

A Comparative Study Between Explanation Approaches for models of Increasing Complexity: A case study using Industry Data

Rafael Mamede

Mestrado em Ciência de Dados (Data Science)

[Departamento de Ciências de Computadores](#)

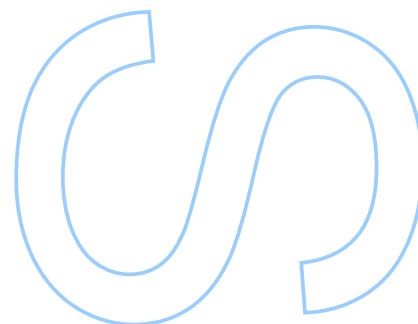
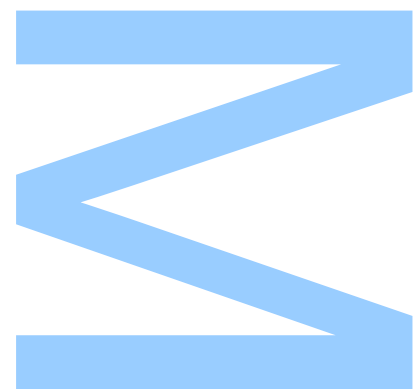
2023

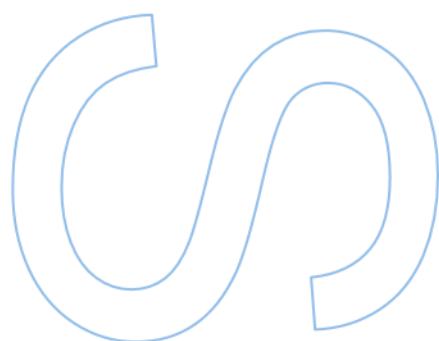
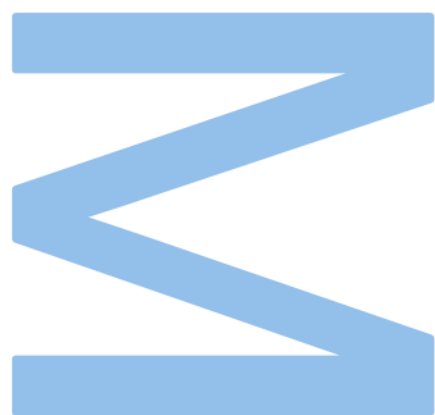
Orientador

[Prof. Dr. João Gama](#), Professor Catedrático, Faculdade de Economia da Universidade do Porto; Professor Catedrático, Faculdade de Ciências da Universidade do Porto; Investigador Coordenador, INESC-TEC

Coorientador

[Prof. Nuno Paiva](#), Sr Data Science Manager & Data Science Chapter, NOS; Colaborador Externo, INESC-TEC





Acknowledgements

I would like to start by thanking everyone who has supported me through this path. To both my supervisors, João Gama and Nuno Paiva, for the incredible support and the amazing opportunities provided. I believe this work would not exist without your direct support, and for that you have my gratitude. I would also like to thank the company in which I interned at for this work for making me grow in what was, for me, a never before explored environment, filled with challenges and opportunities. I would also like to thank my family for the incredible support, not only through my dissertation, but every step of the path that has led me here. Last, but not least, I would like to thank my partner. I couldn't have done this without your support. Thank you!

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto

Departamento de Ciência de Computadores

MSc. Data Science

A Comparative Study Between Models of Increasing Complexity: A case study using Industry Data

by [Rafael MAMEDE](#)

Machine Learning has been overtaken by a growing necessity to explain and understand decisions made by trained models as regulation and consumer awareness have increased. Alongside understanding the inner workings of a model comes the task of verifying how adequately we can model a problem with the learned functions. Hence, in this work we expand on model comparisons beyond traditional aggregate performance metrics. We do so by exploring model robustness, regions of higher error (or weak segments) and quality of extracted explanations. We perform a comparative study of the behaviour of models on industry data, using the Smart Paring application as an experimental environment, evaluating the effect of increasing complexity in additive interpretable models on each of the aforementioned dimensions of the model, and compare the results obtained with a black-box model in the form of XGBoost. Our proposed framework can help practitioners better understand and quantify possible negative effects that increasing model complexity might have in terms of confidence on the trained model.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto

Departamento de Ciência de Computadores

Mestrado em Ciência de dados

Um Estudo Comparativo Entre Modelos de Complexidade Incremental Usando Dados Industriais

por [Rafael MAMEDE](#)

Na área de aprendizagem automática (através de modelos preditivos) tem surgido uma necessidade crescente de explicar e compreender as decisões tomadas por modelos treinados, à medida que a regulamentação e a consciencialização dos consumidores aumentam. A par da compreensão do funcionamento interno de um modelo, surge a tarefa de verificar até que ponto podemos modelar adequadamente um problema com as funções aprendidas em todo o espaço de preditores. Para isso, neste trabalho, expandimos as comparações de modelos para além das métricas tradicionais de desempenho agregado. Fazemo-lo explorando a robustez do modelo, as regiões de maior erro (ou segmentos fracos) e a qualidade das explicações extraídas. Efectuamos um estudo comparativo do comportamento dos modelos em dados do sector, utilizando a aplicação Smart Paring como ambiente experimental, avaliando o efeito do aumento da complexidade em modelos interpretáveis aditivos em cada uma das dimensões do modelo acima mencionadas, e comparamos os resultados obtidos com um modelo de 'caixa-negra' sob a forma de XGBoost. O quadro proposto pode ajudar os profissionais a compreender e quantificar melhor os possíveis efeitos negativos que o aumento da complexidade do modelo pode ter em termos de confiança no mesmo.

Contents

Acknowledgements	iii
Abstract	v
Resumo	vii
Contents	ix
List of Figures	xi
1 Introduction	3
1.1 Motivation	4
1.2 Goals	5
1.3 Contributions	6
1.4 Structure	7
1.5 Smart Pairing: Application Description	7
2 State of the Art	11
2.1 Interpretability and Explainability in Machine Learning	12
2.1.1 Interpretability of models in ML	12
2.2 xAI Taxonomy	13
2.3 Uncertainty Quantification and Error Analysis	18
2.3.1 Understanding the Error	18
2.3.2 Quantifying the Error	19
2.4 Sensitivity Analysis	21
2.4.1 Evaluating Feature Importance through Functionally-Grounded Metrics	23
2.4.2 Morris Sensitivity Analysis	25
2.4.3 SHAP	26
2.5 Robustness in ML	30
2.5.1 Robust Statistics	31
2.5.2 Adversarial Robustness	31
2.5.3 Evasion Attacks in regression problems	34
3 Methodology	37

4	Models	41
4.1	Linear Regression	42
4.2	Generalized Linear Models	43
4.2.1	Practical use cases	45
4.3	Generalized Additive Models	46
4.4	Explainable Boosting Machines	47
4.5	Extreme Gradient Boosting	48
5	Experimental Environment and Results	51
5.1	Data Understanding and Data Analysis	51
5.1.1	Pre-Processing Pipeline	55
5.2	Model Training and Aggregated Performance	56
5.3	Sensitivity Analysis	57
5.3.1	Morris Statistical Indicator Plots	62
5.3.2	Evaluation of Explanations	65
5.4	Error Analysis	68
5.4.1	Understanding Model Error	68
5.4.2	Quantifying Model Error	71
5.5	Adversarial Robustness	73
5.6	Final comparison	75
6	Conclusions	79
6.1	Future Work and Limitations	80
A	Selection of Number of Categories to Encode	83
B	Morris Statistical Plots	89
C	Residual Trees and Error Analysis	93
	Bibliography	97

List of Figures

1.1	Smart Pairing impact on retention lines. Without Smart Pairing the Customer would always be paired with the operator that has the highest performance indicators in a general environment, even if the best operator for that specific client is not that operator.	8
2.1	Morris statistical variable plot (variance of elemental effects as a function of feature importance) for a given model - Each variable in a model would be represented by a point, belonging to one of the highlighted regions . . .	27
2.2	Every possible coalition generated (b) from an initial set of players in a cooperative game containing 4 elements represented by a color (a). (Image adapted from [77])	28
2.3	Evaluation of each coalition in 2.2 through the value function f	28
2.4	Adversarial attack taxonomy according to IBM's adversarial robustness toolbox [86].	33
3.1	High-level overview of the methodology employed for comparison and selection of models. (RQ: Research Question)	38
4.1	PyGAM example of fitting a GAM to data. The splines are represented as the coloured lines and the continuous dark line represents the function built on the combination of them. (figure from [105])	47
5.1	Distribution of Smart Pairing target variable	53
5.2	Proportion of categories in categorical predictors of Smart Pairing	53
5.3	Proportion of outliers in numerical predictors of Smart Pairing	54
5.4	Global Feature Importance of predictors in the Ridge Regression model using SHAP and Morris Method	59
5.5	Global Feature Importance of predictors in the GLM model using SHAP and Morris Method	60
5.6	Global Feature Importance of predictors in the GAM model using SHAP and Morris Method	60
5.7	Global Feature Importance of predictors in the EBM model using SHAP and Morris Method	61
5.8	Global Feature Importance of predictors in the XGBoost model using SHAP and Morris Method	62
5.9	Summary of the proportion of features in each category regarding Independence/monotonicity according to the performed Morris sensitivity analysis	64

5.10	Monotonicity of the Feature Explanations obtained (Morris Sensitivity analysis and SHAP) for each of the trained models. On the left we present the histograms of the monotonicity values for each of the 100 individuals. On the right we present the average values of the Spearman correlation in this sample.	67
5.11	Absolute Residuals for each model in observations where the customer churned.	69
5.12	Jaccard Similarity between observations (pairings of customer and operator) in each model's Highest Error Cohort (HEC).	70
5.13	Local Error (cohort MAE / Global MAE) and cumulative error for each identified cohort	71
5.14	Local coverage and mean interval width assessment using both described strategies. The dashed line represents the desired coverage selected, 90%. We can visually verify a higher stability of coverages using our locally fitted approach, as well as a more granular interval width over test error cohorts.	72
5.15	Sample and Adversarial Error (MAE) for each of the trained models, with a perturbation radius of 0.1.	75
5.16	Final comparison for each trained model.	77
A.1	Error of XGBoost model using only variable <i>ohp_area_dsc</i>	83
A.2	Error of XGBoost model using only variable <i>ohp_direction_dsc</i>	84
A.3	Error of XGBoost model using only variable <i>ohp_function_dsc</i>	84
A.4	Error of XGBoost model using only variable <i>ab_catalog_if_name</i>	85
A.5	Error of XGBoost model using only variable <i>ab_catalog_profile_pack</i>	85
A.6	Error of XGBoost model using only variable <i>ohp_team_dsc</i>	86
A.7	Error of XGBoost model using only variable <i>ab_catalog_tv_box</i>	86
A.8	Error of XGBoost model using only variable <i>ab_catalog_tv_name</i>	87
B.1	Morris statistical variable plot (variance of elemental effects as a function of feature importance) for Ridge Model - Each point represents a feature used by the model	89
B.2	Morris statistical variable plot (variance of elemental effects as a function of feature importance) for GLM Model - Each point represents a feature used by the model	90
B.3	Morris statistical variable plot (variance of elemental effects as a function of feature importance) for GAM Model - Each point represents a feature used by the model	90
B.4	Morris statistical variable plot (variance of elemental effects as a function of feature importance) for EBM Model - Each point represents a feature used by the model	91
B.5	Morris statistical variable plot (variance of elemental effects as a function of feature importance) for XGBoost Model - Each point represents a feature used by the model	91
C.1	Residual distribution for the MVP model segmented by resolution (meta-information)	93
C.2	Trained Residual Tree for Ridge Regression - Obtained in the test set	94
C.3	Trained Residual Tree for GLM Model - Obtained in the test set	94
C.4	Trained Residual Tree for GAM Model - Obtained in the test set	95

C.5	Trained Residual Tree for EBM Model - Obtained in the test set	95
C.6	Trained Residual Tree for XGBoost Model - Obtained in the test set	96

Declaração de Honra

Eu, Rafael de Carvalho Maia Parente Mamede, inscrito(a) no Mestrado em Data Science 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 reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação, 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 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.

Rafael de Carvalho Maia Parente Mamede

Porto, 2 de Outubro de 2023

Chapter 1

Introduction

As the world evolves to be increasingly more data-driven, a need for a widespread understanding of machine learning models and their decision-making processes arises for practitioners and everyone affected by them. In recent years, the increasing regulation of data products reflects this necessity. For example, the European Union General Data Protection Regulation ^{*}, in Recital 71, states that "[the data subject should have] the right ... to obtain an explanation of the decision reached", effectively providing a legal right to an explanation.

Even without considering the regulation, one can easily see the gap that exists if no efforts are made to make decisions made by machine learning algorithms explainable to the end user. When used in sensitive domains such as medical applications, banking, security, law, and many others, traditional performance metrics alone don't allow for an accurate understanding of the decisions made and whether they are being affected by undesired factors such as biases in the training dataset.

Here is where the growing domain of Explainable AI (xAI) enters to provide a bridge between the model and the people (ML practitioners and non-practitioners) affected by said model. The process of 'opening' machine learning models or designing interpretable approaches for a given problem is a rather complex process, without a clear 'one-size fits all' approach. As such, studies on real-world applications of xAI are extremely relevant, not only for the development of the field, but also to the promotion of responsible modeling approaches in industry applications that deal with decisions that affect a broad range of individuals.

^{*}Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) [2016] OJ L 119/1

1.1 Motivation

To comply with regulation and stimulate usage of interpretable machine learning algorithms in industry applications, we frame this work as a way of assessing real world performance, sensitivity, robustness and quality of explanations of models of different levels of complexity and interpretability. This study is initially motivated by Rudin's [1][2] work on pushing for the usage of inherently interpretable machine learning models in high stakes applications, rather than pursuing the simpler path of using black-box models and leaving explanations as a second goal to be extracted via approximations.

As described by Rudin, there exists a widespread belief that more complex and less interpretable models lead to better predictive performances in the generality of cases. Although intuitive, as more complex models often have larger search spaces for the decision function, this is not necessarily the case. If a decision function can be well approximated with a transparent model, the usage of overly complex models comes at no additional benefit. Furthermore, approximations of black-box models through surrogate models often lead to a false sense of confidence over the decision process since if the surrogate model was completely faithful to its less interpretable root, there would be no need for using the black-box model over its transparent approximate.

Research on the usage of state of the art transparent models has shown results that agree with Rudin's perspective. Nori *et al* [3] revealed positive results in terms of performance for transparent models, showing applications where they over-perform black-box models. Although promising, results on low dimensionality datasets, with limited size are hardly an indicator of real industry applications, where data tends to have more volume and attributes.

It is hard to break the standard of using highly adaptive black-box models in a formulaic way in these Big Data environments. This study goes through the path of comparing them with simpler modeling approaches to justify or argue against the usage of these types of models. We argue that by performing a systematic comparison between complex and simpler modeling approaches, we can empirically verify if we have a gain in using more complex models, or whether we are in a use-case where a simpler approach approximates the true relation between predictors and targets in a better and more reliable way.

1.2 Goals

In this thesis, we explore interpretable modeling approaches in a machine learning problem of a telecommunications provider with the main goal of assisting in the model choice through a thorough analysis of model behaviour that goes beyond the traditional performance metric employed in an evaluation step. The main **research questions** that are posed are the following:

1. **RQ1:** What is the effect of increasing model complexity on the predictive performance in real-world applications? - Although less complex transparent models provide benefits for understanding the decision process, at the end of the day, the performance of the ML task is still a very important point. If decisions are not admissible due to high error, understanding them becomes less relevant since the model lacks the ability to move to production, that is, being deployed for automated decisions. Substantiating the usage of a model passes through the verification of performance.
2. **RQ2:** Where do the models of different complexity fail? Can we identify regions of higher error in the feature space? - Error of the model is traditionally not uniform across the feature space. Characterizing errors [4] [5] leads to a better granular understanding of local model behaviour and can be used to understand regions of subpar model fitting. We can expand this to understand if more adaptable models tend to have a higher disparity of prediction error.
3. **RQ3:** What is the effect of increasing model complexity on the susceptibility to perturbations? - Closely related to the study of uncertainty, the evaluation of sensitivity of models provides insight to how numerically unstable the decision process is [6] [7]. By understanding how sensitive models are to fluctuations of the predictors we can conclude how the presence of noise can lead to errors. We can also draw conclusions on the simplicity of the decision process quantitatively. Exploring the effectiveness of adversarial examples [8] also helps us identify which models have more tendency to fail against adversarial attacks, leading to a risk quantification that can help justify the selection of a given model.
4. **RQ4:** Do less complex models provide better quality explanations? - With post-hoc explanations relying on approximations, it is useful to understand the extent to

which they represent the decision process. With studies developed around fooling traditional methods of producing explanations [9], it is necessary to assess their validity, and if a relationship exists between the increase in model complexity and the decrease in explanation validity.

1.3 Contributions

This thesis, and the work developed during the corresponding internship, provide the following contributions:

- **Advancement of xAI Field:** By providing a comprehensive review and revision of technical aspects within the explainable artificial intelligence (xAI) field and responsible Machine Learning for high-risk applications.
- **Practical Application in Industry:** This study focuses on a practical application within an industry environment, emphasizing tangible product requirements. Although studies on synthetic data and openly available data-sets are important for scientific development, studies on real-world data complement them by showing both impact and integration of scientific knowledge with business requirements.
- **Promoting Explainability in the Company's Data Science Practice:** This study directly contributes to the promotion and advancement of explainability within the company's Data Science environment.
- **Academic Recognition:** This work resulted in the acceptance of a paper for the 2023 Discovery Science conference, showcasing its academic impact.
- **Speaking Engagement at Data Makers Fest:** Associated with this work, a talk has been accepted at the 2023 Data Makers Fest, further disseminating its knowledge and findings to a wider audience.
- **Direct Impact on Smart Pairing Model:** Work done in the exploration of the data has led to changes in the pre-processing pipeline of the current Smart Pairing model. Additionally, this work positively affected the current development pipeline by including exploratory methodologies developed, leading model evaluation to go beyond aggregate performance metrics.

1.4 Structure

This document is structured in different chapters, organizing the work developed in the best way possible. Firstly, we conclude this chapter with a description of the industry use-case in question. We then follow this introduction by Chapter 2, where we provide a literature review on the field of xAI and other important aspects of responsible Machine Learning. Chapter 3 goes through the methodology used in this comparative study applied to Smart Pairing. In chapter 4 we explore the theoretical basis of each of the models compared in this work. Chapter 5 exposes our experimental environment and the results obtained in this work. Finally, chapter 6 briefly summarizes and concludes the work developed, as well as providing our proposals for future work.

1.5 Smart Pairing: Application Description

Smart Pairing is a project developed by a Telecommunication Provider that seeks to optimize the pairing of operators and customers calling a retention line regarding the possibility of changes to their current telecommunications plan. The clients calling these retention lines can fall into several categories, such as customers that are seeking to change their current plan for a more favourable one, customers scanning the market to assess how competitive the offers provided by the company are, or customers unhappy with their current service. In either case, there is a risk of churn (customer abandonment) associated with these requests.

When a customer calls a retention line and more than one operator is available, there needs to be a process of selection that determines who answers the call. One naive way to implement this decision would be by randomly selecting one of the available operators for each received call. This, of course, is not very optimal, since more experienced operators tend to have an edge over new entrants, and if an operator is quite good then choosing them over others would lead to a better experience for the customer and a better chance of customer retention. A traditional operational approach is to have some indicators that rank these operators based on past performance for determining the optimal pairing at time of call. This approach, although more sophisticated, completely leaves out of the equation the customer side, providing the same priority of operators regardless of the caller.

Even when just dealing with the portion of B2C (Business to Consumer), consumer profiles and needs vary widely from case to case, and so does operator experience for each of the many types of products the company provides. The next level of optimization in the retention lines involves introducing this knowledge in the process of the decision at the time of the call. Smart Pairing proposes a pairing between operators and clients based on analytics, providing a custom experience for each client. For a given customer, we seek to recommend the best operator, where an ideal pairing system minimizes churn and maximizes contracted revenue. To exemplify the leap in the decision process, figure 1.1 compares the non-tailored previous solution with the one provided by the project.

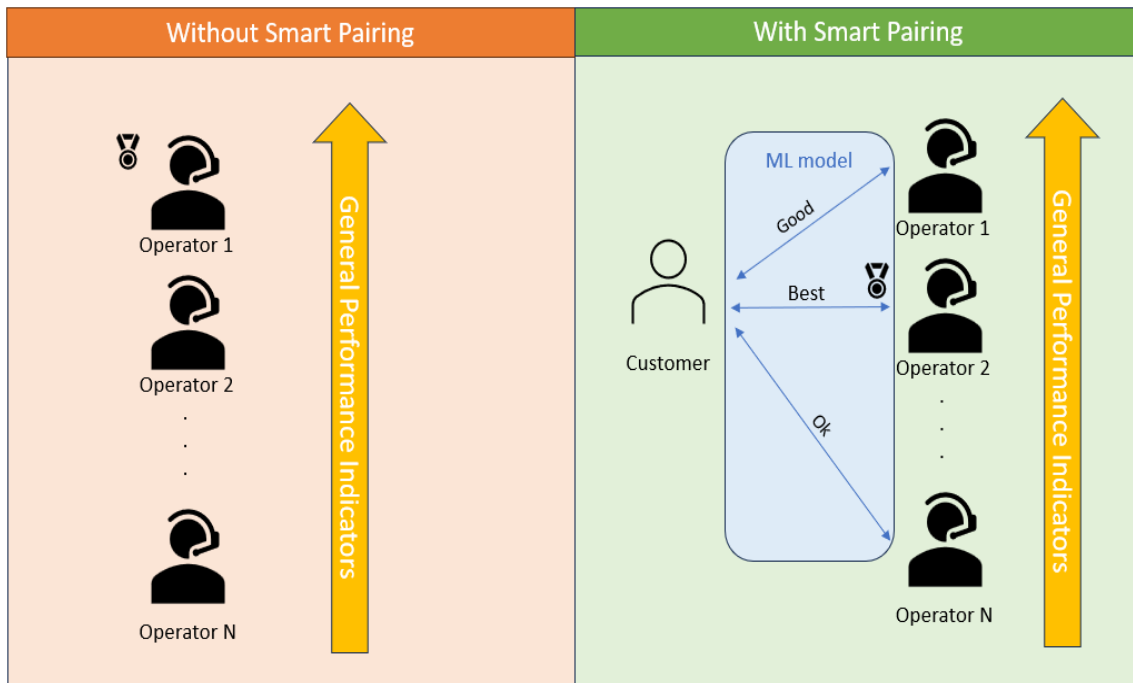


FIGURE 1.1: Smart Pairing impact on retention lines. Without Smart Pairing the Customer would always be paired with the operator that has the highest performance indicators in a general environment, even if the best operator for that specific client is not that operator.

As this application deals closely with both operator and customer data there is a strong necessity of explainability to comply with Privacy Impact Assessment requirements. The main risk of an application such as Smart Pairing arises from the possibility of biased decisions towards specific groups and demographics. To prevent this, interpretability and explainability of the model used to generate the pairing is fundamental to control this risk.

Details on the information used for the modeling approach, as well as a machine learning problem formulation, are further elaborated in the methodological description

in Chapter [3](#).

Chapter 2

State of the Art

Artificial Intelligence was a term first coined by Alan Turing in the 1950's [10] and since then it has been booming ever since, both in terms of research and real world applications. Later in the 50's, Arthur Samuel introduced the term of machine learning [11], which has also boomed as one of the more rapidly expanding fields of artificial intelligence. Machine learning itself can be seen as a pathway to artificial intelligence [12], producing decisions for applications that learn from data as a sort of mapping that links observations (predictors) with a decision. The development of better models and the increase in computational power have led to the popularity of using these models for automated decision making today, across multiple fields. The field of eXplainable AI (xAI) introduces a way to view this decision process as something to be understood by humans rather than an automated black-box process. This field dates back to the late 70's [13] [14] [15], although the term was only coined in 2004 [16], and it, too, has had an impressive expansion, specially in later years due to regulation and increase of awareness within society.

The growth of xAI as a field has, however, been slightly controversial. The main critiques have revolved around lack of consensus on the definition of the field [17] and loose interpretations and definitions of areas surrounding it [18]. These criticisms substantiate the need for a review of the field and the contextualization of terminology and definitions used in this work.

We start this chapter with defining concepts around model interpretability and explainability in section 2.1, followed by a revision of taxonomies in the xAI field (chapter 2.2). We follow these sections by exposing a revision on the topics used to enhance model understanding, namely error analysis (chapter 2.3) for local model behaviour, sensitivity analysis and feature importance (chapter 2.4), and model robustness (Chapter 2.5). This

revision provided us with the tools to approach the proposed research questions in our industry context.

2.1 Interpretability and Explainability in Machine Learning

An initial challenge, posed to every newcomer in the Explainable AI research space, is to find consensual definitions of utilized terms as most don't have a closed mathematical definition. Such is the case of interpretability, defined by Miller [19] as the degree to which a human can understand the cause of a decision. Another definition of the same term is given by Kim *et al.* as the degree to which a human can consistently predict the models result. Molnar [20], in his book, opts for the use of Interpretability and Explainability as interchangeable terms.

For this work, we will differentiate between interpretability and explainability in a manner similar to Rudin [2], by tying interpretability more closely with the inherent ability of a model's decision to be understood by a human by virtue of the model's transparency, without a post-hoc approximation. In contrast, explainability is the process of taking a machine learning model and producing explanations, that is, insights to model behaviour, regardless of using the model's inherent properties, or external approximations or analyses.

Definition 2.1. Interpretable and Explainable ML [adapted from [2]]

An Interpretable machine learning model is constrained in model form so that it is either useful to someone, or obeys structural knowledge of the domain, such as monotonicity, causality, structural (generative) constraints, additivity, or physical constraints that come from domain knowledge.

An Explainable machine learning model relies on post-hoc explanations, and does not provide inherent explanations to the decision process.

2.1.1 Interpretability of models in ML

Model interpretability is not a property that is truly binary, meaning that there is no simple on-off indicator that can tell us whether a model is or is not transparent. Rather, we often can interpret this property more as a spectrum. Although models like regression/decision trees are often considered interpretable, we can conceive an example where, with high enough depth, we increase complexity of the decision function enough that it

becomes hard to interpret all decisions on a global level. Furthermore, not all machine learning models that have been classified as interpretable in literature, have the same properties leading to model transparency. As such, a more fruitful way of classifying a model as interpretable, proposed by Sudjianto *et al.* [21], is through a qualitative assessment of the following properties:

- Additivity - If a model takes an additive or modular form, meaning the decision function can be decomposed through a sum of interpretable functions of predictors;
- Sparsity - If features or model components are regularized in order to use a smaller set of features. Having fewer features or components tends to be more interpretable;
- Linearity - If feature effects are linear (constant as special case). Linear or constant feature effects are easy to interpret;
- Smoothness - If feature effects are continuous and smooth (no sharp discontinuities), hence being simpler to interpret;
- Monotonicity - If feature effects can be modeled to be monotone. Increasing/decreasing effects are desired by expert knowledge;
- Visualizability - If the feature effects can be directly visualized. Visualization facilitates the final model diagnostics and interpretation.

Alongside these characteristics, projection (if projection is used for feature combination) and segmentation (if segmentation is used for heterogeneous modeling) are also proposed features for assessing model interpretability but the models used in this work do not take advantage of any of these characteristics and as such we won't explore them deeply.

The authors propose that for each model, we can assess how well each category can be upheld either globally, for the general decision function, or locally, for the neighbourhood of a data point in the feature space.

2.2 xAI Taxonomy

For the review of the xAI ecosystem it is useful to start with motivating the field by exposing the needs it seeks to address. Adadi *et.al* [17] frames xAI as a field that tries to provide

explanations with different goals. Not only are we interested in understanding model behaviour, we are interested in what we can do with that knowledge once we obtain it. The authors frame the end goals of xAI across the following categories:

- explain to justify - the decisions made by utilising an underlying model should be explained to increase their justifiability;
- explain to control - explanations should enhance the transparency of a model and its functioning, allowing its debugging and the identification of potential flaws;
- explain to improve - explanations should help scholars improve the accuracy and efficiency of their models;
- explain to discover - explanations should support the extraction of novel knowledge and learning relationships and patterns.

Although framing the questions around xAI is important for context, it is not enough to understand the inner workings of the field. Methodologically, many authors posed ways of categorizing procedures in the field to achieve these common goals. In Oliveira's review [22], the author classifies procedures according to 6 criteria: locative, agnosticity, scope, transparency, output, and methodology.

The locative criterion [20] frames a method according to its position relative to the machine learning task, either applied before modeling (pre-model), while building the model (in-model) or after the modelling (post-hoc). Agnosticity [20] refers to how general a certain method is, meaning, whether it is dependent on the model choice, or it is model-agnostic (applicable to multiple models). The scope criterion [20] divides local approaches that provide insight to a single decision from global approaches that provide a generic reasoning on model behaviour. Transparency criterion [18] is more closely related to the inner-working of the model choice (in an in-modeling fashion), and refers to properties such as simulatability, decomposibility and algorithmic transparency of the model. The output criterion [20] refers to the result of the explainability efforts, meaning either feature summaries, surrogate models, understanding of internal model behaviour, or data points that represent typical or atypical model behaviour (through prototypes and counterfactuals). Finally, the methodology criterion [23] relates closely to Deep Neural Networks (DNN), and defines the procedure by function or purpose by achieving, for example, representations of the DNN or taking advantage of its processing.

Schwalbe *et al.* propose yet another different taxonomy alongside pertinent definitions after a thorough revision of surveys on xAI [24]. Rather than focusing on different criteria, the authors propose a categorization in a procedural manner, according to the steps for building an explanation system. In doing so, they achieve the following levels of taxonomy:

1. Problem Definition - In this step, an xAI problem needs to have the ML task well defined (type of machine learning problem: regression, classification, segmentation, etc.; and type of data dealt with: tabular, image, natural language, ...) and the practitioner should review the type of explanandum (model to be explained) that is used in the task. In this choice, one can take the approach of selecting an inherently interpretable model, that is functionally understandable by a human [25], opt for constrained model choices (such as enforcing monotonicity constraints) via an 'interpretable by design' [26] solution, or opt for a full black-box model with no constraints and move to a complete post-hoc approach to interpretability.
2. Explanator Properties - Here the authors view the explanator, the tool or model that provides explanations, as a function that maps a certain input to an explanation. As such, to define it, we need to understand the inputs it will take, the outputs it will produce, the function class (or the mathematical properties the explanator will assume), and the degree of interactivity it will have with the user.
 - (a) Input - Starting with the input, it defines what the explanator object needs to produce explanations. Some methods might require full access to the explanandum or data samples, but other methodologies might require user-feedback. Other topics surrounding the inputs revolve around how easily, if at all, an explanator can be ported to other types of explanandum, by review to the agnosticity criterion of an explanator. Additionally, the scope criterion also plays a role in this step, as it is necessary to understand whether the explanator will be providing local or global explanations.
 - (b) Output - The output of an explanator is defined by 3 important points: the explanator object (what is explained), the output type (how it is explained), and the way the explanation is presented. The objects of explanation can be either representations of the inner workings of a model (such as latent space representations found in Deep Neural Networks), processing objects of the model

to try to provide reason to how decisions are made (for example through the study of feature attributions or decision boundaries), stages of the development or training of the model (for example through exploration of effects of parameters during model training[27]), uncertainty (through the quantification and explanation of the error and uncertainty associated with a prediction), and data (through data analysis and feature mining). The output type describes the type of information presented to the explainee, be it by example instance (such as word clouds of related terms in Natural Language Processing problems), by presenting contrastive/counterfactual/near miss examples (including adversarial examples), generation of prototype examples (data instances that are representative of the data [20]), obtaining of feature importances (what features most contribute to a prediction), obtaining of decision rules (for example through decision trees, or hyperplane rules [28]), defining dimensionality reduction strategies (through finding lower dimensionality sub-spaces to represent the data that maintain interpretability), or through visual representations (through dependency plots [17] or explanatory graphs [29]). Finally, the presentation of the outputs relates both to the form the explanation takes (commonly graphical via diagrams, graphs, and heatmaps; or textual through natural language or formal form) and to the levels of abstraction of the information (what information is being used, such as features or derived features, and the level of complexity of the explanations). Associated with the presentation, the practitioner should also review the anonymization of sensitive information, in order not to infringe on privacy regulation [30].

- (c) Interactivity - This point refers to the degree of interaction of the explanation to the end-user. Explanations can be either static or interactive, accepting user feedback as explanation input. The work on this thesis revolves around static interactions so this particular aspect will not be explored in high detail.
- (d) Mathematical constraints of the explainer - An important aspect of producing an explanation passes through disclosing the properties that an explainer possesses. Many methods of explanations are constrained to have linearity or monotonicity to ease interpretation for the end user. Additionally, disclosing whether an explanation takes into account feature interactions, or even the totality of features is rather important for understanding the production of the

explanation. Other parameters such as size constraints of surrogate models (for example maximum depth in a surrogate decision tree) or number of iterations in models that require several queries to the explanandum are also important for replicability and transparency of the process of generating the explainer.

3. Metrics of Evaluation - Complementing the implementation of xAI systems, comes the phase of assessing its quality. Metrics for evaluating an xAI system fall into one of the 3 following categories [31]:

- (a) Functionally-grounded - These metrics, unlike the ones in the other categories, do not require any human feedback and instead measure the formal properties of the explainer. Examples of such metrics include: stability or robustness [32], measuring the change of output given a change in the input samples and will be explored more in depth later in the chapter; faithfulness [33] and fidelity [34] all measure how accurately the behavior of the explainer conforms with that of the explanandum or the object of explanation; Completeness, or coverage, measures the validity range of an explanation. These types of metrics bring more insight to how good an explanation is by highlighting possible oversimplifications due to mathematical constraints of the explainer.
- (b) Human-grounded - These metrics require direct human involvement on proxy tasks, measuring a psychological effect of explanations. They tend to be a bit more subjective, although they are, nevertheless, important for assessing the final impact of an explanation model. Properties such as the degree of interpretability of the explanations, effectiveness, and time efficiency for a user to build a mental model of the explanation are all concepts regarded as Human-grounded evaluations.
- (c) Application-grounded - These metrics are similar to Human-grounded, differing on the human target. Application-grounded metrics work closely with the end-user of an xAI system and gather information on satisfiability, novelty provided by the system, automation capability, and other concerns that can be relevant for auditing and maintaining the AI system.

2.3 Uncertainty Quantification and Error Analysis

Uncertainty in machine learning models refers to the lack of complete certainty or precision in their predictions. ML models are often trained on limited or noisy data, making it challenging to make definitive predictions. Uncertainty can arise due to factors such as inherent variability in the data, model assumptions, parameter estimation, and inherent stochasticity in the underlying processes [35]. Quantifying and understanding uncertainty in ML models is important as it provides insights into the reliability of predictions.

We subdivide this section into two relevant parts in the process of analysing the error:

- Understanding and diagnosing the error that occurs in the inference step of the model with the main goal of identifying regions of model weakness;
- Quantifying the error in the predictions made by the model.

2.3.1 Understanding the Error

Understanding error in prediction tasks (Error Analysis - EA) is a task that has been around for a while, with works such as Patrick Hall's *Machine Learning for High-Risk Applications* [36] highlighting the importance of the task for model understanding. Here traditional methods like residual plotting, in order to visually identify patterns remaining on the residuals for given features have become standard practice, specially applied to linear regressions to verify if residuals are consistent with normally distributed noise. When problems scale to having high dimensionality and complex feature interactions, these elbow-grease methods fall a bit short, as it is not feasible to simply analyse the residuals of every possible feature for meaningful results. As such more sophisticated methods for mining patterns still remaining on residuals have arisen, such as through their modeling.

The necessity to identify and diagnose machine learning model failures in large scale has led to the development of tools for practitioners' use, such as Uber's MANIFOLD [37] or Microsoft's Error Analysis [38], that allow performing this task in an exploratory way. Here we highlight the Error Analysis package in particular since it follows an approach more common in the area, modeling the residuals through either decision or regression trees (depending on the machine learning model). This approach has been highlighted in other works both using model features[36][39] or meta information[40]. The result of this sort of approach is a segmentation of a given feature space (or a feature space with

meta-information), S , into n sub-regions, $R_i \in S$ ($R_i \in \{R_1, \dots, R_n\}$ with $\bigcup_{i=1}^n R_i = S$, and $R_i \cap R_j = \emptyset$ for $i \neq j$), where local model behaviour can be explored. The segmentation can be either using only the features available for inference or meta-information about the observation that is not accessible to the model. The latter can be either information that is deemed sensitive, such as gender or ethnicity, not included in the training for fairness concerns or information regarding characteristics of the data that are not seen by the model (such as the presence of a cat in a binary image classifier that identifies images of dogs). These two complementary ways of approaching EA can lead to different steps for model improvement.

Identifying regions using boundaries or rules in the feature space is the more direct way to find improvements for an existing model. One way to do so is to train a different model for data in the feature space's high error region, or regions, to complement the existing model. Another way to improve performance in under-performing regions is to increase the training observations in the area, either by obtaining fresh data or performing data augmentation.

Using meta-information of the observations can lead to less direct ways of improving existing models since new observations might lack the feature at the time of inference, or even the use of this information might collide with fairness concerns. As such, ways to improve the developed model with this type of EA tend to follow an exploratory path, auditing the model by framing hypotheses on why it under-performs in the region [41] and complementing the work with other xAI approaches to understand the difference between local and global behaviour. There have been approaches suggested for automating the process of finding patterns that lead to higher error in applications related to image and text ML problems [40], such as using a pipeline based on meta-feature extraction from examples and a rules-based classifier for the prediction of errors.

2.3.2 Quantifying the Error

Uncertainty quantification in machine learning (ML) is a critical field that addresses the inherent uncertainty and variability in predictive models. While traditional ML models often focus on producing point estimates, uncertainty quantification aims to provide a more comprehensive understanding of the confidence or uncertainty associated with these predictions. Although many ways of quantifying model error have been proposed,

in this work we will expose one popular way to achieve prediction intervals through Conformal Prediction which has the advantage of being usable for an arbitrary model[42].

For implementation of Conformal methods the MAPIE framework [43] already possesses multiple ways of estimating uncertainties associated with the predictions of Machine Learning models to better understand the robustness and predictive power of predictions. Mathematically, the Conformal Prediction problem can be defined as follows: given a training dataset $(x, y) = (x_1, y_1), \dots, (x_n, y_n)$, we seek to create an estimator for the prediction interval, $\hat{C}_\alpha$, such that the probability that the real value of the target of a new observation falls on the estimated interval is $P(y_{n+1} \in \hat{C}_\alpha(x_{n+1})) = 1 - \alpha$. The value of $1 - \alpha$ is defined as the target coverage and is a user-determined parameter based on desired risk, α , which depends on how lenient the constructed interval is on error.

To understand how these prediction intervals are constructed let us analyse the example of the slip-conformal method. Here we first consider a separate dataset to the training and test ones that we call the calibration set. Then, for a selected level of risk, α , we determine the estimated quantile of the conformity score distribution [44] (here selected to be the absolute residual score) associated in the calibration set and use this value to construct boundaries around the prediction. Thus a new prediction that we make takes the following form:

$$f(X) \pm (\text{Quantile}_{1-\alpha}\{|Y_1 - f(X_1)|, \dots, |Y_N - f(X_N)|\})$$

, where $\{(X_1, Y_1), \dots, (X_N, Y_N)\}$ are the calibration set examples and $f(\cdot)$ our machine learning model.

Other approaches such as cross-conformal predictions base themselves on a similar principle, fitting a desired model on each of the folds and assessing out-of-fold conformity scores using the selected conformity score function. The prediction interval is then constructed from estimating the quantile associated with risk α from the conformity scores.

One of the limitations of these methods is the lack of ability to correctly undertake heteroscedastic noise (noise not uniformly distributed in the feature space). Expanding on this weakness, Conformalized Quantile Regression [45] offers an alternative approach to regular conformal methods, better adapting the width of prediction intervals to regions of higher noise. This implementation requires compatible models, that is, models that take a quantile regression as the machine learning task.

Still on the note of conformal predictions, Mondrian Conformal predictors [46] expand on the conformal methods by locally fitting conformal predictors. This is done through a mapping of each observation in the calibration set into disjoint subsets according to a Mondrian taxonomy with k categories (a function that maps each instance into a unique sub-set of the feature space), and where a conformal prediction is produced for each sub-set. This procedure can be achieved, for example, through binning a certain variable or a set of variables[47] or through algorithms that segment the feature space [48].

Also of note, in uncertainty quantification, IBM's Uncertainty Quantification 360 [49] provides alternatives to obtaining prediction intervals by employing models that intrinsically produce uncertainty estimates and offering alternative post-hoc methodologies for obtaining prediction intervals on already trained models.

2.4 Sensitivity Analysis

Sensitivity Analysis (SA) is described broadly, by Razavi *et. al.*, as "the study of how the outputs of a system are related to, and are influenced by, its inputs" [50]. In fact, this field of study is not necessarily bounded to machine learning, with applications including any area that develops a modeling approach for a specific problem, such as physical models for atmospheric sciences [51].

Sensitivity Analysis extends to machine learning problems quite naturally as a way to open black-box models in an explainable machine learning fashion [52]. In fact, a strong parallel between Sensitivity Analysis and another well defined xAI problem, feature importance, is quite noticeable as many adaptations of this area are applied to understand variable contributions for predictions [53]. Some authors don't differentiate between some established feature importance methods such as SHAP and LIME [54] and the area of Sensitivity Analysis, opting instead to categorize them as part of the same field [55].

SA can be subdivided into two categories based on the scope of its results: *global* SA and *local* SA. Global SA methods focus on varying all available inputs and seek to obtain values for the contribution, or importance, of each feature for the model in a global sense. In contrast, local SA methods focus on obtaining contributions of each variable to explain one particular prediction [55]. Both local and global SA methods have their place in the xAI ecosystem; however, when dealing with large amounts of data, instance based explanations are less useful for explaining the way predictions are obtained. Rather,

local SA has a larger contribution when given to the end user to comprehend the model behavior on their single case application.

Even with the target set on Global SA for identifying model behaviour, obtaining feature contributions is hardly a task with just one path, with different approaches appearing to appease different models and characteristics of the data (such as high dimensionality or high correlation of attributes) [56]. On the revision of Stein *et. al.* [53], the authors segment different Global SA approaches as being either: derivative-based, variance-based, density-based or model-based.

Derivative-based methods involve analyzing the sensitivity of a model's output with respect to changes in input parameters by estimating derivatives. Common derivative-based approaches include the Morris method [57] and Derivative-based Global Sensitivity Measures (DGSM) [58] [59].

Variance-based methods quantify contributions of specific features by quantifying how changing the values of that specific feature would result in changes in the output of the model. This, unlike derivative-based approaches, takes a more probabilistic view of the problem and attempts to estimate the fluctuation of the average response given a change in a feature, maintaining the others fixed, $\text{Var}(E_{\mathbf{x}_{-i}}(Y|x_i))$. Variance-based methods include Sobol's method [60] and the Fourier Amplitude Sensitivity Test (Fast) [61] [62].

Density-based SA methods, as the name implies, take into consideration the Probability Density Function or Cumulative Density Function of the model response to obtain metrics for the contributions of each feature. These methods tend to be hard to scale to problems of high dimensionality due to computing time. Common methods of density-based approaches include DELTA [63] and PAWN [64].

Model-based approaches are tied closely with interpretable-ML, as inherent ways to obtain importance values for specific ML models, such as regression coefficients in Linear Regression, are placed in this category. Interestingly, the authors also placed the model-agnostic approach SHapley Additive exPlanations (SHAP) into this category, seemingly expanding it to include explainable-ML approaches. Although this inclusion makes the definition of this category less closed, as it is now not completely tied with a specific model choice as suggested, it is quite an interesting way of including well known feature importance methods into the SA umbrella term. Following the inclusion of SHAP under

this category, LIME seems like another feature importance method that could be described under this category [54].

Culminating the review of these methods, the authors highlight the strengths of the Morris method, both in scalability and reliability of the importance measures, and thus this method was included in this study and will be described more in depth. Alongside the Morris method, SHAP has had an increase in popularity in industry applications due to its robust implementations and properties [65]. Due to the high amount of resources available surrounding this method, its agnosticity and strong theoretical foundation on game theory it will also be further expanded on and applied in this work.

2.4.1 Evaluating Feature Importance through Functionally-Grounded Metrics

In this section, we have reviewed ways to achieve feature importances (or feature attributions) explanations, through different explanators. A question that can follow from the existence of different feature attributions for the same explanandum is: 'Which one is correct?'. Different methods will have different degrees of validity, depending on how well the explainer's mathematical restrictions and assumptions align with the explanandum. Just because an explanation is understandable to humans does not mean it correctly reflects the reality of the model. Here, functionally-grounded metrics can help understand whether we are providing appropriate explanations for our problem.

One challenge of evaluating attributions is that we do not possess ground-truth values for what the methods are assessing [66]. If we did, there would be no need for sensitivity analysis. Here, for the implementation of metrics for determining the quality of feature importance, we follow the reasoning of Nguyen *et. al* [67] and evaluate desirable properties of these explanations. Firstly, the authors posit that 'the importance of a feature should be proportional to how imprecise would the prediction be if we did not know its value'. Here it is useful to consider the extreme case when the importance of a feature is 0, meaning that it has no contribution to the prediction of the model, and note that if we replaced that value for another value in the feature's domain the prediction should be unchanged. The opposite effect is also plausible, meaning that if a feature has a high impact on the prediction, altering its value to another in its domain should result in a high fluctuation of the prediction.

Moving to a more mathematical formulation of this idea, let us consider a machine learning problem where the predictor function, $f : R^n \rightarrow R$, maps n predictors to a prediction value y , such that for a data point, $x^* = (x_1^*, \dots, x_n^*)$, $y^* = f(x^*)$ estimates the target of the point. For each predictor, a feature attribution method (using some form of sensitivity analysis) will result in attributions a_i , $i \in \{1, \dots, n\}$. For a loss function, $l(\cdot)$, of interest, such as the mean absolute error, the author posit that:

$$|a_i| \propto E(l(y^*, f_i)|x_{-i}^*) = \int l(y^*, f_i(x_i))p(x_i)dx_i$$

, where f_i is the restriction of the model to feature i at a point x^* , meaning that the only variable changing is feature i and all of the other features are fixed at x^* ($x_1 = x_1^*, \dots, x_{i-1} = x_{i-1}^*, x_{i+1} = x_{i+1}^*, \dots, x_n = x_n^*$). The probability density, $p(x_i)$, can be either conditional on the fixed values of the other features (true to the data [68]) or considered independent on the values of the other features (true to the model [68]). This formula reflects the property posed initially by the authors: a higher (absolute value) feature attribution should reflect a higher expected fluctuation of the prediction (given by the loss between the initial prediction and predictions obtained by randomly fluctuating the feature corresponding to the attribution).

Given this desired property, the authors propose two related metrics that are indicative of how faithful a feature attribution explanation is:

Definition 2.2. Monotonicity of feature attributions [adapted from [67]]

The monotonicity for feature attributions is defined as the Spearmans correlation coefficient, $\rho_S(\mathbf{a}, \mathbf{e})$, where $\mathbf{a}$ is a vector containing the absolute value of the feature attributions ($\mathbf{a} = (|a_1|, \dots, |a_n|)$) and $\mathbf{e}$ the vector of estimated expectations ($\mathbf{e} = (E(l(y^*, f_1|x_{-1}^*)), \dots, E(l(y^*, f_n|x_{-n}^*)))$)

Definition 2.3. Non-sensitivity of feature attributions [adapted from [67]]

The non-sensitivity metric is defined as the symmetric difference of set A_0 and X_0 , $|A_0 \Delta X_0|$, where set A_0 is defined as the subset of indices assigned with zero attribution ($A_0 \subset \{1, \dots, N\}$, where $a_i = 0$) and X_0 is the subset of features with no functional dependence on the model.

We can delve a bit deeper into these definitions and tie them with the original idea of the authors: a higher (absolute value) feature attribution tied with expected fluctuation of the prediction. If this increase is always true, then the relationship between these two quantities will be perfectly monotonic. The monotonicity metric is less restrictive than

verifying the Pearson Correlation as it allows a relationship between both quantities to be more complex than the initially framed linear dependency ($|a_i| \propto E(l(y^*, f_i)|x_{-i}^*)$). Non-sensitivity is more closely tied with features with zero feature attribution, and the verification that they in fact have no impact in the model. This proved less relevant in our work as having zero feature importance was not common in our analysis, but nevertheless its exposure is important to understand the evaluation of these explanations.

2.4.2 Morris Sensitivity Analysis

The Morris Method [57] for sensitivity analysis, also known as the Elementary Effect Test [69] [70], is a Global Sensitivity method based on one-at-a-time variation of the parameters locally that is then averaged to create a global sensitivity measure.

For the assessing of feature importance, we consider that each parameter to be varied is a feature of a given machine learning model, $f : R^n \rightarrow R$, with n features. Each model parameter is then mapped to an n -dimensional unit hypercube and we consider that for a given point, $x = (x_1, \dots, x_n)$, y is the evaluation of that point through our machine learning model ($y = f(x)$). The Morris [57] procedure starts by defining the trajectories that we will consider for a given point. Here we consider a p -spacing of the hypercube that defines the possible grid jumps, Δ , that we will consider for a given feature. As such, the fluctuation of feature k will be defined as $\mathbf{x} + \Delta = (x_1, \dots, x_k + \Delta_k, \dots, x_n)$, where $\Delta_k \in \{\frac{1}{(p-1)}, \dots, 1 - \frac{1}{(p-1)}\}$, such that this jump creates a point within the domain of the feature [71].

Determining the importance of each of these features passes through the computation of what is defined as the elementary effects of varying feature k :

$$EE_k = \frac{f(\mathbf{x} + \Delta) - f(\mathbf{x})}{\Delta_k}$$

Considering that for a certain feature we obtain n_r elementary effects through different trajectories of this feature, we can obtain a statistical summary of these elementary effects to explain the effect of this feature in the model. Three quantities are usually computed for this effect:

1. The mean, $\mu_k = \frac{1}{n_r} \sum_{i=1}^{n_r} EE_k^{(i)}$, which identifies the aggregate effect of varying a certain feature;
2. The standard deviation, $\sigma_k = \sqrt{\frac{1}{n_r} \sum_{i=1}^{n_r} (EE_k^{(i)} - \mu_k)^2}$, which gives information regarding the presence of nonlinearity and/or interactions between the k^{th} parameter

and other parameters (without, however, distinguishing between either effect). On the extreme case where this variance is null, we are in the presence of a feature that is completely linear and non-interactive;

3. The Campolongo mean [72], $\mu_k^* = \frac{1}{n_r} \sum_{i=1}^{n_r} |EE_k^{(i)}|$, that seeks to create a quantity closely related to the mean that is robust to changes of sign of the elementary effects. This quantity is more useful to estimate importance in non-monotonic models or models that possess a high number of interaction between features.

From these quantities, we can visually understand the effects of each feature by plotting the statistical indicators μ^* (or μ when not using the Campolongo correction) and σ on a plane. This visual interpretation leads to the identification of features that:

1. Have negligible effect in the model - low μ^* and low σ ;
2. Linear and additive important features with low interaction - high μ^* and low σ
3. Non-linear or highly interactive features - high σ .

Some works [73] [74] also propose the usage of the ratio $\frac{\sigma}{\mu^*}$ as a way to categorize features into groups that possess different levels of independence and monotonic impact. For values $\frac{\sigma}{\mu^*} < 0.1$ we are in the presence of highly independent features with high linearity, values such that $0.1 < \frac{\sigma}{\mu^*} < 0.5$ indicate either low interactions or strong monotonic influence of the feature, for values such that $0.5 < \frac{\sigma}{\mu^*} < 1$ we can infer non-negligible interactions with other features or almost monotonic influence in the model, and finally, for values such that $\frac{\sigma}{\mu^*} > 1$ we infer high feature interaction or highly non-monotonic influence of the feature. This ratio can expand our visual interpretation of the summary statistics by including plots of these lines for ease of identification of what group certain features fall on. In figure 2.1 we demonstrate the different regions in the Morris Statistical indicators plot.

2.4.3 SHAP

SHapley Additive exPlanations (SHAP) [54] [75] is a method that allows for the estimation of feature attributions (or feature importance) through the calculation of shapley values, a concept borrowed from coalitional game theory [20]. The main idea behind the shapley value is to find the marginal contribution of specific features for a certain prediction. First, let us consider that each specific feature is a 'player' in a cooperative game and for a

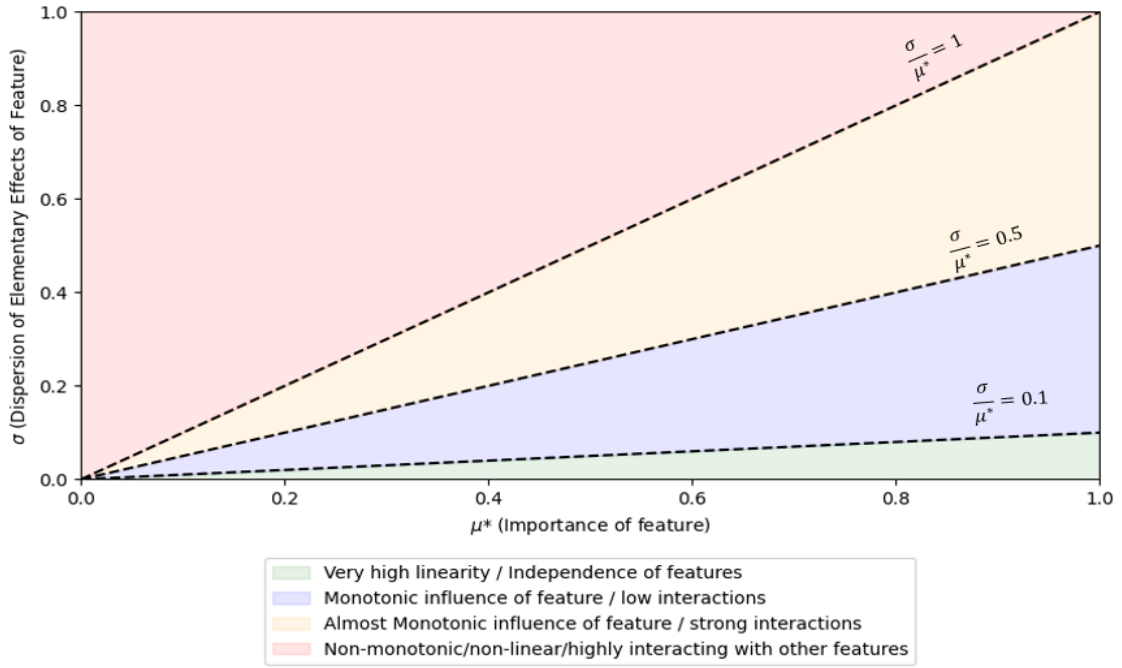


FIGURE 2.1: Morris statistical variable plot (variance of elemental effects as a function of feature importance) for a given model - Each variable in a model would be represented by a point, belonging to one of the highlighted regions

certain result to be obtained in this game, we consider that each player contributes to the outcome with some value. Let us consider that the result of this game is modeled by a function, $f(\cdot)$, dependant on the player's contributions. In order to obtain the value of the game we can consider that each player offers a linear contribution, such that by summing up each contribution we get the result of the game. These contributions, however, don't need to be the same. Depending on the utility of the player to the game, or in our case the importance of a feature to the prediction, these marginal contributions should vary accordingly. The shapley values, ϕ_i , that estimate the contribution of player i (feature i) are given by [76]:

$$\phi_i = \sum_{S \in N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} [f(S \cup \{i\}) - f(S)]$$

, where s is the number of elements in coalition S , $s = |S|$, n the number of elements in the coalition with all elements N , $n = |N|$. Each coalition is a sub-group of players (features) that can possibly contribute to the game, with N being the case when all players are considered. Before delving deeper into the reasoning of the shapley value, it is useful to understand how we can form these coalitions. Figure 2.2 exposes an example of all possible coalitions formed by a group of 4 players in a game. For each individual coalition

we form, we can obtain the result of the game by scoring them with our value function $f(\cdot)$, imposing that if no players are active in the coalition the score is 0, as in figure 2.3 .

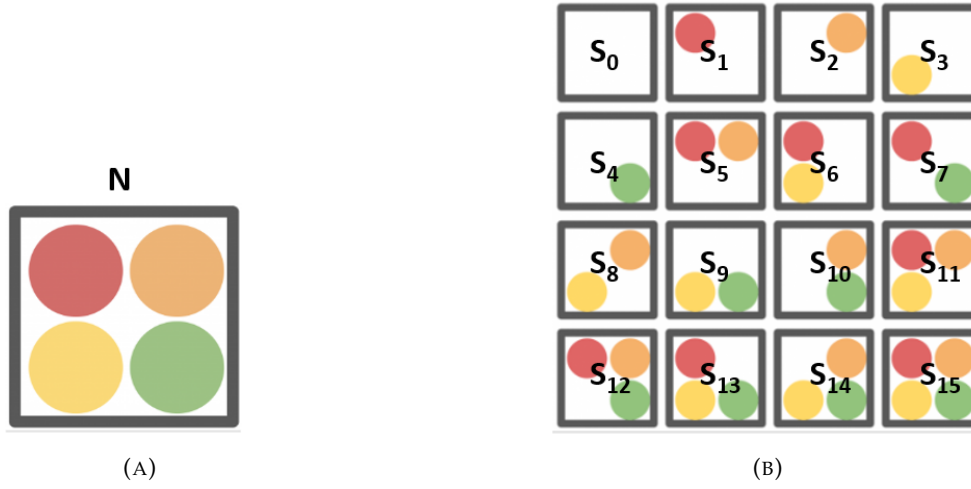


FIGURE 2.2: Every possible coalition generated (b) from an initial set of players in a cooperative game containing 4 elements represented by a color (a). (Image adapted from [77])

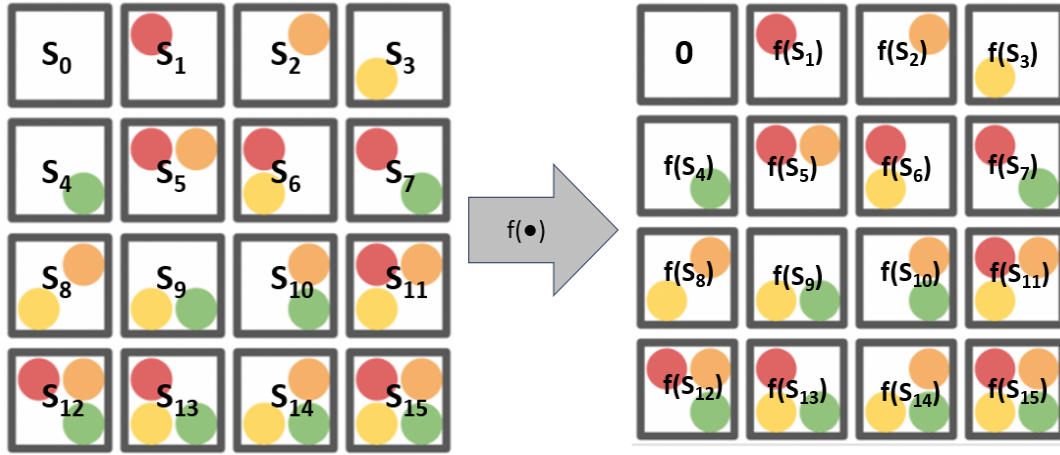


FIGURE 2.3: Evaluation of each coalition in 2.2 through the value function f .

With a better understanding of coalitions, we can start decomposing the shapley value formula. The shapley value for player i will be a weighted average over the marginal contributions, $[f(S \cup \{i\}) - f(S)]$, of adding this player to every sub-group that does not contain it. The weight factor, $\frac{s!(n-s-1)!}{n!}$, can be understood by looking at each factor individually. The denominator, $n!$, corresponds to how many ways we can form the coalition of n players by adding a player at a time. We can rewrite the numerator as $s! \cdot 1 \cdot (n-s-1)!$ and decompose it as the ways the coalition of n players can be created by first creating a

coalition of s players, $s!$, adding player i to this coalition, 1, and subsequently filling the remaining spots, $(n - s - 1)!$. As such we view this weight as the probability of having player i join a coalition of n players at the $s+1$ position when building a coalition of n players.

When we sum these weighted contributions over all the coalitions that player i can join, we obtain an expected marginal contribution of adding player i , which defines the shapley value of this player.

This leaves us with one last remaining piece to be better defined, the value function, $f(\cdot)$, that assigns a value for a coalition of players. Here, it is useful to ground ourselves to the machine learning problem. Let us say that we possess a machine learning problem where we have n features and one target variable, y . A machine learning model will attempt to model the features through a function, $g : R^n \rightarrow R$, that estimates our target variable given the values of the predictors. The value function, f , will be mapping a set of variables used, S , to this same prediction. To make the connection between these two functions we take the marginalized fluctuation of prediction over the features not included in S . So, given a datapoint, $x = (x_1, \dots, x_n)$, the value function is obtained through:

$$f(S) = \int g(x_1, \dots, x_n) dP_{x_i \notin S} - E_X(g(X))$$

Using Molnar's [20] concrete example, for a machine learning model, g , with 4 features and a data point $x = (x_1, x_2, x_3, x_4)$ we evaluate the prediction for the coalition S consisting of feature values x_1 and x_3 through:

$$f(S = \{1, 3\}) = \int \int g(x_1, X_2, x_3, X_4) dP_{X_2} dP_{X_4} - E_X(g(X))$$

It is to be noted that these contributions are, in fact, local. By obtaining multiple local values, this is expanded into global sensitivity measures by aggregating these results through averages.

SHAP expands on Shapley values by bringing a simpler and more computationally efficient way of calculating them. It does so by creating a surrogate explanation model, $h : \{0, 1\}^n \rightarrow R$, that maps a coalition vector, $\mathbf{z}$, to the output of the model. This coalition vector represents whether a certain feature is missing or not in the prediction. The form of this surrogate model is additive, following the lines of a simple linear regression, and takes the form of:

$$h(z) = \phi_0 + \sum_i \phi_i z_i$$

, where ϕ_i is the shapley value associated with feature i and ϕ_0 is the model background, a base prediction, which the presence of each feature can alter. Kernel SHAP takes this linear model and fits it by changing the value of each inactive feature in the coalition vector to a random value sampled by a given mask, for example, the values of the feature for other data points. It performs the fitting as a weighted linear regression, giving more weight to coalitions with a small number of features (highlighting independent feature effects) or to coalitions with almost all the features (highlighting the impact of all interactions) [78]. A large group of coalition vectors is then generated, with corresponding points in the feature space, and the Shapley values for each feature are estimated through the fitting of this surrogate model. The SHAP framework also provides other ways of obtaining Shapley value estimates that are model specific, like Tree SHAP for tree based models and the additive explainer for generalized linear models [75].

2.5 Robustness in ML

In computer science, robustness is associated with the ability of a given system to deal with errors during execution and erroneous inputs. In theory, a truly robust system is capable of dealing with all types of possible errors and inputs evaluated. In practice, a truly robust system is hardly achievable due to the sheer amount of possible situations that occur in the real world for which the system was not familiar with.

In the context of machine learning, robustness is associated with different scenarios a model can encounter, both in its predictive ability, as well as susceptibilities it might have regarding privacy and security. Here, robustness is usually tied with Adversarial Robustness [8], that encapsulates ways to either extract private information of the model (Intellectual Property stealing) or provoke erroneous decisions, and robust statistics, that are closely tied to improving estimators of model parameters in the context of more abnormal data distributions [79].

In the following subsections we cover these two areas where robustness is prevalent.

2.5.1 Robust Statistics

The field of robust statistics is a branch of statistics that focuses on methods and techniques that are resistant to the presence of outliers or departures from the assumed statistical model. Traditional statistical methods, such as least squares regression, can be sensitive to extreme values or violations of assumptions, leading to unreliable or biased estimates. To mitigate this issue, robust regression approaches expand on usual loss functions by introducing other loss metrics that are more resilient to abnormal observations such as outliers in the training process.

To obtain the regression parameters, an M-Estimation[80] method is used. This method is a generalization of the maximum likelihood estimation, that minimizes a given loss function rather than just maximizing likelihood. The choice of the loss function then becomes crucial in the optimization process, where, for outlier resilience, it is typically chosen to be a convex function that penalizes large residuals more than small residuals. One common choice of such a function is the Huber loss [81], a combination of squared loss for small residuals and absolute loss for large residuals that offers a compromise between the robustness of the absolute loss and the efficiency of the squared loss.

This area poses an important starting point for model robustness, as changes in the learning process through outlier resilient loss functions lead to a more aware learning process, that can connect insights extracted from unvaried analysis of the distributions of the predictors into active considerations for modeling. In general, this area takes a proactive approach to robustness, focusing more on modeling considerations than verification of robustness of pre-existing models.

2.5.2 Adversarial Robustness

Adversarial machine learning is a field of study related to creating and evading attacks to a given model, making it sturdier in real-world applications. These attacks range in scope, from threats to security of the intellectual property and user information, to generating instances that are prone to being miss-evaluated by a model. Barreno *et. al.* proposed a taxonomy for these types of attacks, by proposing three axis for its classification[82]:

1. Influence - We can classify the influence an attack has on the model as either **exploratory** or **causative**. If an attack needs manipulation of the training step of a model, via manipulating it to weaken or bias the training process, such as data poisoning [83], it is regarded as a causative attack. If, instead, the attack does not rely

on manipulating the training stage, only needing the predictions, and focuses on crafting examples to discover and exploit weaknesses of the decision process, it is regarded as exploratory.

2. Security Violation - This category is primarily aimed to models that perform a classification task. Where it is proposed that an attack can either lead to an **integrity** violation if it leads to abnormal production of false positives, or an **availability** violation if it leads to an abnormal production of false negatives. In a sense, we could think in a regression task that overshooting or undershooting predictions could replace this axis of classification of an attack.
3. Specificity - Here, the attack can either be **targeted** to a specific instance or **indiscriminate** if it targets a broad group of observations.

Although these axes are useful for gauging how an attack works at a high level, they are not really complete, as some other attacks one can perform on models are completely out of this spectrum. Such attacks include model stealing [84], where a malicious agent tries to replicate the intellectual property usually by creating a model with high fidelity to it, and membership inference [85], where an agent tries to identify whether an observation belongs to the training set.

Because of the incompleteness of this definition, another useful taxonomy for adversarial machine learning is exposed in IBM's adversarial robustness toolbox [86] presented on figure 2.4. Here, we define 4 categories for an adversarial attack:

1. Poisoning - A type of causative attack that consists of contaminating the training dataset with data designed to increase errors in the output. These sorts of attacks have a high impact on industry applications [87], as malicious manipulation of the data can undermine the entire decision process and compromise the integrity of the data product.
2. Inference - A type of attack that encapsulates the previously described membership inference approaches. These types of attacks lead to various security concerns. For example, membership inference, which is usually simpler in highly over-fitted models [88], can lead to identification of sensitive user information and raise data protection concerns.
3. Extraction - This attack focuses on obtaining information about the model in such a way that can, in extreme cases, lead to complete model stealing. Surrogate models

can be used in this context where an adversary tries to extract sensitive information or replicate a target model using limited access to its predictions or training data.

4. Evasion - These attacks consist of exploiting the imperfections of a trained model [89]. They tend to follow an exploratory influence strategy, with either targeted or untargeted specificity. They achieve their goal by creating adversarial perturbations to model inputs that lead to high disparity of model predictions, leveraging their high numerical instability. Applications of adversarial evasion attacks are common in classification problems with image data, where single pixel manipulation can easily lead to wrongful predictions.

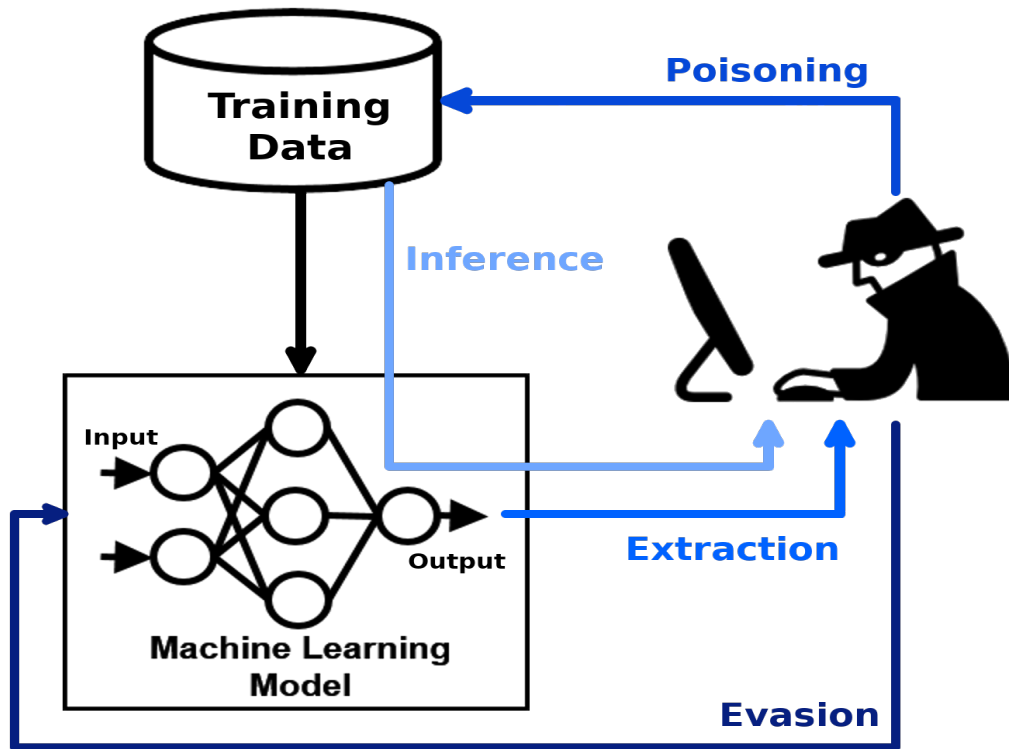


FIGURE 2.4: Adversarial attack taxonomy according to IBM's adversarial robustness toolbox [86].

Adversarial machine learning covers both specific types of attacks and defences to reduce model vulnerability to them. A simple example of such a defence is to take advantage of regularization-based defenses to reduce over-fitting mitigating the effectiveness of membership inference attacks [90].

For our work, the most relevant attack types are the evasion attacks. In a way, we can view this as a continuation of our work in weak segment detection and error analysis, but

rather than verifying model error across feature space, we explore numerical instability of our algorithms. As such, let us explore evasion more in depth in the following subsection.

2.5.3 Evasion Attacks in regression problems

As we exposed previously, adversarial examples are data points crafted in such a way as to look "normal" but cause misclassification to a machine learning model. They are achieved by perturbing inputs of a data point with some form of "noise" in order to abuse numerical instability of a model. A simple unguided way to achieve this is through covariate perturbations of the inputs, for example, using Gaussian noise injection. [91]

Gaussian noise injection adds random perturbations drawn from a Gaussian distribution to the input features. By carefully adjusting the magnitude of the noise, an attacker can manipulate the predictions of the regression model. The noise is typically generated to maximize the difference between the original and adversarial predictions. Gaussian noise can also be used as a data augmentation strategy for increasing model robustness[92], and is a common technique in image classification. An alternative to assessing robustness through covariate perturbation is also implemented in the Python Interpretable Machine Learning repository (PiML) [91], as quantile perturbation. The main idea of this perturbation is to overcome shortcomings of Gaussian noise in the case of highly skewed or discrete data. Here, we discretize each feature using its corresponding quantile values, add a uniform noise perturbation to that value, subject to a perturbation size parameter, and reobtain the new corresponding feature value after perturbation. To exemplify, let us consider a feature X_i , with its corresponding deciles being $\{q_1^i, q_2^i, \dots, q_{10}^i\}$. Given a value of this feature, x_i , we start by rounding it to the value of the nearest decile q_j , ($j \in \{1, \dots, 10\}$). We then perturb this quantile with a uniform noise, $e \sim U(-0.5\lambda, 0.5\lambda)$, with a parameter λ representing perturbation size. The new quantile corresponding to this value will be given by $q_j + e$. We round this new calculated value again to the nearest quantile considered (in this example one of the deciles $\{q_1^i, q_2^i, \dots, q_{10}^i\}$) and map it back to the corresponding value of that quantile in the feature space.

Both Gaussian and quantile perturbations have a randomness factor that can highly impact the quality of adversarial examples generated. It is possible that a small perturbation to an example leads to no significant changes in the prediction, but that same perturbation in another direction of the feature space leads to massive prediction changes. As such robustness can be assessed using more sophisticated attacks that seek to identify the

most optimal change made to an input that leads to a more impactful change in prediction. These types of attacks are regarded as white-box, when the attacker has full access to the model (model type and its learnt parameters and defined hyper-parameters), or black-box if the attacker doesn't have this information regarding the model [93].

Although in real-world scenarios, white-box attacks to ML models are less realistic, since attackers rarely have complete access to the model, in terms of assessing robustness they lead to an evaluation of the worst-case performance of a system, and therefore are the best option for in-house robustness evaluation [93]. Chen *et.al.* [93] define the general procedure for an adversarial attack in the context of multi-class classification as follows: given a model, $F : \mathbb{R}^m \rightarrow \{1, \dots, K\}$, and an observation x_0 , with $F(x_0) = y_0$, an adversarial example, x^* , is such that $F(x^*) \neq y_0$ with $x^* = x_0 + \Delta$, where Δ is a small perturbation. The perturbation Δ can be found using an optimization framework, where we try to find x^* through the following optimization problem:

$$\arg \max_{x^*} g(x^*), \quad \text{s.t. } x^* \in B(x_0)$$

, where $g(\cdot)$ is a function that measures the success of the attack, and $B(x_0)$ is a small region in the feature space surrounding the point x_0 . This region tends to be defined using an l_p norm leading to the following parameterization $B(x_0; \epsilon) = \{x : \|x - x_0\|_p < \epsilon\}$.

Most literature on these attacks tends to focus on classification problems with deep neural-networks, where the optimization step uses properties of the network to reduce the complexity of the problem (such as Fast Gradient Sign Method [94] or Projected Gradient Descent [95]). Although less studied, adversarial examples in regression problems, and defences to these attacks have also been explored in literature in the context of deep learning [96].

The optimization problem for finding adversarial examples can be adapted to generic regression problems [97], by first selecting a norm (for example l-1, or l-2), the radius ϵ , and the function $g(\cdot)$ as a suitable success metric such as the magnitude of the difference of prediction (a metric of disagreement between the prediction of the original and perturbed example). We are then just left with an optimization problem where a suitable solver, such as gradient-based methods, can lead to identification of an adversarial example.

Chapter 3

Methodology

In this chapter we expose the approach used for this work in the Smart Pairing context. We start by recalling the research questions and expanding on them given the literature review performed. First, in order to understand the research questions, we can frame them in the light of an accurate assessment of risk regarding PIA (Privacy and Impact Assessment) requirements. In order to deploy a model in production (the environment in which the model will be used to generate leads used by the business) we need to make sure that we can understand the decisions made, where the models fail, and whether model complexity is justified. Although leaping into a complex adaptive model might yield satisfactory results, we need to make sure that the more transparent interpretable-ML approaches are not sufficient to solve the problem, as they possess important properties that are advantageous for future auditing. As such, in this work we will start the process with simple interpretable-ML models and gradually increase complexity until we achieve a model that is non-interpretable. We will then compare each of the trained models in the light of the explainability methods and metrics described in the State of the Art section (Chapter 2).

Before addressing the model choices used in this problem, we provide a brief explanation of the ML context of Smart Pairing to understand what types of models can be used for this problem. The implementation strategy of Smart Paring consists of two parts:

- A **regression problem**, where given a pair of an operator and a customer we predict contracted revenue (our target variable), given by the number of contracted months multiplied by the monthly value contracted;

- A **ranking problem**, where given a model to predict contracted revenue, we predict the target variable for each pairing possible for a certain customer and return a custom list of the best operators to address the customer.

The regression problem is where explainability can add value. By guaranteeing that we fundamentally understand the predictions, we are not only able to comply with risk requirements, but are also able to use knowledge extracted from explanations to propose changes that can positively affect the company, such as providing training to the operators that have lower performances. The ranking part is purely done as a consequence of the ML regression model and, as such, understanding how the regression decisions are made also leads to an understanding of ranking decisions.

The types of interpretable-ML models considered in this work are generalizations of linear regressions, where at each new model, some of the constraints imposed in the previous model are relaxed for a higher search space. These models are, in increasing order of complexity: Linear Regression, Generalized Linear Models, Generalized Additive Models, and Explainable Boosting Machines. For our explainable-ML approach, we opt for the widely used eXtreme Gradient Boosting Trees. The models used and their properties will be explained in depth in the following chapter (Chapter 4).

For comparing each of the selected models, we use a methodology based on answering each of the proposed research questions, with the knowledge exposed in the State of the Art section (Chapter 2). Figure 3.1 exposes a high-level overview of said methodology.

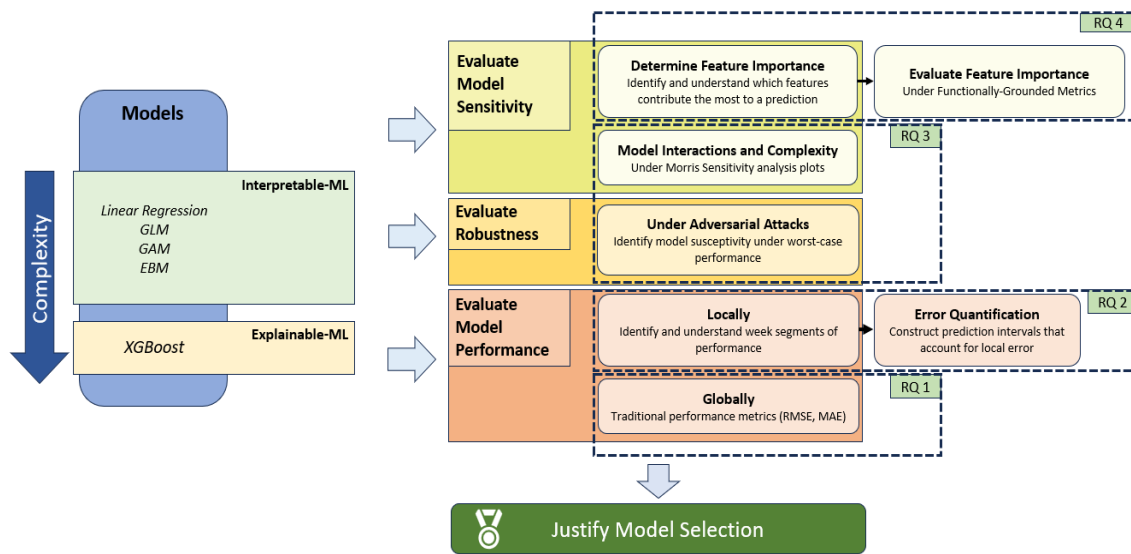


FIGURE 3.1: High-level overview of the methodology employed for comparison and selection of models. (RQ: Research Question)

Before the training and assessment of the models, an initial exploratory analysis of the data was performed in order to determine key characteristics of our dataset and help define the pre-processing pipeline. Each model is then trained on a training dataset, with hyperparameter tuning performed by using a grid-search in a cross-validation fashion. For evaluation of model performance, we first obtain global performance metrics, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), in a separate test set for assessing generalization error of the models.

For finding the local error, we follow an approach similar to Microsoft’s Error Analysis Package [98], where we use regression trees fitted on the absolute residuals of the prediction in order to locate the regions of different error (tree leaves). We can interpret these results by studying each cohort of different errors and extracting insights regarding model error. We then expand each one of the regressors into a conformal regressor, allowing for a prediction interval with a certain coverage, rather than a point prediction, in order to make each prediction take into account the different error regions. We assess coverage locally in each region to make sure our conformal predictors maintain an actual coverage near the target imposed.

For evaluating robustness we use an adversarial attack in the l-2 fashion (euclidean distance) using the differential evolution algorithm implemented in scikit-opt [99]. We do so by sampling points in each different error region extracted in the error analysis step, and finding artificial data points around a perturbation radius that lead to the highest increase in error. We then compare the average error of the adversarial points for each of the models.

Finally, the sensitivity analysis is performed using the Morris Method [57] and SHAP [54] [75], for extracting feature importances of each predictor. We use these importances to understand model behaviour, and use the monotonicity of feature importances as a measure of their quality and compare this metric across each of the models.

At the end, we are left with a high number of metrics to justify model selection in a responsible way, by empirically verifying whether an increase in model complexity comes at costs or gains in the proposed metrics when compared with simpler and more transparent approaches. We comment on the results obtained in the conclusion section (Chapter 6).

Chapter 4

Models

Rather than starting with complex Explainable ML models and work our way up the interpretability ladder, we propose an opposite approach for the modeling, starting from high interpretable models and expanding on additive models if justified by performance: from Linear Regression to Explainable Boosting Machines. As we move up, complexity of the models also increases but we maintain some important properties throughout, them being: additive components making it easy to understand feature contributions to the prediction, and a control over feature interactions, either by limiting them or not considering them at all.

Alongside the additive models, we also consider an Explainable ML model, Extreme Gradient Boosting. This will act as a way to compare our results with the path of not taking any model interpretability measures.

We propose an initial qualitative assessment of the models used, table 4.1, following the structure of Sudjianto *et al.* [21]. It is to be noted that there is some subjectivity associated with this classification, as the metrics themselves are not quantitative.

Models	Additivity	Sparcity	Linearity	Smoothness	Monotonicity	Visualizability
Linear Regression	****	***	****	****	****	****
GLM	****	***	****	****	***	****
GAM	****	***	**	***	***	****
EBM	****	***	*	**	***	***
XGBoost	*	**	*	**	***	**

TABLE 4.1: Qualitative Assessment Template for Model Interpretability (Global).

Ratings are given in accordance to [21] such that:

* - model cannot satisfy the constraint

** - model can be roughly constrained

*** - model can be easily constrained

**** - model automatically satisfies the constraint

Before introducing a revision of the models used in this work, we define some notation that will be used throughout this section. Here we consider a machine learning problem with M examples in a training dataset. Each one of our predictors, $\mathbf{x}^{(p)}$, is a vector with N components such that $\mathbf{x}^{(p)} = (x_1^p, \dots, x_N^p)$, with $p \in \{1, 2, \dots, M\}$ (for examples in the training set) being an identifier of the example from the sample. Each one of our predictors, $\mathbf{x}^{(p)}$, has a corresponding target variable, $y^{(p)}$, and we seek to model this relation. The estimator of our target variable will be denoted as $\hat{y}$.

4.1 Linear Regression

Linear Regression is a type of statistical model that maps N predictor variables, $x_i^{(p)}$, with $i \in \{1, 2, \dots, N\}$, with a scalar response, $y^{(p)}$, using a linear relationship between them. As such, the estimation of our predictor is given by:

$$\hat{y}^{(p)} = \beta_0 + \sum_{i=1}^N \beta_i x_i^{(p)}$$

Typically, parameters β are estimated using Ordinary Least Squares (OLS), so that $\beta = \arg \min_{\beta} S(\beta)$, where $S(\beta) = \sum_{j=1}^M (y^{(j)} - (\beta_0 + \sum_{i=1}^N \beta_i x_i^{(j)}))^2$. The estimation of these parameters using OLS assumes some conditions are met [20], namely:

- **Linearity:** The mean of the response variable, given the predictors, can be described as a linear combination of said predictors;
- **Fixed features (Weak exogeneity):** The predictor variables are considered fixed and not statistical variables, implying that they are free of measurement errors;
- **Normality:** The target variable, given our predictors, $y|x$, follows a normal distribution;
- **Homoscedasticity (constant variance):** Such that the normal distribution of $y|x$ has some constant variance σ^2 ;
- **Independence:** Assuming that there is no dependency relation between observations (data points);
- **Absence of perfect multicollinearity:** So that no predictor can be written as a linear combination of other predictors, which would affect the estimation of the weights.

Other methods for estimating the linear coefficients have been proposed in order to circumvent some of the problems that follow from these restrictions, namely due to high correlated predictors. Ridge Regression adds to the loss function $S(\beta)$ a regularization term with the norm of the weights (L2 regularization), such that $S_{Ridge}(\beta) = \sum_j^M (y^{(p)} - (\beta_0 + \sum_{i=1}^N \beta_i x_i^{(p)}))^2 + \alpha \|\beta\|_2^2$. Another example would be Lasso Regression that adds a penalty term with L1 regularization.

4.2 Generalized Linear Models

Generalized Linear Models, GLM, were proposed by John Nelder and Robert Wedderburn [100] as an extension of the already existing linear regression. Rather than predicting the target variable using a simple linear combination of the predictors, GLM allows for a more general approach by allowing a linear prediction to be transformed by a function before obtaining the predicted value. The general formulation of the GLM is given by:

$$g\left(\mathbb{E}\left(y^{(p)}|\mathbf{x}^{(p)}\right)\right) = \beta_0 + \sum_i^N \beta_i x_i^{(p)}$$

This formulation can be decomposed into 3 components, as described by McCullagh and Nelder [101]:

- The systematic component, being the result of the linear combination of the predictors: $\beta_0 + \sum_i^N \beta_i x_i^{(p)}$. This component will always yield the same result if two data points have the same values, $\mathbf{x}^{(p)} = \mathbf{x}^{(p')} (p \neq p')$;
- The random component, $\mathbb{E}\left(y^{(p)}|\mathbf{x}^{(p)}\right)$, being the expected value of our target variable for a given data point. The randomness comes from the fact that, for some specific $\mathbf{x}^{(p)}$, we can think that $y^{(p)}$ is then sampled from a conditional distribution. In practice this means that if two data points have the same values, $\mathbf{x}^{(p)} = \mathbf{x}^{(p')} (p \neq p')$, they might have different values for the target variable;
- The link function, $g(\cdot)$, that maps our expected result (random component) with the responses from the systematic component.

The systematic component, sometimes called linear predictor [20], contains the weights, β_i , which are the parameters learnt in the modeling task. For the random component in

a generalized linear model, each outcome, $y^{(p)}$, of the dependent variables, $\mathbf{x}^{(p)}$, is assumed to be generated from a particular distribution in the exponential family given by the following expression:

$$y^{(p)}|\mathbf{x}^{(p)} = Y^{(p)} \sim f(Y^{(p)}; \theta^{(p)}, \psi) = \exp\left(\frac{Y^{(p)}\theta - b(\theta^{(p)})}{a(\psi)} + c(Y^{(p)}, \psi)\right) \quad (4.1)$$

Here $\theta^{(p)}$, ψ are parameters and a, b, c are functions. The parameter $\theta^{(p)}$ is called the natural parameter (or canonical parameter[102]), and the parameter ψ is the dispersion parameter. We consider that each $Y^{(p)}$ follows a distribution of the same kind, however with different parameters $\theta^{(p)}$ relating to the values of $\mathbf{x}^{(p)}$. Although we allow this consideration for the canonical parameter, we fix the dispersion parameter [103]. Here, we can recall a simple one dimension linear regression example to build the intuition, that would be modeled by $Y^{(p)} = \beta_0 + \beta_1 x^{(p)} + \epsilon$, where a noise factor, ϵ , follows $\epsilon \sim N(0, \sigma^2)$, making $Y^{(p)} \sim N(\beta_0 + \beta_1 x^{(p)}, \sigma^2)$.

Many distributions fall under this category such as Normal, Exponential, Gamma, and Poisson distributions. The choice of distribution relates to the data characteristics of the target variable. If the target variable is a count, for example, then a suitable choice of distribution could be the Poisson distribution. If we are dealing with only positive data, one choice could be the Exponential distribution. The support of the distribution chosen should match the domain of our target variable.

For all considered distributions of $Y^{(p)}$ there exists at least one canonical link function and, typically, a set of frequently used link functions [102]. A link function is called a canonical link function if it links the expected value of our target variable, given the predictors, $\mathbb{E}(y^{(p)}|\mathbf{x}^{(p)})$, with the canonical parameter $\theta^{(p)}$ such that $g\left(\mathbb{E}(y^{(p)}|\mathbf{x}^{(p)})\right) = \theta^{(p)}$. With this choice of link function, $\theta^{(p)} = \beta_0 + \sum_i^N \beta_i x_i^{(p)}$, meaning now that each canonical parameter can be written as a function of the coefficients β_i .

Now, considering a distribution $f(Y^{(p)}; \theta^{(p)}, \psi)$ and a canonical link function, optimization of the coefficients β_i follows first from a maximum likelihood estimation of the parameters $\theta^{(p)}$. Consider the likelihood function of $\boldsymbol{\theta} = (\theta_1, \dots, \theta_M)$, given a sample of examples $S = \{(\mathbf{x}^{(p)}, y^{(p)}); p \in \{1, 2, \dots, M\}\}$:

$$L(\boldsymbol{\theta}; S) = \prod_{p=1}^M f(y^{(p)}; \theta^{(p)}, \psi)$$

As such, the log-likelihood will be given by:

$$\log(L(\theta; S)) = l(\theta; S) = \sum_{p=1}^M \frac{y^{(p)}\theta^{(p)} - b(\theta^{(p)})}{a(\psi)} + \sum_{p=1}^M c(y^{(p)}, \psi)$$

Since we will be differentiating the expression to obtain a maximum with respect to the parameters θ , the factor of the sum over $c(y^{(p)}, \psi)$ will disappear under differentiation and the constant $a(\psi)$ will disappear when we make the derivative equal to 0. As such let us only consider those factors under:

$$l^*(\theta; S) = \sum_{p=1}^M (y^{(p)}\theta^{(p)} - b(\theta^{(p)}))$$

Recalling that under the canonical link function, $\theta^{(p)} = \beta_0 + \sum_i^N \beta_i x_i^{(p)}$, we can actually make this a function of the parameters β . We then use this for a basis for optimizing the weight parameters, finding those that maximize $l^*(\beta; S)$ using an iteratively reweighted least squares algorithm.

$$l^*(\beta; S) = \sum_{p=1}^M (y^{(p)}(\beta_0 + \sum_i^N \beta_i x_i^{(p)}) - b(\beta_0 + \sum_i^N \beta_i x_i^{(p)}))$$

4.2.1 Practical use cases

When using this model, we need to take into consideration just two aspects. One is the distribution for the response variable $Y^{(p)}$ and the other is the link function. We can select the distribution using insight on the type of response that we are expecting, given some predictors: knowing whether we are working with a classification problem, a regression problem with only positive values, etc. We can then, knowing the distribution selected, use the corresponding canonical link function. This can be obtained by rewriting the distribution in the exponential family such as in (4.1) and writing the canonical parameter as a function of the expectation of the distribution, $\theta^{(p)} = g\left(\mathbb{E}(y^{(p)} | \mathbf{x}^{(p)})\right)$. However, most of these link functions are already derived and can be easily consulted for well known distributions in the exponential family. The table 4.2 shows some examples of canonical link functions for some distributions.

Distribution	Canonical Link Function
Bernoulli $B(\eta)$	$\log \left(\frac{\mu}{1-\mu} \right)$
Binomial $B(k, \mu)$	$\log \left(\frac{\mu}{k-\mu} \right)$
Poisson $P(\mu)$	$\log(\mu)$
Exponential $Exp(\mu)$	$\frac{1}{\mu}$
Gamma $G(\mu, \psi)$	$\frac{1}{\mu}$
Normal $N(\mu, \psi^2)$	μ
Inverse Gaussian $IG(\mu, \psi^2)$	$\frac{1}{\mu^2}$

TABLE 4.2: Distributions and canonical links [102]

4.3 Generalized Additive Models

Generalized Additive Models, GAM, work as an extension of GLMs, relaxing the condition of linearity in the systematic component but maintaining the overall structure. Here, rather than taking a linear combination of the predictors, we now consider that the outcome will be modeled by arbitrary functions of the predictors $f(\cdot)$, that are not enforced to be linear. As such, the mathematical formulation of the model is described by:

$$g \left(\mathbb{E} \left(y^{(p)} | \mathbf{x}^{(p)} \right) \right) = \beta_0 + \sum_i^N f_i(x_i^{(p)})$$

There are many ways to define the general smooth functions $f(\cdot)$, using non-parametric approaches. One common way to do so is using a scatterplot smoother. This approach, described more in depth by Hastie *et. al* [104], divides the predictor space for a given $x_i^{(p)}$ into multiple sections connected by knots and, using a combination of basis functions in each section (such as cubic smoothing splines) we are able to estimate $f(x_i^{(p)})$. At the knots, continuity is enforced in order not to have discontinuous abrupt jumps in the prediction.

In general, not all of the functions $f(\cdot)$ need to be nonlinear. We can easily mix in linear functions like the ones used in GLM for some predictors and non linear functions for others. This is useful for when some of the variables are categorical variables (factors) [104].

The loss function used for optimization can also be updated to include a term that penalizes high acceleration/deceleration of $f(x_i^{(p)})$, meaning we prioritize a smooth continuous function over a continuous function with a lot of abrupt increases/decreases. The traditional RSS can be replaced with Penalized Residual Sum of Squares (PRSS):

$$PRSS = RSS + \sum_{i=1}^N \lambda_i \int f_i''(t)^2 dt$$

Where $\lambda_i > 0$ are regularization factors that control the strength of the regularization.

To build intuition on how these functions are built from basis functions, in order to model a target variable with a predictor, we can consult figure 4.1 [105], containing a practical example, displaying both the basis functions used and the end model prediction.

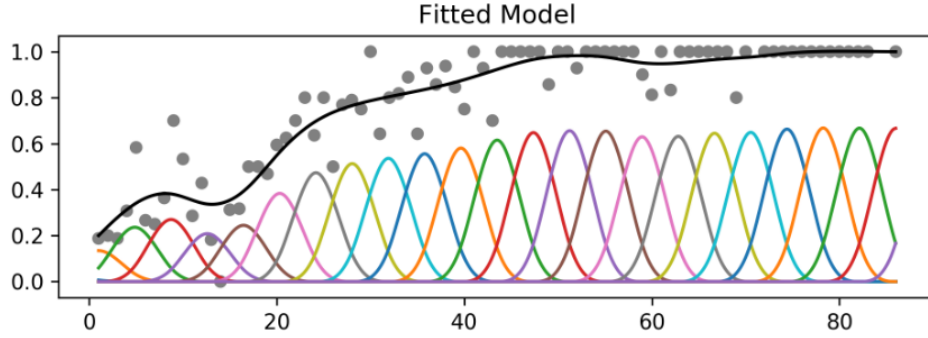


FIGURE 4.1: PyGAM example of fitting a GAM to data. The splines are represented as the coloured lines and the continuous dark line represents the function built on the combination of them. (figure from [105])

4.4 Explainable Boosting Machines

Explainable Boosting Machines, EBM, are a type of Generalized Additive Model, implemented in InterpretML [106], based on the Generalized Additive Model plus Interactions (GA²M) algorithm [107]. Here, the general form of GAMs is extended to include pairwise feature interaction for some group of features $i \in S$, such that:

$$g\left(\mathbb{E}\left(y^{(p)}|\mathbf{x}^{(p)}\right)\right) = \beta_0 + \sum_i^N f_i(x_i^{(p)}) + \sum_{\substack{i>j \\ i,j \in S}}^N f_{i,j}(x_i^{(p)}, x_j^{(p)})$$

The main two details about the implementation are the way that feature functions, $f_i(\cdot)$, are estimated and the selection of the features to interact, S . Regarding the feature functions, rather than using some basis functions for estimation, EBMs use modern machine learning techniques such as bagged boosted trees described in this context by Lou *et. al* [108]. The main idea is to use a round robin pass through the features with a small learning rate so the order of the features does have a high impact on the result. We can think of this process as ordering our features in some manner, for example $\{1, \dots, N\}$, and start by fitting a shallow tree (weak learner) for feature 1 to predict our target variable. We

then pass the residual values to feature 2 and train another shallow tree using only that feature, update the residuals and repeat the steps until we have N models, one for each predictor. After this we update the residuals and pass them back to feature 1 repeating the entire process a given number of iterations until either convergence or a certain stopping criteria is met.

The second nuance of this algorithm relates to how S , the set of feature interactions to model, is determined. The main idea behind the process consists on considering the predictions made by the best additive model so far (obtained by training the one dimensional feature functions) and detect interactions in the residuals. This is performed in a greedy fashion, iterating through possible pairs of features, building a shallow tree using the pair and updating the model with the best model for the residual values. We can update the residuals with the newly found best additive model, now containing a factor with an interaction between two predictors. We can repeat this process through the remaining pairs until some convergence condition is met and we are left with the set S of feature interactions considered. EBM contains a fast implementation of this algorithm, taking advantage of parallelization and multiprocessing capabilities [106].

In the end, we are left with a model that maintains interpretability due to the fact that we can understand and visualize how a predictor influences the prediction by plotting both the feature functions and feature interaction terms containing it.

4.5 Extreme Gradient Boosting

Extreme Gradient Boosting, XGBoost [109], is a state-of-the-art tree based boosting algorithm that gained a lot of popularity due to the great performance in many machine learning competitions [109][104][110]. Like other gradient tree boosting algorithms, the model is constructed by initially starting with a constant prediction, calculating pseudo-residuals for said prediction, following with iteratively fitting weak learners on the pseudo-residuals and updating them for the next weak learner. At each step of the iteration, the model is updated by adding the prediction of the trained weak learner, scaled by a factor γ , that minimizes a regularized loss between the true value of the target variable and $F(\mathbf{x}) + \gamma h(\mathbf{x})$, where $F(\mathbf{x})$ is the decision function learnt in the previous iteration and $h(\mathbf{x})$ is the tree learnt in the current iteration [111][104].

One of the main advantages of XGBoost, that makes it appealing to use in real world applications, is the scalability it offers. Alongside the use of distributed computing, XGBoost also implements approximate tree learning with a weighted quantile sketch for fitting the weak classifiers, which leads to similar results to the classical complete explorations of possible branch cuts, but in much less time. The split finding algorithm is also sparsity aware, making the training faster at dealing with a very sparse predictor matrix, common when employing pre-processing strategies like one-hot encoding [109][112][110].

Additionally, XGBoost allows for the introduction of monotonic constraints. These can be implemented when there is a need or strong belief that the increase in the value of some predictors should result in an monotonic increase or decrease in the target variable, regardless of the value of other predictors. These restrictions to the modeling step can increase model interpretability as:

- The existence of a monotonic constraint in a variable allows reasoning regarding the behaviour of the prediction with that variable independently, which builds a degree of intuition regarding the model predictions [113];
- The ability of imposing these restrictions during the modeling phase allows for the introduction of known causality relations, reducing the search space to find solutions that agree with real world understanding of the data [114].

Chapter 5

Experimental Environment and Results

5.1 Data Understanding and Data Analysis

Before delving into details regarding model training and subsequently approach each research question, we start by having a quick overview of the data for better comprehension of the machine learning problem. In this subsection, we go through a dataset overview, focusing on: comprehending the type of data available, some typical behaviour of the predictors, the distribution of the target variable, uni-variate and bi-variate analysis.

In the regression phase of the Smart Pairing recommendation problem each data-point represents a match between an operator and a customer. The target variable is the contracted revenue closed on the call multiplied by the number of months remaining in the contract, representing the total value a given customer is expected to pay for the active contract. Accurately predicting this value, for a pair of operator and customer, results in understanding the yield of different matches of a given customer and the available operators. The predictors in this machine learning problem contain information regarding 3 main groups of characteristics:

- Operator historical performance (OHP) - where we measure volumes of Service Requests answered by each operator over a certain period in time, both as total volume and segmented by categories. These categories can regard the type of technology or products associated with each request, as well as the type of resolution the request had. We also include information regarding the team, function, direction, and area that a certain operator has within the company.

- Customer historical service requests (CHS) - where we measure volume of Service Requests the customer made regarding their service, over certain aggregation periods.
- Customer active bundles (AB) - where we use information regarding the types of technology, number of SIM cards, television service, and other relevant details of the service provided by the company that a given customer possesses.

Each variable used for prediction falls in one of these groups with the exception of the average revenue per unit (ARPU) at time of call for a customer, that complements the information on the client’s active bundles. Table 5.1 condenses some general information of the dataset, including the division of predictors in each of the described categories.

Number of observations	Number OHP predictors	Number CHS predictors	Number of AB predictors
299178	92	4	28

TABLE 5.1: Characteristics of the dataset

We possess a continuous numerical target variable that represents the total contracted value after the service request over the contracted period. If we look at the distribution of these values, figure 5.1, the first thing we notice is a spike on the value of 0. The reason for this high spike is justified by the nature of the data, since it represents the use case where a customer abandons the service, terminating the contract and thus having no contracted revenue after the call. We possess an indicator that segments each service request regarding the type of resolution it had that we do not use as a predictor since it refers to information not available at time of inference, that we can use to justify this reasoning. If we use this indicator to exclude the service requests that ended in churn of the customer, we can visualize a distribution of our target variable that no longer contains this abrupt spike at 0, figure 5.1. At time of model training, we include these churned examples in order to remain coherent with the reality of the data at time of inference.

Our dataset contains a mix of numerical and categorical predictors. The majority of the predictors have a numerical nature, with only 9 of them having a categorical nature. Out of these 9, 5 of them are related to information regarding the operator (OHP), representing their position in the company’s internal structure, and their status, that is, if they are active or not in the position. The remaining 4 categorical variables refer to kinds of products in the customer portfolio (AB). The number of categories in each of these features

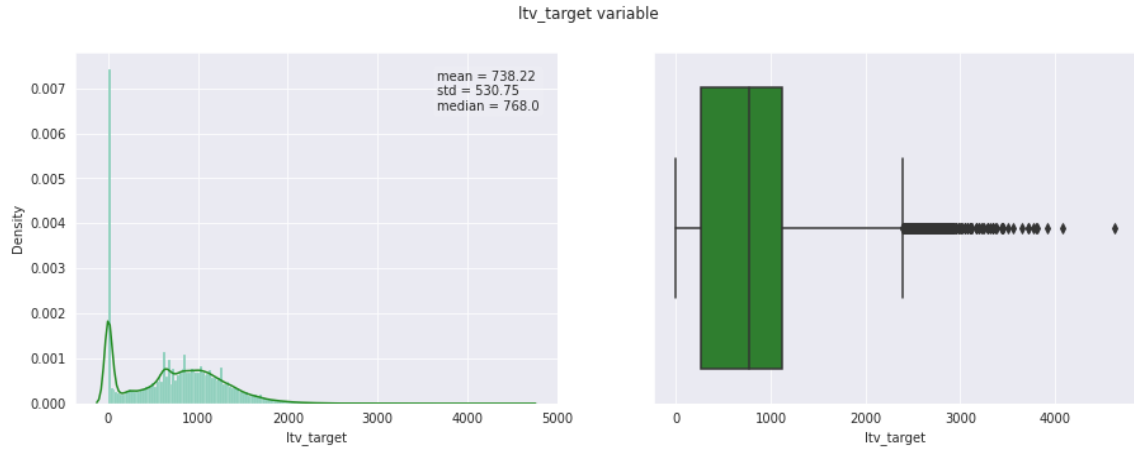


FIGURE 5.1: Distribution of Smart Pairing target variable

varies but infrequent values are prevalent in each one. Figure 5.3 exposes the proportion of each category (either in totality or top and bottom 5 most prevalent categories in the case of high number of categories) confirming the existence of infrequent categories. The categories in each predictor are masked to preserve data confidentiality.

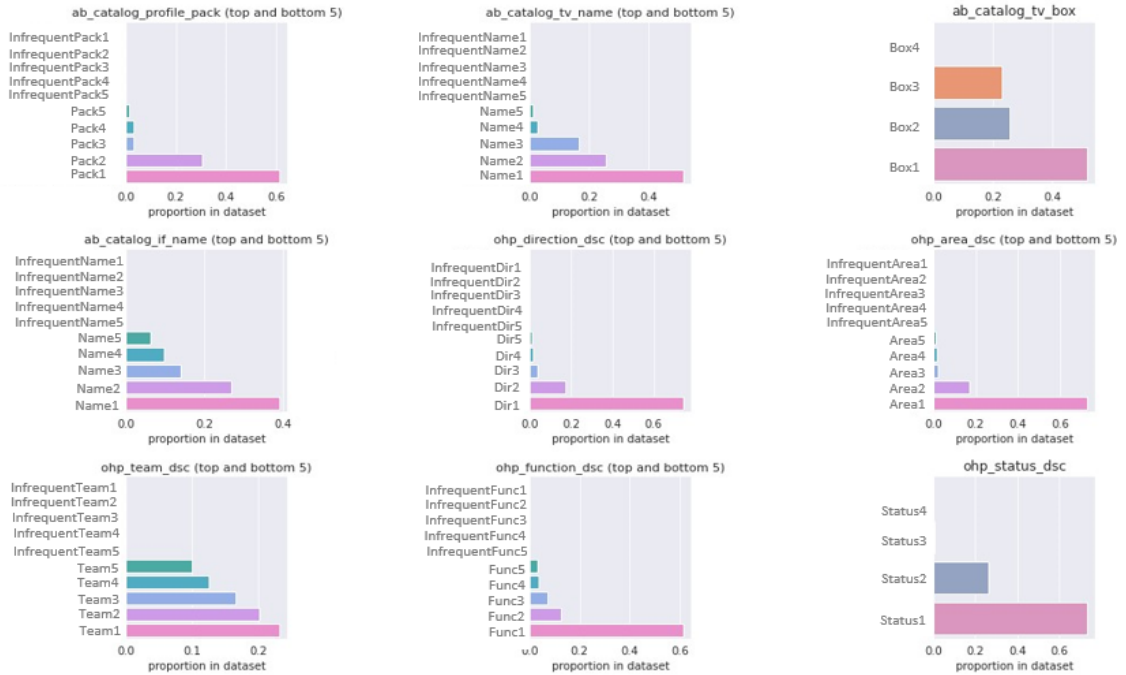


FIGURE 5.2: Proportion of categories in categorical predictors of Smart Pairing

In order to comprehend the impact of these categories in the target variable we performed one-way ANOVA hypothesis test [115], with the intent of verifying whether each variable divides our data into groups (for each internal category) with different population means of the target variable. From the p-values obtained ($p < 0.05$ for all categorical

variables) we can reject the null hypothesis that all categories in the features come from a distribution with the same mean, and conclude that they follow different distributions. This indicates that the predictors contain information that segments the data into groups that have different distributions, which can be used by the models to predict the target.

Due to the high amount of numerical predictors, an unvaried analysis of each one individually is not practical to present in this document. Instead, to provide information on the behaviour of these predictors we present a plot containing information regarding the amount of outliers in each variable to justify the use of median imputation rather than mean imputation for the missing values (since the median is less affected by the high number of outliers).

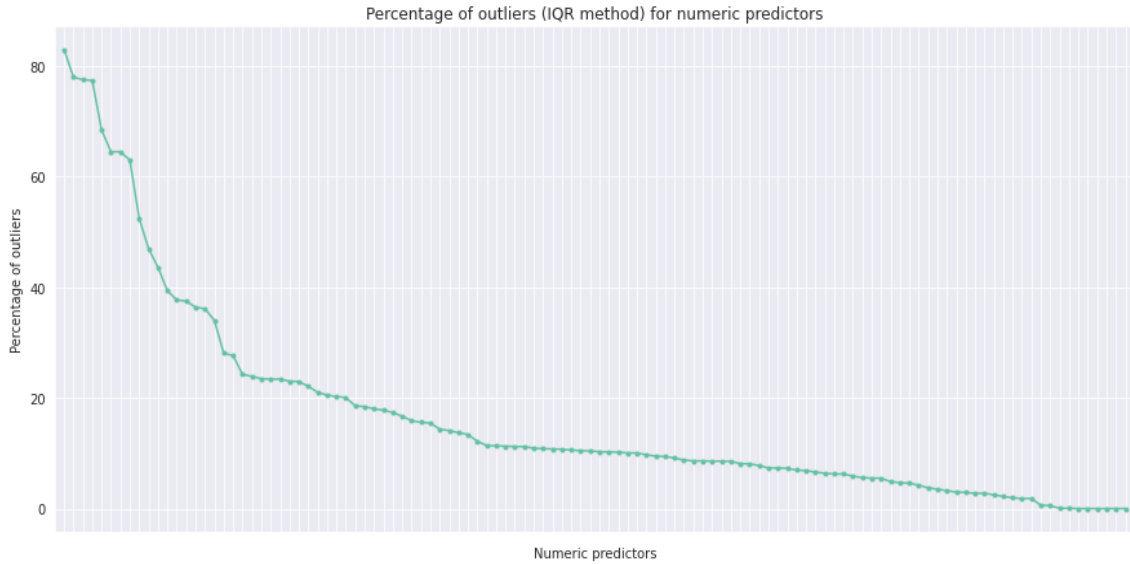


FIGURE 5.3: Proportion of outliers in numerical predictors of Smart Pairing

For assessing internal dependencies of the predictor variables we calculated the Variance Inflation Factor (VIF) [116]. This factor is calculated, for the i -th predictor belonging to the set of predictors P , by using $P \setminus \{i\}$ predictors to model a linear regression to it. We subsequently use the coefficient of determination R_i^2 to calculate this factor, given by $VIF_i = \frac{1}{1-R_i^2}$. Using this method we determined a sub-set of variables with perfect multicollinearity (infinite VIF due to $R_i^2 = 1$), meaning they can be written as a linear combination of other existing predictors. This occurs due to the existence of predictors that track the total amount of calls an operator answered in a given aggregation period, and predictors that track this volume segmented in different categories, so that the sum of this volume over the categories leads to the volume over the aggregation period. As such, it is advisable to drop the columns tracking the total volume over an aggregation

period (ohp_t_srs_3m, ohp_t_srs_6m, ohp_t_srs_12m) when dealing with models that are negatively impacted by having perfect multicollinearity. Eliminating extremely correlated features also helps enhancing explainability, as methods such as feature attributions will not have its effects spread out across multiple highly correlated features.

To conclude the analysis, we also obtained the Pearson correlation between each numerical predictor and the target, obtaining the highest absolute correlation for the predictor related to the number of voice mobile cards a client has (ab_catalog_vm_cards - absolute correlation of 0.55; correlation of 0.55), followed by the quantity of services active in a customer (ab_services_qty - absolute correlation of 0.43; correlation of 0.43), and the cost of the customer active service (ab_catalog_pack_revenue_base - absolute correlation of 0.41; correlation of 0.41). For the remaining predictors, the correlation with the target is much lower (values $\in] - 0.30, 0.30[$). These correlation values seem to indicate that if we consider each isolated predictor, there is a stronger linear relation between variables related to the customer information. We also can infer the absence of severe leakage between predictors and target, as there is no predictor that perfectly predicts our target variable.

5.1.1 Pre-Processing Pipeline

For defining the pre-processing pipeline used for data preparation we used knowledge extracted from the data analysis stage. Here we followed the following procedure:

- removed unique identifiers of customer and operator;
- removed features with high number of missing values ($> 90\%$)
- removed columns tracking the total volume over an aggregation period (ohp_t_srs_3m, ohp_t_srs_6m, ohp_t_srs_12m) due to perfect co-linearity with other variables
- performed one-hot encoding of categorical variables, keeping only the most relevant categories (top x most frequent categories, where x was determined for each individual feature)
- performed standard scaling for numerical variables, as some of the selected models can be highly influenced by variables with different scales;
- used median imputation for remaining missing values.

For determining the number of categories to keep in each categorical feature, we fitted a simple XGBoost model (default hyperparameters) using each individual feature encoded for the top x categories. We visually determined the point to which encoding further categories lead to no significant gain in performance (Appendix A). This lead to keeping:

- the 9 (8+ infrequent) most prevalent categories for the type of package a customer possesses (*ab_catalog_profile_pack*);
- the 5 (4 + infrequent) most prevalent categories for the type of tv product a customer possesses (*ab_catalog_tv_name*);
- the 3 (2 + infrequent) most prevalent categories for the type of tv box a customer possesses (*ab_catalog_tv_box*);
- the 7 (6 + infrequent) most prevalent categories for the type of internet a customer possesses (*ab_catalog_if_name*);
- the 5 (4 + infrequent) most prevalent categories for the direction to which an operator reports to (*ohp_direction_dsc*);
- the 12 (11 + infrequent) most prevalent categories for the area an operator belongs to in the company (*ohp_area_dsc*);
- the 14 (13 + infrequent) most prevalent categories for the team an operator enrolls in the company (*ohp_team_dsc*);
- the 13 (12 + infrequent) most prevalent categories for the function an operator is assigned to in the company (*ohp_function_dsc*);

5.2 Model Training and Aggregated Performance

After defining the pre-processing pipeline, we performed a train-test split using the hold-out method, keeping 80% of the data for training and 20% for testing. The pre-processing pipeline was then fitted in the train-set, an the data transformed accordingly (scaling and encoding procedures). Each of the proposed models was first trained in a 5-fold cross validation fashion, in order to tune their hyper-parameters using a grid search. A more sophisticated method such as Bayesian hyper-parameter optimization could also have been implemented, however due to time constraints we opted for the grid-search approach,

with parameters typically spanning over some orders of magnitude or typical values for a certain model. The following grid parameters were considered in this step (selected ones in bold):

- Ridge Regression - α (regularization parameter) $\in \{0.1, 1.0, \mathbf{3}, 5, 7, 10\}$
- GLM - α (regularization parameter) $\in \{1, \mathbf{0.1}, 0.01, 0.001\}$, p (power of the tweedie distribution) $\in \{0, \mathbf{0.03}, 0.06, \dots, 2.97\}$, logarithmic link function.
- GAM - λ (regularization parameter) $\in \{\mathbf{0.01}, 0.1, 1, 10\}$, linear terms for binary variables, 20 splines for continuous numerical predictors.
- EBM - learning rate $\in \{\mathbf{0.1}, 0.01, 0.001\}$
- XGBoost - learning rate $\in \{\mathbf{0.1}, 0.2, 0.3, 0.4\}$, maximum depth $\in \{4, 6, \mathbf{8}\}$

After defining the model hyper-parameters, we retrain the models using the entire train set and the best performing parameters. In table 5.2 we expose the aggregate performance of each of the trained models in the separate test set, using as performance metrics the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We verify that increasing model complexity affects positively the performance of the models in this context, leading us to conclude that the less complex models do not sufficiently model all the relations between predictors and target variable.

Metric	Ridge Regression	GLM	GAM	EBM	XGBoost
RMSE	371.6	372.6	366.3	353.1	335.5
MAE	283.7	285.1	278.1	260.2	237.6

TABLE 5.2: Model Aggregated performance in the test set

5.3 Sensitivity Analysis

In order to analyse the influence of each variable in the trained models we performed both Morris sensitivity analysis and obtained shapley values for feature contributions using SHAP. In this section we go over the obtained results and use them to understand model behaviour. We then performed a quantitative assessment of explanator quality using the monotonicity of feature attributions metric described in 2.4.1.

Before delving into the results for each of our models of increasing complexity, it is useful to expose how the metrics were obtained. In order to obtain the SHAP values, due to the high computational cost of obtaining the values for each observation, we sub-sampled 1000 points from the test set, obtained the corresponding SHAP values, and averaged the absolute values for each feature in order to obtain a global feature attribution for each predictor. We also use the SHAP method that best suits each model to ease computational strain: Linear SHAP for Ridge Regression, Additive SHAP for GLM and GAM, Kernel SHAP for EBM, and TreeShap for XGBoost. For the Morris method we use 1000 trajectories with 10 levels (each feature is divided into 10 segments) and extract the domain of each variables from the training set. Here a few limitations are introduced in order to maintain this practical experiment, namely on the selection of parameters for the Morris Method, where there is the risk that nuanced effects will not be captured with our grid. The sub-sampling of the test-set for the SHAP attributions can also not be completely representative of the entirety of the population, and even the different SHAP methods to obtain the Shapley values might introduce inconsistencies when comparing data.

Figures 5.4, 5.5, 5.6, 5.7, and 5.8 expose the importance of features for each trained model using each methodology. On each image we expose the bottom and top 5 most important features as well as the importance of each one. The importances are sorted by the sum of normalized FI (for a given method and a predictor i , $F_i \rightarrow \frac{F_i}{F_\alpha}$, where $F_\alpha = \max_{j \in \text{Features}} F_j$) of both methods.

In our Ridge Regression model (Figure 5.4) we observe a relatively high level of discordance between the attribution methods. We also observe a high importance of the variable *arpu_in_amt*, meaning that the initial value of the customer package has a high impact on the revenue expected from a customer after the call. Some details regarding the type of active service of the customer also show a strong impact on the model, namely the ones regarding the type of internet specifications (upload and download speeds). We also see high importance of features regarding operator performance (the number of successful service requests over the last 3 months, *ohp_t_srs_3m_res_rca*, and the volume of calls answered for a specific technology, *ohp_t_srs_3m_ctl_tech_dth*), which reinforces the impact of the choice of operator on the result of the SR. These most relevant operator features are all obtained using the shortest aggregation period, meaning that the model is favouring the more recent performance metrics over the long term metrics (with 6 and 12 months as an aggregation period).

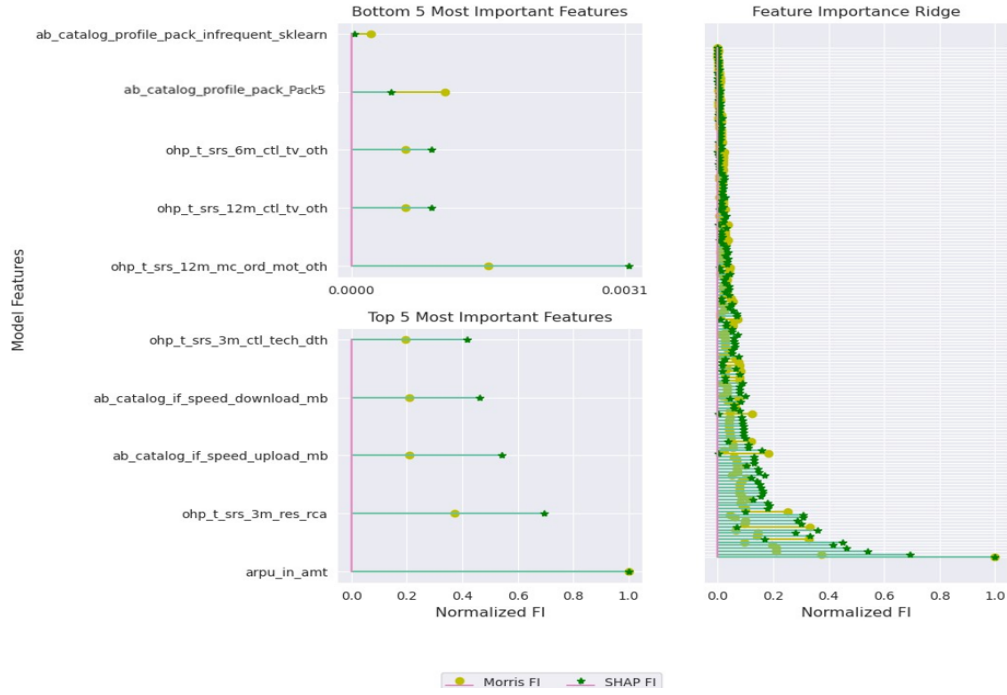


FIGURE 5.4: Global Feature Importance of predictors in the Ridge Regression model using SHAP and Morris Method

The GLM model (Figure 5.5) also shows a high impact of the *arpu_in_amt* of the variable. Here we also see a notable example of explanation discordance in a variable linked to the operator's area within the company (*ohp_area_dsc_Area10*) where the Morris method notes it as having the highest level of importance while SHAP gives it no significance in model predictions. In general both attribution methods pose a lot of importance on variables related with the operator that answers the Service Request.

The GAM model (Figure 5.6) seems to have its predictions more closely tied with the operator side, as no customer information is ranked on the top 5 most important features, a behaviour that is not seen on any of the other models. We also see the high level of discordance between explanations, specially in the most important features.

The sensitivity analysis performed on the EBM model (Figure 5.7) suggests that it takes an opposite approach to its less complex GAM counterpart, by prioritizing features related to customer information. Namely, *chs_t_srs_6m* and *chs_t_srs_12m*, highlighted mostly through the Morris method, suggest a high importance on the volume of service requests a customer was involved in over the 6 and 12 months aggregation period respectively. We also see the resurgence of *arpu_in_amt* as a variable both methods agree is the most important for the regression task.

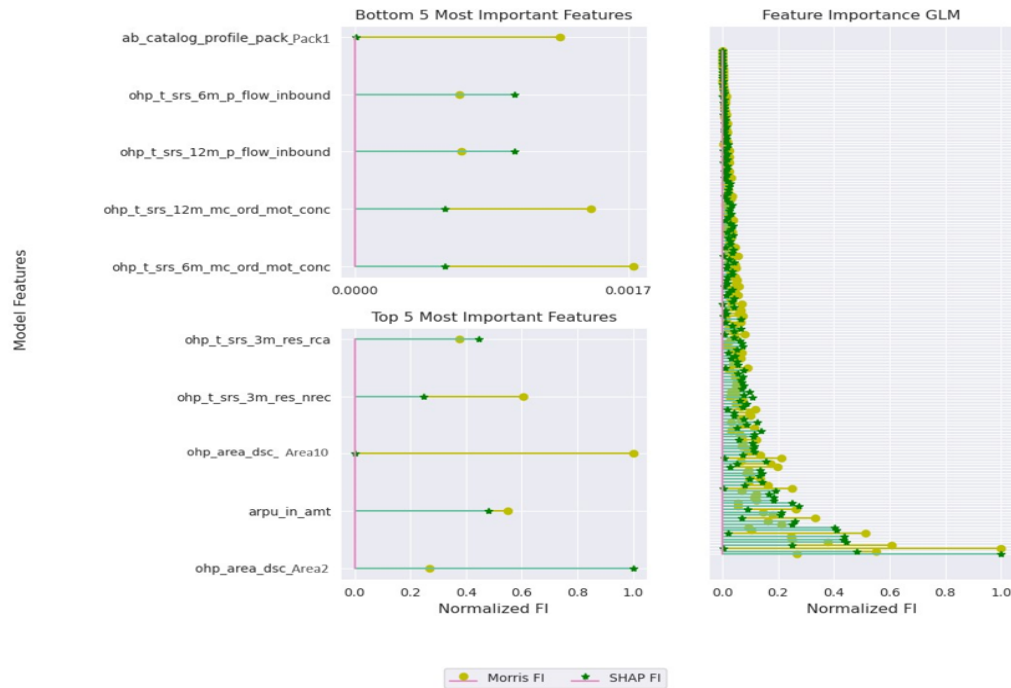


FIGURE 5.5: Global Feature Importance of predictors in the GLM model using SHAP and Morris Method

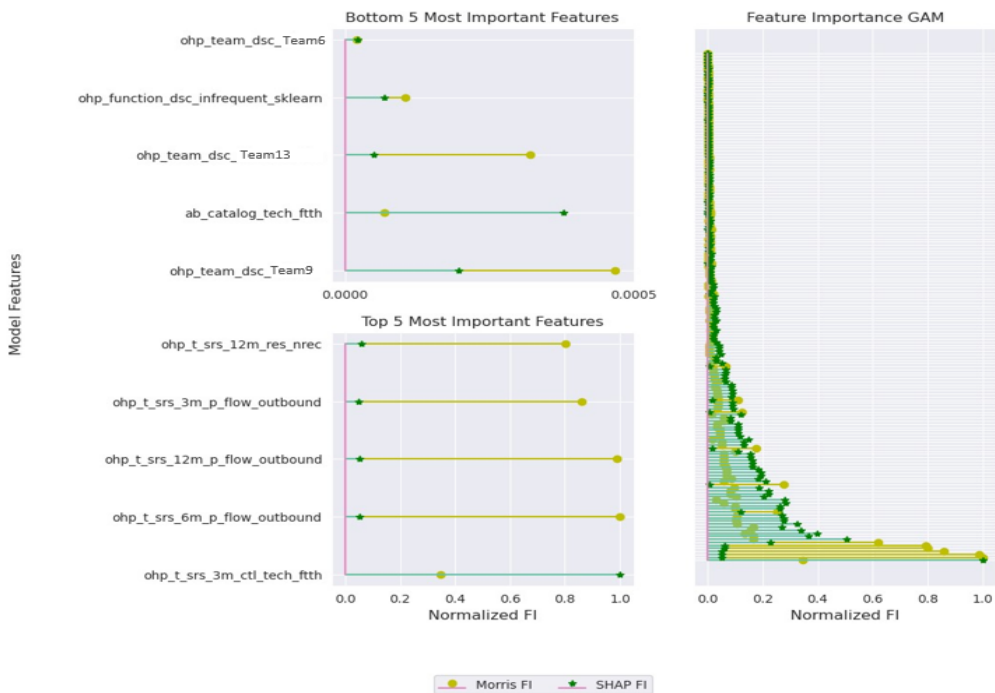


FIGURE 5.6: Global Feature Importance of predictors in the GAM model using SHAP and Morris Method

Finally, XGBoost (Figure 5.8) is the model that shows more disparity between the sensitivity methods used, with the Morris method typically imposing higher values for the

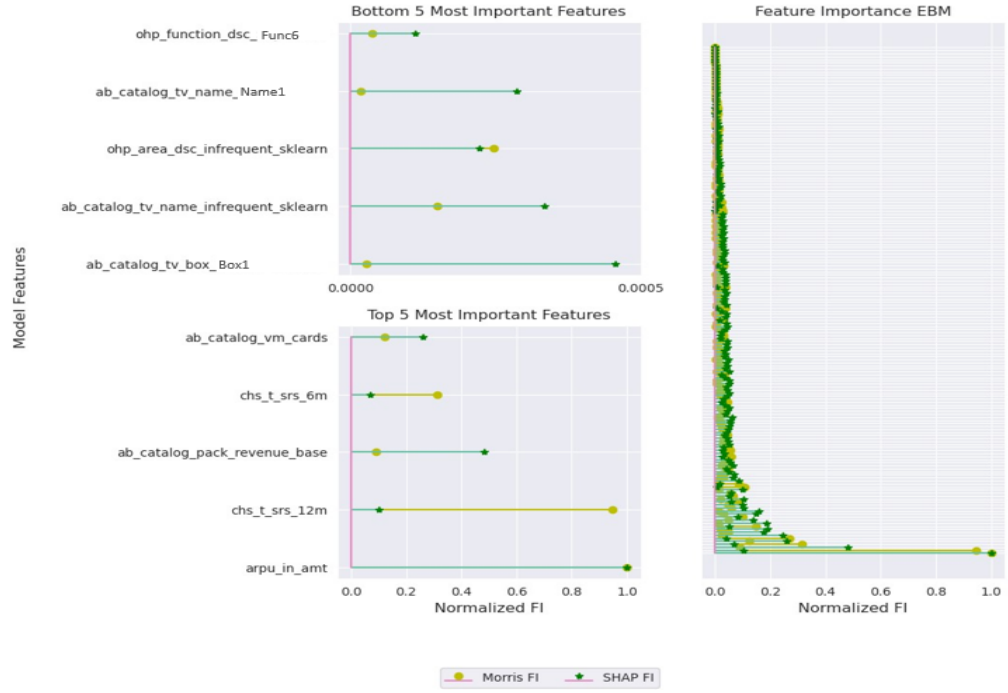


FIGURE 5.7: Global Feature Importance of predictors in the EBM model using SHAP and Morris Method

attributions. Like the EBM model, we see that the most important features regard customer information, with *arpu_in_amt* as the most important variable for both attribution methods. We can also observe that the least important features do not contribute anything to model predictions according to both methods, meaning that they could be excluded from future iterations of the model without any loss of predictive power.

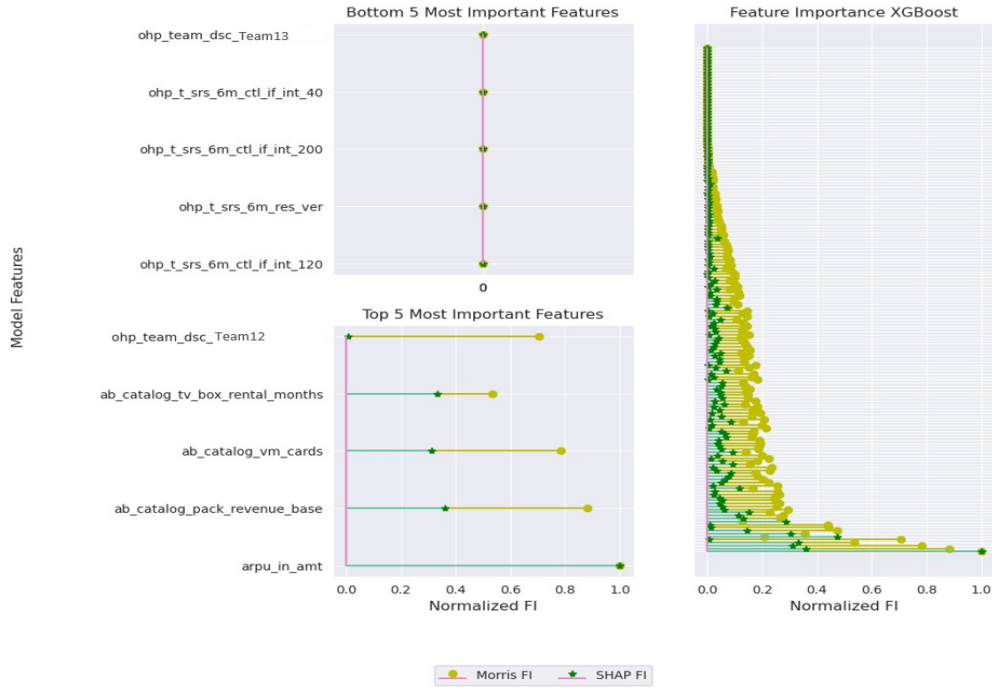


FIGURE 5.8: Global Feature Importance of predictors in the XGBoost model using SHAP and Morris Method

5.3.1 Morris Statistical Indicator Plots

Recalling section 2.4.2, we can visualize the plots of the statistical indicators extracted from the Morris method to better understand the behaviour of the features. As such we present the plots of the importance (Campolongo mean [72], μ^*) and the corresponding variance of the elementary effects (σ). Recall that we can use the ratio of these statistical indicators to infer the degree of interactions/monotonicity that predictors present [73] [74].

The plots for our Ridge Regression and GLM models (figure B.1 and B.2, respectively) are available in appendix B. Both of the interpretations of these plots are rather straightforward, with our Ridge model showing all variables as monotonic and non-interactive, which is to be expected from model construction. It is to be noted that in the plot B.1, variance is expressed in the order of 10^{-11} , which is close to the expected value of 0 (due to model constraints) and the difference is likely due to numerical precision errors. For our GLM model, we note that all variables are categorized as highly interactive/non-monotonic. Although this might not seem intuitive at first, this result is a direct consequence of the use of logarithmic link function, as it transforms the linear portion of the model into a multiplicative relation between target and predictors:

$$\begin{aligned}
\log \left(\mathbb{E} \left(y^{(p)} | \mathbf{x}^{(p)} \right) \right) &= \beta_0 + \sum_i^N \beta_i x_i^{(p)} \\
\Leftrightarrow \mathbb{E} \left(y^{(p)} | \mathbf{x}^{(p)} \right) &= \exp \left(\beta_0 + \sum_i^N \beta_i x_i^{(p)} \right) \\
\Leftrightarrow \mathbb{E} \left(y^{(p)} | \mathbf{x}^{(p)} \right) &= \exp(\beta_0) \prod_i^N \exp(\beta_i x_i^{(p)})
\end{aligned} \tag{5.1}$$

, as such it is to be expected that the features are highly interactive with one-another.

For the GAM model (figure B.3 in appendix B), we can begin our analysis of the statistical plots by recalling model training, where an identity link function was chosen and we only considered functions of a single predictor (enforcing not having feature interactions). Although typically in this analysis we cannot differentiate between a feature being non-monotonic or interacting, since we know that no interactions are considered in the model, we know that only the monotonicity of the function of each predictor can affect the placement of a variable in different regions of the plot. We note that the top 6 most influential features according to the Morris method are classified as showing monotonic tendencies, meaning that the relation between predictors and target are more simple and can be better explained. We can also note that the presence of non-monotonic features occurs only in regions of low feature importance, meaning that the majority of the decision can be explained using features with simpler relations with the target. The EBM model expands the complexity of the GAM to include feature interactions, meaning now we can't differentiate a variable as being non-monotonic or highly interactive. Regardless the results show (figure B.4 in appendix B) that the top 3 features either show monotonic influence/low interactions (*arpu_in_amt*, and *chs_t_srs_6m*) or very high linearity/Independence of features (*chs_t_srs_12m*). We note that the presence of features with more complex relations with the target appears not at a higher importance (starting with the 8th most important feature) meaning that explaining model decisions accurately involves understanding the interactions and specific behaviour of features and makes the process of understanding model through feature importances alone more erroneous.

Lastly, our XGBoost model B.5 shows the highest degree of feature complexity (excluding the GLM with logarithmic link function), which is to be expected from model construction. The features with highest importance (*arpu_in_amt* and *ab_catalog_pack_revenue_base*) show signs of high feature interaction/non-monotonic relations with the target, a trend

that is present in the majority of the model features. This indicates that accurately producing general explanations for model behaviour becomes a very difficult task, and the simplification done through feature importances might not be sufficient to fully understand predictive behaviour. We summarise the results obtained in this step through figure 5.9, which exposes the proportion of features in each of the categories considered. Here we see that, in general, increasing model complexity translates into increasing complexity of feature shapes and interactions when predicting the target variable.

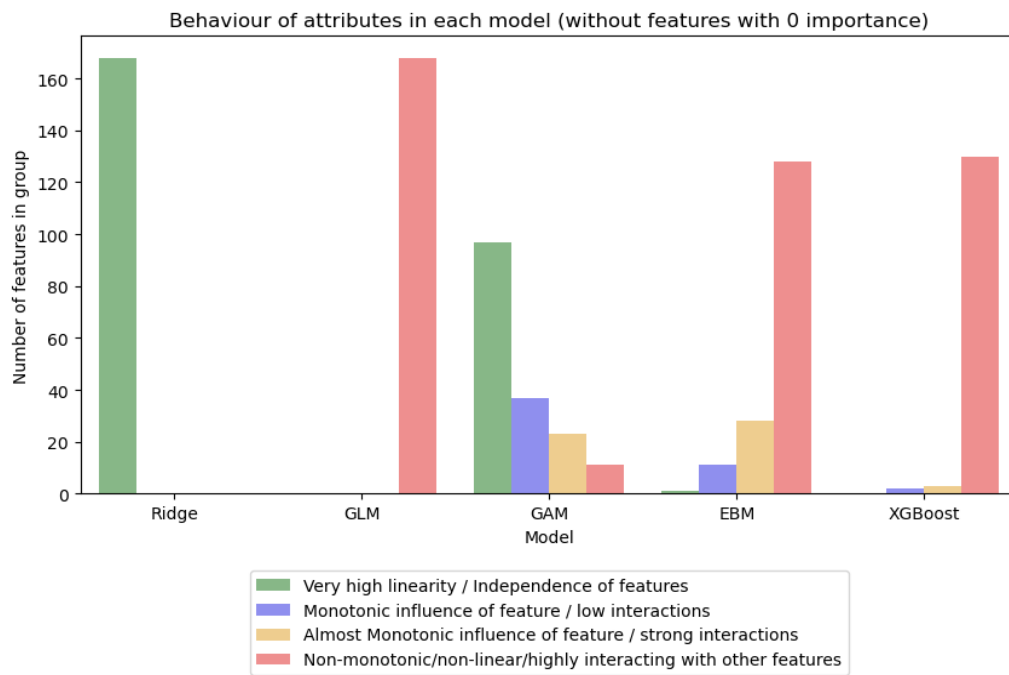


FIGURE 5.9: Summary of the proportion of features in each category regarding Independence/monotonicity according to the performed Morris sensitivity analysis

5.3.2 Evaluation of Explanations

Having analysed the results of our sensitivity analysis, concluding on the importance and behaviour of features, we now progress to the evaluation of our explanators. For this we use the monotonicity of feature attributions metric described on 2.4.1. As described in the work of Nguyen *et. al* [67], for the calculation of this metric we need two elements:

- The absolute value of the feature attributions for each variable i , $|a_i|$, which we already calculated previously using SHAP and the Morris method;
- The expected loss of not knowing the true value for the variable i in a given example, given by $E(l(y^*, f_i)|x_{-i}^*) = \int l(y^*, f_i(x_i))p(x_i)dx_i$.

For the obtainment of the expected losses we first need to define how we obtain the probability density, $p(x_i)$, the loss function used, $l(\cdot, \cdot)$, and the evaluation function, $f_i(x_i)$. Additionally, we will need to expose how the integral is calculated, alongside how this formula was updated for binary variables.

Starting with the most straightforward detail, let us recall that the function $f_i(x_i)$ is nothing more than the evaluation of the model on a given data point with the value x_i at feature i and with all other variables restricted at a data point $\mathbf{x}^* = (x_1^*, \dots, x_n^*)$. As such, once a data point $\mathbf{x}^*$ is selected, for a machine learning model $f : R^n \rightarrow R$, $f_i(x_i)$ is nothing more than $f(x_1^*, \dots, x_i, \dots, x_n^*)$. For the loss function, since we are considering a regression problem, we opted for the mean squared error.

Obtaining of the probability density function, $p(x_i)$, is more tricky. For now, let us exclude binary variables and consider only the continuous ones. Since this work is in the context of a real world application and not synthetic data, we do not possess the true distribution of each variable. As such the best way to calculate this metric is by first estimating how this distribution might be. First we opted for a 'true to the model' approach [67], meaning that we will consider probability density functions that are independent of one-another, in other words we don't use a conditional distribution for feature i based on the values fixed by the data point $\mathbf{x}^*$. We then estimate these probability density functions by kernel density estimation using Scipy's adaptation of the Gaussian Kernel Estimation method [117]. This method provides an approximation of the probability distribution by summing weighted contributions of different normal distributions. We used the data available in the train dataset for fitting the kernel estimation.

Still considering only the continuous predictors, the integral is calculated via the trapezoid method of numerically calculating integrals (also implemented in Scipy [118]), done by summing the area of a finite number of trapezoids under the function curve to estimate the area under a curve given certain limits of integration. The integral was performed under the domain of each of the variables, making sure that the majority of the distribution density (here selected as 95% of the distribution) was contained in this domain.

For the binary variables we possess, either naturally or through the one-hot encoding performed in the pre-processing step, we start by adapting the formula for these expected losses. First we note that the objective of the integral is to calculate the expected loss: $E(l(y^*, f_i) | x_{-i}^*)$. In the case of variables with a finite number of outcomes, the expected value of a quantity $g(x)$ can be calculated with a sum rather than an integral, $E(g(x)) = \sum g(x)p(x)$ [119]. The probability of each outcome was estimated using the proportions available in the train dataset.

With all of this in mind, a routine was created for computing the expected losses for each variable (considering whether they are binary - done through a sum- or continuous - done through an integral). In the cases where an integral was performed, we used 100 steps for the trapezoid method (the domain of integration was divided into 100 equally spaced steps for estimation of the integral). Due to time complexity of the implemented routine we randomly selected 100 points from our test set as the candidates for x^* , and we use the global feature attributions to describe them. The use of global feature attributions rather than local ones, regards the fact that we are trying to estimate how adequate using the global attributions can describe the behaviour of the model, regardless of the region selected.

Before delving into the results of this implementation it is useful to disclose different sources of error that might affect the calculation. Firstly the use of an estimation of the density function of our variables, although a good approximation of its true behaviour, is nevertheless an estimation. The calculation done in synthetic data could yield more accurate results as we would be using the actual data distribution. Secondly, the integral is done numerically, rather than analytically since we do not possess an actual functional form of the quantity being calculated. As such numerical errors of integration are also a factor since we are only obtaining an approximate form of the actual area under the curve. With this in mind, the implementation tried to mitigate some of these problems by selecting a relatively high number of steps of integration, and a verification that the

domain of integration is adequate.

After obtaining the expected losses, the calculation of the metric is straightforward, done through the calculation of the Spearman Ranked Correlation implemented in Scipy [120].

The results, in figure 5.10, show that the attributions, generally, are highly monotonic with the expected losses, consolidating their validity according to this metric. We see, in general, SHAP attributions have a higher degree of monotonicity, when compared with the Morris Method, with the exception of the GLM model. With this in mind, we posit that, with the exception of GLM, for explaining model behaviour through attributions we should consider the ones provided with SHAP. It is also to be noted that this metric indicates that for the GLM model the explanations are not as adequate as for the other ones, probably due to the use on a logarithmic link function that creates a multiplicative relation between predictors which clashes with the additive nature of these explanations. We can also note that, according to this metric, XGBoost explanations underperform slightly when compared to the Ridge Regression, GAM, and EBM models.

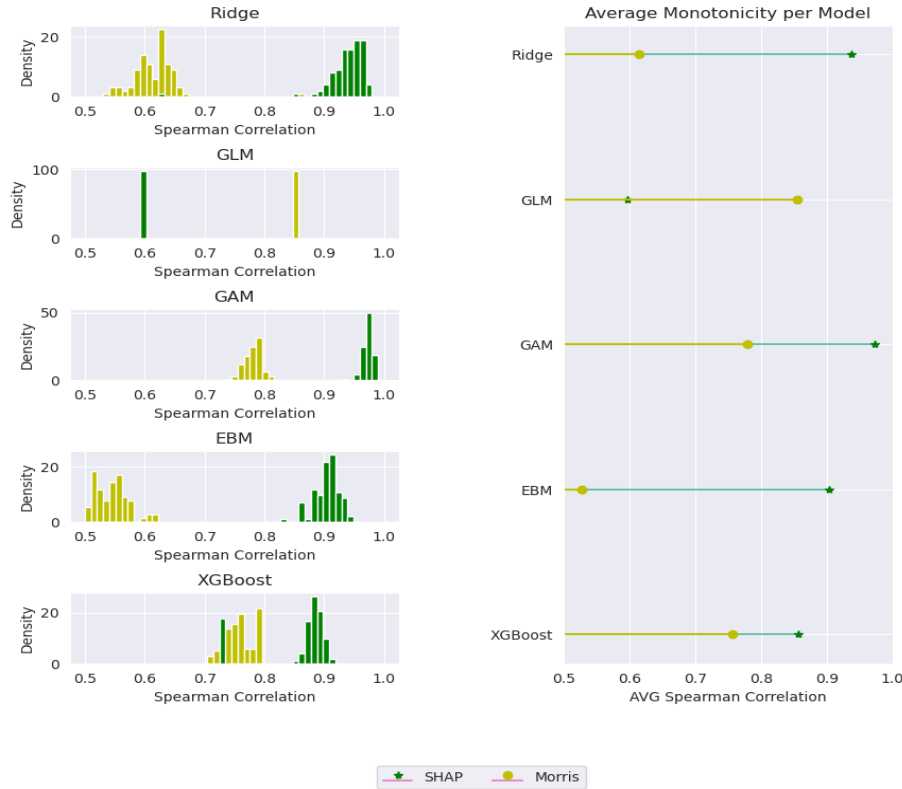


FIGURE 5.10: Monotonicity of the Feature Explanations obtained (Morris Sensitivity analysis and SHAP) for each of the trained models. On the left we present the histograms of the monotonicity values for each of the 100 individuals. On the right we present the average values of the Spearman correlation in this sample.

5.4 Error Analysis

In this section we cover the steps taken to understand the regions (cohorts of examples) where the different trained models have different performances. Additionally, we try to expand on the trained regressors, in order for predictions to account for model error by obtaining prediction intervals rather than point predictions. The utility of this last step will be explained more in depth in [5.4.2](#).

5.4.1 Understanding Model Error

In a real-world application with high dimensionality, understanding model residuals through traditional residual plots becomes somewhat infeasible both due to the high amount of information to visualize and the disregard of possible interactions between residuals for different variables. One way to gain an understanding of model error, described in section [2.3](#) is through the fitting of interpretable models on the residuals of the prediction, in order to segment our feature space in regions where we can observe differences between predictive power. Additionally, the use of meta-information (using features unavailable to the model at time of inference) can also yield valuable information regarding model performance.

Starting with the use of meta-information, each call or Service Request takes one of the following possible outcome:

- *N Recuperado* (churned customer) meaning that the customer abandoned the service after the call, turning their *ltv_target* (our target variable) into null;
- *Recuperado Sem Alteração* (unchanged customer contract) meaning that after the call no alterations were made to the customer's current plan;
- *Recuperado Com Alteração* (changed customer contract) meaning that the result of the call was a successful alteration to the customer's contract (either expanded, with alteration to customer plan, or both)

At time of inference, before the customer call, this information is not available and cannot be used to produce model predictions. However when analysing results we can observe how the model behaves for each of these categories. Figure [C.1](#), in annex [C](#), shows how the residuals of the MVP model (a gradient boosting tree) behave in each of these categories. Notably, we see that the behaviour of residuals in calls classified as

churned (*N Recuperado*) is more erroneous than for the other possible states. This indicates that the model has difficulties predicting customer churn, leading room for improvement by adding variables that can better help predict this state (which were added in following MVP model iterations). The results for each of our trained models were similar, figure 5.11, showing values of residuals that highly surpass the MAE of each model (average of absolute residuals).

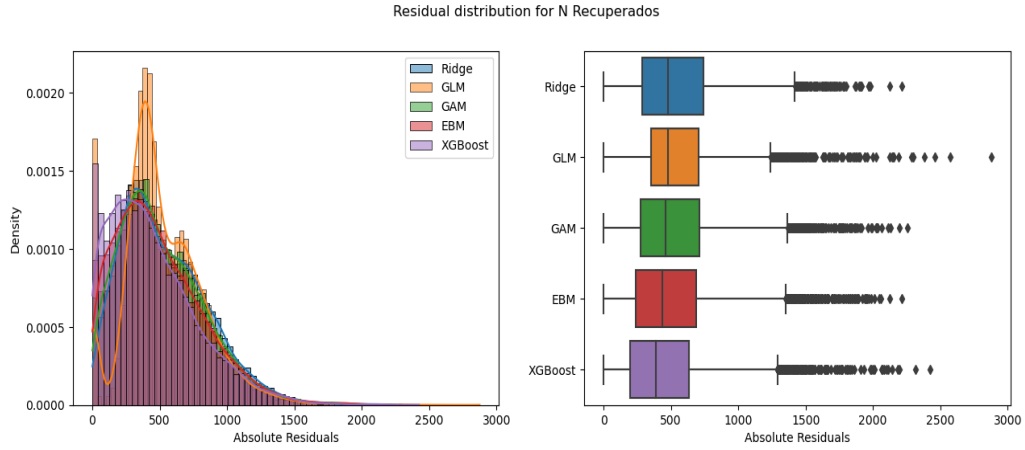


FIGURE 5.11: Absolute Residuals for each model in observations where the customer churned.

After performing this analysis using meta-features, we can move to try to devise insights from the features that the model actually uses. For this we used an interpretable regressor - shallow regression trees - trained on the absolute prediction residuals for identifying high error regions. Given a model, $f : S \rightarrow \mathbb{R}$, and data points, (X, y) with $X \in S$, $y \in \mathbb{R}$, the absolute residuals are calculated as $|f(X) - y|$.

The residual trees were trained and assessed on the test set, with the main objective of mining patterns that are still present after model prediction. We used a maximum depth of 3 (maintaining high cohort interpretability), and a minimum number of samples required to be at a leaf node of 500 to avoid creating outlier cohorts, instead focusing on obtaining larger groups of higher error. Alternatively, this work could have been performed using Microsoft's Error Analysis framework [38], however for better control of tree parameters and a more concise visualization (using DTreeViz [121]) we opted for a manual approach.

By fitting our shallow regression trees on the residuals of the predictions for each model on the test data, we noticed that our models seem to under-perform for customers with high-paying packages and high number of voice and mobile cards. This is a scenario

that occurred regardless of the model used which indicates that those customers are hard to model with the available data, and suggests that improvements could be made by acquiring more examples in these critical regions. Figures C.2,C.3, C.4,C.5, C.6, in appendix C, expose each of the trained trees with their respective cohorts.

We can expand on the proximity of the highest error cohort (HEC) in each of our models by calculating the Jaccard Similarity between the sets of individuals formed by this cohort. The Jaccard Similarity is defined, for two sets A and B , as the ratio between the size of the interception of the sets and the size of their union, $\frac{|A \cap B|}{|A \cup B|}$. From figure 5.12, we notice high similarity between the HEC of our increasingly complex additive models, with the HEC of XGBoost being the most dissimilar of the group. This likely occurs since the first cut performed on the XGBoost residual tree occurs for an operator variable (*ohp_t_srs_12m_p_flow_inbound*) regarding Service Request flows over the 12 month aggregation period, showing a higher influence of error on the operator side before addressing customer profiles.

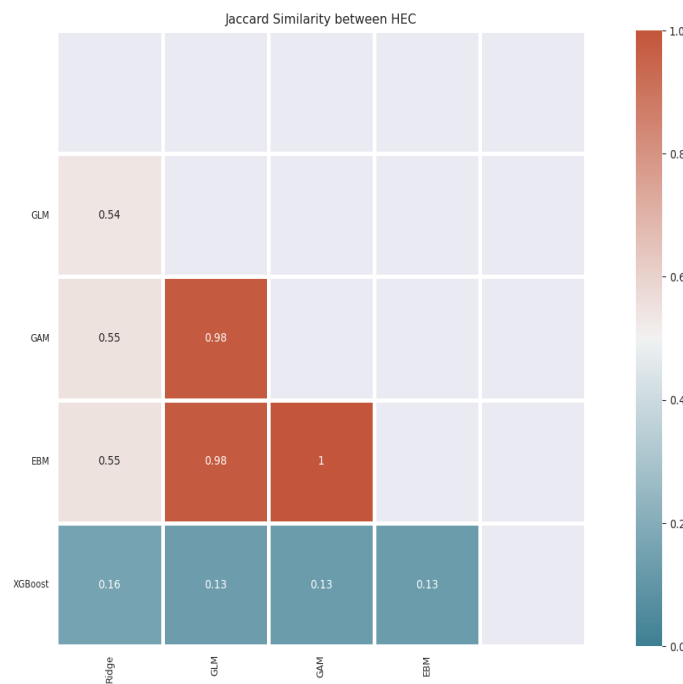


FIGURE 5.12: Jaccard Similarity between observations (pairings of customer and operator) in each model’s Highest Error Cohort (HEC).

Finally, we compare the cohort error in each of the models. The local error in each test cohort, alongside cumulative error, for each model can be found in figure 5.13. From this, we extract that all of the trained models have a significantly higher local error in the HEC, between 1.6 to 1.9 times the global error of the model. Out of all the models, XGBoost

has a slightly higher percentage increase in error on the HEC. We also note that the first 3 cohorts of observations account, in general, for the majority of the error in the test set and, analysing the residual trees (appendix C) we note that these cohort reflect, at some degree, the price of customer packages, reinforcing the notion that the models struggle to predict contracted revenue for high paying customers.

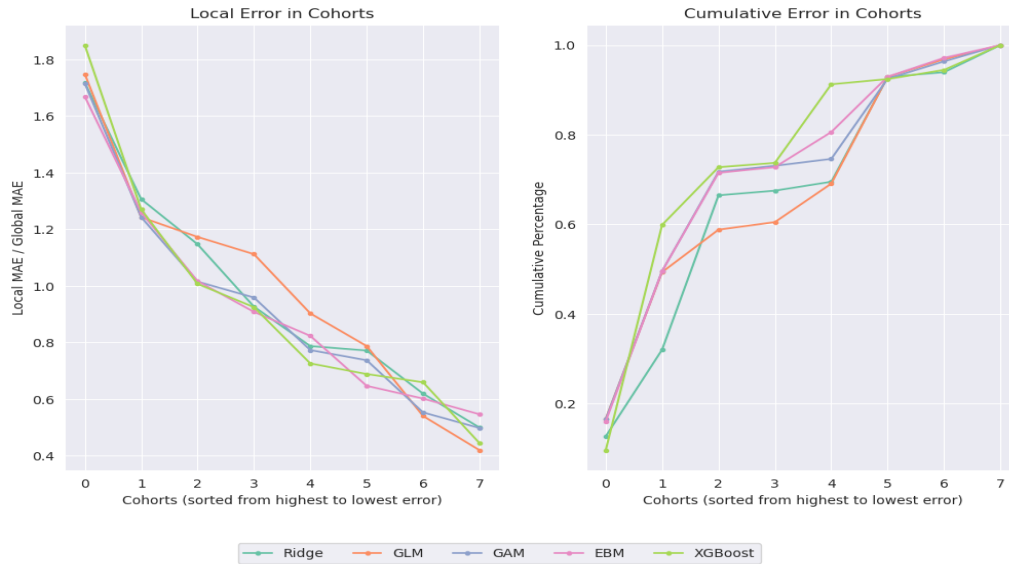


FIGURE 5.13: Local Error (cohort MAE / Global MAE) and cumulative error for each identified cohort

5.4.2 Quantifying Model Error

After identifying the regions of higher error, a question arises regarding how to address the disparity of model performances. Here, a few options can be considered from the insights extracted: adding variables regarding customer churn probabilities, opting not to enforce analytics on groups where models show higher degree of error, or even retrain models in the erroneous cohorts in the hope of improving local performance. Another alternative to these exposed ones derives from the context of Smart Pairing. Since the end goal of the regression problem is to provide a way to construct a ranking system, we can improve on the results if we can quantify prediction error. If rather than a point prediction we provide an interval of prediction, on the ranking problem we can choose not to differentiate between operators with overlapping prediction values, effectively guaranteeing that we only consider an operator A, as being a better choice for a customer, over operator B if all values that fall on the prediction interval of A are higher than all the values that fall on the interval for B.

For constructing these intervals we used two different approaches based on conformal regression using the MAPIE library [43]: one of the approaches was using a traditional split conformal regressor, and another following the Mondrian conformal regressor methodology, with the disjoint sets being generated through residual trees.

For both methods we used a separate calibration set, and select a target coverage of 90%, meaning we expect 90% of the true values of our target to fall on the desired prediction intervals.

In figure 5.14, we can observe the limitations of using the traditional split-conformal approach of MAPIE, which leads to prediction intervals that fail to maintain the local coverage near the desired target since the prediction interval width does not vary. As such, these prediction intervals grossly overestimate uncertainty in regions where model error is low, leading to coverages above the target (valid but overly conservative), and under-estimate in regions of high model error.

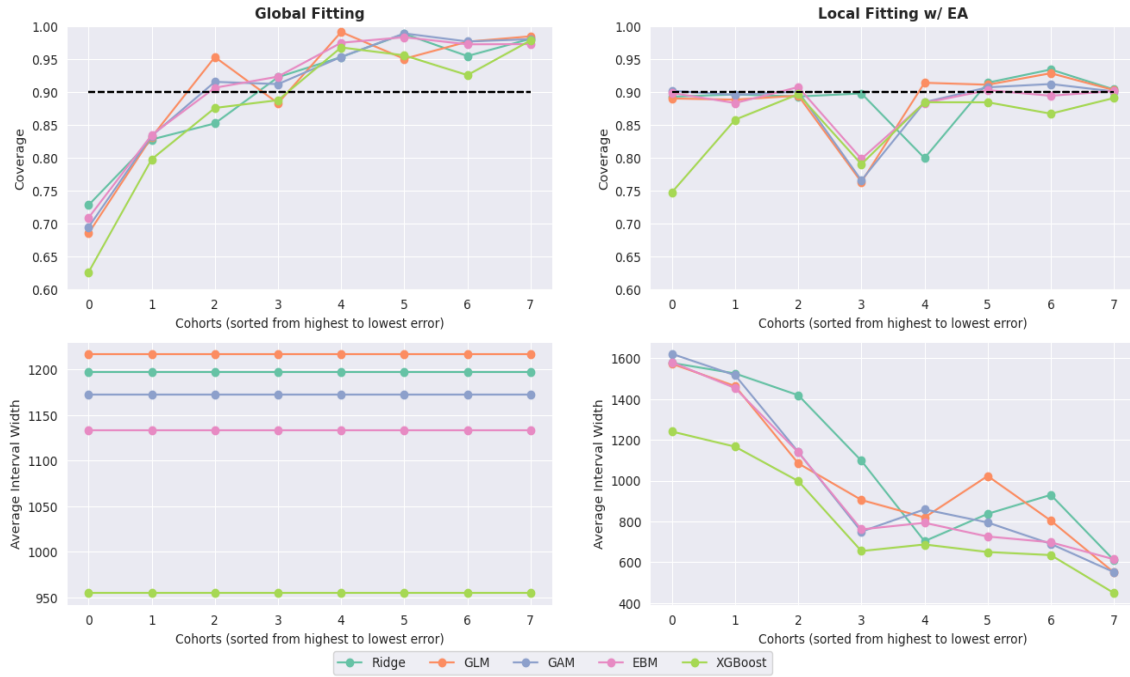


FIGURE 5.14: Local coverage and mean interval width assessment using both described strategies. The dashed line represents the desired coverage selected, 90%. We can visually verify a higher stability of coverages using our locally fitted approach, as well as a more granular interval width over test error cohorts.

In contrast, our approach to locally fit an estimator in each error cohort in the calibration set leads to more granular interval estimates that locally maintain the coverage near the desired target in each test cohort.

In table 5.3 we present the average absolute deviation between measured coverage and desired coverage over each cohort, for each of the trained models in both approaches. We conclude that our approach leads to a significant average improvement over a global fitting approach.

		Models				
		Ridge	GLM	GAM	EBM	XGBoost
Approach	Global MAPIE	7.4%	8.2%	7.5%	7.4%	8.0%
	Local MAPIE	2.2%	2.8%	2.2%	1.9%	4.8%

TABLE 5.3: Mean absolute deviation between target coverage and measured coverage over each cohort (lower is better), for each prediction interval approach.

5.5 Adversarial Robustness

After assessing global and local error, performing and evaluating sensitivity analysis, and having an understanding of the degree of feature monotonicity/interactions, our analysis concludes with an understanding of the ease of which we can create adversarial examples for each model. As mentioned in chapter 2.5, we will follow an approach of an l_2 attack, selecting a small perturbation radius ($r = 0.1$, an order of magnitude smaller than the features' standard deviation) and using the differential evolution algorithm implemented in scikit-opt [99] as the solver for the optimization problem. Recall that the method for determining an adversarial example that we follow is: given a real data-point find a point in the feature space within a sphere of radius r that maximizes the error between its prediction and the true value of the data-point's target.

We implemented a routine to perform this task given a subset of data-points, creating an adversarial point for each one in the given subset. After obtaining the corresponding points, we will compare the MAE of the subset of real points and the MAE of the adversarial points generated from it, and derive our conclusions by comparing both metrics. Here we also enforce that the adversarial points retain characteristics that we expect from the data, namely the binary variables were enforced, meaning that the adversarial points can only take the values of our binary variables in the features that possess this behaviour. We also enforce that the values taken by these points are in the domain of each feature (using our train dataset) so that we are not generating points on regions where a feature took no values during training.

As, computationally, it would be unfeasible to perform the optimization task for the entire test-set we sampled 1000 data-points for each model (with the exception of GAM, which will be discussed further). As we saw that the error is not uniformly distributed in the feature space, in order to avoid creating subsets of points that fall non-uniformly on different error cohorts (for example a subset of points that have the majority of them in a high error region for model A, and a subset of points that fall mostly on low error regions for model B) we sampled 125 points from each of the cohorts identified in 2.3 for each of the models. As such, each group has a unique group of observations selected in such a way that some have high error and some have low error.

The GAM model proved to have a very high computational cost in performing inference, which led to slow predictions that made the process of optimization extremely costly. As such, for this model we followed a similar procedure, sampling from the different error cohorts, but used a smaller number of observations. We used instead 80 observations (rather than 1000, with 10 from each error cohort). This detail can impact the comparison between it and the other models, as we are using a smaller sample size, however we still verify statistical significant performance disparities between the group of observations and the adversarial examples extracted from it.

For our solver we used a population size of 50, meaning that for each data point we perform optimization using 50 candidate solutions, with a maximum number of iterations of 500. Like most optimization problems, it is useful to disclose that we can get local optimum solutions (rather than the desired global optimum), reason to which the number of candidates was raised from the default value of 15.

Our results, presented on figure 5.15, show that in general, an increase in model complexity seems to affect the ease to which adversarial examples can be extracted. This reflects a higher numerical instability of the model's decisions around our data points. As such, when faced with a new data-point, having prototypes or other observations in the vicinity of it might be insufficient to determine model behaviour. The GAM model proved to have a higher degree of instability than expected, as the ratio of error between the adversarial examples generated and the sample of observation is near 3.5, higher than even XGBoost. Here a couple of factors might influence this result. Firstly there is the fact that we obtained these values using a smaller sample size, which can lead to more variance of the adversarial MAE. However, another factor that can play a role in this is the fact that the model was trained using a small number of splines for each predictor,

which can lead to an oversimplification of the relation between features and target that might have been leveraged by our optimization procedure.

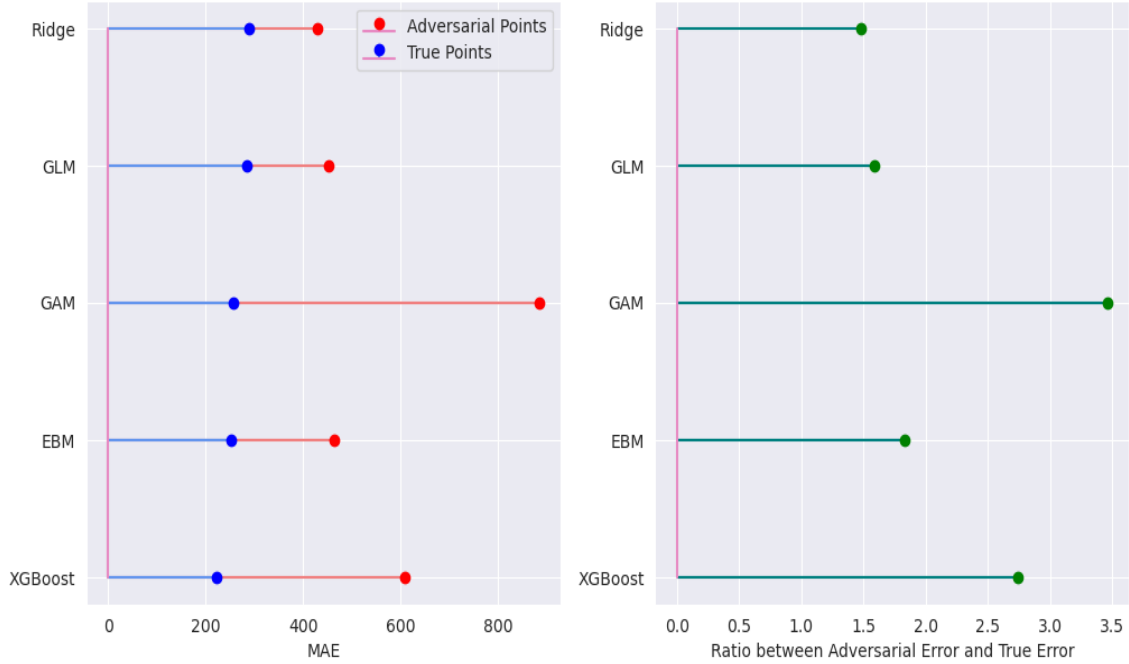


FIGURE 5.15: Sample and Adversarial Error (MAE) for each of the trained models, with a perturbation radius of 0.1.

5.6 Final comparison

In this section we expose a final comparison between our models, using metrics derived from each of the previous sections. We will look at the models in the light of 5 different dimensions:

1. Model performance - Measured through MAE and RMSE, exposed in section 5.2;
2. Model Simplicity - Referring to the degree of monotonicity/interactions of features, extracted from Morris sensitivity analysis (in section 5.3.1). We use the proportion of features with $\frac{\sigma}{\mu^*} < 0.1$, $\frac{\sigma}{\mu^*} < 0.5$, $\frac{\sigma}{\mu^*} < 1$ with respect to the number of influential features ($\mu^* > 0$). We used a naming convention for these indicators according to the color of the regions shaded in the Morris statistical plots, in appendix B. Each of the aforementioned regions are, respectively: G, (G+B), (G+B+O);

3. Explanation Quality - Measured for the explanations extracted with Sensitivity Analysis. Here we consider the monotonicity of the Feature Importance metric that best performed in each model (section 5.3.2);
4. Local Error - Where we compare the MAE error of the HEC with the baseline MAE (section 5.4.1), identifying the disparity of prediction error of each model;
5. Model Robustness - Assessed through the extraction of adversarial examples, and using the ratio between the error of the adversarial examples (MAE) and the sample that generated them as our metric (section 5.5).

In figure 5.16 we expose the condensed results using each of our evaluation dimensions. This final plot was constructed, for ease of interpretation, in a way that the y-axis for each of the dimensions is oriented such that the desired values are situated in the upper regions of the plot (meaning that error metrics such as RMSE and MAE are oriented such that the values grow from top to bottom). The first thing we can remark from this plot is that model selection through these metrics, in the context of Smart Pairing, does not have a clean cut choice. In general, there is no model that over-performs all the others in all categories, and there is a trade-off to be considered when choosing a model over another. In general, increasing model complexity does seem to impact performance in such a way that by choosing less complex models we are giving away some of the predictive power. XGBoost, although a very well performing model, lacks behind the other ones when we take into account the numerical instability that lead to more pronounced adversarial examples, alongside being a less simple model, with lower explanation quality and higher disparity of error between baseline MAE and HEC MAE. On the other side of the spectrum, the Ridge Regression performs well in most of our categories, but the sacrifice of model performance might be too harsh for a real world application. GAM and EBM have a more interesting middle ground. The GAM model shows good results regarding quality of explanations, and the simplicity of the model (by not considering interactions between features) makes it simpler to explain individual relations between a feature and the target variable. It under-performs when creating adversarial attacks, which can possibly be mitigated through more thorough training and assessment of this quantity (see section 5.15). It is also worth mentioning that, its long inference time made it hard to shine a more positive light on the model for this application. The EBM model is perhaps the strongest contestant to a balanced choice, as it performs the closest to XGBoost with

less sacrifice in the other dimensions. Finally, the GLM model under-performs. The use of the log link function proved to be a problem in the model simplicity, and its low performance does not give it a strong support for this loss. In fact, for this application it under-performs even when compared to the Ridge Regression, showing an indicator that more complexity does not necessarily mean a gain in some scenarios.

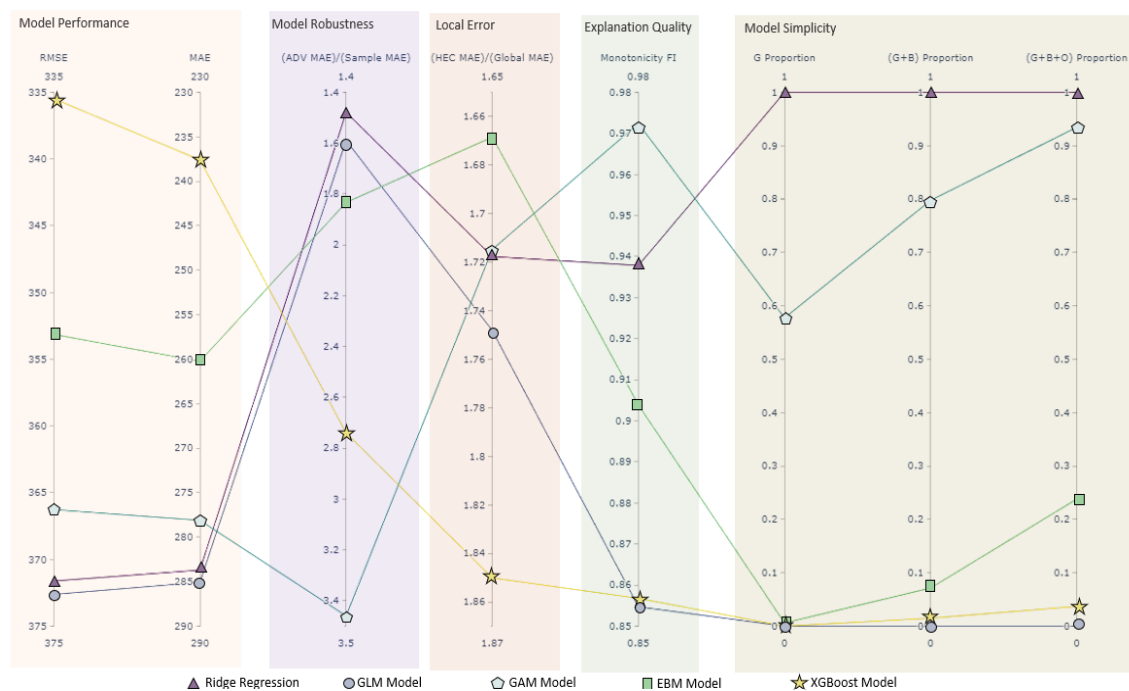


FIGURE 5.16: Final comparison for each trained model.

Chapter 6

Conclusions

In this work we have expanded model comparison beyond traditional aggregate performance metrics, including other relevant dimensions to model understanding, namely robustness, local performance, simplicity and quality of explanations. We took this framework to compare increasingly complex interpretable additive models, with XGBoost (a model only explained via post-hoc methods), in the context of a real-world application in a TelCo provider.

We have concluded, through our framework, that increasing model complexity in search of higher performance has a measurable cost in other relevant areas, which is something to be considered by the practitioner and the owners of the data product. Rather than looking at performance only, our more holistic view of model behaviour leads to a more well rounded model understanding and we propose this way of critically evaluating the models should be carried out in high-risk applications that deal with customer information to verify the integrity of the data product.

We also explored expanding on point predictions through the use of prediction intervals, and propose a way of obtaining such intervals based on segmentation of a calibration set through residual trees, which leads to widths of the prediction interval that are tied with different associated error regions. This means that we can explore the residual trees on the calibration set to better understand which regions of the feature space the model will have higher uncertainty associated with the predictions. Our exploratory approaches to error analysis also lead to a way of diagnosing model behaviour that can be applied regardless of model choice.

Perhaps the most important point that this study takes into account is that exploring and critically analysing the models should be standard practice in this growing field of

Data Science that more and more has a consequence on our daily lives. This work poses a way for teams in the company to have a quick-start in responsible AI, by transposing xAI problems into tangible engineering requirements.

6.1 Future Work and Limitations

Left out of this comparative study were dimensions regarding Fairness [122] [123] of model predictions. These metrics pose an important step to guaranteeing that predictions are generated in such a way that they don't wrongfully harm specific demographics in the prediction process, and pose an important step in responsible AI. As such for future work, we believe that integrating these factors in model comparisons should also constitute a requirement for model understanding. Other metrics regarding quality of explanations for other forms of explanations rather than just feature importance should also be included in future work, such as assessing quality of prototypes [31].

Also of note, in this work we opted for a comparison with additive models of increasing complexity for our interpretable baseline. It is worth mentioning that not all models that have inherent interpretability need to be of this kind [20], further studies on other types of interpretable models using this, or similar, frameworks should also be conducted. Further constraints applied on models, such as enforcing monotonic relations in XGBoost or Neural Networks[114], could also be included in future comparisons.

As for limitations of this work, as exposed throughout the experimental section, computational constraints led to the application of some methods in sub-samples of the data, meaning that there is risk that some results might be influenced by the sampling process. As such, a further study could be conducted using either a larger sample or the entirety of the data, using stronger computational resources. Furthermore, conducting a survey with stakeholder on requirements for the data product, having the research questions of the work as a basis, could also contribute for a more Human-in-the-loop processes, further integrating business requirements with the Scientific task at hand.

One additional limitation of this work comes at the point of the selection of the hyper-parameters to perform the grid-search for each of the models chosen. Here we add a degree of restriction for the modeling solutions by having a low number of possible hyper-parameters and hyper-parameter combinations for the optimization, and more robust Auto-ML approaches should be selected for guarantee a fairer comparison between model potentials.

In sum, even with the concerns regarding computational limitation and possible future work that could still be developed relating to this task, we believe the work developed poses a step into the direction of more responsible Machine Learning through the monitoring of relevant aspects of model behaviour.

Appendix A

Selection of Number of Categories to Encode

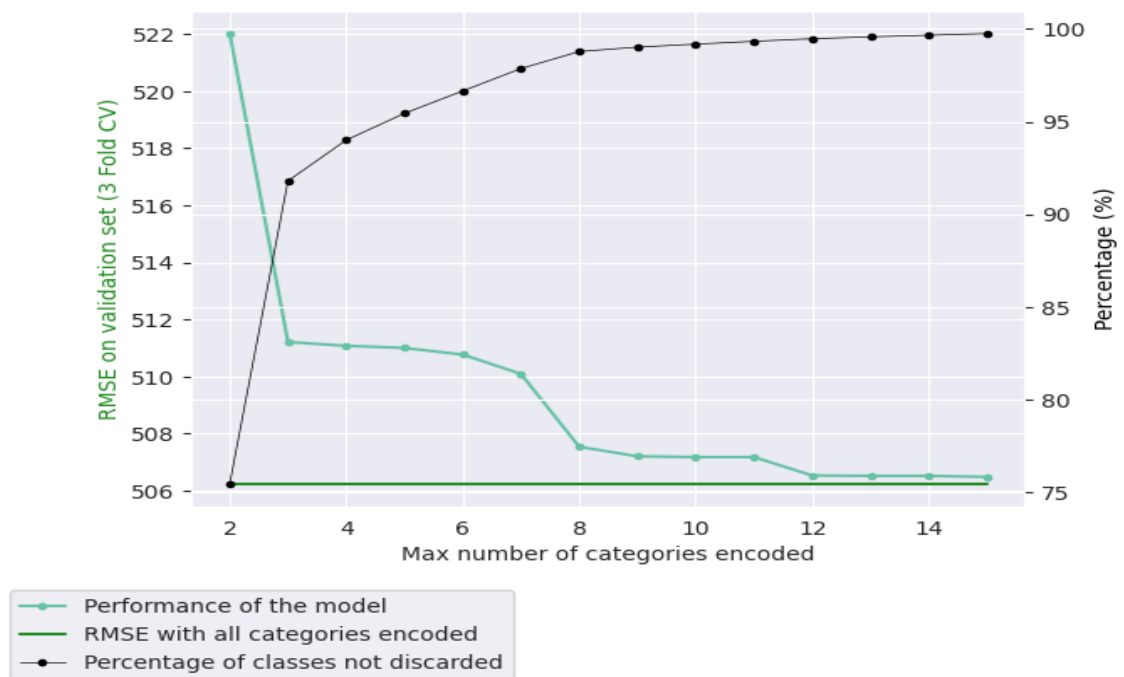


FIGURE A.1: Error of XGBoost model using only variable *ohp_area_dsc*

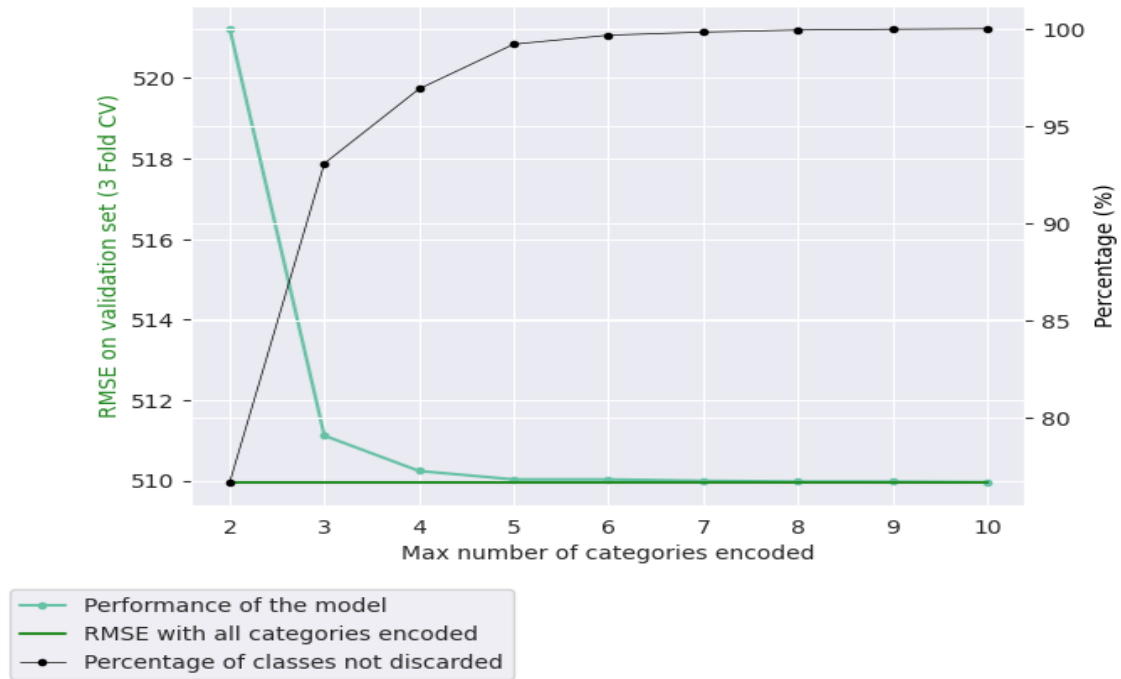


FIGURE A.2: Error of XGBoost model using only variable *ohp_direction_dsc*

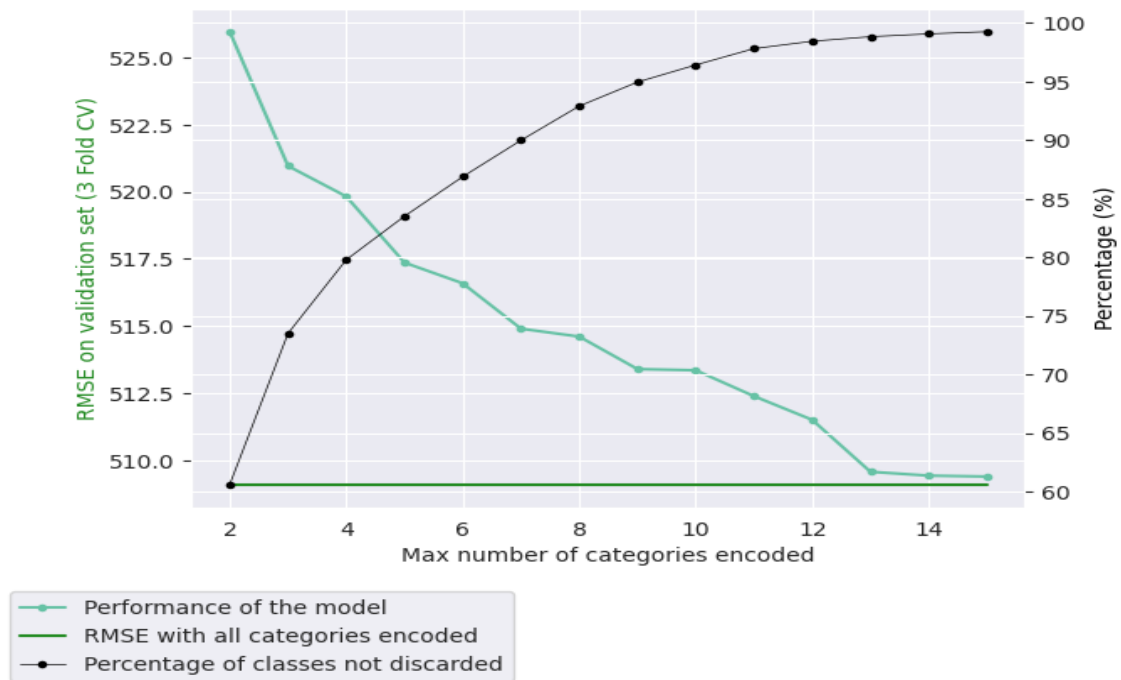
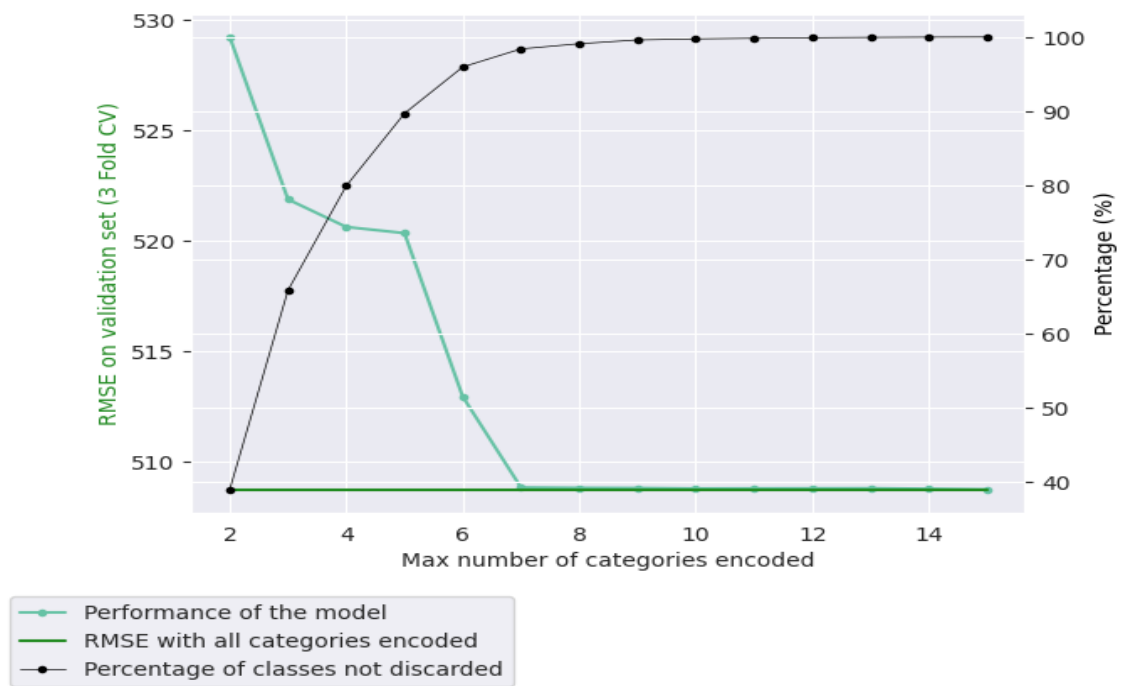
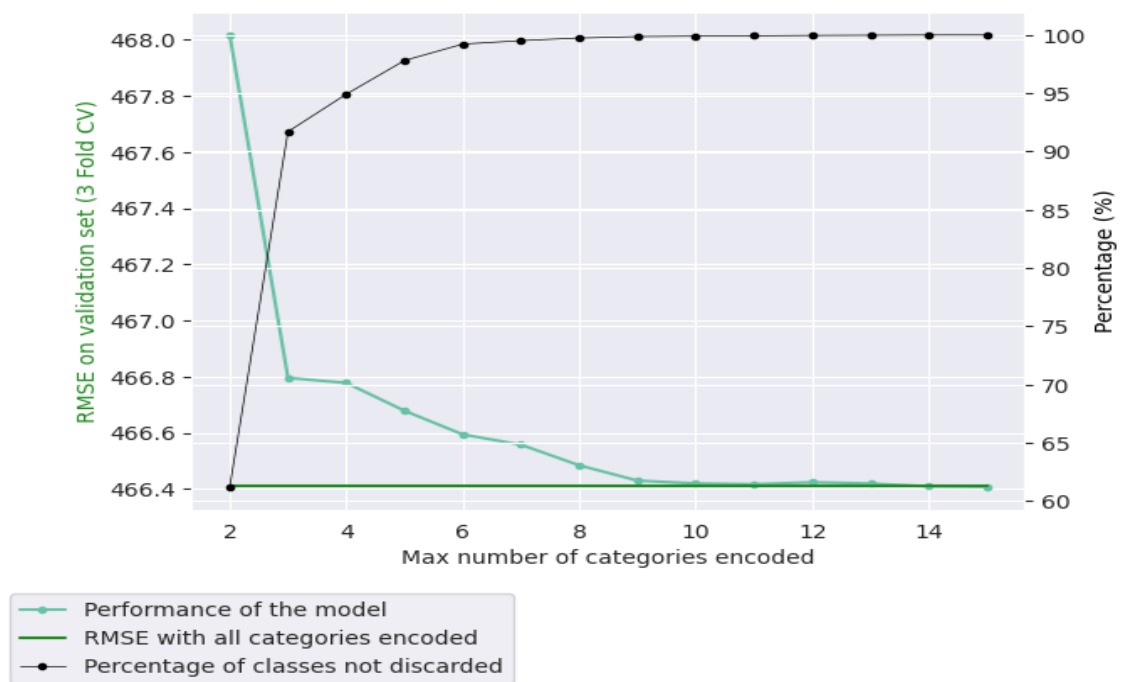


FIGURE A.3: Error of XGBoost model using only variable *ohp_function_dsc*

FIGURE A.4: Error of XGBoost model using only variable *ab_catalog_if_name*FIGURE A.5: Error of XGBoost model using only variable *ab_catalog_profile_pack*

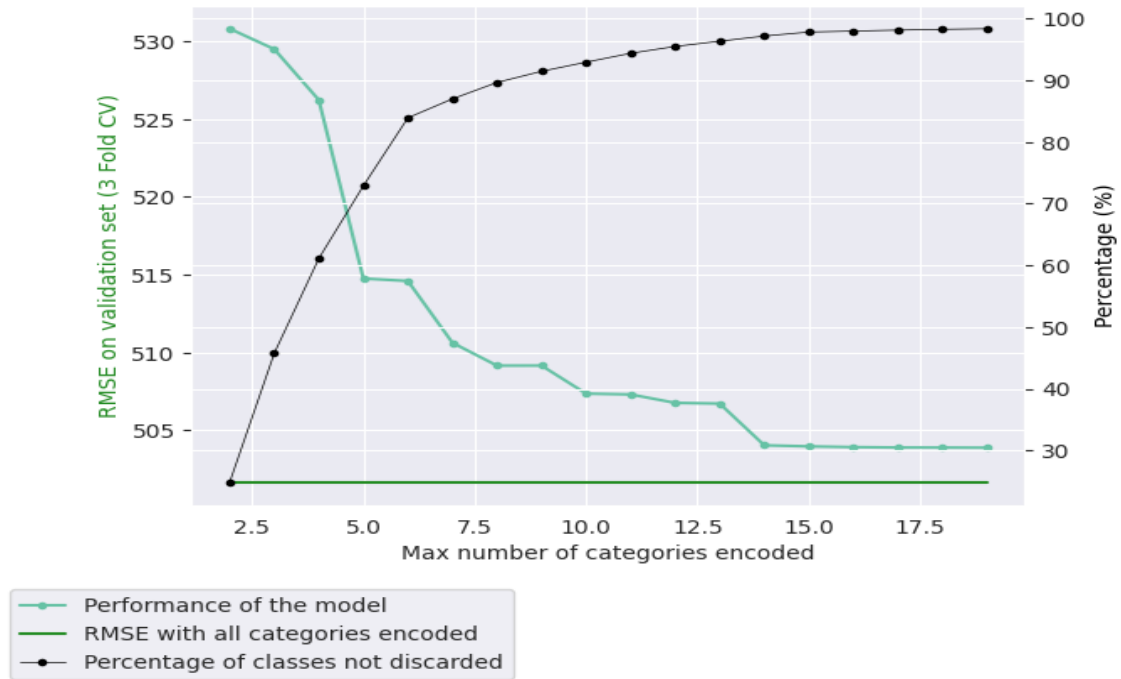


FIGURE A.6: Error of XGBoost model using only variable *ohp_team_dsc*

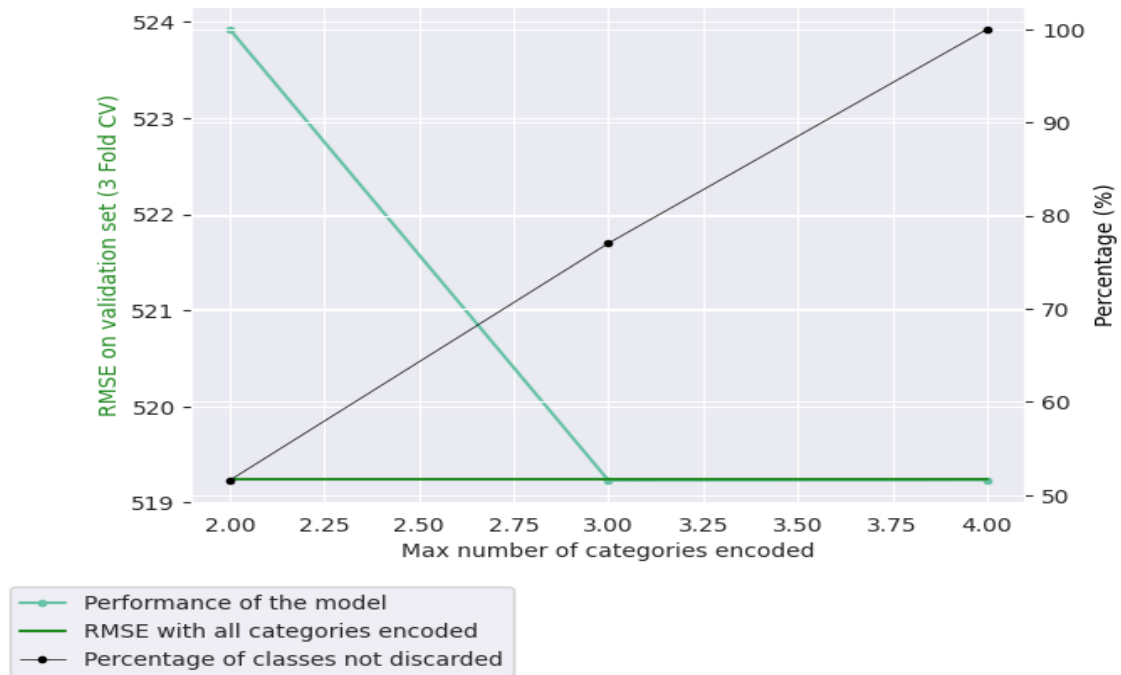
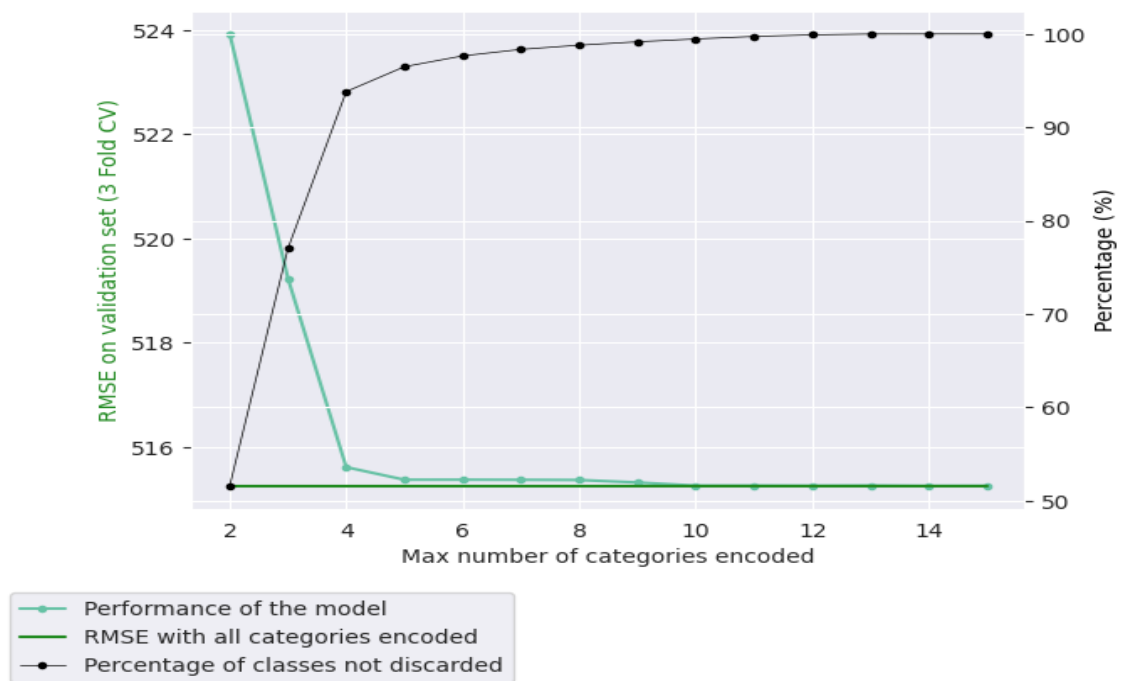


FIGURE A.7: Error of XGBoost model using only variable *ab_catalog_tv_box*

FIGURE A.8: Error of XGBoost model using only variable *ab_catalog_tv_name*

Appendix B

Morris Statistical Plots

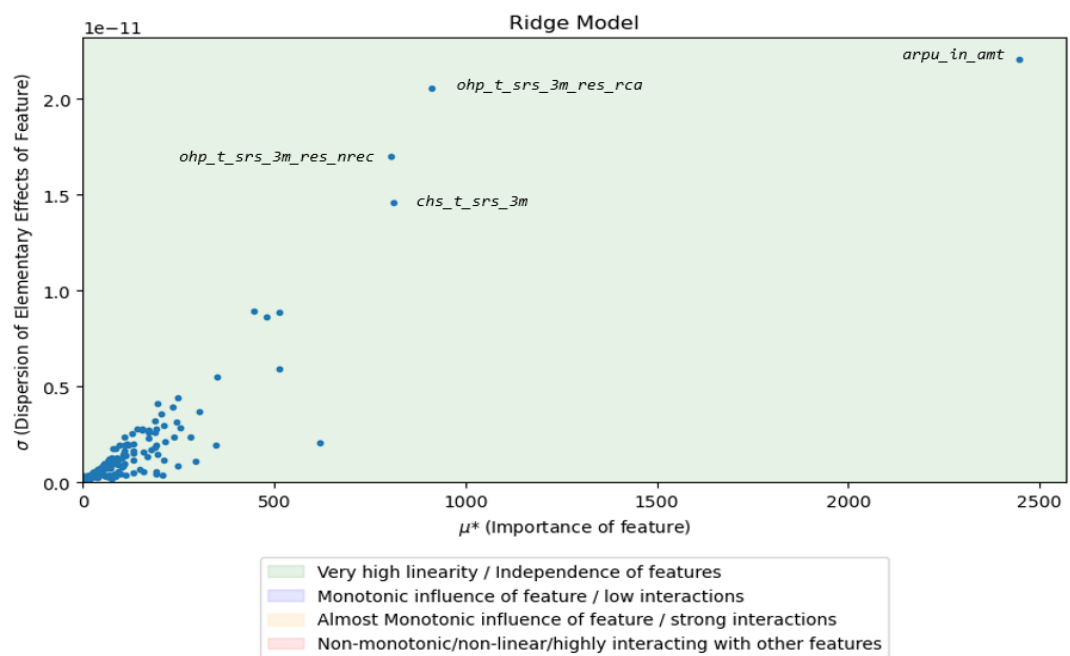


FIGURE B.1: Morris statistical variable plot (variance of elemental effects as a function of feature importance) for Ridge Model - Each point represents a feature used by the model

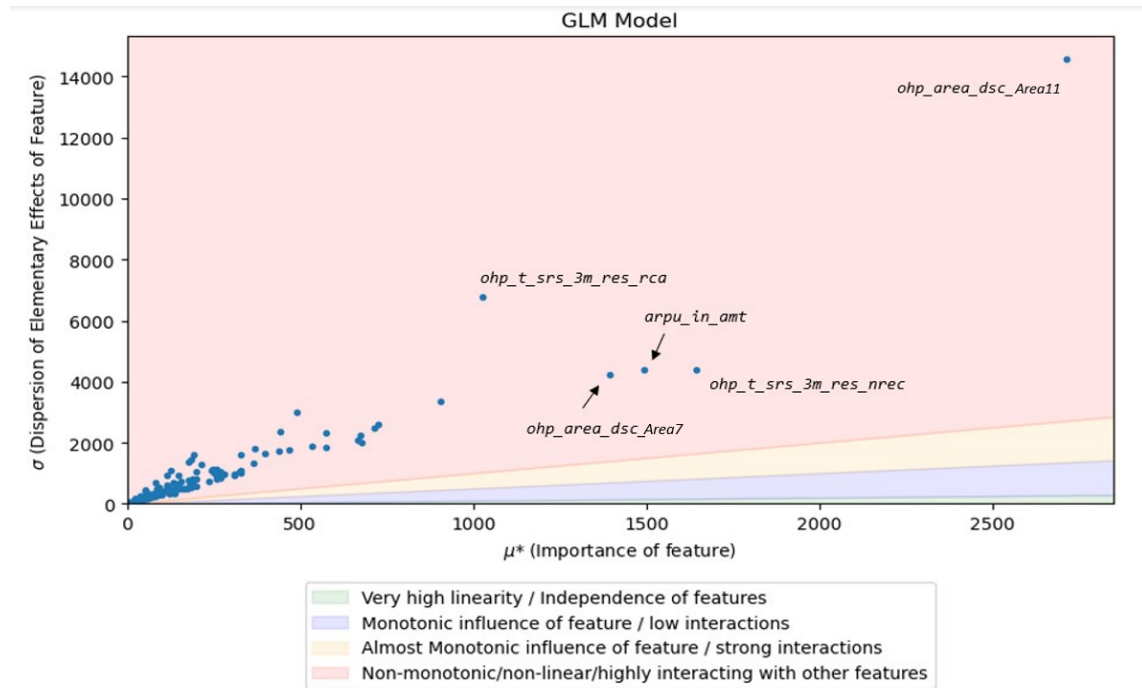


FIGURE B.2: Morris statistical variable plot (variance of elemental effects as a function of feature importance) for GLM Model - Each point represents a feature used by the model

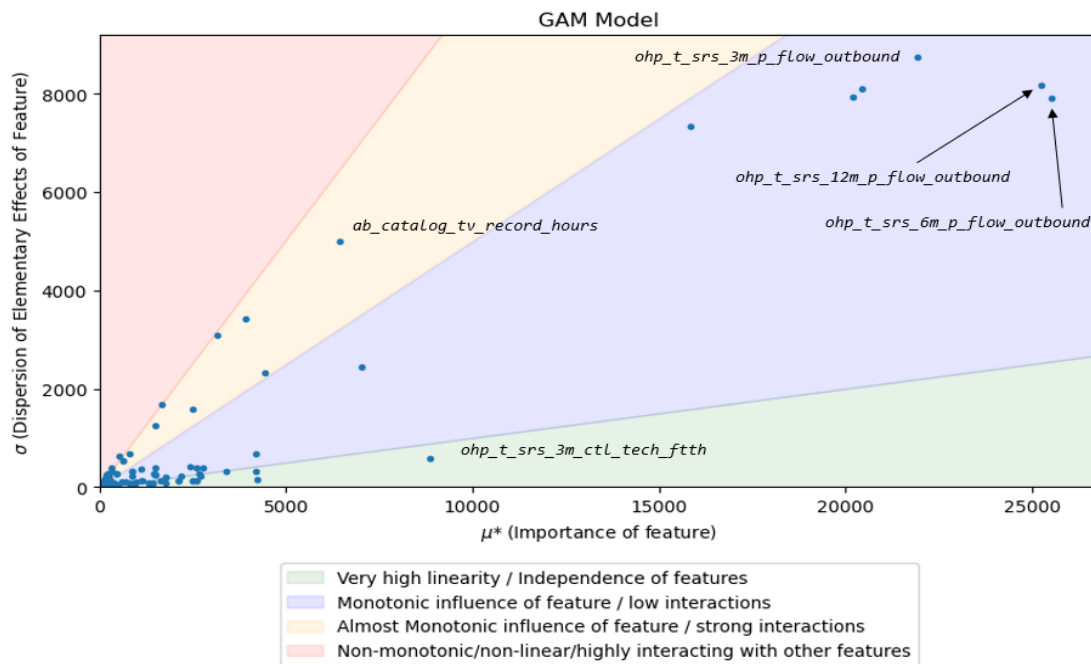


FIGURE B.3: Morris statistical variable plot (variance of elemental effects as a function of feature importance) for GAM Model - Each point represents a feature used by the model

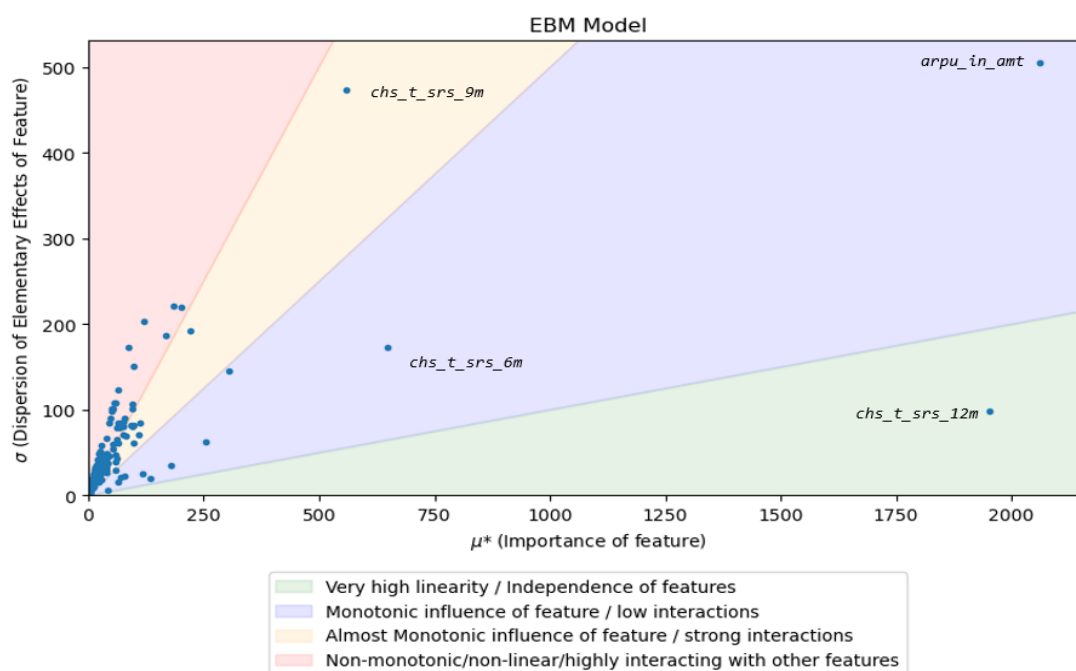


FIGURE B.4: Morris statistical variable plot (variance of elemental effects as a function of feature importance) for EBM Model - Each point represents a feature used by the model

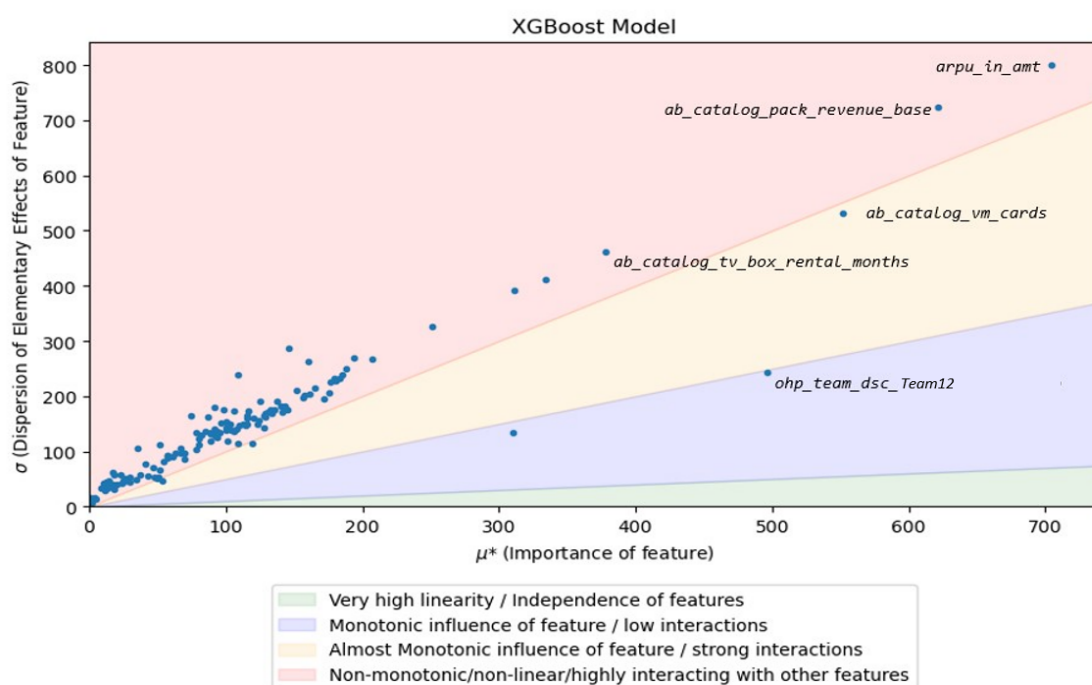


FIGURE B.5: Morris statistical variable plot (variance of elemental effects as a function of feature importance) for XGBoost Model - Each point represents a feature used by the model

Appendix C

Residual Trees and Error Analysis

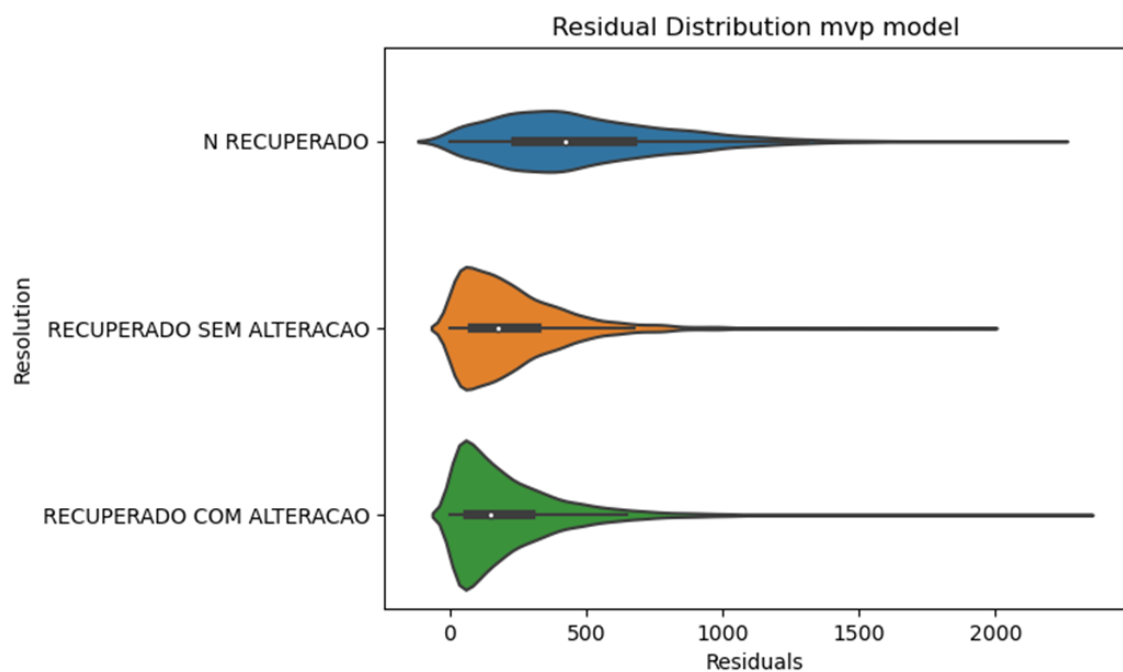


FIGURE C.1: Residual distribution for the MVP model segmented by resolution (meta-information)

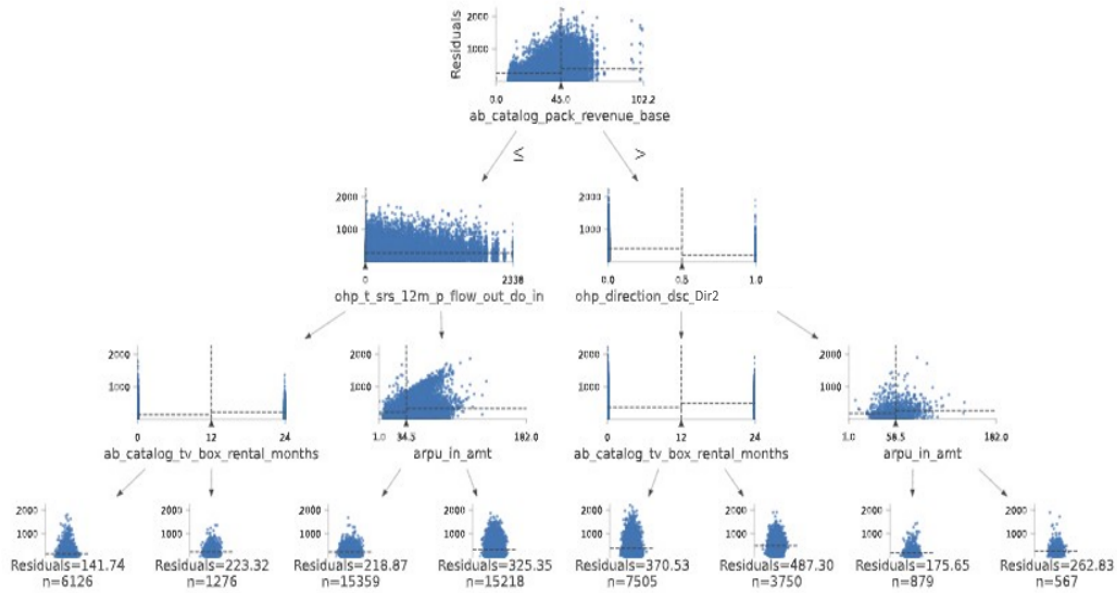


FIGURE C.2: Trained Residual Tree for Ridge Regression - Obtained in the test set

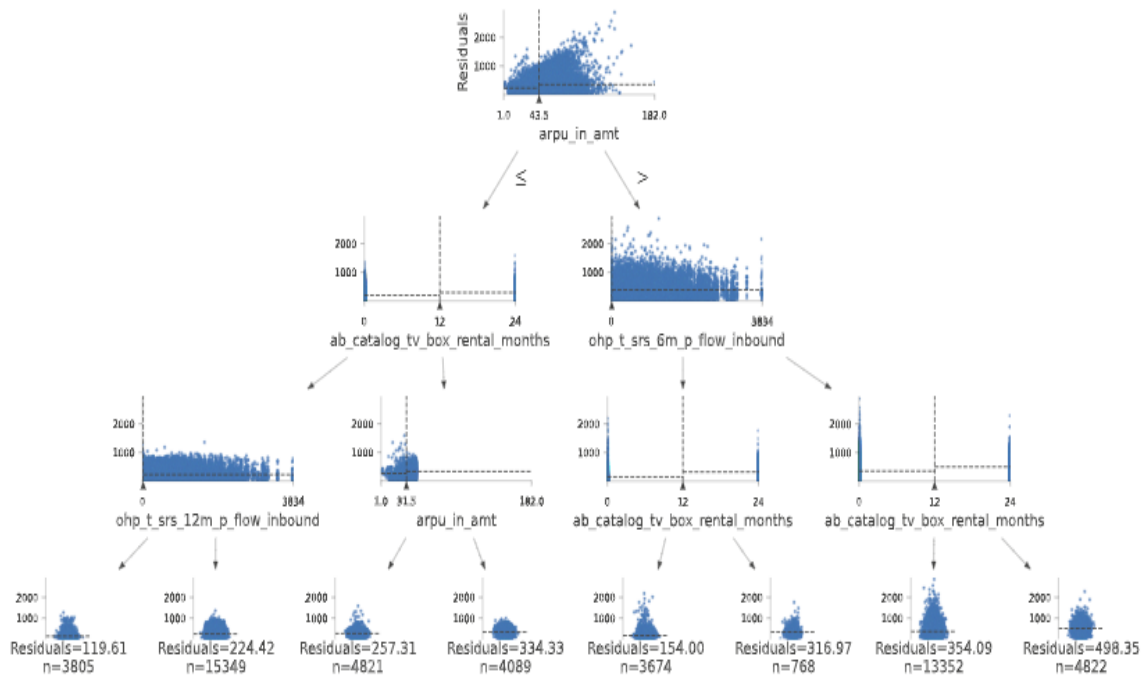


FIGURE C.3: Trained Residual Tree for GLM Model - Obtained in the test set

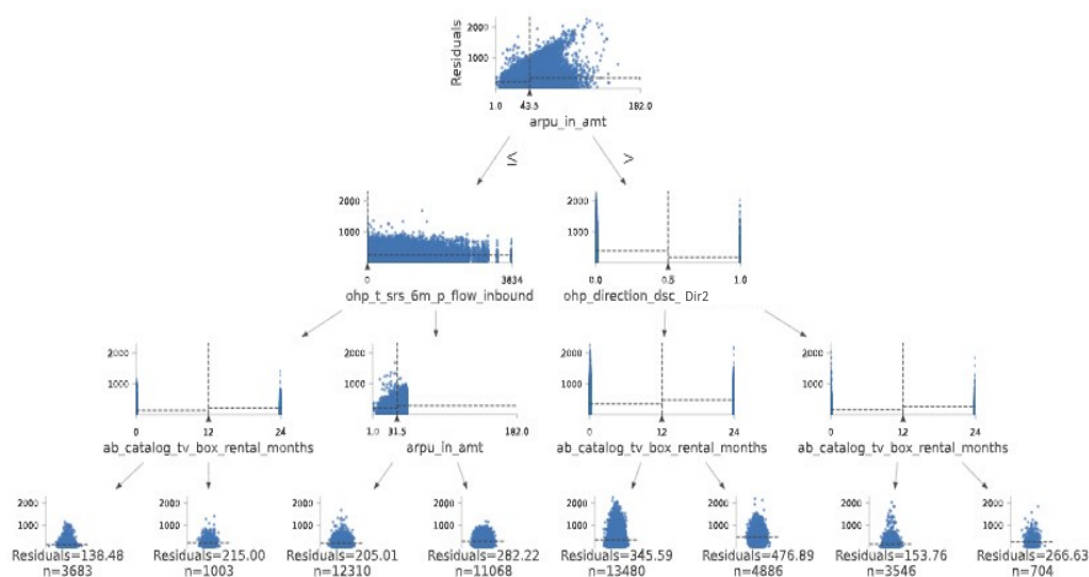


FIGURE C.4: Trained Residual Tree for GAM Model - Obtained in the test set

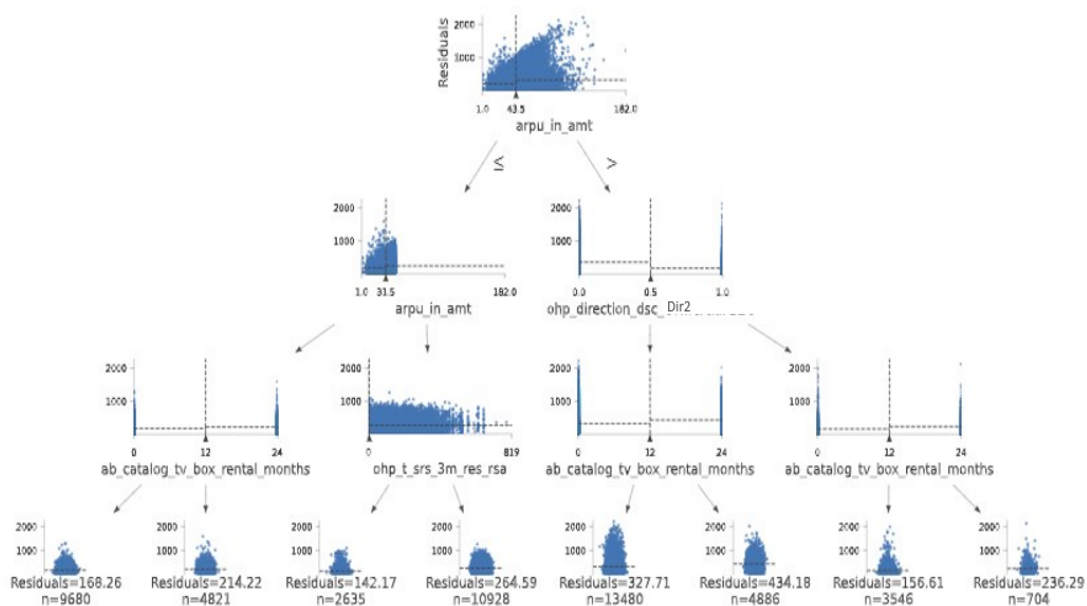


FIGURE C.5: Trained Residual Tree for EBM Model - Obtained in the test set

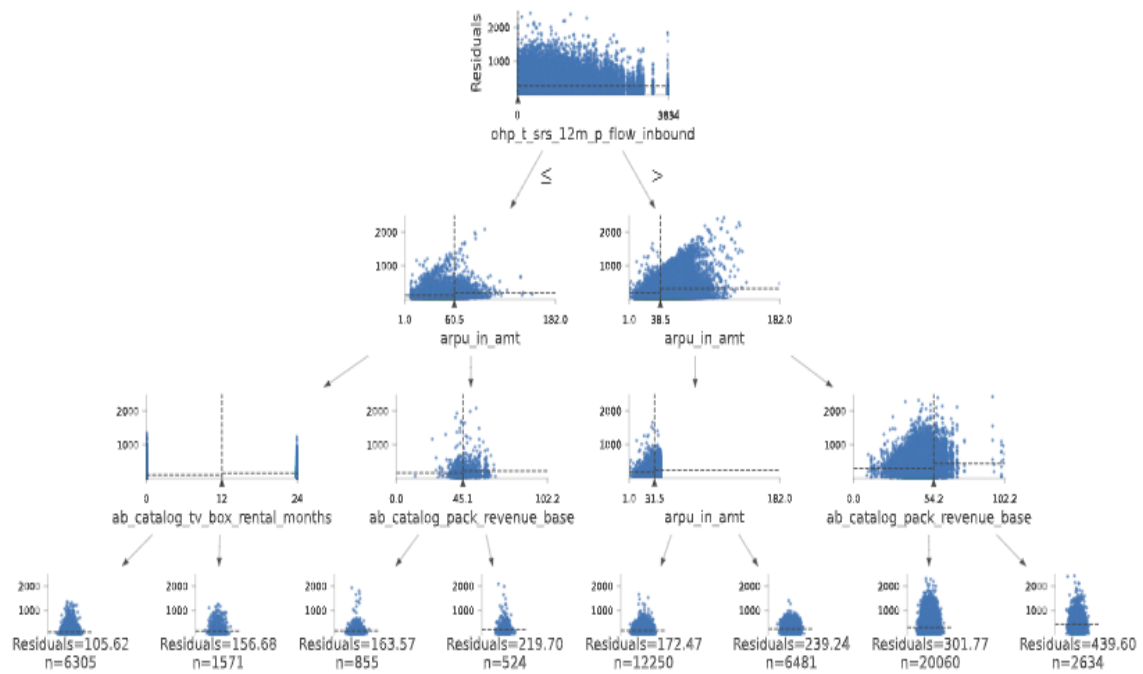


FIGURE C.6: Trained Residual Tree for XGBoost Model - Obtained in the test set

Bibliography

- [1] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, “Interpretable machine learning: Fundamental principles and 10 grand challenges,” 3 2021. [Online]. Available: <http://arxiv.org/abs/2103.11251> [Cited on page 4.]
- [2] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” pp. 1–20. [Cited on pages 4 and 12.]
- [3] H. Nori, “Interpretml: A unified framework for machine learning interpretability,” *ArXiv*, vol. abs/1909.09223, 2019. [Cited on page 4.]
- [4] “Responsible machine learning with error analysis,” <https://towardsdatascience.com/responsible-machine-learning-with-error-analysis-a7553f649915>, accessed 13-4-2023. [Cited on page 5.]
- [5] T. Gao, “Towards automated error analysis: Learning to characterize errors,” *ArXiv*, vol. abs/2201.05017, 2022. [Cited on page 5.]
- [6] *Introduction to Sensitivity Analysis*. John Wiley Sons, Ltd, 2007, ch. 1, pp. 1–51. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470725184.ch1> [Cited on page 5.]
- [7] A. Saltelli, “Sensitivity analysis for importance assessment,” *Risk Analysis*, vol. 22, no. 3, pp. 579–590, 2002. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/0272-4332.00040> [Cited on page 5.]
- [8] M. Shafique, M. Naseer, T. Theodoridis, C. Kyrkou, O. Mutlu, L. Orosa, and J. Choi, “Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead,” *CoRR*, vol. abs/2101.02559, 2021. [Online]. Available: <https://arxiv.org/abs/2101.02559> [Cited on pages 5 and 30.]

- [9] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods." Association for Computing Machinery, Inc, 2 2020, pp. 180–186. [Cited on page 6.]
- [10] A. M. TURING, "I.COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, 10 1950. [Online]. Available: <https://doi.org/10.1093/mind/LIX.236.433> [Cited on page 11.]
- [11] G. Wiederhold and J. McCarthy, "Arthur samuel: Pioneer in machine learning," *IBM Journal of Research and Development*, vol. 36, pp. 329 – 331, 06 1992. [Cited on page 11.]
- [12] "Columbia engineering - artificial intelligence (ai) vs. machine learning." [Cited on page 11.]
- [13] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable ai: A brief survey on history, research areas, approaches and challenges," in *Natural Language Processing and Chinese Computing*, J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan, Eds. Cham: Springer International Publishing, 2019, pp. 563–574. [Cited on page 11.]
- [14] W. R. Swartout, "Explaining and justifying expert consulting programs," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 815823. [Cited on page 11.]
- [15] A. C. Scott, W. J. Clancey, R. Davis, and E. H. Shortliffe, "Explanation capabilities of production-based consultation systems," *American Journal of Computational Linguistics*, pp. 1–50, Feb. 1977, microfiche 62. [Online]. Available: <https://aclanthology.org/J77-1006> [Cited on page 11.]
- [16] M. van Lent, W. Fisher, and M. Mancuso, "An explainable artificial intelligence system for small-unit tactical behavior," in *Proceedings of the 16th Conference on Innovative Applications of Artificial Intelligence*, ser. IAAI'04. AAAI Press, 2004, p. 900907. [Cited on page 11.]
- [17] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (xai)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 9 2018. [Cited on pages 11, 13, and 16.]

- [18] Z. C. Lipton, "The mythos of model interpretability," *Communications of the ACM*, vol. 61, pp. 35–43, 2018. [Cited on pages 11 and 14.]
- [19] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," 6 2017. [Online]. Available: <http://arxiv.org/abs/1706.07269> [Cited on page 12.]
- [20] C. Molnar, "Interpretable machine learning a guide for making black box models explainable." [Online]. Available: <http://leanpub.com/interpretable-machine-learning> [Cited on pages 12, 14, 16, 26, 29, 42, 43, and 80.]
- [21] A. Sudjianto and A. Zhang, "Designing inherently interpretable machine learning models," 2021. [Cited on pages 13 and 41.]
- [22] I. C. R.-T. de Oliveira, "Producing decisions and explanations: A joint approach towards explainable cnns," Master's Thesis, University Porto, Faculty of Engineering, 2019. [Cited on page 14.]
- [23] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89. [Cited on page 14.]
- [24] G. Schwalbe and B. Finzel, "A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts," 5 2021. [Online]. Available: <http://arxiv.org/abs/2105.07190><http://dx.doi.org/10.1007/s10618-022-00867-8> [Cited on page 15.]
- [25] A. Pérez, "The pragmatic turn in explainable artificial intelligence," *Minds and Machines*, vol. 29, no. 3, pp. 441–459, 2019. [Cited on page 15.]
- [26] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," pp. 245–317, 2021. [Cited on page 15.]
- [27] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," 2017. [Cited on page 16.]
- [28] T. Hailesilassie, "Rule extraction algorithm for deep neural networks: A review," 2016. [Cited on page 16.]

- [29] Q. Zhang, R. Cao, F. Shi, Y. N. Wu, and S.-C. Zhu, "Interpreting cnn knowledge via an explanatory graph." [Online]. Available: www.aaai.org [Cited on page 16.]
- [30] R. Calegari, G. Ciatto, and A. Omicini, "On the integration of symbolic and sub-symbolic techniques for xai: A survey," *Intelligenza Artificiale*, vol. 14, pp. 1–25, 09 2020. [Cited on page 16.]
- [31] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017. [Cited on pages 17 and 80.]
- [32] R. Calegari, G. Ciatto, and A. Omicini, "On the integration of symbolic and sub-symbolic techniques for xai: A survey," *Intelligenza Artificiale*, vol. 14, pp. 1–25, 09 2020. [Cited on page 17.]
- [33] X.-H. Li, Y. Shi, H. Li, W. Bai, Y. Song, C. C. Cao, and L. Chen, "Quantitative evaluations on saliency methods: An experimental study," 2020. [Cited on page 17.]
- [34] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," 8 2019. [Cited on page 17.]
- [35] C. Gillmann, D. Saur, and G. Scheuermann, "How to deal with uncertainty in machine learning for medical imaging?" in *2021 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*, 2021, pp. 52–58. [Cited on page 18.]
- [36] P. Hall, *Machine learning for high-risk applications: Techniques for responsible ai*. O'Reilly Media, Incorporated, 2023. [Cited on page 18.]
- [37] J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert, "Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models," 8 2018. [Online]. Available: <http://arxiv.org/abs/1808.00196><http://dx.doi.org/10.1109/TVCG.2018.2864499> [Cited on page 18.]
- [38] "Error analysis identify diagnose errors build responsibly - a toolkit to help analyze and improve model accuracy." [Online]. Available: <https://erroranalysis.ai> [Cited on pages 18 and 69.]
- [39] W. Wu, "Identify error-sensitive patterns by decision tree." [Cited on page 18.]
- [40] T. Gao, S. Singh, and R. J. Mooney, "Towards automated error analysis: Learning to characterize errors," 1 2022. [Online]. Available: <http://arxiv.org/abs/2201.05017> [Cited on pages 18 and 19.]

- [41] “Mlops notes 3.2: Error analysis for machine learning models.” [Online]. Available: <https://pub.towardsai.net/mlops-3-2-error-analysis-750ef03c96e9> [Cited on page 19.]
- [42] R. Tibshirani, “Conformal prediction,” 2023. [Cited on page 20.]
- [43] V. Taquet, V. Blot, T. Morzadec, L. Lacombe, and N. Brunel, “Mapie: an open-source library for distribution-free uncertainty quantification,” 7 2022. [Online]. Available: <http://arxiv.org/abs/2207.12274> [Cited on pages 20 and 72.]
- [44] “Theoretical description for conformity scores.” [Online]. Available: https://mapie.readthedocs.io/en/latest/theoretical_description_conformity_scores.html [Cited on page 20.]
- [45] Y. Romano, E. Patterson, and E. J. Candès, “Conformalized quantile regression,” 5 2019. [Online]. Available: <http://arxiv.org/abs/1905.03222> [Cited on page 20.]
- [46] H. Papadopoulos, V. Vovk, and A. Gammerman, “Regression conformal prediction with nearest neighbours,” pp. 815–840, 2011. [Cited on page 21.]
- [47] H. Boström and U. Johansson, “Mondrian conformal regressors,” in *International Symposium on Conformal and Probabilistic Prediction with Applications*, 2020. [Online]. Available: <https://api.semanticscholar.org/CorpusID:221606511> [Cited on page 21.]
- [48] U. Johansson, H. Boström, T. Löfström, and H. Linusson, “Regression conformal prediction with random forests,” vol. 97. Kluwer Academic Publishers, 2014, pp. 155–176. [Cited on page 21.]
- [49] S. Ghosh, Q. V. Liao, K. N. Ramamurthy, J. Navratil, P. Sattigeri, K. R. Varshney, and Y. Zhang, “Uncertainty quantification 360: A holistic toolkit for quantifying and communicating the uncertainty of ai,” 6 2021. [Online]. Available: <http://arxiv.org/abs/2106.01410> [Cited on page 21.]
- [50] S. Razavi, A. Jakeman, A. Saltelli, C. Prieur, B. Iooss, E. Borgonovo, E. Plischke, S. Lo Piano, T. Iwanaga, W. Becker, S. Tarantola, J. H. Guillaume, J. Jakeman, H. Gupta, N. Melillo, G. Rabitti, V. Chabridon, Q. Duan, X. Sun, S. Smith, R. Sheikholeslami, N. Hosseini, M. Asadzadeh, A. Puy, S. Kucherenko, and H. R. Maier, “The future of sensitivity analysis: An essential discipline for

- systems modeling and policy support,” *Environmental Modelling Software*, vol. 137, p. 104954, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364815220310112> [Cited on page 21.]
- [51] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener, “Sensitivity analysis of environmental models: A systematic review with practical workflow,” *Environmental Modelling Software*, vol. 79, pp. 214–232, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364815216300287> [Cited on page 21.]
- [52] P. Cortez and M. J. Embrechts, “Opening black box data mining models using sensitivity analysis.” [Cited on page 21.]
- [53] A. Razmjoo, P. Xanthopoulos, and Q. P. Zheng, “Online feature importance ranking based on sensitivity analysis,” *Expert Systems with Applications*, vol. 85, pp. 397–406, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417417303329> [Cited on pages 21 and 22.]
- [54] S. M. Lundberg and S. Lee, “A unified approach to interpreting model predictions,” *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: <http://arxiv.org/abs/1705.07874> [Cited on pages 21, 23, 26, and 39.]
- [55] N. V. Stein, E. Raponi, Z. Sadeghi, R. V. Ham, B. V. Stein, N. Bouman, R. C. H. J. V. Ham, and T. Bäck, “A comparison of global sensitivity analysis methods for explainable ai with an application in genomic prediction experience-based computation: Learning to optimise (ecole-itn) view project discovering the preference hypervolume view project a comparison of global sensitivity analysis methods for explainable ai with an application in genomic prediction.” [Online]. Available: <https://www.researchgate.net/publication/363080132> [Cited on page 21.]
- [56] A. A. Saltelli, *Global sensitivity analysis : the primer*. John Wiley, 2008. [Cited on page 22.]
- [57] M. D. Morris, “Factorial sampling plans for preliminary computational experiments,” *Technometrics*, vol. 33, no. 2, pp. 161–174, 1991. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1080/00401706.1991.10484804> [Cited on pages 22, 25, and 39.]

- [58] S. Kucherenko and B. Iooss, *Derivative-Based Global Sensitivity Measures*. Cham: Springer International Publishing, 2016, pp. 1–24. [Online]. Available: https://doi.org/10.1007/978-3-319-11259-6_36-1 [Cited on page 22.]
- [59] I. Sobol and S. Kucherenko, “Derivative based global sensitivity measures,” *Procedia - Social and Behavioral Sciences*, vol. 2, no. 6, pp. 7745–7746, 2010, sixth International Conference on Sensitivity Analysis of Model Output. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877042810013492> [Cited on page 22.]
- [60] I. Sobol, “Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates,” *Mathematics and Computers in Simulation*, vol. 55, no. 1, pp. 271–280, 2001, the Second IMACS Seminar on Monte Carlo Methods. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0378475400002706> [Cited on page 22.]
- [61] R. I. Cukier, C. M. Fortuin, K. E. Shuler, A. G. Petschek, and J. H. Schaibly, “Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. I. Theory,” , vol. 59, no. 8, pp. 3873–3878, Oct. 1973. [Cited on page 22.]
- [62] A. Saltelli, S. Tarantola, and K. P.-S. Chan, “A quantitative model-independent method for global sensitivity analysis of model output,” *Technometrics*, vol. 41, no. 1, pp. 39–56, 1999. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1080/00401706.1999.10485594> [Cited on page 22.]
- [63] E. Plischke, E. Borgonovo, and C. L. Smith, “Global sensitivity measures from given data,” *Eur. J. Oper. Res.*, vol. 226, pp. 536–550, 2013. [Online]. Available: <https://api.semanticscholar.org/CorpusID:1126558> [Cited on page 22.]
- [64] F. Pianosi and T. Wagener, “A simple and efficient method for global sensitivity analysis based on cumulative distribution functions,” *Environmental Modelling Software*, vol. 67, pp. 1–11, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1364815215000237> [Cited on page 22.]
- [65] “An introduction to shap values and machine learning interpretability,” accessed 13-4-2023. [Online]. Available: <https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability> [Cited on page 23.]

- [66] M. Yang and B. Kim, "Benchmarking attribution methods with relative feature importance," *arXiv: Learning*, 2019. [Online]. Available: <https://api.semanticscholar.org/CorpusID:207780661> [Cited on page 23.]
- [67] A. phi Nguyen and M. R. Martínez, "On quantitative aspects of model interpretability," 7 2020. [Online]. Available: <http://arxiv.org/abs/2007.07584> [Cited on pages 23, 24, and 65.]
- [68] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf> [Cited on page 24.]
- [69] O. Tsvetkova and T. B. Ouarda, "A review of sensitivity analysis practices in wind resource assessment," *Energy Conversion and Management*, vol. 238, p. 114112, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0196890421002880> [Cited on page 25.]
- [70] M. Andersson, M. Streb, J. Y. Ko, V. Löfqvist Klass, M. Klett, H. Ekström, M. Johansson, and G. Lindbergh, "Parametrization of physics-based battery models from inputoutput data: A review of methodology and current research," *Journal of Power Sources*, vol. 521, p. 230859, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0378775321013458> [Cited on page 25.]
- [71] "Salib - python library." [Online]. Available: https://gsa-module.readthedocs.io/en/stable/implementation/morris_screening_method.html#id20 [Cited on page 25.]
- [72] F. Campolongo, A. Saltelli, and J. Cariboni, "From screening to quantitative sensitivity analysis. a unified approach," *Computer Physics Communications*, vol. 182, pp. 978–988, 4 2011. [Cited on pages 26 and 62.]

- [73] D. G. Sanchez, B. Lacarrière, M. Musy, and B. Bourges, "Application of sensitivity analysis in building energy simulations: Combining first- and second-order elementary effects methods," *Energy and Buildings*, vol. 68, pp. 741–750, 1 2014. [Cited on pages 26 and 62.]
- [74] K. Menberg, Y. Heo, and R. Choudhary, "Sensitivity analysis methods for building energy models: Comparing computational costs and extractable information b-bem: The bayesian building energy management portal view project sensitivity analysis methods for building energy models: Comparing computational costs and extractable information," *Energy and Buildings*, vol. 133, pp. 433–445, 2016. [Online]. Available: <http://creativecommons.org/licenses/by/4.0/> [Cited on pages 26 and 62.]
- [75] "Shap - python library," accessed 13-4-2023. [Online]. Available: <https://pypi.org/project/shap/> [Cited on pages 26, 30, and 39.]
- [76] E. Algaba, V. Fragnelli, and J. Sánchez-Soriano, "Handbook of the shapley value." [Cited on page 27.]
- [77] R. Geada, "Trustyai shap: Overview and examples." [Online]. Available: <https://blog.kie.org/2021/11/trustyai-shap-overview-and-examples.html> [Cited on pages xi and 28.]
- [78] "Understanding the shap interpretation method: Kernel shap," accessed September of 2023. [Online]. Available: https://data4thought.com/kernel_shap.html [Cited on page 30.]
- [79] P. J. Huber, *Robust statistics*. New York: John Wiley Sons, Inc., year =1981, ISBN = 0-471-41805-6. [Cited on page 30.]
- [80] D. de Menezes, D. Prata, A. Secchi, and J. Pinto, "A review on robust m-estimators for regression analysis," *Computers Chemical Engineering*, vol. 147, p. 107254, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0098135421000326> [Cited on page 31.]
- [81] 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 31.]

- [82] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, pp. 121–148, 11 2010. [Cited on page 31.]
- [83] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," 4 2018. [Online]. Available: <http://arxiv.org/abs/1804.00308> [Cited on page 31.]
- [84] H. Hu and J. Pang, "Stealing machine learning models: Attacks and countermeasures for generative adversarial networks." Association for Computing Machinery, 12 2021, pp. 1–16. [Cited on page 32.]
- [85] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," 10 2016. [Online]. Available: <http://arxiv.org/abs/1610.05820> [Cited on page 32.]
- [86] "Adversarial robustness toolbox (art) for machine learning security." <https://adversarial-robustness-toolbox.readthedocs.io/en/latest/index.html>, accessed 13-4-2023. [Cited on pages xi, 32, and 33.]
- [87] R. S. Siva Kumar, M. Nyström, J. Lambert, A. Marshall, M. Goertzel, A. Comis-soneru, M. Swann, and S. Xia, "Adversarial machine learning-industry perspectives," in *2020 IEEE Security and Privacy Workshops (SPW)*, 2020, pp. 69–75. [Cited on page 32.]
- [88] B. Dickson, ""machine learning: What are membership inference attacks?"," <https://bdtechtalks.com/2021/04/23/machine-learning-membership-inference-attacks/>, accessed 19-06-2023. [Cited on page 32.]
- [89] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," 8 2017. [Online]. Available: <http://arxiv.org/abs/1708.06131>http://dx.doi.org/10.1007/978-3-642-40994-3_25 [Cited on page 33.]
- [90] S. Hidano, T. Murakami, and Y. Kawamoto, "Transmia: Membership inference attacks using transfer shadow training," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–10. [Cited on page 33.]

- [91] "Python interpretable machine learning - a python toolbox for interpretable machine learning." https://selfexplainml.github.io/PiML-Toolbox/_build/html/guides/testing/robustness.html, accessed 19-06-2023. [Cited on page 34.]
- [92] A. Shafahi, A. Ghiasi, F. Huang, and T. Goldstein, "Label smoothing and logit squeezing: A replacement for adversarial training?" 10 2019. [Online]. Available: <http://arxiv.org/abs/1910.11585> [Cited on page 34.]
- [93] in *Adversarial Robustness for Machine Learning*, P.-Y. Chen and C.-J. Hsieh, Eds. Academic Press, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128240205000351> [Cited on page 35.]
- [94] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2015. [Cited on page 35.]
- [95] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 6 2017. [Online]. Available: <http://arxiv.org/abs/1706.06083> [Cited on page 35.]
- [96] A. T. Nguyen, E. Raff, and B. A. Hamilton, "Adversarial attacks, regression, and numerical stability regularization." [Cited on page 35.]
- [97] A. H. Ribeiro and T. B. Schön, "Overparameterized linear regression under adversarial attacks," 4 2022. [Online]. Available: <http://arxiv.org/abs/2204.06274><http://dx.doi.org/10.1109/TSP.2023.3246228> [Cited on page 35.]
- [98] "Error analysis - identify diagnose errors build responsibly. (a toolkit to help analyze and improve model accuracy)," accessed 19-06-2023. [Online]. Available: <https://erroranalysis.ai> [Cited on page 39.]
- [99] "Scikit-opt - swarm intelligence in python." accessed 19-06-2023. [Online]. Available: https://scikit-opt.github.io/scikit-opt/#/en/README?id=_1-differential-evolution [Cited on pages 39 and 73.]
- [100] J. A. Nelder and R. W. M. Wedderburn, "Generalized linear models," *Journal of the Royal Statistical Society. Series A (General)*, vol. 135, no. 3, pp. 370–384, 1972. [Online]. Available: <http://www.jstor.org/stable/2344614> [Cited on page 43.]

- [101] C. E. McCulloch, "Generalized linear models," *Journal of the American Statistical Association*, vol. 95, no. 452, pp. 1320–1324, 2000. [Online]. Available: <http://www.jstor.org/stable/2669780> [Cited on page 43.]
- [102] M. Müller, "Generalized linear models." 2004. [Cited on pages 44 and 46.]
- [103] "Generalized linear models: Advanced methods for data analysis(36-402/36-608). spring 2014." retrieved April 19, 2021. [Online]. Available: <https://www.stat.cmu.edu/~ryantibs/advmethods/notes/glm.pdf> [Cited on page 44.]
- [104] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*, 02 2009. [Cited on pages 46 and 48.]
- [105] D. Servén and C. Brummitt, "pygam: Generalized additive models in python," Mar. 2018. [Online]. Available: <https://doi.org/10.5281/zenodo.1208724> [Cited on pages xi and 47.]
- [106] H. Nori, S. Jenkins, P. Koch, and R. Caruana, "Interpretml: A unified framework for machine learning interpretability," 9 2019. [Online]. Available: <http://arxiv.org/abs/1909.09223> [Cited on pages 47 and 48.]
- [107] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*. ACM Press, 2013. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/accurate-intelligible-models-pairwise-interactions/> [Cited on page 47.]
- [108] Y. Lou, R. Caruana, and J. Gehrke, "Intelligible models for classification and regression," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2012. [Cited on page 47.]
- [109] T. Chen and C. Guestrin, "Xgboost : Reliable large-scale tree boosting system," 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:33260578> [Cited on pages 48 and 49.]
- [110] A. Boschetti and L. Massaron, *Python Data Science Essentials: A practitioners guide covering essential data science principles, tools, and techniques, 3rd Edition*. Packt

- Publishing, 2018. [Online]. Available: <https://books.google.pt/books?id=cP1wDwAAQBAJ> [Cited on pages 48 and 49.]
- [111] 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 48.]
- [112] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," 08 2016, pp. 785–794. [Cited on page 49.]
- [113] A. phi Nguyen and M. R. Martínez, "Mononet: Towards interpretable models by learning monotonic features," 2019. [Cited on page 49.]
- [114] J. Trindade, J. Vinagre, K. Fernandes, N. Paiva, and A. Jorge, "Partially monotonic learning for neural networks." [Cited on pages 49 and 80.]
- [115] "One-way anova - scipy documentation." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.f_oneway.html [Cited on page 53.]
- [116] <https://www.investopedia.com/terms/v/variance-inflation-factor.asp>. [Cited on page 54.]
- [117] "Gaussian kde - scipy open-source software for mathematics, science, and engineering." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.gaussian_kde.html [Cited on page 65.]
- [118] "Integration through trapezoid method - scipy open-source software for mathematics, science, and engineering." [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.trapezoid.html> [Cited on page 66.]
- [119] J. Devore, K. Berk, and M. Carlton, *Modern Mathematical Statistics with Applications*, 01 2021. [Cited on page 66.]
- [120] "Spearman correlation coefficient - scipy open-source software for mathematics, science, and engineering." [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html> [Cited on page 67.]
- [121] Accessed: February 2023. [Online]. Available: <https://github.com/parrrt/dtreviz> [Cited on page 69.]

- [122] S. Mitchell, E. Potash, S. Barocas, A. D'Amour, and K. Lum, "Algorithmic fairness: Choices, assumptions, and definitions," *Annual Review of Statistics and Its Application*, vol. 8, no. 1, pp. 141–163, 2021. [Online]. Available: <https://doi.org/10.1146/annurev-statistics-042720-125902> [Cited on page 80.]
- [123] S. Verma and J. Rubin, "Fairness definitions explained." IEEE Computer Society, 5 2018, pp. 1–7. [Cited on page 80.]