UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Daniel de Moura e Almeida

Understanding model performance

Belo Horizonte
2025

Daniel de Moura e Almeida

Understanding model performance

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Renato Vimieiro

Belo Horizonte
2025

Almeida, Daniel de Moura e.

A447u Understanding model performance [recurso eletrônico] / Daniel de Moura e Almeida. Belo Horizonte – 2025.
1 recurso online (51 f. il., color.) : pdf.

Orientador: Renato Vimieiro.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 47-51.

1. Computação – Teses. 2. Aprendizado do computador – Teses. 3. Ciência de dados – Teses. 4. Controle preditivo – Teses. 5. Inteligência artificial – Teses. I. Vimieiro, Renato. . II. Universidade Federal de Minas Gerais Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Understanding Model Performance

DANIEL DE MOURA E ALMEIDA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Renato Vimieiro

PROF. RENATO VIMIEIRO - Orientador
Departamento de Ciência da Computação - UFMG

R I R L :.

PROF. RICARDO BASTOS CAVALCANTE PRUDENCIO
Centro de Informática - UFPE

Wagner meira jun

PROF. WAGNER MEIRA JÚNIOR
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 13 de março de 2025.

Acknowledgments

First I would like to thank my advisor, Professor Renato, for all the help and all technical, scientific and even motivational advice during the last years. I have learned a lot from you, and cannot express how much your patience and perseverance have helped me reach this mark in my academic formation, and also grow as a researcher and professional.

To my family, thank you for all the love and emotional support during this journey, conciliating work and academic formation. To Mari, I cannot express how important you were for this achievement; without your help, discipline, and motivation, I would not have made it so far. Thank you very much, my love.

And for all others involved in this work, I also would like to express my thanks to Professor Ricardo for all the helpful discussions and ideas during the development of the pipeline, Professor Wagner for the valuable suggestions and for the participation in my thesis' committee, and finally Luis and Igor for the insights into the practical aspects of the pipeline as an open-source tool for the community.

Resumo

Este trabalho introduz um novo *pipeline* de avaliação de modelos de aprendizado de máquina supervisionado, construído para identificar e descrever regiões em um conjunto de dados nas quais os modelos exibem um erro excepcional em suas previsões. Por meio da integração de técnicas de descoberta de subgrupos e visualização de dados, o *pipeline* fornece um entendimento mais granular sobre a performance preditiva do modelo, detectando padrões de erros não triviais, que métricas globais mais tradicionais podem não elucidar. A abordagem é agnóstica ao tipo de modelo e interpretável, oferecendo *insights* valiosos sobre os subgrupos dos dados que contribuem desproporcionalmente para as taxas de erro globais. Resultados experimentais demonstram a efetividade do método em contribuir com a avaliação de modelos, ao aplicá-lo a conjuntos de dados bem conhecidos. Este trabalho contribui para o campo de explicabilidade em IA ao fornecer uma ferramenta prática e *open-source* para análise de performance e identificação de melhorias em modelos preditivos.

Palavras-chave: aprendizado de máquina; ciência de dados; descoberta de subgrupos; explicabilidade em IA.

Abstract

This work introduces a novel evaluation pipeline for supervised machine learning models, designed to identify and describe regions within a dataset where models exhibit exceptional predictive errors. By integrating Subgroup Discovery and Data Visualization techniques, the pipeline provides a more granular understanding of model performance, uncovering non-trivial error patterns that conventional global metrics may obscure. The approach is model-agnostic and interpretable, offering valuable insights into data subgroups that disproportionately contribute to overall error rates. Experimental results demonstrate the pipeline's effectiveness in enhancing model evaluation, by applying it to well-known datasets. This work contributes to Explainable AI (xAI) by providing an open source, practical tool for targeted model improvements and performance analysis.

Keywords: machine learning; data science; subgroup discovery; explainable AI.

Contents

Chapter 1

Introduction

Machine Learning (ML) models have been widely used for quite some time now, and they are increasingly present in people's lives. From the expected time of arrival on our way to work, up to the limits of our credit cards, many aspects of the modern world are powered by ML models, and with the increasing availability of data to train them, they are becoming a go-to solution for many problems. One of the reasons that result in this ubiquity is that we can use them to solve a plethora of tasks: classification, regression, clustering, anomaly detection and item recommendation are some of the most famous ML tasks, and the variety of these applications often results in challenges in building, maintaining and evaluating all these models. Especially in a rapidly evolving world, the dynamic nature of most data sources forces us to constantly question whether the models are still good enough. The problem is that even determining what can be considered good performance may not be as simple as it seems.

In an effort to quantify the fitness of models, many performance metrics have been been proposed. Depending on the task at hand, different metrics are used. In the context of Supervised Learning, measuring the predictive performance of a model on a given dataset is typically quite straightforward. For classification problems, Accuracy, Precision, Recall, F-Measure and area under the Receiving Operating Characteristic (ROC) curve [15] are some of the most used ones, while Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) [5] are some of the most commonly used metrics for regression problems. While they provide a way to objectively calculate how good a model is at a specific task, they are, in general, global measures and make use of aggregated (usually averaged) metrics to summarize model performance.

The problem with using only global measures to determine whether a model is good enough is that there may be cases where the model performs unacceptably badly, but by averaging the performance, their effect on the final metric is not significant. There are many examples like that in safety-critical applications, such as medical [19] and autonomous driving [37] problems. Imagine an autonomous driving model that detects whether there is a person in front of the car or not, and for 99% of the cases it predicts the output correctly, but when the person in front of the car is wearing a shiny purple

coat, it incurs a false negative. Even having 99% accuracy, the model may not be good enough for this application, since this false negative may result in running over this person in a real-world scenario. Although objective metrics are reasonably consolidated as a way to measure model performance, they are not enough to conclude whether a model is adequate for a certain use case, since they often cannot translate how reliable it can be. Model evaluation should be, in this sense, viewed from a more holistic perspective.

Another way to increase confidence in the predictions of an ML model is to understand the reasoning it used to reach a given output. But with the advancement of computing power, ML models are becoming increasingly complex, and while this typically allows them to achieve better predictive performance, it also imposes a challenge in understanding the reasons behind the model's predictions. Interpreting the output of a deep learning model, composed of multiple layers of neurons connected to each other, that learn high-level abstract concepts such as the presence of emotion in a picture of a person's face, is significantly more difficult than that of a simple Logistic Regression, where the coefficients themselves show how much each feature contributes to the final output. Moreover, this extra complexity also enables models to learn decision boundaries (for instance, in classification problems) that are extremely difficult to visualize and understand, and which frequently lead to scenarios such as the example described above: good average predictive performance, but with obvious errors that render them useless for practical purposes.

So, in face of this challenge, Explainable Artificial Intelligence (xAI) has been a major topic of research in recent years [14], and its main goal is to create ways to interpret the output of complex, and often opaque-box models. This field of study focuses on explaining the reasons behind the model's predictions, and generally follows two lines of thought. The first is the use of local surrogate models that are simple enough to interpret directly, and that approximate the behavior of the original model on a certain prediction's vicinity. A well-known example of this type of method is LIME [38] (Local Interpretable Model-agnostic Explanations), and it typically makes use of interpretable models such as linear (or logistic) regression, where we can inspect the values of the coefficients, and decision trees, which are naturally similar to the human decision-making process. The problem with this approach is that the explanations are local, and each surrogate model is applicable only to a small part of the data, which makes it difficult to build a bigger picture of the reasons behind the model's predictions. The second are methods that attribute some kind of weight on the features values, estimating the impact each one had on the final model predictions, also called feature importance. SHAP [27] (Shapley values) are one of the most widely used methods of this kind, in which principles of game theory are applied to measure the additive impact of each feature on the output of the final model. While these indications can be extremely useful in understanding the features that influence the model the most, they still provide only a limited contribution to the

advancement of our understanding of why a model is generating an incorrect output in a certain situation.

Since ML models can be seen as a simplified representation of the original system that generated the data used to train them, errors are, of course, expected. And by diving deeper into the situations in which the model is struggling to predict correctly, we can have a better understanding of its overall performance. This can be seen from two perspectives: analyzing how the predictions differ from the truth, and analyzing the behavior of the data that leads to the model errors. One area of research that focuses on the model's predictions to enhance the confidence we can have on its predictions is the field of Conformal Prediction [39]. The main goal in this case is to determine a region around the predicted value where the true label is going to be present with high probability. It can be done both for regression, where more intuitively one could estimate a confidence interval around the original prediction, but also for classification, by predicting, instead of a single one, a set of classes in a multi-class classification setup, for example. Another, which aims to measure how confident we can be in the output of a model, is the field of Uncertainty Quantification [1]. Beyond the quantitative analysis of the uncertainty, it is often also useful to characterize the types of uncertainty that are affecting the model's confidence or performance. They are generally distinguished [16] as aleatoric uncertainty, the kind of uncertainty that is inherent to the data (or to the process that generates it), and therefore cannot be reduced, and epistemic uncertainty, the uncertainty related to the choice of the model, data collection, lack of quality or any other factors that can be, in one way or another, reduced by either collecting more data or changing the assumptions made in the model fitting. Knowing the type of uncertainty that is affecting a model's predictions can also give guidance as to whether it can be improved, rather than accepting the uncertainty as natural to that kind of problem. But both of these strategies for explaining model performance focus more on enhancing the model's predictions (by turning it into a prediction region, by attributing them a trust score, or characterizing the type of uncertainty in it) than on the main ingredient of all ML recipes: the data.

1.1 Aims

In this sense, characterizing the data that typically lead to higher errors becomes a crucial part of model evaluation, since it will also point out the situations in which the model can be least trusted. The goal of this work is to create a pipeline that aids the performance analysis of data professionals by finding and describing, in an interpretable fashion, the regions in a dataset where a given pre-trained ML model incurs in the most

errors in its predictions. In order for the pipeline to be model-agnostic, and thus generalizable and widely applicable, we infer the level of error the model incurs when predicting instances located in certain regions, and we characterize these regions by defining association rules that correlate intervals in the feature values and an exceptional performance. But the definition of exceptional performance may not be as straightforward, since the model's errors can be in different orders of magnitude. So the task becomes not only to search over regions in the dataset, but also to compare and iteratively update the interestingness levels found among the errors in the predictions of the instances the regions cover.

This problem is common in the data mining field, and one of the methods we can use to search this space for somewhat special groups is Subgroup Discovery [2]. This technique focuses on skimming through a dataset, detecting and characterizing regions that are distinguishable with respect to a given quality function. By using a measure of prediction error as a quality function, we can use a subgroup discovery algorithm to mine the regions where the predictive performance is exceptionally bad (or good) and enhance model evaluation. This approach has the advantage of being flexible enough to cover both local, more specific patterns in model performance (such as the autonomous car driving example cited above), but also more global patterns, only by calibrating the importance of the size of the subgroup in the chosen algorithm.

After all, explaining the regions where an ML model has poor performance can be seen as a kind of xAI, and given its importance, it should be part of the process of model evaluation. This work proposes an evaluation pipeline that makes use of subgroup discovery techniques to mine well-defined regions (slices) in a dataset, where a given ML model (or group of models) has exceptional performance. We propose a multi-step pipeline that aims to (i) return a set of non-redundant, human-readable subgroups that describe regions where a model has either especially good or bad predictive performance, (ii) visualize how the mined subgroups interact with samples in the dataset that the model was evaluated with, and (iii) compare the exceptional performance regions found for two or more models trained on the same dataset, giving indications as to what strengths and weaknesses each model has, and also a notion of instance hardness. By building a tool to run the designed pipeline automatically given a dataset, a pre-trained model (or set of models), and a performance metric, we can increase the productivity of data science professionals, reducing the time spent on the step of model evaluation, since this work also aspires to make this tool open-source and available to all practitioners in the community.

This work is organized as such: following this introduction, Chapter 2 delves into the related work, setting the stage by discussing existing literature and research in the field. Chapter 3 lays the foundational concepts and definitions crucial for the comprehension of the subsequent material. In Chapter 4 the research methods employed are detailed, offering insight into the procedural framework of the study. Chapter 6 showcases

the practical implications and real-world applications of the proposed pipeline. Finally, Chapter 7 synthesizes the research outcomes, reflecting on the implications, limitations, and potential avenues for future research.

Chapter 2

Related Work

With the recent rise in articles concerning xAI, there is some literature regarding how to search for patterns that lead to higher or lower errors, in the context of Supervised Learning. Duivesteijn and Thaele [10] proposed the SCaPE method, a type of Exceptional Model Mining [25] class that strives to obtain interpretable subgroups of exceptional performance. It focuses on finding subgroups where a soft classifier, one that returns continuous values for each class rather than just the class prediction, aligns exceptionally well or poorly with the ground truth. This is achieved by introducing the use of the Average Ranking Loss metric, a way to measure model performance according to the number of true positives that received a lower score than true negatives. Then, by using a quality measure that compares the values of the Average Ranking Loss within and outside the subgroups, they mine the most distinctive ones making use of Cortana [28]. However, they mine subgroups composed of only one attribute at a time, in order to maintain interpretability. But this choice, while beneficial in the sense of computational power needed to search the subgroup space, significantly limits the complexity of the insights returned by the algorithm, since it becomes incapable of identifying interactions between the attributes and how they influence the model's performance.

In order to capture the effect of the interaction between features in the analysis of model performance, Smith-Miles and Muñoz [41] created a 2-dimensional Instance Space to identify regions of strong model performance. This Instance Space is created by linearly projecting all the features into a 2D plane, and building a surface based on the model performance at each point of the plane. This space can be used to guide the collection of additional samples located around areas where the performance of the model is weak, in order to improve it, and the total area in which the model has good performance can also be interpreted as a measure of model robustness. Known as the algorithm footprint, this Instance Space Analysis (ISA) can also aid in algorithm selection, by comparing the footprint left by various algorithms in a given dataset. The primary limitation of this procedure is the potential loss of information when linearly projecting features into a 2D space, especially for non-linear models. Moreover, the projected space provides little interpretability insight regarding the raw feature values and their relation to model performance. The pipeline proposed in this work provides a way to visualize both features

in a space built with its raw values, enhancing interpretability.

Another way to tackle the problem of finding the regions where an ML model makes the most errors is changing the focus from the model to the hardness of the instances themselves. This concept operates on the premise that each instance harbors an intrinsic difficulty level, which correlates with the likelihood of misclassification by a predictive model. Smith et al. [40] apply this idea by calculating a set of hardness measures (such as the number of k-disagreeing neighbors, tree-depth, and class-balance) that act as instrumental features in computing an instance's difficulty. By incorporating these measures into a classifier's error function, one can devise an Informative Error approach, enhancing the model's focus on simpler instances and potentially boosting its robustness. There is a strong relation between a model's errors in a given region in a dataset and the hardness of the instances in it, so calculating and aggregating the instance hardness could, in a way, aid in identifying the regions of most uncertainty in a model's predictions. In this sense, Valeriano et al. [43] combine the ideas described above of both ISA and instance hardness measures by building an instance space projecting meta features generated by calculating various hardness measures. The authors were able to visually identify samples that are representative of regions where the instances are especially hard, and these regions were also the ones where trained ML models had the highest errors. Furthermore, these were also the instances where humans also made the most mistakes in the medical datasets analyzed in their work. A primary limitation of this methodology is that the interpretability of the groups of instances identified as the hardest ones is not straightforward, requiring a post-processing step in order to explain the conditions in which hard instances appear, for instance, the range of their feature values, which is returned by the proposed model evaluation pipeline.

In an effort to generate more interpretable descriptions of these regions and contribute to the understanding of the reasons behind its difficulty, Pimentel et al. [35] proposed a way to search the n-dimensional space (for a dataset composed of n features) using subgroup mining in order to find regions where regression models' performance dips below average. This is assessed through the analysis of the prediction error across data regions, introducing Error Distribution Rules that link feature subgroups to error distributions. The Kolgomorov-Smirnov test is used for the comparison of error distributions, enabling the statistical identification of significant differences in model performance in critical parts of the dataset. Torgo et al. [42] presented a similar procedure, but for classification models, using a chi-squared test to compare the confusion matrices that result from comparing the predicted class and the ground truth for each subgroup. However, the main limitation in these approaches lies in the interpretability of the error measure: by looking at two subgroups that yielded significant values in the KS-test, or confusion matrices that significantly differ from the global model's performance, how can a domain expert understand how well the model is doing in each of the situations? Simplifying

the error measure to a mean error, as is done in the proposed pipeline, could yield more interpretable results.

In order to tackle this challenge, Prudêncio and Silva Filho [36] define the concept of Local Performance Regions. These are subspaces in the dataset where the model has exceptional performance, either especially good performance, the Local Easy Regions, or the exceptionally bad ones, the Local Hard Regions. In their work, a procedure is proposed in order to efficiently mine meta-rules that associate regions, defined by intervals of feature values, to unusual values of average model error. This is done by constructing a new dataset in the attribute space, with a measure of model error as the target, and inducing a decision tree to mine the most relevant rules that lead to higher average errors. The rules were filtered both by a minimum coverage threshold, and a minimum difference from the dataset average error. Since the meta-rules are mined using at most two different features at a time, they use a simple 2D scatter plot to show both the local performance regions and the dataset instances in a plane, making it possible to easily visualize the samples that fall inside a local performance region. This also enables a domain expert to understand the scenarios in which a given model is having trouble getting the predictions right, and exactly how bad the average error is in each case. The difficulty in this procedure lies in calibrating the threshold values to balance the classic trade-off between the size of the subgroup (support in the dataset samples) and how exceptional its errors are. By inducing the rules with a decision tree, this balance can be difficult to find, especially when dealing with different datasets, since each one can have different characteristics. This is another advantage of using subgroup discovery to find the groups, because by tweaking the quality function, we can better control the trade-off between size and exceptionality in the errors.

Similarly to the Local Hard Regions, Chung et al. [7] define problematic *slices* as subsets of the dataset where a model's error rate is significantly higher compared to the rest of the dataset in the Slice Finder framework. In the field of slice discovery, the goal is to identify these problematic slices, given a trained model and its predictions on a dataset. Unlike traditional methods that compare errors in a subset with the entire dataset, Slice Finder compares errors with the complement of the subset. The framework employs two main techniques: decision tree training, which uses the leaves of a decision tree trained to identify misclassified examples as slices, and lattice search, which performs a top-down search by adding new filters to the top-k best slices until a significantly different slice is found, in a breadth-first manner. To validate its effectiveness, Slice Finder uses both synthetic and real datasets, artificially generating errors to evaluate the accuracy of identifying problematic slices across varying numbers of top-k suggestions. However, the search is based on the assumption that when the slice has a statistically different error than its complement in the dataset, it is not worth drilling down, which may not always be the case, since one slice may be exceptionally problematic due to different reasons that could be decomposed, in case the search were to continue inside it. In our work,

we propose a novel solution that strives to find these regions using subgroup discovery, automatically addressing this balance between support and exceptionality, and enabling an interpretable analysis of how the samples interact with the mined subgroups.

Table 2.1 shows a comparison between the analyzed ways to solve the problem and how they compare to the pipeline proposed in this work. The dimensions chosen for comparison were the maximum number of features used to characterize each region/subgroup, whether it uses the raw features in the selectors (rather than processed combinations of them), whether the errors are averaged over the subgroups (or the full distribution is analyzed) and the rule mining procedure. These dimensions will also appear in the next section, where we will introduce some of the conceptual background needed to understand the proposed pipeline.

Table 2.1: Comparison of approaches for the detection of regions of exceptional performance

Approach	Max features	Raw features	Error is averaged	Rule Mining
[10]	1	Yes	Yes	Exceptional Model Mining
[41]	inf	No	No	Manual Inspection
[43]	inf	No	No	Manual Inspection
[35]	inf	Yes	No	Modified Apriori
[42]	inf	Yes	No	Modified Apriori
[36]	2	Yes	No	Decision Tree
[7]	inf	Yes	No	Decision Tree/Lattice Search
This work	2	Yes	Yes	Subgroup Discovery (Beam Search)

Chapter 3

Background

As a way to visualize the applications of the concepts described in this chapter, consider a toy example, chosen from the iris dataset, composed of four features and a binary target, shown in Table 3.1. Moreover, consider a given Machine Learning model that strives to classify between the classes in the target column. Its predicted probabilities for the positive class are also shown in Table 3.1.

3.1 Performance Metric

The first core concept is related to how to objectively quantify what can be considered good or bad in terms of predictive efficacy. The performance metric, defined as $E(M, d)$ is needed to quantify how well model M did in the prediction of each instance d in the dataset D. Depending on the Machine Learning task, different metrics can be used. For classification tasks, we can use both a sample prediction error, which would result in penalizing the situations where the model has a lot of confidence in the wrong answer, or simply a binary variable identifying if the classification was correct or not, treating all misclassifications as equally bad. For regression tasks, we could use the absolute error, which would penalize more the instances where the absolute value of the predictions most differs from the real target variable, independently from the scale of the target, or by choosing the percentage error, the errors would be always scaled by the value of the target for each instance in the dataset.

For the pipeline described in this work, it is important for the performance metric to be calculated at an instance level, so that it can be sufficiently robust when being aggregated even for small subgroups. In this sense, metrics such as the area under the ROC Curve and the F-Score, although very useful in many model evaluation situations, are not adequate for this pipeline. This is mainly because they make much more sense when calculated over a bigger number of instances, and suffer to produce a significant and interpretable result when dealing with small numbers. Since the goal of this work is

to be able to determine the local areas where the ML model has poor performance, it is necessary for the performance metric to be able to adequately quantify it even for less densely populated regions in the dataset.

For the toy example, since it is a classification task, we can use a measure of prediction error, calculated as the absolute difference between the predicted value and the actual target. The calculation of the prediction error for the sample dataset is shown in Table 3.1.

3.2 Subgroup Discovery

Subgroup discovery is a data mining technique that consists of searching a dataset for subsets that are exceptional regarding a given property of interest [2]. It is a task within the broader field of descriptive learning, which aims to extract useful knowledge out of a dataset. In contrast to predictive learning, which is focused on learning rules in order to be able to predict a certain variable adequately, the field of descriptive learning is engaged in inducing rules that are useful for understanding the phenomenon. It has been successfully applied in different areas, such as clinical trials [11] and agriculture [30]. It is also related to other rule-inducing techniques, widely used in the field of data mining, such as association rules mining, since the subgroup descriptions are often very similar to the antecedents of association rules, and many of the subgroup discovery algorithms were adapted from association rules mining. The key aspect that sets subgroup discovery apart, in this context, lies in the fact that maximizing the exceptionality in the distribution of a given target variable inside the subgroup, when compared to the rest of the dataset, is the main focus of the subgroup discovery task.

One of the advantages of using subgroup discovery techniques is that its goal is to generate interpretable descriptions of the subgroup, in the form of rules that relate conditions against which the samples inside the subgroup are successfully evaluated, and an exceptional behavior of the target variable. The conditions, also called *selectors*, define slices of the dataset based on an interval of values that each feature can assume. For instance, given a dataset D composed of a set of m numeric attributes $\mathcal{A} = a_1, a_2, \ldots, a_m$, a selector s would be a condition such as

$$s : a_i \in [\alpha, \beta[, \alpha, \beta \in dom(a_i), \tag{3.1}$$

where α and β are the limits of the interval of values the instances selected by s can assume for the attribute a_i. In the case of categorical attributes, instead of having an interval of values for the attribute, the selector would include a specific category in the

Table 3.1: Example dataset sampled from the iris dataset

id	sepal length	sepal width	petal length	petal width	target	prediction	error
1	7.00	3.20	4.70	1.40	0	0.73	0.73
2	6.40	2.80	5.60	2.10	1	0.82	0.18
3	5.90	3.20	4.80	1.80	0	0.52	0.52
4	7.10	3.00	5.90	2.10	1	0.64	0.36
5	6.50	2.80	4.60	1.50	0	0.46	0.46
6	6.10	2.90	4.70	1.40	0	0.02	0.02
7	6.00	2.90	4.50	1.50	0	0.20	0.20
8	6.30	2.90	5.60	1.80	1	0.81	0.19
9	5.60	3.00	4.10	1.30	0	0.44	0.44
10	6.80	3.00	5.50	2.10	1	0.89	0.11

domain of the attribute. The rules are composed of a conjunction of selectors that lead to a certain behavior of the target variable, quantified by the average of the target variable, in the form

$$r : s_1 \ \wedge \ s_2 \wedge \cdots \wedge \ s_k \rightarrow \bar{E}(M, D_r) \tag{3.2}$$

with each s_* being a selector built using one of the attributes in $\mathcal{A}$. Together, they define the set of samples D_r, against which the performance metric $E(M, d)$ is calculated, sample by sample, and averaged. The result of the procedure is usually a list of subsets mined over the dataset, made of the subgroups with the most quality that were found.

Returning to the example dataset shown in Table 3.1, one subgroup that could be built using the given samples is s : sepal length $\geq 6.40 \ \wedge$ petal width $\in [1.40, 2.10)$. The instances that meet the requirements of both the selectors are those with ID 1 and 5, so these instances are said to be *covered* by the subgroup shown above. The coverage set of a subgroup is the set containing the indices of the samples it covers and is another way to characterize it, since there may be multiple subgroup descriptions that effectively cover the same set of samples. For example, the subgroup s_2 : petal width $\geq 6.50 \ \wedge$ petal length $\in [4.60, 4.80)$ is such that

$$coverset(s) = coverset(s_2) = \{1, 5\}.$$

Beyond the predicate, the subgroup is also associated to a certain behavior shown in the target variable, called the *target concept*. In the case of the example, we could define it as the average value in the target column: the average prediction error. Then, the complete description of the subgroup would be

$$\text{sepal length} \geq 6.40 \wedge \text{petal width} \in [1.40, 2.10) \rightarrow \bar{E}(M, D_r) = 0.595.$$

Furthermore, it is noteworthy that, when compared to the average value of the target variable in the whole dataset (0.321), this subgroup has an especially high value, deeming

it interesting in the context of subgroup discovery, since it selects a set of instances with an exceptionally different average target value.

The rule mining is done basically by searching between some or all of the possible combinations of intervals in the values of the features in the dataset, and calculating a quality measure that is used to quantify how special the subgroup is. This quality measure is a function of the target variable and, typically, of the size of the subgroup. The result of the subgroup discovery procedure is a set of the most interesting subgroups found during the mining phase, according to the pre-defined quality measure.

There are three main choices one has to make when applying the subgroup discovery procedure over a dataset: defining the search space, the quality measure, and the search algorithm. These concepts are explained in the following subsections.

3.2.1 Search Space definition

Looking at the task of subgroup discovery as a search over the possible subgroups in order to find the ones with the best quality, the first step is then defining what is this space over which the search happens. This can be seen from two perspectives: one would be based on the samples themselves, which could form a big but limited number of subsets (if the dataset has n instances, the number of possible subsets would be 2^n), and the second, based on the features that describe the samples. While the first perspective could be easier to enumerate, the subset resulting from this approach would provide little information regarding what the characteristics of the samples in it are. On the other hand, by basing the search space on top of the feature values, we can obtain much more interpretable descriptions of the subgroups mined.

In this sense, while the interpretability of the results is much greater, the number of possible subgroups skyrockets, since datasets are usually comprised of multiple features, and each one has a range of values of its own. So the left-hand side of the subgroup description defines a predicate, usually built by conjunctions of values some of the features in the dataset can assume. These values can be both categorical, in which case the conjunction would be if the feature value is equal to a specific categorical value, or numeric, in the form of an interval. Then, if an instance's feature values are equal to the categorical value or inside this interval, it is said to be *covered* by the subgroup.

When dealing with numeric features, there is virtually an infinite number of possible subgroup descriptions, since there is an infinite number of ways to define intervals over continuous values. So most subgroup discovery algorithms involve a step of discretization of the continuous features. The discretization can be done in multiple ways, like choosing

a set of bins that intuitively partition the domain of the feature. For example, we could choose to split an attribute of age of the person applying for a loan into bins such as 18 up to 25, grouping the young adults, 25 until 35, maybe an age where most start thinking about having children, and so on. Another, more straightforward way to discretize a numeric feature is by breaking it into quantiles, and redefining it as a new, categorical feature, based on the selected quantiles. As an example, we could discretize the feature *sepal length* by using its quintiles: $(-\infty, 6.00), [6.00, 6.30), [6.30, 6.50), [6.50, 7.00)$ and $[7.0, \infty)$.

This discretization step can occur a priori or during the subgroup discovery, deciding the best splits according to estimations of the target metric calculated during the execution of the search algorithm. Once the features are discretized, the intervals behave just like any other categorical attribute, and the search space can then be defined as all the possible combinations of features, and for each combination of features, all possible combinations of their intervals (or categorical values).

3.2.2 Quality Measure

The quality measure, also referred to as quality function, is the main reference used to quantify how interesting a given subgroup is, during the subgroup discovery procedure. It is typically a trade-off between how special the subgroup is, measured by an aggregation over the target variable, and how relevant the subgroup is in the population. When analyzing the subgroups mined over a dataset, one would question both these aspects: is this subgroup representative enough to be worth diving deeper into its understanding? And how different are its values of the target variable when compared to the rest of the dataset? Both these questions are to be balanced and objectively addressed by the value of the quality measure calculated over the subgroup.

Depending on the type of the target variable, the quality function will have a different form. For example, when the target variable is discrete, the target function is usually built upon the share of times each possible value for the target variable occurs inside the subgroup, compared to how it behaves on the dataset as a whole. In this sense, the quality function for binary and nominal target variables can be generalized and written as in Equation 3.3, discussed in Atzmueller [2], which defines the quality of subgroup P of size n. There, a is a factor that controls how much the size of the subgroup influences its quality, $t_P^{v_i}$ is the share of times that the target assumes the value v_i inside subgroup P, and $t_0^{v_i}$ is its equivalent, but on the whole dataset.

$$q^a(P) = n^a \cdot \sum_{v_i} (t_P^{v_i} - t_0^{v_i}), a \in [0, 1] \tag{3.3}$$

Alternatively, if the target variable is numeric, then a more classic type of aggregation can be used, such as taking the mean, median, variance, or even statistics calculated over the whole distribution of the target variable inside the subgroup. Equation 3.3 could be rewritten by changing the shares of positive values ($t_P^{v_i}$ and $t_0^{v_i}$) by the mean of the target variables (m_P and m_0), arriving at Equation 3.4. Although there are other types of quality functions for numeric concepts that closely relate to the use case in this work, such as the AUC measure, proposed in Pieters et al. [34], they lack interpretability for a final user to understand, especially compared to the simpler and considerably useful mean-based quality measures. Furthermore, they also impose the limitation of a minimum support of the subgroup in terms of the number of samples covered, discussed in Section 3.1, because the calculation of the ROC AUC itself is not robust to be applied to a small number of observations. In order for the pipeline proposed in this work to be generalizable for multiple datasets of different characteristics (including sizes), it is preferable to choose a quality measure that is not as strongly affected by small subsets of the data.

$$q^a(P) = n^a \cdot (m_P - m_0), a \in [0, 1] \tag{3.4}$$

There is a plethora of quality measures, thoroughly described in Lemmerich [26]. A simple way to answer both these questions is by multiplying the average of the target variable in the subgroup by the number of observations in it, as is done in the Piatetsky-Shapiro quality measure (substituting $a = 1$ in Equation 3.4. If we use this quality measure in the example subgroup sepal length $\geq 6.40 \wedge$ petal width $\in [1.40, 2.10)$, we would reach a quality of

$$q_{ps} = n(m - m_0) = 2(0.595 - 0.321) = 0.548,$$

which would be used by the subgroup discovery algorithm as the measure of how interesting this subgroup is. When compared to the quality of another possible subgroup, $sepalwidth = 2.90$, which has three samples (6, 7 and 8) and a quality of 0.410, the previous example subgroup would be preferred, since it has a higher quality, even though it covers fewer examples. Both subgroups are highlighted in Table 3.2 for a clearer visualization of the referred instances, in light gray for the two samples covered by the subgroup of quality 0.548, and in darker gray for the three samples that reach a quality of 0.410.

Table 3.2: Example dataset highlighted according to the subgroups that cover some of the samples.

id	sepal length	sepal width	petal length	petal width	target	prediction	error
1	7.00	3.20	4.70	1.40	0	0.73	0.73
2	6.40	2.80	5.60	2.10	1	0.82	0.18
3	5.90	3.20	4.80	1.80	0	0.52	0.52
4	7.10	3.00	5.90	2.10	1	0.64	0.36
5	6.50	2.80	4.60	1.50	0	0.46	0.46
6	6.10	2.90	4.70	1.40	0	0.02	0.02
7	6.00	2.90	4.50	1.50	0	0.20	0.20
8	6.30	2.90	5.60	1.80	1	0.81	0.19
9	5.60	3.00	4.10	1.30	0	0.44	0.44
10	6.80	3.00	5.50	2.10	1	0.89	0.11

3.2.3 Search Algorithm

The search algorithm defines how the navigation through the search space is performed. Its goal is to find, in the smallest number of iterations possible, how to create subgroup descriptions that lead to the highest values for the quality measure. This search can be done exhaustively, with the guarantee that the final solution is the best possible solution for the dataset, or heuristically, aiming to find a good solution that may not necessarily be the best of all. The most famous exhaustive methods include the Depth-First Search (DFS) and the Breadth-First Search (BFS) as alternatives for the enumeration strategy [26], varying only in the order that the subgroups are evaluated. In the EXPLORA algorithm [22], for example, both enumeration strategies are supported. These enumeration strategies are normally used along with intelligent pruning strategies that identify subgroups that are not worth evaluating, since their quality can be shown to be necessarily equal to or lower than other subgroups that were already visited, as is done in DDPMine [6]. Among the most used heuristic methods is the Beam Search, where a set of candidate hypotheses is iteratively refined until the end of the execution.

In the context of subgroup discovery applied to the performance analysis of ML models, the datasets used are composed of the original model's features. Thus, the possible number of subgroups, built by filtering distinct values (or intervals as discussed in 3.2.1) can become significantly large, depending on the number of features and the diversity of their values. In this sense, for larger datasets, exhaustive search algorithms may be less adequate due to the computational complexity of their execution.

Throughout time, several search algorithms were designed, mostly by adapting existing algorithms from other related disciplines to the context of subgroup discovery. From the field of classification, the algorithms EXPLORA and MIDOS [45] were the first

ones to be proposed, and they work by using decision trees to extract useful splits, and considering them part of the conjunctions for a subgroup. Then, these algorithms were modified and extended, creating a new generation of subgroup discovery algorithms based on classification rules, such as SubgroupMiner [23] and CN2-SD [24], with the former being adapted from the CN2 classification rule learning approach. Another category of subgroup discovery algorithms derives from the field of association rules mining, a problem that is closely related to subgroup discovery, in the sense that its goal is to find relations between the variables in the dataset, but regardless of the order of this relation (each feature can appear in the antecedent or the consequent of the association rule). From this field of study, the algorithms APRIORI-SD [20], derived from the APRIORI algorithm, SD-Map [3] and Merge-SD [13], derived from the FP-growth method were proposed. Finally, another field that inspired some subgroup discovery algorithms is the area of genetic algorithms, from which derived the algorithms SDIGA [9] and MESDIF [4], that focus on the extraction of fuzzy rules that describe the subgroups, with an evolutionary strategy.

These concepts will be important when discussing the design choices made in the construction of the pipeline presented in the next chapter. There are many possibilities for the performance metric, the definition of the search space (usually its discretization), the quality measure and for the search algorithm, but the decisions made in the proposed pipeline strive to achieve a balance between generality and effectiveness, electing options that work well in most cases.

Chapter 4

Proposed Pipeline Description

The main components of the proposed pipeline are shown in Figure 4.1. Two instances of the pipeline are displayed in the Figure, since it can also be used to compare multiple ML models trained on the same dataset, and it is composed of four numbered steps. The pipeline starts with the calculation of the model's error (1). Then, based on the errors and the model's features, a subgroup discovery procedure is run (2), mining the most relevant subgroups, according to a given criterion. Since the subgroup discovery algorithm used in this paper does not account for the potential redundancy between the rules mined, the next step of this pipeline is to summarize the redundant subgroups by grouping them into clusters that will be represented by the most relevant of the subgroups (3). By doing that, we obtain the summarized subgroups, which can then be analyzed in two different ways: the first is by visualizing the subgroups in a 2D scatter plot, which shows both the subgroups' boundaries and the samples, as well as the metrics calculated for the subgroups (4); the second way is used when comparing multiple models, over which a set of subgroups is mined, and some of the subgroups coincide. Then, by identifying how many subgroups are equal and different between the models, a Venn Diagram is built (5), and some conclusions can be drawn from the results.

4.1 Error Calculation

The first step to understand the strengths and weaknesses of a given trained model is to determine how to measure its errors, as discussed in Section 3.1 and it is defined by the function $E(M, d)$. This step takes the model's predictions and the ground truth labels as inputs, and generates the errors as the output. The simplest error measure in the context of binary classification is the sample prediction error, given by Equation (4.1), where y is the correct label for sample d and $\hat{p_M}(1|d)|$ is the probability score for class 1 returned by the model given the features of sample d. This choice of error function is due to a special interest in how confident the model is when predicting the class of each

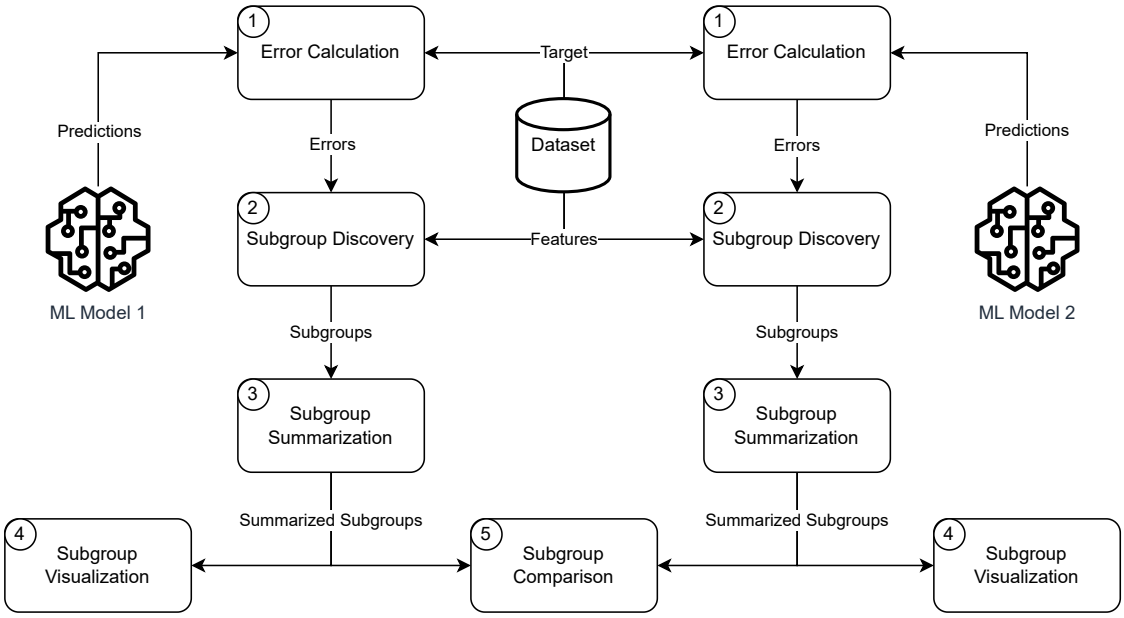

Figure 4.1: The components of the model evaluation pipeline proposed.

sample. In terms of understanding the behavior of a model's errors, the situations in which the model generates a very high probability score for one class in the prediction of an instance, and it is not the correct class, are certainly of high interest.

$$E(M, d) = |y - p_{\hat{M}}(1|d)|, y \in 0, 1 \tag{4.1}$$

Although this paper focuses on applications for classification problems, it could naturally be extended to regression problems, as discussed in [35], just by changing the error measure to one specific to this type of problem. Moreover, it can also be applicable to multi-class classification problems, by fitting a one-vs-all binary classifier for each class, as will be done in Chapter 6. This way, each binary classifier will have its own error measure, given by Equation 4.1, with y indicating whether each sample is of the class the classifier is trying to predict. In this case, the performance metric becomes

$$E(M_k, d) = |y_k - p_{\hat{M}_k}(1|d)|, y_k \in 0, 1 \tag{4.2}$$

where M_k is the binary classifier for the prediction of class k, and y_k is an indicator variable that assumes the value 1 when the instance d is of class k.

After choosing an error measure, the following step is to calculate its value for every sample in the (test) dataset. This way, an auxiliary dataset is created, composed of the original model features and the prediction error $E(M, d)$ as the target. An example of the format that this dataset assumes can be seen in Table 3.1.

4.2 Subgroup Discovery

The dataset built in the previous step is then given as an input to the subgroup discovery procedure, represented by step 2 in Figure 4.1. In this work, numeric features are discretized into quintiles, for an easier application of the subgroup discovery algorithms, but one could naturally skip this step if using a different algorithm that supports numeric features, or choose another discretization method if the dataset used has features that make more sense to be discretized in another way, such as the ones described in Subsection 3.2.1. Another specificity of this pipeline is the choice of the maximum number of selectors in the subgroup descriptions at two. This is extremely useful to be able to visualize, in a two-dimensional plane, the subgroup boundaries and how they relate to the placements of the samples, shown in the same plane, as will be discussed in Section 4.4.

The target of the subgroup discovery procedure is the error calculated in the previous step, so in order to find interesting subgroups regarding this variable, we must choose a quality measure designed for a numeric target. One simple way to proceed is to use the *mean test* as the quality function, displayed in Equation (4.3):

$$q = \left(n_{SG}\right)^{0.5}(\mu_{SG} - \mu) \tag{4.3}$$

where n_{SG} is the number of samples in the subgroup, while μ_{SG} and μ are the average values of the target in the subgroup and in the whole dataset, respectively. This quality function attempts to balance between subgroups which are not too small, representing relevant parts of the whole dataset, while preserving an unusual average value regarding the target variable. It is important to note that, while this function will generate subgroups that lead to a higher prediction error, by changing the order of the subtraction $(\mu - \mu_{SG})$, it can separate subgroups which lead to a lower prediction error. These regions can also be useful for assessing a model's performance over a dataset. The choice of 0.5 as the exponent also has a theoretical basis, since [34] showed that it is order equivalent to the standardized z-score of a subgroup. Beyond that, [29] also discussed the different options for the value of this exponent, and how they impact the result of the subgroup discovery procedure, and their findings showed that there is no clear correct answer. So, we choose the value of 0.5, but this value can be adjusted in case the subgroups being pointed out as hard are too small or too generic in a given application of this pipeline.

Another noteworthy aspect of the choice of this quality measure, and also a consequence of using Subgroup Discovery to mine the exceptional subgroups, is to aggregate the errors by averaging them inside the subgroup. This is useful because it provides an out-of-the-box explanation, to a domain expert, as to why a given subgroup is exceptional. It is easier for a non-technical person to understand a statement such as "the model has

a 30% higher average error in this subgroup" than something like "the model presents an error distribution that is statistically different from the global error distribution" as is done in previous works [7, 36, 35], even though it is a more naive way to define quality in a subgroup.

Having defined the quality measure, the last step that defines how the subgroup discovery is going to take place is the choice of the search algorithm. For this work, the chosen algorithm is the Beam Search, one of the most efficient heuristic strategies, and given its anytime characteristic, it is flexible enough to generate a good result for any existing time constraints. Considering the application of this pipeline over huge datasets, which are fairly common in the data science field, it is advisable to prefer using a heuristic approach rather than a complete or exhaustive one to search for the optimal subgroups, since as the number of features and the number of samples increase, the computational complexity of sifting through all possible subgroups becomes rapidly overwhelming. The output of this step is the complete list of the best subgroups found by the subgroup discovery procedure.

4.3 Subgroup Summarization

The subgroups returned by the subgroup discovery algorithms typically include a combination of the model features, bounded by numeric or categorical constraints. But in real world data, it is reasonably common to observe some correlation between features, and in this case, it may result in the same group of samples being described by more than one rule found in the subgroup discovery. Even though many subgroup discovery algorithms try to prevent this, such as the relevance mechanism in SD [12], or the dominance pruning in DS-SD [44], some redundancy usually remains. So, in order to tackle this redundancy, the next step, number 3 in Figure 4.1, is to identify which rules cover the same examples, pruning these rules from the final set.

By comparing the coverage sets, different descriptions of basically the same subgroup can be pruned from the final result. One way to measure the similarity between these sets is to calculate one minus the Jaccard Index [17] (also known as Jaccard Distance), and join the ones with a value below a certain threshold. Moreover, the Jaccard Distance can also be calculated over the subgroup descriptions themselves, instead of over the coverage sets, by treating each selector in the rule's predicate as an element of a selector set. In this way, subgroups that share a selector in their predicates would automatically have a Jaccard Distance lower than one. This intuition can be useful when the Jaccard Indexes calculated only on the coverage sets are too low, for example in situations

where the dataset is unbalanced, and there are only a few samples of one class, shared between subgroups populated with different portions of the majority class. Combining both these ideas, we reach the following distance measure:

$$D(s_1, s_2) = \min(1 - Jaccard(Cov_{s_1}, Cov_{s_2}), 1 - Jaccard(Sel_{s_1}, Sel_{s_2})), \tag{4.4}$$

where Cov_{s_1} and Cov_{s_2} are the coverage sets of subgroups s_1 and s_2, respectively, and Sel_{s_1} and Sel_{s_2} are the selector sets of s_1 and s_2, respectively. This idea is adapted from [44], where the authors discuss how to define the concept of redundancy in the results of subgroup discovery. The authors argue that there are three main forms of redundancy: in the subgroups' descriptions, in the subgroups' coverage, and in the models (applicable only for Exceptional Model Mining, a correlated field). They also affirm that all three kinds of redundancy should ideally be reduced, and if possible, simultaneously. This is what the proposed distance measure does, since by taking the minimum between the two Jaccard distances, if any of the two types of redundancy is high, the subgroups will have a low distance.

Having calculated the dissimilarities between all of the top subgroups previously mined, the pairs of subgroups that are too similar can be pruned from the final list. Starting with the lowest distance pair overall, since the two subgroups are very similar, we choose the one with the highest quality as the representative of the pair, and then follow on to the next lowest distance. This is done iteratively until all of the remaining pairs of subgroups have a distance of at least a given predefined threshold. But in order to choose this threshold effectively, it helps to have a notion of what is the order of the distances between the pairs of subgroups, and which subgroups would be merged once that threshold is chosen. So, a way to execute this merging procedure and also assist in the choice of an adequate distance threshold is to use a Hierarchical Clustering Analysis, with an Agglomerative Clustering Algorithm [31], executed with the UPGMA linkage method. The output of this kind of algorithm is typically shown in the form of a dendrogram, which enable the user to see at which level (distance) each pair (or set of pairs) of subgroups are merged, an thus choose an appropriate distance threshold. Using the dendrogram can also be useful to understand, from the problem's domain point of view, what are the main types of subgroup that the model makes the most errors in, since the subgroups that are merged later in the procedure tend to be significantly different from each other.

4.4 Subgroup Visualization

Having extracted a set of non-redundant subgroups that describe the samples on which a ML model makes the most errors, it is finally ready to be analyzed. Since the maximum depth of the subgroups is set to two, the subgroup descriptions will have at most two features, which can be easily visualized in a plane. Although up to three features could be plotted in a 3D graph, in practice this type of visualization is rarely used, mainly due to the high level of interactivity needed for a given user to understand a 3D figure, and the boundaries of a subgroup would become a parallelepiped. For this reason, we keep the maximum number of selectors at two, for a more straightforward visualization.

While for numerical features the visualization of the samples is straightforward by using a scatter plot, an adaptation can also be made of it to support categorical features, by arbitrarily setting integer values for each category, and only slightly jittering the integer values with random noise, for better visualization. In this way, each subgroup description, which can also be seen as a region in the dataset, can be visualized along with the samples that fall inside (are covered by) or next to it, colored by class for easier identification. Beyond being able to visually identify which samples fall into which subgroups, this plot can also inform the user about how hard each subgroup was for the model. This is done by annotating, next to the boundaries of each subgroup, the average error of the model inside the subgroup, and over all the dataset.

If more than one subgroup shares features in their selectors, it is also possible to visualize more than one subgroup at the same time, rendering the user an understanding of the regions in the dataset, for each pair of features, that the model is having the most trouble predicting. This can, in turn, be extremely useful in devising new ways to improve the model, since the hardest regions will be clear and could indicate the reasons behind the model's poor performance, such as lack of data, data quality issues, outliers, and more.

4.5 Subgroup Comparison

When creating an ML model for a given problem, it is common to experiment with different types of models, in order to understand which one performs best for the task at hand. With an increasingly high availability of software tools to easily execute the most diverse types of models, it becomes even more straightforward to test different

alternatives, which will result in different outcomes. The last step of the proposed model evaluation pipeline aims to help the user understand how the trained models are related, based on the regions where they make the most errors.

By running the pipeline for each model, a set of subgroups will be mined for each of them, and we can compare which subgroups were identified as hard (or easy) for each model by plotting a Venn Diagram. It will show the number of subgroups that fall under each zone of possible intersection between the different models. Subgroups that fall under the intersection of multiple models probably indicate that these regions are hard by themselves, probably being a difficulty inherent to the dataset. On the other hand, when a certain subgroup is identified as hard for only one model (or few models), it could mean that that specific model (or set of models) is having trouble predicting the instance's targets, be it due to the model's structure, its assumptions, or even its training hyperparameters.

Chapter 5

Experiments

In order to evaluate the pipeline's ability to precisely identify the regions in a dataset where the error is high, a synthetic dataset was used. The **moons** dataset is available in the scikit-learn Python package [33], and is shown in Figure 5.1. A Logistic Regression model was trained on this dataset in order to distinguish between the two classes. However, since the dataset is not linearly separable, there were naturally some errors, as can be seen in Figure 5.1.

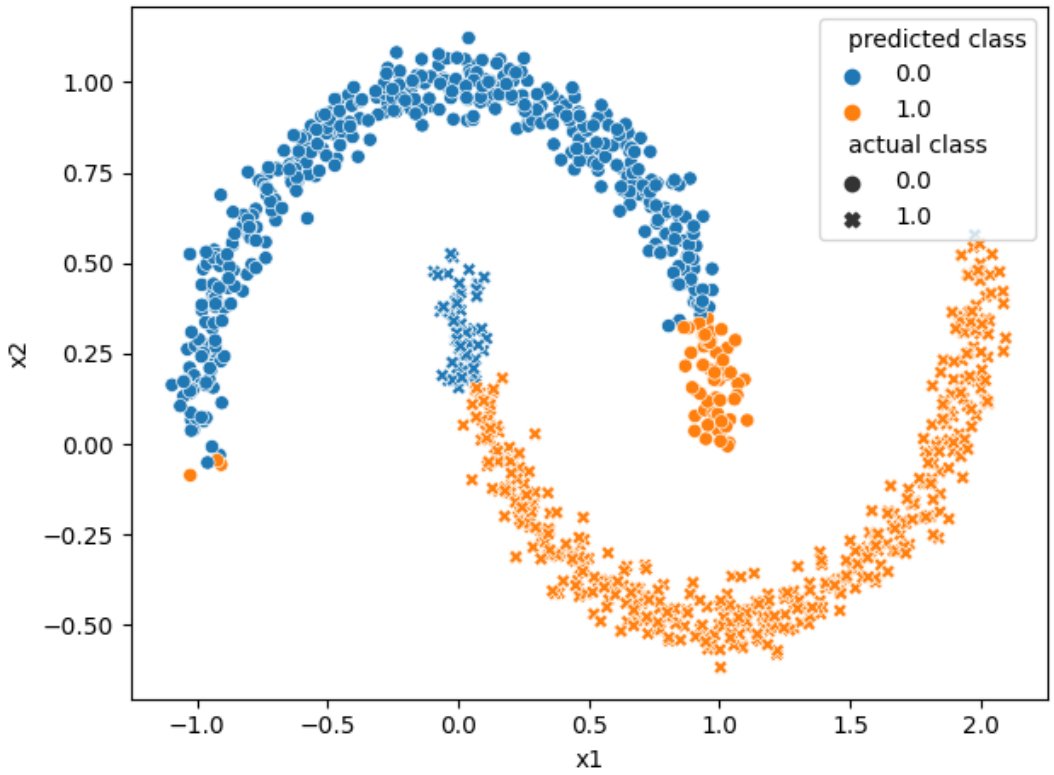

Figure 5.1: Moons dataset along with the predictions of a Logistic Regression classifier trained on it, using 0.5 as the threshold for classification.

For this experiment, we focus on identifying the samples that are incorrectly classified when using the classification threshold of 0.5. As baselines for comparison with the proposed pipeline, the Local Performance Regions (LPR) [36] and the Slice Finder framework (SF) [7] were reproduced and used to identify the regions where the model

exhibited the highest errors. It is important to emphasize that, as can be seen in Table 2.1, while the LPR approach makes use of Decision Trees to find the interesting regions in the dataset, the SF approach was run with the Lattice Search algorithm, providing two different search strategies to be compared to the proposed one that uses Subgroup Discovery.

After running the proposed pipeline with the default parameters over this dataset, 7 subgroups were identified as hard regions, with an error above the mean of the dataset. These subgroups can be seen in Figure 5.2. As expected, the incorrectly classified samples are detected by the proposed pipeline, and only one of the subgroups contains no misclassified samples. Since the proposed pipeline selected 7 subgroups, the top 7 regions/slices were selected from the output of each of the baselines. The resulting regions found for the LPR and the SF framework are shown in Figure 5.3 and Figure 5.4 respectively. When comparing the results qualitatively, we see that while the LPR baseline found larger subgroups, mainly by selecting an interval of values on only one feature, the SF baseline focused on smaller subgroups, trying to select only high error samples, but failing to cover some of them. This corroborates with the argument made in Chapter 2, that one of the main advantages of using Subgroup Discovery to mine the regions of high error is its innate ability to balance between the size of the subgroups and how special they are.

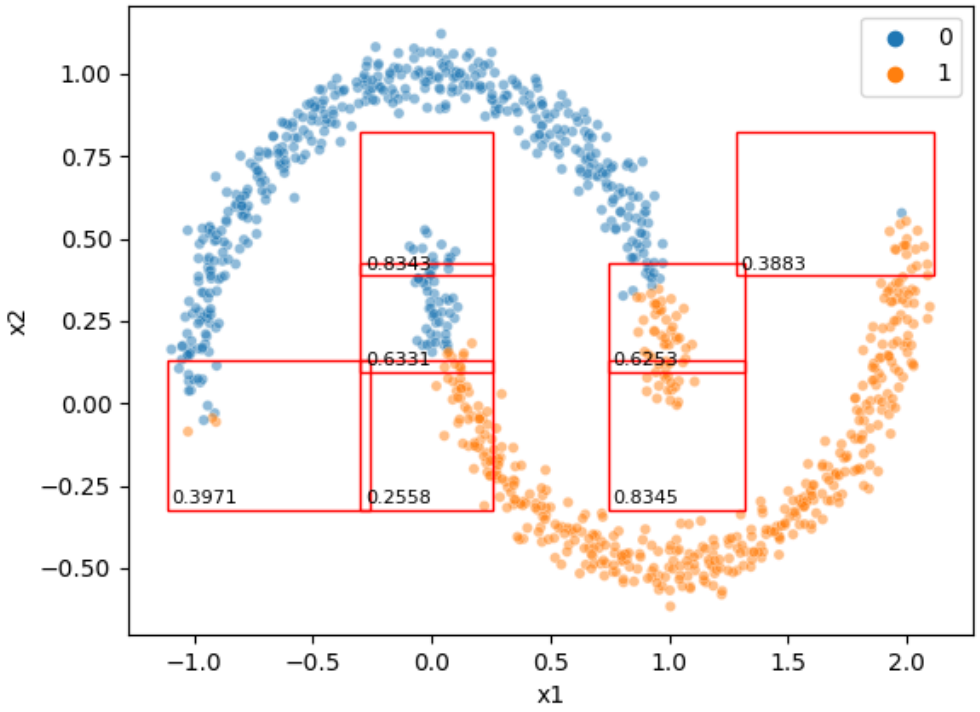

Figure 5.2: Subgroups found when running the proposed pipeline for the Moons Dataset.

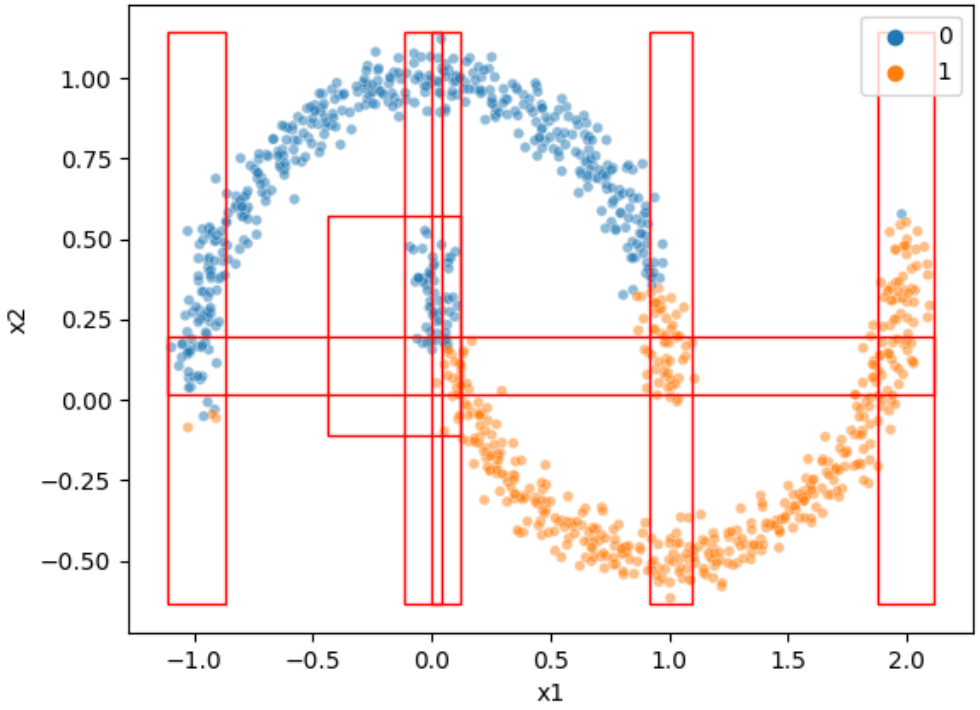

Figure 5.3: Subgroups found when running the Local Performance Regions baseline for the Moons Dataset.

For a more quantitative comparison, we analyze the cumulative precision and recall in the identification of misclassified samples between the proposed pipeline and the baseline methods. This precision is calculated as the percentage of misclassified samples in the identified groups, and the recall is the coverage of misclassified samples in the proposed subgroups. They are calculated by the number of subgroups, so we can see which method more precisely detects the highest portion of the high error samples, in the smallest number of groups. In Figure 5.5, we notice that in terms of precision, the proposed method outperforms almost all baselines, only displaying a lower value than the SF baseline (the most selective one) when considering all 7 subgroups. In terms of recall, the LPR baseline found the best first subgroup but, for all others, the proposed pipeline did a better job in covering the high error samples with its subgroups. In Figure 5.6, we plot the cumulative coverage of the high error samples by the percentage of all samples selected, for each number of groups, and we can see that the proposed pipeline is able to cover a higher percentage of the high error samples, while selecting a lower percentage of samples. So, in terms of accurately identifying the samples with high error, with the goal of understanding the model's weaknesses and devising ways to improve it, the proposed pipeline has shown to be significantly more effective than the baseline methods in this dataset.

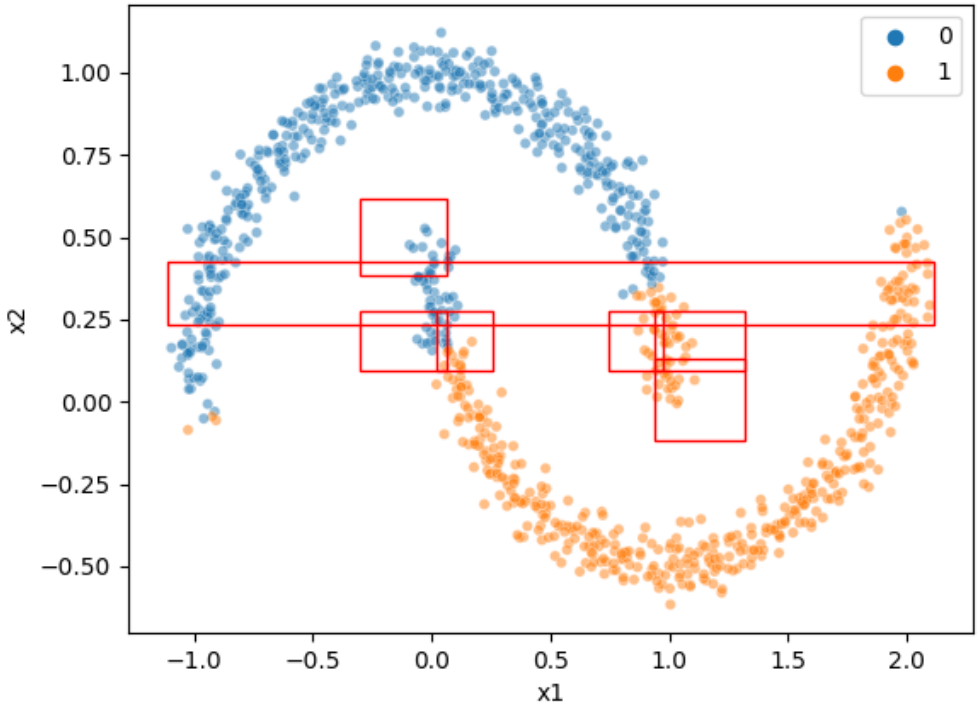

Figure 5.4: Subgroups found when running the Slice Finder baseline for the Moons Dataset.

Finally, to evaluate the pipeline's effectiveness in identifying the high error samples in various scenarios, four other datasets of various sizes and domains were selected from the UCI repository [21]. For each dataset, a Logistic Regression model was fitted and the proposed pipeline (with default parameters) and the two baselines were executed. For a more concise and objective comparison of the results, after calculating the cumulative precision and recall as described in this chapter, we also combined these metrics into one indicator of pipeline performance by calculating a cumulative F1-Score. Then, the approaches were ranked for each number of subgroups analyzed according to the highest F1-Score, and the average rank was calculated for each dataset. The results are shown in Table 5.1, with the lowest ranks highlighted in bold, and the execution times of each pipeline are shown in Table 5.2.

Table 5.1: Average rank of the baselines and the proposed pipeline over different datasets

Dataset	Size	Average Rank		
		LPR Baseline	SF Baseline	Proposed Pipeline
Diabetes	253680	2.00	3.00	**1.00**
Iris	150	2.29	**1.43**	2.29
Moons	1000	2.71	2.00	**1.29**
Student Performance	649	3.00	2.00	**1.00**
Wine	6497	2.00	3.00	**1.00**

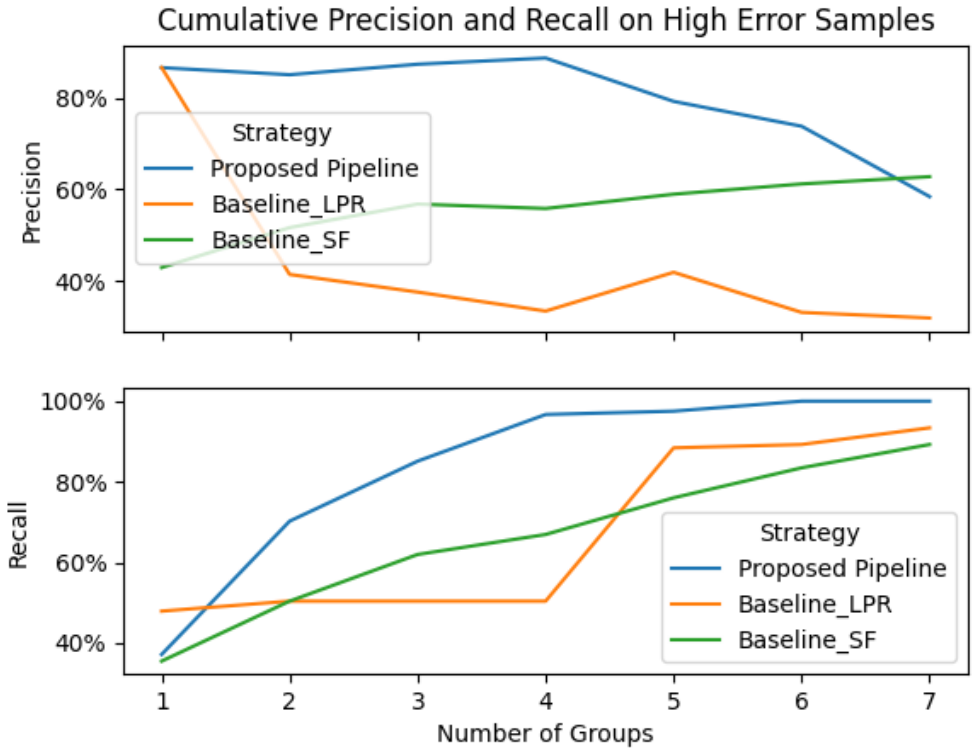

Figure 5.5: Cumulative precision and recall over high error samples for the baselines and the proposed approach, by number of subgroups.

Table 5.2: Execution times of the proposed pipeline and the baselines over different datasets

Dataset	Size	Run Time (min)		
		LPR Baseline	SF Baseline	Proposed Pipeline
Diabetes	253680	49.22	21.98	**0.11**
Iris	150	0.18	0.34	**0.00**
Moons	1000	0.07	0.31	**0.01**
Student Performance	649	6.43	0.21	**0.01**
Wine	6497	1.99	0.58	**0.01**

It is clear to notice that, for most datasets, the proposed pipeline achieves better ranks in the identification of high error samples than the baselines. Furthermore, the execution times are also significantly lower, showing that it can find subgroups with higher precision and recall in the identification of regions of high error, in less time than the baselines. In the next chapter, an application of this pipeline will be shown from the perspective of a user of the pipeline, trying to identify what regions a given model is struggling to get the predictions right, with the goal of understanding its performance and devising ways to enhance it.

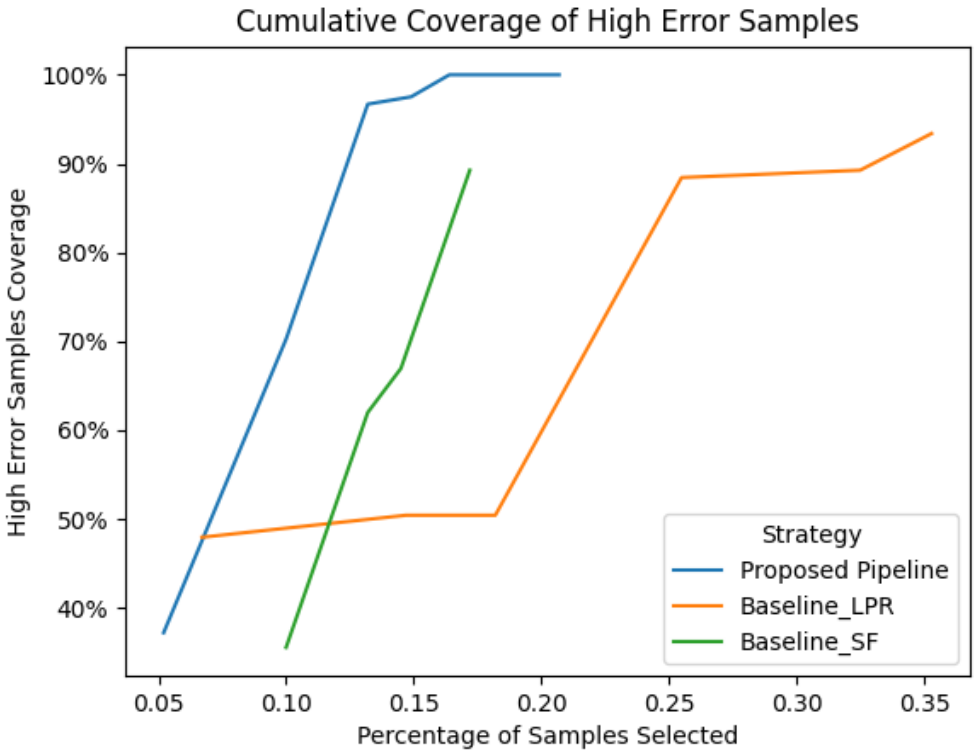

Figure 5.6: Cumulative coverage of high error samples for the baseline and the proposed approach.

Chapter 6

Application

To demonstrate the application of the pipeline described in Chapter 4, we will make use of the Bank Account Fraud (BAF) dataset [18]. This dataset was built using real-world bank account opening fraud data, and by using state-of-the-art data generation techniques, the authors created a realistic benchmark for the application of Machine Learning methods on tabular data. The goal of this dataset is to be able to predict whether each bank account opening request would be a fraud or not, based on some information about the applicant such as their income, age and personal information, and information about the request itself, for example, the initial amount transferred to the account, the credit payment plan, and the proposed credit limit. For that, first we choose three model classes to understand how well they can fit and predict the target in this dataset: Logistic Regression, Random Forest and Gradient Boosting. The idea is to find out what are the regions where each of these models most struggles to get their predictions right and also assess whether the regions found for each model have a significant intersection. After training the three models, we evaluate each model's performance on a previously separated test set, comprised of 30% of the one million original samples, by calculating the prediction error described in Equation 4.1. The distributions of these errors are shown in Figure 6.1, and the average errors for the Logistic Regression, Random Forest and Gradient Boosting classifiers were 33.03%, 2.00% and 2.03%, respectively.

Then, for each classifier, we apply the proposed pipeline using Equation (4.3) as the quality function, and Beam Search as the search algorithm. The output of the subgroup discovery is a list of the 20 hardest subgroups found for each of the three classes, an example of which is shown in Table 6.1 (for the Random Forest classifier). The subgroup found as hard for that classifier that displayed the most quality, according to the chosen measure, is when the credit risk score is 192 or more and the applicant has no other credit cards. This subgroup covers 42,260 samples, and the model has an average error of 5.2% when predicting the class of these samples, 160% higher than the average error of this model on the entire dataset.

At a first glance, we can already see that the majority of the subgroups found are composed of one feature that intuitively is related to fraud and one that is not. While high credit risk scores, high proposed credit limits (which can increase the monetary impact

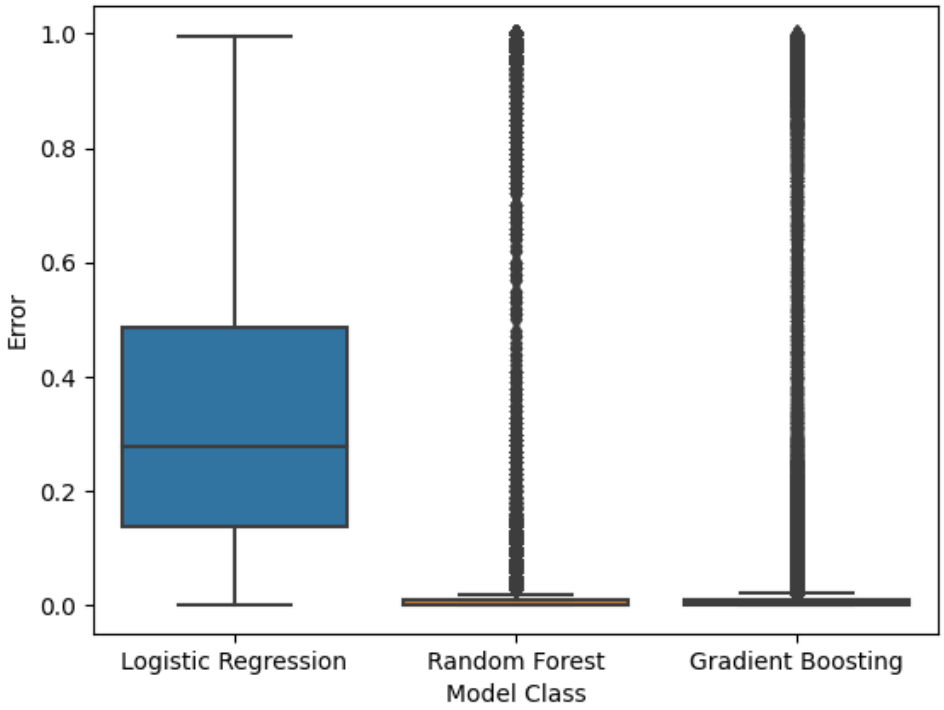

Figure 6.1: Box plots of the sample prediction errors of each classifier on the BAF dataset

Table 6.1: Subgroups mined for the Random Forest classifier in the BAF dataset

Quality	Subgroup	Size	Average Error
6.659	credit_risk_score $\geq$ 192 $\wedge$ has_other_cards = 0	42260	0.052
6.482	credit_risk_score $\geq$ 192 $\wedge$ keep_alive_session = 0	26503	0.060
6.439	credit_risk_score $\geq$ 192 $\wedge$ prev_address_months_count: [-1:6[	46342	0.050
6.315	credit_risk_score $\geq$ 192 $\wedge$ income $\geq$ 0.90	18432	0.067
6.114	credit_risk_score $\geq$ 192 $\wedge$ phone_home_valid = 0	34628	0.053
6.044	customer_age $\geq$ 50 $\wedge$ phone_home_valid = 0	22347	0.060
5.970	keep_alive_session = 0 $\wedge$ proposed_credit_limit $\geq$ 1000	29901	0.055
5.942	credit_risk_score $\geq$ 192 $\wedge$ customer_age $\geq$ 50	15497	0.068
5.846	income $\geq$ 0.90 $\wedge$ proposed_credit_limit $\geq$ 1000	19513	0.062
5.817	income $\geq$ 0.90 $\wedge$ prev_address_months_count: [-1:6[	47580	0.047
5.675	customer_age $\geq$ 50 $\wedge$ keep_alive_session = 0	25253	0.056
5.603	prev_address_months_count: [-1:6[$\wedge$ proposed_credit_limit $\geq$ 1000	53645	0.044
5.599	customer_age $\geq$ 50 $\wedge$ has_other_cards = 0	39538	0.048
5.596	has_other_cards = 0 $\wedge$ proposed_credit_limit $\geq$ 1000	49999	0.045
5.589	income $\geq$ 0.90 $\wedge$ keep_alive_session = 0	29816	0.052
5.565	customer_age $\geq$ 50 $\wedge$ proposed_credit_limit $\geq$ 1000	17665	0.062
5.464	has_other_cards = 0 $\wedge$ income $\geq$ 0.90	48496	0.045
5.354	credit_risk_score $\geq$ 192 $\wedge$ current_address_months_count: [73:155[	13462	0.066
5.330	credit_risk_score $\geq$ 192 $\wedge$ proposed_credit_limit $\geq$ 1000	42483	0.046
5.328	income $\geq$ 0.90 $\wedge$ phone_home_valid = 0	37841	0.047

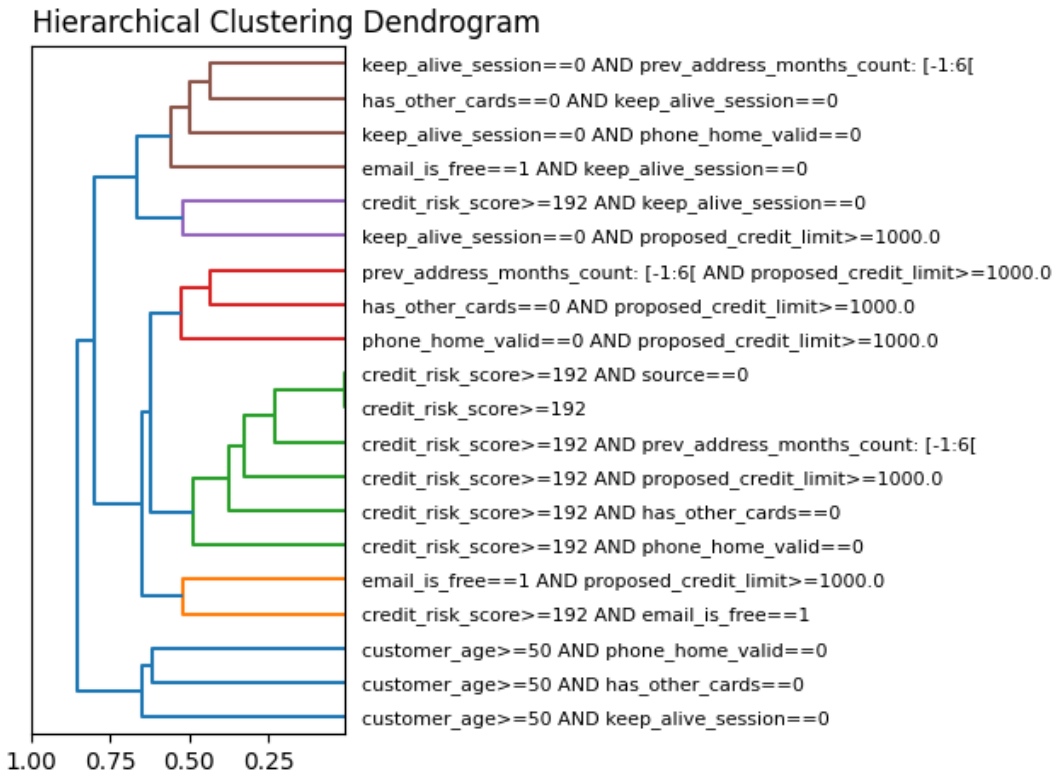

Figure 6.2: Dendrogram showing the 20 subgroups found for the Logistic Regression model clustered according to their similarity

of the fraud) and not having a valid home phone number are probably related to higher rates of fraud, on the other side, being more than 50 years old, having a high income (greater than 0.9), and not having multiple credit cards may be related to a lower risk of fraud. This behavior in the hard subgroups is consistent with the findings of [36], where they also found that the hard regions were commonly composed of features associated with conflicting effects on the target variable.

The next step is to run the Subgroup Summarization procedure, as described in Section 4.3, for the subgroups mined for each classifier. The result of the hierarchical clustering procedure can be visualized in Figure 6.2, which shows an example for the Logistic Regression classifier. We can see that one pair of subgroups had their distance close to zero, which means they cover basically the same set of samples. This means that all instances in the test set that had a credit risk score of 192 or more also had the source of the account opening application as the internet (the value 1 for **source** means the application was done through the mobile app).

This visualization is also useful for organizing the subgroups into more abstract situations that can be perceived by the user. For instance, the final three junctions in the dendrogram (that display distances of more than 0.75) can be separated as customers that did not opt for the keep-alive in their online session, subgroups associated with credit risk

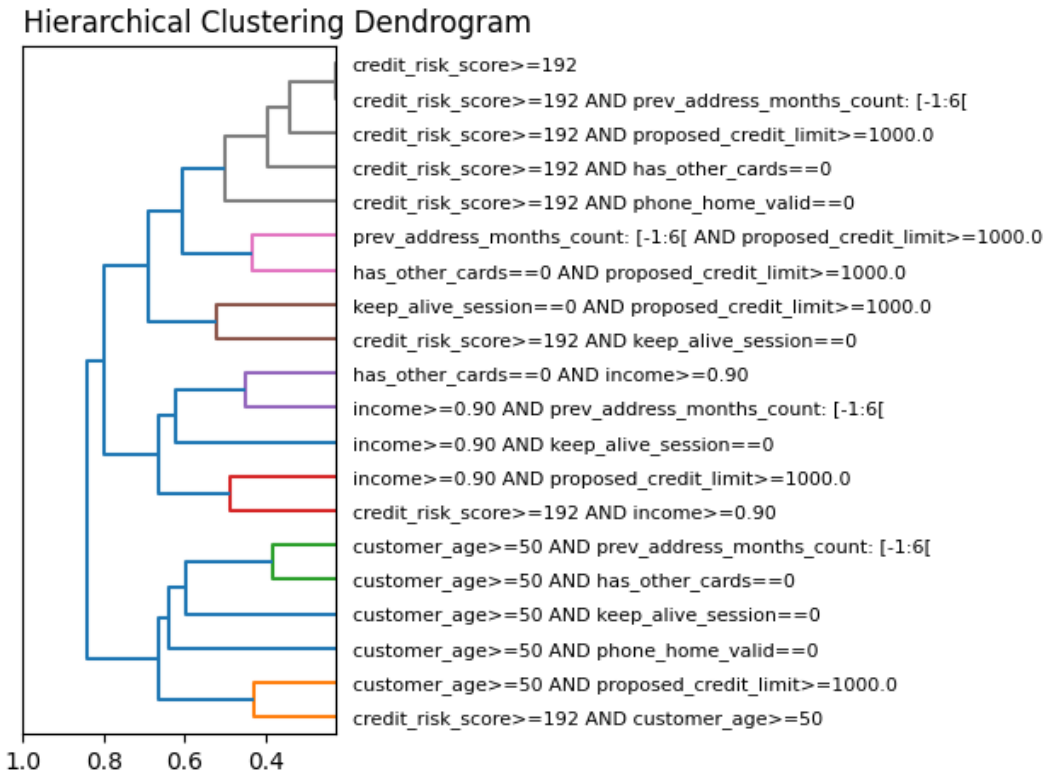

Figure 6.3: Dendrogram showing the 20 subgroups found for the Random Forest model clustered according to their similarity

(high-risk score or high proposed limit), and older customers. Based on the splits seen in the dendrogram, we chose a Jaccard distance threshold of 0.5 for the redundancy filter, in an effort to eliminate the subgroups that covered basically the same set of samples. This same threshold was applied to the Random Forest and Gradient Boosting models, and their dendrograms are shown in Figure 6.3 and 6.4, respectively. This threshold pruned seven subgroups from the result set of the Logistic Regression model, eight from the Random Forest, and six from the Gradient Boosting classifier.

After reducing the redundancy in the subgroups mined, we can choose some of them to visualize and understand how they relate to the distribution of the data itself. Figure 6.5 shows one of the subgroups mined for the Gradient Boosting classifier, with a zoomed view for better visualization of the classes. There seems to be a higher rate of fraud in this subgroup, and the model is having trouble separating the classes inside the subgroup. So, from the point of view of a user of the proposed pipeline, one way to improve this model would be to better study the behavior of the fraudulent cases that have higher incomes, what kind of frauds this group attempted, and what other information could help the model distinguish the fraudulent cases in this situation. Another way to interpret some results is by focusing on how the model is used, and what is the expected business outcome. Since high income customers tend to generate higher revenues for a

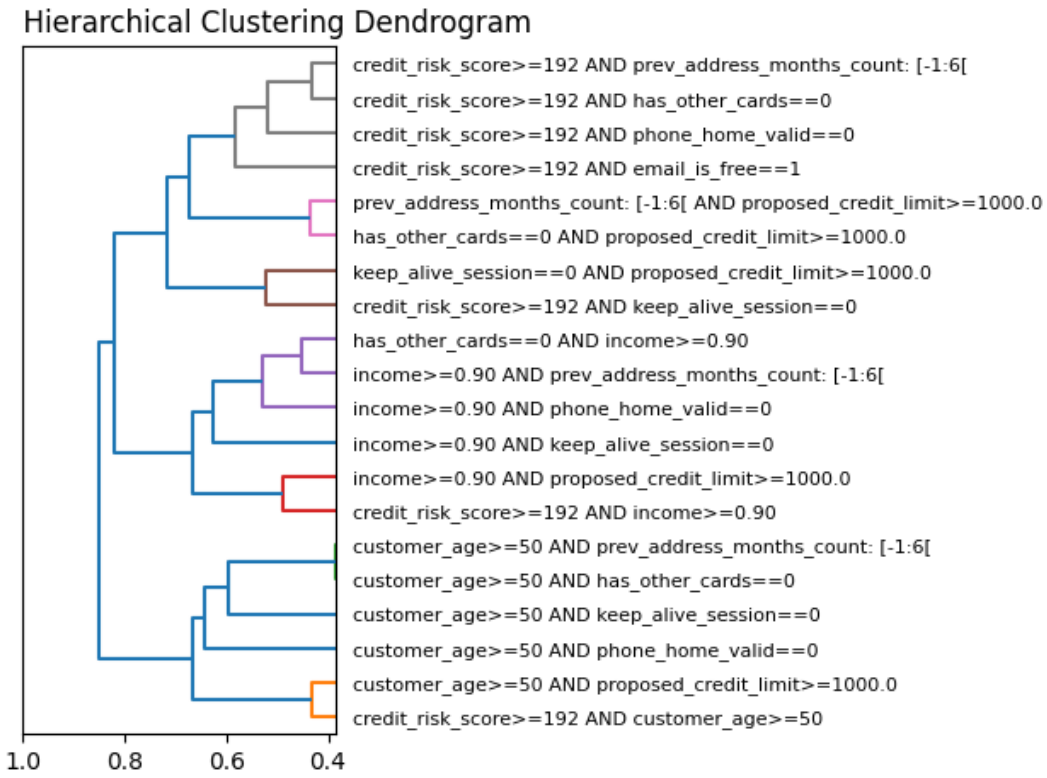

Figure 6.4: Dendrogram showing the 20 subgroups found for the Gradient Boosting model clustered according to their similarity

bank, it could be interesting to segment the usage of the model for this kind of customer. One could devise a strategy that, for most customers, relies heavily on the model's outputs to approve or not the account opening request, but for higher income customers with high credit risk scores, the analysis would also be double-checked by a fraud operations desk.

Finally, the last step of the pipeline is to run the Subgroup Comparison step, described in Section 4.5, to understand how different the hardest subgroups mined for each classifier are from each other. For that, we use a Venn diagram, shown in Figure 6.6. For the Logistic Regression classifier, the one which had the highest errors, we can see that a significant part (5) of the hardest subgroups are unique to that model class. This uniqueness in the hard subgroups found for this model can be explained by the significantly worse average performance, probably due to the model being linear and struggling to accommodate non-linear behaviors. Between the Random Forest and the Gradient Boosting model, there is much more intersection in the hardest subgroups, and seven subgroups appear as hard for all classifiers, shown in Table 6.2.

When a particular subgroup is consistently challenging across multiple classifiers, it indicates that these models both face difficulties in accurately predicting that group of samples. This could be due to the complexity of the samples themselves or the inadequacy of features provided to the machine learning models for distinguishing between

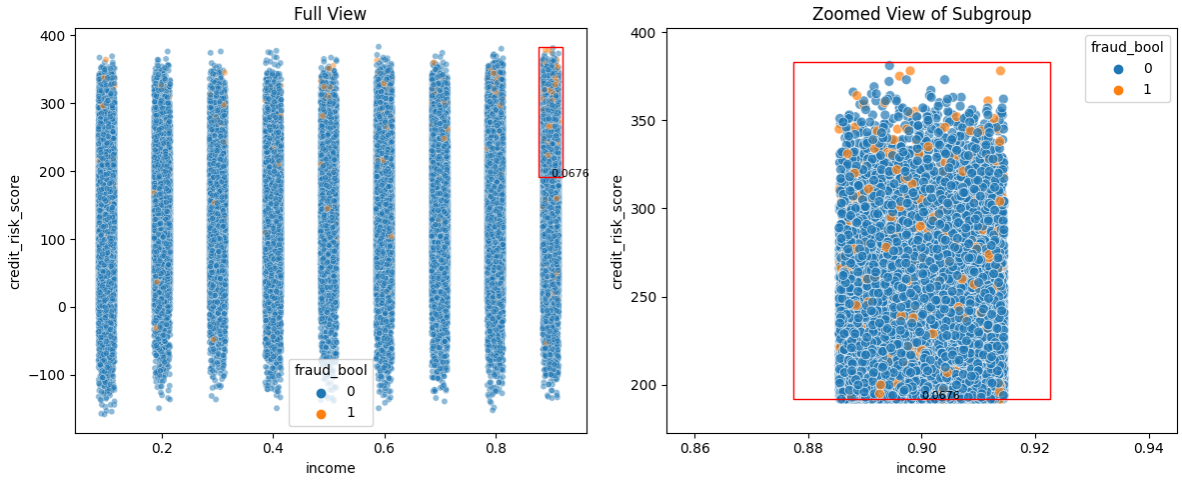

Figure 6.5: Subgroup visualization for one hard subgroup mined for the Gradient Boosting classifier.

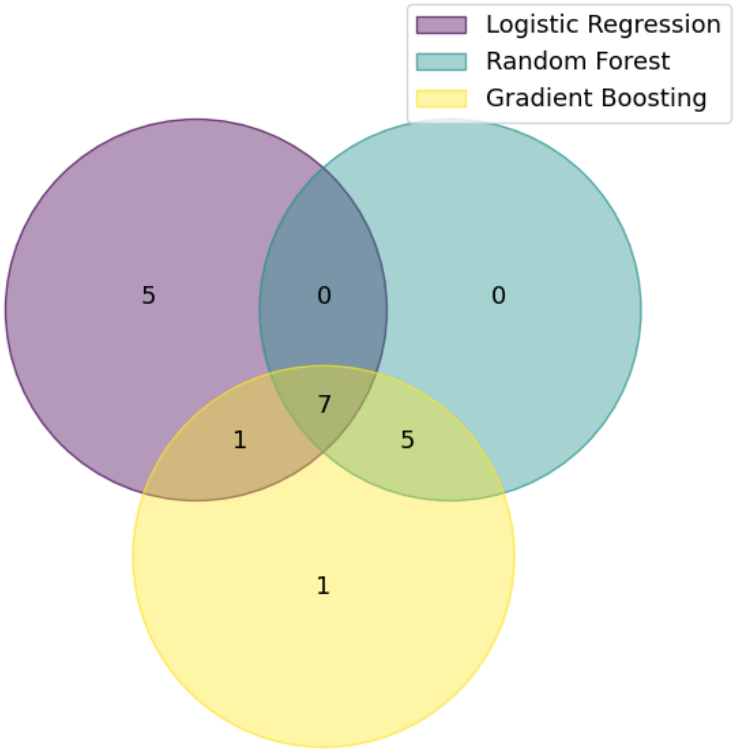

Figure 6.6: Venn Diagram of the hardest subgroups mined for each model class

Table 6.2: Subgroups that were selected as hard for all classifiers

Subgroup	Size
has_other_cards $= 0 \wedge$ proposed_credit_limit ≥ 1000	49999
customer_age $\geq 50 \wedge$ keep_alive_session $= 0$	25253
keep_alive_session $= 0 \wedge$ proposed_credit_limit ≥ 1000	29901
customer_age $\geq 50 \wedge$ phone_home_valid $= 0$	22347
credit_risk_score $\geq 192 \wedge$ keep_alive_session $= 0$	26503
credit_risk_score $\geq 192 \wedge$ has_other_cards $= 0$	42260
customer_age $\geq 50 \wedge$ has_other_cards $= 0$	39538

the two classes effectively. From the pipeline user's perspective, it is beneficial to understand that these observed errors likely do not stem from the model's assumptions, as is the case with subgroups identified solely by the Logistic Regression model. Such visualizations are instrumental in pinpointing areas within the dataset impacted by different uncertainties. Subgroups identified as difficult by several classifiers are likely dominated by aleatoric uncertainty, as explored in [32]. These are inherently challenging subgroups for any classifier, suggesting that the difficulty is intrinsic and that this type of uncertainty cannot be reduced through model adjustments or training parameter tweaks. On the other hand, subgroups that present difficulties to one classifier but not others might illustrate epistemic uncertainty, where the model struggles specifically with recognizing certain sample classes due to its own assumptions. This kind of uncertainty might be mitigated by altering model assumptions or acquiring more problem-specific information. For a data professional, distinguishing between the uncertainties affecting the model's outcomes is crucial, as it directs efforts towards reducing epistemic uncertainty, rather than investing in addressing an irreducible aleatoric uncertainty.

Chapter 7

Conclusion

This research introduces a novel pipeline for assessing the predictive performance of supervised Machine Learning models by focusing on areas within datasets where models are most prone to errors. By employing Subgroup Discovery techniques alongside detailed visualization methods, the proposed approach offers a more nuanced view of model performance, highlighting and describing specific regions in the dataset where traditional performance metrics may fail to reveal critical weaknesses.

The key advantage of this pipeline lies in its ability to automate the identification of high-error regions, which is invaluable for data scientists working with models where localized mistakes can have significant consequences, such as in healthcare or autonomous systems. The interpretability of the subgroups identified by this method allows practitioners not only to detect problematic areas but also to gain insight into the features that contribute to these errors. This interpretability is essential for improving model performance and making informed decisions about model usage in real-world contexts. Furthermore, the pipeline is built with open-source Python packages and is made available in [8] enabling its reproduction by any member of the community.

However, the pipeline has some limitations. The use of heuristic search methods like Beam Search, while efficient, may not always guarantee the discovery of the most optimal subgroups, particularly in large or complex datasets, and other subgroup discovery algorithms could be used in order to enhance its ability to identify these regions even in larger datasets. Moreover, the current approach to reducing redundancy in subgroups, though effective, could be further refined by integrating it into the subgroup discovery algorithm, as is done in SD-Map and other methods, and to improve the clarity and utility of the final results presented to users.

For future work, we intend to tackle these limitations by further investigating the connection between the Subgroup Discovery algorithm and the redundancy reduction step, in order to generate a set of subgroups that already limits the amount of redundancy. Moreover, another aspect that can be enhanced is the search space enumeration, in order to generate more flexible subgroups, rather than using percentiles of the numeric feature values. This limitation can also be addressed by choosing a Subgroup Discovery algorithm that natively supports the use of numeric features, such as SD-Map* or evolutionary

algorithms.

In summary, the pipeline presented in this thesis assists the data professional in model evaluation for supervised machine learning. By moving beyond global performance metrics to a more detailed data-driven analysis of errors, it offers new opportunities to improve the robustness and reliability of the model. As Machine Learning continues to play a crucial role in decision-making processes, tools like the one proposed here will be increasingly important to ensure that models perform reliably across all regions of the dataset.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

[2] Martin Atzmueller. Subgroup discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1):35–49, 2015.

[3] Martin Atzmueller and Frank Puppe. Sd-map–a fast algorithm for exhaustive subgroup discovery. In *Knowledge Discovery in Databases: PKDD 2006: 10th European Conference on Principles and Practice of Knowledge Discovery in Databases Berlin, Germany, September 18-22, 2006 Proceedings 10*, pages 6–17. Springer, 2006.

[4] Francisco Berlanga, María José Del Jesus, Pedro González, Francisco Herrera, and Mikel Mesonero. Multiobjective evolutionary induction of subgroup discovery fuzzy rules: a case study in marketing. In *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining: 6th Industrial Conference on Data Mining, ICDM 2006, Leipzig, Germany, July 14-15, 2006. Proceedings 6*, pages 337–349. Springer, 2006.

[5] Alexei Botchkarev. Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*, 2018.

[6] Hong Cheng, Xifeng Yan, Jiawei Han, and S Yu Philip. Direct discriminative pattern mining for effective classification. In *2008 IEEE 24th International Conference on Data Engineering*, pages 169–178. IEEE, 2008.

[7] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1550–1553. IEEE, 2019.

[8] Almeida Daniel. A new pipeline for understanding model performance based on subgroup discovery, May 2025. URL https://github.com/ddmealmeida/uncertainty-regions.

[9] María José Del Jesus, Pedro González, Francisco Herrera, and Mikel Mesonero. Evolutionary fuzzy rule induction process for subgroup discovery: a case study in marketing. *IEEE Transactions on Fuzzy Systems*, 15(4):578–592, 2007.

[10] Wouter Duivesteijn and Julia Thaele. Understanding where your classifier does (not) work–the scape model class for emm. In *2014 IEEE International Conference on Data Mining*, pages 809–814. IEEE, 2014.

[11] Cyril Esnault, May-Line Gadonna, Maxence Queyrel, Alexandre Templier, and Jean-Daniel Zucker. Q-finder: an algorithm for credible subgroup discovery in clinical data analysis—an application to the international diabetes management practice study. *Frontiers in artificial intelligence*, 3:559927, 2020.

[12] Dragan Gamberger and Nada Lavrac. Expert-guided subgroup discovery: Methodology and application. *Journal of Artificial Intelligence Research*, 17:501–527, 2002.

[13] Henrik Grosskreutz and Stefan Rüping. On subgroup discovery in numerical domains. *Data mining and knowledge discovery*, 19:210–226, 2009.

[14] Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable ai methods-a brief overview. In *International workshop on extending explainable AI beyond deep models and classifiers*, pages 13–38. Springer, 2022.

[15] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.

[16] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning*, 110 (3):457–506, 2021.

[17] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[18] Sérgio Jesus, José Pombal, Duarte Alves, André Cruz, Pedro Saleiro, Rita P. Ribeiro, João Gama, and Pedro Bizarro. Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation. *Advances in Neural Information Processing Systems*, 2022.

[19] Yan Jia, John McDermid, Tom Lawton, and Ibrahim Habli. The role of explainability in assuring safety of machine learning in healthcare. *IEEE Transactions on Emerging Topics in Computing*, 10(4):1746–1760, 2022.

[20] Branko Kavšek and Nada Lavrač. Apriori-sd: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20(7):543–583, 2006.

[21] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The uci machine learning repository, 2017. URL https://archive.ics.uci.edu.

[22] Willi Klösgen. *Explora: a multipattern and multistrategy discovery assistant*, page 249–271. American Association for Artificial Intelligence, USA, 1996. ISBN 0262560976.

[23] Willi Klösgen and Michael May. Spatial subgroup mining integrated in an object-relational spatial database. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 275–286. Springer, 2002.

[24] Nada Lavrac, Branko Kavsek, Peter Flach, and Ljupco Todorovski. Subgroup discovery with cn2-sd. *J. Mach. Learn. Res.*, 5(2):153–188, 2004.

[25] Dennis Leman, Ad Feelders, and Arno Knobbe. Exceptional model mining. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19*, pages 1–16. Springer, 2008.

[26] Florian Lemmerich. *Novel techniques for efficient and effective subgroup discovery*. Bayerische Julius-Maximilians-Universitaet Wuerzburg (Germany), 2014.

[27] Scott Lundberg. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[28] Marvin Meeng and Arno Knobbe. Flexible enrichment with cortana–software demo. In *Proceedings of BeneLearn*, pages 117–119, 2011.

[29] Marvin Meeng and Arno Knobbe. For real: a thorough look at numeric attributes in subgroup discovery. *Data Mining and Knowledge Discovery*, 35(1):158–212, 2021.

[30] Puck JAM Mulders, Edwin R van den Heuvel, Pytrik Reidsma, and Wouter Duivesteijn. Introducing exceptional growth mining—analyzing the impact of soil characteristics on on-farm crop growth and yield variability. *Plos one*, 19(1):e0296684, 2024.

[31] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[32] M Elisabeth Paté-Cornell. Uncertainties in risk analysis: Six levels of treatment. *Reliability Engineering & System Safety*, 54(2-3):95–111, 1996.

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

[34] Barbara FI Pieters, Arno Knobbe, and Sašo Dzeroski. Subgroup discovery in ranked data, with an application to gene set enrichment. In *Proceedings preference learning workshop (PL 2010) at ECML PKDD*, volume 10, pages 1–18, 2010.

[35] João Pimentel, Paulo J Azevedo, and Luis Torgo. Subgroup mining for performance analysis of regression models. *Expert Systems*, 40(1):e13118, 2023.

[36] Ricardo BC Prudêncio and Telmo M Silva Filho. Explaining learning performance with local performance regions and maximally relevant meta-rules. In *Brazilian Conference on Intelligent Systems*, pages 550–564. Springer, 2022.

[37] Martin Rabe, Stefan Milz, and Patrick Mader. Development methodologies for safety critical machine learning applications in the automotive domain: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 129–141, 2021.

[38] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[39] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[40] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. An instance level analysis of data complexity. *Machine learning*, 95:225–256, 2014.

[41] Kate Smith-Miles and Mario Andrés Muñoz. Instance space analysis for algorithm testing: Methodology and software tools. *ACM Computing Surveys*, 55(12):1–31, 2023.

[42] Luis Torgo, Paulo Azevedo, and Ines Areosa. Beyond average performance–exploring regions of deviating performance for black box classification models. *arXiv preprint arXiv:2109.08216*, 2021.

[43] Maria Gabriela Valeriano, João Luiz Junho Pereira, Carlos Roberto Veiga Kiffer, and Ana Carolina Lorena. Explaining instances in the health domain based on the exploration of a dataset's hardness embedding. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1598–1606, 2024.

[44] Matthijs Van Leeuwen and Arno Knobbe. Diverse subgroup set discovery. *Data Mining and Knowledge Discovery*, 25:208–242, 2012.

[45] Stefan Wrobel. An algorithm for multi-relational discovery of subgroups. In *European symposium on principles of data mining and knowledge discovery*, pages 78–87. Springer, 1997.