Erasmus School of Economics

Master Thesis Quantitative Finance

February 2, 2023

Interpretable Machine Learning for Credit Scoring

Author:

Qamar Suleri

Student id:

493256

University Supervisor:

Nick Koning

Company First Supervisor:

Michiel Cornelissen

Company Second Supervisor:

Robert Jan Sopers

Second Assessor:

Hakan Akyuz

Abstract

Accurate credit scoring is critical for loan providers as it enables them to manage credit risk exposure within strict regulatory limits. While Logistic Regression has traditionally been used for this purpose due to its straightforward interpretability, tree-based ensemble models have been shown to be more effective but lack transparency. Rule-based models, on the other hand, can extract important relationships from tree-based models while remaining interpretable. In this thesis, we introduce a novel model called *2-fold Unbiased Penalised Logistic Tree Regression (2-UPLTR)* and compare it to various existing rule-based models. Additionally, we introduce a new metric to quantify model complexity and examine the relationship between model complexity and performance. Our results from both simulation and empirical studies demonstrate that 2-UPLTR outperforms other models. Therefore, we recommend that financial institutions adopt 2-UPLTR for their credit scoring needs as it provides better predictions without sacrificing interpretability.

Keywords: Credit scoring, model interpretability, unbiased machine learning

Contents

1 Introduction

Credit risk management is a critical aspect of banking, as loans pose the greatest risk for financial institutions. Credit scoring models are used to evaluate the likelihood that loan applicants will be able to fulfill their financial obligations, allowing banks to accept applicants with a low probability of default (PD) and reject those with a higher PD. As banks hold billions of dollars in consumer loans, even slight improvements in the performance of credit scoring models can result in significant financial benefits.

Traditionally, the modelling of the PD has been done using Logistic Regression (LR) due to its clear interpretability. However, LR fails to take nonlinear relationships between the features and the target variable into account. As a result, various machine learning (ML) models have been shown to be more effective credit scoring models, particularly tree-based ensemble models (Addo et al., 2018; Gunnarsson et al., 2021; Lessmann et al., 2015; Petropoulos et al., 2019). In addition, with the growth in computing power and data availability, ML models can increasingly achieve better predictive performance, further surpassing LR.

Despite the benefits of ML models, financial institutions have been hesitant to adopt them due to their lack of transparency. This reluctance has been compounded by the global financial crisis of 2007, which increased the demand for insight into credit scoring models from credit regulators. Furthermore, since the General Data Protection Regulation took effect as law across the EU in 2018, algorithmic decisions about customers should be explainable to them (Goodman and Flaxman, 2017). That being said, the European Banking Authority recently acknowledged the benefits of using ML for risk differentiation[1]. To leverage the superior predictive power of ML models, it is necessary to cope with their lack of transparency. There is a growing body of literature on the topic of intrinsically interpretable ML models, but a comprehensive comparison of these models for credit scoring is currently lacking.

This research aims to fill this gap by comparing the most promising intrinsically interpretable credit scoring models in terms of complexity and performance. As tree-based models have been shown to be successful in credit scoring, we compare variations of the following interpretable tree-based models to the industry standard LR: Penalised Logistic Tree Regression (PLTR), RuleFit and RUle eXtraction (RUX). In addition, we analyse the tree-based ensemble models: Random Forest (RF), Extremely Randomized Trees (ERT), Adaptive Boosting (AdaBoost) and eXtreme Gradient

[1] https://www.eba.europa.eu/regulation-and-policy/model-validation/discussion-paper-machine-learning-irb-models Date accessed: February 2, 2023.

Boosting (XGBoost) as supplemental benchmarks. We introduce two unbiased versions of PLTR, 1-fold Unbiased PLTR (1-UPLTR) and 2-fold Unbiased PLTR (2-UPLTR). For RuleFit and RUX we analyse the following versions: RF-RuleFit, Ada-Rulefit, RF-RUX and Ada-RUX. We compare the models in a simulation and empirical study using the area under the receiver operating curve (AUC) and the Brier score (BS). For the simulation study, we use data-generating processes (DGPs) that are based on different degrees of non-linearity. For the empirical study, we use data that consists of aggregated United States state-level data with LendingClub's loan book covering the period from 2008 to 2019.

In short, our research makes the following contributions to the current literature:

- We provide a thorough comparison of the most promising intrinsically interpretable credit scoring models using both a simulation and empirical study.
- We propose a novel intrinsically interpretable credit scoring model called *Unbiased PLTR*, and demonstrate that it outperforms other existing interpretable models in the field.
- We introduce a new metric to quantify the complexity of credit scoring models.
- We investigate the relationship between model interpretability and predictive performance.

In the simulation study, we find 2-UPLTR to be the best-performing interpretable model across different variations of complexity scores and DGPs. 2-UPLTR and RF-RuleFit are the only two interpretable models that consistently significantly outperform LR. All ensemble models consistently outperform LR as well. In the empirical study, we find 2-UPLTR to be the best-performing model followed by 1-UPLTR and Ada-RUX. All models are ranked higher than LR, with the exception of RF-RUX. Furthermore, when testing for significance, we find 2-UPLTR to significantly outperform both PLTR and LR. Lastly, we provide an overview of the performance of the interpretable models for different levels of complexity. For financial institutions that prefer models that are not more complex than LR, we recommend using 2-UPLTR. For those that allow for models that can exceed LR in terms of complexity, we recommend considering Ada-RuleFit and Ada-RUX as they demonstrate the best performance in our empirical study.

This paper is structured in the following manner. In Section 2 we describe interpretability for credit scoring models. Subsequently, in Section 3 we present a short literature overview regarding credit scoring models. In Section 4, we describe our models and performance metrics. In Section 5 we introduce a score to quantify model complexity. Next, in Sections 6 and 7, we provide a simulation and empirical study, respectively. Finally, we conclude our research in Section 8.

2 Interpretability

Miller (2019) generally defines interpretability as the extent to which a human can understand the cause of a decision in a given context. Then, in the case of credit scoring models, we define interpretability to be the extent to which credit managers of banks and other stakeholders can understand the underlying logic and decision-making processes of the model. There are three key stakeholders in terms of model interpretability: credit managers, potential borrowers, and regulators, who all aim for different types of interpretability. Namely, model interpretability can be separated into two levels: global interpretability and local interpretability. Global interpretability concerns the general impact of features on the output of the model, whereas local interpretability concerns individual predictions.

The stakeholders of credit scoring models are shown in Figure 1. Credit managers of banks seek both global and local interpretability, as this allows for careful model validation, in which the model's accuracy and reliability are assessed. This is typically achieved through the comparison of the model's predictions to the actual outcomes of the loans, allowing for the identification of any potential issues or biases in the model. Potential borrowers are interested in local interpretability, as they want to know how the model has determined their credit rating and how they can improve it. This is especially the case for applicants who have been rejected. Hence, potential borrowers want to get insight into relevant counterfactuals. For example: " What would have happened to my credit scoring if I had 10% higher income?" or "What would have happened to my credit scoring if I did not have any outstanding loans?". Regulators prioritize global interpretability, as they want to see a clear link between the risk drivers and the target variable, to determine whether the model makes sense economically. It is important to note that model interpretability does not ensure trustworthiness, as for determining model trustworthiness expert judgement is necessary. Model interpretability simply ensures that experts have the necessary tools to determine model trustworthiness.

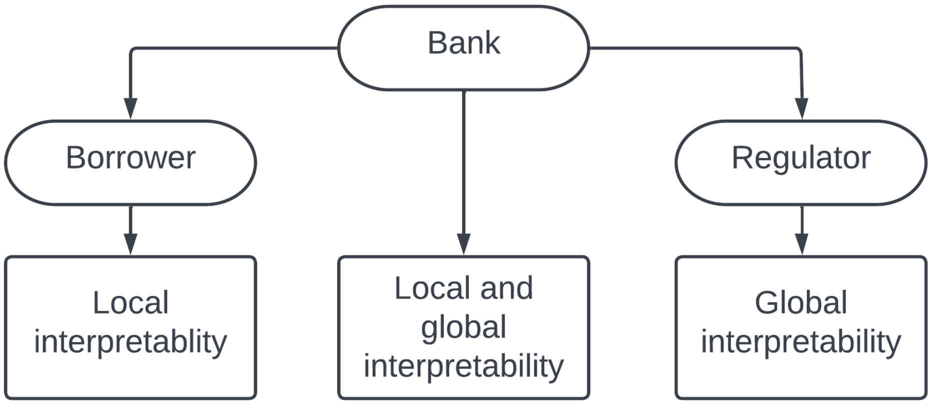


Figure 1: The stakeholders of banks in terms of model interpretability.

The use of uninterpretable machine learning models in high-stakes decision-making fields has prompted the development of methods for explaining them on a local level, such as LIME (Ribeiro et al., 2016) and Shapley values (Bussmann et al., 2021), and on a global level, such as partial dependence plots (Friedman, 2001) and accumulated local effects (Apley and Zhu, 2020). Rudin (2019) states that trying to explain black box models, rather than creating models that are interpretable in the first place, is likely to sustain bad practices and can potentially cause grave harm to society. Namely, relying on post-hoc explanations of black box models can create a false sense of understanding and trust in the model's predictions. This can lead to a lack of critical scrutiny and oversight, which can have serious consequences if the model is making decisions with significant impacts on people's lives, such as in the case of credit scoring models. Additionally, the lack of interpretability in black box models can make it difficult or impossible to hold the creators of these models accountable for their decisions and actions, which can undermine trust in the model and the institutions using it.

The trade-off between model interpretability and performance is a common assumption in the literature, as interpretability often comes at the cost of limiting the class of models that can be used. However, this trade-off may not always hold in practice, particularly when the data is structured and contains few or no noisy features (Razavian et al., 2015). As different fields have varying requirements for model interpretability, interpretability needs to be defined in a domain-specific manner. However, literature regarding metrics that quantify interpretability for specific fields is still gravely lacking (Murdoch et al., 2019).

3 Literature

The objective of credit scoring models is to differentiate between applicants that are likely and unlikely to default. Consequently, credit risk problems can be approached as classification problems. Due to its simplicity and intrinsic interpretability, LR has remained the industry workhorse for the last decades. LR has been compared to various ML models within the context of credit risk. It turns out that individual classifiers show only limited added performance in comparison to LR (Baesens et al., 2003; Yeh and Lien, 2009).

On the other hand, tree-based ensemble models, which combine the output of multiple individual decision trees (DTs), have been shown to be quite successful credit scoring models. Namely, in a study comparing 41 credit scoring models, Lessmann et al. (2015) find ensemble models to obtain the best performance and propose Random Forest (RF) as the benchmark instead of the industry-standard LR. Moreover, Gunnarsson et al. (2021) find eXtreme Gradient Boosting (XGBoost) to be the best ranking classifier in their credit scoring study. Furthermore, Addo et al. (2018) and Petropoulos et al. (2019) find tree-based models to be better credit scoring models than deep learning models. Additionally, Gunnarsson et al. (2021) do not recommend the usage of deep learning models for credit scoring as they do not outperform their shallower counterparts and have substantially more computational costs.

Despite their superior performance as credit scoring classifiers, tree-based models are not widely adopted by banks due to their lack of interpretability (Dumitrescu et al., 2022). In addition to a local and global level, model interpretability can be separated into two types: intrinsic and agnostic. The former restricts the complexity of the model to retain interpretability, whereas the latter separates the explanations from the model by applying auxiliary methods that analyse the model after training.

We primarily focus on intrinsically interpretable models due to their advantages on both a global and local level. Specifically, the overall weight of each risk driver in determining the target variable can easily be obtained and changed in these models. While some model-agnostic methods may approximate the overall weight of each risk driver, they do not provide the same level of ease in adjusting these weights as intrinsically interpretable models. As expert judgment regarding the role of risk drivers plays a significant role in fine-tuning credit scoring models in practice, the ease of manipulation provided by intrinsically interpretable models makes them particularly appealing. Furthermore, intrinsically interpretable models can be directly interpreted on both a global and local level, whereas different model-agnostic methods are required for local and global interpretation, making the use of intrinsically interpretable models more efficient.

Recently, Dumitrescu et al. (2022) introduced Penalised Logistic Tree Regression (PLTR), an intrinsically interpretable credit scoring model that compares competitively to RF. It operates by combining two steps: first, it creates decision rules based on short-depth decision trees, and then incorporates these decision rules as regressors, together with the original features, in a penalised logistic regression. Decision rules are "IF-THEN" statements consisting of conditions followed by a prediction; "IF" the conditions are met "THEN" make a certain prediction. For example: "IF loan > 5000 & age < 25, THEN default = true." As such, decision rules can be viewed as dummy variables.

Friedman and Popescu (2008) introduced another intrinsically interpretable classification model named RuleFit, which is an ensemble of decision rules. This algorithm is an ensemble of decision rules that operates similarly to PLTR, but obtains decision rules through a different process. Although this algorithm has not been previously applied in a credit scoring study, it has been shown to demonstrate comparable performance to tree-based ensemble models across 100 datasets. Furthermore, Akyüz and Birbil (2021) proposed RUle eXtraction (RUX), an approach that extracts interpretable rules from tree-based models using linear programming. This method allows for the assignment of cost coefficients based on different attributes of the decision rules, such as rule length (the number of conditions in a decision rule) and estimator weights.

As tree-based ensemble models have been demonstrated to be high-performing credit scoring models, they form the central focus of this research. We compare the above-mentioned methods to determine the optimal method for use in a credit scoring context. As a comparison between decision rule-based methods is lacking in the current credit scoring literature, this research aims to provide a comprehensive overview of different underlying levels of model complexity for these methods.

4 Methodology

This section presents an overview of the methods used in this thesis. We begin by discussing LR and Decision Tree (DT) as standard models. LR serves as the baseline as it is currently the industry standard for credit scoring, while DT functions as a benchmark as it is the fundamental component of all tree-based models. Next, we present the "black-box" tree-based ensemble models. Lastly, we discuss the intrinsically interpretable models PLTR, RuleFit and RUX, and introduce our novel method: Unbiased PLTR.

4.1 Standard models

Let $(x_i, y_i), i = 1, \ldots, n$ represent a sample of n independently and identically distributed observations, where $x_i \in \mathbb{R}^p$ is a vector containing p features of borrower i, and y_i is a binary variable indicating whether borrower i defaults (i.e., $y_i = 1$) or not (i.e., $y_i = 0$). The objective of credit scoring models is to estimate the posterior probability $P(y_i = 1|x_i)$, which represents the probability that borrower i defaults given their features x_i (Dumitrescu et al., 2022).

4.1.1 Logistic Regression (LR)

The first baseline model is standard (LR), which models the posterior probability of default using the logistic cumulative distribution function (CDF) $F(\cdot)$ as:

$$P(y_i = 1|x_i) = F(\eta(x_i;\beta)) = \frac{1}{1+\exp(-\eta(x_i;\beta))} \tag{1}$$

where $\eta(x_i;\beta) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{i,j}$, and $\beta = (\beta_0, \ldots, \beta_{p-1}, \beta_p) \in \mathbb{R}^{p+1}$ represents the vector of unknown parameters that can be estimated by maximizing the log-likelihood function:

$$\mathcal{L} = \sum_{i=1}^{n} \left\{ y_i \log\{F(\eta(x_i;\beta))\} + \big(1-y_i\big)\big(1-\log\{F(\eta(x_i;\beta))\}\big) \right\} + \lambda_1 \sum_{i=0}^{p} |\beta_i| + \lambda_2 \sum_{i=0}^{p} \beta_i^2. \tag{2}$$

where the hyperparameters λ_1 and λ_2 control the strength of the elastic net penalty terms (Zou and Hastie, 2005). The elastic net penalty terms are used to deal with data that contains a high number of variables. The first term of Equation 2 is the L_1 norm, which uses the sum of the absolute coefficients to shrink the regression coefficients of variables with a minor contribution to the outcome towards zero. Similarly, the second term of Equation 2 is the L_2 norm, which uses the sum of the squared coefficients to shrink the regression coefficients of variables with a minor contribution to the outcome close to zero.

The main advantage of LR is its interpretability, as it only searches for a single linear decision boundary in the feature space. The core assumption for finding this boundary is that the function $\eta(x_i;\beta)$ and the features are linearly related. As a result, the marginal contribution of each feature to the probability of default can be obtained as:

$$\frac{\partial P(y_i = 1|x_i)}{\partial x_{i,j}} = \beta_j \frac{\exp(\eta(x_i;\beta))}{[1+\exp(\eta(x_i;\beta))]^2}, \tag{3}$$

4.1.2 Decision tree (DT)

DTs are non-parametric models that can capture non-linear relationships by recursively partitioning the feature space into smaller regions. The partitions are chosen such that the observations in each region are as similar as possible, which enables the final nodes, called leaves, to effectively discriminate among the potential outcomes.

Let D_m contain the observations and $\theta_m = (j_m, t_{m,j})$ be a candidate split for a given node m, where $j_m = 1, 2, \dots p$ specifies an according threshold value $t_{m,j}$. The Classification and Regression Tree (CART) algorithm (Breiman et al., 1984) separates the observations into two data sets $D_{m,1}(\theta_m)$ and $D_{m,2}(\theta_m)$, where

$$D_{m,1}(\theta_m) = (x_i, y_i) | x_{i,j} < t_{m,j} \quad \text{and} \quad D_{m,2}(\theta_m) = (x_i, y_i) | x_{i,j} \geq t_{m,j}, \tag{4}$$

and θ_m is estimated as

$$\widehat{\theta}_m = (\widehat{j}_m, \widehat{t}_{m,j}) = \arg\max_{\theta_m} \mathcal{H}(D_m) - \frac{1}{2}\Big(\mathcal{H}(D_{m,1}(\theta_m)) + \mathcal{H}(D_{m,2}(\theta_m))\Big). \tag{5}$$

The best split is chosen by maximizing a criterion $\mathcal{H}(\cdot)$ that measures the diversity of the observations in the child nodes. We use the Gini impurity, which is defined as:

$$G = \sum_{i=1}^{C} p(i)(1 - p(i)), \tag{6}$$

where C denotes the number of classes and $p(i)$ denotes the probability of randomly picking an element of class i. Hence, the Gini impurity yields the likelihood of misclassifying a randomly chosen observation, given that it is labelled according to the class distribution in the data set. By minimizing the Gini impurity, we can find thresholds that succeed most in discriminating among the potential outcomes.

To illustrate the workings of a decision tree, consider the example in Figure 2. The tree is built using the features `Income`, `Age`, and `Interest`. The splits at the nodes are determined by comparing the feature values to a threshold value. For example, at the first node, the tree splits the observations into two groups based on whether the income is less than or equal to 30,000. This process is repeated for each child node until the observations reach the leaf nodes. At the leaf nodes, the tree calculates the percentage of defaults for the observations in that node. This percentage is used to classify an unseen observation that ends up in that leaf as either a default or non-default.

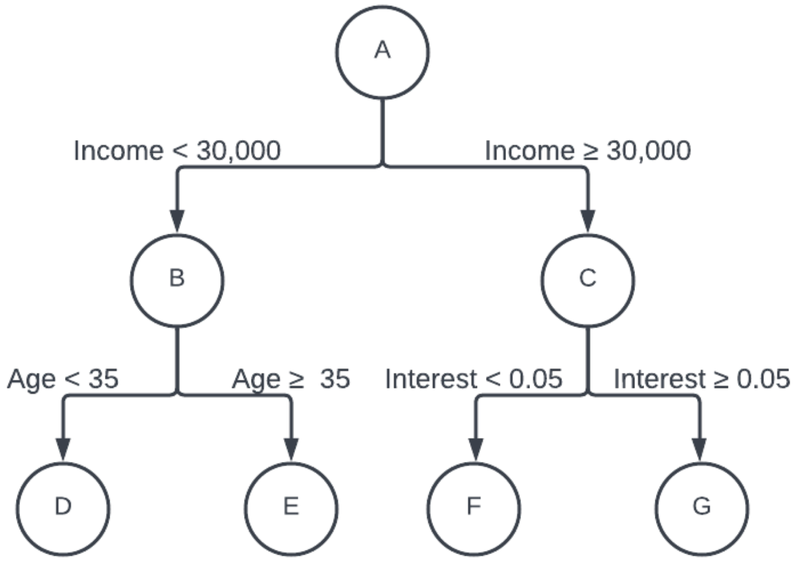


Figure 2: An example of a decision tree. At nodes A-C, the observations are branched to two nodes in the next layer. At leaf nodes D-G, unseen observations are assigned classes based on majority rule.

4.2 Tree-based ensemble models

Decision trees, while being simple and interpretable, can be prone to overfitting and high variance due to their sensitivity to outliers. As a result, ensemble methods have been widely adopted in practice as an alternative (Bramer, 2007). These methods include bagging techniques like Random Forest and Extremely Randomized Trees, as well as boosting methods like eXtreme Gradient Boosting and Adaboost. These ensemble methods have been shown to improve the overall performance of the model by reducing the variance and bias of the base models, resulting in enhanced generalization performance. They have been proposed as a benchmark over traditional LR for credit scoring applications.

4.2.1 Random Forest (RF)

RF is a widely used ensemble method that improves the predictive performance of a model by combining the output of multiple independently grown decision trees. The method utilizes bootstrap sampling, drawing with replacement from the full dataset, to grow each tree. To decrease correlation among the trees, RF only selects a random subset of $\sqrt{p}$ features of a data set containing p features, for forming the splits in each tree. The final prediction is determined by majority voting among the trees. The number of trees is a crucial hyperparameter to be optimized, as increasing the number of trees improves the generalization error of the model (Breiman, 2001).

4.2.2 Extremely Randomized Trees (ERT)

ERT is an ensemble method that, like RF, combines the output of multiple decision trees. However, while RF grows trees using bootstrap subsamples, ERT builds trees using the entire training set. Additionally, instead of using the Gini impurity in Equation 6 to determine the best split, ERT randomly selects cut-points for a subset of features and selects the best cut-point to split the data. This approach tends to reduce variance but increases bias in the model.

4.2.3 Adaptive Boosting (AdaBoost)

Whereas the main goal of bagging models is to decrease the variance, the main goal of boosting models is to decrease the bias. Boosting consists of growing decision trees recursively, where each new tree aims to correct the errors made by the previously trained tree. AdaBoost is a popular boosting algorithm that is primarily designed to tackle binary classification problems (Freund et al., 1996). The algorithm begins by fitting a decision tree to the training data, with all samples initially given an equal weight of $w_{i,t} = 1/N$, where $w_{i,t}$ indicates the importance of correctly classifying sample i after fitting tree t. The misclassified data points are then assigned a higher weight, making them more likely to be classified correctly in the next decision tree. This process is repeated until the ensemble consists of T trees.

The impact of each decision tree t is denoted by

$$\alpha_t = \frac{1}{2} \log \frac{1 - \text{error}}{\text{error}}. \tag{7}$$

As shown in Figure 3, a tree with few misclassifications corresponds to a large positive impact, whereas a tree with many misclassifications corresponds to a large negative impact. This allows the boosting algorithm to assign higher weights to the observations that are misclassified by previous trees, thus reducing the bias of the model. The updated weights are calculated using the following equation:

$$w_{i,t} = \begin{cases} w_{i,t-1} \exp(-\alpha_t), & \text{if } \widehat{y}_{i,t} = y_i; \\ w_{i,t-1} \exp(\alpha_t), & \text{otherwise}, \end{cases} \tag{8}$$

where $\widehat{y}_{i,t}$ represents the predicted value of the target variable after the t^{th} tree is fitted. When an observation is correctly classified, α_t is positive, resulting in a decrease in weight for that observation. On the other hand, when an observation is misclassified, α_t is negative, resulting in an increase in weight for that observation.

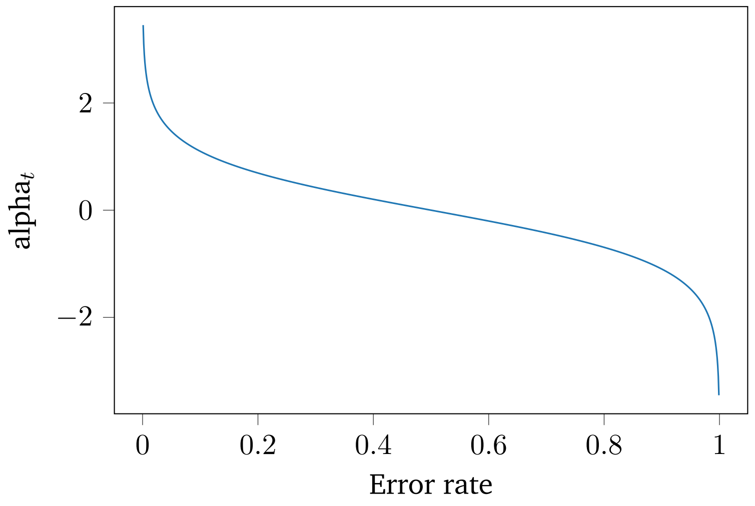


Figure 3: Impact curve of decision trees.

4.2.4 eXtreme Gradient Boosting (XGBoost)

XGBoost is an ensemble method similar to AdaBoost, which combines decision trees to improve the predictive performance of the model. Instead of using simple majority voting like AdaBoost, XGBoost uses gradient descent to optimize the loss function and correct the errors made by previous trees. This method is regularized to prevent overfitting and uses second-order gradients for more efficient optimization. The loss function for node selection and splitting for a tree structure q is given by

$$\mathcal{L}(q) = -\frac{1}{2}\sum_{j=1}^{J}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda}, \tag{9}$$

where J is the number of leaves in the tree, I_j is the set of observations in leaf node j, g_i is the gradient of the loss function, h_i is the hessian of the loss function, and λ is a regularization parameter.

Due to the computational hardship of enumerating through all possible tree structures in order the find the optimal tree structure, a greedy algorithm that iteratively adds branches to a tree is used instead (Chen and Guestrin, 2016). The loss reduction after the split is given by

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\sum_{j=1}^{J} \frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right], \tag{10}$$

where I_L and I_R are the sets containing the observations in the left and right nodes after the split, respectively.

4.3 Intrinsically interpretable tree-based models

We describe PLTR, RuleFit and RUX, which all use tree-based models to extract decision rules and then use these decision rules as features in a linear model. Additionally, we propose two unbiased variants of PLTR.

4.3.1 Penalised Logistic Tree Regression (PLTR)

PLTR seeks to improve the predictive performance of LR while maintaining its interpretability. This is accomplished by incorporating decision rules extracted from short-depth decision trees as additional features in a penalized LR model, alongside the original features. The interpretability of the model is a crucial aspect of PLTR, and it is achieved by keeping the decision trees short. Decision trees are known to be transparent and easy to interpret when they are short, but as the depth increases, the transparency of the model decreases. As the depth of the tree increases, the number of leaf nodes increases, making it more challenging to understand the underlying decision rules. However, using decision trees with low depth can limit the performance of the model. To address this issue, PLTR employs various decision trees with a maximum depth of two layers and combines them in a penalized LR model. The method consists of two steps.

Step 1: extracting decision rules from one-split and two-split decision trees. One-split trees are trees with only one layer. For example, the tree in Figure 2 is a two-split tree, but if nodes D-G were removed, so that B and C form the leaf nodes, it would be a one-split tree. To extract decision rules from one-split trees, a decision tree is fitted using each of the p features, resulting in two leaf nodes, one of which is kept for the next step. For two-split trees, decision trees are fitted for each pair-wise combination of the features. However, when one feature is more informative than the other, this can result in redundant bivariate threshold effects. To avoid this, redundant threshold effects are removed, resulting in at most $\frac{p(p-1)}{2}$ decision rules.

Step 2: incorporating the extracted decision rules in a logistic regression:

$$P(y_i = 1|\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) = \frac{1}{1 + \exp(-\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta)}, \tag{11}$$

where

$$\eta(x_i; \mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) = \beta_0 + \sum_{j=1}^{p} \alpha_j x_i + \sum_{j=1}^{p} \beta_j \mathcal{V}_{i,1}^{(j)} + \sum_{j=1}^{p-1} \sum_{k=j+1}^{p} \gamma_{j,k} \mathcal{V}_{1,2}^{j,k}, \tag{12}$$

which contains the vector $\Theta = (\beta_0, \alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_p, \gamma_{1,2}, \ldots, \gamma p - 1, p)$ and binary variables $\mathcal{V}_{i,1}^{(j)}$, representing univariate threshold effects based on feature j and and bivariate threshold effects based on features j and k, respectively. The parameter vector Θ can be estimated by maximizing the log-likelihood:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \left\{ y_i \log\{F(\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta))\} + \big(1 - y_i\big)\big(1 - \log\{F(\eta(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta))\}\big) \right\}. \tag{13}$$

As adding the decision rules based on short-depth trees as regressors results in a rather large number of features, the adaptive lasso estimator of (Zou, 2006) is used to drop uninformative features. Using fewer features means that the model output is dependent on fewer variables, and hence increases model interpretability. Lasso estimates are obtained as

$$\widehat{\theta}_{alasso} = \arg\min_{\Theta} -\mathcal{L}(\mathcal{V}_{i,1}^{(j)}, \mathcal{V}_{i,2}^{(j,k)}; \Theta) + \lambda |\widehat{\Theta}^{(0)}|^{-1} |\Theta|, \tag{14}$$

where the initial estimator $\widehat{\theta}^{(0)}$ is the value obtained from the logistic regression with a ridge penalty.

4.3.2 Unbiased Penalised Logistic Tree Regression (UPLTR)

PLTR uses all data points for extracting the decision rules in step 1 and fitting the penalized LR in step 2. As this means that the information of each observation is used twice, this could lead to a high bias in the model. To address this issue, we introduce two variants of PLTR: 1-fold Unbiased PLTR (1-UPLTR) and 2-fold Unbiased PLTR (2-UPLTR).

In 1-UPLTR, the data is split into two distinct subsamples, $\mathcal{A}$ and $\mathcal{B}$, where subset $\mathcal{A}$ is used for extracting the decision rules and subset $\mathcal{B}$ for fitting the penalized LR. The roles of the subsamples can then be reversed such that subset $\mathcal{B}$ is used for extracting the decision rules and subset $\mathcal{A}$ for fitting the penalized LR. In 2-UPLTR, the LR coefficients of the two subsets are averaged, which reduces the bias in the model. These unbiased variants of PLTR provide a more robust and reliable solution to the bias issue while maintaining the interpretability of the model.

4.3.3 RuleFit

RuleFit fits a sparse linear model using decision rules extracted from a tree-based ensemble in addition to the original features. Let S_j be the set of all possible values for feature x_j, and s_{jm} be a specified subset that contains m of those values. Then each decision rule can be noted as:

$$r_m(x) = \prod_{j=1}^{n} \mathbf{1}(x_j \in s_{jm}), \tag{15}$$

where $\mathbf{1}(\cdot)$ is an indicator function which is equal to 1 if its argument is satisfied. Consequently, the decision rule is binary, being only equal to 1 if all of its input variables are within their respective subsets $\{x_j \in s_{jm}\}_1^n$. For variables that assume orderable values, the subsets are taken to be intervals $s_{jm} = (l_{jm}, u_{jm}]$. where l_{jm} and u_{jm} denote a lower and upper bound, respectively.

For some variables, the subset of values appearing is equal to the full set of values such that $s_{jm} = S_j$. As $x_j \in s_{jm}$ will always be satisfied in this case, Equation 15 can be simplified to:

$$r_m(x) = \prod_{s_{jm} \neq S_j}^{n} \mathbf{1}(x_j \in s_{jm}). \tag{16}$$

Thus, the more variables there are that have a subset of values which is equal to the full set of values ($s_{jm} = S_j$), the shorter the decision rules are. Moreover, the shorter the decision rules are, the easier they can be interpreted.

To generate an ensemble of decision rules, fast algorithms such as RF and AdaBoost can be used. Let $\{f_m(x)\}_{m=1}^{M}$ denote a tree ensemble containing M decision trees, each with output $f_m(x)$. The set of all decision rules derived from that ensemble is denoted as $r_k(x)_{k=1}^{K}$, where the total number of rules is

$$K = \sum_{m=1}^{M} 2(l_m - 1), \tag{17}$$

where l_m denotes the number of leaf nodes for tree m. The forecasts are then made using the model

$$F(x) = \widehat{a}_0 + \sum_{k=1}^{K} \widehat{a}_k r_k(x), \tag{18}$$

where

$$\{\widehat{a}_k\}_0^k = \arg\min_{\{a_k\}_0^k} \sum_{i=1}^{N} \Big(L(y_i, a_0 + \sum_{k=1}^{K} a_k r_k(x_i) \Big) + \lambda \sum_{k=1}^{K} |a_k|, \tag{19}$$

which uses the squared-error ramp loss

$$L(y, F) = [y - \min(-1, \max(1, F))]^2, \tag{20}$$

and hyperparameter λ for determining the impact of the Lasso penalty. Optimizing this loss function has been shown to produce a comparable performance to other commonly used loss criteria but with increased robustness against misclassified observations (Friedman and Popescu, 2008)

4.3.4 RUle eXtraction (RUX)

RUX extracts decision rules from trained ensemble models by using linear programming and hence is scalable to large data sets. Let $\mathcal{J}$ be a group of decision rules and let rule $j \in \mathcal{J}$ assign the vector $R_j(x_i)$ to observation x_i if that observation is covered by rule j. To predict the class of observation x_i with the rules in group $\mathcal{J}$, a set of nonnegative weights w_j are associated with the rules and the following equation is evaluated:

$$\widehat{y}_i(x_i) = \arg\max \left(\sum_{j \in \mathcal{J}} a_{ij} R_j(x_i) w_j \right), \tag{21}$$

where a_{ij} is equal to 1 if rule j covers observation i and equal to 0 otherwise.

The hinge loss is used to determine the total classification loss, which is defined as

$$\sum_{i=1}^{n} \max\{1 - \sum_{j \in \mathcal{J}} \widehat{a}_{ij} w_j, 0\}, \tag{22}$$

where $\widehat{a}_{ij} = a_{ij} R_j(x_i)^\intercal y_i(x_i)$. The use of this loss function allows for the formulation of a linear programming model that aims to find the set of rules that minimize the total loss. To accomplish this, auxiliary variables v_i for $i = 1, \ldots, n$, defined as $v_i \geq \max\{1 - \sum_{j \in \mathcal{J}} \widehat{a}_{ij} w_j, 0\}$, are introduced. The resulting problem can be formulated as follows:

$$\text{minimize} \quad \sum_{i=1}^{N} v_i + \sum_{j \in \mathcal{J}} c_j w_j \tag{23}$$

Subject to

$$\sum_{j\in\mathcal{J}} \widehat{a}_{ij}w_j + v_i \geq 1, \qquad i = 1\dots n; \tag{24}$$

$$\sum_{j\in\mathcal{J}} a_{ij}w_j \geq \varepsilon, \qquad i = 1\dots n; \tag{25}$$

$$v_i \geq 0, \qquad i = 1\dots n; \tag{26}$$

$$w_j \geq 0, \qquad j \in \mathcal{J}, \tag{27}$$

where cost coefficients $c_j \geq 0, j \in \mathcal{J}$ prevent (1) rules from being too long and (2) involving many non-zero weights.

4.4 Feature importance

To understand the role of different features in determining the final output of the model, we calculate feature importance. This allows banks to identify the key drivers of the probability of default. For interpretable models, this can be done directly by analyzing the decision rules that are generated, as they are transparent. However, for ensemble models, this transparency is not present and feature importance can only be calculated for the original features, not the decision rules that are created internally.

4.4.1 Feature importance of interpretable models

To understand the relative importance of different features in a linear model, Friedman and Popescu (2008) propose calculating feature importance (FI) as the product of the absolute value of the estimated coefficient and the standard deviation of the standardized data for the feature. This can be represented mathematically as:

$$\mathrm{FI}_j = |\widehat{\beta}_j| \times \sigma(\dot{x}_j), \tag{28}$$

where $\widehat{\beta}_j$ is the estimated coefficient of the model for feature j, $\sigma(\cdot)$ is the standard deviation and $\dot{x}_j$ contains the standardized data instances of feature j.

For decision rules, the importance of the feature is calculated as

$$\mathrm{FI}_k = |\widehat{\beta}_k| \times \sqrt{s_k(1 - s_k)}, \tag{29}$$

where $\widehat{\beta}_k$ is the estimated coefficient of the model for decision rule k and s_k is the support of the feature in the data, which is defined as the ratio of data instances to which the decision rule applies. The support of the feature can be calculated as:

$$s_k = \frac{1}{n}\sum_{i=1}^{n} D_k(x_i), \tag{30}$$

where $D_k(x_i)$ is equal to 1 if decision rule D_k is applied to data instance x_i.

To fully understand the working of our tree-based models, we analyze their interpretability on a global and local level. Global explanations focus on the overall impact of features on the model's predictions, whereas local explanations focus on individual predictions.

4.4.2 Feature importance of ensemble models

To calculate feature importance for ensemble models, we use the Gini importance, also known as the mean decrease in impurity, which is defined as the normalized total depletion of the Gini impurity caused by the feature. To obtain feature importance, we first calculate the importance of each node. Node importance (ni) can be calculated as:

$$\mathrm{ni}_j = w_j G_j + w_j^{left} G_j^{left} + w_j^{right} G_j^{right}, \tag{31}$$

where w_j and G_j denote the weight and Gini impurity of node j, respectively, and the left and right superscripts refer to the left and right child node of node j, respectively. Feature importance (fi) for feature i can then be calculated as the weighted fraction of node importances for which feature i is responsible:

$$\mathrm{fi}_i = \frac{\sum_{j\in\mathcal{J}} \mathbf{1}_{i,j}\mathrm{ni}_j}{\sum_{j\in\mathcal{J}} \mathrm{ni}_j}, \tag{32}$$

where $\mathcal{J}$ denotes a set containing all nodes and the indicator function $\mathbf{1}_{i,j}$ is equal to 1 when node j uses feature i to split the data and 0 otherwise. Finally, the normalized feature importance (FI) for feature i can be obtained as:

$$\mathrm{FI}_i = \frac{\mathrm{fi}_i}{\sum_{i\in\mathcal{F}} \mathrm{fi}_i}, \tag{33}$$

where $\mathcal{F}$ denotes a set containing all features.

4.5 Performance measures

Based on the findings of Lessmann et al. (2015), the performance of credit scoring models can be evaluated using the area under the receiver operating characteristic curve (AUC) and the Brier Score (BS). The AUC measures the model's discriminatory ability and the BS assesses the accuracy of the predicted probabilities.

4.5.1 Area under the receiver operating curve (AUC)

The AUC is used to compare the receiver operating characteristic (ROC) curves of different models. ROC curves are based on the values of confusion matrices. An example is shown in Table 1.

		Predicted class	
		Positive	Negative
Actual class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Table 1: A confusion matrix.

Confusion matrices offer a comprehensive representation of a model's predicted and actual outcomes. In the context of binary classification, such as credit scoring, *True Positive (TP)* and *True Negative (TN)* are used to denote the number of observations that are correctly classified as positive (default) and negative (non-default) respectively. Then, the sensitivity measures the proportion of positive examples that are predicted to be positive:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \tag{34}$$

while the specificity measures the proportion of negative observations that are predicted to be negative:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}. \tag{35}$$

The ROC curve is a two-dimensional graph which is created by plotting the sensitivity against [1-specifity], which is also known as *the probability of false alarm*, at various threshold settings.

Two representative Receiver Operating Characteristic (ROC) curves are illustrated in Figure 4. To evaluate the performance of different models, the Area Under the Curve (AUC) is commonly used as it provides a single scalar metric. The AUC can be interpreted as an approximation of the probability that a randomly selected positive instance will be ranked higher than a randomly selected negative instance. As such, it can be observed from Figure 4 that Model 1 exhibits superior discriminatory ability as its AUC is higher than that of Model 2. Additionally, it is important to note that for an adequate model, the AUC should be significantly greater than 0.5, which corresponds to the AUC of a random classifier

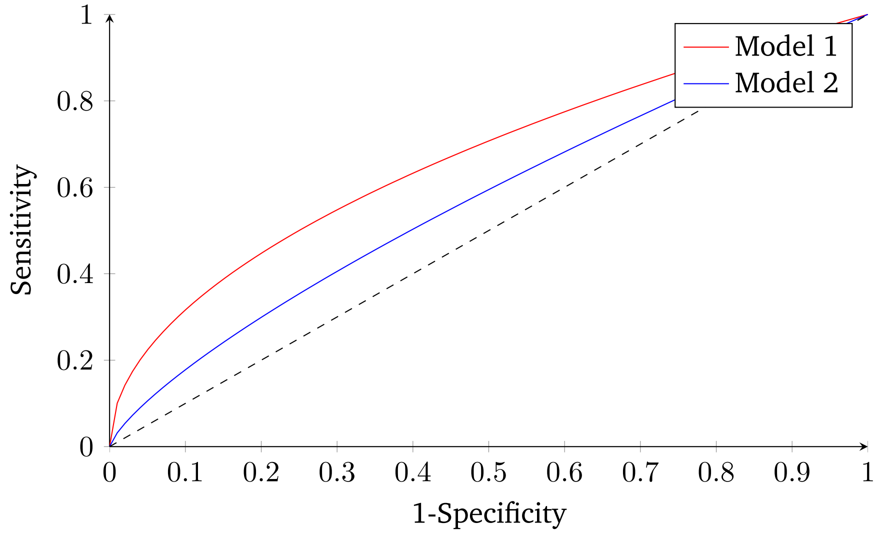


Figure 4: The ROC-curve for two models.

Hanley et al. (1983) propose a method for determining the statistical significance of the difference in AUC between two models that are based on the same observations. The test statistic is given by:

$$Z = \frac{\text{AUC}_A - \text{AUC}_B}{\sqrt{\text{SE}_A^2 + \text{SE}_B^2 - 2r\text{SE}_A\text{SE}_B}}, \tag{36}$$

where AUC_M and SE_M are the mean of the observed AUCs and the estimated corresponding standard error for model $M \in \{A, B\}$, respectively, and r is the estimated Pearson product-moment correlation coefficient between the AUCs of models A and B.

4.5.2 Brier Score (BS)

The BS calculates the mean squared error between the predicted probabilities and the observed values for the target variable:

$$\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(\widehat{y}_i - y_i)^2, \tag{37}$$

5 Complexity score

Interpretability is a crucial aspect in the selection of a credit scoring model, as it allows financial institutions to understand and explain the model's decision-making process. However, a lack of a standardized metric for quantifying model complexity has hindered comparisons of interpretability between models. To address this gap, we propose a novel complexity score for credit scoring models that is based on the number and length of the decision rules used in the model. Our score captures the idea that a model with a larger number of longer decision rules is more complex than one with fewer and shorter rules. This relationship between complexity and predictive performance is crucial for making informed recommendations to financial institutions.

Our complexity score is defined as the sum of the penalties for each decision rule, plus the number of ordinary features (features that are not decision rules) in the model:

$$C = \sum_{d=1}^{D} P_d + F, \tag{38}$$

where penalty P_d is defined as the penalty for decision rule d, and D and F are the number of decision rules and ordinary features in the model, respectively. We consider three straight-forward cases for decision rule penalty P_d, resulting in the following complexity scores:

$$C_k = \sum_{d=1}^{D} P_{d,k} + F = \begin{cases} \sum_{d=1}^{D} L_d + F, & \text{if } k = 1 \text{ ;} \\ \sum_{d=1}^{D} (L_{d,} - 0.5) + F, & \text{if } k = 2 \text{ ;} \\ \sum_{d=1}^{D} 2^{L_{d,}-1} + F, & \text{if } k = 3 \text{ ,} \end{cases} \tag{39}$$

where rulelength L_d is defined as the number of conditions in decision rule d. For example, the decision rule `loan` > 5000 & `age` < 25 has a length of 2.

The baseline complexity score C_1 treats the penalty of a decision rule of length 1 as equivalent to the penalty of a single ordinary feature, implying that a decision rule of length 1, such as `age` < 25, has the same ease of interpretation as a single feature. The complexity score C_2, reduces the penalty of a decision rule of length 1 to half the penalty of a single ordinary feature, implying that a decision rule of length 1 has more ease of interpretation than a single feature. According to this complexity score, a model with many short decision rules is preferred over a model with few but long decision rules. The relative advantage of subtracting a half from the penalty decreases as the decision rule becomes longer, which means that compared to C_1, especially the short decision rules are favoured when using C_2, while the penalty for long decision rules remains approximately the same. For the last complexity score C_3, longer decision rules are penalized exponentially harder, while the penalty for short decision rules remains the same as for C_1. An example of the penalties of the complexity scores for decision rules of different lengths are shown in Table 2.

Table 2: Decision rule penalties for the different complexity scores.

Decision rule	C_1	C_2	C_3	Rule length
`loan` > 5000	1	0.5	1	1
`loan` > 5000 & `age` < 25	2	1.5	2	2
`loan` > 5000 & `age` < 25 & `gender` = male	3	2.5	4	3

6 Simulation study

In this section, we evaluate the performance of all models across different complexity scores and data generating processes (DGPs). The DGPs are designed to reflect varying levels of underlying non-linearity, specifically by incorporating first, second, and third-degree feature interactions. For each DGP, we report the AUC along with its 95% confidence interval. Due to computational limitations, the number of simulations per DGP is restricted to 100.

6.1 Data-generating process (DGP)

We compare the performance of the models based on three different data-generating processes (DGPs) using a set-up similar to that outlined in Dumitrescu et al. (2022). The data used for this comparison consist of 10 standard normal features, designated as $x_{i,j}$ where $j = 1, \ldots, p$ and $i = 1, \ldots, n$, with a sample size of 2,000. 80% of these observations are used to train the data, while the remaining 20% are used for testing the models. The underlying functions in the DGPs contain nonlinear effects using thresholds and interactions, as these are commonly seen in real-world data. For example, in the context of credit scoring, an income threshold effect can be observed, where the probability of default decreases significantly above a certain income level

The functions in the DGPs utilize a vector of parameters $(\beta_0, \beta_1, \ldots, \beta_p, \beta_{1,2}, \ldots, \beta_{p-1,p}, \beta_{1,2,3}, \ldots, \beta_{p-2,p-1,p})'$ which contains components that are randomly drawn from a uniform [-1,1] distribution and a vector of threshold values $(\gamma_1, \ldots, \gamma_p, \delta_1, \ldots, \delta_p, \theta_1, \ldots, \theta_p)'$ that are randomly selected from the support of each feature after excluding the 10% highest and lowest values. The underlying functions for DGPs 1-3 are simulated as follows:

$$\begin{aligned}
\eta_1(x_i;\beta) &= \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{1}(x_{i,j} \leq \gamma_j), \\
\eta_2(x_i;\beta) &= \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{1}(x_{i,j} \leq \gamma_j) + \sum_{j=1}^{p-1} \sum_{k=j+1}^{p} \beta_{j,k} \mathbf{1}(x_{i,j} \leq \delta_j) \mathbf{1}(x_{i,k} \leq \delta_k), \\
\eta_3(x_i;\beta) &= \beta_0 + \sum_{j=1}^{p} \beta_j \mathbf{1}(x_{i,j} \leq \gamma_j) + \sum_{j=1}^{p-1} \sum_{k=j+1}^{p} \beta_{j,k} \mathbf{1}(x_{i,j} \leq \delta_j) \mathbf{1}(x_{i,k} \leq \delta_k) \\
&\quad + \sum_{j=1}^{p-2} \sum_{k=j+1}^{p-1} \sum_{l=j+2}^{p} \beta_{j,k,l} \mathbf{1}(x_{i,j} \leq \theta_j) \mathbf{1}(x_{i,k} \leq \theta_k) \mathbf{1}(x_{i,l} \leq \theta_l),
\end{aligned} \tag{40}$$

where the indicator function $\mathbf{1}(\cdot)$ is equal to 1 if the restriction within brackets is satisfied and 0 otherwise. Subsequently, the default variable is simulated for DGPs k = 1,2,3 as

$$y_{i,k} = \begin{cases} 1, & \text{if } P_k(y_i = 1|x_i) = \frac{1}{1+\exp(-\eta_k(x_i;\beta))} > \pi_k; \\ 0, & \text{otherwise,} \end{cases} \tag{41}$$

where π_k denotes the median of the generated probabilities for each DGP. As can be seen in Equation 40, the DGPs 1-3 include first, second and third-degree interactions among features, respectively. By evaluating the performance of each model, we can determine the relative performance of each model with respect to the nonlinearity present in the underlying DGP.

6.2 Results

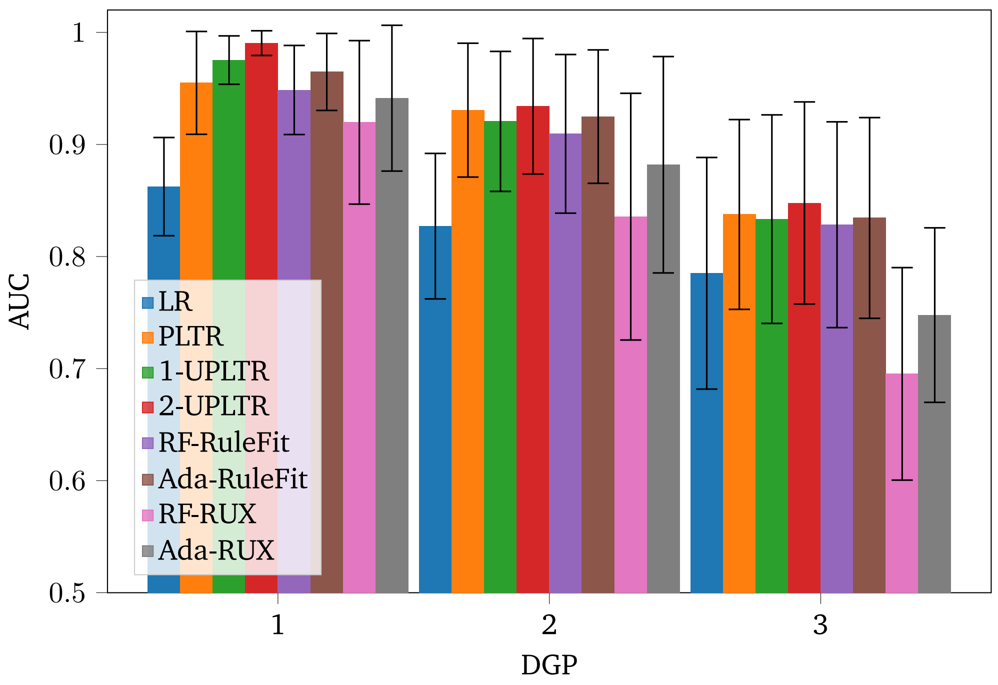
The performance of interpretable models is presented in Figure 5 for varying levels of non-linearity in the DGPs and complexity scores. As expected, the performance of the models deteriorates as the non-linearity in the DGP increases. A complexity restriction of $C_k \leq 20$ is imposed on the models for $k = 1, 2, 3$, ensuring that their complexity score never exceeds double that of the LR. This restriction is set to double the complexity of LR due to the limited number of features in the data. In situations where the data comprises a large number of features, the complexity limit can be set equal to that of LR.

When considering only the models with the baseline complexity score (Figure 5a), we observe that 2-UPLTR consistently achieves the highest performance. For DGP 1, all models outperform LR, with only RF-RUX and Ada-RUX having overlapping confidence intervals with LR. 2-UPLTR has the best performance, followed by 1-UPLTR and Ada-RuleFit. As the volatility in performance increases for DGPs 2 and 3, all confidence intervals overlap. For DGP 2, all models still outperform LR, with PLTR and 2-UPLTR having the best performance, followed by Ada-RuleFit. For DGP 3, not all models outperform LR, with RF-RUX and Ada-RUX having lower performance. 2-UPLTR remains the best performer among all models.

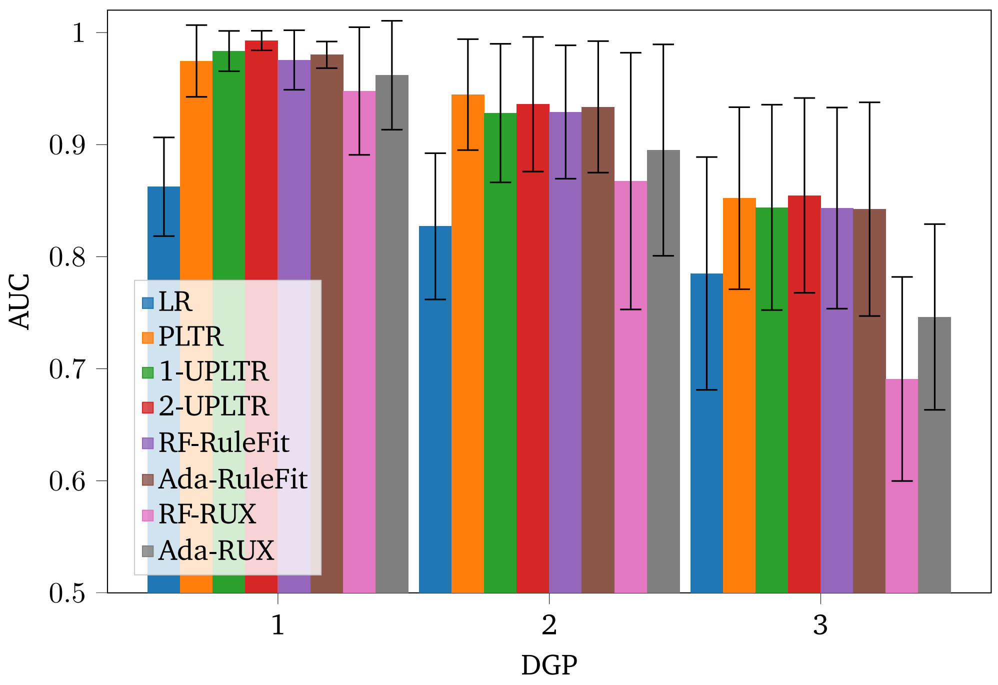
The performance of the models improves when using the second complexity score (Figure 5b) which reduces the penalty for short decision rules. For DGP 1, all models outperform LR, with only RF-RUX having overlapping confidence intervals with LR. 2-UPLTR has the best performance, followed by 1-UPLTR and Ada-RuleFit. For DGP 2, all models still outperform LR, with PLTR having the best performance followed by 2-UPLTR. For DGP 3, RF-RUX and Ada-RUX have lower performance than LR. 2-UPLTR remains the best performer among all models.

Finally, when using the third complexity score (Figure 5c), which increases the penalty for long decision rules, 2-UPLTR consistently has the highest performance. For DGP 1, all models outperform LR, with RF-RUX and Ada-RUX having overlapping confidence intervals with LR. 2-UPLTR has the best performance, followed by 1-UPLTR and Ada-RuleFit. For DGP 2, all models still outperform LR, with 2-UPLTR having the best performance followed by Ada-RuleFit and PLTR. For DGP 3, all models except RF-RUX and Ada-RUX outperform LR. 2-UPLTR again outperforms all others.

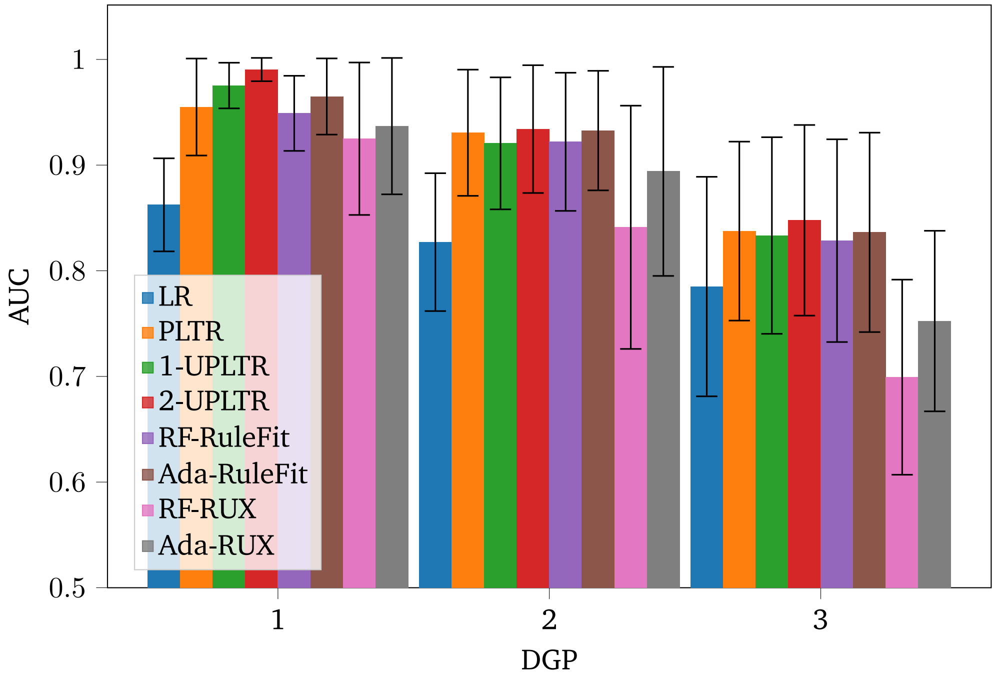
The performance of tree-based models without interpretability constraints is shown in Figure 6. From these ensemble models, it is observed that the boosting models consistently achieve the highest performance. The relative order of performance remains consistent across the DGPs, with RF consistently outperforming ET. However, the performance gap between RF and ET decreases for DGP 3. When comparing the ensemble models to the interpretable models, it is observed that the best-performing interpretable model, 1-UPLTR, performs similarly to the ensemble models. However, as the non-linearity in the underlying DGP increases, the gap in performance between the interpretable and ensemble models widens.



(a) Performance when using complexity score C_1.



(b) Performance when using complexity score C_2



(c) Performance when using complexity score C_3

Figure 5: Simulation performance for interpretable model in terms of AUC per DGP, with 95% confidence intervals based on 100 simulations.

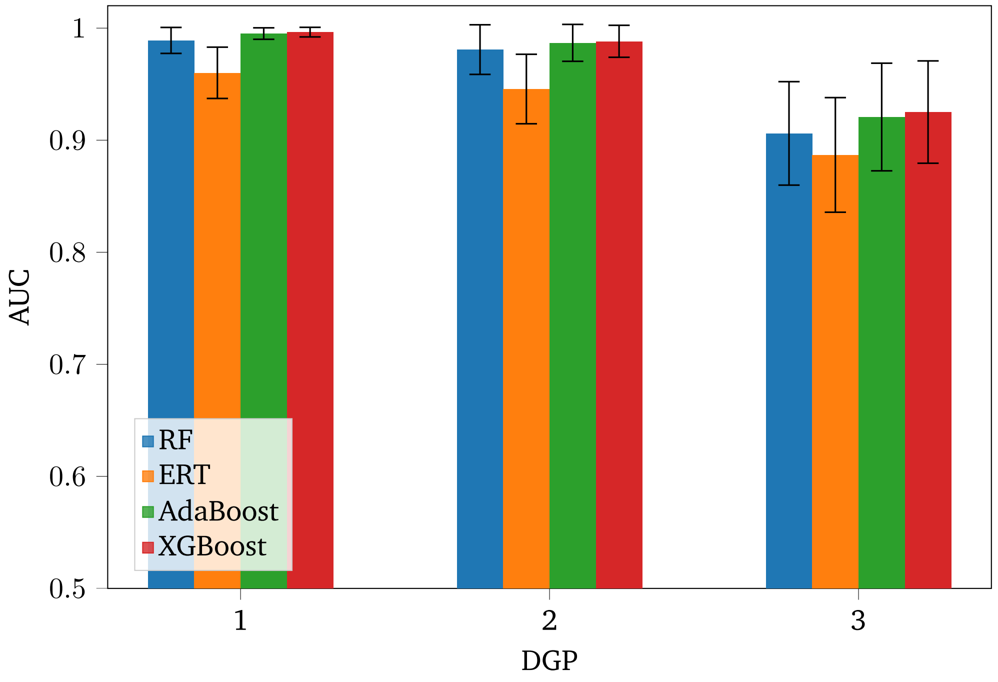


Figure 6: Simulation performance for ensemble models in terms of AUC per DGP, with 95% confidence intervals based on 100 simulations.

Table 3 presents the performance ranking for each model based on different complexity scores for each DGP. For DGPs 1 and 3, the ranking based on the baseline complexity score C_1 is consistent with the average ranking. However, for DGP 2, the ranking based on the baseline complexity score differs from the average ranking for 1-UPLTR and RF-RuleFit, which are ranked 4th and 5th respectively based on the baseline complexity score and vice-versa for the average ranking. Overall, the performance ranking of the models remains stable across the different complexity scores. Therefore, for the remainder of this thesis, we focus solely on the results based on the baseline complexity score C_1.

Table 3: Performance ranking of the models.

Ranking	LR	PLTR	1-UPLTR	2-UPLTR	RF-RuleFit	Ada-RuleFit	RF-RUX	Ada-RUX
				DGP 1				
C1	8	5	2	1	4	3	7	6
C2	8	5	2	1	4	3	7	6
C3	8	4	2	1	5	3	7	6
Average	8	5	2	1	4	3	7	6
				DGP 2				
C1	8	2	4	1	5	3	7	6
C2	8	1	5	2	4	3	7	6
C3	8	3	5	1	4	2	7	6
Average	8	2	5	1	4	3	7	6
				DGP 3				
C1	6	2	4	1	5	3	8	7
C2	6	2	4	1	5	3	8	7
C3	6	2	4	1	5	3	8	7
Average	6	2	4	1	5	3	8	7

Table 4 presents the Z-statistics for all models to test whether there is a significant difference in performance compared to the baseline model LR. We observe that among the interpretable models, only 2-UPLTR and RF-RuleFit significantly outperform LR for all DGPs. With the exception of the RUX models, all interpretable models outperform LR for DGPs 1 and 2. All ensemble models also outperform LR for all DGPs.

Table 4: Z-statistics for model performance in comparison to LR. **Bold** values indicate values that are significant at a level of 5%.

Model	DGP 1	DGP 2	DGP 3
PLTR	**2.82**	**3.66**	1.47
1-UPLTR	**5.50**	**3.38**	1.72
2-UPLTR	**6.62**	**4.29**	**2.33**
RF-RuleFit	**4.40**	**3.38**	**2.44**
Ada-RuleFit	**4.46**	**4.06**	1.93
RF-RUX	1.37	0.25	-1.59
Ada-RUX	1.87	1.06	-0.79
RF	**6.18**	**4.69**	**2.93**
ET	**5.05**	**4.07**	**2.67**
AdaBoost	**6.55**	**5.01**	**3.36**
XGBoost	**6.65**	**5.01**	**3.42**

Figure 7 displays the minimum complexity score required for each interpretable model to achieve an AUC that is at least as high as that of LR. We can see that as the underlying DGP increases in its degree of nonlinearity, the complexity scores of the models increase. Additionally, we observe that only 2-UPLTR is able to consistently achieve performance comparable to LR while maintaining a lower or equal complexity score. When examining the results for DGP 1, we see that several models, including 1-UPLTR, 2-UPLTR, RF-RUX and Ada-RUX, exhibit similar performance to LR while having a significantly lower complexity score. PLTR also has a lower complexity score, while RF-RuleFit and Ada-RuleFit have slightly higher scores. For DGP 2, we note that all models except RF-RUX have complexity scores that are lower than that of LR. In particular, PLTR, 1-UPLTR, 2-UPLTR, and Ada-RUX have low complexity scores. Finally, for DGP 3, we find that all models except 2-UPLTR are unable to keep their complexity scores below that of LR. Notably, RF-RUX and Ada-RUX have high complexity scores, exceeding even the complexity limit, which is set at twice the complexity score of LR.

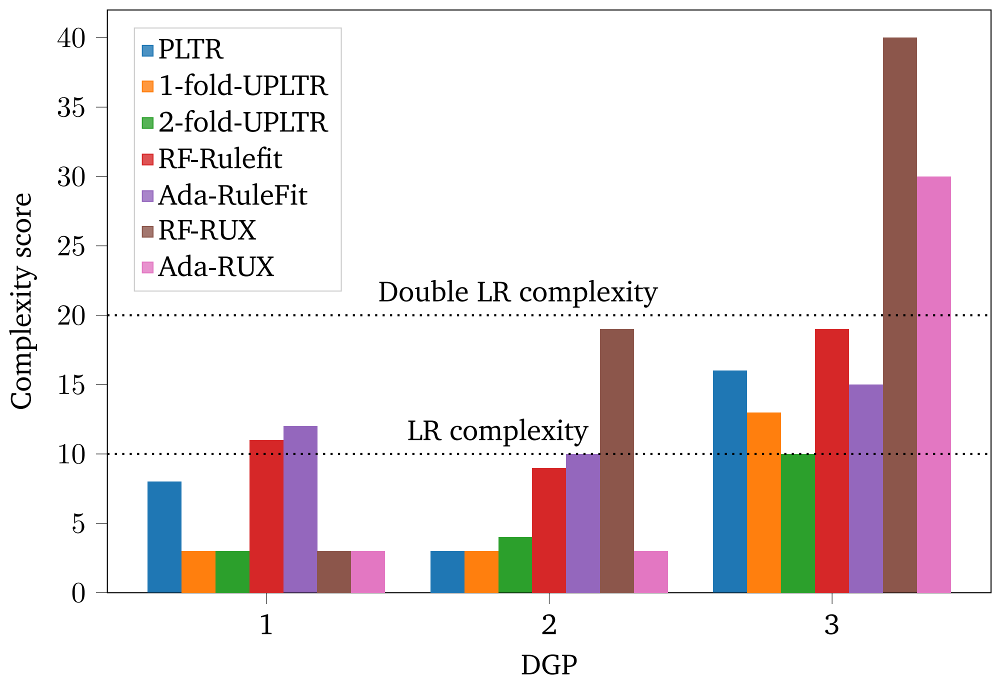


Figure 7: The minimum complexity score required for the interpretable models to achieve performance comparable to that of LR in terms of AUC.

7 Empirical study

In this section, we conduct an empirical evaluation of the models using real-world data. We compare the performance of the models in terms of AUC and BS, and demonstrate how their performance varies with different levels of complexity. Additionally, we provide an in-depth examination of the interpretability of the models.

7.1 Data

The data that we use consists of aggregated United States (US) state-level data with LendingClub's loan book covering the period from 2008 to 2019. As one of the largest peer-to-peer (P2P) lending platforms, LendingClub has amassed a total revenue of 1.2 billion USD in 2022[2]. The dataset includes 32 features and 2,703,430 observations, with 8% of the observations indicating defaulted loans. The target variable is a binary indicator of loan status, with a value of 1 indicating default and 0 indicating otherwise. This dataset provides a valuable opportunity to examine the P2P lending market within the context of macroeconomic variables, as it combines diverse loan, borrower, and state-specific features (Nigmonov et al., 2022). The explanatory variables are described in Table 5.

[2] https://ir.lendingclub.com/news/news-details/2022/LendingClub-Reports-Fourth-Quarter-and-Full-Year-2021-Results Date accessed: February 2, 2023.

Table 5: Description of the explanatory variables (Nigmonov et al., 2022).

Variable	Description
Loan specific variables	
AMOUNT	The total amount committed to the loan at that point in time.
INTRATE	The interest rate on the loan.
INQLAST	The number of inquiries in past 6 months.
OPENACC	The number of open credit lines in the borrower's credit file.
PUBREC	The number of derogatory public records.
DESLENGTH	The past-due amount owed for the accounts on which the borrower is now delinquent.
PCTTL	Percent of trades never delinquent.
TOTHI	Total high credit/credit limit.
Loan type variables (categorical)	
RATING	The assigned loan grade.
TERM	The number of payments on the loan. Values are in months and can be either 36 or 60.
PYMNTPLAN	Indicates if a payment plan has been put in place for the loan.
PURPOSE	A category provided by the borrower for the loan request.
TYPE	Indicates whether the loan is an individual application or a joint application with two co-borrowers.
INITIAL	The initial listing status of the loan.
Borrower specific variables	
INCOME	The self-reported annual income provided by the borrower during registration.
DTI	The average debt-to-income (DTI) score of borrower.
DELINQ	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years.
TAXLIENS	The number of tax liens.
EMPLENGTH	The employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
HOMEOWNER	The home ownership status provided by the borrower during registration or obtained from the credit report.
VERIFTYPE	Indicates if income was verified or not.
Economy specific variables	
EARNINGS	Average weekly earnings of all employees in each state (logarithm of values, monthly, in U.S. dollars).
UNEMP	Unemployment rate for each state (monthly, seasonally adjusted, percentage points).
NEWBUS	Share of established new businesses in total number of businesses in each state (monthly).
INFLATION	Monthly change in seasonally adjusted consumer price index (CPI) for all goods by state (percentage points, proxied by urban centres and U.S. regions).
MUNIRATE	One-year municipal bond yields for each state (monthly average of daily yield rates).

Table 5: Description of the explanatory variables (continued).

Variable	Description
GDPCONTRIB	Contributions to percentage change in real GDP (quarterly, percentage points).
RISKPREM	Risk premium on lending for banks in the USA (lending rate minus treasury bill rate, percentage points).
Demographic variables	
POPUL	Estimated population for each state (logarithm of population estimates reported for 2018).
INTUSER	Number of internet users at any location by state for each year from 2008 to 2016 (logarithm of values, yearly).
REP	Percentage of voters who voted for Republican candidate for each state (based on US Presidential election results 2008, 2012 and 2016).
RELIGIOUS	Percentage of adults who say they believe in God by state (time-invariant).

We preprocess the categorical features by applying one-hot encoding to the nominal features and ordinal encoding to the ordinal features. This ensures a linear interpretability of these features by creating binary indicator variables for each category, allowing for a clear and direct understanding of the impact of each category on the model's predictions.

0.27% of the data consists of missing values. We handle these missing values in the data by utilizing k-nearest neighbor imputation. Specifically, we first initialize the missing values of each variable as the mean. Then, we calculate the Euclidean distance between each of the observations. Starting from the feature with the most missing values, we iteratively impute the missing values using the average of the values from the k nearest neighbors that have a value for that feature. We choose a value of $k = 15$, as suggested by Troyanskaya et al. (2001). To ensure robustness, we repeat this process for 3 imputation rounds.

Moreover, to balance the representation of defaults and non-defaults in the data, we employ random undersampling. Specifically, we randomly select a subset of non-default observations to remove, until the number of non-defaults is equal to that of defaults. This reduces the number of observations to 433,244, while also reducing the computational cost of fitting the machine learning models.

In order to ensure the representativeness of the sample, we follow a similar procedure to Addo et al. (2018) by creating five distinct datasets, each of which is split into training, validation, and test sets in a proportion of 60%, 20%, and 20%, respectively. To mitigate computational limitations, we limit the size of each dataset to 5,000 observations. Additionally, we ensure that the ratio of defaults to non-defaults remains consistent across all splits. The use of multiple test sets allows us to evaluate the robustness of the models and test for significant differences in performance metrics. The specific hyperparameters used for tuning the models on the validation set are reported in Appendix A.

7.2 Results

Table 6 presents the performance of the interpretable models in terms of AUC and BS. The complexity score of each of the models is equal to that of LR. Both AUC and BS are used as performance metrics, with higher AUC and lower BS indicating better performance. The results indicate that 2-UPLTR outperforms the other models, followed by 1-UPLTR and Ada-RUX. All models, except for RF-RUX, outperform LR. Moreover, the AUC drives the difference in ranking between Ada-RuleFit, PLTR and Ada-RUX, as these models all have the same BS. This indicates that even though these models have similar accuracy, they have different discriminatory ability. Lastly, this ranking of interpretable models align slightly more with the ranking for DGP 1 in the simulation study, which contains a lower degree of nonlinearity, compared to DGPs 2 and 3.

Table 6: The average performance of the interpretable models.

Metric	RF-RUX	LR	RF-RuleFit	Ada-RuleFit	PLTR	Ada-RUX	1-UPLTR	2-UPLTR
AUC	0.749	0.757	0.806	0.811	0.813	0.817	0.825	0.832
BS	0.229	0.203	0.180	0.179	0.179	0.179	0.173	0.167
Ranking	8	7	6	5	4	3	2	1

Table 7 shows the performance of the ensemble models in terms of AUC and BS. Both metrics lead to the same ranking of the models, with XGBoost achieving the highest performance, obtaining a slightly higher AUC compared to AdaBoost, but with a particularly lower BS. The results are consistent with those of the simulation study, with XGBoost being followed by AdaBoost, RF and ERT, respectively. All ensemble models outperform the interpretable models.

Table 7: The average performance of the ensemble models.

Metric	ERT	RF	AdaBoost	XGBoost
AUC	0.841	0.872	0.878	0.879
BS	0.168	0.151	0.142	0.132
Ranking	4	3	2	1

Table 8 presents Z-statistics comparing the AUC performance of each combination of two models. It should be noted that due to multiple testing problems, these results should be interpreted with caution. The results show that all models, except for RF-RUX, significantly outperform LR. Furthermore, we find 2-UPLTR to significantly outperform PLTR and RF-RuleFit, while 1-UPLTR outperforms RF-RuleFit. We observe no other significant differences among the interpretable models. Among the ensemble models, RF is found to significantly outperform ERT, while ERT is found to significantly outperform PLTR, RF-RuleFit and Ada-RuleFit. Additionally, Adaboost and XGBoost are found to significantly outperform all models except for RF-RUX. RF-RUX is the only interpretable model that does not have any significant Z-statistics, likely due to its relatively volatile AUCs that are weakly correlated with the AUCs of the other models.

In Figure 8, we present the performance of various interpretable models across varying levels of complexity. We observe that 2-UPLTR exhibits the highest performance among all interpretable models, reaching its peak at a complexity level of 30 before gradually decreasing due to overfitting. At complexity scores below 60, 2-UPLTR is followed closely by 1-UPLTR and PLTR. As complexity increases, 1-UPLTR surpasses PLTR in performance. All PLTR variants consistently outperform LR. Beyond a complexity of 60, Ada-RuleFit obtains the highest performance, followed by Ada-RUX. RF-RuleFit and RF-RUX eventually outperform LR but never outperform the remaining models. The higher performance of AdaBoost variations in comparison to their RF counterparts aligns with previous findings that RF generally performs worse than AdaBoost. Additionally, we observe that the performance curves of PLTR variants are relatively stable, while the performance of other models increase significantly with increasing complexity. This can be attributed to the fact that PLTR variants only incorporate decision rules of maximum length two, thus limiting the flexibility of the model when fitting the data, in contrast to the other models that can incorporate longer decision rules.

Table 8: Z-statistics for model performance of the row model in comparison to the column model in terms of AUC. **Bold** values indicate values that are significant at a level of 5%. Positive values indicate that the row model outperforms the column model.

	LR	PLTR	1-UPLTR	2-UPLTR	RF-RuleFit	Ada-RuleFit	RF-RUX	Ada-RUX	RF	ERT	AdaBoost
PLTR	**9.16**										
1-UPLTR	**5.13**	1.10									
2-UPLTR	**7.11**	**2.05**	0.79								
RF-RuleFit	**5.69**	-0.86	**-3.73**	**-3.07**							
Ada-RuleFit	**2.93**	-0.14	-1.21	-1.64	0.34						
RF-RUX	-0.16	-1.07	-1.27	-1.28	-0.97	-1.09					
Ada-RUX	**2.10**	0.15	-0.28	-0.61	0.40	0.18	0.79				
RF	**14.48**	**7.30**	**3.82**	**8.24**	**6.59**	**3.62**	1.85	**2.34**			
ERT	**11.43**	**5.38**	1.46	1.00	**4.30**	**2.51**	1.59	0.76	**-3.44**		
AdaBoost	**13.15**	**6.49**	**3.89**	**6.67**	**6.35**	**3.43**	1.88	**2.97**	**1.97**	**3.13**	
XGBoost	**13.17**	**7.02**	**4.64**	**8.31**	**7.48**	**3.70**	1.92	**3.00**	**2.16**	**3.39**	0.47

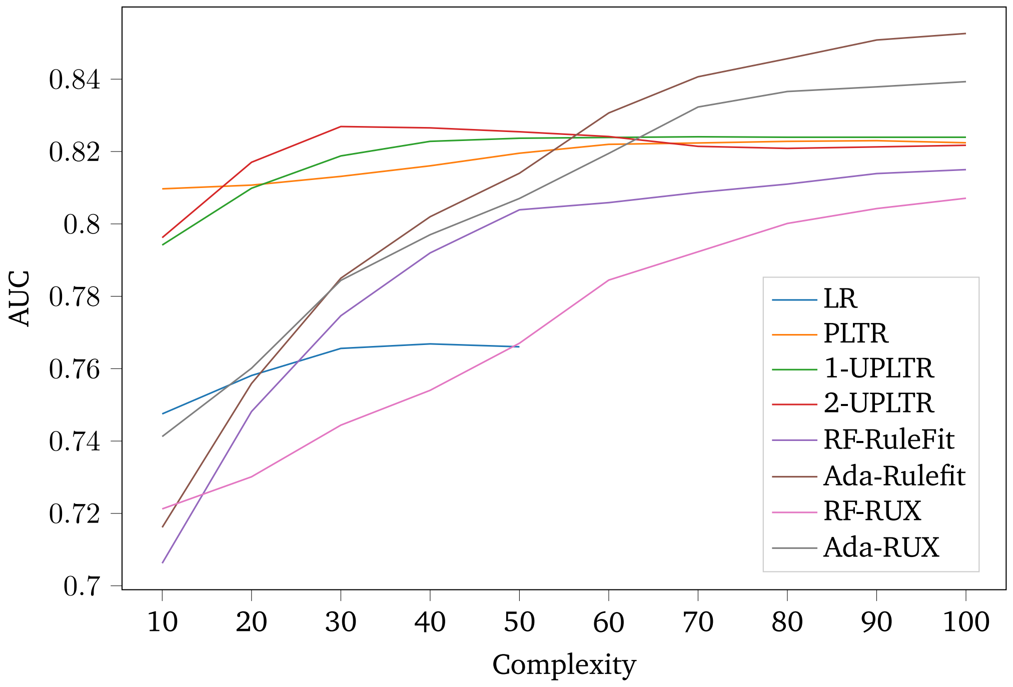


Figure 8: The performance of the interpretable models in terms of AUC for different levels of complexity.

7.3 Feature interpretation

In this section, we conduct an analysis of feature importance for all interpretable models using one of the data sets, with a complexity score of 10. The relative importance of each feature is quantified by their standardized regression coefficients, and the results are presented in tables 9 - 16. The findings reveal that the majority of models contain features related to INFLATION, RATING, and RISKPREM. These feature rankings can be employed to interpret individual predictions, as they allow us to comprehend how changes in specific feature values affect the predicted probability of default (PD). This information can assist banks in identifying the main drivers of default for individuals with a high PD, and can also provide borrowers with deeper insight into their credit score.

As shown in Table 9, the RATING feature is the most significant risk driver for LR. This is expected, as RATING is the assigned loan grade to the borrower, and a higher value for RATING indicates a higher risk of default. Additionally, features such as UNEMP, which represents the unemployment rate for each state, and INFLATION, which represents the inflation rate, are positively related to the predicted PD. This is because a higher unemployment rate or inflation rate can indicate an adverse economic environment, increasing the likelihood of borrower default. Conversely, the feature NEWBUS, which represents the share of established new businesses relative to the total number of businesses in each state, is negatively related to the PD. This is because an increase in the share of established new businesses can indicate an improvement in the economy, resulting in a decrease in the PD.

Table 9: Feature ranking of LR. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	🟩 RATING	22
2	🟩 UNEMP	16
3	🟥 NEWBUS	15
4	🟩 INFLATION	13
5	🟥 INTRATE	8
6	🟩 RISKPREM	8
7	🟩 VERIFTYPE = VERIFIED	5
8	🟥 TOTHI	5
9	🟥 MUNIRATE	4
10	🟥 TYPE = INDIVIDUAL	4

The feature rankings for the PLTR variations are presented in Tables 10 - 12. The models show that RATING is the most influential feature and the top three drivers are mainly responsible for the predictions. For PLTR, the decision rule 0.18 <INFLATION <= 0.55, which represents a specific range of high inflation levels, has a positive relationship with the PD. Additionally, the decision rule 15.09 <INTUSER <= 15.10, which indicates a specific number of internet users per state, has a negative relationship with the predicted PD. This decision rule indicates a relatively high number of internet users and applies to 46% of the observations in the data set. However, this decision rule may be influenced by omitted variable bias as the data set does not include information on the state of residence of the borrower. In 1-UPLTR, the decision rule INFLATION <= 0.18 has a negative impact on the PD, consistent with the previously established relationship between inflation and PD in PLTR. Furthermore, RISKPREM, which is the risk premium on lending for banks, has a positive impact on the PD as it suggests that borrowers in states with high risk premia may have difficulty repaying their loans. Lastly, in 2-UPLTR, the decision rule RISKPREM >3.21 & INTUSER <= 15.11 has a negative impact on the PD, indicating that borrowers in states with high risk premia and moderate levels of internet users may have a lower PD.

Table 10: Feature ranking of PLTR. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite.The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	🟩 RATING	47
2	🟩 0.18 <INFLATION <= 0.55	28
3	🟥 15.09 <INTUSER <= 15.10	20
4	🟥 RISKPREM >3.21 & TYPE = INDIVIDUAL	2
5	🟥 INTRATE<= 0.12	1
6	🟩 inf >0.00 & TYPE = INDIVIDUAL	1

Table 11: Feature ranking of 1-UPLTR. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	impact (%)
1	🟩 RATING	45
2	🟥 INFLATION <= 0.18	27
3	🟩 RISKPREM	11
4	🟩 INTUSER <= 15.09	5
5	🟥 RATING <= 2.50	4
6	🟥 RISKPREM >3.21 & INTUSER <= 15.11	3
7	🟥 RISKPREM >3.21 & inf <= 0.09	3
8	🟥 GDPCONTRIB <= 0.03	2

Table 12: Feature ranking of 2-UPLTR. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	🟩 RATING	34
2	🟥 INFLATION <= 0.18	26
3	🟥 RISKPREM >3.21 & INTUSER <= 15.11	20
4	🟥 RATING <= 2.50	9
5	🟩 INTUSER <= 15.09	4
6	🟥 RISKPREM >3.21 & Verified <= 0.50	3
7	🟥 RISKPREM	3
8	🟥 GDPCONTRIB <= 0.03	1

Tables 13 - 14 present the feature rankings for the RuleFit models. In contrast to previous models, the impact of features in RF-RuleFit is relatively evenly distributed. The primary driver is the decision rule 0.16 <INFLATION <= 0.18 & INTUSER >15.09, which has a negative effect on the PD. This rule pertains to a moderate level of inflation and a high number of internet users. In the case of Ada-RuleFit, the features are generally shorter as no feature comprises more than two conditions. Also, the first five features are responsible for the majority of the prediction.

Table 13: Feature ranking of RF-RuleFit. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	0.16 <INFLATION <= 0.18 & INTUSER >15.09	22
2	NEWBUS	17
3	RATING	17
4	INFLATION >0.18	16
5	UNEMP	15
6	INFLATION <= 0.16 & INTRATE >0.12 & INTUSER <= 15.09	13

Table 14: Feature ranking of Ada-RuleFit. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	RATING	29
2	INFLATION >0.18	14
3	NEWBUS	13
4	UNEMP	12
5	INFLATION >0.14& INFLATION <= 0.18	11
6	INTRATE	8
7	INFLATION	6
8	TYPE = INDIVIDUAL	3
9	TOTHI	2

the RUX models. For both models, there are no dominant features in terms of impact, as the impact of the features gradually decreases along the ranking. In the case of RF-RUX, the decision rule 10.90 <DTI <= 11.00 is the primary driver, which is negatively related to the PD. DTI represents the average debt-to-income ratio of the borrower. As the decision rule pertains to a relatively low debt-to-income ratio, it indicates a less risky borrower and, thus, results in a lower PD. For Ada-RUX, the decision rule 0.14 <INFLATION <= 0.16 is the primary driver, which is positively related to the PD. The second most important feature is the decision rule TOTHI <= 16641.00 & GDPCONTRIB >0.08, which is positively related to the PD. TOTHI represents the credit limit for the borrower and GDPCONTRIB represents the contribution to the percentage change in the real GDP. As a low credit limit indicates that the bank perceives the borrower as too risky to grant a larger loan, it makes sense that this results in an increased PD. It appears that this effect is magnified if the borrower resides in a state with a high GDP contribution.

Table 15: Feature ranking of RF-RUX. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	🟥 10.90 <DTI <= 11.00	21
2	🟥 INFLATION>0.55	20
3	🟥 RISKPREM >3.21 & INFLATION <= 0.18	17
4	🟩 INFLATION >0.09 & RISKPREM>3.21	16
5	🟥 PURPOSE = WEDDING >0.50 & INFLATION >0.18	14
6	🟥 INFLATION <= 0.18	12

Table 16: Feature ranking of Ada-RUX. Green indicates a feature which is positively related to the probability of default, whereas red indicates the opposite. The impact of each of the features on the final predicted PD is given in percentage points and is based on the feature coefficients.

Ranking	Feature	Impact (%)
1	🟩 0.14 <INFLATION <= 0.16	26
2	🟩 TOTHI<= 16641.00 & GDPCONTRIB >0.08	23
3	🟥 INFLATION >0.55	16
4	🟥 AMOUNT <= 1950.00	12
5	🟥 PURPOSE = MEDICAL >0.50 & NEWBUS<= 15788.50	12
6	🟥 INTRATE<= 0.06	12

Table 17 presents the top five drivers for ensemble models. Unlike previous models, which possess inherent interpretability, it is not possible to demonstrate the direct relationship between features and internally generated decision rules with the PD due to a lack of transparency. Therefore, we only display the most significant original features for the models. All models have INFLATION as their primary driver. RISKPREM and INTRATE are the next most crucial drivers. INFLATION and RISKPREM were also the most vital underlying drivers for interpretable models.

Table 17: Top five drivers of ensemble models. **Bold** features indicate features that are present across all models.

Ranking	RF	ERT	AdaBoost	XGBoost
1	**INFLATION**	**INFLATION**	**INFLATION**	**INFLATION**
2	**INTRATE**	**RISKPREM**	**RISKPREM**	**INTRATE**
3	**RISKPREM**	RATING	RATING	TOTHI
4	INTUSER	**INTRATE**	**INTRATE**	DTI
5	RATING	INTUSER	TOTHI	**RISKPREM**

8 Conclusion

In this thesis, we have examined the use of various interpretable machine learning models for credit scoring, and compared their performance to that of the industry standard Logistic Regression (LR). We have evaluated variations of Penalised Logistic Tree Regression (PLTR), RuleFit, and Rule eXtraction (RUX), as well as tree-based ensemble models: Random Forest (RF), Extremely Randomized Trees (ERT), Adaptive Boosting (AdaBoost), and eXtreme Gradient Boosting (XGBoost). Additionally, we have introduced two unbiased versions of PLTR, namely 1-fold Unbiased PLTR (1-UPLTR) and 2-fold Unbiased PLTR (2-UPLTR), and specifically analysed the performance of RF-RuleFit, Ada-Rulefit, RF-RUX and Ada-RUX versions of RuleFit and RUX. To facilitate the comparison of the interpretable models, we have introduced a complexity score, which can be used to evaluate the complexity of rule-based models relative to that of LR.

Through a simulation study, we have employed three data generating processes (DGPs) based on different degrees of underlying non-linearity to compare the performance of the models in terms of the area under the receiver operating characteristic curve (AUC). Our findings indicate that 2-UPLTR is the best performing interpretable model for different variations of complexity scores and DGPs. We have also found that 2-UPLTR and RF-RuleFit are the only interpretable models that consistently outperform LR, while all ensemble models consistently outperform LR.

In addition, we have conducted an empirical study using aggregated United States state-level data with LendingClub's loan book covering the period from 2008 to 2019. Our results reveal that when ranking the interpretable models in terms of AUC and Bries Score (BS), 2-UPLTR was the best performing model, followed by 1-UPLTR and Ada-RUX. All models were ranked higher than LR except RF-RUX. Furthermore, all ensemble models were ranked higher than the interpretable models, particularly the boosting models. In terms of statistical significance, we have found that 2-UPLTR significantly outperforms both PLTR and LR.

We have also provided an overview of the performance of the interpretable models for different levels of complexity. We have found that 2-UPLTR obtains the highest performance when the model complexity is limited to that of LR. However, for higher complexity levels, Ada-RuleFit and Ada-RUX were found to obtain the highest performance. Additionally, we have demonstrated how the models can be interpreted, showing the relationship of features with the predicted probability of default and their impact on the final prediction, which allows for a check on whether the features are used in the model in alignment with economic reasoning. We found that most interpretable models use features related to inflation, the loan grade of borrowers assigned by LendingClub, and the risk premium on lending for banks.

In conclusion, we recommend that banks use intrinsically interpretable machine learning for credit scoring, as it provides direct interpretability for stakeholders, including local interpretability for borrowers and global interpretability for regulators. For banks that prefer models that are not more complex than LR, we recommend using 2-UPLTR. For banks that allow for models that can exceed LR in terms of complexity, we recommend considering Ada-RuleFit and Ada-RUX. However, it is possible that banks may prefer a complexity score that is substantially different from our proposed complexity score, in which case other models may be more suitable.

For future research we recommend conducting a survey study among credit managers of banks regarding their ability to understand decision rules of different lengths. This would allow creating a complexity score which is truly tailored to human cognitive behaviour. As financial regulators and borrowers are also stakeholders of model interpretability, such a survey could be extended to include them as well. Moreover, in our simulation study, we were limited to using only three different complexity scores and DGPs on a relatively small data set due to computational constraints. In future research this simulation could be extended.

References

P. M. Addo, D. Guegan, and B. Hassani. Credit risk analysis using machine and deep learning models. *Risks*, 6(2):38, 2018.

M. H. Akyüz and S. I. Birbil. Discovering classification rules for interpretable learning with linear programming. *CoRR*, abs/2104.10751, 2021. URL https://arxiv.org/abs/2104.10751.

D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(4):1059–1086, 2020.

B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.

M. Bramer. Avoiding overfitting of decision trees. *Principles of Data Mining*, pages 119–134, 2007.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Routledge, 1984.

N. Bussmann, P. Giudici, D. Marinelli, and J. Papenbrock. Explainable machine learning in credit risk management. *Computational Economics*, 57(1):203–216, 2021.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

E. Dumitrescu, S. Hue, C. Hurlin, and S. Tokpavi. Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *European Journal of Operational Research*, 297(3):1178–1192, 2022.

Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The annals of applied statistics*, 2(3):916–954, 2008.

B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a “right to explanation”. *AI magazine*, 38(3):50–57, 2017.

B. R. Gunnarsson, S. Vanden Broucke, B. Baesens, M. Óskarsdóttir, and W. Lemahieu. Deep learning for credit scoring: Do or don’t? *European Journal of Operational Research*, 295(1):292–305, 2021.

J. A. Hanley, B. J. McNeil, et al. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983.

S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136, 2015.

T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

A. Nigmonov, S. Shams, and K. Alam. Macroeconomic determinants of loan defaults: evidence from the us peer-to-peer lending market. *Research in International Business and Finance*, 59:101516, 2022.

A. Petropoulos, V. Siakoulis, E. Stavroulakis, A. Klamargias, et al. A robust machine learning approach for credit risk analysis of large loan level datasets using deep learning and extreme gradient boosting. *IFC Bulletins chapters*, 49, 2019.

N. Razavian, S. Blecker, A. M. Schmidt, A. Smith-McLallen, S. Nigam, and D. Sontag. Population-level prediction of type 2 diabetes from claims data and analysis of risk factors. *Big Data*, 3(4): 277–287, 2015.

M. T. Ribeiro, S. Singh, and C. Guestrin. “Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

I.-C. Yeh and C.-h. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2):2473–2480, 2009.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

H. Zou and T. 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 Hyperparameter tuning

Table A1 displays the hyperparameters that are tuned for the models. All hyperparameters are tuned on the validation set. The candidate values are created around the default values for the hyperparameters.

Table A1: The tuned hyperparameters per model and their candidate values.

Model	Hyperparameter	Type	Lower	Upper	Step Size
LR	elastic net	numeric	0	1	0.1
PLTR	lasso	numeric	0.01	0.5	0.0025
1-UPLTR	lasso	numeric	0.01	0.5	0.0025
2-UPLTR	lasso	numeric	0.01	0.5	0.0025
RF-RuleFit	tree size	integer	2	10	1
Ada-RuleFit	tree size	integer	2	10	1
RF-RUX	maximum depth	integer	2	10	1
Ada-RUX	maximum depth	integer	2	10	1
RF	number of trees	integer	100	1500	200
ET	number of tress	integer	100	1500	200
AdaBoost	number of trees	integer	100	1500	200
XGBoost	number of trees	integer	100	1500	200