uc3m | Universidad Carlos III de Madrid

Master's Degree in Statistics

2023-2024

Master's Thesis

Interpretable Machine Learning for Credit Scoring

Juan Antonio Montero de Espinosa Reina

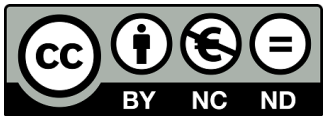
Javier Carrasco Serrano

Rebeca Peláez Suárez

Madrid, 2024

AVOID PLAGIARISM

The University uses the **Turnitin Feedback Studio** for the delivery of student work. This program compares the originality of the work delivered by each student with millions of electronic resources and detects those parts of the text that are copied and pasted. Plagiarizing in a TFM is considered a **Serious Misconduct**, and may result in permanent expulsion from the University.



This work is licensed under Creative Commons **Attribution – Non Commercial – Non Derivatives**

SUMMARY

Logistic regression is the leading technique used to predict the probability of default in credit scoring models (Siddiqi, 2017), with strong support in both policy (Knutson, 2020) and academic (e.g., Wiginton, 1980, Thomas, 2000) circles. This Master's thesis explores the trade-off between discriminatory power and interpretability in credit scoring based on logistic regression and interpretable machine learning (ML) techniques. First, a classical credit scoring analysis - based on logistic regression - is constructed as a baseline following Siddiqi, 2017 and Carrasco Serrano, 2023. This yields both a traditional credit scorecard and a framework for interpretability. Additionally, the trade-off between discriminatory power and interpretability for ML tree-based models is examined. Of these models, the XGBoost model outperforms logistic regression in discriminatory power. Then, limitations for interpretability in credit scoring of general tree-based models are discussed. The analysis demonstrates that scorecards similar to traditional credit scorecards can be constructed from constrained yet powerful tree-based models (e.g., XGBoost). Finally, the thesis compares these results against those obtained from model-agnostic interpretability methods - counterfactuals, individual conditional expectation plots and polynomial representations - in the context of neural networks.

Keywords: credit scorecard, credit scoring, discriminatory power, interpretability, interpretable machine learning, machine learning, model-agnostic interpretability methods, neural networks, tree-based models, XGBoost

DEDICATION

A mis padres, por tanto que me habéis dado y me seguís dando, que va dando su fruto.

A Carla, porque, a pesar de la distancia, has sabido hacerme llegar tu apoyo como si estuviéramos cerca.

A Rafa, por nuestros paseos por Madrid (y las partidas de ajedrez) tan necesarios para, entre el bullicio, tomar aliento.

A mis supervisores por vuestra profunda dedicación y atención para con este trabajo, que desde el primer momento me han animado a perseverar. En particular, por las extensas reuniones en las que tanto he aprendido y que son el terreno sobre el que se ha desarrollado esta tesis.

Por último, me gustaría agradecer a los muchos proyectos de código abierto sin los cuáles estudios de Ciencia de Datos como este serían inviables.

CONTENTS

1. INTRODUCTION

The word 'credit' originates from the Latin *credere* meaning belief or trust. When a lender lends money to a borrower, the lender must trust the borrower to return the principal plus interest. But how can the borrower know which requests for credit they should trust? Credit scores essentially provide a numerical answer to this question by ranking credit applications using observed related features to estimate the borrower's creditworthiness. One of the main applications of these scores, among others such as risk-based pricing, is to establish a threshold over which an application is considered trustworthy enough to be accepted (Petrides et al., 2022).

This Master's thesis studies a promising step forward in the evolution of scoring for credit applications. By using interpretable machine-learning (ML) techniques in the era of Big Data, credit practitioners can better extract the patterns that make a client reliable without sacrificing much interpretability, if any. This research proposes that, at the very least, interpretable ML models should become additional options in a credit modeller's toolkit. Kim et al., 2016 define interpretability[1] as a property of a method that enables users to correctly and efficiently predict the output of the method and, thus, to explain it correctly to stakeholders. However, any loss in explainability must be fully justified given the criticality of credit scoring models for the solvency of lending institutions and, ultimately, of the financial system. This is the basis for regulatory constraints that limit the use of complex, non-interpretable models, as every decision must be explainable to the regulator and the client, as noted in Altman et al., 1994, Lucas and Jurgovsky, 2020, and Knutson, 2020.

A brief historical review of credit scoring based on Thomas, 2000 can help motivate the utility of interpretable ML models. Originally, decisions on whether to grant a loan were made by credit analysts using rules based on their own judgment and experience. This practice gave rise to the five C's for qualitative credit analysis: character (payment history), capacity (income), collateral, conditions (e.g., prospects for the economy), and capital ('skin in the game'). It was not until World War Two in the United States that, amidst a shortage of credit analysts, the few remaining available experts wrote down their practical rules for credit granting. Based on these rules, an expert-driven system was developed, whereby non-experts would apply the written rules as guidance for credit decisions. A decade later, in 1956, Fair and Isaac created the first credit scoring consultancy: the Fair and Isaac Company, nowadays widely known for their popular FICO score.

Durand, 1941 was the first to realise that Fisher, 1936's statistical work on differentiating varieties of iris flowers could be employed for credit scoring purposes. As a signal of creditworthiness, Durand proposed to separate past applications into two classes, in his own words: "those with satisfactory repayment experience and those entailing seri-

[1]The term *explainability* is used interchangeably with *interpretability* across this work.

ous collection difficulties"[2]. Nowadays, the former class is called 'good' (non-default) applications and the latter 'bad' (default) applications[3]. The actual definition of bad is frequently adjusted to fit the context and application (Siddiqi, 2017). The developer must consider both the default event (typically, a delay in payment longer than 90 days as in Petrides et al., 2022) and the horizon (e.g., a 12-month period), yielding a classification problem. The probability of an application being bad in this classification problem is known as the probability of default (PD), which is to be modelled with observed features available at the time of application. The present work follows common industry practice and rather focuses on the capacity of algorithms to statistically separate between goods and bads, also known as discriminatory power or ability.

Once Statistics started being used to optimise and create credit allocation rules, there was no turning back. The advent of credit cards in the 1960s was the definitive nurturing ground for the expansion of credit scoring: the massive size of the credit card market required automation of the decision process; and credit scores were available to meet this requirement, improving results at the same time. Eventually, credit scores started being constructed as the logit transformation of the PD as introduced in Thomas et al., 2002 and Verbraken et al., 2014. Subsequently, credit scoring models ranked applications using these scores, separating between goods and bads by means of a continuous order[4]. Over time, logistic regression became the reference model for estimation of credit scores (Siddiqi, 2017) thanks to its simplicity, interpretability, and robustness. The logistic regression model for credit scoring has been extensively studied in the literature by Wiginton, 1980, Thomas, 2000, and Siddiqi, 2017, among others.

Nonetheless, many studies that find ML techniques that outperform logistic regression already exist. For instance, Lessmann et al., 2015 carry out a systematic review of modelling options for credit scores, with the logistic regression model ranking in the middle (20^{th} out of 41) for the Receiver Operating Characteristic - Area Under the Curve (ROC AUC) statistic. On top of that, Big Data is likely to widen the gap between logistic regression and ML techniques, since the latter better exploit the increasing variety and volume of features in new datasets. This research contributes to the existing literature by applying ML tools to the Home Credit (HC) dataset, with some of the models demonstrating better discriminatory power than the logistic regression model.

But why are interpretable ML tools needed, especially if standard ML techniques already tend to outperform traditional ones? Molnar, 2022 shares a narrative ('Trust Fall') where a failure in a scoring system deemed infallible casts a citizen out of society for no good reason. In the context of credit, non-interpretable models could lead to financial exclusion in similarly obscure ways. In addition, the financial crisis of 2007/2008 arose,

[2]This introduces a well-known selection bias in the sample, since only accepted applications are examined. *Reject inference* is used in the credit industry to tackle this issue, but it falls outside the scope of this work.

[3]From here on referred to as 'goods' and 'bads', respectively.

[4]Thus, the goal of credit scoring is not as much finding a perfect separation between goods and bads, but establishing a continuous ranking of creditworthiness.

amongst other causes, as a consequence of excessive subprime lending. A subprime mortgage in the US is precisely characterised by a too low credit score - a warning that fell on deaf ears. As a result of this crisis, more stringent supervision crystallised in more transparent and auditable processes for developing models (Siddiqi, 2017). Therefore, scoring models must both display a large discriminatory power and meet some interpretability standards in order to prevent financial exclusion and guarantee the auditability and transparency of credit processes. This further supports why interpretable ML models show large promise for credit scoring.

There is already growing literature on the application of interpretable ML for credit scoring. This thesis explores two main research lines: using an intrinsically interpretable model, and applying model-agnostic *ex-post* interpretability techniques. As an example of the former, Dumitrescu et al., 2022 developed the Penalised Logistic Regression Tree, a two-step classifier that first builds decision trees and then selects them using adaptive lasso logistic regression. Bracke et al., 2019 apply clustering methods and a model-agnostic mechanism to explore the drivers of predictions in black-box models for credit, with mixed results. Finally, an influential reference for the development of ML-based scorecards in this study has been the ideas advanced for AdaBoost by Edwards, 2023.

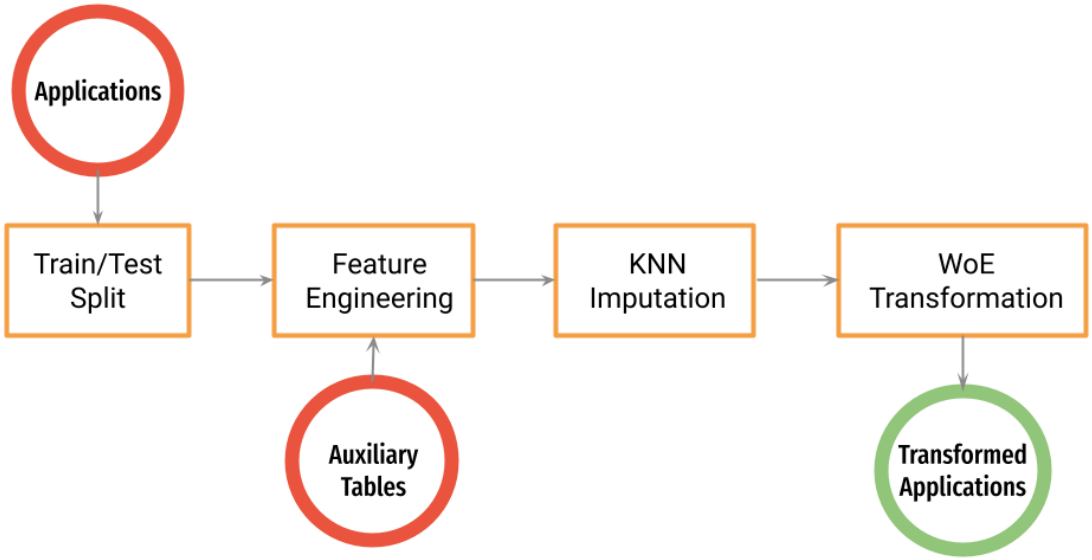
The remainder of this thesis is structured as follows. Chapter 2 delves straight into discussing and processing the HC dataset used for the empirical analysis. Chapter 3 develops a classical credit scoring exercise following Siddiqi, 2017 and Carrasco Serrano, 2023, serving as a baseline for discriminatory power and explainability. Chapter 4 discusses the interpretability and discriminatory power of tree-based models, and formalises and implements methods to develop credit scorecards from additive tree-stump-based models. Chapter 5 covers model-agnostic tools mainly applied to neural networks. Lastly, Chapter 6 draws conclusions from this work and proposes future lines of research. The source code for this project is hosted on the GitHub project Interpretable Credit Scoring.

2. DATASET AND PREPROCESSING

The data used in this thesis is the Home Credit Default Risk dataset available from Kaggle (2018). This dataset is one of the largest real-world datasets for credit scoring currently freely available on the internet [5]. The data is made available with a relational design, with the main table for applications under evaluation containing 307,511 credit (generic) applications. Auxiliary tables in the dataset are bureau applications, previous HC applications, credit card status, and cash balances. These tables are used to extend the features available for training, emulating an actual credit scoring exercise.

The rest of the chapter introduces the target and predictor features in the dataset, followed by a discussion on engineered features, imputation, the WoE transformation, and a correlation study. Figure 2.1 graphically displays a summary of the data transformations applied.

Figure 2.1
Overview of data preprocessing. Circles represent datasets (red for raw, green for processed) and rectangles the data transformations.



2.1. Analysis of Target

In this dataset, bads are identified directly using the binary variable Target. Unfortunately, the description of the dataset does not provide neither the definition of default nor

[5]This dataset was originally shared for a Kaggle competition without an Academic license. Nonetheless, the underlying company Home Credit, now EmbedIT, has granted the author permission for academic use of the data directly via email.

the horizon for this variable. Typically, the horizon follows the recommended 12-month horizon in Basel II with 90 days past due being used as a measure of delay. Ultimately, the definition of bads together with its relevance is borrowed from the data provider.

A train-test random split is applied using the industry norm of 70% for train and 30% as a holdout sample for testing[6] with stratified sampling by the target variable following Siddiqi, 2017. Table 2.1 displays the number of observations and bads in the train and test samples. This shows that the percentage of bads is close in train and test as a direct result of the stratified sampling.

	Train	Test	Total
N	215,257	92,254	307,511
N Bads	17,377	7,448	24,825
% Bads	8.07	8.07	8.07

Table 2.1
Total observations and total and percentage of bads in train and test splits.

On the one hand, Crone and Finlay, 2012 find that for credit scoring based on logistic regression (as in Chapter 3), at least 5,000 bad applications are recommended to build application scorecards. On the other hand, Siddiqi, 2017 is more lenient in this respect, recommending a minimum of 2,000 bad (and good) applications. In any case, the split applied meets these recommendations for a robust credit study by a wide margin.

Next, the features used in this research are introduced, starting with those already available for analysis and then moving on to the engineered features that required additional preprocessing.

2.2. Available Features

Available features in the main applications table can be divided into loan- or client-level features. The loan-level has many features, including the credit amount, the annuity (normalised installment), whether the loan is cash or revolving, and flags for documents presented.

At the client-level, there are about 90 features that can be classified into:

- General features: Gender, age, education, family information (status, number of children), region, etc.
- Financial: Amount of income, type of income (including unemployed), type of occupation/organisation, days on the current job, number of enquiries to credit bureau (using different time frames), external ratings, etc.

[6] Alternatively, a 80%/20% train/test split would also have been a valid option, leading to more information used for training but less for testing.

- Housing-related: owner/renter, normalised facts about housing (46 variables, the mean, median, and mode of many elements, from number of apartments to land area).
- Socially-related: number of defaults in social circles.

Given the considerable volume and variety, this dataset is a good representation of currently available datasets. An additional Excel file is available upon request with summary statistics at the column level plus additional metadata.

2.3. Engineered Features

Some of the engineered features required merging several tables in the dataset, while others are based on features available in the main applications table. The former features complement the data at the client level, while the latter do so at the loan level.

2.3.1. Loan Level

These only require access to the main table with applications. They include loan-to-value (value of the good[7] divided by the credit amount), a lower bound for credit length (credit amount divided by annuity), credit to income, and categorical predictors for daytime (morning, afternoon, evening, and night) and workday (workday, weekend).

2.3.2. Client Level

These features (a total of 20) require the application data to be merged with other auxiliary tables, and they are linked to the credit history of the client. They are listed next:

- Time since earliest credit provided separately for bureau and previous HC applications.
- Time since last previous credit for bureau and previous applications.
- Number of credits contracted during the last year for bureau and previous applications.
- Numbers of active credits for bureau and previous applications.
- Average credit card consumption based on credit card balances.
- Count of times that a credit card exceeded the credit limit (for last month and quarter).

[7]This feature is available for 99.91% of observations, so the dataset likely mostly consists of consumer credit applications.

- Total credit exposure to income, where the total exposure only considers active credits.
- Recent bad credits. For bureau credits, number of loans overdue more than 30 or 60 days at the time of the application. For previous HC applications, number of times that a loan was overdue more than 30 days (with small loan amounts being ignored) for different time periods.

2.4. Treatment of Missing Data

The presence of missing values in the predictor features requires careful data preprocessing to prevent losing valuable information for scoring. To this end, the significance of missing values on bad applications is studied. For the subset of variables with *non-significant* missing values, imputation by K-Nearest-Neighbours (KNN) Regression is employed to preserve these observations. Conversely, *significant* missing observations are dealt with directly with the Weight of Evidence (WoE) transformation, that assigns a WoE value to the missing observations in line with their effect on default[8]. The WoE transformation is discussed in more depth in the next section.

To analyse the significance of missing values, a regression of the target on a dummy indicator of missing value for each predictor is run, and the p-value on the coefficient of the dummy retrieved. For most variables, the output p-value is zero with three digits of precision (rounded), implying that missing values for that predictor have a significant effect on default at least for any confidence level larger than 0.05%.

Variable	N Missing	P-Value
External Scoring 2	469	0.752
Loan-to-Value	202	0.937
Asset Value	202	0.937
Loan Annuity	8	0.995
Credit Length (Lower Bound)	8	0.995
# Family Members	2	0.967
Days Last Phone Change	1	1.000

Table 2.2
Features with non-significant missing values.

Table 2.2 displays the exceptions to that finding, which are characterised by a relatively small number of missing observations. The missing values of these features are treated by KNN Imputation as implemented in the R library `recipes` (Kuhn et al., 2024). KNN Imputation is a robust technique to replace missing values that relies on finding the

[8] Similar to what a Target Encoder does in Scikit-learn.

closest neighbour to the observation with missing values. The closest neighbour is found by computing the distance between observations based on the intersection of observed features for each pair of observations. This last aspect makes the methodology especially robust for imputation purposes.

Next, the WoE transformation is presented, which is a convenient encoding technique that helps address significant missing values, amongst other issues.

2.5. Weight of Evidence (WoE) Transformation

Siddiqi, 2017 proposes the WoE transformation as a core data processing step for credit scoring. The WoE transformation is essentially an encoding technique that can be applied to both numerical and categorical values to obtain groupings that have a significant effect on default, coding each resulting grouping with its WoE value. To do so, numerical features must be previously pre-binned using an appropriate technique, such as quantiles.

Carrasco Serrano, 2023 provides a transparent definition of the WoE value. Let G, and B represent the number of total good and bad loans, respectively; G_k and B_k the number of good and bad loans in group k of a variable - remember, numerical variables should be discretised in advance. Then the WoE statistic for group k is given by the following formula:

$$WoE_k = log\left(\frac{G_k/G}{B_k/B}\right)$$

To understand this value, note that when $\frac{G_k}{G} = \frac{B_k}{B}$, $WoE_k = 0$. Using the natural logarithm comes handy as it maps this positive ratio to the real line, with groups with a higher ratio of goods to bads having a higher WoE, and conversely[9].

The WoE transformation is standard practice in the scoring industry for a variety of reasons. First, and conveniently, the groupings made by the WoE transformation will be the items of the credit scorecard. Also, the WoE transformation deals with missing values and outliers. For the former, it creates a separate category with its own WoE or incorporates them into another category with similar WoE; while for the latter, it assigns them to a wider interval mitigating their effects. Next, it provides interpretable relationships displayable in charts that can account for non-linear effects. Last but not least, it transforms variables into a common scale: the WoE scale.

This latter fact is especially valuable in itself. First, the relative importance ('weight') of each variable can be measured by tracking its dispersion on the common scale. Additionally, it renders the magnitude of the effects of the WoE-transformed features directly comparable. Finally, within a variable, a bin with higher WoE always has a lower default rate, as shown in Appendix B.1.

[9] A small drawback of the WoE is that groups with no goods/bads have a WoE of $-\infty/+\infty$, respectively. Implementations of the WoE transformation deal with this case directly by setting a mininum number of goods and bads for each bin.

In a nutshell, the WoE transformation consists of three main steps:

1. Transform all variables into a categorical type, i.e., bin any numerical features using methods like quantiles or decision trees.

2. Combine existing categories in order to maximise some supervised criterion - in this application, the Information Value defined in Subsection 3.1.2. If the characteristic is numerical or ordinal, then the groups represent partitions of the real line; if nominal, the groups are combinations of the original categories.

3. For each of these groups, the procedure substitutes the original value of a variable for the WoE statistic for this group.

Specifically, this analysis utilises the Python library OptBinning in order to implement the WoE transformation. This library combines a pre-binning step based on a Classification And Regression Tree (CART) with a convex mixed-integer programming formulation in order to refine the initial bins (pre-bins) to maximise the Information Value of the grouping. See Navas-Palencia, 2020 and Navas-Palencia, 2021 for more technical details on the implementation.

The implementation applies a monotonic trend to the Event Rate (the proportion of bads in a bin). In Appendix B.1, this is shown to be equivalent to a monotic WoE trend. Generally, the minimum fraction of observations in a (pre-)bin is set so to the default 5%. This value is thinned for some zero-inflated variables, such as the number of credit cards with credit over the credit limit, to be able to use their information in the analysis.

In turn, the maximum number of (pre-)bins is fixed to 20. For categorical variables, an "Other" category integrating any category that has less than 5% of the total observations is created. The technique also constructs a separate category for missing values.

Lastly, the resulting bins are refined by comparing the Event Rates between consecutive bins by means of a z-test[10]. If the p-value of a test of the difference in means between two consecutive bins is larger than 0.5%, these bins are merged. Some features are dropped on account of having non-significant bins[11]. The remainder of parameters are set to defaults, that can be consulted at the Optbinning library website.

As an illustration, the WoE transformation for Loan-to-Value (LTV) is displayed in Figure 2.2. The WoE transformation presents a decreasing trend since the larger the principal of the loan compared to the good purchased, the more likely it is a bad loan. An extra category for missing values is created: since there are no missing values for LTV, missing values are assigned a zero WoE[12]. The 'Special' category is automatically made

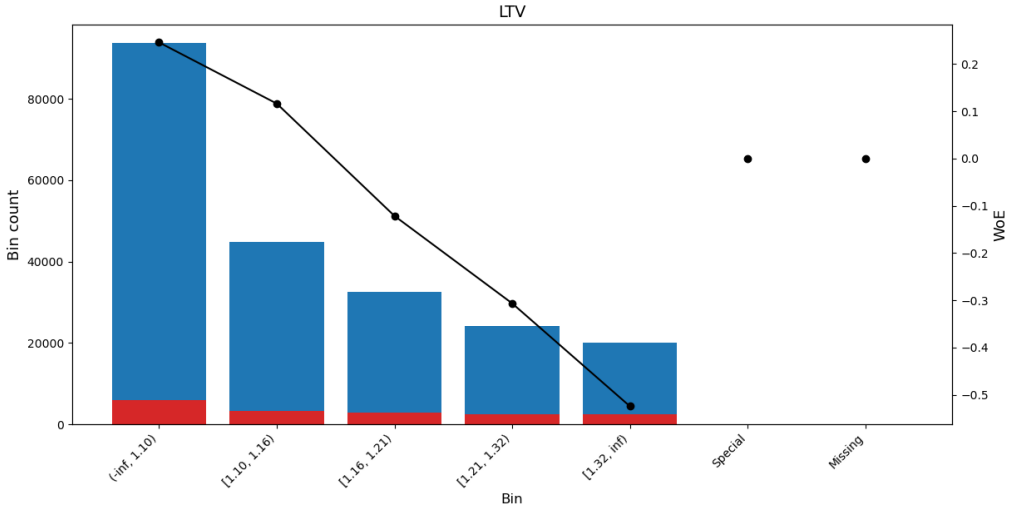
[10]This test was introduced to reduce the number of output groupings; the discriminatory ability of the resulting classical model was monitored, with no significant drop being observed.

[11]These features are indicator variables on email provided, permanent address different than work address, workday for application, bureau debt service ratio.

[12]This reasonably presumes for unobserved categories an implicit ratio of good/bad equal to the sample good/bad ratio.

available by the Optbinning library but is not used in this study.

Figure 2.2
Weight-of-Evidence (WoE) transformation for Loan-To-Value. The bars display counts (left axis), the line the WoE for each bin (right axis).



2.6. Correlation Analysis

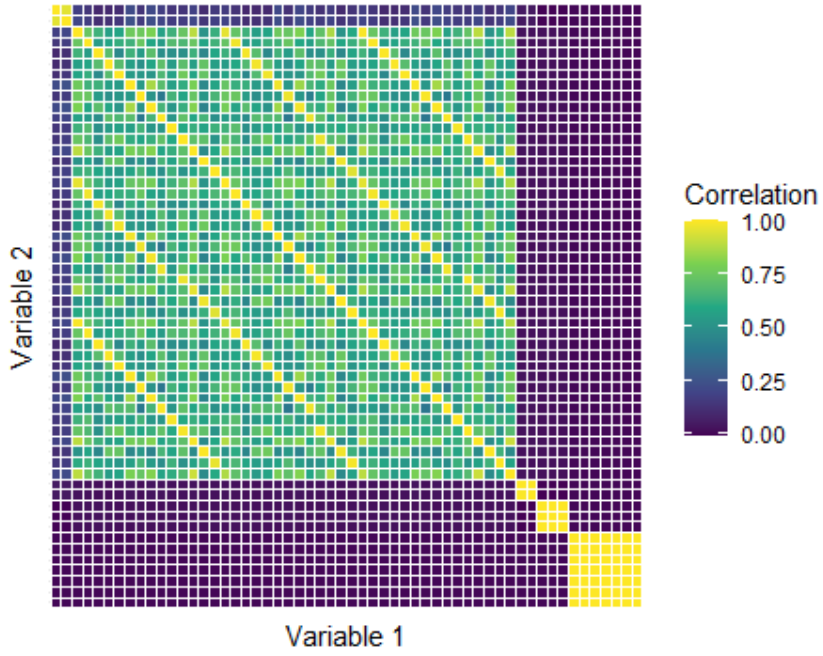
Multicorrelation in the pre-processed dataset must be studied as it may have unintended effects for interpretability, as addressed in Section 3.3. The starting point for this analysis are the features after the WoE transformation.

Figure 2.3 displays a correlation heatmap with only the predictors that have a correlation higher than 0.9, a total of 56 variables. There is some acute correlation between a number of subgroups of the predictors (sorted from larger to smaller subgroup size):

- Housing variables (42 features). The yellow subdiagonals with almost perfect correlation are originated from having several statistics on the same feature, e.g., the median, mode and average for the living area owned by the applicant.
- Characteristics related to previous bads (last seven features). This includes the number of bad bureau and previous HC loans, and indicators of having credit cards over the limit.
- The amount of requirements to the credit bureau about the client in the last hour, day, and week (3 variables).
- Two groups of two variables: the region for rating purposes and observed bads in the applicant's social circle.

Figure 2.3

Correlation Heatmap for Predictors with Highest Correlation after WoE transformation. The chart only includes predictors with a correlation higher than 0.9.



Despite these findings, multicorrelation is not dealt with directly for at least two reasons. First, it would reduce the inherent interpretability of the dataset. For example, combining the amount of requirements to the credit bureau and the apartments would create a hardly interpretable new feature. Second, Siddiqi, 2017 advises that multicorrelation is not a significant concern in credit scoring, because multicollinearity can be handled directly using variable selection - as done in the next chapter.

3. CLASSICAL CREDIT SCORING

In this chapter, a classical credit scorecard is developed. A scorecard assigns a score to ranges of values of predictors, with the aim that the added score across predictors turns into a ranking that effectively separates between good and bad loan applications. A scorecard is named 'classical' in this thesis when it follows the structure presented below in Section 3.4.

Siddiqi, 2017 argues that developing a scorecard is as much a statistical process as a business process. Therefore, the process described here is slightly more stylised than in practice; this is done to preserve a focus on Statistics. A corollary of the first statement is that the resulting product must be *interpretable*, i.e., that can be understood by many users across the organisation (again, following the definition of interpretability by Kim et al., 2016). Scorecards thus serve as a golden standard of interpretable results in this domain, and lead the way to the introduction of a framework for interpretability of credit scoring models at the end of this chapter.

The development flow for the classical credit scorecard in this chapter is as follows:

1. Bivariable stepwise selection based on F-test and IV criterion.
2. Stepwise selection via AIC and BIC.
3. Fit logistic regression models using the WoE-transformed variables.
4. Transform the regression output into the scorecard format.

The first two steps reduce the set of predictors that are actually used in the scorecard. The last step involves a transformation from the estimated logistic coefficients to the credit scores that make up a credit scorecard. Just the train dataset is used in these steps, leaving the test dataset for benchmarking only.

3.1. Bivariate Variable Selection

Mays and Yuan, 2004 propose an approach that encompasses bivariate variable selection using the Information Value and additional measures such as the χ^2 statistic and the Pearson rank correlation between the predictor and the target.

In turn, Carrasco Serrano, 2023 introduces the IV of a feature as:

$$IV = \sum_{k=1}^{K} \left(\frac{G_k}{G} - \frac{B_k}{B} \right) WOE_k$$

with G_k, B_k, G, and B bearing the same meaning as in the definition of the WOE_k above. In essence, the IV of a feature is a sum across bins of the differences in the percentage of goods and bads weighted by the WoE of each bin.

Here the focus is on using the IV to select variables as in Siddiqi, 2017. However, the F-statistic is calculated first in order to obtain a broader picture on the significance of predictors.

3.1.1. F-statistic

This brief subsection studies the statistical significance of each predictor separately. To that purpose, a simple regression[13] of the target default on the predictor is run separately for each predictor. Then, the F-statistic is defined following Wood, 2017 as:

$$F = \frac{(RSS_0 - RSS_1)/q}{RSS_1/(N-1)}$$

where RSS_0 and RSS_1 are the Residual Sum of Squares corresponding to the null and full model, q is the number of coefficients tested (q=1 in the application) and N is the sample size. The F-statistic with a large sample size should be effective thanks to its asymptotic properties (Wood, 2017).

The main finding is that almost every WoE-transformed variable has a significant effect on default. The exception to the rule at a 1% significance level are only a few dummy variables related to, for example, providing documentation, not providing an email or whether the application was started on a workday. All other engineered features are found to be significant. Therefore, the F-statistic while informative, is not really practical to filter out variables in this application.

3.1.2. Information Value

The attentive reader may be wondering, however, how the Information Value (IV) statistic can be translated into anything actionable for variable selection. To that aim, Siddiqi, 2017 proposes the following rule-of-thumb for the values of the IV:

- IV < 0.02: generally unpredictive
- IV $\in [0.02, 0.1)$: weak
- IV $\in [0.1, 0.3)$: medium
- IV $\in [0.3, 0.5)$: strong
- IV ≥ 0.5: potential overfitting

[13] Remember that every predictor is numerical after the WoE transformation, that is why the regression is also simple for categorical features.

The analysis is aligned with this recommendation and initially filters out any variables with IV lower than 0.02. 64 characteristics are selected, with a sample of the selected attributes according to the IV being exhibited in Table 3.1.

Strength	Attributes
Medium	External Scoring 1
Strong	External Scoring 2, External Scoring 3
Weak	Age, Age in Bureau, Loan-to-Value, CC Consumption (Qrt, %), Occupation Type, Income Type, Education Level, # Times CC Over Lim (Qrt), Asset Value, Region/City Rating, (51 more)

Table 3.1
Classification of attributes according to their Information Value for prediction of default.

3.2. Stepwise Variable Selection

In order to further narrow down the number of selected variables in the scorecard, a second round of variable selection is applied using stepwise selection, in both directions. It is worth noting that, as described in García-Portugués, 2024, stepwise selection is a greedy algorithm where each iteration considers including or dropping any possible variable by minimising an information criterion. As a starting point, the null model is considered. Solutions are computed using both the AIC and BIC.

Bear in mind that stepwise variable selection has an advantage over bivariate variable selection as it considers the relationship between target and predictor conditioning on the effect of variables already incorporated into the model. Table 3.2 exhibits the selected features using AIC and BIC, with 28 and 25 variables being selected, correspondingly. Note that the variables are displayed columnwise in the order of selection[14], starting from the left column, and the presence of asterisks note when a variable was selected only under a single criterion. Clearly, BIC is more selective than AIC, with most of the variables selected under BIC being selected under AIC, but one (Region/City Rating). While using the former model leads to a somewhat more parsimonious scorecard, these models should still be compared in terms of discriminatory power. In fact, picking between the set of predictors chosen by AIC or BIC requires exploring the trade-off between discriminatory power and model interpretability.

[14]Interestingly, the order of selection was the same for the AIC and the BIC criteria.

External Scoring 3	# Active Bureau Credits (Curr)
External Scoring 2	Age in Bureau
External Scoring 1	# Times CC Over Lim (Qrt)
Loan-to-Value	Organisation Type
CC Consumption (Qrt, %)	Different Contact City
Occupation Type	Time Last Bureau Credit
Income Type	Time ID Change
Time Employed	Time Registration
Education Level	Family Status
Gender[a]	Median Usage Age
Flag Document 3	CC Consumption (Current, %) (*)
Credit Length (Lower Bound)	Mode Total Area (*)
Car Age	Days Last Phone Change (*)
Median Max # Floors (*)	Region/City Rating (**)
Age in Home Credit	

Table 3.2
Variable selection for classical credit model using AIC and BIC. Variables selected either with the AIC or the BIC criterion are highlighted using an asterisk (*) or double asterisk (**), respectively.

[a]Including variables such as gender is not generally allowed in professional settings on the basis of no discrimination. Here they are preserved as the focus is strictly academic.

3.2.1. VIF criterion

High multicollinearity may be a large issue as different effects in the linear regression may compensate each other, resulting in an overfitted regression model down the line. To tackle this issue, multicollinearity is reduced using the Variance Inflation Factor (VIF) as an additional selection method. The definition of the VIF for feature i is the following (O'Brien, 2007):

$$\text{VIF}_i = \frac{1}{1 - R_i^2}$$

where R_i^2 is the R-squared of the regression of feature i on the remainder of features. The threshold for variable selection is set to a maximum VIF_i of 10, that is equivalent to a $R_i^2 > 90\%$, that indicates a high degree of multicollinearity.

3.3. Estimating Logistic Regression Models

Setting $Y_i := 1$ if application i is a bad loan, the object of interest is $p_i := P(Y_i = 1)$. The logistic regression model is then defined as follows:

$$\eta_i = log\left(\frac{p_i}{1-p_i}\right) \tag{3.1}$$

$$\eta_i = \beta_0 + \beta_1 WOE(x_{1i}) + \cdots + \beta_K WOE(x_{Ki}) = \beta_0 + WOE(x_i)\beta \tag{3.2}$$

where η_i is the logit of the probability of default, $WOE(\cdot)$ is the WoE transformation applied to x_{ki}, the observation i of covariate k, and β_k is the k-th coefficient, with β_0 as the intercept. The features and coefficients are collected in vectors $x_i := (x_{1i}, \ldots, x_{Ki})$ and $\beta := (\beta_1, \ldots, \beta_K)^T$ in the last equality.

Observe that the vector of covariates x_i differs depending on the selection criteria described in Section 3.2. Also, logistic regression is fitted in an iterative fashion, dropping features with positive coefficients and reestimating the regression until all estimated coefficients are negative. This is done with interpretability in mind: a larger WoE should always imply a lower level of default. Therefore, positive coefficients are a clear sign of overfitting.

The output of the logit models with covariates selected by AIC and BIC is presented in Tables A.1 and A.2 in the Appendix. Both models display generally significant variables at a 5% significance level. However, the AIC-based model, presents three non-significant variables at this significance level, such as the median maximum number of floors among the applicant's addresses; the BIC-based model has all variables statistically significant at the 5% significance level.

Generally, measuring discriminatory power is a complex process, usually implemented with a variety of measures as in Chapter 10 in Siddiqi, 2017. For example, the Kolmogorov-Smirnov statistic is frequently used to measure the divergence between the distribution of scores for bads and goods. Yet, the Gini statistic, as described in Carrasco Serrano, 2023, is the main reference to measure discriminatory power used in this research for the purpose of simplicity and comparability.

The Gini statistic is defined as:

$$\text{Gini} := 2(R - 0.5) = 2R - 1$$

where R the ROC AUC. The Gini essentially quantifies how much the distribution of scores for bads is accumulated before the distribution of scores for goods. As the ROC AUC ranges from 0 to 1, the Gini actually ranges from -1 to 1 and can be interpreted as a continuum in the following way:

- When the Gini equals 1, the model perfectly separates between goods and bads, with goods always having larger scores than bads.

- When it equals 0, the model does as well as a random ranking model.
- Negative Gini represents performance worse than random, and should thus not be generally expected.

For the curious reader, Appendix B.2 dives deeper into the grounds for this interpretation.

Table 3.3 provides a summary of the train and test Gini for the logistic regression model using the AIC and BIC selected variables. The outcomes are slightly better for the train than the test dataset, perhaps pointing towards some slight overfitting, probably negligible in practice.

	Train	Test
AIC	50.70%	49.58%
BIC	50.70%	49.59%

Table 3.3
Gini statistic for train and test datasets for logistic regression models after AIC and BIC variable selection.

Both the AIC-based and BIC-based models get similar results on discriminatory power. The latter is used to build the scorecard for three reasons. First, the difference between the test AIC statistic (49.58%) and the BIC statistic (49.59%) is very small, practically insignificant. Second, all variables in the BIC-based model are significant while in the AIC-based model they are not. Third, the BIC-based models result in a somewhat leaner, easier-to-interpret scorecard since it has slightly fewer variables than the AIC-based model.

3.4. Building a Classical Scorecard

In this section, the general procedure to build classical credit scorecards is described. Next, the text introduces the transformation from the logistic regression estimates to the scorecard format.

Following Siddiqi, 2017, the score of an application is defined as:

$$\hat{S} := a + b\hat{\eta},\ a \in \mathbb{R}, b < 0 \tag{3.3}$$

This is, the score is a linear transformation of logit estimates that inverts their ranking, so that a higher logit (estimated probability of default) has a lower score and the other way around. By Equation (3.2), knowing x determines $\hat{\eta}$ and consequently $\hat{S}$. Following this line of thinking, one can define $\hat{S}(x)$, the function that maps from covariates to a score.

A good starting point to build a scorecard is to visualise it. Table 3.4 displays a template for a classical scorecard. In its most compact version, a scorecard consists of three columns: variable, bin, and score. Importantly, the bins (intervals extended with the missing value category) are also required to form a partition of the support of the variable to which they correspond (again, the support including missing values).

As discussed, the configuration of a classical scorecard cannot be considered strictly from a mathematical viewpoint as it renders a standard, interpretable output that facilitates decision making in the industry (Siddiqi, 2017). Yet, its relevance serves as motivation to explore under which mathematical conditions such a result can be constructed.

Variable	Bin	Score
Variable 1	$(a, b) \cup \text{Missing}$	$S_1((a, b) \cup \text{Missing})$
$\vdots$	$\vdots$	$\vdots$
Variable k	$B_{k,m}$	$S_k(B_{k,m})$
$\vdots$	$\vdots$	$\vdots$
Variable K	$B_{K,n}$	$S_K(B_{K,n})$

Table 3.4

Template for a classical scorecard. $B_{k,m}$ represents the m-th bin for variable k. $S_k(B_{k,m})$ is the score (a real value) for that bin. K is the number of variables included in the scorecard.

To use this scorecard for a new incoming application, one must follow three simple steps:

1. Find the value of the variables in the table for the application (if unavailable, set to missing).
2. Select the rows with bins that include these values. For each variable, only one bin is selected.
3. Add up the scores of the selected rows, getting the score of the application.

With knowledge of these steps, the technical conditions required to build a classical credit scorecard can be captured in the following result.

Result. *A classical credit scorecard can be constructed for score estimates $\hat{S}$ as long as there exists a sequence of bins $\{B_{k,m}\}_{k=1,m=1}^{K,M_k}$ [15] and sequence of functions $\{S_k\}_{k=1}^{K}$, such that,*

$$\hat{S}(x) = \sum_{k=1}^{K} \sum_{m=1}^{M_k} S_k(B_{k,m}) I(x_k \in B_{k,m}), \ \forall x \tag{3.4}$$

[15] Bins defined as above, intervals extended with missing values that form a partition of the support of the predictor.

where $I(\cdot)$ is the indicator function, $I(C) = 1$ if and only if condition C is true otherwise equals zero, K is the number of variables, and M_k the number of bins for feature k in the scorecard. This result is useful to understand when alternative techniques can produce a classical scorecard.

The classical methodology, as presented in Siddiqi, 2017 and Carrasco Serrano, 2023, considers the bins calculated by the WoE transformation as $\{B_{k,m}\}_{k=1,m=1}^{K,M_k}$. Then, it defines the score function for variable k, $S_k^C(\cdot)$[16], as:

$$S_k^C(B_{k,m}) = b\left(\hat{\beta}_k WOE(B_{k,m}) + \hat{\beta}_0/K\right) + a/K \tag{3.5}$$

It is straightforward to prove that this definition can be represented as a classical scorecard using the result above. Plugging the definition of $S_k^C(\cdot)$ in (3.5) into the right-hand side of Equation (3.4), $\forall x$:

$$\begin{aligned}\sum_{k=1}^{K}\sum_{m=1}^{M_k} S_k^C(B_{k,m})I(x_k \in B_{k,m}) &= \sum_{k=1}^{K}\sum_{m=1}^{M_k}\left(b\left(\hat{\beta}_k WOE(B_{k,m}) + \hat{\beta}_0/K\right) + a/K\right) I(x_k \in B_{k,m}) \\ &= \sum_{k=1}^{K} b\left(\hat{\beta}_k WOE(x_k) + \hat{\beta}_0/K\right) + a/K \\ &= b\left(\hat{\beta}_0 + \sum_{k=1}^{K}\hat{\beta}_k WOE(x_k)\right) + a = b\hat{\eta}(x) + a = \hat{S}(x) \qquad \square\end{aligned}$$

where in the second equality it has been applied the fact that $\sum_{m=1}^{M_k} WOE(B_{k,m})I(x_k \in B_{k,m}) = WOE(x_k)$ for the unique WoE bin that contains x_k.

In sum, it has been proved that one can build a classical credit scorecard employing:

- WoE bins as bins.
- Score functions $S_k^C(B_{k,m})$, as per Equation (3.5).

The derived result does not depend on the choice of a and b, that remain to be set to some human-friendly values. This work conforms to common practices in the industry (Carrasco Serrano, 2023) and use the Point at Even Odds (PEO) and Point to Double Odds (PDO) that are defined so that:

- The PEO is the score value such that the estimated odds equal 1 (estimated default probability of 0.5).
- The PDO is the number of points that is required to double the odds.

Appendix B.3 shows that this leads to a choice of a and b equal:

$$\begin{aligned} a &= PEO \\ b &= -PDO/\ln(2) \end{aligned}$$

[16] Upperscript C for classical.

Throughout this work PEO and PDO are set to 600 and 20, respectively, as in Carrasco Serrano, 2023[17]. Table 3.5 displays a sample of the resulting classical scorecard for the numerical and categorical variables with largest weights, respectively. The complete version is available in Appendix A.2.

The weights $\{W_k\}_{k=1}^{K}$ compare the magnitude of standardised features for variable k. The larger the weight, the more important the effect of a variable in a scorecard is. Carrasco Serrano, 2023 defines the weights in a classical scorecard as:

$$W_k = \frac{\hat{\beta}_k \sigma_k}{\sum_k \hat{\beta}_k \sigma_k}$$

where σ_k is the standard deviation of the WoE-transformed feature k. The weights add up to one and are non-negative since the $\hat{\beta}_k$s are ensured to be always negative (by construction), leading to both a negative numerator and denominator.

Attribute	Coeff.	Weight (%)	Bin	Count (%)	Bad Rate (%)	WoE	Points
External Scoring 2	-0.73	15.11	(-inf, 0.15)	5.79	21.44	-1.13	2.86
			[0.15, 0.26)	7.12	13.93	-0.61	13.85
			[0.26, 0.39)	11.90	11.02	-0.34	19.49
			[0.39, 0.46)	7.63	9.27	-0.15	23.53
			[0.46, 0.55)	14.77	8.03	0.01	26.85
			[0.55, 0.62)	14.77	6.63	0.21	31.18
			[0.62, 0.66)	10.75	5.64	0.38	34.80
			[0.66, 0.70)	13.36	4.60	0.60	39.35
			[0.70, 0.74)	6.87	3.50	0.88	45.31
			[0.74, inf)	7.03	2.78	1.12	50.31
Education Level	-0.55	5.05	Incomplete higher, Lower Secondary, Academic Degree	4.65	9.10	-0.13	24.64
			Secondary / secondary special	71.07	8.98	-0.12	24.89
			Higher education	24.28	5.23	0.46	34.04

Table 3.5
Sample of classical scorecard for External Credit Score 2 (normalised) and Education Level.

The additional columns in the presented scorecard are the coefficient ($\hat{\beta}_k$), the weight, the within-feature count (%), the bad rate (the event rate above), and the WoE. The estimated coefficients are negative, between 0 and 1. In the full scorecard, the first six variables with largest weights account for more than 50% of the total weight. Conversely, the last six variables have a respective weight lower than 2% (e.g., type of the organisation at which the creditor works) and so possibly a low influence in the final scoring. These differences in weights are reflected in the within-feature scores, with significant variation

[17] Ultimately, the PEO/PDO transformation is applied so that the scores thus obtained are in a meaningful, accessible integer scale after rounding.

in the assigned points for External Credit Score 2, ranging from 3 to 50 points (rounded), and somewhat less variation for the Education Level, from 25 to 34 (rounded). Finally, the ordering of the bins appears logical - a higher external scoring or higher education level leads to a higher score. This ordering of the bins is a result of the enforced monotonicity in WoE. As discussed below, this result plays a fundamental role for interpretability.

3.5. Interpretability Framework for Credit Scoring

The beginning of this chapter argued that a scorecard can be considered the golden standard of interpretability in the credit scoring industry. In this small section, some key elements for interpretability of scorecards are abstracted in order to further analyse interpretable alternatives to scorecards below.

As stated in the introduction, Kim et al., 2016 qualify a method as interpretable or explainable "if a user can correctly and efficiently predict the method's results". In the context of credit scoring, the user can be any of the different roles that must understand the results of a credit scoring exercise, from the scorecard developer to any stakeholder (e.g., the porfolio manager, or the supervisor).

Siddiqi, 2017 presents a compelling case for the use of credit scorecards as an interpretability tool in the credit industry. He poses at least these reasons why this is the case:

- Scorecards can assign scores.
- Scorecards can easily provide reasons for low and high scores, and hence for decisions concerning the acceptance or rejection of an application.
- Monotonic scores can be enforced.

Starting with the first point, scores are a fundamental tool to rank applications, ideally differentiating default and non-default cases. Basically, scores are a cornerstone of interpretable credit scoring analysis. Fortunately, by Equation (3.1) and (3.3), any model that outputs probability estimates may produce scores, so that is already attainable by many models of estimation of PD.

Moreover, the additivity of a scorecard by feature can facilitate the understanding of which features drive a lower or higher score. In addition, one can understand - using weights - which variables contribute more or less to the variation in scores. Adverse Codes (Siddiqi, 2017) is a commonly used technique that exploits this additivity property. Adverse Codes rely on calculating the weighted average score for each feature first, and then reporting the application features that display the largest negative difference with respect to this weighted average. In Section 4.2, the trade-off between the interpretability provided by additivity and the cost in terms of discriminatory power is further explored. Yet, it is worth noting that additivity is only a way to provide explanations

about decision-making. More generally, stable explanations are required - i.e., that if the applicant follows the given advice, their chances to get an approval increase or at least do not decrease.

Another important factor for interpretability is that scorecards can enforce monotonicity. In a scorecard, marginal effects are clearly presented just by observing the change from one bin to another. Crucially, increasing/decreasing scores with respect to a feature is a property that resonates with credit adjudicators' previous experience, helping build adequate risk profiles. For instance, an applicant with a higher salary or account balance will tend to repay more, and that should translate into a higher credit score. Monotonicity thus yields model outcomes that are easier to understand and predict by model developers and users, as in Kim et al., 2016's definition.

In summary, the discussion here has described an interpretability framework that will be used to evaluate the interpretability of credit scoring models in the next chapters.

4. TREE-BASED MODELS FOR CREDIT SCORING

In this part, the usage of different tree classifiers are investigated, taking into account their discriminatory power and interpretability potential for credit scoring purposes.

The tree-based ML methods to be explored are first introduced. Then, after drawing a comparison between the discriminatory power of these methods in the HC dataset, a discussion on the interpretability of these techniques follows, and their limitations are highlighted. Lastly, Section 4.5 closes investigating a specially advantageous particular case that allows the developer to replicate a classical scorecard with tree-based models.

All ML models in this thesis are fitted on the WoE-transformed dataset. This implies that tree-based models are estimated based on the output of a pre-binning step that already relies on decision trees in the Optbinning library. While it could be slightly cleaner to fit these models directly using the raw features with a separate treatment of missing values and outliers, this document centers on the WoE-transformed dataset to simplify the discussion.

4.1. Introduction to Tree-Based Methods

A presentation of the estimated tree-based models follows, blending a brief theoretical description with comments on the implementation.

The decision tree classifier, as implemented in Scikit-learn (Pedregosa et al., 2011), stratifies the data in terminal nodes called *leafs* $R_1, \ldots, R_J$, and assigns the average of the target feature in each leaf as a prediction. To do this, it proceeds recursively splitting the data at a node (set Q_m) into left and right nodes (data sets Q_m^{left} and Q_m^{right}) in a way that maximises the sample-weighted Gini:

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} G(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} G(Q_m^{right}(\theta))$$

where G is the Gini function, θ is a candidate split and n_m, n_m^{left}, n_m^{right} are the number of observations in Q_m, Q_m^{left}, and Q_m^{right}, respectively. This procedure is repeated until any stopping criterion is met, such as maximum depth and minimum proportion of sample in leaf being used here. As per the default options in Scikit-learn, no post cost-complexity pruning is applied.

The estimated Random Forest models also follow the Scikit-learn implementation and fit many decision trees subject to two sources of random perturbation in order to reduce overfitting: bootstrap samples (bagging) and a random sample of $\sqrt{K}$ features taken as split candidates at each split. Predictions are based on an average of the predictions of

the individually fitted trees. In this work, a slight modification is explored: logit-based predictions, and why these are especially attractive for interpretability in credit scoring.

Lastly, the tree-based Gradient Boosting model made available by the XGBoost package (Chen and Guestrin, 2016) is also explored in this chapter. As a boosting model, it approximates the predictions by adding predictions of weak learners (here, trees) to an initial estimate. New trees are derived by minimising the following objective function[18] with respect to the node splits and w_j the predicted value of leaf j:

$$\begin{aligned}\text{obj}^{(t)} &\approx \sum_{i=1}^{n}[g_i w_{q(x_i)} + \frac{1}{2}h_i w^2_{q(x_i)}] + \gamma J + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{J}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma J\end{aligned}$$

where $obj^{(t)}$ is the objective function at step t, I_j is the set of indices of observations belonging to the j-th leaf, $q(\cdot)$ is a function that maps from the covariate space to the assigned leaf, g_i and h_i are the gradient and hessian of the loss function with respect to $\hat{\eta}^{(t-1)}$, J represents the number of leafs and λ and γ penalty parameters for the complexity of the leaf scores and the number of leafs of the new tree, correspondingly. Further details can be found in this XGBoost tutorial.

All implementations discussed here enforce monotonicity constraints. This means that, at each decision split, only a negative effect of WoE on default rate is permitted, otherwise discarding the candidate split. This ensures that the following monotonicity condition is satisfied for $\hat{p}$ the estimated probability of default:

$$\hat{p}(x_1, \ldots, x, \ldots, x_n) \geq \hat{p}(x_1, \ldots, x', \ldots, x_n)$$

for $x' > x$, with $x_1, \ldots, x_n$ representing the WoE-transformed variables. While this ensures local monotonicity, Sections 4.3 and 4.4 discuss why this may not be sufficient to satisfy the interpretability requirements for credit scoring laid out in previous Section 3.5.

4.2. Discriminatory Power of Tree-based Models

In this section, a brief comparison of the discriminatory power displayed by the fitted tree-based models is performed. To fit the models, five-fold cross-validation over a lean grid of hyper-parameters is performed. Not much emphasis is however placed on cross-validation, because for this piece of research it suffices to show that some ML-based options are competitive alternatives against the classical approach.

Table 4.1 displays the selected hyperparameters, and the Gini statistic in the train and test datasets for each of the fitted models. Two versions of each model are estimated: an

[18] The objective function results from the sum of a second-order Taylor approximation to the loss function (typically, the negative log-likelihood) and a penalty complexity factor.

unconstrained version, and a constrained (explainable) version. The explainable version constrains decision trees to have a maximum depth of five, whereas for random forests and gradient booster models this sets the maximum depth to one - below, it is seen below why this brings about enhanced interpretability.

Model	**Parameters**	**Train Gini**	**Test Gini**
Decision Tree	Max Depth: 10, Min Samples Leaf: 0.5%	37.25%	37.27%
Decision Tree (explainable)	Max Depth: 5, Min Samples Leaf: 0.5%	37.16%	37.15%
Random Forest	N Estimators: 500, Max Depth: 5, Min Samples Leaf: 0.25%	45.59%	44.78%
Random Forest (explainable)	N Estimators: 500, Max Depth: 1, Min Samples Leaf: 0.25%	44.43%	44.17%
Gradient Booster	N Estimators: 250, Max Depth: 5, Learning Rate: 0.1	54.64%	51.1%
Gradient Booster (explainable)	N Estimators: 100, Max Depth: 1, Learning Rate: 0.3	51.85%	50.57%

Table 4.1

Comparison of discriminatory power among tree-based models estimated on the WoE transformed train dataset by 5-fold cross-validation in the specified parameters.

The results show that the Gradient Booster is the most competitive tree-based model in terms of train/test Gini, followed by the Random Forest and the Decision Tree - this is true for both the unconstrained and explainable versions. As expected, the explainable versions display poorer performance in discriminatory power than the unconstrained versions as the parameter space of the former has been reduced.

Still, the explainable version of Gradient Booster in the train and test dataset (51.56% and 50.35% Gini, correspondingly) outperforms the Train/Test Gini results for logistic regression given in Table 3.3 (50.70% and 49.59%). This outcome is especially valuable as Section 4.5 shows that a classical scorecard can be constructed for this particular model.

4.3. Interpretability of Decision Trees

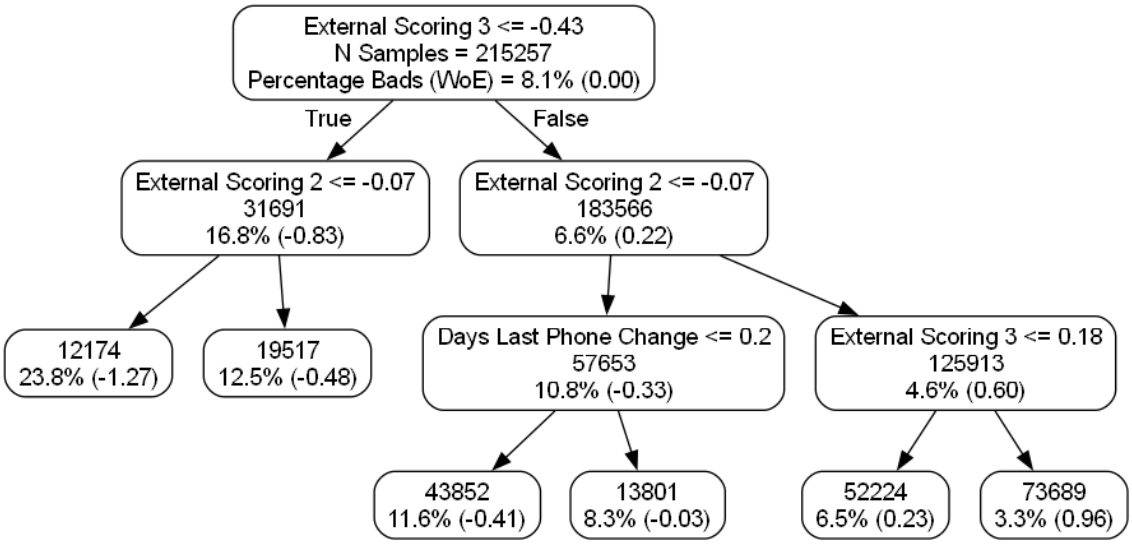
The discussion of interpretability in trees begins by showing why decision trees are generally considered easily interpretable (James et al., 2023) and then proceeds to challenge

that view in the context of credit scoring. Since the goal is to explain acceptance or rejection for each application, interpretable global metrics for trees like feature importance are not covered here.

Figure 4.1 displays the best-performing shallow decision tree fitted. Following Kim et al., 2016's definition, this solution is interpretable as one can easily understand how this decision tree makes predictions. For instance, for low values of External Scoring 3 (≤ -0.43) and low values of External Scoring 2 (≤ -0.07) it assigns a low score corresponding to an Event Rate of 23.8 %. One can proceed in a similar fashion for any other leaf of the decision tree, providing explanations for any result.

Figure 4.1

Explainable Decision Tree fitted in WoE-transformed dataset, with maximum depth 3 and monotonicity constraints. From top to bottom, each split node contains the decision rule, the number of observations, and the percentage of bads with the WoE value in parentheses.



While it is straightforward to extract the rules that determine scores, constructing stable (and thus, usable) explanations of a decision about an application may not be so. To clarify this argument, Table 4.2 extracts the splitting node conditions dividing them by rules (up to 3, the depth of the tree) and calculating the corresponding scores for each terminal node. The scores are calculated from the estimated probability by the usual formula:

$$\hat{S} = a + b\text{logit}(\hat{p})$$

where a and b are derived again from the PEO/PDO method.

For illustration purposes, suppose that the cutoff for approval/rejection in this setting is 660. An application belongs to the rejected leaf with a 658 scoring (row 3), characterised by a high External Scoring 3, low External Scoring 2 and a recent phone change (associated with a lower WoE)[19]. Following Molnar, 2022, human-friendly explanations

[19] In practice, days since the last phone change should not be included in a scorecard because an applicant

consists of mostly two, three reasons. Because of this, a recent phone change and a too low External Score 2 are provided as reasons for rejection. The client would then focus on these two factors, but may still suffer a slight drop in their External Scoring 3, that gets lower than -0.43 (in the WoE scale). This low External Score 3 would entail an automatic rejection - as all its corresponding leaves have a score lower than 660 (row 1 and 2). In summary, misleading reasons for rejection (or acceptance) would have been provided to a client, and thus this cannot be a valid explanation method.

Rule 1	Rule 2	Rule 3	Points
External Scoring 3 ≤ -0.43	External Scoring 2 ≤ -0.072	-	633
External Scoring 3 ≤ -0.43	External Scoring 2 > -0.072	-	656
External Scoring 3 > -0.43	External Scoring 2 ≤ -0.072	Days Last Phone Change ≤ 0.196	658
External Scoring 3 > -0.43	External Scoring 2 ≤ -0.072	Days Last Phone Change > 0.196	669
External Scoring 3 > -0.43	External Scoring 2 > -0.072	External Scoring 3 ≤ 0.179	676
External Scoring 3 > -0.43	External Scoring 2 > -0.072	External Scoring 3 > 0.179	697

Table 4.2
Extracted rules from shallow decision tree with scores. Scores assigned based on PDO and PEO equal to 20 and 600, correspondingly.

The problem only gets worse for deeper trees, since the deeper the tree, the longer the chain of rules that induces a score. In addition, as seen in Table 4.1, decision trees display worse performance if their depth is constrained. In a nutshell, the longer the chain, the easier that a change in a far ancestor invalidates the explanation provided. The conclusion is that deep decision trees cannot be considered as valid explainable tools for credit scoring, as they give rise to unstable explanations. Next section studies another important factor influencing whether additive tree-based learners provide (un)stable explanations.

4.4. The Curse of Interactions

Here, the usage of additive tree-based learners for prediction of log odds is investigated. This section also introduces the 'curse of interactions' in regard to the interpretability of this type of models. The 'curse of interactions' captures that, whenever unconstrained interactions in weak learners are allowed, there can be non-intuitive changes in the explanation for a rejection.

The prediction function of additive tree-based learners can be represented as:

$$\eta(x) = \sum_{l=1}^{L} T_l(x) + BS \tag{4.1}$$

where η is the log-odds function for covariates x, T_l the prediction function of tree l, and BS is the so-called base score. These models are especially attractive for interpretability

can easily modify it to 'game' the application without substantially altering their risk profile. Nevertheless, these 'gameable' variables are maintained to keep the focus on the statistical methods.

because they are additive in scores - one can obtain a logit estimate for each separate tree, and then transform it into the score scale using Equation (3.3).

The Curse of Interactions is now illustrated via an example. Table 4.3 displays the scores for a fictional company where the optimal strategy is to take into account the interactions between an external score and the debt-to-income ratio. Recall that the total score of a client is obtained by adding up the scores obtained by each learner. Note that monotonocity indeed holds for every split in the way expected: higher external scores and lower debt/income ratios lead to higher scores.

Decision Tree	Rule 1	Rule 2	Points
Learner 1	Mid External Score	-	300
Learner 1	High External Score	-	350
Learner 2	Mid External Score	High Debt/Income Ratio	200
Learner 2	Mid External Score	Medium Debt/Income Ratio	300
Learner 2	High External Score	Medium Debt/Income Ratio	250
Learner 2	High External Score	Low Debt/Income Ratio	350

Table 4.3

Illustrative rules extracted from an additive learner based on trees of maximum depth 2.

The aim is to provide an explanation for the outcome of a client that has a mid External Score (300 from first learner) and a Medium Debt/Income Ratio (300 from second learner), again assuming a threshold of 660. Unfortunately for this client, given that threshold, their application is rejected. As the debt/income is already optimal in the current leaf, a simple recommendation for this case is to improve the external score to secure an acceptance next time.

But here comes the catch: once there is an improvement in the external score, one moves from row 2 to 3 in learner 2 (row 4 to row 5 in the table), losing the 50 points that would be won from learner 1. Why could this happen? Because the algorithm detects that conditional on a high external score, a medium debt/income ratio is a poorer signal than for medium external scorers. While this may be perfectly fine from the point of view of separating goods and bads, this produces unreliable explanations. This is why we are naming it a 'curse'.

Naturally, in this simple illustration, there is a simple solution: recommend the client to increase both the external score and decrease the debt/income ratio. But in real applications, with hundreds of learners, it may be very hard to track all potential interactions. This becomes even more challenging for explanations involving multiple factors (e.g., a decrease in debt to income and an increase in salary and balance), as it would require to explore all the trees using any of the factors in the explanation.

Answering how relevant this effect is for different types of tree-based additive model is deemed out of the scope of this thesis, only being briefly explored using individual

conditional expectation plots in Chapter 5. In what follows, the discussion centers on a clear case where there is no curse of interactions nor unstable explanations (as for decision trees): additive learners based on trees of depth one. In the context of boosting, these trees of depth one are also known as tree stumps.

4.5. Classical Scorecards from Additive Tree-based Models

Here it is shown how the necessary and sufficient condition to build a classical scorecard introduced in Section 3.4 holds for additive tree-based models based on tree stumps. This discussion is heavily influenced and generalises the ideas introduced for AdaBoost on a SAS Global Forum presentation by Edwards, 2023, and recently implemented for XGBoost by Burakov, 2024.

The first step is to group the scores by features, as in any classical scorecard. This is especially straightforward with tree stumps, given that each of them only use one feature. With Equation (4.1) holding by assumption, and L_k the number of trees using feature k, the expression for the logit becomes:

$$\eta(x) = \sum_{l=1}^{L} T_l(x) + BS = \sum_{k=1}^{K} \sum_{l=1}^{L_k} T_{kl}(x_k) + BS \tag{4.2}$$

where importantly, we have introduced new functions T_{kl}, that unlike T_l, depend only on the single value of the feature being used by that learner, x_k.

Now, the fact that weak learners are constrained to be tree stumps can be exploited. With c_{kl} the cutoff-point of learner T_{kl} [20] and $\{c_{kl}\}_{l=1}^{L_k}$ an increasing sequence, a *fine partition* of the range of feature k can be defined as:

$$(-\infty, c_{k1}), [c_{k1}, c_{k2}), \ldots, [c_{k,L_k}, \infty) =: B_{k1}, \ldots, B_{k(L_k+1)} =: \{B_{km}\}_{m=1}^{M_k}$$

where M_k, the number of bins for feature k, equals $L_k + 1$, the number of decision stumps using feature k plus one, and $\{B_{km}\}_{m=1}^{M_k}$ is the sequence of scorecard bins as in the classical setting. This fine partition is employed so that each of these bins are always contained within one and only one leaf of the tree stumps using feature k.

Based on these partitions, we propose the following score functions - with respect to B_{km} - for additive tree-based learners:

$$S_k^{AT}(B_{k,m}) = b\left(\sum_{l=1}^{L_k} T_{kl}(B_{k,m}) + BS/K\right) + a/K, \quad k = 1, \ldots, K \tag{4.3}$$

where here the tree stump predictions play the role of the coefficient times WoE in the logistic regression, and the base score the role of the intercept - see Equation (3.5).

[20] The process described can be adapted to categorical or mixed features (e.g., with missing values) as well, but since all features have been WoE-transformed, the discussion is restrained to numerical features to focus on the essentials.

As done in the logistic-regression setting, one can prove that these bins combined with these score functions meet the necessary and sufficient condition to represent a classical scorecard. Again, plugging the definition of $S_k^{AT}(\cdot)$ into the right-hand side of Equation (3.4):

$$\sum_{k=1}^{K}\sum_{m=1}^{M_k} S_k^{AT}(B_{k,m})I(x_k \in B_{k,m}) = \sum_{k=1}^{K}\sum_{m=1}^{M_k}\left(b\left(\sum_{l=1}^{L_k} T_{kl}(x_k) + BS/K\right) + a/K\right)I(x_k \in B_{k,m})$$

To simplify the inner sum, recall that for fixed k, $\{B_{km}\}$ is a *fine partition* of the range of k. Thus, for all x_k there exists a unique $m^*(x_k)$ such that $x_k \in B_{km^*}$, $T_{kl}(B_{km^*}) = T_{kl}(x_k)$, and otherwise $I(x_k \in B_{k,m}) = 0$ so[21]:

$$\begin{aligned}
\sum_{k=1}^{K}\sum_{m=1}^{M_k} S_k^{AT}(B_{k,m})I(x_k \in B_{k,m}) &= \sum_{k=1}^{K}\left(b\left(\sum_{l=1}^{L_k} T_{kl}(B_{km^*}) + BS/K\right) + a/K\right) \\
&= \sum_{k=1}^{K}\left(b\left(\sum_{l=1}^{L_k} T_{kl}(x_k) + BS/K\right) + a/K\right) \\
&= b\left(\sum_{k=1}^{K}\sum_{l=1}^{L_k} T_{kl}(x_k) + BS\right) + a \\
&= b\hat{\eta}(x) + a = S(x),\ \forall x \qquad \square
\end{aligned}$$

Finally, scorecard developers remain free to select a and b as in the logistic regression setting. Below the same values of PDO and PEO are used as before (20 and 600, respectively).

4.5.1. Classical Scorecard from Random Forest

As explained in Section 4.1, random forests simply fit a range of trees subject to stochastic perturbations and then aggregate their results. These decision trees are fit just as standard decision trees. Consequently, there is no restriction against aggregating their predictions in different ways, as long as these predictions reduce the variance of single predictions. This principle is leveraged by modifying the prediction method in Scikit-learn's random-forest to produce probability predictions based on an average of logits, which can be adapted to meet Equation (4.1). Defining *logit* as the logit function and $\hat{p}_l$ as the probability estimate by decision tree l, the new prediction method simply produces new logit estimates $\hat{\eta}$ as:

$$\hat{\eta} = \sum_{l=1}^{L} logit(\hat{p}_l(x))/L \tag{4.4}$$

with $\hat{T}_l$, the logit value contributed by tree l in Equation (4.2), equal to $logit(\hat{p}_l)/L$. Note that Random Forests utilise no base score, so $BS = 0$.

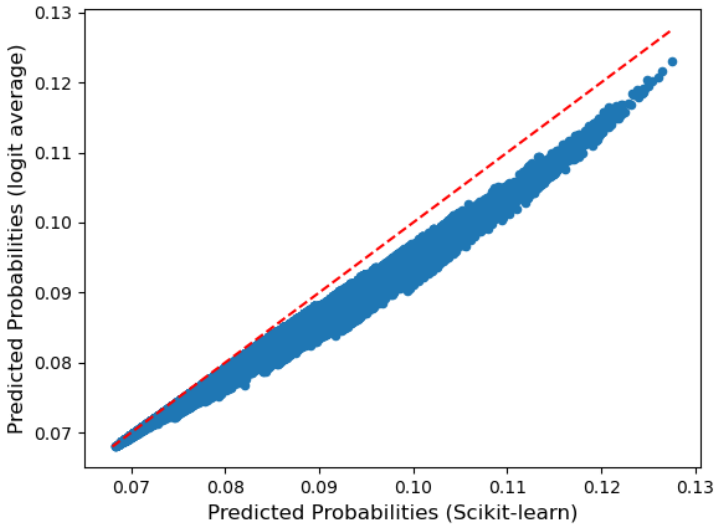
Figure 4.2 compares the default predictions made by Scikit-learn against the predictions based on the new method. The proposed prediction method tends to underestimate

[21] Precisely a fine partition was introduced to get this property.

the original probabilities as it averages over low logit values. Yet, this is not especially concerning for three reasons. First, averaging on the logit scale may be more accurate to produce logit estimates, and scores are directly derived from these. Second, this deviation is common in practice and could be solved via recalibration as in Siddiqi, 2017[22]. Lastly, both methods result in similar train/test Gini statistics, so the estimated ordering of the observations is not significantly affected by the transformation.

Figure 4.2

Comparison of predicted probabilities made by Scikit-learn as per the default setting against proposed predictions based on average of logits.



Based on this prediction method, the score function derived from Equation (4.3) equals:

$$S_k^{RF}(B_{k,m}) = b\left(\sum_{l=1}^{L_k} logit(B_{k,m})/L\right) + a/K \tag{4.5}$$

where $logit(B_{k,m}) = logit(x_k)$ for $x_k \in B_{k,m}$. At the implementation level, creating a fine partition $\{B_{k,m}\}$ is not even required when applying this scoring method to the WoE-transformed data. This is because as the WoE transformation yields a finite covariate space, one can evaluate Equation (4.5) for every WoE value of the features used by the random forest, and then simplify the result by merging contiguous bins with the same value of points.

Table 4.4 displays a sample of the random-forest-based scorecard for the same attributes as the sample of the classical scorecard in Table 3.4. It is remarkable that, for the same feature, the new scorecard only has a subsample of the bins in the original table (e.g., higher education and not higher education for the education level). This is because

[22]In practice, this can be achieved directly by means of regression to adjust the average predicted event rate to that observed.

the original scorecard has a row for every value of the WoE of selected variables, whereas here that space is simplified. Yet, it may be the case that the new scorecard based on a random forest contains more rows, as it can select more variables than in the classical setting. In fact, here the length of the random-forest scorecard (112 bins) is close to the length of the original (117 bins). Finally, as guaranteed by the monotonicity constraints, a lower event rate in a bin translates into a larger number of points.

Attribute	Weight (%)	Bin	Count (%)	Bad Rate (%)	Points
External Scoring 2	3.88	(-inf, 0.26)	12.91	17.30	52.44
		[0.26, 0.39)	11.90	11.02	68.80
		[0.39, inf)	75.19	6.02	81.90
Education Level	2.72	Not higher education	75.72	8.98	71.13
		Higher education	24.28	5.23	87.76

Table 4.4
Sample of scorecard based on a random forest for (normalised) External Credit Score 2 and Education Level.

Compared to the original scorecard, the weights seem more evenly divided (e.g, External Scoring 2 has a weight of only 3.88% against 15% in the classical scorecard). This may be a consequence of the trees in a random forest not considering the fit of previous trees - so similar rules in the untransformed covariate space are repeatedly found, thus leading to an inefficient extraction of scoring rules. A scorecard based on a Gradient Boosting model should be able to remedy this, as in boosting, new learners consider the fit of previous learners.

4.5.2. Classical Scorecard from Gradient Booster

The XGBoost model is a gradient boosting algorithm that produces predictions in the logit scale in the following additive fashion:

$$\hat{\eta} = \sum_{m=1}^{M} f_m(x_i) + BS \tag{4.6}$$

where each f_m represents a decision tree (and more generally any weak learner, like a linear model) and BS is the base score.

The attentive reader will have realised by now that the prediction function in Equation (4.6) has the same form as the definition of additive tree-based models in Equation (4.2). Consequently, one can follow the same procedure as described in that section: using tree stumps as weak learners, grouping by the feature used in the tree stump, in order to obtain f_{kl} tree functions. Then taking $T_{kl} = f_{kl}$ in Equation (4.6) results in a classical scorecard.

A sample of the resulting scorecard thus built is presented in Table 4.5. Interestingly, the top numerical and categorical feature selected by weight differ from those in

the classical scorecard, although both original top features are also present in this scorecard with slightly lower weights than External Scoring 3 and Gender. The resolution of the bins is similar to the original resolution in the WoE map, only with some slight changes. This demonstrates that the boosting strategy is indeed capable of utilising the full WoE-transformed feature space to assign points, unlike the random forest scorecard. The only slight change in the bins is that the missing category for both External Scoring 3 and Gender is allocated to another bin - maybe indicating a small limitation of the original binning algorithm, that always creates a separate category for missing values.

Attribute	Weight (%)	Bin	Count (%)	Bad Rate (%)	Points
External Scoring 3	13.23	(-inf, 0.18)	5.05	21.76	26.79
		[0.18, 0.32)	9.68	14.26	40.16
		[0.32, 0.41), Missing	29.88	9.57	49.78
		[0.41, 0.47)	7.25	7.87	56.63
		[0.47, 0.57)	13.11	5.93	62.18
		[0.57, 0.70)	20.48	4.42	68.64
		[0.70, inf)	14.56	3.32	71.52
Gender	4.08	Male	34.05	10.21	7.39
		Female, Missing	65.95	6.97	15.38

Table 4.5

Sample of scorecard based on a XGBoost model for top quantitative and categorical feature.

More generally, the usage and explainability of the XGBoost scorecard is exactly the same as the classical scorecard, yet it is backed by stronger train/test Gini results as seen in Section 4.2. A possible criticism of the implementation, though, is that variable selection is being delegated to the XGBoost model, so the scorecard developer loses some modelling flexibility. However, the boosting approach provides at least the same flexibility as the linear model in order to include/exclude variables in the model. The selection of certain variables of interest (or groups of variables) can be ensured by using *training continuation*, whereby one trains an initial model with a group of variables, and then restarts the training with the next group of variables, and so on.

5. MODEL-AGNOSTIC INTERPRETABILITY TOOLS

Model-agnostic tools can, by definition, be applied to any estimation model. This has the substantial and obvious advantage of providing flexibility in model choice. The alternative is to only use interpretable (white-box) models, likely incurring in a loss in predictive power caused by a reduction in model search space.

To narrow down the search for interpretability tools, it is helpful to differentiate between global and local interpretability tools as in Molnar, 2022. While the former type explains the entire model, the latter focuses on individual predictions. Some global model-agnostic methods such as permutation feature importance (Molnar, 2022) can provide valuable summary information about the scoring model. Yet, both explaining decisions regarding applications and monotonocities are properties that must hold for every observation, so the main focus here is on local interpretability tools.

The first item of this chapter is fitting neural networks to be later explained using model-agnostic interpretability tools. Then, explanations based on counterfactuals guided by prototypes are constructed. Individual conditional expectation plots along with partial dependence plots prove to be a useful tool to understand the monotonicity of a black-box model. This last point is complemented with an application of the NN2Poly package to study monotonicity via polynomial approximations to neural network models.

5.1. Fitting Multi-layer Perceptrons

To illustrate an alternative to tree-based models, the analysis centers on the application of model-agnostic tools to the multi-layer perceptron model, a type of neural network (NN)[23], that is widely considered as a hard-to-interpret model (Molnar, 2022).

A Multi-layer Perceptron (MLP), also known as a fully connected feed-forward NN, with L-1 hidden layers and h_l neurons at each layer can be defined recursively as in Morala et al., 2023, with ${}^{(l)}y_j$ the output of neuron j at hidden layer l being given by:

$$ {}^{(l)}y_j = g\left(\sum_{i=0}^{h_{l-1}} w_{ij}\, {}^{(l-1)}y_i\right) \tag{5.1} $$

In the formula, w_{ij} are parameters to be estimated known as weights, and g is the activation function set by the developer. The final output of the MLP equals $\hat{\eta} = {}^{(L)}y_j = \sum_{i=0}^{h_{l-1}} w_{ij}\, {}^{(L-1)}y_i$ in the logit scale.

The NNs presented here are estimated using Scikit-learn, except for the NN2Poly application, that uses the `keras` library in R. In the Scikit-learn implementation, the `MLPClassifier` is fitted via 5-fold cross-validation for the initial learning rate and the

[23] In practice though, only single-layer neural networks are fitted in this chapter for the sake of simplicity.

learning rate method, while the activation function is fixed to the rectified linear unit. The best performing learning rate uses inverse scaling and an initial value of 10^{-3}, and attains a train, test Gini of 54.71% and 51.51%, correspondingly - this discriminatory power is comparable to the optimal results for the XGBoost model found in Section 4.2.

5.2. Explaining Application Decisions

Counterfactual explanations find the factors that, if changed, would have led to a different decision. In particular, for declined applications, an acceptance. The alternative, anchors, finds rules such that a decision is maintained with some high probability. An implicit assumption underlying the discussion is that decisions on acceptance/rejection are entirely based on the scoring model (using a cut-off)[24].

Compared to anchors, counterfactuals have the advantage that provide contrastive explanations, that are more human-friendly (Molnar, 2022). However, both have their uses for credit evaluation. On the one hand, counterfactuals are more helpful for rejected applications, answering: what can the applicant do to get an acceptance next time? On the other hand, anchors are more helpful for accepted applications, answering: why was the application accepted? This thesis focuses on counterfactuals for rejections.

The implementation of counterfactuals employed relies on the Python package `alibi` (Klaise et al., 2021), especifically in the implementation proposed by Looveren and Klaise, 2019. With x_0 a single instance to be explained, the authors study perturbations δ of the type $x_{cf} = x_0 + \delta$ so that the following loss function is minimised:

$$\min_{\delta} L := c \cdot L_{pred} + \beta L_1 + L_2 + L_{proto}$$

where:

- L_{pred} tracks the difference between the probability of the originally predicted class (non-default) and the target class (default).
- $\beta L_1 + L_2$ is an elastic net regulariser term. In this application, β is increased until a sparse perturbation is obtained (max. 10 features).
- L_{proto} measures the distance between the proposed counterfactual and a prototype of default, that is found by taking the k-th nearest non-default observation to x_0 [25].

According to Looveren and Klaise, 2019, the L_{proto} term facilitates the construction of a proper counterfactual by drifting the result x_{cf} towards the distribution of the counterfactual class (non-default) and by accelerating the counterfactual search process.

[24] This assumption may be accurate for credit cards or other consumer loans, not as much for typically larger loans such as mortgages where modelling results would need to be complemented with other methods to reach a decision.

[25] With the process to find the nearest neighbour being optimised by using k-d trees in the `alibi` library.

Table 5.1 exhibits the obtained explanations for two applications with predicted default according to a neural network fit using Scikit-learn. With the adjustments in WoE in the perturbation column, the application would have been granted instead taking 600 as the acceptance threshold. There are a couple of remarks to be made. The first is that both the features selected in the explanation and the perturbations seem to be affected by the correlation between predicting features. Starting with the features selected, they may not be the actual cause leading to a rejection, but rather a proxy for it. This may be the case for the number of bureau requirements in the first counterfactual, likely a proxy for other variables such as age or time since first credit. This issue can be alleviated by applying variable selection as in the classical analysis. More concerning is the sign of perturbations, that can sometimes be negative when working with complex ML models. This happens to the second counterfactual, which suggests a negative perturbation for Age[26] - that is, moving to a group with higher default. Again, this is most probably caused by the interaction of the remainder of features with this change in Age, resulting in invalid explanations (and consequently, invalid estimation models).

Observation	Feature	Current Value (WoE)	Perturbation	Explanation
1	Bureau Reqs (quarter)	Less than 1 (0.028)	+0.037	Increase*
2	Loan Annuity	Larger than 16.5K (-0.04)	+0.08	Decrease*
	Age	From 36.5 to 40.2 (-0.08)	-0.11	Decrease*
	Work Phone	Has Work Phone (-0.204)	+0.09	No Work Phone*
	# Active Bureau Credits (Curr)	More than 5, inclusive (-0.42)	+0.19	Decrease*

Table 5.1

Perturbations in WoE for Different Features and Observations. The asterisks (*) indicate that the perturbation in WoE is not large enough to modify the currently assigned bin.

The second remark is that counterfactuals are not really an adequate technique to be applied after a WoE transformation of the features. This is because the recommended adjustments do not necessarily imply a change in the underlying bin for the feature. As a matter of fact, this is the case for every suggested change in Table 5.1, as indicated by the asterisks. For instance, the loan annuity WoE in the second counterfactual is suggested to be increased by 0.08 to 0.04, but the closest bin in the WoE scale has WoE 0.07, so it is uncertain how to interpret this result in the original feature space.

5.3. Checking monotonicity

In this section, the aim is to employ Individual Conditional Expectations (ICE) plots in order to check monotonicity. According to the Scikit-learn documentation (Pedregosa et al., 2011), features are splitted into X_S (the features of interest), and X_C, their complement. With $X_C^{(i)}$ the values of the complementary features for observation i, the ICE plot draws a

[26] Age should likely be excluded from a counterfactual as it is not quite actionable; variables like LTV or loan amount would be better fit for purpose. Yet, this is ignored here to keep matters simple.

distinct line $\hat{p}(X_S, X_C^{(i)})$ for each, with $\hat{p}(\cdot)$ the output probability of the model, and X_S the values of the features of interest between the 5% and 95% percentiles of X_S.

A clear advantage of this method is that it is extremely easy to explain: the lines show the change in the prediction value as the features of interest vary. However, Molnar, 2022 presents a drawback, suggesting that some of the points represented in the chart may be unlikely in the joint feature distribution - since the full range of the feature for every value of the complement features is evaluated.

Figure 5.1

Individual conditional expectation (ICE) plot from Multi-layer Perceptron for (WoE-transformed) education and loan-to-value. The plot only displays a random subsample of 75 observations, with a line for each observation.

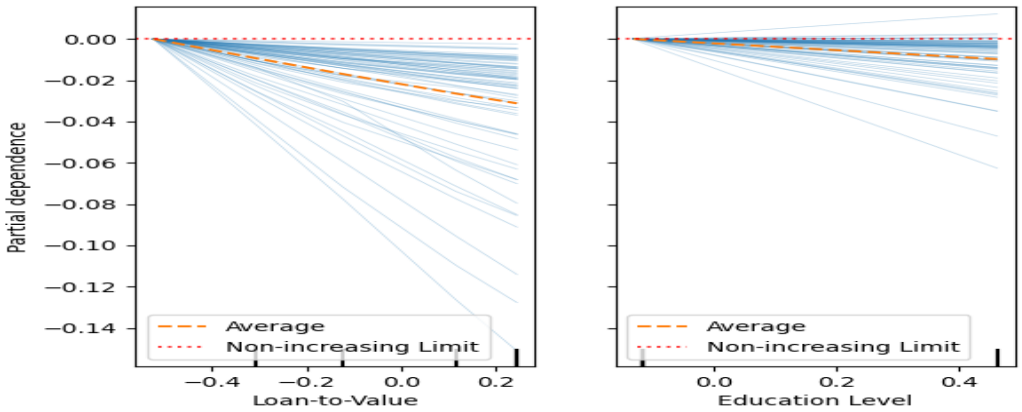


Figure 5.1 depicts ICE plots for a random subsample of 75 observations for loan-to-value and education Level in the train dataset. Note that an application of this neural network model to credit scoring may be problematic for at least two reasons. First, the right chart shows that, in some cases, moving to an education group with a lower bad rate (higher WoE) brings about a higher predicted estimate of default - as it happened with counterfactuals in the previous section. This break of monotonicity is detected thanks to the ICE plot, but can only be remedied by a proper choice of the estimation model. In practice, the ICE plot should rather be replaced by a systematic check of the ICE curves to see how many instances display monotonic behaviour, instead of only looking at a subsample.

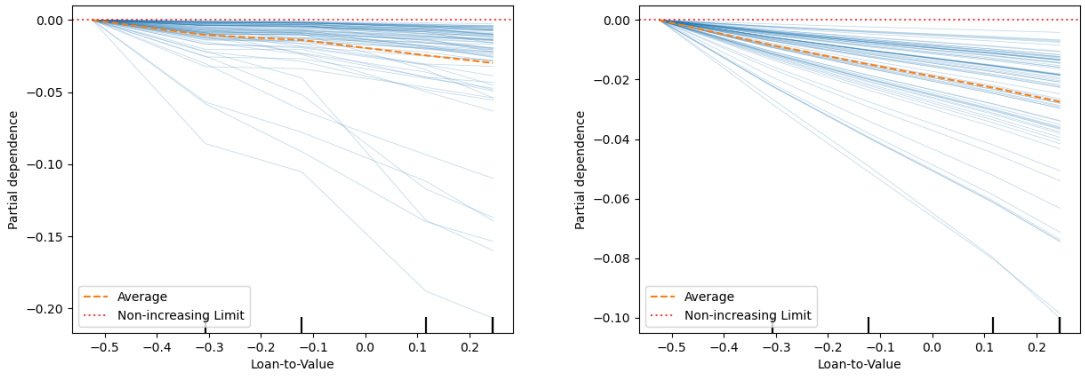
The ICE plot also facilitates identifying another issue: different instances suffer different penalties for the same reduction in, e.g. LTV value. This can be hard for credit practitioners to explain to clients that are provided different explanations for the same change. This can be especially the case in times where information is more frequently and widely shared than ever.

Finally, the same tool is applied to the explainable and non-explainable version of the XGBoost model fitted in Section 4.5.2, with only LTV as the feature of interest. The result

is shown in Figure 5.2. The explainable version shows monotonic effects, as guaranteed by the additivity of tree stumps. Maybe more interestingly, monotonicity also seems to hold for the unconstrained XGBoost model at least for the small subsample displayed in the plot, despite the curse of interactions discussed in Section 4.4.

Figure 5.2

Individual Conditional Expectation (ICE) plots from XGBoost (left) and explainable XGBoost (right) for changes in (WoE-transformed) loan-to-value. The plots only display a random subsample of 75 observations.



Tools to measure the extent of deviations from monotonicity are left for future research. These tools should factor in the likelihood of the points that break monotonicity to provide an accurate statistical measure of monotonicity coverage in the joint feature distribution.

5.3.1. Polynomial Representation via NN2Poly

The idea of a polynomial representation of a NN is explored here. Despite this being a model-specific option, it provides extra insights on the lack of monotonicity of more complex ML models like the NN model used throughout this section. The findings further support that a proper choice of the scoring model cannot be replaced by a model-agnostic tool.

Morala et al., 2023 construct smart Taylor approximations to the activation function in Equation (5.1). To guarantee the validity of the Taylor approximations, the NN is fitted with the inputs scaled to lie in $[-1, 1]$ and the one-norm of the weights at each layer constrained to equal one as per Section IV.A in Morala et al., 2023. Lastly, the maximum order of the approximation is set here to 1 to obtain a linear, interpretable polynomial representation. These restrictions result in a loss in discriminatory power, with the train/test Gini falling to 52.1% and 50.83% compared to the original 54.71% and 51.51% for the unconstrained MLP[27].

[27]Part of this change may also reflect the fact that for this exercise the hyperbolic tangent was used as an activation function.

The main findings of this examination are exhibited in Figures 5.3 and 5.4. Figure 5.3 displays the coefficients of the linear approximation to the neural network. The existence of positive values for the sign of the coefficients of the polynomial breaks the desired monotonicity of the neural network approximation, by increasing the estimated probability of default when moving to a group with lower default. Figure 5.4 exhibits the relationship in the logit scale between the original NN predictions and those of the polynomial representation, showing a striking resemblance.

Figure 5.3

Value of the coefficients of the polynomial (linear) representation obtained via NN2Poly.

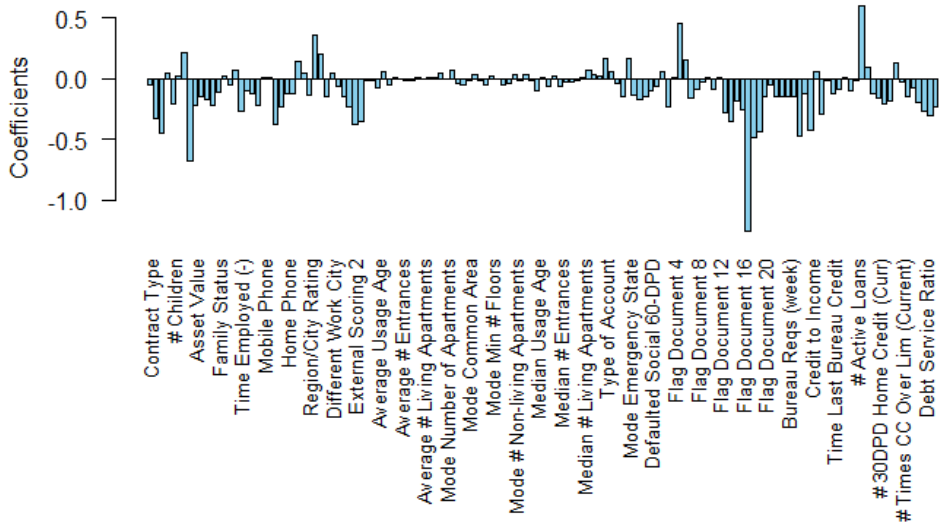
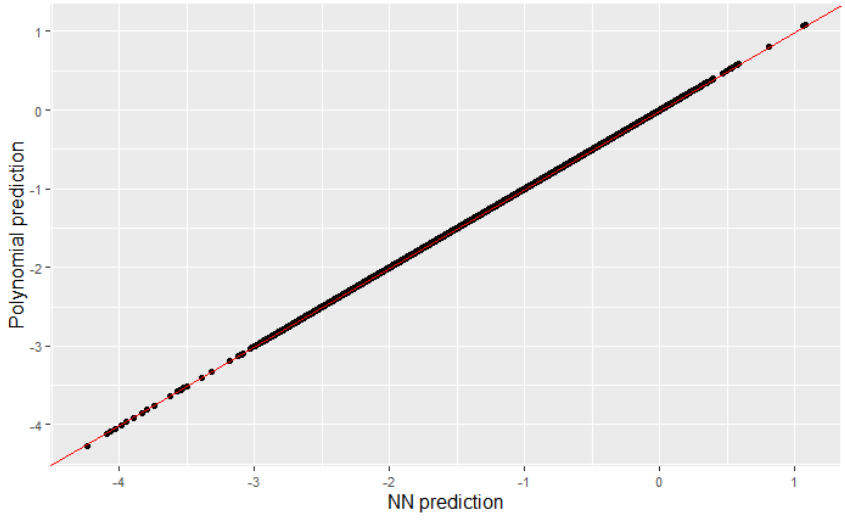


Figure 5.4

Predictions of the Neural Network against its linear representation using NN2Poly.



An important observation is that it is possible to build a classical scorecard from this

linear approximation using the result from Section 3.4. This is a direct consequence of the output of the MLP being in the logit scale and additive across features. However, Figure 5.3 demonstrates that monotonicity is an issue for this model. Maybe this limitation can be resolved by implementing constraints on the sign of the polynomial approximation in the future.

6. CONCLUSION

This Master's thesis conducted some thorough data-processing of the HC dataset[28] and then, using this data source, developed a classical credit scoring analysis. The data-processing included a WoE transformation that simultaneously dealt with missing values and encoded categorical features. The classical analysis consisted of variable selection via Information Value and stepwise selection, the estimation of a logistic regression model, and the construction of a scorecard based on this model. The discriminatory power and interpretability of the results were discussed, building a benchmark to better understand ML alternatives.

Next, we delved into the discriminatory power and explainability of tree-based ML models. With regard to discriminatory ability, most tree-based models performed worse than the classical analysis, with a notable exception: the XGBoost model. This model provided better train/test Gini than the classical analysis, both in its unconstrained and constrained (explainable) versions. We discussed issues with interpretability of deep trees and the 'curse of interactions', and then moved on to develop a technology to build classical scorecards using additive tree-stump-based models. This technology was employed to construct sample scorecards for explainable versions of the random forest and XGBoost models. From these results, one can easily envision how ML-based scorecards may become a valuable tool for credit scoring in the future. The last chapter of this work investigated how model-agnostic tools can be applied to understand the result of any ML technique, empirically focusing in this work on neural networks. Counterfactuals and ICE plots showed great potential for credit-scoring diagnostics. The derived explanations, however, inherited the properties of the scoring model. In particular, it was found that for neural networks these methods provided non-monotonic explanations that would be hard to convey in a professional environment.

In summary, this thesis illustrates the potential of interpretable ML tools for credit scoring in a real-world application. The idea presented for AdaBoost by Edwards, 2023, formalises and extends to any ML model that can meet Equation (3.4). In particular, this work formally proves that this is the case for tree-stump-based additive models, as per Section 4.5. The credit scorecard from a XGBoost model provides better discriminatory power than that from a logistic regression, preserving the same interpretability. Lastly, it is demonstrated that although model-agnostic methods are helpful diagnostic tools, they cannot serve as substitutes for a judicious choice of the scoring model.

This work also contributes to the field of credit scoring by opening several avenues for future research. The main proposal is to develop algorithms and techniques that introduce human-friendly and possibly low-dimensional interactions between the features. A rele-

[28] Home Credit, now EmbedIT, provided the author a license for academic use of the dataset via email, available on request.

vant and useful functionality would be to allow interactions between macroeconomic and individual-level features, making the output model more robust to changes in the overall economy. This way, statistical models would recover one of the five original C's of qualitative credit analysis: 'conditions'. In this respect, the Optbinning library already offers options to construct a 2-dimensional scorecard. However, to the best of the author's knowledge, there is no available statistical tool to select the best interactions with constraints that help preserve the interpretability of the result. In particular, combining these constrained interactions with the XGBoost model should constitute a step in the right direction, especially given that XGBoost can already outperform logistic regression without interactions.

Additionally, the development of refined model-agnostic diagnostic tools should allow researchers to investigate to what extent the new estimation models respect both weak and strict monotonicity. This is key to avoid misleading advice - e.g., recommend to improve a client's income with a null or even negative effect in their application. More precisely, building summary statistics on monoticity coverage that weigh the likelihood of observed monotonicity breaks in ICE plots should help inform the search for better scoring models. Also, it should be worthwhile to research polynomial approximations to machine learning models with constraints in the sign of the coefficients - to guarantee monotonicity. Finally, the development of open-source software to construct ML-based credit scorecards could offer significant educational and professional rewards in this field. This should have the potential improve the discriminatory power and interpretability of credit scoring applications across the industry.

BIBLIOGRAPHY

Altman, E. I., Marco, G., & Varetto, F. (1994). Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the italian experience). *Journal of Banking & Finance*, *18*(3), 505–529. https://doi.org/https://doi.org/10.1016/0378-4266(94)90007-8

Bracke, P., Datta, A., Jung, C., & Sen, S. (2019). Machine learning explainability in finance: An application to default risk analysis. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3435104

Burakov, D. (2024). Boosting scorecards [Accessed: 2024-06-17]. https://github.com/deburky/boosting-scorecards

Carrasco Serrano, J. (2023). De la econometría clásica a los modelos de machine learning: Un enfoque práctico de predicción en economía. https://e-spacio.uned.es/entities/publication/c194cdd8-a8c3-4cb5-8ec4-c3fc6b0cc952

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Crone, S. F., & Finlay, S. (2012). Instance sampling in credit scoring: An empirical study of sample size and balancing [Special Section 1: The Predictability of Financial Markets Special Section 2: Credit Risk Modelling and Forecasting]. *International Journal of Forecasting*, *28*(1), 224–238. https://doi.org/https://doi.org/10.1016/j.ijforecast.2011.07.006

Dumitrescu, E., Hué, S., Hurlin, C., & Tokpavi, S. (2022). Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *European Journal of Operational Research*, *297*(3), 1178–1192. https://doi.org/https://doi.org/10.1016/j.ejor.2021.06.053

Durand, D. (1941). *Risk elements in consumer instalment financing*. National Bureau of Economic Research, Inc. https://EconPapers.repec.org/RePEc:nbr:nberbk:dura41-1

Edwards, P. (2023). Real adaboost [Accessed: 2024-06-17]. https://github.com/pedwardsada/real_adaboost/blob/master/real_adaboost.pptx.pdf

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188. https://doi.org/https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

García-Portugués, E. (2024). *Notes for predictive modeling* [Version 5.10.1. ISBN 978-84-09-29679-8]. https://bookdown.org/egarpor/PM-UC3M/

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning with applications in python*. Springer. https://doi.org/10.1007/978-3-031-38747-0

Kim, B., Khanna, R., & Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 29). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf

Klaise, J., Van Looveren, A., Vacanti, G., & Coca, A. (2021). Alibi explain: Algorithms for explaining machine learning models. *J. Mach. Learn. Res.*, *22*(1).

Knutson, M. L. (2020, April). Credit scoring approaches guidelines [Correspondence, World Bank]. https://www.worldbank.org/

Kuhn, M., Wickham, H., & Hvitfeldt, E. (2024). *Recipes: Preprocessing and feature engineering steps for modeling* [R package version 1.1.0, https://recipes.tidymodels.org/]. https://github.com/tidymodels/recipes

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136. https://doi.org/https://doi.org/10.1016/j.ejor.2015.05.030

Looveren, A. V., & Klaise, J. (2019). Interpretable counterfactual explanations guided by prototypes. *CoRR*, *abs/1907.02584*. http://arxiv.org/abs/1907.02584

Lucas, Y., & Jurgovsky, J. (2020). Credit card fraud detection using machine learning: A survey. *CoRR*, *abs/2010.06479*. https://arxiv.org/abs/2010.06479

Mays, E., & Yuan, J. (2004). Variable analysis and reduction. In E. Mays (Ed.), *Credit scoring for risk managers: The handbook for lenders*. Thomson Learning.

Molnar, C. (2022). *Interpretable machine learning: A guide for making black box models explainable* (2nd ed.). https://christophm.github.io/interpretable-ml-book

Morala, P., Cifuentes, J., Lillo, R., & Ucar, I. (2023). Nn2poly: A polynomial representation for deep feed-forward artificial neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Navas-Palencia, G. (2021). Optimal counterfactual explanations for scorecard modelling. *abs/2104.08619*. http://arxiv.org/abs/2104.08619

Navas-Palencia, G. (2020). Optimal binning: Mathematical programming formulation. *abs/2001.08025*. http://arxiv.org/abs/2001.08025

O'Brien, R. M. (2007). A caution regarding rules of thumb for variance inflation factors. *Quality & Quantity*, *41*(5), 673–690. https://doi.org/10.1007/s11135-006-9018-6

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petrides, G., Moldovan, D., Lize Coenen, T. G., & Verbeke, W. (2022). Cost-sensitive learning for profit-driven credit scoring. *Journal of the Operational Research Society*, *73*(2), 338–350. https://doi.org/10.1080/01605682.2020.1843975

Siddiqi, N. (2017). *Intelligent credit scoring*. John Wiley & Sons, Ltd. https://doi.org/https://doi.org/10.1002/9781119282396.biblio

Thomas, L. (2000). A survey of credit and behavioural scoring: Forecasting financial risk of lending to consumers. *International Journal of Forecasting*, *16*, 149–172. https://doi.org/10.1016/S0169-2070(00)00034-0

Thomas, L., Edelman, D., & Crook, J. (2002, January). *Credit scoring and its applications*.

Verbraken, T., Bravo, C., Weber, R., & Baesens, B. (2014). Development and application of consumer credit scoring models using profit-based classification measures. *European Journal of Operational Research*, *238*(2), 505–513. https://doi.org/https://doi.org/10.1016/j.ejor.2014.04.001

Wiginton, J. C. (1980). A note on the comparison of logit and discriminant models of consumer credit behavior. *The Journal of Financial and Quantitative Analysis*, *15*(3), 757–770. http://www.jstor.org/stable/2330408

Wood, S. N. (2017, May). *Generalized additive models* (2nd Ed.). Chapman; Hall/CRC. https://www.taylorfrancis.com/books/mono/10.1201/9781315370279/generalized-additive-models-simon-wood

A. MODEL OUTPUTS

A.1. Logistic Regression Fits

A.1.1. Logistic Regression based on AIC selection

Dep. Variable:	TARGET	**No. Observations:**	215257
Model:	Logit	**Df Residuals:**	215228
Method:	MLE	**Df Model:**	28
Date:	Tue, 11 Jun 2024	**Pseudo R-squ.:**	0.1157
Time:	18:05:57	**Log-Likelihood:**	-53399.
converged:	True	**LL-Null:**	-60388.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	-2.3413	0.011	-213.965	0.000	-2.363	-2.320
External Scoring 3	-0.7225	0.016	-45.224	0.000	-0.754	-0.691
External Scoring 2	-0.7448	0.015	-49.177	0.000	-0.774	-0.715
External Scoring 1	-0.4324	0.023	-18.682	0.000	-0.478	-0.387
Loan-to-Value	-0.6287	0.030	-20.767	0.000	-0.688	-0.569
CC Consumption (Qrt, %)	-0.4078	0.052	-7.897	0.000	-0.509	-0.307
Occupation Type	-0.2752	0.044	-6.295	0.000	-0.361	-0.190
Income Type	-0.2705	0.048	-5.630	0.000	-0.365	-0.176
Time Employed	-0.5211	0.042	-12.326	0.000	-0.604	-0.438
Education Level	-0.5424	0.040	-13.483	0.000	-0.621	-0.464
Gender	-0.6419	0.043	-14.949	0.000	-0.726	-0.558
Flag Document 3	-0.6177	0.053	-11.760	0.000	-0.721	-0.515
Credit Length (Lower Bound)	-0.4210	0.044	-9.615	0.000	-0.507	-0.335
Car Age	-0.6238	0.060	-10.429	0.000	-0.741	-0.507
Median Max # Floors	-0.1534	0.106	-1.448	0.148	-0.361	0.054
Age in Home Credit	-0.2921	0.055	-5.352	0.000	-0.399	-0.185
# Active Bureau Credits (Curr)	-0.6440	0.061	-10.614	0.000	-0.763	-0.525
Age in Bureau	-0.2791	0.034	-8.172	0.000	-0.346	-0.212
# Times CC Over Lim (Qrt)	-0.2725	0.043	-6.341	0.000	-0.357	-0.188
Organisation Type	-0.2259	0.058	-3.908	0.000	-0.339	-0.113
Different Contact City	-0.2202	0.052	-4.268	0.000	-0.321	-0.119
Time Last Bureau Credit	-0.1917	0.043	-4.485	0.000	-0.276	-0.108
Time ID Change	-0.1737	0.044	-3.924	0.000	-0.260	-0.087
Time Registration	-0.2434	0.053	-4.607	0.000	-0.347	-0.140

Family Status	-0.1518	0.058	-2.603	0.009	-0.266	-0.037
Median Usage Age	-0.3319	0.103	-3.234	0.001	-0.533	-0.131
CC Consumption (Current, %)	-0.1221	0.055	-2.215	0.027	-0.230	-0.014
Mode Total Area	-0.1917	0.102	-1.879	0.060	-0.392	0.008
Days Last Phone Change	-0.0588	0.046	-1.286	0.199	-0.148	0.031

Table A.1
Logistic Regression Estimates after Variable Selection using AIC.

A.1.2. Logistic Regression based on BIC Selection

Dep. Variable:	TARGET	**No. Observations:**	215257
Model:	Logit	**Df Residuals:**	215231
Method:	MLE	**Df Model:**	25
Date:	Tue, 11 Jun 2024	**Pseudo R-squ.:**	0.1159
Time:	18:05:58	**Log-Likelihood:**	-53392.
converged:	True	**LL-Null:**	-60388.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	-2.3496	0.011	-216.362	0.000	-2.371	-2.328
External Scoring 3	-0.7251	0.016	-45.360	0.000	-0.756	-0.694
External Scoring 2	-0.7283	0.015	-47.133	0.000	-0.759	-0.698
External Scoring 1	-0.4300	0.023	-18.585	0.000	-0.475	-0.385
Loan-to-Value	-0.6294	0.030	-20.795	0.000	-0.689	-0.570
CC Consumption (Qrt, %)	-0.4797	0.040	-11.919	0.000	-0.559	-0.401
Occupation Type	-0.2777	0.044	-6.352	0.000	-0.363	-0.192
Income Type	-0.2646	0.048	-5.506	0.000	-0.359	-0.170
Time Employed	-0.5291	0.042	-12.531	0.000	-0.612	-0.446
Education Level	-0.5495	0.040	-13.682	0.000	-0.628	-0.471
Gender	-0.6442	0.043	-15.001	0.000	-0.728	-0.560
Flag Document 3	-0.6102	0.053	-11.614	0.000	-0.713	-0.507
Credit Length (Lower Bound)	-0.4221	0.044	-9.640	0.000	-0.508	-0.336
Car Age	-0.6010	0.060	-10.027	0.000	-0.719	-0.484
Age in Home Credit	-0.3175	0.050	-6.368	0.000	-0.415	-0.220
# Active Bureau Credits (Curr)	-0.6334	0.061	-10.445	0.000	-0.752	-0.515
Age in Bureau	-0.2862	0.034	-8.389	0.000	-0.353	-0.219
# Times CC Over Lim (Qrt)	-0.2907	0.042	-6.860	0.000	-0.374	-0.208
Region/City Rating	-0.2361	0.040	-5.961	0.000	-0.314	-0.158
Organisation Type	-0.2304	0.058	-3.987	0.000	-0.344	-0.117

Different Contact City	-0.2244	0.052	-4.355	0.000	-0.325	-0.123
Time Last Bureau Credit	-0.1886	0.043	-4.412	0.000	-0.272	-0.105
Time ID Change	-0.1817	0.044	-4.107	0.000	-0.268	-0.095
Time Registration	-0.2428	0.053	-4.605	0.000	-0.346	-0.139
Family Status	-0.1569	0.058	-2.690	0.007	-0.271	-0.043
Median Usage Age	-0.5706	0.075	-7.579	0.000	-0.718	-0.423

Table A.2
Logistic Regression Estimates after Variable Selection using BIC.

A.2. Classical Scorecard from Logistic Regression

A '...' symbol is added to long bins for the sake of space. Raw scorecards are available on request in Excel format.

Attribute	Coeff.	Weight (%)	Bin	Count (%)	Bad Rate (%)	WoE	Points
External Scoring 2	-0.73	15.11	(-inf, 0.15)	5.79	21.44	-1.13	2.86
			[0.15, 0.26)	7.12	13.93	-0.61	13.85
			[0.26, 0.39)	11.90	11.02	-0.34	19.49
			[0.39, 0.46)	7.63	9.27	-0.15	23.53
			[0.46, 0.55)	14.77	8.03	0.01	26.85
			[0.55, 0.62)	14.77	6.63	0.21	31.18
			[0.62, 0.66)	10.75	5.64	0.38	34.80
			[0.66, 0.70)	13.36	4.60	0.60	39.35
			[0.70, 0.74)	6.87	3.50	0.88	45.31
			[0.74, inf)	7.03	2.78	1.12	50.31
External Scoring 3	-0.73	15.02	(-inf, 0.18)	5.05	21.76	-1.15	2.55
			[0.18, 0.32)	9.68	14.26	-0.64	13.34
			[0.32, 0.41)	10.05	9.88	-0.22	22.06
			Missing	19.82	9.41	-0.17	23.20
			[0.41, 0.47)	7.25	7.87	0.03	27.30
			[0.47, 0.57)	13.11	5.93	0.33	33.65
			[0.57, 0.62)	7.17	4.84	0.55	38.17
			[0.62, 0.70)	13.30	4.20	0.69	41.27
			[0.70, inf)	14.56	3.32	0.94	46.42
External Scoring 1	-0.43	6.50	(-inf, 0.28)	7.92	15.08	-0.70	17.93
			[0.28, 0.41)	7.62	9.25	-0.15	24.85
			Missing	56.44	8.54	-0.06	25.95

Continued on next page

Attribute	Coefficient	Weight (%)	Bin	Count (%)	Bad Rate (%)	WoE	Points
			[0.41, 0.53)	7.93	6.69	0.20	29.25
			[0.53, 0.62)	5.71	5.46	0.42	31.96
			[0.62, 0.70)	5.12	4.59	0.60	34.22
			[0.70, inf)	9.25	2.99	1.05	39.80
Loan-to-Value	-0.62	5.92	[1.32, inf)	9.33	12.92	-0.52	17.35
			[1.21, 1.32)	11.23	10.66	-0.31	21.24
			[1.16, 1.21)	15.12	9.02	-0.12	24.54
			[1.10, 1.16)	20.82	7.25	0.12	28.79
			(-inf, 1.10)	43.50	6.43	0.25	31.10
Education Level	-0.55	5.05	Incomplete higher, ...	4.65	9.10	-0.13	24.64
			Secondary / secondary special	71.07	8.98	-0.12	24.89
			Higher education	24.28	5.23	0.46	34.04
Gender	-0.65	4.78	M	34.05	10.21	-0.26	21.86
			XNA	0.00	0.00	0.00	26.71
			F	65.95	6.97	0.16	29.70
Time Employed	-0.54	4.78	[-1789.50, inf)	61.71	9.12	-0.13	24.66
			[-2447.50, -1789.50)	9.48	8.02	0.01	26.83
			[-3212.50, -2447.50)	8.66	7.04	0.15	28.99
			[-4650.50, -3212.50)	9.17	6.07	0.31	31.44
			[-6722.50, -4650.50)	5.94	5.17	0.48	34.08
			(-inf, -6722.50)	5.05	4.24	0.68	37.28
CC Consumption (Qrt, %)	-0.49	4.31	[0.64, inf)	6.82	15.40	-0.73	16.48
			Missing	72.30	7.88	0.03	27.08
			[0.00, 0.64)	10.17	7.33	0.10	28.17
			(-inf, 0.00)	10.71	5.41	0.43	32.74
Flag Document 3	-0.63	4.16	1	71.01	8.86	-0.10	24.88
			0	28.99	6.15	0.29	32.02
Car Age	-0.60	3.40	[11.50, inf)	13.67	9.04	-0.12	24.57
			Missing	66.01	8.50	-0.06	25.74
			[7.50, 11.50)	6.45	7.05	0.15	29.24
			(-inf, 7.50)	13.87	5.55	0.40	33.64
Credit Length (Low Bound)	-0.43	3.30	(-inf, 21.46)	59.50	9.24	-0.15	24.86
			[21.46, 34.11)	33.14	6.55	0.23	29.54

Continued on next page

Attribute	Coefficient	Weight (%)	Bin	Count (%)	Bad Rate (%)	WoE	Points
			[34.11, inf)	7.36	5.49	0.41	31.88
# Active Bur. Credits (Curr)	-0.61	3.19	[4.50, inf)	7.94	11.82	-0.42	19.27
			[3.50, 4.50)	7.19	9.23	-0.15	24.14
			[2.50, 3.50)	12.33	8.20	-0.02	26.41
			(-inf, 2.50)	72.55	7.53	0.08	28.05
Age in Bureau	-0.28	2.87	[-582.50, inf)	24.63	11.13	-0.35	23.88
			[-937.50, -582.50)	8.71	9.53	-0.18	25.26
			[-1358.50, -937.50)	11.22	8.71	-0.08	26.05
			[-1910.50, -1358.50)	14.39	7.40	0.09	27.47
			[-2847.50, -1910.50)	35.30	6.10	0.30	29.11
			(-inf, -2847.50)	5.74	5.31	0.45	30.30
Income Type	-0.26	2.45	Unemployed, ...	0.01	25.81	-1.38	16.42
			Working	51.70	9.59	-0.19	25.30
			Commercial associate	23.26	7.54	0.07	27.27
			Pensioner, ...	25.02	5.43	0.43	29.90
Median Usage Age	-0.52	2.39	Missing	48.74	9.21	-0.14	24.55
			(-inf, 0.97)	7.58	8.39	-0.04	26.09
			[0.97, 0.98)	21.85	7.22	0.12	28.54
			[0.98, 0.99)	16.08	6.65	0.21	29.87
			[0.99, inf)	5.75	5.30	0.45	33.49
Age in Home Credit	-0.32	2.18	[-965.50, inf)	38.30	9.51	-0.18	25.03
			[-1179.50, -965.50)	6.59	8.64	-0.07	26.02
			[-2040.50, -1179.50)	21.85	7.73	0.05	27.15
			[-2658.50, -2040.50)	21.68	6.84	0.18	28.39
			(-inf, -2658.50)	11.58	5.93	0.33	29.82
Times CC Over Lim (Qrt)	-0.29	2.06	[1.50, inf)	2.36	19.29	-1.00	18.35
			[0.50, 1.50)	1.51	16.70	-0.83	19.82
			(-inf, 0.50)	96.13	7.66	0.06	27.19
Region/City Rating	-0.24	2.05	3	14.24	11.36	-0.38	24.10
			2	74.69	7.92	0.02	26.85
			1	11.07	4.84	0.55	30.49
Occupation Type	-0.27	2.04	Laborers, ...	23.93	10.77	-0.32	24.27
			Sales staff	10.44	9.50	-0.18	25.35
			Accountants, ...	14.62	8.41	-0.04	26.37

Continued on next page

Attribute	Coefficient	Weight (%)	Bin	Count (%)	Bad Rate (%)	WoE	Points
			Missing	31.37	6.59	0.22	28.39
			Managers, ...	19.64	6.14	0.29	28.97
Organisation Type	-0.23	1.84	Self-employed	12.54	10.17	-0.25	25.05
			Business Entity Type 3	22.20	9.32	-0.16	25.68
			Medicine, ...	41.85	7.99	0.01	26.79
			Other	5.36	7.75	0.04	27.00
			XNA	18.04	5.37	0.44	29.58
Time Registration	-0.25	1.62	[-1084.50, inf)	15.29	9.56	-0.18	25.39
			[-5732.50, -1084.50)	46.90	8.62	-0.07	26.20
			[-7329.50, -5732.50)	11.78	7.86	0.03	26.92
			[-9020.50, -7329.50)	11.02	7.18	0.13	27.63
			(-inf, -9020.50)	15.01	5.67	0.38	29.43
Time Last Bureau Credit	-0.20	1.45	[-57.50, inf)	7.10	12.83	-0.52	23.69
			[-93.50, -57.50)	6.00	10.13	-0.25	25.26
			[-222.50, -93.50)	20.28	8.85	-0.10	26.13
			[-274.50, -222.50)	6.61	7.91	0.02	26.85
			(-inf, -274.50)	60.01	7.06	0.15	27.56
Time ID Change	-0.19	1.43	[-1897.50, inf)	27.69	9.91	-0.23	25.49
			[-3112.50, -1897.50)	19.95	8.76	-0.09	26.23
			[-4068.50, -3112.50)	18.17	7.86	0.03	26.87
			[-4452.50, -4068.50)	15.02	6.76	0.19	27.76
			(-inf, -4452.50)	19.17	5.93	0.33	28.51
Different Contact City	-0.24	1.24	1	7.79	12.35	-0.47	23.45
			0	92.21	7.71	0.05	27.06
Family Status	-0.16	0.84	Single / not married, ...	24.36	9.87	-0.22	25.72
			Unknown	0.00	0.00	0.00	26.71
			Married, ...	70.41	7.62	0.06	27.00
			Widow	5.23	5.82	0.35	28.30

B. MATHEMATICAL DERIVATIONS

B.1. Monotonic Event Rate, monotonic WoE

Result. *Let the event rate be ER_k, where $ER_k := \frac{B_k}{G_k+B_k}$. Then $\{ER_k\}_{k=1}^K$ is a monotonically increasing (decreasing) sequence if and only if $\{WOE_k\}_{k=1}^K$ is a strictly monotonic decreasing (increasing) sequence.*

Proof. Since

$$ER_k = \frac{B_k}{G_k + B_k} = \frac{1}{1 + G_k/B_k}$$

and

$$WoE_k = \log\left(\frac{G_k/G}{B_k/B}\right) = \log\left(G_k/B_k\right) - \log\left(B/G\right)$$

the monotonicity of both expressions (with respect to k) is determined by the monotonicity of $\{\frac{G_k}{B_k}\}_{k=1}^K$. If $\{\frac{G_k}{B_k}\}_{k=1}^K$ is increasing (decreasing), then the ER is decreasing (increasing) and WoE is increasing (decreasing), with no other monotonic behaviour being possible. □

B.2. Connection between Gini and ROC AUC

This part of the Appendix dives deeper into the connection between the Gini and the ROC AUC. Under some assumptions, the ROC AUC can be understood as estimating the population value,

$$P(X_1 \geq X_0)$$

where X_1 and X_0 are the score for goods and bads, respectively[29].

In words, the ROC AUC estimates the probability that a randomly sampled good loan has a score larger than a randomly sampled bad loan. Since a random estimator has $P(X_1 \geq X_0) = 0.5$, the Gini statistic is defined so that 0% matches the ROC AUC of a random estimator, and 100 % matches the ROC AUC of a perfect estimator, leading to the definition:

$$\text{Gini} := 2(R - 0.5) = 2R - 1$$

with R the ROC AUC. This indeed has a Gini equal to 0 for $R = 0.5$, a Gini equal to 1 for $R = 1$. Note that negative values of Gini are possible as long as $R < 0.5$, i.e. the estimator is worse than the random estimator in separating between goods and bads.

[29] Or a monotonically increasing transformation of them, since for such g, $P(g(X_1) \geq g(X_0)) = P(g^{-1}(g(X_1)) \geq g^{-1}(g(X_0))) = P(X_1 \geq X_0)$. This interpretation is thus also valid for the logit estimates.

B.3. Deriving the logit-to-score transformation for PDO/PEO

Result. *In the relationship* $\hat{S} := a + b\hat{\eta}$*, the a and b parameters that define the PDO/PEO score are given by:*

$$a = PEO$$
$$b = -PDO/\log(2)$$

Proof. To find the formulas for parameters a and b, the definitions of PDO and PEO are used in turn and then the parameters of interest are plugged out.

To find a, the odds are set to 1 to get the PEO, so $\eta = 0$ and:

$$PEO = \hat{S} = a$$

To find b, the PDO equals the change in score $\Delta\hat{S} := \hat{S}_1 - \hat{S}_0$ that doubles the 'Good-to-Bad' odds, i.e. halves the 'Bad-to-Good' Odds ($\frac{\text{Odds}_1}{\text{Odds}_0} = 1/2$) so it follows that:

$$\begin{aligned} PDO = \Delta\hat{S} &= (a + b\hat{\eta}_1) - (a + b\hat{\eta}_0) \\ &= b\log\left(\frac{\text{Odds}_1}{\text{Odds}_0}\right) = b\log(1/2) = -b\log(2) \end{aligned}$$

□