FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Optimised Interpretable Ordinal Scorecards

Mafalda Bastos da Costa

For Jury Evaluation



Master in Informatics and Computing Engineering

Supervisor: Jaime Cardoso

Second Supervisor: Luís Filipe Teixeira

September 5, 2025

© Mafalda Bastos da Costa, 2025

Optimised Interpretable Ordinal Scorecards

Mafalda Bastos da Costa

Master in Informatics and Computing Engineering

September 5, 2025

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

The specific Sustainable Development Goals addressed in this work are:

SDG 3 Ensure healthy lives and promote well-being for all at all ages.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialisation and foster innovation.

SGD	Target	Contribution	Performance Indicators and Metrics
3	3.1	Building interpretable and transparent decision support tools to support healthcare professionals.	Accuracy and number of rules in the scorecard.
9	9.1	Contribution to scientific research by proposing machine learning methods to be used by non-experts.	Reproducibility of the code.

Resumo

Os *scorecards* são amplamente utilizados em processos de tomada de decisão devido à sua transparência, simplicidade e interpretabilidade. Um *scorecard* é um modelo preditivo que atribui pontos a variáveis com base na sua contribuição para um resultado. A soma desses pontos produz uma pontuação total que pode ser usada para fazer uma classificação ou estimar a probabilidade de um determinado evento.

As abordagens tradicionais para a construção de scorecards baseiam-se num processo de duas etapas: primeiro, as variáveis contínuas são discretizadas e codificadas; depois, um modelo preditivo calcula os pesos para todas as variáveis. Além disso, muitos métodos de estado de arte focam-se exclusivamente na estimativa de pesos, assumindo uma etapa prévia de pré-processamento para a discretização das variáveis. Consequentemente, não existem algoritmos desenvolvidos que combinem ambas as etapas num processo unificado. De modo a abordar esta falha na literatura, esta dissertação apresenta o *Infinitesimal Bins*, um novo algoritmo de discretização concebido para aproximar a construção de *scorecards* a uma única etapa.

Adicionalmente, a literatura recente enfatiza abordagens baseadas em otimização em vez de abordagens baseadas em *machine learning* para o design de *scorecards*, particularmente para *scorecards* binários, sendo o RiskSLIM o método mais notável. No entanto, a sua extensão para classificação ordinal permanece pouco explorada. Este trabalho aborda esta questão, adaptando o RiskSLIM para dados ordinais através de um algoritmo de replicação de dados.

É dada especial ênfase às aplicações na área da saúde, onde os resultados ordinais são mais comuns e a interpretabilidade e transparência dos modelos são cruciais para o uso clínico. Em particular, esta tese investiga a previsão de resultados estéticos após o tratamento conservador do cancro da mama.

A avaliação experimental mostra que o *Infinitesimal Bins*, em comparação com o discretizador de referência, tem tendência a produzir modelos de maior dimensão. A sua granularidade pode ser benéfica em pequenos conjuntos de dados, mas leva frequentemente a modelos demasiado complexos em conjuntos de dados maiores, indicando a necessidade de refinamento através da fusão ou eliminação de *bins*. No que diz respeito a métodos de codificação, o *Differential Encoding* supera consistentemente o *One-Hot Encoding*, reduzindo o *overfitting*, melhorando a esparsidade e alcançando melhor desempenho preditivo. Entre os classificadores, nos problemas binários o modelo *Generic Generalised Linear Estimator (skglm)*, que induz esparsidade, alcançou o melhor equilíbrio entre tamanho do modelo e capacidade preditiva. Para problemas ordinais, contudo, não existe um classificador que se destaque de forma consistente; portanto, a escolha deve depender do equilíbrio desejado entre precisão e complexidade. Análises exploratórias revelaram que a seleção manual de atributos e as estratégias *ensemble* melhoram a interpretabilidade e o desempenho. Finalmente, a extensão do *RiskSLIM* para a classificação ordinal demonstra a viabilidade de abordagens baseadas em otimização neste contexto.

Palavras-chave: Scorecards ordinais; Machine Learning interpretável; Otimização; Discretização

Abstract

Scorecards are widely used in decision-making due to their transparency, simplicity, and interpretability. A scorecard is a predictive model that assigns points to features based on their contribution to an outcome. The sum of these points produces a total score that can be used to make a classification or estimate the probability of a given event.

Traditional approaches to scorecard construction rely on a two-step process: first, continuous features are discretised and encoded; then, a predictive model calculates weights (points) for all features. Furthermore, current state-of-the-art methods focus solely on weight estimation, assuming a prior pre-processing step for feature discretisation. This leaves a gap in the development of algorithms that combine both steps into a unified process. To address this, this thesis introduces *Infinitesimal Bins*, a novel discretisation algorithm designed to approximate scorecard construction as a one-step algorithm.

Additionally, recent literature emphasises optimisation-based approaches over machine learning-based ones for scorecard design, particularly for binary scorecards, with RiskSLIM being the most notable method. However, its extension to ordinal classification remains underexplored. This work addresses this issue by adapting RiskSLIM for ordinal data through a Data Replication framework.

Special emphasis is placed on healthcare applications, where ordinal outcomes are more common and the interpretability and transparency of models are crucial for clinical usage. In particular, this thesis investigates the prediction of aesthetic results after breast cancer conservative treatment.

The experimental evaluation shows that Infinitesimal Bins, compared to the baseline discretiser, tends to produce larger models. Its granularity can be beneficial for small datasets, but often leads to excessively complex models in larger datasets, indicating the need for refinement through bin merging or pruning. In terms of encoding methods, Differential Coding consistently outperforms One-Hot Encoding by reducing overfitting, improving sparsity, and achieving higher accuracy. Among classifiers, in binary tasks, the sparsity-inducing Generic Generalised Linear Estimator (skglm) model achieved the best balance between compactness and predictive performance. For ordinal tasks, however, there is no single classifier that proves consistently superior; therefore, the choice of classifier should depend on the desired trade-off. Exploratory analyses further revealed that manual feature selection and ensemble strategies enhance interpretability and performance. Additionally, the extension of RiskSLIM to ordinal classification demonstrates the feasibility of optimisation-based approaches in this setting.

Keywords: Ordinal Scorecards; Interpretable Machine Learning; Optimisation; Discretisation

Acknowledgements

I would like to thank my supervisors, Professor Jaime Cardoso and Professor Luís Filipe Teixeira, for their support and guidance.

I am grateful to INESC TEC, more specifically to the Health from Portugal project, for the resources and support provided.

Thank you to all my friends, especially Manuel, Mariana, Inês, and Leonor, for the discussions about my work, for the encouragement and for the great company.

I am very grateful to my family for their love and constant support.

Finally, I would like to thank the researchers and authors whose work served as the basis and inspiration for this dissertation.

Mafalda Costa

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 1
- 1.3 Objectives and Contributions . . . 2
- 1.4 Document Structure . . . 3

2 Background **4**
- 2.1 Binary and Ordinal Classification . . . 4
- 2.2 Generalised Additive Models . . . 4
- 2.3 Regularisation . . . 5
- 2.4 Interpretability . . . 6
- 2.5 Optimisation . . . 6

3 State of the Art **8**
- 3.1 Handcrafted Scoring Systems . . . 8
- 3.2 Automatic Binary Scoring Systems . . . 10
  - 3.2.1 SLIM and RiskSLIM . . . 10
  - 3.2.2 Interval Coded Scoring . . . 12
  - 3.2.3 AutoScore . . . 13
  - 3.2.4 PSORisk . . . 14
  - 3.2.5 FasterRisk . . . 14
- 3.3 Automatic Multiclass Scoring Systems . . . 16
  - 3.3.1 Differential Coding for Binary and Ordinal Scorecards . . . 16
  - 3.3.2 Multi-Objective Symbolic Regression . . . 17
  - 3.3.3 FAIRScoringSystems . . . 18
  - 3.3.4 MISS . . . 18
- 3.4 Applications . . . 19
  - 3.4.1 Optimisation-based Scoring Systems . . . 19
  - 3.4.2 Machine Learning-based Scoring Systems . . . 21
- 3.5 Comparative Summary of Scoring System Methods . . . 22
- 3.6 Conclusion . . . 23

4 Proposed Methodology **24**
- 4.1 Overall Construction of a Scorecard for Continuous Features . . . 24
- 4.2 Single Binary Classifier . . . 25
- 4.3 Discretisation Techniques . . . 26
  - 4.3.1 CAIM . . . 26
  - 4.3.2 Infinitesimal Bins . . . 28

4.4 Encoding Techniques . . . 30
4.4.1 One-Hot Encoding . . . 30
4.4.2 Differential Coding . . . 30
4.5 Classifier . . . 31
4.5.1 Logistic Regression . . . 31
4.5.2 Support Vector Machines . . . 31
4.5.3 Adaptive Lasso . . . 32
4.5.4 Generic Generalised Linear Estimator (skglm) . . . 32
4.5.5 RiskSLIM . . . 33

5 Experimental Evaluation **34**
5.1 Datasets . . . 34
5.1.1 Preprocessing . . . 34
5.1.2 Aesthetic Evaluation Dataset . . . 35
5.1.3 Binary datasets . . . 35
5.1.4 Ordinal datasets . . . 35
5.2 Performance Metrics . . . 36
5.3 Implementation Details . . . 36
5.4 Results . . . 37
5.4.1 Comparison between Discretisation Techniques . . . 37
5.4.2 Comparison between Encoding Techniques . . . 41
5.4.3 Comparison between Classifiers . . . 44
5.5 Exploratory Analysis . . . 48
5.5.1 Manual Feature Selection . . . 48
5.5.2 Ensemble of Scorecards . . . 49
5.5.3 RiskSLIM . . . 51
5.6 Conclusions . . . 51

6 Conclusions and Future Work **54**

References **56**

A Performance Metrics for each Dataset **62**

B Declaration on the Use of AI-assisted Tools **68**

List of Figures

1.1 **Example of a Scorecard**, from the SAS Institute (2006) [55], that predicts whether an applicant should be granted credit based on age, income, and homeownership status. 2

4.1 Data Replication Method example [9]. 27

5.1 **Test accuracy** (y-axis) **versus model size** (x-axis) **for each discretisation technique** across **binary datasets**. Each point represents a classifier combined with one discretisation method: CAIM (blue) or Infinitesimal Bins (orange). Encoding was performed with Differential Coding. 38

5.2 **Test accuracy** (y-axis) **versus model size** (x-axis) **for each discretisation technique** across **ordinal datasets**. Each point represents a classifier combined with one discretisation method: CAIM (blue) or Infinitesimal Bins (orange). Encoding was performed with Differential Coding. 39

5.3 **Test accuracy** (y-axis) **versus model size** (x-axis) **for each encoding technique** across **binary datasets**. Each point represents a classifier combined with one encoding method: One-Hot Encoding (red) or Differential Coding (green). Discretisation was performed with Infinitesimal Bins. 42

5.4 **Test accuracy** (y-axis) **versus model size** (x-axis) **for each encoding technique** across **ordinal datasets**. Each point represents a classifier combined with one encoding method: One-Hot Encoding (red) or Differential Coding (green). Discretisation was performed with Infinitesimal Bins. 43

5.5 Confusion matrices for Logistic Regression (a) and skglm (b) on the `aesthetic ordinal` dataset with Infinitesimal Bins and One-Hot Encoding. 44

5.6 **Test accuracy** (y-axis) **versus model size** (x-axis) **for each classifier** across **binary datasets**. Each point corresponds to a classifier (Logistic Regression in red, SVM in green, Adaptive Lasso in blue, and skglm in orange). Discretisation was performed using CAIM and encoding with Differential Coding. 45

5.7 **Test accuracy** (y-axis) **versus model size** (x-axis) **for each classifier** across **ordinal datasets**. Each point corresponds to a classifier (Logistic Regression in red, SVM in green, Adaptive Lasso in blue, and skglm in orange). Discretisation was performed using CAIM and encoding with Differential Coding. 47

List of Tables

3.1 $CHADS_2$ scorecard for stroke risk assessment. 8
3.2 Apgar scorecard for newborn evaluation. 9
3.3 Example of a multiclass scoring system generated by *FAIRScoringSystems* for the `customer` dataset [49]. Each table corresponds to a scoring system for a single class. The attribute "starts with" is the bias. 18
3.4 MISS model for the `iris` dataset, as presented by Grzeszczyk et al. (2024) [23]. This dataset has three classes: "Class 1" corresponds to *setosa*, "Class 2" to *versicolor*, and "Class 3" to *virginica*. Each row represents a binary feature, and the values indicate how much that feature contributes to the total score of each class. 19
3.5 Comparative summary of scoring system methods. 23

4.1 General structure of the quanta matrix [56]. 28

5.1 Binary datasets used for testing, where n is the number of samples, d is the number of features, and y is target variable. 35
5.2 Ordinal datasets used for testing, where n is the number of samples, d is the number of features, K is the number of target classes, and K_1 to K_7 is the number of observations (and proportion in percentage) of each class. 35
5.3 **Comparing discretisation techniques**: CAIM vs Infinitesimal Bins. Performance results per **binary** dataset, discretisation method, and model. The encoding technique used is Differential Coding. Each model reports the test accuracy and the model size. 40
5.4 **Comparing discretisation techniques**: CAIM vs Infinitesimal Bins. Performance results per **ordinal** dataset, discretisation method, and model. The encoding technique used is Differential Coding. Each model reports the test accuracy and the model size. 40
5.5 **Comparing encoding techniques**: One-Hot Encoding vs. Differential Coding. Performance results per **binary** dataset, encoding method, and model. The discretisation technique used is Infinitesimal Bins. Each model reports the test accuracy and the model size. 41
5.6 **Comparing encoding techniques**: One-Hot Encoding vs. Differential Coding. Performance results per **ordinal** dataset, encoding method, and model. The discretisation technique used is Infinitesimal Bins. Each model reports the test accuracy and the model size. 41
5.7 **Comparing classifiers**: Logistic Regression vs. SVM vs. Adaptive Lasso vs. skglm. Performance results per **binary** dataset and classifier. Each classifier reports the train accuracy, the test accuracy, and the model size. The discretisation was done using CAIM, and the encoding was Differential Coding. 46

5.8 **Comparing classifiers**: Logistic Regression vs. SVM vs. Adaptive Lasso vs. skglm. Performance results per **ordinal** dataset and classifier. Each classifier reports the train accuracy, the test accuracy, and the model size. The discretisation was done using CAIM, and the encoding was Differential Coding. 47
5.9 `aesthetic ordinal` dataset results for different discretisation, encoding, and classifier combinations. The comparison is made between using **all features versus manually selected features**. Performance is reported as test accuracy (in %) and model size, measured by the number of non-zero coefficients. 49
5.10 Existing scorecard for the `aesthetic ordinal` dataset. Points are assigned per feature. 50
5.11 Performance of scorecards without and with the ensemble on the `aesthetic ordinal` dataset. Results are shown for two discretisation techniques (CAIM and Infinitesimal Bins) and three classifiers (Logistic Regression, SVM and skglm), using Differential Coding for encoding. Metrics reported include train accuracy, test accuracy, and model size. 50
5.12 Results on the `aesthetic` **ordinal** dataset with `author = cmavioso` for different discretisation methods and classifiers, including RiskSLIM. The results include test accuracy, train accuracy and model size. For RiskSLIM, two decision thresholds are shown in brackets: 0.50 (default) and the one with the best training performance. For CAIM with RiskSLIM, the maximum model size was restricted to 42 (due to CPLEX limits). The encoding technique used was Differential Coding. 52
5.13 Results on the `aesthetic` **binary** dataset with `author = cmavioso` for different discretisation methods and classifiers, including RiskSLIM. The results include test accuracy, train accuracy and model size. For RiskSLIM, two decision thresholds are shown in brackets: 0.50 (default) and the one with the best training performance. The encoding technique used was Differential Coding. 52

A.1 Performance results of `appendicitis` for different discretisation, encoding and classifier combinations. 62
A.2 Performance results of `mammo` for different discretisation, encoding and classifier combinations. 63
A.3 Performance results of `breast cancer` for different discretisation, encoding and classifier combinations. 63
A.4 Performance results of `aesthetic binary` for different discretisation, encoding and classifier combinations. 63
A.5 Performance results of `aesthetic binary` with **features selected manually** for different discretisation, encoding and classifier combinations. 64
A.6 Performance results of `aesthetic binary` with **features selected manually and** `author = cmavioso` for different discretisation, encoding and classifier combinations. 64
A.7 Performance results of `spambase` for different discretisation, encoding and classifier combinations. 64
A.8 Performance results of `income` for different discretisation, encoding and classifier combinations. 65
A.9 Performance results of `small wine` for different discretisation, encoding and classifier combinations. 65
A.10 Performance results of `aesthetic ordinal` for different discretisation, encoding and classifier combinations. 65

A.11 Performance results of `aesthetic ordinal` with **features selected manually** for different discretisation, encoding and classifier combinations. 66
A.12 Performance results of `aesthetic ordinal` with **features selected manually and** `author = cmavioso` for different discretisation, encoding and classifier combinations. 66
A.13 Performance results of `aesthetic ordinal` with the **ensemble** of scorecards for different discretisation, encoding and classifier combinations. 66
A.14 Performance results of `abalone` for different discretisation, encoding and classifier combinations. 67
A.15 Performance results of `large wine` for different discretisation, encoding and classifier combinations. 67

Abbreviations and Symbols

AI	Artificial Intelligence
APACHE	Acute Physiology and Chronic Health Evaluation
BCCT	Breast Cancer Conservative Treatment
CAIM	Class-Attribute Interdependence Maximisation
CPA	Cutting Plane Algorithm
enICS	elastic net Interval Coded Scoring
FCAIM	Fuzzy Class-Attribute Interdependence Maximisation
GAMs	Generalised Additive Models
GLMs	Generalised Linear Models
GP	Genetic Programming
ICS	Interval Coded Scoring
ICU	Intensive Care Unit
IP	Integer Programming
LASSO	Least Absolute Shrinkage and Selection Operator
LCPA	Lattice Cutting Plane Algorithm
lpICS	linear programming Interval Coded Scoring
MINLP	Mixed-Integer Nonlinear Programming
MIP	Mixed-Integer Programming
MISS	Multiclass Interpretable Scoring Systems
MOSR	Multi-Objective Symbolic Regression
MSE	Mean Squared Error
PSO	Particle Swarm Optimisation
RiskSLIM	Risk-calibrated Supersparse Linear Integer Model
SAPS	Simplified Acute Physiology Score
SBC	Single Binary Classifier
SLIM	Supersparse Linear Integer Model
SOFA	Sequential Organ Failure Assessment
SVM	Support Vector Machines

Chapter 1

Introduction

1.1 Context

Scorecards are among the simplest predictive models. To make a prediction using a scorecard, a user only needs to add and subtract a few small numbers. This can be done without a computer or any special training, making scorecards suitable for real-world high-stakes scenarios.

These models are widely used in areas such as credit scoring and healthcare, not only because they are intuitive and interpretable, but also highly predictive. Interpretability in this context means that a decision maker can easily see, for example, how much a patient's characteristic contributes to their risk score. In high-stakes decisions, this interpretability is key to trusting artificial intelligence (AI) models. In contrast, black-box models require a post hoc explanation, which only approximates their reasoning and lacks transparency [51].

The traditional approach to scorecard development follows two main steps: (1) predictive variables are discretised, meaning that the continuous features are divided into discrete intervals; (2) a model is fitted to the data, learning the scores of each interval of every feature [22]. The final score for an instance is generated by summing the points of the intervals to which each feature belongs. The prediction is then made by comparing this final score to a pre-defined threshold.

Scorecards are primarily designed for binary classification problems - to predict a "yes" or "no" outcome - as in the following example. Figure 1.1 is an illustrated example of a scorecard from the SAS Institute (2006) [55]. In this example, with a threshold of 600 points, the scorecard predicts whether applicants should be accepted for credit based on age, income, and homeownership status. For a 35-year-old applicant (210 points), earning $38000 (225 points), and owning a home (225 points), the total score sums up to 660 points. The applicant would be approved for credit since this score exceeds the 600-point threshold.

1.2 Motivation

As mentioned, scorecards are most often used for binary classification. However, the data is ordinal in many real-world problems, especially in the medical field. That is, there are more than

Feature	Interval (bin)	Scorecard Points
AGE	< 22	100
AGE	[22; 32[	185
AGE	[32; 42[	210
AGE	>= 42	250
HOME	own	225
HOME	rent	110
INCOME	< 1000	120
INCOME	[1000; 35000[	160
INCOME	[35000; 58000[	225
INCOME	>= 58000	260

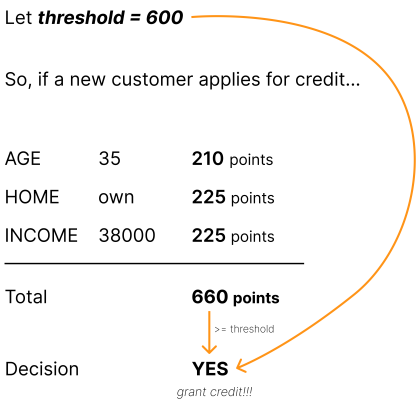


Figure 1.1: **Example of a Scorecard**, from the SAS Institute (2006) [55], that predicts whether an applicant should be granted credit based on age, income, and homeownership status.

two ordered outcomes, but the differences between them are not uniform (e.g., mild, moderate, severe). Aware of this issue, Silva and Cardoso (2015) [56] extended scorecards from binary to ordinal classification using the same traditional two-step approach, but added a pre-processing step: a Data Replication method [9] that transforms the ordinal problem into multiple binary problems, allowing the use of a single binary classifier.

More recently, binary scorecards have been modelled using optimisation-based approaches, formulating the scorecard problem as an exact mathematical optimisation problem instead of using machine learning. The main foundational examples include SLIM [62] and its improved version RiskSLIM [63]. Many other optimisation-based approaches inspired by RiskSLIM have been proposed since [48, 39], including adaptations to multiclass classification [23, 49]. However, no adaptation has been done to ordinal problems, explicitly accounting for the order of the classes. Given these advances, there is a clear gap and interest in developing optimisation-based ordinal scorecards.

Additionally, all existing scorecard methods assume the data is already discretised or include a discretisation step for continuous variables before training. It would therefore be valuable to create a globally optimised model that can produce scorecards in a single step.

1.3 Objectives and Contributions

This dissertation aims to achieve several research objectives and make methodological contributions. These can be summarised as follows.

Objectives:

- Advance the development of interpretable ordinal scorecards using machine learning and mathematical programming approaches.
- Make the scorecard construction algorithm a single-step process.
- Extend existing optimisation techniques, such as RiskSLIM, to ordinal classification.
- Apply the methodology to benchmark and healthcare data, particularly in predicting aesthetic outcomes after breast cancer conservative treatment.

Contributions:

- Design and evaluate multiple combinations of discretisation methods, encoding strategies, and classifiers for binary and ordinal classification tasks.
- Propose and evaluate Infinitesimal Bins, a novel discretisation algorithm approximating scorecard construction to a unified process.
- Adapt RiskSLIM to ordinal classification through the Single Binary Classifier technique.
- Explore ensemble strategies of scorecards and feature selection.
- Validate the methodology using benchmark datasets and healthcare data, focusing on the ordinal problem of predicting aesthetic outcomes after breast cancer conservative treatment.

1.4 Document Structure

This chapter introduces the problem this dissertation addresses, providing the context, motivation, and objectives. The rest of the document is structured as follows: Chapter 2 presents foundational concepts related to the problem at hand, including binary and ordinal classification, Generalised Additive Models, regularisation, interpretability, and optimisation. Chapter 3 reviews the state of the art of scorecards, covering handcrafted scorecards, binary and multiclass classification scorecards, practical applications, and a summary of the current research. Chapter 4 describes the proposed methodology for constructing binary and ordinal scorecards, integrating state-of-the-art techniques and extending them with a proposed discretisation algorithm called Infinitesimal Bins. It also describes the feature encoding methods, models used for learning scores, and the extension of RiskSLIM to ordinal classification. Chapter 5 presents the experimental setup and results. It describes the datasets used and performance metrics, and compares the performance of the different discretisation, encoding, and classification methods. Finally, Chapter 6 summarises the main conclusions of this work and provides suggestions for future research.

Chapter 2

Background

This chapter presents foundational concepts that support this study and the proposed methodology. It begins with **binary and ordinal classification**, the main classification tasks addressed in this work. Following that, it introduces **Generalised Additive Models (GAMs)**, since scorecards are a type of GAM. **Regularisation** is discussed as it is used to induce sparsity and perform feature selection during modelling. The concept of **interpretability** is also presented, as it is an essential aspect of this study. Finally, the basic principles of **optimisation** are introduced, which serve as a basis for techniques explored in the following chapters.

2.1 Binary and Ordinal Classification

Classification is a machine learning problem that involves predicting target labels based on input data. It is a type of supervised learning, meaning that the classification models use labelled datasets to learn from.

The type of classification depends on the nature of the target. Binary classification wants to predict a yes or no outcome. In contrast, multiclass classification addresses more than two target classes. These classes can be either nominal or ordinal. In nominal classification, there is no order among classes (e.g., predicting different types of animals). In ordinal classification, however, the classes have a defined order, even though the differences between consecutive classes may not be uniform (e.g., predicting the quality of a wine).

2.2 Generalised Additive Models

Understanding the relationships between variables is crucial for predictive modelling. Specifically, for a given dependent variable or target Y, it is necessary to learn how the independent variables $X = (X_1, X_2, \ldots, X_p)$ influence it. Generalised Additive Models (GAMs), created by Hastie and Tibshirani (1986) [26], are designed to model these relationships. GAMs are an extension of Generalised Linear Models, which are themselves an extension of Linear Models.

Linear Models model the target as a linear combination of the input variables. The general expression is $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$, where β are the coefficients calculated by the model and ε is an unknown disturbance variable, assumed to follow a normal distribution.

Generalised Linear Models (GLMs), developed by Nelder and Wedderburn (1972) [45], extend Linear Models by allowing for disturbances that follow distributions other than the normal distribution. Besides that, the relationship between the prediction and the expected value of Y is mediated by a link function. As such, GLMs have three parts: a random component that allows Y to be part of an exponential distribution family; a systematic component that defines the predictor η as a linear combination of the input variables, $\eta = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$; and a link function $g(.)$ that aggregates these two components $g(E[Y]) = \eta$.

But GLMs, as Linear Models, assume a linear relationship between Y and X. To model more complex relationships, Hastie and Tibshirani (1986) [26] created Generalised Additive Models. This extension is done by generalising the predictor η as a sum of smooth functions. The expression is $\eta = s_0 + s_1(X_1) + \cdots + s_p(X_p)$, where $s_j(.)$ is a smooth function standardised to satisfy $E[s_j(X_j)] = 0$, to ensure identifiability. A smooth function can capture nonlinearity in the data while maintaining some degree of regularity.

Scorecards are GAMs where the $s_j(.)$ functions are piecewise constant. This means that each $s_j(X_j)$ maps the value of X_j into a discrete set of intervals, maintaining a constant value within each interval [56].

2.3 Regularisation

Regularisation is a technique intended to improve the ability of machine learning models to generalise, which is crucial when predicting outcomes for unseen data. This is done by adding a penalty term to the loss function, which will then be minimised during model training. This prevents overfitting and controls the bias-variance trade-off.

One type of regularisation is **Ridge Regression** [28], also known as **L2 regularisation**. The penalty in the loss function shrinks the coefficients, promoting small weights, and effectively reduces model variance and handles highly correlated predictors. The general objective for linear regression with L2 regularisation is

$$\min \|XW - Y\|^2 + \lambda \|W\|^2, \tag{2.1}$$

where X is the matrix of the input variables, W is the matrix of the coefficients to be calculated by the model, Y is the vector of the real targets, and $\lambda \geq 0$ controls the strength of the regularisation.

The problem with models that use L2 is that typically they have nonzero values for all weights, making the problem harder to interpret. To solve this problem, **Least Absolute Shrinkage and Selection Operator (LASSO)** [58], also called **L1 regularisation**, was created. The penalty added to the loss functions forces multiple coefficients to be zero, doing feature selection and

creating sparse models. The objective is as follows:

$$\min \|XW - Y\|^2 + \lambda \|W\|_1. \tag{2.2}$$

Elastic Net [71] combines both L2 and L1 regularisation methods. It has the sparsity introduced by L1 regularisation, while handling highly correlated features. The general objective is

$$\min \|XW - Y\|^2 + \lambda_1 \|W\|_1 + \lambda_2 \|W\|^2. \tag{2.3}$$

In this equation, λ_1 and λ_2 control the strength of the L1 and L2 regularisations, respectively.

2.4 Interpretability

In the context of machine learning systems, interpretability is defined as "the ability to explain or to present in understandable terms to a human" [14].

Generally, an interpretable model obeys a domain-specific set of constraints that make its internal structure more understandable. These constraints can vary significantly depending on the application domain and user expertise [50].

A typical interpretable supervised learning setup, as presented by Rudin et al. (2021) [51], involves minimising a loss function (to ensure accuracy) while incorporating an interpretability constraint and/or penalty. One common interpretability constraint example is sparsity, which refers to encouraging the model to use only a small number of features or nonzero coefficients. Sparse models are easier to understand, since they reduce the amount of information needed to make and interpret predictions.

There are a few models interpretable by design, as decision trees, sparse linear models, and scoring systems [51]. Notably, in scoring systems, each input feature contributes independently to the final score, allowing practitioners to comprehend how predictions are generated.

As Rudin et al. (2021) [51] discussed, "interpretability is a key element of trust for AI models". This trust becomes especially crucial when these models are used in high-stakes domains, like healthcare, criminal justice, or finance. In contrast, black-box models — "models that by definition are inscrutable" — are more challenging to troubleshoot and use in practice. For these reasons, using inherently interpretable models rather than relying on post hoc explanations of black-box models is more appropriate.

2.5 Optimisation

Optimisation is a discipline within applied mathematics that seeks to find optimal values of variables given a minimisation or maximisation problem, subject to constraints. These problems are modelled as Mathematical Programming Models, and their components are the following [27]:

- **Decision variables**: the unknown variables of the problem. The goal is to determine the values of these variables that will produce the optimal solution to the problem. For example, they could represent quantities of resources to be allocated or the number of units to be produced. They can be represented as $x_1, x_2, ..., x_n$.

- **Objective function**: represents the objective of the problem with respect to the decision variables. The aim is to either maximise or minimise this function. For instance, this could involve maximising the total profit or minimising the total production costs. An example of an objective function is $P = 3x_1 + 2x_2 + ... + 5x_n$.

- **Constraints**: the restrictions or limitations of the values of the variables. A constraint has two components: a function and a constant, which are related by either an equality or an inequality sign. For example, if there is a limit on the quantity of a certain product. A constraint could be expressed as $x_1 + 3x_1x_2 + 2x_2 \leq 10$.

There are several types of Mathematical Models, presented here in increasing order of complexity. The simplest type is Linear Programming, where both the objective function and the constraints are linear functions. If the variables are restricted to be integers, it is denoted as an Integer Programming (IP) problem. This mathematical problem is highly complex to solve as it is combinatorial and NP-hard. Mixed Integer Programming (MIP) problems are linear problems that allow the variables to be either integer or continuous. Nonlinear Programming introduces nonlinear relationships in the objective function and/or constraints. Finally, Mixed Integer Nonlinear Programming (MINLP) combines nonlinearity with integer and continuous variables.

Chapter 3

State of the Art

The chapter provides an overview of the state of the art in scoring systems, organised into five sections. First, the section on **Handcrafted Scoring Systems** details scorecards manually designed by domain experts in healthcare. Next, the **Automatic Binary Scoring Systems** section reviews models to build scorecards automatically from data for two-class problems. Following that, the **Automatic Multiclass Scoring Systems** section extends the discussion to scoring systems designed to handle problems with more than two outcome categories. The **Applications** section presents case studies and practical implementations of scoring systems. The **Comparative Summary of Scoring System Methods** highlights the main characteristics of all presented scoring systems. Finally, the **Conclusion** identifies challenges in the literature and opportunities for future research.

3.1 Handcrafted Scoring Systems

The most well-known scoring system is *CHADS*$_2$ [20] and was handcrafted by a panel of experts to assess the stroke risk. It is based on five binary features, each assigned integer points of 1 or 2, with a maximum total score of 6. Table 3.1 shows the scorecard along with the corresponding predicted risk scores.

Table 3.1: CHADS$_2$ scorecard for stroke risk assessment.

Binary Feature	**Points**
Congestive heart failure history	1
Hypertension	1
Age $\geq$ 75 *years*	1
Diabetes mellitus	1
Stroke or TIA symptoms	2

Score	0	1	2	3	4	5	6
Pr. (%)	1.9	2.8	4.0	5.9	8.5	12.5	18.2

Apgar, created by Apgar (1953) [2], evaluates the condition of a newborn right after childbirth. It evaluates five criteria: heart rate, respiratory effort, muscle tone, reflex irritability, and colour. Each criterion has a score between 0 and 2, and the total score is at most 10.

Table 3.2: Apgar scorecard for newborn evaluation.

Feature	0 points	1 point	2 points
Heart rate	Absent	< 100 ppm	> 100 ppm
Respiratory effort	Absent	Slow and irregular	Vigorous cry
Muscle tone	Absent	Flexed arm and legs	Active
Reflex irritability	Floppy	Minimal response to stimulation	Prompt response to simulation
Colour	Blue or pale	Pink body with blue extremities	Pink

APACHE II [33], which stands for Acute Physiology and Chronic Health Evaluation, calculates the severity score of a disease based on 12 physiological measures taken within the first 24 hours of a patient's Intensive Care Unit (ICU) admission, along with their age and chronic health status. The score ranges from 0 to 71, with a higher score indicating more severe illness and a greater risk of death. All variables are continuous and were discretised by hand, where each interval has its corresponding scores. The original version, *APACHE I* [35], was developed in 1981; however, APACHE II is the most widely used version. A newer version, *APACHE III* [34], was introduced in 1991 and includes 20 physiological measures. Despite this, APACHE II remains the preferred choice because of its simplicity.

SAPS III [40, 41], Simplified Acute Physiology Score, was created based on APACHE and predicts hospital mortality upon ICU admission. It is an extension of SAPS II [21]. It accounts for 17 features, and the scores range from 0 to 217.

The *ABCD rule of dermoscopy* was developed to diagnose melanoma, a type of skin cancer, and was created by Nachbar et al. (1994) [43]. Dermoscopy is a non-invasive skin imaging technique that is used to examine skin lesions with a device called a dermatoscope. This rule was designed to distinguish between benign and malignant melanocytic lesions through an evaluation based on four dermoscopic criteria: Asymmetry (the "A"), Border irregularity (the "B"), Colour variegation (the "C"), and the presence of specific Dermoscopic structures (the "D"). The diagnosis is based on the Total Dermoscopy Score (TDS), calculated using the formula: $\text{TDS} = (A \times 1.3) + (B \times 0.1) + (C \times 0.5) + (D \times 0.5)$. A TDS less than 4.75 suggests benign lesions, scores between 4.8 and 5.45 raise suspicion, and scores above 5.45 strongly suggest melanoma.

The *Sequential Organ Failure Assessment (SOFA)* score [65] was developed to quantify the degree of organ dysfunction in critically ill patients, particularly those with sepsis. It evaluates six organ systems: respiratory, cardiovascular, hepatic, coagulation, renal, and neurological. The score is based on clinical and laboratory measurements, such as the ratio of arterial oxygen partial pressure to fractional inspired oxygen, platelet count, bilirubin level, mean arterial pressure (or use of vasopressors), Glasgow Coma Scale scores and creatinine levels or urine output. Each organ system is assigned a score ranging from 0 to 4 based on predefined clinical thresholds, resulting in a total score ranging from 0 to 24. Higher scores indicate more severe organ dysfunction and are

associated with an increased risk of mortality. The SOFA score was incorporated to define sepsis in the Sepsis-3 consensus definition [57].

While $CHADS_2$, Apgar and ABCD focus on simplicity by using very few features and a small range of scores, APACHE, SAPS, and SOFA have more complex scorecards, making them less straightforward but more comprehensive. All of these scoring systems are still used in the medical field.

3.2 Automatic Binary Scoring Systems

All scoring systems presented in this section focus on binary classification tasks.

It starts with **SLIM** [62] and **RiskSLIM** [63], which are both optimisation-based approaches. RiskSLIM, developed by the same authors as SLIM, is constructed based on SLIM but adapted to provide risk scores instead of scores. Then, **Interval Coded Scoring** [7] combines machine learning with mathematical programming and offers a toolbox with an interface for user interaction. **AutoScore** [69] is focused on using electronic health records and uses machine learning, primarily logistic regression. Finally, **PSORisk** [48] and **FasterRisk** [39] are both adaptations of RiskSLIM. PSORisk makes the algorithm more efficient by using parallel computing, while FasterRisk translates the mathematical programming formulation to heuristics, allowing for more scalability.

3.2.1 SLIM and RiskSLIM

Supersparse Integer Linear Models (SLIM), made by Ustun and Rudin (2016) [62], are scoring systems created using optimisation-based programming, a pioneering approach in this field. It formulates scorecard construction as an integer program with the two main goals of maximising accuracy, by minimising the 0-1 loss, and maximising sparsity, by minimising ℓ_0-seminorm, while constraining the scores to be small coprime integers. The 0-1 loss assigns a loss of 0 if the prediction is correct and a loss of 1 if the prediction is incorrect, so the loss across the whole dataset corresponds to the number of classification errors. The ℓ_0-seminorm counts the number of nonzero coefficients. SLIM allows for the inclusion of operational constraints with no parameter tuning.

Before SLIM, the methods to create integer scoring systems were typically obtained by first calculating real coefficients and then rounding them. This could reduce accuracy and violate operational constraints, so SLIM addresses these issues by directly optimising over integer coefficients.

The IP optimisation problem solved by SLIM is:

$$\begin{aligned} \min_{\lambda} \quad & \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[y_i \lambda^T x_i \leq 0] + C_0\|\lambda\|_0 + \varepsilon\|\lambda\|_1 \\ \text{s.t.} \quad & \lambda \in \mathscr{L}, \end{aligned} \tag{3.1}$$

where:

- N is the number of observations;

- $\lambda \in \mathbb{R}^{P+1}$ are the coefficients of the P features and the intercept λ_0;
- $x \in X \subseteq \mathbb{R}^{P+1}$ is a vector of P features and the intercept term;
- $y \in Y = \{-1, 1\}$ are the binary labels;
- $\|\lambda\|_0 = \sum_{j=1}^{P} \mathbb{1}[\lambda_j \neq 0]$ is the ℓ_0-seminorm;
- $C_0 \in \left[\frac{1}{N}, 1\right]$ controls the trade-off between accuracy and sparsity (user-provided);
- $\|\lambda\|_1$ is an ℓ_1-penalty that restricts coefficients to coprime values;
- ε controls the trade-off between ℓ_1-penalty and accuracy and sparsity. It is small enough to ensure SLIM does not sacrifice accuracy or sparsity to reduce the ℓ_1-penalty (user-provided);
- $\mathscr{L} \in \mathbb{Z}$ is the set of feasible coefficients. Usually $\lambda \in \{-10, \ldots, 10\}^{P+1}$, and may include operational constraints, e.g., $\|\lambda\|_0 \leq 10$ to limit the number of selected features (user-provided).

To address the issue of imbalanced datasets, Ustun and Rudin (2016) [62] suggested using a weighted 0-1 loss function. In the formulation, the standard 0-1 loss is replaced by

$$\frac{W^+}{N} \sum_{i \in \mathscr{I}^+} \mathbb{1}[y_i \lambda^T x_i \leq 0] + \frac{W^-}{N} \sum_{i \in \mathscr{I}^-} \mathbb{1}[y_i \lambda^T x_i \leq 0] \tag{3.2}$$

,

and a new constraint is added:

$$\frac{1}{N^-} \sum_{i \in \mathscr{I}^-} \mathbb{1}[y_i \lambda^T x_i \geq 0] \leq \gamma. \tag{3.3}$$

Here, W^+ and W^- are weights controlling the accuracy trade-off between positive and negative classes. N^- are negative observations that belong to $\mathscr{I}^- = \{i : y_i = -1\}$ and γ is the maximum tolerated error rate on negatively labelled observations.

SLIM includes a data reduction algorithm to speed up computation by discarding training examples that do not affect the final model. It involves first solving a linear programming relaxation of the IP formulation as the surrogate problem on the full dataset. Then, this surrogate problem is solved N times, each time leaving a different data point out. If excluding a given point does not change the optimal solution for the surrogate problem, that point is considered redundant and discarded from the training set. SLIM is then trained using only the identified relevant points.

Ustun and Rudin (2019) [63] later extended SLIM and created *Risk Supersparse Linear Integer Models (RiskSLIM)*. The fundamental difference between the two is that SLIM produces scores, and RiskSLIM produces risk scores. Scores are used for decision-making, where, by comparing the calculated score to a given threshold, a "yes" or "no" decision is made. Risk scores are probabilities used for risk assessment. The higher the risk score, the higher the probability of a given event.

RiskSLIM formulates the *risk score problem* as:

$$\begin{aligned} \min_{\lambda} \quad & l(\lambda) + C_0 \|\lambda\|_0 \\ \text{s.t.} \quad & \lambda \in \mathscr{L}, \end{aligned} \tag{3.4}$$

where:

- $l(\lambda) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + \exp\langle -\lambda, y_i x_i \rangle))$ is the normalised logistic loss;
- $\|\lambda\|_0 = \sum_{j=1}^{P} \mathbb{1}[\lambda_j \neq 0]$ is the l_0-seminorm;
- $\mathscr{L} \in \mathbb{Z}$ is a set of feasible coefficient vectors (user-provided);
- C_0 is a trade-off parameter to balance fit and sparsity (user-provided).

By minimising the logistic loss instead of the 0-1 loss used in SLIM, RiskSLIM not only maximises the accuracy, but also maximises the calibration. In this context, calibration means that an event's predicted risk matches the event's actual frequency in the data. For example, if the model predicts a 30% risk for a group of instances, then 30% of those instances should actually experience the event.

The predicted risk is estimated using the logistic link function:

$$p_i = Pr(y_i = +1 | x_i) = \frac{1}{1 + \exp(\langle -\lambda, y_i x_i \rangle)}. \tag{3.5}$$

RiskSLIM is formulated as a MINLP, which is NP-hard. The problem is mixed-integer because the coefficients are restricted to a discrete set of integer values, and nonlinear because the objective function, logistic loss, is nonlinear. This combination makes the optimisation computationally complex, especially for large datasets.

To address this complexity, Ustun and Rudin (2019) [63] proposed a new Cutting Plane Algorithm (CPA), the Lattice Cutting Plane Algorithm (LCPA). A CPA is an optimisation method that solves a surrogate problem of the original problem and iteratively adds linear constraints ("cuts") to exclude infeasible solutions and converge to the optimal solution. Traditional CPAs stall in non-convex problems, which is an issue in this case since the sparsity constraint makes the problem non-convex. The LCPA solves a MILP version of the original MINLP, and addresses this challenge by alternating between solving linear relaxations of the problem and using a branch-and-bound search.

It is not explicitly mentioned, but both SLIM and RiskSLIM assume the data is binarised before training.

3.2.2 Interval Coded Scoring

Interval Coded Scoring (ICS), created by Billiet et al. (2018) [7], differentiates itself from other binary scoring systems because of its **toolbox** that integrates an **interface** and allows for user

interaction and visualisation of the model. Before modelling, the user can preselect the data in the toolbox.

The modelling begins with the data being discretised and encoded. Binary and categorical variables are **one-hot encoded**. The continuous variables have to be discretised before encoding. Therefore, they are first transformed using **equal frequency**: the range of each variable is divided into a predefined number of intervals, such that each interval contains the same number of observations. The number of intervals for each variable can be determined beforehand, but each interval must contain at least 5 data points; otherwise, the number of intervals is reduced. The continuous variables are then encoded also using one-hot encoding, and all features become binary.

After data transformation, the weights are calculated for all binary variables. The algorithm used is either based on **linear programming** (*lpICS*, derived from **soft-margin SVM**) or on **elastic net** (*enICS*). In both cases, a mathematical programming solver is used. The resulting vector of weights is sparse.

These weights are then tuned through ***iterative reweighting***. The goal is to create a simpler model with fewer changes in the weights between consecutive intervals, to identify general patterns rather than focusing on very specific changes. This tuning is done repeatedly until a stable model is achieved. The user can control the degree of weight smoothing to find the desired balance between model complexity and performance.

Finally, the weights are converted to integer values by scaling and rounding. Then, through logistic regression, the resulting weights are transformed into risk values.

3.2.3 AutoScore

AutoScore, proposed by Xie et al. (2020) [69], is an "automatic clinical score generator", designed with the goal of using electronic health records.

Firstly, **feature ranking** is done using **Random Forest**, which produces an ordered list of the features according to their predictive importance. The number of features selected for the scoring can be determined manually, through grid search, or performance validation assessment, as described in the final step.

Next, the selected features are discretised. Continuous variables are transformed into categorical variables using **equal frequency**, with the number of intervals defined in advance. Domain knowledge can be integrated at this stage by defining or adjusting the cutoff points, either by combining or rounding them.

Afterwards, each category of each variable is assigned a score. The weights/scores are calculated with **logistic regression**. Logistic regression is performed again to ensure all weights are positive with adjusted reference categories. The weights are scaled by dividing each by the smallest weight and rounded to obtain integer scores.

Finally, to determine the optimal number of features to use for scoring, the model is evaluated to identify the smallest set of variables beyond which there is no significant improvement in performance. Once this is determined, the feature discretisation and score calculation processes are repeated using this set of variables, resulting in the final scorecard.

3.2.4 PSORisk

Pham et al. (2020) [48] propose a variant of **RiskSLIM** that uses **Particle Swarm Optimisation (PSO)**, called *PSORisk*. PSO is a population-based meta-heuristic capable of accelerating the optimisation process through **parallel computing**, thus improving the efficiency of RiskSLIM.

In PSO, each particle in the swarm represents a candidate solution defined by its position in the search space. At each iteration, particles adjust their positions based on their previous velocity, the best position it has found so far, and the best position found by the entire swarm. The algorithm guides each particle toward promising areas of the search space. The PSO update rule is the following:

$$\begin{aligned} p_i^{t+1} &= p_i^t + v_i^{t+1} \\ v_i^{t+1} &= w \times v_i^t + c_1 \times r_1 \times (Pbest_i^t - p_i^t) + c_2 \times r_2 \times (Gbest^t - p_i^t). \end{aligned} \tag{3.6}$$

In this formulation, p_i^t is the position of particle i at iteration t, and v_i^t is its velocity; $Pbest_i^t$ and $Gbest^t$ represent, respectively, the best position found by particle i so far (personal best), and the best position found by the entire swarm (global best); w is the inertia weight and controls the influence of the previous velocity; c_1 and c_2 are acceleration constants; and r_1 and r_2 are random numbers between $[0,1]$ that measure how each particle is influenced by the personal and global best positions.

One of the main differences between PSORisk and RiskSLIM is in the loss formulation. While RiskSLIM uses the standard logistic loss (as shown in Equation 3.4), PSORisk adapts it to:

$$l(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i(\alpha_0 + \alpha_1 \langle -\lambda, x_i \rangle))), \tag{3.7}$$

so the distinction is in how the coefficient vector λ contributes to the total score.

To reduce complexity, PSORisk divides the process into two phases: (1) selection of features and determination of coefficients with a PSO-based algorithm, and (2) probability calibration using logistic regression, where α_0 and α_1 are estimated to best fit the scores.

In phase (1), each particle in the PSO algorithm represents a coefficient vector of the selected features. The algorithm initialises particle positions randomly and iteratively evaluates them with a fitness function. It then refines the particles using local search and updates them via the PSO update rule. The coefficients are rounded to the closest integers, considering a feature selected if its rounded coefficient is nonzero. Coefficients are also bound to predefined limits. When the termination condition is met, the best feasible solution is returned and used to construct the total score by linearly combining selected features with their rounded coefficients. Finally, in phase (2), this total score is the input to a logistic regression model to build the classifier.

3.2.5 FasterRisk

FasterRisk was created by Liu et al. (2022) [39] and is inspired by **RiskSLIM** [63]. The authors state that RiskSLIM's optimisation-based approach relies on licensed mathematical programming

tools, which can be slow and struggle with scalability. So, one of the goals is to produce scores "within a few minutes on a small personal computer" by using **heuristics** that approximate the power of those solvers. In addition, it generates a pool of almost optimal solutions rather than a single solution.

The discretisation is not detailed, but it is mentioned that **all variables are binarised before training**.

FasterRisk starts with a formulation similar to RiskSLIM, which minimises logistic loss and has sparsity and integer coefficient constraints, but introduces a multiplier variable m to make the solution space larger. The general formulation of the problem is the following:

$$\begin{aligned} \min_{w,w_0,m} \quad & L(w,w_0,\tfrac{1}{m}D) = \sum_{i=1}^{n} \log\left(1+\exp\left(-y_i \tfrac{x_i^\top w + w_0}{m}\right)\right) \\ \text{s.t.} \quad & \|w\|_0 \le k \\ & w \in \mathbb{Z}^p \\ & w_j \in [-5,5], \quad \forall j \in \{1,\dots,p\} \\ & w_0 \in \mathbb{Z} \\ & m > 0. \end{aligned} \tag{3.8}$$

Here:

- L is the logistic loss function;
- $D = \{1, x_i, y_i\}_{i=1}^n$ is the dataset, where 1 corresponds to the intercept;
- $\frac{1}{m}D = \{\frac{1}{m}, \frac{1}{m}x_i, y_i\}_{i=1}^n$ is the rescaled dataset;
- w are the integer weights to be learned;
- w_0 is the intercept;
- k is the maximum number of nonzero coefficients (sparsity constraint).

To solve this problem, the first step involves working with the original dataset D (not scaled) and without the integer constraint. This results in an optimal sparse continuous solution.

There are multiple models that can achieve a similar performance to the initially found solution. By defining a tolerance gap on the loss, FasterRisk identifies a pool of near-optimal continuous solutions.

Finally, a "start ray" search is performed by exploring and scaling each coefficient vector in the pool by a range of multipliers (m). The scaled coefficients are then rounded to integers using sequential rounding - each coefficient is processed in a predefined order and rounded to the nearest integer that minimises the increase in the loss function. The best multiplier is chosen for each continuous solution, resulting in **multiple near-optimal integer solutions**.

3.3 Automatic Multiclass Scoring Systems

All scoring systems presented in this section focus on multiclass classification tasks, including ordinal targets.

The section starts with **Differential Scorecards for Binary and Ordinal Data** [56], which introduces a novel method called Differential Coding and adapts scorecards to ordinal problems using a Data Replication Method [9]. Next, **Multi-Objective Symbolic Regression** [18] uses Genetic Programming and produces scores as a nonlinear combination of features, differing from the structure of traditional scorecards. Finally, both **FairScoringSystems** [49] and **MISS** [23] extend RiskSLIM to multiclass classification.

3.3.1 Differential Coding for Binary and Ordinal Scorecards

Silva and Cardoso (2015) [56] present a new encoding technique, called Differential Coding, and adapt scorecards for ordinal classification using a data replication method [9]. The study evaluates various combinations of discretisation techniques, encodings, and classifiers.

For discretising continuous variables, the methods used were:

- **Equal frequency**;
- **Equal width**, that splits the range of the variables into intervals of equal size;
- **Class-Attribute Interdependence Maximisation (CAIM)**, which creates intervals that maximise the dependency between the discretised variable and the target;
- **Fuzzy CAIM (FCAIM)**, a variant of CAIM that allows fuzzy boundaries between intervals to handle uncertainty or overlap in data.

For encoding, the following techniques were employed:

- **One-hot encoding**;
- **Weight of Evidence coding**, which encodes categories based on their predictive power for the target, using the log odds of positive outcomes versus negative outcomes;
- The proposed **Differential Coding** captures the relative differences between intervals instead of treating them as independent columns.

To learn the weights, standard classifiers such as **Least Squares**, **Logistic Regression**, and **SVM** were applied. Least Squares was integrated with **elastic net regularisation** to induce sparsity.

These methods were tested on binary datasets and ordinal datasets. The adaptation to ordinal uses a data replication method, which transforms the ordinal problem into multiple binary problems, to then be solved using a Single Binary Classifier.

The methods CAIM, Differential Coding, and the Data Replication Method are used in the proposed methodology and are explained in detail in the following chapter.

3.3.2 Multi-Objective Symbolic Regression

The work by Ferrari et al. (2022) [18] proposes constructing scoring systems using *Multi-Objective Symbolic Regression (MOSR)*. A fundamental distinction from the state-of-the-art scoring systems is that this method learns **scoring models that are not limited to linear combinations of features** - MOSR simultaneously learns the coefficients and the structure of the aggregation formula. Besides that, the variables in the problem are continuous, both the scores and the targets, and enable later classification.

Symbolic regression is a method that uses **Genetic Programming** (GP) to find mathematical expressions that best fit a given dataset. GP is a biologically inspired algorithm where a population of candidate solutions – in this case, formulas represented as trees – evolves over generations through mutation, crossover, and selection [17].

While traditional symbolic regression optimises a single objective, **multi-objective** symbolic regression optimises for multiple objectives simultaneously. In this method, the objectives are: predictive accuracy, represented as a weighted Mean Squared Error; **data balance**, measured with the Wasserstein distance between the predicted and true value distributions; and **data stratification**, calculated with the Kendall-τ correlation between the predicted and real target ordering.

For data preprocessing, the input features are rescaled using **Gaussian standardisation** and "variable dummification". The latter involves first discretising continuous variables using **knowledge-defined thresholds**, followed by **one-hot encoding** of continuous, categorical, and binary features. A similar rescaling process is applied to the target variables. When working with high-dimensional datasets, feature dimensionality reduction is performed using the nonlinear **feature selection** method Hilbert-Schmidt Independence Criterion Lasso (HSIC Lasso), which selects variables based on their statistical dependence with the target. Finally, the empirical distribution of each target variable is calculated using Knuth's method, which determines a discretisation scheme for each target. The relative frequencies for each bin associated with each target are calculated as well.

The GP algorithm used is the NSGA-II algorithm [13], a widely used elitist multiobjective genetic algorithm that preserves population diversity and ranks individuals based on Pareto dominance. In multi-objective optimisation, a solution is said to Pareto dominate another if it is better in at least one objective, and as good in the others.

The final output is a set of Pareto-optimal scoring formulas, from which users can choose according to their application needs.

Building on this work, Guidetti (2024) [24] integrated parametric Bayesian Inference to address both epistemic and aleatoric uncertainty. These stand for, respectively, uncertainty due to limited data and uncertainty from noise in the data. They used uncertainty estimation, together with multi-objective optimisation, to extend the approach to Continual Learning and Federated Learning with non-IID (not independent and identically distributed) data.

3.3.3 FAIRScoringSystems

Rouzot et al. (2023) [49] created the *FAIRScoringSystems*, by adapting **SLIM** [52, 62] from binary to multiclass classification and incorporating fairness constraints. Fairness aims to mitigate bias that may be present in the data or introduced by the model to avoid discrimination [64].

To adapt SLIM to multiclass classification, a ***one-vs-all*** approach is used, which fits one binary classifier per class. This means that one scoring system is created for each class, and its purpose is to identify examples that belong to that specific class, opposed to all other classes. To classify a new example, each scoring system is evaluated, and the class with the scoring system with the highest score is chosen as the prediction. Additionally, each class scoring system has a bias term, which is always added to the respective class score during prediction.

In Table 3.3, an example of a scoring system created by FAIRScoringSystems can be seen, as shown by Rouzot et al. (2023) [49]. It is designed for a dataset that predicts customer categories and has four classes (*A*, *B*, *C* and *D*)[1]. For example, consider a customer that is graduated, has never been married, has work experience of 1 year, belongs to "Cat_6", and has a low spending score. The FAIRScoringSystems would assign the following scores: 0 for class A, 6 for class B, 20 for class C, and 15 for class D. Since class C has the highest score, it would be the predicted class for this customer.

Table 3.3: Example of a multiclass scoring system generated by *FAIRScoringSystems* for the `customer` dataset [49]. Each table corresponds to a scoring system for a single class. The attribute "starts with" is the bias.

Binary Feature	Scores for **Class A**
starts with	9
Graduated	9
Work Experience ≤ 1	-9
Cat == Cat_6	-9

Binary Feature	Scores for **Class B**
starts with	-9
Ever Married	-9
Graduated	9
Work experience ≤ 1	-9

Binary Feature	Scores for **Class C**
starts with	2
Graduated	9
Cat == Cat_6	9

Binary Feature	Scores for **Class D**
starts with	1
Low Spending Score	9
Work Experience ≤ 1	-4
Cat == Cat_6	9

In the problem formalisation, the loss used is the empirical loss (for accuracy) or balanced loss (for balanced accuracy). The constant C_0 **is not user-provided** as in SLIM, but instead it is chosen by the model so that a sparser model is selected only if it does not affect the loss.

3.3.4 MISS

The *Multiclass Interpretable Scoring Systems (MISS)* developed by Grzeszczyk et al. (2024) [23] is, like the FairScoringSystems, an adaptation of **RiskSLIM** to multiclass classification. However, they claim that generating "K models for K classes makes it difficult to use in a feasible manner", which makes them move away from the one-vs-all approach.

[1] https://www.kaggle.com/datasets/vetrirah/customer

Instead of using a one-vs-all strategy, MISS **assigns points to each binary feature for each class**. The total score for a given sample is calculated by summing the points of all the features that are true for that sample. Additionally, a bias term specific to each class is added to each score. The class with the highest score is chosen as the prediction.

An example of a scoring system generated by MISS can be seen in Table 3.4. For this example, the `iris`[2] dataset was used, which has three classes. For instance, a petal length of 5 cm adds 5 points to Class 3 (*virginica*), none to Class 2 (*versicolor*), and subtracts five from Class 1 (*setosa*). And if, for example, Class 3 ends up having the highest score among the three classes, it is chosen for prediction.

Table 3.4: MISS model for the `iris` dataset, as presented by Grzeszczyk et al. (2024) [23]. This dataset has three classes: "Class 1" corresponds to *setosa*, "Class 2" to *versicolor*, and "Class 3" to *virginica*. Each row represents a binary feature, and the values indicate how much that feature contributes to the total score of each class.

Binary Feature	**Class 1**	**Class 2**	**Class 3**
sepal length < 5.4	1	-2	1
$4.8 \leq$ *petal length*	-5	0	5
$0.8 \leq$ *petal width* < 1.75	-5	5	2
+ bias	6	3	1
Score:	$= \ldots$	$= \ldots$	$= \ldots$

The scores generated by MISS can be translated to probabilities for each class by applying the *softmax* function.

In the problem formalisation, the loss used is the cross-entropy loss. Instead of a coefficient vector as in RiskSLIM, MISS **learns a coefficient matrix**, where each entry corresponds to the number of points of a binary feature for a specific class. The first element of the coefficients for each class is the bias for that class.

For data discretisation, Grzeszczyk et al. (2024) [23] use **Feature Discretisation RiskSLIM** (FD-RiskSLIM), as described by Pajor et al. (2022) [47]. This method integrates feature binarisation in the hyperparameter optimisation process, allowing for testing and selecting the binarisation technique best suited to the problem at hand.

3.4 Applications

This section presents studies that have developed scoring systems for specific datasets across various domains, including both optimisation-based and machine learning-based scoring systems.

3.4.1 Optimisation-based Scoring Systems

Adult ADHD [61]. RiskSLIM was used to predict adult Attention Deficit / Hyperactivity Disorder (ADHD) using responses to questions on the World Health Organisation Adult ADHD Self-Report Scale (ASRS). The objective was to develop an optimal integer scoring system, with final

[2] https://archive.ics.uci.edu/dataset/53/iris

scores ranging from 0 to 5, while aligning the scale with the diagnostic criteria of DSM-5. Each ASRS question has four possible ordered responses: "never", "rarely or more", "sometimes or more", and "frequently or more". So, for each question, four nested binary indicators were created. For example, if the response is "sometimes or more", both the "rarely or more" and "sometimes or more" indicators are set to 1. The model was restricted to include a maximum of 8 questions, with as many nested binary indicators per question as useful. The coefficients corresponding to the "never" indicators were restricted to be negative, while those of the other responses were set to be positive.

Obstructive Sleep Apnea (OSA) [53, 62]. SLIM was applied to develop a scoring system for screening obstructive sleep apnea (OSA). The goal was to use only routinely available standard medical information, without patient-reported symptoms due to their unreliability, which is used by other OSA scoring systems. The data used is not publicly available and comes from the Massachusetts General Hospital. Domain experts required operational constraints, including restrictions on model size and the sign of coefficients. The major conclusion was that the ability to incorporate these operational constraints into the mathematical program was a significant advantage in comparison to the machine learning approaches.

Seizure risk prediction in the ICU [53]. RiskSLIM was also used for seizure prediction in critically ill patients, based on continuous electroencephalography (cEEG) recordings, which monitor brain activity, from Massachusetts General Hospital. The goal was to predict whether a patient would experience a seizure within 24 hours using features extracted from 6 hours of cEEG recordings. The data is not publicly available. The motivation of this study was to replace clinicians' subjective manual estimates with an automated tool. The primary objective was to avoid false positives - patients predicted as no risk when they actually had risk. To achieve this, constraints were added to the problem. The results show that SLIM had fewer false positives compared to other machine learning methods, so, once again, it benefited from the ability to include constraints.

Recidivism prediction [53]. RiskSLIM was applied to recidivism prediction. Prior work suggested that traditional statistical models could match the performance of machine learning when combined with preprocessing. However, these traditional models struggle with calibration, because the coefficients were rounded to integers after modelling. RiskSLIM addressed these limitations by integrating constraints for integer coefficients, resulting in improved calibration. The study used the largest publicly available dataset on recidivism: the Recidivism of Prisoners Released in 1994 dataset, collected by the U.S. Department of Justice, Bureau of Justice Statistics [60].

Mortality in patients with cardiogenic shock [70]. A risk score named BOS,MA_2 was developed with RiskSLIM to predict in-hospital mortality among adults admitted to the cardiac ICU with cardiogenic shock. To handle missing values, predictive mean matching was used, a method

that fills in missing data with values from similar patients in the dataset. For the discretisation of the continuous variables, different cutoff points were tested, and the ones with the highest performance were chosen; however, the authors did not elaborate further.

Prediction of Illicit Massage Businesses (IMBs) [59]. Using RiskSLIM, this study predicts whether a massage establishment is an Illicit Massage Business (IMB), which are businesses that force victim workers into sex and labour trafficking. To build the score, the authors created their dataset and collected geographic and socio-demographic features, Yelp customer reviews and business information, massage license records, and data from Rubmaps, a review board where sex buyers report on IMBs. All continuous features were discretised into three bins: “low”, “medium”, and “high”. Most features were divided based on quantiles, and others used natural cutoffs. For example, distances were categorised as "short" if they were less than 20 miles, "medium" if between 20 and 60 miles, and "long" if greater than 60 miles. The “medium” category was used as the reference category for all features and excluded from the model.

3.4.2 Machine Learning-based Scoring Systems

Stroke risk [46]. Optimal Classification Trees (OCT) were used to create a nonlinear risk score for the 10-year risk of stroke, improving upon the Revised Framingham Stroke Risk Score (R-FSRS), and creating the Nonlinear Framingham Stroke Risk Score (N-FSRS). The R-FSRS is based on the original Framingham Stroke Risk Score (FSRS) [68], which was constructed using Cox proportional hazards regression models with coefficients rounded to integers. The revised version, R-FSRS [15], used the same statistical methodology but on updated cohorts, recalibrating the coefficients and incorporating new, contemporary risk factors. Unlike classical approaches, OCT adaptively adjust splits on independent variables, allowing for the capture of nonlinear interactions among the risk factors. The results show that the OCT-based model outperformed the R-FSRS. Additionally, the model was implemented with a user-friendly interface.

Dehydration severity in patients over 5 years [4]. Ordinal classification was used to predict dehydration severity in patients aged more than 5 years with diarrheal disease. No scoring system for this application existed before. The result of this study is the Novel, Innovative Research for Understanding Dehydration in Adults and Kids (NIRUDAK) score, which categorises the dehydration severity into three classes: none, some, or severe. The ordinal regression model used forward stepwise regression, a method that starts with no variables in the model and sequentially adds the most statistically significant features one at a time, stopping when no additional features improve model performance. The variables used in the model were either binary or discretised by hand. The resulting coefficients were divided by a common divisor and rounded to the nearest integer to create a 13-point scoring system. Scores of 0–3 indicate no dehydration, 4–6 indicate some dehydration, and 7–13 indicate severe dehydration.

Predicting ICU admission in COVID-19 patients [19]. A risk score was developed with ordinal regression for patients with COVID-19 admitted to the general care unit using smartwatches to continuously monitor physiological measurements, specifically heart rate (HR), heart rate variability (HRV), and respiratory frequency (RF). The score assesses two outcomes: the probability of hospital discharge and the probability of intensive care admission. To construct the model, explanatory models were first used to identify the features significantly associated with patient outcomes. Because many of these features were highly correlated, Principal Component Analysis was applied to reduce dimensionality. Logistic regression with LASSO regularisation was then employed to select the features with the highest predictive power. Finally, an ordinal regression model with the selected features generated the score. A higher risk score indicates a greater probability of ICU admission and a lower probability of hospital discharge.

Prediction of red blood cell transfusion after orthopaedic surgery [10]. Logistic regression was used to construct a scoring system to predict red blood cell (RBC) transfusion after orthopaedic surgery. The variables were selected with Recursive Feature Elimination (RFE) - predictors contributing less than 5% of the total feature importance were excluded. Continuous variables were discretised based on clinical reference ranges or conclusions taken from partial dependence plots, which show how the predicted outcome changes as one variable varies and the other variables remain constant. The resulting coefficients from the logistic regression model were scaled by multiplying by 10 and rounded to the nearest integer.

Predict sepsis [30]. This study uses a knowledge-transfer-based approach to predict sepsis using electronic health records. This approach integrates established medical scoring systems, including NEWS, SIRS, and qSOFA, with an ordinal logistic regression model. Specifically, the output scores from these existing scoring systems are used as additional features in the ordinal regression model. The ordinal classes in this study are defined as: no sepsis, mild sepsis (patients who did not die during hospitalisation), and severe sepsis. The main goals of this approach are to address class imbalance and limited sample sizes, resulting in improved prediction performance compared to the traditional scoring systems, particularly for the minority classes.

3.5 Comparative Summary of Scoring System Methods

Table 3.5 summarises the various approaches discussed in this chapter to provide a clearer comparison of all scoring systems. The table outlines the types of targets supported, the methodological approaches used, whether the methods are optimisation-based, the application domains, and whether ordinal classification is specifically addressed.

Table 3.5: Comparative summary of scoring system methods.

Method	Discretisation	Encoding	Target Type	Approach	Optimisation-based	Applications
CHADS2, Apgar, APACHE, SAPS, ABCD	Experts	Experts	Binary / Risk Scores	Handcrafted by experts	No	Stroke, newborns, ICU, melanoma
SLIM [62]	Data is assumed to be discretised	Data is assumed to be encoded	Binary / Risk Scores	Integer Programming ($\ell_0 + \ell_1$ regularisation)	Yes	Benchmark and sleep apnea
RiskSLIM [63]	Data is assumed to be discretised	Data is assumed to be encoded	Binary / Risk Scores	MINLP with logistic loss + LCPA	Yes	Healthcare, recividism and benchmark
PSORisk [48]	Data is assumed to be discretised	Data is assumed to be encoded	Binary / Risk Score	Particle Swarm Optimisation (metaheuristic) + RiskSLIM	Yes	Benchmark
FasterRisk [39]	Data is assumed to be discretised	Data is assumed to be encoded	Binary / Risk Scores	Heuristic approximation of RiskSLIM	No (approximate)	Benchmark
ICS [7]	Equal frequency	One Hot Encoding	Binary	ML + Linear Programming / Elastic Net	Partial	Healthcare
AutoScore [69]	Equal frequency (experts adjust thresholds)	One Hot Encoding	Binary	Logistic regression + feature ranking	No	Healthcare
Differential Scorecards [56]	Equal frequency, Equal width, CAIM, FCAIM	One Hot Encoding, Weight of Evidence, Differential Coding	Binary + Ordinal	Differential Coding + Data Replication	No	Healthcare and benchmark
MOSR [18]	Knowledge-defined thresholds	One Hot Encoding	Continuous / Multiclass	Multi-objective Symbolic Regression (Genetic Programming)	No	Healthcare
FAIRScoringSystems [49]	Data is assumed to be discretised	Data is assumed to be encoded	Multiclass	One-vs-All SLIM + fairness constraints	Yes	Benchmark
MISS [23]	FD-RiskSLIM	One Hot Encoding	Multiclass	RiskSLIM extension with points given per class	Yes	Benchmark

3.6 Conclusion

It can be observed that all scoring systems have a common two-step pipeline: first, they discretise the input data, and then assign weights to the resulting features. Although the first step is not explicitly mentioned in all scoring systems, it is always expected that the input data for the model training is binarised. This reveals a gap in the literature regarding a method that optimises the process in a single step.

Another limitation is that there is no appropriate optimisation-based approach to handle ordinal classification. Even though extensions of RiskSLIM have been proposed for multiclass classification, such as in FairScoringSystems and MISS, they assume that the classes are unordered, not capturing the ordinal structure well.

Chapter 4

Proposed Methodology

This chapter presents the methodology developed for constructing binary and ordinal scorecards, integrating several state-of-the-art techniques and extending them to ordinal classification, along with a new discretisation algorithm called **Infinitesimal Bins**. The focus is on problems with continuous attributes, where the common practice in the literature is to pre-process the data using a predefined discretisation algorithm before scorecard construction.

In particular, to discretise the continuous features, the Class-Attribute Interdependence Maximisation (**CAIM**) algorithm [56] is used as a baseline to compare with the proposed Infinitesimal Bins.

To encode the discretised features, two known strategies are evaluated: **one-hot encoding** [56, 7, 69] and **differential coding** [56].

The models used to learn the weights include standard models, sparsity-inducing models, and an optimisation-based model. For the baseline, **Logistic Regression** and **SVMs** with regularisation techniques are employed. To promote sparsity, sparse linear models, such as **Adaptive Lasso** [3] and **skglm** [6], are used. In addition, **RiskSLIM** [63] is extended to ordinal classification.

In this chapter, the **pipeline for scorecard construction** and all the techniques involved are described in detail.

4.1 Overall Construction of a Scorecard for Continuous Features

The general pipeline for scorecard construction in this work is described below, and the specific algorithms used in each step are detailed in the following sections.

First, the discretisation scheme is found. Each continuous variable is divided into intervals based on selected cut-points. For categorical variables, a new binary feature is created for each unique value. The variables are then encoded according to the discretisation scheme.

If the target variable is ordinal, the data is transformed into a binary classification problem using Single Binary Classifier (SBC) reduction. A binary classifier is then trained to learn the weights (or scores) of each encoded feature. After training, if the original target is ordinal, SBC decoding maps the binary predictions back to the ordinal class labels.

4.2 Single Binary Classifier

For problems where the target variable is ordinal, a data replication method, as proposed by Cardoso and Costa (2007) [9], and utilised for ordinal scorecard development by Silva and Cardoso (2015) [56], is used. This approach is a reduction technique that allows ordinal problems to be addressed by using a Single Binary Classifier (SBC).

The main idea is transforming the original K-class ordinal problem, with classes labelled from C_1 to C_K, into $K-1$ binary classification problems to be then solved by an SBC. It is achieved by finding $K-1$ parallel hyperplanes in the feature space, where each hyperplane separates the data into two groups: those belonging to classes less than or equal to a certain threshold, and those belonging to classes above it. So, the i-th hyperplane discriminates classes $C_1,\ldots,C_i$ against classes $C_{i+1},\ldots,C_K$.

For this purpose, the data is projected into an augmented feature space. If the original feature space is $\mathbb{R}^d$, the augmented space becomes $\mathbb{R}^{d+K-2}$; hence, $K-2$ new features are created. For each hyperplane, a replica of each data point is created. Given the sequence e_0 consists of $K-2$ zeros, and e_q is a sequence of zeros with h, a positive constant, in the q-th position. For the new binary problem with target classes $\overline{C_1}$ and $\overline{C_2}$, the feature space is replicated according the following rule:

$$\begin{bmatrix} \mathbf{x_i^{(k)}} \\ e_0 \end{bmatrix} \in \begin{cases} \overline{C_1} \text{ for } k=1 \\ \overline{C_2} \text{ for } k=2,\ldots,K \end{cases}$$
$$\vdots$$
$$\begin{bmatrix} \mathbf{x_i^{(k)}} \\ e_{q-1} \end{bmatrix} \in \begin{cases} \overline{C_1} \text{ for } k=1,\ldots,q \\ \overline{C_2} \text{ for } k=q+1,\ldots,K \end{cases} \tag{4.1}$$
$$\vdots$$
$$\begin{bmatrix} \mathbf{x_i^{(k)}} \\ e_{K-2} \end{bmatrix} \in \begin{cases} \overline{C_1} \text{ for } k=1,\ldots,K-1 \\ \overline{C_2} \text{ for } k=K \end{cases} .$$

In this step, $K-1$ binary problems are defined as a single binary problem in the augmented feature space. A two-class classifier can now be applied to the binary dataset.

For example, given that there are three target classes ($K=3$), the goal is to find two hyperplanes: one discriminating class C_1 against classes $\{C_2,C_3\}$ and the other discriminating classes $\{C_1,C_2\}$ against class C_3. As shown in Figure 4.1(a) to Figure 4.1(b), if the original feature space is $\mathbb{R}^2$, it gets projected into $\mathbb{R}^3$ by replicating each point as follows:

$$\mathbf{x} \in \mathbb{R}^2 \begin{cases} [\mathbf{x}\ 0] \in \mathbb{R}^3 \\ [\mathbf{x}\ h] \in \mathbb{R}^3 \end{cases} \text{, where } h = \text{const} \in \mathbb{R}^+. \tag{4.2}$$

And the new binary problem becomes the following, as it can be seen in Figure 4.1(c):

$$\begin{bmatrix} \mathbf{x}_i^1 \\ 0 \end{bmatrix} \in \overline{C_1}, \begin{bmatrix} \mathbf{x}_i^2 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{x}_i^3 \\ 0 \end{bmatrix} \in \overline{C_2} \quad ; \quad \begin{bmatrix} \mathbf{x}_i^1 \\ h \end{bmatrix}, \begin{bmatrix} \mathbf{x}_i^2 \\ h \end{bmatrix} \in \overline{C_1}, \begin{bmatrix} \mathbf{x}_i^3 \\ h \end{bmatrix} \in \overline{C_2}. \tag{4.3}$$

.

Given the classification of the binary classifier, the prediction has to be transformed back to the original ordinal problem. For all replicas of a given point there are K possible sequences, and the corresponding predicted classes are:

$$\begin{array}{llllll} \overline{C_1}, & \overline{C_1}, & \ldots, & \overline{C_1}, & \overline{C_1} & \Longrightarrow & C_1, \\ C_2, & \overline{C_1}, & \ldots, & \overline{C_1}, & \overline{C_1} & \Longrightarrow & C_2, \\ & & \vdots & & & & \\ \overline{C_2}, & \overline{C_2}, & \ldots, & \overline{C_2}, & \overline{C_1} & \Longrightarrow & C_{K-1}, \\ \overline{C_2}, & \overline{C_2}, & \ldots, & \overline{C_2}, & \overline{C_2} & \Longrightarrow & C_K. \end{array} \tag{4.4}$$

Thus, the target class for a data point is obtained by adding one to the number of $\overline{C_2}$ labels among the replicas.

Returning to the example with the original feature space $\mathbb{R}^2$ and $K = 3$, the binary classification is illustrated in Figure 4.1(d). To transform the binary targets into the original ordinal targets, the following is done:

$$\overline{C_1}\,\overline{C_1} \Longrightarrow C_1, \quad \overline{C_2}\,\overline{C_1} \Longrightarrow C_2, \quad \overline{C_2}\,\overline{C_2} \Longrightarrow C_3. \tag{4.5}$$

And the solution to the problem with the original data is in Figure 4.1(e).

4.3 Discretisation Techniques

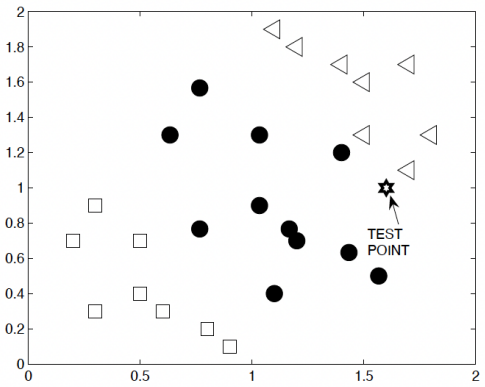
For the discretisation of the training set, each continuous variable is partitioned into discrete intervals using thresholds, also called "cut-points". A cut-point is a value within the range of a continuous variable that “divides the range into two intervals, one interval is less than or equal to the cut-point and the other interval is greater than the cut-point” [36].

Considering a continuous variable $X_j = \{x_{ij}\}_{i=1}^n$, which corresponds to the j-th variable in the dataset D. The set of cut-points is denoted as $T_j = \{t_{j1}, ..., t_{jm}\}$. These cut-points split X_j into $m+1$ intervals as $]-\infty, t_{j1}]$, $]t_{j1}, t_{j2}]$, ..., $]t_{jm}, +\infty[$. Together, the intervals cover the entire range of the variable and do not overlap. Each possible value of the variable X_j belongs to exactly one interval.

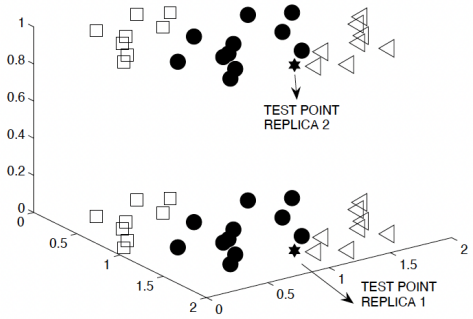
This work uses two discretisation techniques: Class-Attribute Interdependence Maximisation (CAIM) and the proposed Infinitesimal Bins.

4.3.1 CAIM

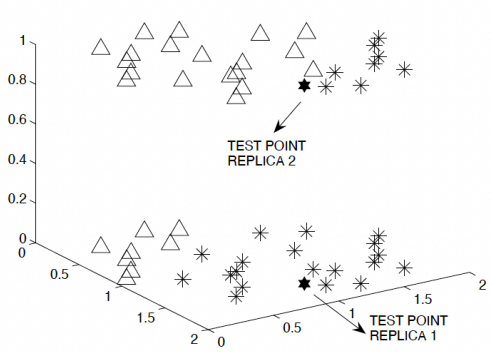
Class-Attribute Interdependence Maximisation (CAIM), proposed by Kurgan and Cios (2004) [37], is a supervised discretisation method whose goal is to maximise the dependence between



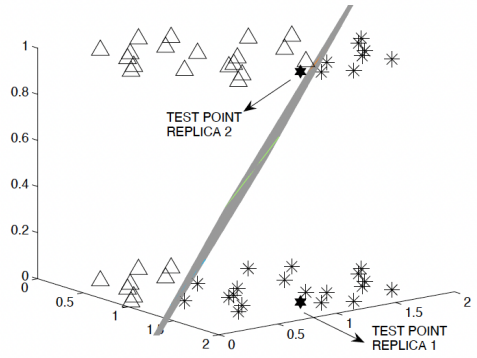
(a) Original data set in $\mathbb{R}^2$ with $K = 3$.



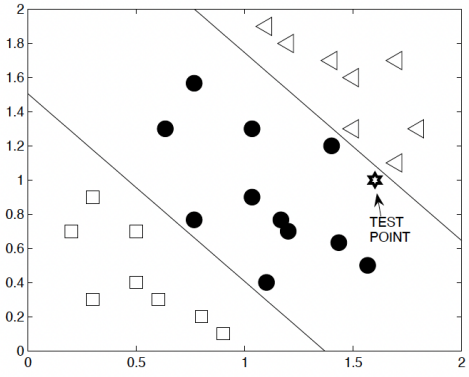
(b) Projection of the dataset in $\mathbb{R}^3$ (with $h = 1$).



(c) Transformation into a binary classification problem.



(d) Linear solution to the binary problem.



(e) Linear solution to the original dataset.

Figure 4.1: Data Replication Method example [9].

the features and the target label, while also minimising the number of intervals.

Let X be the training set consisting of N instances, each belonging to one of K classes. For a continuous feature F in X and a class variable C, let the set of cut-points $T = t_0, t_1, ..., t_m$ discretise F into m intervals.

A quanta matrix (or 2D frequency matrix) can be constructed relating the target variable C to the discretisation of the variable F. The structure of this matrix is shown in Table 4.1. The element q_{ir} represents the number of instances of F that belong to the class C_i and interval $]t_{r-1}, t_r]$. The sum N_{+r} is the total number of instances belonging to the interval $]t_{r-1}, t_r]$, and N_{i+} is the total number of instances belonging to class C_i.

Class	Interval					Class Total
	$[t_0, t_1]$	$\cdots$	$]t_{r-1}, t_r]$	$\cdots$	$]t_{m-1}, t_m]$	
C_1	q_{11}	$\cdots$	q_{1r}	$\cdots$	q_{1m}	N_{1+}
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
C_i	q_{i1}	$\cdots$	q_{ir}	$\cdots$	q_{im}	N_{i+}
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
C_K	q_{K1}	$\cdots$	q_{Kr}	$\cdots$	q_{Km}	N_{K+}
Interval Total	N_{+1}	$\cdots$	N_{+r}	$\cdots$	N_{+n}	N

Table 4.1: General structure of the quanta matrix [56].

The CAIM criterion measures the dependence between the target C and the discretisation T of F. It is defined as:

$$CAIM(C, T|F) = \frac{1}{m} \sum_{r=1}^{m} \frac{max_r^2}{N_{+r}}, \tag{4.6}$$

where m is the total number of intervals, and max_r is the maximum value of the q_{ir} in the r^{th} column. The term $\frac{max_r^2}{N_{+r}}$ favours discretisation schemes in which most values within an interval are concentrated in a single class, thus increasing class-attribute interdependence. The division by m favours discretisation schemes with a smaller number of intervals.

The optimal discretisation scheme for F is the one with the highest value of the CAIM criterion, as a higher CAIM value indicates a higher correlation between T and C.

For this technique, a Python library created by Kurgan and Cios (2004) [38] was used[1], with some adaptations to handle categorical variables.

4.3.2 Infinitesimal Bins

The traditional two-step scorecard design process can limit performance, much like applying conventional machine learning to manually engineered features.

The proposed discretisation technique, called *Infinitesimal Bins*, aims to integrate discretisation and weight learning more tightly, enabling an end-to-end design akin to deep learning-based representation learning.

[1] https://github.com/airysen/caimcaim/blob/master/caimcaim/caimcaim.py

It is motivated by the concept of margin, more specifically, large margin learning, which arose with the development of support vector machines (SVMs). In SVMs, the margin of a classifier refers to the distance of training points to the decision boundary. Large margin learning is based on the principle that classifiers with a larger distance between the data points and the boundary (larger boundary) generalise better on unseen data.

Considering the binary scenario where the label $y \in \{-1, 1\}$ is assigned by a classifier to its input $x \in R^d$. Given the scoring function $f : R^d \mapsto R$ that maps a d-dimensional input x into a one-dimensional space, the binary classification corresponds to $g(\mathbf{x}) = sign(f(\mathbf{x}))$. Since the boundary is where $f(x) = 0$, let S be the set of points in the decision boundary, $S = \{\mathbf{z} : f(\mathbf{z}) = 0\}$.

The instance-specific margin is the distance of that instance to the closest point in S: $d(\mathbf{x}, S) := \inf\{|\mathbf{x} - \mathbf{z}| : \mathbf{z} \in S\}$. The margin over the whole training set, T, is simply given by $m_f := \min_{\mathbf{x} \in T} d(\mathbf{x}, S)$.

Intuitively, maximising the margin during training works as a regulariser, improving robustness and generalisation. By moving the training observations away from the decision boundary, the likelihood that unseen observations will fall on the "wrong side of the boundary" and, therefore, be misclassified, is smaller.

Since $f(\mathbf{x})$ is typically a continuous function, S corresponds to the set of points where the classifier $g(\mathbf{x})$ is discontinuous.

Scorecards do not use smooth functions. Instead, they use piecewise constant functions, where the score changes discretely at bin endpoints. Each bin endpoint, therefore, represents a potential point of discontinuity of the scoring function, even if the decision does not change. Rather than considering the boundary where the decision flips, it can be thought of as the location where the score changes. To generalise the notion of the instance-specific margin, let G be the set of all points of discontinuity of the scoring function $f(\mathbf{x})$ or of the decision function $g(\mathbf{x})$. The instance-specific margin is redefined as the distance between the instance and the set G.

In the discretisation step, the goal is to avoid constraining the classifier learning process through suboptimal binning. For each feature x, the process begins with an infinitesimal binning, where each bin has an infinitesimal width dx.

The binning can safely relax since empty bins do not contribute to the classifier's fitting capacity. This leads from an infinitesimal binning to a finite one, where each bin corresponds to a distinct value of the feature.

To maximise the generalised instance-specific margin, the bin endpoints are placed between consecutive distinct values $x_{[i]}$ and $x_{[i+1]}$ at their average, i.e., $(x_{[i]} + x_{[i+1]})/2$, where $x_{[i]}$ denotes the i-th sorted value of x.

This method places significant emphasis on the weight learning phase instead of relying a lot on the discretisation method, which is carried out on an encoded dataset with a very large number of attributes. As a result, scorecard construction is approximated to a one-step process.

4.4 Encoding Techniques

After each feature is discretised, the dataset is transformed into an encoded representation, where each feature is expanded into multiple binary columns, one for each interval. If feature j is discretised into q_j intervals, the encoded dataset will have a total of $\sum_{j=1}^{p} q_j$ columns, while maintaining the number of rows n.

This work uses two encoding algorithms: one-hot encoding and differential coding.

4.4.1 One-Hot Encoding

In one-hot encoding, each interval of a feature is represented by a binary indicator, where the indicator is set to 1 if the feature value lies in that interval and 0 otherwise. Let $P_{ji} \in \{0,1\}$ be the binary indicator of the j^{th} feature and i^{th} interval, and let W_{ji} be the learned weight for that feature and interval.

The score for a new instance is computed as:

$$score = s_0 + \sum_{j=1}^{p} s_j(X_j), \tag{4.7}$$

where s_0 is the intercept (the baseline score), and $s_j(X_j)$ is the score contribution of feature j. This score is given by:

$$s_j(X_j) = \sum_{i=1}^{q_j} W_{ji} P_{ji}. \tag{4.8}$$

Substituting this into the score equation, and separating the weight of the first interval (W_{j1}) for each feature:

$$score = s_0 + \sum_{j=1}^{p} W_{j1} + \sum_{j=1}^{p} \sum_{i=2}^{q_j} (W_{ji} - W_{j1}) P_{ji}. \tag{4.9}$$

And by absorbing all W_{j1} terms into the intercept, the score can be re-formulated as:

$$score = W_0 + \sum_{j=1}^{p} \sum_{i=2}^{q_j} W_{ji} P_{ji}. \tag{4.10}$$

4.4.2 Differential Coding

Differential coding, introduced by Silva and Cardoso (2015) [56], also expands features into binary variables, but the meaning of the binary indicator is different: a 1 indicates that the feature value lies in a given interval or any higher interval.

This means that the binary indicators, and consequently the weights, are cumulative along the ordered intervals of a feature:

$$W_{ji} = W_{j(i-1)} + \delta_{ji} \implies W_{ji} = \sum_{l=1}^{i} \delta_{jl}, \tag{4.11}$$

where δ_{ji} is, for feature j, the difference in weight between interval $i-1$ and interval i.

With this representation, the score becomes:

$$score = \delta_0 + \sum_{j=1}^{p} \sum_{i=2}^{q_j} \delta_{ji} P^{\star}_{ji}, \tag{4.12}$$

where $P^{\star}_{ji} = 1$ if x_j belongs to an interval $\ell \geq i \geq 2$, and 0 otherwise.

While using regularisation with one-hot encoding "will enforce small values for the weights", in differential coding, regularisation "will promote smooth variations in the score between consecutive bins – a much more intuitive and desirable setting" [56].

4.5 Classifier

Linear classifiers are applied to estimate the weights of each feature of the encoded data. A key requirement for the classifiers is *sparsity* - the model has to assign zero weights to as many features as possible, resulting in a scorecard where only a few features contribute to the final score. This is important because it makes them more interpretable and easier to use. This need for sparsity is even more crucial when using the Infinitesimal Bins discretisation technique, where the number of generated features can be extremely large.

To achieve this, linear classifiers with regularisation (Logistic Regression and SVMs) and specialised supersparse algorithms (Adaptive Lasso and Beyond-L1) and RiskSLIM are employed.

4.5.1 Logistic Regression

Logistic Regression is a generalised linear model (GLM) where the relationship between the features and the probability of the event is modelled using the logit link function.

The logit function transforms the predicted probability P into a linear combination of features:

$$\text{logit}(P) = \log\left(\frac{P}{1-P}\right) = Z = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n, \tag{4.13}$$

where $\beta_0, \beta_2 \ldots, \beta_n$ are the models parameters to be predicted and $X_1, \ldots, X_n$ are the input features.

Maximum Likelihood Estimation (MLE) is used to estimate these parameters. The goal is to determine the parameters for which the observed data has the highest joint probability.

4.5.2 Support Vector Machines

Support Vector Machines (SVMs) are binary classifiers that aim to find the hyperplane that maximises the margin between classes. The margin is the smallest distance between observations of the same class and the hyperplane. Margin maximisation is to choose the hyperplane that gives the largest margin. SVMs employ a soft-margin approach, which allows some misclassification, to balance the trade-off between maximising the margin and minimising classification errors [11].

4.5.3 Adaptive Lasso

Adaptive Lasso was created by Barroso-Luque and Xie (2023) [3] and is an extension of the standard Lasso. While the standard Lasso applies the same penalty to all coefficients, the Adaptive Lasso assigns a unique penalty to each coefficient, thereby improving feature selection.

The two main steps are as follows: First, fit an initial model to obtain estimates of the coefficients. Secondly, use these estimates to define a set of adaptive weights: variables with larger initial coefficients receive smaller penalties, making it less likely that they will become zero in the final model, and variables with very small initial coefficients are penalised more heavily, increasing the chances of being excluded from the model.

Mathematically, the following optimisation problem is solved:

$$\hat{\beta} = \arg\min_{\beta} \left\{ R(\beta) + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right\}, \tag{4.14}$$

where $R(\beta)$ is the loss function, λ is the regularisation parameter, and w_j is the adaptive weight for coefficient β_j.

4.5.4 Generic Generalised Linear Estimator (skglm)

The `skglm` library [42] provides modular estimators for sparse Generalised Linear Models (GLMs).

GLMs assume that the predicted output is a linear combination of features, and they are often combined with a regularisation term to enforce certain properties on the coefficients, such as sparsity or group structure. From an optimisation view, training a GLM involves solving a composite objective that combines data fit with a penalty. The main principle of `skglm` is to treat solvers, data fits, and penalties as separate and interchangeable components. This allows for different combinations of these elements.

`skglm` uses a working set approach to progressively identify a set of nonzero coefficients and solve increasingly smaller subproblems. This reduces the number of variables at each stage. In this work, the solver used is `AndersonCD`, which is a coordinate descent method with Anderson acceleration. Coordinate descent updates one parameter at a time while keeping others fixed, searching along coordinate directions to find the minimum of a function. Anderson acceleration speeds up convergence by using information from the last K points rather than using just the previous point. The update for the next iteration is the following:

$$x^{(k+1)} = \sum_{i=1}^{K} c_i x^{k-K+i}, \tag{4.15}$$

where c_i are the coefficients chosen to minimise the residuals between the current iterate and the fixed-point update.

4.5.5 RiskSLIM

The RiskSLIM [63] was adapted for this work. The optimisation problem was formulated as:

$$\begin{aligned} \min_{\lambda} \quad & \frac{1}{N}\sum_{i=1}^{N} \log\left(1+\exp\left(-y_i\langle\lambda, x_i\rangle\right)\right) + C_0\|\lambda\|_0 \\ \text{s.t.} \quad & \lambda \in \mathscr{L}, \end{aligned} \tag{4.16}$$

where:

- $\|\lambda\|_0 = \sum_{j=1}^{P} \mathbb{1}[\lambda_j \neq 0]$ is the l_0-seminorm;
- $\mathscr{L} \subseteq \mathbb{R}$ is a set of feasible coefficient vectors (user-provided);
- C_0 is a trade-off parameter to balance fit and sparsity (user-provided).

To ensure direct comparability with the other classifiers, the restrictions in $\mathscr{L}$ on the maximum absolute value of coefficients and on the maximum number of nonzero coefficients were removed. The maximum number of coefficients was set to its largest possible value, ensuring that CPLEX [2] could run without requiring a license. C_0 remained the same defined by the authors, 1^{-6}.

After learning the weights, the score is calculated for each instance, using the intercept, λ_0, and the coefficients calculated by RiskSLIM, λ:

$$\text{score}_i = \lambda_0 + \lambda \cdot x_i, \tag{4.17}$$

This score is then transformed to risk probability using the logistic link function:

$$p_i = \frac{1}{1+\exp(-\text{score}_i)}. \tag{4.18}$$

To obtain a binary classification, the probability is compared to a threshold. The RiskSLIM authors recommend that domain experts define the threshold. For the experiments, it was considered both the standard value of 0.5 and the threshold (within the range [0.01, 0.99]) that achieved the best performance on the training set.

[2] https://www.ibm.com/products/ilog-cplex-optimization-studio

Chapter 5

Experimental Evaluation

The experiments done were all combinations of the discretisation methods, encoding methods and classification models described in Chapter 4.

In this chapter, first, the **datasets** used for evaluation are described, with particular detail in the `aesthetic` dataset, which received the most focus in this work. Next, the **performance metrics** used are explained. The **implementation details** - the model setup and hyperparameter tuning - are explained. The **results** are then reported, comparing the performance of the two discretisation methods, two encoding methods, and four classifiers. Finally, an **exploratory analysis** is presented, including results for the `aesthetic` dataset using a manually selected subset of features, experiments with ensembles of scorecards and with RiskSLIM.

5.1 Datasets

A set of publicly available datasets for both binary and ordinal classification tasks was used for testing, considering those used in state-of-the-art scoring systems. All datasets, except for the `aesthetic` and `appendicitis` datasets, were obtained from the UCI Machine Learning Repository [32].

The following subsections describe the preprocessing applied to all datasets, followed by details on the `aesthetic` dataset, and then summaries of the binary and ordinal datasets used for evaluation.

5.1.1 Preprocessing

All datasets went through the same preprocessing steps. First, all rows containing missing values were removed, so the number of observations reported in the following sections corresponds to the number of rows remaining after this removal.

Furthermore, the target variables were mapped into integer values to make them suitable for the models used. For binary datasets, the positive class was mapped to 1 and the negative class to 0. For ordinal datasets, the classes were mapped to ordered integer values, reflecting their natural order (e.g., "Excellent", "Good", "Fair", "Poor" mapped to 1, 2, 3, and 4, respectively).

5.1.2 Aesthetic Evaluation Dataset

The `aesthetic` dataset was obtained from the Portuguese Champalimaud Foundation in the context of the CINDERELLA project [31]. This dataset contains measurements extracted from clinical images of breast cancer patients who underwent conservative breast surgery (BCCT, Breast Cancer Conservative Treatment). The target variable, named "Subjective Evaluation", has the aesthetic results evaluations assigned by three oncoplastic surgeons. These evaluations follow the Harvard Scale introduced by Harris et al. (1979) [25], which categorises results into four ordered classes: "Excellent", breasts are almost identical; "Good", minimal variations between breasts; "Fair", noticeable differences but without significant distortion; and "Poor", significant distortion present.

The images were converted to tabular data by manual detection of anatomical keypoints, followed by automatic computation, as described by Cardoso and Cardoso (2007) [8].

The dataset used in the following sections is represented as `aesthetic ordinal`, and a binary version, `aesthetic binary`, was created by mapping "Poor" and "Fair" to 0, and "Good" and "Excellent" to 1.

5.1.3 Binary datasets

A description of the binary datasets can be found in Table 5.1. The datasets are listed in ascending order of $n \times d$, where n is the number of samples and d is the number of features.

Table 5.1: Binary datasets used for testing, where n is the number of samples, d is the number of features, and y is target variable.

Dataset	n	d	Proportion of samples with $y = 1$	Condition for $y = 1$	Reference
`appendicitis`	106	7	19.8%	person has appendicitis	[66]
`mammo`	830	5	48.2%	mammographic mass is benign	[16]
`breast cancer`	683	9	35.0%	patient has breast cancer	[67]
`aesthetic binary`	1358	23	66.5%	good aesthetic result after BCCT	[31]
`spambase`	4601	57	39.4%	email is spam	[29]
`income`	30162	14	24.9%	person's annual income exceeds $50,000	[5]

5.1.4 Ordinal datasets

The ordinal datasets are summarised in Table 5.2, which are also ordered in ascending order of $n \times d$.

Table 5.2: Ordinal datasets used for testing, where n is the number of samples, d is the number of features, K is the number of target classes, and K_1 to K_7 is the number of observations (and proportion in percentage) of each class.

Dataset	n	d	K	K_1	K_2	K_3	K_4	K_5	K_6	K_7	Classification task	Reference
`small wine`	178	13	3	71 (40%)	59 (33%)	48 (27%)	-	-	-	-	wine quality (1–3)	[1]
`aesthetic ordinal`	1358	23	4	308 (23%)	595 (44%)	353 (26%)	102 (7%)	-	-	-	BCCT aesthetic results	[31]
`abalone`	4177	8	5	189 (5%)	2541 (61%)	1186 (28%)	225 (5%)	36 (1%)	-	-	abalone's age group	[44]
`large wine`	6497	11	7	30 (0.5%)	216 (3%)	2138 (33%)	2836 (44%)	1079 (17%)	193 (3%)	5 (0.1%)	wine quality (0–10)	[12]

5.2 Performance Metrics

The main performance metric used is accuracy, which measures the proportion of correctly classified examples relative to the total number of examples. This metric was also used as the scoring function during the grid search:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{\sum_{i=1}^{n} \mathbb{1}\{\hat{y}_i = y_i\}}{n}, \tag{5.1}$$

where $\hat{y}_i$ is the predicted label, y_i is the real label and n is the total number of samples.

Another crucial metric is model size (sparsity), which is directly related to interpretability. Although there is no standard measure for sparsity, it can be evaluated with the proportion of nonzero coefficients relative to the total number of coefficients after encoding and by counting the number of nonzero coefficients.

$$\text{Model Size} = \text{Number of Nonzero Coefficients} \quad \text{or} \quad \text{Model Size} = \frac{\text{Number of Nonzero Coefficients}}{\text{Total Number of Coefficients}} \tag{5.2}$$

The confusion matrix shows the distribution of predictions for each class. It allows for the identification of systematic misclassification. In the binary case, it is:

$$\text{CM} = \begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix} \tag{5.3}$$

where TN, FP, FN, and TP are the number of true negatives, false positives, false negatives, and true positives, respectively.

Additional metrics such as Balanced Accuracy, Mean Squared Error (MSE) and Execution Time were also computed. These are reported in Appendix A, but are not discussed in this chapter.

5.3 Implementation Details

The train/test split was set to 75%/25%, and hyperparameter tuning was performed using grid search with 5-fold cross-validation.

For the libraries and hyperparameters used for the classifiers, the following configurations were adopted:

- **Logistic Regression**: implemented using the `LogisticRegression` class from the `scikit-learn`[1] library. Hyperparameter tuning was done over the regularisation parameter $\alpha \in \{0.001, 0.01, 0.1, 0.4, 0.6, 0.9, 0.99, 1.0\}$, which corresponds to the inverse parameter $C = 1/\alpha$ in `scikit-learn`. Two solver-penalty combinations were explored: the `liblinear` solver with an L1 penalty, and the `saga` solver with an Elastic Net penalty, with the `l1-ratio`,

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

the parameter that controls the balance between L1 and L2 regularisation, set to 0.4, 0.6, or 0.8.

- **SVM**: linear SVM was implemented using the `SVC` class from `scikit-learn`[2], with the kernel set to `'linear'`. Hyperparameter tuning was performed over the regularisation parameter C, with values from the range $C \in \{2^i \mid i = -10, \ldots, 10\}$ to explore both very small and very large regularisation strengths.

- **Adaptive Lasso**: implemented using the `sparse-lm`[3] Python package. Hyperparameter tuning was done over the regularisation parameter (λ), with values ranging from e^{-10} to e^{10}.

- **skglm**: implemented using the `skglm`[4] library, which is compatible with the `scikit-learn` API. For the data fit, the `Huber` loss function was used, for robustness to outliers, as it applies squared loss for small residuals and absolute loss for large residuals. For the penalties, two non-convex sparsity-inducing penalties were used: `MCPenalty` (Minimax Concave Penalty), which shrinks small coefficients strongly while reducing bias for larger coefficients by applying weaker penalisation as they grow; and `SCAD` (Smoothly Clipped Absolute Deviation), similar to MCP but applies even less shrinkage to large coefficients, resulting in nearly unbiased estimates for strong predictors.

- **RiskSLIM**: implemented by adapting the code from the authors [63][5].

The performance of all models was evaluated using the average results over five runs.

5.4 Results

This section compares the impact of discretisation, encoding, and classifier alternatives on accuracy and model size. Results are presented in tables and supported by plots to illustrate the trade-offs between sparsity and predictive performance.

5.4.1 Comparison between Discretisation Techniques

Tables 5.3 and 5.4 present a comparison between the two discretisation techniques used in this study: CAIM and Infinitesimal Bins, for binary and ordinal datasets, respectively. The encoding method used is Differential Coding. Figures 5.1 and 5.2 also illustrate this comparison by plotting the model size against test accuracy.

Overall, the results show that CAIM consistently produces much more compact models than Infinitesimal Bins. This was expected because of the number of encoded columns Infinitesimal Bins produces compared to CAIM. For example, in the `aesthetic binary` dataset, Infinitesimal Bins leads to thousands of features (up to 22374), while CAIM achieves similar or higher

[2] https://scikit-learn.org/stable/modules/svm.html
[3] https://pypi.org/project/sparse-lm/
[4] https://github.com/scikit-learn-contrib/skglm
[5] https://github.com/ustunb/risk-slim

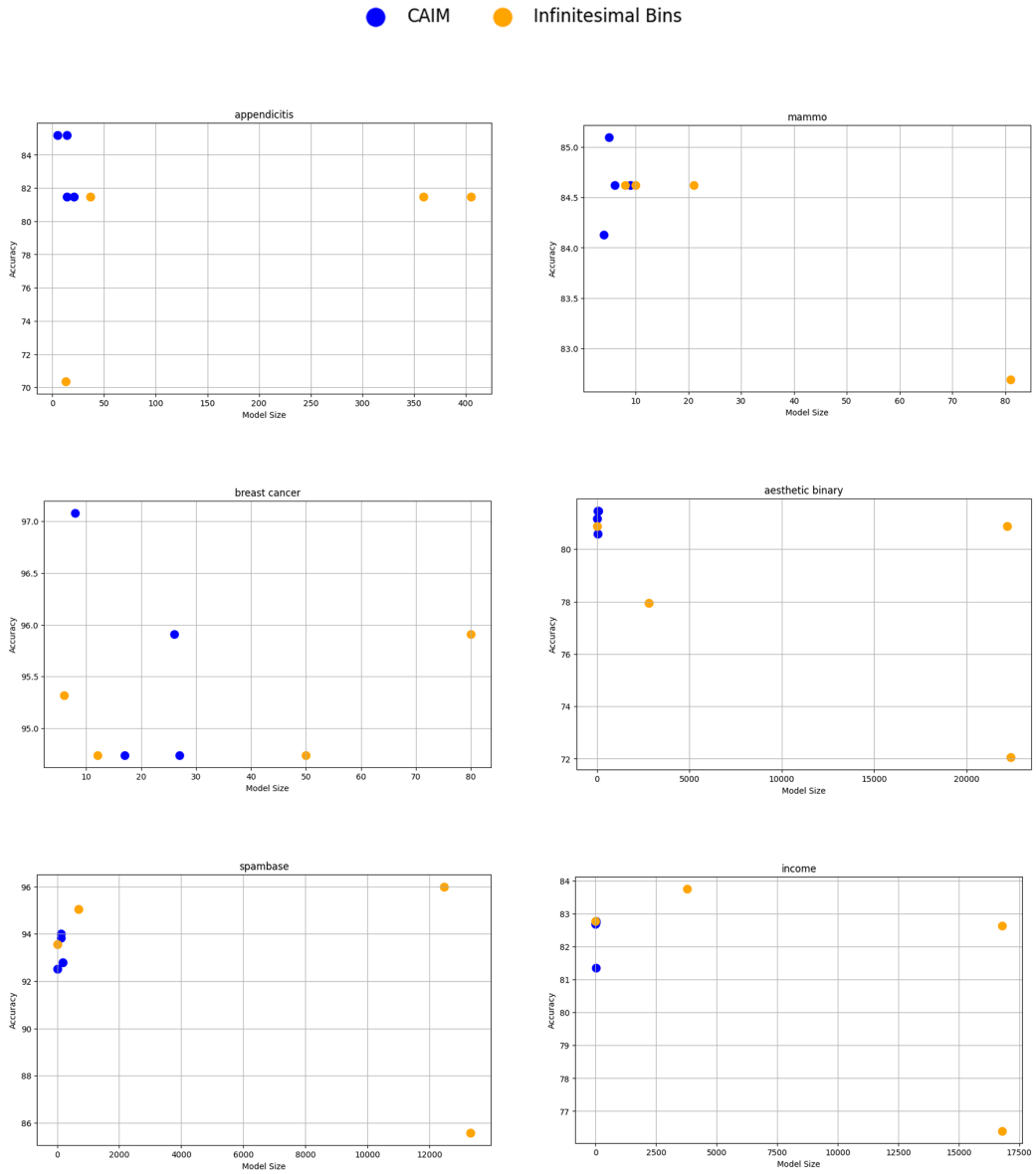


Figure 5.1: **Test accuracy** (y-axis) **versus model size** (x-axis) **for each discretisation technique** across **binary datasets**. Each point represents a classifier combined with one discretisation method: CAIM (blue) or Infinitesimal Bins (orange). Encoding was performed with Differential Coding.

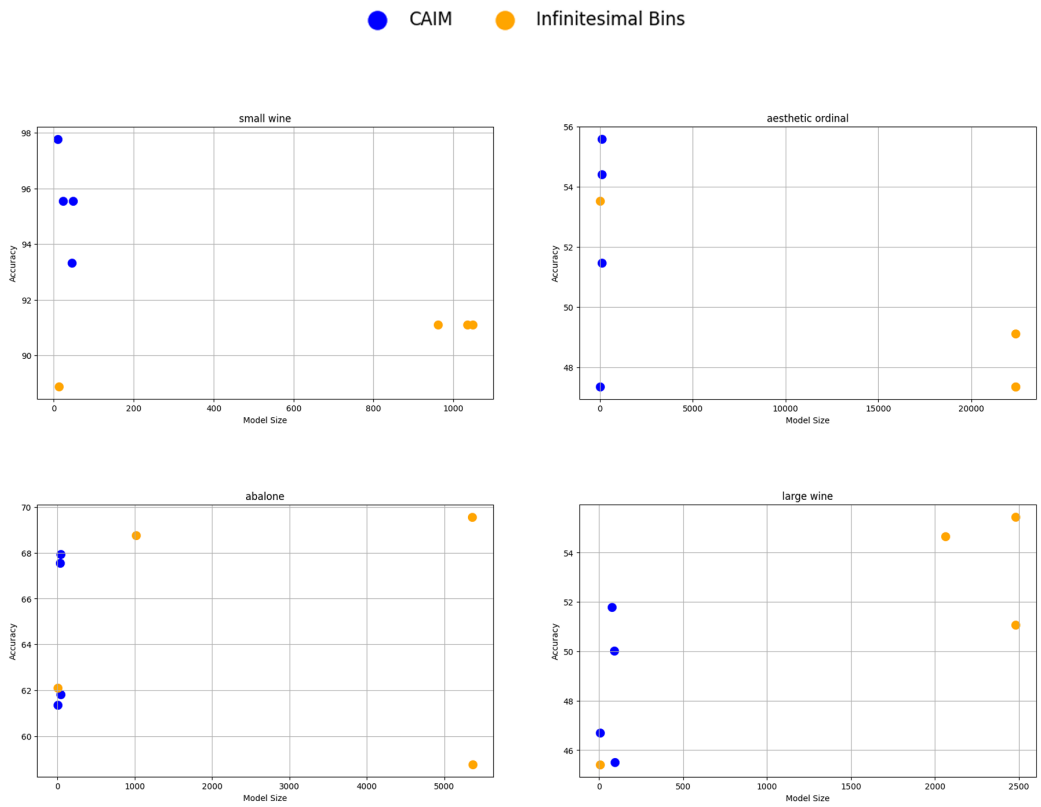


Figure 5.2: **Test accuracy** (y-axis) **versus model size** (x-axis) **for each discretisation technique** across **ordinal datasets**. Each point represents a classifier combined with one discretisation method: CAIM (blue) or Infinitesimal Bins (orange). Encoding was performed with Differential Coding.

Table 5.3: **Comparing discretisation techniques**: CAIM vs Infinitesimal Bins. Performance results per **binary** dataset, discretisation method, and model. The encoding technique used is Differential Coding. Each model reports the test accuracy and the model size.

Binary Dataset	CAIM								Infinitesimal Bins							
	LR		SVM		skglm		Adaptive Lasso		LR		SVM		skglm		Adaptive Lasso	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
appendicitis	81.48	14	85.19	14	**85.19**	**5**	81.48	21	81.48	37	81.48	359	70.37	13	81.48	405
mammo	84.62	6	84.62	9	**85.10**	**5**	84.13	4	84.62	21	82.69	81	84.62	8	84.62	10
breast cancer	94.74	17	95.91	26	**97.08**	**8**	94.74	27	94.74	50	95.91	80	95.32	6	94.74	12
aesthetic binary	81.47	27	80.59	53	**81.18**	**10**	81.47	66	77.94	2802	80.88	22208	80.88	16	72.06	22374
spambase	94.01	125	93.83	131	92.53	14	92.79	171	95.05	686	**96.00**	12460	93.57	14	85.58	13313
income	82.73	17	81.36	17	82.68	5	82.77	18	**83.76**	**3793**	82.77	16783	82.63	8	76.40	16784

accuracy with less than 100 features. Another example is that on mammo, both methods reach an accuracy of 84.62% with Logistic Regression, but CAIM has a model size of 6 compared to 21. The plots confirm this, with CAIM points clustered on the left side of the x-axis (smaller model sizes) while maintaining comparable accuracy. The table and the plots also show that Infinitesimal Bins grows disproportionately as the size and dimensionality of the dataset increase. So, the difference in size between the two discretisers is even more evident in high-dimensional datasets such as spambase and aesthetic binary, suggesting that Infinitesimal Bins does not scale well, while CAIM remains stable. In smaller datasets, such as appendicitis and mammo, accuracy is preserved, and although the model sizes are larger than the ones produced by CAIM, they do not become excessively large, indicating that Infinitesimal Bins is more usable for smaller datasets, where granularity may be valuable. For larger datasets, reducing model size by, for example, merging redundant bins or removing overly specific bins, could substantially improve performance.

Regarding predictive performance, CAIM is either better or comparable to Infinitesimal Bins across most datasets. There are a few exceptions, such as spambase, where Infinitesimal Bins achieves higher accuracy in two models compared to CAIM. Similarly, in spambase, with skglm, both discretisers produced models of equal size, but Infinitesimal Bins led to better accuracy. In the ordinal setting, Infinitesimal Bins achieved the best accuracy in the largest datasets.

In conclusion, these results show that CAIM provides a better balance between model size and accuracy, while Infinitesimal Bins only rarely improves predictive power and does so consistently at the cost of interpretability.

Table 5.4: **Comparing discretisation techniques**: CAIM vs Infinitesimal Bins. Performance results per **ordinal** dataset, discretisation method, and model. The encoding technique used is Differential Coding. Each model reports the test accuracy and the model size.

Ordinal Dataset	CAIM								Infinitesimal Bins							
	LR		SVM		skglm		Adaptive Lasso		LR		SVM		skglm		Adaptive Lasso	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
small wine	93.33	45	95.56	48	**95.56**	**10**	97.78	23	91.11	962	91.11	1035	88.89	12	91.11	1048
aesthetic ordinal	**55.59**	**95**	54.41	107	47.35	9	51.47	112	47.35	22376	47.35	22373	53.53	11	49.12	22376
abalone	67.94	44	67.56	38	61.34	7	61.82	45	68.77	1013	**69.57**	**5358**	62.11	8	58.76	5366
large wine	51.78	77	50.03	92	45.52	7	46.71	93	54.65	2062	**55.45**	2479	45.42	7	51.08	2479

5.4.2 Comparison between Encoding Techniques

Tables 5.5 and 5.6 show the results of the comparison between One-hot Encoding and Differential Encoding, for binary and ordinal datasets, respectively. The discretisation technique used is Infinitesimal Bins. Figures 5.3 and 5.4 also illustrate this comparison by plotting the model size against test accuracy.

Table 5.5: **Comparing encoding techniques**: One-Hot Encoding vs. Differential Coding. Performance results per **binary** dataset, encoding method, and model. The discretisation technique used is Infinitesimal Bins. Each model reports the test accuracy and the model size.

Binary Dataset	One-Hot Encoding								Differential Coding							
	LR		SVM		skglm		Adaptive Lasso		LR		SVM		skglm		Adaptive Lasso	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
appendicitis	81.48	404	81.48	398	**81.48**	**14**	77.78	405	81.48	37	81.48	359	70.37	13	81.48	405
mammo	84.62	74	84.13	75	82.69	5	83.17	83	84.62	21	82.69	81	**84.62**	**8**	84.62	10
breast cancer	94.74	66	94.74	79	90.06	22	94.15	80	94.74	50	95.91	80	**95.32**	**6**	94.74	12
aesthetic binary	70.59	22374	72.35	22374	66.18	0	72.65	14897	77.94	2802	80.88	22208	**80.88**	**16**	72.06	22374
spambase	87.26	12986	88.62	11979	58.73	3	83.32	13313	95.01	683	**96.00**	**12460**	93.57	14	85.58	13313
income	**84.14**	**604**	83.60	8980	77.56	6	76.50	11	83.76	3793	82.77	16783	82.63	8	76.40	16784

Table 5.6: **Comparing encoding techniques**: One-Hot Encoding vs. Differential Coding. Performance results per **ordinal** dataset, encoding method, and model. The discretisation technique used is Infinitesimal Bins. Each model reports the test accuracy and the model size.

Ordinal Dataset	One-Hot Encoding								Differential Coding							
	LR		SVM		skglm		Adaptive Lasso		LR		SVM		skglm		Adaptive Lasso	
	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size
small wine	64.44	1012	64.44	998	57.78	18	73.33	1048	**91.11**	**962**	91.11	1035	88.89	12	91.11	1048
aesthetic ordinal	45.00	22376	45.29	22376	41.47	2	42.65	22376	47.35	22376	47.35	22373	**53.53**	**11**	49.12	22376
abalone	65.55	1310	65.74	5027	61.34	4	62.39	5369	68.77	1013	**69.57**	**5358**	62.11	8	58.76	5366
large wine	53.66	1898	53.91	2405	45.42	4	50.65	2479	54.65	2062	**55.45**	**2479**	45.42	7	51.08	2479

Combining Infinitesimal Bins with One-Hot Encoding resulted in the worst performance of all discretisation–encoding pairs. Although training accuracy often reaches 100% (see Appendix A), except for skglm, test performance drops substantially, showing strong overfitting. This issue is especially evident in the ordinal datasets, where the models tend to predominantly predict the majority class. For example, as shown in the confusion matrix in Figure 5.5a, with the `aesthetic ordinal` dataset, the classifier predominantly predicts class 3 ("Fair"). Specifically, skglm often predicts only the majority class (Figure 5.5b). As Infitesimal Bins produces a lot of features, and each feature is very specific to each example in the data, when combined with One-hot Encoding, the resulting encoded dataset is extremely sparse. Consequently, the models overfit the training data and fail to generalise.

In addition, Differential Coding produces more compact models with comparable or better accuracy. This is visible in the plots, with the green points (Differential Coding) being placed more to the left (smaller model sizes), often close to the upper-left corner (high accuracy with fewer coefficients). This pattern is observed in both binary and ordinal datasets. For instance, on `spambase`, One-Hot Encoding with Logistic Regression achieved 87.26% accuracy with a model size of almost 13,000, while Differential Coding attained an accuracy of 95.01% with less than 700 coefficients. In the ordinal datasets, the difference is even more pronounced. On `small`

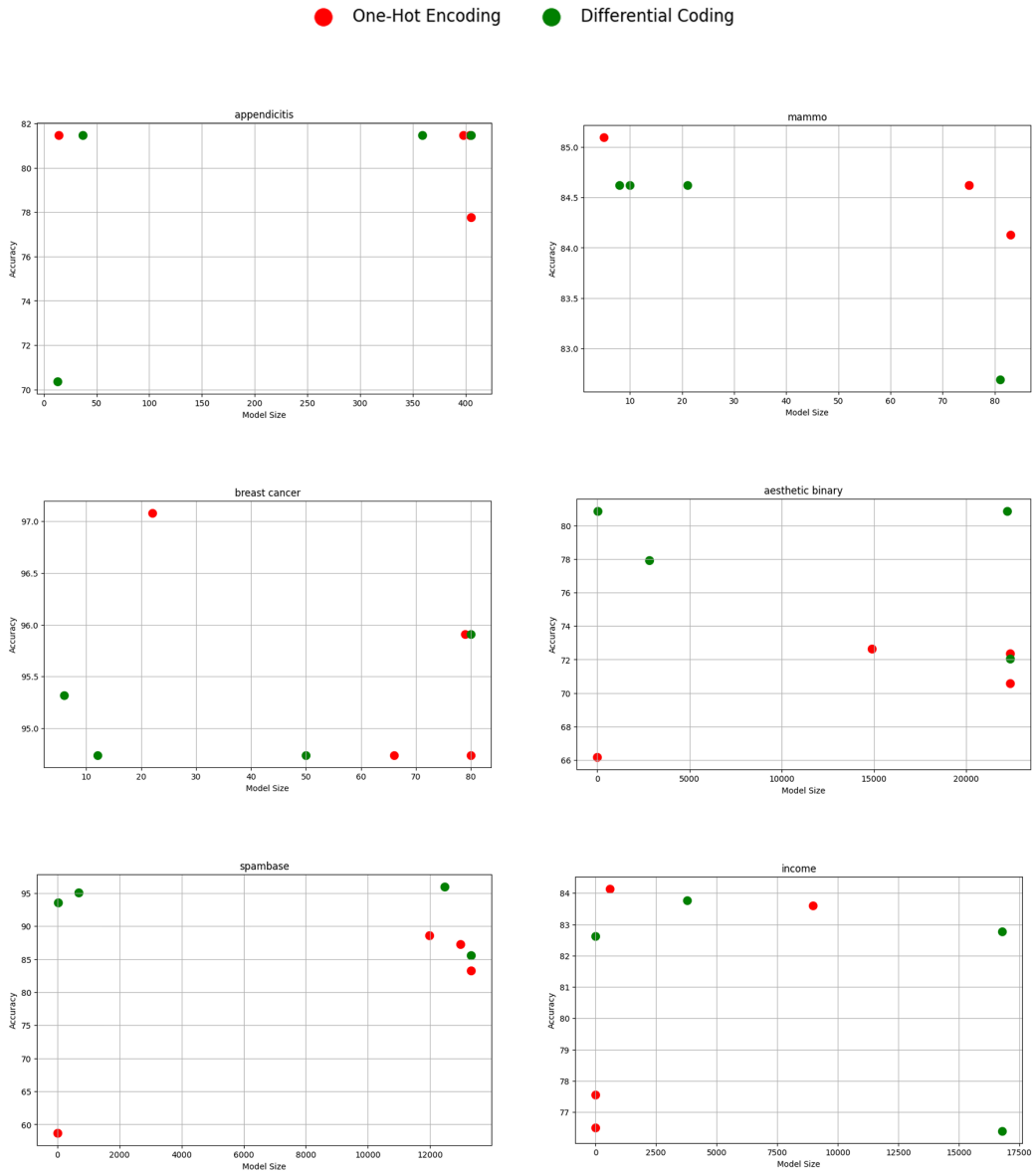


Figure 5.3: **Test accuracy** (y-axis) **versus model size** (x-axis) **for each encoding technique** across **binary datasets**. Each point represents a classifier combined with one encoding method: One-Hot Encoding (red) or Differential Coding (green). Discretisation was performed with Infinitesimal Bins.

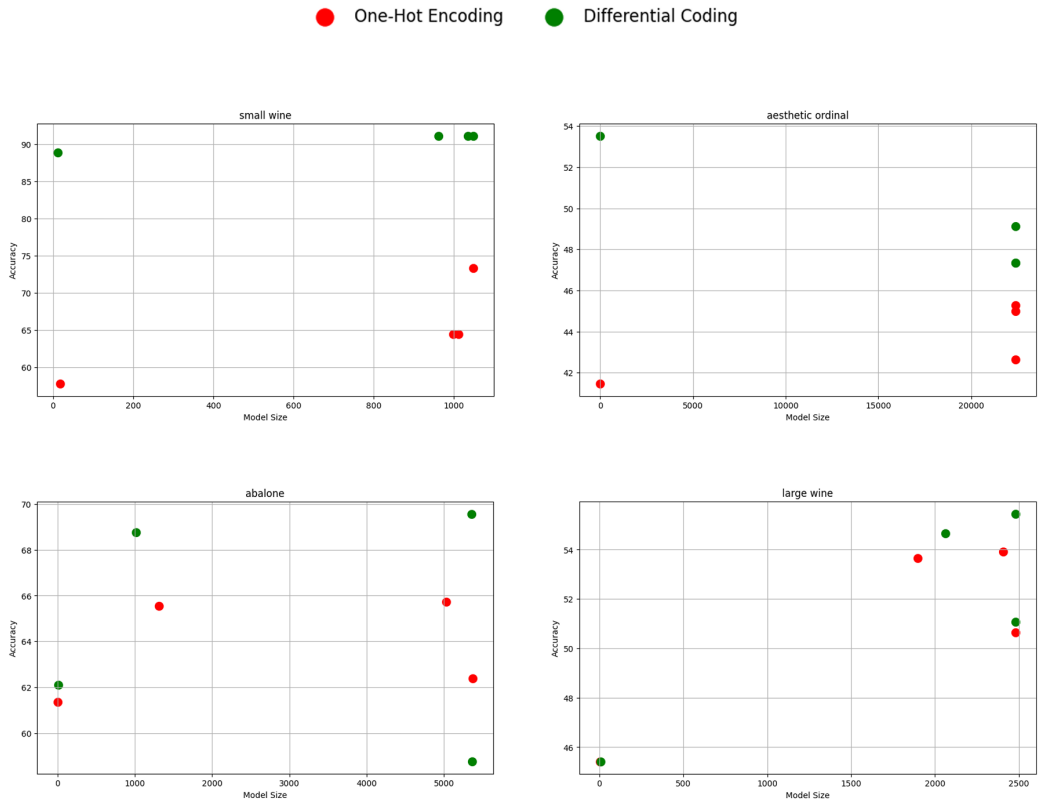
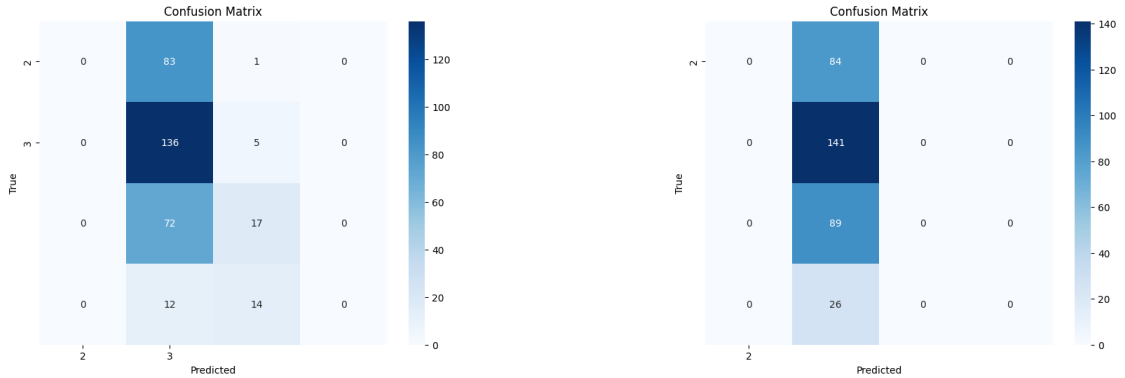


Figure 5.4: **Test accuracy** (y-axis) **versus model size** (x-axis) **for each encoding technique** across **ordinal datasets**. Each point represents a classifier combined with one encoding method: One-Hot Encoding (red) or Differential Coding (green). Discretisation was performed with Infinitesimal Bins.

`wine`, One-Hot Encoding reaches 64.44% accuracy with more than 1000 coefficients, and Differential Coding achieves 91.11% accuracy with a similar model size, and 88.89% with just 12 coefficients. In the plots, especially for the ordinal datasets, the green dots (Differential Coding) consistently appear above the red dots (One-Hot Encoding), indicating that Differential Coding provides stronger predictive power.

With CAIM, the difference between One Hot Encoding and Differential Coding is minimal in accuracy, but Differential Coding slightly decreases the model size (see Appendix A).

Therefore, One-hot Encoding works well with discretisation methods that produce fewer intervals, like CAIM, but it performs poorly with fine-grained methods such as Infinitesimal Bins because of the extreme sparsity it creates. The main conclusion is that the choice of encoding is most critical when Infinitesimal Bins is used, but Differential Coding is preferable in both cases.



(a) Logistic Regression.

(b) skglm.

Figure 5.5: Confusion matrices for Logistic Regression (a) and skglm (b) on the `aesthetic ordinal` dataset with Infinitesimal Bins and One-Hot Encoding.

5.4.3 Comparison between Classifiers

This section compares the performance of four classifiers: Logistic Regression, SVM, Adaptive Lasso, and skglm. The evaluation is presented separately for binary and ordinal classification tasks.

5.4.3.1 Binary Problems

Table 5.7 reports the performance of the four classifiers across the binary datasets, and Figure 5.6 illustrates the relationship between model size and test accuracy.

In general, **skglm** stands out for producing smaller models, achieving significant reductions in model size compared to the other classifiers. This sparsity is evident in the plots, where skglm points (in orange) appear at the lower end of the x-axis. Despite these smaller model sizes, skglm maintains competitive accuracy in both training and test sets, achieving the highest accuracy in three of the six binary datasets. This is visible in the plots because most of the orange points

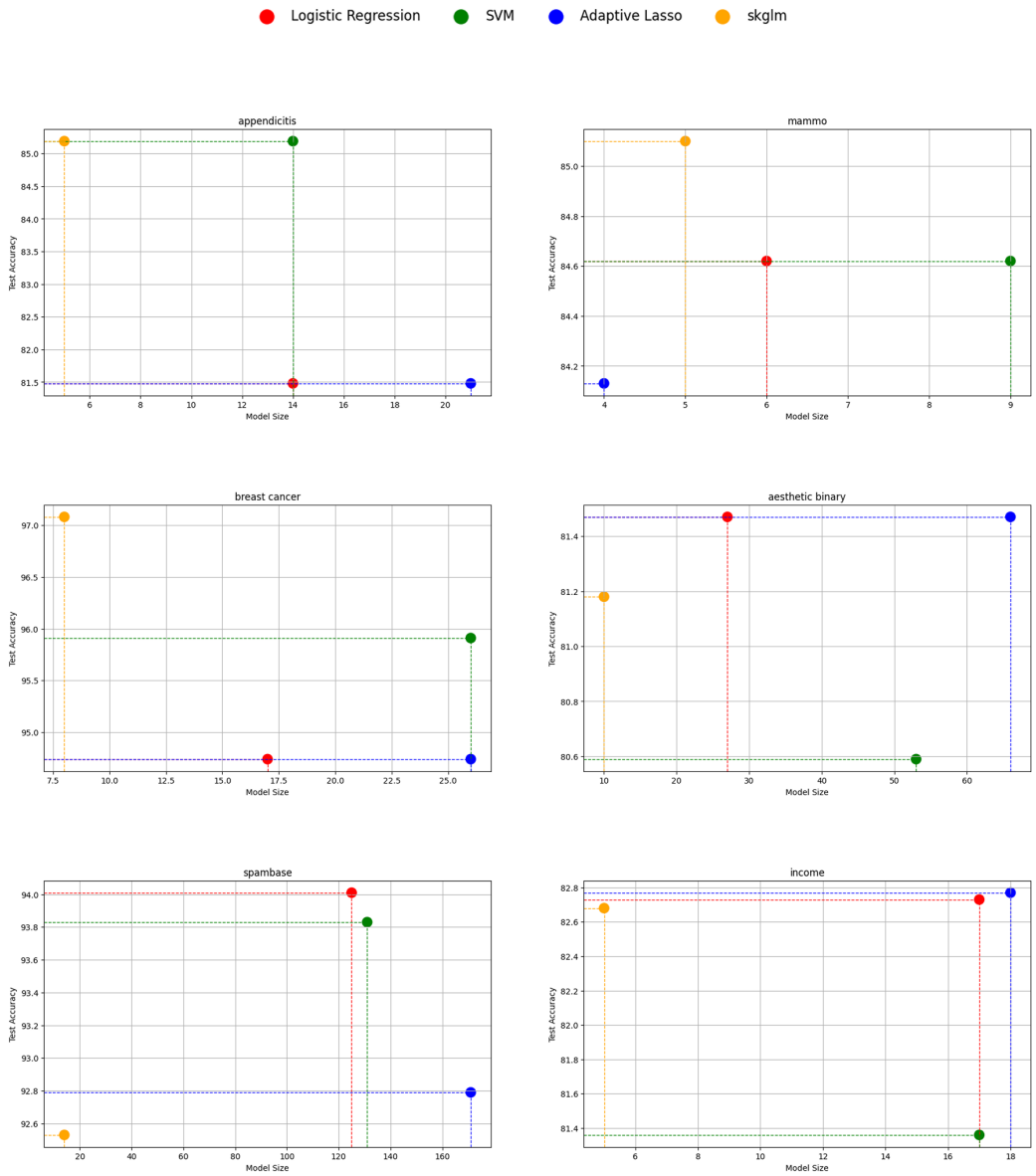


Figure 5.6: **Test accuracy** (y-axis) **versus model size** (x-axis) **for each classifier** across **binary datasets**. Each point corresponds to a classifier (Logistic Regression in red, SVM in green, Adaptive Lasso in blue, and skglm in orange). Discretisation was performed using CAIM and encoding with Differential Coding.

Table 5.7: **Comparing classifiers**: Logistic Regression vs. SVM vs. Adaptive Lasso vs. skglm. Performance results per **binary** dataset and classifier. Each classifier reports the train accuracy, the test accuracy, and the model size. The discretisation was done using CAIM, and the encoding was Differential Coding.

Binary Dataset	Logistic Regression			SVM			skglm			Adaptive Lasso		
	Train Acc.	Test Acc.	Size	Train Acc.	Test Acc.	Size	Train Acc.	Test Acc.	Size	Train Acc.	Test Acc.	Size
appendicitis	91.14	81.48	14	88.61	85.19	14	**89.87**	**85.19**	**5**	89.87	81.48	21
mammo	80.23	84.62	6	79.90	84.62	9	**80.06**	**85.10**	**5**	84.13	84.13	4
breast cancer	97.85	94.74	17	97.66	95.91	26	**97.07**	**97.08**	**8**	97.27	94.74	27
aesthetic binary	**83.89**	**81.47**	**27**	83.01	80.59	53	82.71	81.18	10	84.09	81.47	66
spambase	**94.12**	**94.01**	**125**	94.03	93.83	131	91.88	92.53	14	93.19	92.79	171
income	82.43	82.73	17	81.17	81.36	17	82.25	82.68	5	**82.42**	**82.77**	**18**

are near the upper left corner, representing the desirable combination of high accuracy and small model size. In the remaining three binary datasets, skglm shows a slight decrease in test accuracy compared to the best-performing models, between 1% and 2%. However, the differences are minimal, suggesting that the reduction in the number of coefficients does not significantly compromise the predictive performance. Moreover, this trade-off can be considered favourable because of the increase in interpretability, and justifies the selection of skglm over the more complex models.

Adaptive Lasso was selected for its ability to achieve models with high sparsity. In some datasets, it succeeds, having a reduced model size compared to Logistic Regression or SVM. For example, in `mammo`, it produces one of the smallest models while having high accuracy. However, the graphs show that Adaptive Lasso often results in model sizes larger than expected and uses all coefficients in the encoded dataset, such as in `spambase`, `aesthetic binary`, and `income`. It sometimes even exceeds Logistic Regression. Its performance is less consistent than skglm, showing variability in the trade-off between sparsity and accuracy depending on the dataset.

Logistic Regression serves as a strong baseline, providing competitive accuracy across all datasets. The plots show that Logistic Regression points (shown in red) are often located in the middle or right side of the x-axis, indicating larger model sizes. As a result, while it consistently achieves high accuracy, it is less effective in producing sparse models even with regularisation.

SVM also demonstrates good test accuracy, comparable to Logistic Regression and, in some cases, even higher (e.g., `appendicitis`). Nevertheless, its model size is generally very large, often using more coefficients than Logistic Regression. This is evident in the plots, where the SVM model (shown in green) is located more to the right side.

In conclusion, for binary classification problems, skglm achieves the highest accuracy using a minimal number of coefficients in half of the cases. For the other half, it maintains a low number of coefficients while having only a slight reduction in accuracy compared to the other coefficients. This makes skglm the preferred classifier for this task.

5.4.3.2 Ordinal Problems

Table 5.8 reports the performance of the four classifiers across the ordinal datasets, and Figure 5.7 illustrates the relationship between model size and test accuracy.

Table 5.8: **Comparing classifiers**: Logistic Regression vs. SVM vs. Adaptive Lasso vs. skglm. Performance results per **ordinal** dataset and classifier. Each classifier reports the train accuracy, the test accuracy, and the model size. The discretisation was done using CAIM, and the encoding was Differential Coding.

Ordinal Dataset	Logistic Regression			SVM			skglm			Adaptive Lasso		
	Train Acc.	Test Acc.	Size	Train Acc.	Test Acc.	Size	Train Acc.	Test Acc.	Size	Train Acc.	Test Acc.	Size
`small wine`	98.50	93.33	45	96.99	95.56	48	94.74	95.56	10	**94.74**	**97.78**	**23**
`aesthetic ordinal`	**64.64**	**55.59**	**95**	64.93	54.41	107	55.01	47.35	9	60.02	51.47	112
`abalone`	**68.39**	**67.94**	**44**	68.71	67.56	38	60.66	61.34	7	61.49	61.82	45
`large wine`	**52.74**	**51.82**	**76**	51.21	50.03	92	43.06	45.42	7	45.14	46.71	93

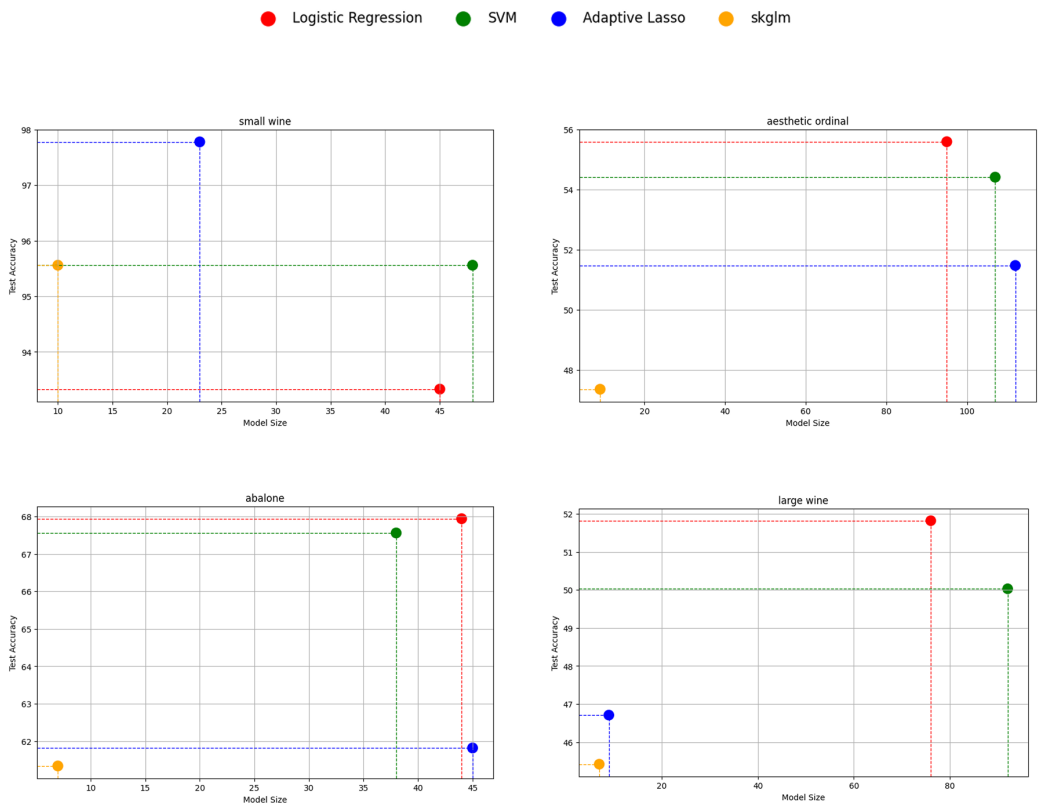


Figure 5.7: **Test accuracy** (y-axis) **versus model size** (x-axis) **for each classifier** across **ordinal datasets**. Each point corresponds to a classifier (Logistic Regression in red, SVM in green, Adaptive Lasso in blue, and skglm in orange). Discretisation was performed using CAIM and encoding with Differential Coding.

In the ordinal datasets, **skglm** is also the model that consistently produces the smallest models, as indicated by the orange dots positioned further to the left in the plots. However, this simplicity comes at the expense of performance - skglm achieves the lowest accuracy overall, with the orange dots positioned in the bottom-left corner. Its balanced accuracy is also poor (see Appendix A), as skglm primarily predicts the majority class.

Adaptive Lasso demonstrates strong performance in certain scenarios, achieving good accuracy while using relatively few coefficients, such as in the `small wine` dataset. However, as in the binary setting, in some instances, it uses all available coefficients, making it unstable. When combined with Infinitesimal Bins (see Appendix A), it frequently reaches 100% training accuracy, which highlights its lack of robustness.

Logistic Regression generally achieves the highest accuracy, and **SVM** is competitive, although both classifiers typically use many coefficients, with the red and green dots clustered in the upper-right corner.

In conclusion, for ordinal problems, classifier choice depends on the objective: interpretability (skglm, occasionally Adaptive Lasso) or predictive accuracy (Logistic Regression and SVM). No single classifier outperforms all others across various datasets.

5.5 Exploratory Analysis

This section presents a set of complementary analyses conducted on the `aesthetic` dataset to investigate different strategies for improving interpretability and predictive performance. These analyses focused on three distinct strategies. First, the effect of **manual feature selection** was studied by comparing features selected by experts against the complete set of features. Second, the potential of **ensemble** methods was explored by integrating an existing scorecard into the modelling pipeline. Finally, the **RiskSLIM** framework was tested as an alternative approach for learning the coefficient weights for ordinal classification.

5.5.1 Manual Feature Selection

To test the impact of feature selection on the `aesthetic` dataset, the original 23 features were reduced to 4 hand-selected features by experts: "pLBC Value", "pBCE Value", "cX2b Value" and "pUNR Value". Three of these features, *pLBC*, *pBCE*, and *pUNR*, correspond to asymmetry measures that capture geometric differences between the treated breast and untreated breast after surgery. *pLBC* (Local Breast Contour) measures the “difference between levels of inferior breast contour”; *pBCE* (Breast Compliance Evaluation) measures the “difference between left and right nipple to inframammary fold distance”; and *pUNR* (Uppward Nipple Retraction) measures the “difference between nipple levels”. In contrast, *cX2b* is a colour difference feature that measures the χ^2 statistic of the *b* channel histogram in the CIE *L*a*b* colour space [8]. The results for both versions of the dataset are shown in Table 5.9, where test accuracy and model size are reported for all combinations of discretisation, encoding, and classifiers.

As expected, manual feature selection consistently reduced model size, leading to improvements in interpretability. For example, model complexity dropped from over 22000 coefficients with Infinitesimal Bins to 4070, and from 107 to 22 with CAIM + SVM. Regarding predictive performance, accuracies remained similar between both datasets, with a slight improvement when using the hand-selected features. The best results were obtained using the manually selected features with CAIM discretisation and logistic regression, where both encoding techniques achieved the same performance, with about 58% accuracy. Differential coding resulted in fewer weights, totalling 15.

Table 5.9: `aesthetic ordinal` dataset results for different discretisation, encoding, and classifier combinations. The comparison is made between using **all features versus manually selected features**. Performance is reported as test accuracy (in %) and model size, measured by the number of non-zero coefficients.

			All Data		Manually Selected Data	
Discretisation	Encoding	Classifier	Test Accuracy (%)	Model Size	Test Accuracy (%)	Model Size
CAIM	One-hot Encoding	Logistic Regression	55.58	56	57.65	22
		SVM	53.85	107	53.82	22
		skglm	48.23	8	50.00	10
		Adaptive Lasso	50.59	44	50.59	22
	Differential Coding	Logistic Regression	55.58	74	**57.65**	**15**
		SVM	54.41	107	53.82	22
		skglm	47.35	9	50.59	10
		Adaptive Lasso	50.88	112	50.59	22
Infinitesimal Bins	One-hot Encoding	Logistic Regression	42.94	11175	42.35	2255
		SVM	45.29	22376	44.12	4070
		skglm	41.47	2	41.47	2
		Adaptive Lasso	42.94	22376	42.84	4070
	Differential Coding	Logistic Regression	51.11	1022	52.94	1487
		SVM	46.77	22376	49.71	4065
		skglm	52.06	11	47.65	9
		Adaptive Lasso	47.35	22376	48.24	4070

5.5.2 Ensemble of Scorecards

Ensemble methods are a common strategy in machine learning to improve predictive performance by combining the strengths of multiple models. The idea is that one scorecard may only capture part of the data, while combining several scorecards can make the model more accurate without sacrificing interpretability.

Motivated by this idea, an ensemble of scorecards was created by incorporating the predictions of an existing scorecard as an additional feature. This scorecard, shown in Table 5.10, contains nine rules and was constructed according to Silva and Cardoso (2015) [56] using private data from the European Project PICTURE [54]. Specifically, the points predicted by this scorecard were added as an additional feature to the dataset and then used as input for training new scorecards.

Table 5.11 reports the performance of scorecards without and with the ensemble. Metrics include train accuracy, test accuracy, and model size, where the ensemble model size includes the nine rules of the existing scorecard.

Table 5.10: Existing scorecard for the `aesthetic ordinal` dataset. Points are assigned per feature.

Feature	Interval	Points
pLBC Value	[0.0174, 0.1012[	0.1538
	[0.1012, 0.1707[	0.4379
	≥ 0.1707	0.5767
pBCE Value	[0.0184, 0.3684[	0.0636
	[0.3684, 0.9219[	0.1938
	≥ 0.9219	0.2128
cX2b Value	[0.0140, 0.0267[	0.0299
	[0.0267, 0.6881[	0.0642
	≥ 0.6881	0.0727

Total Points	Predicted Class
< 0.1738	Excellent
< 0.3720	Good
< 0.7002	Fair
≥ 0.7002	Poor

Table 5.11: Performance of scorecards without and with the ensemble on the `aesthetic ordinal` dataset. Results are shown for two discretisation techniques (CAIM and Infinitesimal Bins) and three classifiers (Logistic Regression, SVM and skglm), using Differential Coding for encoding. Metrics reported include train accuracy, test accuracy, and model size.

		Without Ensemble		With Ensemble	
Discretisation	Classifier	Test Acc. (%)	Model Size	Test Acc. (%)	Model Size
CAIM	Logistic Regression	55.59	95	**58.24**	**20 (+9)**
	SVM	**54.41**	107	54.12	**26 (+9)**
	skglm	47.35	**9**	**51.47**	9 (+9)
Infinitesimal Bins	Logistic Regression	47.35	22376	**50.59**	**4106 (+9)**
	SVM	47.35	22373	**52.06**	**4096 (+9)**
	skglm	**53.53**	**11**	49.41	8 (+9)

The results indicate that the ensemble generally improves test accuracy. For instance, CAIM + LR improved from 55.59% to 58.24%, and Infinitesimal Bins + SVM improved from 47.35% to 52.06%. At the same time, the ensemble often creates more compact scorecards, which is especially noticeable with Logistic Regression and SVM, showing that including the existing scorecard helps capture information more efficiently.

Although the increases in accuracy were modest, the use of ensemble methods resulted in improvements across most tests, highlighting the potential for further exploration of ensemble techniques with scorecards.

5.5.3 RiskSLIM

Since the free version of CPLEX has limitations on the number of variables (determined by the dataset dimension), RiskSLIM could not be integrated and used for comparison in all experiments. However, some tests were conducted on smaller datasets. The target variable in the `aesthetic` dataset was defined by three different authors. To create a smaller dataset, the examples from the author with the fewest cases ("cmavioso") were selected. Both the ordinal dataset and its binary version were tested, using the existing scorecard intervals and CAIM for the discretisation and Differential Coding for the encoding. The classifiers tested in addition to RiskSLIM were logistic regression, SVM, and sklgm. The results for the ordinal dataset are shown in Table 5.12 and for the binary version in Table 5.13.

The RiskSLIM methodology followed the formulation described in Section 4.5.5. For the binary classification task, both the standard threshold of 0.5 and the threshold that achieved the best performance on the training set were evaluated. Table 5.12 and Table 5.13 report the performance for both thresholds.

When applying RiskSLIM with the CAIM on the ordinal dataset, it was not possible to run the model with the default settings due to CPLEX size limitations. So the maximum model size was restricted to the largest value that CPLEX could handle, which was 42. For the Infinitesimal Bins discretisation, despite trying to run the model with a maximum model size of 5 and a maximum coefficient value of 1 for both the ordinal and binary datasets, the solver still had size limitations, so it was not possible to test for this discretisation method.

In both the binary and ordinal datasets, RiskSLIM always has a lower performance compared to the machine learning models. In the binary case, the difference is small, just around 2% lower, making RiskSLIM almost comparable in accuracy while retaining interpretability. In the ordinal case, the best threshold produces about 5% lower accuracy compared to skglm. Nevertheless, these results suggest that the adaptation of RiskSLIM to ordinal data is capable of producing feasible and competitive models.

5.6 Conclusions

The experimental results provide several insights into the impact of discretisation, encoding, and classifier alternatives on the performance of scorecards.

Table 5.12: Results on the `aesthetic` **ordinal** dataset with `author = cmavioso` for different discretisation methods and classifiers, including RiskSLIM. The results include test accuracy, train accuracy and model size. For RiskSLIM, two decision thresholds are shown in brackets: 0.50 (default) and the one with the best training performance. For CAIM with RiskSLIM, the maximum model size was restricted to 42 (due to CPLEX limits). The encoding technique used was Differential Coding.

Discretisation	Classifier	Train Acc. (%)	Test Acc. (%)	Model Size
Existing Scorecard	Logistic Regression	59.63	57.06	11
	SVM	59.53	57.06	11
	skglm	56.09	51.76	9
	RiskSLIM (0.50)	25.54	28.24	11
	RiskSLIM (0.12)	51.57	45.59	11
CAIM	Logistic Regression	80.23	60.00	83
	SVM	80.23	65.00	105
	skglm	66.10	**65.00**	**10**
	RiskSLIM (0.50)	0.00	0.00	25
	RiskSLIM (0.41)	54.42	45.59	25

Table 5.13: Results on the `aesthetic` **binary** dataset with `author = cmavioso` for different discretisation methods and classifiers, including RiskSLIM. The results include test accuracy, train accuracy and model size. For RiskSLIM, two decision thresholds are shown in brackets: 0.50 (default) and the one with the best training performance. The encoding technique used was Differential Coding.

Discretisation	Classifier	Train Acc. (%)	Test Acc. (%)	Model Size
Existing Scorecard	Logistic Regression	82.02	79.71	9
	SVM	82.02	79.71	8
	skglm	81.83	**79.71**	**6**
	RiskSLIM (0.50)	78.88	77.65	9
	RiskSLIM (0.44)	79.08	77.94	9
CAIM	Logistic Regression	92.66	78.33	23
	SVM	88.70	80.00	64
	skglm	90.96	78.33	11
	RiskSLIM (0.50)	83.01	**81.76**	**31**
	RiskSLIM (0.57)	83.79	79.71	31

In terms of discretisation, CAIM produced smaller models than Infinitesimal Bins, sometimes by several orders of magnitude, while maintaining similar or better accuracy. This makes CAIM the better choice for more interpretable models. However, Infinitesimal Bins still shows potential, especially in smaller datasets. Future work should focus on refining Infinitesimal Bins by, for example, merging redundant bins.

When it comes to encoding, One-hot Encoding combined with Infinitesimal Bins resulted in overfitting. In contrast, Differential Coding, besides reducing overfitting, produces more compact models, making it the preferred encoding strategy. With CAIM, both encoding methods performed similarly in terms of accuracy, although Differential Coding sometimes resulted in smaller models.

Regarding classifiers, for binary classification problems, skglm was considered the best option because it consistently produced the sparsest models with minimal or no loss in accuracy, representing the best balance between performance and interpretability. For ordinal problems, skglm produced the most compact models, but always with a significant loss in accuracy. While Logistic Regression and SVM produced high-accuracy models, they were less sparse. Adaptive Lasso showed mixed results: it sometimes achieved sparse and accurate models, but in most datasets, it used all coefficients, undermining its intended goal of promoting sparsity. As such, for ordinal problems, there is no clear better option.

Additionally, manual feature selection on the `aesthetic` dataset demonstrated that feature selection enhances both accuracy and interpretability. Ensemble methods provided modest but consistent improvements in accuracy, indicating their potential. Finally, RiskSLIM experiments confirmed the viability of optimisation-based approaches for interpretable scorecards and that it can be adapted to ordinal classification using the Single Binary Classifier technique.

Chapter 6

Conclusions and Future Work

This dissertation investigated the construction of interpretable scorecards, with a particular focus on ordinal scorecards, which is an underexplored classification task in the literature. The study compared two discretisation methods, two encoding techniques, and five classifiers, using both machine learning and optimisation-based approaches. In particular, RiskSLIM, which had previously been extended only to multiclass classification without accounting for the ordinal structure, was adapted here to ordinal tasks. A fundamental contribution was the proposal of Infinitesimal Bins, a discretisation method that creates bins that correspond directly to data values, therefore approximating scorecard construction as a single-step process. Additional exploration with ensemble scorecards and feature selection was performed because of the potential to improve interpretability and predictive performance.

The experimental evaluation demonstrates that Infinitesimal Bins, while effective for small datasets, often produce overly large models on larger datasets unless combined with strong sparsity-inducing classifiers. Differential Coding consistently outperformed One-Hot Encoding across all settings, confirming its suitability for both binary and ordinal scorecards. For binary problems, the skglm classifier offered the best balance between model size and predictive power. In ordinal problems, smaller models achieved lower accuracy while larger models achieved higher accuracy, indicating that the choice of the classifier should be made according to the application's preferences. Finally, ensemble methods and feature selection resulted in better performance, and RiskSLIM was successfully extended to ordinal classification, although its accuracy does not match that of machine learning classifiers.

Future research could explore several potential options, including the following:

- **Feature selection prior to modelling**. Manual feature selection on the `aesthetic` dataset demonstrated performance improvements in various scorecard configurations. This suggests that integrating an automated feature selection method, or Random Forest for feature ranking, could be beneficial, as demonstrated by AutoScore [69]. In addition, it would enhance interpretability by potentially reducing the number of scorecard columns.
- **Iterative reweighting**. Inspired by ICS [7], iterative reweighting could be integrated into the scorecard learning process. It would be especially beneficial with the Infinitesimal Bins

discretisation technique, since it produces a lot of columns and with high specificity. Tuning the weights would improve sparsity and stability.

- **Try FasterRisk**. Since FasterRisk [39] does not require mathematical programming tools, there is no need for a license. Using it would allow for a direct comparison with state-of-the-art optimisation-based scoring systems, particularly in large datasets.
- **Handle class imbalance**. Although balanced accuracy was evaluated, the imbalance in the data was not addressed during training. In the future, imbalance techniques could be employed.
- **Ensemble of scorecards**. The exploratory results for the ensemble of scorecards suggest that the combination of multiple scorecards could improve robustness and accuracy while still being intuitive. This could be further explored.
- **Interactive interface**. Although the developed scorecard code was implemented as a reusable class for all datasets, its practical use is still limited. An interactive tool could be designed to visualise and customise the learned scorecards. It could allow experts to select features, adjust intervals, and control the trade-off between performance and sparsity.
- **Evaluation criteria besides accuracy**: The optimisation goal could be adapted according to the application. For example, in healthcare, increasing the F1 score (reducing false positives) may be more crucial than maximising overall accuracy.

References

[1] Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

[2] Virginia Apgar. A proposal for a new method of evaluation of the newborn infant. *Current Researches in Anesthesia and Analgesia*, 32(4):250–259, 1953.

[3] Luis Barroso-Luque and Fengyu Xie. sparse-lm: Sparse linear regression models in python. *Journal of Open Source Software*, 8(92):5867, 2023.

[4] M. A. Barry, K. Qu, M. Gainey, C. H. Schmid, S. C. Garbern, S. Nasrin, N. H. Alam, J. A. Lee, E. J. Nelson, R. Rosen, and A. C. Levine. Derivation and internal validation of a score to predict dehydration severity in patients over 5 years with acute diarrhea. *The American Journal of Tropical Medicine and Hygiene*, 105(5):1368–1375, 2021.

[5] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[6] Q. Bertrand, Q. Klopfenstein, P.-A. Bannier, G. Gidel, and M. Massias. Beyond l1: Faster and better sparse models with skglm. In *NeurIPS*, 2022.

[7] Lieven Billiet, Sabine Van Huffel, and Vanya Van Belle. Interval coded scoring: A toolbox for interpretable scoring systems. *PeerJ Computer Science*, 2018, 2018.

[8] Jaime S. Cardoso and Maria João Cardoso. Towards an intelligent medical system for the aesthetic evaluation of breast cancer conservative treatment. *Artificial Intelligence in Medicine*, 40(2):115–126, 2007.

[9] Jaime S. Cardoso and Joaquim F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *Journal of Machine Learning Research*, 8:1393–1429, July 2007.

[10] Yifeng Chen, Xiaoyu Cai, Zicheng Cao, Jie Lin, Wenyu Huang, Yuan Zhuang, Lehan Xiao, Xiaozhen Guan, Ying Wang, Xingqiu Xia, Feng Jiao, Xiangjun Du, Guozhi Jiang, and Deqing Wang. Prediction of red blood cell transfusion after orthopedic surgery using an interpretable machine learning framework. *Frontiers in Surgery*, Volume 10 - 2023, 2023.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[12] Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Wine Quality. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C56S3T.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[14] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017.

[15] Carole Dufouil, Alexa Beiser, Leslie A. McClure, Philip A. Wolf, Christophe Tzourio, Virginia J. Howard, et al. Revised framingham stroke risk profile to reflect temporal trends. *Circulation*, 135(12):1145–1159, 2017.

[16] Matthias Elter. Mammographic Mass. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C53K6Z.

[17] D. Ferrari, V. Guidetti, Y. Wang, and V. Curcin. Multi-objective symbolic regression to generate data-driven, non-fixed structure and intelligible mortality predictors using ehr: Binary classification methodology and comparison with state-of-the-art. In *AMIA ... Annual Symposium Proceedings. AMIA Symposium*, pages 442–451, 2022.

[18] Davide Ferrari, Veronica Guidetti, and Federica Mandreoli. Multi-objective symbolic regression for data-driven scoring system management. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 945–950, 2022.

[19] Simon Föll, Adrian Lison, Martin Maritsch, Karsten Klingberg, Vera Lehmann, Thomas Züger, David Srivastava, Sabrina Jegerlehner, Stefan Feuerriegel, Elgar Fleisch, Aristomenis Exadaktylos, and Felix Wortmann. A scalable risk-scoring system based on consumer-grade wearables for inpatients with covid-19: Statistical analysis and model development. *JMIR Form Res*, 6(6):e35717, Jun 2022.

[20] Brian F. Gage, Amy D. Waterman, William Shannon, Michael Boechler, Michael W. Rich, and Martha J. Radford. Validation of clinical classification schemes for predicting stroke. *Journal of the American Medical Association*, 285(22):2864–2870, 2001.

[21] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (saps ii) based on a european/north american multicenter study. *The Journal of the American Medical Association*, 270(24):2957–2963, 1993.

[22] Tiago Gonçalves, Anna Hedström, Aurélie Pahud de Mortanges, Xiaoxiao Li, Henning Müller, Jaime S. Cardoso, and Mauricio Reyes. *Trustworthy AI for Medical Imaging*, chapter Interpretable AI for Medical Image Analysis: Methods, Evaluation and Clinical Considerations (under preparation). MICCAI/Elsevier, 2025.

[23] Michal K. Grzeszczyk, Tomasz Trzciński, and Arkadiusz Sitek. Miss: Multiclass interpretable scoring systems. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM 2024)*, pages 55–63, 2024. Houston, 18-20 April 2024.

[24] Veronica Guidetti. Symbolic regression for transparent clinical decision support: A data-centric framework for scoring system development. In *CEUR Workshop Proceedings*, 2024.

[25] Jay R. Harris, Martin B. Levene, Goran Svensson, and Samuel Hellman. Analysis of cosmetic results following primary radiation therapy for stages i and ii carcinoma of the breast. *International Journal of Radiation Oncology, Biology, Physics*, 5:257–261, 1979.

[26] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1:297–310, 1986.

[27] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to operations research / Frederick S. Hillier, Gerald J. Lieberman*. McGraw Hill, New York, 7th ed edition, 2001.

[28] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[29] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C53G6X.

[30] Yu Ji, Kaipeng Wang, Yuan Yuan, Yueguo Wang, Qingyuan Liu, Yulan Wang, Jian Sun, Wenwen Wang, Huanli Wang, Shusheng Zhou, Kui Jin, Mengping Zhang, and Yinglei Lai. A knowledge-transfer-based approach for combining ordinal regression and medical scoring system in the early prediction of sepsis with electronic health records. *Computational Biology and Chemistry*, 113:108203, 2024.

[31] Orit Kaidar-Person, Marilia Antunes, Jaime S. Cardoso, Oriana Ciani, Helena Cruz, Rosa Di Micco, Oreste D. Gentilini, Tiago Gonçalves, Pedro Gouveia, Jörg Heil, Pawel Kabata, Daniela Lopes, Marta Martinho, and on behalf of the CINDERELLA Consortium. Evaluating the ability of an artificial-intelligence cloud-based platform designed to provide information prior to locoregional therapy for breast cancer in improving patient's satisfaction with therapy: The cinderella trial. *PLOS ONE*, 18(8):e0289365, 2023.

[32] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository. https://archive.ics.uci.edu, 2024. Accessed: 2025-07-04.

[33] William A. Knaus, Elizabeth A. Draper, Douglas P. Wagner, and Jack E. Zimmerman. Apache ii: a severity of disease classification system. *Critical Care Medicine*, 13(10):818–829, 1985.

[34] William A. Knaus, D.P. Wagner, E.A. Draper, J.E. Zimmerman, Marilyn Bergner, P.G. Bastos, C.A. Sirio, D.J. Murphy, T. Lotring, and A. Damiano. The apache iii prognostic system: risk prediction of hospital mortality for critically ill hospitalized adults. *Chest Journal*, 100(6):1619–1636, 1991.

[35] William A. Knaus, Jack E. Zimmerman, Douglas P. Wagner, Elizabeth A. Draper, and Diane E. Lawrence. Apache – acute physiology and chronic health evaluation: A physiologically based classification system. *Critical Care Medicine*, 9(8):591–597, 1981.

[36] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.

[37] L. A. Kurgan and K. J. Cios. Caim discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.

[38] L. A. Kurgan and K. J. Cios. Caim discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, Feb. 2004.

[39] Jiachang Liu, Chudi Zhong, Boxuan Li, Margo Seltzer, and Cynthia Rudin. Fasterrisk: Fast and accurate interpretable risk scores. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17760–17773. Curran Associates, Inc., 2022.

[40] Philipp G. H. Metnitz, Rui P. Moreno, Eduardo Almeida, Barbara Jordan, Peter Bauer, Ricardo Abizanda Campos, Gaetano Iapichino, David Edbrooke, Maurizia Capuzzo, Jean-Roger Le Gall, and on behalf of the SAPS 3 Investigators. Saps 3—from evaluation of the patient to evaluation of the intensive care unit. part 1: Objectives, methods and cohort description. *Intensive Care Medicine*, 31(10):1336–1344, 2005.

[41] Rui P. Moreno, Philipp G.H. Metnitz, Eduardo Almeida, Barbara Jordan, Peter Bauer, Ricardo Abizanda Campos, Gaetano Iapichino, David Edbrooke, Maurizia Capuzzo, and Jean-Roger Le Gall. SAPS 3 - from evaluation of the patient to evaluation of the intensive care unit. part 2: Development of a prognostic model for hospital mortality at icu admission. *Intensive Care Medicine*, 31(10):1345–1355, 2005.

[42] Badr Moufad, Pierre-Antoine Bannier, Quentin Bertrand, Quentin Klopfenstein, and Mathurin Massias. skglm: improving scikit-learn for regularized generalized linear models. 2023.

[43] F. Nachbar, W. Stolz, T. Merkle, A. B. Cognetta, T. Vogt, M. Landthaler, P. Bilek, O. Braun-Falco, and G. Plewig. The abcd rule of dermatoscopy. high prospective value in the diagnosis of doubtful melanocytic skin lesions. *Journal of the American Academy of Dermatology*, 30(4):551–559, 1994.

[44] Warwick Nash, Tracy Sellers, Simon Talbot, Andrew Cawthorn, and Wes Ford. Abalone. UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C55C7W.

[45] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Royal Statistical Society. Journal. Series A: General*, 135(3):370–384, 1972.

[46] A. Orfanoudaki, E. Chesley, C. Cadisch, B. Stein, A. Nouh, M. J. Alberts, et al. Machine learning provides evidence that stroke risk is not linear: The non-linear framingham stroke risk score. *PLoS ONE*, 15(5):e0232414, 2020.

[47] Andrzej Pajor, Jakub Żołnierek, Bartosz Śnieżyński, and Anna Sitek. Effect of feature discretization on classification performance of explainable scoring-based machine learning model. In Derek Groen, Clement de Mulatier, Maciej Paszyński, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2022*, volume 13352 of *Lecture Notes in Computer Science*, pages 112–125. Springer, Cham, 2022.

[48] Diem Pham, Binh Tran, Su Nguyen, and Damminda Alahakoon. Swarm-based machine learning algorithm for building interpretable classifiers. *IEEE Access*, 8:228136–228150, 2020.

[49] Julien Rouzot, Julien Ferry, and Marie-Jose Huguet. Learning optimal fair scoring systems for multi-class classification. In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, pages 197–204, 2023.

[50] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, 2019.

[51] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges, 2021.

[52] Cynthia Rudin and Berk Ustun. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5), 2018.

[53] Cynthia Rudin and Berk Ustun. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, 48(5):449–466, 2018.

[54] Jörg Sabczynski, Hans Barschdorf, Thomas Bülow, Maria Cardoso, Jaime Cardoso, Alex Clarke, Bjoern Eiben, Pedro Gouveia, David Hawkes, John Hipwell, Mohammed Keshtgar, Rene Lacher, Dominik Kutra, Gerrit-Jan Liefers, Kirsten Meetz, Barbara Molenkamp, João Monteiro, Ash Mosahebi, Helder Oliveira, and Hooshiar Zolfagharnasab. Picture: Predicting the cosmetic outcome of breast cancer surgery. 09 2014.

[55] SAS Institute. Building credit scoring models with sas enterprise miner. Technical report, SAS Institute, 2006. Technical report.

[56] Pedro F. B. Silva and Jaime S. Cardoso. Differential scorecards for binary and ordinal data. *Intelligent Data Analysis*, 19:1391–1408, 2015.

[57] Mervyn Singer, Clifford S. Deutschman, Christopher Warren Seymour, Manu Shankar-Hari, Djillali Annane, Michael Bauer, Rinaldo Bellomo, Gordon R. Bernard, Jean-Daniel Chiche, Craig M. Coopersmith, Richard S. Hotchkiss, Mitchell M. Levy, John C. Marshall, Greg S. Martin, Steven M. Opal, Gordon D. Rubenfeld, Tom van der Poll, Jean-Louis Vincent, and Derek C. Angus. The third international consensus definitions for sepsis and septic shock (sepsis-3). *JAMA*, 315(8):801–810, 2016.

[58] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[59] Margaret Tobey, Ruoting Li, Osman Y. Özaltın, Maria E. Mayorga, and Sherrie Caltagirone. Interpretable models for the automated detection of human trafficking in illicit massage businesses. *IISE Transactions*, 56(3):311–324, 2024.

[60] Bureau of Justice Statistics U.S. Department of Justice. Recidivism of prisoners released in 1994, 2014.

[61] Bedirhan Ustun, Lenard A. Adler, Cynthia Rudin, Stephen V. Faraone, Thomas J. Spencer, Patricia Berglund, Michael J. Gruber, and Ronald C. Kessler. The world health organization adult attention-deficit/hyperactivity disorder self-report screening scale for dsm-5. *JAMA Psychiatry*, 74(5):520–527, 2017.

[62] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102:349–391, 2016.

[63] Berk Ustun and Cynthia Rudin. Learning optimized risk scores. *Journal of Machine Learning Research*, 20:1–75, 2019.

[64] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the International Workshop on Software Fairness*, FairWare '18, page 1–7, New York, NY, USA, 2018. Association for Computing Machinery.

[65] Jean-Louis Vincent, Rui Moreno, Jukka Takala, Steve Willatts, Arnaldo De Mendonça, Hugues Bruining, Charles K. Reinhart, Peter M Suter, and Luciano G. Thijs. The sofa (sepsis-related organ failure assessment) score to describe organ dysfunction/failure. *Intensive Care Medicine*, 22(7):707–710, 1996.

[66] S. M. Weiss and C. A. Kulikowski. Appendicitis dataset, 1991. Proposed in *Computer Systems That Learn*.

[67] WIlliam Wolberg. Breast Cancer Wisconsin (Original). UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5HP4Z.

[68] Philip A. Wolf, Ralph B. D'Agostino, Albert J. Belanger, and William B. Kannel. Probability of stroke: A risk profile from the framingham study. *Stroke*, 22(3):312–318, 1991.

[69] F. Xie, B. Chakraborty, M.E.H. Ong, B.A. Goldstein, and N. Liu. Autoscore: A machine learning-based automatic clinical score generator and its application to mortality prediction using electronic health records. *JMIR Medical Informatics*, 8(10):e21798, 2020.

[70] Eric Yamga, Sreekar Mantena, Darin Rosen, Emily M. Bucholz, Robert W. Yeh, Leo A. Celi, Berk Ustun, and Neel M. Butala. Optimized risk score to predict mortality in patients with cardiogenic shock in the cardiac intensive care unit. *Journal of the American Heart Association*, 12(13):e029232, 2023.

[71] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A

Performance Metrics for each Dataset

This appendix reports the performance results obtained for each dataset across all discretisation, encoding, and classifier combinations. For each combination, the following metrics are recorded: train accuracy, test accuracy, test balanced accuracy, mean squared error (MSE), model size, and execution time. Model size is given as a percentage of total features (non-zero) after encoding compared to before encoding, along with the absolute number of non-zero coefficients in parentheses.

For both `aesthetic binary` and `aesthetic ordinal` datasets, three tables are presented: the first contains results for the whole dataset (Tables A.4 and A.10); the second includes results for the dataset with manually selected features (Tables A.5 and A.11); and the third contains results for the dataset with manually selected features and restricted to samples with `author = cmavioso` (Tables A.6 and A.12). The `aesthetic ordinal` dataset includes an additional table presenting the results of the ensemble of scorecards (Table A.13).

Table A.1: Performance results of `appendicitis` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	91.14	81.48	64.29	0.185	42.86 (9)	0.022
		SVM	90.03	85.19	72.62	0.148	90.48 (19)	0.022
		skglm	91.14	81.48	64.29	0.185	28.57 (6)	0.020
		Adaptive Lasso	89.87	81.48	64.29	0.185	100.00 (21)	0.032
	Differential Coding	Logistic Regression	91.14	81.48	64.29	0.185	66.67 (14)	0.026
		SVM	88.61	85.19	72.62	0.148	66.67 (14)	0.022
		skglm	89.87	85.19	66.67	0.148	23.81 (5)	0.021
		Adaptive Lasso	89.87	81.48	64.29	0.185	100.00 (21)	0.031
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	81.48	58.33	0.185	99.75 (404)	0.212
		SVM	100.00	81.48	58.33	0.185	98.27 (398)	0.047
		skglm	99.68	81.48	64.29	0.185	3.46 (14)	0.010
		Adaptive Lasso	100.00	77.78	55.95	0.222	100.00 (405)	0.080
	Differential Coding	Logistic Regression	93.67	81.48	64.29	0.185	9.14 (37)	0.290
		SVM	92.41	81.48	64.29	0.185	88.64 (359)	0.030
		skglm	98.73	70.37	57.14	0.296	3.21 (13)	0.047
		Adaptive Lasso	100.00	81.48	64.29	0.185	100.00 (405)	0.053

Table A.2: Performance results of `mammo` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	80.23	84.62	84.76	0.154	100.00 (15)	0.018
		SVM	79.58	84.62	84.78	0.154	100 (15)	0.013
		skglm	79.90	84.13	84.29	0.159	46.67 (7)	0.016
		Adaptive Lasso	80.23	84.13	84.29	0.159	100.00 (15)	0.027
	Differential Coding	Logistic Regression	80.23	84.62	84.76	0.154	40.00 (6)	0.012
		SVM	79.90	84.62	84.78	0.154	60.00 (9)	0.016
		skglm	80.06	85.10	85.25	0.149	33.33 (5)	0.008
		Adaptive Lasso	79.74	84.13	84.29	0.159	26.67 (4)	0.032
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	85.69	84.62	84.54	0.154	89.15 (74)	0.041
		SVM	83.60	84.13	84.01	0.159	90.36 (75)	0.014
		skglm	81.83	82.69	82.54	0.173	6.02 (5)	0.007
		Adaptive Lasso	86.01	83.17	83.05	0.168	100.00 (83)	0.034
	Differential Coding	Logistic Regression	84.89	84.62	84.52	0.154	25.30 (21)	0.025
		SVM	84.08	82.69	82.63	0.173	97.59 (81)	0.020
		skglm	85.05	84.62	84.52	0.154	9.64 (8)	0.012
		Adaptive Lasso	84.89	84.62	84.52	0.154	12.05 (10)	0.150

Table A.3: Performance results of `breast cancer` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	97.85	95.32	94.37	0.047	88.89 (24)	0.041
		SVM	97.46	97.66	97.31	0.023	92.59 (25)	0.014
		skglm	97.07	95.91	95.35	0.041	51.85 (14)	0.012
		Adaptive Lasso	97.27	94.74	93.88	0.053	100.00 (27)	0.038
	Differential Coding	Logistic Regression	97.85	94.74	93.63	0.053	62.96 (17)	0.068
		SVM	97.66	95.91	95.10	0.041	96.30 (26)	0.027
		skglm	97.07	97.08	96.82	0.029	29.63 (8)	0.012
		Adaptive Lasso	97.27	94.74	93.88	0.053	100.00 (27)	0.023
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	98.83	94.74	93.63	0.053	82.50 (66)	0.036
		SVM	97.27	94.74	93.63	0.053	98.75 (79)	0.008
		skglm	95.90	90.06	87.75	0.099	27.50 (22)	0.009
		Adaptive Lasso	98.63	94.15	92.90	0.058	100.00 (80)	0.019
	Differential Coding	Logistic Regression	98.83	94.74	93.63	0.053	62.50 (50)	0.207
		SVM	97.27	95.91	95.10	0.041	100.00 (80)	0.012
		skglm	97.27	95.32	94.37	0.047	7.50 (6)	0.005
		Adaptive Lasso	97.46	94.74	93.63	0.053	15.00 (12)	0.048

Table A.4: Performance results of `aesthetic binary` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	83.69	81.41	76.60	0.186	100.00 (66)	0.938
		SVM	82.91	80.29	75.12	0.197	81.82 (54)	1.134
		skglm	83.01	80.00	74.69	0.200	18.18 (12)	0.993
		Adaptive Lasso	84.09	81.18	76.21	0.188	100.00 (66)	1.143
	Differential Coding	Logistic Regression	83.89	81.47	76.65	0.185	40.91 (27)	1.167
		SVM	83.01	80.59	75.56	0.194	80.30 (53)	1.139
		skglm	82.71	81.18	76.00	0.188	15.15 (10)	0.972
		Adaptive Lasso	84.09	81.47	76.43	0.185	100.00 (66)	1.160
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	70.59	58.22	0.294	100.00 (22374)	107.784
		SVM	100.00	72.35	60.83	0.276	100.00 (22374)	4.455
		skglm	66.60	66.18	50.00	0.338	0.00 (0)	0.270
		Adaptive Lasso	100.00	72.65	61.48	0.274	66.58 (14897)	45.026
	Differential Coding	Logistic Regression	99.31	77.94	73.34	0.221	12.53 (2802)	675.322
		SVM	88.80	80.88	75.57	0.191	99.26 (22208)	3.223
		skglm	84.97	80.88	75.57	0.191	0.07 (16)	1.053
		Adaptive Lasso	100.00	72.06	69.32	0.279	100.00 (22374)	70.996

Table A.5: Performance results of `aesthetic binary` with **features selected manually** for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	82.12	79.06	75.25	0.209	100.00 (12)	0.174
		SVM	82.02	78.24	73.78	0.218	100.00 (12)	0.297
		skglm	81.93	78.82	74.86	0.212	41.67 (5)	0.177
		Adaptive Lasso	82.12	78.82	75.07	0.212	100.00 (12)	0.278
	Differential Coding	Logistic Regression	82.12	79.00	75.21	0.210	100.00 (12)	0.228
		SVM	82.12	78.24	73.78	0.218	66.67 (8)	0.262
		skglm	82.02	79.12	75.29	0.209	50.00 (6)	0.177
		Adaptive Lasso	82.12	78.82	75.07	0.212	100.00 (12)	0.291
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	69.41	57.33	0.306	100.00 (4068)	14.993
		SVM	100.00	69.71	57.56	0.303	100.00 (4068)	1.162
		skglm	66.90	66.18	50.00	0.338	0.07 (3)	0.053
		Adaptive Lasso	100.00	70.29	64.16	0.297	100.00 (4068)	1.290
	Differential Coding	Logistic Regression	89.29	78.71	73.71	0.213	5.31 (216)	0.363
		SVM	83.10	80.00	74.26	0.200	98.75 (4017)	0.668
		skglm	83.01	81.47	75.58	0.185	0.17 (7)	0.335
		Adaptive Lasso	100.00	72.35	68.27	0.276	100.00 (4068)	4.269

Table A.6: Performance results of `aesthetic binary` with **features selected manually and** `author = cmavioso` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	88.70	81.67	65.56	0.183	100.00 (12)	0.028
		SVM	89.27	81.67	67.78	0.183	100.00 (12)	0.029
		skglm	88.70	81.67	65.56	0.183	75.00 (9)	0.029
		Adaptive Lasso	88.70	81.67	65.56	0.183	100.00 (12)	0.042
	Differential Coding	Logistic Regression	88.70	81.67	65.56	0.183	100.00 (12)	0.037
		SVM	88.14	81.67	65.56	0.183	100.00 (12)	0.027
		skglm	88.70	81.67	65.56	0.183	75.00 (9)	0.030
		Adaptive Lasso	88.70	81.67	65.56	0.183	100.00 (12)	0.034
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	76.67	53.33	0.233	100.00 (704)	0.436
		SVM	81.36	75.00	50.00	0.250	93.18 (656)	0.026
		skglm	81.36	75.00	50.00	0.250	0.00 (0)	0.007
		Adaptive Lasso	100.00	75.00	54.44	0.250	100.00 (704)	0.102
	Differential Coding	Logistic Regression	96.05	81.67	67.78	0.183	6.11 (43)	0.053
		SVM	95.48	81.67	67.78	0.183	99.29 (699)	0.052
		skglm	90.96	80.00	71.11	0.200	1.99 (14)	0.060
		Adaptive Lasso	91.53	80.00	68.89	0.200	1.28 (9)	1.088

Table A.7: Performance results of `spambase` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	94.12	93.83	93.25	0.062	95.32 (163)	6.496
		SVM	94.23	93.83	93.21	0.062	76.61 (131)	1.217
		skglm	91.62	92.09	91.11	0.079	7.60 (13)	2.899
		Adaptive Lasso	93.19	92.79	91.89	0.072	100.00 (171)	1.182
	Differential Coding	Logistic Regression	94.12	94.01	93.46	0.060	73.10 (125)	2.061
		SVM	94.03	93.83	93.25	0.062	76.61 (131)	1.400
		skglm	91.88	92.53	91.57	0.075	8.19 (14)	1.756
		Adaptive Lasso	93.19	92.79	91.89	0.072	100.00 (171)	1.237
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	99.94	87.26	86.00	0.127	97.55 (12986)	571.928
		SVM	98.93	88.62	87.40	0.114	89.98 (11979)	28.830
		skglm	61.22	58.73	50.00	0.413	0.02 (3)	0.762
		Adaptive Lasso	99.94	83.32	82.54	0.167	100.00 (13313)	19.770
	Differential Coding	Logistic Regression	97.82	95.01	94.63	0.050	5.13 (683)	1.315
		SVM	96.43	96.00	95.69	0.40	93.59 (12460)	7.495
		skglm	93.07	93.57	92.84	0.064	0.11 (14)	1.682
		Adaptive Lasso	99.94	85.58	85.50	0.144	100.00 (13313)	65.374

Table A.8: Performance results of `income` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	82.43	82.73	70.42	0.173	22.41 (26)	5.328
		SVM	81.18	81.36	64.49	0.186	18.97 (22)	10.547
		skglm	82.10	82.48	69.61	0.175	4.31 (5)	5.525
		Adaptive Lasso	82.42	82.77	70.59	0.172	22.41 (26)	5.536
	Differential Coding	Logistic Regression	82.43	82.73	70.42	0.173	15.74 (17)	5.369
		SVM	81.17	81.36	64.49	0.186	15.74 (17)	9.631
		skglm	82.25	82.68	70.26	0.173	4.63 (5)	5.374
		Adaptive Lasso	82.42	82.77	70.59	0.172	16.67 (18)	5.565
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	85.31	84.14	72.12	0.159	3.58 (604)	8.505
		SVM	87.80	83.60	69.79	0.164	53.19 (8980)	2602.464
		skglm	77.12	77.56	60.51	0.224	0.04 (6)	9.688
		Adaptive Lasso	76.11	76.50	57.38	0.235	0.07 (11)	30.773
	Differential Coding	Logistic Regression	85.50	83.76	72.26	0.162	22.48 (3793)	61.106
		SVM	82.47	82.77	67.02	0.172	99.46 (16783)	1338.413
		skglm	82.22	82.63	69.02	0.174	0.05 (8)	20.081
		Adaptive Lasso	96.26	76.40	68.28	0.236	99.47 (16784)	13369.422

Table A.9: Performance results of `small wine` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	97.74	88.89	88.89	0.111	56.60 (30)	0.088
		SVM	96.99	91.11	90.74	0.089	92.45 (49)	0.168
		skglm	93.23	88.89	90.74	0.111	20.75 (11)	1.715
		Adaptive Lasso	96.99	91.11	91.67	0.089	100.00 (53)	0.141
	Differential Coding	Logistic Regression	98.50	93.33	93.52	0.067	84.91 (45)	0.177
		SVM	96.99	95.56	96.30	0.044	90.57 (48)	0.173
		skglm	94.74	95.56	96.30	0.044	18.87 (10)	0.074
		Adaptive Lasso	94.74	97.78	98.15	0.022	43.40 (23)	0.181
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	64.44	60.37	0.356	96.56 (1012)	1.526
		SVM	99.25	64.44	60.37	0.356	95.23 (998)	0.045
		skglm	66.92	57.78	55.74	0.689	1.72 (18)	0.042
		Adaptive Lasso	100.00	73.33	70.74	0.267	100.00 (1048)	0.161
	Differential Coding	Logistic Regression	100.00	91.11	92.59	0.089	91.79 (962)	3.190
		SVM	100.00	91.11	91.67	0.089	98.76 (1035)	0.076
		skglm	96.99	88.89	89.81	0.111	1.15 (12)	0.088
		Adaptive Lasso	100.00	91.11	92.59	0.089	100.00 (1048)	0.464

Table A.10: Performance results of `aesthetic ordinal` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	64.44	55.59	48.16	0.462	62.50 (70)	13.749
		SVM	64.54	53.82	48.31	0.497	95.54 (107)	14.405
		skglm	54.13	50.00	38.53	0.571	8.04 (9)	14.128
		Adaptive Lasso	60.02	51.76	41.13	0.553	100.00 (112)	14.282
	Differential Coding	Logistic Regression	64.64	55.59	48.72	0.471	85.00 (95)	13.773
		SVM	64.93	54.41	48.66	0.491	95.54 (107)	13.183
		skglm	55.01	47.35	35.68	0.588	8.04 (9)	15.428
		Adaptive Lasso	60.02	51.47	40.83	0.555	100.00 (112)	21.552
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	45.00	28.89	0.665	100.00 (22376)	385.689
		SVM	100.00	45.29	29.17	0.653	100.00 (22376)	18.233
		skglm	44.60	41.47	25.00	0.814	0.01 (2)	1.427
		Adaptive Lasso	100.00	42.65	26.23	0.741	100.00 (22376)	57.747
	Differential Coding	Logistic Regression	100.00	47.35	45.05	0.729	100.00 (22376)	1156.21
		SVM	99.80	47.35	43.40	0.747	99.99 (22373)	19.460
		skglm	57.27	53.53	42.43	0.544	0.05 (11)	2.847
		Adaptive Lasso	100.00	49.12	42.25	0.685	100.00 (22376)	420.980

Table A.11: Performance results of `aesthetic ordinal` with **features selected manually** for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	60.41	57.53	52.64	0.486	98.18 (21)	2.738
		SVM	59.72	53.82	50.92	0.565	100.00 (22)	2.725
		skglm	54.81	50.00	39.15	0.562	45.45 (10)	3.298
		Adaptive Lasso	54.42	50.59	39.65	0.565	100.00 (22)	2.599
	Differential Coding	Logistic Regression	60.41	57.65	52.72	0.485	68.18 (15)	2.726
		SVM	59.43	53.82	50.92	0.565	100.00 (22)	2.728
		skglm	54.42	50.59	42.81	0.574	45.45 (10)	2.457
		Adaptive Lasso	54.32	50.59	39.65	0.556	100.00 (22)	2.713
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	44.60	42.35	26.46	0.718	55.41 (2255)	2.198
		SVM	100.00	44.12	31.26	0.744	100.00 (4070)	8.000
		skglm	44.60	41.47	25.00	0.815	0.05 (2)	0.211
		Adaptive Lasso	100.00	42.94	29.66	0.721	100.00 (4070)	1.284
	Differential Coding	Logistic Regression	73.28	52.94	48.17	0.582	35.56 (1447)	51.673
		SVM	65.13	49.71	46.38	0.659	99.88 (4065)	5.008
		skglm	53.93	47.65	34.45	0.612	0.22 (9)	0.382
		Adaptive Lasso	100.00	48.24	41.96	0.700	100.00 (4070)	46.905

Table A.12: Performance results of `aesthetic ordinal` with **features selected manually and** `author = cmavioso` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	71.19	65.00	68.85	0.450	73.64 (16)	0.184
		SVM	63.28	61.67	41.95	0.533	100.00 (22)	0.174
		skglm	68.93	61.67	66.82	0.483	59.09 (13)	0.113
		Adaptive Lasso	69.49	61.67	66.46	0.483	100.00 (22)	0.150
	Differential Coding	Logistic Regression	71.19	66.67	69.82	0.433	68.18 (15)	0.195
		SVM	65.54	61.67	65.64	0.533	100.00 (22)	0.161
		skglm	65.54	61.67	43.25	0.483	40.91 (9)	0.118
		Adaptive Lasso	69.49	61.67	66.46	0.483	100.00 (22)	0.194
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	37.33	27.75	0.827	49.83 (351)	0.0388
		SVM	87.57	41.67	29.71	0.883	100.00 (22)	0.062
		skglm	53.11	31.67	25.00	0.783	0.99 (7)	0.041
		Adaptive Lasso	100.00	38.33	28.61	0.817	100.00 (706)	0.111
	Differential Coding	Logistic Regression	75.71	63.33	67.89	0.467	9.26 (65)	0.084
		SVM	81.92	61.67	64.93	0.533	99.58 (703)	0.085
		skglm	63.28	55.00	63.44	0.450	1.27 (9)	0.069
		Adaptive Lasso	68.93	61.67	66.93	0.433	2.27 (16)	3.658

Table A.13: Performance results of `aesthetic ordinal` with the **ensemble** of scorecards for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	59.53	58.24	53.01	0.497	74.07 (20)	2.401
		SVM	59.82	54.12	51.20	0.562	100.00 (27)	2.520
		skglm	51.96	47.94	35.20	0.609	33.33 (9)	2.436
		Adaptive Lasso	57.37	52.35	44.85	0.547	100.00 (27)	2.497
	Differential Coding	Logistic Regression	59.53	58.24	53.11	0.497	70.37 (19)	2.637
		SVM	59.82	54.12	51.20	0.562	96.30 (26)	2.443
		skglm	53.24	51.47	37.19	0.556	33.33 (9)	3.916
		Adaptive Lasso	57.37	52.06	44.56	0.550	100.00 (27)	2.425
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	100.00	49.12	41.44	0.665	99.61 (4090)	53.040
		SVM	100.00	47.35	39.83	0.700	100.00 (4106)	5.221
		skglm	50.10	48.24	32.08	0.676	0.15 (6)	0.239
		Adaptive Lasso	100.00	47.06	39.24	0.694	100.00 (4106)	1.403
	Differential Coding	Logistic Regression	98.72	50.59	45.70	0.665	100.00 (4106)	161.298
		SVM	68.57	52.06	46.67	0.635	99.76 (4096)	5.074
		skglm	52.95	49.41	33.72	0.594	0.19 (8)	0.577
		Adaptive Lasso	100.00	48.53	43.86	0.688	100.00 (4106)	53.275

Table A.14: Performance results of `abalone` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	68.44	67.75	38.55	0.399	95.83 (46)	2.890
		SVM	67.75	66.60	39.15	0.426	95.83 (46)	3.201
		skglm	60.66	61.34	20.00	0.618	14.58	2.729
		Adaptive Lasso	61.49	61.82	20.75	0.586	95.83 (46)	2.938
	Differential Coding	Logistic Regression	68.39	67.94	39.24	0.397	93.62 (44)	3.466
		SVM	68.71	67.56	37.95	0.401	80.85 (38)	3.907
		skglm	60.66	61.34	20.00	0.618	14.89 (7)	2.827
		Adaptive Lasso	61.49	61.82	20.75	0.586	95.74 (45)	2.921
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	79.47	65.55	35.57	0.435	24.40 (1310)	39.480
		SVM	79.63	65.74	35.61	0.436	93.63 (5027)	71.699
		skglm	60.66	61.34	20.00	0.618	0.07 (4)	1.062
		Adaptive Lasso	63.12	62.39	21.42	0.553	100.00 (5369)	4.847
	Differential Coding	Logistic Regression	80.91	68.77	47.59	0.357	18.88 (1013)	17.306
		SVM	76.44	69.57	46.58	0.364	99.81 (5358)	...
		skglm	61.56	62.11	20.73	0.576	0.15 (8)	1.618
		Adaptive Lasso	90.36	58.76	29.57	0.589	99.96 (5366)	484.702

Table A.15: Performance results of `large wine` for different discretisation, encoding and classifier combinations.

Discretisation	Encoding	Classifier	Train Acc. (%)	Test Acc. (%)	Balanced Acc. (%)	MSE	Model Size (%)	Exec. Time (s)
CAIM	One-Hot Encoding	Logistic Regression	52.94	51.94	20.03	0.642	88.17 (82)	2.881
		SVM	51.21	50.03	19.61	0.742	100.00 (93)	14.179
		skglm	43.06	45.42	14.29	0.780	7.53 (7)	2.580
		Adaptive Lasso	45.14	46.71	15.00	0.756	100.00 (93)	2.076
	Differential Coding	Logistic Regression	52.74	51.78	20.00	0.642	83.01 (77)	5.291
		SVM	51.21	50.03	19.61	0.742	98.92 (92)	12.957
		skglm	43.06	45.42	14.29	0.780	7.53 (7)	1.901
		Adaptive Lasso	45.14	46.71	15.00	0.756	100.00 (93)	2.707
Infinitesimal Bins	One-Hot Encoding	Logistic Regression	68.25	53.66	24.61	0.616	76.60 (1898)	139.851
		SVM	67.96	53.91	22.23	0.622	97.01 (2405)	2658.126
		skglm	43.06	45.42	14.29	0.780	0.16 (4)	7.493
		Adaptive Lasso	57.84	50.65	17.74	0.788	100.00 (2479)	13.032
	Differential Coding	Logistic Regression	66.47	54.65	26.49	0.610	83.20 (2062)	340.854
		SVM	65.62	55.45	23.38	0.622	100.00 (2479)	325.272
		skglm	43.06	45.42	14.29	0.780	0.28 (7)	1.708
		Adaptive Lasso	57.84	51.08	19.43	0.745	100.00 (2479)	117.860

Appendix B

Declaration on the Use of AI-assisted Tools

This work used Grammarly to identify grammatical errors, improve sentence construction, and improve clarity. ChatGPT was used to guide document organisation, to suggest improvements in the clarity of the writing, and to support the understanding of specific excerpts from research papers. DeepL was used to translate Portuguese to English, and Copilot assisted in reproducing code blocks and helped fix bugs.