Boosting the Prediction of Extreme Values

Aníbal Silva
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2022

Orientador
Prof. Dra. Rita Ribeiro, Faculdade de Ciências

Coorientador
Prof. Dr. Nuno Moniz, Faculdade de Ciências

U. PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas
pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/________

Declaração de Honra

Eu, Aníbal José Ferreira e Silva, inscrito(a) no Mestrado em Ciências dos Computadores da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação/relatório de estágio/projeto [indicar o aplicável] reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação/relatório de estágio/projeto “Boosting the Prediction of Extreme Values”, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação/relatório de estágio/projeto [indicar o aplicável] quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor.

Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Aníbal José Ferreira e Silva

Ermesinde, 13 de outubro de 2022

Universidade do Porto

Masters Thesis

Boosting the Predictions of Extreme Values

Author:

Aníbal Silva

Supervisor:

Rita Ribeiro

Co-supervisor:

Nuno Moniz

A thesis submitted in fulfilment of the requirements

for the degree of MSc. Computer Science

at the

Faculdade de Ciências da Universidade do Porto

Departamento de Ciências de Computadores

December 12, 2022

“ *When convention and science offer us no answers, might we not finally turn to the fantastic as a plausibility?* ”

David Duchovny as *Fox Mulder*

Acknowledgements

I would like to start by giving a word of appreciation to my supervisors, Rita Ribeiro and Nuno Moniz, for providing me the opportunity of developing this work, for helping me improve my critical thinking and mature my scientific writing.

Secondly, I would like to show my gratitude towards my parents for always believing and supporting my decisions.

Finally, I would like to thank my friends, not only for supporting me when the “waters are shaking”, but also for just being there.

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciências de Computadores

MSc. Computer Science

Boosting the Predictions of Extreme Values

by Aníbal Silva

Imbalanced domain learning aims to produce accurate models in predicting instances that, though underrepresented, are of utmost importance for the domain. Research in this field has been mainly focused on classification tasks. Comparatively, the number of studies carried out in the context of regression tasks is negligible. One of the main reasons for this is the lack of loss functions capable of focusing on minimizing the errors of extreme (rare) values. Recently, an evaluation metric was introduced: Squared Error Relevance Area ($SERA$). This metric posits a bigger emphasis on the errors committed at extreme values while also accounting for the performance in the overall target variable domain, thus preventing severe bias. However, its effectiveness as a loss function is unknown. The main goal of this thesis is to study the impacts of using $SERA$ as an optimization criterion in imbalanced regression tasks. Using gradient boosting algorithms as proof of concept, an experimental study with 36 data sets of different domains and sizes is performed. Results show that models that used $SERA$ as an objective function are practically better than the models produced by their respective standard boosting algorithms at the prediction of extreme values. This confirms that $SERA$ can be embedded as a loss function into optimization-based learning algorithms for imbalanced regression scenarios. After this proof of concept, a study on the impact of using $SERA$ in the residuals of Gradient Boosting Machines will be performed.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto

Departamento de Ciências de Computadores

Mestrado em Ciências dos Computadores

Boosting Predictions of Extreme Values

por Aníbal Silva

Domínios desbalanceados visam construir modelos precisos em prever instâncias que, embora sub-representadas, são o maior foco no domínio. A investigação neste campo tem sido, contudo, maioritariamente focada em tarefas de classificação. Comparativamente, o número de estudos no contexto de problemas de regressão é praticamente insignificante. Um dos motivos é a falta de funções de custo capazes de minimizar erros de valores extremos (raros). Recentemente, uma métrica foi introduzida: Squared Error Relevance Area ($SERA$). Esta métrica não só enfatisa erros cometidos em valores extremos, como também tem em consideração todo o domínio da variável predictiva. Contudo, a sua eficácia como função de custo é desconhecida. Nesta tese, o objectivo é estudar os impactos do uso de $SERA$ como uma função de custo em problemas de regressão desbalanceada. Usando algoritmos de gradient boosting como prova de conceito, um estudo experimental com 36 conjuntos de dados de diferentes domínios e tamanhos será feito. Os resultados mostram que os modelos que usam $SERA$ como função de custo são praticamente melhores do que modelos optimizados por uma função de custo padrão na previsão de valores extremos. Isto confirma que $SERA$ pode ser embutida como uma função de custo num modelo de optimização para cenários de regressão desbalanceada. Depois de desta prova de conceito, um estudo sobre o impacto do $SERA$ nos resíduos usados nos modelos de gradient boosting será efectuado.

Contents

List of Figures

List of Tables

List of Algorithms

Symbols

$\mathcal{D}$	Training data set
x	Independent features
y	Dependent feature
$\hat{y}$	Predicted feature
$\mathcal{X}$	Independent feature space
$\mathcal{Y}$	Dependent feature space
X	Independent random variables
Y	Dependent random variable
f	Optimal learner
β	Slope coefficients
$\boldsymbol{\Theta}$	Set of parameters
$\mathcal{L}$	Loss function
P	Probability density function
$\mathbb{E}$	Expected value
$\mathcal{N}$	Gaussian probability density function
$\mathbf{I}$	Identity matrix
ϕ	Relevance function
σ	Instance contribution to $SERA$ or weights
F	Strong learner
h	Weak learner
r	Pseudo-residuals

$\boldsymbol{a}$	Weak learner learnable parameters
γ	Estimated slope coefficients
η	Learning rate
T	Number of intervals taken to discretize $SERA$
$\mathcal{W}$	Workflow
$\mathcal{M}$	Model
R	ROPE

Glossary

$SERA$	Squared Error Relevance Area
SER_t	Squared Error-Relevance
MSE	Mean Squared Error
SSE	Sum of Squared Errors
SE	Squared Error
EPE	Expected Prediction Error
AE	Absolute Error
MAE	Mean Absolute Error
$LINEX$	Linear Exponential
$LIN - LIN$	Linear Linear
$QUAD - QUAD$	Quadratic Quadratic
NN	Neural Network
FDS	Feature Distribution Smoothing
LDS	Label Distribution Smoothing
k-**NN**	k-Nearest Neighbors
pchip	Piece Cubic Hermite Interpolating Polynomials
XGBoost	Extreme Gradient Boost
LGBM	Light Gradient Boosting Machines
Smote	Synthetic Minority Over-sampling Technique
SmoteR	Synthetic Minority Over-sampling Technique for Regression
SmoGN	Synthetic Minority over Gaussian Noise

DIR	Deep Imbalanced Regression
KDE	Kernel Density Estimation
MC	Medcouple
IQR	Inter quartile range
$O()$	Worst-case time complexity
ROPE	Region Of Practical Equivalence

Chapter 1

Introduction

Supervised machine learning assumes the existence of an unknown function mapping a set of independent to dependent variables. The goal of supervised learning algorithms is to approximate such an unknown function through optimization processes. To achieve this, a key decision rests on choosing which preference criterion, e.g. a loss function, should be used. Such a decision entails critical definitions and assumptions on what should be considered a successful approximation. By assuming that all values are equally important, traditional optimization processes tend to focus on the most common observations, i.e. those near the average or median. Such focus on error reduction in the most common values of a domain traditionally come at the expense of neglecting a proper function approximation for values that are considered extreme, known for entailing high-importance events in some application domains.

Imbalanced regression appears in this context, where it is paramount to predict such rare and extreme target values accurately.

Examples of this type of predictive task include multiple real-world applications in different areas, such as predictive maintenance, where the user is interested in estimating the lifetime of a given product [1], environmental sciences, where the goal is to predict hazardous concentrations of a given molecule in the air [2], age estimation [3] where the age of a person is inferred from its visual aspects and web content ranking [4], where the goal is to predict the popularity of a given content in social media applications.

Several challenges impose the non-triviality of predicting extreme values.

From a supervised learning perspective, these include two main ones: 1) the definition of suitable and non-uniform preferences over a continuous and possibly infinite domain of the target variable; 2) map such preference regarding the extreme values into an evaluation metric that would adequately allow model selection and, possibly, optimization. Regarding the first challenge, a proposal [5, 6] exists that suggests a mapping of the target variable domain into a well-defined space (the relevance space), which gives information about the relevance of a given instance based on its target value. As for the second challenge, while there are some proposals for specially tailored evaluation metrics [7–9] in an imbalanced regression scenario, very few works exist on including such metrics in the optimization process.

1.1 Motivation

Typically, in standard regression tasks, the used loss functions are symmetric w.r.t. to a set of observations and thus favor models that are accurate in the prediction of values around the center of the distribution of the target variable. In an imbalanced regression scenario, the most common approaches are based on the definition of an asymmetric loss function. One example includes [10], where the authors focused on the prediction of extreme values by defining a branched asymmetric loss function in the residual space, using the Gradient Boosting algorithm as a training model. Here, the loss function branches into a quadratic (for a loss close to zero) or exponential (for higher losses, i.e., higher residuals) function depending on a threshold defined in the residual space.

In the same context of imbalanced regression, an evaluation metric was proposed in [6] - Squared Error Relevance Area ($SERA$). By using the definition of relevance associated with the target variable domain, this metric explicitly gives the notion of asymmetry regarding the loss in different ranges of the target variable. $SERA$ allows for errors of equal magnitude to have different impacts depending on the relevance of the target values. Moreover, while it focuses on errors in cases with extreme target values, it also accounts for the errors committed across all the rest of the target values, preventing a severe bias towards the extreme values. In addition, it does not depend on any hyper-parameter, which is common in proposed asymmetric loss functions found in the literature.

1.2 Objectives

The main objectives of this thesis are:

1. Provide a brief review of recent research developed in the field of Imbalanced Regression;

2. Review $SERA$ from a numerical perspective;
3. Provide the main tools to embed $SERA$ as an optimization loss function in machine learning algorithms – with a focus on Gradient Boosting Machines;
4. Perform an extensive experimental study to show $SERA$ ability to optimize extreme and normal values;
5. Perform a study on the impact of $SERA$ in the residuals of Gradient Boosting Machines.

1.3 Thesis organization

This thesis is organized as follows. In Chapter 2, a review of recent research in the field of Imbalance Regression will be presented. In Chapter 3, the problem definition will be initially introduced. Next, the relevance space, where $SERA$ is built upon, will be introduced. Here, this space will be revisited and the non-parametric function that maps it onto a continuous domain will be introduced. Finally, $SERA$ will be introduced and an equivalence with the Sum of Squared Errors (SSE) will be demonstrated. In Chapter 4 the models that will be used as a baseline to assess the quality of $SERA$ as a loss function in a variety of data sets will be introduced, such as the necessary modifications to conform $SERA$ with them. Next, a grid-search procedure over the chosen models will be performed under a variety of data sets. Given the best parameters, a hypothesis test will used to evaluate the statistical significance between models optimized with $SERA$ and MSE. Finally, model quality will be assessed on an out-of-sample set. In Chapter 5, a discussion on how $SERA$ influences the residuals computed in Gradient Boosting Machines will be performed. In Chapter 6, conclusions will be drawn and future work will be provided.

1.4 Bibliographic Notes

The main work presented in this thesis has been accepted and will be soon published:

- Silva A., Ribeiro R., Moniz N., Model Optimization in Imbalanced Regression. In Proc. of the 25th International Conference on Discovery Science. Springer

An **R** package was also developed where all the tools presented throughout this thesis are used: `https://github.com/anibalsilva1/ModelOptimizationIR`

Chapter 2

Related Work in Imbalanced Learning

The study of imbalanced learning has been advocated over the years, as it poses well-known challenges to standard predictive learning tasks [11]. One of the major differences between standard and imbalanced predictive learning tasks is that, while the first does not have a preference over the predictive focus of the target variable, which, by consequence, lies around the mean, the latter aims at predicting values that are uncommonly seen, i.e., values that considerably deviate from normality. As such, new methods emerged in order to circumvent this averaged predictive focus.

There are three main strategies to cope with imbalanced domain learning problems: data-level, algorithmic-level, and hybrid. A short review of such strategies is provided in this chapter (Section 2.1, Section 2.2 and Section 2.3, respectively).

2.1 Data-level Approaches

Data-level approaches are the most common and oldest ones. Their main advantage is that they can be embedded in any learning task as a pre-processing step, without the need to worry about further modifications to the learning algorithms. Generally speaking, we can group them into under-sampling, over-sampling, generation of synthetic examples, or their combination.

Under-Sampling methods aim at standardizing the target variable distribution which we want to predict by randomly removing samples that, for example, have a high probability density. In a regression task, one way to accomplish this is by mapping our target variable into the relevance space introduced in [5], and defining a threshold t that separates extreme from non-extreme values, randomly removing the latter instances. Conversely, over-sampling standardize a target variable distribution by adding samples that have a low probability density. By the same token, one way to apply over-sampling can be done by sampling with replacement instances that have a relevance above a given threshold in the relevance space. How to define the amount of under/over-sampling in these methods is another natural question imposed by the methods. Usually, a user-specified ratio is provided.

Regarding synthetic methods, one that stood its ground in data-level approaches for Imbalanced Domains is the Synthetic Minority Over-sampling Technique (Smote) [12]. Generally speaking, Smote over-samples the minority class by creating synthetic examples in a two-classification problem setting. This synthesis is done by evaluating the k-nearest neighbors (k-NN) for a given minority sample (one at a time). Given these neighbors, the algorithm generates new random samples conditioned to a line segment that connects a given neighbor with the minority sample, i.e., by linear interpolation. This algorithm was later on adapted to learning algorithms and provided good results, however, when determining the nearest neighbors it is not assured that an instance that represents a minority class is picked, and thus it has the potential risk of introducing a class mixture to the data distribution.

Although initially proposed in the context of classification, some extensions to regression problems were developed. The first one was SmoteR [13]. SmoteR shares the same backbone as Smote, i.e., it uses k-NN to find the nearest neighbors and synthesizes a new sample by linear interpolation. In addition, the following strategies are also adopted: *i)* The definition of an extreme value lies in the relevance space, where a user-predefined threshold must be provided to split normal and extreme instances; *ii)* Since in a regression task we may have a different range of values that may be of interest, the algorithm also has the ability to create samples corresponding to low and high extreme values; *iii)* Performs under-sampling on normal instances; *iv)* It deals with both numeric and nominal data, where the latter is randomly picked between a given rare case or its neighbor; *v)* Since the target variable of the generated sample may not be the same, it is given by a weighted averaged mean between an extreme and its neighbor, where the weights are given by the inverse distance function between the extreme and neighbor to the newly generated samples, respectively. Using the same reasoning as for Smote, nearest neighbors which have a low relevance (i.e., are common values) might be picked for interpolation.

To circumvent this mixture problem, a more refined method was proposed. This method was called Synthetic Minority Over Gaussian Noise (Smogn). With the spirit of SmoteR, Smogn also uses the notion of the relevance space to separate normal and extreme instances and performs under-sampling. The main differences between these two methods follow: *i)* Instead of splitting instances that correspond to low and high extreme values, this algorithm orders the data set by ascending the value of the target variable. Given this ordered data set, consecutive partitions (bins) are created, given the relevance threshold defined, for both normal and rare instances. Regarding normal bins, standard random under-sampling is performed; *ii)* Regarding rare bins, like SmoteR, k-NN is used to determine the nearest neighbors w.r.t. a given rare case. Next, the distance between this rare case and the remaining elements in the bin is determined. The median of these distances acts as a threshold that defines which method will be used to generate a new sample – if the distance of the randomly chosen nearest neighbor is below it, then SmoteR is applied, otherwise Gaussian Noise will be used to perturb the rare case and generate a new data point. With the introduction of this threshold, the probability of generating samples with a low relevance drops significantly as in this case Gaussian noise is applied to the extreme value at hand. This section is closed by enunciating some of the drawbacks of data-level approaches:

1. Under-sampling techniques may remove important information from the data set;
2. Over-sampling techniques may add information that does not reflect the reality of the problem at hand;
3. Smote-like methods mainly use the notion of distance to create synthesized observations – This notion of distance drops for highly-dimensional data sets, which nowadays are becoming more popular.

2.2 Algorithm-level Approaches

Another common strategy to face Imbalance Domains lies at the algorithmic level. Here, two main branches have been exhaustively studied by the research community.

The first one is known as cost-sensitive learning. Historically, cost-sensitive learning is a field that was developed to introduce costs/benefits in the spectrum of a target variable distribution. The main purpose of this field was to assign costs to a given class during the optimization phase of a learning algorithm independently of its density/frequency. The thought of connecting it to Imbalance Domains only came after the latter started to gain popularity. As the name indicates, cost-sensitive learning aims at providing different costs to instances during the learning phase of a machine learning algorithm such that the model becomes more sensitive to instances to which higher costs are provided than others. In this sense, model optimization is biased toward instances that have a higher cost.

The second one revolves around modifications to the loss function. As this thesis focus on this approach, it will be individually discussed in Section 2.4.

Focusing on cost-sensitive learning, while there are several proposals for classification tasks (e.g. [14–16]), rather few have been made for regression. For the latter, a recently proposed method called DenseWeight is used to calculate costs associated with a continuous variable [17]. This method is built upon Kernel Density Estimation (KDE) to approximate the density function of the target variable. A hyperparameter α responsible for controlling the degree of weighting is also included. Having the 1-to-1 correspondence between the target variable and the weights, the authors couples the latter with the loss function (which they call DenseLoss) such that during the optimization phase the model's internal parameters take into account these costs. Although the model selected was a Neural Network (NN), any learning algorithm which relies on gradient descent and can handle sample weighting should have no trouble incorporating this method.

2.3 Hybrid Approaches

Hybrid approaches are a combination of both data and algorithm methods and have been gaining popularity in the past few years. These hybrid approaches usually resort to ensemble algorithms such as boosting and bagging. Once again, most of the efforts have been made in classification tasks [18–20]. Regarding regression tasks, two examples will be provided below.

The first discussed one is SMOTEBoost [21]. Generally speaking, SMOTEBoost is a combination of the pre-processing data-level approach described in Section 2.1 SmoteR and several well known-variants of the AdaBoost algorithm [22–24]. The algorithm starts by uniformizing sample weights of the training set. Then, for each iteration, a sample with replacement of the training set is performed. With this sample, the embedding of SmoteR is done such that, a given percentage of highly relevant cases are synthesized (note that normal cases are not down-sampled). With this new synthesized set, a Regression Tree is trained, and predictions are made on the original training set. The rest of the algorithm is standard AdaBoost (e.g. [23]). This method proved to provide better results than vanilla implementations, however, the same drawbacks as SmoteR are still present.

The second one lies in the field of Deep Learning and was dubbed Deep Imbalanced Regression (DIR) [3] and brings new contributions as it also has the ability to deal with high-dimensional data. DIR is composed of two algorithms, one that handles the target variable and possible missing values by smoothing its distribution, called Label Distribution Smoothing (LDS), and another that calibrates the feature map statistics (mean and standard deviation) called Feature Distribution Smoothing (FDS).

The construction of the first algorithm, LDS, was motivated by noting that there was a significant difference in the correlation between the error and target variable distribution when comparing the same Deep Learning architecture on two different data sets – both with the same distribution and target range, but one had a continuous target variable (which provided a low correlation), and another a categorical one (which provided a high correlation). While the correlation for the classification problem was expected, as high-density classes are the common ones, and thus their error should be smaller, the correlation for the regression problem was not (it was smooth). This low correlation was justified by stating that the seen (empirical) target distribution did not reflect the real target distribution due to the dependence between data samples at nearby targets (e.g. face images of people ages 30 and 29 should have a stronger relationship than face images of people with age 30 and 70). To circumvent this problem, it was proposed a symmetric kernel that convolves with this empirical target distribution to compute the effective target density distribution. This new distribution can thus be used as a cost-sensitive re-weighting method.

The construction of the second algorithm, FDS, was motivated by the intuition that continuity in the target space should create a corresponding continuity in the feature space [3]. Here, they noted that there was an unrealistic similarity in the learned feature map statistics between different ages (e.g. the mean similarity between age of 30 and 0, where the latter age is a "few-shot" region, i.e., its density is low, was about the same as the similarity between the age of 30 and 25, where both targets had a close density distribution). Motivated by these observations, they created a feature-calibrated layer after the final feature map which smooths the statistics of it using again the similarity kernel. Finally, these statistics are updated across each epoch using momentum update.

Although some interesting points have been considered and DIR proved to give better results than SMOGN, the LDS smoothness technique lies in modifying the distribution of the target variable, to which some of its values may lose representativeness of the real world.

2.4 Loss Functions

One of the most well-explored sub-fields at the algorithmic level lies in the proposals of new loss functions that have the ability to take into account the imbalanced distribution of the target variable. As this branch is the main motivation of this thesis, it is revisited below.

2.4.1 Preliminaries

Let $X \in \mathbb{R}^n$ be a set of independent random variables and $Y \in \mathbb{R}$ the target and dependent random variable. In a supervised setting, a learning algorithm characterized by a set of parameters $\boldsymbol{\Theta}$ aims at finding the optimal function $f : (X, \boldsymbol{\Theta}) \to Y$ that better describes the data. A loss function $\mathcal{L}(f(X, \boldsymbol{\Theta}), Y)$ is usually defined as an indicative measure that tells if the algorithm is *learning* $\mathcal{L} : (f(X, \boldsymbol{\Theta}), Y) \to C \in \mathbb{R}$. Ideally, C should be close to 0.

From a statistical point of view, let $P(X, Y)$ be the joint probability density distribution that describes these variables. Let's also assume that $f(X, \boldsymbol{\Theta}) = f(X)$ without loss of generality. The Expected Prediction Error (EPE) is defined as [25]

$$EPE = \mathbb{E}_{X,Y}[\mathcal{L}(Y, f(X))] \quad . \tag{2.1}$$

In the case where the joint distribution is continuous, EPE is given by

$$\begin{aligned} EPE &= \int_X \int_Y \mathcal{L}(Y', f(X'))P(X', Y')dX'dY' \\ &= \int_X \int_Y \mathcal{L}(Y', f(X'))P(Y'|X')P(X')dX'dY' \\ &= \int_X \mathbb{E}_{Y|X'}[\mathcal{L}(Y, f(X'))|X']P(X')dX' \\ &= \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathcal{L}(Y, f(X))|X]] \quad . \end{aligned} \tag{2.2}$$

It is desirable that EPE is minimum. The unknown function f that minimizes EPE can be determined as

$$\hat{f}(X) = \underset{f(X)}{\arg\min}\, \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathcal{L}(Y, f(X))|X]] \quad . \tag{2.3}$$

Which is equivalent to minimizing EPE pointwise

$$\hat{f}(x) = \underset{c}{\arg\min}\, \mathbb{E}_{Y|X}[\mathcal{L}(Y, c)|X = x] \quad . \tag{2.4}$$

Two common loss functions used are the Squared Error (SE)

$$\mathcal{L}(Y, f(X)) = (Y - f(X))^2 \quad , \tag{2.5}$$

where minimization implies that

$$\hat{f}(x) = \mathbb{E}(Y|X = x) \quad , \tag{2.6}$$

i.e., the EPE is minimum when the learned function follows the expected value of the target variable, and the Absolute Error (AE)

$$\mathcal{L}(Y, f(X)) = |Y - f(X)| \quad , \tag{2.7}$$

where minimization implies

$$\hat{f}(x) = \text{Median}(Y|X = x) \quad , \tag{2.8}$$

implying that the EPE is minimum when the learned function follows the median of the target variable. If there is a linear relationship between Y and X, one of the foundations of Machine Learning is recovered, i.e., the Linear Regression model

$$f(X) = X^T\beta \quad , \tag{2.9}$$

where T denotes the transpose of the vector X and β are the unknown regression coefficients to be determined[*]. Using the Squared Error, EPE Equation (2.1) is given by

[*]Here, the intercept with y axis β_0 is also considered. The first component of X, X_0 is a vector of 1's.

$$\begin{aligned} EPE &= \mathbb{E}[(Y - X^T\beta)^2] = \mathbb{E}[(Y - X^T\beta)(Y - X^T\beta)^T] \\ &= \mathbb{E}[(Y - X^T\beta)(Y^T - \beta^T X)] \\ &= \mathbb{E}[YY^T - Y\beta^T X - X^T\beta Y^T + X^T\beta\beta^T X] \\ &= \mathbb{E}[YY^T - 2\beta XY^T - \beta^T XX^T\beta] \quad , \end{aligned} \tag{2.10}$$

minimization of EPE w.r.t. β yields

$$\hat{\beta} = \arg\min_{\beta} EPE = \mathbb{E}(XX^T)^{-1}\mathbb{E}(X^T Y) \quad .^* \tag{2.11}$$

Most of the time, however, a linear relationship between the independent and dependent variables is insufficient to describe the data. For these cases, a more complex expression for $f(X)$ may be provided. Nowadays, the common approach is to use a machine learning algorithm that *learns* $f(X)$ by itself through some optimization technique.

2.4.2 Asymmetric Loss Functions

One type of loss functions families for imbalanced regression tasks is the so-called asymmetric loss function. Asymmetric loss functions are usually defined as a branched function that changes their dependence on the residuals for a given condition and have been exhaustively studied in the field of Econometrics. This condition usually differentiates over from under predictions – if the residue of a prediction is greater (lower) than a given threshold then the loss function changes its behavior. One that is well explored is the well-known $LINEX$ [7]:

$$\mathcal{L}(x) = b[e^{ax} - ax - 1], \quad , a \in \mathbb{R} \setminus \{0\}, \quad b \in \mathbb{R}_+ \quad , \tag{2.12}$$

where $x = y - \hat{y}$, a is a constant that controls the asymmetry of the estimation. Setting $a > 0$, for under-predictions ($x < 0$) the linear term wins and for over-predictions ($x > 0$) the exponential term wins, and conversely when $a < 0$, hence the name. An interesting property comes when $a \to 0$. In this limit, second-order Taylor expansion yields $e^{ax} \sim 1 + ax + (ax)^2/2$. Replacing this approximation in Equation (2.12) the constant and linear

*Note that in reality, $P(X)$ and $P(Y)$ are always unknown. As such, the expected value $\mathbb{E}(.)$ is replaced by the sample mean (i.e., the training data). In this approximation, the known matrix form solution takes place $\beta = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}^T\mathbf{Y}$, where $\mathbf{X} \in \mathbb{R}^{m\times(n+1)}$, $\mathbf{Y} \in \mathbb{R}^{m\times 1}$ and m is the number of instances in the training set.

terms will be canceled and SE is recovered. $LINEX$ can thus be seen as a generalized version of SE where symmetry is recovered as $a \to 0$.

Another well-known branched loss is the $LINLIN$ [8] loss function:

$$\mathcal{L}(x) = \begin{cases} c_o|x|, & \text{if } \ x > 0 \ , \\ 0, & \text{if } \ x = 0 \ , \\ c_u|x|, & \text{if } \ x < 0 \ , \end{cases} , \tag{2.13}$$

where, c_o and c_u are costs assigned to over and under predictions, respectively. $LINLIN$ is linear for both under and over predictions. In the case where the costs are the same, the Absolute Error (AE) is recovered. For a fixed x, high values of c_o or c_u will imply an high value of $\mathcal{L}(x)$.

Another early proposed loss, which is a combination of SE and AE is the Huber loss [26]:

$$\mathcal{L}(x) = \begin{cases} \frac{1}{2}x^2, & |x| \leq \delta \ , \\ \delta(|x| - \frac{1}{2}\delta), & \text{otherwise} \ . \end{cases} \tag{2.14}$$

Here δ is a control parameter that controls the robustness of predictions. If the residuals are limited by δ, then the squared error is used, while if the residuals are above it, absolute error is used instead, which is more robust (or less sensitive) to outliers. The behavior of the three enunciated loss functions above is depicted in Figure 2.1.

In the context of condition-based maintenance (CBM) and in the field of Deep Learning, the authors in [1] used combinations of linear, quadratic, and logarithmic losses and proposed four asymmetric loss functions for making early predictions on the Remaining Useful Life (RUL) of aircraft gas turbine engines

$$LIN - MSE = \begin{cases} ax, & \text{if } \ x \leq 0 \\ x^2, & \text{otherwise} \end{cases} , MSLE - MSE = \begin{cases} (\log \hat{y} - \log y)^2, & \text{if } \ x \leq 0 \\ x^2, & \text{otherwise} \end{cases} ,$$

$$QUAD - QUAD = \begin{cases} 2ax^2, & \text{if } \ x \leq 0 \\ 2(a + (1 - 2a))x^2, & \text{otherwise} \end{cases} , \ \text{Eq.(2.13)} \ . \tag{2.15}$$

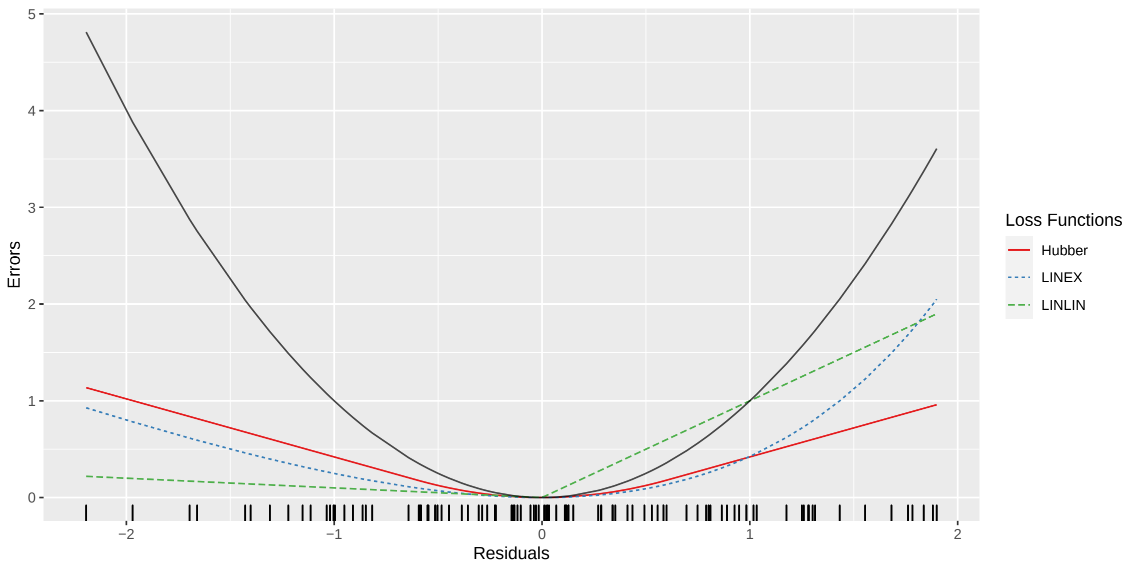

FIGURE 2.1: Errors as a function of the residuals for Hubber (red line, with $\delta = 0.6$), $LINEX$ (dashed blue line, with $a = 0.1, b = 1$), $LINLIN$ (dashed green line, with $c_o = 1, c_u = 0.1$) and SE (black line). Residuals were generated as $x \sim \mathcal{N}(0, 1)$, and their density is represented as ticks on the residual axis (vertical lines).

Results showed that $LIN - LIN$ and $QUAD - QUAD$ outperformed MSE and MAE for different types of architectures. However, these results are sensitive to the degree of asymmetry imposed by the chosen parameters.

Recently, the authors of [27] revisited MSE from a statistical point of view and proposed Balanced MSE. Here, they try to close the gap between Imbalanced Classification and Regression tasks by adapting a technique known as logit adjustment [28] to regression tasks. With this, plus the principle that using a balanced metric on an arbitrary test set is equivalent to using an overall metric on a balanced test set that hypothetically exists [29], they proposed Balanced MSE.

Let $\boldsymbol{x} \in \mathcal{X} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathcal{Y} \in \mathbb{R}^d$. Balanced MSE is built on the following assumptions: *i)* The training and test set are drawn from different joint distributions, i.e., $P_{\text{train}}(\boldsymbol{x}, \boldsymbol{y}) \neq P_{\text{bal}}(\boldsymbol{x}, \boldsymbol{y})$; *ii)* While $P_{\text{train}}(\boldsymbol{y})$ is skewed, $P_{\text{bal}}(\boldsymbol{y})$, the test set distribution, is uniform.; *iii)* The label-conditional probability $P(\boldsymbol{x}|\boldsymbol{y})$ is the same for both training and test set.

Instead of estimating $P_{\text{train}}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Theta})$, their goal is to estimate

$$P_{\text{bal}}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Theta}) = \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}_{\text{pred}}, \sigma^2_{\text{noise}}\mathbf{I}) \quad .^{*} \tag{2.16}$$

*For completeness, σ_{noise} is the standard deviation of an i.i.d. error term $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}}\mathbf{I})$.

This goal is achieved by letting $P_{\text{bal}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$ be estimated by the regressor parameters $\boldsymbol{\Theta}$ determined by minimizing the negative log-likelihood of $P_{\text{train}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$, i.e., the loss function is defined as

$$\mathcal{L} = -\log P_{\text{train}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) \quad , \tag{2.17}$$

where

$$P_{\text{train}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) = \frac{P_{\text{bal}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) \cdot P_{\text{train}}(\boldsymbol{y})}{\int_Y P_{\text{bal}}(\boldsymbol{y'}|\boldsymbol{x};\boldsymbol{\Theta}) \cdot P_{\text{train}}(\boldsymbol{y'})d\boldsymbol{y'}} \quad . \tag{2.18}$$

The last expression is derived using the Bayes Theorem. Replacing Equation (2.18) in (2.17) leads to

$$\mathcal{L} \cong -\log \mathcal{N}(\boldsymbol{y};\boldsymbol{y}_{\text{pred}},\sigma^2_{\text{noise}}\mathbf{I}) + \log \int_Y \mathcal{N}(\boldsymbol{y'};\boldsymbol{y}_{\text{pred}},\sigma^2_{\text{noise}}\mathbf{I}) \cdot P_{\text{train}}(\boldsymbol{y'})d\boldsymbol{y'} \quad , \tag{2.19}$$

where they hide the term $-\log P_{\text{train}}(\boldsymbol{y})$. Thus, Balanced MSE aims at modeling $P_{\text{bal}}(\boldsymbol{y}|\boldsymbol{x})$, while minimizing $\mathcal{L}$ w.r.t. $P_{\text{train}}(\boldsymbol{y}|\boldsymbol{x})$. Notice that MSE can be recovered from Equation (2.19) when $P_{\text{train}}(\boldsymbol{y})$ is uniform. In this case, the integral termnnnnnnnnnnnnnnnnnnnnnnnnnnn of Equation (2.19) is independent of $P_{\text{train}}(\boldsymbol{y'})$ and Gaussian's integral is constant. The loss function becomes $\mathcal{L} = \textit{constant} - \log \mathcal{N}(\boldsymbol{y};\boldsymbol{y}_{\text{pred}},\sigma^2_{\text{noise}}\mathbf{I})$, and thus parameters estimation are the same as in MSE. Thus, Balanced MSE can be seen as a generalized version of MSE.

During the training phase, $P_{\text{bal}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$ is initially predicted, converted into $P_{\text{train}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$ and $\mathcal{L}$ is computed to update $\boldsymbol{\Theta}$ during back-propagation. During the test phase, the output $\boldsymbol{y}_{pred}$ is directly computed from $P_{\text{bal}}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$. To compute the integral term of Equation (2.19), they provided two methods – an exact and a numerical one. A detailed explanation regarding them can be found in their work [27].

Finally, their experimental setup lied in the analyses of uni and multi-dimensional targeted data sets ($\boldsymbol{y} \in \mathbb{R}^d$, $d \geq 1$) and results showed the ability to provide better results than DIR described in Section 2.3, showing that the embedment of alternative loss functions posits a good direction to face imbalanced problems.

Chapter 3

Imbalanced Regression

In this chapter the building blocks needed to introduce Squared Error Relevance Area ($SERA$), the loss function used to optimize models when the target variable faces an asymmetric distribution, will be introduced. In Section 3.1, a general definition of the problem faced will be provided. In Section 3.2 the relevance function will be introduced. At the end of this chapter, Section 3.3, $SERA$ will be introduced and its equivalence with the weighted Sum of Squared Errors will be demonstrated.

3.1 Problem definition

Consider $\mathcal{D}$ a training set defined as $\mathcal{D} = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ is a feature vector of the feature space $\mathcal{X} \in \mathbb{R}^m$ and y_i an instance of the feature space $\mathcal{Y}$ that depends on the feature space $\mathcal{X}$. As enunciated in Chapter 2, a supervised machine learning algorithm objective is to find a function f that maps the feature space $\mathcal{X}$ onto $\mathcal{Y}$, $f : \mathcal{X} \rightarrow \mathcal{Y}$. Depending on the nature of $\mathcal{Y}$, two different types of problems may emerge – if $\mathcal{Y}$ is discrete, a classification one, while if $\mathcal{Y}$ is continuous, a regression one.

To obtain the best approximation function of f, the standard approach in supervised learning is to consider a loss function $\mathcal{L}$, responsible for the optimization of a set of parameters $\boldsymbol{\Theta}$ which tune a model to extract predictions that better describes new instances from the feature space $\mathcal{Y}$.

In this thesis, the focus will be on the problem of imbalanced regression, i.e., when the target variable $\mathcal{Y} \in \mathbb{R}$ presents a skewed distribution and the most important values for the prediction task are extreme (rare) values.

The most commonly used loss function in regression is the Mean Squared Error (MSE). However, this metric is not adequate for our prediction task. The constant which minimizes MSE is the mean of the target variable, as shown in Section 2.4, which is counterintuitive for the predictive focus of extreme values. In an imbalanced regression scenario, an appropriate loss function should search the parameter space $\mathbf{\Theta}$ such that it encompasses a good predictive power for both common (around the mean) and uncommon (extremes) instances of the target domain $\mathcal{Y}$, i.e., it should be agnostic to the imbalance faced by the target variable distribution. However, this is not a trivial task to accomplish.

3.2 Relevance Function

The notion of relevance function was initially proposed in the context of Utility-based Regression in [2] to face the attribution cost problem to a target variable $\mathcal{Y}$. Its formal definition is presented below.

Definition 1. A relevance function $\phi : \mathcal{Y} \rightarrow [0, 1]$ is a continuous function that expresses the application-specific bias concerning the target variable domain $\mathcal{Y}$ by mapping it into a $[0, 1]$ scale of relevance, where 0 and 1 represent the minimum and maximum relevance, respectively.

Due to the generality of this definition, ϕ may take a wide spectrum of forms. Probably, the most straightforward proposition is to consider the normalized inverse of the density function of $\mathcal{Y}$ [2]. Here, instances that have a high-density are considered normal, while instances that have low-density, extreme values. Considering the inversion of the density, extreme values will have higher relevance than normal ones. Let $q(\mathcal{Y})$ be defined as the inverse of the density function of $P(\mathcal{Y})$. For an instance $y_i \in \mathcal{Y}$

$$\phi(y_i) = \frac{q(y_i)}{\sum_{\mathcal{Y}} q(y_i)} \quad . \tag{3.1}$$

The difficulty of this proposal lies in the determination of the probability density function $P(\mathcal{Y})$, which most of the time is unknown. In addition, for real-world data, it is very uncommon for the target variable to follow a well-behaved density function – like a Gaussian one. As such, most of the time the density function is approximated using kernel-density-based methods. Another possible proposal to determine ϕ lies in interpolation methods.

The method that will be used to determine $\phi(y)$ throughout this thesis will now be presented and it was initially proposed in [2]. It is based on a polynomial interpolation method which uses a set of control points obtained from box plot statistics.

Let $S = \{\langle y_k, \varphi(y_k), \varphi'(y_k)\rangle\}_{k=1}^{s}$ be a set of control points, where each triple $(y_k, \varphi(y_k), \varphi'(y_k))$ denotes the values of an instance, the relevance of an instance y_k and the derivative of a relevance at y_k, respectively. A proper interpolation algorithm must have the ability to conform the relevance function to these control points while preserving the properties enunciated in Definition 1. To this end, a shape-preserving interpolation method, called Piece Cubic Hermite Interpolating Polynomials (*pchip*) [30], is used. This method takes as inputs the set of control points S, generating an interpolation function such that it preserves positivity, continuity, and monotonicity, piecewise.

In a perfect world scenario, $s = N$, i.e., for each instance y_i, there should be knowledge about the relevance and its derivatives. However, this is most unlikely in the real world. In this absence, there is a need to find a way to determine reasonable control points automatically. The solution proposed in [6] lies in the use of statistics derived from a modified version of the well-known Tukey's box plot – the adjusted box plot [31]. As opposed to Tukey's box plot, where the outliers are determined assuming symmetry of the underlying probability distribution of a variable, the adjusted box plot takes into account the asymmetry that a variable might face. This accountability is taken by coupling an exponential term to the interquartile range, that depends on the medcouple [32]. Mathematically, the adjusted box plot considers a given instance as an outlier if it is outside the limits

$$\begin{aligned} &[Q_1 - 1.5e^{-4MC}IQR, Q_3 + 1.5e^{3MC}IQR], \text{ if } MC \geq 0 \quad , \\ &[Q_1 - 1.5e^{-3MC}IQR, Q_3 + 1.5e^{4MC}IQR], \text{ if } MC \leq 0 \quad , \end{aligned} \tag{3.2}$$

where $Q_{1,3}$ are the first and third quartiles, respectively, $IQR = Q_1 - Q_3$ is the interquartile range, and MC is the medcouple given by the kernel

$$h(y_i^+, y_j^-) = \frac{(y_i^+ - Q_2) - (Q_2 - y_j^-)}{y_i^+ - y_j^-} \quad , \tag{3.3}$$

where the points y_i^+ belong to the set of points $Y^+ = \{y_i \mid y_i \geq Q_2\}$ and y_j^- belong to the set of points $Y^- = \{y_j \mid y_j \leq Q_2\}$. Thus, for each pair (y_i^+, y_j^-), $h(y_i^+, y_j^-)$ is determined, and the medcouple is the median of all the computed values $H = \{h(y_i^+, y_j^-)\}$

$$MC = \text{median } H \quad . \tag{3.4}$$

Notice that this solution is non-parametric, i.e., there is no dependence on the target variable probability density function and it only relies on the order statistics of the variable.

With this in mind, the automatic approach proposed determines a set of control points S such that: 1) the median denotes a point of no relevance, i.e., $\varphi(y_k) = 0$; 2) the upper and lower limit of the Equation (3.2) denotes points of highest relevance, i.e., $\varphi(y_k) = 1$; 3) as these points correspond to values of the lowest and highest relevance of $\phi(y)$, $\varphi'(y_k) = 0$. Thus, these three sets of control points are fed into *pchip*, which returns an interpolation function of $\phi(y)$. In the end, constant extrapolation is used to encompass all the domain of the variable. In Figure 3.1, it is shown an example of this interpolation method when a variable faces the two types of extreme values (low and high).

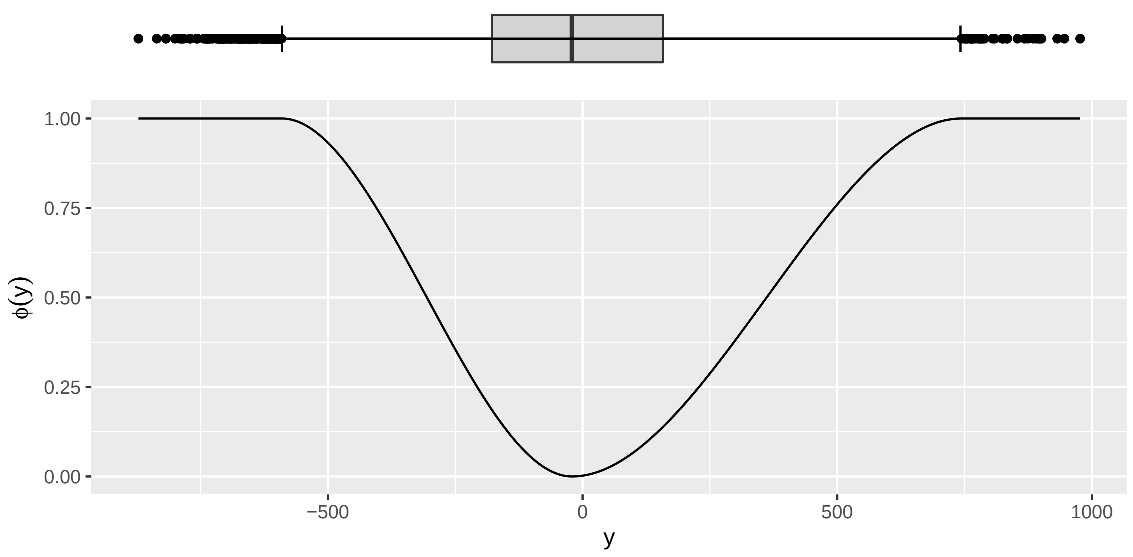

FIGURE 3.1: **Top**: Adjusted boxplot; **Bottom**: Relevance function obtained by considering the interpolation method *pchip* with the set of control points obtained from the adjust-boxplot statistics for ailerons data set.

Due to its high versatility, several studies under this method have been performed in the past years. Such studies include the aforementioned SMOTER [13] and SMOTEGN [33] for data level approaches, utility-based learning [5] for algorithmic level approaches and proposals of new evaluation metrics [2, 4].

3.3 Squared Error Relevance Area (*SERA*)

In the previous section, it was provided the main recipe to approximate the general definition of the relevance function into a non-parametric form. In this section, the evaluation metric proposed in [6] is introduced, which is the center point of the study of this thesis.

Let $\mathcal{D} = \{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{N}$ be a data set and $\phi : \mathcal{Y} \to \{0, 1\}$ a relevance function defined for the target variable Y. Considering the subset $\mathcal{D}^t \subseteq \mathcal{D}$ of instances such that $\mathcal{D}^t = \{\langle \boldsymbol{x}_i, y_i \rangle \in \mathcal{D} \,|\, \phi(y_i) \geq t\}$, the Squared Error-Relevance (SER_t) is defined, for a given cutoff t, as

$$SER_t = \sum_{y_i \in \mathcal{D}^t} (\hat{y}_i - y_i)^2 \quad . \tag{3.5}$$

This is the Sum of Squared Errors (SSE) for all the instances such that the relevance of the target value is bounded by a given threshold t. Since this metric only depends on instances such that $\phi(y_i) \geq t$, the following property holds: for any given $\delta \in \mathbb{R}^+$, s.t. $t + \delta \leq 1$: $SER_{t+\delta} \leq SER_t$. Finally, its maximum and minimum value are ascertained when $t = 0$ and $t = 1$, respectively.

In the same work, the authors took a step further and integrated this estimate w.r.t. all possible cutoff values (i.e., between 0 and 1). There, they defined this area as the Squared Error Relevance Area ($SERA$), and it is given by

$$SERA = \int_0^1 SER_t \; dt = \int_0^1 \sum_{y_i \in \mathcal{D}^t} (\hat{y}_i - y_i)^2 \; dt \quad . \tag{3.6}$$

This area has some important properties that allow understanding the performance of models in an imbalanced regression setting. First, it encompasses all the possible relevance thresholds constrained in the definition of SER_t, removing the need to explicitly define a threshold. Secondly, it is a decreasing and monotonic function. Note that integration is performed under all possible relevance values. Since by definition $\mathcal{D}^{t+\delta} \subseteq \mathcal{D}^t$, the higher the relevance threshold is, the lower will be the number of instances considered, but also more relevant. Thus, on one hand, values of SER_t which have a high relevance will have a greater contribution to this area when compared with instances where the relevance is small. The squared errors of these latter instances are accounted for fewer times for $SERA$, when compared to the high relevance instances. On the other hand, the area will be smaller at points where the relevance is high (as only high relevance instances are taken into consideration). With this, explicit penalization for high relevance errors is performed, which are usually harder to optimize while keeping the entire data domain. Finally, since this metric is built by integrating SER_t over all the relevance domain, which is convex, convexity is also preserved. Also, this metric must be differentiable. By the same token, given that SER_t is differentiable, so is $SERA$. One intuitive and clear way to visualize the errors committed by a given model under $SERA$ is by plotting SER_t across all the relevance domain. Such example is depicted in Figure 3.2. Since this plot will be used throughout this thesis, it is dubbed as $SERA$ curve [6].

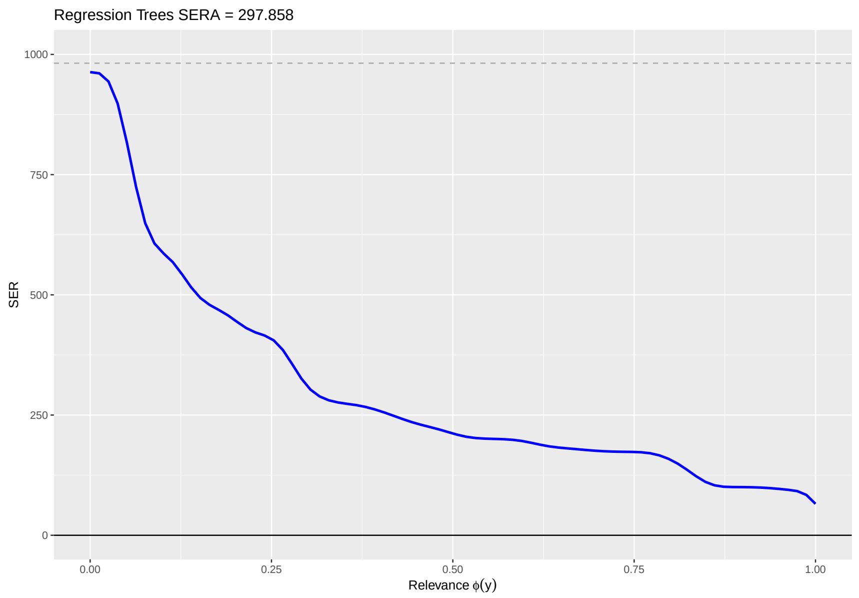

Figure 3.2: $SERA$ curve for predictions obtained from a Regression Tree.

An aspect that needs to be taken into the account is the method used to evaluate $SERA$ numerically. In [6], the trapezoidal rule with a uniform grid discretized into T equally spaced steps, s, is used *. In this thesis, the same principle is followed, with the same settings. Under this approximation, $SERA$ is given by

$$\begin{aligned} SERA &= \int_0^1 SER_t \, dt \\ &\approx \frac{1}{2T}\sum_{k=1}^{T}\Big(SER_{t_{k-1}} + SER_{t_k}\Big) \\ &= \frac{1}{2T}\left(SER_{t_0} + 2SER_{t_1} + \ldots + 2SER_{t_{T-1}} + SER_{t_T}\right) \\ &= \frac{1}{T}\Bigg(\sum_{k=1}^{T-1} SER_{t_k} + \frac{SER_{t_0} + SER_{t_T}}{2}\Bigg) \\ &= \frac{1}{T}\Bigg(\frac{1}{2}\sum_{y_i\in\mathcal{D}^{t_0}}(\hat{y}_i - y_i)^2 + \sum_{y_i\in\mathcal{D}^{t_1}}(\hat{y}_i - y_i)^2 + \ldots \\ &\qquad \ldots + \sum_{y_i\in\mathcal{D}^{t_{T-1}}}(\hat{y}_i - y_i)^2 + \frac{1}{2}\sum_{y_i\in\mathcal{D}^{t_T}}(\hat{y}_i - y_i)^2\Bigg) \ . \end{aligned} \tag{3.7}$$

*By default, $s = 0.001$, which means that the number of intervals taken to approximate the integral using the trapezoidal rule is $T = 1000$.

Regarding its time complexity, the trapezoidal rule has a computational complexity of $O(T)$. SER_t has a computational complexity of $O(|\mathcal{D}^t|)$, where $|\mathcal{D}^t|$ is the number of instances with relevance higher or equal to a given threshold t. $SERA$ will consider $|\mathcal{D}^{t_0}| + |\mathcal{D}^{t_1}| + ... + |\mathcal{D}^{t_T}|$ instances for all the T steps of the trapezoidal rule. In the worst-case scenario, all the target values have a constant and maximum relevance equal to 1. In that case, $|\mathcal{D}|$ is the number of instances accounted for all steps. Thus, $SERA$ will have a computational complexity of $O(T \times |\mathcal{D}|)$.

3.3.1 Equivalence with weighted Sum of Squared Errors

In this section, a proof of how $SERA$, under the approximation of the trapezoidal rule, can be interpreted as a weighted version of SSE, is provided. Starting the last equality from Equation (3.7)

$$\begin{aligned} SERA = \frac{1}{T}\Bigg(\frac{1}{2}\sum_{y_i \in \mathcal{D}^{t_0}} (\hat{y}_i - y_i)^2 + \sum_{y_i \in \mathcal{D}^{t_1}} (\hat{y}_i - y_i)^2 + ... \\ + \sum_{y_i \in \mathcal{D}^{t_{T-1}}} (\hat{y}_i - y_i)^2 + \frac{1}{2}\sum_{y_i \in \mathcal{D}^{t_T}} (\hat{y}_i - y_i)^2\Bigg) \quad , \end{aligned} \tag{3.8}$$

consider a particular instance, y_j, and lets evaluate its contribution to $SERA$, denoted by C_j. If this instance has the lowest relevance possible, $\phi(y_j) = 0$, it will only be considered in the first summation of the Equation (3.8), i.e., $C_j = (y_j - \hat{y}_j)^2/2$. Conversely, if it has the highest relevance value, $\phi(y_j) = 1$, its contribution will be over all possible intervals $k \in [0, T]$ and $C_j = T(y_j - \hat{y}_j)^2$. For a relevance $0 \leq \phi(y_j) < 1$, the error will be taken into the account for $k = \{0, 1, ..., K\}$, i.e.

$$C_j = \left(\frac{1}{2} + n_j\right)(\hat{y}_j - y_j)^2 \quad , \tag{3.9}$$

where

$$n_j = \sum_{k=1}^{K} \mathbf{1}\left(y_j \in \mathcal{D}^{t_k}\right), \tag{3.10}$$

here, $\mathbf{1}(.)$ is an indicator that takes the value of 1 if the argument holds, and 0 otherwise. Thus, under the trapezoidal rule, $SERA$ can be also given by

$$SERA = \sum_{i=1}^{N} \sigma_i (\hat{y}_i - y_i)^2 \quad , \tag{3.11}$$

where σ_i is the ratio between the number of times a given instance contributes to $SERA$ and the total number of intervals T. The algorithm to evaluate it is presented in Algorithm 1.

Algorithm 1 Evaluate σ_i. Determines the number of times a given instance contributes to $SERA$.

```
    Input: $\phi(y_i)$ : The relevance of the instance $y_i$; steps : A vector with equally spaced steps of size $T + 1$.
    Output: $\sigma_i$.
 1: $\sigma_i \leftarrow 1/2$                                    $\triangleright$ $\phi_i = 0$ contribution.
 2: if $\phi(y_i) = 0$ then
 3:     return $\sigma_i = 1/2T$
 4: else if $\phi(y_i) = 1$ then
 5:     return $\sigma_i = 1$
 6: else
 7:     for $k \leftarrow 2$ to $T$ do
 8:         if $\phi(y_i) \geq steps[k]$ then
 9:             $\sigma_i \leftarrow \sigma_i + 1$
10:         end if
11:     end for
12:     return $\sigma_i/T$
13: end if
```

Thus, under the trapezoidal rule, $SERA$ can be seen as a cost-sensitive re-weighting method when using SSE as a loss function, where weights arise naturally as a counting measure of a given instance contribution to $SERA$.

Chapter 4

Model Optimization in Imbalanced Regression

In the previous chapter, $SERA$, the evaluation metric used to assess the quality of a model when trained under a data set that presents extreme values in the target variable, was introduced. In this chapter, the main work of this thesis, an experimental study that shows $SERA$ ability as a loss function to improve the predictions of extreme and normal values will be provided.

To accomplish this goal, two Gradient Boosting Machines models will be considered – Extreme Gradient Boosting Machines (XGBoost) and Light Gradient Boosting Machines (LGBM) under a variety of data sets that have a target variable that presents a skewed distribution.

In Section 4.1 the necessary modifications to implement $SERA$ in these algorithms will be provided. In Section 4.2, an initial description of the data sets used will be provided. In addition, the set of tasks that will be performed across this chapter to justify why $SERA$ is an appropriate loss function to be used when the target variable faces an unbalanced distribution will be enunciated. In Section 4.3, a grid-search procedure with cross-validation will be performed for the considered models described in Section 4.1 to search for the better parameters of each model for a given data set. Results will be assessed using the Bayes-Sign Test and conclusions from it will be drawn. Finally, in Section 4.4 results in the out-of-sample estimations will be evaluated under $SERA$ curves.

4.1 Optimization Loss Function for Imbalanced Regression

As previously stated, the objective of this thesis is to study the impact of embedding $SERA$ as a loss function in supervised learning algorithms. To this end, efforts are driven by using two well-known implementations of Gradient Tree Boosting Machines – XGBoost [34] and LGBM [35]. Its use in already implemented learning algorithms only requires the proposal of a custom loss function, where the only thing needed to provide are the first and second-order derivatives. In this work, it was used the implementations that can be found in the packages `xgboost` [36] and `lightgbm` [37] from **R** programming language [38].

The first-order derivative of $SERA$ w.r.t. a given prediction $\hat{y}_j$ is given as follows

$$\begin{aligned}\frac{\partial SERA}{\partial \hat{y}_j} &= \frac{\partial}{\partial \hat{y}_j} \int_0^1 \sum_{y_i \in \mathcal{D}^t} (\hat{y}_i - y_i)^2 \; dt \\ &= 2 \int_0^1 \sum_{y_i \in \mathcal{D}^t} (\hat{y}_i - y_i) \; \delta_{ij} \; dt\end{aligned} \quad , \tag{4.1}$$

where δ_{ij} is the Kronecker's delta, which takes value of 1 if $i = j$ and 0 otherwise. Since the derivative must be taken into account over all the possible relevance values a given instance is encompassed in, the equation above is expressed as

$$\frac{\partial SERA}{\partial \hat{y}_j} = 2 \int_0^1 (\hat{y}_j - y_j)\Big|_{y_j \in \mathcal{D}^t} \; dt \quad . \tag{4.2}$$

The second derivative w.r.t. a given prediction $\hat{y}_j$ is obtained by

$$\frac{\partial^2 SERA}{\partial \hat{y}_j^2} = 2 \int_0^1 \mathbf{1}(y_j \in \mathcal{D}^t) \; dt \quad . \tag{4.3}$$

To approximate the values of the two derivatives, recall from Section 3.3.1 that under the trapezoidal rule $SERA$ is given by Equation (3.11)

$$SERA = \sum_{i=1}^{N} \sigma_i (\hat{y}_i - y_i)^2 \quad . \tag{4.4}$$

The first derivative w.r.t. a prediction $\hat{y}_j$ is given by

$$\begin{aligned}\frac{\partial SERA}{\partial \hat{y}_j} &\approx \frac{\partial}{\partial \hat{y}_j} \sum_{i=1}^{N} \sigma_i (\hat{y}_i - y_i)^2 \\ &= 2 \sum_{i=1}^{N} \sigma_i (\hat{y}_i - y_i) \delta_{ij} \\ &= 2\sigma_j (\hat{y}_j - y_j)\end{aligned} \quad , \tag{4.5}$$

while the second derivative w.r.t. a prediction $\hat{y}_j$ yields

$$\frac{\partial^2 SERA}{\partial \hat{y}_j^2} \approx 2\sigma_j \quad . \tag{4.6}$$

Due to the nature of σ, the first and second-order derivatives will be biased towards points of high relevance (as $\sigma \rightarrow 0$ for common values). Interestingly enough, in the context of LGBM, which uses Gradient-based One-Side Sampling (GOSS) to sample higher gradient instances in the tree optimization phase (thus focusing on instances that usually have higher errors), instances with higher relevance will be preferred over lower relevance ones in this algorithm.

This section is closed by performing a study on the degree of error committed by using the approximations above (Equations (3.11), (4.5) and (4.6)) against the use of the trapezoidal rule directly on Equations (3.6), (4.2) and (4.3). For that, the predictions obtained for XGBoost optimized with $SERA$ across the data sets described in Section 4.2 are used. The rationale is the following: 1) for each data set, $SERA$, the first and second derivatives are computed, using both methods; 2) the absolute difference between the results obtained from both methods is measured; 3) an average of these differences over all instances is performed. The results obtained using this evaluation are depicted in the left box plot of Figure 4.1.

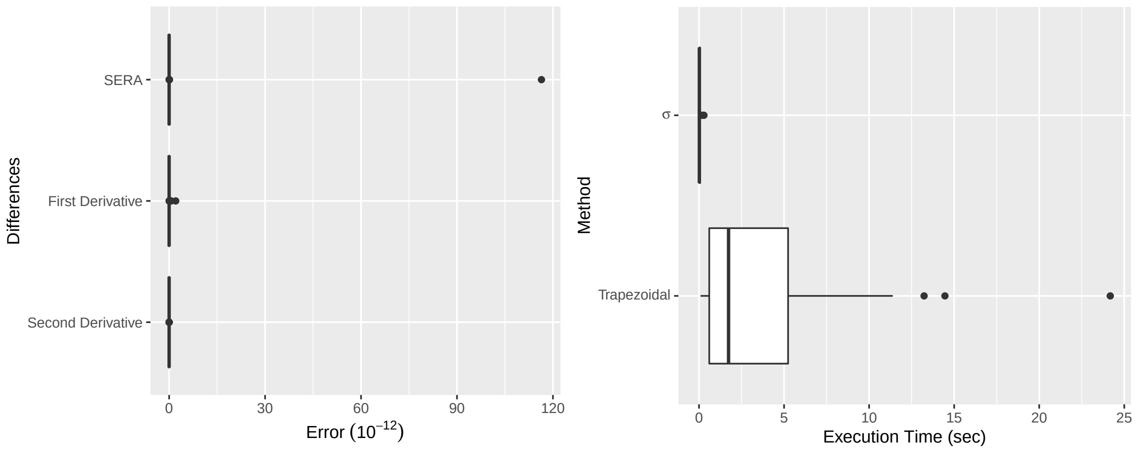

Figure 4.1: **Left**: Absolute error differences for $SERA$, the first and second derivatives between the proposed approximations and the trapezoidal rule. **Right**: Execution time (in seconds) of $SERA$, the first and second derivative for a given data set, under σ approximation and the trapezoidal rule.

Results show that $SERA$, first and second derivative approximations have a minor difference w.r.t. the trapezoidal implementation (order of 10^{-12}). On the right box plot of Figure 4.1, it is also shown the difference in execution time using both methods for each data set. Here, the execution time is measured as the time taken to evaluate $SERA$, the first and second derivatives. The box plot shows that there is a non-negligible difference between our approximation and the trapezoidal rule as the size of a data set increases.

From this point on, XGBoost and LGBM models optimized with $SERA$ are designated as XGBoost^S and LGBM^S, while models optimized with MSE, which will be the baseline metric that will be used to compare $SERA$ with, are designated as XGBoost^M and LGBM^M , respectively.

4.2 Experimental Setup

To study the effects of using $SERA$ as a loss function, a wide range of data sets from several domains in the context of imbalanced regression are used. These data sets, with their respective main properties, are presented in Table 4.1. From them, it was extracted the number of instances $|\mathcal{D}|$, the number of nominal (Nom) and numerical (Num) variables. In addition, and to give a notion of the imbalance present in the target variable, the automatic method proposed in [6] that is based on the adjusted box plot is used. An extreme value is considered if it has a relevance of 1. The number of instances that satisfy this condition is represented as $|\mathcal{D}_R|$. The Imbalance Ratio (IR) is calculated as the ratio between $|\mathcal{D}_R|$ and the number of "normal" instances (in this case, $\phi(y) < 1$) as $|\mathcal{D}_R|/|\mathcal{D}| \times 100\%$. Finally, the type of imbalance of each target variable is also included. It is given by the following rule – if the adjusted box plot only presents outliers below or above the respective fence, the type of extremes is low (L) or high (H), respectively, while if it presents outliers below and above the fences, the type is both (B).

To assess the effectiveness of each model, a random partition for each data set is performed. This partition is made such that 80% of the data will be used to tune the models while the remaining 20% to make predictions under the best model configuration found in a given data set. To tune the parameters for each model, a grid-search approach with a 10-fold stratified cross-validation is used. A workflow of a given algorithm j is defined as the tuple $\boldsymbol{W}^{(j)} = (\boldsymbol{M}_j, \boldsymbol{\Theta}^{(j)}) = \{\mathcal{W}_q^{(j)}\}_{q=1}^{e}$, where e is the number of different workflows considered for a given tuple, $\boldsymbol{M}_j$ denote the algorithm used, $\boldsymbol{\Theta}^{(j)}$ the respective set of parameters, which are described in Table 4.2.

Given the workflows obtained from the grid-search, the following methodology to answer the question that motivated this thesis is provided. It consists of the following tasks:

Table 4.1: Data sets description: $|\mathcal{D}|$ - nr of instances, **Nom** - nr. of nominal attributes, **Num** - nr. of numeric attributes, $|\mathcal{D}_R|$ - nr. of extreme (rare) instances, i.e. $\phi(y) = 1$ IR - imbalance ratio and Type - type of extremes.

id	dataset	$\lvert\mathcal{D}\rvert$	Nom	Num	$\lvert\mathcal{D}_R\rvert$	IR	Type
1	diabetes	35	0	3	4	12.90	H
2	triazines	151	0	61	4	2.72	B
3	a7	160	3	9	7	4.58	H
4	autoPrice	165	10	16	3	1.85	L
5	elecLen1	399	0	3	4	1.01	H
6	housingBoston	407	0	14	40	10.90	B
7	forestFires	416	0	13	7	1.71	H
8	wages	429	7	4	1	0.23	B
9	strikes	501	0	7	1	0.20	H
10	mortgage	841	0	16	60	7.68	L
11	treasury	841	0	16	79	10.37	L
12	musicorigin	848	0	118	15	1.80	B
13	airfoild	1203	0	6	11	0.92	H
14	acceleration	1387	3	12	30	2.21	B
15	fuelConsumption	1413	12	26	27	1.95	B
16	availablePower	1443	7	9	75	5.48	B
17	maxTorque	1442	13	20	43	3.07	B
18	debutenizer	1918	0	8	90	4.92	H

id	data set	$\lvert\mathcal{D}\rvert$	Nom	Num	$\lvert\mathcal{D}^R\rvert$	IR	Type
19	space_ga	2487	0	7	21	0.85	B
20	pollen	3080	0	5	32	1.05	B
21	abalone	3343	1	8	374	12.60	B
22	wine	5199	0	12	1022	24.47	H
23	deltaAilerons	5705	0	6	528	10.20	B
24	heat	5922	3	9	39	0.66	B
25	cpuAct	6555	0	22	227	3.59	L
26	kinematics8fh	6556	0	9	50	0.77	B
27	kinematics32fh	6556	0	33	53	0.82	B
28	pumaRobot	6556	0	33	91	1.41	B
29	deltaElevation	7615	0	7	1802	31	H
30	sulfur	8065	0	6	606	8.12	B
31	ailerons	11003	0	41	186	1.72	B
32	elevators	13280	0	18	1598	13.68	B
33	calHousing	16513	0	9	23	0.14	L
34	house8H	18229	0	9	305	1.70	B
35	house16H	18229	0	17	303	1.69	B
36	onlineNewsPopRegr	31716	0	60	2879	9.98	B

Table 4.2: Models parameters considered for grid-search.

Model	**R Package**	**Parameters**

XGBoost
LGBM
	xgboost [36]
	lightgbm [35]
		nrounds = $\{250, 500\}$
		max_depth = $\{3, 5, 7\}$
		$\eta = \{10^{-3}, 10^{-2}, 10^{-1}\}$

T1: For each data set, and for each model in $\boldsymbol{M}$, the workflow that had the lowest score according to $SERA$ is selected. This score is calculated by averaging the results obtained by cross-validation on the 80% partition.

T2: Given the best workflows, they are compared using the Bayes Sign Test [39]. This task will be performed in Section 4.3. A workflow that is optimized using a standard loss function (MSE) is designated as $\mathcal{W}^M$, while if optimized with $SERA$, $\mathcal{W}^S$.

T3: Finally, the best workflows for each data set obtained with the partitioned 80% are trained, and with the remaining 20% it will be assessed the quality of $SERA$ as an optimization loss function. This quality will also be evaluated by plotting $SERA$ curves. This task will be performed in Section 4.4.

4.3 Results on Model Optimization

With the top workflows from each model obtained by **T1**, the performance of the models can be assessed in task **T2**. For that, the Bayes Sign Test is used. Briefly, this test compares two models on a multi-data set scenario by measuring their score difference (a prior probability) for all data sets, returning a probability measure (the posterior) hinting if a model is practically better than another, or if they are equivalent. This equivalence is measured in a given interval and is defined as the Region Of Practical Equivalence (ROPE), R [40]. The prior z_i, where i indicates a given data set, is determined by averaging the normalized difference below for all k-folds

$$z_i = \frac{1}{10}\sum_{k=1}^{10} \frac{\mathcal{L}_k(\mathcal{W}^S) - \mathcal{L}_k(\mathcal{W}^M)}{\mathcal{L}_k(\mathcal{W}^M)} \quad , \tag{4.7}$$

and taking $\mathcal{L}$ as $SERA$ or MSE. After determining this mean difference for all data sets, the Bayes Sign Test receives as input the vector $\boldsymbol{z}$ and a ROPE between $[-1\%, 1\%]$, returning the posterior probability $p(z)$ that a given model is practically better or equivalent than the other. A more detailed explanation with an illustrative example is provided in Appendix A.

The results from this evaluation are depicted in Figure 4.2 and provide two perspectives according to the considered error metrics. Regarding MSE (left column), the standard models are practically better (with a $p(z)$ of 0.92 for XGBoost^M and a $p(z)$ of 0.97 for LGBM^M). Concerning $SERA$ (right column), results show that both algorithms when optimized with $SERA$ are practically better (with a $p(z)$ of 0.67 for XGBoost^S and a $p(z)$ of 0.70 for LGBM^S).

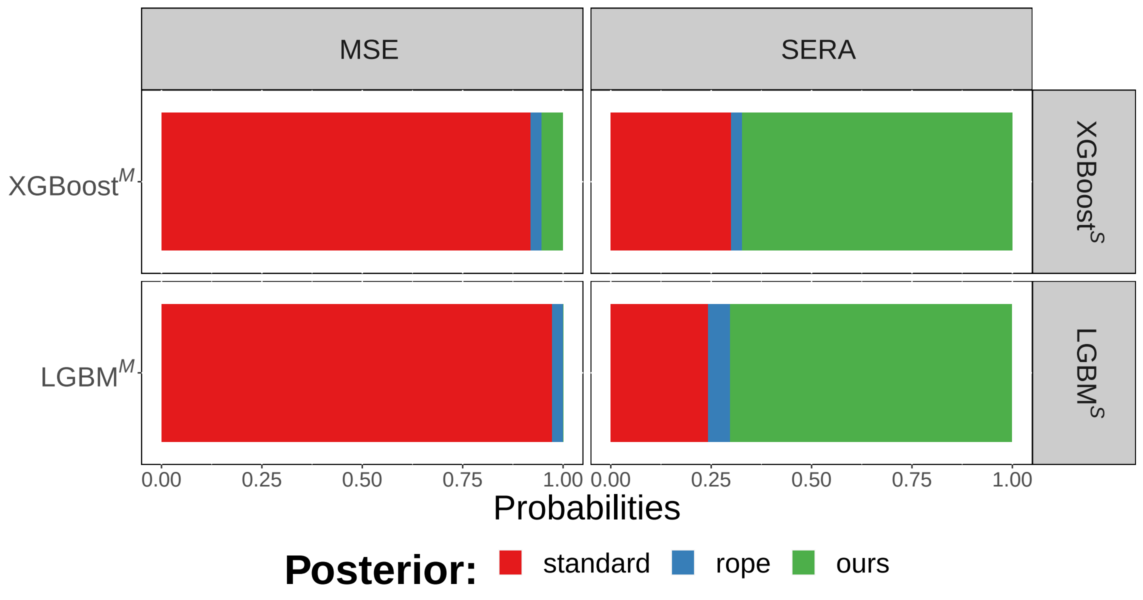

FIGURE 4.2: Comparison between the models optimized with $SERA$, LGBM^S and XGBoost^S, against the standard models LGBM^M and XGBoost^M. Each color denotes the probability of the proposed implementation (with $SERA$, green) or standard (with MSE, red) being practically better or equivalent (blue) to one another according to the Bayes Sign Test with the ROPE interval $[-1\%, 1\%]$. The left and right columns denote the results of the Bayes Sign Test with MSE and $SERA$, respectively.

Thus, from a statistical point of view and in a model optimization scenario, there is a clear trade-off between the standard and the models which use $SERA$ as a loss function when assessing their scores with different metrics. This was somewhat expected as these metrics have a different predictive focus as was already mentioned above. Nevertheless, from this test, it is possible to infer the ability of $SERA$ as a loss function to lower the errors obtained in a problem of Imbalance Regression. With this, the second task **T2** is finished and partially answered the question that motivated this thesis. Next, the objective is to show that $SERA$ does improve the predictive power for both common and extreme values in an out-of-sample scenario (i.e., using the test data).

4.4 Results in Out-of-Sample

Using the parameters found in the best workflows obtained for each model and each data set in task **T1**, the models are trained and the predictions with the (20%) out-of-sample data are assessed. Given these predictions, $SERA$ and MSE are calculated for all the models in the considered data sets (cf. Tables B.1 and B.2 in Appendix B).

From the obtained results, a rank evaluation of the proposed models is considered. For that purpose, and for a given data set, the rank of 1 is assigned to the model which provided the lowest score. Figure 4.3 depicts the rank distribution for each model over all the considered data sets.

Focusing on the left-side of Figure 4.3, where the rank was evaluated under MSE, LGBM^M was the top performer, followed by XGBoost^M (with a median rank of 2) and, finally, XGBoost^S and LGBM^S with a respective median rank of 3 and 4. Focusing on the right-side of the same figure, where the rank was evaluated under $SERA$, both LGBM^S and XGBoost^S had a median rank of two (although LGBM^S had a better performance overall), followed by LGBM^M and XGBoost^M both with a median rank of 3. From these ranking distributions, the somewhat expected conclusion can be taken: models that are optimized with their respective evaluation metric will be the top performers.

Next, each algorithm is considered independently by counting the number of times the models optimized with $SERA$ had the lowest error w.r.t. the standard ones. While for XGBoost, XGBoost^S had the lowest score in 4 and 27 of the data sets when evaluated by MSE and $SERA$, respectively, for LGBM, LGBM^S had a better score in 1 and 26 of the data sets when evaluated by MSE and $SERA$. These results show that $SERA$ is

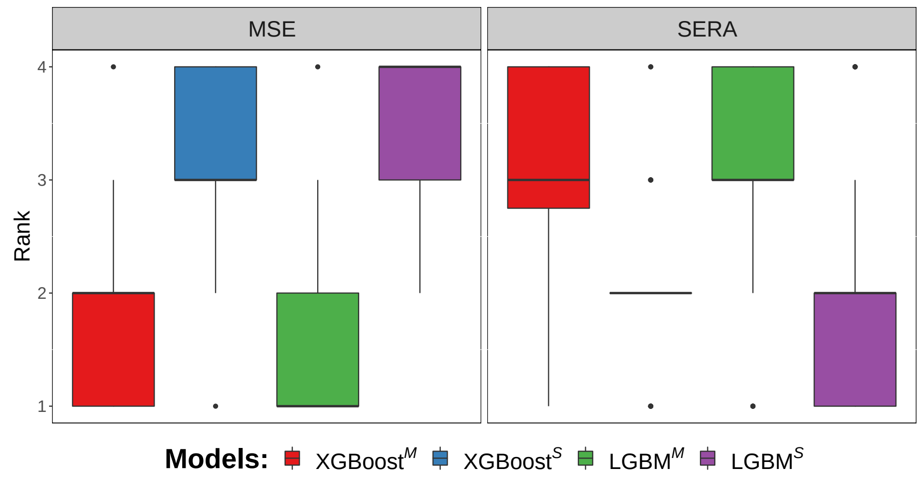

FIGURE 4.3: Rank distribution of models by MSE and $SERA$ results in out-of-sample.

model selection dependent as different models optimized by it may lead to significant error differences when compared with its respective baseline.

Finally, from all the considered models, the model with the lowest MSE and $SERA$ errors for a given data set is extracted. Results show that LGBM variants were the top performers for the considered metrics, having the lowest score in 20 and 16 of the data sets for MSE and $SERA$, respectively. Regarding MSE, LGBM^M was followed by XGBoost^M, having the lowest score in 15 of the considered data sets, and finally XGBoost^S outperforming in only one of the data sets. Regarding $SERA$, LGBM^S was followed by XGBoost^S, with the lowest score in 8 of the considered data sets, XGBoost^M with the lowest score in 7 of the data sets, and finally, LGBM^M outperforming in the remainder.

The results presented above answers the question that motivates this thesis, i.e., $SERA$ is an adequate loss function to be used when the predictive focus are extreme values.

The following study aims to show that, even in data sets where models optimized with MSE provided a better $SERA$ estimation, there it still exists a domain in the relevance space where models optimized with $SERA$ surpass the previous ones – such domain is lower bounded by a point of relevance defined as a turning point. Such point is defined as the minimum relevance value φ for which a model optimized with $SERA$ has a $SERA$ estimate for all the values with a relevance greater or equal to φ, i.e. $SERA_{\phi(.)\geq\varphi}(\mathcal{M}^S)$, lower than the $SERA$ estimate obtained by the standard model in the same conditions, i.e. for all the values with a relevance greater or equal to φ, i.e. $SERA_{\phi(.)\geq\varphi}(\mathcal{M}^M)$. More formally, and for a specific data set, the turning point is then a threshold ϕ_t obtained by

$$\phi_t = \min\{\varphi \in [0,1] \mid SERA_{\phi(.)\geq\varphi}(\mathcal{M}^S) < SERA_{\phi(.)\geq\varphi}(\mathcal{M}^M)\} \quad . \tag{4.8}$$

These turning points are included in the study of $SERA$ curves, for six selected data sets. Recall that these curves are built by calculating the error SER_t as the relevance threshold t for $\phi(y)$ increases and are shown in Figure 4.4.

In the curves of Figure 4.4, the turning points are represented by dashed lines, and the shadowed regions represent the relevance domain for which the condition above holds.

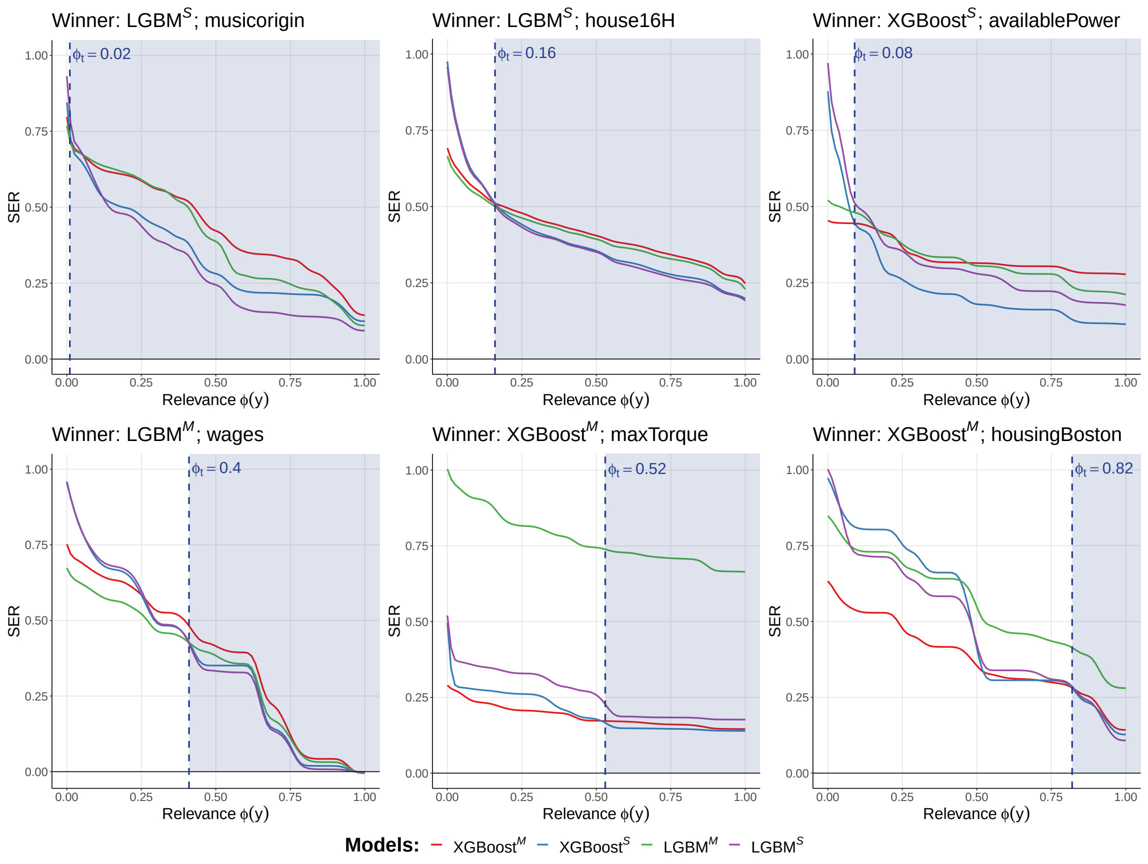

Figure 4.4: $SERA$ curves for six selected data sets. The first row provides data sets where our models had the lowest error, while the second row provides data sets where XGBoost^M had the lowest error. The highlighted region in each graph depicts the turning point where models optimized with $SERA$ started to have a lower error w.r.t. standard models.

The plots from the first row of Figure 4.4 show us $SERA$ curves where models that were optimized with $SERA$ had the lowest score. Regardless of data set, note that the model which obtained a better estimate had a turning point at a relatively low relevance value, showing the capability of $SERA$ as an optimization loss function to lower the prediction error not only on high relevance points, but on low relevance points as well.

Regarding the second row of plots in Figure 4.4, where the chosen data sets had a better estimate when optimized by MSE, the curves depicted show that even when models optimized by $SERA$ are surpassed, they still have the ability to provide a better estimate on different ranges of the relevance domain ($\phi_t = 0.4$, $\phi_t = 0.52$, $\phi_t = 0.82$ for wages, maxTorque and housingBoston data sets, respectively).

From this analysis, the conclusion that $SERA$ can be used as a loss function to reduce errors in both extreme and common values (first row of Figure 4.4) can be taken. In addition, even when models optimized with $SERA$ do not provide the best score across the whole relevance domain, they can still perform better for different relevance domains (second row from Figure 4.4). With this, task **T3** is finished and the question that motivated this thesis is answered, i.e., $SERA$ can be used as an optimization loss function to optimize the reduction of extreme values.

Chapter 5

Theoretical Discussion

In the previous chapter an experimental study was performed to demonstrate that $SERA$ can be used as a loss function to optimize common and extreme values – the main motivation of this thesis. In this chapter, a theoretical discussion that studies the impact of $SERA$ during the optimization process of Gradient Boosting Machines will be provided.

In Section 5.1, an overview of Gradient Boosting Machines, the baseline used, will be revisited. In Section 5.2, the necessary modifications to embed $SERA$ in Gradient Boosting Machines will be presented. In Section 5.3, a case study will be performed which shows the impact of using different values of intervals T used to discretize $SERA$ under the trapezoidal rule (cf. Equation (3.8)) will be performed.

5.1 Review on Gradient Boosting Machines

This review follows closely the paper by Friedman where Gradient Boosting Machines was initially proposed [41].

As discussed in Chapter 2, from a statistical point of view, a Machine Learning algorithm aims at minimizing EPE

$$EPE = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathcal{L}(Y, F(X))|X]] \quad , \tag{5.1}$$

w.r.t. the unknown function $F(X)$

$$\hat{F}(X) = \underset{F(X)}{\arg\min}\, \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathcal{L}(Y, F(X))|X]] \quad . \tag{5.2}$$

However, knowing the specific form of $F(X)$ becomes intractable in most applications. The standard approach to overcome this difficulty comes from defining a parametric function that depends on the inputs X and on a set of parameters $\boldsymbol{a}$. With this in mind, and motivated by Boosting Machines, $F(X)$ is restricted to a family of functions of the form

$$F\left(X,\{\beta_m,\boldsymbol{a}_m\}_{m=1}^{M}\right)=\sum_{m=1}^{M}\beta_m h\left(X,\boldsymbol{a}_m\right) \quad . \tag{5.3}$$

Here, the summation up to M denotes the number of iterations (boosters) that are taken into consideration, and $h(X,\boldsymbol{a}_m)$ is yet another family of functions (weak-learners) that depends on the inputs X and it is parametrized by the vector of parameters $\boldsymbol{a}_m$. These weak-learners may take many forms, such as a Neural Network, an SVM, or a CART. Minimizing EPE in this setting implies finding the best set of parameters $\{\beta_m^*,\boldsymbol{a}_m^*\}_{m=1}^{M}$ such that

$$\{\beta_m^*,\boldsymbol{a}_m^*\}_{m=1}^{M}=\underset{\{\beta_m,\boldsymbol{a}_m\}_{m=1}^{M}}{\arg\min}\ \mathbb{E}_X\left[\mathbb{E}_{Y|X}\left[\mathcal{L}\left(y,\sum_{m=1}^{M}\beta_m h\left(X,\boldsymbol{a}_m\right)\right)|X\right]\right] \quad . \tag{5.4}$$

Given the form in Equation (5.3), to compute the parameters $\{\beta_m,\boldsymbol{a}_m\}_1^M$ which minimizes the Expected Prediction Error it is standard to use numerical optimization methods such as steepest descent. This optimization technique, instead of being used in the parameter space, is used in the function space.

In this space, the objective is to construct the additive and final approximation function

$$F(X)=F_0(X)+\sum_{m=1}^{M}F_m(X) \quad , \tag{5.5}$$

where $F_0(X)$ can be defined, for example, as the constant that minimizes the EPE

$$F_0(X)=\underset{c}{\arg\min}\,\mathbb{E}_X[\mathbb{E}_{Y|X}[\mathcal{L}(Y,c)|X=x]] \quad , \tag{5.6}$$

and $\{F_m(X)\}_1^M$ are incremental functions or boosters obtained by the optimization model, defined as

$$F_m(X)=-\gamma_m\left[\frac{\partial\mathcal{L}(y,F(X))}{\partial F(X)}\right]_{F(X)=F_{m-1}(X)}=-\gamma_m r_m(X) \quad , \tag{5.7}$$

$r_m(X)$ are the so-called pseudo-residuals and γ_m can be obtained by performing the line search

$$\gamma_m = \arg\min_{\gamma} \mathbb{E}_{y,X}[\mathcal{L}(y, F_{m-1}(X) - \gamma r_m(X))] \quad . \tag{5.8}$$

When dealing with finite data, this method breaks down because the sample data that is given cannot be assumed to represent a population. With this in mind, Equation (5.4) is approximated to the known sample

$$\{\beta_m^*, \boldsymbol{a}_m^*\}_{m=1}^M = \arg\min_{\{\beta_m, \boldsymbol{a}_m\}_{m=1}^M} \left(\sum_{i=1}^N \mathcal{L}(y_i, \sum_{m=1}^M \beta_m h\left(\boldsymbol{x}_i, \boldsymbol{a}_m\right)) \right) \quad .^* \tag{5.9}$$

This forms a set of M first order (and possibly non-linear, depending on the form of $\mathcal{L}$) differential equations, which can become infeasible to solve. Motivated by Boosting Machines, instead, Friedman proposed a greedy-stagewise approach which consists of the following steps.

First, instead of solving the system of equations above, the problem is reduced to the determination of $\{\beta_m^*, \boldsymbol{a}_m^*\}$, stagewise, i.e., for $m = 1, 2, ..., M$

$$(\beta_m^*, \boldsymbol{a}_m^*) = \arg\min_{\beta, \boldsymbol{a}} \sum_{i=1}^N \mathcal{L}(y_i, F_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i, \boldsymbol{a})) \quad , \tag{5.10}$$

and then $F_m(\boldsymbol{x})$ is updated as

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \beta_m h(\boldsymbol{x}, \boldsymbol{a}_m) \quad . \tag{5.11}$$

In situations where Equation (5.10) is still hard to solve, instead of performing optimization in the parameter space, the function space is used. This construction is done by noting that the best greedy step in the steepest descent towards the optimal solution for a given iteration m is done in the direction s.t. $F_m(\boldsymbol{x})$ is optimal. This greedy step is done in the direction of the pseudo-residuals $\gamma_m r_m(\boldsymbol{x})$, and it is achieved by seeking $\beta_m h(\boldsymbol{x}, \boldsymbol{a}_m)$ that is most parallel to the pseudo-residuals as possible. With this, the optimization problem reduces to approximating the unknown function $\beta_m h(\boldsymbol{x}, \boldsymbol{a}_m)$ into the pseudo-residuals, $\gamma_m r_m(\boldsymbol{x})$. Since these pseudo-residuals are known, any learning algorithm can be used to learn $h(\boldsymbol{x}, \boldsymbol{a}_m)$.

Finally, the line-search defined in Equation (5.8) is obtained by

[*] Note that randomness of the variable X was dropped to $\boldsymbol{x}$ because this minimization is done under a sample, i.e., the training set.

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} \mathcal{L}(y_i, F_{m-1}(\boldsymbol{x}_i) + \gamma h_m(\boldsymbol{x}_i, \boldsymbol{a}_m)) \quad , \tag{5.12}$$

and at each iteration m the final approximated function is updated as

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \gamma_m h(\boldsymbol{x}, \boldsymbol{a}_m) \quad . \tag{5.13}$$

The pseudo-code for Gradient Boosting Machines is depicted in Algorithm 2.

Algorithm 2 Gradient Boosting Machines Pseudo-Code.

```
Input: $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N} \equiv \mathcal{D}$: Training dataset; $M$: Number of Iterations; $\mathcal{L}$: Evaluation Loss function.
Output: $F_M = F_0 + \sum_{m=1}^{M} \gamma_m h_m$.
1: Initialize $F_0 = \arg\min_{\gamma} \mathcal{L}(\gamma, y_i)$
2: $m \leftarrow 1$
3: while $m \leq M$ do
4:     Calculate the pseudo-residuals: $r_{i,m} = -\left[\frac{\partial \mathcal{L}}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x}_i)=F_{m-1}(\boldsymbol{x}_i)}$
5:     Fit a weak learner providing $\{\boldsymbol{x}_i, r_{i,m}\}$ and get back the hypothesis: $h_m$
6:     Compute $\gamma_m = \arg\min_{\gamma} \mathcal{L}(y_i, F_{m-1}(\boldsymbol{x}_i) + \gamma h_m(\boldsymbol{x}_i))$
7:     Update the regressor $F_m = F_{m-1} + \gamma_m h_m$
8: end while
9: return $F_M = F_0 + \sum_{m=1}^{M} \gamma_m h_m$
```

From the pseudo-code presented in Algorithm 2, the mechanism behind the construction of the best-approximated function is better understood – Start with an initial solution (e.g. if $\mathcal{L} = MSE$, the mean of the target variable, $\bar{y}$) and to this, additively add the optimal residuals found at each iteration m.

5.1.1 Gradient Tree Boosting

In the same paper, it was proposed the algorithm known nowadays as Gradient Boosting Machines Regression Trees (GBRT). Here, the weak-learner is a regression tree (usually a CART [42]). In GBRT, instead of considering a "global" coupling weight γ_m for each iteration, consider that each j-terminal node in a regression tree is a weak-learner. Defining $\{R_{m,j}\}_{j=1}^{J}$ all the (disjoint) regions that collectively cover the space of all joint values of the predictor variables $\boldsymbol{x}$, for each iteration m, a given weak-learner is given by

$$h_m = \sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) \quad . \tag{5.14}$$

Given this, the line search Eq. (5.12) becomes

$$\{\gamma_m^*\}_{j=1}^{J} = \underset{\{\gamma_m\}_{j=1}^{J}}{\arg\min} \sum_{i=1}^{N} \mathcal{L}(y_i, F_{m-1}(\boldsymbol{x}_i) + \sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j})) \quad , \tag{5.15}$$

which, by the disjoint nature of the leaves produced by the regression trees, it becomes

$$\gamma_{m,j} = \underset{\gamma}{\arg\min} \sum_{\boldsymbol{x}_j \in R_{m,j}} \mathcal{L}(y_j, F_{m-1}(\boldsymbol{x}_j) + \gamma) \quad , \tag{5.16}$$

for a given leaf j.

5.1.2 Optimization Techniques

Here a brief review of techniques in Boosting Gradient Machines to avoid overfitting will be provided.

5.1.2.1 Shrinking

In [43], it was also proposed the introduction of a hyper-parameter in line 7 of the Gradient Boosting Algorithm 2. The introduction of this parameter changes the update rule such that

$$F_m = F_{m-1} + \eta\gamma_m h_m \quad , \tag{5.17}$$

this hyper-parameter, η, is the so-called learning rate and it is usually constrained to the interval $0 < \eta < 1$. Due to this constrain, the effect of the learning rate is to penalize every weak-learner prediction. This penalization translates into a slower convergence. As this convergence is slower, usually there is the need to take into consideration a higher number of boosters M and hence there is a trade-off between these two parameters – the lower the learning rate, the higher should be the number of boosters needed to produce convergence. Although one of the caveats of the introduction of this hyper-parameter is the addition in the algorithm's time complexity cost, due to the need of increasing M, it has a high impact on the predictive power of the model when assessing predictions.

5.1.2.2 Subsampling

Another method to avoid overfitting in Gradient Boosting Machines was proposed in [44] and its implementation on Gradient Boosting gave rise to the Stochastic Gradient Boosting algorithm. This method, called subsampling, consists of introducing randomness in the fitting procedure. Instead of considering all the training data to fit a weak-learner, a sample is drawn (in the original paper without replacement, although replacement is also possible) from the data. Denoting N as the size of our training set and $\tilde{N}$ the size of our sample, the so-called bag-fraction b is defined as

$$b = \tilde{N}/N \quad . \tag{5.18}$$

The use of this hyper-parameter needs to be used with care as it depends on the number of instances of our data – choosing a low bag-fraction on a training set with only a few instances can cause the model to underfit the data.

5.2 *SERA* implementation in Gradient Boosting Machines

Given the review on Gradient Boosting Machines, the necessary conditions to embed $SERA$ as a loss function in Gradient Boosting will now be established. Note that both exact (cf. Equation (3.6)) and numerical (cf. Equation (3.11)) solutions will be presented below. The full derivation will be performed for the exact solutions, while the numerical ones will only be enunciated.

Due to the initialization of Gradient Boosting Machines, the first step is to determine the constant that minimizes $SERA$. This constant will correspond to the first "booster", $F_0(\boldsymbol{x}) = F_0$.

Recalling the definition of $SERA$ from Equation (3.6)

$$SERA = \int_0^1 SER_t \; dt = \int_0^1 \sum_{y_i \in \mathcal{D}^t} (\hat{y}_i - y_i)^2 \; dt \quad , \tag{5.19}$$

its derivative with respect to F_0 gives

$$\frac{\partial SERA}{\partial F_0} = \frac{\partial}{\partial F_0} \int_0^1 \sum_{y_i \in \mathcal{D}^t} (F_0 - y_i)^2 \; dt \quad , \tag{5.20}$$

setting the derivative to 0 and solving for F_0

$$F_0 = \frac{\int_0^1 \sum_{i \in \mathcal{D}^t} y_i \; dt}{\int_0^1 |\mathcal{D}^t| \; dt} \approx \frac{\sum_{i=1}^N \sigma_i y_i}{\sum_{i=1}^N \sigma_i} \quad . \tag{5.21}$$

Now the pseudo-residuals. Recalling from Equation (5.7)

$$r_m(\boldsymbol{x}) = - \left[\frac{\partial \mathcal{L}(y, F(\boldsymbol{x}))}{\partial F(\boldsymbol{x})} \right]_{F(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x})} \quad , \tag{5.22}$$

for a given instance i

$$\begin{aligned} r_{i,m} &= -\frac{\partial SERA}{\partial F(\boldsymbol{x}_i)}\bigg|_{F(x_i)=F_{m-1}(\boldsymbol{x}_i)} \\ &= -2\int_0^1 \sum_{y_i \in \mathcal{D}^t} (F_{m-1}(\boldsymbol{x}_i) - y_i)\ dt \\ &\approx -2\sigma_i(F_{m-1}(\boldsymbol{x}_i) - y_i) \quad . \end{aligned} \tag{5.23}$$

From now on, to ease the notation, $F_m(\boldsymbol{x}_i) \equiv F_{i,m}$ and $h_m(\boldsymbol{x}_i) \equiv h_{i,m}$. The, minimization of γ_m found in Equation (5.12) yields

$$\begin{aligned} \frac{\partial SERA}{\partial \gamma_m} &= \frac{\partial}{\partial \gamma_m}\int_0^1 \sum_{y_i \in \mathcal{D}^t} (F_{i,m-1} + \gamma_m h_{i,m} - y_i)^2\ dt \\ &= 2\int_0^1 \sum_{y_i \in \mathcal{D}^t} h_{i,m}(F_{i,m-1} + \gamma_m h_{i,m} - y_i)\ dt \quad . \end{aligned} \tag{5.24}$$

Setting the derivative to 0

$$\begin{aligned} \frac{\partial SERA}{\partial \gamma_m} = 0 &\Leftrightarrow 2\int_0^1 \sum_{y_i \in \mathcal{D}^t} h_{i,m}(F_{i,m-1} + \gamma_m h_{i,m} - y_i)\ dt = 0 \\ &\Leftrightarrow \int_0^1 \sum_{y_i \in \mathcal{D}^t} h_{i,m}(y_i - F_{i,m-1})\ dt = \gamma_m \int_0^1 \sum_{y_i \in \mathcal{D}^t} h_{i,m}^2 dt \quad , \end{aligned} \tag{5.25}$$

solving for γ_m

$$\gamma_m = \frac{\int_0^1 \sum_{y_i \in \mathcal{D}^t} h_{i,m}(y_i - F_{i,m-1})\ dt}{\int_0^1 \sum_{y_i \in \mathcal{D}^t} h_{i,m}^2\ dt} \approx \frac{\sum_{i=1}^N \sigma_i h_{i,m}(y_i - F_{i,m-1})}{\sum_{i=1}^N \sigma_i h_{i,m}^2} \quad . \tag{5.26}$$

The determination of γ_m in Equation (5.26) can be used for any generic booster $h_m(\boldsymbol{x})$. If, however, the booster is a Regression Tree, recalling Equation (5.16)

$$\gamma_{m,j} = \arg\min_{\gamma} \sum_{\boldsymbol{x}_j \in R_{m,j}} \mathcal{L}(y_j, F_{m-1}(\boldsymbol{x}_j) + \gamma) \quad , \tag{5.27}$$

for a given terminal node k, and replacing the weak-learner from Equation (5.14), the derivative of $SERA$ w.r.t. to $\gamma_{m,k}$ implies that

$$\begin{aligned}
\frac{\partial SERA}{\partial \gamma_{m,k}} &= \frac{\partial}{\partial \gamma_{m,k}} \int_0^1 \sum_{y_i \in \mathcal{D}^t} \left(F_{i,m-1} + \sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) - y_i \right)^2 dt \\
&= 2 \int_0^1 \sum_{y_i \in \mathcal{D}^t} (F_{i,m-1} + \sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) - y_i) \frac{\partial}{\partial \gamma_{m,k}} \left(\sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) \right) dt \\
&= 2 \int_0^1 \sum_{y_i \in \mathcal{D}^t} (F_{i,m-1} + \sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) - y_i) \left(\sum_{j=1}^{J} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) \delta_{j,k} \right) dt \\
&= 2 \int_0^1 \sum_{y_i \in \mathcal{D}^t} (F_{i,m-1} + \sum_{j=1}^{J} \gamma_{m,j} \mathbf{1}(\boldsymbol{x} \in R_{m,j}) - y_i) \times \mathbf{1}(\boldsymbol{x} \in R_{m,k}) \; dt \\
&= 2 \int_0^1 \sum_{\substack{\boldsymbol{x}_i \in R_{m,k} \\ y_i \in \mathcal{D}^t}} (F_{i,m-1} + \gamma_{m,k} - y_i) \; dt \quad .
\end{aligned} \tag{5.28}$$

Setting the derivative to 0

$$\begin{aligned}
\frac{\partial SERA}{\partial \gamma_{m,k}} = 0 \Leftrightarrow & \\
& \Leftrightarrow 2 \int_0^1 \sum_{\substack{\boldsymbol{x}_i \in R_{m,k} \\ y_i \in \mathcal{D}^t}} (F_{i,m-1} + \gamma_{m,k} - y_i) \; dt = 0 \\
& \Leftrightarrow \int_0^1 \sum_{\substack{\boldsymbol{x}_i \in R_{m,k} \\ y_i \in \mathcal{D}^t}} (y_i - F_{i,m-1}) \; dt = \gamma_{m,k} \int_0^1 \sum_{\substack{\boldsymbol{x}_i \in R_{m,k} \\ y_i \in \mathcal{D}^t}} dt \quad .
\end{aligned} \tag{5.29}$$

Solving for $\gamma_{m,k}$

$$\begin{aligned}
\gamma_{m,k} &= \frac{\int_0^1 \sum_{\substack{\boldsymbol{x}_i \in R_{m,k} \\ y_i \in \mathcal{D}^t}} (y_i - F_{i,m-1}) \; dt}{\int_0^1 |\mathcal{D}|_{\substack{\boldsymbol{x}_i \in R_{m,k} \\ y_i \in \mathcal{D}^t}} \; dt} \\
&\approx \frac{\sum_{\boldsymbol{x}_i \in R_{m,k}} \sigma_i (y_i - F_{i,m-1})}{\sum_{\boldsymbol{x}_i \in R_{m,k}} \sigma_i} \quad .
\end{aligned} \tag{5.30}$$

All these derivations are the weighted versions of the same derivations under SSE.

5.3 Case study

To give a more detailed description of the impact of using $SERA$ as a loss function in Gradient Boosting Machines, a case study is performed. Here, it will be shown that different values of T may lead to a trade-off between generalization and extreme values optimization. The main reason lies on the dependence of residuals on σ (cf. Algorithm 1).

The considered data set was already considered on previous studies (e.g. [6]) and measures the logarithmic concentration of NO_2 in the air. Its estimated probability density function is depicted in Figure 5.1.

This data set is interesting for the case study as there is domain knowledge of the target variable, where a concentration above $\log(150\mu g/m^3) \approx 5$ is considered hazard to a human being (i.e., it is an extreme value). As such, the following set of control points are considered: $S = \{(1.1, 0, 0), (3.7, 0, 0), (5, 1, 0)\}$, where the first and second set denote low and annual mean concentrations, respectively.

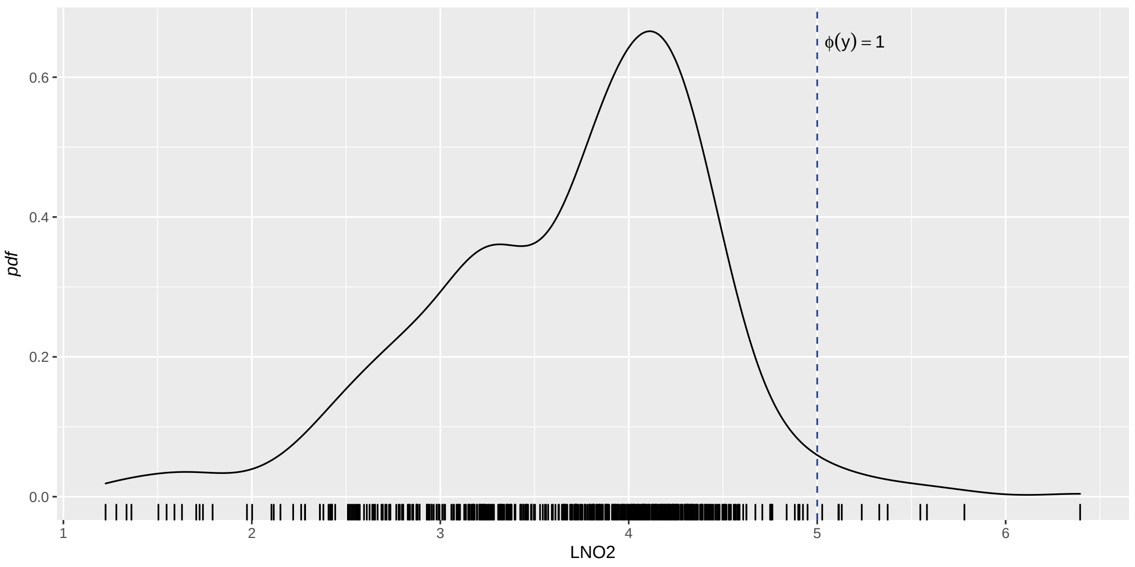

Figure 5.1: Logarithmic NO_2 density function. Values above $LNO_2 = 5$ are considered hazardous.

The independent variables are related to meteorological conditions and road traffic. The data set consists of 500 instances and an initial random partition of 80/20% is performed.

The study consists of comparing instances obtained during the training phase which have a high and low relevance value for three Gradient Tree Boosting variants:

V1: $\mathcal{L} = MSE$, with $F_0 = \bar{y}$;

V2: $\mathcal{L} = SERA$, with $F_0 =$ Equation (5.21);

V3: $\mathcal{L} = SERA$, with $F_0 = \bar{y}$.

The hyper-parameters considered are fixed for the three variants, which are the learning rate $\eta = 10^{-2}$, and the number of boosters $max\_iter = 1000$. As training is performed, the predictions and pseudo-residuals (cf. Equation (5.7)) are collected from each iteration.

The first case study compares the three variants for an extreme value, $y = 5.37$. Its evolution during the training phase is depicted in Figure 5.2. As it is shown, variants that use $SERA$ as an optimization loss function (**V2**, **V3**), regardless of their initialization, give a better approximation of the true value than the one optimized with MSE (**V1**). Also, it is interesting to point out that, although the third variant initializes at the mean of the target variable, stability is achieved at approximately the same number of iterations as the second variant, which uses a weighted mean based on σ (cf. Equation (5.21)). Comparing the stability from these variants (which is achieved at approximately 500 iterations) with the one from the first variant (which is achieved at approximately 200 iterations), there is a clear trade-off between the learning capability and the number of boosters needed to achieve stability – a more accurate prediction of an extreme value comes at the cost of considering more boosters until the learning phase is stabilized.

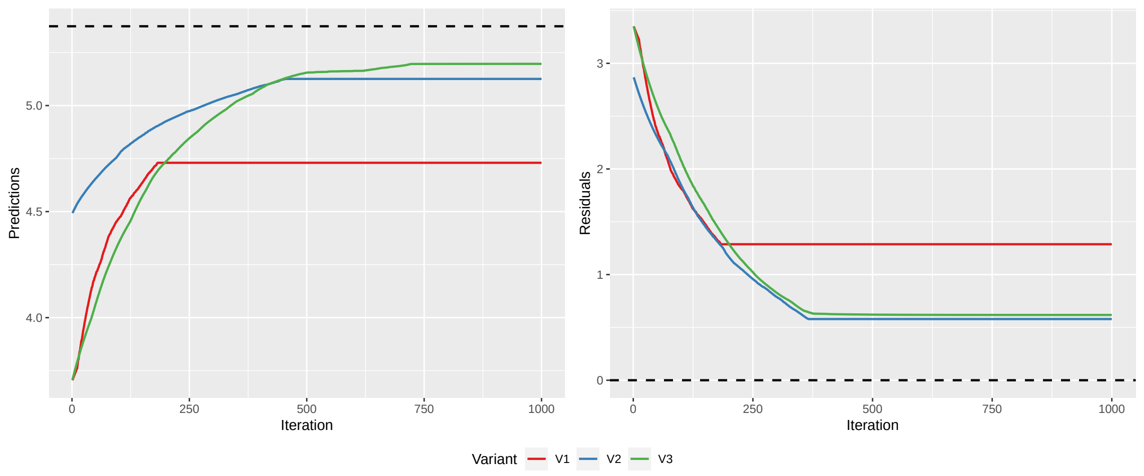

FIGURE 5.2: **Left:** Predictions as a function of the iteration for an extreme value. The black dashed line denotes the ground truth ($y = 5.37$). **Right:** Residuals as a function of the iteration. The black line denotes the optimal residual ($r = 0$).

The second case study compares these three variants for a common value $y = 3.35$. Their evolution across training phases is depicted in Figure 5.3.

As it is shown, the second and third variants could not properly provide a good prediction for a low relevance value (although variant **V2** was able to slowly converge until it stopped around the same value as variant **V3**, however, the latter diverged from its initial "guess"). The reason seems to come from the residuals dependence on σ (cf. Equation

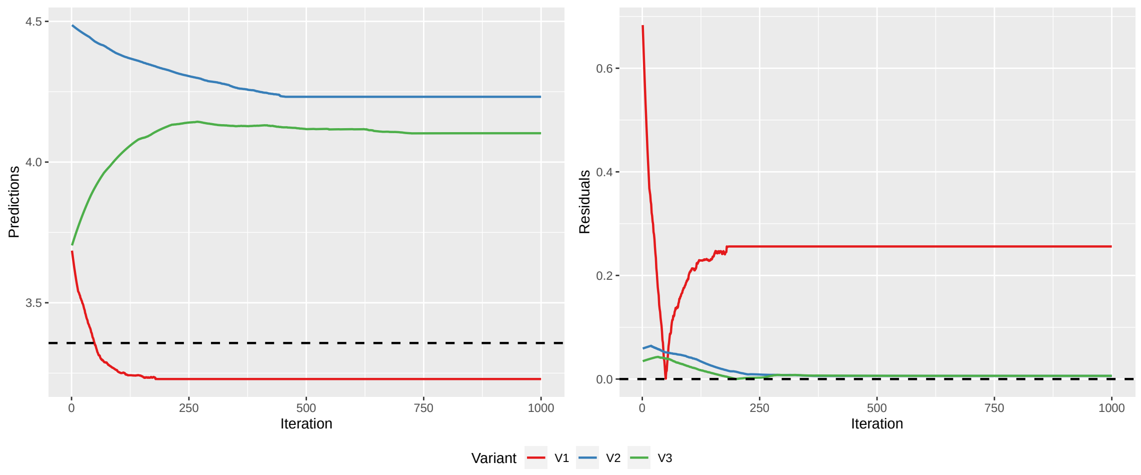

Figure 5.3: **Left:** Predictions as a function of the iteration for a common value. The black dashed line denotes the ground truth ($y = 3.35$). **Right:** Residuals as a function of the iteration. The black dashed line denotes the optimal residual ($r = 0$).

(5.23)). In this case, the residual of this instance is always close to 0 across all iterations for variants **V2**, **V3** and its effect can be seen in the right plot of Figure 5.3. As it was pointed out in Section 3.3.1, there is an equivalence between $SERA$ and the weighted SSE. These weights arise naturally from the construction of $SERA$ and depend on two values: 1) the number of times a given instance contributes to $SERA$, which is counted by comparing its relevance value with each step taken to discretize the relevance space; 2) the total number of intervals T taken into consideration. Recalling that, for a point of no relevance (cf. Algorithm 1)

$$\sigma = \frac{1}{2T} \quad , \tag{5.31}$$

for $T = 1000$, $\sigma = 0.5 \times 10^{-3}$. Comparing the scale of σ with the scale of the target variable from the case study (which goes from $[1, 6]$, approximately, Figure 5.1), σ will dominate over the target variable when computing the residuals for low relevance values. As such, residuals won't be representative of the actual error committed at a given iteration, as it will always be practically negligible. As an example, suppose that the third variant is being considered, where $F_0 = \bar{y} \sim 3.7$, and the residual at the first iteration is being evaluated. Considering two instances, one where $y_1 = 1.1$ and another where y_2 is close as to the mean as possible. The "unweighted" residuals for both instances, will be, in absolute and respectively, $r_1 = 2.6$ and $r_2 \sim 0$. Coupling the weights, while r_2 remains unchanged, $r_1 \approx 0$ as well. This may imply under-fitting during tree construction since the partition space may be over-simplified as points of no relevance are the most common ones. As such, a scaling problem is faced and care must be taken when there is a discrepancy between the number of intervals T and the domain of the target variable. Although these conclusions are being built upon Gradient Tree Boosting machines, other learning algorithms that rely on optimization methods such as steepest/gradient descent, where residuals are evaluated, should posit the same conclusions.

Given this, a second scenario is considered where instead of using the default value $T = 1000$, T has now the same scale as the target variable, $T = 10$. Thus, for a common value, the weight attributed to an instance will now be $\sigma = 0.5 \times 10^{-1}$, relaxing the severe penalization faced when $T = 1000$. Results are presented in Figure 5.4.

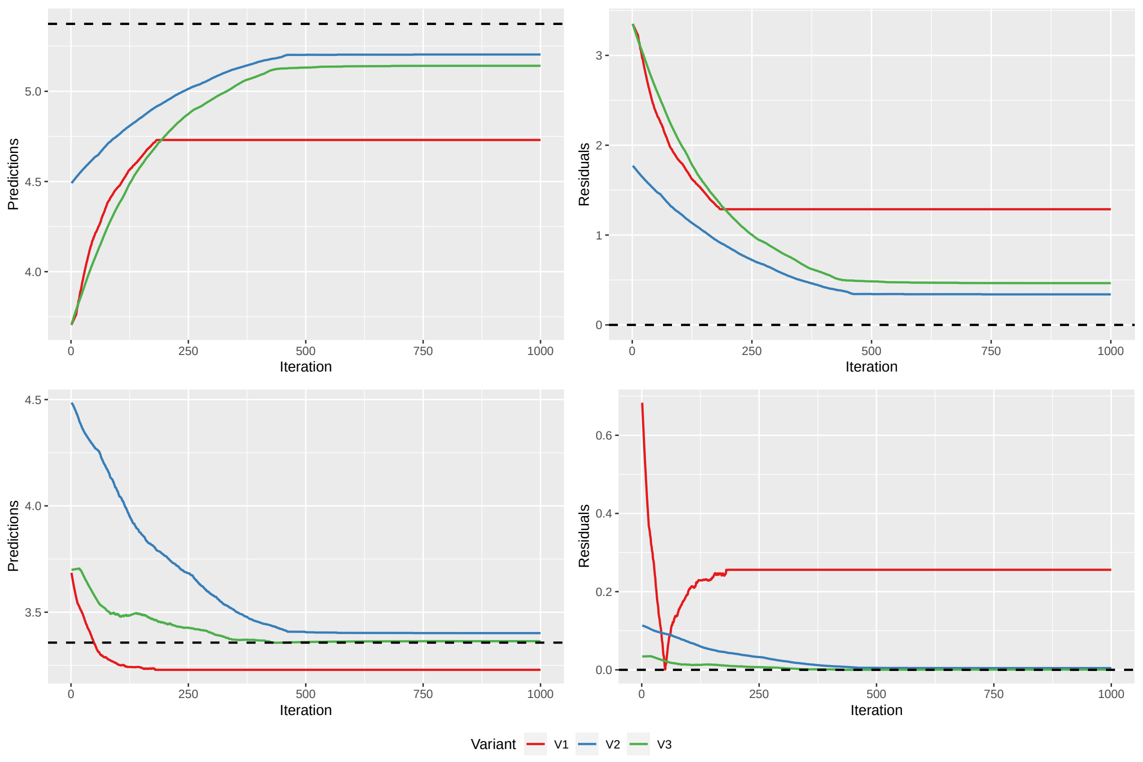

FIGURE 5.4: **Top:** Predictions and residuals as a function of the iteration for the same extreme value ($y = 5.37$, dashed-line). Predictions did not vary after setting $T = 10$. **Bottom:** Predictions and residuals as a function of the iteration for the same point of no relevance $y = 3.35$. Note that in this case predictions converged in the direction of the true value.

As it is depicted, **V2** and **V3** are now able to provide a better approximation of the instance $y = 3.35$, while maintaining the approximation of the extreme value $y = 5.37$ close to its true value (as expected, since regardless of T, $\sigma = 1$ for extreme values). To conclude this study, predictions are assessed for the two different values of T. Results from these evaluations are presented in Table 5.1.

	MSE		*SERA*	
Variants	$T = 1000$	$T = 10$	$T = 1000$	$T = 10$
V1	0.22	0.22	8.28	8.28
V2	0.84	0.24	5.52	5.67
V3	0.75	0.25	5.69	6.84

Table 5.1: Errors obtained during inference phase. *SERA* was evaluated using the default setting, $T = 1000$.

As it is shown, as T is reduced to conform with the domain of the target variable, a better generalization is achieved (e.g. for **V2**, *MSE* drops from 0.84 to 0.24), however, at the cost of a worst predictive power for extreme values (e.g. for **V3**, *SERA* raises from 5.69 to 6.84).

5.3.1 Discussion

In the previous section, it was shown that different T intervals lead to different results, both in optimization and inferring phases. The reason behind this is that the weight assigned to each instance may posit an unrepresentativeness of the actual error committed of a given prediction. So what is the cost of considering different intervals when optimizing a given model under *SERA*? Turning the attention to the example from the case study, initially, it was considered $T = 1000$. Results showed that under this setting extreme values were "prioritized" as residuals for common values were close to 0. When lowering the number of intervals to $T = 10$, results showed that a better generalization was achieved at the cost of lowering the predictive power of extreme values. In addition, when lowering the number of intervals, *SERA* may deviate from its true value because the number of trapezoids taken into consideration will be smaller, losing the notion of continuity.

Chapter 6

Conclusions

In this thesis, it was addressed the problem of embedding $SERA$ as a loss function in supervised learning algorithms to improve the predictive power for both extreme and normal values. As such, it was used as baseline two variants of Gradient Boosting Machines, XGBoost, and LGBM. This evaluation was done by comparing models optimized with MSE. Results showed that the embedment of $SERA$ provides the ability to reduce errors for both extreme and normal values. In addition, when underperformed by standard models, models optimized with $SERA$ were still able to provide a more accurate prediction in a high relevance domain.

6.1 Contributions

Some contributions were made in this thesis. They include:

- a brief review on recent research in Imbalance Regression (Chapter 2);
- the proof of equivalence between $SERA$ and the weighted SSE (Section 3.3.1);
- the tools needed to embed $SERA$ as a loss function in Gradient Boosting Machines models with an extensive experimental study that shows the ability of $SERA$ to improve the predictive focus of both common and extreme values (Chapter 4);
- a study where T, the number of intervals taken to discretize $SERA$ under the trapezoidal rule, should be a parameter that may require some fine-tuning, or a method to automatically propose an appropriate value (Section 5.3);

6.2 Future Work

Future directions of this work should include:

- find an appropriate approach to find the optimal number of intervals T used to discretize $SERA$ for a given data set – a straightforward solution comes from considering T such that it has the same order of magnitude as the mean of the target variable;
- as mentioned in Chapter 2 and 3, recent research has been made at the algorithmic level, namely in the realm of Deep Learning; future work should focus on the assessment of the predictive focus of $SERA$ in this field, where high-dimensional data is used; another aspect that it was not studied regards the use of $SERA$ on multi-target regression problems, which should also be taken into consideration;
- finally, compare the predictive power of $SERA$ with other recently proposed loss functions in the context of Imbalanced Regression (e.g. Balanced MSE).

Appendix A

Bayes Sign Test

The Bayes Sign Test aims at solving the problem of comparing two models on multiple data sets by determining the posterior probability distribution of a variable z that encompasses information relative to the two models. This is done by assuming a Dirichlet Process over the prior probability distribution of z composed by $\boldsymbol{z} = \{z_1, z_2, ..., z_d\}$ independent and identically distributed (i.i.d.) observations of z, where each observation represents a score over a given dataset in $\boldsymbol{D}$. In the methodology presented in this thesis, z_i will be the mean difference of each k-fold score between a workflow $\mathcal{W}_1$ and $\mathcal{W}_2$, normalised by $\mathcal{W}_2$, i.e., if $\mathcal{L}$ is our evaluation metric

$$z_i = \frac{1}{10} \sum_{k=1}^{10} \frac{\mathcal{L}_k(\mathcal{W}_1) - \mathcal{L}_k(\mathcal{W}_2)}{\mathcal{L}_k(\mathcal{W}_2)} \quad . \tag{A.1}$$

The reason why it is considered this normalization is explained below. To determine the posterior of z, $p(z)$, the procedure from [39] is followed. In their work, the authors considered the prior mean G_0 as a delta Dirac's centered at each z_i. The posterior was constructed as a linear combination of delta Dirac's distributions. Given the posterior $p(z)$, it is divided in three intervals in the probability density space: $(-\infty, -R]$, $[-R, R]$, $[R, \infty)$. Here, R is the parameter that defines the Region of Practical Equivalence (ROPE) and it is considered as a region in the probability density space where the two classifiers are practically equivalent [40]. Thus, given two workflows $(\mathcal{W}_1, \mathcal{W}_2)$, the following posterior probabilities are defined:

- $P(z < -R)$, the posterior probability that the mean difference between the scores of workflows $\mathcal{W}_1$ and $\mathcal{W}_2$ is practically negative;

- $P(|z| \leq R)$, the posterior probability that the mean difference between the scores of workflows $\mathcal{W}_1$ and $\mathcal{W}_2$ is practically equivalent;

- $P(z > R)$, the posterior probability that the mean difference between the scores of workflows $\mathcal{W}_1$ and $\mathcal{W}_2$ is practically positive;

As an illustrative example, suppose that two workflows $\mathcal{W}_1$ and $\mathcal{W}_2$ are being evaluated and the following scores are obtained: $P(z < -R) = 0.9,\ P(|z| \leq R) = 0.1,\ P(z > R) = 0$, then, the following conclusion can be drawn – the posterior probability of the mean difference between them is practically negative, i.e., $\mathcal{W}_1$ is practically better than $\mathcal{W}_2$. From now on, it is adopted this nomenclature, reminding that in the end, these conclusions are always inferred. Finally, an adequate value to ROPE must be provided. For classification problems, it is usually used 0.01, while for regression problems, 0.1. Here, it is maintained $R = 0.01$, as it is more restricted, however, due to the high variance of errors for different data sets in the context of regression, normalize each difference is normalized according to Equation (A.1).

Appendix B

Tables of Results

In this section, we report the MSE and $SERA$ results obtained in out-of-sample of each data set.

Table B.1: MSE results in out-of-sample, with the best models per data set in bold. Model (#wins): LGBM^M (20), XGBoost^M (15), XGBoost^S (1), LGBM^S (0).

id	XGBoost^S	XGBoost^M	LGBM^S	LGBM^M
1	3.00e-01	**8.54e-01**	1.47e-01	9.53e-01
2	2.14e-01	2.50e-01	3.30e-01	**2.31e-01**
3	**1.33e+03**	1.74e+03	6.63e+02	1.83e+03
4	4.25e+06	7.87e+06	3.98e+06	**2.97e+06**
5	9.79e+06	**1.19e+07**	1.07e+07	1.37e+07
6	8.63e+02	**6.47e+02**	8.16e+02	9.51e+02
7	2.80e+03	3.96e+03	1.31e+04	**2.63e+03**
8	8.76e+02	9.11e+02	8.65e+02	**8.10e+02**
9	4.76e+07	5.43e+07	4.15e+07	**5.45e+07**
10	4.66e-01	1.22e-01	1.37e-01	**1.46e-01**
11	1.06e+00	6.76e-01	6.22e-01	**5.94e-01**
12	2.52e+01	3.24e+01	2.23e+01	**2.99e+01**
13	1.75e+02	**1.90e+02**	1.47e+02	1.94e+02
14	4.78e+01	**4.35e+01**	5.66e+01	5.09e+01
15	1.55e+01	**1.46e+01**	4.02e+01	2.06e+01
16	4.76e+03	**6.72e+03**	6.28e+03	6.54e+03
17	3.28e+03	**3.09e+03**	4.23e+03	1.27e+04
18	6.90e-01	**9.20e-01**	5.83e-01	9.17e-01
19	2.29e+00	2.30e+00	2.19e+00	**2.16e+00**
20	4.82e+02	6.05e+02	4.86e+02	**5.96e+02**
21	1.94e+03	2.26e+03	1.94e+03	**2.20e+03**
22	3.12e+01	**1.52e+02**	3.31e+01	1.62e+02
23	1.18e-05	1.65e-05	1.09e-05	**1.42e-05**
24	3.08e+02	**3.48e+02**	5.61e+02	4.70e+02
25	1.02e+03	**1.02e+03**	1.05e+03	1.08e+03
26	9.12e-01	1.12e+00	8.95e-01	**1.08e+00**
27	2.91e+01	3.82e+01	2.96e+01	**3.69e+01**
28	2.26e-02	2.82e-02	2.16e-02	**2.50e-02**
29	5.82e-04	1.13e-03	5.77e-04	**1.13e-03**
30	2.57e+00	2.93e+00	2.55e+00	**2.68e+00**
31	2.70e+05	**2.23e+05**	3.67e+05	3.48e+05
32	6.32e-03	**6.20e-03**	5.94e-03	6.51e-03
33	2.57e+11	8.38e+11	2.57e+11	**1.01e+12**
34	2.77e+12	3.25e+12	2.71e+12	**3.11e+12**
35	2.16e+12	2.33e+12	2.12e+12	**2.26e+12**
36	7.57e+11	7.81e+11	7.32e+11	**7.40e+11**

Table B.2: *SERA* results in out-of-sample, with the best models per data set in bold. Model (#wins): LGBM^S (16), XGBoost^S (8), XGBoost^M (7), LGBM^M (5).

id	XGBoost^S	XGBoost^M	LGBM^S	LGBM^M
1	3.00e-01	8.54e-01	**1.47e-01**	9.53e-01
2	**2.14e-01**	2.50e-01	3.30e-01	2.31e-01
3	1.33e+03	1.74e+03	**6.63e+02**	1.83e+03
4	4.25e+06	7.87e+06	3.98e+06	**2.97e+06**
5	**9.79e+06**	1.19e+07	1.07e+07	1.37e+07
6	8.63e+02	**6.47e+02**	8.16e+02	9.51e+02
7	2.80e+03	3.96e+03	1.31e+04	**2.63e+03**
8	8.76e+02	9.11e+02	8.65e+02	**8.10e+02**
9	4.76e+07	5.43e+07	**4.15e+07**	5.45e+07
10	4.66e-01	**1.22e-01**	1.37e-01	1.46e-01
11	1.06e+00	6.76e-01	6.22e-01	**5.94e-01**
12	2.52e+01	3.24e+01	**2.23e+01**	2.99e+01
13	1.75e+02	1.90e+02	**1.47e+02**	1.94e+02
14	4.78e+01	**4.35e+01**	5.66e+01	5.09e+01
15	1.55e+01	**1.46e+01**	4.02e+01	2.06e+01
16	**4.76e+03**	6.72e+03	6.28e+03	6.54e+03
17	3.28e+03	**3.09e+03**	4.23e+03	1.27e+04
18	6.90e-01	9.20e-01	**5.83e-01**	9.17e-01
19	2.29e+00	2.30e+00	2.19e+00	**2.16e+00**
20	**4.82e+02**	6.05e+02	4.86e+02	5.96e+02
21	**1.94e+03**	2.26e+03	1.94e+03	2.20e+03
22	**3.12e+01**	1.52e+02	3.31e+01	1.62e+02
23	1.18e-05	1.65e-05	**1.09e-05**	1.42e-05
24	**3.08e+02**	3.48e+02	5.61e+02	4.70e+02
25	1.02e+03	**1.02e+03**	1.05e+03	1.08e+03
26	9.12e-01	1.12e+00	**8.95e-01**	1.08e+00
27	2.91e+01	**3.82e+01**	2.96e+01	3.69e+01
28	2.26e-02	2.82e-02	**2.16e-02**	2.50e-02
29	5.82e-04	1.13e-03	**5.77e-04**	1.13e-03
30	2.57e+00	2.93e+00	**2.55e+00**	2.68e+00
31	2.70e+05	**2.23e+05**	3.67e+05	3.48e+05
32	6.32e-03	6.20e-03	**5.94e-03**	6.51e-03
33	2.57e+11	8.38e+11	**2.57e+11**	1.01e+12
34	2.77e+12	3.25e+12	**2.71e+12**	3.11e+12
35	2.16e+12	2.33e+12	**2.12e+12**	2.26e+12
36	7.57e+11	7.81e+11	**7.32e+11**	7.40e+11

Bibliography

[1] D. Rengasamy, B. Rothwell, and G. Figueredo, “Asymmetric loss functions for deep learning early predictions of remaining useful life in aerospace gas turbine engines,” in *2020 International Joint Conference on Neural Networks (IJCNN)*, 04 2020. [Cited on pages 1 and 13.]

[2] R. Ribeiro, “Utility-based regression,” Ph.D. dissertation, Faculty of Sciences - University of Porto, 2011. [Cited on pages 1, 18, 19, and 20.]

[3] Y. Yang, K. Zha, Y. Chen, H. Wang, and D. Katabi, “Delving into deep imbalanced regression,” *CoRR*, vol. abs/2102.09554, 2021. [Online]. Available: https://arxiv.org/abs/2102.09554 [Cited on pages 1 and 9.]

[4] N. Moniz, “Prediction and ranking of highly popular web content,” Ph.D. dissertation, Faculty of Sciences - University of Porto, 2017. [Cited on pages 1 and 20.]

[5] L. Torgo and R. Ribeiro, “Utility-based regression,” in *Knowledge Discovery in Databases: PKDD 2007*. Springer Berlin Heidelberg, 2007, pp. 597–604. [Cited on pages 2, 6, and 20.]

[6] R. Ribeiro and N. Moniz, “Imbalanced regression and extreme value prediction,” *Machine Learning*, vol. 109, pp. 1–33, 2020. [Cited on pages 2, 19, 20, 22, 28, and 43.]

[7] H. R. Varian, “A bayesian approach to real estate assessment,” *Studies in Bayesian econometric and statistics in Honor of Leonard J. Savage*, pp. 195–208, 1975. [Cited on pages 2 and 12.]

[8] C. W. J. Granger, “Prediction with a generalized cost of error function,” *OR*, vol. 20, no. 2, pp. 199–207, 1969. [Online]. Available: http://www.jstor.org/stable/3008559 [Cited on page 13.]

[9] C. Granger, “Outline of forecast theory using generalized cost functions,” *Spanish Economic Review*, vol. 1, no. 2, pp. 161–173, 1999. [Cited on page 2.]

[10] L. Ehrig, D. Atzberger, B. Hagedorn, J. Klimke, and J. Döllner, “Customizable asymmetric loss functions for machine learning-based predictive maintenance,” in *2020 8th International Conference on Condition Monitoring and Diagnosis (CMD)*, 2020, pp. 250–253. [Cited on page 2.]

[11] P. Branco, L. Torgo, and R. P. Ribeiro, “A survey of predictive modeling on imbalanced domains,” *ACM Comput. Surv.*, vol. 49, no. 2, 2016. [Cited on page 5.]

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: Synthetic minority over-sampling technique,” *J. Artif. Int. Res.*, vol. 16, no. 1, p. 321–357, 2002. [Cited on page 6.]

[13] L. Torgo, R. P. Ribeiro, B. Pfahringer, and P. Branco, “Smote for regression,” in *Progress in Artificial Intelligence.* Springer, 2013, pp. 378–389. [Cited on pages 6 and 20.]

[14] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, “Cost-sensitive boosting for classification of imbalanced data,” *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007. [Cited on page 8.]

[15] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, “Adacost: Misclassification cost-sensitive boosting,” in *16th International Conference on Machine Learning*, ser. ICML ’99. Morgan Kaufmann Publishers Inc., 1999, p. 97–105.

[16] C. Elkan, “The foundations of cost-sensitive learning,” *17th International Conference on Artificial Intelligence*, vol. 1, pp. 973–978, 2001. [Cited on page 8.]

[17] M. Steininger, K. Kobs, P. Davidson, A. Krause, and A. Hotho, “Density-based weighting for imbalanced regression,” *Machine Learning*, vol. 110, no. 8, pp. 2187–2211, 2021. [Online]. Available: https://doi.org/10.1007/s10994-021-06023-5 [Cited on page 8.]

[18] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, “Smoteboost: Improving prediction of the minority class in boosting,” in *Knowledge Discovery in Databases: PKDD 2003.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 107–119. [Cited on page 8.]

[19] C. Chen and L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, 01 2004.

[20] A. Estabrooks and N. Japkowicz, "A mixture-of-experts framework for learning from imbalanced data sets," in *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, ser. IDA '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 34–43. [Cited on page 8.]

[21] N. Moniz, R. Ribeiro, V. Cerqueira, and N. Chawla, "Smoteboost for regression: Improving the prediction of extreme values," in *IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 150–159. [Cited on page 8.]

[22] H. Drucker, "Improving regressors using boosting techniques," *Proceedings of the 14th International Conference on Machine Learning*, 08 1997. [Cited on page 8.]

[23] D. Solomatine and D. Shrestha, "Adaboost.rt: a boosting algorithm for regression problems," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, vol. 2, 2004, pp. 1163–1168 vol.2. [Cited on page 9.]

[24] P. Kankanala, S. Das, and A. Pahwa, "Adaboost(+): An ensemble learning approach for estimating weather-related outages in distribution systems," *Power Systems, IEEE Transactions on*, vol. 29, pp. 359–367, 01 2014. [Cited on page 8.]

[25] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001. [Cited on page 10.]

[26] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73 – 101, 1964. [Online]. Available: https://doi.org/10.1214/aoms/1177703732 [Cited on page 13.]

[27] J. Ren, M. Zhang, C. Yu, and Z. Liu, "Balanced mse for imbalanced visual regression," 2022. [Online]. Available: https://arxiv.org/abs/2203.16427 [Cited on pages 14 and 15.]

[28] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, "Long-tail learning via logit adjustment," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.*

OpenReview.net, 2021. [Online]. Available: https://openreview.net/forum?id=37nvvqkCo5 [Cited on page 14.]

[29] K. H. Brodersen, C. S. Ong, K. Stephan, and J. Buhmann, "The balanced accuracy and its posterior distribution," *Pattern Recognition, International Conference on*, vol. 0, pp. 3121–3124, 08 2010. [Cited on page 14.]

[30] R. Dougherty, A. Edelman, and J. Hyman, "Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic hermite interpolation," *Mathematics of Computation*, vol. 52, pp. 471–494, 04 1989. [Cited on page 19.]

[31] M. Hubert and E. Vandervieren, "An adjusted boxplot for skewed distributions," *Computational Statistics & Data Analysis*, vol. 52, pp. 5186–5201, 2008. [Cited on page 19.]

[32] G. Brys, M. Hubert, and A. Struyf, "A robust measure of skewness," *Journal of Computational and Graphical Statistics*, vol. 13, pp. 996–1017, 2004. [Cited on page 19.]

[33] P. Branco, L. Torgo, and R. P. Ribeiro, "SMOGN: a pre-processing approach for imbalanced regression," in *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, ser. Proceedings of Machine Learning Research, P. B. Luís Torgo and N. Moniz, Eds., vol. 74. PMLR, 22 Sep 2017, pp. 36–50. [Online]. Available: https://proceedings.mlr.press/v74/branco17a.html [Cited on page 20.]

[34] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. ACM, 2016, p. 785–794. [Cited on page 26.]

[35] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Curran Associates Inc., 2017, p. 3149–3157. [Cited on pages 26 and 29.]

[36] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, M. Li, J. Xie, M. Lin, G. Yifeng, Y. Li, and J. Yuan, *xgboost: Extreme Gradient Boosting*, 2022. [Online]. Available: https://CRAN.R-project.org/package=xgboost [Cited on pages 26 and 29.]

[37] Y. Shi, G. Ke, D. Soukhavong, J. Lamb, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, and N. Titov, *lightgbm: Light Gradient Boosting Machine*, 2022. [Online]. Available: https://CRAN.R-project.org/package=lightgbm [Cited on page 26.]

[38] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020. [Online]. Available: https://www.R-project.org/ [Cited on page 26.]

[39] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *Journal of Machine Learning Research*, vol. 18, no. 77, pp. 1–36, 2017. [Cited on pages 29 and 51.]

[40] J. Kruschke and T. Liddell, "The bayesian new statistics: Two historical trends converge," *SSRN Electronic Journal*, 2015. [Cited on pages 30 and 51.]

[41] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001. [Online]. Available: https://doi.org/10.1214/aos/1013203451 [Cited on page 35.]

[42] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984. [Cited on page 38.]

[43] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986 [Cited on page 39.]

[44] J. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002, nonlinear Methods and Data Mining. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167947301000652 [Cited on page 39.]