permitem uma geração de amostras mais adaptativa, mas sofrem muitas vezes de elevadas exigências computacionais e de reduzida interpretabilidade. Neste trabalho, apresentamos uma versão modificada de um método de geração de dados sintéticos baseado no algoritmo CART, especificamente concebido para problemas de regressão desequilibrada. A nossa abordagem incorpora métodos de relevância e de densidade para concentrar a geração de dados em áreas sub-representadas da variável objetivo, eliminando a necessidade de limiares fixos ao confiar num processo de geração sem limiares e guiado pelas caraterísticas dos dados, e possibilitando a aprendizagem em tipos de dados heterogéneos bem como intercorrelações e distribuições complexas. No nosso estudo experimental, validamos o nosso método através de experiências que visam a previsão de valores raros e extremos em conjuntos de dados tabulares de referência para este tipo de tarefa. Os resultados mostram que o nosso método tem um desempenho competitivo em relação a outras abordagens de reamostragem e generativas, ao mesmo tempo que oferece vantagens notáveis em termos de velocidade e interpretabilidade. Em geral, a técnica proposta surge como uma solução transparente e escalável ao nível dos dados para melhorar o desempenho da regressão em contextos desequilibrados.

Palavras-chave: Aprendizagem a partir de Domínios Desequilibrados, Regressão Desequilibrada, Previsão de Valores Extremos, Dados Tabulares, Geração de Dados Sintéticos, Amostragem

Abstract

Addressing the issue of imbalanced target distribution in regression problems remains a critical challenge, particularly in tabular data scenarios where sparsely populated regions can degrade predictive accuracy. Data-level approaches, such as random sampling and SMOTE-inspired techniques, have attempted to tackle this by extending classification strategies to regression tasks. However, these methods often rely on strict, manually set thresholds for the target variable, which are inherited from approaches used for imbalanced classification tasks, and can lead to arbitrary and unintuitive problem definitions. More recent solutions using generative models such as Generative Adversarial Networks and Variational Autoencoders allow for more adaptive sample generation but often suffer from high computational demands and reduced interpretability. In this work, we present a modified version of a CART-based synthetic data generation method, specifically designed for imbalanced regression problems. Our approach incorporates both relevance and density information to focus sampling on underrepresented areas of the target space, eliminating the need for fixed thresholds by relying instead on a feature-guided, threshold-free generation process, making it suitable for heterogeneous data types, complex inter-correlations, and intricate column-wise distributions. We validate our method through experiments targeting the prediction of rare or extreme values across benchmark datasets. The findings show that our method performs competitively with other resampling and generative approaches, while offering notable advantages in speed and interpretability. Overall, the proposed technique emerges as a transparent and scalable data-level solution for enhancing regression performance in imbalanced settings.

Keywords: Imbalanced Domain Learning, Imbalanced Regression, Extreme Value Prediction, Tabular Data, Synthetic Data Generation, Sampling

Table of Contents

List of Tables viii
List of Figures ix
List of Abbreviations xi

1. Introduction 1
   1.1. Motivation 1
   1.2. Objectives 2
   1.3. Bibliographic Notes 2
   1.4. Organisation 3
2. Background 4
   2.1. Imbalanced Regression 4
   2.2. Application Domains 5
      2.2.1. Cybersecurity 5
      2.2.2. Environment and Meteorology 5
      2.2.3. Finance 6
      2.2.4. Utilities 6
   2.3. Challenges in Imbalanced Regression 7
      2.3.1. Target Variable Preferences Specification 7
         2.3.1.1. Relevance Function 8
         2.3.1.2. DenseWeight 9
      2.3.2. Evaluation Metrics 11
         2.3.2.1. Threshold Dependent 12
         2.3.2.2. Relevance-Aware 12
      2.3.3. Modelling Approaches 14
         2.3.3.1. Data-Level Approaches 14
         2.3.3.2. Algorithm-level Approaches 14
         2.3.3.3. Hybrid Approaches 15
3. Related Work 16
   3.1. Sampling-based Approaches 17
      3.1.1. Random Undersampling (RU) 17
      3.1.2. Random Oversampling (RO) 17
      3.1.3. WEighted Relevance-based Combination Strategy (WERCS) 18
   3.2. Synthetic Data Generation Approaches 19
      3.2.1. Introduction of Gaussian Noise (GN) 19

3.2.2. Synthetic Minority Oversampling Technique (SMOTE) 20
3.2.2.1. SMOTE for Regression (SmoteR) 20
3.2.2.2. SmoteR with Gaussian Noise (SMOGN) 21
3.2.2.3. Weighting SMOTE for Regression (WSMOTER) 22
3.2.2.4. Geometric SMOTE for Regression (G-SMOTER) 22
3.2.3. Deep Learning Approaches 23
3.2.3.1. DIRVAE 24
3.2.3.2. IRGAN 25
3.2.3.3. DAVID 25
3.2.4. Other Strategies 26
3.2.4.1. COVERT 26
3.2.4.2. Data Augmentation + Weighted Relevance (DA-WR) 26
3.2.4.3. k-Nearest Neighbour Oversampling - Regression (KNNOR-Reg) ... 27
3.3. Comparative Analysis 28
4. CART-based Data Augmentation for Imbalanced Regression 30
4.1. CART-based Synthetic Data Generation 30
4.2. Proposed Adaptation for Imbalanced Regression 32
5. Experimental Study 37
5.1. Experimental Setup 37
5.1.1. Datasets 37
5.1.2. Preprocessing Strategies 38
5.1.3. Learning Algorithms 39
5.1.4. Evaluation Metrics 40
5.2. Empirical Results 41
5.2.1. Target Variable Distribution Analysis 41
5.2.2. Error Performance Analysis 41
5.2.3. Runtime Performance Analysis 47
5.3. Discussion 48
6. Conclusion 50
6.1. Main Contributions 50
6.2. Limitations and Future Work 51

Bibliography 51
A. Comparative Analysis of Related Algorithms 64
B. Experimental Results 68
B.1. Data-Level Algorithms Variants and Hyperparameters 68
B.2. Aggregated Results 70
B.3. Wilcoxon Signed-Rank Test Results 76

B.4. Friedman/Post-Hoc Nemenyi Test Results and Critical Difference Diagrams.....87
B.5. Bayesian Signed-Rank Test Results.....90
B.6. Runtime Results 94
C. Implementation Details.....95

List of Tables

3.1. Comparison of Data-Level Algorithms by Key Design Characteristics. 29

5.1. Benchmark regression datasets used in the experiments 38
5.2. Data-level strategies and hyperparameters used in the experiments..... 39
5.3. Hyperparameter settings considered for the regression algorithms 40
5.4. Counts of best scores by preprocessing strategy / learner for each metric 43

A.1. Comparison of preprocessing strategies by key design characteristics 64
A.2. Detailed comparison of preprocessing strategies 65

B.1. Preprocessing strategy variants and respective hyperparameters..... 68
B.2. Best RMSE scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue. 70
B.3. Best RW-RMSE scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue. 71
B.4. Best SERA scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue. 72
B.5. Best DW-RMSE scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue. 73
B.6. Best DW-SERA scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue. 74
B.7. Best preprocessing strategy + learner combination per dataset and metric 75
B.8. RMSE results by (significant) wins/losses vs. baseline scenario 82
B.9. RW-RMSE results by (significant) wins/losses vs. baseline scenario 83
B.10. SERA results by (significant) wins/losses vs. baseline scenario..... 84
B.11. DW-RMSE results by (significant) wins/losses vs. baseline scenario 85
B.12. DW-SERA results by (significant) wins/losses vs. baseline scenario..... 86
B.13. Friedman test statistics per metric and learner, with statistically significant p-values $(p < 0.05)$ highlighted 87
B.14. Bayesian Signed-Rank Test: CARTGen-IR vs. WSMOTER 91
B.15. Runtime average (std. dev.) per preprocessing strategy and dataset 94

C.1. Code Metadata 96

List of Figures

2.1. Balanced and Imbalanced Domain Learning Taxonomy (adapted from [9]) 4
2.2. Importance of values for the target variable in the *housingBoston* dataset (adapted from [9]) 7
2.3. Distribution of a target variable (black) with its relevance function (blue) and the adjusted boxplot (adapted from [1, 42]) 9
2.4. DenseWeight Approach [39] 10
2.5. SERA metric for a model (adapted from [9]) 13

3.1. Taxonomy of main data-level approaches used in Imbalanced Regression 16
3.2. Distribution of a target variable before and after applying the RO strategy, indicated by the histogram and density curves, respectively, according to a relevance function $\phi()$ and a threshold of 0.8 18
3.3. Density of the target variable before and after applying the WERCS strategy, indicated by the histogram and density curves, respectively, according to a relevance function $\phi()$ 18
3.4. Distribution of a target variable before and after applying sampling strategies, indicated by the histogram and density curves, respectively, with a relevance threshold of 0.8 19
3.5. Distribution of a target variable before and after applying the GN strategy, indicated by the histogram and density curves, respectively, according to a relevance function $\phi()$ and a threshold of 0.8 20
3.6. Synthetic Generation Example of the SMOGN Algorithm [23] 21
3.7. Comparison of synthetic instance generation in SMOTER and G-SMOTER [68] 23
3.8. Distribution of a target variable before and after applying synthetic data generation strategies, indicated by the histogram and density curves, respectively, with a relevance threshold of 0.8 24
3.9. Distribution of a target variable before and after applying deep learning strategies, indicated by the histogram and density curves, respectively. 28

4.1. CART tree example (adapted from [83]) 30
4.2. Consecutive variable-wise data generation using CART 31
4.3. CARTGen-IR's data augmentation process flowchart 36

5.1. Distribution of the target variable before and after applying preprocessing strategies, indicated by the histogram and density curves, respectively. 42
5.2. Comparison of (significant) wins and losses of selected preprocessing strategies across all datasets, according to different metrics - RMSE, RW-RMSE and SERA 44
5.3. Critical Difference diagrams of all data-level strategies for a metric-learner pair, considering all datasets at 95% confidence level. 45
5.4. Bayesian Posterior Ternary Plots – CARTGen-IR vs. WSMOTER 47
5.5. Boxplot of runtimes per preprocessing strategy 48

B.1. Comparison of (significant) wins and losses of selected preprocessing strategies across all datasets, according to different metrics - RMSE, DW-RMSE and DW-SERA 76
B.2. RMSE results by (significant) wins/losses vs. baseline scenario 77
B.3. RW-RMSE results by (significant) wins/losses vs. baseline scenario 78
B.4. SERA results by (significant) wins/losses vs. baseline scenario 79

B.5. DW-RMSE results by (significant) wins/losses vs. baseline scenario....80
B.6. DW-SERA results by (significant) wins/losses vs. baseline scenario....81
B.7. Critical Difference diagrams for RMSE across learners....87
B.8. Critical Difference diagrams for RW-RMSE across learners....88
B.9. Critical Difference diagrams for SERA across learners....88
B.10. Critical Difference diagrams for DW-RMSE across learners....89
B.11. Critical Difference diagrams for DW-SERA across learners....89
B.12. Bayesian Posterior Ternary Plot for RMSE: CARTGen-IR vs. WSMOTER....90
B.13. Bayesian Posterior Ternary Plot for RW-RMSE: CARTGen-IR vs. WSMOTER....90
B.14. Bayesian Posterior Ternary Plot for SERA: CARTGen-IR vs. WSMOTER....90
B.15. Bayesian Posterior Ternary Plot for DW-RMSE: CARTGen-IR vs. WSMOTER....91
B.16. Bayesian Posterior Ternary Plot for DW-SERA: CARTGen-IR vs. WSMOTER....91
B.17. Bayesian Posterior Distributions for RMSE: CARTGen-IR vs. WSMOTER....92
B.18. Bayesian Posterior Distributions for RW-RMSE: CARTGen-IR vs. WSMOTER....92
B.19. Bayesian Posterior Distributions for SERA: CARTGen-IR vs. WSMOTER....92
B.20. Bayesian Posterior Distributions for DW-RMSE: CARTGen-IR vs. WSMOTER....92
B.21. Bayesian Posterior Distributions for DW-SERA: CARTGen-IR vs. WSMOTER....93

C.1. Scripts Interdependencies....95

List of Abbreviations

BN Bayesian Networks. 32

CART Classification and Regression Trees. ix, 2, 3, 30– 34, 50, 51

CARTGen-IR CART-based Generation for Imbalanced Regression. 32– 34, 38, 41– 51

CD Critical Difference. 45, 46, 48

CGAN Conditional Generative Adversarial Network. 26, 66

CTGAN Conditional Tabular Generative Adversarial Network. 32

CV Cross-Validation. 40, 41

DA Data Augmentation. 26

DA-WR Data Augmentation + Weighted Relevance. 38

DIRVAE Deep Imbalanced Regression Variational Autoencoder. 24, 25, 38

DistSMOGN Distributed SMOGN for Imbalanced Regression Problems. 22

DL Deep Learning. 16

DW-RMSE DenseWeight-RMSE. 40, 42, 43, 46, 47

DW-SERA DenseWeight-SERA. 40, 42, 43, 46

G-SMOTE Geometric SMOTE. 22

G-SMOTER Geometric SMOTE for Regression. 22, 23, 38, 41, 48

GAN Generative Adversarial Network. 25, 29, 31, 32, 66

GN Introduction of Gaussian Noise. ix, 19, 20, 23, 38, 50, 65, 66

HTTP Hypertext Transfer Protocol. 5

IR Imbalanced Regression. ix, 16, 30

IRGAN Imbalanced Regression Generative Adversarial Network. 25, 38

IRLSTM Imbalanced Regression Long Short-Term Memory. 24

k-NN k-Nearest Neighbour. 22, 26, 27, 67

KDE Kernel Density Estimation. 2, 9, 15, 26, 27, 33, 42, 65, 66

KNNOR k-Nearest Neighbour Oversampling. 27

KNNOR-Reg k-Nearest Neighbour Oversampling - Regression. 27, 28, 38, 41, 44, 47– 49

LSTM Long Short-Term Memory. 66

MAD Mean Absolute Deviation. 11

MAE Mean Absolute Error. 6, 11, 40

MSE Mean Squared Error. 4, 11, 40

NLP Natural Language Processing. 16

NSIR Negative Skewed Imbalanced Regression. 24

PSIR Positive Skewed Imbalanced Regression. 24

RF Random Forest. 27, 32, 39, 42, 43, 46

RMSE Root Mean Squared Error. ix, 12, 40, 41, 43, 44, 46, 48

RO Random Oversampling. ix, 17– 20, 38, 47

ROPE Region of Practical Equivalence. 46

RROC ROC space for regression. 11

RU Random Undersampling. 17, 19, 38, 42, 46, 47

RVAE Regression Variational Autoencoder. 24

RW-RMSE Relevance-Weighted RMSE. ix, 12, 40, 42– 46, 48

SER Squared Error-Relevance. 13

SERA Squared Error-Relevance Area. ix, 13, 40, 42– 46, 48, 50

SMOGN Synthetic Minority Oversampling with Gaussian Noise. ix, 21– 23, 33, 38, 50

SMOTE Synthetic Minority Oversampling Technique. 15, 19, 20, 26, 27, 31, 32, 67

SMOTER Synthetic Minority Oversampling Technique for Regression. 20– 23, 27, 33, 38, 42, 48, 50, 65, 66

SPD Synthetic Patient Data. 31

SSRF Server Side Request Forgery. 5

SVR Support Vector Regressor. 39, 43, 46, 47

UNIR Un-Normal Imbalanced Regression. 24

VAE Variational Autoencoder. 25, 26, 29, 66

WERCS WEighted Relevance-based Combination Strategy. ix, 18, 19, 33, 38, 44, 47, 50

WR Weighted Resampling. 27

WSMOTER WEighted SMOTE for Regression. 22, 23, 33, 38, 42, 44– 48

XGB XGBoost. 39

XXE XML External Entity. 5

1. Introduction

Imbalanced domain learning involves predictive modelling tasks characterised by two key aspects: a skewed distribution of the target variable and the need to accurately predict infrequent but often critical cases. While the majority of research in this area has focused on classification tasks with tabular data, particularly improving performance on minority classes [1], similar challenges arise in regression settings, where the target variable is continuous.

In many real-world domains, accurately forecasting rare or extreme values is essential. Examples include predicting wildfire risks [2–4], cybersecurity threats [5, 6], urban water consumption patterns [7], and financial frauds [8]. In such contexts, the primary concern is not general predictive accuracy, but rather the precise prediction of uncommon and impactful outcomes.

As in classification, standard regression methods face two main challenges in imbalanced domains [1]. First, they implicitly treat all target values as equally important. Second, commonly used evaluation metrics, such as Mean Squared Error (MSE) and Mean Absolute Error (MAE), tend to emphasise accuracy near the mean, often overlooking rare or extreme values. Moreover, the continuous nature of the target variable adds complexity, requiring models to identify important values across a range rather than among discrete categories.

Over the past two decades, the field of imbalanced domain learning has seen significant advances [1, 9–13]. Core areas of research include formalising the problem, recognising the limitations of standard learning algorithms, proposing tailored solutions, and refining evaluation metrics. Nonetheless, most contributions have concentrated on classification problems, especially binary classification, leaving imbalanced regression relatively underexplored.

1.1. Motivation

Among the strategies to handle imbalanced domain learning problems, data-level approaches are widely adopted. These techniques alter the training data to highlight important cases, typically during preprocessing. Their model-agnostic nature makes them

compatible with a variety of machine learning algorithms [1]. However, current data-level methods for regression often come with notable drawbacks: reliance on black-box models [14–16], the need for difficult-to-tune hyperparameters [17–21], and dependence on user-defined relevance thresholds that require discretising the continuous target space [17–19, 21].

This thesis proposes an adaptation of a previously introduced Classification and Regression Trees (CART)-based data augmentation technique [22] to make it suitable for imbalanced regression tasks. Furthermore, unlike most existing data-level techniques for imbalanced regression [17, 18, 23], our proposal does not depend on user-defined thresholds within the target variable domain, features an interpretable process regarding data generation due to the visual component of CART and avoids relying on hyperparameters difficult to input without domain knowledge.

1.2. Objectives

The main goals of this thesis are as follows:

1. provide a literature review of the problem as well as the challenges in the field, comparing existing state-of-the-art data-level strategies;
2. release a public repository of benchmark datasets for imbalanced regression;
3. propose a CART-based data augmentation method specifically adapted for imbalanced regression tasks that avoids the use of user-defined relevance thresholds;
4. implement new Kernel Density Estimation (KDE)-based evaluation metrics tailored to imbalanced regression tasks;
5. conduct a comprehensive experimental evaluation comparing state-of-the-art data-level strategies using benchmark datasets.

1.3. Bibliographic Notes

The following materials are associated with the main work presented in this thesis:

- A. P. Pinheiro, R. P. Ribeiro, ”CART-Based Synthetic Tabular Data Generation for Imbalanced Regression”, 40th Annual AAAI Conference on Artificial Intelligence *(under review)*

- A. P. Pinheiro and R. P. Ribeiro, “Synthetic Tabular Data Augmentation for Imbalanced Regression” [Source Code], DOI:10.24433/CO.7826905.v2

1.4. Organisation

The remainder of this thesis is structured as follows. Chapter 2 delineates the problem definition, the challenges in the field, and the application domains of an imbalanced regression task. Chapter 3 provides a concise overview of the pertinent literature, including existing sampling and data augmentation techniques. Chapter 4 presents our proposed solution, a CART-based data-level augmentation approach and its underlying mechanisms. Chapter 5 discusses the outcomes of an extensive experimental evaluation conducted with benchmark datasets, widely-used learning models, and sampling/augmentation strategies. Concluding remarks are provided in Chapter 6, as well as prospective future developments in the field.

2. Background

This chapter provides an overview of the main topics related to imbalanced regression, with real-world examples demonstrating the importance of addressing the problem. It also discusses the current challenges in this field of predictive modelling and reviews existing methods to tackle these problems. Additionally, the chapter examines various approaches aimed at addressing this research area.

2.1. Imbalanced Regression

Predictive modelling refers to the task of approximating an unknown function $Y = f(X)$ using supervised learning methods applied to a dataset composed of input variables X and a target variable Y [1]. When Y is continuous, the problem is formulated as regression, and models are trained to minimise a loss function, typically the Mean Squared Error (MSE), through parameter optimisation [1, 9].

In many practical settings, however, not all target values are equally important. Rare or extreme values often hold greater significance but are poorly represented in the data, giving rise to imbalanced regression problems [1]. These cases become especially challenging when combined with non-uniform domain preferences, where some regions of the target space are more relevant to the user than others. As illustrated in Figure 2.1, learning tasks can be categorised based on whether the target distribution is balanced or imbalanced, and whether domain preferences are uniform or not [9].

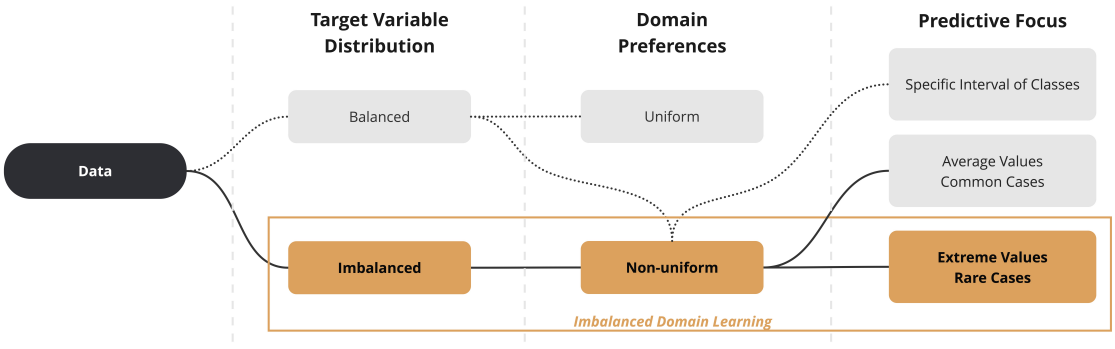


Figure 2.1: Balanced and Imbalanced Domain Learning Taxonomy (adapted from [9])

In the context of regression, imbalance is typically associated with the presence of rare and extreme values. Addressing this scenario involves two central challenges: guiding learning algorithms to focus on the most relevant instances, and adopting evaluation metrics that reflect model performance specifically on those rare cases [1]. Without

tackling these issues, models risk bias toward frequent values and fail to reflect the true goals of the application domain.

2.2. Application Domains

In many real-world imbalanced problems, the goal is to accurately predict extreme values of a continuous target variable that exhibits a skewed distribution. In this section, some areas of practical applications are presented involving problems whose configuration underscores the importance of the imbalanced regression task and highlights the rationale and importance for the contributions made in this study.

2.2.1 Cybersecurity

While the majority of machine learning tasks within the cybersecurity domain are classification-oriented, there are also important regression problems that involve imbalanced datasets. In the context of network intrusion detection, the authors of [24] constructed a dataset with features resembling real-world traffic, suitable for predicting risk scores or time-to-detection. Likewise, [25] demonstrated that a dataset developed by the Canadian Institute for Cybersecurity could support an intrusion detection system using variables like flow durations, packet counts, and other time-based statistics to estimate bandwidth usage, anomaly scores, or attack intensity. In addition, the authors of [26] show how the ISCX Botnet Dataset [27] can be applied to predict network congestion and perform anomaly scoring. Regression techniques are also used in other cybersecurity areas such as estimating the likelihood of an attack, or predicting network packet parameters and comparing them to normal behaviour.

In endpoint protection and application security, regression can help predict the number/sequence of system calls for executable processes and compare them to their real executions. It can also detect anomalies in Hypertext Transfer Protocol (HTTP) requests involving XML External Entity (XXE) injection and Server Side Request Forgery (SSRF) attacks, as well as abnormal user activity like unusual login times [5, 6].

2.2.2 Environment and Meteorology

Environmental and meteorological domains present clear cases of imbalanced regression problems, where rare but impactful events must be predicted from limited data. As noted by [28], extreme weather events, such as heavy rainfall exceeding 25 mm/h, are critical to forecast but occur infrequently, sometimes fewer than ten times per decade

per location. Similarly, wildfire prediction faces the challenge of scarce training data for high-risk events, influenced by many environmental and human factors [29–31]. Studies such as [2–4] emphasise the rarity of data representing severe wildfire conditions, like those from Portugal's 2017 fires, and advocate for data augmentation to support model training [2]. Additionally, in air pollution modelling, limited occurrences of dangerously high pollutant levels, such as concentrations above 150 μg/m³, pose similar challenges [32].

2.2.3 Finance

The application of regression techniques in financial market forecasting is inherently tied to challenges posed by imbalanced regression problems. Financial datasets often exhibit characteristics such as high nonlinearity, noise, and dynamic dependencies, with very sparse data as indicated in the research done by [8]. These traits underscore the importance of addressing imbalances in regression modelling. Additionally, in financial contexts, extreme market movements (e.g., crashes or surges) may be rare but carry disproportionate importance for stakeholders. The need to accurately predict these underrepresented outcomes is paramount for practical applications such as risk management and portfolio optimisation. The unique properties of financial data require innovative approaches to model design and evaluation, ensuring that the models are robust against the skewed distribution of outcomes and capable of delivering actionable insights [8].

2.2.4 Utilities

Electricity load and urban water consumption forecasting are key examples of regression tasks with substantial operational and economic impact. Accurate electricity load forecasts support critical decisions in energy market operations and can yield major cost savings, up to USD$1.6 million annually from just a 1% reduction in the Mean Absolute Error (MAE) for a 10,000 MW generator [33, 34]. Similarly, forecasting water demand is vital for effective resource distribution, pricing, and system management [7].

In both cases, the challenge intensifies during peak or unusual demand periods, which are rare and lack sufficient historical data. This scarcity leads to imbalanced regression problems, where high-impact instances are underrepresented and harder to predict accurately [9].

2.3. Challenges in Imbalanced Regression

Imbalanced regression presents three main challenges, as outlined by [9]. The first is how to formally represent non-uniform preferences over continuous target domains. Figure 2.2 contrasts these two views: the dashed green line reflects uniform preferences, while the solid black line illustrates a preference for extremes.

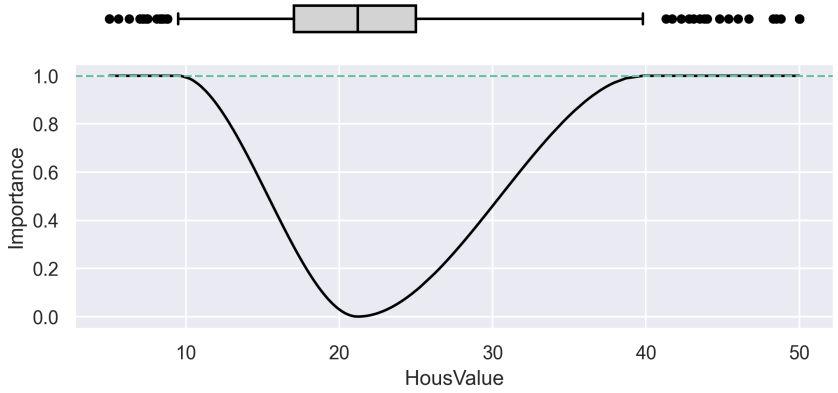


Figure 2.2: Importance of values for the target variable in the *housingBoston* dataset (adapted from [9])

In contrast to classification, where it is typically easy to define a specific class of interest labeled as positive, regression involves expressing preferences over a potentially infinite range of target values. This complexity requires some form of automated approximation for preference specifications. However, as highlighted by many studies [9, 35–37], these methods must not rely on fixed assumptions, such as normality.

The second challenge involves developing evaluation criteria that prioritise predictive performance on rare but important cases without introducing significant bias. Traditional regression metrics focus on average performance and fail to reflect the importance of extreme values [9]. Alternative metrics often either ignore preference variation or overemphasise extremes, reducing generalisability [38].

The third challenge, as referred in [9], relates to the modelling phase, specifically to bias the learning algorithms to train models that follow the mentioned metrics as preference criteria.

2.3.1 Target Variable Preferences Specification

Translating the domain knowledge to user domain preferences plays a pivotal role. It influences both the evaluation and comparison of alternative models, as well as the tailoring of the learning process to construct models that effectively meet and prioritise these

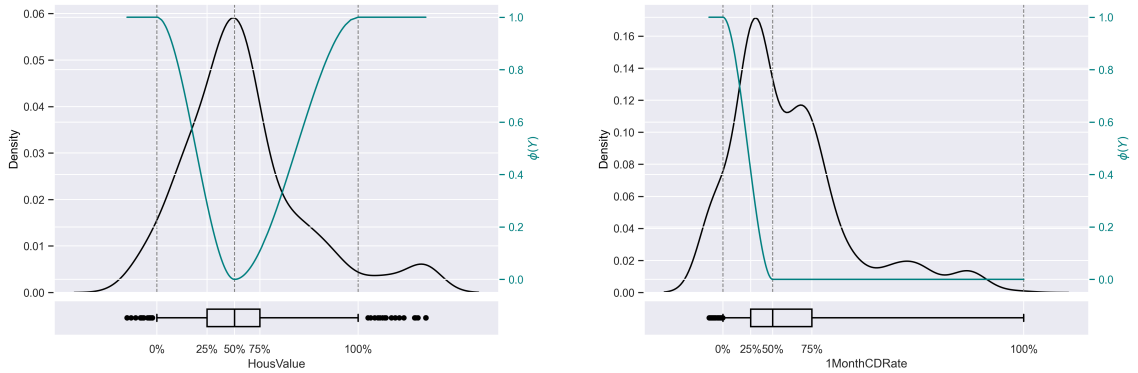
preferences, ensuring they are as aligned as possible with user-defined objectives [9]. Furthermore, possessing the complete specification of the domain knowledge represents the ideal scenario. However, such information is often challenging to acquire in practical applications, particularly in regression tasks where the target variable possesses an infinite domain. Among the numerous hypotheses and approaches to addressing this challenge, this subsection presents two approaches formalised by [9] and [39], respectively.

2.3.1.1 Relevance Function

To address imbalanced regression in the absence of domain knowledge, [9] propose a non-parametric method for automatically defining a relevance function, building on earlier work by [40]. A relevance function $\phi(Y) : Y \rightarrow [0, 1]$ quantifies the importance of accurately predicting each target value within a continuous domain [9, 40].

Given the impracticality of manually specifying relevance across an infinite target space, the method approximates the function using a set of control points - target values with predefined relevance values and respective derivatives - combined with interpolation [9]. To automate the generation of control points, the approach identifies extreme values using adjusted boxplots, which account for skewness by incorporating the medcouple statistic [41]. Extreme target values are assigned high relevance, while frequent values are assigned low relevance [9].

As illustrated in Figure 2.3, the resulting relevance function depends on the underlying distribution of the target variable. If the data sample is representative of the application domain, the generated function aligns closely with relevance functions informed by expert knowledge. Otherwise, it reflects distribution-specific priorities [9].



(a) *housingBoston* dataset (b) *treasury* dataset

Figure 2.3: Distribution of a target variable (black) with its relevance function (blue) and the adjusted boxplot (adapted from [1, 42])

Interpolation is performed using Piecewise Cubic Hermite Interpolating Polynomials (pchip) [43], which ensures smooth and shape-preserving curves [9]. This method maintains key mathematical properties, such as monotonicity, local positivity, and convexity, and applies linear extrapolation outside the range defined by the control points [9]. Overall, this approach enables a flexible and robust specification of non-uniform relevance in imbalanced regression settings, without requiring prior expert knowledge of the domain.

2.3.1.2 DenseWeight

The DenseWeight method, introduced by [39], addresses domain preference in imbalanced regression by assigning sample weights inversely proportional to the estimated density of the target variable. This ensures that rarer outcomes receive higher importance during model training. While conceptually related to relevance functions, DenseWeight uses KDE rather than heuristics such as boxplot statistics [9].

The weighting function $f_w(\alpha, y)$ depends on a tunable hyperparameter $\alpha \in [0, \infty[$, which controls the sensitivity to target rarity. When $\alpha = 0$, weights are uniform; as α increases, rare values receive greater weight. To maintain stability, all weights are positive, strictly non-zero, and normalised to have a mean of 1 [39].

Density Estimation via KDE: Rarity is estimated through KDE, which provides a smooth approximation of the target distribution. Using a Gaussian kernel and Silverman's bandwidth rule [44], the estimated density $p(y)$ is normalised to the $[0, 1]$ range, assigning lower values to rare targets [39].

Weighting Function: The final weight for each target value is computed in three steps:

1. Initial weighting:

$$f'_w(\alpha, y) = 1 - \alpha \cdot p'(y) \tag{2.1}$$

2. Clipping to prevent zero or negative weights:

$$f''_w(\alpha, y) = \max(1 - \alpha \cdot p'(y), \epsilon) \tag{2.2}$$

3. Normalisation across the dataset:

$$f_w(\alpha, y) = \frac{f''_w(\alpha, y)}{\frac{1}{N}\sum_{i=1}^{N} f''_w(\alpha, y_i)} \tag{2.3}$$

As [39] show, higher α values amplify the contrast between rare and common instances, allowing flexible control over how much emphasis is placed on underrepresented targets. However, the optimal value for α is task-specific and typically requires empirical tuning.

DenseWeight is compatible with data-level preprocessing, relevant to this thesis, as well as with learning algorithms that support weighted loss functions [39], as is illustrated in Figure 2.4.

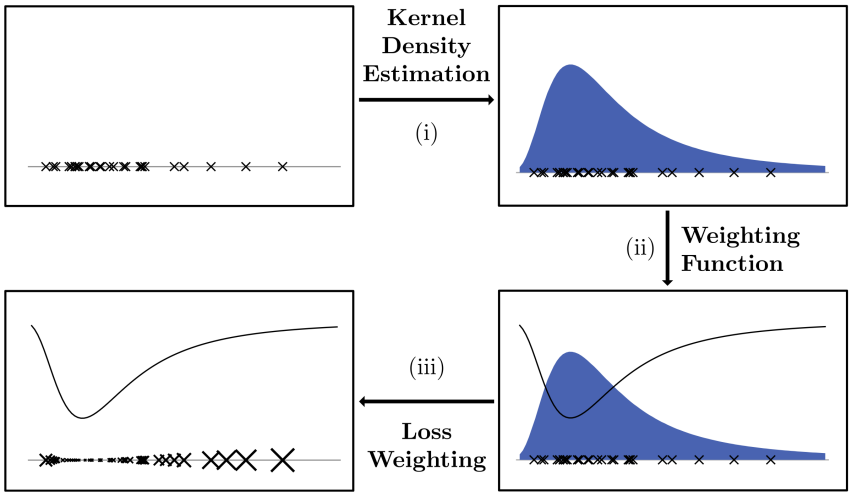


Figure 2.4: DenseWeight Approach [39]

2.3.2 Evaluation Metrics

Imbalanced domains arise when the most critical target values, from the user's perspective, are poorly represented in the training data, creating a major challenge: standard evaluation metrics become ineffective, as they favour average performance, dominated by common cases [1].

This issue is particularly evident in regression tasks, where widely used error metrics such as the MSE and MAE, also referred to as Mean Absolute Deviation (MAD), are the standard criteria (cf. Equations 2.4 and 2.5) [9].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{2.4}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{2.5}$$

These metrics assume uniform domain preferences and focus on minimising the average error, naturally favouring frequent values concentrated around the mean. As a result, model performance is often biased toward these central values, with rare or extreme cases being neglected [9, 16, 45]. This can lead to misleading conclusions when working with skewed distributions [46, 47]. For instance, in the distribution shown in Figure 2.3, traditional metrics would favour accurate predictions in high-density, low-value regions, while neglecting rare but important upper-end values.

Therefore, the choice of evaluation metrics is critical when addressing data imbalance. Metrics must align with the user's objectives and help guide the learning process by emphasising errors in the most relevant regions of the target domain [1, 48]. Several alternative evaluation metrics have been proposed to address the limitations of standard error measures in regression. These include asymmetric loss functions, differentiating between under- and over-predictions [38, 49–53] and graphical tools like the ROC space for regression (RROC) curve [54], which adapt ideas from classification to regression. However, such approaches often remain inadequate for imbalanced domains, as they only enable the identification of certain error types or those occurring within specific ranges of the true target variable, thus overlooking the varying importance of prediction errors across different target value ranges [1, 9, 55].

This section reviews the key performance evaluation metrics specifically created for imbalanced regression that consider non-uniform domain preferences.

2.3.2.1 Threshold Dependent

To overcome the limitations of traditional metrics in imbalanced regression, the authors of [40, 56] proposed a utility-based evaluation framework. At its core, this approach defines a utility function $u(\hat{y}, y)$ that expresses the usefulness of a prediction in terms of both numeric accuracy and the relevance of the predicted and true values. The function assumes both positive and negative values, reflecting a balance between prediction benefits and associated costs [17, 40].

Building on this formulation, several evaluation metrics were adapted from classification to regression. Definitions for true positives, false positives, false negatives, precision, and recall were introduced in the context of regression using this utility perspective [18]. Furthermore, combined performance metrics such as the utility-based F_1-score and an adapted G-Mean were also proposed to assess models under imbalanced conditions [17]. These contributions provide a principled way to evaluate models with an emphasis on both rare, high-relevance predictions and more common cases.

However, this framework depends on a user-defined relevance threshold to identify relevant cases. This results in sharp and artificial boundaries, which do not match the natural continuous nature of the target variable and limit the generalisation ability of the models [9, 17].

2.3.2.2 Relevance-Aware

One immediate alternative to overcome the limitations of earlier evaluation metrics in imbalanced regression is Relevance-Weighted RMSE (RW-RMSE) [57] (cf. Eq. 2.6) was defined. It enhances Root Mean Squared Error (RMSE) by incorporating a relevance function $\phi()$ directly into the error calculation, thereby capturing not only the magnitude but also the contextual significance of prediction errors.

$$RW - RMSE = \sqrt{\frac{\sum_{i=1}^{n} \phi(y_i) \cdot (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} \phi(y_i)}} \tag{2.6}$$

However, this metric has the drawback of ignoring performance on more common values, which have near-zero relevance scores and tend to favour models that are heavily

biased toward rare values, which have maximum relevance scores. This can lead to selecting models with poor generalisation ability.

To address the limitations of earlier evaluation metrics in imbalanced regression, [9] proposed a new metric: the Squared Error-Relevance Area (SERA), designed to assess a model's ability to predict extreme values while penalising substantial bias. It builds on the concept of relevance functions from [40] and avoids the need for arbitrary thresholds.

The metric is based on the Squared Error-Relevance (SER), which calculates the total squared error over a subset of data with relevance above a given cutoff t:

$$SER_t = \sum_{i \in D_t} (\hat{y}_i - y_i)^2, \tag{2.7}$$

where $D_t = \{\langle x_i, y_i \rangle \in D \mid \phi(y_i) \geq t\}$ and $\phi(y) \in [0,1]$ is the relevance function. By computing SER_t across all possible thresholds $t \in [0,1]$, a monotonic decreasing curve is formed, from which the SERA metric is derived (cf. Equation 2.8), as illustrated in Figure 2.5:

$$SERA = \int_0^1 \sum_{i \in \mathcal{D}_t} (\hat{y}_i - y_i)^2 \, dt \tag{2.8}$$

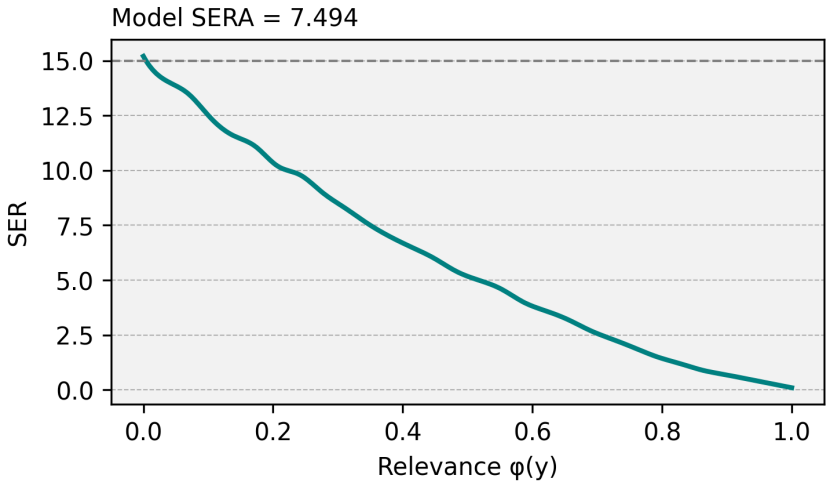


Figure 2.5: SERA metric for a model (adapted from [9])

This approach gives greater weight to errors on high-relevance (extreme) cases, while still incorporating model performance on average values. It enables fairer evaluation in skewed domains and supports model comparison without relying on hard relevance thresholds, unlike previous utility-based methods [9, 56]. Additionally, SERA allows

asymmetric, non-uniform error weighting through relevance functions, without assuming distribution symmetry, offering a more robust alternative to traditional metrics [9].

2.3.3 Modelling Approaches

To address the inadequacy of the models produced regarding imbalanced regression problems, different types of solutions have been introduced in the literature. These approaches can be categorised into three types of strategies: (i) preprocessing adjustments to the data before training, (ii) modifications to the learning algorithms, and (iii) hybrid strategies that integrate multiple approaches [1, 58].

2.3.3.1 Data-Level Approaches

Data-level methods are among the most widely used techniques in machine learning [1]. These methods operate at the preprocessing stage, modifying the training data distribution to emphasise cases that are most relevant to the user. They typically follow two strategies: adjusting the distribution of examples to better represent important cases (distribution change), or weighting the data space according to prediction costs to minimise high-impact errors [58]. Common approaches include undersampling, oversampling, the generation of synthetic examples, or combinations of these techniques.

Data-level methods offer flexible, model-agnostic preprocessing that aligns learning with user-defined goals [58]. However, mapping those goals to a suitable data distribution is challenging [1], and techniques like undersampling, oversampling, and synthetic generation each involve trade-offs, particularly in high-dimensional settings [58].

This thesis will explore data-level methods in detail, including a comprehensive review of existing algorithms and the proposal of a new technique.

2.3.3.2 Algorithm-level Approaches

Algorithm-level methods tackle imbalanced domains by modifying the learning process itself. Two main strategies have been explored: cost-sensitive learning, which assigns different costs to instances to steer the optimisation toward high-cost cases [1], and loss function modification, which adapts the error criterion to reflect data imbalance [58].

While these approaches can yield models that align more directly with user goals and offer improved interpretability [59], they require in-depth knowledge of both the learning algorithm and user preferences. Defining appropriate utility functions and implementing custom loss functions can be technically demanding and may reduce flexibility in choosing algorithms [1].

In regression, progress in cost-sensitive methods has been limited. A notable recent development is DenseLoss [39], which combines KDE-based rarity estimation with a custom loss function parameterised by α, allowing continuous-valued cost sensitivity to be integrated into model training [39, 58].

2.3.3.3 Hybrid Approaches

Hybrid methods integrate techniques from multiple strategies, combining data-level and algorithm-level methods to leverage their strengths and achieve superior overall performance. [58] In recent years, this approach has gained increasing popularity, with several methods that combine elements from the fundamental approaches discussed earlier emerging in the research literature. [1]

Although the majority of research has focused on classification tasks [58], a notable example of such an approach is SMOTEBoost for regression [60], an adaptation of the algorithm of SMOTEBoost developed for classification. It comprises four variants, each differing from the others while sharing a common baseline: the combination of the data preprocessing approach known as Synthetic Minority Oversampling Technique (SMOTE) with a boosting algorithm.

3. Related Work

Numerous data, algorithm, and prediction-level strategies have been proposed to address imbalanced data in Imbalanced Regression (IR). Within this topic there are two main groups of research lines: IR, i.e. modelling the probability distributions of the dependent variable in structured tabular datasets, and Deep IR, generalising it to unstructured data domains like images and Natural Language Processing (NLP) using Deep Learning (DL). This thesis focuses on the former, namely data-level preprocessing strategies for structured tabular data.This chapter provides an overview and analysis of existing methods for handling imbalanced tabular data in regression, highlighting their advantages, disadvantages, and underlying mechanisms. Figure 3.1 presents a taxonomy of existing data-level strategies for IR that will be discussed throughout the chapter.

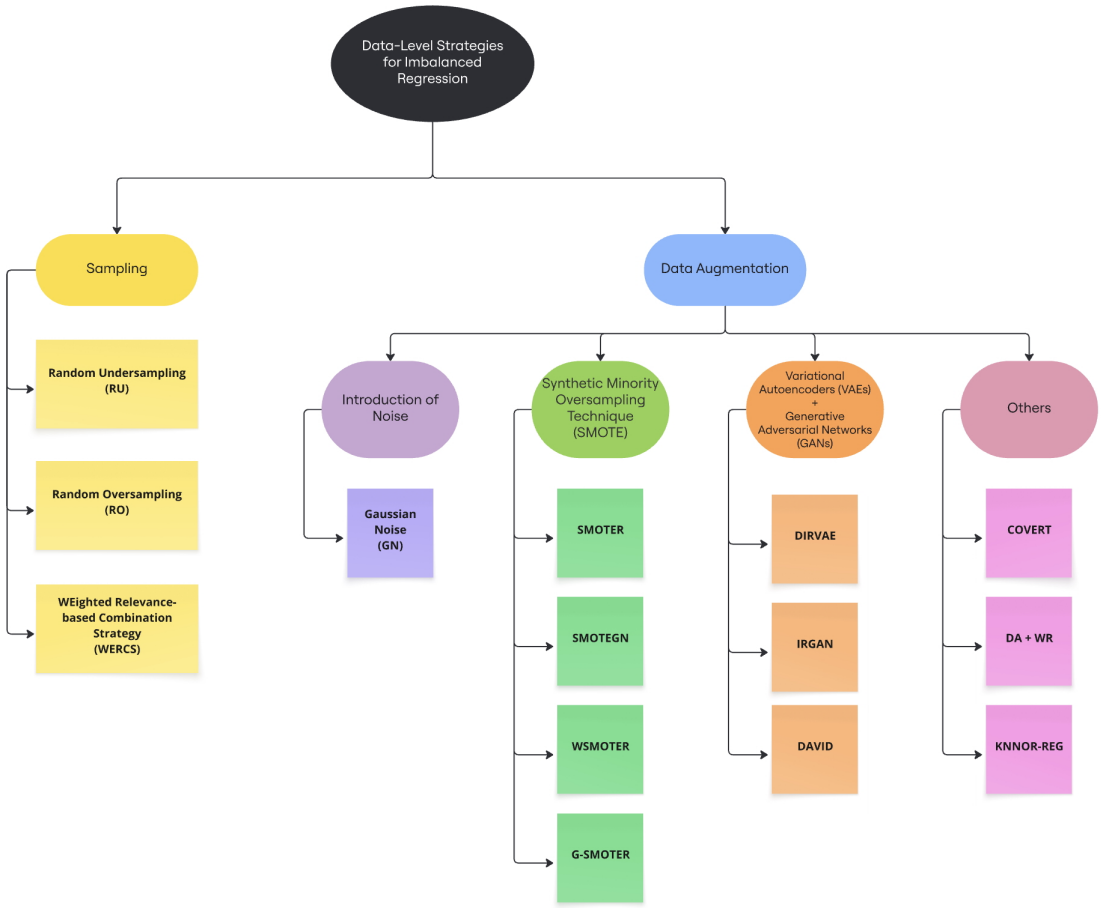


Figure 3.1: Taxonomy of main data-level approaches used in Imbalanced Regression

3.1. Sampling-based Approaches

Three basic approaches rely on random sampling guided by the relevance of the target values: undersampling, oversampling, and a combination of the two.

3.1.1 Random Undersampling (RU)

The Random Undersampling (RU) method, proposed by [18] reduces low-relevance observation frequency by using a relevance function and a user-defined threshold to rebalance regression datasets. It retains all high-relevance cases and randomly samples from the rest, based on a defined ratio n_u [17, 18].

Its main advantage is simplicity and reduced computational cost, while still improving performance over using imbalanced data [18, 61]. However, it risks discarding useful data, relies on an arbitrary threshold, and may distort the data distribution near relevance boundaries [17, 18, 62].

3.1.2 Random Oversampling (RO)

The algorithm proposed by [17] adapts the traditional Random Oversampling (RO) technique, commonly used in classification, for imbalanced regression. Instead of working with class labels, it uses a relevance function [9, 40] to discretise the continuous target variable into bins. These bins are split into relevant ($Bins_R$) and normal ($Bins_N$) ranges based on a user-defined relevance threshold (t_R). Relevant bins contain rare, important cases and are oversampled; normal bins contain common cases and are undersampled.

After binning, the algorithm duplicates the relevant examples according to a user-defined percentage. Two automatic strategies for setting this percentage are included: “balance”, which equalises the size of each bin, and “extreme”, which increases the number of rare cases beyond that of common ones [17].

The main advantage of this approach is that it preserves all original data while boosting rare cases [17]. However, it can significantly increase dataset size, leading to higher computational costs and a greater risk of overfitting [17], especially when many duplicates are added. The effects of this algorithm are shown in Figure 3.2, which compares the target variable distribution before and after applying different RO settings.

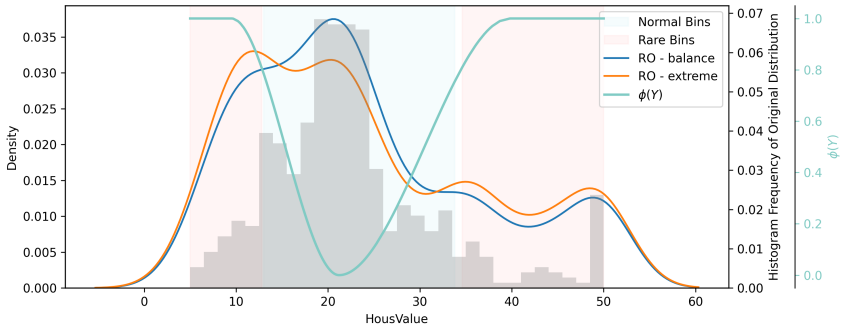


Figure 3.2: Distribution of a target variable before and after applying the RO strategy, indicated by the histogram and density curves, respectively, according to a relevance function $\phi()$ and a threshold of 0.8

3.1.3 WEighted Relevance-based Combination Strategy (WERCS)

The WEighted Relevance-based Combination Strategy (WERCS) algorithm, proposed by [17], addresses imbalanced regression by guiding under- and oversampling using the continuous nature of the relevance function [9, 40], rather than relying on predefined thresholds or bins. It assigns replication probabilities proportional to each instance's relevance and removal probabilities based on its complement. Thus, highly relevant instances are more likely to be duplicated, while low-relevance ones are more likely to be removed [17].

This probabilistic sampling avoids abrupt distinctions between similar cases, unlike threshold-based methods, and requires only two parameters: the proportions of over- and undersampling, $\%o$ and $\%u$, respectively. The effect of adjusting these parameters is shown in Figure 3.3, which illustrates how the distribution of the target variable evolves after applying WERCS.

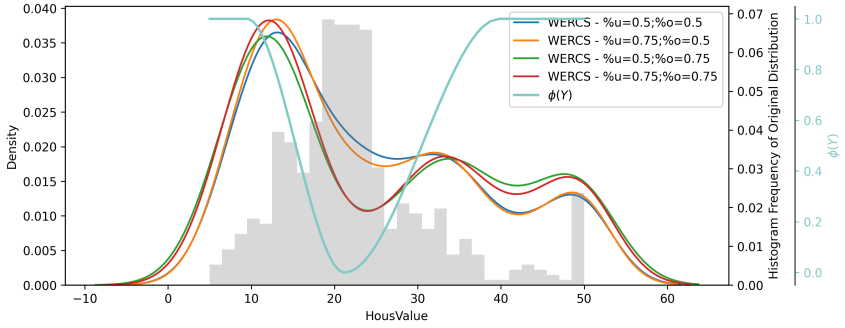


Figure 3.3: Density of the target variable before and after applying the WERCS strategy, indicated by the histogram and density curves, respectively, according to a relevance function $\phi()$

The main advantages of WERCS are its flexibility, its respect for the continuous

nature of the target variable, and its ability to treat similar instances accordingly [17]. However, it also has limitations: it relies on simple duplication without noise or neighbourhood structure, which may lead to overfitting [17]. Additionally, the choice of sampling proportions (%u and %o) is arbitrary and may affect the final dataset quality. Finally, some information may be lost during undersampling [17].

Figure 3.4 illustrates how the sampling strategies RU, RO and WERCS modify the distribution of a target variable.

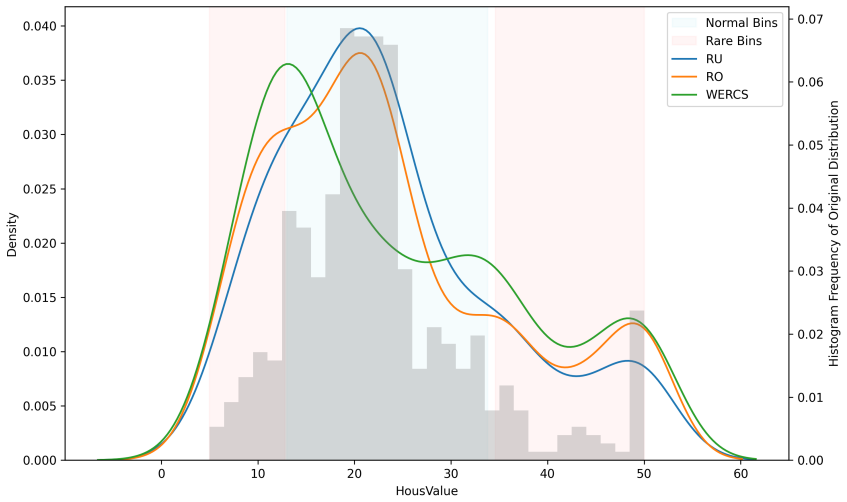


Figure 3.4: Distribution of a target variable before and after applying sampling strategies, indicated by the histogram and density curves, respectively, with a relevance threshold of 0.8

3.2. Synthetic Data Generation Approaches

To avoid the overfitting that can result from oversampling, three main data augmentation approaches have been proposed for generating new synthetic data in an imbalanced regression scenario. These approaches include the use of Gaussian noise, SMOTE-based methods, and deep learning techniques.

3.2.1 Introduction of Gaussian Noise (GN)

The Introduction of Gaussian Noise (GN) strategy, proposed by [17], enhances imbalanced regression datasets by undersampling low-relevance cases and generating new high-relevance ones. Like the RO method, it uses binning based on a relevance threshold to identify which cases to reduce or augment. For oversampling, it adds Gaussian noise (δ) to numeric attributes and target values of relevant cases, and samples nominal attributes based on their frequency [17, 63].

The number of synthetic cases per bin is, like in RO, automatically set using “balance” or “extreme” options. The impact of this noise-based generation is illustrated in Figure 3.5, which shows how the target variable distribution is reshaped after applying the method.

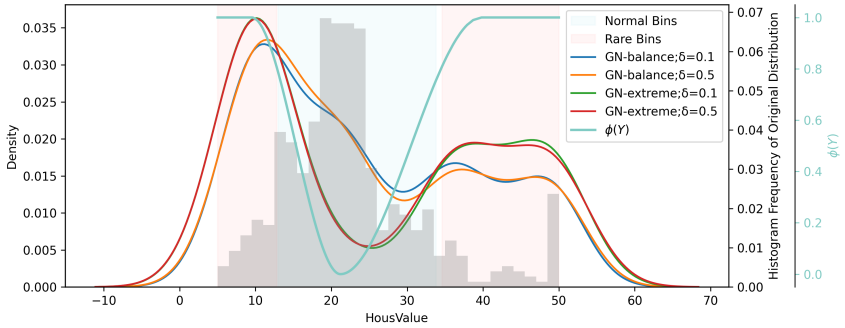


Figure 3.5: Distribution of a target variable before and after applying the GN strategy, indicated by the histogram and density curves, respectively, according to a relevance function $\phi()$ and a threshold of 0.8

This method generates more diverse synthetic examples while maintaining relevance, and simplifies usage through automated parameter definition. However, it still relies on a user-defined relevance threshold for binning, which may affect consistency and introduce discretisation issues [17].

3.2.2 Synthetic Minority Oversampling Technique (SMOTE)

Building on SMOTE [64], a well-known strategy used in imbalanced classification tasks, four data-level strategies have been proposed to tackle imbalanced regression.

3.2.2.1 SMOTE for Regression (SmoteR)

The Synthetic Minority Oversampling Technique for Regression (SMOTER) algorithm, introduced by [65], adapts the original SMOTE method for regression by focusing on improving predictions for rare target values. It does so by first identifying relevant cases using a relevance function and threshold [9, 40], then oversampling those cases and undersampling the less important ones [18, 61]. Synthetic samples are generated from a set of k nearest neighbours, by interpolating numeric attributes and randomly selecting nominal ones. The target value is computed as a weighted average based on the distance to each seed instance.

One of its strengths is its versatility, as it handles both numeric and nominal features and supports generating rare cases at both ends of the target distribution [65]. However, it has notable limitations: it relies on a user-defined relevance threshold, which can introduce inconsistencies [18], and its interpolation process may create unrealistic or noisy synthetic cases, especially in high-dimensional settings [66]. Additionally, setting the right number of neighbours k is non-trivial, and the combination of undersampling and oversampling may lead to data loss or overfitting [18, 61].

3.2.2.2 SmoteR with Gaussian Noise (SMOGN)

The Synthetic Minority Oversampling with Gaussian Noise (SMOGN) algorithm, proposed by [23], is a hybrid oversampling technique for imbalanced regression that combines SMOTER-based interpolation with controlled Gaussian noise injection. It aims to improve generalisation and data diversity while reducing the risk of generating unrealistic synthetic points.

The method first partitions the data based on a relevance function into rare $Bins_R$ and normal $Bins_N$ cases using a relevance threshold. It then applies random undersampling to normal bins and selectively oversamples relevant bins. If a seed and its neighbour are close in the feature space, SMOTER is used; otherwise, Gaussian noise (δ) is applied. This decision is based on whether the distance to the neighbour is less than half the median distance within the bin. Figure 3.6 shows an example of this neighbour selection logic, for two attributes X1 and X2 and target variable Tgt, where cases marked with a cross are normal, common instances and ones highlighted with bullet points represent relevant samples.

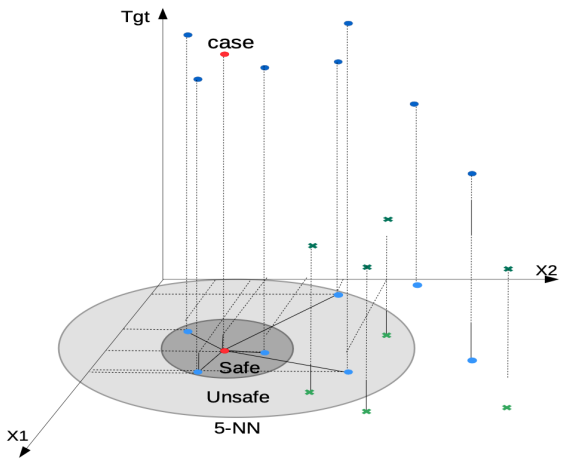


Figure 3.6: Synthetic Generation Example of the SMOGN Algorithm [23]

Advantages of SMOGN include better control over overfitting and improved synthetic diversity [23] by dynamically switching between interpolation and noise-based generation. It balances fidelity and variability, making it effective in capturing rare patterns. However, it still relies on a relevance threshold for binning and requires tuning of k for the k-Nearest Neighbour (k-NN) search, which can complicate optimisation [9, 23].

Notably, SMOGN also inspired Distributed SMOGN for Imbalanced Regression Problems (DistSMOGN) [67], a distributed version designed for large-scale datasets, extending its utility to big data scenarios.

3.2.2.3 Weighting SMOTE for Regression (WSMOTER)

The WEighted SMOTE for Regression (WSMOTER) algorithm, introduced by [20], extends SMOTER by incorporating a rarity-based weighting system using the DenseWeight method [39] to quantify instance rarity. Instances are assigned selection probabilities based on the sparsity of their target values, with rarer cases being more likely to be chosen. Once a rare instance is selected as seed, a second instance is chosen through a directional search by shifting β positions right or left, depending on if the first seed's target value is above or below the median, respectively, in a sorted target value list and selects a neighbouring instance within that range, and then generating a synthetic instance using interpolation, as in SMOTER [20].

Its main advantages include removing rigid relevance thresholds, enabling a continuous and density-aware sampling process, and focusing generation on meaningful sparse regions [20, 39]. This results in better coverage of rare cases without assuming predefined partitions, improving flexibility over SMOTER [18]. However, it requires manual specification of the parameter k, increases dataset size and training time, and may still risk overfitting by relying on interpolation of existing data [20].

3.2.2.4 Geometric SMOTE for Regression (G-SMOTER)

The Geometric SMOTE for Regression (G-SMOTER) algorithm, proposed by [68], adapts the Geometric SMOTE (G-SMOTE) [69] method for regression by generating synthetic data within a geometrically defined region rather than along a straight line, as done in SMOTER. This region is formed by transforming a hypersphere around selected instances using a truncation and deformation factor - θ and ξ, respectively - allowing for

greater diversity and reducing the risk of creating near-duplicate points. Figure 3.7 illustrates the key difference in synthetic data generation between SMOTER and G-SMOTER.

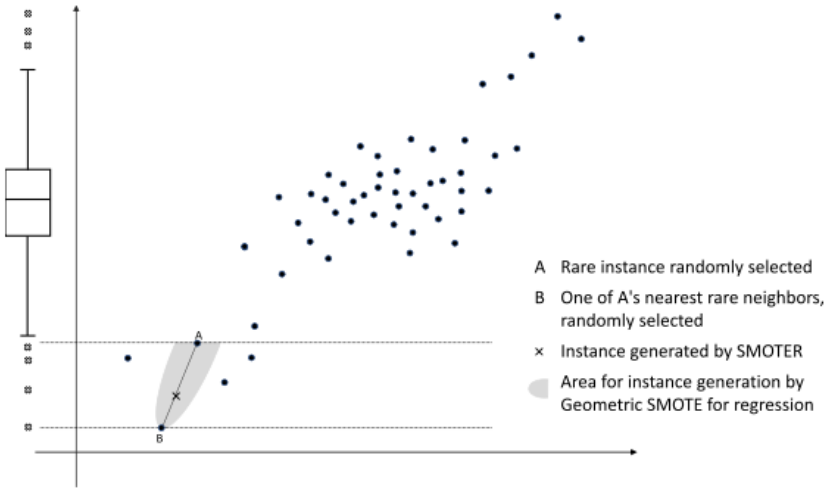


Figure 3.7: Comparison of synthetic instance generation in SMOTER and G-SMOTER [68]

Instance selection for oversampling can follow one of three strategies: minority, majority, or combined, based on relevance values derived from a relevance function [9, 40]. The target value of each synthetic point is a weighted average influenced by the geometric closeness to its source instances. The augmentation ratio is expressed parameter η.

Advantages of G-SMOTER include increased diversity in the generated data, reduced overfitting risk, and support for custom relevance functions, making it flexible and generalisable across regression tasks [20, 68]. However, it still depends on a user-defined relevance threshold to identify rare cases, which may introduce subjectivity and affect reproducibility [68].

Figure 3.8 illustrates how the synthetic data generation strategies GN, SMOTER, SMOGN, WSMOTER and G-SMOTER modify the distribution of a target variable.

3.2.3 Deep Learning Approaches

In deep learning, some recent approaches have been proposed to address imbalanced regression tasks at the data level. These primarily include adaptations of Variational Autoencoders and Generative Adversarial Networks.

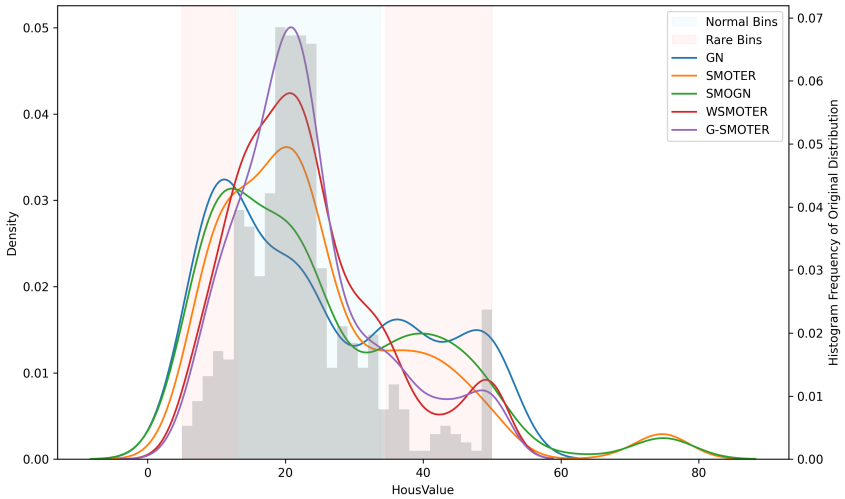


Figure 3.8: Distribution of a target variable before and after applying synthetic data generation strategies, indicated by the histogram and density curves, respectively, with a relevance threshold of 0.8

3.2.3.1 DIRVAE

The Deep Imbalanced Regression Variational Autoencoder (DIRVAE) algorithm, introduced by [14], is a generative method designed to synthesise underrepresented and missing samples in imbalanced regression tasks. It follows an adversarial learning framework composed of a generator and a discriminator, both of which are trained in a minimax game to achieve a Nash equilibrium. The generator is specifically built to create realistic regression samples aligned with predefined target values, using a dual-model structure that combines global and local information from the original data [14].

To better address different imbalance patterns, DIRVAE categorises the target value distribution into three fundamental modes: Positive Skewed Imbalanced Regression (PSIR), Un-Normal Imbalanced Regression (UNIR), and Negative Skewed Imbalanced Regression (NSIR), each corresponding to a different type of skewness in the distribution [14]. These modes serve as simplified representations of complex imbalanced regression distributions and guide the sample generation process.

The generator consists of two components. The first is an Regression Variational Autoencoder (RVAE), which encodes original samples into a latent space and decodes them back, ensuring global consistency with the original distribution [14]. The second is an Imbalanced Regression Long Short-Term Memory (IRLSTM), which incorporates nearest-neighbour information to learn local relationships between adjacent target values. These two intermediate outputs are merged via a fully connected layer to produce the

final synthetic sample. This architecture allows DIRVAE to model both the overall data structure and finer-grained local patterns [14].

This approach effectively captures spatial distribution characteristics and enhances generalisation by mapping data to distributions rather than fixed points [14, 70]. However, a key limitation is its reliance on specific distribution shapes - such as skewed or Gaussian distributions - which may limit its applicability to more diverse real-world regression problems [71].

3.2.3.2 IRGAN

The Imbalanced Regression Generative Adversarial Network (IRGAN) algorithm proposed by Liu et al. [15] addresses the problem of imbalanced regression by generating synthetic samples for underrepresented regions of the target variable. It combines optimisation techniques with adversarial learning, relying on a framework composed of multiple sample pools, including original, real, fake, and balanced datasets [15]. The generator produces synthetic data by injecting Gaussian noise and optimising a composite loss function that includes both adversarial objectives with the JS divergence metric and the Mahalanobis distance, which is particularly effective in high-dimensional, correlated feature spaces [15].

To stabilise the training process, especially in early stages, a correction module is used to align generated samples with realistic transitions, guiding the generator through reinforcement learning strategies [15]. Meanwhile, a discriminator based on convolutional neural networks differentiates between real and generated samples, helping to refine the quality of the synthetic data. The entire adversarial system evolves toward a Nash equilibrium, producing a more balanced dataset for regression [15].

Despite its strengths, such as robustness to complex feature distributions and the inclusion of corrective mechanisms, IRGAN is still a black-box model and remains vulnerable to the vanishing gradient problem common in Generative Adversarial Network (GAN)-based architectures [15].

3.2.3.3 DAVID

The DAVID algorithm, proposed by [16], generates synthetic data through a two-phase process in the latent space of a Variational Autoencoder (VAE). First, a seed is drawn from the mean representation μ_i of a rare observation, as encoded by the VAE.

Then, a new latent point z^* is generated using a smoothed bootstrap applied to the matrix of mean latent representations. This bootstrap process involves selecting a seed with a given weight and adding noise from a kernel density estimator, typically based on multivariate kernels such as the Gaussian and a rarity exponent α. Finally, the latent vector z^* is decoded into a new synthetic observation (x^*, y^*) [16].

This approach allows for non-parametric, flexible data generation that captures the underlying structure of the data without requiring discretisation of the target variable. It is particularly suited to handling complex distributions in imbalanced regression scenarios [16]. Nevertheless, combining VAEs with smoothed bootstrap introduces higher computational overhead and increases sensitivity to noise parameters, which may impact the robustness of the method [72].

3.2.4 Other Strategies

This subsection presents strategies that combine multiple techniques previously described.

3.2.4.1 COVERT

COVERT, introduced by [71], is a data rebalancing algorithm designed for continuous industrial process datasets. It avoids discretising the target variable and instead identifies rare anchor points using KDE, then generates synthetic data by combining these anchors with neighbouring points, weighted and adjusted using the data's covariance structure. This allows the method to iteratively improve coverage in sparse regions while accounting for uncertainty [71].

COVERT is effective in high-dimensional, real-valued datasets and preserves anchor rarity during generation. However, it is domain-specific, struggles with categorical or Boolean features, does not focus on extreme target values, and incurs higher computational cost due to its iterative process [71].

3.2.4.2 Data Augmentation + Weighted Relevance (DA-WR)

The algorithm proposed by [73] tackles imbalanced covariates regression by first applying Data Augmentation (DA) to expand the input space using a range of methods - from noise injection [74], ROSE [75], and k-NN SMOTE [64] to deep learning or machine learning generative models like Conditional Generative Adversarial Network (CGAN) [76]

and Random Forest (RF) [77]. It then applies Weighted Resampling (WR) to resample the data so that the covariate distribution matches a specified target distribution, using KDE to calculate sampling probabilities [73].

An advantage of this approach is its simplicity: it avoids complex model assumptions and offers flexibility by allowing a wide choice of augmentation techniques. It also improves covariate representativeness and reduces overfitting risks by separating augmentation from resampling [73]. However, the method has key drawbacks as it cannot generate truly novel or out-of-distribution values, often replicating existing patterns. Its effectiveness heavily depends on the selected generator, and it does not address imbalance in the target variable, limiting its applicability to broader imbalanced regression tasks [73].

3.2.4.3 k-Nearest Neighbour Oversampling - Regression (KNNOR-Reg)

The k-Nearest Neighbour Oversampling - Regression (KNNOR-Reg) algorithm, proposed by [21], adapts the original k-Nearest Neighbour Oversampling (KNNOR) method from classification to imbalanced regression by leveraging the relevance function framework introduced in [9, 40]. It combines an enhanced SMOTE-like oversampling technique with k-NN to generate synthetic data that accounts for noise, small disjuncts, and intra-class imbalances. A key innovation lies in its filtering strategy, which selects minority instances that best represent the overall data distribution.

KNNOR-Reg offers three tailored strategies depending on the dataset's characteristics. For low-population data, it uses the KNNOR approach to interpolate new synthetic feature vectors and computes the associated target values via a weighted distance function. This formulation generalises SMOTER when $k > 2$, making it effective for small datasets where traditional methods may fall short [21]. For high-population data, two deep learning variants are introduced. When dealing with high-dimensional data, KNNOR-Reg integrates KNNOR with a Target-Aware Autoencoder, which compresses the input space via an encoder and reconstructs features and targets through a decoder trained with an exponential loss function. For low-dimensional data, the algorithm uses a Target-Aware AutoInflater, which expands the feature space before compressing it again to recover both features and target values. These neural architectures aim to enhance the estimation of target values and allow the oversampling process to better reflect the original data structure [21].

This approach is particularly versatile, handling datasets with varying population sizes and feature dimensionalities. It also addresses typical oversampling weaknesses by mitigating noise sensitivity and capturing non-linear relationships through multiple neighbourhoods. Nevertheless, it comes with limitations. The method requires manual setting of a relevance threshold for oversampling decisions and depends on domain discretisation. Additionally, the choice of the parameter k is non-trivial, and the deep learning components introduce interpretability issues due to their black-box nature [21, 78].

Figure 3.9 illustrates how the deep learning strategies DAVID and KNNOR-Reg modify the distribution of a target variable.

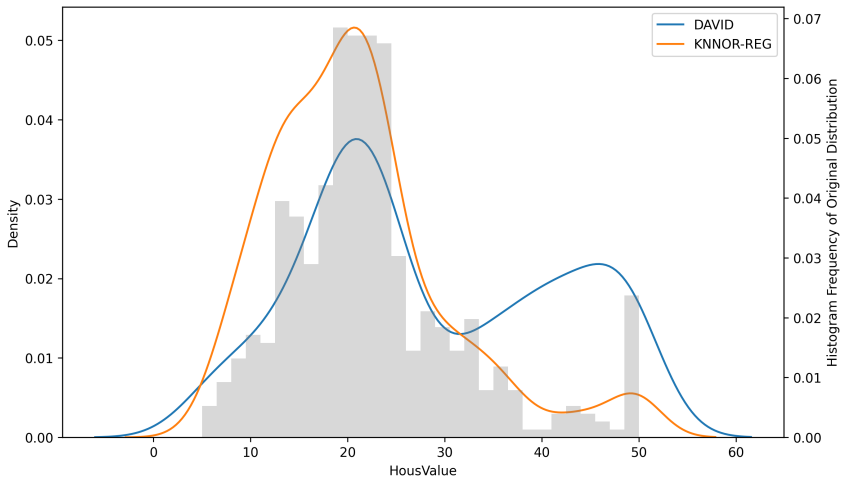


Figure 3.9: Distribution of a target variable before and after applying deep learning strategies, indicated by the histogram and density curves, respectively.

3.3. Comparative Analysis

The diverse set of methods proposed for this machine learning task underscores the significance of this subject. At this point, it is important to emphasise both the benefits and drawbacks of the proposals previously discussed. For example, while certain methods may excel in handling high-dimensional data, others may prove more suitable for datasets featuring a combination of feature types. Therefore, understanding the strengths and weaknesses of each preprocessing strategy is crucial. Tables 3.1 and A.2 from Appendix A provide concise summaries of the advantages and disadvantages of the data-level strategies presented in the literature.

Although the existing strategies previously presented encompass a broad spectrum of methodologies, primarily based on parametric methods that utilise distance metrics, the advent of synthetic data generation has led to the emergence of novel approaches

Table 3.1: Comparison of Data-Level Algorithms by Key Design Characteristics.

Strategy	Neighbour Based	Noise Introduction	Relevance Threshold	Target Domain Discretisation	Other Hyperparameters
RU [18]	✗	✗	✓	✓	✗
RO [17]	✗	✗	✓	✓	✗
WERCS [17]	✗	✗	✗	✗	✓
GN [17]	✗	✓	✓	✓	✓
SMOTER [18, 65]	✓	✗	✓	✓	✓
SMOGN [23]	✓	✓	✓	✓	✓
WSMOTER [20]	✓	✗	✗	✗	✓
G-SMOTER [68]	✓	✗	✓	✓	✓
DIRVAE [14]	✗	✓	✗	✗	✓
IRGAN [15]	✗	✗	✗	✗	✓
DAVID [16]	✗	✓	✗	✗	✓
COVERT [71]	✓	✗	✗	✗	✓
DA + WR [73]	✓	✓	✗	✗	✓
KNNOR-REG [21]	✓	✗	✗	✓	✓

which employ non-parametric methods [79] such as classification and regression trees, random forests, Bayesian techniques, and, more recently, deep learning methods like GANs [80] and VAEs [81], as previously discussed. The subsequent chapter presents a novel and adapted methodology based on the non-parametric classification and regression trees, originally designed from the ground up for imbalanced regression tasks.

4. CART-based Data Augmentation for Imbalanced Regression

This chapter presents CARTGen-IR, a novel data augmentation method for imbalanced regression. It first reviews prior CART-based generation approaches in classification, and then proposes an adaptation tailored to regression that does not require thresholding. The method combines rarity-based resampling with variable-wise data generation using the CART algorithm.

4.1. CART-based Synthetic Data Generation

Among the non-parametric strategies in the literature which have been underexplored within the IR context is the CART [82] algorithm. It is a versatile method for determining the conditional distribution of a univariate result given multivariate features. Using recursive binary splits, CART builds a decision tree that divides the predictor space in a way that creates partitions of the input space, which are progressively more uniform with respect to the target variable. Depending on the learning task, the preference criterion to choose the best splits is based on the minimisation of the Gini impurity or the sum of squared errors, respectively. Figure 4.1 showcases an example of a CART tree, considering two numeric attributes.

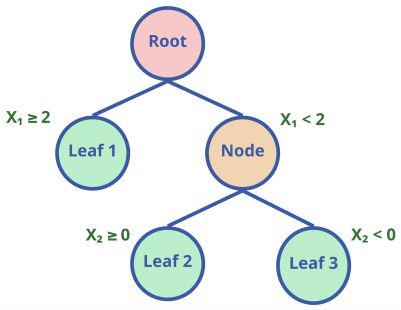


Figure 4.1: CART tree example (adapted from [83])

Reiter [83] has proposed using CART as a method to generate synthetic data and prevent the disclosure of sensitive information in the context of releasing microdata collected from censuses or surveys. According to the author, CART offers several methodological advantages compared to parametric models. Its non-parametric nature makes it particularly suitable for datasets exhibiting irregular or unknown distributional forms, where parametric assumptions may be violated or difficult to infer. Additionally, CART

is capable of detecting complex interactions and non-linear dependencies among predictors without explicit specification, providing a flexible modelling alternative [79, 84]. The algorithm's recursive partitioning mechanism facilitates a semi-automated selection of relevant variables, which is particularly beneficial in high-dimensional settings with numerous potential predictors of mixed data types [79, 83, 85]. This mechanism also enables the smooth handling of missing values, eliminating the need for artificial imputation. However, these advantages must be weighed against certain limitations. CART models can encounter interpretability issues due to abrupt changes at partition boundaries and may underperform in situations where the true underlying relationships are accurately captured by correctly specified parametric models [83].

In a recent work [22], the aforementioned principles [83] were applied to conduct a comparative study of several methods designed to compare performances in synthetic tabular data generation for class imbalance and fairness. Based on the CART-generated decision tree, they employed a sequential, column-wise data generation approach by sampling within the tree leaves, as illustrated in Figure 4.2. This sampling procedure adheres to the conditional distribution estimates to generate the new synthetic data. To mitigate the class imbalance, the researchers proposed employing CART trees for data generation through sampling techniques applied to the leaf nodes. In their study [22], the authors demonstrated the effectiveness of CART models in tackling imbalance in the target variable compared to other traditional methods, such as SMOTE and GANs, particularly in mitigating class imbalance.

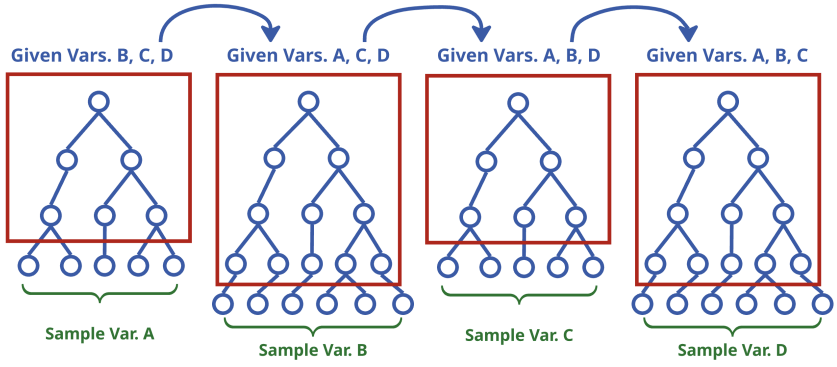


Figure 4.2: Consecutive variable-wise data generation using CART

In another recent study [86], a comparative analysis was conducted of the generation of Synthetic Patient Data (SPD) for survival analysis in oncology trials, aiming to expedite clinical development. The authors concluded that CART emerged as the most effective method for generating SPD from limited datasets, such as clinical trial data, compared

to RF, Bayesian Networks (BN), and Conditional Tabular Generative Adversarial Network (CTGAN) and determined that future research could enhance CART-based generation methods by incorporating feature engineering and employing other techniques [86].

Additionally, the authors of [85] compared various model-independent generative techniques for oversampling tabular datasets and addressing class imbalance. While traditional methods rely on SMOTE [64], the authors evaluated advanced approaches like Gaussian Copula Synthetizer (SDV-GC) [87], CTGAN [76], and generative non-parametric CART trees. All methods were trained on tabular datasets and used to generate synthetic samples. The CART method outperformed others in terms of performance and efficiency and the authors concluded that its non-parametric nature ensures consistent generation, even for underrepresented subgroups under extreme data scarcity [85].

This method is a promising data augmentation technique that captures complex relationships in the original data while avoiding the distortions associated with other synthetic data generation algorithms, such as SMOTE and GANs. The existing body of research, as highlighted in the works of [22, 79, 83, 85, 86], has predominantly concentrated on addressing classification problems characterised by imbalanced datasets. These studies have provided valuable insights and methodologies tailored to the challenges posed by such scenarios, where the distribution of classes is skewed, and the minority class is often of greater interest. However, a notable gap persists in the literature concerning the application of CART to regression problems that exhibit similar imbalances. To date, as far as we know, there has been no comprehensive comparative study that rigorously evaluates the performance, predictive capabilities, and feature selection efficacy of CART when applied to imbalanced regression tasks. This represents a significant gap in the current research landscape.

4.2. Proposed Adaptation for Imbalanced Regression

We propose CART-based Generation for Imbalanced Regression (CARTGen-IR), a method that employs CART to generate synthetic tabular data to tackle imbalanced regression problems. We aim to leverage the good performance that CART sample generation has shown in imbalanced classification problems and apply it to imbalanced regression. Furthermore, we aim to develop a strategy that removes the need for arbitrary user-defined thresholds, especially when identifying relevant or rare cases. This strategy enables us to avoid crisp divisions over the continuous domain of the target variable into

bins or partitions, thereby preventing domain discretisation, which is necessary in other previously proposed algorithms, such as SMOTER [18, 65] or SMOGN [23].

The main idea behind CARTGen-IR is as follows: given a dataset, a weighting mechanism is applied to the target domain values to rank them by rarity and relevance. Each instance is assigned a weight, with rarer and more relevant cases receiving a higher weight and vice versa. The original dataset is then resampled according to these weights. This ensures that relevant cases have a higher probability of being selected, and the final resampled dataset will primarily consist of rare cases related to the target variable domain. After this step, the algorithm is prepared to implement CART-based oversampling. To achieve this, we use the consecutive column-wise generation process, as described in [22] and illustrated in Figure 4.2. The algorithm first fits a CART tree for each column in the dataset, using the other columns as predictors, including the target variable. After fitting, it generates synthetic samples based on the constructed model.

The relevance-based weighting mechanism for rare target values is not new and is used in methods like WERCS [17] and WSMOTER [20] to prioritise relevant instances. However, CARTGen-IR stands out by resampling the original dataset and excluding cases with frequent target values from data augmentation. This approach minimises the inclusion of frequent cases in the final training dataset, resulting in more focused augmentation.

The main steps of CARTGen-IR, outlined in Algorithm 1 and in Flowchart 4.3, are:

1. **Assign a rarity score to each instance in the original dataset.** Our method employs three weighting techniques defined by the $density$ hyperparameter: a baseline KDE method customised for this algorithm, the DenseWeight method [39], and the relevance function [9]. These techniques account for fluctuations in rarity across samples without assuming a specific distribution for rare cases or extremes. The density estimation helps compute a rarity score, which is determined by taking the inverse of the density (plus a small constant to avoid division by zero). However, any other weighting technique can be used. Final weight values are adjusted by a rarity exponent α and normalised to fit a probability distribution, which involves dividing each score by the total sum of all weights.

2. **Resample the original dataset using rarity scores as weights.** The resampling process uses replacement to produce a dataset dominated by rare target values, with the number of selections linked to the desired synthetic samples. We introduced

a hyperparameter η (range: [0,$+\infty$[) to set the proportion of cases generated. For example, $\eta = 0.5$ creates 50% of the original dataset size, while higher values increase synthetic cases significantly. Following common practices in synthetic data generation for imbalanced learning [64, 88], we target five synthetic samples per selected instance to balance diversity and computational efficiency. While other ratios can be used, avoiding very low or high values is crucial to prevent underfitting or overfitting. The selection is based on rarity, ensuring minimal inclusion of common instances, without a preset threshold.

3. **Optionally input noise into duplicated instances.**The resampling process with replacement can create duplicated instances, potentially causing overfitting. To mitigate this, a small amount of noise can be added to the duplicates using the hyperparameter δ. The amount of noise is independent from the standard deviation as numeric features are normalised.

4. **Fit CART to resampled dataset and generate synthetic samples.** To generate synthetic data, a CART decision tree is fitted for each column in the dataset, using the other columns as predictors in a sequential, column-wise generation process.

Tables A.1 and A.2 from Appendix A compare the advantages and disadvantages of CARTGen-IR with the methods already found in the literature.

The following chapter presents an experimental study aimed at evaluating the performance of this new method and comparing it to other existing data-level strategies found in the literature.

Algorithm 1: CARTGen-IR Algorithm

Input: X, Y, α, η, $density$, δ

$\mathbf{X}$: set of predictors
Y: target variable
α : rarity exponent
$density$: density strategy $\in \{"kde_baseline", "denseweight", "relevance"\}$
η : sampling proportion $\in \{"balance", "extreme"\}$
δ: noise

Output: $X_{\text{augmented}}, Y_{\text{augmented}}$

function `CARTGen-IR`(α, $density$, η, δ):
- $ws \leftarrow$ `ComputeRarityWeights`(Y, α, $density$)
- **if** η = $balance$ **then**
  - $N \leftarrow \lfloor 0.5 \cdot |Y| \rfloor$ // nr. cases to generate: 50% of original size
- **else if** η = $extreme$ **then**
  - $N \leftarrow \lfloor 0.75 \cdot |Y| \rfloor$ // nr. cases to generate: 75% of original size
- $(\mathbf{X}_{\text{resampled}}, Y_{\text{resampled}}) \leftarrow$ `sample`($data = (\mathbf{X}, Y), size = N/5, weights = ws, replace =$`True`)
- **if** $\delta > 0$ **then**
  - $\mathbf{X}_{\text{resampled}} \leftarrow$ `AddGaussianNoise`($\mathbf{X}_{\text{resampled}}$,$\delta$) // add noise to duplicates
- **end**
- $model \leftarrow$ `CART`($\mathbf{X}_{\text{resampled}}, Y_{\text{resampled}}$) // fit a CART tree model
- $(X_{\text{synth}}, Y_{\text{synth}}) \leftarrow$ `CARTGen`($model$,N) // column-wise generation from fitted model
- **return** $(\mathbf{X}, Y) \cup (\mathbf{X}_{synth}, Y_{synth})$

function `ComputeRarityWeights`(Y, α, $density$):
- **if** $density = "kde"$ **then**
  - $ws \leftarrow \left(1/(\exp(\text{KDE}(Y, Gaussian) + 1e^{-5}\right)$
  - $ws \leftarrow ws^{\alpha}$
- **else if** $density = "denseweight"$ **then**
  - $ws \leftarrow$ `DenseWeight`(Y, α) // algorithm presented in [39]
- **else if** $density = "relevance"$ **then**
  - $ws \leftarrow \phi(Y)$ // algorithm presented in [9]
  - $ws \leftarrow ws^{\alpha}$
- **return** $ws / \sum ws$

function `AddGaussianNoise`($\mathbf{X}$,δ):
- $idx \leftarrow$ `FindDuplicates`($\mathbf{X}$)
- $attrs \leftarrow$ `GetNumericAttrs`($\mathbf{X}$)
- $\varepsilon \sim \mathcal{N}(0, \delta^2)$ of shape $|idx| \times |attrs|$ // Gaussian noise
- $\mathbf{X}[idx, attrs] \leftarrow \mathbf{X}[idx, attrs] + \varepsilon$ // add noise to numeric features of duplicates
- **return** $\mathbf{X}$

$X_{\text{augmented}}, Y_{\text{augmented}} \leftarrow$ `CARTGen-IR`(α, $density$, η, δ)

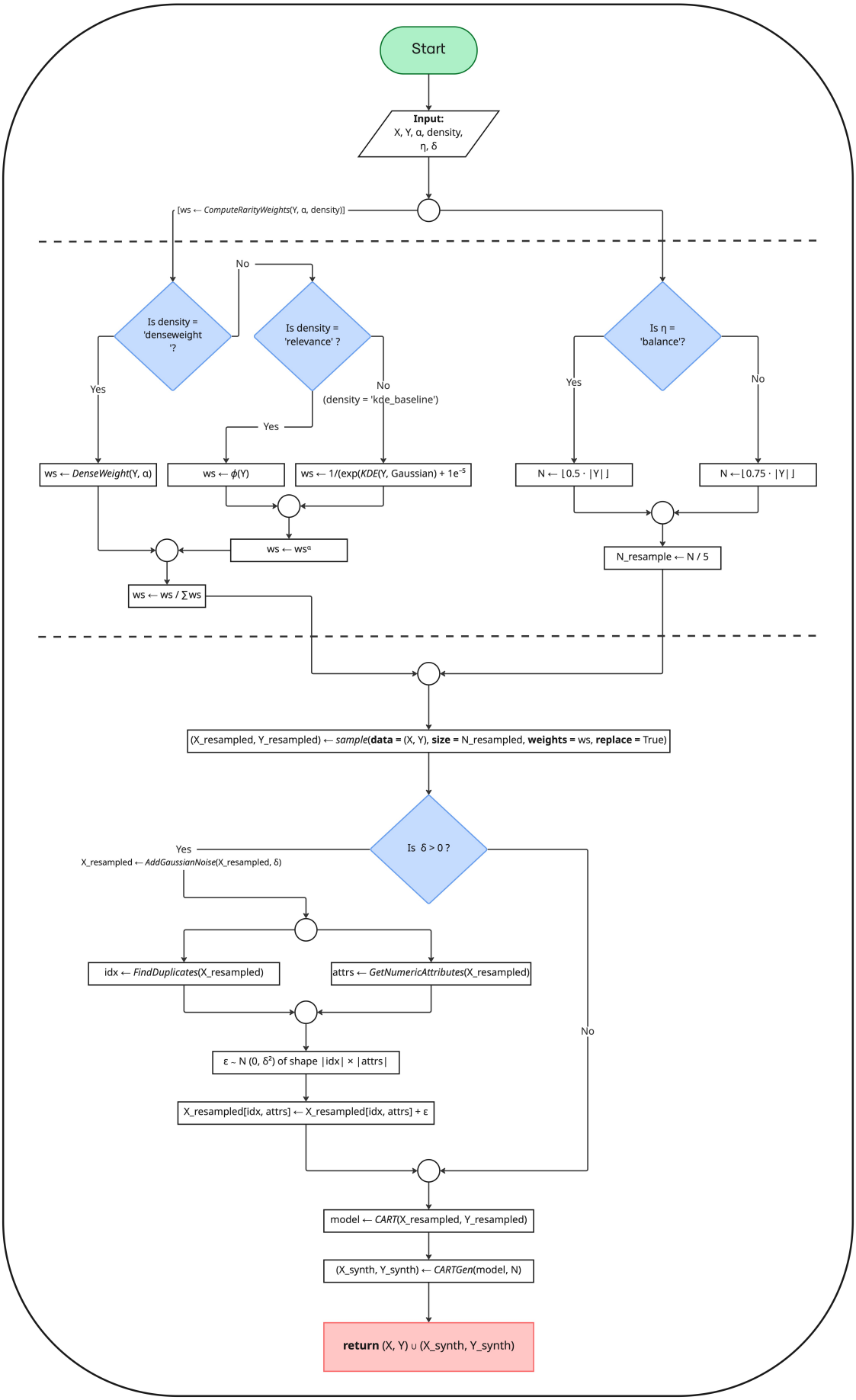


Figure 4.3: CARTGen-IR's data augmentation process flowchart

5. Experimental Study

Our primary objective is to evaluate CARTGen-IR performance and capabilities within the broader context of a comparative study against other state-of-the-art data-level strategies for imbalanced regression tasks. In this section, with the experimental study, we aim to answer the following research questions:

RQ1: Is CARTGen-IR effective for imbalanced regression scenarios?

RQ2: How does it compare to the main state-of-the-art data-level methods proposed for imbalanced regression tasks for tabular data?

RQ3: What tradeoff do these methods offer regarding predictive performance and execution time?

We begin by outlining the experimental setup, which includes a description of the datasets, preprocessing strategies and learning algorithms used in the experiments. Then, we present and discuss the results obtained in light of the aforementioned research questions.

5.1. Experimental Setup

5.1.1 Datasets

We have created a publicly accessible repository with 62 datasets for imbalanced regression research from various application domains. This collection includes datasets referenced in studies such as [17, 18, 20, 23, 68] and can be found at the following repository: Datasets-ImbalancedRegression. Each dataset is briefly described and available for download in ARFF and CSV formats. The repository aims to serve as a standardised benchmark for investigating imbalanced regression issues across diverse real-world contexts.

For this study, we used 15 regression datasets from the repository. The key attributes of these datasets are summarised in Table 5.1, which also reports the absolute and relative frequencies of rare instances, defined according to a relevance threshold of 0.8. To accomplish this, we derived a relevance function for each dataset using the

automated approach outlined in [9]. The chosen datasets present a broad range of characteristics, representative of the overall repository, regarding numeric and nominal features, instances, types of extremes, and rare occurrences, with the aim of evaluating the performance of data augmentation methods across a diverse set of tabular datasets.

Table 5.1: Benchmark regression datasets used in the experiments

Dataset	N	Features	Nominal Features	Numeric Features	Missing	Extreme Type	# Rare	% Rare
strikes	625	6	0	6	No	High	15	2.40%
forestFires	517	12	0	12	No	High	15	2.90%
ele-1	495	2	0	2	No	High	21	4.24%
cpuSm	8192	12	0	12	No	Low	371	4.53%
airfoil	1503	5	0	5	No	High	80	5.32%
fuelConsumption	1764	37	12	25	No	Both	167	9.47%
heat	7400	11	3	8	No	Both	833	11.26%
sensory	576	11	0	11	No	Both	69	11.98%
mortgage	1049	15	0	15	No	Low	133	12.68%
maxTorque	1802	32	13	19	No	Both	235	13.04%
treasury	1049	15	0	15	No	Low	137	13.06%
availablePower	1802	15	7	8	No	Both	305	16.93%
housingBoston	506	13	0	13	No	Both	105	20.75%
abalone	4177	8	1	7	No	Both	1033	24.73%
servo	167	4	4	0	No	Both	59	35.33%

5.1.2 Preprocessing Strategies

We also considered a comprehensive set of preprocessing strategies to address data imbalance in regression tasks. The approaches considered include RU, RO, WERCS, GN, SMOTER, SMOGN, WSMOTER, G-SMOTER, DAVID, KNNOR-Reg and CARTGen-IR. To incorporate the G-SMOTER method into this experimental study, we independently implemented the method in Python, adhering to the algortihm's procedures described in [68] to the best of our ability. This decision was made due to the white-box nature of this algorithm and despite our request to the authors for access to their code, which was not provided. Moreover, it is important to note that a direct comparison with the DIRVAE approach proposed by [14], as well as the IRGAN approach proposed by [15], was not possible, as the corresponding source codes are not publicly available. Furthermore, we excluded the COVERT approach, proposed by [71], and the Data Augmentation + Weighted Relevance (DA-WR) method, designed by [73], as they focused, respectively, on uniform domain preferences and imbalance in covariates rather than in the target variable, which did not align with the main objectives of this experimental study. All strategies were tuned, through their hyperparameters, to guarantee that the augmented datasets

had similar sizes, ensuring fairness for their comparison. A complete overview of the 72 resampling configurations is presented in Table 5.2.

Table 5.2: Data-level strategies and hyperparameters used in the experiments

Strategy	Description	Hyperparameters
None	No sampling applied	–
RU [18]	Random Undersampling	%u = {balance, extreme}
RO [17]	Random Oversampling	%o = {balance, extreme}
WERCS [17]	Weighted Relevance-based Combination Strategy	%u = {0.5, 0.75}, %o = {0.5, 0.75}
GN [17]	Gaussian noise applied after under/oversampling	%u/%o = {balance, extreme}, δ = {0.05, 0.1, 0.5}
SMOTER [18, 65]	SMOTE for Regression	%u/%o = {balance, extreme}, k = {5}
SMOGN [23]	SMOTER with Gaussian Noise	%u/%o = {balance, extreme}, δ = {0.05, 0.1, 0.5}, k = {5}
WSMOTER [20]	Weighted SMOTER	ratio = {1.5, 1.75}, β = {1, 2}, k = {5}
G-SMOTER [68]	Geometric SMOTER for Regression	strategy = {minority, majority, combined}, ξ = {0.7}, η = {0.75}, θ = {-0.5, 0.5}, k = {5}
DAVID [16]	Variational autoencoder-based augmentation	α = {1, 2}
KNNOR-REG [21]	KNNOR for Regression	–
CARTGen-IR	CART-based synthetic data generation for imbalanced regression	α = {1, 1.5, 2.0}, η = {0.5, 0.75}, density = {kde, denseweight, relevance}, δ = {0.001, 0}

5.1.3 Learning Algorithms

All these methods are sampling algorithms or ones that enhance the training set by generating additional synthetic instances. To evaluate their effectiveness, they must be coupled with a predictive learning model. In this study, we employed three such models: RF, Support Vector Regressor (SVR), and XGBoost (XGB). The experimental setup, shown in Table 5.3, comprised 14 different hyperparameter combinations for tuning these learning models. Each model was evaluated across 15 regression datasets under all 72 preprocessing conditions, resulting in a total of 15 120 experiments ($14 \times 15 \times 72$).

Table 5.3: Hyperparameter settings considered for the regression algorithms

Learner	Hyperparameters	Python Package
Random Forest (RF)	n_estimators = {100, 200}, max_features = {sqrt, log2}	scikit-learn [89]
Support Vector Regressor (SVR)	kernel = {rbf}, C = {1, 10, 100}, epsilon = {0.1, 0.5}	scikit-learn [89]
XGBoost (XGB)	n_estimators = {100, 200}, max_depth = {3, 6}	xgboost [90]

5.1.4 Evaluation Metrics

The evaluation of predictive performance in imbalanced regression remains a methodological challenge due to the absence of a universally accepted metric. Various studies in the literature employ different evaluation criteria, which hinders the comparability of results across works. While alternative metrics tailored to imbalanced regression have been proposed, as previously described in Chapter 2, there is still no consensus on a standard benchmark. This lack of uniformity is further compounded by the limitations of conventional metrics such as MAE and MSE, which are widely used despite not being optimised for imbalanced settings.

Therefore, in this study, we used both RW-RMSE and SERA as evaluation metrics to better assess model performance under imbalanced conditions. We also included RMSE as a standard regression error metric to assess the generalisation capability of the models. Furthermore, since there is no uniformity in the weighting mechanism employed by the preprocessing strategies, with some utilising the relevance function [9] and others employing DenseWeight [39], or both, we implemented an adapted version of both RMSE and SERA with DenseWeight [39] - DenseWeight-RMSE (DW-RMSE) and DenseWeight-SERA (DW-SERA), which were originally reliant on the relevance function method [9]. This approach ensures that they incorporate the DenseWeight methodology as their weighting foundation in their calculations, thereby improving and enabling fairer comparisons between strategies that use such a mechanism for their processes. This is the first time DenseWeight-based metrics have been proposed and originally implemented, as far as we know.

All evaluation metrics were computed using a repeated 2×5-fold stratified Cross-Validation (CV) procedure to ensure robust and reliable performance estimates. This

cross-validation setup means that in each run, the dataset was partitioned into five distinct folds, each with similar target variable distributions. For each iteration, one fold was designated as the test set while the remaining four were used for training. This process was repeated five times per repetition, allowing each fold to serve once as the test set, thereby completing a full cross-validation cycle. Additionally, to enhance the robustness of the evaluation, the entire procedure was repeated with a different random partitioning of the data, completing the 2x5-fold stratified CV procedure.

5.2. Empirical Results

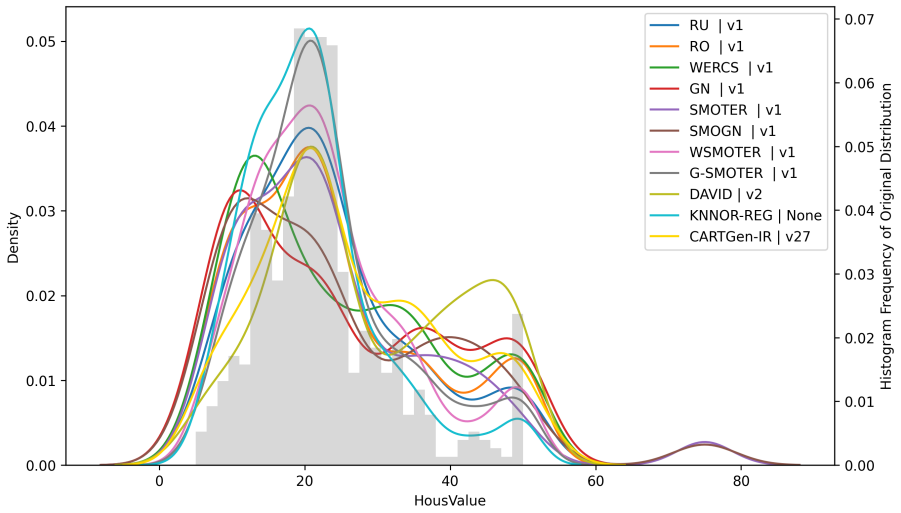
5.2.1 Target Variable Distribution Analysis

To illustrate the impact of the preprocessing strategies on the target variables, we present the density plots of the target variable for two benchmark datasets: *housing-Boston* and *treasury* in Figure 5.1. Each plot overlays the histogram of the original target distribution with the kernel density estimates produced by various sampling and data augmentation techniques, enabling a visual comparison of how each method reshapes the distribution. This comparison focuses on three aspects: the preservation of the original distributional shape, the concentration and spread of density across value ranges, and the overall support coverage.

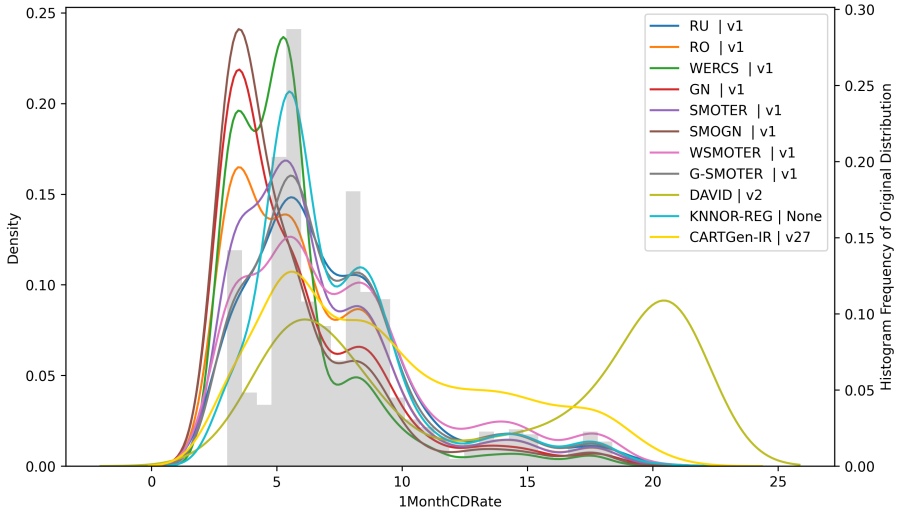
From the plots, we observe that certain methods, such as KNNOR-Reg and G-SMOTER, closely follow the original distribution, especially around the high-density regions, indicating a conservative augmentation approach. In contrast, techniques like DAVID and CARTGen-IR exhibit broader coverage, extending density into the tails of the distribution, suggesting stronger emphasis on rare or extreme values. This behaviour is particularly relevant in imbalanced regression, where performance on tail regions is often critical. Thus, these plots not only expose how well each technique preserves the original characteristics but also reveal their capacity to enhance underrepresented areas of the target space, information that cannot be captured by performance metrics alone. The specific hyperparameter combination used by each variant of the methods is detailed in Table B.1 from Appendix B.1.

5.2.2 Error Performance Analysis

The number of datasets where each pair of resampling strategy and learner model achieves the best error estimates, according to each metric, is presented in Table 5.4. This table was extracted from the analysis of Tables **??**-**??** from Appendix B.1. RMSE



(a) KDE plot of *housingBoston* target variable distribution



(b) KDE plot of *treasury* target variable distribution

Figure 5.1: Distribution of the target variable before and after applying preprocessing strategies, indicated by the histogram and density curves, respectively.

results indicate that, although WSMOTER demonstrates good performance, CARTGen-IR emerges as the superior technique, standing out in this metric. It is noteworthy that the proposed method exhibits exceptional efficacy compared to the original dataset (None) and against widely recognised methods such as SMOTER or RU. This scenario is further corroborated by the RW-RMSE, SERA, DW-RMSE and DW-SERA metrics. As observed previously, CARTGen-IR emerges as the top performer for all metrics, demonstrating its superior performance more than 40% of the experiments with no other method approaching its dominance in any metric. Additionally, the results are uniformly distributed among learners for all preprocessing strategies, with no learner demonstrating significantly superior results. Nevertheless, it is noteworthy that for the CARTGen-IR method, both RF

and XGBoost exhibit better performance, while SVR displays the least favourable performance of all learners for this strategy. Furthermore, this is not true for the rest of the strategies, which perform well with SVR and often achieve their best results with such a model.

Table 5.4: Counts of best scores by preprocessing strategy / learner for each metric

Metric	Learner	None	RU	RO	WERCS	GN	SMOTER	SMOGN	WSMOTER	G-SMOTER	DAVID	KNNOR-REG	CARTGen-IR
RMSE	RF	2	0	1	0	0	0	0	3	0	0	2	**7**
	SVR	1	0	4	0	0	0	0	3	1	0	1	**5**
	XGBoost	1	0	1	0	0	0	0	3	3	0	0	**7**
	Total	4	0	6	0	0	0	0	9	4	0	3	**19**
RW-RMSE	RF	0	0	0	4	0	1	0	2	0	0	0	**8**
	SVR	0	0	4	3	0	0	1	1	0	0	0	**6**
	XGBoost	1	0	1	1	0	0	0	3	2	0	0	**7**
	Total	1	0	5	8	0	1	1	6	2	0	0	**21**
SERA	RF	0	0	0	3	0	1	1	2	0	0	0	**8**
	SVR	0	0	4	2	0	0	2	2	0	0	0	**5**
	XGBoost	1	0	0	1	0	0	0	3	1	0	0	**9**
	Total	1	0	4	6	0	1	3	7	1	0	0	**22**
DW-RMSE	RF	0	0	1	2	0	0	0	2	0	0	0	**10**
	SVR	0	0	3	2	0	0	0	4	1	0	0	**5**
	XGBoost	1	0	1	0	1	0	0	5	1	0	0	**6**
	Total	1	0	5	4	1	0	0	11	2	0	0	**21**
DW-SERA	RF	0	0	1	0	0	0	1	4	0	0	1	**8**
	SVR	0	0	3	0	1	0	0	**5**	2	0	0	4
	XGBoost	1	0	1	0	0	1	0	4	2	0	0	**6**
	Total	1	0	5	0	1	1	1	13	4	0	1	**18**

This observation is further supported by Table B.7 from Appendix B.2, which presents the optimal combination of preprocessing strategy and learner for each pair of metric and dataset. Notably, CARTGen-IR consistently achieves the highest number of best results among all preprocessing strategies. It is noteworthy that the RF learner rarely produces the optimal outcomes, regardless of the preprocessing strategy it is paired with. Conversely, both SVR and XGBoost demonstrate split dominance in the learner category.

Furthermore, Figure 5.2 provides a comprehensive summary of the wins and losses (including significant wins and losses) achieved by each data augmentation technique used in this study across three metrics - RMSE, RW-RMSE and SERA. Figure B.1 from Appendix B.3 illustrates the (significant) wins and losses for the remaining metrics - DW-RMSE and DW-SERA. The analysis is conducted relative to the scenario of using the original imbalanced data as the baseline. The statistical significance of the results was determined using a Wilcoxon Signed Rank Test, with a predefined 95% threshold for statistical significance. Due to the comparative study's evaluation of 36 parameter combinations for the CARTGen-IR model, it's impractical to include all versions in the figure. We have selected the six best-performing versions of that strategy, along with all versions of the competing approaches, for visualisation. Figures B.2-B.6 from Appendix

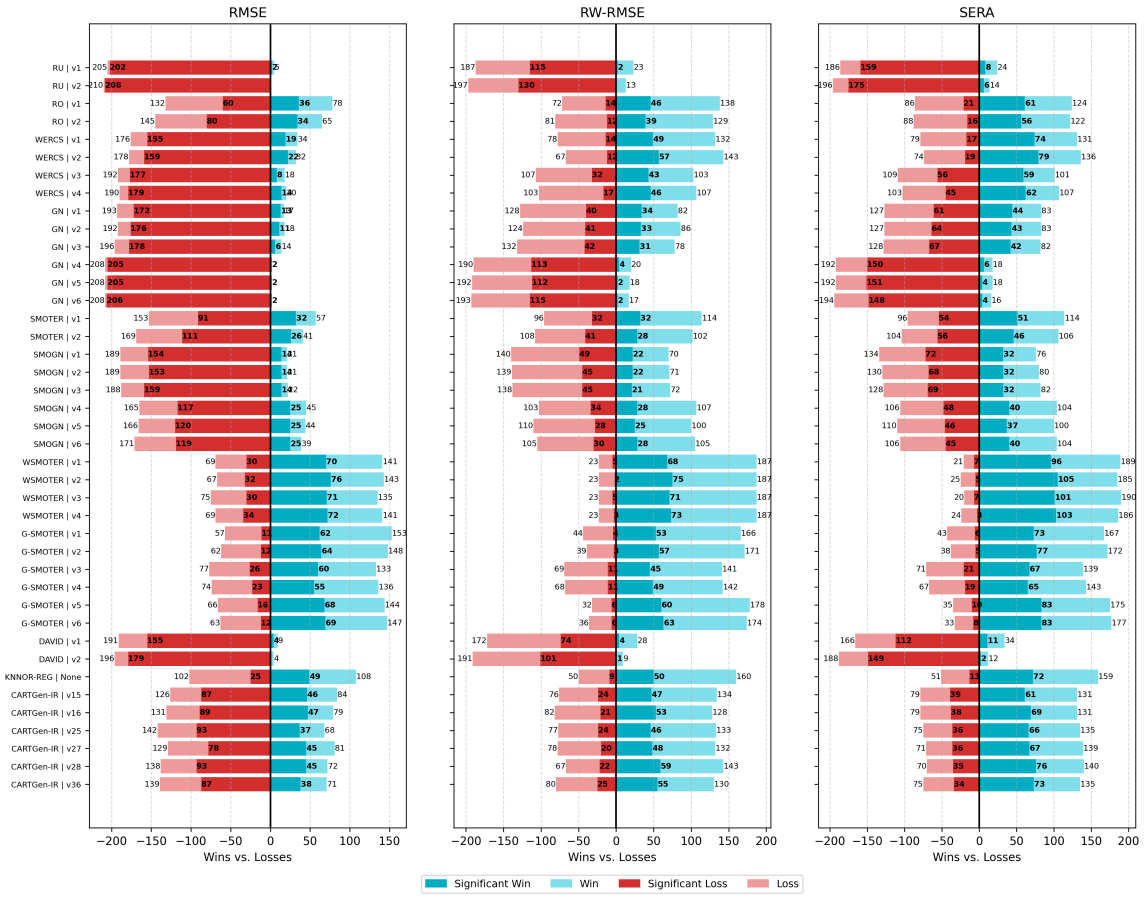


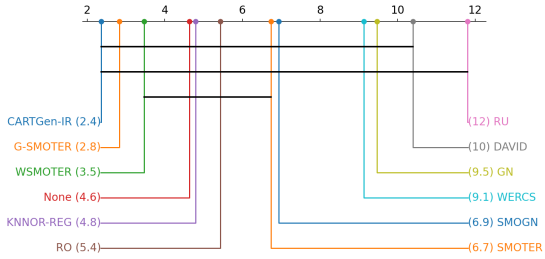
Figure 5.2: Comparison of (significant) wins and losses of selected preprocessing strategies across all datasets, according to different metrics - RMSE, RW-RMSE and SERA

B.3 present the summary of wins and losses for all the preprocessing strategies variants evaluated in the study. By the analysis of this figure, we can conclude that WSMOTER emerges as the most consistent method in our study across all evaluated metrics, followed by KNNOR-Reg. CARTGen-IR also exhibits satisfactory results, generally ranking third in consistency across all metrics and all versions included. This is a characteristic that other methods do not exhibit. For instance, WERCS consistently performs better in its first two versions, where oversampling surpasses undersampling in proportion. However, its performance diminishes in the other two versions, as undersampling weights more heavily than oversampling. It is also noteworthy that while CARTGen-IR may not be the most consistent winner, it demonstrates a superior significant win-to-lose ratio compared to similarly performing methods in these results. This implies that for the same number of wins, CARTGen-IR's are generally more significant than other methods at that level. Moreover, we observe that the most effective versions of CARTGen-IR exhibit a common characteristic: the introduction of Gaussian noise into the numerical variables. Additionally, the α values for these versions are set to 1.5 and 2, suggesting that this method

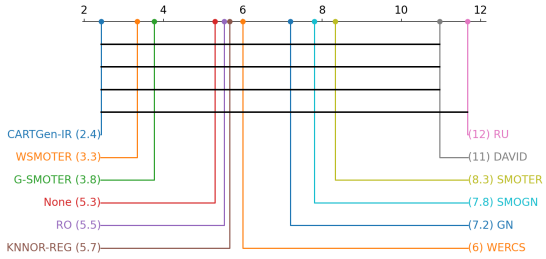
appears to benefit from higher values for this hyperparameter. In terms of other parameters, both DenseWeight and Relevance demonstrate superior performance when it comes to the calculation of the rarity score and the number of synthetic samples does not appear to have a significant impact on the performance of these methods. All results from the Wilcoxon Signed-Rank test are presented in Appendix B.3.

We also used multiple statistical tests to compare the resampling strategies across the 15 datasets. The Friedman F-test showed that the tested strategies have distinct performance characteristics for all metrics, since with a 95% significance level, we rejected the null hypothesis. Table B.13 from Appendix B.4 shows the test results.

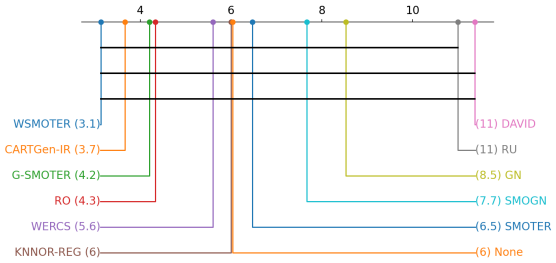
Given the rejection of the null hypothesis, we conducted the post-hoc Nemenyi test to identify methods with significantly different results. This test, conducted again with a 95% significance threshold, generated the Critical Difference (CD) diagrams proposed by [91], presented in Figures B.7- B.11 from Appendix B.4. The diagrams display the average rankings of the strategies being compared, with connections made when the differences are not statistically significant. Overall, the results indicate that there are no significant differences among the strategies. However, CARTGen-IR consistently shows the best overall performance across all metrics, closely followed by WSMOTER. It holds top rank positions, particularly for RW-RMSE and SERA, across all learners (Figures 5.3a



(a) RMSE – RF



(b) SERA – XGboost



(c) DW-SERA – SVR

Figure 5.3: Critical Difference diagrams of all data-level strategies for a metric-learner pair, considering all datasets at 95% confidence level.

and 5.3b). Even when not ranked first (e.g., with SVR under RMSE or DW-SERA - Figure 5.3c), its performance is nearly equivalent to the best, with no significant difference. SVR remains the weakest learner for CARTGen-IR, while DAVID and RU are the poorest performing strategies overall.

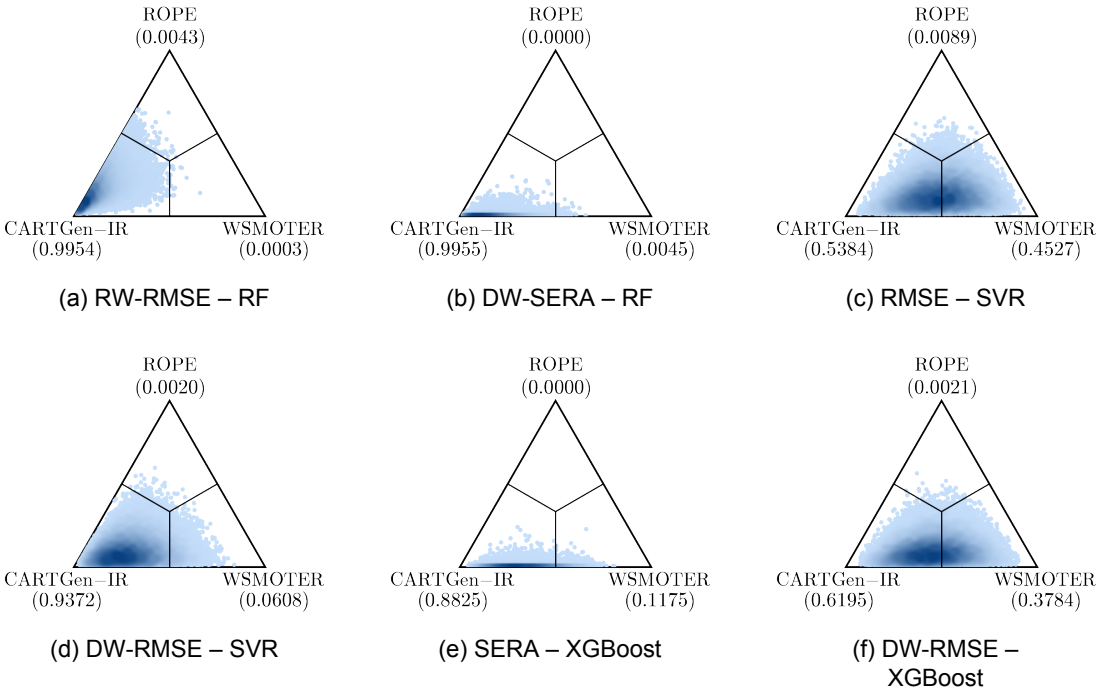
Given that CARTGen-IR and WSMOTER both demonstrate superiority in some of the results previously analysed, we decided to perform a Bayesian Signed-Rank test to compare them. The Bayesian signed-rank test offers a probabilistic interpretation of model comparisons that complements the insights provided by CD diagrams. While CD diagrams illustrate average rank differences across datasets and highlight statistically significant differences based on frequentist tests like the Nemenyi test [91], they do not quantify the probability that one method is better than another. In contrast, the Bayesian signed-rank test estimates the posterior probabilities that one method outperforms another, is worse, or that their performances are practically equivalent, incorporating a Region of Practical Equivalence (ROPE) [92]. This enables more nuanced conclusions by capturing uncertainty and avoiding binary decisions based solely on p-values, providing richer evidence for strategy comparison [93]. For this experimental study, we defined the ROPE interval to be $[-1\%, 1\%]$. The results of this test are presented in Figure 5.4, with a representative selection of six of the generated Bayesian ternary plots. The full collection of ternary plots, along with a table that presents the details of this test are displayed in Appendix B.5.

A detailed examination of the ternary plots reveals that CARTGen-IR substantially outperforms WSMOTER when used with RF across all metrics, with the posterior probability that CARTGen-IR is superior to WSMOTER standing at around 99% across all metrics - Figures 5.4a and 5.4b. These results provide strong evidence that CARTGen-IR consistently delivers better performance when paired with RF.

When coupled with SVR, the results are more balanced. For both RMSE and RW-RMSE, the probabilities that CARTGen-IR is better than WSMOTER hover around 53%, while WSMOTER is favored in approximately 46% of the cases - Figure 5.4c. For these metrics, the probabilities for equivalence remain low, suggesting that while there is no clear dominance, the performances are not practically indistinguishable. In contrast, for metrics like SERA and DW-RMSE, CARTGen-IR reflects a dominant position, with probabilities exceeding 90% - Figure 5.4d.

With XGBoost, the results indicate a moderate to dominant advantage for CARTGen-IR. For nearly all metrics, the posterior probabilities exceed 70%, and reaching as high as 90% - Figure 5.4e. The worst performing metric for XGBoost is DW-RMSE, with a probability that CARTGen-IR outperforms WSMOTER of about 61% - Figure 5.4f.

Overall, the Bayesian signed-rank test confirms that CARTGen-IR outperforms WSMOTER in all of the tested scenarios, across all metrics and especially when using ensemble-based learners, but remains competitive even with the SVR model.



(a) RW-RMSE – RF (b) DW-SERA – RF (c) RMSE – SVR

(d) DW-RMSE – SVR (e) SERA – XGBoost (f) DW-RMSE – XGBoost

Figure 5.4: Bayesian Posterior Ternary Plots – CARTGen-IR vs. WSMOTER

5.2.3 Runtime Performance Analysis

To conclude our experimental study, we conducted a comparative analysis of the runtime for all data augmentation techniques. To ensure fairness and provide comparable working conditions for each method, we gathered data solely from the data augmentation procedure under consistent parallelisation conditions. As shown in Figure 5.5, which displays a boxplot of runtime values in seconds for each strategy, CARTGen-IR emerges as one of the fastest-performing techniques in this study. RU and RO, along with WERCS, are sampling techniques that do not produce new synthetic data and demonstrate minimal execution times. Among the other data augmentation techniques that create new synthetic samples, CARTGen-IR is only outperformed by KNNOR-Reg in terms of runtime execution. Yet, it has a lower standard deviation, as is indicated in Table B.15

from Appendix B.6, where all the runtime average and standard deviation values for all datasets are presented, ensuring consistency during its runs. While WSMOTER and G-SMOTER operate within a similar timeframe, all other methods, particularly those based on SMOTER, show poor runtime values. Notably, DAVID, the slowest method, requires an amount of time equivalent to 31 runtimes of CARTGen-IR's.

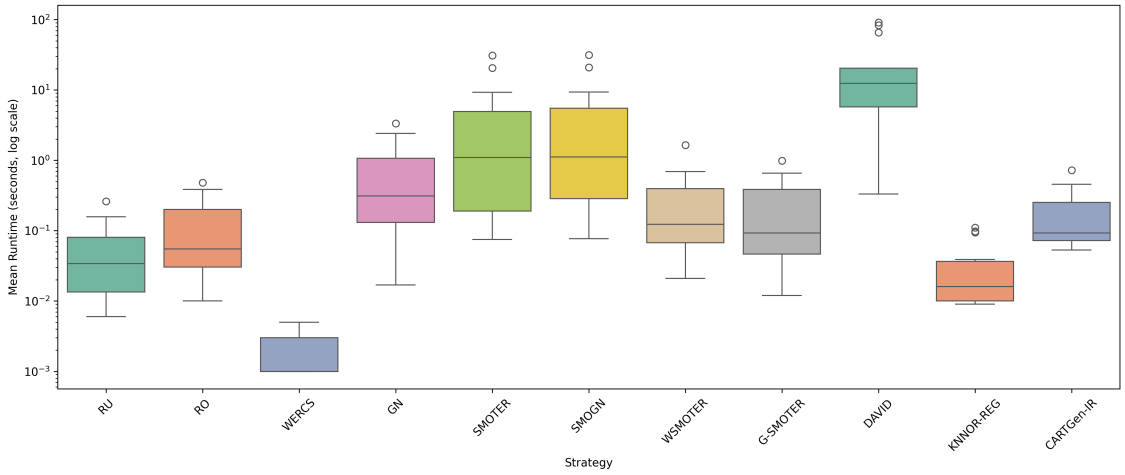


Figure 5.5: Boxplot of runtimes per preprocessing strategy

5.3. Discussion

In response to **RQ1** and **RQ2**, CARTGen-IR has proven to be effective for imbalanced regression tasks. It consistently achieves strong performance across all datasets and ranks the highest overall. When compared to the leading state-of-the-art resampling strategies, CARTGen-IR either improves upon or matches these methods, particularly in balancing the focus between rare-valued and common-valued cases. This is a key performance observation for comparison to other strategies, such as KNNOR-Reg, which performs well with metrics that account for all cases, no matter their relevance, but lacks focus power for rare instances, or vice-versa. For the RW-RMSE and SERA metrics, which are specifically designed to assess performance in imbalanced regression tasks, WSMOTER and CARTGen-IR emerge as clear winners, while also demonstrating solid performance for RMSE. This indicates that CARTGen-IR generalises well across the entire domain without compromising the overall predictive performance. Furthermore, it exhibits a superior win-to-loss ratio compared to the other two methods. The CD diagrams and the Bayesian Signed-Rank test further corroborate these findings, showcasing CARTGen-IR's strong generalisation capabilities while maintaining focus on rare cases.

As for **RQ3**, when analysing the characteristics of each preprocessing strategy, we can categorise the methods into sampling and augmentation techniques. Sampling

techniques, due to their simplicity in implementation, result in the lowest execution times; however, they also yield inconsistent and lower-ranked scores. The other methods, which generate new synthetic samples, can then be compared. The best method in terms of runtime is KNNOR-Reg, which, as previously mentioned, struggles with rare-valued target cases. CARTGen-IR ranks second in execution time and exhibits strong performance, effectively balancing the trade-off between the two.

6. Conclusion

This thesis focused on imbalanced regression problems, where the goal is to predict rare values of the continuous target variable with a skewed distribution, a still-challenging class of problems in machine learning. In this chapter, we present the primary contributions of this study, while also highlighting the limitations of the proposed algorithm and perspectives for future research.

6.1. Main Contributions

The main contribution of this study is the proposal of CARTGen-IR, a non-parametric method within data-level approaches, specifically designed for imbalanced regression, which is based on a previously proposed CART-based synthetic generation method for tabular data [22, 83]. Unlike other state-of-the-art resampling strategies proposed for imbalanced regression, this method does not rely on any specific threshold set for the continuous target variable. By leveraging a CART-based mechanism, CARTGen-IR can model complex relationships in the data while maintaining computational efficiency and interpretability on the data generation process. The empirical evaluation, conducted across a diverse set of benchmark datasets, learning models, evaluation metrics, and state-of-the-art data-level strategies, confirms the competitiveness of the proposed approach in imbalanced regression scenarios.

As additional contributions, this work includes:

(i) the extension and adaptation of relevance-based techniques, such as random under and oversampling, WERCS, GN, SMOTER and SMOGN, to use adjusted boxplot statistics, to better account for target variable skewness;

(ii) the first implementation in the Python programming language of a cost-sensitive learner - XGBoost - with an imbalanced regression-specific metric - SERA - as the optimising metric;

(iii) the first implementation of imbalanced regression-specific metrics that use the DenseWeight [39] as their weighting mechanism;

(iv) and the public release of a curated repository comprising 62 benchmark datasets for imbalanced regression tasks.

All the code used in this study is publicly available on GitHub for reproducibility at `https://github.com/antoniopedropi/SyntheticTabularDataAugmentation-IR.git`. Furthermore, to facilitate the replication of the experimental study presented in this thesis, a reproducible capsule was developed and published, being available with DOI 10.24433/CO.7826905.v2 [94] and at `codeocean.com/capsule/7826905/tree`. Regarding the implementation, it is important to mention that we used the *python-synthpop* package [95], which provides the CART algorithm implementation in Python, essential to develop the CARTGen-IR strategy. Additional information regarding the code and reproducible capsule is provided in Table C.1 and Figure C.1 of Appendix C.

6.2. Limitations and Future Work

Overall, CARTGen-IR offers a promising direction for future work on robust and interpretable solutions for imbalanced regression problems. Conversely, this method may encounter interpretability challenges due to abrupt transitions at partition boundaries. It may exhibit suboptimal performance in scenarios where appropriately specified parametric models accurately represent the underlying true relationships. This could render the model unreliable and significantly underperform in the presence of such parametric relationships within the data. Furthermore, CARTGen-IR is dependent on tunable hyperparameters - a rarity exponent (α), a sample size ratio (η), and noise proportion (δ) - which significantly influence results and require careful fine tuning.

Even though CARTGen-IR demonstrates promise for tackling imbalanced regression problems, several directions for future work include: (i) expanding the comparative study to a broader dataset pool, particularly those containing target domain rare intervals that are not extremes; (ii) incorporating learners with a cost-sensitive approach to imbalanced regression-specific metrics, such as SERA, into the comparative study (e.g. XGBoost with SERA as optimising metric [58]); and (iii) identifying techniques to address the unsuitability of CARTGen-IR for parametric distributions, as partition boundaries may underperform in effectively capturing the true data relationships.

References

[1] P. Branco, L. Torgo, and R. P. Ribeiro, “A survey of predictive modeling on imbalanced domains,” *ACM Comput. Surv.*, vol. 49, no. 2, 2016. [Online]. Available: https://doi.org/10.1145/2907070 [Cited on pages ix, 1, 2, 4, 9, 11, 14, and 15.]

[2] V. A. Bento, D. C. Lima, L. C. Santos, M. M. Lima, A. Russo, S. A. Nunes, C. C. DaCamara, R. M. Trigo, and P. M. Soares, “The future of extreme meteorological fire danger under climate change scenarios for iberia,” *Weather and Climate Extremes*, vol. 42, p. 100623, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212094723000762 [Cited on pages 1 and 6.]

[3] NHSJS, “A comparative analysis of machine learning models for wildfire prediction,” 2024, accessed: 2024-12-29. [Online]. Available: https://nhsjs.com/2024/a-comparative-analysis-of-machine-learning-models-for-wildfire-prediction/

[4] S.-E.-T. Chowdhury, K. Zhu, and Y. Zhang, “Mitigating greenhouse gas emissions through generative adversarial networks based wildfire prediction,” 08 2021. [Cited on pages 1 and 6.]

[5] T. D. Science, “Machine learning for cybersecurity 101,” 2019, accessed: 2024-12-29. [Online]. Available: https://towardsdatascience.com/machine-learning-for-cybersecurity-101-7822b802790b [Cited on pages 1 and 5.]

[6] AltexSoft, “Ai in cybersecurity: Challenges, opportunities, and future trends,” 2023, accessed: 2024-12-29. [Online]. Available: https://www.altexsoft.com/blog/ai-cybersecurity/ [Cited on pages 1 and 5.]

[7] M. Herrera, L. Torgo, J. Izquierdo, and R. Pérez-García, “Predictive models for forecasting hourly urban water demand,” *Journal of Hydrology*, vol. 387, no. 1, pp. 141–150, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022169410001861 [Cited on pages 1 and 6.]

[8] O. Akbilgic, H. Bozdogan, and M. E. Balaban, “A novel hybrid rbf neural networks model as a forecaster,” *Statistics and Computing*, vol. 24, no. 3, pp. 365–375,

2014. [Online]. Available: https://doi.org/10.1007/s11222-013-9375-7 [Cited on pages 1 and 6.]

[9] R. P. Ribeiro and N. Moniz, "Imbalanced regression and extreme value prediction," *Machine Learning*, vol. 109, no. 9, pp. 1803–1835, 2020. [Online]. Available: https://doi.org/10.1007/s10994-020-05900-9 [Cited on pages ix, 1, 4, 6, 7, 8, 9, 11, 12, 13, 14, 17, 18, 20, 22, 23, 27, 33, 35, 38, and 40.]

[10] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*. Berlin: Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-98074-4

[11] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025513005124

[12] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016. [Online]. Available: https://doi.org/10.1007/s13748-016-0094-0

[13] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, 1st ed. New York: Wiley-IEEE Press, 2013. [Cited on page 1.]

[14] H. Tian, C. Tian, K. Li, and W. Jia, "Unbalanced regression sample generation algorithm based on confrontation," *Information Sciences*, vol. 642, p. 119157, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025523007429 [Cited on pages 2, 24, 25, 29, 38, 64, and 66.]

[15] X. Liu and H. Tian, "Research on imbalanced data regression based on confrontation," *Processes*, vol. 12, no. 2, 2024. [Online]. Available: https://www.mdpi.com/2227-9717/12/2/375 [Cited on pages 25, 29, 38, 64, and 66.]

[16] S. Stocksieker, D. Pommeret, and A. Charpentier, "Data augmentation with variational autoencoder for imbalanced dataset," 2024. [Online]. Available: https://arxiv.org/abs/2412.07039 [Cited on pages 2, 11, 25, 26, 29, 39, 64, and 66.]

[17] P. Branco, L. Torgo, and R. P. Ribeiro, "Pre-processing approaches for imbalanced distributions in regression," *Neurocomputing*, vol. 343, pp. 76–99, 2019, learning

in the Presence of Class Imbalance and Concept Drift. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219301638 [Cited on pages 2, 12, 17, 18, 19, 20, 29, 33, 37, 39, 64, and 65.]

[18] L. Torgo, P. Branco, R. P. Ribeiro, and B. Pfahringer, "Resampling strategies for regression," *Expert Systems*, vol. 32, no. 3, pp. 465–476, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12081 [Cited on pages 2, 12, 17, 20, 21, 22, 29, 33, 37, 39, 64, and 65.]

[19] P. Branco, L. Torgo, and R. P. Ribeiro, "SMOGN-LIDTA17: Implementation of smogn for imbalanced regression," 2017, gitHub repository. [Online]. Available: https://github.com/paobranco/SMOGN-LIDTA17 [Cited on pages 2 and 65.]

[20] L. Camacho and F. Bacao, "Wsmoter: a novel approach for imbalanced regression," *Applied Intelligence*, vol. 54, no. 19, pp. 8789–8799, 2024. [Online]. Available: https://doi.org/10.1007/s10489-024-05608-6 [Cited on pages 22, 23, 29, 33, 37, 39, 64, and 65.]

[21] S. B. Belhaouari, A. Islam, K. Kassoul, A. Al-Fuqaha, and A. Bouzerdoum, "Oversampling techniques for imbalanced data in regression," *Expert Systems with Applications*, vol. 252, p. 124118, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417424009849 [Cited on pages 2, 27, 28, 29, 39, 64, and 67.]

[22] E. Panagiotou, A. Roy, and E. Ntoutsi, "Synthetic tabular data generation for class imbalance and fairness: A comparative study," 2024. [Online]. Available: https://arxiv.org/abs/2409.05215 [Cited on pages 2, 31, 32, 33, and 50.]

[23] P. Branco, L. Torgo, and R. P. Ribeiro, "SMOGN: a pre-processing approach for imbalanced regression," in *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, ser. PMLR, vol. 74, 2017, pp. 36–50. [Cited on pages ix, 2, 21, 22, 29, 33, 37, 39, 64, and 65.]

[24] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. [Cited on page 5.]

[25] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," *International Journal of Engineering Technology*, vol. 7, pp. 479–482, 01 2018. [Cited on page 5.]

[26] P.-F. Marteau, "Random partitioning forest for point-wise and collective anomaly detection—application to network intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2157–2172, 2021. [Cited on page 5.]

[27] E. Biglar Beigi, H. Hadian Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*, Oct 2014, pp. 247–255. [Cited on page 5.]

[28] M. Schultz, C. Betancourt, B. Gong, F. Kleinert, M. Langguth, L. Leufen, A. Mozaffari, and S. Stadtler, "Can deep learning beat numerical weather prediction?" *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, vol. 379, 02 2021. [Cited on page 5.]

[29] K. Varga, C. Jones, A. Trugman, L. M. V. Carvalho, N. McLoughlin, D. Seto, C. Thompson, and K. Daum, "Megafires in a warming world: What wildfire risk factors led to california's largest recorded wildfire," *Fire*, vol. 5, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2571-6255/5/1/16 [Cited on page 6.]

[30] W. Ma, L. Zhai, A. Pivovaroff, J. Shuman, P. Buotte, J. Ding, B. Christoffersen, R. Knox, M. Moritz, R. A. Fisher, C. D. Koven, L. Kueppers, and C. Xu, "Assessing climate change impacts on live fuel moisture and wildfire risk using a hydrodynamic vegetation model," *Biogeosciences*, vol. 18, no. 13, pp. 4005–4020, 2021. [Online]. Available: https://bg.copernicus.org/articles/18/4005/2021/

[31] G. Yu, Y. Feng, J. Wang, and D. B. Wright, "Performance of fire danger indices and their utility in predicting future wildfire danger over the conterminous united states," *Earth's Future*, vol. 11, no. 11, p. e2023EF003823, 2023, e2023EF003823 2023EF003823. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2023EF003823 [Cited on page 6.]

[32] M. Aldrin and I. H. Haff, "Generalised additive modelling of air pollution, traffic volume and meteorology," *Atmospheric Environment*, vol. 39, no. 11, pp. 2145–2155, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S135223100500021X [Cited on page 6.]

[33] I. Koprinska, M. Rana, and V. Agelidis, “Yearly and seasonal models for electricity load forecasting,” 07 2011, pp. 1474–1481. [Cited on page 6.]

[34] B. Hobbs, S. Jitprapaikulsarn, S. Konda, V. Chankong, K. Loparo, and D. Maratukulam, “Analysis of the value for unit commitment of improved load forecasts,” *IEEE Transactions on Power Systems*, vol. 14, no. 4, pp. 1342–1348, Nov 1999. [Cited on page 6.]

[35] A. Hald, *A History of Mathematical Statistics from 1750 to 1930*, ser. Wiley Series in Probability and Statistics. Wiley-Interscience, 1998. [Cited on page 7.]

[36] R. R. Wilcox, “Comparing the means of two independent groups,” *Biometrical Journal*, vol. 32, no. 7, pp. 771–780, 1990.

[37] ——, *Introduction to Robust Estimation and Hypothesis Testing*. Elsevier, 2012. [Cited on page 7.]

[38] S. F. Crone, S. Lessmann, and R. Stahlbock, “Utility based data mining for time series analysis: cost-sensitive learning for neural network predictors,” in *Proceedings of the 1st International Workshop on Utility-Based Data Mining*, ser. UBDM ’05. New York, NY, USA: Association for Computing Machinery, 2005, p. 59–68. [Online]. Available: https://doi.org/10.1145/1089827.1089835 [Cited on pages 7 and 11.]

[39] M. Steininger, K. Kobs, P. Davidson, A. Krause, and A. Hotho, “Density-based weighting for imbalanced regression,” *Machine Learning*, vol. 110, no. 8, pp. 2187–2211, 2021. [Online]. Available: https://doi.org/10.1007/s10994-021-06023-5 [Cited on pages ix, 8, 9, 10, 15, 22, 33, 35, 40, and 50.]

[40] R. P. Ribeiro, “Utility-based regression,” PhD thesis, Faculty of Sciences, University of Porto, 2011. [Cited on pages 8, 12, 13, 17, 18, 20, 23, and 27.]

[41] M. Hubert and E. Vandervieren, “An adjusted boxplot for skewed distributions,” *Computational Statistics Data Analysis*, vol. 52, no. 12, pp. 5186–5201, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167947307004434 [Cited on page 8.]

[42] N. Moniz, P. Branco, and L. Torgo, “Resampling strategies for imbalanced time series,” 10 2016. [Cited on pages ix and 9.]

[43] R. Dougherty, A. Edelman, and J. Hyman, "Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic hermite interpolation," *Mathematics of Computation*, vol. 52, pp. 471–494, 04 1989. [Cited on page 9.]

[44] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. London: CRC Press, 1986, vol. 26. [Cited on page 9.]

[45] J. Ren, M. Zhang, C. Yu, and Z. Liu, "Balanced mse for imbalanced visual regression," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 7916–7925. [Cited on page 11.]

[46] R. Ranawana and V. Palade, "Optimized precision - a new measure for classifier performance evaluation," in *2006 IEEE International Conference on Evolutionary Computation*, July 2006, pp. 2254–2261. [Cited on page 11.]

[47] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009. [Cited on page 11.]

[48] J. Van Hulse, T. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," vol. 227, 06 2007, pp. 935–942. [Cited on page 11.]

[49] M. Cain and C. Janssen, "Real estate price prediction under asymmetric loss," *Annals of the Institute of Statistical Mathematics*, vol. 47, no. 3, pp. 401–414, 1995. [Online]. Available: https://doi.org/10.1007/BF00773391 [Cited on page 11.]

[50] A. Zellner, "Bayesian estimation and prediction using asymmetric loss functions," *Journal of the American Statistical Association*, vol. 81, no. 394, pp. 446–451, 1986. [Online]. Available: http://www.jstor.org/stable/2289234

[51] P. F. Christoffersen and F. X. Diebold, "Optimal prediction under asymmetric loss," *Econometric Theory*, vol. 13, no. 6, pp. 808–817, 1997. [Online]. Available: http://www.jstor.org/stable/3532894

[52] C. W. Granger, "Outline of forecast theory using generalized cost functions," *Spanish Economic Review*, vol. 1, no. 2, pp. 161–173, 1999. [Online]. Available: https://doi.org/10.1007/s101080050007

[53] T.-H. Lee, "Loss functions in time series forecasting," *International Encyclopedia of the Social Sciences*, 04 2007. [Cited on page 11.]

[54] J. Hernández-Orallo, "Roc curves for regression," *Pattern Recognition*, vol. 46, no. 12, pp. 3395–3411, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320313002665 [Cited on page 11.]

[55] N. Japkowicz, *Assessment Metrics for Imbalanced Learning*. Wiley, 06 2013, pp. 187–206. [Cited on page 11.]

[56] L. Torgo and R. Ribeiro, "Utility-based regression," in *Knowledge Discovery in Databases: PKDD 2007*, J. N. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 597–604. [Cited on pages 12 and 13.]

[57] S. R. Tulon, J.-G. Gaudreault, and P. Branco, "imbalance-metrics: A python package for evaluating imbalanced datasets," https://pypi.org/project/imbalance-metrics/, 2025, available via PyPI. [Cited on page 12.]

[58] A. Silva, "Boosting the predictions of extreme values," Master's Thesis, Faculdade de Ciências da Universidade do Porto, Porto, Portugal, December 2022, supervisors: Prof. Dra. Rita Ribeiro, Prof. Dr. Nuno Moniz. [Cited on pages 14, 15, and 51.]

[59] R. P. Ribeiro, *Imbalanced Domain Learning*, 2023, fraud Detection - 2023/2024, Lecture Notes. [Cited on page 15.]

[60] N. Moniz, R. Ribeiro, V. Cerqueira, and N. Chawla, "Smoteboost for regression: Improving the prediction of extreme values," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2018, pp. 150–159. [Cited on page 15.]

[61] S. El-Amir and I. El-Henawy, "An improved model using oversampling technique and cost-sensitive learning for imbalanced data problem," *Information Sciences with Applications*, vol. 2, pp. 33–50, 03 2024. [Cited on pages 17, 20, and 21.]

[62] Master's in Data Science, "What is undersampling?" 2022, accessed: 2025-01-21. [Online]. Available: https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/ [Cited on page 17.]

[63] S. S. Lee, "Noisy replication in skewed binary classification," *Computational Statistics Data Analysis*, vol. 34, no. 2, pp. 165–191, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016794739900095X [Cited on page 19.]

[64] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, “Smote: Synthetic minority over-sampling technique,” *J. Artif. Intell. Res. (JAIR)*, vol. 16, pp. 321–357, 06 2002. [Cited on pages 20, 26, 32, and 34.]

[65] L. Torgo, R. P. Ribeiro, B. Pfahringer, and P. Branco, “Smote for regression,” in *Progress in Artificial Intelligence*, L. Correia, L. P. Reis, and J. Cascalho, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 378–389. [Cited on pages 20, 21, 29, 33, 39, 64, and 65.]

[66] J. C. Olamendy, “How to tackle unbalanced data with smote: A comprehensive guide,” 2023, accessed: 2025-01-21. [Online]. Available: https://medium.com/@juanc.olamendy/how-to-tackle-unbalanced-data-with-smote-a-comprehensive-guide-706347ad37ad [Cited on page 21.]

[67] X. Y. Song, N. Dao, and P. Branco, “Distsmogn: Distributed smogn for imbalanced regression problems,” in *Proceedings of the Fourth International Workshop on Learning with Imbalanced Domains: Theory and Applications*, ser. Proceedings of Machine Learning Research, N. Moniz, P. Branco, L. Torgo, N. Japkowicz, M. Wozniak, and S. Wang, Eds., vol. 183. PMLR, 23 Sep 2022, pp. 38–52. [Online]. Available: https://proceedings.mlr.press/v183/song22a.html [Cited on page 22.]

[68] L. Camacho, G. Douzas, and F. Bacao, “Geometric smote for regression,” *Expert Systems with Applications*, vol. 193, p. 116387, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742101678X [Cited on pages ix, 22, 23, 29, 37, 38, 39, 64, and 66.]

[69] G. Douzas and F. Bacao, “Geometric smote a geometrically enhanced drop-in replacement for smote,” *Information Sciences*, vol. 501, pp. 118–135, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519305353 [Cited on page 22.]

[70] J. Wang, “Variational autoencoder (vae),” 2024, accessed: 2025-02-05. [Online]. Available: https://medium.com/@jimwang3589/variational-autoencoder-vae-7609893c80f4 [Cited on page 25.]

[71] I. Severinsen, W. Yu, T. Walmsley, and B. Young, “Covert: A classless approach to generating balanced datasets for process modelling,” *ISA Transactions*, vol.

144, pp. 1–10, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019057823004925 [Cited on pages 25, 26, 29, 38, 64, and 66.]

[72] A. A. Course, “Bootstrapping in machine learning,” 2024, accessed on 2025-02-14. [Online]. Available: https://www.appliedaicourse.com/blog/bootstrapping-in-machine-learning/ [Cited on page 26.]

[73] S. Stocksieker, D. Pommeret, and A. Charpentier, “Data augmentation for imbalanced regression,” 2023. [Online]. Available: https://arxiv.org/abs/2302.09288 [Cited on pages 26, 27, 29, 38, 64, and 66.]

[74] S. S. Lee, “Noisy replication in skewed binary classification,” *Computational statistics & data analysis*, vol. 34, no. 2, pp. 165–191, 2000. [Cited on page 26.]

[75] G. Menardi and N. Torelli, “Training and assessing classification rules with imbalanced data,” *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 92–122, 2014. [Online]. Available: https://doi.org/10.1007/s10618-012-0295-5 [Cited on page 26.]

[76] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, “Modeling tabular data using conditional gan,” 2019. [Online]. Available: https://arxiv.org/abs/1907.00503 [Cited on pages 26 and 32.]

[77] B. Nowok, G. M. Raab, and C. Dibben, “synthpop: Bespoke creation of synthetic data in r,” *Journal of Statistical Software*, vol. 74, no. 11, p. 1–26, 2016. [Online]. Available: https://www.jstatsoft.org/index.php/jss/article/view/v074i11 [Cited on page 27.]

[78] A. Islam and S. B. Belhaouari, “Class aware auto encoders for better feature extraction,” in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, June 2021, pp. 1–5. [Cited on page 28.]

[79] C. Little, R. Allmendinger, and M. Elliot, “Synthetic census microdata generation: A comparative study of synthesis methods examining the trade-off between disclosure risk and utility,” *Journal of Official Statistics*, vol. 0, no. 0, p. 0282423X241266523, 2024. [Online]. Available: https://doi.org/10.1177/0282423X241266523 [Cited on pages 29, 31, and 32.]

[80] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” 2014. [Online]. Available: https://arxiv.org/abs/1406.2661 [Cited on page 29.]

[81] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations, ICLR 2014*, Banff, AB, Canada, April 2014. [Online]. Available: https://arxiv.org/abs/1312.6114 [Cited on page 29.]

[82] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, 1st ed. Chapman and Hall/CRC, 1984. [Cited on page 30.]

[83] J. Reiter, “Using cart to generate partially synthetic, public use microdata,” *Journal of Official Statistics*, vol. 21, 01 2005. [Cited on pages ix, 30, 31, 32, and 50.]

[84] H. Chipman, E. George, and R. McCulloch, “Bayesian cart model search,” *J. Amer. Statist. Assoc.*, vol. 93, 12 1998. [Cited on page 31.]

[85] E. Panagiotou, A. Roy, and E. Ntoutsi, “Synthetic tabular data generation for class imbalance and fairness: A comparative study,” 09 2024. [Cited on pages 31 and 32.]

[86] I. Akiya, T. Ishihara, and K. Yamamoto, “Comparison of synthetic data generation techniques for control group survival data in oncology clinical trials: Simulation study,” *JMIR Med Inform*, vol. 12, p. e55118, Jun 2024. [Online]. Available: https://medinform.jmir.org/2024/1/e55118 [Cited on pages 31 and 32.]

[87] N. Patki, R. Wedge, and K. Veeramachaneni, “The synthetic data vault,” in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2016, pp. 399–410. [Cited on page 32.]

[88] H. He, Y. Bai, E. A. Garcia, and S. Li, “Adasyn: Adaptive synthetic sampling approach for imbalanced learning,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 1322–1328. [Cited on page 34.]

[89] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12,

no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html [Cited on page 40.]

[90] T. Chen, M. Benesty, Y. Tang, N. Zhu, J. Yuan, H. Cho, R. Mitchell, and H. Liu, "Xgboost (python package)," https://pypi.org/project/xgboost/, 2025, apache-2.0 license, released via PyPI. [Cited on page 40.]

[91] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 01 2006. [Cited on pages 45 and 46.]

[92] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *Journal of Machine Learning Research*, vol. 18, no. 77, pp. 1–36, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-305.html [Cited on page 46.]

[93] A. Benavoli, F. Mangili, G. Corani, M. Zaffalon, and F. Ruggeri, "A bayesian wilcoxon signed-rank test based on the dirichlet process," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. II–1026–II–1034. [Cited on page 46.]

[94] A. P. Pinheiro and R. P. Ribeiro, "Synthetic tabular data augmentation for imbalanced regression," https://doi.org/10.24433/CO.7826905.v2, 2025, published: 2025-07-15. [Cited on pages 51 and 96.]

[95] Algorithm Audit, "python-synthpop: A python port of the synthpop framework for synthetic data generation," https://pypi.org/project/python-synthpop/, 2025, published via PyPI. [Cited on page 51.]

[96] L. Torgo, "Expert systems resources," 2015, accessed: 2025-01-21. [Online]. Available: https://www.dcc.fc.up.pt/~ltorgo/ExpertSystems/ [Cited on page 65.]

[97] ——, "Epia 2013 resources," 2013, accessed: 2025-01-21. [Online]. Available: http://www.dcc.fc.up.pt/~ltorgo/EPIA2013 [Cited on page 65.]

[98] P. Branco, "Datasets for imbalanced regression," https://paobranco.github.io/DataSets-IR/, 2020, accessed: 2025-01-25. [Cited on page 65.]

[99] ——, "Pre-processing approaches for imbalance regression," https://github.com/paobranco/Pre-processingApproachesImbalanceRegression, 2020, accessed: 2025-01-25. [Cited on page 65.]

[100] L. Camacho and F. Bacao, “Wsmoter: Source code for imbalanced regression,” 2024, accessed: 2025-01-31. [Online]. Available: https://drive.google.com/drive/folders/1h6Q5sKB5bnqk0Kh01sH1uc6LTp4KSPbb [Cited on page 65.]

[101] S. Stocksieker, “DAVID: A deep learning approach for data analysis and visualization,” 2024, gitHub repository. [Online]. Available: https://github.com/sstocksieker/DAVID [Cited on page 66.]

[102] I. Severinsen, “Covert: Uncertainty estimation for black-box models,” 2023, gitHub repository. [Online]. Available: https://github.com/Isaac-Severinsen/covert [Cited on page 66.]

[103] S. Stocksieker, “Dair - data and ai readiness,” 2024, accessed: 2025-02-10. [Online]. Available: https://github.com/sstocksieker/DAIR [Cited on page 66.]

[104] A. Islam, “Comparesotareg: Comparing state-of-the-art regression models,” 2024, gitHub repository. [Online]. Available: https://github.com/ashhadulislam/CompareSOTAReg/tree/main [Cited on page 67.]

[105] A. P. Pinheiro and R. P. Ribeiro, “Synthetictabulardataaugmentation-ir,” https://github.com/antoniopedropi/SyntheticTabularDataAugmentation-IR, 2025, published: 2025-07-02. [Cited on page 67.]

A. Comparative Analysis of Related Algorithms

Table A.1: Comparison of preprocessing strategies by key design characteristics

Strategy	Neighbour Based	Noise Introduction	Uses Relevance Threshold	Domain Discretisation	Other User-Defined Parameters
RU [18]	✗	✗	✓	✓	✗
RO [17]	✗	✗	✓	✓	✗
WERCS [17]	✗	✗	✗	✗	✓
GN [17]	✗	✓	✓	✓	✓
SMOTER [18, 65]	✓	✗	✓	✓	✓
SMOGN [23]	✓	✓	✓	✓	✓
WSMOTER [20]	✓	✗	✗	✗	✓
G-SMOTER [68]	✓	✗	✓	✓	✓
DIRVAE [14]	✗	✓	✗	✗	✓
IRGAN [15]	✗	✗	✗	✗	✓
DAVID [16]	✗	✓	✗	✗	✓
COVERT [71]	✓	✗	✗	✗	✓
DA + WR [73]	✓	✓	✗	✗	✓
KNNOR-REG [21]	✓	✗	✗	✓	✓
CARTGen-IR	✗	✓	✗	✗	✓

Table A.2: Detailed comparison of preprocessing strategies

Strategy	Description	Software Packages	Advantages	Disadvantages
Random Undersampling (RU) [18]	Performs random undersampling by randomly removing cases from the most common and less relevant target variable values.	[96, 97]	• Balanced dataset • Reduced computational cost	• Loss of information • Bias in low-relevance sample selection • Threshold-dependent • Domain discretisation
Random Oversampling (RO) [17]	Consists of randomly selecting and replicating examples from the relevant bin of cases, while maintaining all the common cases in the ”normal” bin.	[98, 99]	• Maintains all information • Automatic method of determining oversampling percentage	• Larger dataset • Higher likelihood of overfitting • Threshold-dependent • Domain discretisation
WEighted Relevance-based Combination Strategy (WERCS) [17]	Employs an informed under and oversampling strategy, based on the relevance function, which assigns weights to cases that influence their likelihood of being replicated or removed from the dataset.	[98, 99]	• No need for user-defined relevance thresholds • No domain discretisation	• Loss of information • Higher likelihood of overfitting • Need for specification of under and oversampling percentages by user • Simplistic replication of seed cases without incorporating neighbouring information or noise
Introduction of Gaussian Noise (GN) [17]	Combines the undersampling of typical, prevalent cases with the oversampling of rare, pertinent cases, through a synthetic generation process that introduces normally distributed noise to the seed cases.	[98, 99]	• Compatible with both numeric and nominal attributes • Synthetic cases with high relevance • Balanced dataset • Maintains all information • Automatic method of determining under/oversampling percentage	• Threshold-dependent • Domain discretisation • Hard to find optimal amount of noise to add to rare cases
Synthetic Minority Oversampling Technique for Regression (SMOTER) [18, 65]	Performs random undersampling for the low-relevance pool of cases and oversampling for the high-relevance pool of cases using a linear interpolation strategy for the predictors and a weighted average for the target variable values.	[96, 97]	• Balanced dataset • Handles both numeric and nominal variables	• Loss of information • Flawed synthetic sample generation process • Hard to find the optimal value for k • Risk of overfitting • Less effective in high-dimensional data • Threshold-dependent • Domain discretisation
SMOTER with Gaussian Noise (SMOGN) [23]	Combines an undersampling process for ordinary cases and an oversampling strategy for pertinent cases, wherein synthetic cases are generated through SMOTER for seeds with high-proximity cases and through GN for seeds distant from other cases.	[19]	• Mitigates risk of overfitting • Preserves local data distributions • High data fidelity and diversity	• Hard to find the optimal value for k • Threshold-dependent • Domain discretisation
Weighting SMOTE for Regression (WS-MOTER) [20]	This algorithm weights instances by their rarity (via KDE) to probabilistically guide synthetic data generation, ensuring sparse regions are augmented without restrictive assumptions about where rarity occurs.	[100]	• No need for relevance thresholds • No need for splitting data into bins or partitions • No assumptions about the location of rare data points	• Larger dataset • Hard to find the optimal value for k

(Continued on the next page)

Strategy	Description	Software Packages	Advantages	Disadvantages
Geometric SMOTE for Regression (G-SMOTER) [68]	Generates synthetic instances within a transformed hyperspherical region and assigns labels via weighted interpolation, ensuring alignment with target distributions through three distinct selection strategies for the data generation process.	-[1]	• Mitigates overfitting risk • Agnostic to relevance methods • Expands the sample generation region • Diversity of generated instances • Enhances generalisation capability of SMOTER	• Threshold-dependent • Domain discretisation
Deep Imbalanced Regression Variational Autoencoder (DIRVAE) [14]	Synthesises data using a GAN-based framework with a dual-model architecture: a VAE for latent distribution and an Long Short-Term Memory (LSTM) for learning neighbouring sample relationships. A discriminator ensures generated samples match the original data distribution. It categorises imbalanced regression into three distribution types to simplify target value distributions, effectively generating minority samples.	-[1]	• Synthetic sample generation is more similar to the original data distribution • Increased diversity in sample generation • Considers the characteristics of neighbouring samples but also captures the spatial distribution features of the original samples • Mitigates the risk of overfitting	• Dependent on specific distribution shapes • Not suitable for more diverse or complex distribution characteristics • Black-box model
Imbalanced Regression Generative Adversarial Network (IRGAN) [15]	Integrates generative, correction and discriminative modules, generating synthetic samples to balance datasets. A correction module guides sample generation, reducing training instability, and optimising adversarial learning.	-[1]	• Works in high-dimensional data • Handles non-independent or non-identical distribution features • Correction network improves the similarity and authenticity of generated samples	• Black-box model • Vanishing gradient problem
DAVID [16]	Generates synthetic data by combining VAEs with a smoothed bootstrap technique, using a balanced loss function that inversely weights samples based on their density via a kernel estimator.	[101]	• More pertinent data generation • No domain discretisation • Captures non-linear correlations • Non-parametric model • Suitable for complex data distributions	• Increased computational cost • Sensitivity to noise parameters
COVERT [71]	Designed to balance imbalanced datasets in process modelling by using KDE to identify sparse regions in the data and iteratively generate synthetic samples to create a more uniform distribution.	[102]	• Works for highly dimensional feature spaces • No domain discretisation or relevance threshold • Leverages uncertainty information relevant to the dataset • High adaptability due to its iterative nature	• Applicable only to industrial data • High computational cost • Focus on uniform target domain relevance • Not compatible with highly quantised, categorical or boolean attributes
Data Augmentation + Weighted Relevance (DA+WR) [73]	This method integrates a data augmentation technique, encompassing various strategies such as GN and CGAN, with a weighted approach that calculates resampling probabilities based on weights for specific cases.	[103]	• Mitigates overfitting risk • Distinctive separation between the resampling and augmentation phases • No complex parametrisation is required • Addresses the issue of selection bias in sampling	• Addresses covariate imbalance rather than the target variable imbalance • Generated values exhibit excessive similarity to the imbalanced sample • Direct adaptation of existing data augmentation imbalanced classification methods to imbalanced regression • Sensitivity of the data generation method

[1] The authors did not publicly disclose any data, software package, or implementation code. Even after direct requests to the authors, they did not make it available.

(Continued on the next page)

Strategy	Description	Software Packages	Advantages	Disadvantages
k-Nearest Neighbour Oversampling - Regression (KNNOR-Reg) [21]	Leverages a relevance function and integrates an enhanced SMOTE variant with k-NN to generate synthetic data points, with the addition of deep learning-based techniques (Target-Aware Autoencoder and AutoInflater) for high-population data.	[104]	• Versatility in terms of feature dimensionality and population data • Addresses noisy and ambiguous samples • Incorporates non-linearity • Reduces risk of information loss or distortion • No need for user-defined relevance thresholds	• Hard to find the optimal value for k • Domain discretisation based on user-defined bins • Target Aware Autoencoder/AutoInflater are black-box models
CARTGen-IR	Generates synthetic samples using CART-based column-wise synthesis, guided by a rarity-weighted resampling strategy that emphasises rare target values without relying on discrete thresholds or domain discretisation.	[105]	• Versatility in terms of feature dimensionality, missing data and complexity • High computational efficiency • Incorporates non-linearity • Interpretability • No need for user-defined relevance thresholds and domain discretisation	• Hard to find the optimal value for hyperparameters • Suboptimal performance for partition boundaries • Not recommended for data described by parametric relationships

B. Experimental Results

B.1. Data-Level Algorithms Variants and Hyperparameters

Table B.1: Preprocessing strategy variants and respective hyperparameters

Strategy	Variant	Hyperparameters Combination
RO/RU	v1	(%u / %o = 'balance')
	v2	(%u / %o = 'extreme')
WERCS	v1	(%u = 0.5, %o = 0.5)
	v2	(%u = 0.5, %o = 0.75)
	v3	(%u = 0.75, %o = 0.5)
	v4	(%u = 0.75, %o = 0.75)
GN	v1	(%u/%o = 'balance', δ = 0.05)
	v2	(%u/%o = 'balance', δ = 0.1)
	v3	(%u/%o = 'balance', δ = 0.5)
	v4	(%u/%o = 'extreme', δ = 0.05)
	v5	(%u/%o = 'extreme', δ = 0.1)
	v6	(%u/%o = 'extreme', δ = 0.5)
SMOTER	v1	(%u/%o = 'balance', k = 5)
	v2	(%u/%o = 'extreme', k = 5)
SMOGN	v1	(%u/%o = 'balance', k = 5, δ = 0.05)
	v2	(%u/%o = 'balance', k = 5, δ = 0.1)
	v3	(%u/%o = 'balance', k = 5, δ = 0.5)
	v4	(%u/%o = 'extreme', k = 5, δ = 0.05)
	v5	(%u/%o = 'extreme', k = 5, δ = 0.1)
	v6	(%u/%o = 'extreme', k = 5, δ = 0.5)
WSMOTER	v1	(β = 1, ratio = 1.5, k = 5)
	v2	(β = 1, ratio = 1.75, k = 5)
	v3	(β = 2, ratio = 1.5, k = 5)
	v4	(β = 2, ratio = 1.75, k = 5)
G-SMOTER	v1	(ξ = 0.7, k = 5, η = 0.75, strategy = 'combined', θ = -0.5)
	v2	(ξ = 0.7, k = 5, η = 0.75, strategy = 'combined', θ = 0.5)
	v3	(ξ = 0.7, k = 5, η = 0.75, strategy = 'majority', θ = -0.5)
	v4	(ξ = 0.7, k = 5, η = 0.75, strategy = 'majority', θ = 0.5)
	v5	(ξ = 0.7, k = 5, η = 0.75, strategy = 'minority', θ = -0.5)
	v6	(ξ = 0.7, k = 5, η = 0.75, strategy = 'minority', θ = 0.5)
DAVID	v1	(α = 1)
	v2	(α = 2)
KNNOR-REG	v1	-

(Continued on the next page)

Strategy	Variant	Hyperparameters Combination
CARTGen-IR	v1	(α = 1, density = 'denseweight', δ = 0.00, η = 'balance')
	v2	(α = 1, density = 'denseweight', δ = 0.00, η = 'extreme')
	v3	(α = 1, density = 'denseweight', δ = 0.01, η = 'balance')
	v4	(α = 1, density = 'denseweight', δ = 0.01, η = 'extreme')
	v5	(α = 1, density = 'kde_baseline', δ = 0.00, η = 'balance')
	v6	(α = 1, density = 'kde_baseline', δ = 0.00, η = 'extreme')
	v7	(α = 1, density = 'kde_baseline', δ = 0.01, η = 'balance')
	v8	(α = 1, density = 'kde_baseline', δ = 0.01, η = 'extreme')
	v9	(α = 1, density = 'relevance', δ = 0.00, η = 'balance')
	v10	(α = 1, density = 'relevance', δ = 0.00, η = 'extreme')
	v11	(α = 1, density = 'relevance', δ = 0.01, η = 'balance')
	v12	(α = 1, density = 'relevance', δ = 0.01, η = 'extreme')
	v13	(α = 1.5, density = 'denseweight', δ = 0.00, η = 'balance')
	v14	(α = 1.5, density = 'denseweight', δ = 0.00, η = 'extreme')
	v15	(α = 1.5, density = 'denseweight', δ = 0.01, η = 'balance')
	v16	(α = 1.5, density = 'denseweight', δ = 0.01, η = 'extreme')
	v17	(α = 1.5, density = 'kde_baseline', δ = 0.00, η = 'balance')
	v18	(α = 1.5, density = 'kde_baseline', δ = 0.00, η = 'extreme')
	v19	(α = 1.5, density = 'kde_baseline', δ = 0.01, η = 'balance')
	v20	(α = 1.5, density = 'kde_baseline', δ = 0.01, η = 'extreme')
	v21	(α = 1.5, density = 'relevance', δ = 0.00, η = 'balance')
	v22	(α = 1.5, density = 'relevance', δ = 0.00, η = 'extreme')
	v23	(α = 1.5, density = 'relevance', δ = 0.01, η = 'balance')
	v24	(α = 1.5, density = 'relevance', δ = 0.01, η = 'extreme')
	v25	(α = 2.0, density = 'denseweight', δ = 0.00, η = 'balance')
	v26	(α = 2.0, density = 'denseweight', δ = 0.00, η = 'extreme')
	v27	(α = 2.0, density = 'denseweight', δ = 0.01, η = 'balance')
	v28	(α = 2.0, density = 'denseweight', δ = 0.01, η = 'extreme')
	v29	(α = 2.0, density = 'kde_baseline', δ = 0.00, η = 'balance')
	v30	(α = 2.0, density = 'kde_baseline', δ = 0.00, η = 'extreme')
	v31	(α = 2.0, density = 'kde_baseline', δ = 0.01, η = 'balance')
	v32	(α = 2.0, density = 'kde_baseline', δ = 0.01, η = 'extreme')
	v33	(α = 2.0, density = 'relevance', δ = 0.00, η = 'balance')
	v34	(α = 2.0, density = 'relevance', δ = 0.00, η = 'extreme')
	v35	(α = 2.0, density = 'relevance', δ = 0.01, η = 'balance')
	v36	(α = 2.0, density = 'relevance', δ = 0.01, η = 'extreme')

Table B.2: Best RMSE scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue.

Learner	Strategy	abalone	airfoil	availablePower	cpuSm	ele-1	forestFires	fuelConsumption	heat	housingBoston	maxTorque	mortgage	sensory	servo	strikes	treasury
RF	None	**2.157 (0.087)**	1.774 (0.091)	6.322 (1.884)	2.826 (0.102)	646.809 (78.54)	55.657 (36.566)	0.486 (0.134)	2.265 (0.177)	3.236 (0.64)	12.28 (2.774)	0.136 (0.019)	0.717 (0.038)	0.607 (0.129)	506.85 (143.141)	0.217 (0.043)
	RU	2.897 (0.604)	9.907 (0.307)	26.469 (1.298)	25.323 (2.222)	1079.04 (110.776)	225.951 (47.534)	1.42 (0.135)	13.573 (0.813)	8.676 (2.178)	56.236 (4.354)	0.653 (0.058)	0.973 (0.222)	1.09 (0.31)	2337.521 (234.62)	0.657 (0.054)
	RO	2.181 (0.089)	1.87 (0.114)	**6.045 (1.713)**	2.861 (0.13)	651.264 (82.385)	62.492 (33.583)	0.484 (0.14)	2.218 (0.205)	3.21 (0.556)	11.43 (2.457)	0.138 (0.02)	0.721 (0.034)	0.558 (0.124)	520.809 (146.245)	0.219 (0.043)
	WERCS	2.301 (0.075)	2.612 (0.208)	6.968 (1.969)	2.915 (0.094)	666.804 (69.497)	71.909 (27.18)	0.521 (0.118)	2.844 (0.155)	3.239 (0.48)	13.362 (3.069)	0.199 (0.017)	0.745 (0.051)	0.736 (0.16)	631.342 (133.774)	0.284 (0.106)
	GN	2.324 (0.072)	2.504 (0.167)	7.998 (1.604)	2.936 (0.078)	743.947 (100.74)	77.188 (26.563)	0.559 (0.122)	3.71 (0.185)	3.494 (0.4)	15.804 (3.339)	0.169 (0.027)	0.744 (0.051)	0.6 (0.117)	561.74 (128.888)	0.252 (0.042)
	SMOTER	2.561 (0.117)	1.87 (0.128)	6.608 (2.03)	2.882 (0.14)	700.95 (103.09)	59.493 (35.276)	0.484 (0.131)	2.189 (0.205)	3.247 (0.561)	11.924 (1.971)	0.153 (0.026)	0.736 (0.044)	0.546 (0.145)	508.507 (140.601)	0.235 (0.045)
	SMOGN	2.611 (0.166)	1.98 (0.109)	7.061 (2.992)	2.838 (0.089)	710.91 (110.186)	67.621 (32.052)	0.482 (0.137)	2.213 (0.204)	3.215 (0.512)	11.127 (1.657)	0.166 (0.019)	0.725 (0.032)	0.553 (0.123)	525.012 (143.887)	0.247 (0.041)
	WSMOTER	2.206 (0.083)	**1.742 (0.124)**	6.122 (1.81)	2.81 (0.087)	656.571 (84.306)	61.585 (33.542)	0.472 (0.137)	2.104 (0.168)	**3.037 (0.499)**	11.517 (2.808)	**0.133 (0.019)**	0.712 (0.029)	0.559 (0.145)	511.707 (139.193)	0.202 (0.035)
	G-SMOTER	2.158 (0.09)	1.763 (0.096)	6.256 (1.908)	2.793 (0.086)	641.887 (76.037)	55.873 (37.16)	0.481 (0.141)	2.127 (0.186)	3.146 (0.576)	11.529 (3.092)	0.135 (0.022)	0.713 (0.026)	0.526 (0.12)	507.028 (139.136)	0.215 (0.041)
	DAVID	2.313 (0.095)	4.113 (0.233)	39.369 (8.618)	3.608 (0.159)	645.271 (72.465)	77.981 (25.867)	0.666 (0.355)	27.501 (1.204)	3.953 (0.818)	24.59 (26.382)	0.197 (0.022)	0.768 (0.046)	1.701 (0.26)	639.218 (133)	0.297 (0.072)
	KNNOR-REG	2.175 (0.077)	1.786 (0.11)	6.238 (1.895)	2.826 (0.105)	**641.016 (76.971)**	56.116 (36.575)	0.486 (0.14)	2.17 (0.168)	3.237 (0.627)	12.533 (3.054)	0.137 (0.019)	0.721 (0.037)	0.607 (0.165)	**507.018 (141.403)**	0.215 (0.038)
	CARTGen-IR	2.161 (0.087)	1.778 (0.088)	6.05 (1.828)	**2.777 (0.063)**	649.240 (85.39)	58.644 (35.935)	**0.471 (0.135)**	**1.949 (0.178)**	3.045 (0.546)	**11.06 (3.324)**	0.137 (0.019)	**0.709 (0.029)**	**0.497 (0.12)**	507.314 (142.911)	**0.199 (0.035)**
SVR	None	**2.118 (0.084)**	2.789 (0.155)	8.216 (1.551)	3.481 (0.21)	736.059 (125.845)	50.486 (42.224)	0.592 (0.153)	0.754 (0.227)	3.234 (0.678)	14.09 (5.586)	0.066 (0.008)	0.769 (0.054)	0.478 (0.072)	518.313 (126.646)	0.183 (0.032)
	RU	2.403 (0.095)	6.298 (0.391)	18.158 (1.758)	8.118 (0.317)	1036.317 (93.201)	50.028 (41.956)	0.951 (0.108)	6.726 (0.229)	5.403 (0.757)	30.91 (3.156)	1.376 (0.194)	0.947 (0.152)	1.013 (0.253)	527.223 (126.247)	1.632 (0.209)
	RO	2.399 (0.157)	3.74 (0.336)	7.727 (1.385)	3.584 (0.244)	764.579 (100.299)	50.753 (39.733)	0.603 (0.142)	**0.441 (0.057)**	3.291 (0.558)	**10.886 (1.964)**	**0.066 (0.007)**	0.813 (0.055)	**0.465 (0.076)**	533.552 (138.892)	0.184 (0.031)
	WERCS	2.269 (0.067)	3.423 (0.264)	8.529 (1.175)	3.398 (0.183)	715.259 (65.68)	56.143 (36.115)	0.629 (0.128)	0.502 (0.104)	3.621 (0.616)	13.478 (3.21)	0.097 (0.016)	0.893 (0.086)	0.714 (0.136)	562.514 (135.315)	0.213 (0.04)
	GN	2.389 (0.083)	4.324 (0.397)	9.633 (1.717)	4.353 (0.522)	867.891 (88.391)	61.972 (33.534)	0.668 (0.116)	1.621 (0.219)	3.714 (0.525)	15.745 (2.889)	0.082 (0.012)	0.877 (0.074)	0.589 (0.106)	569.876 (145.308)	0.201 (0.038)
	SMOTER	2.531 (0.157)	3.749 (0.335)	11.929 (1.266)	3.557 (0.265)	733.421 (75.814)	51.306 (39.317)	0.607 (0.141)	0.547 (0.106)	3.399 (0.617)	14.604 (4.146)	0.066 (0.008)	0.85 (0.089)	0.483 (0.096)	531.828 (151.555)	0.184 (0.032)
	SMOGN	2.871 (0.355)	3.795 (0.337)	13.539 (1.848)	3.751 (0.275)	816.114 (105.264)	57.019 (35.126)	0.607 (0.15)	0.544 (0.106)	3.4 (0.558)	15.426 (5.098)	0.076 (0.023)	0.816 (0.055)	0.5 (0.099)	546.138 (156.987)	0.188 (0.031)
	WSMOTER	2.127 (0.07)	2.786 (0.171)	**7.688 (1.373)**	3.422 (0.216)	666.744 (115.59)	50.207 (42.222)	0.598 (0.146)	0.509 (0.158)	**3.128 (0.5)**	12.716 (3.592)	0.066 (0.009)	0.807 (0.047)	0.466 (0.073)	503.601 (133.424)	**0.182 (0.031)**
	G-SMOTER	2.113 (0.084)	**2.785 (0.184)**	8.103 (1.467)	3.391 (0.185)	698.015 (120.957)	50.378 (42.188)	0.591 (0.146)	0.619 (0.188)	3.179 (0.58)	12.311 (3.068)	0.066 (0.008)	0.784 (0.052)	0.488 (0.078)	514.786 (127.707)	0.181 (0.033)
	DAVID	3.306 (0.33)	5.14 (0.519)	42.333 (7.702)	4.841 (0.519)	736.059 (125.845)	149.361 (37.514)	0.757 (0.367)	22.273 (7.918)	3.785 (0.56)	28.259 (28.828)	0.133 (0.014)	0.926 (0.086)	1.686 (0.263)	523.404 (119.47)	0.26 (0.064)
	KNNOR-REG	2.143 (0.08)	2.842 (0.142)	8.328 (1.631)	3.363 (0.205)	698.7 (132.074)	50.48 (42.297)	0.591 (0.144)	0.821 (0.21)	3.222 (0.712)	14.825 (5.008)	0.07 (0.009)	**0.782 (0.052)**	0.485 (0.074)	507.162 (134.736)	0.187 (0.035)
	CARTGen-IR	2.141 (0.078)	2.871 (0.156)	8.458 (1.584)	**3.209 (0.174)**	**643.363 (70.038)**	**49.285 (41.758)**	**0.585 (0.13)**	0.692 (0.122)	3.237 (0.56)	12.738 (2.314)	0.068 (0.008)	0.788 (0.052)	0.483 (0.083)	**503.230 (133.99)**	0.192 (0.032)
XGBoost	None	2.199 (0.075)	1.45 (0.108)	6.078 (1.546)	2.766 (0.171)	727.643 (101.967)	63.309 (33.893)	0.462 (0.107)	**1.203 (0.057)**	3.222 (0.642)	5.742 (2.153)	0.142 (0.021)	0.722 (0.04)	0.298 (0.073)	571.419 (152.277)	0.245 (0.027)
	RU	3.135 (0.421)	7.915 (0.217)	22.452 (2.441)	6.001 (1.868)	996.917 (67.866)	325.664 (123.877)	1.571 (0.297)	10.357 (0.979)	9.773 (2.491)	46.341 (6.211)	0.495 (0.046)	1.293 (0.131)	1.481 (0.683)	2799.445 (346.28)	0.541 (0.097)
	RO	2.339 (0.08)	1.528 (0.177)	5.921 (1.763)	2.835 (0.188)	715.421 (104.689)	65.558 (37.552)	0.468 (0.125)	1.352 (0.05)	3.227 (0.534)	6.925 (1.975)	0.14 (0.016)	0.729 (0.034)	**0.298 (0.077)**	587.919 (162.925)	0.234 (0.036)
	WERCS	2.452 (0.08)	2.076 (0.204)	7.67 (1.65)	2.98 (0.152)	733.381 (93.748)	90.426 (29.594)	0.557 (0.1)	1.823 (0.072)	3.289 (0.532)	7.981 (2.107)	0.212 (0.038)	0.805 (0.05)	0.652 (0.319)	700.592 (117.604)	0.311 (0.069)
	GN	2.348 (0.086)	2.22 (0.261)	7.801 (1.656)	2.99 (0.073)	797.44 (113.26)	88.621 (26.068)	0.598 (0.119)	3.043 (0.158)	3.866 (0.552)	12.211 (3.728)	0.178 (0.028)	0.789 (0.08)	0.336 (0.084)	701.053 (160.419)	0.271 (0.049)
	SMOTER	2.701 (0.132)	1.594 (0.231)	7.359 (2.044)	2.916 (0.069)	750.828 (108.07)	68.845 (28.836)	0.486 (0.123)	1.604 (0.133)	3.503 (0.704)	9.041 (4.452)	0.164 (0.02)	0.762 (0.047)	0.358 (0.097)	583.903 (145.187)	0.272 (0.046)
	SMOGN	2.689 (0.12)	1.721 (0.183)	7.445 (1.785)	2.99 (0.08)	776.211 (109.248)	83.963 (30.147)	0.477 (0.13)	1.627 (0.086)	3.455 (0.688)	7.859 (2.963)	0.178 (0.023)	0.715 (0.031)	0.305 (0.117)	604.471 (159.955)	0.29 (0.036)
	WSMOTER	2.252 (0.075)	1.514 (0.13)	5.807 (1.51)	2.733 (0.054)	712.924 (97.447)	70.059 (37.382)	0.454 (0.127)	1.5 (0.054)	**2.976 (0.444)**	**5.39 (1.261)**	0.135 (0.023)	0.719 (0.039)	0.31 (0.07)	551.085 (169.158)	**0.213 (0.043)**
	G-SMOTER	2.188 (0.076)	**1.421 (0.127)**	5.681 (1.206)	**2.721 (0.068)**	710.517 (90.44)	62.276 (35.402)	0.455 (0.128)	1.302 (0.077)	2.979 (0.541)	6.338 (2.354)	**0.133 (0.018)**	0.705 (0.032)	0.305 (0.077)	567.979 (137.351)	0.23 (0.041)
	DAVID	2.452 (0.108)	3.709 (0.33)	35.467 (8.861)	3.668 (0.27)	727.643 (101.967)	99.689 (46.34)	0.624 (0.297)	15.342 (1.739)	4.017 (0.882)	17.015 (24.148)	0.21 (0.029)	0.817 (0.059)	1.701 (0.247)	729.166 (165.934)	0.301 (0.063)
	KNNOR-REG	2.232 (0.069)	1.448 (0.104)	5.658 (1.734)	2.75 (0.054)	713.414 (85.503)	64.284 (33.102)	0.474 (0.137)	1.513 (0.098)	3.116 (0.575)	6.707 (2.628)	0.143 (0.015)	0.721 (0.034)	0.298 (0.084)	563.362 (140.201)	0.238 (0.048)
	CARTGen-IR	**2.181 (0.075)**	1.488 (0.117)	**5.303 (1.503)**	2.771 (0.08)	**696.308 (84.072)**	**61.971 (37.88)**	**0.442 (0.112)**	1.8 (0.077)	3.11 (0.442)	6.595 (2.144)	0.141 (0.02)	**0.69 (0.04)**	0.311 (0.056)	**547.091 (139.975)**	0.230 (0.056)

Table B.3: Best RW-RMSE scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue.

Learner	Strategy	abalone	airfoil	availablePower	cpuSm	ele-1	forestFires	fuelConsumption	heat	housingBoston	maxTorque	mortgage	sensory	servo	strikes	treasury
RF	None	2.778 (0.167)	1.869 (0.293)	10.717 (5.699)	4.014 (0.553)	829.122 (371.593)	202.643 (145.721)	0.743 (0.4)	3.083 (0.717)	3.65 (0.869)	18.333 (9.397)	0.066 (0.013)	1.018 (0.097)	0.533 (0.33)	1730.972 (363.864)	0.174 (0.131)
	RU	3.04 (0.702)	5.182 (3.675)	22.117 (6.964)	12.219 (5.504)	1081.435 (148.981)	238.235 (88.858)	1.316 (0.247)	10.759 (2.166)	7.342 (2.269)	47.947 (22.273)	0.641 (0.127)	1.263 (0.389)	1.04 (0.782)	1657.321 (176.062)	0.633 (0.114)
	RO	2.744 (0.17)	1.871 (0.315)	9.778 (5.138)	4.165 (0.719)	802.176 (354.137)	199.648 (145.895)	0.729 (0.399)	3.029 (0.68)	3.598 (0.825)	16.631 (8.159)	0.063 (0.01)	1.005 (0.122)	0.495 (0.274)	1726.502 (374.669)	0.173 (0.127)
	WERCS	**2.537 (0.107)**	1.909 (0.915)	9.926 (5.811)	3.676 (0.389)	795.194 (335.743)	**185.06 (140.214)**	**0.671 (0.377)**	2.916 (0.623)	3.483 (0.855)	16.873 (9.273)	0.063 (0.01)	1.01 (0.137)	0.673 (0.451)	**1621.218 (400.789)**	0.265 (0.343)
	GN	2.619 (0.138)	2.248 (0.447)	11.101 (5.56)	4.028 (0.501)	762.669 (340.907)	188.98 (148.155)	0.76 (0.384)	4.259 (1.045)	3.671 (0.934)	21.696 (13.888)	0.078 (0.015)	0.981 (0.137)	0.503 (0.232)	1701.047 (378.476)	0.204 (0.149)
	SMOTER	2.738 (0.213)	1.863 (0.291)	**9.481 (3.742)**	4.268 (0.763)	782.471 (336.469)	197.622 (147.312)	0.724 (0.398)	2.955 (0.679)	3.564 (0.739)	18.797 (11.292)	0.129 (0.051)	1.073 (0.131)	0.493 (0.296)	1726.165 (364.242)	0.253 (0.086)
	SMOGN	2.779 (0.25)	1.947 (0.292)	10.245 (7.264)	3.991 (0.443)	760.554 (341.624)	193.565 (146.612)	0.671 (0.377)	2.983 (0.609)	3.514 (0.742)	16.803 (8.648)	0.139 (0.039)	1.014 (0.11)	0.473 (0.282)	1713.167 (367.849)	0.268 (0.093)
	WSMOTER	2.638 (0.175)	**1.813 (0.36)**	10.182 (5.314)	3.778 (0.485)	781.626 (337.697)	197.737 (146.218)	0.703 (0.401)	2.769 (0.608)	3.337 (0.625)	17.235 (9.008)	**0.059 (0.012)**	0.963 (0.126)	0.489 (0.298)	1718.533 (371.403)	0.16 (0.117)
	G-SMOTER	2.755 (0.164)	1.845 (0.283)	10.354 (5.642)	3.895 (0.548)	790.756 (343.329)	200.671 (144.599)	0.726 (0.398)	2.858 (0.623)	3.457 (0.854)	17.387 (9.034)	0.063 (0.013)	0.979 (0.128)	0.448 (0.287)	1716.164 (358.437)	0.169 (0.12)
	DAVID	3.112 (0.231)	5.087 (0.727)	68.533 (35.766)	5.792 (0.992)	820.955 (363.69)	195.877 (140.247)	1.058 (0.635)	39.636 (12.43)	4.605 (1.045)	45.672 (65.65)	0.107 (0.022)	1.089 (0.13)	1.174 (1.225)	1641.071 (387.724)	0.249 (0.139)
	KNNOR-REG	2.711 (0.163)	1.839 (0.281)	10.651 (5.462)	3.913 (0.513)	810.077 (353.151)	201.724 (145.162)	0.738 (0.406)	2.926 (0.584)	3.577 (0.856)	19.098 (9.721)	0.066 (0.012)	1.004 (0.101)	0.522 (0.313)	1717.077 (383.916)	0.174 (0.127)
	CARTGen-IR	2.690 (0.145)	1.842 (0.333)	9.831 (5.5)	**3.631 (0.436)**	**672.071 (262.872)**	190.003 (144.594)	0.688 (0.406)	**2.606 (0.592)**	**3.299 (0.863)**	**15.498 (8.073)**	0.061 (0.013)	**0.948 (0.15)**	**0.418 (0.244)**	1629.396 (343.249)	**0.151 (0.11)**
SVR	None	2.786 (0.149)	3.458 (0.704)	12.749 (4.45)	6.021 (1.33)	1138.685 (599.032)	212.25 (144.593)	0.9 (0.402)	1.402 (0.636)	3.614 (1.033)	20.616 (12.343)	0.033 (0.004)	1.029 (0.18)	0.439 (0.14)	1829.381 (318.857)	0.154 (0.086)
	RU	2.849 (0.257)	3.939 (2.371)	16.353 (4.232)	8.617 (1.441)	1449.322 (249.568)	188.064 (122.603)	1.108 (0.211)	5.963 (2.066)	5.396 (0.855)	40.559 (21.023)	0.282 (0.116)	1.365 (0.194)	1.028 (0.763)	1536.925 (309.118)	1.2 (1.41)
	RO	2.638 (0.179)	3.632 (0.599)	**11.446 (4.361)**	5.735 (1.041)	**720.32 (277.912)**	186.487 (142.406)	0.894 (0.397)	**0.671 (0.117)**	3.522 (0.983)	**12.529 (4.876)**	0.032 (0.003)	1.016 (0.196)	0.387 (0.16)	1439.71 (332.128)	0.157 (0.084)
	WERCS	2.516 (0.189)	**2.196 (0.784)**	11.945 (3.786)	4.239 (0.587)	791.668 (246.73)	**178.177 (141.263)**	0.857 (0.355)	0.689 (0.375)	3.921 (1.009)	15.767 (7.686)	**0.031 (0.003)**	1.063 (0.232)	0.636 (0.356)	1468.433 (340.399)	0.184 (0.148)
	GN	2.644 (0.178)	3.561 (1.063)	13.241 (4.864)	5.847 (0.955)	760.188 (245.113)	182.057 (144.429)	0.915 (0.349)	2.496 (0.691)	3.941 (0.865)	20.442 (10.857)	0.05 (0.01)	1.018 (0.113)	0.515 (0.198)	1431.701 (332.061)	0.167 (0.094)
	SMOTER	2.667 (0.228)	3.58 (0.6)	17.628 (5.636)	5.594 (1.023)	739.805 (293.592)	189.302 (143.316)	0.9 (0.376)	0.919 (0.238)	3.611 (1.085)	19.501 (8.368)	0.032 (0.004)	1.129 (0.254)	0.4 (0.216)	1437.286 (330.412)	0.157 (0.096)
	SMOGN	2.811 (0.29)	3.518 (0.627)	19.047 (6.278)	5.644 (0.82)	722.55 (274.979)	183.349 (144.711)	0.89 (0.382)	0.914 (0.241)	3.577 (1.002)	16.805 (8.586)	0.032 (0.004)	1.031 (0.123)	0.412 (0.196)	**1429.1 (335.515)**	0.16 (0.087)
	WSMOTER	2.503 (0.133)	3.171 (0.717)	11.64 (4.237)	5.558 (1.263)	951.804 (490.344)	203.598 (147.167)	0.907 (0.391)	0.886 (0.388)	3.511 (0.882)	17.361 (8.413)	0.032 (0.002)	0.982 (0.192)	0.401 (0.226)	1782.977 (334.862)	**0.15 (0.078)**
	G-SMOTER	2.681 (0.161)	3.256 (0.646)	12.537 (4.642)	5.483 (1.138)	1065.396 (548.715)	211.461 (144.689)	0.886 (0.394)	1.097 (0.448)	3.514 (1.028)	16.493 (7.293)	0.034 (0.004)	0.993 (0.187)	0.425 (0.263)	1819.209 (322.482)	0.153 (0.087)
	DAVID	3.352 (0.316)	4.925 (0.838)	74.915 (40.359)	7.558 (1.235)	1138.685 (599.032)	209.139 (110.381)	1.2 (0.665)	31.314 (9.895)	4.345 (1.089)	52.33 (73.147)	0.077 (0.013)	1.1 (0.16)	1.163 (1.215)	1448.585 (328.368)	0.222 (0.118)
	KNNOR-REG	2.603 (0.14)	3.378 (0.658)	12.86 (4.747)	5.508 (1.228)	1037.281 (534.642)	211.601 (144.67)	0.89 (0.399)	1.528 (0.568)	3.52 (1.034)	21.275 (12.001)	0.035 (0.004)	1.025 (0.187)	0.437 (0.306)	1790.726 (333.022)	0.159 (0.097)
	CARTGen-IR	**2.468 (0.132)**	2.401 (0.128)	12.693 (3.431)	**4.122 (0.681)**	720.549 (304.087)	184.155 (141.16)	**0.812 (0.345)**	1.015 (0.314)	**3.347 (0.922)**	15.545 (5.803)	0.036 (0.01)	**0.971 (0.164)**	**0.377 (0.221)**	1616.316 (324.765)	0.158 (0.084)
XGBoost	None	2.797 (0.162)	1.555 (0.322)	9.864 (4.017)	3.875 (0.936)	909.944 (455.037)	208.38 (137.905)	0.68 (0.38)	**1.403 (0.263)**	3.69 (0.84)	7.237 (4.967)	0.075 (0.009)	0.923 (0.126)	0.259 (0.197)	1760.546 (336.539)	0.211 (0.128)
	RU	3.17 (0.52)	4.399 (2.459)	19.629 (4.362)	6.249 (0.848)	993.745 (365.984)	329.65 (139.384)	1.485 (0.353)	7.151 (0.734)	7.859 (2.467)	36.73 (9.396)	0.444 (0.203)	1.413 (0.145)	1.142 (1.084)	2045.225 (454.847)	0.505 (0.125)
	RO	2.732 (0.187)	1.529 (0.328)	10.006 (3.927)	4.082 (1.092)	901.859 (454.269)	200.918 (139.944)	0.665 (0.4)	1.615 (0.296)	3.731 (0.697)	8.944 (6.66)	0.081 (0.01)	0.914 (0.147)	**0.259 (0.175)**	1771.008 (355.222)	0.21 (0.141)
	WERCS	2.611 (0.118)	1.795 (0.38)	11.217 (5.912)	3.831 (0.916)	886.384 (438.004)	200.592 (135.764)	0.686 (0.352)	1.68 (0.371)	3.516 (0.869)	9.316 (6.442)	0.079 (0.022)	0.953 (0.119)	0.568 (0.626)	**1652.766 (415.57)**	0.286 (0.265)
	GN	2.657 (0.116)	1.99 (0.257)	10.994 (5.187)	4.25 (0.736)	815.815 (380.997)	200.999 (136.731)	0.756 (0.387)	3.276 (0.534)	3.797 (0.494)	15.713 (7.801)	0.084 (0.012)	0.927 (0.114)	0.29 (0.21)	1738.66 (397.418)	0.222 (0.164)
	SMOTER	2.876 (0.272)	1.618 (0.304)	10.434 (4.663)	4.339 (0.521)	870.453 (433.712)	199.127 (145.613)	0.705 (0.393)	1.81 (0.41)	3.82 (0.884)	14.163 (16.995)	0.157 (0.031)	0.992 (0.135)	0.298 (0.234)	1756.15 (364.143)	0.295 (0.14)
	SMOGN	2.85 (0.247)	1.702 (0.323)	11.716 (6.999)	4.413 (0.576)	826.594 (387.609)	200.615 (143.492)	0.687 (0.354)	1.814 (0.319)	3.685 (0.849)	11.814 (14.91)	0.155 (0.039)	0.926 (0.07)	0.295 (0.274)	1752.211 (311.355)	0.314 (0.13)
	WSMOTER	2.653 (0.194)	1.621 (0.348)	9.221 (4.387)	**3.558 (0.421)**	869.937 (409.268)	202.679 (146.806)	0.665 (0.381)	1.673 (0.311)	3.4 (0.571)	**5.899 (3.401)**	**0.07 (0.008)**	0.893 (0.145)	0.28 (0.227)	1744.496 (313.715)	0.186 (0.118)
	G-SMOTER	**2.188 (0.076)**	**1.492 (0.349)**	8.833 (3.833)	3.65 (0.458)	880.081 (433.029)	206.272 (143.831)	0.665 (0.378)	1.51 (0.23)	3.341 (0.787)	7.96 (5.141)	0.075 (0.01)	0.903 (0.141)	0.266 (0.209)	1718.038 (313.928)	0.205 (0.137)
	DAVID	3.264 (0.289)	4.596 (0.922)	64.914 (37.018)	5.499 (1.139)	909.944 (455.037)	224.515 (144.464)	0.954 (0.571)	24.79 (5.089)	4.56 (0.868)	34.043 (61.589)	0.115 (0.016)	0.969 (0.175)	1.191 (1.198)	1792.512 (308.193)	0.258 (0.138)
	KNNOR-REG	2.743 (0.155)	1.555 (0.308)	9.425 (4.562)	3.609 (0.454)	882.475 (406.867)	203.695 (145.579)	0.675 (0.404)	1.723 (0.407)	3.541 (0.711)	9.474 (6.868)	0.075 (0.014)	0.918 (0.169)	0.275 (0.201)	1750.294 (359.657)	0.215 (0.159)
	CARTGen-IR	2.716 (0.161)	1.554 (0.382)	**8.499 (4.582)**	3.563 (0.43)	**662.75 (259.706)**	**179.673 (123.084)**	**0.637 (0.382)**	2.088 (0.352)	**3.336 (0.61)**	8.432 (6.654)	0.074 (0.009)	**0.854 (0.147)**	0.260 (0.196)	1676.084 (364.082)	**0.178 (0.101)**

Table B.4: Best SERA scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue.

Learner	Strategy	abalone	airfoil	availablePower	cpuSm	ele-1	forestFires	fuelConsumption	heat	housingBoston	maxTorque	mortgage	sensory	servo	strikes	treasury
RF	None	2344.629 (329.112)	159.874 (104.774)	11842.204 (10101.166)	3450.568 (727.225)	16367575.153 (12686092.018)	381956.514 (594335.436)	57.768 (60.279)	3879.483 (1140.02)	412.586 (292.205)	36044.139 (36404.312)	0.232 (0.111)	12.235 (16.742)	5.334 (6.336)	25381108.028 (20326797.014)	2.547 (3.431)
	RU	2986.899 (1689.722)	1539.924 (2280.611)	44649.644 (16301.308)	71514.552 (143308.467)	26871543.795 (14303528.629)	419649.224 (464583.668)	161.974 (41.712)	47088.893 (8743.784)	1601.324 (870.252)	200265.979 (96480.171)	21.205 (13.041)	22.961 (42.111)	17.361 (19.906)	25124734.817 (15364389.077)	23.109 (9.65)
	RO	2287.8 (328.378)	159.585 (102.068)	9893.052 (8241.677)	3735.535 (1073.682)	15134141.125 (11563659.547)	378064.642 (590846.781)	55.768 (57.248)	3753.244 (1178.119)	409.321 (320.28)	29022.864 (25808.217)	0.212 (0.084)	12.188 (17.24)	4.581 (5.499)	25307620.666 (20410124.679)	2.469 (3.225)
	WERCS	**1956.194 (221.563)**	179.906 (184.53)	10443.236 (9913.754)	2993.994 (935.359)	14445323.579 (10293091.551)	**336583.166 (537789.206)**	**47.942 (54.044)**	3469.371 (882.333)	372.906 (296.079)	31519.447 (30809.908)	0.234 (0.087)	12.205 (17.016)	7.301 (7.506)	22787844.929 (18708879.658)	8.546 (18.943)
	GN	2083.289 (268.835)	232.506 (151.967)	12210.932 (9645.757)	3455.946 (550.262)	11968789.183 (6617308.456)	361156.676 (575640.712)	58.711 (56.387)	7292.694 (1598.04)	431.813 (294.159)	46320.201 (36996.721)	0.328 (0.162)	11.711 (16.642)	4.164 (4.31)	24656158.611 (20064721.769)	3.41 (4.259)
	SMOTER	2281.524 (374.114)	158.863 (102.624)	**9130.566 (6466.139)**	3939.978 (1218.266)	13694771.614 (9134749.211)	378625.72 (595499.223)	54.795 (55.265)	3601.57 (1227.412)	392.205 (264.063)	31342.497 (21364.349)	0.924 (0.821)	13.283 (17.67)	4.76 (6.248)	25243992.48 (20261619.591)	4.034 (2.862)
	SMOGN	2353.047 (400.998)	171.942 (104.984)	10932.5 (7141.813)	3422.919 (691.123)	11952047.611 (7582370.172)	368072.609 (575257.301)	54.679 (56.628)	3681.22 (1182.118)	388.919 (263.076)	**26550.38 (17780.248)**	1.04 (0.639)	12.36 (17.289)	4.308 (5.374)	24872393.023 (19819229.069)	4.506 (3.078)
	WSMOTER	2115.173 (281.047)	**153.412 (111.868)**	10610.548 (9394.498)	3062.003 (635.043)	14750136.036 (11472349.9)	375758.535 (585849.453)	51.693 (54.736)	3148.818 (949.233)	338.573 (211.384)	32071.863 (30998.145)	**0.189 (0.085)**	10.986 (15.051)	4.663 (5.999)	25039470.188 (20107614.304)	2.053 (2.625)
	G-SMOTER	2305.142 (322.886)	156.092 (103.836)	11071.001 (10286.391)	3249.952 (702.864)	14670397.943 (11017383.605)	381117.043 (598012.436)	55.406 (58.757)	3362.736 (1069.953)	375.569 (272.512)	32648.701 (29620.273)	0.212 (0.099)	11.462 (15.925)	4.057 (5.201)	24967333.226 (20042091.677)	2.314 (2.965)
	DAVID	2949.324 (501.444)	1153.73 (680.296)	463592.284 (254649.999)	7188.322 (1854.144)	15797215.668 (11905308.992)	364136.037 (587393.326)	142.45 (173.653)	711394.315 (75241.932)	673.693 (500.009)	298767.092 (555450.496)	0.595 (0.284)	13.905 (18.934)	38.957 (52.556)	23372002.803 (19642431.759)	4.307 (3.906)
	KNNOR-REG	2230.848 (300.32)	154.855 (101.328)	11577.116 (9745.126)	3298.454 (765.286)	15612799.187 (12087715.203)	384958.607 (592964.064)	56.981 (60.512)	3526.569 (1021.95)	405.258 (317.901)	38831.558 (36039.75)	0.231 (0.102)	11.999 (16.523)	5.521 (7.446)	25066060.335 (20304976.391)	2.409 (2.956)
	CARTGen-IR	2199.465 (285.904)	156.399 (106.685)	10126.531 (9286.489)	**2819.941 (501.566)**	**9641359.653 (5505522.995)**	356122.41 (590855.567)	48.97 (51.719)	**2803.56 (971.341)**	**315.151 (261.439)**	27661.357 (28286.847)	0.202 (0.094)	**10.74 (14.975)**	**3.399 (4.221)**	**22785445.93 (18799517.479)**	**1.675 (2.235)**
SVR	None	2355.9 (310.263)	554.399 (376.766)	15351.21 (9992.859)	7527.234 (1887.381)	32064626.075 (26203374.519)	412490.983 (628747.3)	81.528 (74.531)	833.438 (534.35)	418.307 (307.444)	51280.201 (67675.031)	0.059 (0.023)	12.695 (17.584)	2.862 (2.168)	28055720.18 (21851073.743)	1.639 (1.641)
	RU	2492.601 (558.615)	824.673 (1038.763)	24329.207 (9871.24)	18256.144 (5921.535)	47996792.684 (28190828.129)	324995.359 (485462.252)	118.019 (51.651)	13906.587 (3191.954)	859.162 (460.116)	143234.883 (91885.511)	6.853 (4.259)	23.633 (36.225)	16.677 (19.806)	21757880.642 (17095214.904)	159.81 (229.183)
	RO	2117.475 (323.715)	596.363 (359.891)	**11761.124 (6796.224)**	6868.794 (1205.497)	11224972.97 (6448306.638)	343367.67 (541688.101)	79.765 (67.691)	**195.967 (89.426)**	407.207 (327.602)	**14604.24 (6648.162)**	**0.052 (0.018)**	12.91 (18.327)	2.47 (2.845)	18521496.436 (14994794.633)	1.67 (1.589)
	WERCS	1928.343 (319.295)	**230.799 (175.604)**	12880.305 (6349.914)	3904.678 (983.664)	14135500.213 (9152008.193)	**322776.497 (524862.456)**	73.156 (61.376)	203.172 (156.828)	483.092 (341.423)	26571.895 (25840.841)	0.056 (0.019)	13.596 (18.879)	6.078 (5.216)	18789098.822 (15261494.873)	2.615 (3.557)
	GN	2127.906 (312.399)	585.888 (404.364)	15691.596 (9208.897)	7208.147 (1251.531)	12376319.269 (5692001.862)	334001.257 (488164.602)	81.682 (56.094)	2573.382 (1009.264)	497.927 (328.345)	37612.765 (24720.492)	0.129 (0.06)	13.185 (19.2)	4.154 (3.921)	18364960.064 (14872271.472)	1.892 (2.099)
	SMOTER	2169.691 (393.304)	580.944 (353.118)	27903.371 (12219.001)	6548.497 (1215.25)	11583303.921 (6065409.576)	351271.414 (551697.757)	80.762 (65.027)	365.242 (178.385)	418.359 (312.224)	40727.981 (35869.241)	0.054 (0.022)	14.692 (19.54)	3.04 (4.015)	18416233.475 (14861776.589)	1.693 (1.599)
	SMOGN	2414.716 (496.354)	563.024 (347.75)	32985.734 (16279.546)	6713.197 (949.984)	**10966667.486 (5475754.39)**	339598.258 (543656.629)	79.517 (68.581)	362.744 (180.353)	406.262 (291.589)	37022.391 (48892.196)	0.057 (0.02)	13.153 (18.67)	2.75 (3.192)	**18305541.987 (14893393.673)**	1.708 (1.692)
	WSMOTER	**1903.951 (257.877)**	477.638 (340.714)	12344.66 (7705.377)	6449.697 (1760.759)	22986794.798 (19925692.542)	394428.678 (612752.095)	81.99 (70.694)	350.447 (266.03)	385.739 (252.113)	32825.982 (32491.021)	0.055 (0.019)	11.666 (16.296)	2.803 (2.188)	26716329.875 (21000882.466)	**1.502 (1.373)**
	G-SMOTER	2183.212 (311.589)	492.017 (336.495)	14550.106 (9246.16)	6259.215 (1411.275)	27946286.58 (22864952.192)	410544.395 (626935.762)	78.855 (71.712)	532.524 (388.526)	401.777 (312.53)	27572.641 (20945.364)	0.06 (0.023)	11.861 (16.508)	3.01 (1.763)	27773504.303 (21692936.103)	1.624 (1.646)
	DAVID	3424.804 (655.687)	1128.355 (752.581)	520367.479 (271239.123)	12310.664 (3375.807)	32064626.075 (26203374.519)	362356.139 (489475.382)	175.109 (199.657)	393281.404 (117931.623)	552.653 (299.609)	382397.394 (689362.773)	0.309 (0.133)	14.867 (21.049)	38.335 (51.951)	18883708.766 (14739834.783)	3.383 (3.074)
	KNNOR-REG	2058.452 (270.144)	526.847 (355.705)	15375.059 (9466.357)	6325.31 (1677.963)	27232262.334 (23458432.973)	410941.196 (627645.542)	80.065 (72.016)	980.978 (529.666)	399.308 (284.112)	52870.274 (65651.06)	0.064 (0.024)	12.631 (17.536)	2.921 (2.167)	26947129.081 (21155835.418)	1.795 (1.82)
	CARTGen-IR	1969.096 (256.007)	270.408 (177.737)	14952.781 (8199.041)	**3593.039 (657.41)**	11686222.95 (8975257.553)	334704.455 (540543.308)	**62.952 (42.19)**	429.148 (197.123)	**342.082 (312.657)**	24691.581 (19290.679)	0.071 (0.059)	**11.617 (17.394)**	**2.316 (2.805)**	22264488.25 (18081697.318)	1.682 (1.61)
XGBoost	None	2374.63 (315.825)	114.114 (84.907)	10333.806 (7106.522)	3309.938 (1695.902)	20309229.004 (16544918.56)	392611.838 (600577.722)	45.416 (40.823)	**804.96 (168.172)**	384.962 (164.654)	6716.264 (9788.335)	0.299 (0.13)	10.015 (13.427)	1.408 (1.707)	26114025.238 (20530250.69)	3.206 (3.501)
	RU	3159.822 (1237.196)	1016.727 (1160.408)	36797.841 (16101.916)	8836.559 (3119.928)	20884157.891 (8882706.261)	773824.908 (628287.664)	216.884 (79.959)	22056.092 (3828.349)	1870.9 (1137.224)	121705.089 (46014.264)	11.713 (12.075)	25.151 (38.408)	29.278 (43.458)	38489061.935 (27418018.346)	15.565 (7.265)
	RO	2269.276 (339.902)	109.777 (82.774)	10837.97 (7786.519)	3685.209 (1917.929)	20174218.594 (17052237.315)	374657.306 (564693.61)	48.235 (50.288)	1057.109 (145.916)	380.169 (98.899)	10298.889 (13196.577)	0.346 (0.151)	9.992 (13.848)	1.412 (1.801)	26388078.191 (20442964.158)	3.309 (4.045)
	WERCS	**2068.621 (223.018)**	148.521 (94.004)	13155.86 (9354.292)	3304.37 (1765.408)	18417245.406 (13412957.046)	362648.51 (524111.045)	47.852 (45.87)	1134.906 (158.51)	335.859 (133.114)	9920.806 (9541.495)	0.361 (0.117)	10.695 (14.151)	6.148 (10.857)	**23924043.178 (19940055.66)**	7.28 (11.116)
	GN	2145.011 (262.828)	180.22 (103.939)	11396.207 (7530.917)	3839.478 (851.278)	13909400.053 (8218362.772)	371582.52 (544714.258)	57.433 (49.686)	4400.29 (823.995)	419.463 (181.487)	24274.669 (18845.454)	0.384 (0.167)	10.426 (14.325)	1.664 (1.902)	25664595.998 (20226280.29)	3.925 (4.984)
	SMOTER	2525.622 (496.765)	123.666 (91.756)	12592.249 (8220.734)	4015.236 (688.486)	17699029.194 (13833270.398)	379066.795 (583625.135)	50.77 (51.274)	1349.953 (411.495)	405.062 (194.848)	22094.709 (36893.24)	1.242 (0.622)	11.505 (15.437)	2.013 (2.153)	25969036.535 (20511157.366)	5.564 (5.064)
	SMOGN	2477.367 (453.462)	132.262 (85.61)	11969.874 (8672.033)	4191.919 (972.52)	14795899.44 (9233563.429)	380365.974 (568255.764)	50.615 (40.505)	1342.113 (229.843)	411.589 (246.251)	14785.074 (22289.047)	1.24 (0.922)	10.304 (13.869)	1.751 (2.148)	25868632.476 (20132394.918)	6.216 (4.349)
	WSMOTER	2144.37 (357.838)	123.478 (90.783)	8337.861 (5503.207)	**2705.585 (469.048)**	18396364.726 (15355163.144)	383546.067 (584262.211)	45.913 (48.298)	1133.376 (149.84)	321.923 (98.189)	**4014.111 (5050.882)**	**0.263 (0.115)**	9.102 (11.796)	1.598 (1.829)	25609492.618 (20005870.733)	2.5 (3.021)
	G-SMOTER	2312.018 (284.14)	**103.29 (68.287)**	7813.816 (5962.692)	2830.653 (324.49)	18732280.531 (13900104.982)	395086.438 (597406.663)	45.926 (40.395)	944.259 (247.455)	310.932 (106.89)	7873.349 (10802.499)	0.291 (0.118)	9.562 (12.731)	1.502 (1.863)	25157588.615 (20344865.898)	3.076 (3.88)
	DAVID	3243.745 (605.703)	951.449 (618.198)	394455.063 (249480.83)	6280.156 (1282.365)	20309229.004 (16544918.56)	448793.315 (624424.57)	112.37 (133.031)	249130.73 (46005.809)	627.154 (410.414)	216714.301 (445559.766)	0.688 (0.28)	11.915 (18.141)	38.568 (51.737)	27182079.895 (21291380.413)	4.462 (4.169)
	KNNOR-REG	2283.211 (289.559)	111.598 (76.382)	9368.686 (7900.754)	2832.752 (435.948)	18752687.214 (14360894.651)	391747.858 (608570.172)	49.385 (47.831)	1205.362 (302.127)	356.006 (141.084)	11562.583 (16617.648)	0.304 (0.155)	9.77 (13.154)	1.533 (1.835)	25905696.216 (20724694.494)	3.529 (4.802)
	CARTGen-IR	2242.262 (304.122)	115.159 (84.221)	**7237.189 (5866.996)**	2765.906 (578.441)	**9777311.079 (7080594.562)**	**306885.035 (548003.586)**	**41.407 (39.829)**	1725.244 (411.988)	**310.266 (79.017)**	9196.557 (12388.776)	0.291 (0.15)	**8.941 (13.217)**	**1.39 (1.599)**	24286622.221 (19592553.796)	**2.11 (2.555)**

Table B.5: Best DW-RMSE scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue.

Learner	Strategy	abalone	airfoil	availablePower	cpuSm	ele-1	forestFires	fuelConsumption	heat	housingBoston	maxTorque	mortgage	sensory	servo	strikes	treasury
RF	None	2.92 (0.198)	2.123 (0.178)	10.127 (3.588)	3.615 (0.18)	924.336 (137.244)	235.845 (222.075)	0.676 (0.302)	3.361 (0.341)	4.253 (1.193)	19.157 (6.816)	0.193 (0.027)	1.012 (0.047)	1.022 (0.256)	1199.886 (290.26)	0.304 (0.079)
	RU	3.067 (0.299)	11.131 (0.847)	27.704 (1.806)	19.296 (2.36)	1173.218 (109.322)	285.517 (140.677)	1.461 (0.116)	14.583 (0.81)	7.18 (1.379)	65.831 (6.575)	0.338 (0.035)	1.205 (0.425)	1.22 (0.539)	1865.492 (137.6)	0.441 (0.094)
	RO	2.894 (0.208)	2.236 (0.191)	**9.522 (3.192)**	3.679 (0.259)	915.572 (138.442)	234.203 (219.853)	0.665 (0.301)	3.283 (0.349)	4.219 (1.038)	17.552 (5.66)	0.197 (0.031)	1.006 (0.056)	0.878 (0.267)	1204.221 (277.818)	0.309 (0.092)
	WERCS	**2.729 (0.145)**	3 (0.287)	10.091 (3.791)	3.431 (0.169)	900.222 (113.68)	**223.233 (207.953)**	0.634 (0.279)	3.583 (0.275)	3.937 (1.045)	18.973 (6.365)	0.296 (0.029)	0.994 (0.048)	0.957 (0.287)	1181.453 (275.702)	0.432 (0.197)
	GN	2.764 (0.186)	2.962 (0.31)	11.509 (3.076)	3.67 (0.117)	944.196 (121.59)	228.284 (217.014)	0.713 (0.27)	5.09 (0.401)	4.199 (0.998)	23.715 (6.762)	0.227 (0.036)	0.98 (0.067)	0.861 (0.186)	1203.569 (269.139)	0.366 (0.092)
	SMOTER	2.962 (0.189)	2.242 (0.19)	10.451 (3.954)	3.731 (0.295)	927.668 (118.191)	232.165 (222.47)	0.661 (0.3)	3.219 (0.355)	4.131 (1.157)	18.174 (5.002)	0.214 (0.042)	1.052 (0.061)	0.942 (0.309)	1203.622 (279.281)	0.333 (0.097)
	SMOGN	3.006 (0.186)	2.361 (0.191)	11.274 (5.565)	3.621 (0.15)	936.438 (133.245)	233.534 (214.512)	0.661 (0.301)	3.264 (0.383)	4.059 (0.975)	16.976 (3.792)	0.229 (0.03)	1.011 (0.056)	0.885 (0.239)	1208.075 (269.481)	0.349 (0.081)
	WSMOTER	2.767 (0.192)	**2.049 (0.206)**	9.788 (3.467)	3.48 (0.165)	902.408 (150.317)	231.265 (221.475)	0.639 (0.296)	3.072 (0.327)	3.877 (0.983)	18.032 (5.732)	**0.19 (0.033)**	0.957 (0.04)	0.943 (0.254)	1197.397 (290.735)	0.281 (0.074)
	G-SMOTER	2.902 (0.201)	2.09 (0.172)	10.02 (3.651)	3.541 (0.168)	903.714 (131.571)	233.809 (222.232)	0.661 (0.3)	3.134 (0.366)	4.159 (1.13)	18.017 (6.746)	0.195 (0.034)	0.981 (0.034)	0.905 (0.287)	1196.563 (285.214)	0.308 (0.084)
	DAVID	3.402 (0.237)	5.336 (0.489)	66.921 (15.534)	4.92 (0.389)	922.214 (124.793)	233.284 (214.59)	0.986 (0.613)	45.255 (1.942)	5.244 (1.469)	40.225 (44.842)	0.275 (0.04)	1.101 (0.083)	3.37 (0.467)	1196.801 (281.741)	0.396 (0.103)
	KNNOR-REG	2.857 (0.187)	2.087 (0.169)	9.955 (3.623)	3.586 (0.19)	890.623 (138.064)	233.137 (222.583)	0.666 (0.298)	3.17 (0.31)	4.279 (1.16)	19.697 (6.647)	0.194 (0.026)	1.005 (0.063)	1.046 (0.329)	1194.647 (281.591)	0.3 (0.068)
	CARTGen-IR	2.827 (0.167)	2.074 (0.168)	9.641 (3.443)	**3.429 (0.133)**	**815.268 (97.456)**	225.149 (222.263)	**0.627 (0.29)**	**2.908 (0.384)**	**3.676 (1.048)**	**16.846 (6.433)**	0.195 (0.035)	**0.95 (0.043)**	**0.836 (0.238)**	**1157.292 (280.529)**	**0.272 (0.083)**
SVR	None	2.979 (0.187)	3.363 (0.331)	12.043 (3.177)	4.72 (0.426)	1193.388 (236.711)	244.989 (225.197)	0.819 (0.303)	1.289 (0.422)	4.118 (1.284)	20.231 (11.535)	0.091 (0.015)	1.064 (0.037)	0.565 (0.163)	1268.993 (268.458)	0.226 (0.043)
	RU	3.036 (0.181)	8.048 (0.605)	17.426 (1.854)	8.22 (0.435)	1408.921 (198.08)	227.137 (219.324)	1.056 (0.185)	8.592 (0.517)	5.568 (0.628)	39.835 (10.793)	2.036 (0.226)	1.266 (0.245)	1.185 (0.721)	1247.205 (271.103)	2.596 (0.27)
	RO	2.881 (0.219)	4.498 (0.631)	10.966 (2.495)	4.685 (0.315)	881.783 (102.697)	220.44 (213.163)	0.82 (0.293)	**0.686 (0.112)**	4.103 (1.123)	**12.75 (3.073)**	0.09 (0.015)	1.053 (0.083)	**0.541 (0.157)**	1169.337 (289.106)	0.225 (0.042)
	WERCS	2.752 (0.164)	3.703 (0.414)	11.646 (2.201)	3.935 (0.279)	883.457 (110.487)	**214.851 (209.279)**	0.803 (0.275)	0.72 (0.212)	4.095 (1.067)	16.62 (6.643)	0.139 (0.032)	1.068 (0.069)	0.773 (0.283)	**1124.905 (254.144)**	0.28 (0.063)
	GN	2.871 (0.219)	5.243 (0.71)	13.076 (2.85)	5.299 (0.554)	937.751 (79.562)	217.107 (214.597)	0.847 (0.239)	2.561 (0.403)	4.305 (0.935)	20.47 (5.015)	0.114 (0.022)	1.058 (0.083)	0.641 (0.156)	1149.565 (266.568)	0.254 (0.062)
	SMOTER	2.94 (0.209)	4.48 (0.618)	17.385 (2.576)	4.626 (0.324)	876.158 (74.85)	223.178 (214.966)	0.824 (0.287)	0.896 (0.207)	4.157 (1.083)	20.086 (8.602)	0.09 (0.015)	1.088 (0.106)	0.633 (0.167)	1177.734 (287.773)	0.226 (0.042)
	SMOGN	3.129 (0.187)	4.53 (0.633)	19.303 (3.379)	4.774 (0.218)	909.489 (116.769)	217.993 (215.087)	0.82 (0.318)	0.887 (0.198)	4.25 (1.278)	20.347 (10.2)	0.108 (0.046)	1.06 (0.078)	0.568 (0.211)	1125.461 (262.899)	0.234 (0.044)
	WSMOTER	2.674 (0.156)	**3.158 (0.334)**	**10.896 (2.837)**	4.449 (0.425)	990.113 (239.432)	237.521 (224.49)	0.826 (0.3)	0.84 (0.301)	3.865 (1.017)	17.599 (7.451)	**0.088 (0.014)**	**1.005 (0.032)**	0.585 (0.159)	1232.223 (277.49)	0.222 (0.034)
	G-SMOTER	2.885 (0.183)	3.299 (0.374)	11.862 (3.235)	4.468 (0.34)	1097.764 (237.155)	244.231 (225.109)	0.806 (0.299)	1.041 (0.35)	4 (1.195)	16.325 (6.088)	0.091 (0.015)	1.034 (0.029)	0.562 (0.151)	1261.193 (271.767)	**0.221 (0.043)**
	DAVID	4.428 (0.564)	5.629 (0.835)	71.247 (14.023)	6.45 (0.773)	1193.388 (236.711)	258.632 (164.379)	1.118 (0.643)	37.555 (13.323)	4.358 (0.733)	44.886 (50.002)	0.181 (0.025)	1.127 (0.076)	3.339 (0.494)	1181.84 (270.945)	0.337 (0.078)
	KNNOR-REG	2.814 (0.167)	3.36 (0.321)	11.945 (3.174)	4.444 (0.413)	1068.214 (262.096)	244.316 (225.01)	0.813 (0.297)	1.407 (0.367)	4.074 (1.267)	20.332 (11.229)	0.096 (0.024)	1.059 (0.043)	0.574 (0.166)	1238.466 (274.88)	0.232 (0.047)
	CARTGen-IR	**2.597 (0.158)**	3.301 (0.31)	12.195 (3.166)	**3.881 (0.265)**	**827.663 (123.126)**	218.193 (213.738)	**0.764 (0.276)**	1.096 (0.266)	**3.843 (1.18)**	16.505 (4.212)	0.092 (0.014)	1.011 (0.068)	0.561 (0.264)	1142.667 (265.98)	0.233 (0.044)
XGBoost	None	2.969 (0.179)	1.614 (0.158)	9.518 (3.062)	3.488 (0.391)	1055.46 (189.036)	240.286 (217.404)	0.62 (0.24)	**1.656 (0.117)**	3.955 (0.919)	8.153 (4.392)	0.192 (0.036)	0.923 (0.038)	0.447 (0.145)	1238.849 (250.243)	0.334 (0.054)
	RU	3.243 (0.226)	8.37 (0.852)	25.208 (7.247)	6.174 (0.842)	1133.98 (140.053)	371.205 (148.127)	1.647 (0.254)	10.312 (0.788)	7.561 (1.471)	49.59 (10.673)	0.355 (0.081)	1.32 (0.219)	1.529 (0.581)	2316.066 (346.842)	0.469 (0.093)
	RO	2.906 (0.165)	1.726 (0.301)	9.958 (2.979)	3.619 (0.445)	986.264 (176.578)	232.777 (212.071)	0.626 (0.276)	1.908 (0.102)	3.8 (0.534)	10.427 (4.703)	0.187 (0.026)	0.897 (0.029)	**0.444 (0.133)**	1252.491 (257.86)	0.308 (0.077)
	WERCS	2.829 (0.158)	2.235 (0.28)	11.265 (3.412)	3.583 (0.346)	990.351 (196.679)	239.831 (202.87)	0.658 (0.244)	2.282 (0.092)	3.621 (0.55)	10.903 (4.483)	0.285 (0.044)	0.937 (0.064)	0.824 (0.452)	1208.675 (295.492)	0.446 (0.132)
	GN	**2.81 (0.172)**	2.389 (0.298)	11.465 (3.127)	3.8 (0.21)	992.515 (165.465)	239.246 (202.604)	0.722 (0.265)	4.111 (0.222)	3.914 (0.422)	18.687 (7.229)	0.249 (0.045)	0.919 (0.074)	0.507 (0.133)	1272.351 (208.576)	0.354 (0.078)
	SMOTER	3.153 (0.186)	1.795 (0.292)	11.347 (4.139)	3.809 (0.123)	1010.044 (180.685)	233.312 (216.032)	0.646 (0.288)	2.187 (0.254)	4.098 (0.848)	14.027 (9.235)	0.21 (0.029)	0.944 (0.024)	0.547 (0.209)	1245.079 (242.758)	0.367 (0.082)
	SMOGN	3.12 (0.187)	1.919 (0.24)	12.367 (4.43)	3.842 (0.209)	1015.726 (140.732)	235.342 (211.526)	0.639 (0.289)	2.205 (0.207)	3.921 (0.773)	11.856 (7.763)	0.229 (0.03)	0.894 (0.024)	0.454 (0.261)	1260.953 (235.563)	0.394 (0.066)
	WSMOTER	2.811 (0.215)	1.653 (0.217)	8.692 (2.713)	**3.317 (0.096)**	1005.623 (196.776)	238.441 (221.651)	0.606 (0.272)	1.989 (0.09)	**3.341 (0.46)**	**7.129 (2.393)**	**0.181 (0.043)**	0.888 (0.054)	0.481 (0.137)	1229.51 (234.82)	**0.276 (0.066)**
	G-SMOTER	2.93 (0.181)	**1.582 (0.174)**	8.505 (2.549)	3.383 (0.087)	1014.73 (173.499)	239.669 (222.817)	0.609 (0.278)	1.798 (0.149)	3.525 (0.641)	9.031 (5.366)	0.184 (0.026)	0.905 (0.041)	0.458 (0.162)	1213.51 (266.327)	0.305 (0.077)
	DAVID	3.599 (0.289)	4.561 (0.426)	60.275 (16.373)	4.893 (0.498)	1055.46 (189.036)	273.904 (210.792)	0.884 (0.526)	25.595 (2.767)	4.543 (0.978)	28.123 (42.066)	0.274 (0.051)	1.037 (0.131)	3.363 (0.432)	1308.543 (269.219)	0.383 (0.091)
	KNNOR-REG	2.935 (0.186)	1.634 (0.152)	9.112 (3.134)	3.42 (0.12)	1006.931 (175.13)	235.075 (221.809)	0.631 (0.301)	2.057 (0.192)	3.492 (0.654)	9.832 (4.621)	0.187 (0.025)	0.914 (0.056)	0.446 (0.148)	1228.85 (273.657)	0.322 (0.086)
	CARTGen-IR	2.874 (0.166)	1.659 (0.203)	**8.124 (2.951)**	3.402 (0.161)	**868.638 (140.029)**	**232.449 (223.512)**	**0.581 (0.249)**	2.417 (0.138)	3.634 (0.407)	10.251 (4.151)	0.182 (0.026)	**0.871 (0.014)**	0.46 (0.127)	**1185.64 (251.872)**	0.299 (0.072)

Table B.6: Best DW-SERA scores (mean ± std. dev.) across learners, strategies, and datasets. Best results per learner are bolded. The overall best per dataset is shaded in light blue.

Learner	Strategy	abalone	airfoil	availablePower	cpuSm	ele-1	forestFires	fuelConsumption	heat	housingBoston	maxTorque	mortgage	sensory	servo	strikes	treasury
RF	None	3063.686 (315.567)	696.008 (105.798)	14713.931 (9766.508)	9961.877 (911.263)	34293590.29 (8683724.258)	406490.503 (587012.675)	75.245 (59.643)	6799.771 (1201.566)	869.292 (384.754)	52087.907 (32284.98)	3.469 (1.019)	50.93 (5.608)	10.414 (4.414)	30758888.253 (16444661.9)	8.123 (3.204)
	RU	4299.561 (1516.909)	21220.032 (2621.136)	156095.039 (16578.953)	497994.338 (127210.123)	78468121.194 (15700372.17)	1679923.024 (532266.52)	485.048 (86.975)	172591.69 (18146.244)	3589.373 (1570.649)	835750.859 (150587.857)	15.99 (3.263)	80.02 (39.502)	24.332 (17.003)	275477537.56 (68380758.65)	20.974 (6.348)
	RO	3065.51 (301.521)	783.79 (151.284)	**13100.489 (7768.404)**	10232.126 (1202.465)	34821453.445 (9307489.269)	408137.772 (573676.113)	73.862 (59.379)	6467.945 (1236.04)	847.403 (322.089)	44443.819 (22918.945)	3.628 (1.205)	50.616 (4.889)	8.376 (3.603)	31389200.218 (16255770.793)	8.414 (3.787)
	WERCS	2959.675 (207.059)	1479.499 (224.23)	16557.348 (10317.141)	9537.697 (809.029)	34393510.668 (7928284.03)	401364.681 (537730.289)	72.764 (51.549)	9215.508 (1113.08)	782.165 (330.776)	57050.658 (29478.958)	7.703 (1.676)	51.558 (6.942)	12.133 (5.471)	35727168.08 (14874742.142)	16.906 (16.908)
	GN	3053.995 (250.21)	1423.719 (262.475)	20407.339 (8918.28)	10518.551 (678.645)	44669338.558 (13036225.859)	439062.939 (545014.415)	90.507 (53.978)	16967.483 (2137.052)	903.787 (290.539)	83385.485 (38262.197)	5.287 (1.958)	51.187 (4.865)	8.693 (3.271)	32966239.557 (15518535.774)	11.315 (4.685)
	SMOTER	3618.018 (215.39)	787.967 (149.989)	16147.696 (10926.356)	10415.94 (1347.517)	39210217.772 (10162457.362)	403192.372 (581634.279)	73.009 (54.402)	6268.72 (1262.93)	842.75 (323.934)	46815.903 (19426.14)	4.573 (1.51)	53.967 (6.865)	9.236 (4.813)	31077571.149 (16106083.976)	10.19 (4.908)
	SMOGN	3743.384 (354.3)	883.306 (126.237)	19886.543 (18814.97)	9978.262 (708.448)	41678512.651 (13277891.352)	424393.534 (581392.825)	73.151 (57.031)	6451.886 (1385.662)	808.415 (293.586)	**40652.475 (13315.052)**	5.347 (1.299)	51.128 (5.233)	8.271 (3.558)	31768613.086 (15905984.564)	11.234 (4.599)
	WSMOTER	**2926.104 (284.182)**	**659.873 (112.609)**	13708.831 (8990.19)	9460.925 (783.669)	34630592.431 (10755863.441)	405600.051 (571410.523)	68.979 (55.062)	5798.785 (1038.561)	741.104 (300.192)	46226.715 (27108.993)	**3.366 (1.241)**	**48.397 (4.229)**	8.849 (4.366)	30833899.785 (16545628.291)	7.097 (2.796)
	G-SMOTER	3048.844 (312.81)	679.742 (102.513)	14432.485 (9271.957)	9638.079 (769.178)	33614395.868 (8432997.905)	403833.96 (588603.93)	72.595 (58.184)	5967.876 (1213.052)	833.056 (355.506)	46914.128 (29649.93)	3.533 (1.337)	49.287 (3.991)	8.29 (4.269)	30636518.495 (16713376.146)	8.307 (3.465)
	DAVID	4054.506 (460.701)	4297.799 (637.376)	576551.459 (228043.705)	17690.194 (2167.274)	34185729.541 (8330204.17)	433342.481 (553054.27)	179.862 (210.905)	1113963.412 (97205.103)	1317.152 (620.311)	436497.205 (823621.156)	7.174 (1.601)	60.742 (8.603)	97.2 (28.104)	40027896.083 (14666619.393)	15.538 (9.117)
	KNNOR-REG	3020.846 (271.948)	678.972 (114.274)	14272.09 (9630.148)	9959.417 (948.97)	**32903942.165 (8969120.849)**	398913.094 (585324.72)	73.513 (57.844)	6121.449 (1053.311)	880.985 (370.019)	54733.937 (31051.347)	3.475 (0.993)	51.194 (5.911)	10.778 (4.965)	30521607.079 (16196990.7)	7.815 (2.637)
	CARTGen-IR	2965.266 (256.169)	685.842 (101.507)	13449.293 (9041.481)	**9366.835 (522.581)**	33442589.029 (8894899.313)	**392015.386 (585516.798)**	**67.141 (54.483)**	**5166.143 (1235.402)**	**693.745 (240.824)**	42500.815 (26985.19)	3.564 (1.279)	48.485 (3.976)	**7.158 (3.358)**	**30035702.78(15272267.781)**	**6.753 (3.628)**
SVR	None	3133.162 (282.18)	1704.437 (296.798)	21215.813 (9631.805)	15802.58 (2185.367)	49266539.354 (18641734.057)	428340.194 (620606.619)	107.779 (74.885)	900.773 (533.074)	872.3 (421.234)	66500.803 (69098.884)	0.79 (0.221)	59.754 (8.778)	4.329 (2.194)	34539305.394 (16242455.211)	5.172 (1.557)
	RU	3612.783 (394.585)	9798.548 (1154.732)	61161.186 (8480.996)	62122.903 (5168.52)	88637652.007 (16072261.585)	406079 (558632.711)	221.16 (47.231)	56608.487 (4371.987)	1877.492 (314.65)	262055.677 (87620.53)	342.887 (84.416)	81.769 (24.858)	21.985 (16.985)	33579597.822 (16314199.926)	539.475 (143.188)
	RO	3344.041 (403.79)	3261.675 (824.571)	18405.417 (7631.957)	16224.465 (2408.097)	41674561.106 (11174845.348)	372648.505 (531868.677)	108.366 (71.212)	**275.025 (77.875)**	859.516 (352.209)	**28964.563 (10027.699)**	0.786 (0.224)	60.644 (6.53)	**4.074 (2.513)**	31697957.904 (17121353.202)	5.164 (1.543)
	WERCS	2985.447 (211.841)	2516.912 (578.197)	20986.3 (6458.226)	12955.123 (1820.231)	36629377.509 (6945822.52)	370343.631 (523685.734)	108.796 (63.756)	344.941 (161.371)	895.165 (344.564)	47997.32 (28301.153)	1.859 (0.745)	69.521 (13.508)	9.868 (5.692)	30447341.443 (15595640.417)	7.435 (2.444)
	GN	3262.398 (285.215)	4435.079 (1086.47)	27368.104 (10389.467)	23266.086 (5293.89)	50879927.197 (10183809.423)	**366985.455 (530011.997)**	126.649 (56.292)	3657.072 (1011.7)	973.575 (281.63)	67821.754 (26798.749)	1.236 (0.411)	67.258 (10.918)	5.915 (2.689)	32401345.347 (18568737.826)	6.387 (2.084)
	SMOTER	3618.931 (385.485)	3251.676 (832.143)	43006.442 (10784.397)	15942.046 (2364.68)	39158062.137 (7934667.264)	377672.574 (542139.848)	109.88 (69.381)	442.914 (170.561)	878.228 (341.78)	65759.843 (44729.199)	0.788 (0.224)	67.552 (13.695)	5.053 (2.122)	31505239.116 (18136764.505)	5.209 (1.544)
	SMOGN	4336.42 (692.399)	3356.329 (812.403)	54932.42 (16098.691)	17393.287 (2107.584)	46863318.03 (14481764.182)	369011.4 (535417.864)	110.642 (78.821)	433.288 (164.435)	927.536 (363.318)	72051.909 (62370.346)	1.179 (0.981)	61.167 (7.28)	4.717 (2.647)	31918701.028 (18783531.702)	5.49 (1.485)
	WSMOTER	**2750.039 (233.535)**	**1551.757 (300.481)**	**17782.537 (7859.63)**	14437.464 (2143.644)	37220480.484 (14756629.173)	413361.411 (603550.207)	107.893 (72.968)	404.143 (261.261)	**773.885 (305.115)**	48492.712 (33851.15)	**0.776 (0.227)**	58.514 (7.03)	4.439 (2.157)	31994604.979 (16538180.958)	5.056 (1.445)
	G-SMOTER	3027.473 (245.27)	1675.542 (310.466)	20551.811 (9076.08)	14666.592 (1768.217)	43048327.443 (16894442.68)	426090.929 (619007.225)	105.114 (73.016)	603.58 (385.345)	823.834 (360.763)	42590.325 (22926.378)	0.792 (0.219)	**57.581 (7.816)**	4.322 (2.02)	34006930.454 (16288424.7)	**5.016 (1.59)**
	DAVID	8787.029 (2115.836)	5412.664 (1540.198)	654191.803 (218824.092)	31092.589 (6999.086)	49266539.354 (18641734.057)	863529.757 (361954.646)	220.399 (235.784)	817084.746 (344579.114)	1019.792 (307.716)	540407.8 (996717.516)	3.241 (0.761)	75.156 (12.165)	95.48 (28.302)	81404690.361 (33467428.057)	11.885 (7.165)
	KNNOR-REG	2989.753 (256.174)	1730.94 (301.995)	21277.056 (9634.957)	14476.208 (2053.143)	42393208.827 (18263106.37)	426632.926 (619860.391)	106.746 (72.165)	1038.27 (495.56)	856.064 (414.12)	66982.465 (67183.603)	0.904 (0.399)	60.403 (8.638)	4.461 (2.209)	32419559.887 (16659721.191)	5.414 (1.758)
	CARTGen-IR	2772.314 (205.791)	1781.062 (287.259)	22127.625 (10007.737)	**12205.585 (1701.26)**	**30627661.746 (6506506.35)**	369156.735 (533866.515)	**97.96 (63.135)**	711.83 (327.564)	784.962 (382.48)	45293.097 (16095.379)	0.833 (0.221)	59.842 (8.414)	4.359 (2.013)	**30114540.04 (15934306.007)**	5.646 (1.812)
XGBoost	None	3211.277 (275.142)	421.725 (68.497)	12903.247 (6541.686)	9497.453 (1638.331)	44648501.494 (13543124.514)	433786.134 (577340.058)	63.886 (39.895)	**1811.424 (226.283)**	796.514 (340.985)	11249.385 (9326.267)	3.786 (1.12)	47.341 (5.557)	2.557 (1.255)	35007168.022 (15999962.145)	9.851 (1.899)
	RU	4835.697 (954.647)	12957.799 (2466.872)	137634.049 (86370.603)	31902.839 (9389.234)	68470527.52 (19029705.793)	3423449.838 (2388622.93)	617.763 (220.525)	96021.637 (15062.128)	4238.555 (1963.969)	508976.891 (209636.653)	16.555 (3.411)	120.842 (26.887)	46.218 (36.904)	392230589.846 (102877533.396)	24.021 (9.081)
	RO	3259.597 (226.457)	476.97 (169.042)	13670.078 (7098.552)	10086.404 (1876.009)	40760938.311 (12653498.345)	438357.865 (599729.578)	66.826 (48.715)	2344.623 (204.228)	746.926 (190.26)	16144.843 (11421.348)	3.387 (0.804)	45.864 (5.181)	**2.466 (0.985)**	36019791.833 (16535971.216)	8.91 (2.792)
	WERCS	3295.31 (185.858)	887.565 (243.994)	19285.769 (9457.554)	10462.355 (1477.205)	41183340.716 (11831690.79)	548942.337 (483910.089)	80.938 (44.735)	3997.627 (303.867)	715.016 (215.886)	19983.44 (11548.668)	8.757 (3.962)	52.826 (5.526)	13.383 (17.417)	39738459.495 (14774878.941)	17.389 (8.18)
	GN	3096.112 (236.127)	1011.338 (251.779)	19803.988 (10880.334)	11136.063 (808.446)	50216326.391 (12566078.503)	529676.051 (520530.609)	97.853 (53.071)	11524.673 (1067.856)	916.805 (207.08)	52799.049 (19140.972)	5.94 (1.875)	52.431 (5.269)	3.198 (1.544)	42992739.498 (15808988.569)	11.86 (3.835)
	SMOTER	4063.045 (266.769)	520.186 (166.656)	18247.342 (12624.62)	10773.253 (584.444)	45495969.55 (12182485.884)	**413876.439 (548848.654)**	69.283 (43.421)	3218.123 (625.801)	874.406 (319.815)	33591.808 (39392.33)	4.685 (1.119)	52.531 (6.062)	3.555 (2.396)	36407874.255 (16099604.962)	12.533 (3.558)
	SMOGN	4028.257 (386.108)	593.367 (141.467)	22329.708 (13425.669)	11079.3 (905.44)	49952292.218 (15709157.054)	486110.575 (513307.117)	70.892 (53.128)	3287.337 (526.036)	823.324 (298.529)	21601.532 (21968.385)	5.6 (1.365)	45.34 (3.312)	2.853 (2.965)	37472415.216 (17032090.828)	14.3 (3.088)
	WSMOTER	3062.493 (353.109)	452.197 (93.053)	11265.359 (5367.456)	**8808.972 (468.837)**	41357520.429 (13202960.592)	439309.662 (588876.388)	62.596 (47.418)	2702.361 (231.521)	**610.88 (180.497)**	**8298.074 (4412.356)**	3.411 (1.088)	45.189 (4.711)	2.747 (1.324)	33399123.308 (17903244.522)	**7.337 (2.194)**
	G-SMOTER	3131.928 (268.063)	**407.463 (76.815)**	10793.339 (5210.669)	9066.62 (585.782)	41631455.932 (11892889.465)	437014.389 (574011.262)	64.179 (44.371)	2127.186 (304.821)	649.741 (216.466)	13445.144 (9901.135)	**3.259 (0.955)**	45.485 (4.291)	2.668 (1.357)	34206650.029 (15066124.085)	8.667 (3.029)
	DAVID	4626.768 (581.155)	3256.593 (753.657)	472174.159 (229144.712)	18193.805 (2976.618)	44648501.494 (13543124.514)	715992.967 (757284.92)	146.946 (161.259)	348548.201 (78033.023)	1132.969 (466.031)	290695.24 (596691.894)	7.907 (2.704)	63.412 (11.067)	96.741 (27.238)	53772661.548 (25086116.547)	14.888 (6.586)
	KNNOR-REG	3216.178 (282.747)	445.828 (71.39)	12522.729 (7219.348)	9314.888 (515.396)	41376123.988 (11853150.645)	420521.129 (584041.049)	68.629 (56.723)	2849.068 (446.619)	646.373 (219.594)	14628.986 (10600.758)	3.589 (0.82)	46.576 (5.106)	2.524 (1.446)	34245656.462 (15922972.894)	9.564 (3.732)
	CARTGen-IR	**3036.923 (260.672)**	465.474 (99.858)	**9982.248 (6136.984)**	9354.382 (704.922)	**35902236.762 (11882600.681)**	417193.347 (588393.216)	**58.864 (40.91)**	3925.756 (456.985)	691.773 (123.001)	16169.796 (10889.623)	3.514 (0.867)	**44.558 (3.427)**	2.585 (1.048)	**32256480.806 (14715460.832)**	8.426 (2.757)

Table B.7: Best preprocessing strategy + learner combination per dataset and metric

Dataset	RMSE	RW-RMSE	SERA	DW-RMSE	DW-SERA
abalone	None + SVR	G-SMOTER + XGBoost	WSMOTER + SVR	CARTGen-IR + SVR	WSMOTER + SVR
airfoil	G-SMOTER + XGBoost	G-SMOTER + XGBoost	G-SMOTER + XGBoost	G-SMOTER + XGBoost	G-SMOTER + XGBoost
availablePower	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost
cpuSm	G-SMOTER + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost
ele-1	KNNOR-REG + RF	CARTGen-IR + XGBoost	CARTGen-IR + RF	CARTGen-IR + RF	CARTGen-IR + SVR
forestFires	CARTGen-IR + SVR	WERCS + SVR	CARTGen-IR + XGBoost	WERCS + SVR	GN + SVR
fuelConsumption	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost
heat	RO + SVR	RO + SVR	RO + SVR	RO + SVR	RO + SVR
housingBoston	WSMOTER + XGBoost	CARTGen-IR + RF	CARTGen-IR + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost
maxTorque	WSMOTER + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost	WSMOTER + XGBoost
mortgage	RO + SVR	WERCS + SVR	RO + SVR	WSMOTER + SVR	WSMOTER + SVR
sensory	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost	CARTGen-IR + XGBoost
servo	RO + XGBoost	RO + XGBoost	CARTGen-IR + XGBoost	RO + XGBoost	RO + XGBoost
strikes	CARTGen-IR + SVR	SMOGN + SVR	SMOGN + SVR	WERCS + SVR	CARTGen-IR + RF
treasury	WSMOTER + SVR	WSMOTER + SVR	WSMOTER + SVR	G-SMOTER + SVR	G-SMOTER + SVR

B.3. Wilcoxon Signed-Rank Test Results

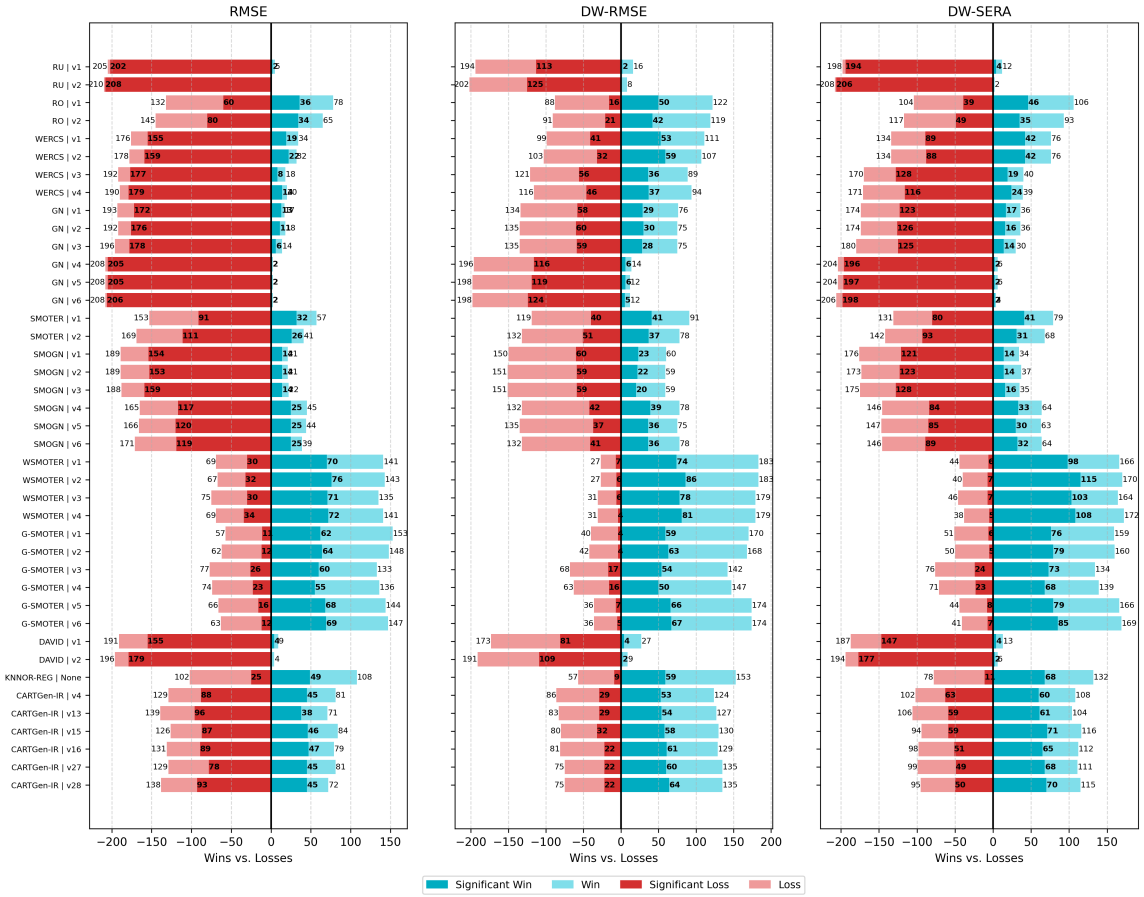


Figure B.1: Comparison of (significant) wins and losses of selected preprocessing strategies across all datasets, according to different metrics - RMSE, DW-RMSE and DW-SERA

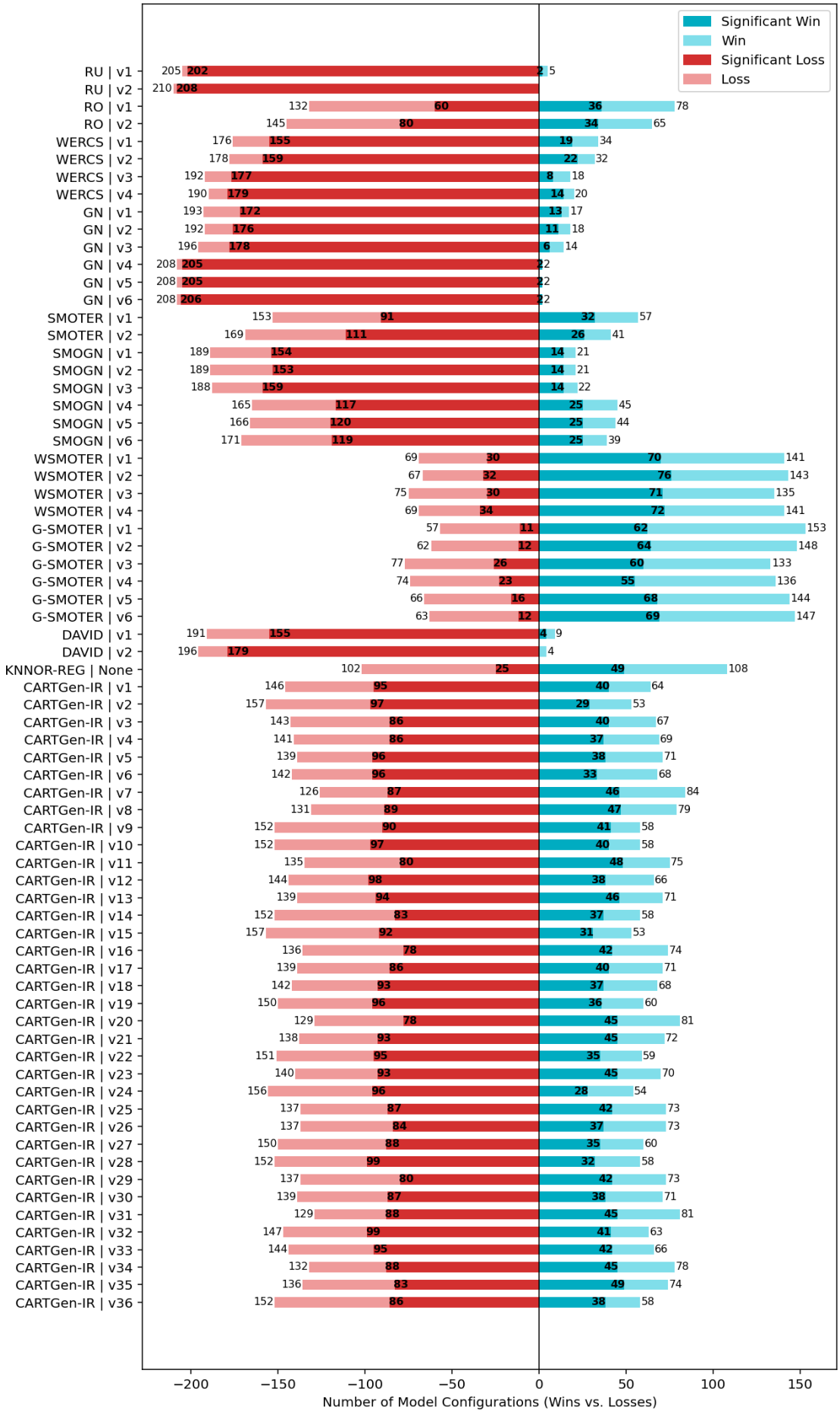


Figure B.2: RMSE results by (significant) wins/losses vs. baseline scenario

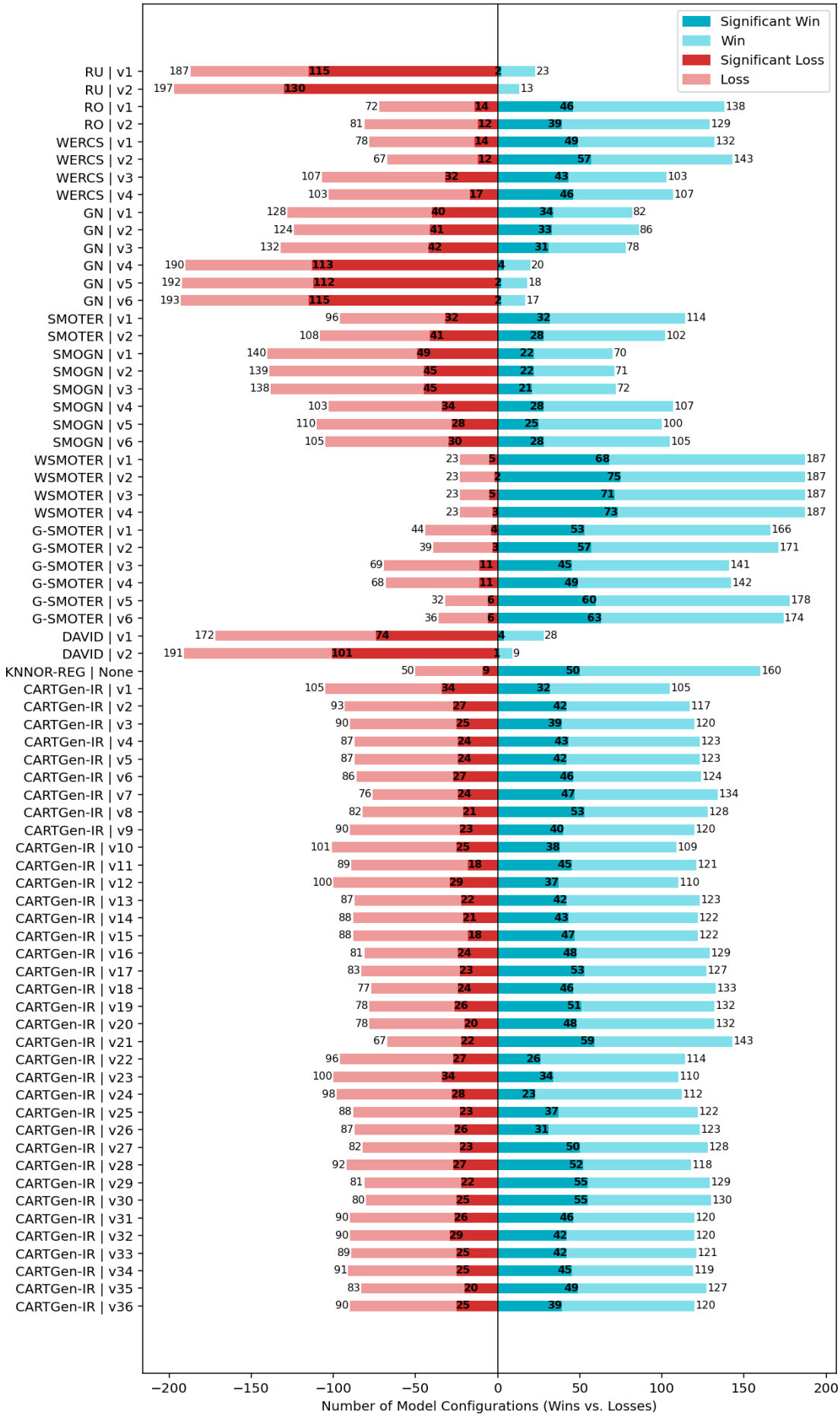


Figure B.3: RW-RMSE results by (significant) wins/losses vs. baseline scenario

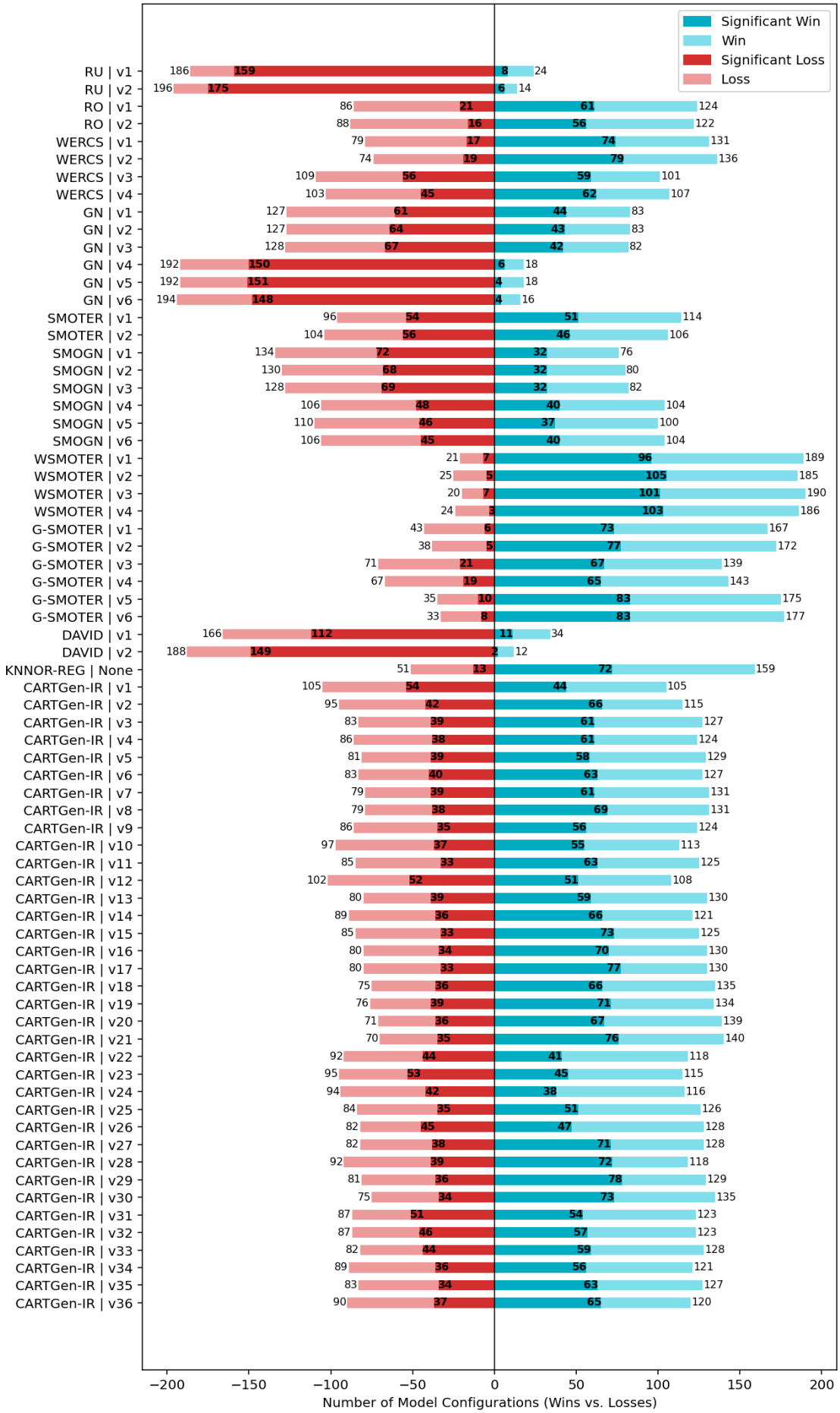


Figure B.4: SERA results by (significant) wins/losses vs. baseline scenario

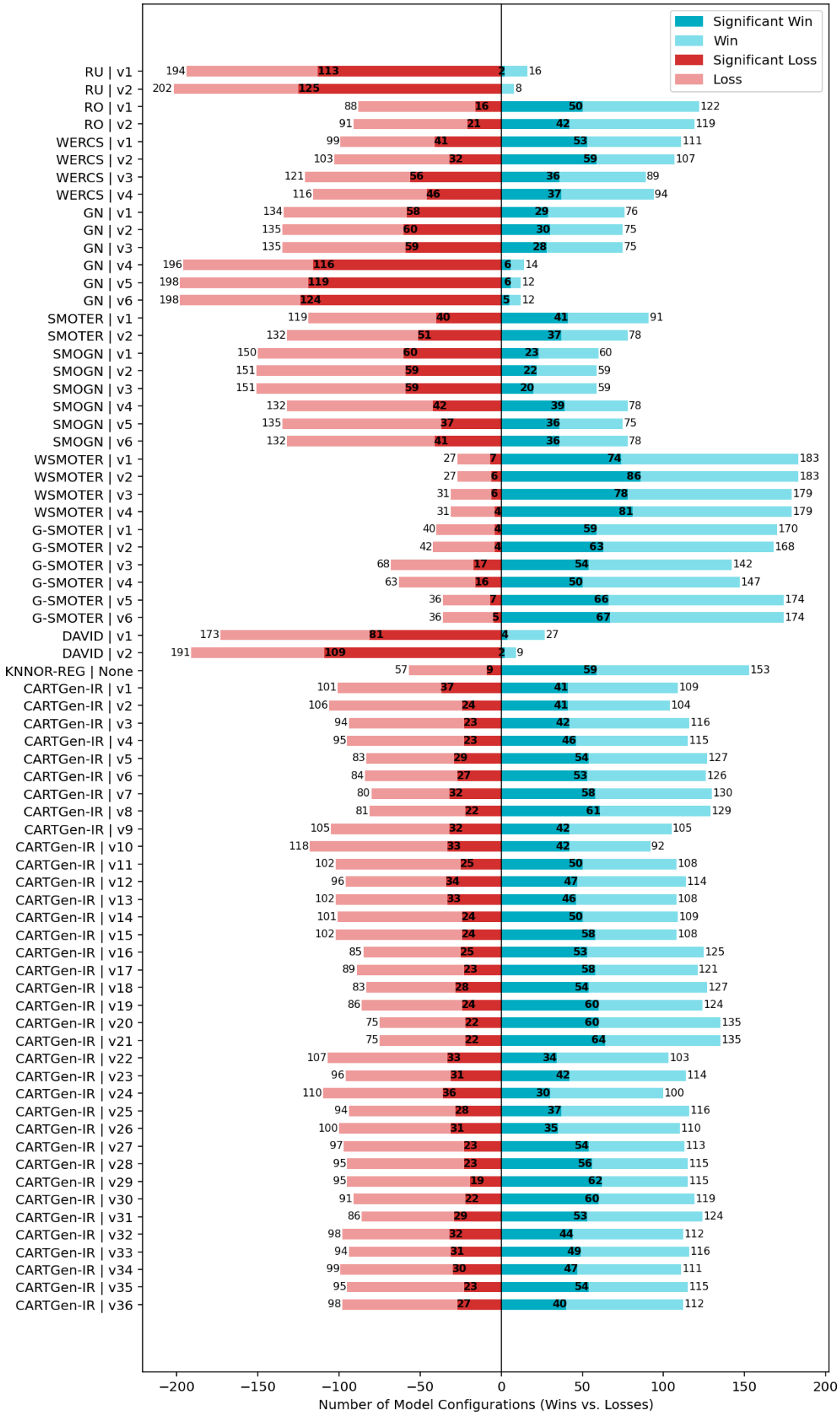


Figure B.5: DW-RMSE results by (significant) wins/losses vs. baseline scenario

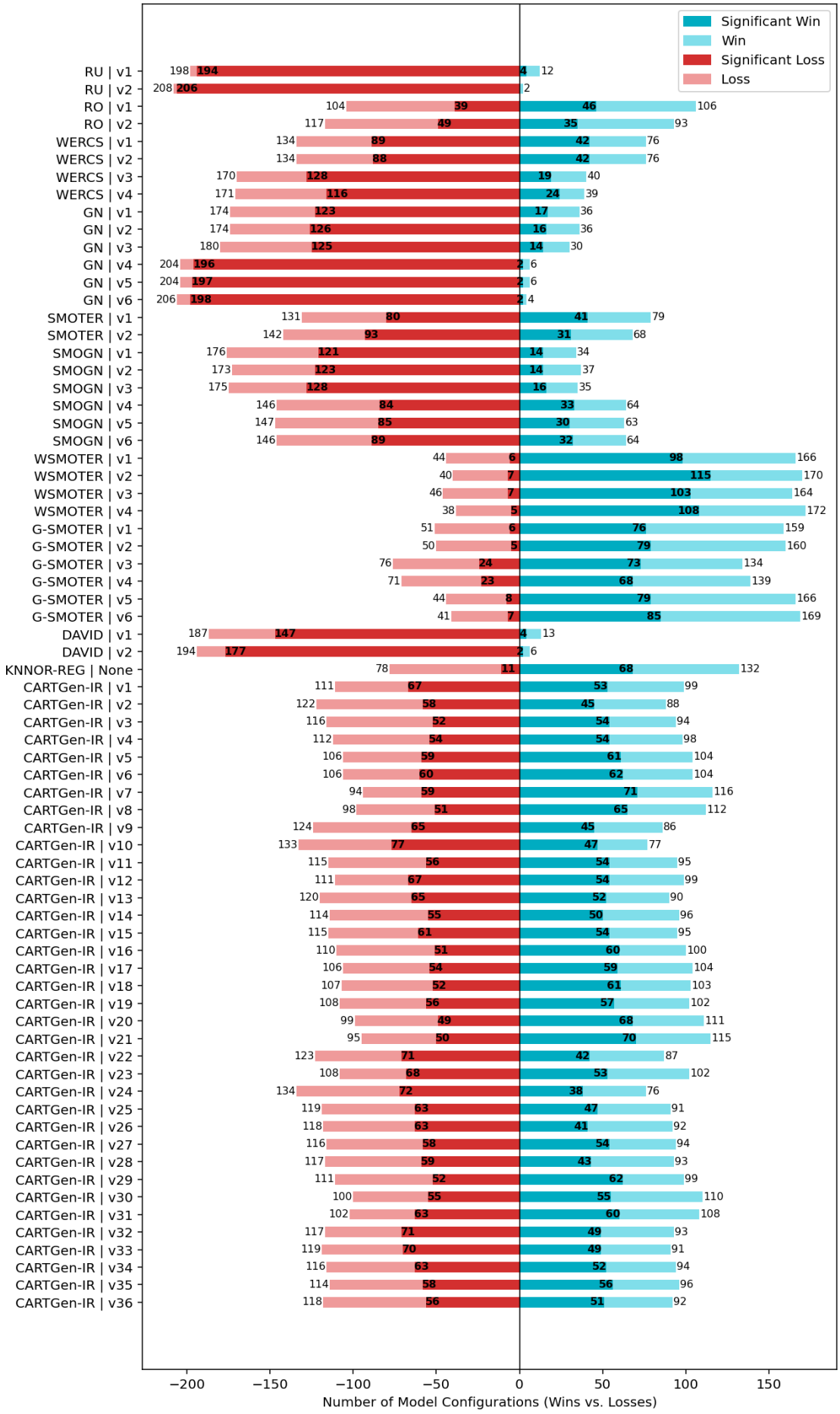


Figure B.6: DW-SERA results by (significant) wins/losses vs. baseline scenario

Table B.8: RMSE results by (significant) wins/losses vs. baseline scenario

Strategy	RF (Max. Wins = 60)		SVR (Max. Wins = 90)		XGBoost (Max. Wins = 60)	
	Win	**Loss**	**Win**	**Loss**	**Win**	**Loss**
RU | v1	0(0)	60(60)	5(2)	85(82)	0(0)	60(60)
RU | v2	0(0)	60(60)	0(0)	90(88)	0(0)	60(60)
RO | v1	23(7)	37(17)	42(29)	48(24)	13(0)	47(19)
RO | v2	19(6)	41(17)	34(26)	56(42)	12(2)	48(21)
WERCS | v1	0(0)	60(51)	32(19)	58(52)	2(0)	58(52)
WERCS | v2	2(0)	58(53)	30(22)	60(54)	0(0)	60(52)
WERCS | v3	0(0)	60(60)	18(8)	72(60)	0(0)	60(57)
WERCS | v4	0(0)	60(60)	20(14)	70(61)	0(0)	60(58)
GN | v1	1(0)	59(55)	16(13)	74(65)	0(0)	60(52)
GN | v2	2(0)	58(56)	16(11)	74(67)	0(0)	60(53)
GN | v3	0(0)	60(56)	14(6)	76(67)	0(0)	60(55)
GN | v4	0(0)	60(60)	2(2)	88(85)	0(0)	60(60)
GN | v5	0(0)	60(60)	2(2)	88(85)	0(0)	60(60)
GN | v6	0(0)	60(60)	2(2)	88(86)	0(0)	60(60)
SMOTER | v1	15(6)	45(28)	38(26)	52(27)	4(0)	56(36)
SMOTER | v2	15(5)	45(31)	24(21)	66(41)	2(0)	58(39)
SMOGN | v1	0(0)	60(46)	21(14)	69(62)	0(0)	60(46)
SMOGN | v2	0(0)	60(44)	21(14)	69(62)	0(0)	60(47)
SMOGN | v3	0(0)	60(46)	22(14)	68(62)	0(0)	60(51)
SMOGN | v4	17(5)	43(33)	22(19)	68(52)	6(1)	54(32)
SMOGN | v5	17(5)	43(32)	22(19)	68(53)	5(1)	55(35)
SMOGN | v6	15(4)	45(31)	20(20)	70(52)	4(1)	56(36)
WSMOTER | v1	43(14)	17(9)	69(52)	21(5)	29(4)	31(16)
WSMOTER | v2	43(19)	17(8)	66(50)	24(11)	34(7)	26(13)
WSMOTER | v3	42(17)	18(10)	66(49)	24(7)	27(5)	33(13)
WSMOTER | v4	43(20)	17(11)	66(47)	24(9)	32(5)	28(14)
G-SMOTER | v1	38(11)	22(2)	78(49)	12(5)	37(2)	23(4)
G-SMOTER | v2	35(12)	25(3)	77(50)	13(5)	36(2)	24(4)
G-SMOTER | v3	36(9)	24(5)	66(43)	24(14)	31(8)	29(7)
G-SMOTER | v4	35(12)	25(1)	66(41)	24(15)	35(2)	25(7)
G-SMOTER | v5	34(16)	26(5)	70(49)	20(1)	40(3)	20(10)
G-SMOTER | v6	39(14)	21(2)	69(49)	21(1)	39(6)	21(9)
DAVID | v1	3(0)	57(49)	6(4)	78(60)	0(0)	56(46)
DAVID | v2	4(0)	56(55)	0(0)	84(74)	0(0)	56(50)
KNNOR-REG | None	31(3)	29(4)	52(45)	38(13)	25(1)	35(8)
CARTGen-IR | v1	9(3)	51(30)	40(34)	50(38)	15(3)	45(27)
CARTGen-IR | v2	10(4)	50(31)	40(32)	50(38)	16(2)	44(29)
CARTGen-IR | v3	18(6)	42(25)	38(35)	52(41)	14(4)	46(27)
CARTGen-IR | v4	25(8)	35(21)	40(31)	50(41)	16(6)	44(26)
CARTGen-IR | v5	16(7)	44(26)	37(31)	53(39)	10(3)	50(34)
CARTGen-IR | v6	16(8)	44(24)	42(32)	48(34)	8(2)	52(37)
CARTGen-IR | v7	24(8)	36(23)	41(32)	49(37)	13(5)	47(28)
CARTGen-IR | v8	24(14)	36(21)	41(32)	49(34)	9(3)	51(28)
CARTGen-IR | v9	16(6)	44(24)	34(30)	56(40)	8(2)	52(22)
CARTGen-IR | v10	11(3)	49(26)	34(24)	56(44)	8(2)	52(27)
CARTGen-IR | v11	22(8)	38(22)	31(30)	59(42)	14(2)	46(22)
CARTGen-IR | v12	24(12)	36(22)	34(22)	56(41)	11(3)	49(23)
CARTGen-IR | v13	17(4)	43(29)	44(31)	46(36)	10(3)	50(31)
CARTGen-IR | v14	20(6)	40(28)	38(26)	52(39)	10(1)	50(29)
CARTGen-IR | v15	25(9)	35(22)	45(32)	45(38)	14(5)	46(27)
CARTGen-IR | v16	28(14)	32(27)	39(29)	51(36)	12(4)	48(26)
CARTGen-IR | v17	10(6)	50(19)	40(33)	50(34)	8(2)	52(37)
CARTGen-IR | v18	9(6)	51(22)	40(32)	50(38)	9(2)	51(37)
CARTGen-IR | v19	22(13)	38(17)	40(33)	50(35)	13(2)	47(28)
CARTGen-IR | v20	20(12)	40(25)	39(32)	51(37)	12(2)	48(32)
CARTGen-IR | v21	13(5)	47(19)	34(30)	56(42)	11(2)	49(22)
CARTGen-IR | v22	15(4)	45(22)	33(25)	57(46)	5(2)	55(24)
CARTGen-IR | v23	26(9)	34(19)	34(32)	56(40)	14(1)	46(19)
CARTGen-IR | v24	27(11)	33(18)	32(25)	58(46)	12(4)	48(22)
CARTGen-IR | v25	13(4)	47(29)	44(32)	46(34)	11(1)	49(30)
CARTGen-IR | v26	13(5)	47(28)	38(28)	52(39)	9(3)	51(29)
CARTGen-IR | v27	26(8)	34(19)	42(32)	48(33)	13(5)	47(26)
CARTGen-IR | v28	24(14)	36(24)	38(27)	52(39)	10(4)	50(30)
CARTGen-IR | v29	10(6)	50(21)	41(28)	49(38)	8(1)	52(36)
CARTGen-IR | v30	11(5)	49(20)	39(22)	51(39)	4(1)	56(37)
CARTGen-IR | v31	21(12)	39(19)	39(28)	51(39)	13(2)	47(29)
CARTGen-IR | v32	21(12)	39(15)	39(22)	51(37)	13(3)	47(32)
CARTGen-IR | v33	20(5)	40(23)	35(28)	55(40)	5(2)	55(25)
CARTGen-IR | v34	21(7)	39(21)	32(24)	58(48)	5(1)	55(30)
CARTGen-IR | v35	25(9)	35(20)	34(30)	56(41)	14(3)	46(19)
CARTGen-IR | v36	26(10)	34(18)	34(25)	56(46)	11(3)	49(23)

Table B.9: RW-RMSE results by (significant) wins/losses vs. baseline scenario

Strategy	RF (Max. Wins = 60)		SVR (Max. Wins = 90)		XGBoost (Max. Wins = 60)	
	Win	**Loss**	**Win**	**Loss**	**Win**	**Loss**
RU | v1	4(0)	56(34)	19(2)	71(45)	0(0)	60(36)
RU | v2	0(0)	60(40)	13(0)	77(50)	0(0)	60(40)
RO | v1	46(11)	14(1)	76(33)	14(1)	16(2)	44(12)
RO | v2	41(8)	19(1)	71(28)	19(2)	17(3)	43(9)
WERCS | v1	39(12)	21(0)	69(33)	21(5)	24(4)	36(9)
WERCS | v2	51(17)	9(0)	69(36)	21(2)	23(4)	37(10)
WERCS | v3	24(12)	36(11)	61(28)	29(8)	18(3)	42(13)
WERCS | v4	28(12)	32(4)	66(30)	24(5)	13(4)	47(8)
GN | v1	19(4)	41(10)	51(27)	39(13)	12(3)	48(17)
GN | v2	20(4)	40(9)	52(27)	38(12)	14(2)	46(20)
GN | v3	20(4)	40(9)	45(25)	45(12)	13(2)	47(21)
GN | v4	0(0)	60(34)	19(4)	71(41)	1(0)	59(38)
GN | v5	0(0)	60(34)	18(2)	72(41)	0(0)	60(37)
GN | v6	1(0)	59(33)	16(2)	74(46)	0(0)	60(36)
SMOTER | v1	35(4)	25(8)	64(28)	26(6)	15(0)	45(18)
SMOTER | v2	33(6)	27(8)	56(22)	34(12)	13(0)	47(21)
SMOGN | v1	15(0)	45(13)	48(22)	42(14)	7(0)	53(22)
SMOGN | v2	15(0)	45(10)	48(22)	42(14)	8(0)	52(21)
SMOGN | v3	21(0)	39(10)	46(21)	44(14)	5(0)	55(21)
SMOGN | v4	32(6)	28(8)	63(22)	27(8)	12(0)	48(18)
SMOGN | v5	30(3)	30(4)	63(22)	27(6)	7(0)	53(18)
SMOGN | v6	35(6)	25(4)	60(22)	30(7)	10(0)	50(19)
WSMOTER | v1	57(24)	3(0)	85(41)	5(2)	45(3)	15(3)
WSMOTER | v2	60(30)	0(0)	84(39)	6(0)	43(6)	17(2)
WSMOTER | v3	59(21)	1(0)	85(43)	5(1)	43(7)	17(4)
WSMOTER | v4	59(28)	1(0)	85(40)	5(0)	43(5)	17(3)
G-SMOTER | v1	49(14)	11(0)	82(39)	8(0)	35(0)	25(4)
G-SMOTER | v2	50(16)	10(0)	81(39)	9(0)	40(2)	20(3)
G-SMOTER | v3	44(12)	16(0)	69(30)	21(8)	28(3)	32(3)
G-SMOTER | v4	46(14)	14(0)	68(30)	22(9)	28(5)	32(2)
G-SMOTER | v5	56(18)	4(0)	83(38)	7(0)	39(4)	21(6)
G-SMOTER | v6	54(20)	6(0)	83(38)	7(0)	37(5)	23(6)
DAVID | v1	9(0)	51(24)	19(4)	65(26)	0(0)	56(24)
DAVID | v2	7(1)	53(31)	2(0)	82(40)	0(0)	56(30)
KNNOR-REG | None	48(14)	12(1)	76(32)	14(4)	36(4)	24(4)
CARTGen-IR | v1	26(5)	34(5)	53(26)	37(17)	26(1)	34(12)
CARTGen-IR | v2	33(10)	27(3)	50(26)	40(13)	27(1)	33(13)
CARTGen-IR | v3	37(7)	23(4)	51(26)	39(17)	22(1)	38(13)
CARTGen-IR | v4	41(14)	19(0)	52(30)	38(14)	27(2)	33(12)
CARTGen-IR | v5	42(12)	18(0)	58(29)	32(12)	20(1)	40(17)
CARTGen-IR | v6	40(13)	20(1)	64(28)	26(9)	17(1)	43(15)
CARTGen-IR | v7	41(14)	19(2)	61(29)	29(11)	17(2)	43(12)
CARTGen-IR | v8	45(19)	15(0)	69(29)	21(8)	13(1)	47(12)
CARTGen-IR | v9	42(12)	18(1)	57(27)	33(15)	21(0)	39(9)
CARTGen-IR | v10	38(11)	22(1)	57(30)	33(12)	22(1)	38(14)
CARTGen-IR | v11	43(11)	17(1)	56(26)	34(14)	21(2)	39(10)
CARTGen-IR | v12	45(13)	15(2)	59(30)	31(9)	19(0)	41(13)
CARTGen-IR | v13	43(11)	17(0)	61(31)	29(11)	19(0)	41(13)
CARTGen-IR | v14	42(14)	18(2)	58(31)	32(11)	24(1)	36(14)
CARTGen-IR | v15	47(14)	13(1)	61(31)	29(12)	26(2)	34(11)
CARTGen-IR | v16	47(18)	13(1)	59(33)	31(11)	22(2)	38(9)
CARTGen-IR | v17	40(15)	20(1)	64(24)	26(7)	16(1)	44(15)
CARTGen-IR | v18	36(12)	24(0)	62(26)	28(7)	11(0)	49(18)
CARTGen-IR | v19	43(21)	17(0)	62(23)	28(7)	16(1)	44(11)
CARTGen-IR | v20	44(17)	16(0)	63(25)	27(5)	16(0)	44(17)
CARTGen-IR | v21	44(12)	16(1)	59(28)	31(12)	19(3)	41(8)
CARTGen-IR | v22	47(13)	13(0)	58(31)	32(10)	17(3)	43(8)
CARTGen-IR | v23	46(18)	14(1)	59(28)	31(14)	24(2)	36(9)
CARTGen-IR | v24	49(18)	11(1)	58(30)	32(13)	20(5)	40(9)
CARTGen-IR | v25	46(11)	14(1)	65(35)	25(10)	22(0)	38(13)
CARTGen-IR | v26	45(16)	15(0)	65(34)	25(12)	22(1)	38(14)
CARTGen-IR | v27	51(12)	9(0)	64(34)	26(9)	17(2)	43(11)
CARTGen-IR | v28	51(20)	9(2)	66(34)	24(9)	26(5)	34(11)
CARTGen-IR | v29	37(6)	23(1)	64(20)	26(10)	13(0)	47(16)
CARTGen-IR | v30	37(5)	23(0)	62(18)	28(9)	13(0)	47(19)
CARTGen-IR | v31	42(15)	18(1)	62(20)	28(6)	18(2)	42(16)
CARTGen-IR | v32	46(14)	14(0)	58(17)	32(9)	19(0)	41(17)
CARTGen-IR | v33	45(14)	15(2)	60(33)	30(13)	23(3)	37(8)
CARTGen-IR | v34	43(19)	17(2)	59(31)	31(12)	16(2)	44(13)
CARTGen-IR | v35	48(19)	12(2)	60(32)	30(11)	21(4)	39(9)
CARTGen-IR | v36	50(19)	10(2)	60(33)	30(12)	20(3)	40(11)

Table B.10: SERA results by (significant) wins/losses vs. baseline scenario

Strategy	RF (Max. Wins = 60)		SVR (Max. Wins = 90)		XGBoost (Max. Wins = 60)	
	Win	**Loss**	**Win**	**Loss**	**Win**	**Loss**
RU | v1	2(0)	58(46)	20(8)	70(57)	2(0)	58(56)
RU | v2	0(0)	60(57)	14(6)	76(62)	0(0)	60(56)
RO | v1	40(13)	20(1)	71(46)	19(3)	13(2)	47(17)
RO | v2	38(10)	22(2)	69(43)	21(2)	15(3)	45(12)
WERCS | v1	34(17)	26(0)	70(50)	20(5)	27(7)	33(12)
WERCS | v2	45(20)	15(0)	69(54)	21(3)	22(5)	38(16)
WERCS | v3	24(19)	36(14)	58(38)	32(19)	19(2)	41(23)
WERCS | v4	35(16)	25(9)	58(40)	32(17)	14(6)	46(19)
GN | v1	20(5)	40(18)	51(36)	39(19)	12(3)	48(24)
GN | v2	22(4)	38(19)	50(37)	40(18)	11(2)	49(27)
GN | v3	24(5)	36(17)	46(35)	44(23)	12(2)	48(27)
GN | v4	0(0)	60(45)	18(6)	72(52)	0(0)	60(53)
GN | v5	0(0)	60(46)	18(4)	72(52)	0(0)	60(53)
GN | v6	0(0)	60(42)	16(4)	74(55)	0(0)	60(51)
SMOTER | v1	38(4)	22(16)	62(45)	28(11)	14(2)	46(27)
SMOTER | v2	36(6)	24(16)	57(40)	33(14)	13(0)	47(26)
SMOGN | v1	21(0)	39(22)	49(32)	41(22)	6(0)	54(28)
SMOGN | v2	22(0)	38(18)	49(32)	41(23)	9(0)	51(27)
SMOGN | v3	28(0)	32(18)	49(32)	41(22)	5(0)	55(29)
SMOGN | v4	30(5)	30(12)	63(35)	27(9)	11(0)	49(27)
SMOGN | v5	30(3)	30(9)	64(34)	26(10)	6(0)	54(27)
SMOGN | v6	31(6)	29(8)	62(34)	28(10)	11(0)	49(27)
WSMOTER | v1	60(29)	0(0)	85(64)	5(2)	44(3)	16(5)
WSMOTER | v2	60(36)	0(0)	85(62)	5(0)	40(7)	20(5)
WSMOTER | v3	60(28)	0(0)	86(65)	4(2)	44(8)	16(5)
WSMOTER | v4	59(33)	1(0)	86(65)	4(0)	41(5)	19(3)
G-SMOTER | v1	48(14)	12(0)	84(57)	6(0)	35(2)	25(6)
G-SMOTER | v2	49(18)	11(0)	84(57)	6(0)	39(2)	21(5)
G-SMOTER | v3	45(14)	15(3)	68(50)	22(13)	26(3)	34(5)
G-SMOTER | v4	45(15)	15(2)	68(49)	22(13)	30(1)	30(4)
G-SMOTER | v5	53(21)	7(0)	84(54)	6(0)	38(8)	22(10)
G-SMOTER | v6	54(22)	6(0)	85(55)	5(0)	38(6)	22(8)
DAVID | v1	12(1)	48(38)	22(10)	62(39)	0(0)	56(35)
DAVID | v2	8(2)	52(45)	4(0)	80(58)	0(0)	56(46)
KNNOR-REG | None	47(18)	13(0)	75(51)	15(6)	37(3)	23(7)
CARTGen-IR | v1	26(6)	34(8)	55(37)	35(26)	24(1)	36(20)
CARTGen-IR | v2	30(12)	30(8)	52(38)	38(23)	26(1)	34(21)
CARTGen-IR | v3	38(7)	22(7)	53(37)	37(26)	24(1)	36(20)
CARTGen-IR | v4	44(12)	16(4)	52(41)	38(26)	27(1)	33(21)
CARTGen-IR | v5	41(14)	19(2)	63(42)	27(17)	19(1)	41(27)
CARTGen-IR | v6	42(14)	18(4)	65(45)	25(18)	21(0)	39(22)
CARTGen-IR | v7	42(12)	18(2)	61(42)	29(15)	18(2)	42(19)
CARTGen-IR | v8	44(19)	16(1)	68(44)	22(12)	15(0)	45(21)
CARTGen-IR | v9	39(19)	21(1)	58(45)	32(21)	23(1)	37(15)
CARTGen-IR | v10	37(17)	23(3)	58(48)	32(19)	20(1)	40(20)
CARTGen-IR | v11	43(16)	17(4)	57(44)	33(18)	27(1)	33(17)
CARTGen-IR | v12	45(13)	15(2)	61(48)	29(18)	18(0)	42(18)
CARTGen-IR | v13	44(12)	16(0)	61(46)	29(18)	24(0)	36(21)
CARTGen-IR | v14	40(17)	20(1)	61(44)	29(18)	26(2)	34(21)
CARTGen-IR | v15	44(14)	16(2)	62(45)	28(18)	25(2)	35(19)
CARTGen-IR | v16	48(20)	12(2)	63(47)	27(17)	20(2)	40(19)
CARTGen-IR | v17	42(17)	18(3)	63(39)	27(11)	19(0)	41(21)
CARTGen-IR | v18	38(14)	22(2)	62(41)	28(11)	13(0)	47(24)
CARTGen-IR | v19	43(24)	17(0)	63(38)	27(14)	19(1)	41(19)
CARTGen-IR | v20	47(21)	13(3)	64(38)	26(10)	19(0)	41(26)
CARTGen-IR | v21	40(17)	20(0)	59(47)	31(20)	22(2)	38(16)
CARTGen-IR | v22	46(18)	14(1)	60(51)	30(18)	19(4)	41(14)
CARTGen-IR | v23	43(22)	17(2)	59(45)	31(17)	28(3)	32(15)
CARTGen-IR | v24	48(20)	12(1)	59(50)	31(16)	23(7)	37(16)
CARTGen-IR | v25	46(15)	14(2)	68(51)	22(15)	21(0)	39(19)
CARTGen-IR | v26	45(18)	15(1)	68(50)	22(16)	21(3)	39(22)
CARTGen-IR | v27	51(16)	9(0)	68(49)	22(17)	20(2)	40(19)
CARTGen-IR | v28	50(21)	10(1)	66(50)	24(16)	24(5)	36(18)
CARTGen-IR | v29	39(8)	21(5)	66(33)	24(16)	13(0)	47(23)
CARTGen-IR | v30	39(8)	21(1)	61(30)	29(14)	16(0)	44(27)
CARTGen-IR | v31	43(17)	17(3)	65(32)	25(10)	18(2)	42(22)
CARTGen-IR | v32	48(14)	12(3)	61(31)	29(17)	19(2)	41(25)
CARTGen-IR | v33	46(17)	14(3)	59(51)	31(21)	23(3)	37(14)
CARTGen-IR | v34	43(20)	17(1)	59(49)	31(18)	16(3)	44(20)
CARTGen-IR | v35	46(22)	14(5)	62(51)	28(16)	21(5)	39(15)
CARTGen-IR | v36	50(17)	10(3)	63(52)	27(16)	22(4)	38(15)

Table B.11: DW-RMSE results by (significant) wins/losses vs. baseline scenario

Strategy	RF (Max. Wins = 60)		SVR (Max. Wins = 90)		XGBoost (Max. Wins = 60)	
	Win	Loss	Win	Loss	Win	Loss
RU | v1	0(0)	60(32)	16(2)	74(43)	0(0)	60(38)
RU | v2	0(0)	60(36)	8(0)	82(49)	0(0)	60(40)
RO | v1	39(13)	21(0)	65(36)	25(3)	18(1)	42(13)
RO | v2	35(12)	25(1)	62(29)	28(6)	22(1)	38(14)
WERCS | v1	31(12)	29(8)	60(37)	30(8)	20(4)	40(25)
WERCS | v2	34(15)	26(8)	61(42)	29(5)	12(2)	48(19)
WERCS | v3	28(8)	32(16)	55(28)	35(12)	6(0)	54(28)
WERCS | v4	32(9)	28(13)	54(28)	36(7)	8(0)	52(26)
GN | v1	18(6)	42(21)	50(20)	40(15)	8(3)	52(22)
GN | v2	15(7)	45(18)	48(21)	42(20)	12(2)	48(22)
GN | v3	19(5)	41(20)	45(21)	45(18)	11(2)	49(21)
GN | v4	4(0)	56(32)	10(6)	80(44)	0(0)	60(40)
GN | v5	4(0)	56(33)	8(6)	82(47)	0(0)	60(39)
GN | v6	4(0)	56(35)	8(5)	82(50)	0(0)	60(39)
SMOTER | v1	28(7)	32(8)	55(34)	35(10)	8(0)	52(22)
SMOTER | v2	24(9)	36(12)	47(28)	43(16)	7(0)	53(23)
SMOGN | v1	11(0)	49(16)	44(23)	46(18)	5(0)	55(26)
SMOGN | v2	9(0)	51(14)	44(22)	46(19)	6(0)	54(26)
SMOGN | v3	11(0)	49(14)	44(20)	46(20)	4(0)	56(25)
SMOGN | v4	22(11)	38(9)	48(28)	42(13)	8(0)	52(20)
SMOGN | v5	22(9)	38(5)	48(27)	42(13)	5(0)	55(19)
SMOGN | v6	22(8)	38(5)	48(28)	42(13)	8(0)	52(23)
WSMOTER | v1	59(25)	1(0)	83(45)	7(2)	41(4)	19(5)
WSMOTER | v2	59(30)	1(0)	81(45)	9(1)	43(11)	17(5)
WSMOTER | v3	59(24)	1(0)	80(46)	10(2)	40(8)	20(4)
WSMOTER | v4	57(31)	3(0)	80(44)	10(0)	42(6)	18(4)
G-SMOTER | v1	49(12)	11(0)	80(46)	10(0)	41(1)	19(4)
G-SMOTER | v2	45(16)	15(0)	80(46)	10(0)	43(1)	17(4)
G-SMOTER | v3	45(13)	15(0)	69(36)	21(10)	28(5)	32(7)
G-SMOTER | v4	46(12)	14(1)	69(36)	21(8)	32(2)	28(7)
G-SMOTER | v5	48(20)	12(0)	85(45)	5(0)	41(1)	19(7)
G-SMOTER | v6	52(18)	8(0)	85(45)	5(0)	37(4)	23(5)
DAVID | v1	11(0)	49(29)	16(4)	68(30)	0(0)	56(22)
DAVID | v2	4(0)	56(35)	5(2)	79(42)	0(0)	56(32)
KNNOR-REG | None	48(15)	12(1)	73(40)	17(3)	32(4)	28(5)
CARTGen-IR | v1	29(9)	31(5)	57(31)	33(18)	23(1)	37(14)
CARTGen-IR | v2	37(10)	23(1)	56(35)	34(17)	21(2)	39(16)
CARTGen-IR | v3	38(10)	22(0)	55(31)	35(17)	21(1)	39(14)
CARTGen-IR | v4	47(13)	13(0)	56(36)	34(17)	21(4)	39(12)
CARTGen-IR | v5	43(14)	17(0)	54(30)	36(13)	15(0)	45(19)
CARTGen-IR | v6	44(14)	16(1)	57(34)	33(11)	15(1)	45(19)
CARTGen-IR | v7	43(15)	17(1)	55(30)	35(13)	13(2)	47(16)
CARTGen-IR | v8	45(19)	15(1)	58(34)	32(9)	12(1)	48(13)
CARTGen-IR | v9	39(10)	21(2)	53(29)	37(14)	20(1)	40(11)
CARTGen-IR | v10	37(10)	23(0)	50(30)	40(10)	17(1)	43(14)
CARTGen-IR | v11	45(12)	15(0)	52(28)	38(12)	19(2)	41(11)
CARTGen-IR | v12	47(14)	13(0)	50(31)	40(11)	18(1)	42(12)
CARTGen-IR | v13	47(15)	13(0)	60(37)	30(12)	20(2)	40(17)
CARTGen-IR | v14	44(15)	16(0)	59(35)	31(13)	23(3)	37(14)
CARTGen-IR | v15	48(19)	12(0)	59(37)	31(14)	23(2)	37(18)
CARTGen-IR | v16	50(21)	10(0)	59(38)	31(11)	20(2)	40(11)
CARTGen-IR | v17	36(12)	24(0)	53(30)	37(12)	16(0)	44(20)
CARTGen-IR | v18	33(12)	27(0)	49(30)	41(14)	10(0)	50(19)
CARTGen-IR | v19	42(20)	18(1)	52(29)	38(13)	14(1)	46(11)
CARTGen-IR | v20	40(17)	20(1)	52(29)	38(13)	16(0)	44(19)
CARTGen-IR | v21	40(14)	20(0)	52(32)	38(11)	17(4)	43(13)
CARTGen-IR | v22	42(16)	18(0)	52(37)	38(12)	14(5)	46(12)
CARTGen-IR | v23	47(19)	13(0)	56(31)	34(14)	22(3)	38(11)
CARTGen-IR | v24	48(17)	12(0)	55(34)	35(11)	18(7)	42(12)
CARTGen-IR | v25	44(15)	16(1)	63(38)	27(11)	20(1)	40(16)
CARTGen-IR | v26	41(17)	19(0)	63(39)	27(9)	20(4)	40(15)
CARTGen-IR | v27	53(19)	7(0)	66(38)	24(10)	16(3)	44(12)
CARTGen-IR | v28	47(22)	13(0)	63(38)	27(9)	25(4)	35(13)
CARTGen-IR | v29	37(10)	23(0)	54(24)	36(13)	12(0)	48(20)
CARTGen-IR | v30	38(6)	22(0)	50(24)	40(12)	12(0)	48(24)
CARTGen-IR | v31	42(13)	18(0)	55(24)	35(11)	19(0)	41(17)
CARTGen-IR | v32	43(12)	17(0)	50(23)	40(12)	17(0)	43(19)
CARTGen-IR | v33	40(13)	20(0)	56(35)	34(12)	17(6)	43(11)
CARTGen-IR | v34	45(18)	15(1)	54(35)	36(10)	16(3)	44(12)
CARTGen-IR | v35	47(19)	13(0)	53(37)	37(10)	15(6)	45(9)
CARTGen-IR | v36	47(19)	13(0)	54(36)	36(8)	18(5)	42(14)

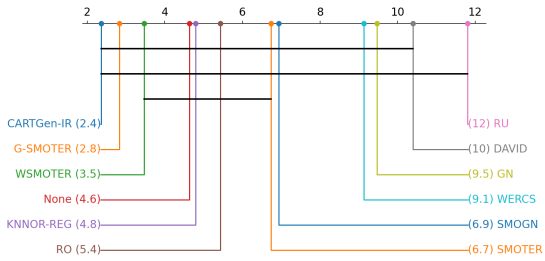
Table B.12: DW-SERA results by (significant) wins/losses vs. baseline scenario

Strategy	RF (Max. Wins = 60)		SVR (Max. Wins = 90)		XGBoost (Max. Wins = 60)	
	Win	Loss	Win	Loss	Win	Loss
RU | v1	0(0)	60(60)	12(4)	78(74)	0(0)	60(60)
RU | v2	0(0)	60(60)	2(0)	88(86)	0(0)	60(60)
RO | v1	29(7)	31(4)	58(39)	32(17)	19(0)	41(18)
RO | v2	25(7)	35(6)	50(27)	40(23)	18(1)	42(20)
WERCS | v1	22(5)	38(27)	48(36)	42(24)	6(1)	54(38)
WERCS | v2	20(8)	40(21)	50(34)	40(26)	6(0)	54(41)
WERCS | v3	0(0)	60(37)	38(19)	52(39)	2(0)	58(52)
WERCS | v4	3(0)	57(28)	34(24)	56(36)	2(0)	58(52)
GN | v1	5(0)	55(40)	29(17)	61(43)	2(0)	58(40)
GN | v2	7(0)	53(41)	27(16)	63(45)	2(0)	58(40)
GN | v3	3(0)	57(38)	25(14)	65(47)	2(0)	58(40)
GN | v4	0(0)	60(56)	6(2)	84(80)	0(0)	60(60)
GN | v5	0(0)	60(56)	6(2)	84(81)	0(0)	60(60)
GN | v6	0(0)	60(56)	4(2)	86(82)	0(0)	60(60)
SMOTER | v1	25(6)	35(24)	52(35)	38(24)	2(0)	58(32)
SMOTER | v2	22(6)	38(27)	42(25)	48(31)	4(0)	56(35)
SMOGN | v1	1(0)	59(31)	32(14)	58(48)	1(0)	59(42)
SMOGN | v2	3(0)	57(33)	32(14)	58(48)	2(0)	58(42)
SMOGN | v3	2(0)	58(35)	32(16)	58(48)	1(0)	59(45)
SMOGN | v4	20(8)	40(23)	37(24)	53(32)	7(1)	53(29)
SMOGN | v5	20(6)	40(22)	37(24)	53(32)	6(0)	54(31)
SMOGN | v6	20(7)	40(22)	39(24)	51(31)	5(1)	55(36)
WSMOTER | v1	51(27)	9(0)	82(65)	8(0)	33(6)	27(6)
WSMOTER | v2	50(38)	10(0)	79(67)	11(1)	41(10)	19(6)
WSMOTER | v3	47(31)	13(0)	78(66)	12(2)	39(6)	21(5)
WSMOTER | v4	50(37)	10(0)	82(62)	8(0)	40(9)	20(5)
G-SMOTER | v1	42(14)	18(1)	81(61)	9(1)	36(1)	24(4)
G-SMOTER | v2	42(17)	18(0)	82(60)	8(1)	36(2)	24(4)
G-SMOTER | v3	39(13)	21(3)	70(52)	20(14)	25(8)	35(7)
G-SMOTER | v4	40(14)	20(2)	70(52)	20(14)	29(2)	31(7)
G-SMOTER | v5	44(16)	16(1)	86(57)	4(1)	36(6)	24(6)
G-SMOTER | v6	47(20)	13(0)	85(58)	5(1)	37(7)	23(6)
DAVID | v1	3(0)	57(45)	10(4)	74(58)	0(0)	56(44)
DAVID | v2	4(0)	56(55)	2(2)	82(72)	0(0)	56(50)
KNNOR-REG | None	43(11)	17(1)	64(54)	26(4)	25(3)	35(6)
CARTGen-IR | v1	23(9)	37(13)	54(40)	36(30)	22(4)	38(24)
CARTGen-IR | v2	26(8)	34(13)	54(43)	36(29)	19(3)	41(25)
CARTGen-IR | v3	30(8)	30(11)	52(41)	38(32)	20(4)	40(25)
CARTGen-IR | v4	36(15)	24(8)	54(41)	36(32)	18(4)	42(23)
CARTGen-IR | v5	31(8)	29(9)	51(39)	39(31)	11(2)	49(31)
CARTGen-IR | v6	29(8)	31(9)	51(39)	39(29)	11(2)	49(32)
CARTGen-IR | v7	33(11)	27(9)	50(39)	40(29)	11(2)	49(25)
CARTGen-IR | v8	34(15)	26(10)	49(39)	41(25)	13(2)	47(23)
CARTGen-IR | v9	28(10)	32(8)	47(38)	43(29)	17(3)	43(19)
CARTGen-IR | v10	27(9)	33(9)	47(34)	43(28)	14(2)	46(21)
CARTGen-IR | v11	31(15)	29(6)	44(38)	46(28)	19(1)	41(18)
CARTGen-IR | v12	33(15)	27(6)	48(36)	42(30)	17(3)	43(18)
CARTGen-IR | v13	33(13)	27(8)	55(46)	35(24)	16(2)	44(27)
CARTGen-IR | v14	33(15)	27(9)	55(44)	35(26)	16(3)	44(25)
CARTGen-IR | v15	40(16)	20(7)	56(49)	34(26)	20(6)	40(26)
CARTGen-IR | v16	38(15)	22(6)	55(47)	35(24)	19(3)	41(21)
CARTGen-IR | v17	25(6)	35(10)	47(38)	43(25)	14(1)	46(30)
CARTGen-IR | v18	21(7)	39(14)	44(39)	46(29)	12(1)	48(34)
CARTGen-IR | v19	30(14)	30(7)	48(39)	42(28)	17(1)	43(21)
CARTGen-IR | v20	29(14)	31(12)	45(37)	45(23)	16(1)	44(30)
CARTGen-IR | v21	29(10)	31(6)	50(40)	40(30)	17(0)	43(19)
CARTGen-IR | v22	31(11)	29(11)	51(40)	39(32)	13(3)	47(18)
CARTGen-IR | v23	33(17)	27(7)	48(41)	42(29)	19(2)	41(15)
CARTGen-IR | v24	36(17)	24(9)	51(38)	39(28)	17(4)	43(17)
CARTGen-IR | v25	29(10)	31(6)	60(49)	30(20)	14(2)	46(26)
CARTGen-IR | v26	29(9)	31(9)	58(46)	32(21)	15(2)	45(26)
CARTGen-IR | v27	38(13)	22(8)	58(50)	32(19)	15(5)	45(22)
CARTGen-IR | v28	37(18)	23(6)	61(49)	29(20)	17(3)	43(24)
CARTGen-IR | v29	27(7)	33(10)	49(34)	41(28)	11(1)	49(33)
CARTGen-IR | v30	22(5)	38(9)	47(32)	43(30)	7(1)	53(33)
CARTGen-IR | v31	29(13)	31(9)	44(33)	46(28)	18(1)	42(26)
CARTGen-IR | v32	27(12)	33(6)	48(28)	42(29)	17(1)	43(28)
CARTGen-IR | v33	30(12)	30(10)	51(41)	39(30)	13(1)	47(18)
CARTGen-IR | v34	31(6)	29(8)	50(35)	40(30)	12(2)	48(21)
CARTGen-IR | v35	32(18)	28(8)	50(41)	40(30)	17(3)	43(14)
CARTGen-IR | v36	37(16)	23(9)	54(35)	36(28)	19(4)	41(18)

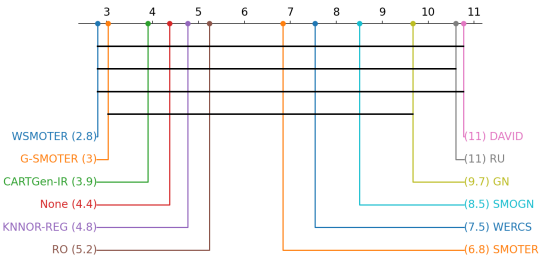
B.4. Friedman/Post-Hoc Nemenyi Test Results and Critical Difference Diagrams

Table B.13: Friedman test statistics per metric and learner, with statistically significant p-values ($p < 0.05$) highlighted

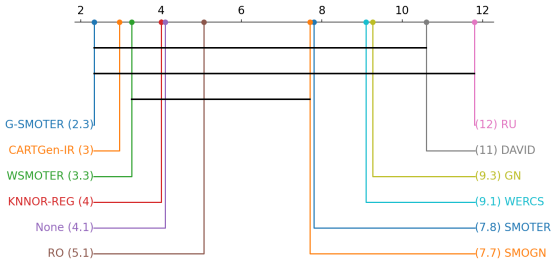
Metric	Learner	Friedman Statistic	p-value	Significance
RMSE	RF	123.1373	4.2e-21	True
	SVR	106.6007	8.7e-18	True
	XGBoost	135.0944	1.6e-23	True
RW-RMSE	RF	86.3691	8.5e-14	True
	SVR	75.6623	1.0e-11	True
	XGBoost	104.0617	2.8e-17	True
SERA	RF	88.2847	3.6e-14	True
	SVR	70.9370	8.1e-11	True
	XGBoost	103.3661	3.8e-17	True
DW-RMSE	RF	95.7860	1.2e-15	True
	SVR	84.0775	2.4e-13	True
	XGBoost	106.9165	7.6e-18	True
DW-SERA	RF	109.4205	2.4e-18	True
	SVR	92.3933	5.7e-15	True
	XGBoost	125.1599	1.7e-21	True



(a) RMSE – RF

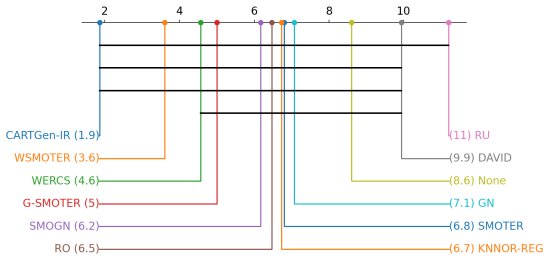


(b) RMSE – SVR

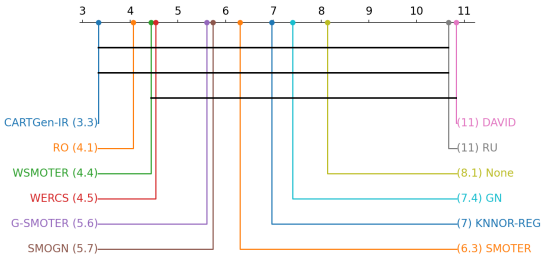


(c) RMSE – XGBoost

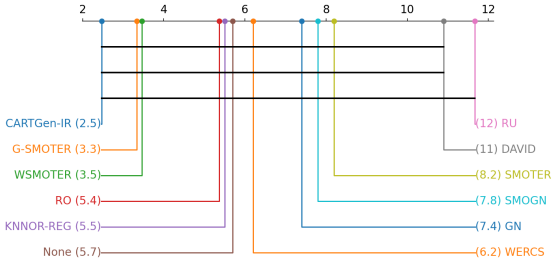
Figure B.7: Critical Difference diagrams for RMSE across learners



(a) RW-RMSE – RF

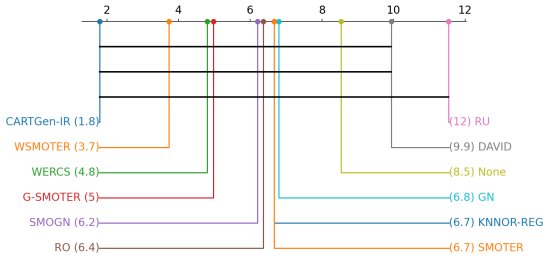


(b) RW-RMSE – SVR

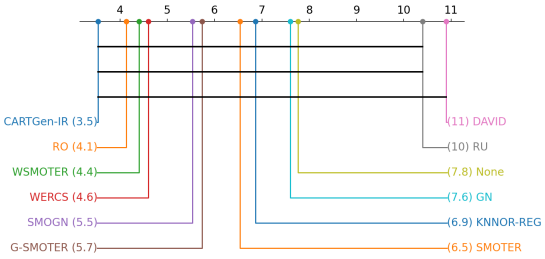


(c) RW-RMSE – XGBoost

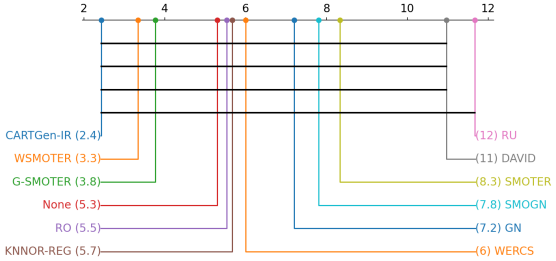
Figure B.8: Critical Difference diagrams for RW-RMSE across learners



(a) SERA – RF

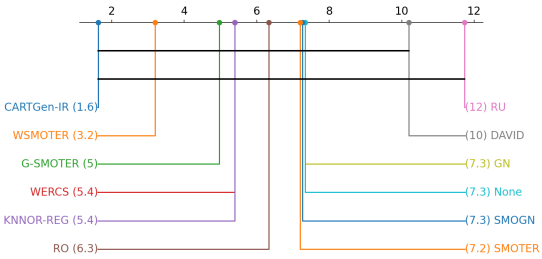


(b) SERA – SVR

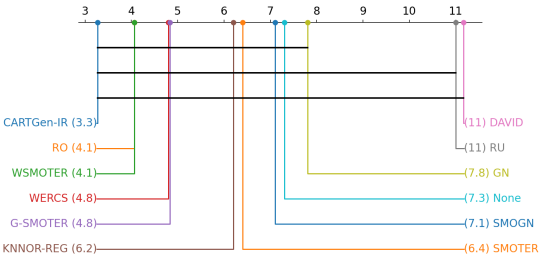


(c) SERA – XGBoost

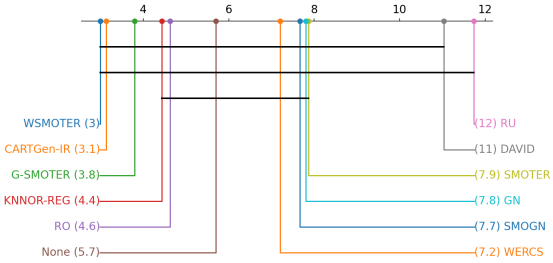
Figure B.9: Critical Difference diagrams for SERA across learners



(a) DW-RMSE – RF

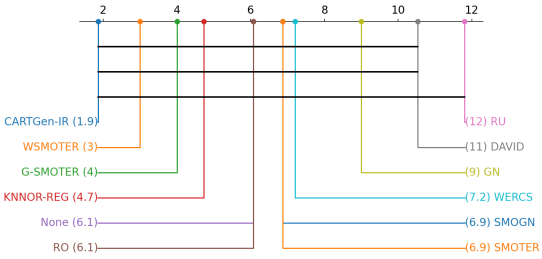


(b) DW-RMSE – SVR

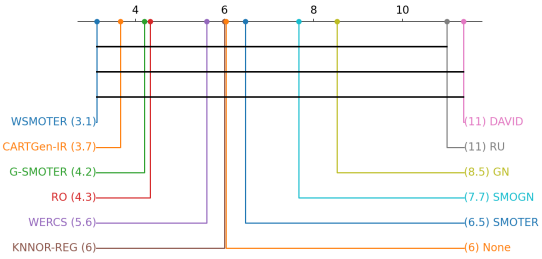


(c) DW-RMSE – XGBoost

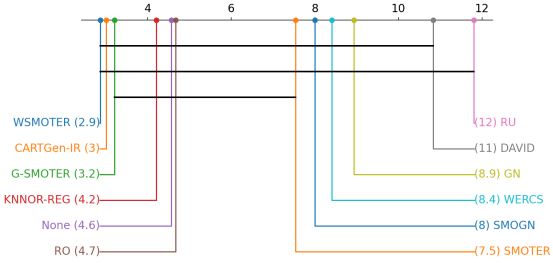
Figure B.10: Critical Difference diagrams for DW-RMSE across learners



(a) DW-SERA – RF



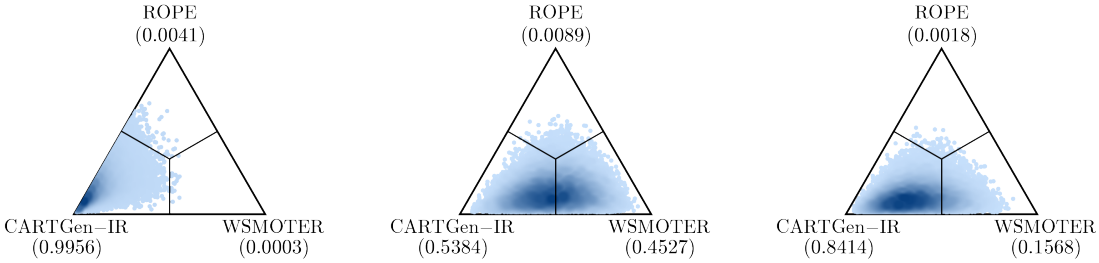
(b) DW-SERA – SVR



(c) DW-SERA – XGBoost

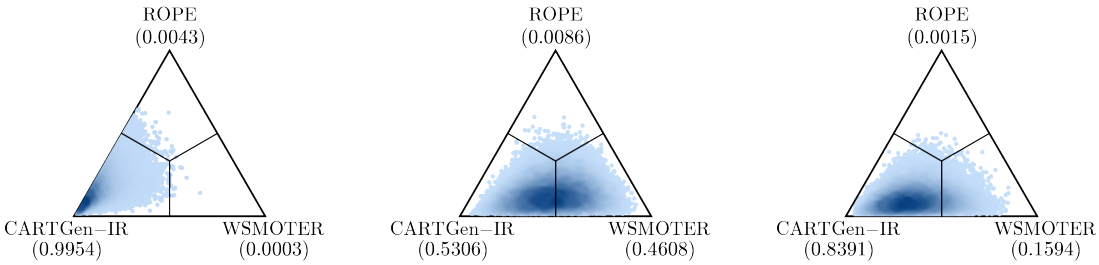
Figure B.11: Critical Difference diagrams for DW-SERA across learners

B.5. Bayesian Signed-Rank Test Results



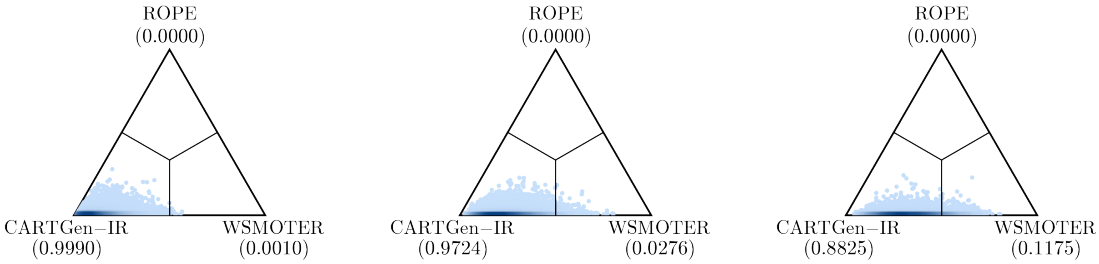
(a) RMSE – RF (b) RMSE – SVR (c) RMSE – XGBoost

Figure B.12: Bayesian Posterior Ternary Plot for RMSE: CARTGen-IR vs. WSMOTER



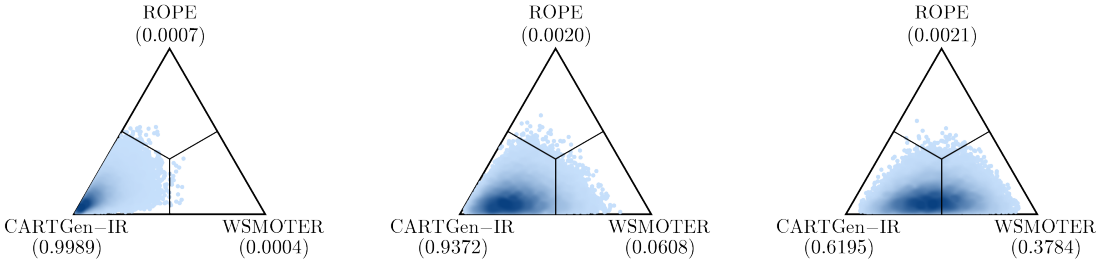
(a) RW-RMSE – RF (b) RW-RMSE – SVR (c) RW-RMSE – XGBoost

Figure B.13: Bayesian Posterior Ternary Plot for RW-RMSE: CARTGen-IR vs. WSMOTER



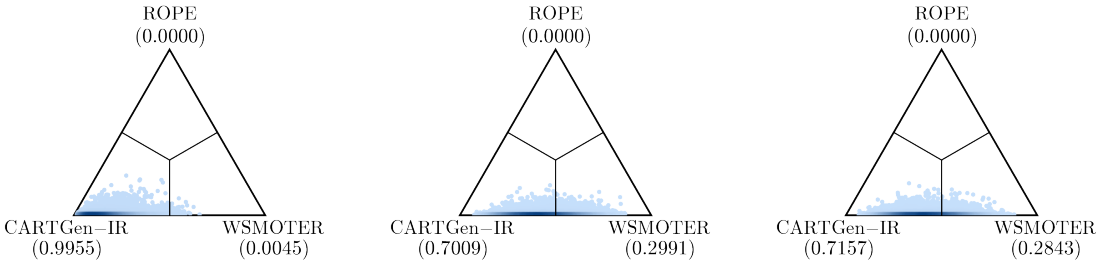
(a) SERA – RF (b) SERA – SVR (c) SERA – XGBoost

Figure B.14: Bayesian Posterior Ternary Plot for SERA: CARTGen-IR vs. WSMOTER



(a) DW-RMSE – RF (b) DW-RMSE – SVR (c) DW-RMSE – XGBoost

Figure B.15: Bayesian Posterior Ternary Plot for DW-RMSE: CARTGen-IR vs. WSMOTER

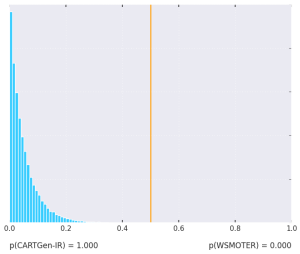


(a) DW-SERA – RF (b) DW-SERA – SVR (c) DW-SERA – XGBoost

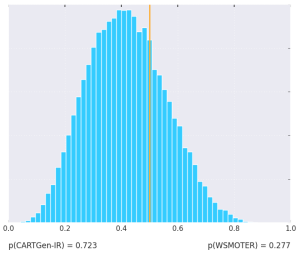
Figure B.16: Bayesian Posterior Ternary Plot for DW-SERA: CARTGen-IR vs. WSMOTER

Table B.14: Bayesian Signed-Rank Test: CARTGen-IR vs. WSMOTER

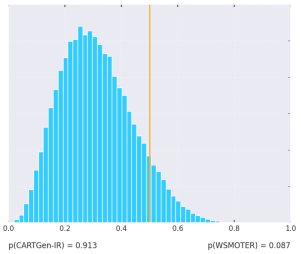
Metric	**Learner**	**Bayesian Probabilities**			**Number of Paired Results**
		$P(\text{CARTGen-IR} > \text{WSMOTER})$	$P(\text{Equivalent})$	$P(\text{WSMOTER} > \text{CARTGen-IR})$	
RMSE	RF	0.9956	0.0041	0.0003	150
	SVR	0.5384	0.0089	0.4527	150
	XGBoost	0.8414	0.0018	0.1568	150
RW-RMSE	RF	0.9954	0.0043	0.0003	150
	SVR	0.5306	0.0086	0.4608	150
	XGBoost	0.8391	0.0015	0.1594	150
SERA	RF	0.9990	0.0000	0.0010	150
	SVR	0.9724	0.0000	0.0276	150
	XGBoost	0.8825	0.0000	0.1175	150
DW-RMSE	RF	0.9989	0.0007	0.0004	150
	SVR	0.9372	0.0020	0.0608	150
	XGBoost	0.6195	0.0021	0.3784	150
DW-SERA	RF	0.9955	0.0000	0.0045	150
	SVR	0.7009	0.0000	0.2991	150
	XGBoost	0.7157	0.0000	0.2843	150



(a) RMSE – RF

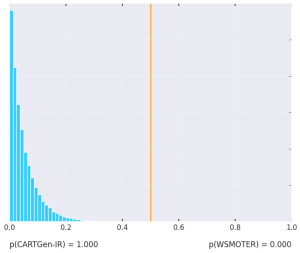


(b) RMSE – SVR

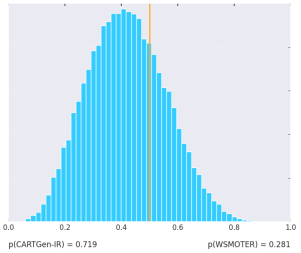


(c) RMSE – XGBoost

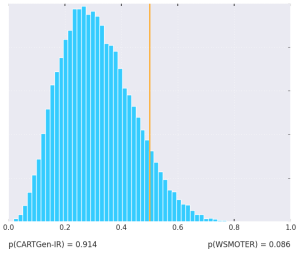
Figure B.17: Bayesian Posterior Distributions for RMSE: CARTGen-IR vs. WSMOTER



(a) RW-RMSE – RF

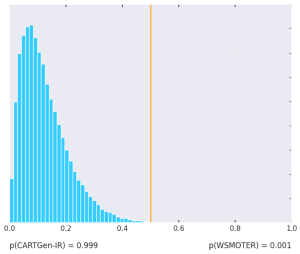


(b) RW-RMSE – SVR

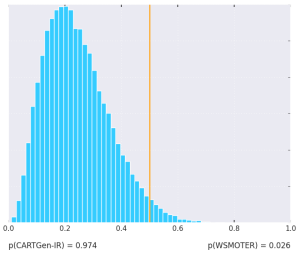


(c) RW-RMSE – XGBoost

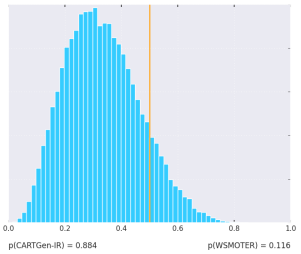
Figure B.18: Bayesian Posterior Distributions for RW-RMSE: CARTGen-IR vs. WSMOTER



(a) SERA – RF

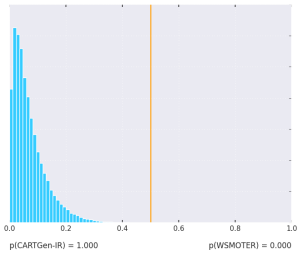


(b) SERA – SVR

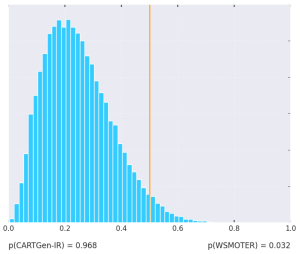


(c) SERA – XGBoost

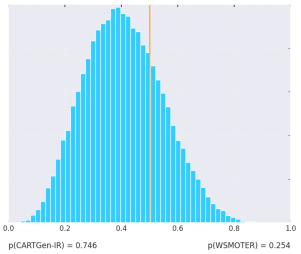
Figure B.19: Bayesian Posterior Distributions for SERA: CARTGen-IR vs. WSMOTER



(a) DW-RMSE – RF

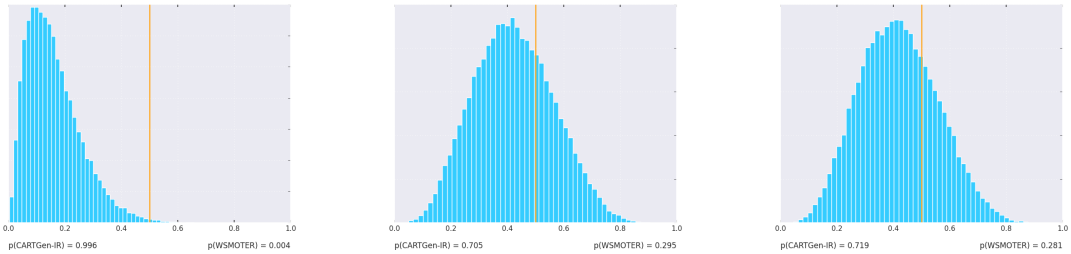


(b) DW-RMSE – SVR



(c) DW-RMSE – XGBoost

Figure B.20: Bayesian Posterior Distributions for DW-RMSE: CARTGen-IR vs. WSMOTER



(a) DW-SERA – RF (b) DW-SERA – SVR (c) DW-SERA – XGBoost

Figure B.21: Bayesian Posterior Distributions for DW-SERA: CARTGen-IR vs. WSMOTER

Table B.15: Runtime average (std. dev.) per preprocessing strategy and dataset

Dataset	RU	RO	WERCS	GN	SMOTER	SMOGN	WSMOTER	G-SMOTER	DAVID	KNNOR-REG	CARTGen-IR
abalone	0.1 (0)	0.15 (0)	0.003 (0)	0.894 (0.002)	20.576 (0.068)	20.836 (0.064)	0.396 (0.008)	0.392 (0)	90.638 (0.276)	0.034 (0.003)	0.262 (0.001)
airfoil	0.022 (0)	0.038 (0.001)	0.001 (0)	0.193 (0)	0.416 (0.001)	0.429 (0.001)	0.123 (0.002)	0.982 (0.002)	10.372 (0.002)	0.02 (0.001)	0.093 (0.001)
availablePower	0.043 (0)	0.118 (0.001)	0.002 (0)	0.591 (0.003)	4.175 (0.014)	4.362 (0.003)	0.406 (0.002)	0.131 (0.001)	16.357 (0.054)	0.098 (0.002)	0.122 (0.002)
cpuSm	0.262 (0.001)	0.481 (0.001)	0.004 (0)	3.361 (0.001)	9.303 (0.082)	9.341 (0.065)	0.693 (0.004)	0.655 (0.002)	65.834 (0.004)	0.039 (0.001)	0.723 (0.003)
ele-1	0.006 (0)	0.01 (0)	0.001 (0)	0.034 (0)	0.075 (0)	0.077 (0)	0.047 (0.001)	0.059 (0)	0.333 (0.002)	0.01 (0.001)	0.053 (0.002)
forestFires	0.016 (0)	0.029 (0)	0.001 (0)	0.131 (0)	0.13 (0.002)	0.215 (0.001)	0.245 (0)	0.252 (0)	5.758 (0.035)	0.009 (0.001)	0.08 (0)
fuelConsumption	0.083 (0)	0.282 (0.002)	0.004 (0)	1.499 (0.008)	2.793 (0.008)	3.704 (0.002)	0.384 (0.007)	0.032 (0.002)	20.292 (0.323)	0.015 (0.001)	0.255 (0.002)
heat	0.158 (0.002)	0.387 (0.002)	0.005 (0)	2.421 (0)	30.896 (0.041)	31.38 (0.119)	1.648 (0.003)	0.484 (0.002)	83.531 (0.081)	0.094 (0.066)	0.457 (0.001)
housingBoston	0.015 (0)	0.032 (0)	0.001 (0)	0.13 (0.001)	0.391 (0.001)	0.421 (0.002)	0.044 (0.001)	0.023 (0)	6.482 (0.078)	0.011 (0.001)	0.087 (0.001)
maxTorque	0.078 (0.001)	0.25 (0)	0.003 (0)	1.235 (0.003)	5.746 (0.008)	6.593 (0.018)	0.396 (0.003)	0.38 (0.001)	20.411 (0.121)	0.111 (0.003)	0.249 (0.002)
mortgage	0.034 (0)	0.058 (0.001)	0.001 (0)	0.324 (0.003)	1.101 (0.003)	1.116 (0.006)	0.088 (0.001)	0.093 (0)	12.86 (0.125)	0.016 (0)	0.113 (0.002)
sensory	0.012 (0.003)	0.033 (0)	0.001 (0)	0.142 (0.001)	0.25 (0.002)	0.357 (0)	0.073 (0.005)	0.065 (0.001)	5.227 (0.018)	0.009 (0.001)	0.079 (0)
servo	0.006 (0)	0.01 (0)	0.001 (0)	0.017 (0)	0.1 (0)	0.11 (0.004)	0.021 (0.001)	0.012 (0)	4.895 (0.015)	0.01 (0)	0.053 (0)
strikes	0.01 (0)	0.02 (0)	0.001 (0)	0.086 (0)	0.104 (0)	0.144 (0)	0.062 (0.001)	0.064 (0)	5.738 (0.005)	0.01 (0.001)	0.066 (0.003)
treasury	0.047 (0.01)	0.055 (0)	0.001 (0)	0.312 (0)	1.154 (0.002)	1.166 (0.002)	0.089 (0.001)	0.034 (0)	12.427 (0.1)	0.016 (0.001)	0.056 (0.001)
Average	0.059	0.130	0.002	0.758	5.147	5.350	0.314	0.244	24.077	0.034	0.183
Std. Dev.	0.001	0.001	0.000	0.002	0.016	0.019	0.003	0.001	0.083	0.005	0.001

C. Implementation Details

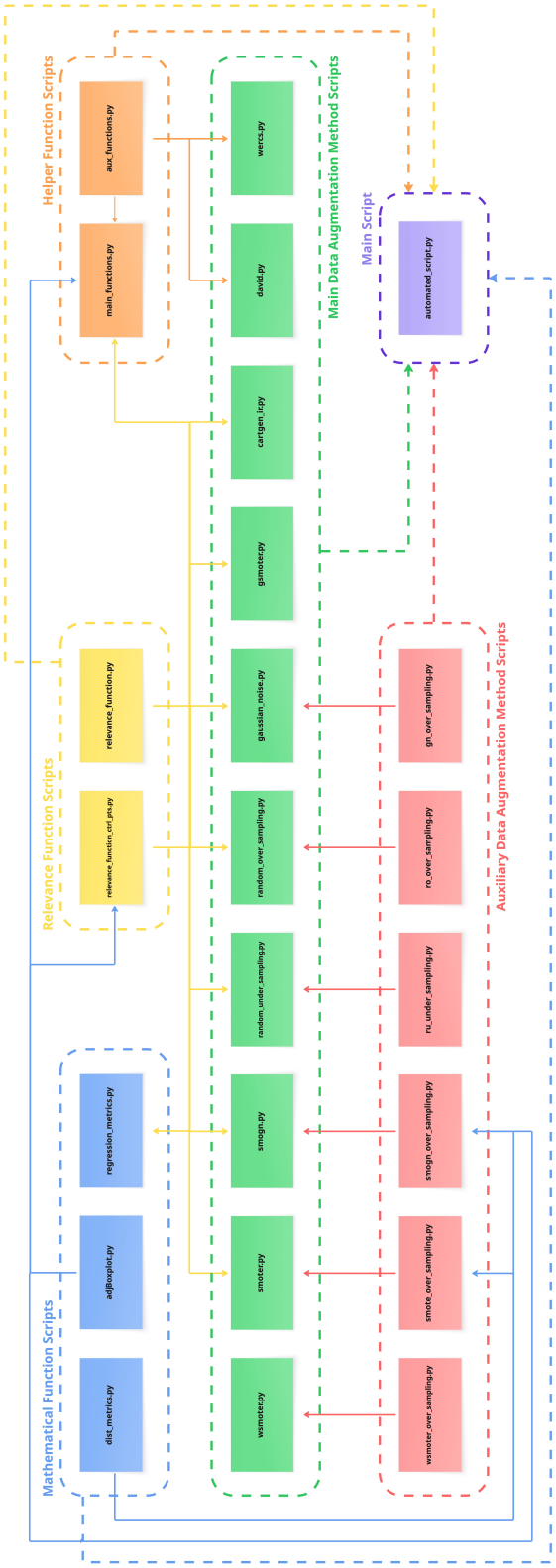


Figure C.1: Scripts Interdependencies

Table C.1: Code Metadata

Field	**Description**
Repository Link	`antoniopedropi/SyntheticTabularDataAugmentation-IR`
Reproducible Capsule Link	`codeocean.com/capsule/7826905/tree`
DOI	10.24433/CO.7826905.v2 [94]
Code/Software Licence	MIT
Data Licence	No Rights Reserved (CC0)
Version Control System	git
Software Code Language	Python
Package Requirements	denseweight==0.1.2
	imbalanced_learn==0.13.0
	joblib==1.4.2
	knnor_reg==0.0.3
	matplotlib==3.10.3
	numpy==2.3.0
	pandas==2.3.0
	python_synthpop==0.1.1
	scikit_learn==1.6.1
	scikit_posthocs==0.11.4
	scipy==1.15.3
	seaborn==0.13.2
	statsmodels==0.14.4
	torch==2.7.0
	tqdm==4.67.1
	xgboost==3.0.2
Support Contact	`up201704931@up.pt`