UCLouvain

FACULTE DES SCIENCES

Faculté des sciences

Loss Given Default Modelling for corporate bonds

Author: **Olivier CHEFFERT**
Supervisors: **Donatien HAINAUT, Jean DESSAIN**
Reader: **Pierre ARS**
Academic year 2023–2024
Master [120] in Actuarial Science

Abstract

The estimation of Loss Given Default (LGD) is a critical aspect of credit risk assessment, influencing loan pricing, capital allocation, and risk management strategies in financial institutions. This master's thesis aims to develop a rigorous and regulation-compliant methodology when modelling LGD or the Recovery Rate. The research is guided by five key objectives.

Firstly, the study embarks on an exploration of preprocessing techniques tailored specifically to LGD datasets. The goal is to develop a standardised and effective methodology that addresses the characteristics and challenges of LGD data.

Secondly, the thesis investigates various modelling approaches for LGD estimation, leveraging advanced techniques such as glassbox models. By capturing the technicalities of credit risk dynamics, these models aim to enhance predictive accuracy and reliability.

In addition, the research expands the existing LGD literature by exploring alternative loss functions beyond the traditional Mean Squared Error (MSE). These alternative functions are evaluated for their efficacy in capturing the underlying risk factors associated with LGD.

Furthermore, a comprehensive economic comparison of different models and loss functions is conducted to assess their performance and suitability in real-world financial applications. This analysis provides valuable insights into the economic implications of LGD modelling decisions.

Finally, state-of-the-art interpretability methods are employed to elucidate the results obtained from the LGD models. By enhancing transparency and regulatory compliance, these methods contribute to a deeper understanding of the underlying risk factors driving LGD estimates.

The goal of this research is to build a robust and regulation-compliant framework for LGD modelling, offering valuable insights for financial institutions and regulators alike.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Donatien Hainaut, whose guidance and expertise have been invaluable throughout this journey. His mentorship and support has been instrumental in shaping this master thesis into a culmination of my studies in actuarial science.

I am also immensely grateful to my internship supervisors, Geoffrey, Jean, and Nora, for their continuous support and encouragement. Their insightful feedback and technical guidance have been indispensable in refining this work to be both practical and rigorous.

Besides, I thank Pierre Ars for agreeing to be part of the thesis jury as reader.

I extend my thanks to my colleagues at Reacfin, whose camaraderie and collaboration made my time there truly enjoyable and enriching.

Lastly, I am indebted to my friends and family for their unwavering support and encouragement throughout my 7 years academic journey. Their belief in me has been a constant source of motivation and strength.

Contents

1 Introduction **1**
- 1.1 Literature review . . . 1
- 1.2 Objectives . . . 3
- 1.3 Contributions . . . 3

2 Data analysis **5**
- 2.1 Dataset and descriptive analysis . . . 5
- 2.2 Adding variables . . . 7
  - 2.2.1 Company's financials . . . 8
  - 2.2.2 Macro-economic variables . . . 8
  - 2.2.3 Are they Gaussian? . . . 9
- 2.3 Outliers detection . . . 11
- 2.4 Imputation techniques . . . 13

3 Algorithms and loss functions **15**
- 3.1 Data preparation . . . 15
- 3.2 Loss functions . . . 16
  - 3.2.1 Mean squared error . . . 16
  - 3.2.2 Dense Loss . . . 16
  - 3.2.3 Beta deviance . . . 18
- 3.3 Algorithms . . . 19
  - 3.3.1 Basic models . . . 20
  - 3.3.2 Tree based models . . . 20
  - 3.3.3 Feed forward neural networks . . . 21
  - 3.3.4 Glassbox models . . . 22
- 3.4 Autocalibration . . . 27
- 3.5 Economic metric . . . 28
- 3.6 Results . . . 32
  - 3.6.1 Autocalibration . . . 34
- 3.7 Risk management tools . . . 35
  - 3.7.1 Scenario testing . . . 35

3.7.2 Statistical approach 36

4 Interpretability **37**
4.1 Intrinsic Interpretability 37
4.1.1 LocalGLMNet 38
4.1.2 EBM 39
4.1.3 GAMI-Net 42
4.2 Shapley values 44
4.2.1 Global interpretability 44
4.2.2 Local interpretability 45
4.3 AI risk measurement 46
4.3.1 Accuracy 47
4.3.2 Robustness 48
4.3.3 Explainability 49

Conclusion and Perspectives **51**

Bibliography **54**

Appendix **I**
A Kmeans and Burt's distance I
B Financial Ratios II
C Autocalibration III
D Full results V
E RGA statistical test IX

Chapter 1

Introduction

Loss Given Default (LGD) modelling plays a pivotal role in credit risk assessment and management within the financial industry. Understanding LGD, which represents the extent of loss incurred when a borrower defaults on a loan, is crucial for estimating the overall credit risk exposure of financial institutions. LGD models aim to quantify this loss, providing insights that inform decision-making processes related to lending, portfolio management, and risk mitigation strategies.

In recent years, LGD modelling has gained increased attention due to its significance in regulatory compliance, risk management practices, and capital allocation within financial institutions. The accuracy of LGD estimates directly impacts various aspects of the banking sector, including loan pricing, provisioning, and capital adequacy assessments.

Given its importance, LGD modelling has evolved significantly, incorporating advanced statistical and machine learning techniques to enhance predictive accuracy and robustness. Traditional approaches relied on simplistic models or rule-based methodologies, often failing to capture the complexity inherent in credit risk dynamics. However, advancements in data analytics and computational capabilities have enabled the development of sophisticated LGD models capable of handling large and diverse datasets while accounting for nuanced risk factors.

This introduction sets the stage for exploring the fundamentals of LGD modelling, its relevance in credit risk management, and the contemporary approaches employed in the field. Through a comprehensive examination of LGD estimation methodologies, including statistical techniques, machine learning algorithms, and regulatory considerations, this study aims to provide insights into the evolving landscape of LGD modelling and its implications for financial institutions.

1.1 Literature review

The estimation of Loss Given Default (LGD) is a crucial component in credit risk modelling, playing a significant role in various financial applications such as loan pricing, capital

allocation, and risk management. The literature on LGD modelling encompasses a wide range of methodologies and approaches, reflecting the complexity and importance of accurately quantifying the loss incurred when a borrower defaults. This review will delve into key studies and methodologies employed in LGD modelling, highlighting advancements, challenges, and areas of ongoing research.

Siddiqi and Zhang (2004) proposed a basic model using Beta regression and OLS. The usage of the Beta distribution is quite natural since its domain is the unit interval exactly like the LGD. However, transforming the result of the OLS through probit and Beta CDF makes it inefficient as pointed by Gert Loterman (2013). This study is the largest LGD benchmarking study and shows that non linear models are more accurate than linear ones. However, models exhibit low R^2 which indicates that LGD is hard to estimate accurately.

In fact, Qi and Zhao (2011) shows that the poor performance of this model come from the fact that such transforms are very sensitive to ϵ used to clip values from the closed unit interval to the open one such that 0 are replaced by ϵ and 1 by $1 - \epsilon$. Moreover, they have shown that using these transformations does not outperform a simple linear regression even when choosing the optimal ϵ. Their comparison also shows that complex models outperform simpler ones. This was also the conclusion of Kaposty et al. (2019), Bellotti et al. (2021) and Gambetti et al. (2022b). This last work explores the meta-learning approaches with meta-learners that combine classical algorithms. They have shown that a liner combination of ML models improves the accuracy. However, the price would be a lower interpretability.

From a dataset perspective, Bellotti et al. (2021) have shown that incorporating macro economic variables improves the prediction. In Peer-to-Peer lending, Zhou et al. (2018) have shown that the balance sheet structure is important such as the leverage ratio. On top of that, real-asset intensive companies such as utilities have lower LGD. This means that the financials of the company are crucial when estimating the LGD. Finally, Gambetti et al. (2022b) shows that uncertainty measures are crucial when modelling LGD. They have identified three types of uncertainty measures: survey based, news based and volatility based. The survey based measures such as the inflation uncertainty are realised by the Federal reserve and can be the dispersion of forecasts from professional forecasters. The news based uncertainty can reflect the volume of newspaper with respect to a given topic while the volatility based uncertainty is well described by the VIX index. Taking these informations into account, this work will use these conclusions to integrate such variables in the input dataset.

Estimating the LGD is important but there is an asymmetry when considering the PD (Probability of Default) and LGD predictions. Settlements and Supervision (2005) uses a formula to obtain a stressed version of the PD that reflects downturn economic conditions. This is not the case for LGD for which regulators ask to take macro economy into account. That difference can lead to underestimated expected loss as shown by Barbagli and Vrins (2023) due to a non linear dependency as studied by Lauria (2019). Indeed, they have shown that using a best estimate for the LGD instead of a stressed version can lead to a security level of 81% instead of 99.9% as asked by the regulator.

1.2 Objectives

LGD literature lacks in some area. In fact, Internal Rating Based models must be interpretable and robust over time. These LGD models are defined by the Basel framework Settlements and Supervision (2005). In Europe, the regulator asks explicitly to obtain interpretable models for which an institution should be able to explain the output from a given input EBA (2021). Moreover, the global explanation should also be available and grounded. This leads to the fact that these models must collaborate with humans and be based on their expert judgement. Due to these strong requirements, the institutions use simple models such as historical means or linear regression EBA (2023). However, there exist more powerful alternatives with an equal interpretability such as glassbox models.

On top of that, the LGD literature is based on the use of MSE as the loss function but this has not been assessed and this choice, which is the Gaussian deviance, might not be relevant. Finally, the preprocessing of the LGD dataset is not developed but the quality of the input data is an important subject for the European regulator.

Therefore, this work aims to achieve the following five objectives:

1. Investigate preprocessing techniques to develop an appropriate methodology.
2. Model LGD using powerful techniques, including glassbox models.
3. Expand the LGD literature by exploring alternative loss functions beyond Mean Squared Error (MSE).
4. Conduct a comparison of different models and loss functions.
5. Interpret the results obtained from these models using state-of-the-art interpretability methods.

1.3 Contributions

Based on these objectives, this study contributes in the following ways:

1. Conducting an in-depth investigation into the particularities of LGD datasets and proposing appropriate preprocessing methods.
2. Comparison of glassbox models for corporate bonds LGD: LocalGLMNet, Explainable Boosting Machine and GAMI-Net.
3. Usage of other loss functions than MSE and their comparison.
4. Usage of a bias removal method used until now in insurance field.
5. New economic metric to compare models and loss functions.

6. Comparison of interpretability tools: intrinsic interpretability, Shapley and SafeAI. SafeAI is a new tool developed to satisfy regulatory requirements such as the robustness assessment.

7. Optimizing the SafeAI Python package for scalability with larger datasets.

In summary, this study aims to provide a comprehensive, robust, and practical methodology for modelling LGD, addressing various aspects of data preprocessing, model selection, loss function optimization, bias mitigation, economic metric development, interpretability assessment, and computational efficiency enhancement.

Chapter 2

Data analysis

This chapter outlines the process of completing, analyzing, and imputing the dataset. Ensuring data cleanliness is crucial for deriving meaningful results and drawing accurate conclusions. This study relies on the Moody's Default and Recovery Database (DRD), encompassing defaulted corporate bonds spanning from 1993 to 2019. These bonds meet the Basel requirements, stipulating a minimum duration of seven years and at least one period characterized by economic downturn conditions (§472 Settlements and Supervision (2005)). Additionally, it is emphasized that the prediction should be linked to macroeconomic variables. Consequently, the variables will undergo analysis to enhance comprehension of the dataset. Subsequently, financial ratios and macroeconomic variables will be incorporated. The particular distribution of these variables will then be studied, necessitating an appropriate outlier detection method and subsequent imputation.

2.1 Dataset and descriptive analysis

The DRD covers bond defaults of global corporate/financial entities. It is one of the reference databases used in the literature (Lauria (2019), Olson et al. (2021),Gambetti et al. (2022a)) and like in those articles, this thesis will focus on corporate and financial bonds. Grace period defaults and cross defaults are not included and Moody's measured the recovery rate as the price of the bond 30 days after default expressed as percentage of par. However, a lot of those instruments are illiquid such that the 30 days post-default price may not exist. In such a case, Moody's measures the recovery rate from a price between 20 and 40 days after default. If no price is found, this field is left blank which prevents this target variable from being used in regression models. Therefore, all bonds with a missing recovery rate are excluded from the dataset.

Below are described the variables obtained through Moody's and used to predict the recovery rate. This quick overview provides insight into the strongest risk drivers that lead to a high or low recovery rate. Those variables are the characteristics of the bond (such as coupons or seniority) or the issuer (such as the country). More variables will be added in a later section.

Regions

Moody's labels issuers according to their effective domicile. Due to many countries having only one registered issuer, the country variable is excluded, and the region is represented by the continent variable instead as shown in Table 2.1. Because of the limited number of bonds issued from Africa, they are grouped with Asian bonds.

Table 2.1: Descriptive statistic of recovery rates per region

Region	Counts	Recovery Rate		
		Mean	Median	Standard deviation
America	2594	28.11%	12.5%	0.27
Europa	709	28.5%	11%	0.32
Asia-Africa	84	48.65%	44%	0.28

From these statistics, it appears that emerging markets are riskier than developed countries. However, it's essential to note that Europe has the highest standard deviation, suggesting a higher dispersion than in other regions. Most studies have focused on the US market, which has the lowest standard deviation (Qi and Zhao (2011), Nazemi and Fabozzi (2018), Olson et al. (2021), Gambetti et al. (2022a)).

Sectors

Moody's classifies each issuer into one of its 11 activity sectors, and the differentiation between these sectors is crucial. This significance arises from the fact that certain sectors are characterized as real-asset-intensive such as utilities, directly influencing the recovery rate. This impact is shown in Figure 2.1.

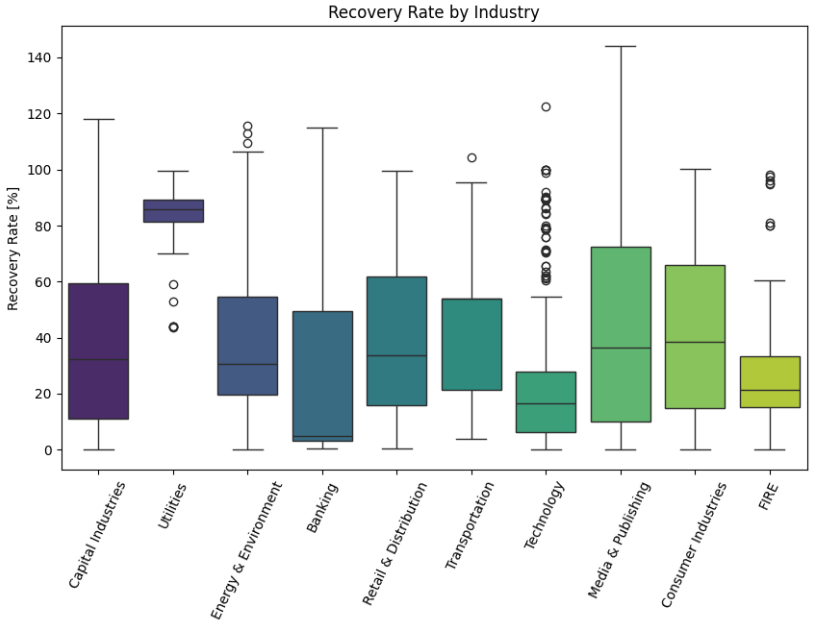

Figure 2.1: Boxplot of the recovery rates in each sector

While some sectors seem to have similar recovery rates, Utilities obtain higher recovery rates (as obtained by Zhou et al. (2018)), probably due to their balance sheet profile, which is more real-asset intensive. On the other hand, Technology obtains low recovery rates since it often belongs to the services industry with a higher proportion of intangible assets. Note that FIRE stands for Finance, Insurance, and Real Estate and is very imbalanced, with a majority of bonds from Lehman Brothers. Since it represents too high a proportion of the total dataset (nearly 20%), their bonds are reduced to those with different characteristics.

Seniorities

Seniorities describes the order of repayment priority in case of default or liquidation. The 4 main levels are given in Table 2.2 in decreasing order of priority.

Table 2.2: Descriptive statistic of recovery rates per seniority

Priority level	Counts	Recovery Rate		
		Mean	Median	Standard deviation
Senior Secured	471	56.57%	54%	0.27
Senior Unsecured	1553	31.54%	22.28%	0.29
Subordinated	340	31.5%	22.98	0.29
Preferred Stock	140	22.6%	8%	0.28

This table demonstrates that a higher priority corresponds to a higher expected recovery rate, which is intuitively reasonable. However, this variable alone cannot fully explain the priority order, as it doesn't quantify how much debt is senior to a particular bond. For instance, a subordinated bond might exhibit a higher recovery rate when no higher-priority bonds exist compared to cases where senior bonds from the same company are present. To address this, a seniority index is introduced, indicating the proportion of debt with superior priority compared to a given bond. This index is constructed in a manner analogous to Qi and Zhao (2011) which says "The variable seniority index is a variable [...] constructed using percentage above plus one-half of the percentage at the same rank".

2.2 Adding variables

To differentiate companies and estimate potential repayments in case of default, it is crucial to consider the financials of the companies. The balance sheet provides an inventory of how much the company owes and owns but also how the assets are invested. In contrast, the income statement measures the value created in a given period (typically a year), while the cash flow statement describes the cash movements from operating to financing activities.

However, there are instances, such as during a crisis, when one year may be riskier than another due to exogenous factors. This underscores the necessity of incorporating macroeconomic factors to capture the impact of such external events. Indeed, as demonstrated by

Bellotti et al. (2021), macroeconomic factors play a crucial role. Furthermore, Gambetti et al. (2022a) highlights that incorporating uncertainties — such as survey-based (e.g., inflation uncertainties), news-based (e.g., economic policy uncertainty), and volatility-based (e.g., VIX) — enhances predictive capabilities.

These types of variables will be added and studied in the following parts.

2.2.1 Company's financials

The financial standing of a company can be delineated using ratios for comparability across companies of varying sizes. These ratios can be categorized into groups:

- Leverage:
  - ➢ Total Debt over Equity
  - ➢ Net Debt over Equity
  - ➢ Net Debt over Free Cashflow
  - ➢ Net Debt over EBITDA
- Profitability:
  - ➢ Operating margin
  - ➢ ROE
  - ➢ ROA
- Liquidity:
  - ➢ Current ratio
  - ➢ Quick ratio
- Coverage:
  - ➢ Debt service ratio (EBIT/Interest)

Additionally, indicators for the sign of equity, EBITDA, and Free Cashflow have been incorporated. A more in-depth analysis of the distributions of these variables is provided in the Annex B. Note that the Debt service ratio has been inversed afterwards to allow companies with no charge of interest.

All the financial ratios describing the companies have been extracted using Bloomberg.

2.2.2 Macro-economic variables

As the dataset predominantly comprises US bonds, the macroeconomic factors were sourced from the Federal Reserve Bank of St. Louis website[1]. Moreover, the US economy holds a

[1] `https://fred.stlouisfed.org/`

pivotal position as the world's largest, exerting a substantial influence on global economic dynamics.

Each macro-economic variable has been introduced as an evolution over a time period ending just before the default date. The length of this time period, the lag, has been fixed by fitting a linear regression between several lags (1, 3, 6, 12, 18 and 24 months) and the recovery rate. The lags with the highest coefficient in absolute value have been kept and are summarized in Table 2.3:

Indicator	Lag
3 months T bills rate	1 month
High yield spread	6 months
US GDP	3 months
US Employment	1 month
Industrial Production	12 months
US Construction spending	12 months
Economic uncertainty (news based)	6 months
Inflation	1 month
VIX	6 months
S&P 500	3 months

Table 2.3: Macro-economic variables

The first two indicators serve as proxies for the financing cost, whereas the remaining variables describe the US economy as recommended by Bellotti et al. (2021), along with survey-based, news-based, and volatility-based uncertainties as recommended by Gambetti et al. (2022a).

2.2.3 Are they Gaussian?

When dealing with outliers, the classical interquartile range technique rejects a lot of variables. Let's take a look at the Interest/EBIT ratio for example:

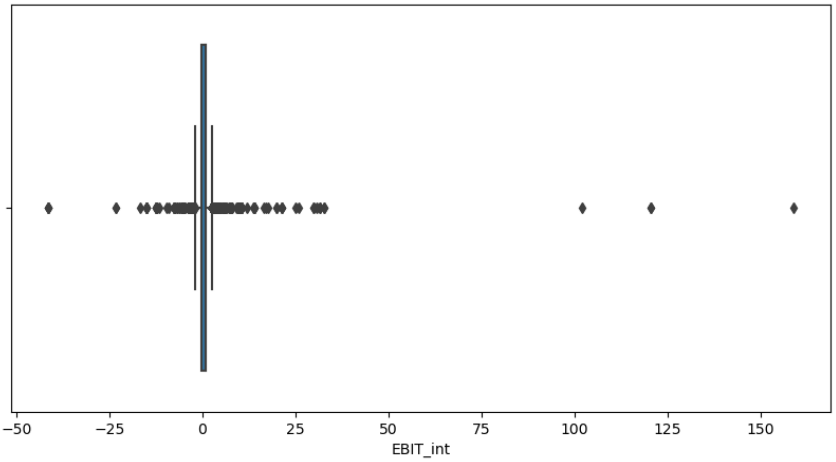

Figure 2.2: Interquartile range for the interest coverage ratio

This technique detects outliers for 18.15% of this variable, 25.47% for ROE, 21.61% for NetDebt/EBITDA, 1.27% for Debt/EBITDA, ... As suggested in section B, the financial ratios seem to have fat tails which explains this behaviour. In order, to check this characteristic, the kurtosis and skewness of each of these variables have been computed:

Table 2.4: Gaussianity: Kurtosis and Skewness

Variable	**Kurtosis**	**Skewness**
DE	37.16	4.39
NDE	40.46	4.67
Interest/EBIT	9.64	-0.08
debt/ebitda	22.18	-2.73
ND/FCF	29.38	-3.34
ND/ebitda	12.25	-2.40
current ratio	3.72	1.68
quick ratio	3.71	1.85
Operating margin	14.79	-3.57
ROE	23.17	-0.01
ROA	10.15	-1.64
debt/assets	8.46	2.47
SP500	0.98	-0.04
VIX	1.64	1.58
Inflation	0.60	-0.40
Economic uncertainty	11.19	2.69
Construction	-0.74	-0.58
Production	0.23	-0.92
Unemployment	-0.34	0.12
High Yield spread	3.50	1.30
T bill rate	41.65	6.12
US GDP	0.34	-1.08

The kurtosis measures the thickness of the tails of a distribution with a value of 3 for the Gaussian distribution. On the other end, the skewness of a distribution is its asymmetry measure which is 0 for the Gaussian distribution. The Table 2.4 indicates that the majority of these variables have fat tails, indicating leptokurtic distributions. This results in a high proportion of outliers with the interquartile range (IQR) technique.

None of these variables are Gaussian according to the Jarque-Bera test that checks the null hypothesis: the random variable follows the gaussian distribution. Indeed, none of the p-value are greater than 0. The Jarque Bera statistics is given by

$$JB = n\left(\frac{\hat{S}^2}{6} + \frac{(\hat{K}-3)^2}{24}\right)$$

and follows under the null hypothesis an asymptotic χ^2_2 distribution. $\hat{S}$ and $\hat{K}$ are the estimators of skewness and kurtosis for a given sample of n observations.

2.3 Outliers detection

Outliers detection techniques for skewed and heavy tailed distributions are rare. However, Gregg and Moore (2023) have just found a technique to deal with skewed, heavy tailed, monotonic and/or bimodal univariate distributions. This method named "STAR outliers" stands for "Skew and Tail-heaviness Adjusted Removal of Outliers". The approach relies on a generalized method based on IQR, as proposed by Verardi and Vermandele (2018), which defines the concept of symmetrical outlyingness (ASO). Initially, a ratio is calculated by subtracting each data point from the median and then dividing by the interquartile range. Subsequently, this ratio is scaled to fit within the unit interval and subjected to a probit transformation. If these ASO values are uniformly distributed on the unit interval before the probit transform, the final values are normally distributed. However, if it's not the case, this gives a transformed normal distribution with skew and tail heaviness. Such a distribution is called a Tukey g and h distribution first introduced by Tukey (1977) which adds two parameters to the normal distribution: a skewness and a kurtosis parameter.

The methodology's workflow is illustrated in Figure 2.3, while the specific transformations applied within this workflow are depicted in Figure 2.4. These figures have been sourced from the authors of STAR, as noted in Gregg and Moore (2023). The workflow diagram shows how the method addresses issues such as monotonicity, multimodality, and skewness. Notably, the step labeled "outliership test" is the one of the method outlined by Verardi and Vermandele (2018). Additionally, the detection of multimodality, based on the Cumulative Distribution Function (CDF), employs Hartigan's dip test as referenced in Hartigan and Hartigan (1985). The idea behing this test is to compare the largest distance (called dip) between the empirical CDF of the given sample and the CDF of an unimodal distribution. More precisely, this statistic measures the maximum difference between the empirical distribution function of the dataset and the best fitting unimodal distribution function. If the dataset is unimodal, this maximum difference will be relatively small.

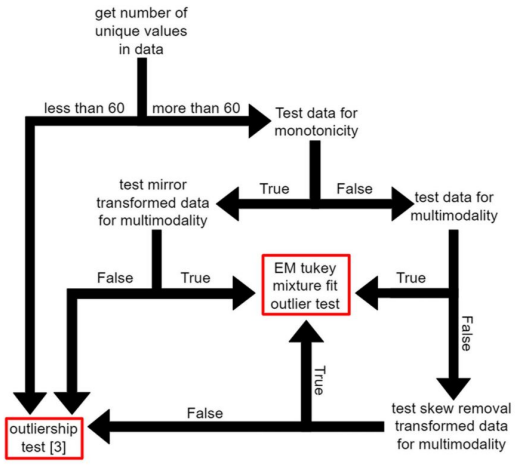

Figure 2.3: Workflow of STAR

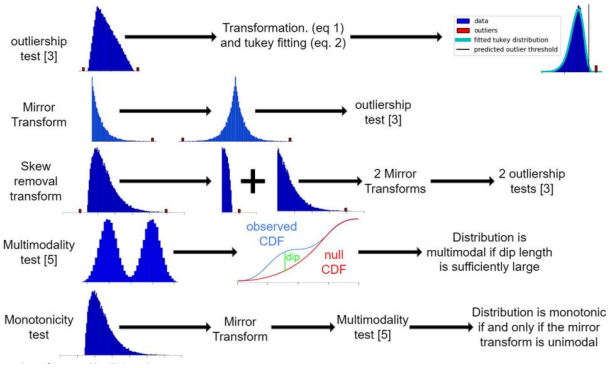

Figure 2.4: Transformations of STAR

When applying this outlier detection method to the financial ratios depicted in Figure 2.2 (interest coverage), the resulting analysis yields:

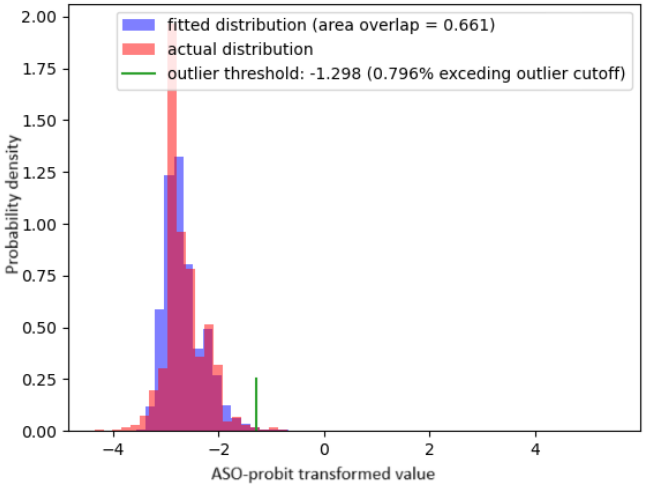

Figure 2.5: STAR outlier detection fot he interest coverage ratio

For non-transformed values, any data falling outside of the range [-16.6, 14] is classified as an outlier. In this study, outliers are managed by capping their values for each variable within the bounds of the interval established by STAR. Since this method operates on a univariate basis, this procedure is repeated for each feature. However, future work could explore extending the STAR method to its multivariate version, as discussed by Verardi and Vermandele (2018).

2.4 Imputation techniques

Some variables had missing values, prompting the need for an imputation technique. Initially, the coupon rate had 108 missing values, which were addressed using an Iterative Linear Imputer, as detailed in Warnauts (2023) and Buck (1960). This method employs a round-robin approach, estimating the missing features through linear regression based on the other features.

Additionally, two categorical variables required imputation: the rating and a boolean indicating the presence of collateral. For each of these variables, six different imputation methods were tested and compared to determine the most effective approach for filling in the missing values. The hyperparameters of each of these methods have been found by a 10 cross validation.

Since the variable *Collateral* is a boolean, a logistic regression has been tested as well as a Multilayer perceptron (Choudhury and Pal (2019)). The perceptron has been cross validated on its architecture (layers, neurons, dropouts) as well as on the learning rate of the optimizer.

For both models, the random forest has been tested on all other variables and solely on the numerical ones. Troyanskaya et al. (2001) have demonstrated the good performance of an imputer based on the KNN algorithm; therefore, this study will also compare the KNN imputer to other methods. The KNN imputer will undergo cross-validation for its neighborhood size, which is employed to determine the best prediction based on the closest elements in the dataset. Finally, a particular kind of Kmeans has been tested based on the Burt's distance (see Annex A) that takes into joint frequencies of modalities.

In Tables 2.5 and 2.6, the results of the comparison are given where the metric is the proportion of misclassified elements.

Table 2.5: RATING

Model	Error [%]
RF on numerical var.	**7.96**
RF on all variables	9.77
KNN	14.88
KMeans-Burt	10.82
Mode (Upper medium)	68.4

Table 2.6: Collateral

Model	Error [%]
Logistic	10.67
Perceptron	10.07
KNN	14.58
RF on numerical var.	**9.32**
RF on all variables	9.32
KMeans-Burt	9.47
Mode (False)	9.47

In both cases, random forest applied on numerical variables gives the lowest number of misclassifications. Hence, this technique has been used to impute these two categorical variables which allows to study their impact on the recovery rates in Figures 2.6 and 2.7.

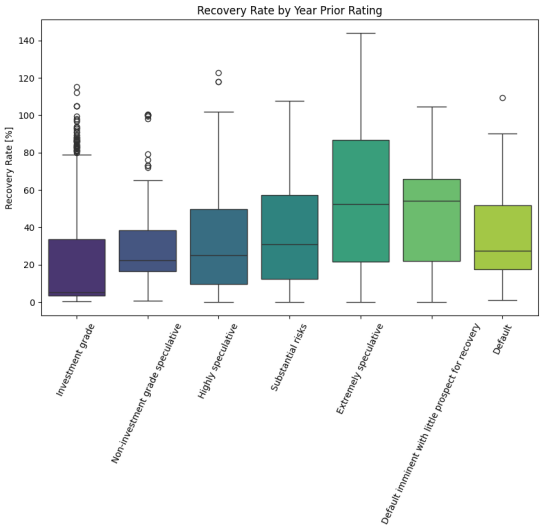

Figure 2.6: Boxplot of the recovery rates by ratings

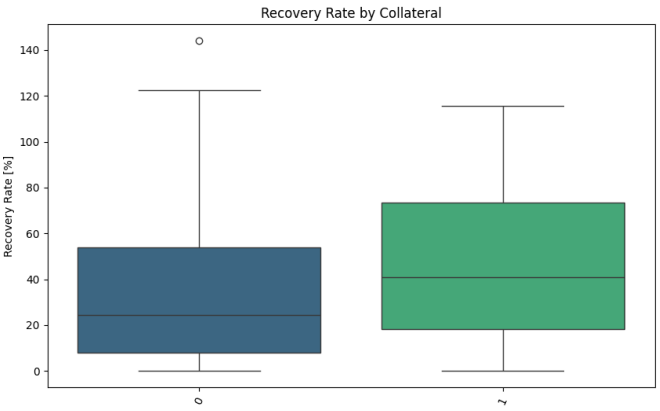

Figure 2.7: Impact of the collateral on the recovery rates

The presence of collateral helps to obtain a higher recovery rate. However, a better rating does not necessarily lead to a higher recovery rate, which may mean that the rating might not be accurate enough one year upfront or that other elements (such as the sector or the economic conditions) also play a role. Indeed, the rating often reflects the default probability and not the recovery rate.

Chapter 3

Algorithms and loss functions

This chapter covers the LGD modelling from cleaned data. Given the prevalent use of MSE in the LGD literature, one objective is to assess its performance against other loss functions. Thus, section 3.2 outlines the various loss functions compared in this study. These loss functions are optimized by the 10 machine learning algorithms described in section 3.3. In addition to basic, tree-based, and neural network models, the chapter details glassbox models, which are inherently explainable yet capable of capturing more complex trends than linear regression. Their explainability is further elaborated in the next chapter.

Furthermore, section 3.4 elaborates on autocalibration, a method that extends the Method of Marginal Totals, aiming to prevent transfers between subcategories and mitigate bias.

For a fair comparison, a metric developed in section 3.5 quantifies bond and portfolio valuation bias based on predefined allocations. The comparison results are discussed in section 3.6, while section 3.7 demonstrates the utilization of the tools developed in this chapter from a risk management perspective.

3.1 Data preparation

Ensuring the robustness and stability of models over time is a critical concern for regulators (EBA (2021)). In this regard, Olson et al. (2021) has demonstrated that employing sequential cross-validation yields superior stability and out-of-time performance compared to shuffled cross-validation. With sequential cross-validation, the chronological order remains unchanged. Accordingly, this work will use this sequential split with the training set being the first 80% of the dataset and the testing set the last 20%. Each time a validation set is used, the chronological order is kept unchanged. Like Dessain et al. (2023), this study will use a two-step grid search strategy to cross-validate the models. Initially, a broad grid is employed, followed by a more refined one. Moreover, all the variables are scaled between 0 and 1 and the categorical variables are dummified.

3.2 Loss functions

A loss function is the objective function to be minimized by an algorithm. In the LGD literature, the usual loss function is the Mean Squared Error (MSE) which assumes a Gaussian distribution for the residuals. Indeed, it can be shown that the MSE is the Gaussian deviance. However, the distribution of LGD does not follow a Gaussian distribution as explained by Gert Loterman (2013) which leads to the question of the relevance of the MSE. This work will compare this commonly used MSE to its rebalanced version and to the Beta deviance.

3.2.1 Mean squared error

The MSE penalizes big errors by squaring the residuals:

$$L_{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^n$ are the target and the estimate respectively.

Note that since residuals are squared, this loss function heavily emphasizes outliers, underscoring the necessity for a robust outlier detection technique. However, the literature on LGD has not yet addressed this aspect, whereas this study devotes a section to discussing the appropriate outlier detection technique for LGD datasets (refer to section 2.3).

3.2.2 Dense Loss

As mentioned in other works (Bastos (2010), Gert Loterman (2013), Gürtler and Hibbeln (2013)), the unconditional distribution of the recovery rates is imbalanced with a high proportion of low recovery rates. This is also the case for this dataset as shown in Figure 3.1 where the recovery rates are plotted before scaling.

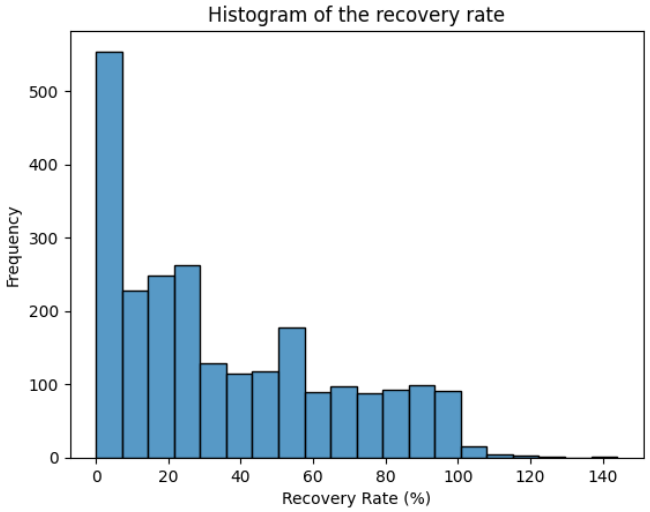

Figure 3.1: Unconditional distribution of the recovery rates

An imbalanced dataset can affect prediction and this can be corrected by a resampling approach such as SMOGN (Branco et al. (2017)). However, Steininger et al. (2021) proposed a weighting function that takes into account the imbalancement of the training dataset and has demonstrated the outperformance with respect to the resampling approach. Another advantage is that this approach leaves the dataset unaltered.

The goal of this method is to weight a given loss function $L(\boldsymbol{y}, \hat{\boldsymbol{y}})$ by a function $f_w(\alpha, \mathbf{y})$ that depends on a hyperparameter α. This work will consider $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = L_{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}})$ like in the original paper. Concerning the weight function, the underlying intuition is an inverse link with the frequency of a given target in the dataset like this can be easily applied for classification (Huang et al. (2016), Wang et al. (2017)). The idea is that a target in a densely populated part of the part should not penalise the loss function as a much as a target in a sparse area of the dataset bringing the idea of rebalancing the loss function. Consequently, a measure of frequency can be used but the difficulty is to apply it on a continuous domain. The authors solved this issue by fitting a Gaussian Kernel Density Estimator (Gaussian KDE) on the distribution of the recovery rates in the dataset. Therefore, for each target in the continuous domain, this KDE gives a measure of frequency in the dataset.

Additionally, the weight function must satisfy 4 constraints:

1. The weights of rarer samples must be higher than for more common ones.
2. The weights of rarer samples grow with the hyperparameter α and a null value for this α means a uniform weight function.
3. The weights are strictly positive.
4. The mean of the weights must be 1 over the samples to prevent an impact on the learning rate of an algorithm minimizing with this loss.

Steininger et al. (2021) proposed the following loss function $L_{DenseLoss}$:

$$L_{DenseLoss}(\alpha, \boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n} \sum_{i=1}^{n} f_w(\alpha, y_i) L(y_i, \hat{y}_i)$$

with:

$$f_w(\alpha, \mathbf{y}) = \frac{\max\left(\mathbf{1} - \alpha p'(\mathbf{y}), \epsilon\right)}{\frac{1}{N} \sum_{i=1}^{N} \left(\max\left(1 - \alpha p'\left(y_i\right), \epsilon\right)\right)}$$

where

- $p'(\mathbf{y})$ is a scaled version of the KDE's pdf between 0 and 1 as shown in Figure 3.2.
- ϵ makes the weights strictly positive (10^{-8})
- The weight is divided by the mean to satisfy the last constraint.

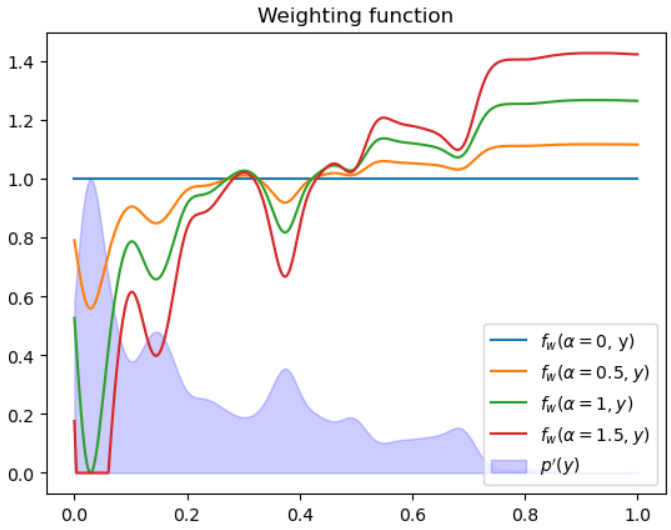

Figure 3.2: Weighting function for each scaled recovery rate

A Kernel Density Estimation (KDE) is a non parametric density given by:

$$p(y) = \frac{1}{n\,h} \sum_{i=1}^{n} K\left(\frac{y - y_i}{h}\right)$$

with h the bandwidth and K the kernel function. When K is the Gaussian density, this is called the Gaussian KDE. Interested readers are referred to Silverman (2018).

Figure 3.2 shows the shape of the weighting function for 4 values of α from which its impact can be observed. A higher α weights more heavily the sparser region of the dataset while a null value represent uniform weights giving $L_{DenseLoss} = L = L_{MSE}$ in this case.

This method has the advantage of being easily implemented since it doesn't modify the underlying loss function but rather incorporates weights. This work will use the 4 different values of the hyperparameter depicted in Figure 3.2.

3.2.3 Beta deviance

Due to its unit domain, the Beta distribution is a natural choice to model the recovery rate (or LGD). Besides, Siddiqi and Zhang (2004) based their approach on the CDF of the Beta distribution before applying an ordinary least square and this technique has been used frequently afterwards (Qi and Zhao (2011), Gert Loterman (2013)).

However, it is possible to model directly the recovery rates with a conditional beta distribution. Indeed, Smithson and Verkuilen (2006) show a reparametrization of the Beta distribution providing a more comprehensive insight into the parameters and applied this model to psychological data.

The Beta distribution $Y \sim \mathcal{B}(\omega, \tau)$, where ω and τ are strictly positive shape parameters,

can pose challenges in interpreting their conditional expectations. But, it's known that:

$$\mathbb{E}[Y] = \frac{\omega}{\omega + \tau} \quad \text{and} \quad \mathbb{V}[Y] = \frac{\omega\tau}{(\omega+\tau)^2(\omega+\tau+1)}$$

Therefore, it is possible to use a reparametrization with $\mu = \mathbb{E}[Y]$ and $\phi = \omega + \tau$ giving $\omega = \mu\phi$ and $\tau = \phi - \omega\phi$. The first parameter is the expectation of the Beta distribution while ϕ is a dispersion parameter. Indeed, for a fixed expectation μ, increasing ϕ decreases the variance since the variance can be written as: $\mathbb{V}[Y] = \frac{\mu(1-\mu)}{1+\phi}$.

Knowing that the pdf of $Y \sim \mathcal{B}(\omega, \tau)$ is given by:

$$f(y; \omega, \tau) = \frac{\Gamma(\omega+\tau)}{\Gamma(\omega)\Gamma(\tau)} y^{\omega-1}(1-y)^{\tau-1}$$

where Γ is the gamma function, the log-likelihood is given by:

$$log\left(L(y; \omega, \tau)\right) = log(\Gamma(\omega+\tau)) - log(\Gamma(\omega)) - log(\Gamma(\tau)) + (\omega-1)log(y) + (\tau-1)log(1-y)$$

ω and τ are replaced by their expressions such that the log-likelihood is expressed as a function of μ and ϕ.

Then, they proposed to use an extended GLM such that:

$$g_1(\mu_i) = \boldsymbol{\beta}^T \boldsymbol{X}_i \quad \text{and} \quad g_2(\phi_i) = \boldsymbol{\delta}^T \boldsymbol{X}_i$$

with g_1 and g_2 link functions, and $\boldsymbol{X}_i, \boldsymbol{\beta}, \boldsymbol{\delta} \in \mathbb{R}^m$. $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ are the m features for each of the n samples. Since the expectation μ must lie in the unit interval and ϕ must be positive, they proposed the logit and negative log links:

$$g_1(\mu_i) = log\left(\frac{\mu_i}{1-\mu_i}\right) = \boldsymbol{\beta}^T \boldsymbol{X}_i \quad \text{and } g_2(\phi_i) = -log(\phi_i) = \boldsymbol{\delta}^T \boldsymbol{X}_i$$

This formulation enables the modelling of heteroskedasticity through the dispersion submodel. Additionally, this study will explore a generalization of this form with:

$$g_1(\mu_i) = f_{\text{algo}_1}(\boldsymbol{X}_i) \quad \text{and } g_2(\phi_i) = f_{\text{algo}_2}(\boldsymbol{X}_i)$$

where f_{algo_1} and f_{algo_2} are the output of a given ML algorithm (such as neural networks).

3.3 Algorithms

Previous works have shown that more complex ML algorithms outperform simpler ones (Bastos (2010), Qi and Zhao (2011), Gert Loterman (2013), Kaposty et al. (2019), Bellotti et al. (2021), Gambetti et al. (2022a)). However, as previously mentioned, they have compared their models using (R)MSE or MAE, which may not be suitable for a valid analysis. Nevertheless, this work will apply commonly used algorithms with the loss functions already described in the previous section, in addition to the glassbox models. Glassbox models refer to models that are inherently interpretable, such as linear regression for the simplest one, to LocalGLMNet, Explainable Boosting Machine, and GAMI-Net, which are described in further details in their respective subsections in this section.

3.3.1 Basic models

The most basic models rely on linear regression with or without transformation. A lot of studies have used the classical ordinary least squares for the linear regression but the regularization helps to reduce the variance of the parameters and to obtain more stable results.

Linear regression

The linear regression used in this work will use a Lasso (L1) regularization. This helps to obtain smaller and more robust coefficients and tends to give a sparse model. The model is:

$$\hat{y}_i = \boldsymbol{\beta} \cdot \boldsymbol{X}_i$$

for i=1...n, with $\boldsymbol{\beta} \in \mathbb{R}^m$ the coefficient optimized by:

$$\min_{\beta} \; L(\boldsymbol{y}, \boldsymbol{X} \cdot \boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1$$

The hyperparameter λ is chosen based on a sequential cross-validation.

Logit regression

The logit regression employs a traditional linear regression model on a transformed version of the target variable, which is modified by the following element-wise transformation:

$$\begin{aligned} y' : [0,1] &\to \mathbb{R} \\ y &\mapsto y' = log\left(\frac{y}{1-y}\right) \end{aligned}$$

3.3.2 Tree based models

Bastos (2010) and Qi and Zhao (2011) have explored regression trees for LGD regression. However, traditional regression methods might overfit the training set, whereas random forest is less prone to overfitting. Boosted trees can also be parameterised based on cross-validation to mitigate overfitting.

Random forest

Random Forest is an ensemble learning method that builds multiple decision trees during training. Each tree is trained on a random subset of the training data and features, reducing overfitting and improving generalization. Predictions are made by averaging the outputs of individual trees for regression. By combining the predictions of multiple trees, Random Forest achieves high accuracy and robustness. The hyperparameters like the

number of trees, the depth and the minimum samples in each node are set by sequential cross-validation. The algorithm is implemented in the scikit-learn python package. Further details are given in Breiman (2001) and Geurts et al. (2006).

Gradient boosting machine

Gradient Boosting Machine (GBM) constructs an ensemble of weak learners (decision trees) sequentially by minimizing the ensemble's loss function through gradient descent. Each weak learner is trained to correct the errors of the existing ensemble, enhancing predictive accuracy. Regularization methods like shrinkage and tree pruning are employed to prevent overfitting by limiting the complexity of individual trees. GBM's iterative approach incrementally improves model performance by optimally fitting the residuals of previous iterations. Further details can be found in Denuit et al. (2020).

Note that two popular implementations exist in Python: LightGBM (Ke et al. (2017)) and XGBoost (Chen and Guestrin (2016)). XGBoost uses a level-wise tree growth strategy, where it splits nodes one level at a time across all features, whereas LightGBM uses a leaf-wise tree growth strategy, which splits nodes based on the leaf with the maximum reduction in loss, leading to faster training times for large datasets. Both allow L1 regularization which has been set as well as the other hyperparameters by sequential cross-validation.

3.3.3 Feed forward neural networks

Neural networks are computational models inspired by the human brain's neural structure, composed of interconnected nodes (neurons) organised in layers. They learn from data by adjusting connections (weights) between neurons to perform tasks such as classification, regression, and pattern recognition. Feedforward neural networks (FNN) are a fundamental type of artificial neural network where information flows in one direction, from input nodes through hidden layers to output nodes, without any feedback loops.

Wüthrich and Merz (2023) describe a FNN layer l with activation function ϕ by:

$$\begin{aligned} z^{(l)} : \{1\} \times \mathbb{R}^{q_{l-1}} &\to \{1\} \times \mathbb{R}^{q_l} \\ z &\mapsto z^{(l)}(z) = \left(1, z_1^{(l)}(z), \ldots, z_{q_l}^{(l)}(z)\right)^\top \end{aligned}$$

having neurons for $1 \leq j \leq q_l$

$$z_j^{(l)}(\boldsymbol{z}) = \phi\left\langle \boldsymbol{w}_j^{(l)}, \boldsymbol{z} \right\rangle = \phi\left(\sum_{l=0}^{q_{l-1}} w_{l,j}^{(l)} z_l\right),$$

with weights $\boldsymbol{w}_j^{(l)} = \left(w_{l,j}^{(l)}\right)_{0 \leq l \leq q_{l-1}} \in \mathbb{R}^{q_{l-1}+1}$. A layer is therefore a transformation (ϕ) of a linear mapping of the input and the weights taking into account an intercept. Based on this foundation, we can define a FNN as:

$$\begin{aligned} \boldsymbol{z}^{(d:1)} : \{1\} \times \mathbb{R}^m &\to \{1\} \times \mathbb{R}^{q_d} \\ \boldsymbol{x} &\mapsto \boldsymbol{z}^{(d:1)}(\boldsymbol{x}) = \left(\boldsymbol{z}^{(d)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}) \end{aligned}$$

with $\boldsymbol{x}$ a sample of $\boldsymbol{X}$.

Like for the linear regression, a L1 regularisation is implemented by adding a penalty term to the loss function during training, which penalises large parameter values (large weights). This kernel regularisation is used to mitigate overfitting and promote sparsity in the model parameters. Indeed, the model is encouraged to shrink less important weights towards zero, effectively pruning them from the model. This simplifies the model and makes it more interpretable, while also reducing the risk of overfitting by discouraging the model from becoming overly complex.

A recursive version of the networks has been incorporated, wherein three networks are stacked sequentially, and the prediction of one network is used as an input in the subsequent one alongside the features.

Dropout

Another approach employed in neural networks to mitigate overfitting is Dropout. It functions by randomly disabling a portion of neurons within the network during the training phase. This means that the output of these neurons, as well as any connections associated with them, are temporarily ignored during a training iteration. Dropout is applied independently to each neuron with a certain probability that can be different for each layer.

During training, dropout effectively creates an ensemble of neural networks, as different sets of neurons are active or inactive at each iteration. This forces the network to learn more robust and generalizable features, as it cannot rely too heavily on any particular set of neurons. Dropout acts as a form of regularization by adding noise to the network and preventing it from memorising the training data too closely.

At test time, dropout is typically turned off, and the full network is used to make predictions.

Hyperparameters are optimised through sequential cross-validation to ensure reliable results. The architectures tested range from 1 to 3 hidden layers, with the number of neurons being a power of 2. Additionally, the subsequent layer cannot have a larger number of neurons. The regularization parameter is tested across values of $[0, 0.2, 0.4, 0.6]$, while the learning rate is cross-validated as a power of 10. Thanks to multiprocessing and GPU acceleration, more than 2000 architectures have been evaluated.

Further details about neural networks are given in Denuit et al. (2019b).

3.3.4 Glassbox models

Glassbox models are designed with explicit interpretability, producing precise and easily understandable explanations. A drawback of more complex models like forests, GBM, or FNNs is their lack of intrinsic and accurate explanations. Post-hoc techniques such as LIME or Shapley values (discussed in section 4.2) provide approximate explanations.

However, this doesn't imply they shouldn't be used; instead, they should be applied strategically to enable explicit interpretability. This sections will cover three algorithms based on this idea.

First, LocalGLMNet extends the traditional Generalized Linear Model (GLM) by incorporating neural networks. This hybrid approach aims to leverage the flexibility of neural networks while retaining the interpretability characteristic of GLMs. Finally, the study explores the Explainable Boosting Machine (EBM) and the GAMI-Net models. Both extend the Generalized Additive Model (GAM) by incorporating pairwise interactions, with a focus on selecting the most relevant ones. However, while EBM replaces splines in GAMs with boosting, GAMI-Net replaces them with neural networks. Notably, GAMI-Net imposes stricter constraints than EBM, aligning with the goal of achieving a more interpretable model.

LocalGLMNet

The classical GLM is given by:

$$g(y) = \boldsymbol{\beta} \cdot \boldsymbol{X}_i + \beta_0$$

with $\boldsymbol{\beta} \in \mathbb{R}^m, \beta_0 \in \mathbb{R}$ the parameters to optimise. This is a generalised version of the linear model since there is a strictly monotone link function g that can be called a canonical link function under some conditions (refer to Denuit et al. (2019a)). This model is common in actuarial works to model in particular when modelling the claim frequency in non life insurances with a log link function (canonical link of Poisson distribution). This log link function gives a multiplicative structure in the response. Thus, GLMs allow to obtain a better model than a pure linear model ($g(x) = x$) while keeping the intrinsic interpretability. Indeed, the interpretability of GLMs is based on the magnitude of their parameters $\boldsymbol{\beta}$.

Therefore, Wüthrich and Merz (2023) proposed a way to incorporate Neural Networks (FNNs in particular) in GLMs by making $\boldsymbol{\beta}$ varying as a function of the inputs given by a FNNs. This model uses a skip connection that uses the input in deeper layers. Locally, near a given sample, $\boldsymbol{\beta}$ should be nearly constant which is why this model is called a Local GLM Network. In line with the idea to keep the generalised linear form, its mathematical representation given by the following:

$$\boldsymbol{x} \mapsto g(y) = \beta_0 + \langle \boldsymbol{\beta}(\boldsymbol{x}), \boldsymbol{x} \rangle = \beta_0 + \sum_{j=1}^{m} \beta_j(\boldsymbol{x}) x_j$$

with $\boldsymbol{\beta}$ the attention layer given by:

$$\begin{aligned} \boldsymbol{\beta} : \mathbb{R}^m &\to \mathbb{R}^m \\ \boldsymbol{x} &\mapsto \boldsymbol{\beta}(\boldsymbol{x}) \overset{\text{def.}}{=} \boldsymbol{z}^{(d:1)}(\boldsymbol{x}) = \left(\boldsymbol{z}^{(d)} \circ \cdots \circ \boldsymbol{z}^{(1)}\right)(\boldsymbol{x}). \end{aligned}$$

following the definitions of a FNNs given in section 3.3.3. An example of this network can be visually represented as follows:

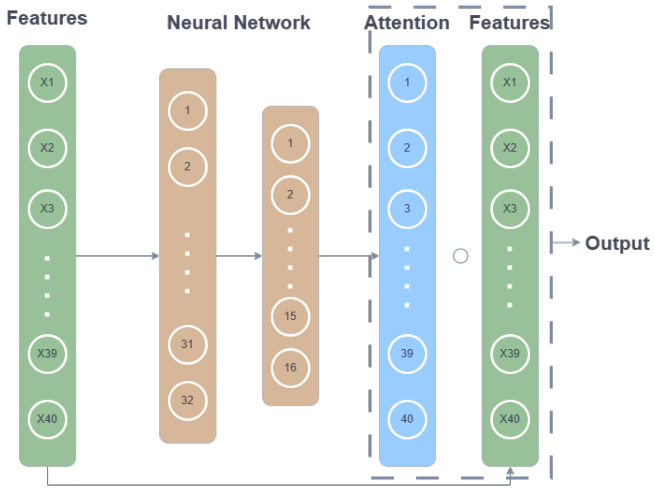

Figure 3.3: Architecture of LocalGLMNet

Based on the attention layer, the interpretability comes from the study of of each individual term $\beta_j(\boldsymbol{x})x_j$ such that:

- $\beta_j(\boldsymbol{x})x_j = 0$ means that this term can be removed. Note that it does not mean that the feature x_j should be dropped since it can influence other β_i $(i \neq j)$.
- $\beta_j(\boldsymbol{x})x_j = K$ means that the coefficient should remain constant, exactly like in a GLM.
- $\beta_j(\boldsymbol{x})x_j = \beta_j(x_j)x_j$ means that this term does not interact with the other features. This does not mean that there is no interaction between x_j and any other feature x_i since it can be that $\beta_i(\boldsymbol{x})x_i$ depends on x_j. Instead, this should be understood as the fact that the other features do not influence the importance of the variable x_j.

However, the balance property does not hold anymore such that it's recommended to apply a bias correction step. This has been implemented and the hyperparamaters such as the architecture, the learning rate, the L1 regularization coefficient have been selected by sequential cross validation as explained for the FNNs.

An example of interpretability will be developed in its own part of this work (section 4.1.1).

Explainable boosting machine

Generalized Additive models (GAMs) have the following form:

$$g(\mathbb{E}[y]) = \beta_0 + \sum_j f_j(x_j)$$

with g the link function and f_j a function for each term. The interpretability of GAM arises from the ability to discern the importance of each feature from the graph of f_j (more on GAMs in Denuit et al. (2019a)). However, a limitation of this approach is the

absence of interactions, which more complex models such as forests and neural networks are capable of capturing.

However, Lou et al. (2013) have proposed a new version of GAMs that takes into account pairwise interactions named "GA^{2}Ms":

$$g(\mathbb{E}[y]) = \beta_0 + \sum_j f_j(x_j) + \sum_{i,j} f_{i,j}(x_i, x_j)$$

This remains as interpretable as GAMs since a pairwise interaction can be described by a heatmap as it will be shown in a dedicated part (section 4.1.2).

First, the algorithm will fit a classical GAM without any interaction. The main effect function f_j can be chosen as splines (Wood (2003)) or tree based methods such as regression trees or tree ensembles (Bauer and Kohavi (1999)). Fitting these functions can be realized based on backfitting (Hastie (2017)) or gradient boosting (Friedman (2001)). But Lou et al. (2012) have shown that using shallow regresion trees with gradient boosting outperforms the other methods mentioned above. Therefore, the EBM method builds f_j using gradient boosting and bagging. The procedure trains f_j for each feature $j \in \{1, ..., m\}$ one at a time in round robin fashion with a small learning rate. This has been chosen such that the feature order does not influence the result and the colinearity is mitigated. At this point, the main effects have been fitted and interactions effects will be calibrated on the residuals while maintaining the one-dimensional functions fixed.

However, the primary challenge lies in the number of pairs to consider when comparing interactions, as this requires an $\mathcal{O}(m^2)$ complexity, which may not be practical for large datasets. Fortunately, Lou et al. (2013) have also proposed an efficient pairwise detection algorithm named "FAST". The concept involves calculating the decrease in the residual sum of squares (RSS) for each pair (x_i, x_j) when modelling $f_{ij}(x_i, x_j)$ instead of $f_i(x_i) + f_j(x_j)$. A significant interaction will result in a considerable decrease in RSS, while ensuring that the already fitted $f_i(x_i)$ and $f_j(x_j)$ accommodate the presence of $f_{ij}(x_i, x_j)$. While constructing $f_{ij}(x_i, x_j)$ can be computationally expensive, the idea is to develop a cost-effective approximation instead.

The FAST algorithm divides each feature domain into q_j bins, resulting in four quadrants for problems with two variables (pairwise interaction). The interaction predictor T_{ij} is constructed by averaging the points within each quadrant. This process utilises dynamic programming, wherein the value at a cut is computed based on the value at a previously computed smaller cut. The cuts then iterate to find the optimal interaction predictor for the feature pair. Subsequently, the top K pairs are selected, and the full function $f_{i,j}$ is constructed similarly to principal effects f_j. The authors have demonstrated that the FAST algorithm has a complexity of $\mathcal{O}(n + b^2)$ for each pair, where b represents the number of bins. While not explicitly stated by the authors, this implies that maintaining b constant for increasing problem sizes (n) results in linear complexity with respect to dataset size. The default number of bins is set to 256.

For the hyperparameter tuning, the number of bins, interactions, leaves in each tree, minimal sample in leaf and inner bags has been realised by sequential cross validation.

GAMI-Net

Yang et al. (2021) proposed another implementation of the GA²M algorithm where the idea is to fit the functions f_j and f_{ij} with neural networks. The FAST algorithm for pairwise interaction detection is also applied. However, it's no more a two stages algorithm but a three stages algorithm. Indeed, a new stage has been added and the predictor must satisfy 3 constraints (supported by the meta-analysis study Li et al. (2006)):

- **Sparsity** involves retaining only the crucial factors and pruning both the main and interaction predictors. The aim is to eliminate effects with minimal importance, which aids in mitigating overfitting and reducing the computational costs. The importance of each effect is determined by its sample variance, and effects with low importance are set to zero.
- **Heredity** dictates that a pairwise interaction is only considered if at least one of its corresponding main effects (higher-order effect) demonstrates sufficient importance.
- **Marginal clarity** makes the main and interaction effects orthogonal. The goal is to avoid that one absorbs the other which leads to multiple representation of the same model. This helps the model to stay stable overtime and further improve interpretability by clearly separating the two effects. An interaction and its parent main effect are orthogonal if:

  $$\int f_j(x_j) f_{ij}(x_i, x_j) dF(\boldsymbol{x}) = 0$$

  where the joint cumulative distribution function is given by F. Empirically, it's possible to define a degree of orthogonality by:

  $$\Omega(f_j, f_{ij}) = \frac{1}{n} |\sum f_j(x_j) f_{ij}(x_i, x_j)|$$

  The smaller the degree of orthogonality, the more orthogonal the main and interaction effects are. However, it may be possible that those effects are not exactly orthogonal so that the degree of orthogonality is not exactly 0. Hence, the degree of orthogonality is incorporated into the objective function as a penalisation term, with its strength determined by a hyperparameter λ.

The first stage involves training the main effects by sequentially adding effects from the most important to the least important while ensuring the sparsity constraint is met. Practically, the addition of effects stops when the validation loss is smaller or equal than $(1 + \eta) min(l_0, ..., l_m)$ where l_i is the validation loss when i variables have been added and η an hyperparameter tuned through cross-validation. In the second stage, interactions satisfying the heredity constraint are added, reducing the number of interactions to consider. Pruning follows the same procedure as for main effects, with the same relative tolerance η. For example, when fitting the MSE, GAMI-Net has chosen the following number of effects:

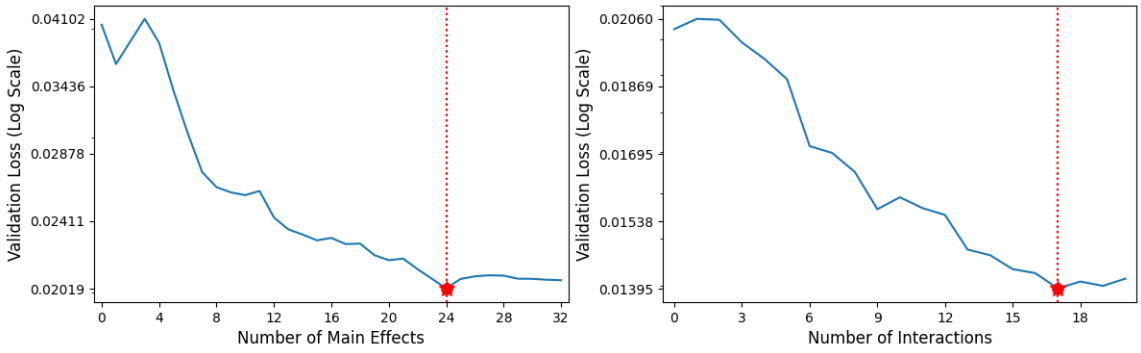

Figure 3.4: Selection of effects with the sparsity constraint

In the final step, both subnetworks are retrained simultaneously, while the marginal clarity constraint is still enforced on the selected effects. This helps mitigate bias stemming from the removal of trivial main and interaction effects. This step further improves the accuracy of the model as shown in Figure 3.5

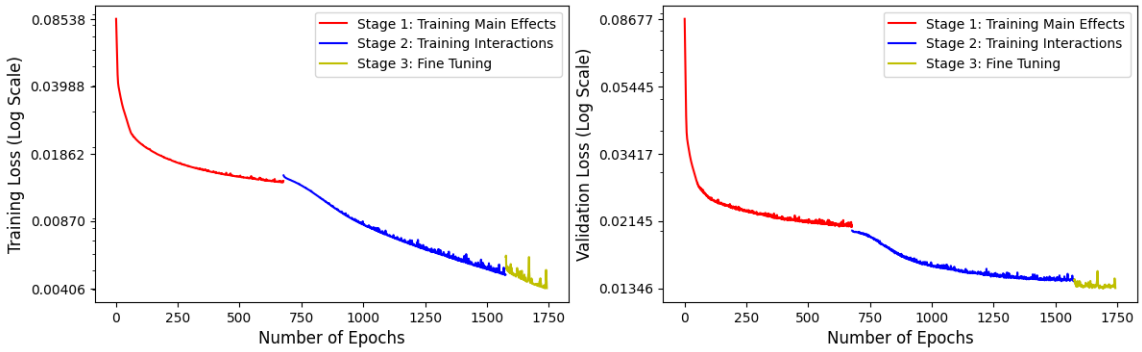

Figure 3.5: Effect of the three stages when training GAMI-Net

Interpreting the model involves looking at the graphs of f_j, similar to classical GAMs, and studying heatmaps for the interactions f_{ij} as for EBMs. Further details on interpreting GAMI-Net will be provided in its dedicated section (section 4.1.3).

The architectures of the main and interaction effects subnetworks have been determined through cross-validation, along with the learning rates, maximum number of interactions, and the regularization parameter λ.

3.4 Autocalibration

In contrast to the previous algorithms, this section discusses a post-hoc technique applied after any of these algorithms. Autocalibration aims to reinstate both global balance and local equilibrium, adhering to the principles of the Method of Marginal Totals (MMT). The goal is that the sum of the predictions ($\hat{y}$) closely aligns with the sum of the targets (y). Thus, the concept of MMT revolves around the notion that an acceptable premium set should mirror the observed outcomes within sub-portfolios corresponding to different

risk factors (like gender or age in non-life insurance) as well as across the entire portfolio. Essentially, this implies that there should be no transfer from one subgroup to another and no bias.

Global balance and local equilibrium are inherently satisfied in GLMs but not for neural networks or GBMs. Autocalibration solves this problem by adding an extra step and keeps the accuracy of these flexible algorithms. In context of insurance, a premium is computed by a predictor $\hat{\pi}$ as:

$$\hat{\pi}(\boldsymbol{x}) \approx \mu(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{X} = \boldsymbol{x}]$$

However, this may be a biased prediction. In order to solve this problem, a new notion has to be introduced defined by Krüger and Ziegel (2019): "X is an autocalibrated forecast of y if $\mathbb{E}[Y|X] = X$ almost surely". This means that the prediction X of Y can be used 'as is' without bias correction. Denuit et al. (2021) introduced this concept in insurance pricing by stating that an autocalibrated predictor must satisfy $\hat{\pi}(\boldsymbol{x}) = \mathbb{E}[y|\hat{\pi}(\boldsymbol{x})]$ meaning that there is no transfer between sub-portfolios.

Furthermore, they have proposed a way to autocalibrate a given predictor $\hat{\pi}$ to obtain a balance-corrected predictor:

$$\hat{\pi}_{BC}(\boldsymbol{x}) = \mathbb{E}[y|\hat{\pi}(\boldsymbol{x})]$$

whose expectation is $\mathbb{E}[y]$ as shown in their proposition 5.1. This implies that it is feasible to acquire a balance-corrected version of a given predictor $\hat{\pi}$ derived from neural networks or GBMs for example. While the bias correction step explained in the neural networks section restores the global balance property, it does not address local balance. To address this, within a neighborhood defined by the predictor $\hat{\pi}$, a solution is to enforce a local GLM likelihood equation, aligning the predictor with the constraints of MMT:

$$\sum_{i=1}^{n} \nu_i(\hat{\pi}(\boldsymbol{x}))y_i = \sum_{i=1}^{n} \nu_i(\hat{\pi}(\boldsymbol{x}))\hat{\pi}_{BC}(\boldsymbol{x})$$

for a given weight function ν. The weight function employed can take various forms such as tricube, rectangular, Gaussian or Epanechnikov. However, in line with the authors, this study utilizes the rectangular function. In this setup, within a specified window, the weights cancel out, leading to a principle of mutuality, while outside the window, the weights are set to zero. The size of the window is determined for each predictor/algorithm (those detailed in previous sections) through cross-validation to minimize bias, as shown in details in Appendix C.

3.5 Economic metric

This metric aims to compare the algorithms and loss/fitting functions on a fair basis. Finding a fair metric to compare the results can be quite challenging but this work explores the impact onto the economic valuation of these bonds through a valuation bias. Therefore, the first step is to fit each model (section 3.3) with each one of the loss functions (section 3.2). Additionally, the second step is to obtain autocalibrated versions of these fitted

models to compare the models with and without autocalibration (section 3.4). Once all these models are fitted, the economic metric will give the valuation bias on separate bonds and on the overall portfolio.

The economic metric introduced in this study is a bond valuation bias resulting from errors in estimating the LGD.

The bond valuation formula is as follows:

$$\pi^b := (1 - LGD) \cdot PD + \mathbb{E}[\text{valuation}|\text{No default}] \cdot (1 - PD)$$

where PD is the probability of default and $\mathbb{E}[\text{valuation}|\text{No default}]$ independent of LGD. However, the true value of LGD is not know in advance such that an estimate $L\hat{G}D$ is computed and the valuation is approximated by:

$$\hat{\pi}^b := (1 - L\hat{G}D) \cdot PD + \mathbb{E}[\text{valuation}|\text{No default}] \cdot (1 - PD)$$

From this expression, the valuation bias B due to the error in estimating the LGD is:

$$B := \hat{\pi}^b - \pi^b = PD \cdot (LGD - L\hat{G}D)$$

The sign of this expression is informative since a negative value indicates that the bond is undervalued while a positive sign suggests an overvaluation.

From a portfolio perspective, the valuation of multiple bonds is calculated, aiming to assess the overall portfolio value. In this context, a portfolio manager employs a specific allocation vector $\boldsymbol{w} \in [0, 1]^n$, where $\mathbf{1}^T\boldsymbol{w} = 1$, representing the allocation weights of individual bonds within the portfolio. This bias can be computed as:

$$\boldsymbol{B}^T \cdot \boldsymbol{w} := \Big(\boldsymbol{PD} \odot (\boldsymbol{LGD} - \boldsymbol{L\hat{G}D})\Big) \bullet \boldsymbol{w}$$

where $\boldsymbol{PD}, \boldsymbol{LGD}$ and $\boldsymbol{L\hat{G}D}$ are vectors of size n. This definition uses an Hadamard (element-wise) product and enables the bias valuation of each bond in the portfolio to offset one another.

This expression is highly flexible and can be rearranged for specific allocations such as equally weighted or label-weighted portfolios. A label-weighted portfolio follows an allocation that weights some characteristics of bonds. For example, the weight of European bonds can be fixed by the regulator or the manager can decide to limit its exposure to some bond ratings.

Proposition 1 *The average valuation bias over all non empty equally weighted portfolios of any size is equivalent to the average valuation bias over all the bonds of the dataset:*

$$\frac{1}{2^n - 1} \sum_{\boldsymbol{w}} \boldsymbol{w}^T \cdot (\boldsymbol{PD} \odot (\widehat{\boldsymbol{LGD}} - \boldsymbol{LGD})) = \frac{\sum_{i=1}^n PD_i(\widehat{LGD}_i - LGD_i)}{n}$$

Proof: This proposition is based on the fact that there exist $2^n - 1$ allocations. Indeed, denote $\mathbf{a} \in \{0, 1\}^n$ a decision vector of binary values where a_i is 1 if the bond i is in the

portfolio ($\boldsymbol{w}$ is in fact given by $\boldsymbol{a}/\sum_i a_i$). There exist 2^n different decision vectors $\boldsymbol{a}$ but the empty allocation is not acceptable since the constraint $\mathbf{1}^T\boldsymbol{w} = 1$ must be satisfied. Therefore, it is an average over all allocation $\boldsymbol{w}$ that can be developed as:

$$\begin{aligned}\frac{1}{2^n-1}\sum_{\boldsymbol{w}} \boldsymbol{w} \bullet (\boldsymbol{PD} \odot (\widehat{\boldsymbol{LGD}} - \boldsymbol{LGD})) &= \frac{1}{2^n-1}\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{\binom{n-1}{k-1}}{k} PD_i(\hat{LGD}_i - LGD_i) \\ &= \frac{1}{2^n-1}\left(\sum_{i=1}^{n} PD_i(\hat{LGD}_i - LGD_i)\right)\sum_{k=1}^{n}\frac{\binom{n-1}{k-1}}{k}\end{aligned}$$

The first sum allows to give a weight w_i for each asset i. The second sum compute all the bias for all portfolio of size k where these assets are chosen among the $n-1$ other assets than i. Thus, this sum will compute the bias for the asset i over all portfolios in which it is present. Moreover, in a portfolio of size k, it will receive a weight of $1/k$ since the portfolio is equally weighted. This formulation allows to generalise the result for all equally weighted sub-portfolio.

Then, using combination properties:

$$\begin{aligned}\frac{1}{2^n-1}\sum_{k=1}^{n}\frac{\binom{n-1}{k-1}}{k} &= \frac{1}{2^n-1}\sum_{k=1}^{n}\frac{(n-1)!}{k!(n-k)!} \\ &= \frac{1}{2^n-1}\frac{1}{n}\sum_{k=1}^{n}\frac{n!}{k!(n-k)!} \\ &= \frac{1}{n(2^n-1)}\sum_{k=1}^{n}\binom{n}{k} \\ &= \frac{1}{n(2^n-1)}\left(-1+\sum_{k=0}^{n}\binom{n}{k}\right) \\ &= \frac{1}{n(2^n-1)}\left(-1+2^n\right) = \frac{1}{n}\end{aligned}$$

which ends the proof ■

This result means that taking the average valuation bias assumes an equally weighted portfolio. However, this may not be representative for other allocations in which case this result is not relevant. This observation leads to the need to define a label-weighted allocation.

Proposition 2 *Suppose a label-allocation $\boldsymbol{w}^l$ is given for which a weight $\boldsymbol{w}_i^l$ is the weight of a group of bonds (a label) within a portfolio such that $\mathbf{1}^T\boldsymbol{w}^l = 1$. Inside each group of a given label, this sub portfolio can be any equally weighted portfolio of bonds of this label.*

The average valuation bias for a portfolio following this label-allocation is:

$$\boldsymbol{w}^l \bullet \overline{\boldsymbol{B}}^l$$

where $\overline{\boldsymbol{B}}^l$ is the average bias inside the label l.

Proof: As in the previous proposition, the average valuation bias can be written as the average valuation bias over all possible allocation $\boldsymbol{w}$:

$$\frac{1}{|W|}\sum_{\boldsymbol{w}} \boldsymbol{w} \bullet \boldsymbol{B}$$

where W is the set of all allocations $\boldsymbol{w}$ (at bonds level) that satisfy the allocation $\boldsymbol{w}^l$ (at the labels level).

This sum can be decomposed along the labels:

$$\frac{1}{|W|}\sum_{\boldsymbol{w}} \boldsymbol{w} \bullet \boldsymbol{B} = \frac{1}{|W|}\sum_{l}\sum_{\boldsymbol{w}} \boldsymbol{w}_l \bullet \boldsymbol{B}_l$$

where $\boldsymbol{w}_l$ (resp. $\boldsymbol{B}_l$) are the elements of $\boldsymbol{w}$ (resp. $\boldsymbol{B}$) referring to the bonds of label l.

The sum over $\boldsymbol{w}$ can be further developed for a given label l:

$$\frac{1}{|W|}\sum_{\boldsymbol{w}} \boldsymbol{w}_l \bullet \boldsymbol{B}_l = \frac{1}{|W|}\sum_{\boldsymbol{w}_l} N_l(\boldsymbol{w}_l)\boldsymbol{w}_l \bullet \boldsymbol{B}_l$$

in which $N_l(\boldsymbol{w}_l)$ refers to the number of allocations $\boldsymbol{w}$ in which the bonds of label l have the allocation $\boldsymbol{w}_l$. However, this number is constant over all the $\boldsymbol{w}_l$ since all combinations are tested and the allocation in $\boldsymbol{w}_l$ do not interfere with the remaining allocation in $\boldsymbol{w}$. The label receives its weight through $\boldsymbol{w}^l$ and all the equally weighted allocations are taken into account inside this label l. Since $N_l(\boldsymbol{w}_l)$ is constant, it can be rewritten as $N_l(\boldsymbol{w}_l) = N_l$ from which, the development becomes:

$$\begin{aligned}
\frac{1}{|W|}\sum_{\boldsymbol{w}} \boldsymbol{w}_l \bullet \boldsymbol{B}_l &= \frac{1}{|W|}\sum_{\boldsymbol{w}_l} N_l(\boldsymbol{w}_l)\boldsymbol{w}_l \bullet \boldsymbol{B}_l \\
&= \frac{N_l}{|W|}\sum_{\boldsymbol{w}_l} \boldsymbol{w}_l \bullet \boldsymbol{B}_l \\
&= \frac{1}{|W_l|}\sum_{\boldsymbol{w}_l} w^l_+ \frac{\boldsymbol{w}_l}{w^l_+} \bullet \boldsymbol{B}_l
\end{aligned}$$

where $N_l/|W| = 1/|W_l|$ with W_l is the set of all possible $\boldsymbol{w}_l$ and w^l_+ the sum of $\boldsymbol{w}_l$. This w^l_+ is constant for all $\boldsymbol{w}_l$ and equal to the corresponding entry in $\boldsymbol{w}^l$. Then, the sum satisfies the assumptions of Proposition 1 which means that:

$$\frac{1}{|W_l|}\sum_{\boldsymbol{w}_l} \frac{\boldsymbol{w}_l}{w^l_+} \bullet \boldsymbol{B}_l = \overline{B}_l$$

where $\overline{B}_l$ is the average of $\boldsymbol{B}_l$ over all bonds in the label l.

From this development, it comes that:

$$\frac{1}{|W|}\sum_{\boldsymbol{w}} \boldsymbol{w} \bullet \boldsymbol{B} = \sum_{l} w^l_+ \overline{B}_l = \boldsymbol{w}^l \bullet \overline{\boldsymbol{B}}^l$$

which ends the proof ■

This is a powerful result since there is no restriction on the labels so that they can be chosen as the ones that suit the best a portfolio manager. Indeed, the average valuation bias

over all portfolios satisfying its allocation will be computed with an algorithmic complexity of $\mathcal{O}(n)$ without having to generate each of these portfolios (exponential complexity). Moreover, the expression of the bias $B = PD \cdot (LGD - \hat{LGD})$ can be replaced by any other expression needed.

In this work, a, existing allocation will be used: the allocation of Belfius Plan High used in the L'Oréal pension fund whose allocation is:

Répartition des obligations

Qualité du credit | Fourchette du coupon | Échéancier

Qualité	Fonds %
AAA	8,18
AA	20,00
A	29,83
BBB	29,95
BB	7,03
B	0,73
inférieure à B	0,24
sans cote	4,05

Figure 3.6: Allocation of Belfius Plan High

This implies that the labels are determined by the ratings such that the vector $\boldsymbol{w}_l$ is given in Figure 3.6. Employing Proposition 2, to derive the average valuation bias across all portfolios conforming to Belfius' allocation, the average bias is calculated for each label, and this outcome is multiplied by the allocation vector $\boldsymbol{w}_l$ through scalar product. Moreover, if this institution adheres to additional regulations like geographical allocation, this can be factored in by establishing appropriate labels.

For each bond, PD has been set as the default probability according to its rating. Indeed, PD can be computed for each rating as done by Dessain et al. (2023).

3.6 Results

This section present a summarised version of the full results given in annex D. All the models have been fitted on the described loss function and their economic result is compared through the economic metric explained in the previous section on equally weighted (EW) and Belfius' allocation. All these results are also given with and without autocalibration to observe the impact of this method.

To follow the same order as for the descriptions of the loss functions and algorithms, the loss functions will be studied first. The Table 3.1 shows the average rank of each of the 5 loss functions over the 10 models. Specifically, for each model, the loss functions were ranked based on their absolute bias for two allocation.

	MSE	Dense Loss $\alpha = 0.5$	Dense Loss $\alpha = 1$	Dense Loss $\alpha = 1.5$	Beta deviance
EW	2.6	2	2.6	3.6	3.25
Belfius	3.4	2.1	2.6	2.5	3.5

Table 3.1: Average rank for each loss function over the models

The table illustrates the superior performance of the Dense Loss functions with $\alpha = 0.5$ for both allocations. This suggests that better results can be achieved by adjusting the loss with weights. It also implies that since the LGD literature typically evaluates LGD modelling, comparison, and analysis based on MSE, these results may not be economically relevant and indicate a larger valuation bias than other loss functions. Moreover, while MSE does not exhibit poor performance with an equally weighted allocation, its performance notably declines with Belfius' allocation compared to other losses. Additionally, it appears that regardless of the value of α, the Dense Loss outperforms MSE. In practice, when employing a specific allocation (such as Belfius'), MSE might yield subpar results, emphasising the importance of selecting the appropriate α, which is as crucial as choosing the right model. Across both allocations, the Dense Loss with $\alpha = 0.5$ achieves the best performance and remains consistently strong in terms of relative performance, akin to $\alpha = 1$.

Furthermore, the comprehensive results table for Belfius' allocation (Table A6 in the annex) demonstrates that regardless of the chosen algorithm, a Dense loss function is capable of outperforming MSE. Specifically, except for Linear Lasso, the Dense loss with $\alpha = 0.5$ consistently outperforms MSE regardless of the model used to fit the recovery rates. This implies that if Belfius opts for MSE as suggested in the literature, it will consistently achieve poorer performance compared to incorporating weights. Consequently, the efforts invested in calibrating the best model may be less beneficial than selecting the appropriate loss function.

The same table can be made for the models based on their ranking for each loss function as a generalization of what is made in the literature with MSE:

	LR L1	Logit	RF	LGBM	XGB	EBM	NN	rec NN	GLMNet	GAMI-Net
EW	4.25	8.2	6	3.25	2.5	4.5	5.8	6.4	3.8	6
Belfius	7	6.8	3.75	4	4	3.75	5.86	7.1	6	5.5

Table 3.2: Average rank for each model over the loss functions

In this table, "LR L1" means linear regression with Lasso regularisation, "RF" random forest, "rec NN" recursive neural networks. The algorithms follow their order of description in section 3.3 except for EBM grouped with the other boosting algorithms.

Table 3.2 highlights that boosting algorithms (LGBM, XGB, and EBM) exhibit an advantage over basic models or neural networks. It appears that basic functions may not adequately capture nonlinearities in their models, while neural networks may suffer from insufficient data or require a higher number of layers. However, GLMNet performs well on an equally weighted allocation.

Boosting algorithms consistently deliver stable performance across allocations, and an explainability cost is observed for the equally weighted allocation, which disappears for Belfius' allocation. This concept of cost of explainability, as defined by Dessain et al. (2023), represents the trade-off between accuracy and interpretability when employing glassbox models instead of less interpretable ones. Overall, EBM enables accurate results while maintaining full explainability, as demonstrated in the next chapter. Other boosting algorithms can be included in comparisons.

Contrary to many papers, linear regression has been enhanced through regularization, yet this does not prevent other models from outperforming it. Particularly, this model with Belfius' allocation is notably surpassed by others. A report by the EBA (EBA (2021)) on the usage of ML models in the financial sector indicates that logistic and linear regression are frequently employed. However, utilising such simple models with MSE entails employing subpar models with a poor-performing loss function.

Using Table 3.1, the best Dense loss function is obtained with $\alpha = 0.5$ such that the Table 3.2 can be built without $\alpha = 1, 1.5$ to give:

	LR L1	Logit	RF	LGBM	XGB	EBM	NN	NN réc	GLMNet	GAMI-Net
EW	5	7	6	5	3	3	6.33	6	3.2	4

Table 3.3: Average rank for each model over the loss functions

Similarly to Table 3.2, the out-performance of the boosting algorithms remains and the GLMNet algorithm confirms its satisfying performances. However, the explainability cost disappears between EBM and XGBoost which shows that EBM is an interesting candidate when modelling the LGD/Recovery rate. This also shows that the selection of α for the Dense Loss is an important topic for which the ranking between XGBoost and EBM may vary but the overall trend is the same: basic models produce bigger valuation bias and boosting algorithms seem to outperform. The choice between XGBoost and EBM is up to the user and might be based on the explainability of the models.

3.6.1 Autocalibration

Autocalibration enhances the absolute bias in 56.1% of cases with the equally weighted (EW) allocation and 48.78% with Belfius' allocation. Additionally, aggressive results (indicating overvaluation) become more conservative (indicating undervaluation) in 15 out of 16 cases for the EW allocation and 8 out of 8 cases for Belfius' allocation. Among these aggressive results, 11 out of 16 (EW allocation) and 3 out of 8 (Belfius' allocation) achieve a smaller absolute bias.

A more detailed analysis can be conducted by examining Figures 3.7 and 3.8, where high bias results generally show improvement while low bias results tend to deteriorate. Tables A5 and A7 in the annex reveal that neural networks benefit significantly from autocalibration, whereas boosting algorithms do not. This implies that users should assess whether to apply autocalibration on a case-by-case basis for each algorithm.

Moreover, it's noteworthy that the original paper (Denuit et al. (2021)) applied this technique on a non-life insurance dataset larger than the one used in this study or in the corporate bonds LGD literature. Therefore, it's strongly recommended to independently test autocalibration due to its dependence on the dataset size. Nonetheless, the results demonstrate that autocalibration can enhance valuation bias and make aggressive results more conservative.

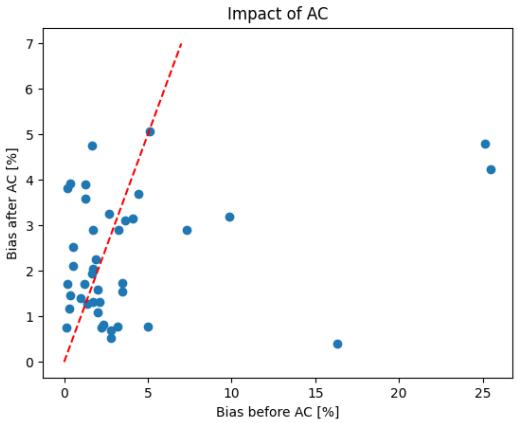

Figure 3.7: Equally weighted allocation

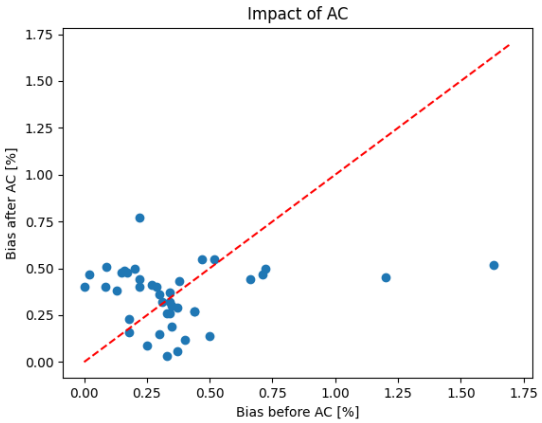

Figure 3.8: Belfius' allocation

Figure 3.9: Impact of autocalibration

3.7 Risk management tools

While risk management falls beyond the scope of this study, this section briefly illustrate how the model can be applied in credit risk. Specifically, the aim is to estimate the downturn Loss Given Default (LGD), a requisite of the European regulator. The approaches outlined here include scenario testing and a statistical approach.

3.7.1 Scenario testing

When modelling LGD, addressing economic downturn conditions is crucial but recommended by the ECB (European Central Bank (2024)). The ECB emphasises the importance of incorporating relevant macroeconomic and economic factors as drivers in this process. This study has integrated several macroeconomic and economic variables, aligning with this requirement.

Subsequently, obtaining a stressed LGD estimate involves supplying the model with stressed economic variables. For instance, GDP growth (alongside other economic variables) can be set to the worst GDP growth observed in recent history to derive a downturn LGD from the model.

This approach offers the advantage of intuitive modelling of downturn LGD, requiring minimal effort if the model is already calibrated. However, this is a data-driven perspective

that relies on past observations. Severe downturn conditions are extreme events for which observations are rare. In this context, extreme value theory may help to address this problem and generate extreme scenarios.

3.7.2 Statistical approach

For each bond, a Beta distribution has been fitted (cfr section 3.2.3). This means that a distribution is available at the bond level from which classical risk management tools can be applied. Indeed, for a given distribution, it's possible to study the distribution of the recovery rates for each entry by its mean, variance, kurtosis, Value-at-Risk (VaR), expected shortfall, tail VaR, ...

Nevertheless, some of these values are not additive (e.g. VaR) which means that it cannot be easily obtained for the whole portfolio. Two solutions are possible:

1. Using additive risk management measures such as the expected shortfall
2. Simulate the portfolio distribution through Monte Carlo simulations

This approach offers the advantage to connect to the risk management framework but at the price of a possible higher computational costs if the distribution of the portfolio is required.

Chapter 4

Interpretability

This chapter explores how to interpret model such as the one described in the previous chapter. Two ways to interpret models machine learning models exists. First, intrinsic interpretability is covered through the three glassbox models studied in the previous chapter. Glassbox models are inherently explainable due to their structure like GLMs or GAMs.

However, not all models are intrinsically interpretable which leads to the post-hoc methods. Shapley values will be covered since it's one of the most used method to explain global and local predictions. Local explanation explains a single prediction while global explanation describes the whole logic of the models from the features to the output. Finally, a brand new tool will be shown: SafeAI. Its goal is to develop statistical tests on the accuracy, the robustness and the explainability of models based on Lorenz curves.

4.1 Intrinsic Interpretability

Three glassbox models have been studied in this work. This section demonstrates how to gain insight into the inner workings of these models. Linear regression is the most basic type of explainable model where the explainability is done by looking at the coefficients of each feature. The focus is on understanding how LocalGLMNet, EBM, and GAMI-Net operate internally, providing valuable insights into their processes.

All the examples shown in this chapter are fitted with the MSE loss function but the process is the same for all loss functions and the goal of this section is to understand how to interpret the predictions of a given model.

4.1.1 LocalGLMNet

First, recall (section 3.3.4) that the model is given by:

$$\boldsymbol{x} \mapsto g(\mathbb{E}[y]) = \beta_0 + \langle \boldsymbol{\beta}(\boldsymbol{x}), \boldsymbol{x} \rangle = \beta_0 + \sum_{j=1}^{m} \beta_j(\boldsymbol{x}) x_j$$

Wüthrich and Merz (2023) proposed to look at the values of the coefficients for a given feature. In particular, the coefficients for the Debt-Equity ratio is given by:

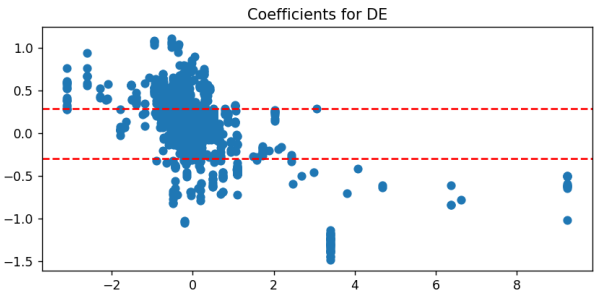

Figure 4.1: Importance: $\beta_{DE}(\boldsymbol{x})$ in function of x_{DE}

The horizontal lines represent the 90% confidence interval for noise $\mathcal{N}(0, 1)$ added to the model inputs. This necessitates the normalisation of all features, explaining why the range of values for these inputs differs from before.

In Figure 4.1, no clear trend is discernible from the coefficients. However, a drawback of this method is the presence of noisy coefficients near $x_{DE} = 0$ since the contribution $\beta_{DE}(\boldsymbol{x})\ x_{DE}$ remains constant. With more data, a decreasing trend may emerge, with left values (where x_{DE} is negative) possibly exceeding right values (where x_{DE} is positive).

Another tool has proven more effective as it eliminates noise near zero by directly showing the contribution:

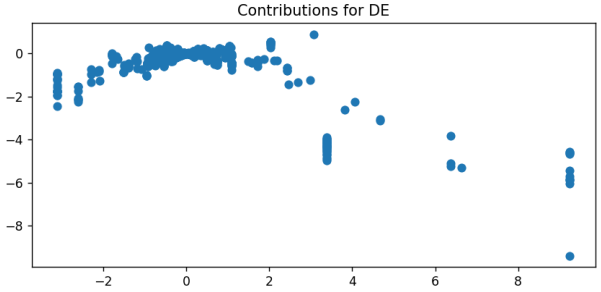

Figure 4.2: Contribution: $\beta_{DE}(\boldsymbol{x})\, x_{DE}$ in function of x_{DE}

Now, a clear trend emerges with negative contributions elsewhere than near 0, with the reference point provided by β_0. The other features exhibit similar behaviour, with coefficients that are difficult to interpret but have interpretable contributions. This does not imply that coefficients should not be computed, but rather they should be utilised

to eliminate unnecessary features when their values lie within the confidence interval. Wüthrich and Merz (2023) have further advanced this method to demonstrate feature selection. Additionally, feature selection by Lasso regularization is also addressed.

Unlike other glassbox models that only accommodate pairwise interactions, this model enables higher-order interactions through the coefficients $\boldsymbol{\beta}(\boldsymbol{x})$, which depend on the entire network structure between the input and the attention layers. An interaction is computed as the gradient of a coefficient i with respect to a variable j, denoted as $\partial\beta_i(\boldsymbol{x})/\partial x_j$. This should be interpreted as the change in the importance of variable i in relation to the value of x_j.

Figure 4.3 show the impact of 3 variables (TBill 1 month, US GDP growth 3 months and Debt-Equity) on the coefficient β_{DE}:

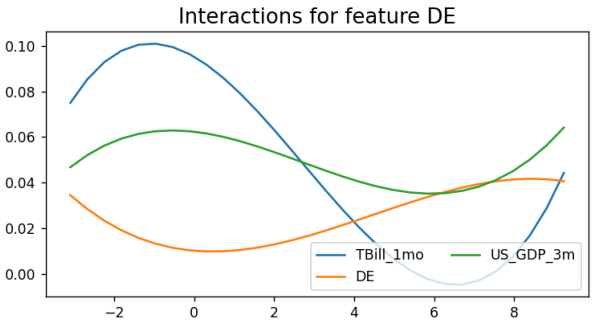

Figure 4.3: Interaction: $\partial\beta_{DE}(\boldsymbol{x})/\partial x_j$

This interpretation suggests that a high value of the Tbill rate growth over 1 month diminishes the variation of importance of the Debt to Equity variable, and vice versa. The US GDP growth has a similar impact but with a smaller magnitude, while the Debt-Equity variable exhibits an inverse relationship, with higher variations of importance at higher values. However, the shape of this interaction might depend on the architecture of the neural networks and its activation functions in particular. A future study could explore the influence of these activation functions further.

Moreover, a local explanation is as easy as the one of GLMs since for each data $\mathbf{x}_i$, the model gives a set of coefficients $\boldsymbol{\beta}(\boldsymbol{x_i})$ from which the contribution can be computed and added to obtain the prediction.

Finally, a feature importance calculation is possible by taking, for each coefficient, the average of its absolute value over the dataset. The feature importance will be given later to compare with the other two glassbox models.

4.1.2 EBM

As explained in section 3.3.4, the EBM model is given by:

$$g(\mathbb{E}[y]) = \beta_0 + \sum_j f_j(x_j) + \sum_{i,j} f_{i,j}(x_i, x_j)$$

For LGD/recovery rates regression, the identity link function has been used. This form allows to explain the main effect through their function f_j and the interactions $f_{i,j}$ with a heatmap. For example, the impact of the most important variable of the EBM model on the prediction is shown in Figure 4.4. The seniority index is a variable describing the proportion of debt more senior than a given bond. Therefore, the model uses an intuitive trend where a small seniority index impacts positively the recovery rate while a high seniority index lowers the recovery rate.

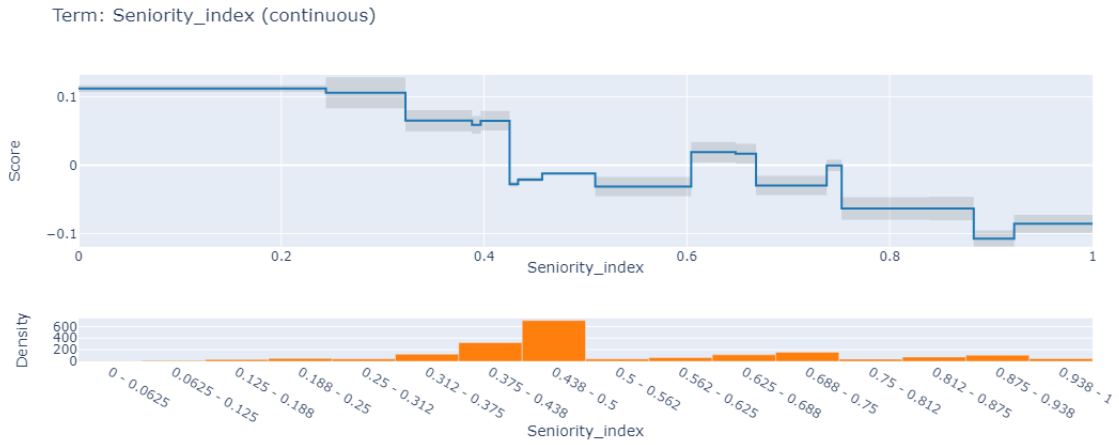

Figure 4.4: Impact of the seniority index

However, the graph shows some non monotone areas that are hard to interpret. Fortunately, EBM allows to monotonize these main effects functions thanks to isotonic regression. An isotonic regression solves the following optimization problem for a decreasing function:

$$\min_{\hat{y}} \sum_i (y_i - \hat{y}_i)^2 \quad \text{such that } \hat{y}_i \geq y_i \text{ whenever } X_i \leq X_j$$

while the inequality symbol of the constraint on $\hat{y}$ is reversed to obtain an increasing function. It's possible to let the program find the trend based on the Spearman's rank coefficient.

Constraining the function of the seniority index to be a monotone decreasing function gives:

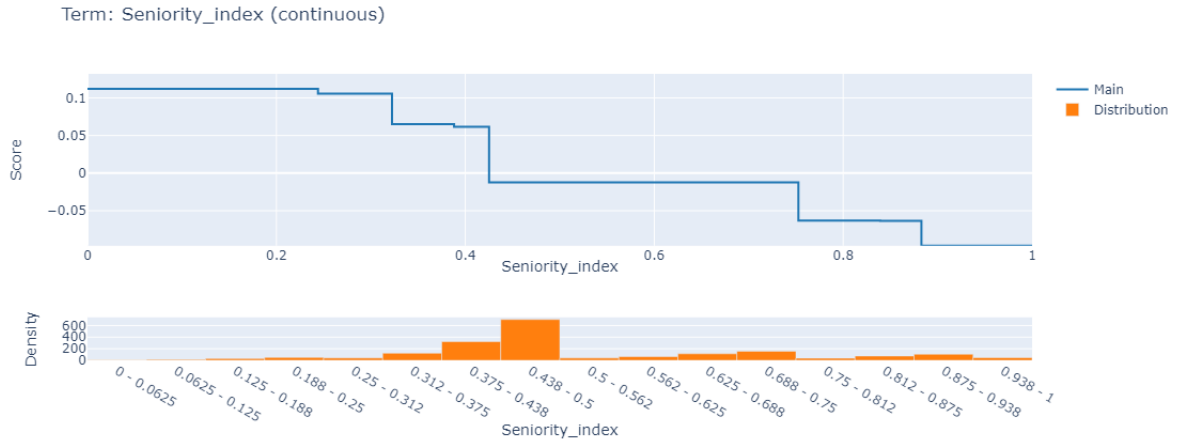

Figure 4.5: Impact of the seniority index, monotonized version

This feature becomes particularly useful when deep explainability is necessary.

On the other hand, pairwise interactions must be explained based on heatmaps such as in Figure 4.6. This figure shows that the EBM model gives a bigger positive impact of the recovery rate when the bond come from a company of the consumer industries (Beverage, Food, Healthcare, Hotel, consumer goods, ...) with a high operating margin than if the company is not from this sector.

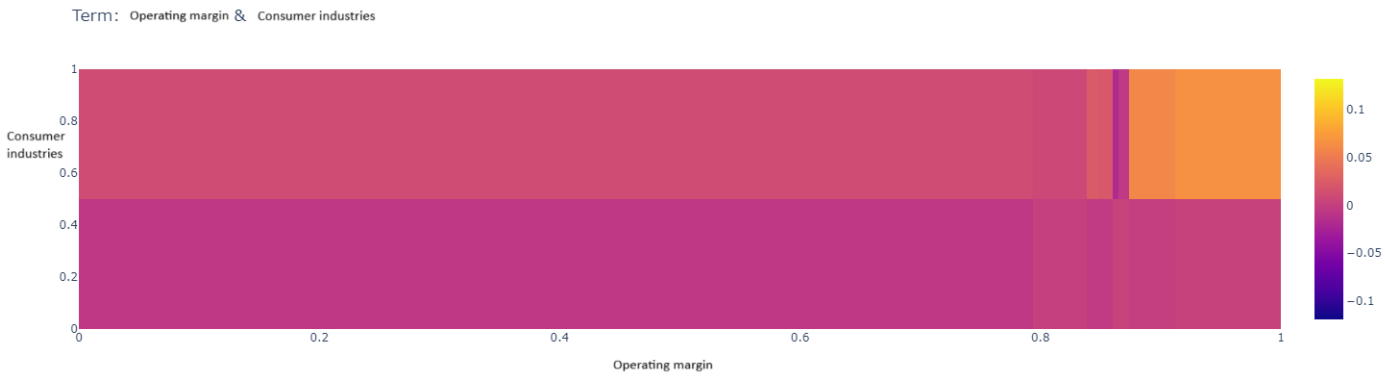

Figure 4.6: Interaction of Consumer industries with the operating margin

Finally, as demonstrated in figures 4.4 and 4.6, the main and interaction effects values provide insights for local explanations. An illustrative example is presented in Figure 4.7, where a low seniority index increases the prediction, while a high Net Debt compared to free cash flow lowers it. Such local explanations are obtained by computing the values for each f_j and $f_{i,j}$ like for GAMs.

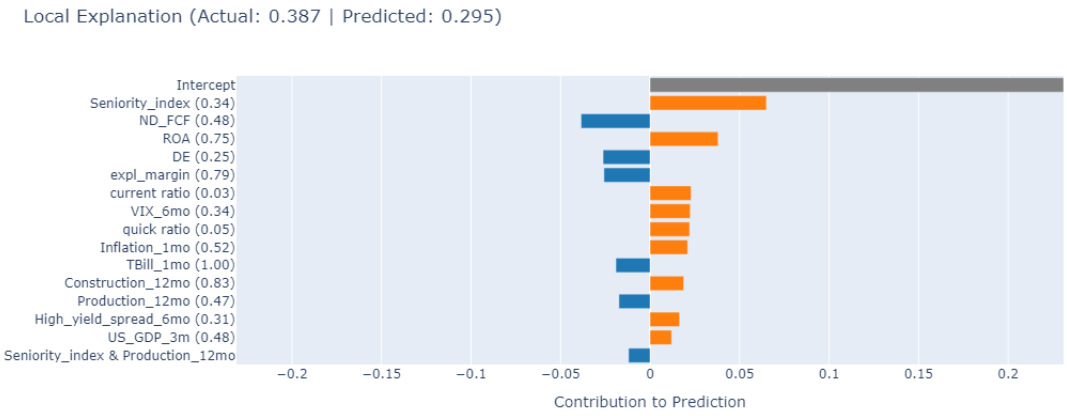

Figure 4.7: Local explanation

4.1.3 GAMI-Net

Since GAMI-Net uses the same equation as EBM, interpreting the model follows a similar approach. Therefore, the aim of this section is to primarily observe the effects of the neural networks on both the main and interaction factors.

The influence of the seniority index is depicted in Figure 4.8, while the effect of the growth of US production is illustrated in Figure 4.9. Similar to EBM, the seniority index demonstrates a noteworthy impact. Regarding the US production growth, it offers insights into the vitality of the US economy, with a discernible increase in impact on the recovery rate corresponding to higher production growth.

As Yang et al. (2021) explained, using smooth activation functions such as tanh (used here) or sigmoid makes smooth shape functions. This is different than EBM which computes the effects using boosting and trees. This difference in the internal process gives different shapes between these models. The GAMI-Net implementation does not offer the possibility to monotonize the effects but the usage of smooth activation functions gives smoother and more interpretable functions as in Figures 4.8, 4.9 and 4.10.

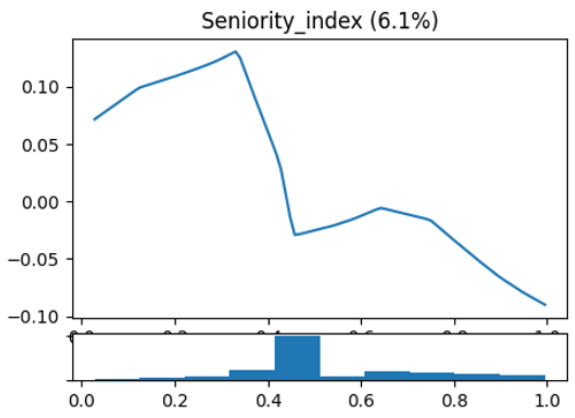

Figure 4.8: Impact of the seniority index

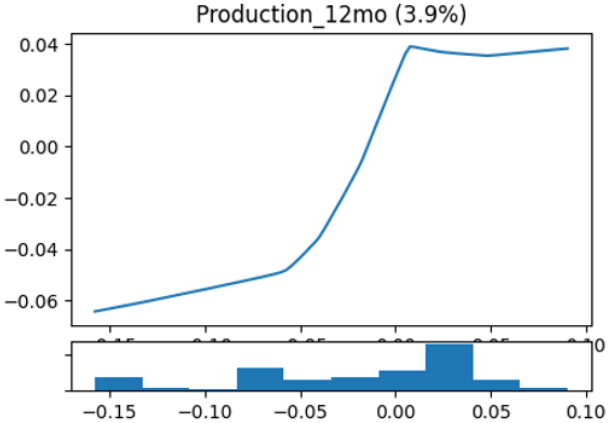

Figure 4.9: Impact of the US Production

The interaction effect is described by a heatmap like for EBM:

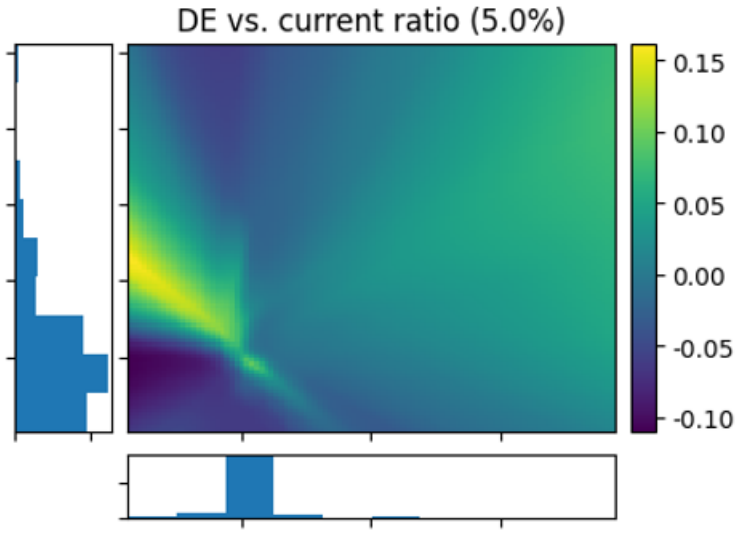

Figure 4.10: Interaction between Debt-Equity and current ratio

The values of the axis follows the classical convention with low values at the bottom left corner and high values at the top right. GAMI-Net gives more variable impact with a low current ratio. Low and high debt to equity ratios have negative impacts on the recovery rates since they correspond to negative or low equity.

The feature importance of these 3 glassbox models can be compared. For sake of clarity a subset of variables is chosen being the five most important features of EBM:

	GLMNet	EBM	GAMI-Net
Seniority index	3.13	3.96	6.1
ROA	2.58	2.1	0.8
TBill 1 month	4.988	1.95	2.1
Operating margin	4.41	1.77	2.3
ROE	3.32	1.66	Removed by sparsity constraint

Table 4.1: Feature importance [%]

GLMNet exhibits higher values since there is no additional interaction effects while GAMI-Net obtains one high value thanks to its sparsity constraint that removes some effects giving more weight to the most crucial ones. ROE is not fully removed from the GAMI-Net model since it appears through interactions. All models consider the seniority index as important, consistent with findings in the literature, but the weights for the other ones differ.

4.2 Shapley values

The Shapley values come from the game theory literature. The idea is to measure the contribution of a feature i to the prediction and its equation is given by:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[v_{S \cup \{i\}} \left(x_{S \cup \{i\}} \right) - v_S \left(x_S \right) \right] \tag{4.1}$$

for a set of feature F, a subset of features S and where:

$$v_S(x_S) = \mathbb{E}[f(\mathbf{X}|X_j : \ j \in S)] - \mathbb{E}[f(\mathbf{X})]$$

Equation 4.1 might seem complex at first glance, but it can be explained in simpler terms, originally introduced in the context of game theory by Myerson (1991) and adapted here in a machine learning perspective. Suppose that the features are lined up in a queue and added to the model one by one. With this approach, there are potentially many different ways to order the features. Specifically, there are $|F|!$ possible permutations of the features.

Now, consider a scenario where certain features, denoted as S, are placed before feature i in the queue. In this case, there are $|S|!(|F| - |S| - 1)!$ ways to arrange the features, where $|S|$ represents the number of features preceding i, and $|F| - |S| - 1$ represents the number of features following i in the set $F \setminus (S \cup \{F_i\})$. These are essentially all the possible orderings where feature i is added after the features in set S.

Given that all orderings are equally probable, the probability of feature i being added to a model containing features S is calculated as $|S|!(|F| - |S| - 1)!/|F|!$. Finally, the marginal contribution of feature i when it's added to the model can be computed as the difference before and after the addition of i as $v_{S \cup \{i\}} \left(x_{S \cup \{i\}} \right) - v_S \left(x_S \right)$.

Although this method provides precise insights into the marginal contribution of each feature, it comes with a high computational cost as it requires computing all possible subsets S, resulting in exponential complexity. However, given the relatively modest size of LGD corporate bonds datasets, this computational overhead is manageable. One approach to alleviate this issue is to calculate the marginal contribution on a subset of the dataset. In this study, the SHAP package was used, and the marginal contribution was computed across the entire dataset.

This section will show how to leverage this tool for interpreting machine learning models. It is particularly useful when the model lacks intrinsic explainability or fails to cover aspects that can be addressed with Shapley. However, for comparison, this section will use EBM (fitted with MSE) to compare to its intrinsic explanation as a reference.

4.2.1 Global interpretability

The influence of a variable can be examined through the Shapley values for each dataset entry, as shown in the scatter plot in Figure 4.12. This can be compared to the main effect

of seniority in EBM, depicted in Figure 4.4. The general trend is consistent, but there is more noise in the Shapley plot, possibly due to the fact that Shapley values also account for the seniority index through its interactions. The ICE plot in Figure 4.11 provides further insight into the impact of the seniority index. Here, the thin blue lines illustrate how the estimated recovery rate changes as the seniority index variable varies on a subset of samples (while keeping all other features constant). The interpretation aligns with the other tools: a higher seniority index corresponds to a lower expected recovery rate, and vice versa.

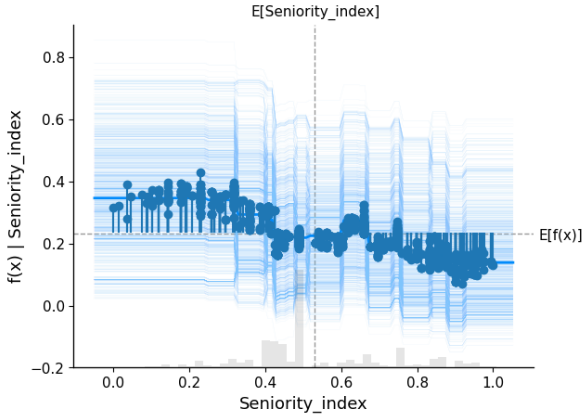

Figure 4.11: Impact of the seniority index with ICE plot

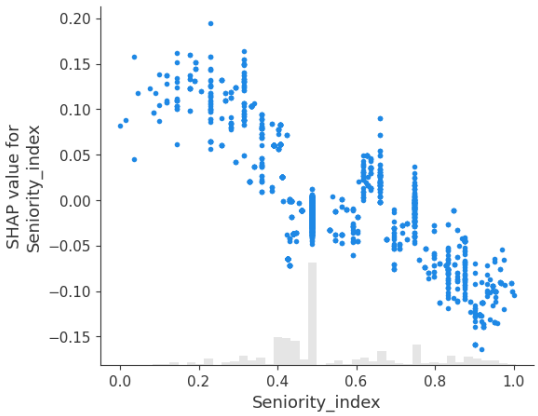

Figure 4.12: Impact of the seniority index by scatter plot

Feature importances can be computed by taking the mean absolute SHAP value, and the results will be presented in the last section of this chapter to compare the post-hoc methods.

4.2.2 Local interpretability

Given a particular bond, its shapley values can be computed for each feature which gives:

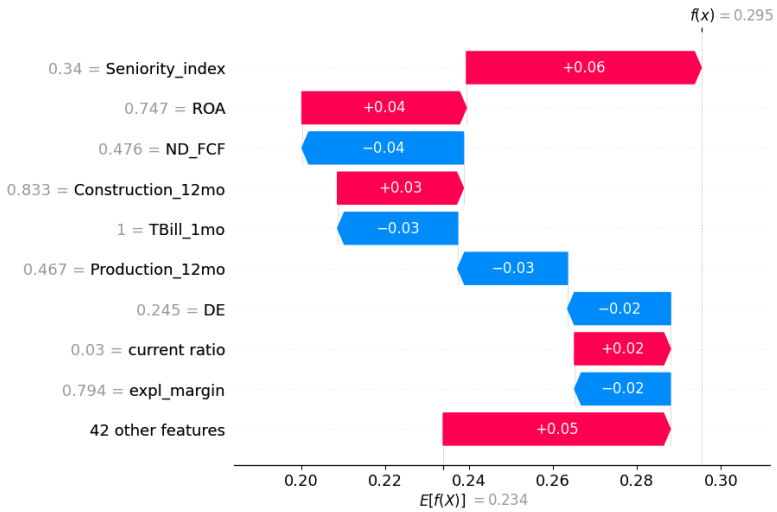

Figure 4.13: Local explanation

The explanation is nearly the same as the one given by EBM itself in Figure 4.7. The difference comes from the fact that Shapley computes the importance of a variable also through its interactions while EBM separates the main and interaction effects.

To contrast these post-hoc explanations with the intrinsic one provided by EBM, we need to analyse Figures 4.7 and 4.13. This involves aggregating their contributions to the prediction for the five most significant features according to Shapley. The comparison is made in the following table:

	Intrinsic	Shapley
Seniority index	0.065	0.0636
ROA	0.038	0.391
Net Debt/FCF	-0.0385	-0.3847
Construction growth	0.0187	0.03
Tbill growth	-0.019	-0.0286
β_0	0.2309	0.2338

Table 4.2: Local explanation comparison

Shapley values closely align with the reference values provided by EBM, with the exception of Construction growth. The disparity arises because the EBM model treats pairwise interactions separately, whereas Shapley computations aggregate the contribution of a given feature i.

4.3 AI risk measurement

Babaei et al. (2024) introduced a novel tool designed to address the requirements outlined in the EU AI act, focusing on Security, Accuracy, Fairness, and Explainability. In this study, we delve into Security, Accuracy, and Explainability aspects, as Fairness is not relevant in this context. The Security assessment evaluates the model's robustness and its resistance to perturbations. Accuracy testing involves comparing the model against a simplified version, while the Explainability component reveals the contribution of each variable and assesses their significance.

Their tool uses Lorenz and concordance curves, which are commonly employed for evaluating model performance. (Denuit et al. (2019c)). Given two statistical variables Y^* and Y^{**}, the Lorenz curve of Y^* can be constructed by sorting them in a non decreasing way such that $L_{Y^*} \equiv \left(i/n, \sum_{j=1}^{i} y^*_{r^*_j} / \left(n\bar{y}^*\right)\right)$ where r^*_j are the non-decreasing ranks of Y^* and $\bar{y}^*$ the mean. The dual Lorenz curve is built following the same idea but based on the non-increasing ranks r^*_{n+1-j}: $L'_{Y^*} \equiv \left(i/n, \sum_{j=1}^{i} y^*_{r^*_{n+1-j}} / \left(n\bar{y}^*\right)\right)$. The concordance curve C is obtained by ordering the Y^* values according to the non-decreasing ranks of Y^{**}: $C \equiv \left(i/n, \sum_{j=1}^{i} y^*_{r^{**}_j} / \left(n\bar{y}^*\right)\right)$. Therefore, comparing the concordance curve to the Lorenz curves is akin to assessing the distance of mismatch between Y^* and Y^{**} ranks. It can be illustrated with a figure taken from Babaei et al. (2024):

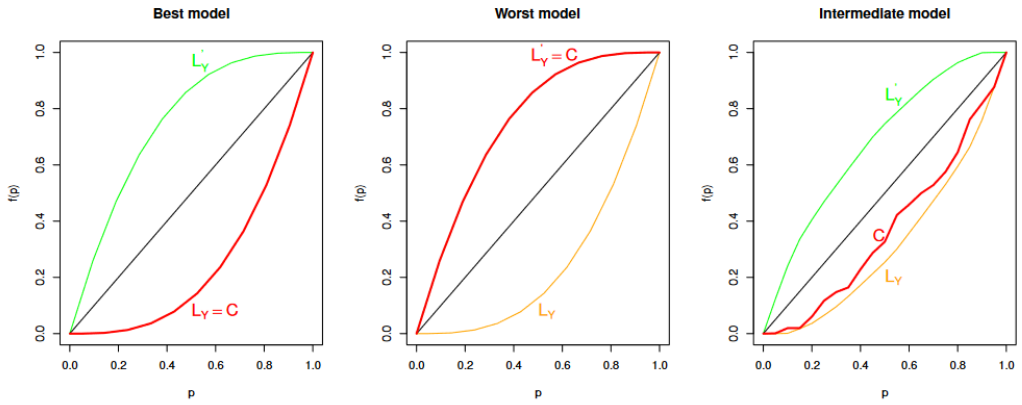

Figure 2: The L_Y and L'_Y Lorenz curves, the concordance curve C and the 45-degree line in the cases of: the best possible model; the worst possible model; an "intermediate" model

Figure 4.14: Comparing the concordance and Lorenz curves

The best model is reached when $r_j^* = r_j^{**}$ for all j so that the concordance curve and the Lorenz curve are superposed. However, if $r_{n+1-j}^* = r_j^{**}$ for all j, the concordance curve coincides with the dual Lorenz curve. A natural measure is to define how far are the Lorenz and concordance curves. The whole theory behind this tool is based on the definition of a statistical test based on a distance between these curves and on the definition of Y^* and Y^{**}. This method allows to quantity the improvement of the model against a random predictor based on the distance to the 45° line.

4.3.1 Accuracy

The accuracy test is computed using $Y^* = Y$ the target and $Y^{**} = \hat{Y}$ the estimation. Defining the accuracy measure RGA as a ratio between the concordance curve and the dual Lorenz curve (normalized by the maximum distance) gives:

$$RGA = \frac{\sum_{i=1}^{n} \left\{ \frac{1}{n\overline{y}} \left(\sum_{j=1}^{i} y_{r_{n+1-j}} - \sum_{j=1}^{i} y_{\hat{r}_j} \right) \right\}}{\sum_{i=1}^{n} \left\{ \frac{1}{n\overline{y}} \left(\sum_{j=1}^{i} y_{r_{n+1-j}} - \sum_{j=1}^{i} y_{r_j} \right) \right\}}$$

where r_j (resp. r_{n+1-j}) are the non decreasing (resp. non increasing) ranks of Y, $\hat{r}_j$ the non decreasing ranks of $\hat{Y}$ and $\overline{y}$ the mean value of Y.

This measures have the following properties:

1. $RGA \in [0, 1]$ with 0, 0.5 and 1 indicating a perfectly discordant, random and perfectly concordant model respectively.

2. RGA is the same as AUROC for binary Y (proof in Giudici and Raffinetti (2024))

3. It is possible to derive a statistical test to check if two models are significantly different (see annex E on how to obtain a statistical test from RGA).

The RGA value for this EBM model is 0.644 with the given curves:

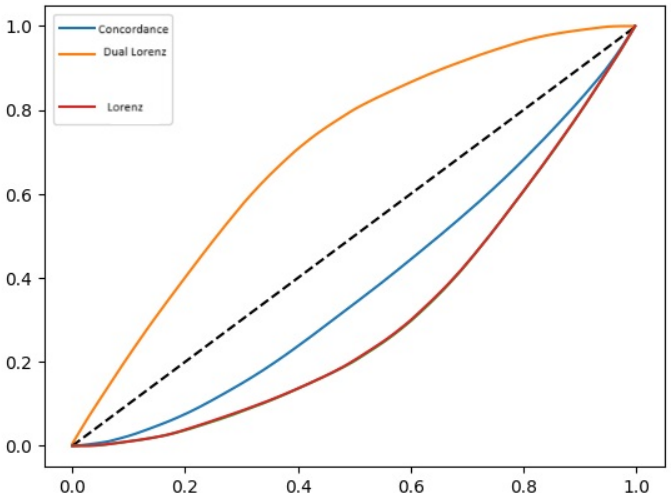

Figure 4.15: Lorenz and concordance curves for accuracy

This shows that the model makes some error as previously observed but is closer to the Lorenz curve than its dual. However, even if the distance to the diagonal line is observable, the test allows to confirm that EBM is significantly different than the random model with a 0 p-value. Testing this model against the reduced one without Seniority index gives also a p-value under 5% which means that not using this variable leads to significantly different predictions.

4.3.2 Robustness

The Robustness statistics is obtained using $Y^* = \hat{Y}$ and $Y^{**} = \hat{Y}^P$ with $\hat{Y}^P$ the prediction over a perturbed input dataset. The perturbation suggested by the authors is to swap the values below the α quantile and above $1-\alpha$ for a given α. Defining RGR in the same spirit as RGA gives:

$$RGR = \frac{\sum_{i=1}^{n} \left\{ \frac{1}{n\hat{y}} \left(\sum_{j=1}^{i} \hat{y}_{r_{n+1-j}} - \sum_{j=1}^{i} \hat{y}_{r_j^p)} \right) \right\}}{\sum_{i=1}^{n} \left\{ \frac{1}{n\hat{y}} \left(\sum_{j=1}^{i} \hat{y}_{r_{n+1-j}} - \sum_{j=1}^{i} \hat{y}_{r_j} \right) \right\}}$$

Moreover, it shares similar properties:

1. $RGR \in [0,1]$ with 0 and 1 indicating a full perturbed and full robust model respectively. RGR=0.5 means that the perturbations have turned the model into a random model.

2. It is possible to derive a statistical test to test the resilience of two different models (the details are given in Babaei et al. (2024) but are very similar to the one of RGA developed in annex E).

EBM demonstrates remarkable resilience to perturbations, especially on its most crucial variable. For instance, a perturbation of 0.1 (equivalent to 20% of input data being swapped) results in an RGR of 0.8 and applying a 0.1 perturbation across all variables yields an RGR of 0.66. A statistical test indicates a p-value of 0 when comparing EBM's resilience against Linear Regression, one of the most commonly used models. This suggests that EBM exhibits significantly greater resilience, as evidenced by its higher RGR.

This capability addresses one of the European regulator's requirements for a stable and robust model over time. By providing a quantifiable measure of resilience, it enables comparison of resilience levels among models, utilising the same theoretical basis as RGA statistics.

4.3.3 Explainability

In this toolbox, the explainability feature is determined using $Y^* = \hat{Y}$ and $Y^{**} = \hat{Y}^{-X_k}$, where X_k represents the removal of variable k from the input dataset. It's important to note that when the concordance curve aligns with the Lorenz curve, it indicates maximum unexplainability of variable k because removing the feature doesn't alter the prediction. Conversely, maximum explainability is achieved when the concordance curve matches the dual Lorenz curve. This reversed approach affects the definition of the RGE measure::

$$RGE = 1 - \frac{\sum_{i=1}^{n} \left\{ \frac{1}{n\bar{\hat{y}}} \left(\sum_{j=1}^{i} \hat{y}_{r_{n+1-j}} - \sum_{j=1}^{i} \hat{y}_{r_j^{-x_k}} \right) \right\}}{\sum_{i=1}^{n} \left\{ \frac{1}{n\bar{\hat{y}}} \left(\sum_{j=1}^{i} \hat{y}_{r_{n+1-j}} - \sum_{j=1}^{i} \hat{y}_{r_j} \right) \right\}}$$

This approach shares similarities with Shapley in that it evaluates the explainability of a variable by comparing the model's performance with and without a particular feature. However, this measures comes with classical properties of this toolbox:

1. $RGE \in [0, 1]$ with 0 indicating that the k-th feature does not contribute to the model explanation while $RGE = 1$ means that this feature gives the maximum explanation. RGE=0.5 means that the models have turned the model into a random without this feature.

2. It is possible to derive a statistical test to test if the given feature provides a significant explanation (the details are given in Babaei et al. (2024) but are very similar to the one of RGA developed in annex E)

This statistical test provides a p-value of 0 for the Seniority index feature meaning that Seniority index carries a signification explanation in the model.

Based on this RGE measure, feature importances can be computed and compared to Shapley as done in Table 4.3. Again, the five most important features according to the intrinsic interpretability of EBM are compared:

	EBM	Shapley	SafeAI
Seniority index	3.96	4.72	7.94
ROA	2.1	2.077	3.21
TBill 1 month	1.95	1.89	1.98
Operating margin	1.77	1.99	1.625
ROE	1.66	1.67	1.6215

Table 4.3: Feature importance [%]

SafeAI shows higher values for the most important features which can be explained by the fact that it compares models based on ranking and not on values while Shapley does compare based the values. Shapley exhibits closer numbers but the order is not respected while SafeAI finds the correct order. EBM assigns a lower importance to the Seniority index compared to the post-hoc methods. This difference arises because Shapley and SafeAI capture the contribution of the Seniority index through interactions, whereas EBM separates the main and interactions effects.

From an implementation point of view, this toolbox has its own Python package created by the authors of the papers. Nevertheless, this has some flaws such as the computation speed due to some inefficiencies in the code. For this work, the package has been improved in two ways:

1. A graphical visualization has been added to plot the (dual) Lorenz and concordance curves like in Figure 4.15.

2. Speed and space optimizations have been implemented to obtain a speed-up from 20 to 300 depending on the functions. Vectorization is an essential element when implementing scientific code in Python. This allows to run a part of the code in a C/Fortran compiled version instead of pure Python. Furthermore, numerical libraries like NumPy provides very efficient codes with clever algorithms that avoid as much as possible cache misses.

Conclusion and Perspectives

In the introduction, 5 goals were mentioned. The first goal was to show adequate preprocessing techniques which has been addressed by the first chapter. Imputation techniques have been compared, a new outlier technique taking into account skewness and excess kurtosis has been applied and the dataset has been completed with marcro economic variables.

Another goal was to model LGD using powerful models, including glassbox models, leading to the subsequent goal of proposing loss functions beyond the commonly used MSE. Chapter two addressed this by covering basic, tree-based, neural network, and glassbox models. Autocalibration was explored as a technique for bias removal. Additionally, two additional loss functions were introduced to extend the LGD literature: the Dense Loss, which rebalances a given loss function, and the Beta deviance, which models recovery rates with a Beta distribution, providing risk management tools. Furthermore, this chapter introduced a new economic metric that demonstrated useful properties for comparing models and loss functions.

Finally, interpretability was addressed in the last chapter. The first approach to interpreting a model is to use its intrinsic interpretability if applicable (e.g., for glassbox models). Conversely, post-hoc methods complement these tools and allow for the interpretation of complex models such as FNNs or GBMs. The SafeAI toolbox introduced new tools to statistically assess the accuracy, robustness, and explainability of given models.

Main results

- Input features exhibit skewness and excess kurtosis, making the interquartile range outlier detection technique inadequate. The STAR outlier detection technique effectively handles such behavior. Additionally, the Random Forest imputer appears to be a promising imputation method for such datasets, as comparisons have shown.
- Choosing the right loss function is as important as choosing the right model. The choice of the loss function has not been covered in the LGD literature. Modelling LGD using Linear Regression and MSE will lead to high valuation bias in practice.
- MSE can be improved using weights, similar to techniques used in life insurance. In this study, the Dense Loss function outperforms the MSE on both equally weighted and Belfius' allocations. The hyperparameter of this loss function can be optimized through cross-validation. Additionally, the Beta deviance approach models the

distribution of the recovery rate for each bond, providing valuable tools for risk management.

- Boosting algorithms have demonstrated superior performance compared to other models. Specifically, XGBoost was the best model for the equally weighted allocation, while EBM outperformed for the Belfius' allocation. Overall, EBM appears to be a promising candidate for accurately modeling LGD/recovery rates, offering the added benefit of intrinsic interpretability.
- Glassbox models provide precise local and global explanations. Similarly, post-hoc techniques like Shapley values can also accurately explain predictions both locally and globally for a given model.
- The European regulator requires the predictions to be interpretable and estimates robust. The SafeAI toolbox can be used to assess a model's robustness, accuracy, and explainability, ensuring compliance with these regulatory demands.

Improving paths

The work can be pushed further in the following directions:

- Integrating an estimated probability of being in downturn conditions based on the input features computed with a specific model is a valuable addition to LGD modelling:

  $LGD = \mathbb{E}[LGD|\text{Downturn}]\ P[\text{Downturn}] + \mathbb{E}[LGD|\text{not Downturn}]\ (1 - P[\text{Downturn}])$

  This approach allows for a more nuanced understanding of LGD dynamics under different economic scenarios. However, it's important to consider how this addition may impact the interpretability of the model.

  One potential approach to mitigate the interpretability challenge is to provide transparency around how the estimated probability of downturn conditions is calculated and its influence on the final LGD estimate. This could involve visualizations or explanations that illustrate how changes in input features affect the likelihood of downturn conditions and subsequent LGD estimates.

- Incorporating a time series analysis of yearly average LGD adds another dimension to LGD modelling, allowing for insights into trends and patterns over time. Time series models such as ARIMA, LSTM, or CNN can be employed to forecast the yearly average LGD based on historical data. Once the time series model is trained and validated, the forecasted yearly average LGD can be used as an additional feature in the prediction for each bond. By leveraging this average expected LGD, the model can account for broader trends and fluctuations in LGD estimates over time.

  However, it's important to note that implementing this approach effectively requires a sufficiently large dataset to train the time series model robustly.

- Incorporating a variable to differentiate between regulatory and accounting frameworks over time may enhance the LGD modelling process by capturing the impact of evolving accounting rules on asset and liability valuation. This variable would serve as an indicator of changes in the regulatory and accounting environment, allowing the model to adjust its predictions accordingly.
- Study the PD-LGD dependency for example with copulas since the dependency is not simply a correlation (cfr Lauria (2019)).
- Experiment with additional allocation strategies to compare the performance of models and loss functions across a wider range of allocation scenarios.
- Create an algorithm to smooth or monotonize the pairwise interactions of EBM. This would enhance the interpretability of the model by reducing abrupt changes, making it easier to understand. Additionally, it would align with the requirement for increased human-model interactions outlined by the regulator.
- Include additional loss functions in the comparison to assess whether there are superior alternatives to Dense Loss and if the performance varies depending on the allocation. Additional algorithms can also be tested such as boosting with neural networks (Hainaut et al. (2022)).

Bibliography

Golnoosh Babaei, Paolo Giudici, and Emanuela Raffinetti. Safeaipackage: A python package for ai risk measurement. *SSRN Electronic Journal*, 01 2024. doi: 10.2139/ssrn.4744576.

Matteo Barbagli and Frédéric Vrins. Accounting for pd-lgd dependency: A tractable extension to the basel asrf framework. *Economic Modelling*, 125:106321, 2023.

João A. Bastos. Forecasting bank loans loss-given-default. *Journal of Banking Finance*, 34(10):2510–2517, 2010.

Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 07 1999. doi: 10.1023/A:1007515423169.

Anthony Bellotti, Damiano Brigo, Paolo Gambetti, and Frédéric Vrins. Forecasting recovery rates on non-performing loans with machine learning. *International Journal of Forecasting*, 37(1):428–444, 2021.

Arup Bose and Snigdhansu Chatterjee. *U-statistics, Mm-estimators and Resampling*. Springer, 2018.

Paula Branco, Luís Torgo, and Rita Ribeiro. Smogn: a pre-processing approach for imbalanced regression. 09 2017.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

S. F. Buck. A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society. Series B (Methodological)*, 22(2):302–306, 1960.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, August 2016. doi: 10.1145/2939672.2939785.

Suvra Jyoti Choudhury and Nikhil R Pal. Imputation of missing data with neural networks for classification. *Knowledge-Based Systems*, 182:104838, 2019.

Michel Denuit, Donatien Hainaut, and Julien Trufin. *Effective Statistical Learning Methods for Actuaries I: GLMs and Extensions*. 01 2019a. ISBN 978-3-030-25819-1. doi: 10.1007/978-3-030-25820-7.

Michel Denuit, Donatien Hainaut, and Julien Trufin. *Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions*. 01 2019b. ISBN 978-3-030-25826-9. doi: 10.1007/978-3-030-25827-6.

Michel Denuit, Dominik Sznajder, and Julien Trufin. Model selection based on lorenz and concentration curves, gini indices and convex order. *Insurance: Mathematics and Economics*, 89:128–139, 2019c. ISSN 0167-6687. doi: https://doi.org/10.1016/j.insmatheco.2019.09.001.

Michel Denuit, Donatien Hainaut, and Julien Trufin. *Effective Statistical Learning Methods for Actuaries II: Tree-Based Methods and Extensions*. 01 2020. ISBN 978-3-030-57555-7. doi: 10.1007/978-3-030-57556-4.

Michel Denuit, Arthur Charpentier, and Julien Trufin. Autocalibration and tweedie-dominance for insurance pricing with machine learning, 2021.

Jean Dessain, Nora Bentaleb, and Fabien Vinas. Cost of explainability in ai: An example with credit scoring models. pages 498–516, 10 2023. ISBN 978-3-031-44063-2. doi: 10.1007/978-3-031-44064-9_26.

EBA. Eba discussion paper on machine learning for irb models. Technical report, EBA, 2021.

EBA. Machine learning for irb models: Follow-up report from the consultation on the discussion paper on machine learning for irb models. Technical report, EBA, 2023.

European Central Bank. Ecb guide to internal models. Technical report, ECB, 2024.

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

Paolo Gambetti, Francesco Roccazzella, and Frédéric Vrins. Meta-learning approaches for recovery rate prediction. *Risks*, 10(6), 2022a.

Paolo Gambetti, Francesco Roccazzella, and Frédéric Vrins. Meta-learning approaches for recovery rate prediction. *Risks*, 10(6), 2022b. ISSN 2227-9091. URL `https://www.mdpi.com/2227-9091/10/6/124`.

Gert Loterman. *Predicting loss given default*. PhD thesis, Ghent University, 2013.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

Paolo Giudici and Emanuela Raffinetti. Rga: a unified measure of predictive accuracy. *Advances in Data Analysis and Classification*, 01 2024. doi: 10.1007/s11634-023-00574-2.

John Gregg and Jason Moore. Star_outliers: a python package that separates univariate outliers from non-normal distributions. *BioData Mining*, 16, 09 2023.

Marc Gürtler and Martin Hibbeln. Improvements in loss given default forecasts for bank loans. *Journal of Banking Finance*, 37(7):2354–2366, 2013. ISSN 0378-4266. doi: https://doi.org/10.1016/j.jbankfin.2013.01.031.

Donatien Hainaut, Julien Trufin, and Michel Denuit. Response versus gradient boosting trees, glms and neural networks under tweedie loss and log-link. *Scandinavian Actuarial Journal*, 2022(10):841–866, 2022.

J. A. Hartigan and P. M. Hartigan. The Dip Test of Unimodality. *The Annals of Statistics*, 13(1):70 – 84, 1985. doi: 10.1214/aos/1176346577. URL `https://doi.org/10.1214/aos/1176346577`.

Trevor J Hastie. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

Wassily Hoeffding. A Class of Statistics with Asymptotically Normal Distribution. *The Annals of Mathematical Statistics*, 19(3):293 – 325, 1948. doi: 10.1214/aoms/1177730196.

Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5375–5384, 2016.

Charlotte Jamotton, Donatien Hainaut, and Thomas Hames. Insurance analytics with clustering techniques. Technical report, Institut de Statistique, Biostatistique et Sciences Actuarielles, UCLouvain, 2023.

Florian Kaposty, Johannes Kriebel, and Matthias Löderbusch. Predicting loss given default in leasing: A closer look at models and variable selection. *International Journal of Forecasting*, 36, 11 2019. doi: 10.1016/j.ijforecast.2019.05.009.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*, 2017.

Fabian Krüger and Johanna F. Ziegel. Generic conditions for forecast dominance, 2019.

Alessandro Lauria. Measuring probability of default and loss given default in a real dataset, 2019.

Xiang Li, Nandan Sudarsanam, and Daniel D Frey. Regularities in data from factorial experiments. *Complexity*, 11(5):32–45, 2006.

Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. pages 623–631, 08 2013. doi: 10.1145/2487575.2487579.

Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991. ISBN 9780674341166. URL `http://www.jstor.org/stable/j.ctvjsf522`.

Abdolreza Nazemi and Frank J. Fabozzi. Macroeconomic variable selection for creditor recovery rates. *Journal of Banking Finance*, 89:14–25, 2018. ISSN 0378-4266.

Luke M. Olson, Min Qi, Xiaofei Zhang, and Xinlei Zhao. Machine learning loss given default for corporate debt. *Journal of Empirical Finance*, 64:144–159, 2021. ISSN 0927-5398.

Min Qi and Xinlei Zhao. Comparison of modeling methods for loss given default. *Journal of Banking Finance*, 35(11):2842–2855, 2011.

Emanuela Raffinetti. A rank graduation accuracy measure to mitigate artificial intelligence risks. *Quality & Quantity*, 57(Suppl 2):131–150, 2023.

Bank Settlements and Basel Supervision. Basel ii: International convergence of capital measurement and capital standards: A revised framework. *Bank for International Settlements, Basel*, 01 2005.

Nadeem A. Siddiqi and Meiqing Zhang. A general methodology for modeling loss given default. *The RMA Journal*, pages 92–95, 2004.

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

Michael Smithson and Jay Verkuilen. A better lemon squeezer? maximum-likelihood regression with beta-distributed dependent variables. *Psychological methods*, 11:54–71, 03 2006. doi: 10.1037/1082-989X.11.1.54.

Michel Steininger, Konstantin Kobs, Padraig Davidson, Anna Krause, and Andreas Hotho. Density-based weighting for imbalanced regression. *Machine Learning*, 110:1–25, 08 2021.

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays . *Bioinformatics*, 17(6):520–525, 06 2001.

John W Tukey. Modern techniques in data analysis. In *Proceedings of the NSF-Sponsored Regional Research Conference*, volume 7. Southern Massachusetts University North Dartmouth, MA, USA, 1977.

Vincenzo Verardi and Catherine Vermandele. Univariate and multivariate outlier identification for skewed or heavy-tailed distributions. *The Stata Journal: Promoting communications on statistics and Stata*, 18:517–532, 09 2018. doi: 10.1177/1536867X1801800303.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. *Advances in neural information processing systems*, 30, 2017.

Aymeric Warnauts. Embedding layer and localglmnet feature selection in actuarial setting, 2023.

Simon N. Wood. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):95–114, 2003. doi: https://doi.org/10.1111/1467-9868.00374.

Mario Wüthrich and Michael Merz. *Statistical Foundations of Actuarial Learning and its Applications*. 01 2023. ISBN 978-3-031-12408-2. doi: 10.1007/978-3-031-12409-9.

Zebin Yang, Aijun Zhang, and Agus Sudjianto. Gami-net: An explainable neural network based on generalized additive models with structured interactions. *Pattern Recognition*, 120:108192, 2021.

Guangyou Zhou, Yijia Zhang, and Sumei Luo. P2p network lending, loss given default and credit risks. *Sustainability*, 10(4):1010, 2018.

Appendix

A Kmeans and Burt's distance

The Burt's distance is based on a matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ where the boolean d_{ij} indicates if the sample i has the modality j. A categorical variable c can have m_c modalities and m is the total number of modalities with q the number of categorical variables such that $m = \sum_{c=1}^{q} m_c$. The Burt's matrix is defined as:

$$\mathbf{B} = \mathbf{D}^T\mathbf{D}$$

whose elements can been interpreted as a joint modalities frequency. And the weighted Burt's matrix $\mathbf{B}^W$ is given by:

$$\begin{aligned} \mathbf{C} &= diag(\mathbf{B})^{-1/2} \\ \mathbf{B}^W &= \frac{1}{q}\mathbf{CBC} \end{aligned}$$

This definition leads to the Burt's distance based on the weighted Burt's matrix:

$$d(i,j) = \left\| \mathbf{D}_{i,\cdot}\mathbf{B}^W/q - \mathbf{D}_{j,\cdot}\mathbf{B}^W/q \right\|_2$$

From this distance, a clustering algorithms such as KMeans can be applied and will take into account the frequency of modalities in the dataset through the Burt's distance. This method is covered in more details in Jamotton et al. (2023).

B Financial Ratios

The distribution of the financials ratios is shown in Figure A16 but note that certain extreme values are omitted from the graphs for clarity.

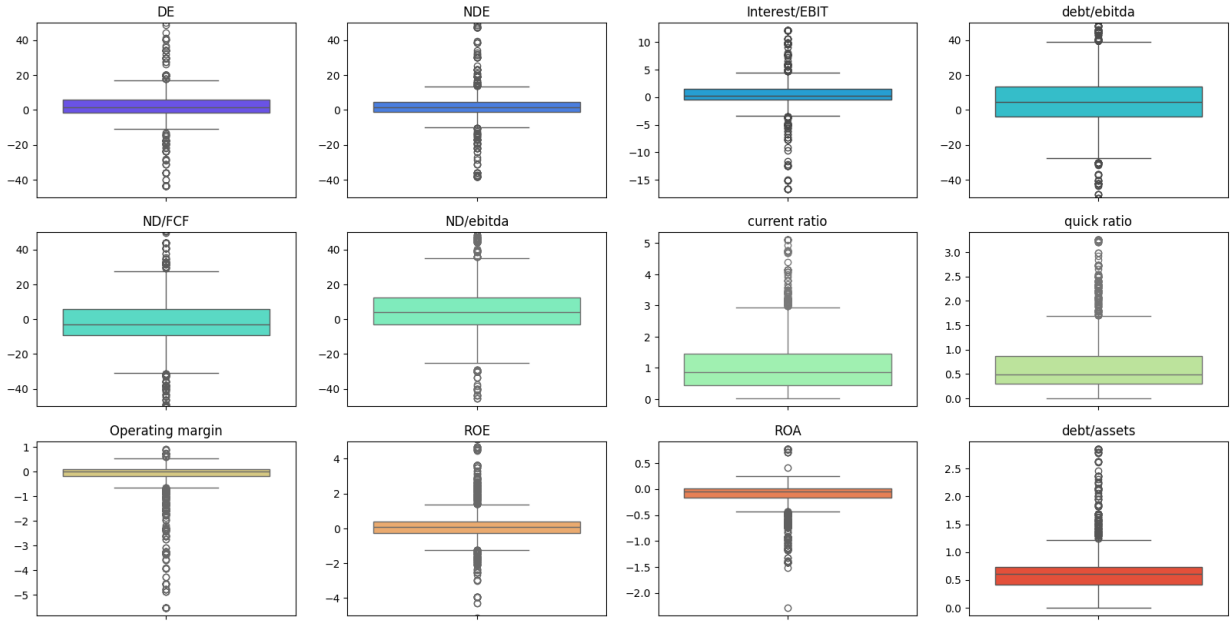

Figure A16: Boxplots des ratios financiers

The plots indicate that leverage ratios exhibit large values with heavy tails, signifying high kurtosis. These elevated values suggest a significant amount of debt and/or a low level of equity, with negative values indicating negative equity. A diminished equity also impacts the Return on Equity (ROE), which can assume large values. In comparison, Return on Assets (ROA) reveals fewer large values, and a substantial portion of them corresponds to a negative return on assets.

The Operating Margin displays numerous negative values, indicating an activity that fails to generate cash independently of investing cash flows. Regarding liquidity ratios, there is a decline from the current to the quick ratio, implying that certain assets are held as inventories. This may be observed in companies facing challenges in selling their products.

The prevalence of extreme values, both positive and negative, signifies companies grappling with challenges such as markedly low profitability (EBITDA, EBIT, Free Cashflow, net result) and/or a balance sheet featuring low equity and high debt. Consequently, it is logical to identify such patterns in these financial ratios, contributing to non-normality marked by heavy tails.

C Autocalibration

In order to determine the appropriate neighborhood size (α), the training set is divided into three subsets: a training set (60%), a smoothing set (20%), and a validation set (20%), following the methodology of Denuit et al. (2021). The training set is utilized to fit the model, the smoothing set is employed for autocalibration, and the validation set is used to select the optimal neighborhood size while considering bias. Bias is defined as the disparity between the mean of the real data in the validation set and the predicted values. This bias is illustrated in Figure A17. The first plot demonstrates that the autocalibrated neural network exhibits lower bias with autocalibration (NN_AC) than without (NN). The second plot depicts the impact on the loss (in this case, the Mean Squared Error, MSE), indicating that the loss can be minimized with auto-calibration.

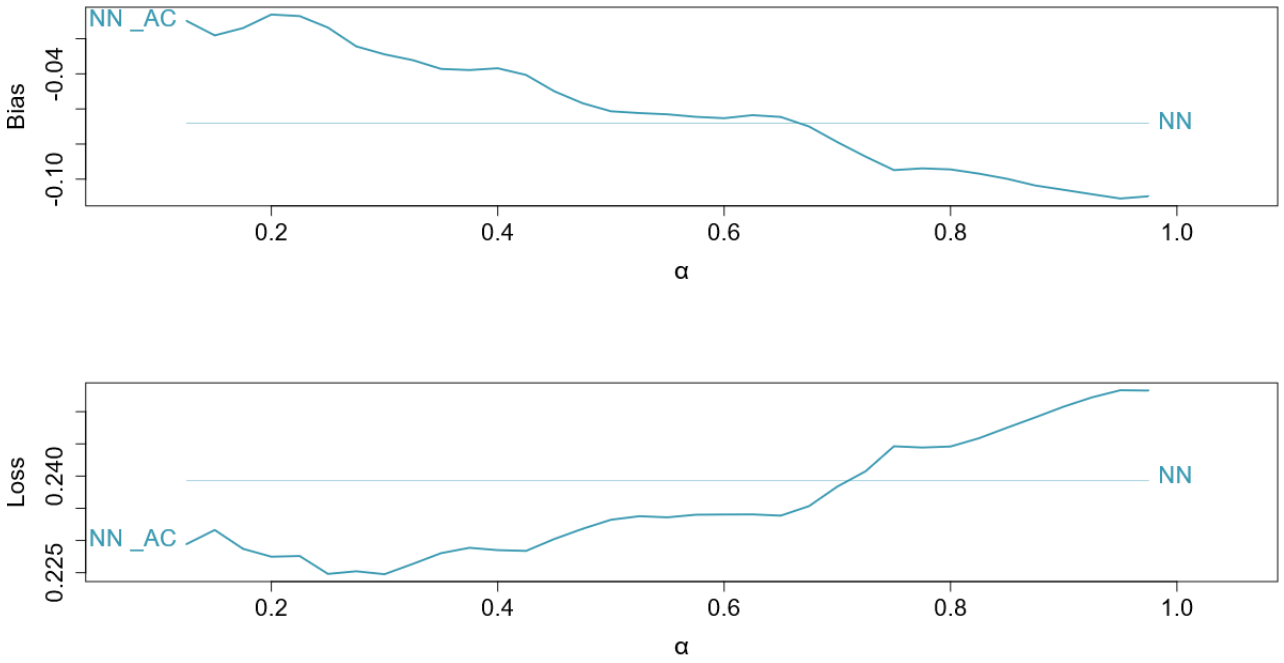

Figure A17: Size of neighborhood selection

After selecting the neighborhood size, the difference between models with and without autocalibration can be observed in Figure A18. The first plot is a QQ plot comparing the predictions of the neural network with and without autocalibration. The second plot illustrates the function $s \mapsto \mathbb{E}[Y|\pi(X) = s]$, where Y represents the target, X the features, and π the predictor (in this case, a neural network). This figure indicates that, according to the autocalibration technique, the neural network tends to underestimate the recovery rate, except around a prediction of 0.35.

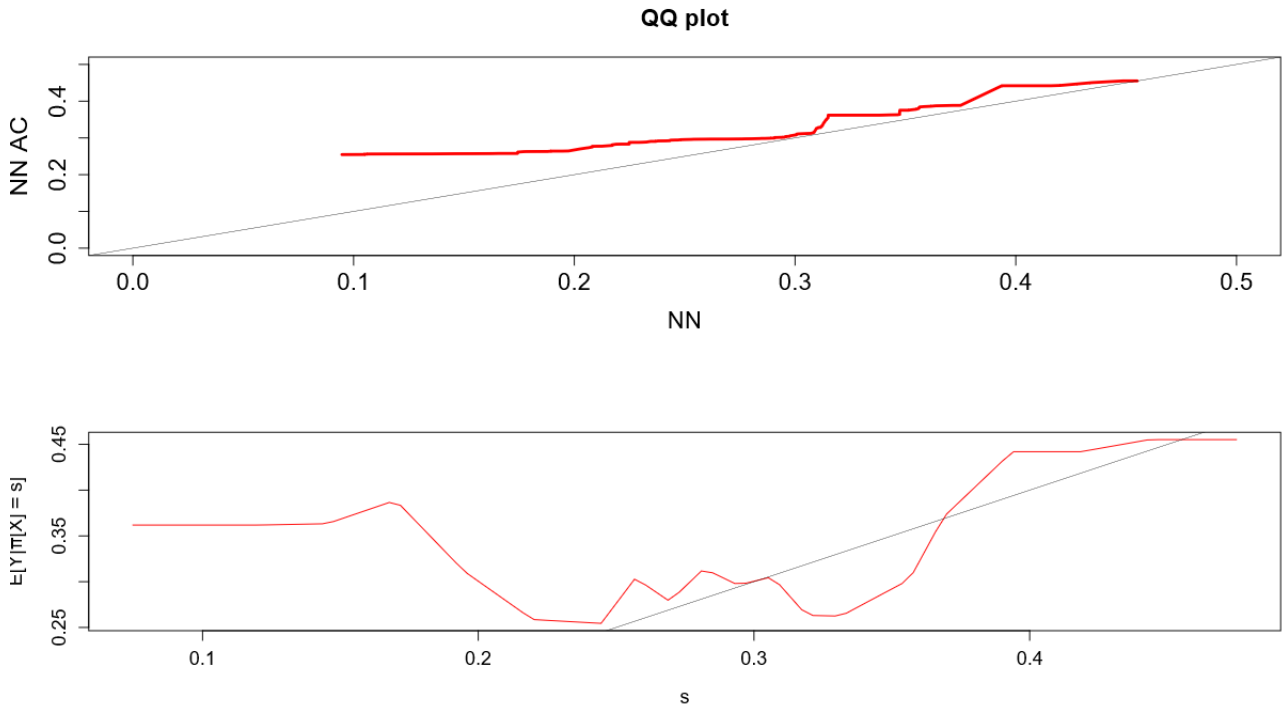

Figure A18: From neural network to autocalibrated neural network

D Full results

Economic results are presented for equally-weighted portfolios in Tables A4 and A5, and for a Belfius allocation in Tables A6 and A7. Each cell denotes the economic metric for an algorithm (as indicated by the row) fitted by a loss function (as indicated by the column). The most accurate result for each algorithm is highlighted in bold. Additionally, each cell has another value in parentheses that gives the economic metric in absolute value without any offset between bonds in the portfolio. It can be seen as a measure of individual PD-weighted precision. As explained in its dedicated section, the economic metric is a valuation bias.

Models	MSE	Dense Loss $\alpha = 0.5$	Dense Loss $\alpha = 1$	Dense Loss $\alpha = 1.5$	Beta deviance
Linear Lasso	**-0.2%** (16%)	-5.13% (**14%**)	0.29% (15.84%)	3.63% (16.6%)	✘
Logit	16.29% (21.29%)	9.87% (18.13%)	25.12% (26.5%)	25.5% (26.74%)	**-0.15%** (**15.36%**)
Random Forest	-4.46% (**14%**)	**-1.25%** (14.89%)	1.27% (15.14%)	4.1% (16.06%)	✘
LGBM	-1.92% (13.88%)	-1.67% (14.09%)	0.39% (14.64%)	**0.17%** (**13.37%**)	✘
XGBoost	-1.7% (14.22%)	-0.96% (**13.13%**)	**0.39%** (13.52%)	2.66% (13.55%)	✘
EBM	-1.67% (15.11%)	-1.41% (15.67%)	**1.19%** (**14.65%**)	5.02% (14.86%)	✘
NN	**-2.02%** (**15.78%**)	-2.25% (16.33%)	-3.2% (16.08%)	-3.27% (16.23%)	-4.31% (16.52%)
Recursive NN	-2.77% (16.69%)	**-2.1%** (16.35%)	-3.5% (**15.97%**)	-3.47% (16.4%)	-4.06% (16.81%)
LocalGLMNet	-2.02% (**14.64%**)	**-0.54%** (16.38%)	-1.75% (16.11%)	-2.32% (16.1%)	1.06% (15.96%)
GAMI-Net	**-0.51%** (**13.44%**)	-1.71% (17.32%)	2.82% (14.68%)	7.33% (16.85%)	✘

Table A4: Economic results on equally-weighted portfolio

Red crosses indicate that it is not possible to fit the model for the given loss function. In Table A4, models marked with red crosses are unable to output two values, or in the case of linear regression, the output space lacks coherence with a Beta distribution. A potential solution could be to apply the sigmoid function to the result, which corresponds to Logit regression.

Regarding autocalibration, in the following table, the generated values on the smoothing set may not sufficiently represent the input space, thereby hindering the effectiveness of the autocalibration technique. In this scenario, the mean in a neighborhood may not adequately cover a different value in the validation set.

Models	MSE	Dense Loss $\alpha = 0.5$	Dense Loss $\alpha = 1$	Dense Loss $\alpha = 1.5$	Beta deviance
Linear Lasso	-1.72% (**16.5%**)	-5.08% (16.65%)	**1.18%** (17%)	-3.12% (18.88%)	✘
Logit	**0.41%** (17.39%)	3.2% (16.14%)	4.8% (19.24%)	-4.23% (17.96%)	0.76% (**15.4%**)
Random Forest	-3.7% (**13.33%**)	-3.6% (14.23%)	-3.9% (14.78%)	**-3.16%** (13.74%)	✘
LGBM	**-2.26%** (13.81%)	-4.75% (14.35%)	-3.92% (15.18%)	-3.83% (**13.24%**)	✘
XGBoost	-2.05% (14.31%)	**-1.41%** (13.14%)	-1.46% (13.27%)	-3.25% (**12.41%**)	✘
EBM	-1.94% (15%)	-1.27% (14.95%)	-1.72% (14.52%)	**-0.77%** (**14.03%**)	✘
NN	1.59% (**15.18%**)	**0.75%** (16.96%)	0.78% (17.9%)	2.9% (17.8%)	✘
Recursive NN	**0.7%** (**15.93%**)	1.33% (17.43%)	1.73% (17.32%)	1.56% (18.49%)	✘
LocalGLMNet	1.1% (16.6%)	2.52% (16.47%)	1.32% (17%)	**-0.83%** (**16.19%**)	✘
GAMI-Net	-2.12% (15.6%)	-2.9% (**13.77%**)	**-0.53%** (13.95%)	2.9% (15.01%)	✘

Table A5: Economic results on equally-weighted portfolio with autocalibration

Models	MSE	Dense Loss 3 $\alpha = 0.5$	Dense Loss 3 $\alpha = 1$	Dense Loss 3 $\alpha = 1.5$	Beta deviance
Linear Lasso	-0.52% (**0.97%**)	-0.66% (0.86%)	-0.22% (0.99%)	**0.2%** (1.2%)	✘
Logit	0.3% (1.12%)	**-0.15%** (1.02%)	1.2% (1.6%)	1.63% (1.93%)	-0.71% (**0.99%**)
Random Forest	-0.34% (**1.08%**)	-0.22% (1.11%)	**0.003%** (1.19%)	0.31% (1.35%)	✘
LGBM	-0.38% (**0.95%**)	-0.22% (1%)	-0.27% (0.99%)	**-0.086%** (1.1%)	✘
XGBoost	-0.47% (**0.89%**)	-0.18% (1.12%)	-0.17% (1.06%)	**0.02%** (1.2%)	✘
EBM	-0.18% (1.4%)	-0.44% (1.3%)	-0.16% (**1.2%**)	**0.13%** (1.33%)	✘
NN	-0.34% (0.93%)	**-0.33%** (**0.83%**)	-0.37% (1.02%)	-0.35% (0.87%)	-0.41% (1.06%)
Recursive NN	-0.4% (0.99%)	**-0.33%** (**0.83%**)	-0.37% (0.96%)	-0.34% (0.87%)	-0.55% (0.93%)
LocalGLMNet	-0.5% (0.92%)	-0.25% (**0.09%**)	-0.3% (0.96%)	-0.35% (1.03%)	**0.17%** (1.01%)
GAMI-Net	-0.44% (**0.97%**)	**-0.09%** (1.4%)	-0.29% (1.21%)	-0.72% (1.19%)	✘

Table A6: Economic results on Belfius's allocation

Models	MSE	Dense Loss $\alpha = 0.5$	Dense Loss $\alpha = 1$	Dense Loss $\alpha = 1.5$	Beta deviance
Linear Lasso	-0.55% (1%)	**-0.44%** (**1%**)	-0.77% (1.14%)	-0.5% (1.04%)	✘
Logit	**-0.36%** (1.01%)	-0.48% (1.07%)	-0.45% (1.08%)	-0.52% (1%)	-0.47% (**0.97%**)
Random Forest	-0.37% (**1.03%**)	-0.44% (1.05%)	-0.4% (1.06%)	**-0.32%** (1.16%)	✘
LGBM	-0.43% (**0.93%**)	**-0.4%** (0.98%)	-0.41% (1%)	-0.4% (1.03%)	✘
XGBoost	-0.55% (**0.88%**)	**-0.23%** (1.11%)	-0.48% (0.98%)	-0.47% (1.11%)	✘
EBM	**-0.16%** (1.33%)	-0.27% (1.3%)	-0.49% (**1.07%**)	-0.38% (1.12%)	✘
NN	0.26% (1.38%)	**0.03%** (1.18%)	-0.06% (**1.13%**)	0.3% (1.25%)	✘
Recursive NN	**0.12%** (**1.24%**)	0.26% (1.27%)	0.29% (1.44%)	0.32% (1.24%)	✘
LocalGLMNet	0.14% (1.21%)	**0.09%** (1.12%)	0.15% (1.23%)	-0.19% (**1.05%**)	✘
GAMI-Net	**-0.27%** (1.03%)	-0.51% (**0.99%**)	-0.4% (1.04%)	-0.5% (1.08%)	✘

Table A7: Economic results on Belfius's allocation with autocalibration

E RGA statistical test

This development is based on the one done by the authors of the method in Babaei et al. (2024) and completed by some developments for a better understanding.

Raffinetti (2023) proved that the RGA measures can be written with covariances:

$$RGA = \frac{\sum_{i=1}^{n}\left\{\frac{1}{n\bar{y}}\left(\sum_{j=1}^{i} y_{r_{n+1-j}} - \sum_{j=1}^{i} y_{\hat{r}_j}\right)\right\}}{\sum_{i=1}^{n}\left\{\frac{1}{n\bar{y}}\left(\sum_{j=1}^{i} y_{r_{n+1-j}} - \sum_{j=1}^{i} y_{r_j}\right)\right\}} = \frac{1}{2}\frac{cov(Y_{r(\hat{Y})}, F(Y))}{cov(Y, F(Y)} + \frac{1}{2}$$

with $Y_{r(\hat{Y})}$ the reordered Y following the ranks of the corresponding predictions $\hat{Y}$ and F the CDF of Y.

Briefly, the left expression is rewritten using Lorenz curves proprieties and the covariances appear based on its definition $Cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$.

Therefore, RGA is computed as a linear function of:

$$\psi(Y, \hat{Y}) = \frac{cov(Y_{r(\hat{Y})}, F(Y))}{cov(Y, F(Y))}$$

being a statistics that can be used to compare two given models (Mod_1, Mod_2):

$$H_0 : \psi(Y, \hat{Y}_{Mod_1}) = \psi(Y, \hat{Y}_{Mod_2}) \quad vs \quad H_1 : \psi(Y, \hat{Y}_{Mod_1}) \neq \psi(Y, \hat{Y}_{Mod_2})$$

A statistical test can be introduced using two U-statistics as an estimator of ψ:

$$\hat{\psi}(Y, \hat{Y}_{Mod_1}) = \frac{U_1}{U_3} = \frac{\frac{1}{4\binom{n}{2}}\sum_{i=1}^{n}(2i - 1 - n)Y_{r(\hat{Y}_{i\,Mod_1})}}{\frac{1}{4\binom{n}{2}}\sum_{i=1}^{n}(2i - 1 - n)Y_{r(Y_i)}}$$

Following the same idea, for the second model: $\hat{\psi}(Y, \hat{Y}_{Mod_2}) = \frac{U_2}{U_3}$

Here is a development that proves this result since it does not appear in the papers of the authors:

$$cov(Y, F(Y)) = \binom{n}{2}^{-1} \sum_{1 \leq i_i \leq i_2 \leq n} \frac{1}{2}(F(Y_{i1}) - F(Y_{i2}))\ (Y_{i1} - Y_{i2}) \tag{4.2}$$

$$= \frac{1}{2}\binom{n}{2}^{-1} \sum_{1 \leq i_i \leq i_2 \leq n} \left(\frac{i_1}{n} - \frac{i_2}{n}\right)\ (Y_{i1} - Y_{i2}) \tag{4.3}$$

$$= \frac{1}{2}\binom{n}{2}^{-1} \sum_{i=1}^{n} Y_i \left[\left(\sum_{j=i}^{n} \frac{i}{n} - \frac{j}{n}\right) - \left(\sum_{j=1}^{i-1} \frac{j}{n} - \frac{i}{n}\right)\right] \tag{4.4}$$

$$= \frac{1}{2}\binom{n}{2}^{-1} \sum_{i=1}^{n} Y_i \left(\sum_{j=1}^{n} \frac{i}{n} - \frac{j}{n}\right) \tag{4.5}$$

$$= \frac{1}{2}\binom{n}{2}^{-1} \sum_{i=1}^{n} Y_i \frac{2i - n - 1}{2} \tag{4.6}$$

The first equation 4.2 uses the definition of a U-estimator of order 2 with kernel

$$h((x_1, y_1), (x_2, y_2)) = \frac{1}{2}(x_1 - x_2)(y_1 - y_2)$$

The goal of U-statistics is to build unbiased estimators but since it's out of the scope of this work, interested readers will find more detailed explanations on the basics of U-statistics in Bose and Chatterjee (2018).

The equation 4.3 uses the Empirical CDF of the sample $Y_{i=1,\dots,n}$. Equation 4.4 simplifies the above equation with the first term in brackets being the coefficient of Y_i when it takes the role of Y_{i1} and the second when it takes the role of Y_{i2}.

Then, define a function $\hat{\delta}$ that depends on 3 dependent U-statistics:

$$\hat{\delta} = \psi(Y, \hat{Y}_{Mod_1}) - \psi(Y, \hat{Y}_{Mod_2}) = \frac{U_1}{U_3} - \frac{U_2}{U_3}$$

Hoeffding (1948) proved that a function of dependent U-statistics converges to a normal distribution which means that the test statistics for testing the null hypothesis H_0 is given by:

$$Z = \frac{\hat{\delta}}{\sqrt{\widehat{\mathrm{Var}(\hat{\delta})}}} \to N(0, 1)$$

If $|Z| \geq z_{\alpha/2}$, the null hypothesis is rejected and the models Mod_1 and Mod_2 are significantly different.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
Faculté des sciences
Place des sciences, 2 bte L6.06.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/sc**