Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (33^{th} cycle)

Pattern-based algorithms for Explainable AI

Eliana Pastor

* * * * * *

Supervisor
Prof. Elena Baralis

Doctoral Examination Committee:
Dr. Francesco Bonchi, Referee, ISI Foundation, Italy
Prof. Paolo Merialdo, Referee, Universitá Roma Tre, Italy
Dr. Sihem Amer-Yahia, CNRS, University of Grenoble Alpes, France
Prof. Luca de Alfaro, University of California, Santa Cruz (UCSC), USA
Prof. Paolo Garza, Politecnico di Torino, Italy

Politecnico di Torino
2021

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Eliana Pastor
Torino, 2021

Summary

Machine learning models are increasingly adopted in a wide range of critical areas. However, most high-performing models lack interpretability. Especially in critical tasks as health care, criminal justice, and finance, understanding the model behavior is of fundamental importance.

The thesis addresses the problem of the lack of interpretability of classification models. We propose post-hoc techniques to analyze the behavior of classifiers, from the perspective of individual instance predictions and data subgroups. The proposed techniques build on the notion of patterns. Patterns are conjunctions of attribute-value pairs intrinsically interpretable. We leverage patterns to capture relevant associations of attribute values and to define subgroups in the attribute domain. The proposed techniques are model agnostic because they do not rely on the knowledge of the inner workings of any classification paradigm.

At the level of individual instance predictions, we consider the lack of understanding of the reasons behind individual predictions for black-box models. We propose a rule-based explanation method that explains the prediction of any classifier on a specific instance by analyzing the joint effect of feature subsets on the classifier prediction. The approach relies on a local rule-based model to identify the relevant patterns determining locally the prediction. The extracted local patterns provide a qualitative understanding of the reasons behind predictions. We provide a quantitative understanding through the notion of prediction difference. We exploit a removal-based technique to compute the influence on individual predictions of feature values and subsets of feature values derived from patterns. We then propose an interactive tool that leverages the rule-based explanation method for a human-in-the-loop inspection of the reasons behind model predictions.

From the subgroup perspective, we investigate the behavior of models on data subgroups. Specifically, we address the problem of identifying and characterizing data subgroups in which a classification model behaves differently. The identification of these critical data subgroups plays an important role in many applications, for example, model validation and testing, evaluation of model fairness, and identification of bias. We introduce the notion of divergence to capture the different behavior of the model on data subgroups with respect to the overall behavior. We characterize data subgroups via patterns and we leverage frequent pattern mining

techniques for their automatic extraction. We use the notion of Shapley value to quantify the contribution to the pattern divergence of the attribute values identifying a data subgroup. We also introduce a generalization of the Shapley value to estimate the global contribution to the divergent model behavior. We then propose an interactive system for the exploration of divergent subgroups which supports drill-down operations and human-in-the-loop inspections of peculiar subgroup behaviors.

The work is supported by theoretical analysis and experimental evaluations, showing the effectiveness of the proposed approaches to reveal the behavior of the model at the individual instance and subgroup level.

Acknowledgements

I would like to thank my supervisor Prof. Elena Baralis. Her guidance, support, and advice were precious during the course of my Ph.D.

I would like to thank all my colleagues and members of the DBDMG research group. Thank you for the continuous interaction, your time, and your friendship.
I would like to generally extend my gratitude to all the colleagues and researchers that I had the pleasure to meet during these years.

I would like to thank my parent and my sister Viviana for supporting me throughout this journey. Thank you for all you have done for me. Thank you Viviana for being there when I need you the most and for always believing in me.

A special acknowledgment to my grandmother who helped shape me into who I am today.

Finally, I would like to thank all my friends for their understanding and support.

To my parents
and my sister Viviana

Contents

List of Tables

List of Figures

Chapter 1

On the need of Explainable AI: Concepts and taxonomy

The chapter explores the relevant concepts of Explainable Artificial Intelligence (XAI). Section 1.1 firstly illustrates the importance of interpretability in machine learning models and its strict relations with goals and desiderata of machine learning research. In Section 1.2, we introduce notions, taxonomies, and evaluation approaches in XAI research. Finally, Section 1.3 presents an overview of the thesis and its main contributions.

1.1 On the need of explainability

Machine learning models are extensively adopted for a wide range of applications. This adoption includes critical areas as health care, criminal justice, finance, and insurance. Many of the adopted models are considered black boxes which do not disclose their internal logic yielding the prediction. In these high-stakes applications, the lack of interpretability may have serious consequences [142]. Without the understanding of the behavior of machine learning models, the model predictions cannot be legitimated and justified. Insights into how a model arrives at its decisions allow evaluating if the model can be trusted. Only by inspecting the reasons behind predictions, domain experts can detect if the model has learned wrong associations or potential bias. The analysis can reveal if a disparate behavior and impact is observed for different groups of observations. It may disclose the presence of potentially discriminatory behavior towards groups characterized by sensitive data as race or gender. Considering the important implications especially for high-risk tasks, in recent years there is an increasing demand for transparency and explainability of machine learning models. The call comes from multiple stakeholders, ranging from the developers, the data scientists, the ethicists demanding ethical AI to end-users [134].

The demand for improving the understandability of models and their outcomes comes also from institutions. The European Union approved the General Data Protection Regulation (GDPR) [54], a regulation for ensuring personal data protection. Paragraph 1 of Article 15 states that the data subject has the right to be informed of different aspects related to its data and to receive "meaningful information about the logic involved" in case of automated decision-making [54]. The recital, which provides additional information and clarifies the intention of the regulation, states that the data subject should have "the right to obtain [...] an explanation of the decision reached". For some authors, this requirement legally mandates a "right to explanation" [64]. The extension of the "right to explanation" is highly debated [54, 148, 165]. Authors as Wachter et al. state that the GDPR refers to a "right to be informed" [165]. The authors additionally debate on the scope of the right of explanation, distinguishing among *ex-ante* and *ex-post* explanation. An *ex-ante* explanation refer to an explanation provided before the actual use of personal data in the automated decision-making process [165]. It hence concerns the functionality of the system itself as the system logic and the considered attributes. Ex-ante explanations should provide the indication of the consequences of the application of automatic-decision making, specifying whether a different impact based on the input data of the subject is expected on the outcome function. The *ex-post* explanation refers to the logic of specific decisions [165]. Therefore, ex-post explanations should directly provide indications of the reasons behind the predictions. Despite the debate, both forms of explanations are hard to achieve in the case of black-box models. Since they do not disclose their inner working, data controllers cannot directly investigate the impact of the systems on data subjects and why individual predictions are made.

The recent demand for enhancing the comprehensibility of machine learning models has brought to the increasing work in this direction. Explainable AI (XAI) in an emerging field of artificial intelligence research to enable the understandability of models and their decision to humans [12]. The research interest in XAI is also motivated by its strict link with multiple concepts, also referred to as *desiderata* [98] or goal [12] of XAI research. These goals represent objectives of real-world applications which are enabled or achieved with the understanding of the model behavior. In the following part of the section, we outline the main desiderata and their link with model understanding.

- **Trust.** The adoption of a machine learning model, especially in critical applications, depends on the trust of domain experts place in it. If they do not trust it, users probably will not deploy it [138]. Several researchers consider the understandability of model behavior as a prerequisite for trustworthiness [138, 135]. Domain experts can decide if they can trust a model or its predictions only if are able to understand it or understand why predictions are made [138]. We can hence identify two form of trustworthiness: *trust a*

model or *trust a prediction*. The former considers the understanding of the model as a whole and trusting it will perform well in real scenarios. The latter refers to the confidence in individual predictions and take decisions based on them. In both cases, trusting requires a degree of understanding of the model behavior, comparing it with the prior domain knowledge of the problem, and decide on their coherence.

- **Knowledge.** This desiderata is also refer to as informativeness [98], interestingness [22] and scientific understanding [48]. The understanding of how a model works can allow experts to discover new patterns and associations. The comprehensibility of model behavior is of key importance for knowledge discovery in data [13]. The discovered form of knowledge should have the characteristic of being previously unknown, enough certain, and *interesting* according to user measure of interest [58]. In addition, the explanations of the model behavior should be *informative* for supporting the decision making [98].

- **Causality.** Domain experts can leverage the discover knowledge capture by a model to potentially find causality among data. While correlations do not imply causation, correlations among the data can be in fact exploited to search for causal relations [12]. Again, novel patterns and associations in the data can be inspected only if users can grasp at a certain degree the reasons behind model predictions.

- **Error analysis and debugging.** If experts can analyze why decisions are made they could potentially find wrong associations. These can be inspected, studied, and potentially solved. As Ribeiro et al. illustrate, the insights of the model behavior could be helpful "to convert an untrustworthy model into a trustworthy one" [138]. A compelling example is discussed by Caruana et al. in their pneumonia risk case study, where the goal was to estimate the probability of death of patients with pneumonia [28]. The model behavior could be analyzed only because of its interpretable nature. Patients with high-risk conditions as asthma, chronic lung disease, or a history of chest pain were considered by the model at lower risk of dying. Thanks to the interpretability of the model, domain experts were able to discover these unexpected behaviors, study them, and understand their source. Patients with these high-risk conditions are admitted directly to the intensive care unit and treated with greater attention, lowering their actual mortality rate. Data scientists were then able to fix the model, by eliminating or editing the model variables. The analysis of the model can also reveal if differences in the model behavior occur for subgroups of data. The identification of data subgroups in which a model performs poorly can help data scientists in model debugging.

- **Fairness.** The demand for explainable AI can be framed in the broader context of the need for a more responsible AI [12]. Responsible AI research aims to consider fairness, accountability, and privacy in the development of AI techniques. The deployment of machine learning models in high-risk applications can negatively affect people's lives and impact their equal participation and treatment. Model decisions may be inherently discriminatory [18]. Machine learning models may in fact learn from biased and unfair data. Data are collected from our society that is discriminatory and unequal. This results in systematically learning biased models, even if unintended. Models can hence embed pre-existing bias and re-iterate it.
  The concern of potentially unfair and discriminating machine learning is also expressed by institutions [54]. Recital 71 of the European GDPR recognizes the urgency of ensuring fair and transparent data processing. It requires data controllers to apply appropriate procedures and implement measures that prevent discriminatory effects on the basis of sensitive data as ethnic origin, political opinion, and sexual orientation [54]. Only the inspection of model decisions can provide insights into the results and the relations influencing the predictions. This allows users to audit the fairness of the model and perform ethical analysis. The model understanding may reveal the presence of bias in the model and of possible discriminatory decisions.
  Societal bias [18] is a growing concern and researchers are increasingly working on measuring and ensuring fairness. The evaluation of fairness often considers two notions for the assessment of the disparate behavior of models: disparate treatment and disparate impact [18]. The *disparate treatment* assesses if decisions are (in some measure) based on sensitive and protected attributes as race and gender [18]. In Chapter 4, we address this issue from the *individual* perspective. We expose the reasons behind individual predictions made by a generic classifier. The explanation can reveal if decisions are based on sensitive attributes and hence the disparate treatment of the classifiers.
  The notion of *disparate impact* considers the different impact of a machine learning model. A disparate impact occurs if the outcomes of the model disadvantage (or benefit) groups of people characterized by protected attributes. To this aim, human experts are frequently required to manually identify the sensitive attributes or subgroups that may potentially be affected by a different model behavior. In Chapter 6, we propose an automatic approach to identify subgroups of data where a divergent model behavior is observed. We audit the classifier behavior at the *group* level to reveal the subgroups affected by a disparate behavior and the attribute values that determine it. The identification of the potential discriminatory factors and differently treated subgroups allows practitioners to manage the negative effects of the model, identify appropriate solutions and turn an unfair model into a fair one [64, 98].

- **Transferability.** Understanding the reasons behind predictions allows users to assess if the model is able to generalize and it could be applied in different scenarios [98]. The assessment of the ability of the model transferability is particularly relevant in the case of concept drift when the properties and relations between the input data and the target variable change over time.

- **Interactivity.** Another desideratum is the interaction between the model and the users. Users should be able to actively interact with the model by investigating their assumptions on its behavior [73, 90]. In Chapters 5 and 7, we address this desideratum by proposing two novel interactive systems that allow users to inspect the reasons behind predictions of individual observations [127] and the behavior of the model on data subgroups, respectively [130].

Other desiderata of explainable AI research are *confidence* [12] as a generalization of the *robustness* and *stability* of the model behavior, *privacy awareness* [12, 51] and *accessibility* [12].

1.2 Explainability: Notions and Taxonomy

The section firstly outlines the multiple definitions proposed for explainable AI. We then classify explainable AI approaches, illustrating their scope and arguing on the problematic nature of interpretability assessment.

1.2.1 The notion of explainability

In the XAI literature, several terms have been proposed to refer to the understanding of the model behavior. The multitude of terms and the disagreement on their use is also attributable to the subjective nature of the understanding process itself and its strict link to humans and their cognitive ability. The term "*to interpret*" itself comprises several definitions, from "to clarify or explain the meaning of; elucidate" [45] to "make understandable", "to translate", "to have or show one's own understanding of the meaning of; construe" [46], "to describe the meaning of something; examine in order to explain" [47] These definitions, despite the differences, all involve the notion of explaining in understandable terms to someone. Commonly used terms as interpretability, understanding, comprehensibility, and explainability are often used interchangeably in the explainable AI literature [112, 114] and we will follow this line through the thesis. Some authors instead specifically distinguish the concepts of interpretability and explainability [12].

In the following, we outline the most commonly used terms and their definitions.

A widely adopted term is *interpretability*. Doshi-Velez and Kim derive its definition in the context of machine learning directly from the definition of to interpret.

They define it as "the ability to explain or to present in understandable terms to a human" [48]. Biran and Cotton in [23] and Miller in [112] define interpretability as "the degree to which an observer can understand the cause of a decision". While in [112] the term is considered equivalent and synonym of explainability, other authors differentiate it and consider it only as a synonym of model transparency [12]. In this context, interpretability is considered a passive characteristic of the model itself and refers to its degree of understandability to humans [12].

Another commonly used term is *understandability* [22]. The term focuses on the characteristic of the model behavior to be understood, by humans, in a reasonable amount of time [22]. The time constraint is introduced because we could say that any model could be understood in an infinite time [22]. The term *comprehensibility* refers to the representation of the model behavior in a form comprehensible to humans. Authors as Askira-Gelman focuses on the comprehensibility of the model workings as a fundamental for knowledge discovery [13]. Other commonly used terms are *intelligibility* [28] and *mental fit* [55, 167] that refer to a representation of the model behavior that can be grasped and evaluated "in mind" by humans.

Explainable and *explainability* are often used as synonyms of the previously mentioned terms and put the emphasis on the ability of the model or its result to provide an understanding of the relations between the input and the outcome predictions in an informative and understandable way. Explanations reveal the reasons behind the model predictions. For some authors, explainability differs from interpretability since it is considered as an active characteristic of the model [12]. In this context, explainability refers to procedures to allow users to understand the inner behavior of the models. It can hence be considered as a broader notion of interpretability [12]. An interpretable model can be explained using its own ability to present the result in understandable terms.

The outlined definitions, despite the different focus, all consider the notion of the ability of human beings to interpret and understand the behavior of the model. A recent definition of explainable AI specifically considers the audience for which explainability is meant [12]. The model behavior should be provided in a comprehensible and clear way depending on the audience [114].

1.2.2 Explainable AI approaches and their taxonomy

Explainable AI techniques to enable the understanding of the model behavior can be categorized into two categories: interpretable by design and post-hoc explainability [12].

The first line of work in Explainable AI is the definition of interpretable models. Interpretable models, also referred to as transparent models, are models that are interpretable by design. Considering the increasing relevance of interpretability, recently several works consider interpretability constraints directly into the optimization problem, in order to allow the model comprehensibility while obtaining

high performance [7, 89, 113, 143]. Transparent models explain the reasons behind their predictions to a certain degree. The level of comprehensibility and the form of the explanations vary for the different interpretable models. An overview of interpretable models is presented in Section 2.1.1.

The second class of approaches aims at enhancing the interpretability of classification models that are not intrinsically transparent. Post-hoc explainability provides insight into the behavior of the classifier and is applied after model training. Considering the wide application of black-box models also in high-risk tasks and the need to interpret their results, several approaches have been proposed to explain the reasons behind the predictions, identifying the most relevant attributes or an explanatory set of observations. We present an overview of these approaches in Section 2.1.2.

Post-hoc explainability techniques can be classified according to two dimensions: the generality and the scope.

Degree of generalizations. Some techniques are tailored to explain only some specific classification models [12]. We refer to these approaches as *model-dependent* and we outline them in Section 2.1.2. *Model-agnostic* solutions or model-independent are instead designed to provide insights into the behavior of a generic classification model. In Section 2.1.2, we provide an overview of these solutions and we illustrate their advantages over model-dependent ones. The approaches proposed in this thesis belong to this category.

Scope of explainability. *Global explanations* aims to provides insights on the behavior of the entire model [67]. *Local explanations* explain the reasons behind individual predictions [67]. Our explanation approach LACE [129] illustrated in Chapter 4 and the interactive tool [127] for the inspection of predictions outlined in Chapter 5 have a local scope.

Recently, several works aim to characterize the behavior of the classifier at the *subgroup level* [26, 34]. The analysis specifically targets the identification of peculiar behaviors of the model across data subgroups. An overview of these methods is presented in Section 2.2. In Chapters 6 we propose a novel approach [126] to identify and describe subgroups with different classification behavior than the overall one.

1.2.3 Evaluating explainability

This section discusses the difficulties in measuring interpretability and overview existing evaluation approaches. We first distinguish among two possible targets of the evaluations. The former considers as target the interpretability of the representation of the behavior of the model itself. The latter considers the evaluation of the quality of the explanations provided by the post-hoc explainability approaches.

The major difficulty in the definition of a measure of interpretability derives from its subjective nature. Despite the multitude of terms, all definitions consider the human audience as active subjects of the understanding process. The actual degree of understanding of the model behavior depends on the background, education, expertise, level of attention of the user [77].

Another obstacle derives from the lack of ground truth. For the evaluation of classification performance, data scientists can leverage a multitude of different and well-known metrics as classification accuracy, precision, recall, or F-measure. All these metrics are based on the knowledge of the true label. True explanations of the model behavior are normally not available. In almost all cases, the true association between attribute values and class value is not known. Indeed, classifiers are specifically used to capture these associations from the data.

We can identify two major approaches to evaluate explainability: functionally-grounded evaluations and human-based evaluations [48].

Functionally-grounded evaluations. Functionally-grounded evaluations [48] also refer to as proxy tasks [48] or heuristic approaches [22] are quantitative and objective evaluations of interpretability that do not require human intervention. These approaches also rely on proxies. A widely adopted heuristic measure is the size [22, 48]. The size is considered as a proxy of the model complexity. The larger is the size, the more the representation of the model is complex and so the less it is interpretable. The size can be used as a proxy of interpretability at the global, subgroups, and individual levels. At the local scope, the metric is also referred to as conciseness [4]. For example, at the global level, it can represent the number of nodes or the depth of a tree, at the subgroup level the number of attribute values characterizing the group and for an individual explanation, it can be the number of relevant attributes or the number of nodes in the path of a decision tree. However, large sizes, especially when referring to the model itself at a global level, make their comprehension difficult and time-consuming. We have to consider the human limitations in understanding and processing a big amount of information, linked to the limits of short-term memory. A popular conception in psychology is that humans can deal with seven, plus or minus two, abstract objects in their short memory at the same time [111]. Moreover, the interpretability of the model behavior model is tightly linked to the ability of humans to understand it in a reasonable amount of time. However, models with a small size, while considered interpretable, may be also considered too simple. Users may not trust simple models and reject them, questioning their ability to capture the complexity of the problem under study [59]. An additional problem of the size as a heuristic measure is that it does not allow an easy comparison between different representations of the models. As an example, we cannot easily compare the number of nodes of a tree with the number of neurons of a neural network or the number of non-zero weights of a logistic regression classifier.

The drawback of the heuristics based on proxies is that they consider only syntactical aspects, ignoring the semantics one [22, 59].

Human-based evaluations. Considering the subjective nature of interpretability and its strict relation to the audience, the other adopted approach is based on human-based evaluations. Doshi-Velez and Kim further divide human evaluations into application-grounded and human-grounded evaluation [48]. The main difference derives from the level of expertise of the audience. The *application-grounded* evaluation considers the evaluation performed by domain experts, in the scenario of real tasks and real applications. This evaluation requires designing the experimental setup with great care and involve experts on the application of interest. The *human-grounded* evaluation involves instead general users. Hence, the user study should be conducted for simplified tasks, where no particular expertise or knowledge is required. In both scenarios, the evaluation design should carefully consider the presentation of questions, the background, and expertise of the user, the length of the study to avoid tiredness or motivation loss.

With the rise of post-hoc explainability techniques, tailored solutions have been proposed to assess the quality of explanations of the behavior of black-box models. Despite the recent growth, there is no unified and agreed measure of explanation quality. The lack of consensus is again attributable to the disagreement on the definition of explanation of the model behavior and the lack of ground truth. For the evaluation, we do not have a target ground truth explanation to compare with.

When the explanation is derived from a (local) interpretable surrogate model that mimics (locally) the behavior of the classifier, an often adopted metric is the (local) *fidelity*. It measures how good the surrogate is in approximating the behavior of the original model [38, 67, 138]. In the context of individual explanation, the evaluation is based on comparing the surrogate and the black-box predicted labels [67]. As an alternative approach, the estimation considers a neighborhood of the individual prediction [138]. The notion of fidelity captures the accuracy of the surrogate model and it was also redefined to capture the precision, recall, and F1-score [67]. A recent framework address the evaluation of individual explanations extracted from local linear models [4]. The linear model can be a linear surrogate model (as in LIME [138]) or the local linear function can be derived from explanation in form of feature importances (as in SHAP [104]). The evaluation considers quality metrics as the stability of the explanation results (reiteration similarity) and local fidelity with conciseness constraints as a measure of comprehensibility [4].

Another evaluation approach is to design the experimental setting so that the true associations or the true explanations are known. A class of solutions considers the usage of artificial datasets [33, 129, 153, 154]. For artificial datasets, the true associations among the feature values and the class label are known. The explanations produced by post-hoc explainability models can hence be compared with the

true associations. These solutions have the advantage of allowing the assessment of the ability of the models to capture the true relations among the data. However, an explanation should capture the model behavior and so what the model has learned. A high-quality explanation can differ from the true association if the model failed to learn it. We will show an example of this phenomenon in Section 4.4.2. Hence, we first need to assure the performance of the model for the true associations of interest. Another class of approaches consists of knowing the model behavior itself at the scope of interest (local, group, or global level). This can be achieved by directly learning an interpretable model and using it as a black-box model to explain [81, 129], by controlling the behavior of the classifier [33, 126] or, for local explanations, synthetic transparent classifiers [66]. The quality of the explanation given the true association or explanation is then evaluated by measuring their similarity. The metric used depends on the representation of the explanations themselves. Examples of adopted metric are the cosine similarities [66, 81] and f1-score [66, 81].

1.3 Overview of the thesis and contributions

Considering the relevance and the need for interpretability for multiple objectives, this thesis addresses the problem of the lack of transparency of classification models for structured data from both the perspective of individual predictions and subgroups. We propose post-hoc explainability approaches that leverage the notion of patterns to capture local associations of feature values and identify subgroups. Patterns are conditions of attribute-value pairs. We enhance the interpretability of black-box models by exploiting the following relevant features and properties of patterns.

- **Intrinsic interpretability.** Patterns are typically provided as a list of attribute-value pairs in textual form. Hence, they are intrinsically interpretable and understandable by humans. Their understanding does not require user expertise or training. Users can inspect patterns and directly understand which terms compose them.

- **Capturing associations**. Patterns can be used to model the associations that occur among multiple attribute-value pairs and the class labels. In Chapters 4 and 5, we leverage this feature of capturing association to derive a local explanation of the behavior of the classification model. The explanation reveals the relevant associations of attribute values of an individual prediction and the class label for the classifier.

- **Interpretable data grouping.** Patterns can be exploited to determine data subgroups. A pattern identifies the data instances satisfying its conditions. This corresponds to slicing the data in the attribute domain. The groups

identified by patterns are intrinsically interpretable. As a result, posterior techniques to describe the subgroups are not required. In Chapters 6 and 7, we use the notion of pattern to identify data subgroups for which a classifier behaves differently than overall.

Thesis outline

In the following, we outline the thesis structure and its main contributions.

Related work and background. Chapter 2 outlines existing solutions in explainable AI to enhance the understanding of the behavior of machine learning models. The analysis reviews the techniques on the basis of the illustrated taxonomies. Specifically, we focus on techniques for the understanding of the model behavior from the perspective of individual predictions and data subgroups, outlining limitations and needs for improvements.

Chapter 3 introduces background notions and algorithms used in the thesis.

Individual prediction perspective. Chapters 4 and 5 address the problem of enhancing the interpretability of classification models from the perspective of individual predictions.

Specifically, Chapter 4 describes LACE [129], an explanation method that explains the reasons behind individual predictions via a rule-based model-agnostic approach. The explanation provides a qualitative and quantitative understanding of prediction behavior. The qualitative insight is provided by a local model that captures relevant associations of attribute values in terms of patterns. The quantitative explanation is computed by estimating the prediction change when one or more attribute values derived by patterns are omitted. Existing state-of-the-art techniques generally provide only one form of explanation, qualitative or quantitative. The dual form of LACE explanations improves existing techniques and provides a more complete understanding of the model prediction behavior. Experimental results show the ability of the proposed approach to capture the reasons behind predictions.

Chapter 5 presents x-Plain [127], an interactive tool for human-in-the-loop inspections of individual predictions. It exploits LACE as the underlying explanation method. The design and purpose of the tool directly consider multiple desiderata of explainable AI research as interactivity, debugging, and trust. x-Plain allows interactive explorations of the behavior of the classification models. Through active inspections, users can debug classifiers and assess the trustworthiness of the model individual predictions.

Subgroup perspective. Chapters 6 and 7 address the understanding of model behavior from the perspective of data subgroups.

Chapter 6 describes DivExplorer [126], a novel approach for inspecting data subgroups in which a classification model exhibits peculiar (*divergent*) behaviors that differ from the overall one. Data subgroups are characterized by patterns. As a result, the identified subgroups are already interpretable and can be described by the attribute-value pairs of the patterns themselves.

Differently from existing approaches, we propose a complete exploration of the subgroups with adequate representation in the data. The identification of critical subgroups is performed exploiting frequent pattern mining techniques. We show, both theoretically and experimentally the need for a more complete exploration to identify peculiar behaviors of the model.

Given a data subgroup with peculiar model behavior and the pattern characterizing it, we can then be interested in understanding which are the attribute values of the pattern that mostly contribute. We use the notion of Shapley value [150] to determine the influence of attribute values to the subgroup peculiar behavior. We then propose a generalization of the Shapley value to estimate the global contribution to the model divergent behavior.

In Chapter 7, we consider again the desiderata of explainable AI research for the design of an interactive application to inspect classifier behavior in subgroups. We propose a web app integrating the DivExplorer algorithm that supports interactive searches, explorations, and drill-down operators [124].

Conclusions ad future work. Chapter 8 draws the conclusions and outlines future works.

Chapter 2

Related work

The chapter outlines the main existing explainable AI techniques to enable the understanding of machine learning model behavior. We analyze the approaches both from the perspective of understanding the reason behind prediction and on characterizing peculiar behaviors in data subgroups.

The chapter is organized as follows. Section 2.1 reviews the approaches to understand model predictions. We firstly outline transparent classification models and their degree of understandability (Section 2.1.1). We also review novel approaches for interpretability by design. We then examine the post-hoc techniques to enhance the interpretability of black-box models 2.1.2. The analysis considers the two dimensions of post-hoc explainability techniques, the degree of generalization and the scope of explainability. Section 2.2 outlines the approaches for characterizing the behavior in data subgroups. We firstly review supervised techniques that require human intervention for the identification of interesting subgroups (Section 2.2.1). Finally, Section 2.2.2 then outlines unsupervised techniques with an automatic identification process of the subgroups in which the model has a peculiar behavior.

2.1 Understanding the reasons behind predictions

The section outlines techniques to understand the reasons behind model predictions. We firstly outline classification approaches that are interpretable by design and we focus on their level of understandability. We then outline post-hoc explainability approaches to provide explanations of the classification behavior.

2.1.1 On the transparency of classification models

The section provides a brief description of interpretable classification models and outlines their scope and degree of interpretability. We also review recent interpretability by design solutions that consider interpretability as a target of the

optimization process.

Interpreting classification trees

Classification tree or decision tree (DT) models construct a decision tree representation to classify data [75]. Classification tree models partition the feature space by splitting the data into sub-regions with respect to cutoff feature-value pairs. At each split, the attribute and cutoff attribute value that minimize the heterogeneity of class values in each partition is determined to obtain the best split. Each split partitions the data into different and exclusive subsets. Hence, each instance of the data belongs to one single partition. The data are recursively partitioned until certain stop criteria are reached, as the minimum number of instances in a region or the maximum depth of the whole resultant tree. The nodes of the tree structure are categorized into the root node, internal nodes, and leaf nodes. The topmost node in a tree is the root node and corresponds to the first split. An internal node corresponds to a split of the feature space and denotes a test on an attribute. A leaf node is a terminal node and it is labeled with a class or a probability distribution over the classes, usually determined as the most common class value or the average outcome of the instances of the training data set that fall back in that leaf. The algorithms proposed for building trees may differ on (i) the metric to evaluate the impurity of a node, as the Gini index, the classification error rate and entropy, (ii) the structure of the splits and the number of edges (i.e. the number of partitions), (iii) the stopping criteria, and (iv) how to determine the class or the probability distribution over the classes of the leaf nodes.

Classification tree models are widely adopted for the intrinsic interpretability of their tree graphical representations [59]. Trees allow interpreting the classification results from the global to local perspectives. The entire tree can be inspected by the users. Hence, they can grasp how the model globally works and observe the whole relationships between input features and class labels learned by the tree. DT also provides local interpretability by explaining why particular decisions are made. The local explanation of a single prediction corresponds to the path from the root to the leaf node. Hence, the user can inspect the reasons for an individual instance in a more focused way.

Freitas in his position paper on the interpretability of classification models points that, in addition to the graphical representation of a tree, another advantage of DTs is that they often consider only a subset of the input features [59]. This facilitates the user comprehension of the global picture of the model and the most relevant attributes. Moreover, the hierarchical structure of the tree provides an indication of the relevance of the attributes. Often, the attributes closer to the root node are considered more relevant [59]. Another criterion is to consider the relevance of an attribute as a functusion of the number of instances having the attribute in the decision path [59]. The relevance of attribute values can be useful

both at the global and the local scope. The former allows to understand which attributes are mostly characterizing the model. The second refers to derive the relative importance of the features in the decision path.

However, concerns arise on the actual global interpretability of DT when the size of the tree increases, estimated from the number of nodes or the depth of the tree. As illustrated in Section 1.2.3, the size is a proxy of model complexity. The interpretability of a model decreases when its complexity increases. On one hand, small trees are easier to interpret. However, in real applications as medical ones, small trees are considered too simple to model the complexity of the problem of interest. On the other hand, classification trees can be so large, deep, and with so many nodes to make its understanding difficult and time-consuming. Hence, despite the transparency of trees and the possibility to inspect them, their complexity and large size, especially when considering real applications, may hinder their understandability from a global perspective.

Moreover, considering the close connection between interpretability and human beings, interpretability is highly subjective. As a result, some users may consider decision trees less comprehensible and prefer other representations as decision tables and classification rules [77].

Finally, the general understandability of DT predictions typically corresponds to a lower classification accuracy [114]. This relation is known as the comprehensibility-accuracy trade-off. As a result, classification trees are often not adopted in real applications demanding high accuracy, and other high-performing but obscure models are applied.

Interpreting rules

Classification rules are more commonly represented as propositional if–then rules in the form *IF* (*condition*) *THEN class*. The condition is a conjunction of tests on attributes, i.e. $A_1 = a_1, A_2 = a = 2, ..., A_k = a_k$ [156]. The condition is referred to as rule antecedent while the class is denoted as rule consequent. Multiple approaches have been proposed to derive classification rules [156]. We can firstly classify them into direct and indirect methods. Direct methods learn and extract rules directly from the data [156]. Sequential covering approaches belong to this class of methods. These approaches learn a single rule at the time, selecting it greedily. The heuristic search of the rules proceeds until the entire dataset is covered by the rules. PRISM [30], RIPPER [36], CN2 [35], FOIL [136], and AQ [110] algorithms are examples of sequential covering approaches. A drawback of this class of methods is that the heuristic and greedy process of rule discovery does not guarantee to identify the best set of rules.

Associative classifiers are another class of approaches that exploits concepts of frequent pattern mining and association rule mining to extract rules from the data. Since these algorithms will be exploited to capture local associations in Chapters 4

and 5, an overview of associative classification is reported in Section 3.3. Examples of associative classification algorithms are CPAR [171], CMAR [95], CBA [99], L^3 [17]. Associative classifiers tend to obtain better performance than heuristic sequential covering approaches [171], exploring the whole search space to learn optimal rules.

Indirect methods are a family of approaches that extract rules from classification models [156]. Several approaches extract rules from decision trees as the C4.5 rules algorithm [156]. Considering their high classification performance, several approaches have been proposed to extract rules from artificial neural networks [6]. Rule-based surrogate models can be trained on the predictions of a generic classifier to understand the global behavior of the model. We denote these approaches as interpretable global surrogate models and we review them in Section 2.1.2.

We can divide rules into decision lists and decision sets, according to the strategies for combining multiple rules and dealing with overlapping rules [114]. A decision list is a list of ordered rules. For classification tasks, the first rule (in the ordered list) matching the instance to classify is applied. A decision set is a set of unordered rules. In case the rules in the set are not mutually exclusive, a strategy for resolving conflicts when data are classified is applied to define the class label, like majority voting or quality measures as the confidence of the rule.

As decision trees, classification rules are widely adopted for their intrinsic interpretability. They provide both global and local interpretability. Classification rules are provided to the users in a textual representation, as an ordered (decision list) or unordered (decision set) list of rules. Hence, the users can read the entire list of rules to understand the global behavior. The rule or set of rules matching the instance to predict can then be inspected to understand the reason behind the individual prediction for the local interpretability. Each antecedent of classification rules is a set of unordered conjunction of test on attributes. Different than classification trees, rules have no hierarchical structure. Hence, we cannot directly derive from rules the relevance of attribute values to the predictions. However, as the alternative approach for decision trees, we can derive the relative importance of each attribute by considering the number of labeled instances matching a rule containing that attribute [59]. Considering the subjective nature of interpretability, user studies have been conducted to assess the comprehensibility of rules compared to other interpretable methods as decision trees [77]. The empirical results showed that the decision table representation of rules rather than the textual one and decision trees were considered by human users as easier to use. An advantage of the rule textual representation is their compactness [114]. On the other hand, the textual representation may hinder the comprehensibility from the global perspective and have a full picture of the model [59]. A global comprehension becomes increasingly difficult when the complexity of the rules increases. For classification rules, a common measure of complexity is the number of rules themselves [59]. Additionally, long rules are considered less interpretable [114].

Interpreting logistic regression

A linear classifier classifies the data based on a linear combination of the input features [76]. The classification is characterized by the dot product between a set of coefficients (or weights) β and the feature vector x:

$$y = g(\beta \cdot x) = g\left(\sum_j \beta_j x_j\right) = g(\beta_0 + \beta_1 x_1 + ... + \beta_d x_d) \tag{2.1}$$

where g is a function that maps the dot product to the desired output. The linear classifiers highly differ on how the function g and the coefficients β are defined and derived. An example of a simple linear classifier consists of adopting a threshold function as function g. In this case, for a binary classification problem, the linear classifier determines if $\beta \cdot x$ is greater than a threshold T and the output class is derived accordingly (e.g. the class is 1 if $\beta \cdot x \geq T$).

For the logistic regression, the function g is the logistic function [76]. For binary outcomes, the logistic regression applies the logistic function to model a binary dependent variable. The logistic function allows limiting the output of a linear equation between 0 and 1 that can be interpreted as the output probability. The logistic regression has the following form:

$$P(Y = 1|X = x) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + ... + \beta_d x_d}} \tag{2.2}$$

where Y is the response variable, x is an instance and β_i are the coefficients of the model. The logistic regression can be generalized for multiple (more than two) outcomes. The multinomial logistic regression can be modeled as a set of independent binary regressions in the form:

$$P(Y = K|X = x) = \frac{1}{1 + \sum_{l=1}^{K-1} e^{\beta_0 + \beta_l^T x}} \tag{2.3}$$

where K are the possible outcomes and β_l is the regression coefficient vector associated with a class l.

In linear models as linear regression for regression purposes, the weights can be straightforwardly used to interpret the results. The outcome of the regression is a linear function of the coefficients. Each coefficient β_i indicates the expected change in the outcome for a change of one unite of x_i when all the other attribute values are held fixed [114]. By applying the logistic function, we lose the linear influence of weight on the output probability. Two common ways to interpret logistic regression models are studying the effects of the attributes (i) on the logged odds and (ii) on the odds [122]. The first interpretation directly considers the coefficients β. The linear coefficients are in fact equal to the logarithm of the odds (logged odds):

$$log(odds) = log\left(\frac{P(Y = 1|X = x)}{1 - P(Y = 1|X = x)}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_d x_d \tag{2.4}$$

where the odds of an event is defined as the ratio of the probability of the event to occur and the probability that the event will not occur [76]. The effect of the attributes on the logged odds is hence linear and additive. A change of one unit of an attribute x_i corresponds to increasing the log odds by β_i, the value of the corresponding coefficient.

The effect on the logarithm of the odds may be hard to interpret and less intuitive due to the logarithm function [114]. As a second interpretation, we can consider the exponential of both terms of Equation 2.4 [122]:

$$odds = \frac{P(Y=1|X=x)}{1-P(Y=1|X=x)} = e^{\beta_0+\beta_1 x_1+...+\beta_d x_d} = e^{\beta_0} \times e^{\beta_1 x_1} \times ... \times e^{\beta_d x_d} \quad (2.5)$$

As a result, the effect on the odds is multiplicative and not additive as for the logged odds [122]. To an increase of the value of attribute x_i by one unit corresponds a change in the estimated odds by a factor e_i^{β}. A coefficient e_i^{β} of 1 does not influence the odds, $e_i^{\beta} > 1$ increases the odds while $e_i^{\beta} < 1$ decreases the odds [122].

Logistic regression models are widely adopted, especially in medical fields and social science for their interpretability. However, the understanding of the effect of weights maybe not intuitive for all users. The predictive performance of logistic regression classifiers is often lower than the one of more complex but more obscure models. As a result, high-performing but black-box models are often preferred.

Interpreting GAMs

Generalized additive models (GAMs) are generalized linear modelS (GLMs) [118] in which the outcome is modeled as the sum of arbitrary functions of each attribute [75]. GAMs are expressed in the following form:

$$g(y) = \beta_0 + \sum_{j=1}^{d} h_j(x_j) \quad (2.6)$$

where x is a data instance and y the target outcome, g is a link function and h_j's are generic nonparametric functions. Examples of commonly used function for h_j are step functions and splines. Example of used link functions g are the identity function for regression purposes or the logit function to model classification problems [75].

GAMs are widely adopted for their general high performance and ability to audit the model. GAMs are in fact considered interpretable since the relationship between each attribute and the final outcome can be analyzed by examining h_j [114]. The relations can be visualized by plotting each $h_j(x_j)$ as a function of x_j. The visualization reveals the contribution of a single attribute to the prediction. However, the interpretation of GAMs becomes challenging when the number of attributes increases. The number of attributes is a proxy of the model complexity.

The higher the attributes, the higher is the number of relations of $h_j(x_j)$ with the outcomes to inspect.

Interpreting k-nearest neighbors

K nearest neighbor algorithm (KNN) is an instance-based approach that assigns the outcome for a new instance based on its K most similar instances in the dataset [156]. For classification purposes, the class is usually assigned by majority voting among the K nearest instances, often weighted according to their distance. The interpretation of the behavior of a KNN model cannot be done at the global level as no model is actually trained. A way to interpret locally individual predictions of the KNN classifier is represented by the K neighbors themselves. The K closed instances represent an *explanation by example* of the reasons behind the prediction [114]. The inspection of closed examples reflects how humans often motivate or justify their actions by giving examples and by analogy [98]. However, the interpretation is challenging when the number of features increases, especially for tabular data [114]. Moreover, explanations by example do not provide an indication of the relevant attribute for the class assignment.

Interpreting Naive Bayes classifier

The Naive Bayes classifier is a probabilistic classifier that applies the theorem of Bayes with the naive assumption of independence between features [156]. Let x be the instance to classify among K classes. The class label y is derived as follows:

$$y = \arg \min_{c=1,\dots,K} p(c) \prod_{i=1}^{d} p(x_i|c) \tag{2.7}$$

where $p(c)$ is the class probability and $p(x_i|c)$ is the probability of feature x_i given the class c.

The Naive Bayes model predictions can be interpret by analyzing the probability associated with all the attributes [59]. From probabilities $p(x_i|c)$ and $p(c)$ learned at training time, we can derive $p(c|x_i)$ using the Bayes' Theorem and independence assumption, with $p(c|x_i) = p(c)p(x_i|c)/p(x_i)$. Each term $p(c|x_i)$ indicates the contribution of the feature x_i to the class assignment. Considering the relevance of understanding the reason behind predictions, the probabilistic interpretation is often applied in medical applications [92]. However, some users may not find it easy to deal with probabilities. To increase the understandability, solutions as the visualization approach in [20] have been proposed. However, the strong assumption of independence among features does not handle associations among attributes, reducing the performance of the approach and hindering its applicability.

Targeting interpretability by design

The recent widespread use of machine learning models in high-risk applications as health care, finance, and criminal justice is prompting researchers to consider understanding model behavior as a prerogative. A significant research direction consists of developing interpretable models by design, overcoming the belief of the necessary trade-off between accuracy and interpretability [142]. The goal is to design highly accurate and inherently interpretable models. The understandability of a models is considered earlier in the process of its definition and training. The problem of learning an interpretable classification model is then modeled as an optimization problem where interpretability criteria are directly included [142]. At an high level, the optimization problem is formulated as follows [144]:

$$\min_{f \in F} \sum_{i} Loss(f, x_i, y_i) + \text{InterpretabilityPenalty}(f), \quad \text{subject to Interpretability constraint}(f) \tag{2.8}$$

where f is the target model chosen among the class of functions F, x_i, y_i are the labeled data samples. Hence the objective is to minimize the loss function while minimizing the complexity of the classifier modeled as an interpretability penalty. The proposed approaches differentiate on the class of functions considered (as decision trees, rules, or linear models) and on the interpretability constraints. The latter highly depends on the target model. The interpretability criteria can be the number of nodes in decision trees or the number of rules in rule-based models.

We note that the class of optimized functions predominantly belongs to the existing class of transparent models that we have outlined in the previous sections. However, these novel approaches, despite often leveraging on existing transparent solutions, introduce a novel paradigm: consider interpretability needs directly in the optimization problem. The models are then optimized to achieve high performance while ensuring their interpretability, considering the human cognitive limits [111]. As a result, some of the discussed drawbacks are overcome. A detailed overview of the principles and challenges for interpretable machine learning models is presented in [144].

Works as [97] consider decision trees (DT) as class of function of interest. Equation 2.8 is then defined as [97, 144]:

$$\min_{f \in \text{set of DTs}} \sum_{i} Loss(f, x_i, y_i) + \lambda \cdot \text{Number of leaves}(f) \tag{2.9}$$

where λ is a regularization parameter that control the interpretability constraints. The number of leaves is a measure of the tree complexity. The greater the number of leaves, the more the tree is considered complex and so hard to interpret.

Several works leverage the high understandability of rules to derive interpretable and accurate models [7, 89, 113, 143]. The CORELS algorithm [7] derives rule lists

and considers as interpretability constraint the number of rules in the rule list as a measure of the size of the model and so of its complexity. Lakkaraju et al. propose interpretable decision sets, a framework to learn interpretable and accurate decision sets, i.e. list of unordered rules [89]. In the optimization problem, they consider multiple interpretability constraints. In their work, the interpretability of decision sets is modeled as a function of four criteria: size, length, cover, and overlap. The first two metrics refer to two common criteria to measure the complexity of rules. The size of the model is the number of rules in the decision set, while the length considers the length of rules (i.e. number of elements in the rules). The cover of a rule denotes its clarity and measures how many instances are covered by the rule. Finally, the overlap estimates the overlapping among rules. The lower is the overlapping, the more each rule covers distinct portions of the feature space and the higher is the interpretability. LIBRE is an ensemble approach that learns boolean rules, balancing accuracy and interpretability [113]. The approach combines bottom-up weak learners that generate few and compact rules, allowing their understandability. The results of the weak learners are combined via a union operation to improve the generalization of the model while preserving the interpretability of the final rule set [113].

Other works introduce interpretability constraints to optimize linear classification models as scoring systems [161]. Equation 2.8 can then be reformulated for scoring systems as follows [144]:

$$\min_{f \in \text{linear models}} \sum_{i} Loss(f, x_i, y_i) + \text{ nonzero terms } \beta \tag{2.10}$$

where f is a linear model and β_j are its coefficients. The number of nonzero coefficients is an indicator of the model complexity. The higher is the number of nonzero coefficients, the less the model is considered interpretable. Other interpretability constraints can be integrated by considering the coefficients as small integers or other domain-dependent constraints [144].

Explainable Boosting Machine (EBM) is an extension of GAMs to improve intelligibility while achieving high accuracy, based on the GA^{2}M algorithm [101, 120, 28, 102]. A limitation of GAMs is that they do not directly consider in their definition (Equation 2.6) the interaction among features. EBM includes pairwise interaction terms in the formulation of GAMs:

$$g(y) = \beta_0 + \sum_{j=1}^{d} h_j(x_j) + \sum h_i j(x_j, x_j) \tag{2.11}$$

To limit the number of explored pairs of attributes, it includes an efficient method to measure and rank the pairwise interaction [102]. As for GAMs, the contribution of each attribute to the prediction can be visualized and inspect by plotting h_j.

Moreover, EBMs allow the inspection of the pairwise interaction via a heatmap visualization.

2.1.2 Explanation method for enhancing interpretability

Despite the recent research efforts on designing interpretable classification models, black-box models are extensively adopted also in high-stake tasks. Given the importance of interpretability, many algorithms have been proposed for improving the understandability of classification models. A recent and detailed survey is provided in [67].

As illustrated in Section 1.2, we can identify two main dimensions on which algorithms for enhancing model explainability differ: the generality and the scope. The generality dimension divides the approaches into model-dependent solutions and model-agnostic ones. The scope refers to the target of explanations, global and local. In the following sections, we first present an overview of black-box models and model-dependent solutions. We then outline global model-agnostic approaches. Finally, we present a detailed analysis of model-agnostic solutions for local explanations. The discussion outlines the limits of existing solutions and the need for improvements that are addressed in this thesis.

Model dependent solutions

Model-dependent approaches are solutions to improve the model interpretability which are applicable only for specific classification models because they rely on how the considered classification method operates. The section outlines black-box models and the model-dependent solutions proposed to improve their interpretability.

Artificial neural networks. Artificial neural networks (ANN) are a class of techniques inspired by the biological neural system of the human brain [70]. ANNs are based on a set of weighted connected units, called artificial neurons organized in a network. Neurons are usually structured in connected layers divided into an input layer, one or more hidden layers, and an output layer. The output of each neuron is a function of the weighted sum of its inputs, usually applying a nonlinear function (the activation function). As a result, the ANN model a complex high-dimensional non-linear function. The mapping from the data instance to the final prediction is modeled by the collection of neurons organized in a sequence of multiple layers of the network and the output of each neuron depends on a usually non-linear function of the weighted sums of its inputs. Therefore, as humans, we cannot understand the complex interactions between the data instance and the final prediction [114].

Considering their great accuracy and advantages, a lot of work has been done in the direction of improving neural network interpretability. In particular, many techniques have been proposed for extracting explicit rules from ANN [149, 159]. An overview and taxonomy of techniques for rule extraction from ANNs are presented in [6]. Rules are considered as a good compromise since they can capture the complexity of the problem but at the same time, they are still understandable. Other solutions propose visualizing the neural network graph. As an example, Tzeng and Ma [160] propose to "open the black box" to reveal the relations between the inputs and the outputs of an ANN.

Deep learning models (DL) are increasingly adopted and approaches to explain DL predictions have been proposed for multiple tasks as for sentiment analysis and classification on textual data [11, 163], entity resolution [43] and image recognition [119, 151, 162, 176]. For image recognition, many techniques explaining predictions for individual images [119, 151, 162, 176]. Several techniques provide explanations in form of saliency maps [151], showing how each pixel of a particular image is important for class prediction. Fong and Vedaldi apply a removal-based approach for image classification, approximating the elimination of parts of an image with meaningful perturbations [56]. EBAnO defines interpretable features of the input images through hypercolumn representation and cluster analysis and exploits perturbation to measure the relevance of the extracted interpretable representation [162]. These techniques are specifically designed for deep learning model and deal with unstructured data as text and images. Differently, the techniques proposed in this thesis are suitable for structured data.

Random Forest. Random Forests is an ensemble learning method that learns multiple de-correlated decision trees using bagging and random subsampling methods [25, 75]. For classification tasks, the prediction is assigned by majority voting among the trees. Hence, the final outcome is the class that occurs more commonly in trees.

A random forest consists of a large number of trees. As a result, users cannot conveniently inspect the resulting model and understand its internal process. One way of getting insights into a random forest model was proposed by Breiman himself, in the paper in which he describes his Random Forests algorithm, based on the computation of the feature importance [25]. The provided explanation has a global scope since it evaluates the relevance of each feature for the model. The approach leverages out-of-bag estimates. Hence, it is applicable for approaches exploiting bagging, as random forests [25] and boosted decision trees [75, 60]. Several approaches extract rules from trained random forest models [42, 72, 108].

Support Vector Machines. Support Vector Machines (SVM) are a class of discriminative classifiers that finds the hyperplane or set of hyperplanes that better separate the data by maximizing the margin, i.e. the separation between the classes.

SVMs can handle classes with complex non-linear decision boundaries. Support vector classifiers often achieve high predictive performance. However, their results are difficult to interpret.

One possible way to interpret model is through its support vectors, the subset of observation that lies closest to the decision boundaries [78]. Hence, support vectors provide an *explanation by example*. However, support vectors are a reduction of the number of instances to consider [78]. Hence, they do not directly explain which are the most relevant factors that determine the prediction. Another possibility is to visualize the hyperplanes of the SVM model in the attribute spaces. However, this solution is appropriate only if for two or three-dimensional feature space [78].

Nomograms have been applied to explain the individual decisions of Support Vector Machines (SVM) with linear kernels [78]. Nomograms are a visualization technique that enables visualizing the contribution of feature values. Nomograms have also been applied for Naive Bayes [117] and Logistic regression models [103]. As a result, the behavior of these different classifiers for the same classification problem can be compared. This represents an advantage of the use of nomograms for explaining the model. The comparison allows choosing the most suitable model not only based on performance but also on the model behavior. However, nomograms do not properly handle redundant and highly correlated attributes. Rule extraction techniques have also been developed for the interpretation of SVMs [61, 107, 121]. A complete review can be found in [16]. Other works focus on visualizing SVMs, as the projection technique proposed by Caragea et al. [27] and the open-box visual analysis of Ma et al. [106].

The presented methods address some specific classification algorithms. The explanations of how the models work are presented by means of rules, visualizations, nomograms, global feature importance. Thus, the different explanation approach and form of presentation hinder the comparison among the explanations of different classification techniques in terms of model interpretability. Through model comparison, it is possible to analyze the different behaviors of classifiers, understand what the different models have learned, and, possibly, choose the best model for a specific purpose. Several model-dependent techniques are based on the extraction of rules from the trained model. Through the comparison of rules, it is possible to compare the behavior of different models. However, the (many) extracted rules can be very complex. Thus, model understanding and comparison may become difficult anyway.

Model agnostic solutions

Model agnostic or model-independent solutions derive post-hoc explanations of the model behavior by treating the original machine model as a black box. These techniques are therefore applicable to any classifier, without making any assumption

on the model behavior.

Model agnostic solutions introduce flexibility in the choice of the classifiers, representations and allow model comparison. Ribeiro et al. identify the following advantageous properties of model agnostic solutions [140].

- **Model flexibility.** Model agnostic solutions allow uncoupling model performance and their interpretability. Often, in high-risk tasks, transparent models are preferred over obscure ones, despite the potential lower accuracy. With model-independent solutions, users can apply the best performing classifier for the problem under study. They can then apply post-hoc explainability techniques to inspect and understand the model behavior.

- **Explanation flexibility.** For transparent models, the form of provided explanations depends on the model themselves. For example, we have the tree graphical representation and decision path for decision rules, the set or list of rules for rule-based techniques, and the weights for linear models. Similarly, for model-dependent post-hoc solutions, the explanation type depends on the technique itself. For example, we have feature importance vector for the random forest [25], nomograms [78, 117] or visualization of neural network graph [160]. A first drawback of the constraints on the form of the explanations is that the degree of understandability is subjective. The level of comprehensibility varies for different users [59]. Some users may be familiar with weight or probabilities, other users may prefer to inspect rules while others inspect the importance of each feature to the prediction. Moreover, different forms of explanation hinder the comparison of the reason behind predictions. With model-agnostic solutions, we can apply multiple explanation techniques and derive multiple forms of explanations. As a result, data analysts and end-users can use the most suitable form of explanation for their application and compare explanations.

- **Representation flexibility.** Model agnostic solutions allow decoupling the representation of the input feature for training the model and feature for deriving explanations. We can hence have explanations in terms of words or super-pixels representing a portion of an image for models trained on word embeddings and tensors with three color channels per pixels respectively [140].

- **Lower switch cost.** Model agnostic approaches enable decoupling the model and the explanation representation. The applied model can be changed over time while keeping fixed the form of explanation representation.

- **Model comparison**. The comparison between models is usually performed with respect to final quality indicators of performance. Comparing models in terms of their behavior allows a more comprehensive evaluation and to

decide which model to trust. However, the obscure nature of classifiers or the multitude of forms of explanations hinder the model comparison. With model-agnostic post-hoc techniques, users can compare the behavior of multiple models using a uniform form of explanation.

Model agnostic solutions can have a global or local scope. In the following, we firstly outline model agnostic solutions for global explanations. We then provide an in-depth overview of model agnostic techniques with a local scope to explain individual predictions.

Global solutions

Global interpretability techniques aim at explaining how a model globally works. Global explanation methods allow understanding the entire relationship between inputs and class labels learned by a model [69].

Interpretable global surrogate models. Some solutions try to explain the original model globally by deriving a transparent model that mimics its behavior.

The algorithm TREPAN [38] approximates a generic model f by learning a classification tree (DT) on the predictions of f. The approach aims at optimizing the mimicking ability of the surrogate tree while preserving its comprehensibility. The fidelity of the tree to the original black-box model is measured by estimating the percentage of classification agreement of the proxy model to the original one. The comprehensibility of the model is instead managed by controlling the number of nodes as a proxy measure of tree understandability (Section 2.1.1).

The concern with this solution is that a simple model as a decision tree is used as surrogate global models. It can be argued that the interpretable but simple decision tree may not be able to mimic the complexity of non-linear and complex models as artificial neural networks. The doubts consider the faithfulness of the DT. In the case of a completely faithful surrogate model, we may argue that the transparent model could be applied in the first place, avoiding the drawbacks of obscure models. Moreover, the resulting decision tree can be so complex and large to hamper its global understanding.

The approach of learning a surrogate model on the basis of black-box predictions can be generally applied by using a generic transparent model as surrogates. However, alike concerns may arise. The global explanations could be a too severe approximation of the original model and fail to faithfully mimic complex models.

Partial dependence plots. Partial dependence plots (PDPs) provide the global, average prediction behavior of a variable or a (small) set of variables, by visualizing their marginal effect on the class probability [76]. In classification tasks, PDPs

show the dependence of a set of input features on prediction probabilities. Let f be a generic classification model and d the number of attributes. The dependence of a subset of feature S on predictions can be estimated by averaging in the dataset its marginal effect as follows:

$$\frac{1}{n}\sum_{i=1}^{n} f(x_S, x_{\hat{S}}^{(i)}) \tag{2.12}$$

where n is the number of instances in dataset, $\hat{s}$ are the other $d - S$ features in the dataset and $x_{\hat{S}}$ are actual feature values from the dataset (i.e. from the i-th instance). Intuitively, the partial dependence is the average prediction probabilities when the values of the features in the considered subsets are applied while the other features remain unchanged [114]. The partial dependence functions are plotted as a function of the values of x_S. As a result, subsets of only one feature correspond to two-dimensional representations while subsets of two features correspond to three-dimensional plots. Considering the limitations of the visualization and human perception, the realistic maximum number of features that are considered in partial dependence functions is 2 [114]. PDPs considering one feature value at a time show how prediction globally depends on the single input. Therefore, they have the advantage of being very intuitive. However, PDPs have the strong limitation of assuming independence among features. Moreover, partial dependence plots represent the average effect of a feature on predictions [114]. Hence, they may obfuscate heterogeneous relationships, as positive and negative associations. Finally, the visual inspection of the individual feature relation with predictions may become difficult for the user when the number of features increases [114]. In this case, the analysis requires observing multiple graphical representations. The user may find it challenging to grasp the relations among features and how the model globally works.

Permutation feature importance. Permutation feature importance is a generalization of the feature importance firstly proposed by Breiman for Random Forest classifiers [25]. The approach measures the change in scores (as the accuracy, f1-score, out-of-bag estimate performance scores) when features are permuted [114]. A feature is considered relevant if a change in the score is observed to its permutation. Permutation feature importance belongs to the class of explanation techniques defined removal-based explanations [37]. The permutation operation can in fact be seen as a way to approximate the removal of a feature and observing how the model behavior changes. Permutation feature importance is intuitive and provides a global insight into the model behavior. However, being an average of the model behavior, they do not provide the reasons behind individual predictions.

The variety of different approaches proposed for providing global explanations

of the model behavior are affected by theoretical concerns about the ability of simple explanation models to fully mirror the original and more complex model [69]. For global surrogate solutions [38], doubts arise about their faithfulness and trustworthiness. Furthermore, explanations of an entire model could be too complex to be really understood by humans. Finally, global models may not provide the reasons behind individual decisions.

Local solutions

Local explanation approaches, also refer to as *outcome explanations* [67] or *prediction explanations*, aim at explaining the reasons behind individual predictions. Model-agnostic local explanations reveal why a particular decision is made by a generic classifier rather than its whole logic.

The explanation of single predictions is gaining interest in the research community [68, 104, 138] because of its straightforward interpretability. The explanation of an entire classification model may become excessively complex and difficult to understand. Moreover, concerns arise on the faithfulness of global explanation on capturing the global behavior, especially for complex classifiers. Furthermore, in some cases users, do not need a global comprehension of the model. As an example, in recommendation systems, the reasons why a particular product has been suggested make the model more useful and likely to be trusted [138]. In addition, it is not always possible to provide a global explanation also for legal and contractual reasons. Some algorithms are proprietary and companies do not fully divulge their inner process.

Considering the advantages of local solutions, in Chapter 4, we propose LACE [129], a novel model agnostic approach for explaining individual predictions. In the following, we provide an in-depth review of existing local explanation approaches. Along the lines, we outline the need for improvement and main limitations that are addressed by our approach [129].

Local explanation approaches differ in the way the reason behind an individual prediction is presented. Examples of prevalent form of explanations are local feature importance vector [104, 138, 153, 154, 155], local rules [68, 80, 139], visualization techniques [62] and counterfactual explanations [41, 68, 86, 116, 166].

Feature importance explanations. Several approaches produce local explanations in form of feature importance vectors [104, 138, 153, 154, 155]. A feature importance vector is a local explanation of the prediction of an individual instance x in the form $t(x) = t_1, t_2, \cdots, t_d$ where d is the number of dimensions of the interpretable feature space. As discussed in the section introduction, model-agnostic solutions have the advantage of representation flexibility. Hence, these approaches disentangle the feature representation of the model itself and the one for deriving explanations. Each term t_i represents the relevance of the i-th feature to the prediction for instance x.

Ribeiro et al. introduce LIME, a model-agnostic method for explaining individual predictions by learning an interpretable model in the locality of the prediction to be explained [138]. The local model is a linear model that mimics locally the behavior of the original model. Hence, differently than global surrogate interpretable models that aim to capture the global behavior of the model, LIME learns an interpretable model only in the locality of the instance to derive the relevant feature for the individual label assignment. The instance locality is derived by generating perturbed samples of the instance and the locality is controlled by weighting the samples by their proximity. LIME optimizes the fidelity of the local surrogate model to the original one while preserving its understandability. Since the local model applied is a linear one, the number of non-zero weights of the linear model and so of the features used are a proxy of the model complexity. A drawback of LIME is that the locality has a major impact on the quality of explanations and different approaches have been proposed to derive high-quality neighborhoods [81, 91]. Moreover, the approach may suffer from instability [4, 114]. Close instances may have different explanations and multiple and unstable explanations may be derived for the same instance due to the random sampling neighborhood generation process [114]. Differently, we propose a novel approach that exploits the *actual* neighborhood of the instance to train a rule-based surrogate model. The local model captures the relevant association of attribute values with the class label. Our method leverages an automatic approach to select the appropriate locality scope to obtain high-quality explanations.

Several works, referred to as removal-based explanations [37], study how a prediction changes if parts of the input components are omitted [94, 104, 105, 141, 153, 154, 155]. Removal-based explanation approaches generally differ on (i) how the feature removal is approximated, (ii) the target model behavior of interest, and (iii) on how the contribution of the feature omission is summarized in the feature importance vector. An in-depth overview of removal-based explanations can be found in [37].

Lemaire et al. [94] and Robnik-Šikonja and Kononenko [141] analyze the relevance of each attribute value for the prediction in the case of tabular data by omitting one attribute value at a time. Štrumbelj et al. improve this approach by proposing IME, an individual explanation approach that considers also the omission of more attribute values at a time [155]. In this way, they also consider attribute interactions. The contributions of subsets of attribute values are aggregated using as summarization function the Shapley value [150]. The Shapley value is a concept from coalition game theory to assign a score to the players who cooperate to achieve a certain total score [150]. In the context of prediction explanations, the attribute values of the instance to explain are the players and the prediction probability is the score. However, the exact estimation requires the computation of the omission effect for the power set of the attributes. Hence, the method is affected by an

exponential time complexity. A solution for overcoming this problem is based on a sampling-based approximation [153]. The sampling is quasi-random and adaptive and it is based on a greedy approach, considering data characteristics, as the feature variance.

SHAP is a game-theoretic approach that decomposes individual predictions using the notion of Shapley value [104]. The authors propose two practical approximations for estimating Shapley values, KernelSHAP and TreeSHAP. KernelSHAP [104] is an approximation approach based on local surrogate models. The estimation is based on weighted linear regression models. Despite the approximation, KernelSHAP can be slow [114]. A faster but model-specific alternative is TreeSHAP [105]. It is an efficient estimation approach designed for tree-based models as random forest and gradient boosted trees. However, the feature attribution scores may be unintuitive due to its relying on conditional expected prediction [114]

The mentioned solutions [94, 104, 105, 141, 153, 154, 155] summarize the information on attribute interaction in one single contribution for each attribute value. Hence, the information on interaction relevance is lost. Our approach for individual explanations LACE [129], illustrated in Chapter 4, separately quantifies feature importance both for individual attribute values and relevant subset of attribute values. Our method overcomes the problem of exponential time complexity by exploiting local properties of the original model to be explained. A rule-based local surrogate model is trained on the locality of the prediction to the relevant associations of attribute values and their relevance is estimated via omission.

Rule-based explanations. Several works extract explanations in form of rules [68, 80, 139]. Rule explanations are a set of rules in the form $\{A_i = a_i, ..., A_k = a_k\} \rightarrow c$ where each term $A_j = a_j$ is an attribute-value pair and c is the class. They capture the relevant association of multiple attribute values with the class label. Anchor [139] and LORE [68] approaches both exploit the concept of locality for obtaining local explanations. Anchor derives rules that "anchor" the prediction locally by solving a multi-armed bandit problem [139]. An anchor rule indicates that changes in the other feature not belonging to the rule do not alter the prediction [139]. LORE derives rules by learning a decision tree as a local surrogate model. For the two approaches, the locality is studied by generating perturbed instances randomly in the neighborhood of the instance to be explained [139] or through a genetic algorithm [68]. The returned local rules provide only a qualitative insight into the reason for a prediction. In our work, described in Chapter 4, we focus our work on detecting the different roles played both (i) separately by single attributes and (ii) jointly by correlated attributes derived from local rules and comparing their relevance by means of appropriate metrics. Our approach also exploits rules to provide local prediction interpretation, but it extends with respect to previous work by considering the role of attribute correlation in the explanation. Hence,

our explanations are capable to capture and quantify the interaction effect when multiple attributes influence the individual prediction. Furthermore, we capture the locality of a prediction by means of its actual neighborhood instead of generating perturbed instances randomly [138, 139] or through a genetic algorithm [68]. Hence, only existing relationships among attribute values are studied.

Visualization-based explanations. Visualization techniques provide insight into the model prediction behavior by means of graphical representation [69]. Individual Conditional Expectation (ICE) plots are a novel adaptation of partial dependence plots (PDPs) for the local inspection of the model inner working [62]. While PDPs capture the global behavior by averaging the effect of each feature on prediction, ICE depicts how the model behaves for a single observation [62]. An ICE visualization shows the dependence of a feature on the prediction for each individual instance separately. Hence, we have one line per instance that indicates how the prediction changes as the value of the feature changes. Despite their intuitiveness, ICE, as PDPs, require users to inspect multiple visualizations, one per feature. Moreover, the visualization of individual dependencies in the same plot may hinder the understanding of the reason behind individual predictions, helping more in providing an indication of the average partial relationship. Our work LACE, described in Chapter 4, provides the relevance of each input feature and relevant subsets of input features to the prediction of an individual instance using a bar plot representation. The bar plot visualization is very simple and intuitive, helping users in focusing on the relevant relation of input features and the prediction outcome.

Example-based explanations. Explanation by examples consists in providing a data instance or a set of data instances to explain the model behavior [114]. Example-based explanations can be provided in multiple forms as prototypes, counterfactual explanations, and adversarial examples. Prototypes are representative instances of the individual prediction to explain or representative data instances of all the data for a global understanding of the model behavior [114]. Explanation by data examples reflect how we as human think and justify our actions by examples and analogy [98]. However, the model behavior from examples may be difficult to interpret in the case of tabular data, especially when the number of features increases [114]. Several approaches derive counterfactuals to provide insight into the model behavior [41, 68, 86, 116, 166]. Counterfactual explanations of an individual prediction describe the slightest change to the feature values of the instance that changes the prediction outcome [114]. Counterfactual instances represent the observations closed to the instance of interest that have a different outcome [164]. An overview of approaches for counterfactual explanations is presented in [164]. The notion of counterfactual has been recently explored also for graph classifiers to produce counterfactual graphs which are structurally similar to the explained

graphs but are assigned to a different class [1]. An interesting case of counterfactual examples is represented by adversarial examples. Adversarial examples are counterfactual instances where the small changes are applied to fool machine learning models rather than interpret them [114].

Counterfactual explanations are intuitive since they indicate changes that change the prediction. However, they do not directly provide the reasons behind the prediction but the reason that enables it to change it. Moreover, for a given instance, we may have multiple counterfactual explanations. Therefore, their interpretation can be challenging for users.

2.2 Understanding the model behavior in subgroups

The outlined approaches characterize the behavior of the model at the individual or at the global level. A relevant step in the explanation of the outcome of machine learning algorithms is the characterization of the model behavior in data subgroups. Subgroup analysis of classification models provides an understanding of the model at the data subset granularity. The model may show a different, and potentially anomalous, behavior across data subgroups. The identification of peculiar behaviors of data subgroups finds important applications in the KDD pipeline, such as model validation and testing [34], error analysis [85, 169] and the evaluation of model fairness [26]. In particular, societal bias [18] is becoming a growing concern and researchers are increasingly working on measuring and ensuring fairness in machine learning.

In Chapters 6 and 7, we propose DivExplorer [126, 124], a novel model-agnostic approach to characterize the behavior of classification models in data subgroups. Specifically, we aim to identify and describe the data subsets that show a peculiar and different behavior than the overall dataset.

In the following, we outline the related work to DivExplorer [126] for the identification of peculiar behaviors in data subgroups. Existing techniques for analyzing data subgroups include both supervised and unsupervised techniques. The former refers to user-defined subgroups and human-in-the-loop identification approaches. The latter are a class of approaches for the automatic identification of the subgroups of interest. Our approach belongs to this latter category.

2.2.1 Supervised techniques for subgroup analysis

Data grouping solutions often rely on domain experts to identify the relevant attributes or subgroups of interest.

Classification performance. Several approaches audit the behavior of classification models on user-defined data subsets evaluating whether differences in classification performance occur [5, 19, 85]. In the TensorFlow Model Analysis (TFMA) library [5, 19], the users specify the input features on which to partition the data used for classification performance evaluation. The library provides an overview of the statistics and classification performance metrics associated with the user-specified slices of data. MLCube [85] is an interactive and explorative visualization technique that estimates aggregate statistics and performance metrics over subgroups defined by users. The slices can be identified using multiple conditions on features and further interactively inspected using drill-down operations.

Fairness. Many efforts have been devoted to detecting and mitigating bias in classification tasks. For model fairness, the subgroup diagnosis concentrates on evaluating if a different treatment or performance occur on groups determined by some sensitive or protected attributes (e.g., gender, ethnicity, sexual orientation, degree of disability) [21, 146]. Several works [50, 57, 87, 115] consider fairness for an intersection of multiple sensitive attributes, known as intersectional fairness.

Recently, researchers focused their attention on fairness in other tasks beyond classification as in rankings [174]. Rankings are widely adopted for multiple applications as job recruiting, school admissions, potential friend or partner searches. Fairness concerns arise since ranking systems often score and rank individuals. It is hence urgent to detect if different behaviors of a ranking or scoring system occur across data subgroups. Different works propose measures and mechanisms to audit ranking outputs and mitigate bias over protected groups [29, 170, 173, 175]. Protected groups can be pre-defined or defined over a single or an intersection of sensitive attributes. When the number of sensitive attributes increases the fairness evaluation of all subgroups defined over protected attributes may be hard since their number is exponential. To address this issue, a recent work proposes an automatic approach to partition into groups over the sensitive attributes to obtain the most unfair partitioning with respect to the scoring function [52].

The analysis of group fairness generally assumes that the sensitive attributes that define protected groups are known or specified a priori. However, the enumeration of all the attributes that characterize a potentially biased group can be incomplete and not exhaustive due to proxy phenomena. Other attributes, not directly evident, may be proxies of sensitive attributes [18], as the residential neighborhood can be a proxy attribute for the race.

The identification of problematic attributes might not be straightforward. Supervised approaches largely depend on the domain expertise of users and their knowledge of the data and the classification problem. Requiring human intervention, the process of subgroup selection can be time-consuming. Moreover, it may

prevent identifying previously unknown subgroups of interest or unexpected behaviors.

2.2.2 Unsupervised techniques for subgroup analysis

Unsupervised techniques automatically identify interesting data subgroups with peculiar model behavior. Recently, several approaches have been proposed to directly detect subsets of data showing different classification performance [26, 33, 145, 169]. These works are close to our approach DivExplorer.

Slice Finder. Slice Finder [33, 34] is an interactive framework that automatically identifies large data slices in which the model performs poorly, defined as "problematic" slices. The approach is model agnostic and can be used to detect problematic data slices for a generic classification model. The behavior of the classifier on data slices is compared with their counterparts (i.e. the reciprocals). Slice Finder identifies top-k slices of interest using a top-down lattice search in a breadth-first traversal. The lattice search is controlled by statistical techniques that measure the significance and magnitude of performance discrepancy on subgroups. To identify large and interpretable subgroups, the breadth-first traversal does not proceed if the considered data group is already statistically significant and the model performs sufficiently poorly.

However, the metrics commonly used for assessing model performance on subgroups as error rate, accuracy, false positive, and negative rates are typically non-monotone. As a result, the grade of discrepancy of a group provides no indication on the behavior of its super/sub-groups. In this case, subgroups may have a higher or lower discrepancy given by phenomena of compensation effects can be observed.

We propose a more thorough exploration of the lattice, considering all data slices, identified by itemsets having support greater than a given threshold. Frequency constraints allow us to identify data subsets (i.e., slices) large enough to be of interest. To improve interpretability, concepts of coalition game theory are exploited to characterize subgroup divergence.

SliceLine. SliceLine [145] identifies the top k data slices with adequate representation for which a model performs significantly worse than the entire dataset. As our work DivExplorer, the approach leverages the monotonicity properties of the subgroups. Moreover, the search of data slices is optimized for a specific error metric. As a result, it also considers monotonicity properties of errors and resulting scores of problematicness for effective pruning. The approach leverages a linear-algebra-based enumeration algorithm, which, together with pruning, allows for parallelization and scalability. In our work, we aim at providing an in-depth understanding of the model behavior at the subgroup level. First, we provide the divergence scores for all the subgroups with adequate representation. Hence, we are

able to identify not only the most divergent subgroups with worse performance (i.e. the top k) but also the subgroups in which the model behaves equal or better than the average. We then use the notion of Shapley value to understand the attribute values characterizing a subgroup that mostly contribute to the peculiar subgroup behavior of the model.

FairVIS. Cabrera et al. in [26] propose FairVIS to audit the fairness of machine learning models in data subgroups. FairVIS [26] is a visual analytics system to discover intersectional bias in classifiers using a model-agnostic approach. The subgroups are automatically generated by applying clustering techniques. To obtain an interpretable representation of the subgroups, FairVIS uses feature entropy to derive the dominant feature of each identified group having significant statistical similarity. The performance metrics are evaluated on the clusters and users can interactively perform drill-down operations on specific subgroups. Differently from FairVIS, DivExplorer identify data subgroups directly by slicing attribute domains. As a result, the identified subgroups are already interpretable and no additional post-hoc techniques to derive interpretable representation as feature entropy are needed. Moreover, the notion of Shapley value and interactive visualizations allow us to further improve the transparency of slices characterized by multiple attributes.

Errudite. Data grouping is exploited in Errudite [169] for NLP error analysis to improve and understand NLP models. A domain-specific language is proposed to systematically group instances. Moreover, Errudite enables systematic counterfactual analysis to understand why the model fails for certain data subgroups. Despite the system suggestions and guidance to formulate group queries, data grouping highly depends on users. Differently from [169], DivExplorer deal with structured data and we automatically slice the dataset with respect to the actual attribute domains.

There are other lines of work related to the problem of identifying interesting behavior of classification models in data subgroups. These approaches, rather than characterizing the model behavior in subgroups, identify and characterized data subgroups themselves. Subgroup identification techniques can be considered as building blocks of approaches for understanding the model behavior in subgroups. Several of the introduced approaches so far and DivExplorer itself leverage data exploration techniques. For example, FairVIS [26] exploits clustering analysis, SliceFinder [34] uses lattice search while our approach integrated frequent pattern mining algorithms.

Exploratory data analysis. Clustering analysis is a class of unsupervised techniques that groups data instances so that instances in a group are similar and

different from the ones in other groups [156]. Multiple algorithms have been proposed to derived data clusters, such as centroid models as K-means [100], density models as DBSCAN [53] and OPTICS [9], unsupervised neural networks, and connectivity models as hierarchical clustering. The target of clustering is to group similar data. Differently, we aim to characterize the behavior of classification models, highlighting peculiar behavior as classification performance or labeled class distributions. Moreover, clustering algorithms generally do not provide directly the reasons behind the grouping, as the relevant features of each clustered data or each cluster. Therefore, post-hoc techniques to characterize the clusters should be applied to interpret the results. Clustering techniques can be a step in the pipeline to understand the model behavior in subgroups. As mentioned, FairVIS [26] firstly applies clustering techniques as K-means, DBSCAN, and OPTICS to group data and exploit feature entropy as a post-hoc technique to derive the most dominant feature of the clusters. In [33], clustering is considered as an alternative solution to the lattice search to identify problematic subgroups.

Frequent pattern mining (FPM) is a data exploration technique that finds patterns that occur frequently in a dataset. An overview of FPM techniques is reported in Section 3.2. FPM algorithms as FP-growth [70], Apriori [3] prune the search space over the lattice of attribute values using the anti-monotonicity property of frequency. The data slices identified are already interpretable since FPM techniques slice in the attribute domain. We leverage these techniques in our approach DivExplorer (Chapters 6 and 7) to derive frequent subgroups where a classification model shows peculiar behaviors different than the overall one.

Online Analytical Processing (OLAP) tools allow users to select, extract and inspect multidimensional data [63]. Slicing is a key OLAP operation and enables users to slice the data by specific dimensions. Techniques as Smart Drilldown [84] aim to automatically provide the most interesting data slices. Smart Drilldown is a novel operator which extends the drill-down operator and it derives the top-k most interesting slices in form of rules. While in OLAP systems, the targets of slicing are generally the count or sum aggregates of numerical measures, we focus on characterizing the model behavior.

Local and global explanations. As reviewed in Section 2.1.2 techniques as LACE [129] (described in Chapter 4), LIME [138] and SHAP [104] allows to understand the reasons behind individual model predictions [67]. Differently, explanations at the subgroup scope focus on studying the disparate statistical behavior of a classifier on data subgroups. Multiple individual explanations could be potentially aggregated to derive understanding at the subgroup level. However, it would require the intervention of users to group explanation in a supervised scenario or additional post-hoc data grouping as clustering in an unsupervised one. Rather than understanding individual predictions, DivExplorer provides an analysis of the entire dataset by characterizing the subgroups in which a different behavior

than overall is observed.

As the works [153, 155] and SHAP [104], DivExplorer exploits the concept of Shapley value from coalition game theory. In SHAP, the Shapley value is used to compute the contribution of each feature value to the prediction for a single instance. In our work, the notion of Shapley value is adopted to characterize the contribution of each attribute value to the subgroup different behavior, hence characterizing a subgroup rather than a specific instance. Furthermore, we generalize the notion of Shapley value to estimate the global contribution to divergent behavior of each attribute value over the entire dataset.

Forms of global explanations as rules and decision tree can provide insights into the behavior of the classifier for subgroups of data. Individual rules and paths indicate the reason behind the predictions of the instances they represent. However, these form of explanations is limited to transparent rule and tree-based models as discussed in Section 2.1.1. On the other hand, as outlined in Section in 2.1.2, model agnostic solutions that extract explanations in form of rules or trees may fail to fully mimic the global behavior of the model. In Chapter 6, we propose a model agnostic solution to directly study the different behavior of a generic classification model in data subgroups.

Data coverage. The work of Asudeh et al. [14] studies the lack of adequate coverage in a dataset. Inadequate representation may cause errors in predictions and undesirable outcomes such as algorithmic racism. Uncovered patterns are introduced to identify attribute space regions not adequately covered by the data. Data subgroups, as in our work DivExplorer, are identified by attribute value combinations. However, [14] addresses a different problem, because the target attributes and classification outcome are not considered in the coverage problem, while we explicitly consider classification performance and identify subgroups in which a classification model performs differently with respect to the overall population. Furthermore, differently from [14] which considers underrepresented groups, we consider subgroups with adequate representation selected by a frequency threshold.

Chapter 3

Background

This chapter introduces preliminary definitions and concepts that are used throughout the thesis. We firstly characterize the data structure under analysis and recall key definitions of machine learning (Section 3.1). Then, we review basic concepts of frequent pattern mining. We provide the definitions of pattern and its related terminologies (Section 3.2). We illustrate the key features of algorithms for frequent pattern extraction. Finally, we present an overview of associative classifiers (Section 3.3).

3.1 Preliminary definitions

We firstly provide the definition of basic concepts exploited in the thesis.

3.1.1 Dataset and Itemsets

We consider a dataset D, whose schema is given by d distinct attributes and a class label. Let A be the set of attributes over which D is defined, with A=$\{$ $A_1, A_2, ..., A_d$ $\}$ and $|A| = d$. The domain of each attribute $A_i \in A$, defined as $\mathcal{D}_{A_i}$, can be discrete or continuous. In this thesis, we will focus on classification problems for structured data with a categorical class label. An *instance* $x \in D$ is a set of attribute-value pairs such data $x(A_i) \in \mathcal{D}_{A_i}$ for each $A_i \in A$. For each instance in the training set, the class label y is known.

A *classifier* is a relation $f_c(x)$ that maps an instance x to a class label y. Given a set of n labeled training instances of the form $\{(x_1, y_1), ..., (x_N,\ y_N)\}$, a classifier algorithm learn a function $f_c : X \rightarrow Y$, where X is the attribute space with schema A and Y is the target label space. Many classifiers can be represented as a scoring function $f : X \rightarrow \mathbb{R}$. Given an instance, they predict a probability score or confidence score associated with each class label. The function f_c is then derived from f as the y value that maximizes f: $f_c(x) = \text{argmax}_y\, f(x, y)$, where the argmax operator returns the class that maximizes $f(x, y)$.

Definition 3.1.1. Item An *item* α is an attribute equality $A_i = c$ for $A_i \in A$ and $c \in \mathcal{D}_{A_i}$.

We say that an instance x is covered by the item $\alpha : A_i = c$, referred as $x \models \alpha$, if $x(A_i) = c$. Thus, covering is the equivalent of the logical notion of satisfaction. We denote with $\text{attr}(\alpha)$ the attribute to which an item refers, so that $\text{attr}(A_i = c) = A_i$.

Definition 3.1.2. Itemset (pattern) An *itemset* is a set of items $I = \{\alpha_1, \ldots, \alpha_k\}$ that refer each to a distinct attribute, i.e., such that $\text{attr}(\alpha_i) \neq \text{attr}(\alpha_j)$ for all $1 \leq i < j \leq k$. An itemset $I = \{\alpha_1, \ldots, \alpha_k\}$ can be represented as the conjunction $\alpha_1 \wedge \cdots \wedge \alpha_k$ of its items.

In the thesis, we will use the terms *itemset* and *pattern* interchangeably.

An instance $x \in D$ is covered by an itemset I, written $x \models I$, if $x \models \alpha_i$ for $1 \leq i \leq k$. If $x \models I$, we will say that the instance x satisfies (or matches) I.

The *length* of an itemset is the number of items contained in it. The length can range between 0, for the empty itemset, and d, the number of attributes. We refer to l-itemsets as the itemset with length l. We denote by $\text{attr}(I) = \bigcup_{\alpha \in I} \text{attr}(\alpha)$ the set of attributes included in an itemset. For a subset of attributes $B \subseteq A$, we write $\mathcal{I}_B = \{I \mid \text{attr}(I) = B\}$ for the itemsets over attributes B. In particular, the set $\mathcal{I}_A$ consists of the itemsets that contain all attributes of our dataset.

The *support set* $D(I) = \{x \in D \mid x \models I\}$ of an itemset I consists of the instances that satisfy I. The *support count* refers to the number of instances that contains a particular itemset. The support count of an itemset I is the cardinality of $D(I)$, given by $sup\_count(I) =\mid D(I) \mid$.

Definition 3.1.3. Support The *support* of I is the fraction of instance in the dataset that satisfies I and it is given by $sup(I) = \frac{|D(I)|}{|D|}$.

Definition 3.1.4. Frequent pattern. An itemset (or pattern) I is defined *frequent* if $sup(I)$ is greater a given threshold s.

The support threshold s is also referred to as *minimum support*.

The algorithms for extracting frequent patterns were originally designed for transactional design. As a consequence, patterns need to be defined over attributes with a discrete and finite domain. Where attributes are continuous, we firstly apply a discretization function *discr* that discretizes them. We hence obtain a set of discretized feature A^d for each $A_i \in A$, with $A_i^d = discr(A_i)$ if A_i is continuous, $A_i^d = A_i$ otherwise. Hence, each attribute $A_i^d \in A^d$ can take a discrete, finite set $\mathcal{D}_{A_i^d}$ of values, and we let $m_{A_i^d} = |\mathcal{D}_{A_i^d}|$.

Definition 3.1.5. Association rules. An *association rule* is a implication in the form $A \rightarrow B$, where A and B are disjoint itemsets, with $A \cap B = \varnothing$ [156]. A is referred to as rule body or rule antecedent while B as rule head or rule consequent.

To evaluate the quality of an association rule, the support and confidence metrics are commonly exploited. Similarly to Definition 3.1.3, the support of a rule $s(A \rightarrow B)$ is the fraction of instances in D that satisfies both A and B, with $s(A \rightarrow B) = s(A \cup B)$. Intuitively, the support of a rule determines how often a rule is applicable to a given data set. The confidence of a rule $conf$ is the fraction of instances that satisfies $A \cup B$ over the number of tuples that A, with $conf = \frac{s(A \cup B)}{s(A)}$ Intuitively, the confidence of a rule determines how frequently items in B appear in instances that contain A.

3.2 Algorithms for frequent patterns mining

Frequent pattern mining is a data exploration technique that finds relationships that occur frequently among items in a dataset by extracting frequent itemsets [156, 157]. Frequent patterns are combinations of attribute values that have a frequency of co-occurrence to be considered significant. These patterns may hence represent interesting relations among instances in the dataset. As presented in Definition 3.1.4, the significance of the frequency of the itemset is determined with respect to the minimum support frequency threshold s. Several algorithms for mining frequent items have been proposed [3, 70, 172]. In the following, we will review two well-known and broadly applied mining techniques, the Apriori [3] and FP-growth algorithms [70].

Apriori algorithm. Apriori is a frequent pattern mining algorithm that exploits the support threshold to limit the search space [3]. The approach is based on the Apriori principle that states the following: 'if an itemset is frequent, then all of its subsets must also be frequent' [3]. It holds due to the antimonotone property of the support measure. Given two arbitrary itemsets A and B, if $A \subseteq B$ then $sup(A) \geq sup(B)$.

Apriori consists in a bottom-up and level-wise search, from frequent itemsets of only one item to the longest (i.e. highest number of items) of frequent itemsets. The level-wise search consists of two main operations, the candidate generation and the candidate pruning [3]. In the candidate generation, the k-frequent itemsets are used to generate candidate itemsets of length k+1. The process starts with itemset candidates of length 1. In the candidate pruning phase, candidate k+1-itemsets are pruned according to the Apriori principle. All candidate itemsets of length k+1 which are a superset of an infrequent k-itemset are pruned. Then, the support of the pruned candidate k+1-itemsets is computed and candidates below the support threshold are pruned.

One of the major drawbacks of the Apriori approach is that multiple datasets scans are performed, one for each iteration. In particular, if l is the length of the longest frequent pattern, the total number of scans is l+1. Moreover, the candidate

generation phase generates a large number of candidates and it may be especially critical for 2-itemset candidates.

FP-growth algorithm. FP-growth algorithm performs a depth-first search, scanning the dataset only twice and without candidate generation [70]. The key concept of FP-growth is its encoding of the dataset in a compressed representation called Frequent Pattern tree, FP-tree. Frequent itemsets are directly extracted from FP-trees. The dataset is scanned only twice. The first scan computes the support count of each item. Frequent items are then sorted in descending order. During the second scan, the FPtree is constructed. For each data instance x in the dataset, the items in x are sorted according to the support. The items are then inserted in the FP-tree using an existing path if the prefix is shared or creating a new path if not. The extraction of frequent patterns is performed by recursively visit the FP-tree in a bottom-up fashion applying a divide-and-conquer approach. FP-growth algorithm is generally faster than Apriori algorithm thanks to its recursive search on the compact representation of the FP-tree [70].

3.3 Association Analysis

In this section, we first outline the concept of association rule mining. We then illustrate the definition of associative classifiers. Finally, we present an overview of relevant algorithms for deriving associative classification models.

3.3.1 Association rule mining

Association rule mining is one of the most popular and applied techniques of frequent pattern mining. It is an exploratory approach that models correlations in forms of association rules defined as Definition 3.1.5. Association rule mining techniques are widely adopted in multiple fields such as manufacturing [2, 32, 39], food industry [10, 158], and energy consumption [44] also for their intrinsic transparency.

The problem of association rule mining consists in the extraction of all the association rules having rule support greater than the minimum support threshold s and confidence greater than a minimum confidence min_conf threshold [156]. The thresholds allow controlling the statistical relevance of the extracted rules. The process of association rule mining is often decomposed in two steps [156].

The first step is the computation of frequent itemsets. This is also the step most computationally expensive and the frequent pattern mining techniques outlined in Section 3.2 are applied. The second step is the generation of association rules from frequent itemsets. Let be I a frequent itemset. The rule generation step consist in firstly deriving all the disjoint itemsets A and B from I such that $I = A \cup B$.

Then, the rule $A \rightarrow B$ is generated if the confidence $conf(A \rightarrow B)$ is greater or equal the confidence threshold min_conf.

3.3.2 Associative classifiers

Associative classifiers are classifiers that learn association rules for classification purposes. In this context, association rules are denoted as class association rules.

Definition 3.3.1. Class Association Rules (CARs). A class association rule (CAR) is an association rule $A \rightarrow B$ where B is a class label. CARs highlight the subsets of attribute values that are associated with the class label.

As discussed in Section 2.1.1, one of the main advantages of associative classifiers is their interpretability. CARs are easy to understand and they allow to have both a global and prediction-based understanding of the model behavior. The overall extracted CARs represent the global logic of the model. By inspecting instead the rule or rules determining the prediction, the user can inspect the reason behind the class labeling for an individual instance.

Several algorithms have been propose to extract accurate associative classifiers as CBA [99], CPAR [171], CMAR [95], iCAEP [178], L^3 [17]. The process of deriving high-quality CARs can be divided into three main steps [157]: (i) frequent pattern extraction, (ii) generation of CARs, and (iii) pruning the CARs.

The first step consists in the extraction of frequent patterns similarly to frequent pattern mining and association rule mining as discussed in Section 3.2. Note that differently than association rule mining, the class label must be part of the itemsets under analysis since, in CARs, the label is the consequent of the rule. During the second step, CARs are generated from the frequent patterns based on quality thresholds as the min_conf confidence threshold. Finally, the generated CARs are pruned to obtain a selection of high-quality rules. The number of rules generated can be large, with drawbacks in terms of quality of results, size, and efficiency. Rules are pruned to remove redundancy and misleading rules that may introduce incorrect classifications [157]. Rules are also ranked, generally preferring rules with large support and confidence values.

Existing techniques strongly differ on the applied pruning approaches. Some methods, as CMAR [95], prune rules using the chi-square testing, evaluating correlations among the rule body and the class label. CBA [99] uses pessimistic error rate. Both approaches then consider database coverage for pruning, firstly introduced by the CBA approach. The generated rules are tested against the training data. If a training data instance is covered by at least λ rules, the instance is removed from the training data. The rules that cover at least a training instance are considered. CBA [99] and CMAR [95] differ on the coverage threshold. For CBA $\lambda = 1$ while for CMAR $\lambda \geq 1$ (usually $\lambda = 1$).

The approach L^3 introduced the concept of lazy pruning to select high-quality rules [17]. L^3 will be exploited in Chapter 4 to derive a local surrogate interpretable model. Hence, in the following, we will briefly outline the L^3 main concepts.

L^3 classifier. *Live and Let Live*, L^3, is an associative classifier that relies on a compact representation, similar to FP-growth, and on a lazy pruning approach [17]. The intuition of pruning is to avoid disregarding useful rules. Only rules that may be 'harmful', i.e. that may introduce classification errors, are removed. In the rule selection process, CARs are categorized into three sets: used, spare and harmful rules. Harmful rules are rules that wrongly classify the training dataset. Intuitively, those rules do not increase classification accuracy.

Used rules or level 1 rules are CARs that correctly classify at least one instance of the training set. These rules are considered firstly in the classification process for assigning the class label. Spare rules or level 2 rules are instead rules that do not cover any instance in the training set but are also not harmful. Hence, they might be useful in classifying data. Level 2 rules are considered only if no level 1 rule matches the instance to be classified. L^3 differs from CBA [99] and CMAR [95] also on how the pruned rules are ranked. In the case of rules with the same support and confidence, the more specific rule, with greater length, is preferred to the more general one. The intuition is that a specific rule covering the same training points (i.e. same support) can be considered more specialistic and accurate. Hence, the learned classification model consists of two sets of ranked rules, the level 1 and level 2 rules [17]. Being an associative classifier, classification results can be easily interpreted (i) globally by inspecting the sets of rules and (ii) locally by analyzing the rules that match the prediction of a particular instance.

Chapter 4

LACE: Explaining Black Box Models by means of Local Rules

This chapter illustrates LACE (Local Agnostic attribute Contribution Explanation), a novel method to explain classifier predictions on single instances [129]. This methodology is model-agnostic. Hence, it is applicable to any classification method without making any assumption on its internal logic. The explanation highlights the subset of relevant features and feature values that play a role in the prediction provided by a specific classifier for a particular instance. The explanation is based on the knowledge of the local behavior of the model in the neighborhood of the instance, captured by an interpretable local model. The approach features an automatic technique to select the appropriate locality scope.

The chapter extends the work presented in [129] and is organized as follows. Section 4.1 describes the proposed explanation method. Section 4.2 proposes an estimation of the approximation introduced by the local model and illustrates the heuristic algorithm for the automatic selection of the locality scope. Section 4.3 introduces the experimental setting and the explanation validation measures. Section 4.4 presents experimental results and compares the explanations provided by our method with the prediction explanations given by different existing approaches. Finally, Section 4.5 draws the conclusions.

4.1 LACE explanation method

LACE is a model-agnostic explanation method to explain classifier predictions of individual instances for structured data [123, 128, 129]. LACE explanations provide the relevance of each attribute value and significant sets of attribute values of a particular instance for the prediction of its class label.

Given the particular prediction that we want to explain, LACE omits one or more attribute values at a time and measures how the prediction probability

changes. A change in the probability implies that the omitted attributes are significant for the prediction. In our work, we estimate the attribute influence in terms of prediction difference, by computing the difference of prediction probabilities with respect to a particular target class.

With respect to previous approaches [104, 138, 153, 154, 155], we are interested in providing explanations highlighting not only how each single attribute value is significant for the prediction, but also its interaction with the others. An attribute value can determine the prediction only if it is in conjunction with others. In this case, we need to observe how the prediction changes for sets of attribute values. LACE directly estimates which combinations of attributes are relevant for the particular prediction and computes the prediction difference only for the relevant ones. Without this key characteristic, computing the most relevant attribute configuration would require estimating the prediction difference for all the possible subsets of attribute values as discussed in Section 2.1.2. This step has an exponential time complexity, because it requires the power set computation, making it unfeasible in real applications. In our work, the relevant attribute subsets for the prediction are extracted a priori by a local interpretable model, trained in the neighborhood of the prediction to be explained. The local model provide also a qualitative understanding of the reason behind the prediction in form of patterns.

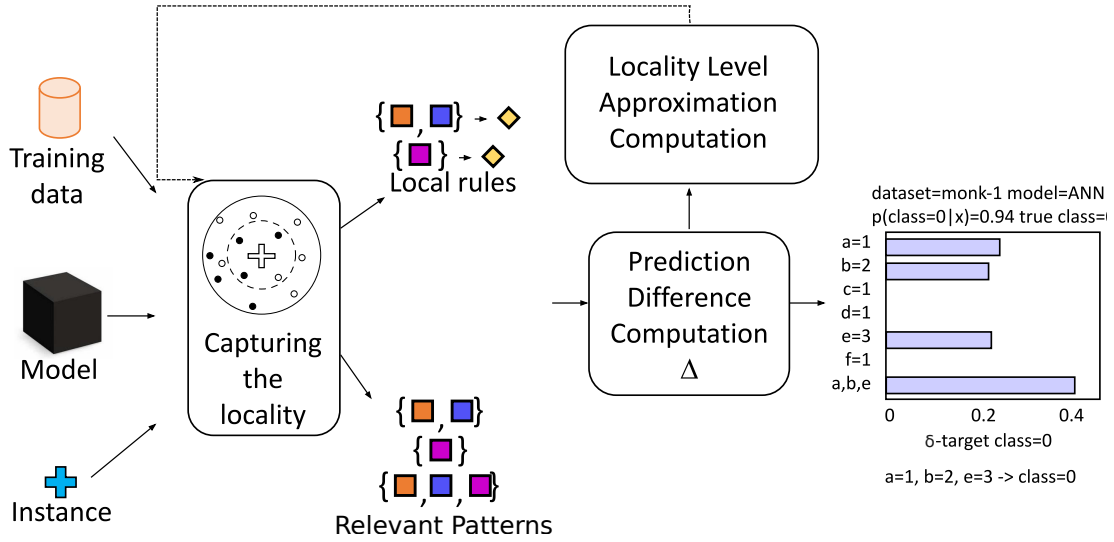


Figure 4.1: The main step of the LACE explanation method.

Figure 4.1 outlines LACE main steps. Let f be a generic trained classification model and x the instance whose prediction made by model f we want to explain. Firstly, LACE captures the locality of x by means of its K neighbors in the training set. The selected instances are labeled by model f and exploited as training data for learning a local model. In this way, the local model studies what model f has

learned in the locality of x. The local model provides a set of local rules from which the relevant patterns, i.e. subsets of feature values are extracted. They represent the subsets of attribute values that jointly determine the prediction for instance x. Next, the contribution to the prediction is estimated for each (i) attribute value and (ii) relevant patterns, in terms of prediction difference. The scope of the studied locality impacts the local rules and, thus, on the explanation itself. Hence, we propose an instance-based automatic approach to detect the appropriate number of neighbors K, which iterates until a good approximation of the prediction difference is reached. Finally, the prediction difference is visualized by means of a bar plot representation. In the following the steps of LACE approach are presented in detail.

4.1.1 Capturing the Locality by means of Local Rules.

The first step of LACE is the investigation of the locality of the individual prediction for the instance x to be explained made by classifier f. The goal is to capture the local behavior of the original model f in the locality of x. The intuition, firstly proposed by Ribeiro et al. in LIME [138], is to approximate the model f with an interpretable model in the locality of the prediction. While a simpler and interpretable model may fail to approximate the model globally, the local model can faithfully mimic the behavior in the locality of the prediction and hence derive locally faithful explanations.

We train a local interpretable model able to extract local rules. Local rules provide a qualitative understanding of the model behavior. From local rules, we derive the patterns as the subsets of attribute values that are relevant for the prediction. The relevant subsets are the only sets of attribute values considered for the prediction difference estimation.

Firstly, we derive the locality as data samples in the neighborhood of the instance. These samples are used as training dataset to fit an interpretable model to mimic the local behavior of the original model f. In the following, these two stages are illustrated.

Deriving the locality of a prediction

The locality of the instance x is the a set of instances $N(x)$ which are similar and close to x. It is used by the local model to learn the local decision boundary. In our work, the locality of x is captured by means of the K instances in the training set that are nearest to the instance x that we want to explain.

More specifically, let D be our training set and ρ a distance measure. The neighborhood of x is defined as $N(x) \subset D$ such that $|N(x)| = K$ and $\forall z \in D \backslash N(x)$, $\rho(x, z) \geq \max_{y \in N(x)} \rho(x, y)$. Each point not in $N(x)$ is at least as far away from x as the furthest point in $N(x)$. The K neighbors are then labeled by the original model f.

We refer to the labels of $N(x)$ as Y_{N_x}. We obtain a local training dataset ($N(x)$, Y_{N_x}) that represent the local behavior of f in x.

Existing approaches study the locality by generating perturbed instances randomly in the locality of the instance [138, 139], through a genetic model [68] or based on local intrinsic dimensionality of the instance [81]. However, in this way also non-existing configurations of attribute values may be generated and then studied by the local model. Considering the neighborhood of instance x in the dataset instead of generated perturbations enables training a local model only on existing attribute value configurations. We remark that the local model is used specifically to extract the set of attribute values that are relevant for the individual prediction in form of local rules. Hence, in our context, it is particularly relevant that the instances $N(x)$ to train the local classifier are indeed possible and observed configurations of feature values. While the access to the actual neighborhood is relevant to capture real associations, in some scenarios we may not have access to the data but only the instance to explain itself. In these situations, we have to opt for generated local samples via data perturbations or generation techniques to retrieve the neighbors and produce explanation with LACE.

The choice of parameter K for capturing the neighborhood determines the locality scope. We propose a heuristic instance-based algorithm for the automatic selection of K. To tune K value, we consider the changes it induces on the local model. A detailed description of the algorithm is presented in Section 4.2.2.

Deriving Local Rules

The K labeled neighbors ($N(x)$, Y_{N_x}) of instance x are used for training the local model. The local model provides the local association of feature values with the predicted label. We use an associative classifier that extracts classification rules from the training data.

Let $A_1, A_2, ..., A_\mathrm{d}$ be the set of attributes characterizing instance x. We recall from Section 3, that an itemset I (or pattern) of length $l \leq d$ is a collection of items in the form $\{A_\mathrm{i} = a_\mathrm{i}, A_\mathrm{j} = a_\mathrm{j}, ..., A_\mathrm{k} = a_\mathrm{k}\}$, where $|I| = l$ (Definition 3.1.2). An associative classifier extract CARs in the form $A \rightarrow B$, where A is an itemset, and B is a class label [99]. We refer the reader to Section 3.3 for an overview of associative classifiers.

In our implementation, we use the L^3 associative classifier [17], but any other associative classifier can be used for the same purpose. The associative classifier is trained with the K neighbors of the instance x that we want to explain and yields *local rules*.

Definition 4.1.1 (Local Rule)**.** Let I be an itemset of length $l \leq d$ defined on the attribute set $\{ A_1, A_2, ..., A_\mathrm{d} \}$ and c a class label. A local rule is a CAR in the form $I \rightarrow c$ returned by an associative classifier model learned in the locality of instance x where x *is covered* by I, $x \models I$.

Local rules are derived from the CARs extracted by the local model. We select the CARs where the instance x satisfies the rule antecedent I, i.e. if $x \models \alpha_i$ for $1 \leq i \leq l$.

Each local rule body is a conjunction of items in the form (*attribute=value*). Thus, each rule gives the indication of which configuration of attribute values *together* determines the prediction.

From the selected local rules, we retrieve patterns, i.e. the subsets of significant feature values. Instead of computing the contribution of each possible subset of x, as done in [155], we consider only the relevant subsets highlighted by the local model. From each local rule, we derive its corresponding relevant pattern. We will refer to relevant patterns also as relevant subsets (of attribute values).

Definition 4.1.2 (Relevant Pattern)**.** Let $\{A_{\mathrm{i}} = a_{\mathrm{i}}, A_{\mathrm{j}} = a_{\mathrm{j}}, ..., A_{\mathrm{k}} = a_{\mathrm{k}}\}$ be an itemset of length l. Let $r : I \rightarrow c$ be a local rule. Its corresponding relevant pattern S_r is the subset of (*attribute=value*) $A_i = a_i, A_j = a_j, ..., A_k = a_k$.

The local model may return multiple local rules. In this case, the prediction of instance x may be determined by more than one rule. By considering only one rule, we might not observe the complete effect of all the different factors affecting the prediction. We represent the presence of multiple rules by considering the union of all the corresponding rule bodies. We will denote the set of subsets extracted from all the local rules as U in the following.

Definition 4.1.3 (Union Itemset)**.** Let $\{\ r_1, r2, ..., r_h\ \}$ be a set of local rules in the form $ri : I_i \rightarrow c_i$ be a local rule. The union itemset U is the union of the relevant patterns $S_{r_1}, S_{r_2}..., S_{r_h}$, $U = \bigcup_{i=1,...,h} S_{r_i}$. The union itemset U is then a set of attribute values in the form (*attribute=value*).

From the local rules, we obtain the relevant patterns and the union itemset, if any. Instead of computing the contribution of each possible subset of x, we consider only the relevant subsets highlighted by the local model for the prediction difference evaluation illustrated in the following section.

4.1.2 Prediction difference

The explanation of the single prediction x made by black-box model f highlights the contribution of each relevant (i) attribute value and (ii) subset of attribute values in the prediction of the class label. The estimation of these contributions is driven by the relevant feature subsets provided by the local rules. The influence of an attribute value or of a relevant pattern on a prediction is estimated in terms of *prediction difference*. The prediction difference expresses how the prediction changes when one or more attribute values are omitted.

Most classification models require all the features as input for making predictions. Hence, we cannot directly provide to classifiers a subset of the features and

observe how the prediction changes. Several approaches have been proposed to simulate the feature removal and summarize their contribution [40, 93, 104, 132, 147, 155, 176]. As outlined in [37], these approaches differ on three dimensions: feature removal, model behavior, and the summary technique. The feature removal dimension refers to the method used to approximate the omission of features. The second choice regards what model behavior to analyze. Finally, the methods differ in the way it is used to summarize the contribution of features.

In the following, we firstly outline preliminary definitions and we review the approaches for removal-based explanations. We then outline and motivate the strategies chosen in this work over the three dimensions. Finally, we illustrate the notion of prediction difference over individual features and relevant subsets.

Preliminaries and overview. Let $X = (X_{A_1}, X_{A_2}, ..., X_{A_d})$ be the attribute space with schema A over which our classifier f is defined (refer to Section 3 for the detailed definitions). X_{A_i} denotes a random variable while x_{A_i} denotes its values. Let S denote a subset of attributes $S \subseteq A$ and $\bar{S}$ its complement, with $\bar{S} = D \setminus S$. The term $x_S = \{x_{A_i} : A_i \in S\}$ is defined over the subsets of features S.

We denote with F a function defined over subsets of attributes that models the prediction of the classifier f [37]. $F(x_S)$ depends only on the attributes in S and it is invariant to $\bar{S}$. We note that $F(x) = F(x_A) = f(x)$. Hence, the output of F when all features are available is equal to the output of the model $f(x)$ itself.

Existing removal-based explanation methods vary on how F is defined [37]. Several approaches approximate the feature omission by simply setting them to zero, with $F(x_s) = f(x_S,0)$ [132, 176] . Other approaches substitute the features to be removed with default values [40, 138]. For example, LIME for image data substitutes super-pixels by means of grey pixels [138]. Another solution consists on marginalizing with marginal [37]. The features to be removed are marginalized using their join marginal distribution $p(X_{\bar{S}})$. As a result, $F(x_s) = \mathbb{E}[f(x_s, X_{\bar{S}})]$. Examples of explanation using this solutions are KernelSHAP [104] and the permutation tests defined by Breiman [25]. Intuitively, the omission of feature values is approximated by sampling values from the feature's marginal distribution. Recent studies as [79] examined the benefits of this approach over other solutions that marginalize using the feature conditional distribution. We, therefore, consider this solution of marginalizing with marginal as the method used to simulate the omission of features in our work.

Removal-based explanation methods then may differ on the target of model behaviors [37]. While some approaches consider the loss of the model [105, 147], most of the methods directly analyze the *prediction* of the model and its probability for the instance x [104, 155, 176, 179]. In this scenario, the target function is $F(x_s)$ itself. Our work can be categorized among these approaches that consider the prediction as the target function. In our definition, we explicitly consider a multi-classification scenario. Let c be a class value of our target class label y. We are

interested in the probability of the instance x to belong to the target class c. Hence, we consider $f(y = c|x)$ and $F(y = c|x_S)$ as our target prediction function.

The last dimension of interest is how to summarize the influence of the features estimated via omission. Some approaches estimate the importance of each individual feature by removing it. The importance of an attribute A_i is then estimated as the difference of the target function when all the attributes are considered and when the attribute A_i is omitted [93, 147, 176, 179]. However, removing individual features does not consider how the feature interacts with the others. Other approaches estimate the feature value importance by also omitting subsets of attribute values in order to consider feature interactions. Works as IME [155] and SHAP [104] use the notion of Shapley value that satisfies different properties as symmetry and efficiency [150]. However, this summarization technique may be computationally challenging since it requires the estimation of the omission for many subsets S of features. The "exact" computation of the Shapley value in fact requires the complete exploration of all 2^d subsets of features. To overcome the exponential time complexity of the problem, several approximations have been proposed. These methods exploit sampling strategies [153], linear regression models [104] or dynamic programming algorithms [105].

These solutions summarize the information on attribute interaction in one contribution for each attribute value. Hence, the information on the interaction of attribute values is lost. Differently, we explicitly consider the relevant association of attribute values and we estimate the feature omission for those subsets of attributes. Our approach (i) estimates the influence of each attribute value for a particular prediction, and (ii) highlights separately each relevant attribute interaction. We overcome the problem of exponential time complexity by considering only the relevant attribute values derived from the local rules.

Prediction difference. We denote the observed change in the classification outcome when attribute values are omitted as *prediction difference*, using $f(y = c|x)$ and $F(y = c|x_S)$ as our target function.

To estimate the influence of a single attribute value, we consider the omission of a single attribute at a time. The prediction difference δ_{i} with respect to attribute A_{i} is computed as follows:

$$\delta_{A_i} = F(y = c|x) - F(y = c|x_{A \setminus A_i}) = f(y = c|x) - F(y = c|x_{\bar{A}_i}) \tag{4.1}$$

We recall that, by definition, $F(y = c|x) = f(y = c|x)$ and corresponds to the probability of the instance x to belong to the target class c for the classifier f. The second term $F(y = c|x_{D \setminus A_i})$ is defined as the probability of the instance x to belong to the target class c when attribute A_{i} is omitted. As mentioned, in our work the approximation of this quantity when the attribute A_{i} is omitted is computed via marginalization [37]. In the following, when clear from the context, we will refer to δ_{A_i} also as δ_i to refer to prediction difference when the i-th attribute is omitted.

Equation 4.1 allows us to compute how the prediction for instance x changes when a single attribute value is omitted, i.e., $\delta_i(x)$, for all the attributes $A_1, A_2, ...,$ A_d. For each attribute A_i, $\delta_i(x)$ shows its relevance in the definition of the class label.

Attributes may contribute *jointly* to a class prediction. Considering the omission of more attributes at a time allows us to quantify how these attributes *jointly* influence the prediction. Hence, we are interested in estimating if attribute values together are significant for a prediction. To estimate the omission of more attribute values at a time, we use again $F(y = c|x_{A\setminus S})$. The jointly influence of the attributes S for the prediction of the class c for the instance x is computed as:

$$\delta_S = F(y = c|x) - F(y = c|x_{A\setminus S}) \tag{4.2}$$

Differently from existing works, we explicitly consider only the relevant attribute values derived from the local rules. We compute the prediction difference (Equation 4.2) for (a) each relevant pattern S_r derived by the local rules and (b) the union itemset U. Considering only the local behavior captured by the local rules allows us to deal with h+1 subsets instead of 2^d (i.e., the cardinality of the power set $P(X)$), where h is the number of local rules and h+1 $\ll |\,\mathrm{P(X)}\,|$.

Each $\delta(x)$ term represents the influence of (i) a single feature, or (ii) jointly more feature values together in the determination of the class label for the prediction on a single instance x made by the black box model f. It ranges from -1 to +1. The larger the value of influence, the more the corresponding attribute values influence the class assignment. A positive contribution means that the attribute values have a positive influence in assigning the class label. A negative one, instead, means that the attribute values are against the assignment of the considered class label.

Evaluating the neighborhood.

To evaluate how well the extracted local rules are able to capture the locality of the prediction, we compare the prediction difference for the union of all rule bodies, $\delta_U(x)$, with the prediction difference when all the attribute values are removed. If the variation is low, it means that removing all rule bodies entails removing all the relevant information from instance x. In this case, the local model is a good approximation in the locality of the prediction. A detailed description of the computation of the approximation is presented in Section 4.2.1. On the contrary, the entire process is repeated varying the value of K. The detailed descriptions of the computation of the approximation of the locality scope and of the instance-based heuristic algorithm for selecting parameter K are presented in Section 4.2.

4.1.3 Visualization

Visualizing the contributions of relevant attributes and subsets enables inspecting quantitatively the relevance of the different factors of an explanation.

All the $\delta(x)$ terms can be visualized through a bar plot representation. The visualization allows users to understand in a simple and uniform way the motivations driving the prediction for instance x made by model f. An example of visualization is the bar plot to the right in Figure 4.1.

Each representation is built to explain the prediction of a target class c performed by a model f for a particular instance x. The single attributes with their corresponding value and all the relevant attribute subsets are shown on the vertical axis. The prediction difference $\delta(x)$ of each element with respect to the target class is plotted on the horizontal axis. More specifically, the diagram plots the corresponding $\delta(x)$ term for each attribute A_i, for each relevant pattern S_r, and, finally, if the prediction is performed by more than one rule, the $\delta_U(x)$ term encompassing the contribution of all relevant subsets (bottom row in the plot).

4.2 Estimation of the neighborhood

The local model has a great impact on the prediction explanations. Only the relevant subsets determined by the local model are considered, instead of the complete power set.

The locality of the instance is controlled by parameter K. The K nearest neighbors of the instance to explain labeled by f are the only input of the local associative classifier. We devise a heuristic approach to automatically tune the parameter K. The proposed technique selects the appropriate locality scope by evaluating the quality of the extracted local rules to capture the local properties of the instance. In the following sections, we first illustrate the notion of faithfulness of the local rules to capture the relevant associations. The intuition is that removing the relevant local associations entails removing the information of the instance itself. We use this notion to assess the approximation introduced by the local rules in capturing the locality. We then propose a heuristic approach that leverages this evaluation of the approximation introduced by the locality scope to adapt the number of samples for the neighborhood.

4.2.1 Approximation evaluation

Through the local model, we obtain local rules. These rules should capture the local behavior in a neighborhood of instance x. From the local rules, we derive the relevant subsets, i.e., the sets of attribute values, that are actually relevant in determining the prediction. These rules are the ones that determine the class for the instance x. Hence, omitting the corresponding subsets of attribute values would be like omitting the complete information of the instance x. Instead, omitting the subsets not belonging to the relevant subsets should not be significant.

We define Π as the prediction difference $\delta_A = \delta_{A_1,A_2,..,A_d}$ when all the attribute

values of the instance are omitted, i.e. when the complete information is removed. It is computed as:

$$\begin{aligned}\Pi = \delta_{A_1,A_2,..,A_d} = \\ = f(y=c|x) - F(y=c|x_{A\setminus A_1,A_2,...,A_\mathrm{d}}) = \\ = f(y=c|x) - F(y=c|x_\emptyset)\end{aligned} \tag{4.3}$$

By applying Equation 4.2, $\delta_{A_1,A_2,..,A_d}$ can be written as a difference of two terms. The first term is the probability of instance x for the classifier f to belong to class c. The second term approximates the class probability when all the attributes A_1, ..., A_d are removed. This term corresponds to the probability of belonging to the class c when no information is available.

To approximate the omission of the complete information, we omit all relevant subsets by omitting the *union itemset U(x)*. We define Π_{approx} as:

$$\begin{aligned}\Pi_{approx} = \delta_U = \\ = f(y=c|x) - F(y=c|x_{A\setminus U}) = \\ = f(y=c|x) - F(y=c|x_{A\setminus\bigcup_{i=1,...,h} S_{r_i}})\end{aligned} \tag{4.4}$$

where $S_{r_1}, S_{r_2}..., S_{r_h}$ are the relevant patterns derived from the local rules $\{ r_1, r_2, ..., r_h\}$. It represents the prediction difference of the union itemset U. The attributes considered in U are a subset of $A_1, \ldots, A_d$.

The terms Π and Π_{approx} are used to estimate the absolute locality approximation. It is the deviation between the exact value and its approximated value:

$$\begin{aligned}\epsilon = |\Pi - \Pi_{approx}| = \\ = |p(y=c|x) - p(y=c|x\setminus A_1, A_2, A_3, ..., A_d) - \delta_U| \\ = \left|F(y=c|x_{A\setminus\bigcup_{i=1,...,h} S_{r_i}} - p(y=c|x\setminus A_1, A_2, A_3, ..., A_d)\right|\end{aligned} \tag{4.5}$$

The *locality approximation* ϵ quantifies the capability of the local rules to capture the local behavior of the original model f in the locality of x. If the approximation value ϵ is low, the local rules actually determine the prediction. The locality approximation can be used to tune parameter K, i.e. the number of neighbors, as presented in the following section.

4.2.2 Automatic Locality Definition

We estimate and visualize how the prediction changes only for the relevant subsets determined by the local model. If the local model captures the behavior in the locality of the instance to explain, the relevant subsets are also the most meaningful ones, in terms of prediction difference. The selection of parameter K

Algorithm 1: Compute Contributions selecting a suitable K

```
   Input: data, x, K, f
   Output:
 1 oldRules = {};
 2 for (K=SK; K < maxK; K=K+SK) do
 3 |   rules=getLocalRules(data, K, x, f);
 4 |   if rules == oldRules && K! = SK then
 5 |   |   continue;
 6 |   end
 7 |   oldRules=rules;
 8 |   rules=removeOverfittingRules(rules);
 9 |   δ(x)=computePredictionDifference(data, x, rules);
10 |   ϵ=computeLocalityApproximation(data, deltas, x);
11 |   if ϵ < ϵ_Threshold then
12 |   |   return δ(x);
13 |   end
14 |   if K! = SK then
15 |   |   if ϵ > ϵ_old then
16 |   |   |   return δ_old(x);
17 |   |   end
18 |   end
19 |   ϵ_old=ϵ;
20 |   δ_old(x)=δ(x);
21 end
22 return δ(x);
```

determines the locality degree for the instance explanation. If K is too low, the K neighbors used for training the local model may be too few and the resulting local model may not be able to capture the local behavior. Larger values of K reduce the effect of noise and outliers. However, if K is too large, the K neighbors will not represent the locality of the instance x anymore. The local model in this case may not be able to correctly mimic the local behavior. For these reasons, choosing an optimal K is of key importance.

The proposed methodology to tune the value of parameter K aims at yielding a model that captures the locality of the instance to be explained. We estimate the quality of the resulting model by means of the locality approximation ϵ defined in the previous section. The selected value of K should yield a low approximation value.

The baseline value for K, SK, is defined by using heuristics already exploited to choose parameter K for the K Nearest Neighbor classifier. In particular, we consider K equal to $\sqrt{n}$, where n is the training dataset size for large datasets [49]. Otherwise, K is proportional to the size, in the original training dataset, of the particular target class [82]. In this way, for larger classes, more nearest neighbors are used. Then K is progressively increased until one of these stopping criteria is reached:

- the approximation value is lower than a predefined threshold
- a local minimum is found (i.e., the new approximation value is greater than the old one),
- value K is too large, thus the K neighbors will not represent the locality of x anymore.

This heuristic technique is outlined in Algorithm 1. The inputs are the training dataset and the baseline K value denoted as SK. The training dataset is used to compute the neighbors. SK value is set using the heuristics already mentioned.

In Line 2, starting with K as SK, K is progressively increased with a step SK until a stopping criterion is satisfied. The incremental step SK is a value greater than one, which depends on the cardinality and properties of the particular dataset from which instance x is extracted. Instead, increasing K value by a step of one would entail adding each time only one instance to the training data for the local model. Being the training data only slightly changed, the local model would probably capture the same local behavior and same local rules as before.

In Line 3, the local model is trained using as training dataset the K neighbors of instance x labeled by model f and the local rules are returned. From the second iteration, we check if the extracted rules are equal to the ones obtained with the previous value of K. In this case, the terms $\delta(x)$ are not re-estimated. The local rules are the same and thus the $\delta(x)$ terms. Moreover, in Line 8, if existing, an over-fitting

rule (i.e., a rule describing the instance x itself) is detected and discarded. This rule does not represent interesting information. An explanation should highlight which are the components of the input that are major determining the predictions. Of course, the instance itself is relevant for the prediction but it does not explain which are the most significant terms.

In Line 9, the terms $\delta(x)$ are estimated for each single attribute value, for each relevant subset and for all the relevant subsets together using Equation 4.2. In Line 10, the locality approximation ϵ is computed as discussed in Section 4.2.1.

Next, stopping criteria are checked. In Line 11, we check if the obtained approximation value ϵ is lower than a predefined threshold representing the approximation we are willing to accept. In the positive case, the local rules and their prediction difference are considered a good approximation of the behavior in the locality of the predictions, and the algorithm returns the $\delta(x)$. Otherwise, in Line 15, if the new approximation value is greater than the previous one, a local minimum is reached. We return the previous values of $\delta(x)$, derived from the local rules obtained with the previous value of K. If no stopping condition is met, the values of the old approximation value and old local rules are updated in Lines 19-20. The heuristic search continues until a stopping criterion is satisfied.

4.3 Experimental Setup

LACE has been developed in Python, exploiting the scikit-learn [131], pandas [109] and numpy [74] libraries. We use the level-1 rules of L^3 associative classifier to capture the local behavior in the neighborhood of the prediction [17] and its Python implementation [15]. In the following, we will introduce the dataset exploited and their pre-processing, and the experimental setup. The source code and data for all the experiments are available [1].

Data sets

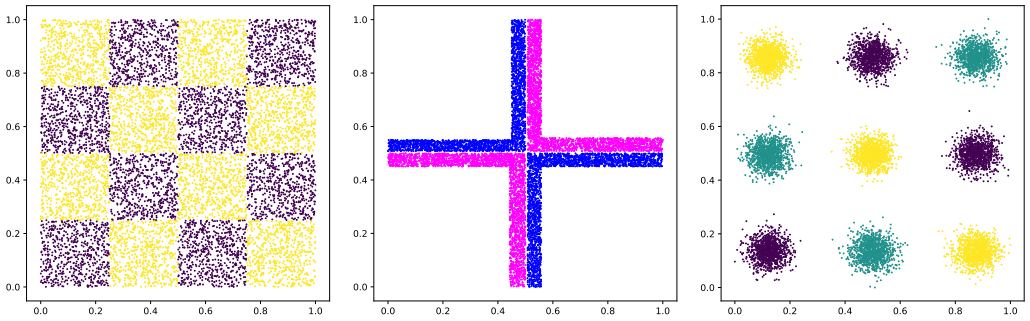
We ran the experiments on both artificial and real-world data sets. The main features of the datasets used in our experiments are reported in Table 4.1.

Artificial data sets are data sets for which the relationship between attributes and the class value is known. Hence, it is possible to validate explanations, by comparing them with the true relationships among attributes. For the artificial dataset, we use the *chess* [141], *cross* [141], *groups* [141] and *monks* [96] dataset. The *chess*, *cross* and *groups* data sets are composed of four attributes [141]. The first two attributes X and Y define the class. The other two attributes, *R1* and *R2*, take random values and are unrelated to the class. The visualization of the

[1] https://lacexplain.github.io/

dataset	$\|D\|$	$\|A\|$	$\|A\|_{cont}$	$\|A\|_{cat}$
adult	45,222	11	4	7
COMPAS	6,172	6	2	4
chess	10,000	4	4	0
cross	10,000	4	4	0
group	10,000	4	4	0
group_10	10,000	10	10	0

Table 4.1: Dataset characteristics. A_{cont} is the set of continuous attributes, A_{cat} of categorical ones.



(a) Chess. (b) Cross. (c) Groups.

Figure 4.2: Visualization of the two relevant attributes X and Y in the *chess*, *cross* and *groups* datasets.

two important attributes of these threes artificial dataset are shown in Figure 4.2. The X and Y attributes take the form of a 4x4 chessboard for the *chess* data set and the class labels are *0* and *1* (Figure 4.2a). The X and Y attributes in the cross dataset draw a cross with two possible labels on the diagonal (Figure 4.2b). In the *group* data set, X and Y are disposed in groups and instances are divided into three possible classes (Figure 4.2c). The *monks-1* data set is composed by 6 discrete attributes *a, b, c, d, e, f* and the class label can take value 0 or 1 [96]. The relationship between the attributes and the class value is known. The class is 1 if *a=b* or if *e=1*, 0 otherwise. To evaluate the impact on explanations of the number of attributes, we propose an extended version of the *group* dataset [141]. We extend the *group* artificial data set by adding a total of 8 random attributes to the two relevant ones X and Y for a total of 10 attributes. We will refer to this extended version as *group_10* dataset.

The real-world tabular data sets we used are *adult* and *COMPAS*. The UCI *adult* data set task is to predict whether the income of a person exceeds 50K per year based on census data [96]. The *COMPAS* data set from Propublica consists

of demographic information labeled with criminal recidivism risk scores [8].

Data processing

L^3 associative classifier used for learning local models and our computation of the prediction difference work with discrete data. As in LIME [138], the continuous attributes of the labeled neighbors $N(x)$ are discretized to derive the local model. For the artificial data sets *chess*, *cross* and *groups*, we used the prior knowledge of the problem for selecting the number of intervals, as done in [141]. We set the number of intervals to 2 for the *cross*, 3 for the *groups* and 4 for the *chess* data sets and used discretization with equal width of intervals. For continuous attributes of real-data sets, we used equal-frequency discretization. We note that this step is for explanation purposes only and the classifier can be generally defined on continuous attributes.

For the *COMPAS* data set we use the following features: sex, age_cat, race, priors_count, c_charge_degree and length_of_stay. We considered two classes of risk of recidivism score, "Low-Medium" and "High". For the *adult* data set we used sex, workclass, education, race, marital-status, occupation, relationship, capital-gain, capital-loss, hours-per-week, age.

Each data set is split into *training* set and *explain* set. Training data set is used to train classification models. LACE explanations are produced for instances in the explain set.

Experimental setup

We consider three different models: Random Forest (RF), Naive Bayes (NB), and Multilayer Perceptron artificial neural networks (MLP). We use the implementations and default parameters of scikit-learn [131] unless otherwise specified. Support and confidence thresholds of the associative classifier are set to the default values, i.e., 1% and 50% respectively. The locality approximation that we are willing to accept in the automatic approach for tuning parameter K is set to 0.02. The explanations are generated considering the predicted class as the target class unless otherwise stated. The experiments were performed on Ubuntu 16.04.1 LTS 64 bit, 16 GB RAM, Intel Core i7.

4.3.1 Explanation evaluation

We evaluate the quality of explanations by measuring how explanations are close to the *true explanations*. Let $\hat{e}_M(x)$ the explanation provided by an explanation method M for the instance x with respect to a generic classifier f and $e(x)$ the true explanation.

We consider two forms of explanation: feature importance vector and rule explanations. A feature importance vector is an explanation in the form $t{=}t_1, t_2, ..., t_d$

where t_i is the importance of the feature A_i. Rule explanations are provided as a set of local rules r (Definition 4.1.1) in the form $I \rightarrow c$ where I is a set of attribute-value pairs and c is the class values.

The form of the explanation $\hat{e}_M(x)$ depends on the explainer M. The explainers SHAP and LIME provide the explanations only in the form of feature importance vectors. Anchor provides explanations in form of rules. As discussed in Section 4.1, LACE provides explanations in form of both feature importance vector and as a set of local rules.

We evaluate the quality of feature importance explanations and of the set of local rules by measuring its adherence with the *true explanations* adopting the following metrics [66, 81].

For feature importance explanations, we measure the quality of explanation by means of two metrics: the feature similarity $f\text{-}sim$ and the f1-score $f1\text{-}feature$ [81].

Let be $\hat{t}(x)$ the feature importance vector provided by the explanation method M and $t(x)$ the *true* feature importances for instance x.

Feature cosine similarity - f-sim. The feature similarity $f\text{-}sim$ measures the similarity of feature importance vectors using the cosine similarity:

$$f\text{-}sim = cosine(\hat{t}(x), t(x)) = \left\| \frac{\hat{t}(x) \cdot t(x)}{\left\|\hat{t}(x)\right\| \|t(x)\|} \right\| \tag{4.6}$$

where $\hat{t}(x) \cdot t(x)$ is the dot product of the two explanation vectors and $||$ is the l^2-norm [81]. The closer $f\text{-}sim$ is to 1, the closer $\hat{t}(x)$ is to $t(x)$ and so the higher the quality of $\hat{t}(x)$.

We consider also the case of *binary explanations* [81]. A binary explanation highlights if the features are relevant or not for the prediction. In this case, $\hat{t}_i(x)$, $t_i(x) \in \{0,1\}$, where $t_i(x)$ is 1 if the feature i is relevant while it is 0 is it does not influence the prediction. For binary explanation, we use the f1-score.

F1-score of feature importance vectors - *f1-feature*. We refer to the f1-score of explanation vectors as $f1\text{-}feature$ and it is defined as follow.

$$\begin{aligned} f1\text{-}feature &= f1\text{-}score(\hat{t}(x), t(x)) = \\ &= \frac{2 * recall(\hat{t}(x), t(x)) * precision(\hat{t}(x), t(x))}{recall(\hat{t}(x), t(x)) + precision(\hat{t}(x), t(x))} \end{aligned} \tag{4.7}$$

where the $recall(\hat{t}(x), t(x))$ measures how many features that are considered as relevant by the explainer M are truly important and the $precision(\hat{t}(x), t(x))$ measures how many features that are truly important are correctly identified by $\hat{t}(x)$ [81]. We note that any *numerical* explanation $t(x)$ (with $t_i(x) \in \mathbb{R}$) can be mapped to a

binary explanation by setting to 1 the relevant terms (with $abs(t_i)(x) \geq 0$) and 0 otherwise [66].

F1-score of rule explanations - *f1-rule.* We use the f1-score also for the evaluation of the quality of explanations in form of local rules [66, 81]. Let be $\hat{r_1}, ..., \hat{r_h}$ the set of h rules provided by the explanation method M and $r_1, ..., r_k$ the set of k *true rules* for instance x. We map each rule to a binary vector representing the presence or absence of a feature in the rule. We then use the $f1$-$score$ to measure the quality of rules. The $f1$-$score(b(\hat{r_i}(x)), b(r_j(x)))$ measures the quality of the binarized rule $\hat{r_i}(x)$ with respect to the true rule $r_j(x)$, where $b(\cdot)$ is the operator that maps the rule to the binary vector representations. We will refer to the $f1$-$score$ for the estimation of rule quality as $f1$-$rule$.

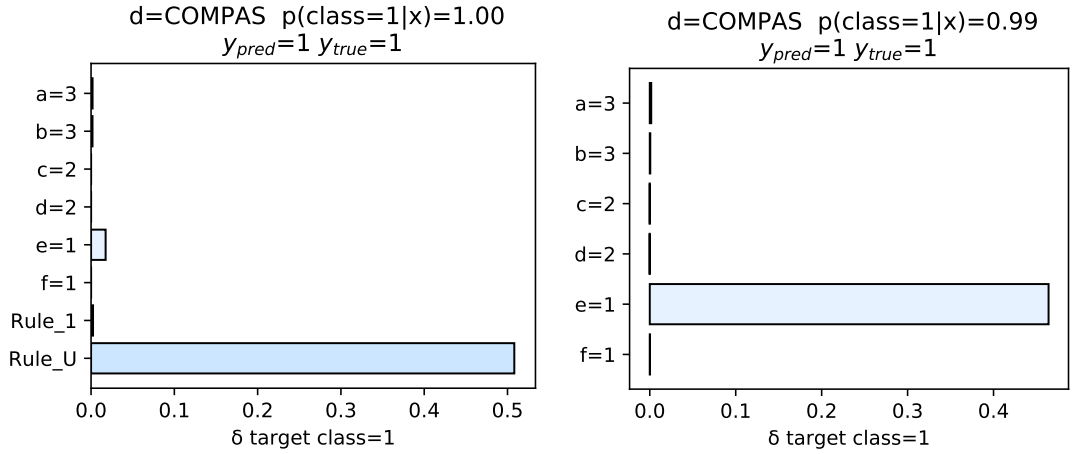
Rule hit - *r-hit.* We also introduce the notion of rule hit *r-hit* to understand the behavior of rule-based explainers. The rule hit *r-hit* $\in \{0,1\}$ assesses if *true* rules are captured by the set of rules provided by the explanation method M. It is 1 if the true rules r are also in $\hat{r_1}, ..., \hat{r_h}$, 0 otherwise. We also evaluate if the true rules are partially covered by the explanation rules. Let be r_i a true rule that finds not matching in $\hat{r_1}, ..., \hat{r_h}$. We distinguish two cases: a subset match and a superset match. We say that an explanation rule $\hat{r}$ is a subset of r_i if all the items composing it appear in r_i: $\forall \alpha_j \in \hat{r}$: $\alpha_j \in r_i$, with $|r_i| > |\hat{r}|$. For example, rule $\{a, b\}$ is a subset of $\{a, b, c\}$. Correspondingly, we say that an explanation rule $\hat{r}$ is a superset of r_i if: $\forall \alpha_j \in r_i$: $\alpha_j \in \hat{r}$, with $|r_i| < |\hat{r}|$. The *r-sub-hit* $\in \{0,1\}$ and *r-sup-hit* $\in \{0,1\}$ assess the type of match, subset and superset match respectively.

The illustrated metrics leverage the knowledge of the *true explanations* behind model predictions. As discussed in Section 1.2.3 the true explanations to evaluate prediction explanations are unfortunately normally not available. In almost all cases, the true association between attribute values and class value is not known. For these reasons, we use the following approaches, illustrated in Section 1.2.3, to evaluate the quality of the provided explanation. On one hand, we consider artificial data sets to validate our explanation method since their true association is known. For real datasets, we consider two settings to known the true explanations a priori. As a first approach, we control the classifier behavior by injecting associations into the data. The classifier will learn the injected associations as relevant for the predictions. We then use the relations with the class label as the true explanation. As a second setting for real datasets, we use white box classifiers to derive ground truth explanations and we exploit them to evaluate the local explanations as used in [66, 81, 129].

4.4 Experimental results

The set of experiments analyzes individual explanations to assess the quality of LACE explanations and highlights LACE ability to provide insights on the reason behind individual predictions. We will firstly qualitatively discuss the results of LACE and compare them with state-of-the-art techniques via a running example with the *monks* dataset. We will then extensively quantitatively evaluate the explanation methods by comparing the extracted explanation with the *true explanations*. We then evaluate the effect of the instance-based automatic approach for detecting the locality on the explanations.

4.4.1 Running example with monks dataset.



(a) MLP. (b) NB.

Figure 4.3: LACE explanation of a MLP (a) and NB predictions of the *monk-1* dataset. Local rules for the MLP prediction: Rule_1: {a=3, b=3} → class=1, Rule_2: {e=1} → class=1, Rule_U: {a=3, b=3, e=1} → class=1.

We present and discuss the explanations produced on the *monk-1* data set [96]. The *monk-1* data set is composed by 6 discrete attributes *a, b, c, d, e, f* and the class label can take value 0 or 1. Being the data set artificial, the relationship between the attributes and the class value is known. The class is 1 if *a=b* or if *e=1*, 0 otherwise. Thus, it is possible to validate the results of our explanation method, by comparing the provided explanations with the true relationships among attributes.

In Figure 4.3a we report the explanation provided by LACE for a multilayered feed-forward artificial neural network (MLP). The explanation is provided for instance x = *(a=3, b=3, c=2, d=2, e=1, f=1)* of the *monk-1* data set. The true class

for x is 1 because *e=1* and *a=b*. Thus, e and both a and b play an important role in the prediction. The MLP correctly predicts the class label as 1 with probability *p(class=1|x)* equal to 1. LACE local model returns the following CARs:

$$\{\text{e=1}\} \rightarrow \text{class=1}$$

$$\{\text{a=3, b=3}\} \rightarrow \text{class=1}$$

These local rules, based on our knowledge of the *monk-1* data set, actually capture the true explanation for instance x.

Given the relevant subsets obtained by the local rules, the prediction differences are shown in Figure 4.3a. The δ_{e} has a small contribution, while the other δ terms for single attributes and the $\delta_{a,b}(x)$ term have a null contribution. For this instance, the joint effect of attributes a, b, e drives the label assignment. For example, if both a and b are removed together, the class label is still predicted as 1 because of e. Consequently, also δ_a, δ_b, δ_e and $\delta_{a,b}$ are nearly 0. When removing the three attributes a, b, e, the prediction probability drastically changes. This is captured by the union itemset:

$$\{\text{a=3, b=3, e=1}\}$$

Hence, $\delta_{a,b,e}$ allows us to effectively quantify the interaction of the three attributes that drives the label assignment.

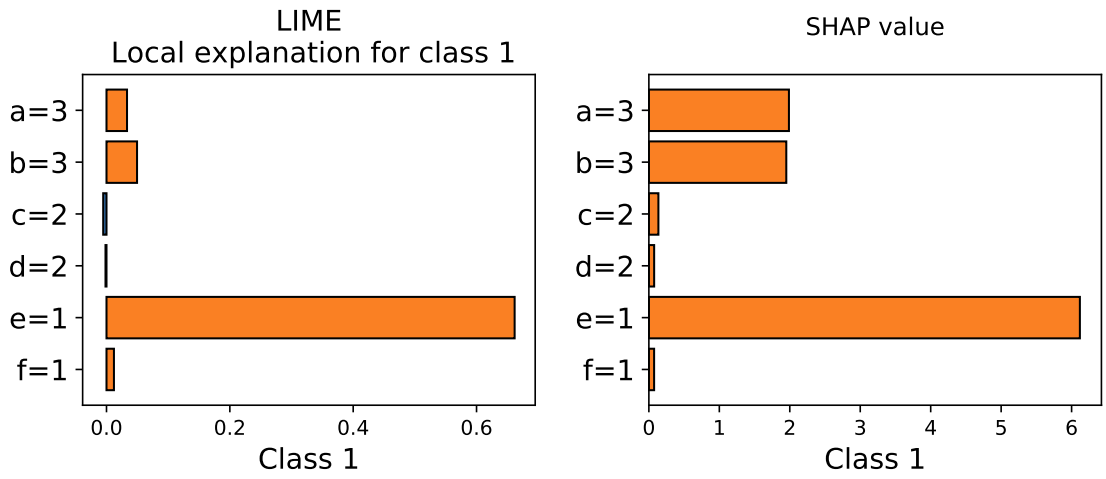
Consider now Figure 4.3b, which reports the explanation for the same instance x and class 1, but classified by the Naive Bayes classifier. LACE local model returns a single relevant rule:

$$\{e = 1\} \rightarrow class = 1$$

Hence, only the value of attribute e is considered relevant for the prediction. The Naive Bayes classifier assigns correctly the instance x to class 1, but only because *e=1*. The local model and the explanation highlight that the Naive Bayes classifier has not learned the association that if *a=b* then *class=1*. The Naive Bayes classifier, because of its assumption of independence between features, is not able to learn the importance that a and b jointly have. Hence, LACE local model and explanation in this case successfully reflect the model behavior.

This experiment highlights that different models work differently. This difference may not be directly visible from the prediction, because the predicted class is the same. The careful analysis of the prediction provided by LACE explanation allows understanding in depth the local behavior of different classifier models.

We compared LACE explanations with the three state-of-the-art approaches LIME [138], Anchor [139] and SHAP [104]. As LACE, these methods are model-agnostic prediction explainers, hence applicable for explaining the prediction of any classification model.



(a) LIME. (b) SHAP.

Figure 4.4: LIME (a) and SHAP (b) explanations of a MLP prediction of the *monk-1* dataset.

LIME learns a linear model in the locality of the instance x to explain. Differently from LACE, the locality is given by random local perturbations of instance x. An explanation is derived directly from the linear model, in form of weights for each feature value. As LACE, an explanation is provided with respect to a target class, by quantifying the relevance of attribute values for the prediction of the target class.

Consider the same instance x = *(a=3, b=3, c=2, d=2, e=1, f=1)* of the *monk-1* dataset. Figure 4.4a shows the explanation of the (same) prediction of the MLP model provided by LIME for class 1 [2]. The largest weight is given to term *e=1*, while terms *a=3* and *b=3* provide a very small contribution. LIME linear local model is not able to clearly capture the joint relevance of *a=3* and *b=3* for the prediction. Similar considerations hold for the explanations provided by LIME for predictions performed on different instances of the *monk-1* dataset. LIME is not able to highlight the information on the joint effect of feature values *a=b*. Hence, all predictions are only explained by the term *e=1* or $e \neq 1$, while for the terms a and b it detects a limited contribution.

SHAP [104] leverages the concept of Shapley value from coalition game theory [150]. The Shapley value is used to compute the contribution of each feature value to the prediction for a single instance. Figure 4.4b shows the explanation of the (same) prediction of the MLP model provided by SHAP for class 1 [2]. The

[2]We use a shared bar plot representation for ease of comparison.

explanation properly capture that the term *e=1* is relevant to the prediction, followed by the terms *a=3* and *b=3*. The contribution to the prediction that subsets of attribute values jointly have is combined in the SHAP values. However, this summarization of the attribute value contribution does not allow us to clearly understand the relevant association of attribute values. From the explanation, we cannot derive that the term *e=1* alone contributes to the prediction while the terms jointly a and b contribute. The local rules derived by LACE instead allow to clearly capture the associations.

We now qualitatively compare our explanation with *Anchor* [139]. Anchor explainer is an extension of the work in [138]. The explanations are provided in form of decision rules. An anchor is a rule that "anchors" locally the prediction and, thus, determines the prediction. Changes to the feature values not belonging to the rule do not change the final prediction [139]. We explain again the prediction performed by the MLP classifier for instance x of the *monk-1* dataset. Anchor explainer provides the following rule: $\{e = 1\} \rightarrow 1$. The rule meets the *Anchor* definition. A change of the other terms, thus also of *a=3* and *b=3*, does not cause a prediction change. However, the Anchor explanation, being a single rule, is not able to capture the complete behavior of the model, which is also driven by the joint values of attributes a and b.

4.4.2 Explanation validation

We evaluate LACE explanations in terms of cosine similarity, f1-score and rule hit, defined in 4.3.1. We compare the results in terms of cosine similarity and the f1-score of feature importance vector $f1\text{-}feature$ with LIME [138] and SHAP [104] and in terms of the f1-score of rule explanation $f1\text{-}rule$ and rule hit with Anchor [139]. We firstly discuss the results for artificial datasets. We then evaluate explanations for real dataset by comparing the results for injected behaviors and using white-box classifiers to derive ground truth explanations.

Explanation validation with artificial datasets

We run the experiments for the artificial datasets *chess*, *cross*, *group* and *group_10* with the *MLP* and random forest (*RF*) classifiers. We omit for this set of experiments the Naive Bayes classifier. As observed in Section 4.4.1, Naive Bayes may fail to capture the true associations within the dataset because of its assumption of independence between features. A good explanation should reflect the behavior of the classifier for the prediction under analysis. As a consequence, the true relationships cannot be considered also as the *true explanations* if the classifier fails to learn them. On the other hand, we observed a high accuracy on the test set of the artificial datasets for the *MLP* and *RF* classifiers. Therefore, we included only the MLP and RF classifiers and we used the true relationships of the datasets as

true explanations. LACE, Anchor [139] and LIME [138] discretize the dataset to produce explanation results. To enable a fair comparison that is not dependent on the internal discretization, for the following set of experiments we firstly discretize the dataset in a shared way.

We train the dataset on the *training* set. We then evaluate performance and explanations on the *explain* set. In particular, we randomly select N=100 instances of the *explain* set and generate explanations.

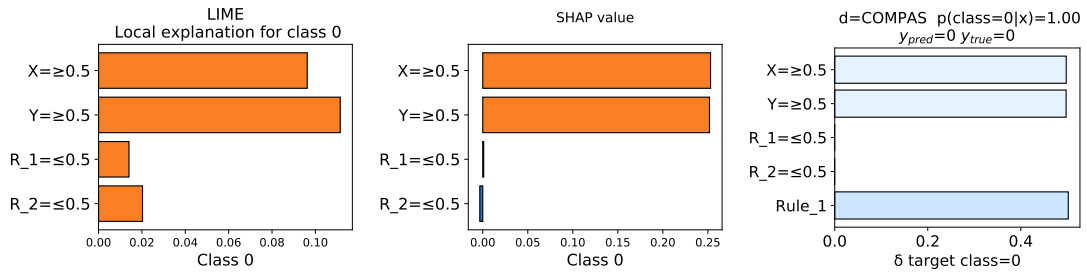
dataset	classifier	LACE	LIME	SHAP
chess_d	RF	**0.99996**	0.86489	0.99956
	MLP	**0.99996**	0.86451	0.99784
cross_d	RF	**0.99998**	0.98791	0.99980
	MLP	**0.99998**	0.98793	0.99905
groups_d	RF	**1.0**	0.97709	0.99987
	MLP	**1.0**	0.97711	0.99973
groups_10_d	RF	0.98250	0.69973	**0.99451**
	MLP	**1.0**	0.72695	0.99783

Table 4.2: Average feature similarity f-*sim* for N=100 explanations of the RF and MLP predictions of the artificial datasets.

dataset	classifier	LACE	LIME	SHAP
chess_d	RF	**1.0**	0.6667	0.6720
	MLP	**0.9240**	0.6680	0.6973
cross_d	RF	**1.0**	0.6667	0.6667
	MLP	**0.9460**	0.6667	0.6667
groups_d	RF	**1.0**	0.6680	0.6680
	MLP	**0.9280**	0.6680	0.6693
groups_10_d	RF	0.3345	**0.3346**	0.3345
	MLP	**0.6600**	0.3343	0.5002

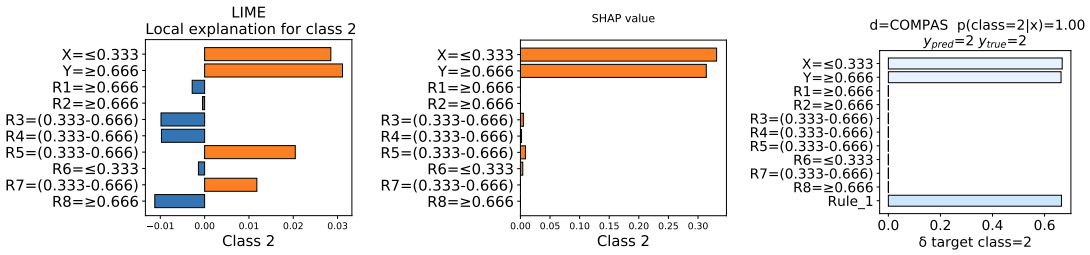
Table 4.3: Average feature f1-score $f1$-$feature$ for N=100 explanations of the RF and MLP predictions of the artificial datasets.

We first evaluate the quality of the feature explanation vectors returned by LACE and compare it with LIME and SHAP in terms of feature similarity f-*sim* and f1-score $f1$-$feature$. We recall from Section 4.3 that the artificial datasets are designed with 2 relevant features X and Y while the other attributes are completely random and not associated with the class label. We consider both features equally important. Hence, the true explanation in form of feature vector is represented by t=[0.5, 0.5, 0, 0] for the 4-dimensional dataset *cross*, *chess* and *group* and t=[0.5,0.5,0,..., 0] with $|t|$=10 for the group_10 dataset, where the first two



(a) LIME. (b) SHAP. (c) LACE.

Figure 4.5: LIME (a), SHAP (b) and LACE (c) explanations of a RF prediction of the *cross* dataset. Rule_1={X≥0.5, Y≥0.5} → 0.



(a) LIME. (b) SHAP. (c) LACE

Figure 4.6: LIME (a), SHAP (b) and LACE (c) explanations of a RF prediction of the *group_10* dataset. Rule_1={X≤0.333, Y≥0.666} → 2.

elements of t refer to the attributes X and Y. To compute the f1-score $f1\text{-}feature$, we map t to its binary representation where we have a one for each not-null element and zero otherwise.

The average cosine feature similarity $f\text{-}sim$ for the N explanations is reported in Table 4.2. The average feature similarity for LIME ranges from 0.7 to 0.99. LACE and SHAP instead generally correctly identify the relevant features as the ones with the highest relevance for all the considered datasets. For both explainers, the $f\text{-}sim$ is high, ranging from 0.98 to 1. In particular, LACE slightly outperforms SHAP in 7 out of 8 experiments.

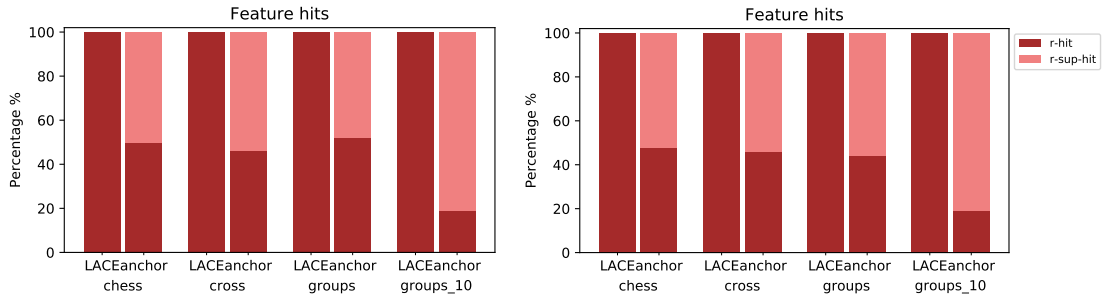
Table 4.3 shows the average f1-score $f1\text{-}feature$. LACE generally outperforms SHAP and LIME in the identification of the two relevant attributes. For the evaluation of the $f1\text{-}feature$ scores, we map each not null term $\hat{f}_i(x)$ to 1. Hence, even small contributions are mapped to 1 in the binary explanation. This is the reason for the lower performance in terms of $f1\text{-}feature$ of the three explainers, compared to the similarity-based evaluation.

To clarify this point, consider Figures 4.5 and 4.6 showing the explanations for two example instances of the *cross* and *group_10* dataset respectively for the LIME, SHAP and LACE. Even if LIME correctly identifies the attributes X and Y

as the most relevant terms, it assigns a (still significant) contribution also to other random features (Figures 4.5a and 4.6a). On the other hand, SHAP and LACE identify as only relevant terms the attributes X and Y (but we note that they assign a really small and negligible feature importance to the random terms R_1 and R_2). However, differently than SHAP, LACE provides in addition to the feature importance vector also the local rules associated to the predictions. For the prediction reported in Figures 4.6c and 4.6c, we can observe that LACE also successfully identifies that the attributes X and Y *together* are relevant for the prediction. It identifies that $\{X \geq 0.5,\ Y \geq 0.5\} \rightarrow 0$ and $\{X \leq 0.333,\ Y \geq 0.666\} \rightarrow 2$ are the relevant rules for the prediction of the *cross* (Figure 4.5c) and *group_10* (Figure 4.6c) datasets respectively. Moreover, LACE also quantifies the prediction difference of the local rules and their values are reported in the bar plots (Figures 4.6c and 4.6c).

dataset	classifier	LACE	Anchor
chess_d	RF	**1.0**	0.85667
	MLP	**1.0**	0.88467
cross_d	RF	**1.0**	0.87733
	MLP	**1.0**	0.87733
groups_d	RF	**1.0**	0.87600
	MLP	**1.0**	0.87800
groups_10_d	RF	**1.0**	0.65959
	MLP	**1.0**	0.69481

Table 4.4: Average rule f1-score $f1\text{-}rule$ for N=100 explanations of the RF and MLP predictions of the artificial datasets.



(a) RF. (b) NN.

Figure 4.7: Rule hit percentage of LACE and Anchor visualization for the artificial datasets for the RF (a) and MLP (b) classifiers

We then evaluate the ability of LACE to capture the true associations in form of local rules. We compare the results with Anchor using the notions of f1-score of rule-based explanation $f1\text{-}rule$ and rule hit defined in Section 4.3.1. For the

artificial datasets, the true explanation in form of rule is represented by r={$X = x_i, Y = x_j$}$\rightarrow c$ since the class is determined by X and Y. To compute the $f1$-$rule$, we map rule r to its binary representation, where we have a one for the features present in the rules. As a result, $b(r)$=[1,1,0,0] for the 4-dimensional dataset *cross*, *chess* and *group* and $b(r)$=[1,1,0,..., 0] with $|r_b|$=10 for the *group_10* dataset. We similarly map the explanation rules $\hat{r}$ obtained by LACE and Anchor, setting to one the feature belonging to the returned rules and zero otherwise. To characterize rules, we compute the rule hit *r-hit* to evaluate if the rules highlighted by the explainers match completely or partially the true explanations. In the latter case, we expose if each rule is a subset or superset of a true explanation. For the artificial datasets, an explanation rule $\hat{r}$ as a *r-hit*=1 if it is composed by the attributes X and Y. A rule $\hat{r}$ is a *subset* of r if it identifies only X or only Y as relevant terms. Finally, a rule $\hat{r}$ composed by X and Y and one or more random attributes is a *superset* of the true explanation r.

Table 4.4 shows the average $f1$-$rule$ for $N = 100$ explanations for the RF and MLP predictions of LACE and Anchor. The results show that LACE outperforms Anchor in the identification of the true rules. To better characterize the type of rules determined by the two explainers, we summarize the type of match in Figure 4.7. The bar plot representation shows the average rule hit *r-hit* and the partial rule hits *r-sup-hit* of the N explanations for the RF and MLP classifiers. LACE successfully captures the relevant associations. It identifies the attributes X and Y for all the N explanations with a average percentage r-hit of 100% for all the experiments. On the other hand, the average *r-hit* of Anchor ranges from 19% to 52% for the RF classifiers and from 19% to 48% for the MLP classifiers, with an average value of 41.75% and 39.25% respectively. The approach Anchor for about half of the explanations finds supersets of the relevant features (*r-sup-hit*). The erroneous identification of supersets of the true explanation rules is increased for the dataset *groups_10*. In this artificial dataset, the number of attributes is higher.

The experimental results on the artificial dataset show that LACE identifies the relevant associations. Differently from Anchor, it additionally provides the results in form of feature relevance. LACE combines the strengths of feature relevance-based explanations and rule-based ones, achieving accurate results.

4.4.3 Explanation hit for real datasets

We evaluate the quality of explanations on the real dataset *COMPAS*. Being a real dataset, the *true explanations* is not known a priori. Hence, we cannot directly validate explanations with the metrics illustrated in Section 4.3.1. For validation purposes, we design two experimental settings. The first analysis consists of injecting a controlled behavior in the classifier. We can then compare the injected associations with explanations. In the second study, we consider a white-box classifier as model f to explain. The ground truth explanation can then be derived

directly from the white-box model. In the following, we discuss the results for the two listed sets of experiments.

Injecting a controlled behavior in classifiers.

We performed a controlled set of experiments in which we artificially injected a controlled behavior in classifiers. Specifically, let be $I=\{\alpha_i, ..., \alpha_k\}$ an itemset and G_I the subgroup of instances satisfying I. In the training set, we modify the class labels of the subgroup G_I, setting them to a target class c. As a result, the trained model will learn the association of the attribute values $\alpha_i, ..., \alpha_k$ and the class c. For all the instances to explain matching I, we consider $\alpha_i, ..., \alpha_k$ as the attributes values in the *true explanation vector* t and $I \rightarrow c$ as the *true local rule* r. For the feature explanation vector, we consider the attributes in I to be equally important.

For the evaluation, we used the *COMPAS* dataset and the *RF* classifier. We repeated the process for 20 random itemsets of length 2 to 3. For each experiment, we generate the explanations for N=100 random instances of the explain set matching the itemset I. We compute the cosine similarity and f1-score for the feature vector and the f1-score and the rule hits for the rule explanations and we averaged the results for the 20 experiments.

	$f\text{-}sim$		
	LACE	**LIME**	**SHAP**
min	0.81	0.44	0.69
max	0.98	0.95	0.97
mean	**0.93**	0.75	0.89
75%	0.96	0.84	0.95
std	0.05	0.14	0.07

Table 4.5: Summary of cosine similarity results $f\text{-}sim$ for the 20 injected behaviors for N=100 *feature vector* explanations of COMPAS dataset (RF classifier).

	$f1\text{-}feature$		
	LACE	**LIME**	**SHAP**
min	0.51	0.5	0.5
max	0.73	0.5	0.5
mean	**0.60**	0.5	0.5
75%	**0.64**	0.5	0.5
std	0.06	0.0	0.0

Table 4.6: Summary of f1-score $f1\text{-}feature$ for the 20 injected behaviors for N=100 *feature vector* explanations of *COMPAS* dataset (RF classifier).

	precision			*recall*		
	LACE	**LIME**	**SHAP**	**LACE**	**LIME**	**SHAP**
min	0.35	0.33	0.33	0.98	1.0	1.0
max	0.58	0.34	0.33	1.0	1.0	1.0
mean	**0.43**	0.33	0.33	**1.0**	**1.0**	**1.0**
75%	0.47	0.33	0.33	1.0	1.0	1.0
std	0.06	0.00	0.00	0.01	0.0	0.0

Table 4.7: Summary of precision and recall for the 20 injected behaviors for N=100 *feature vector* explanations of *COMPAS* dataset (RF classifier).

The summary of the results for the feature cosine similarity f-*sim* is reported in Table 4.5. Specifically, the summary illustrates the minimum, maximum, average, 75% percentiles, and standard deviation of the average f-*sim* for the 20 experiments. The results show that LACE outperforms LIME and SHAP. Our approach has the highest minimum, average and 75% percentiles value for both the f-*sim*. LACE has also the lower standard deviation. The evaluation demonstrates the effectiveness of LACE in identifying the correct features that determine the prediction.

The summary of the results in terms of f1-score $f1$-$feature$ is reported in Table 4.6 . For the explainers, the $f1$-$feature$ ranges from 0.5 to 0.73. To better understand the (on average) low $f1$-$feature$ we also report the summary of the results for the precision and recall (Table 4.7). All three explainers recognize the injected terms as relevant, with a recall on average of 1. On the other hand, they also consider others terms as important for the prediction. We remark that we controlled the behavior for only one subset of features, leaving the rest unaltered. Hence, other features may still be considered relevant for the prediction by the classifier.

	LACE	**Anchor**
min	0.93	0.81
max	1.00	1.00
mean	**0.99**	0.94
75%	1.00	1.00
std	0.02	0.06

Table 4.8: Summary of $f1$-*rule* for the 20 injected behaviors for N=100 *feature vector* explanations of COMPAS dataset (RF classifier).

We now evaluate the ability of the LACE and Anchor of capturing the true (injected) associations in form of local rules. The summary of the results for N = 100 explanations for the 20 experiments in terms of f1-score $f1$-*rule* are reported in

explainer		*r-hit*	*r-sup-hit*	*r-sub-hit*	*r-total*
LACE	**min**	**58.00**	0.00	0.0	95.00
	max	100.00	25.00	26.0	100.00
	mean	**85.70**	9.15	4.9	**99.75**
	std	**10.04**	6.82	7.6	1.12
Anchor	**min**	28.00	0.00	0.00	95.00
	max	100.00	72.00	42.00	100.00
	mean	77.35	19.95	2.40	99.70
	std	23.96	24.10	9.34	1.13

Table 4.9: Rule hits statistics for 20 injected behaviors for the COMPAS dataset (RF classifier).

Table 4.8. LACE achieves the best results, with the highest average $f1$-*rule* (0.99% compared to 0.94%). We also evaluate the rules in terms of rule hit to understand the type of rules returned by the two explainers. The summary of the results in terms of *r-hit*, *r-sup-hit* and *r-sub-hit* are reported in Table 4.9 (the last column *r-total* is the sum of the three terms). LACE, on average, more often identifies fully the injected behavior, with a mean *r-hit* equal to 85.7% (compared to 77.35%) and lower standard deviation (10.04% compared to 23.96%). The explainer Anchor tends more than LACE to recognize a superset of the true local rules as important, with a *r-sup-hit* on average higher (19.95 compared to 9.15). Hence, our approach identifies with a higher rate all and only the terms in the true local rules.

Evaluation with white-box models.

For the following experiments, we leverage the decision tree as classifier f. For this model, the actual explanation of the prediction is already available, because it is the decision path from the root to the leaf node. Hence, we can compare the explanations provided by different explanation methods with the actual path in the tree, which is the correct explanation for this model.

As discussed in Section 2.1.1, the interpretability of a decision tree highly depends on its size [59] since it is a proxy of model complexity. The greater is the size, the less the model is considered understandable. For classification decision trees, the size can be estimated by the number of nodes or by their depth. Hence, in the experiments, we limited the depth of the decision trees and we studied the impact of the depth on explanation quality.

Specifically, we train a decision tree classifier with default parameters and a maximum depth *dep* on the training set. We then explain N=100 instances of the explain set. For each explanation, we consider as true explanation its decision path from the root to the leaf node. In particular, for estimating the *f1-score* for an instance x, we considered as relevant attributes the ones appearing in the decision path. We consider the true explanation t as a binary explanation vector indicating

the presence or absence of the attribute as in [81]. Hence, for this analysis, we do not consider the cosine similarity of feature vectors. Hence, the true explanation is a binary explanation $t(x) = t_1, ..., t_d$ where t_i is 1 if the i-th attribute belong to the decision path, and 0 otherwise. For the rule hit, the decision path itself is consider as the *true* local rule r. The number of relevant features and the length of the local rule can be at most *dep*.

	LACE	**LIME**	**SHAP**
2	**1.0**	0.418	0.870
3	**1.0**	0.514	0.810
4	**1.0**	0.573	0.572
5	**1.0**	0.688	0.688
6	**1.0**	0.777	0.775

Table 4.10: Average feature f1-score $f1\text{-}feature$ for N=100 explanations of decision tree predictions for the *COMPAS* varying the depth of the tree.

	LACE	**Anchor**
2	**0.866**	0.857
3	**0.872**	0.812
4	**0.729**	0.642
5	**0.768**	0.665
6	**0.772**	0.687

Table 4.11: Average rule f1-score $f1\text{-}rule$ for N=100 explanations of decision tree predictions for the *COMPAS* varying the depth of the tree.

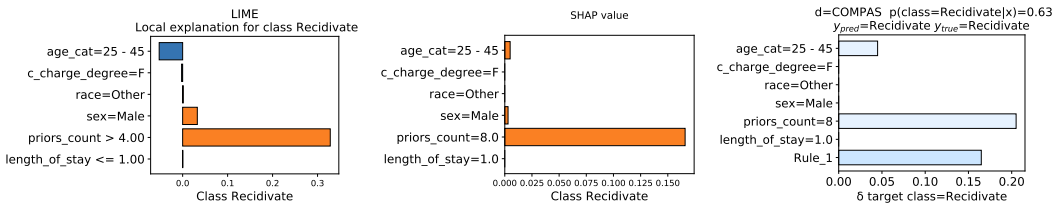
We repeat the experiments varying the depth size *dep* from 2 to 6. The average results for the N=100 explanations with respect to the f1-score $f1\text{-}feature$ are reported in Table 4.10. LACE outperforms LIME and SHAP in all experiments in terms of $f1\text{-}feature$. The results show that LACE is the best performing algorithm in capturing the relevant features belonging to the decision path.

We now compare LACE with Anchor in terms of the f1-score of the rule-based explanation and rule hit. Table 4.11 shows the average $f1\text{-}rule$ for N explanations for each evaluated depth size. The average results varying the depth size in terms of *r-hit*, *r-sup-hit* and *r-sup-hit* are shown in Table 4.12 and the last column *r-total* is the sum of the three terms. Our approach outperforms Anchor both in terms of $f1\text{-}rule$ (Table 4.11) and *r-hit* (Table 4.12).

Consider the decision tree classifier (DT) trained with depth=3 and a random instance x={*age_cat=25 - 45, c_charge_degree=F, race=Other, sex=Male, priors_count=8, length_of_stay=1.0*} of the *explain* set. The true explanation of the prediction is derived from decision path itself. For instance x the path is the

	depth	*r-hit*	*r-sup-hit*	*r-sub-hit*	*r-total*
LACE	**2**	**65**	15	19	99
	3	**60**	9	29	98
	4	**30**	8	60	98
	5	**26**	9	63	98
	6	**16**	9	75	**100**
	avg	**39.40**	10.00	49.20	98.60
Anchor	**2**	63	17	20	**100**
	3	42	26	32	**100**
	4	13	16	71	**100**
	5	8	12	80	**100**
	6	9	1	90	**100**
	avg	27.00	14.40	58.6	**100.0**

Table 4.12: Rule hit results for N=100 explanations of decision tree predictions for the *COMPAS* varying the depth of the tree.



(a) LIME. (b) SHAP. (c) LACE

Figure 4.8: LIME (a), SHAP (b) and LACE (c) explanations of a *DT* prediction of the *COMPAS* dataset. Rule_1={age=25 - 45, #priors=8} → Recidivate.

following: *#priors>2.5 & #prior<=8.5 & age!=Greater than 45 then class = 'Recidivate'*. From the path we derive that *priors_count=8* and *age_cat=25 - 45* are the attribute values with not-null importance in the feature important vector $t(x)$ and $\{$*age=25 - 45*, *#priors=8*$\}$ → *Recidivate* is the only true rule $r(x)$.

The explanations of the *DT* prediction of x for LIME, SHAP and LACE are reported in Figures 4.8a, 4.8b, 4.8c respectively. The Anchor rule for the same instance is the following: *#priors* > 4.00 AND *age = 25 - 45*. LIME identifies the count of prior offenses as relevant but it assigns a negative contribution to the age between 25 and 45. SHAP correctly identifies that both the number of prior and the age between 25 and 45 are significant to the prediction, while it assigns a real small contribution to the term *sex=Male* that do not belong to the decision path. However, SHAP does not clear highlight the relevance that the age and the prior count have together. This is captured by both Anchor and LACE methods. Again, LACE is able to provide both information. Our method identifies the true

path in form of local rule Rule_1={*age=25 - 45*, *#priors=8*} → *Recidivate* and the terms *age=25 - 45* and *#priors=8* are the only relevant ones for the prediction difference (Figure 4.8c).

4.4.4 Analysis of the neighborhood tuning.

		RF	NN	NB
cross_discrete	$diff_\epsilon$	100.00	100.00	99.50
	mean (*min-max*)	5.91 (5-7)	5.91 (5-7)	5.91 (5-7)
chess_d	$diff_\epsilon$	0.00	0.00	24.82
	mean (*min-max*)	1.0 (1-5)	1.0 (1-5)	1.99 (1-5)
groups_d	$diff_\epsilon$	83.00	83.00	84.58
	mean (*min-max*)	1.83 (1-2)	1.83 (1-2)	1.85 (1-2)
groups_10_d	$diff_\epsilon$	0.27	0.00	0.05
	mean (*min-max*)	9.0 (1-16)	1.0 (1-16)	1.65 (1-16)
monks	$diff_\epsilon$	9.41	9.82	27.27
	mean (*min-max*)	5.35 (1-9)	5.46 (1-9)	5.87 (1-9)
adult	$diff_\epsilon$	37.62	39.15	68.90
	mean (*min-max*)	21.33 (2-35)	20.98 (2-35)	26.11 (2-35)
COMPAS	$diff_\epsilon$	74.09	86.35	77.05
	mean (*min-max*)	8.65 (1-13)	9.6 (1-13)	11.61 (1-13)

Table 4.13: Impact of the automatic tuning approach on the locality approximation and on the number of iterations.

We analyze the impact of the neighborhood on explanation results. We consider the 5 artificial datasets and the 2 real datasets *adult* and *COMPAS*. We apply LACE on N=100 randomly selected instances of the explanation sets and we evaluate the improvement in terms of local approximation ϵ introduced by the heuristic approach. Table 4.13 summarizes the results. For each data set, the average (*mean*), minimum and maximum (*min-max*) number of iterations required by the automatic approach for tuning the value of K are reported. The number of iterations depends on the dataset and the classifier to explain.

The table then reports the effect of the automatic tuning approach on the locality approximation. We compare the average approximation obtained when only the heuristic approaches for selecting the parameter K is applied with the one obtained with the proposed automatic approach. For each data set, we report the average difference, referred in Table 4.13 as $diff_\epsilon$. The automatic tuning approach yields an average locality approximation value reduction equal to 47.8%.

4.5 Remarks and conclusions

In this chapter, we have presented LACE, a model-agnostic explanation method to explain individual predictions of any classifier [129]. The original classification model is treated as a black box. The influence of both single attributes and attribute subsets on the prediction for specific instances is quantified by omitting attribute sets and measuring the prediction change. We overcome the exponential time complexity that derives from the computation of the power set of the feature values by learning a local model. The local model is an associative classifier that is learned in the locality of the instance whose prediction is to be explained. It returns the subsets of feature values that are relevant for that particular prediction. Only these subsets are omitted.

The LACE explanations are in form of feature importance vectors and local rules. The feature importance provides a quantitative understanding of the model behavior indicating how each attribute value individually influences the prediction. The local rules capture the relevant association of feature values to the class assignment.

We conducted a wide range of experiments to assess the quality of the provided explanations. Experiments performed both on synthetic and real-world datasets highlighted the ability of the LACE explanation method to capture the relevant attribute subsets that jointly contribute to a single instance prediction.

Chapter 5

X-PLAIN: An interactive framework to inspect model behavior

Machine learning models are increasingly adopted to assist human experts in decision-making. Especially in critical tasks, understanding the reasons behind model predictions is essential for trusting the model itself. Investigating model behavior in an interactive way can provide actionable insights. For example, experts can test their hypothesis on the model inner working, detect model wrong behaviors and actively work on model debugging and improvement. Unfortunately, the black-box nature of most high-performance models hinders the exploration and the understanding of model behavior.

This chapter presents X-PLAIN [127], an interactive tool that allows human-in-the-loop inspection of classifier reasons behind predictions. X-PLAIN leverages on LACE, illustrated in Chapter 4, as explanation method [129] and exploits its properties to enable a comprehensive analysis of model behaviors. X-PLAIN directly addresses relevant desiderata of explainable AI research outlined in Section 1.1 as interactivity. Its support for the local analysis of individual predictions enables users to inspect the local behavior of different classifiers and compare them. The interactive exploration of prediction explanation provides actionable insights for both trusting and validating model predictions and, in case of unexpected behaviors, for debugging and improving the model itself.

The chapter is organized as follows. Section 5.1 outlines the specification of the interactive tool. Section 5.2 illustrate the key functionalities of X-PLAIN. Finally, Section 5.3 draws the conclusions.

5.1 Tool specifications

The X-PLAIN tool is implemented as a web app [127]. The source code is available at [1]. The back-end, which implements the data access layers and the analysis of model behavior, is written in Python. It relies on the *Flask* framework [65]. The analysis operations are implemented on top of the Pandas library for dataset processing [109], and the scikit-learn library for data mining [131]. X-PLAIN tool leverages on LACE as explanation method and on its Python implementation [129]. The front-end is written using *React.js* [137], a JavaScript library for building user interfaces.

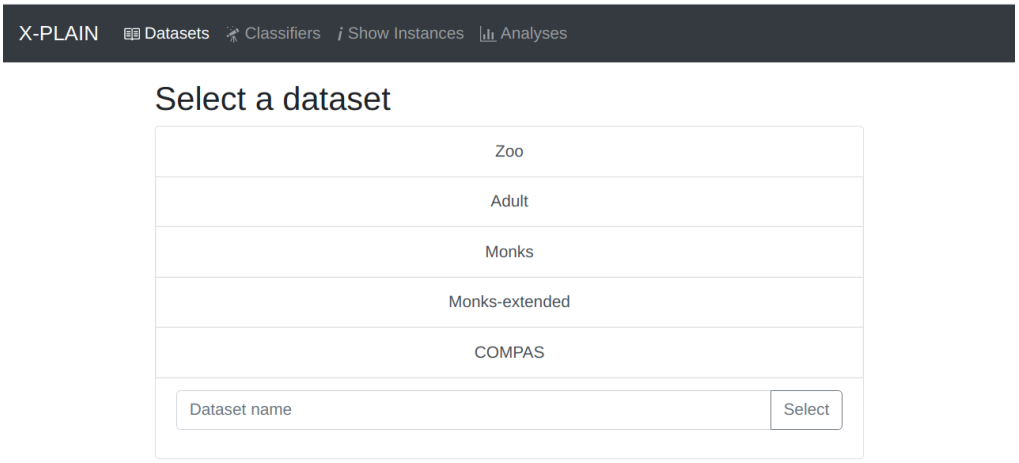


Figure 5.1: Dataset

The X-PLAIN [127] interactive tool focuses on local interpretability for structured (i.e., tabular) data. As a first step, the users specify the structured dataset they want to analyze by selecting an existing one or uploading it, as depicted in Figure 5.1. We have already included in X-PLAIN several datasets so that the users can easily try the tool. As a second step, we have the specification of the classifier. X-PLAIN is model agnostic. Hence, it provides explanations and local inspection for individual predictions of any arbitrary classifier. X-PLAIN requires to access to classifiers to inspect its predictions since the model is used as an oracle. The user can specify the classifier and perform the training. In this case, data set is split into *training* and *explain* set. The training set is used to train classification models. Explanations are produced for instances in the explain set. Alternatively, the user can upload a trained classification model (in pickle format) and the dataset will be considered as *explain* set. As a result, X-PLAIN shows the data instances and the labels predicted by the classification model.

[1] https://github.com/elianap/X-PLAIN-Demo

The user can hence start with the analysis of the behavior of the classification model. The X-PLAIN UI for the selection of the analysis to perform is shown in Figure 5.2. In the following section, we illustrate the key functionalities of X-PLAIN.

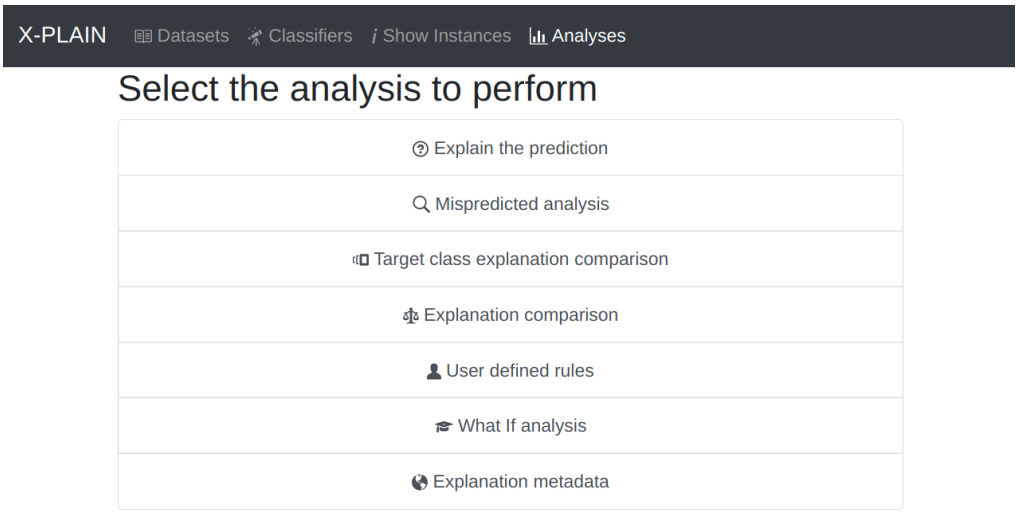


Figure 5.2: X-PLAIN UI - Selection of the dataset.

5.2 Key functionalities of X-PLAIN

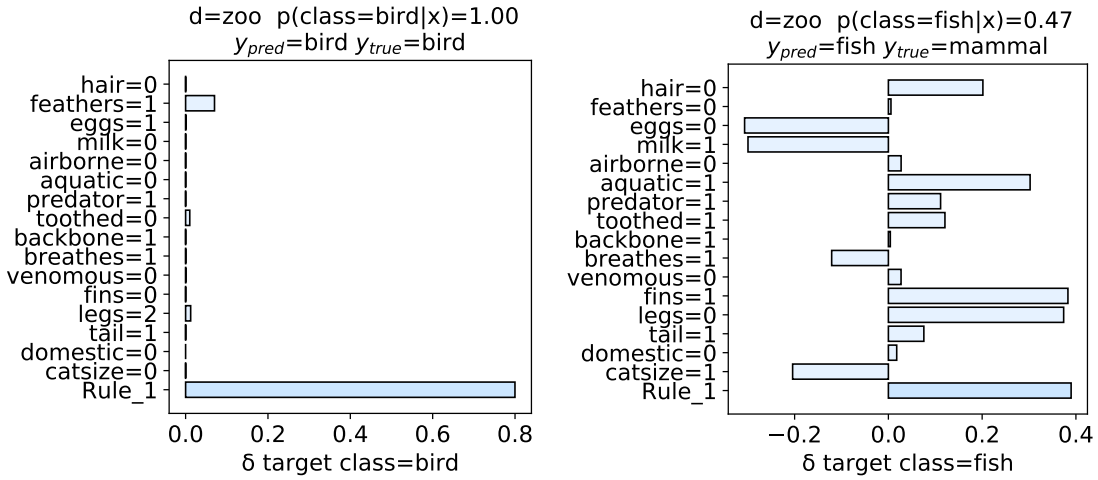
X-PLAIN is an interactive tool that supports local inspection of individual predictions of any classifier for structured data. X-PLAIN can help data analysts and domain experts in understanding the reasons behind specific predictions, inspecting incorrect or unexpected behaviors, and comparing what different classifiers have learned for each different class of the problem under analysis. In the following the key functionalities of the X-PLAIN interactive tool are presented.

Explanation of an instance prediction. X-PLAIN allows the evaluation of attribute value importance for the prediction of each class label, both for correct and mispredicted instances. This feature also enables the comparison of the local behavior for multiple target classes and classifiers.

Human-in-the-loop model analysis. Users may actively speculate and analyze their assumptions on the local model behavior based on their prior domain knowledge and perform what-if analysis by tweaking attribute values of single instances.

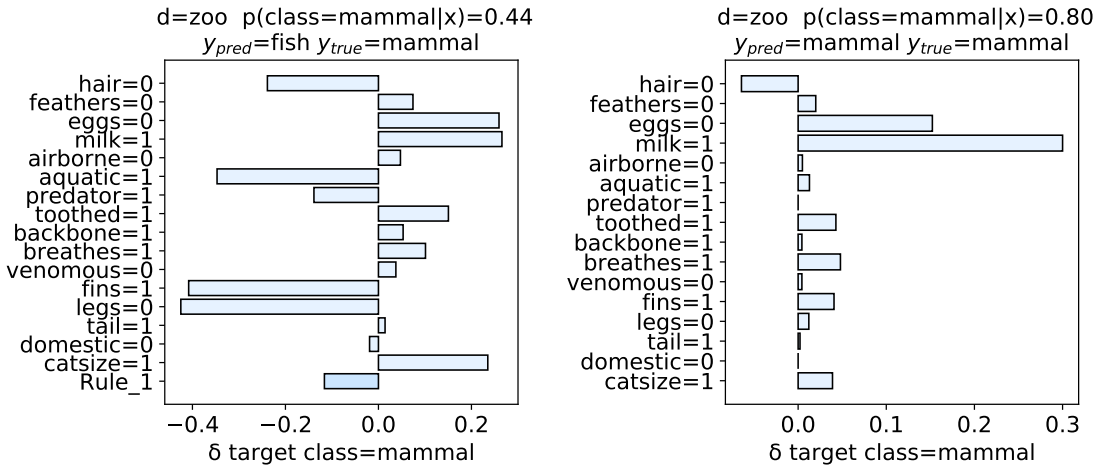
Explanation metadata analysis. The explanations provided by X-PLAIN provide actionable metadata that can be collectively exploited to characterize the global model behavior.

5.2.1 Explanation of an instance prediction



(a) NB example of correct prediction (b) NB example of incorrect prediction

Figure 5.3: Example of explanations for the NB correct prediction of the instance x=kiwi (a) and incorrect prediction y=porpoise (b) of the *zoo* dataset with respect to the predicted class.



(a) NB (b) RF

Figure 5.4: Example of explanation comparison for the prediction of the instance y=porpoise of the *zoo* dataset for the NB (a) and RF (b) classifiers with respect to the *mammal* class.

X-PLAIN leverages on LACE to generate explanations [129]. Hence, given an instance x belonging to the *explain* dataset it provides the prediction explanation

with respect to any arbitrary target class c. The explanation captures what model f has learned in the locality of x for class c in terms of local rules and prediction difference.

X-PLAIN exploits two important properties of LACE to provide explanation-based analyses. The first property is that the explanations are class-dependent. As a result, for an instance x we can observe which are the relevant attribute values and subsets of attribute values for each target class of interest. This is particularly interesting during the validation phase of a classification model for misclassified instances. Varying the target class from the predicted to the true class, we can study why the instance was incorrectly classified. The second property is the model agnostic nature of LACE. Hence, we can compare the explanations of the same instance for multiple classifiers.

In the following, we outline the variety of explanation-based analyses offered by X-PLAIN.

Explanation of correctly classified instances

The explanation of a correctly predicted instance highlights why the classifier has made that particular choice. Users can inspect the motivation behind the prediction. Hence, X-PLAIN supports GDPR compliant (*ex-post*) explanations by providing "meaningful information about the logic involved" [64]. Furthermore, the user can compare the explanation with her prior domain knowledge and determine if the model is "right for the right reasons".

An example of inspection of the reasons behind a correct prediction is presented in Figure 5.3a for the prediction of instance *x=kiwi* of the *zoo* dataset made by a Naive Bayes (NB) classifier. The *zoo* data set belongs to the UCI repository [96]. The classification task is the identification of the biological class of animals, based upon its 16 variables. The NB classifiers correctly assign the animal *kiwi* to the bird class. The only local rules is {*hair=0, feathers=1, eggs=1, milk=0, toothed=0, backbone=1, breathes=1, venomous=0, fins=0, legs=2, tail=1*} → class=*bird*. The confidence of the rule is 100% with support equal to 0.24. Hence, 24% of instances in the locality of the prediction are characterized by these attribute values and for all of them the predicted class is the bird class. In terms of individual attribute value contribution, we can observe from the prediction difference reported in Figure 5.3a that having feathers, being toothless, and having two legs are the terms that mostly contribute to the assignment of the bird class.

Explanation of mispredicted instances

The X-PLAIN tool allows interactively inspecting the classifier behavior for misclassified instances. The explanation highlights the reasons why the classifier wrongly assigned the class label to a particular instance. The user can interactively

select an incorrect prediction to inspect and target class c and the corresponding explanation is presented. Domain experts can inspect it and detect if the model has learned wrong associations. Hence, explanations (a) allow experts to comprehend why decisions are made, (b) enable model debugging and (c) foster model improvements in the case of model incorrect behaviors.

An example of misprediction inspection is presented in Figure 5.3b for the prediction of instance *y=porpoise* of the *zoo* dataset made by the Naive Bayes (NB) classifier. The *porpoise* is an aquatic mammal. However, the NB classifier incorrectly assigns instance *y=porpoise* to class *fish*. By exploiting X-PLAIN, users can inspect the reasons behind the wrong assignment. The extracted local rule is *{hair=0, feathers=0, airborne=0, aquatic=1, toothed=1, backbone=1, fins=1, legs=0, tail=1}* $\rightarrow$ *class='fish'*. The quantitative explanation, reported in Figure 5.3b, highlights that the assignment to the fish is driven by having fins and not legs, followed by being an aquatic animal with no hairs.

Compare the behavior for multiple target classes

X-PLAIN leverages the property of LACE of providing class-dependent explanations to inspect the model behavior of the classifier for multiple target classes. Explanations of instance x prediction provided by the same model f for different target classes can be visually compared. Users can inspect and compare the subsets of attribute values that are critical and significant for each analyzed target class. The analysis is particularly interesting, during the phase of model validation, in case of misclassified instances. The explanation with respect to the true label highlights which attribute values have a negative influence on the true label assignment.

Consider again instance *y=porpoise*. A user may be interested in investigating why the NB classifier does not assign instance y to the *mammal* class. By interactively selecting the different target classes, users can obtain the explanation computed with respect to mammal class. The explanation, reported in Figure 5.4a, highlights as terms that have the most negative influence in the assignment to class *mammal* the characteristics of having fins and no legs and being aquatic. The characteristic of producing milk, one of the main distinctive features of mammals, has a positive influence on the mammal class. However, the NB classifier does not assign to this attribute value enough importance to drive the prediction. Hence, a user can carefully inspect the motivations for different classes and evaluate if the model under analysis indeed captures the distinguishing characteristics of the studied problem.

Compare the behavior for multiple classifiers

X-PLAIN, as its integrated explanation method LACE, is model agnostic. Hence,

it provides local inspection for individual predictions of any classifier. Explanations of the same instance x made by different classifiers allow users to easily compare what the different models have learned. The comparison of the local behaviors may be exploited by domain experts to select the model that best fits a specific purpose. Users may also select which model prediction to trust based on their prior domain knowledge of the problem.

As an example, we consider the prediction of instance y = *porpoise* by a Random Forest model (RF). The RF model correctly identifies instance y as belonging to class *mammal*. Figure 5.4b shows the explanation of instance y for the predicted class. The local rule highlighted by X-PLAIN is *{feathers=0, eggs=0, milk=1, toothed=1, backbone=1, breathes=1, venomous=0}* → *class=mammal*. The term with the highest positive prediction difference is the animal characteristic of producing milk. On the other hand, having no hair has a negative influence on the mammal class assignment. Based on our knowledge of the biological class mammal we can say that RF has captured distinctive characteristics of the mammal class.

5.2.2 Human in the loop explanation

Human-in-the-loop inspections allow users to test their assumptions on the model internal behavior by actively investigating the classifier behavior as follows.

User rule definition

A user may interactively obtain the prediction relevance of additional, user-defined, rules. Based on prior domain knowledge, a user may expect a combination of attribute values to be important for the considered prediction. X-PLAIN directly estimates the prediction difference for the new user rule(s) and includes the new terms in the bar plot representation.

By adding and quantifying the relevance of rules based on domain expert knowledge, domain specialists can confirm if the model has learned the important associations behind the prediction. A small user-rule prediction difference instead indicates that the user-defined subsets of feature values do not drive the prediction. Domain experts can inspect the reasons behind the prediction highlighted by explanations. By investigating them, they can decide if trusting the prediction or not. Moreover, experts may discover new patterns and associations from the motivation highlighted by explanations. As an example, consider again the prediction of the NB model for instance *y=porpoise*. We may be interested in investigating if NB, despite the wrong assignment, has learned some discriminant characteristics of the mammal class. Following the definition of the mammal biological class, we interactively define the new user rule *{toothed=1, backbone=1}*. X-PLAIN directly estimates the relevance of the subset, which is equal to 0.15. The value is slightly larger than the prediction difference of the two terms when considered alone. Hence, it shows that,

in the NB model, this attribute value combination has a (small) positive influence on the mammal class assignment.

What-if analysis on attribute values

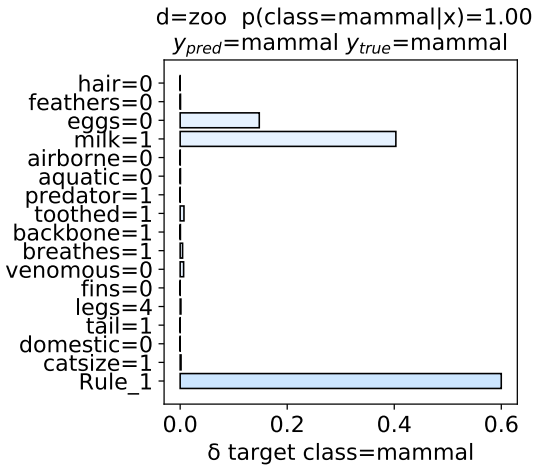


Figure 5.5: Example of what-if analysis. Explanation of tweaked instance y of the *zoo* data set for the NB prediction for *mammal* class.

What-if analysis allows users to examine how and why the prediction of x could change if some of its attribute values were different. Users can interactively change the value of one or more attributes at a time. X-PLAIN directly provides the explanation of the prediction for the instance with the perturbed attribute values. Users can inspect the changes in (i) predicted class label, (ii) local rules, and (iii) prediction differences of the perturbed instance. Hence, they can explore model f labeling behavior when the attributes of interest are replaced with user-defined values.

Consider again instance *y=porpoise* and the NB model. We analyze how the prediction and corresponding explanation would change if the three most discriminant negative terms highlighted by explanation in Figure 5.3b were different. We tweak the *fins*, *legs* and *aquatic* attributes, setting them to 0, 4 and 0 respectively. The resulting explanation computed for class *mammal* is reported in Figure 5.5 with local rule *{feathers=0, eggs=0, milk=1, toothed=1, backbone=1, breathes=1, venomous=0, fins=0}* → *'mammal'*. The prediction and the explanation drastically changes. The perturbed instance is assigned to class mammal. The explanation shows that *milk=1* and *eggs=0* are the only terms that individually influence positively the prediction.

5.2.3 Explanation metadata

Multiple local explanations generated by X-PLAIN may provide global insights on the model by highlighting which attributes and subsets of attribute values characterize the class assignment.

Explanation metadata provides a global understanding of the model behavior by averaging local explanations. Multiple local explanations can be combined to derive global insight into the model behavior [105, 138]. SHAP values in [105] are combined to obtain global explanations as average values for global feature importances, SHAP-based partial dependence plots, and summary plots. In [138], a global insight is provided through a submodular pick algorithm that selects a significant set of instances and their corresponding explanations. Unlike existing solutions, we propose multiple views of the model behavior from individual attributes to patterns.

Explanation metadata are generated by computing prediction explanations of model f for N instances of the *explain* dataset, considering as target class the predicted one, and stored in a knowledge base. Then, the average prediction difference is computed for (i) each attribute, (ii) attribute value, and (iii) pattern derived from local rules, separately for each target class. Finally, attribute value subsets are ranked based on average prediction difference. High-ranked combinations provide a description of the global model behavior for a given target class. By exploiting the metadata provided by a collection of prediction explanations, X-PLAIN may reveal which attribute, individual attribute values, or subsets are overall most discriminating for each class.

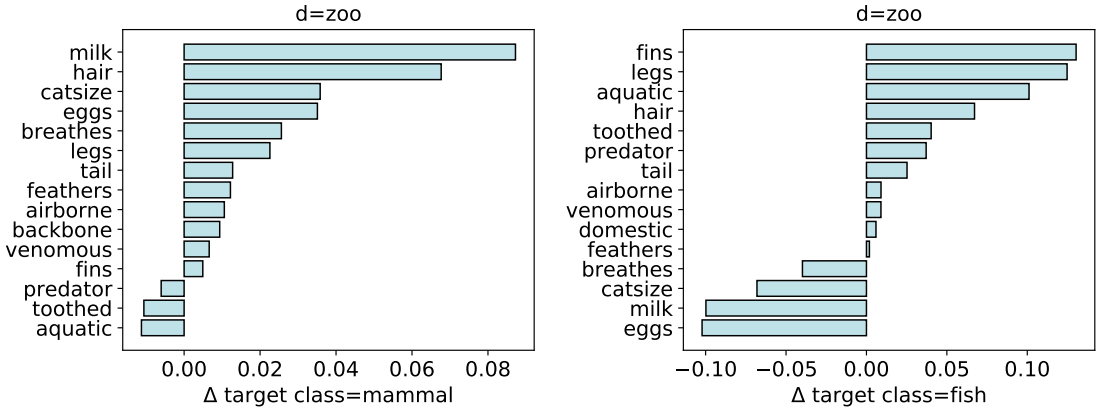
In the following, we formally illustrate the three views of the global model behavior: attribute, item (attribute value), and local rules views.

Attribute view. The attribute view provides the indication of which attributes are mostly considered for the prediction of a particular target class. The attribute-based metadata, denoted as *attribute-meta*, is computed by averaging the prediction differences with respect to a target class c:

$$\textit{attribute-meta}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} \delta(x_j) \tag{5.1}$$

where N_c is the number instances randomly sampled from the *explain* dataset assigned to the class c and $\delta(x_j)$ is the prediction difference for the instance x_j computed with respect to class c. $\delta(x_j)$ is the d-dimensional vector (where d is the number of attributes $A_1, \ldots, A_d$) that indicates the influence of each attribute values for the prediction of x_j. The d-dimensional vector $\textit{attribute-meta}_c$ provides the relevance of each attribute A_i to the prediction.

As an example, consider the explanation metadata of the *zoo* dataset for a NB model and set N as the cardinality of the explain dataset. Two examples of attribute view metadata are reported in Figure 5.6, where only the top-15 terms



(a) Attribute view - mammal

(b) Attribute view - fish

Figure 5.6: Example attribute view for the predictions of the *zoo* dataset for the NB classifier with respect to the *mammal* and *fish* classes.

are shown for visualization purposes. When selecting as target class the *mammal* one, X-PLAIN indicates that the most distinctive attribute is *milk*, followed by *hair* while the attributes aquatic and toothed have in general a negative contribution (Figure 5.6a). If we select fish as target class (Figure 5.6b), we obtain that *fins* and *legs* are the most distinctive attributes while *eggs* and *milk* have a negative influence.

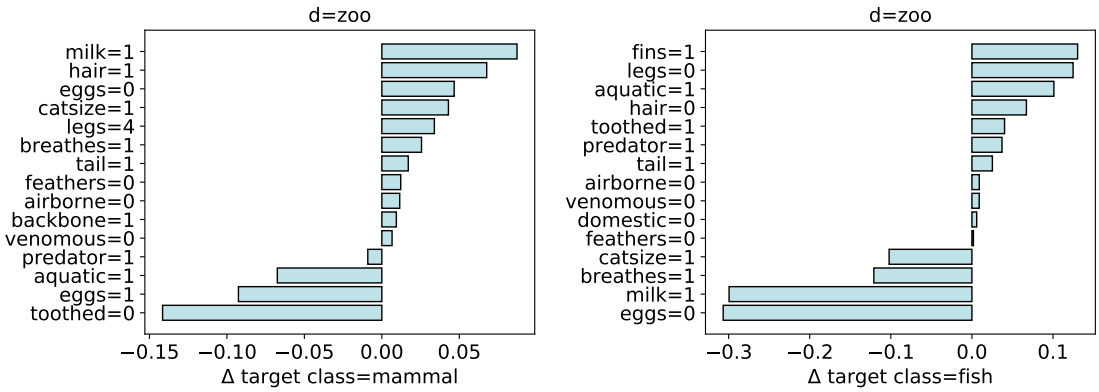
Item view. The item view reveals which attribute values are relevant for the prediction of a specific target class c.

Let $m_{A_i} = |\mathcal{D}_{A_i}|$ be the cardinality of the domain of each attribute $A_i \in \{A_1, \ldots, A_d\}$. In the case of continuous domain, the attributes are discretized before the item view is computed. We are interested in understanding which attribute values drive the prediction. Hence, focusing on ranges of attribute values instead of distinct numerical values can help the user exploration on only the distinctive terms. We remark that the discretization is only for explanation analysis purposes and that the classifiers and the explanations are defined on the original continuous domain. Each prediction difference vector $\delta(x_j)$, with $j = \{1, \ldots, N\}$, is mapped to a m_D-dimensional vector, where $m_D{=}\prod_{A_i \in A} m_{A_i}$. We refer to the mapped prediction difference as δ'. As a result, the not null terms of $\delta'(x_j)$ correspond to the attribute values that are relevant for the prediction of the instance x_j. Finally, we average the mapped prediction difference δ' for N explanations of the *explain* dataset to obtain the item view *item-meta* with respect to the target class c:

$$\textit{item-meta}_c = \frac{1}{N_c} \sum_{j=1}^{N_c} \delta'(x_j) \tag{5.2}$$

where again N_c refers to the prediction assigned to the class c randomly sampled from the *explain* dataset. The item view shows the average contribution of attribute values with respect to the class c.

Consider again our running example. The item view for the *NB* classifier for the *mammal* and *fish* class are shown in Figure 5.7 (only the top-15 items are reported to ease the visualization). The most distinctive items for the mammal class are *milk=1*, followed by the term *hair=1* while being toothless and laying eggs have a strong negative contribution (Figure 5.7a). If the user selects the target class *fish*, X-PLAIN indicates that having fins, no legs, and being aquatic are the most distinctive terms, while not laying eggs and producing milk have a negative contribution to the assignment to the fish class (Figure 5.7b).



(a) Item view - mammal

(b) Item view - fish

Figure 5.7: Example of *item view* for the predictions of the *zoo* dataset for the NB classifier with respect to the *mammal* and *fish* classes.

Local rule view. The local rule view indicates which local rules mostly drive the assignment. It reveals which association of attribute values are generally associated with the class label. The local rule view is computed by averaging the prediction difference separately for each extracted local rule. The *meta local rules* are then sorted with respect to the highest average prediction difference and the top-K are shown to the user.

For our example for the *zoo* dataset, the most determining subset of attribute values for the *mammal* class are *{eggs=0, breathes=1, venomous=0, backbone=1, feathers=0, toothed=1, milk=1, hair=1}* and *{breathes=1, venomous=0, backbone=1, feathers=0, milk=1, hair=1}*. For the fish target class, *{ hair=0, backbone=1, feathers=0, aquatic=1, legs=0, tail=1, airborne=0, toothed=1, fins=1 }* is the most characterizing subset.

5.3 Remark and conclusions

The chapter illustrates X-PLAIN, an interactive framework to inspect model behavior via a comprehensive analysis of explanations of individual predictions [127]. X-PLAIN integrates LACE [129] as explanation method and it exploits its key features for analyzing predictions. The proposed tool enables the inspection of local explanations. The analysis of explanations can be performed with respect to multiple target classes and multiple classifiers for structured data. This enables understanding the key aspects learned by a generic classifier for each different target class. Moreover, domain experts can leverage the tool during the validation of the classification model itself for model testing and debugging.

X-PLAIN supports human-in-the-loop inspections of the model predictions. The users can actively query the model behavior by testing their assumptions on associations of attribute values and by performing what-if analysis. Finally, X-PLAIN summarizes multiple local explanations to derive explanation metadata. The explanation metadata provides global insights on the model behavior at different granularity, from the importance of attribute to subsets of attribute values. Hence, the interactive tool can help data scientists and domain experts to understand and interactively investigate individual decisions made by black-box models.

Chapter 6

DivExplorer: Understanding subgroup divergence via patterns

The chapter focuses on enhancing the understanding of model behavior from the data subgroup perspective. We present DivExplorer [126], a novel approach that identifies and characterizes data subgroups in which a model behaves differently.

We introduce the notion of *divergence* to estimate the different classification behavior in data subgroups with respect to the overall behavior. A subgroup is a subset of the data characterized by a *pattern*, i.e. a conjunction of attribute-value pairs. The *divergence* of the subgroup measures the difference in statistics such as false positive and false negatives on the subgroup compared to the entire dataset. We use the classification performance as a proxy of the behavior of classification models.

The identification of data subgroups in which a machine learning model performs differently is relevant in many applications such as model validation and testing [33, 145], model comparison [33, 85], error analysis [169] and evaluation of model fairness [26, 33]. The analysis of divergent subgroups provides indications of model behavior in data subgroups. Moreover, divergence exploration can reveal in which subgroups a model performs poorly, helping data scientists in model debugging. It may also reveal if divergence from the overall behavior occurs for sensitive attributes.

As an example, consider the *COMPAS* dataset [8] containing demographic information and criminal history of defendants. For each criminal defendant, the *COMPAS* score of recidivism risk assesses the defendant's likelihood of committing another offense in a period of two years. *COMPAS* scores are determined by a proprietary algorithm and we do not have access to its inner workings. We compare the predicted recidivism rate with the actual one. The overall false positive (FPR) and false negative (FNR) rates are 0.088 and 0.698, where the positive class indicates being a recidivist. However, the rates are different when subgroups are considered (see Table 6.1). The subgroup identified by pattern (*age=25-45, #prior>3,*

Itemset	
age=25-45, #prior>3, race=African-Am, sex=Male	FPR=0.308
age>45, race=Caucasian	FNR=0.929
race=African-Am, sex=Male	FPR=0.150
race=African-Am, sex=Male, #prior>3	FPR=0.267
race=African-Am, sex=Male, #prior=0	FPR=0.097

Table 6.1: Example of patterns in the *COMPAS* dataset, along with their FPR or FNR. The overall FPR and FNR are 0.088 and 0.698.

race=African-American, sex=Male) has a FPR equal to 0.308. Instances belonging to this data subset tend to be wrongly assigned to high risk of recidivism with a rate higher than the dataset overall. On the other hand, the FNR of the pattern (*age>45, race=Caucasian*) is 0.929, indicating that Caucasians with age greater than 45 tend to be wrongly labeled with a low risk of recidivism more than in the dataset overall.

Several existing approaches that explore differences in subgroup behavior [19, 85] require users to specify the attributes or attribute values of interest. An overview of these approaches is presented in Section 2.2.1. They require human expertise, and this may hinder the identification of unexpected and previously unknown critical subgroups. Instead, our approach belongs to automatic subgroup detection techniques also refer to as unsupervised techniques. An overview of these approaches is illustrated in Section 2.2.2. Differently from existing methods [26, 33, 83], we introduce algorithms that allow us to efficiently estimate the divergence in classifier behavior for all subgroups with the condition of being sufficiently represented in the dataset. Furthermore, our approach is model agnostic. Hence, it treats the classification model as a black box, without knowledge of its internal working.

The contributions of the approach outlined in this chapter are both theoretical and algorithmic and are implemented in the DivExplorer package [124]. On the theoretical side, we introduce the notion of *divergence* over itemsets, and we provide a way of measuring its statistical significance that is informed by Bayesian statistics. Next, we introduce the use (and generalization) of Shapley values to analyze the contribution of atomic patterns (single-attribute patterns such as *sex=Male*) both within larger patterns, and overall in the dataset.

Recall our example dataset *COMPAS*. Once we determine that the pattern (*age=25-45, #prior>3, race=African-American, sex=Male*) has high divergence, one might wonder about the relative contribution to divergence of the four members of the pattern, to which we refer as *items*. The problem of measuring individual contributions to a collective outcome has been considered in game theory, and the celebrated notion of Shapley value [150] answers precisely this question. We propose to apply Shapley values to divergence analysis. This will enable us to

determine that the item contributing most to the divergence is *#prior>3*, followed by *race=African-American*, with *sex=Male* giving only marginal contribution (see Figure 6.2).

In a dataset such as *COMPAS*, one is often interested not only in analyzing particular patterns where divergence is high (of which there are many; see Table 6.2 and Figure 6.7), but also in understanding what is the role of each item in leading to divergence across all patterns. The simplest approach is to measure the *individual* divergence of the item in isolation. We propose to extend the notion of Shapley value to measure the contribution to divergence of each item, in the context of all other items. The result, which we call *global* divergence, measures how much an item contributes to increasing the divergence when added to patterns. We prove that our generalization satisfies the fundamental axioms of Shapley values, stated in our modified context. Individual and global item divergence have different properties. We argue that among the two, global divergence is often a better measure of the effect of an item on divergence. In fact, individual item divergence is often unable to capture divergence that results from the association of multiple items. Global item divergence captures such associated contributions, due to its basis in the team-analysis underlying Shapley values (see Figure 6.4).

The second class of contributions is algorithmic, and they rest on the realization that item and pattern divergences can be computed efficiently by augmenting well-known *frequent pattern mining* algorithms [156] (illustrated in Section 3.2). This enables us to efficiently compute the divergence of *all* patterns whose support (frequency in the dataset) is above a specified threshold. A boundary on support is reasonable, as patterns with small support are less relevant, due to the few data instances they affect, and measurements on them are more affected by statistical fluctuations. We provide experimental results on multiple real-world datasets showing that our algorithms enable full exploration up to the support threshold typically in a matter of seconds (see Figure 6.6).

The need for a complete exploration derives from the consideration that the considered metrics to estimate differences in classification performance are not monotone. Therefore, from the divergence of a pattern, we cannot make assumptions on the divergence of the patterns that are contained in it. Let G and H be two data subsets of dataset D, with $H \subset G \subset D$. The divergence of H can be higher, equal, or lower than that of its superset G.

Previous approaches, such as [33] illustrated in Section 2.2.2, adopted heuristics to prune the search, stopping when divergence reaches sufficient values, or when a prescribed number of divergent patterns has been found. Our complete exploration not only enables the measurement of metrics such as global divergence, but also makes visible phenomena that might be invisible under pruning. One of the most intriguing is the notion of *corrective items*, which are items that *reduce* the divergence when added to patterns.

In summary, our main contributions, implemented in DivExplorer are as

follows.

- *Divergence.* We introduce the notion of divergence and we characterize each relevant data subgroup by its divergence. We estimate the local contribution of each attribute value to the subgroup divergence through the notion of Shapley value.
- *Global item divergence.* We generalize the notion of Shapley value to estimate the global contribution to divergence of each attribute value.
- *Corrective attribute value.* We introduce the notion of corrective attribute values, which tend to renormalize the divergence.
- *Bayesian treatment of statistical significance.* We propose a way to measure the statistical significance of the results that can be applied to black-box classifications.
- *Divergence computation algorithm.* We propose an efficient algorithm to automatically extract and explore all divergent subgroups with sufficient support.

The chapter is organized as follows. Section 6.1 provides our main definitions of divergence and statistical significance. Section 6.2 uses the notion of Shapley value to define local and global item contribution to divergence. Section 6.3 introduces our algorithm, and Section 6.4 presents experimental results on several real-world datasets, reporting running times and divergence results. Section 6.5 shows how the notion of divergence and its analysis can be generalized to multiple tasks beyond classification. Finally, Section 6.6 draws the conclusions.

6.1 Itemset Divergence

In this section, we firstly introduce basic concepts and definitions. We then define the notion of pattern divergence.

Background. We consider a structured dataset D consisting in a set of instances over a set with schema A. We assume that every attribute $a \in A$ can take a discrete, finite set $\mathcal{D}_a$ of values, and we let $m_a = |\mathcal{D}_a|$. An *instance* $x \in D$ assigns value $x(a) \in \mathcal{D}_a$ to every attribute $a \in A$. We only consider *discretized* attributes; continuous-valued attributes are discretized before our analysis techniques are applied.

6.1.1 Outcome Function and Itemset Divergence

Consider a dataset D with schema A, alongside a function $h : 2^D \mapsto \mathbb{R}$. The function h represents a statistic that can be computed over (subsets of) the dataset, such as the false positive or negative classification rates. For an itemset I, we write

for brevity $h(I)$ for $h(D(I))$, denoting h evaluated on the set of instances that satisfy I.

We define the *h-divergence* over an itemset I as the difference between the statistics h as measured on I, and as measured on the complete dataset.

Definition 6.1.1. (*itemset divergence*). Let I be an arbitrary itemset in dataset D and $h : 2^D \mapsto \mathbb{R}$ a function defined over subsets of the dataset. The *h-divergence* of itemset I is:

$$\Delta_h(I) = h(I) - h(D) \tag{6.1}$$

We do not provide h directly to DivExplorer. Rather, we specify h as the *outcome rate* of an *outcome function*. This will be instrumental in allowing the efficient algorithmic computation of itemset divergences.

Definition 6.1.2. (*outcome function and positive outcome rate*). Given a dataset D, an *outcome function* is a function $o : D \mapsto \{\mathrm{T}, \mathrm{F}, \bot\}$. The *positive outcome rate* $h_o(X)$ of o over a set of instances $X \subseteq D$ is defined as

$$h_o(X) = \frac{|\{x \in X \mid o(x) = \mathrm{T}\}|}{|\{x \in X \mid o(x) \neq \bot\}|} \tag{6.2}$$

Thus, instances x with $o(x) = \bot$ are not considered in the computation of the positive rate. In the thesis, we concern ourselves with classifiers and inspecting their behavior. Therefore, the outcome $o(x)$ will indicate whether x is a false-positive, or false-negative, instance in the classification. In Section 6.5, we introduce a generalization of the notion of divergence to multiple scenarios as scoring and ranker systems.

In classification tasks, if $v : D \mapsto \{\mathrm{T}, \mathrm{F}\}$ is the ground truth, and if $u : D \mapsto \{\mathrm{T}, \mathrm{F}\}$ is the classification outcome, to study the false-positive rate we use

$$o(x) = \begin{cases} \mathrm{T} & \text{if } u(x) \wedge \neg v(x); \\ \mathrm{F} & \text{if } \neg u(x) \wedge \neg v(x); \\ \bot & \text{if } v(x). \end{cases} \tag{6.3}$$

An outcome function reflecting the false-negative rate can be similarly defined. If we wish to study the positive rate of the ground truth, we can obviously set $o(x) = v(x)$ for $x \in X$. The classification outcome u can be the output of a generic classification model, making the approach model agnostic.

We will refer to $\Delta_h(I)$ as the *h-divergence* of I. When o is the false positive outcome, we will call this the *false positive divergence* of I, and so forth. When h is generic or can be understood from the context, for brevity we omit it by using the notation $\Delta(I)$.

By relying on a Boolean outcome function, we can apply DivExplorer to classifiers as black boxes, without the need for accessing their internal loss or classification probability, as would be needed for real-valued outcome functions. As

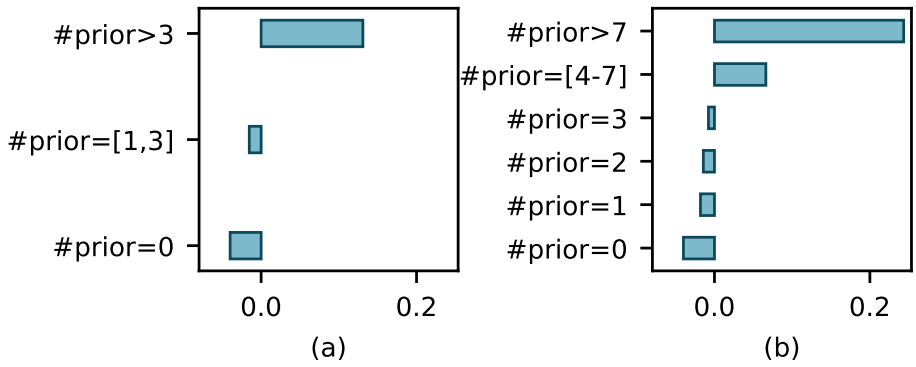


Figure 6.1: Individual item divergence for false-positive rate of *prior* attribute value of the *COMPAS* dataset where the attribute is discretized in 3 (a) and 6 (b) intervals (s=0.05.)

we shall see in Section 6.4.1, the focus on Boolean outcome functions also allows efficiently exploring itemsets and measuring their divergence.

The DivExplorer tool supports multiple metrics to assess classifier performance, such as accuracy, misclassification error, positive predictive value, true positive and negative rates, false discovery, and false omission rates. We will mostly focus on false positive and false negative rates as our measures h of interest. This allows us to study the different classification behavior of the classifier in data subgroups.

The h-divergence satisfies the following property.

Property 6.1.1. (*divergence is not hidden by finer discretization*). Let X be a set of instances, and let $X_1, .., X_n$ be a partition of X, so that $\bigcup_{i=1}^{n} X_i = X$ and $X_i \cap X_j = \emptyset$ for $1 \leq i < j \leq n$. For any h-divergence measure, there is at least one subset X_i, $1 \leq i \leq n$, with h-divergence equal or greater than X in absolute value.

The property holds because the divergence of X is simply the weighted average of $X_1, \ldots, X_n$, where the weight of X_i is the number of instances with non-bottom outcome function in X_i, for $1 \leq i \leq n$. The property has an important implication for the discretization of continuous-valued attributes. If we refine a discretization, for every divergent itemset in the coarser discretization there is at least one finer itemset that has equal or greater divergence. In other words, a finer discretization never hides divergence. This is illustrated in Figure 6.1: when the item *#prior>3* is split into the two finer ones *#prior=[4-7]* and *#prior>7*, the finer *#prior>7* has greater divergence than *#prior>3*.

6.1.2 Statistical Significance

Once an itemset with high divergence is identified, the question arises as to whether the divergence is statistically significant, or whether it originates from statistical fluctuations due to the finite size of the itemset. We can exploit the fact that the outcome function is Boolean, and follow an approach based on Bayesian statistics.

Our aim is to estimate the precision in the knowledge of the positive rate. We reason as follows. Consider a Bernoulli trial (a coin toss) with outcomes T with probability Z, and F with probability $1 - Z$. In our setting, the Bernoulli trial is the evaluation of the outcome function o over an instance in the itemset, and Z is the positive rate in the itemset. Before any trial is carried out, Z is not known, and it is natural to assume a uniform prior, which is the least information prior, $\Pr(Z = z) = 1$ for $0 \leq z \leq 1$. If we then perform trials and we observe k^+ T outcomes and k^- F ones, that is, if

$$k^+ = |\{x \mid x \models I \wedge o(x) = \mathrm{T}\}| \,, \quad k^- = |\{x \mid x \models I \wedge o(x) = \mathrm{F}\}|$$

we can use Bayes' rule to obtain the posterior distribution for the positive rate Z:

$$\Pr(Z = z) = \kappa\, z^{k^+}(1-z)^{k^-} = \mathrm{Beta}(k^+ + 1,\, k^- + 1)(z) \,,$$

where $z \in [0, 1]$, $\mathrm{Beta}(\alpha, \beta)(z) = \kappa\, z^{\alpha-1}(1-z)^{\beta-1}$ is the Beta distribution with parameters α, β, and κ is a normalization constant ensuring the distribution's integral in [0,1] is 1. This states the well-known fact that the Beta distribution is the posterior distribution that results from carrying Bernoulli trials starting from a uniform prior. We can then measure the mean and variance of our positive rate Z via the mean μ_I and standard deviation ν_I of the Beta distribution:

$$\mu_I = \frac{k^+ + 1}{k^+ + k^- + 2} \qquad \nu_I = \frac{(k^+ + 1)(k^- + 1)}{(k^+ + k^- + 2)^2\,(k^+ + k^- + 3)} \tag{6.4}$$

Once mean and variance are known, we can compare the positive rate on I to the positive rate on the whole dataset using Welch's t-test:

$$t = \frac{|\mu_I - \mu_D|}{\sqrt{\nu_I + \nu_D}} \,.$$

The advantage of the form (6.4) with respect to simply considering the mean and variance of the outcome function includes numerical stability when $k^+ + k^- = 0$, which happens when the outcome function is $\perp$ on the itemset (e.g., if we are measuring the FPR in an itemset where all instances have ground-truth $v = \mathrm{T}$).

6.1.3 Frequent Itemsets and DivExplorer

The number of itemsets in a dataset is exponential in the number of attributes. Many itemsets may have very small or empty support, and these itemsets are of lesser interest for divergence analysis, for two reasons. First, in itemsets with small support, the measure of the positive rate of o will be affected by statistical fluctuations, as discussed. Second, it is reasonable to assume that divergence affecting a larger portion of the dataset is more consequential than divergence affecting only a smaller portion of it. For these reasons, DivExplorer will only consider *frequent* itemsets, that is, itemsets I whose support size $sup(I)$ is above a given threshold s specified at the outset of the exploration.

The problem of finding all frequent itemsets in a dataset is a fundamental one in data mining, and much effort has been devoted to developing efficient algorithms for this task; see, e.g., [3, 70]. An overview of these approaches is proposed in Section 3.2. DivExplorer will leverage those algorithms, augmenting them so that the performance statistics h can be computed for all frequent itemset. The detailed algorithms are presented in Section 6.3.

6.1.4 Summarizing divergent itemsets

To provide a compact representation of pattern divergence, we present a post-exploration pruning approach. A pattern I is pruned if there exists an item $\alpha \in I$ whose absolute marginal contribution is lower than a threshold ϵ, i.e. $|\Delta_h(I) - \Delta_h(I \setminus \{\alpha\})| \leq \epsilon$. The pattern $I \setminus \alpha$ captures the divergence of pattern I, since the inclusion of item α only slightly alters the divergence (slightly with respect to the threshold ϵ). In Section 6.4.3, the impact of the ϵ input parameter on the number of resulting itemsets is studied.

6.1.5 Our Running Example: *COMPAS*

As a running example to illustrate the previous definitions, we again consider the *COMPAS* dataset. We compare the predicted recidivism rate with the actual rate, defined as the new occurrence of a misdemeanor or felony offense over a two-year period. For an instance (a person) x, we let v be the ground truth, with $v(x) = \text{T}$ iff recidivism occurred, and $v(x) = \text{F}$ if none occurred. The classification outcome $u(x)$ corresponds to the output of the *COMPAS* system. We let $u(x) = \text{T}$ if *COMPAS* classifies person x as being at high recidivism risk, and F otherwise.

Table 6.2 shows the most divergent patterns with respect to the false positive rate (FPR), false negative rate (FNR), error rate (ER) and accuracy (ACC) for a support threshold s=0.1. The pattern with highest false-positive rate divergence is I_1 = (*age=25-45,#prior>3, race=African-American, sex=Male*) with $\Delta_{FPR}(I_1)$=0.220. The model tends to be biased towards African-Americans with

Itemset	Sup	Δ_{FPR}	t
age=25-45, #prior>3, race=Afr-Am, sex=Male	0.13	0.22	7.1
age=25-45, #prior>3, race=Afr-Am	0.15	0.211	7.4
age=25-45, charge=F, #prior>3, race=Afr-Am	0.11	0.202	6.2
	Sup	Δ_{FNR}	t
age=25-45, stay<week, #prior=0	0.15	0.236	12.1
charge=M, stay<week, #prior=[1,3]	0.10	0.233	12.2
age>45, race=Cauc	0.10	0.231	10.3
	Sup	Δ_{ER}	t
age<25, stay<week, race=Afr-Am	0.10	0.098	4.7
age<25, stay<week, sex=Male	0.13	0.095	5.2
age<25, race=Afr-Am, sex=Male	0.11	0.090	4.5
	Sup	Δ_{ACC}	t
stay<week, #prior=0, race=Cauc	0.12	0.141	8.4
charge=M, stay<week, #prior=0	0.15	0.133	8.6
charge=M, #prior=0	0.16	0.129	8.5

Table 6.2: Top-3 divergent patterns with respect to FPR, FNR, error rate (ER) and accuracy (ACC) for the *COMPAS* dataset. The support threshold is $s = 0.1$.

age in the range 25-45 that have a high number of prior offenses. These three items are shared for all the top-3 FPR divergent patterns. Another influencing item is having been convicted of a felony (charge=F).

The results also indicate that the model has a higher false negative rate for people with fewer than 3 prior offenses, short stays in jail (stay<week), and having a prior conviction of a misdemeanor charge (charge=M) rather than a felony. Caucasians with age greater than 45 also have higher-than-overall FNR. We note that the model has a higher error rate for African-American defendants with age lower than 25 and short stays in jail. The model tends to be more accurate for Caucasian defendants with short stays in jail and no prior offenses.

6.2 Item Contribution to Divergence

Once itemsets with large divergence are identified, such as those of Table 6.2, the question arises as to which of the items appearing in the itemsets are most responsible for the divergence. We introduce methods both for attributing the divergence of an itemset to its items and for estimating the overall impact of an item on divergence. Our definitions are based on the notion of the *Shapley value* of a player in a coalition.

6.2.1 Shapley Value

The *Shapley value* [150] is defined in the context of a cooperative N-player game. Let $v(\sigma)$ be the value that can be attained by a coalition $\sigma \subseteq \{1, \ldots, N\}$ of players. When all players cooperate, they can achieve the value $v(\{1, \ldots, N\}) = v^*$. The Shapley value measures the contribution $\hat{v}(i)$ of each player to v^*, in such a way that $\sum_{i=1}^{N} \hat{v}(i) = v^*$. The Shapley value of player i, for $1 \leq i \leq n$, is given by:

$$\begin{aligned} \hat{v}(i) &= \sum_{\sigma \in \pi(1,\ldots,N)} v(\sigma[:i]^+) - v(\sigma[:i]^-) \\ &= \sum_{\phi \subseteq \{1,\ldots,N\} \setminus \{i\}} \frac{|\phi|!(N - |\phi| - 1)!}{N!} \Big[v(\phi \cup \{i\}) - v(\phi) \Big] , \end{aligned} \tag{6.5}$$

where $\pi(1, \ldots, N)$ is the set of permutations of $1, \ldots, N$, and where, for a permutation σ, $\sigma[: i]^+$ is its prefix up to i included, and $\sigma[: i]^-$ is its prefix up to i excluded. The Shapley value is the unique value assignment that satisfies the four properties of *symmetry*, *efficiency*, *linearity* and *null player*:

- *Symmetry:* all players are treated the same, that is, for $1 \leq i < j \leq N$, if $v(\phi \cup \{i\}) = v(\phi \cup \{j\})$ for all $\phi \subseteq \{1, \ldots, N\}\{i, j\}$, then $\hat{v}(i) = \hat{v}(j)$.
- *Efficiency:* all value is subdivided:

$$\sum_{1 \leq i \leq N} \hat{v}(i) = v(\{1, \ldots, N\}) . \tag{6.6}$$

- *Linearity:* If two games with value functions v, w are combined yielding a single game with value function $v + w$, the Shapley value of each player is given by $\hat{v} + \hat{w}$.
- *Null Player:* if a player does not contribute, i.e., if $v(\phi \cup \{i\}) = v(\phi)$ for all $\phi \subseteq \{1, \ldots, N\} \setminus \{i\}$, then $\hat{v}(i) = 0$.

6.2.2 Item Contribution to Itemset Divergence

The notion of Shapley value directly yields a way to measure the (local) contribution of an item to the divergence of an itemset.

Definition 6.2.1. (*item contribution to itemset divergence*). Given an itemset I and an item $\alpha \in I$, the contribution $\Delta(\alpha \mid I)$ of α to the divergence of I is:

$$\Delta(\alpha \mid I) = \sum_{J \subseteq I \setminus \{\alpha\}} \frac{|J|!(|I| - |J| - 1)!}{|I|!} \Big[\Delta(J \cup \alpha) - \Delta(J) \Big] . \tag{6.7}$$

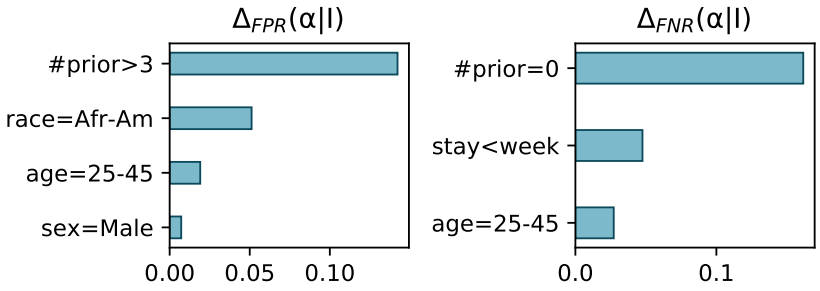


Figure 6.2: Contributions of individual items to the divergence of the *COMPAS* frequent patterns having greatest false-positive and false-negative divergence.

In (6.7), if I is a frequent itemset, then all the itemsets appearing in the formula are frequent, being subsets of I. Therefore, we can compute the local item contributions to frequent itemsets on the basis of the exploration performed by DivExplorer, which is limited to frequent itemsets.

Consider again the *COMPAS* dataset. Figure 6.2 gives the item contributions to the divergence of the frequent itemsets with largest false-positive and false-negative divergence. The item with the greatest influence on the false-positive divergence of the itemset is whether the person has at least 3 prior criminal charges. This is followed by belonging to the African-American race. The *sex=Male* item gives only minor contribution. For the divergence in false-negative rate, the greatest contribution is given by not having prior convictions. In general, we see that the items' contribution to itemset divergence can be quite different.

We note that the Shapley value tends to under-estimate the contribution to divergence of correlated items appearing jointly. For example, consider two fully-correlated items α and β in itemset $I \cup \{\alpha, \beta\}$. When appearing jointly, α and β are attributed only part of the contribution to divergence that they receive in isolation.

This effect is intrinsic to the way in which the Shapley value attributes contribution symmetrically. In DivExplorer, users can explore the lattice around any divergent itemset (see Section 6.4.4). The lattice would show that the divergence of $I \cup \{\alpha, \beta\}$ was already present in $I \cup \{\alpha\}$ and $I \cup \{\beta\}$. Users can appreciate the contribution of the items α and β by looking at their contributions to these shorter itemsets. Furthermore, this situation is mitigated by the redundancy pruning described in Section 6.1.4. According to this pruning, the itemset $I \cup \{\alpha, \beta\}$ is omitted from the output, since it is no more divergent than its subsets $I \cup \{\alpha\}$ and $I \cup \{\beta\}$.

6.2.3 Corrective Items

Divergence is not monotonic: $I \subseteq J$ does not imply $\Delta(I) \leq \Delta(J)$ for itemsets I, J. We call items that decrease divergence when added to an itemset *corrective items*.

Definition 6.2.2. (*corrective item and corrective factor*). Given an itemset I and an item $\alpha \notin I$, we say that α is a *corrective item* for I if $|\Delta(I \cup \alpha)| < |\Delta(I)|$. The *corrective factor* of α w.r.t. I is $|\Delta(I)| - |\Delta(I \cup \alpha)|$.

By performing an exhaustive exploration of all frequent itemsets, DivExplorer can identify the corrective items.

I	corr. item	Δ(I)	Δ(I $\cup\alpha$)	c_f	t
FPR					
race=Afr-Am, sex=Male	#prior=0	0.062	0.009	0.053	2.8
race=Afr-Am	#prior=0	0.051	-0.001	0.051	3.4
stay<week, #prior=0	race=Afr-Am	-0.044	-0.003	0.041	3.1
FNR					
charge=F, race=Afr-Am, sex=Male	#prior=[1,3]	-0.123	-0.011	0.112	3.8
charge=F, race=Afr-Am	#prior=[1,3]	-0.113	0.004	0.109	4.3
race=Afr-Am, sex=Male	charge=M	-0.090	-0.001	0.089	3.3

Table 6.3: Top corrective items for FPR and FNR of *COMPAS* dataset.

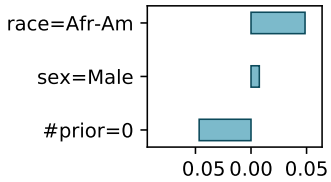


Figure 6.3: An itemset where an item has a negative divergence contribution.

Table 6.3 shows the top corrective items for false-positive and false-negative divergence in *COMPAS*. The FPR-divergence of itemset I_3=(*race=Afr-Am, sex=Male*) drops from 0.062 to 0.009 when item *#prior=0* is included, with a corrective factor of 0.053. The absence of prior convictions tends to lower the wrong assignments of African-American male defendants to the recidivist class to a similar FPR rate to the overall.

Figure 6.3 shows that the corrective effect of having no prior convictions is also reflected in the item contributions to the divergence of the corrected itemset, measured according to the Shapley value.

6.2.4 Global Item Divergence

Given an individual item α, there are two ways of measuring the effect of α on divergence. One is via its divergence $\Delta(\alpha)$ defined in (6.1). This *individual* measurement is the most common way of measuring the effect of an item on divergence. For example, when studying the effect of *race = African-American* on classification, we can measure the false-positive or false-negative divergence for this item, to see if the classifier behaves differently for people in the support-set of the item compared to people at large.

Another way of measuring the effect of an item on classification is to consider the effect of adding the item to other itemsets. As this measures the effect of the item on all itemsets, it provides a *global* measurement of the effect of the item. Roughly, the *global divergence* of an item will tell us whether the item α tends to skew the classification in every possible context. The definition is based on the notion of Shapley value, adapted to account for the fact that only items for different attributes can be part of the same itemset.

Definition 6.2.3. (*global itemset divergence*). Let D be a dataset with schema A, and let Δ be the divergence of its itemsets measured for a given outcome function. We define the *global divergence* $\Delta^g(I)$ of an itemset I of D as:

$$\Delta^g(I) = \sum_{B \subseteq A \setminus \text{attr}(I)} \frac{|B|!(|A| - |B| - |I|)!}{|A|! \prod_{b \in B \cup \text{attr}(I)} m_b} \sum_{J \in \mathcal{I}_B} \Big[\Delta(J \cup I) - \Delta(J)\Big] \,. \tag{6.8}$$

The definition parallels (6.5), except for the additional factor $1/(\prod_{b \in B \cup \text{attr}(I)} m_b)$, which is necessary to normalize the sums, accounting for the number of different itemsets with given attributes. The following theorem gives the properties of the above notion of global divergence. Together, these properties formalize the fact that (6.8) is the generalization of Shapley value to the itemset case.

Theorem 1. (*properties of global divergence*). Consider a dataset D with set A of attributes, alongside a divergence Δ for its itemsets. The global divergence defined as in (6.8) satisfies the following properties:

- *Efficiency:*
$$\sum_{a \in A} \sum_{c \in \mathcal{D}_a} \Delta^g(a = c) \;=\; \frac{1}{|\mathcal{I}_A|} \sum_{I \in \mathcal{I}_A} \Delta(I) \,. \tag{6.9}$$
- *Null items:* if there is an attribute $a \in A$ such that, for all $c \in \mathcal{D}_a$ and all itemsets $I \in \mathcal{I}$ with $a \notin \text{attr}(I)$, $\Delta(I) = \Delta(I \cup \{a = c\})$, then $\Delta^g(a = c) = 0$. Furthermore, under the above hypotheses, removing a from A does not affect the value of $\Delta^g(I)$ for any itemset I not containing a.
- *Symmetry:* if for two itemsets I, I' we have $\Delta(J \cup I) = \Delta(J \cup I')$ for all $J \in \mathcal{I}$ with $\text{attr}(J) \cap \text{attr}(I) = \emptyset$ and $\text{attr}(J) \cap \text{attr}(I') = \emptyset$, then $\Delta^g(I) = \Delta^g(I')$.

- *Linearity:* If two notions of divergence $\Delta_1, \Delta_2 : 2^X \mapsto \mathbb{R}$ are combined into a single one $\Delta = \gamma_1\Delta_1 + \gamma_2\Delta_2$ via a linear combination, for every item I we will have $\Delta^g(I) = \gamma_1\Delta_1^g(I) + \gamma_1\Delta_2^g(I)$, where Δ^g is computed from Δ, and Δ_1^g, Δ_2^g from Δ_1, Δ_2, respectively.

These properties are the generalization of the corresponding properties of Shapley values. The difference in the forms is due to the fact that there is more than one complete itemset.

Accounting for support lower bound. In DivExplorer we cannot use (6.8) directly, as it involves the consideration of all itemsets. Rather, we opt for an approximation of (6.8), in which we limit the summation to frequent itemsets, whose support is at least s. Let $\mathcal{I}_B^\star$ be the set of frequent itemsets with attributes B. We define the global divergence approximated to support s via:

$$\widetilde{\Delta}^g(I, s) = \tag{6.10}$$
$$\sum_{B \subseteq A \setminus \text{attr}(I)} \frac{|B|!(|A| - |B| - |I|)!}{|A|! \prod_{b \in B \cup \text{attr}(I)} m_b} \sum_{J : J \cup I \in \mathcal{I}^\star_{B \cup \text{attr}(I)}} \Big[\Delta(J \cup I) - \Delta(J)\Big].$$

If I is frequent, the summations can be computed in terms of frequent itemsets only, and the approximation can be computed on the basis of the output of DivExplorer, which only outputs frequent itemsets.

6.2.5 Global vs. Individual Item Divergence

For an item α, we can measure both the *individual divergence* $\Delta(\alpha)$ defined by (6.1), and the *global divergence* $\widetilde{\Delta}^g(\alpha, s)$ defined by (6.10). The individual divergence $\Delta(\alpha)$ is independent of the support threshold (provided the item itself is above the threshold). The global divergence $\widetilde{\Delta}^g(\alpha, s)$, on the other hand, depends on the support threshold s chosen for its analysis.

The value of global divergence lies in its ability to highlight the role of items in giving rise to divergence via association with other items. For instance, assume that in a dataset there are two items α, β that cause divergence in the itemset $\{\alpha, \beta\}$, but less so in isolation. The individual divergence of α and β may be low, masking the effect of the items when jointly present. On the other hand, global divergence is able to capture the effect of α and β on divergence, provided the itemset $\{\alpha, \beta\}$ has support above the threshold. We make this observation precise via a theorem, and via an example on an artificial dataset.

Theorem 2. (*individual and global divergence do not coincide*). There is a dataset D with schema A, a minimum support $s > 0$, and items $a = c$ for $a \in A$, $c \in \mathcal{D}_a$, such that $\Delta(a = c) = 0$ but $\widetilde{\Delta}^g(a = c, s) \neq 0$.

To illustrate how global item divergence is able to capture the role of items that cause divergence when joint with other items, we constructed an artificial 10-dimensional dataset, denoted *artificial*, with 50,000 instances and attributes $a, b, c, \ldots, j$ with domain $\{0, 1\}$.

We construct the dataset, and a classifier, so that the itemsets $a = b = c = 1$ and $a = b = c = 0$ are divergent. To this end, we create the instances by setting each of their attributes $a, \ldots, j$ randomly and independently to values 0 and 1, with equal probability. We first train a classifier with respect to a class label that is T when $a = b = c$ and F otherwise. Then, to simulate classification errors, during test, we flip the class label for half of the instances in $a = b = c$ (without retraining the classifier).

The global and individual item divergence for the false positive rate, analyzed with minimum support $s = 0.01$, are given in Figure 6.4. We see that individual item divergence is unable to capture the role of a, b, c, together, to cause high divergence. The divergence of $a = b = c$ is completely masked by statistical fluctuations in the overall dataset, to the point that unrelated items such as $g = 0, g = 1, h = 0, h = 1$ have much larger individual divergence that items for attributes a, b, c. On the other hand, the global divergence is clearly able to identify the attributes a, b, c as those causing divergence when appearing together.

For the *COMPAS* dataset, Figure 6.5 compares the global and individual false-positive divergence for items. Global divergence assigns more importance to racial factors: for instance, being African American introduces almost as much bias to an itemset as having been convicted more than 3 times. This indicates how race plays a role jointly with other factors in creating highly divergent itemsets.

6.3 The DivExplorer Algorithm

The DivExplorer algorithm extracts frequent subsets of attribute values and estimates their divergence. The computation is embedded in the frequent pattern extraction process, and DivExplorer can leverage any frequent pattern mining (FPM) technique [156] to extract frequent subsets. More specifically, when the support of an itemset is estimated, the outcome function o and outcome rate h are also computed. Hence, the performance of DivExplorer directly depends on the efficiency of the selected FPM algorithm, because the dataset is accessed as many times as the selected underlying FPM method does.

FPM algorithms require discrete data. Thus, continuous attributes (if any) are firstly discretized. This discretization is only performed after the classification process. In particular, DivExplorer does not require the classification algorithm to rely on discretization.

Algorithm 2 outlines the main steps of DivExplorer. Given the input data set D, the ground truth v, the classification outcome u, the outcome function o, and

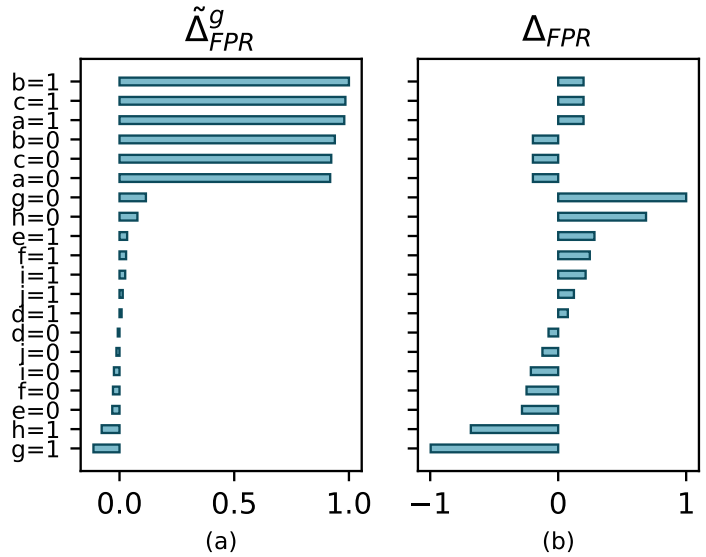


Figure 6.4: Relative magnitudes of $\widetilde{\Delta}^g(\cdot, s)$ and individual item divergence, for false-positive rate in the artificial dataset. The attributes a, b, c give raise to divergence when appearing together: global divergence captures this.

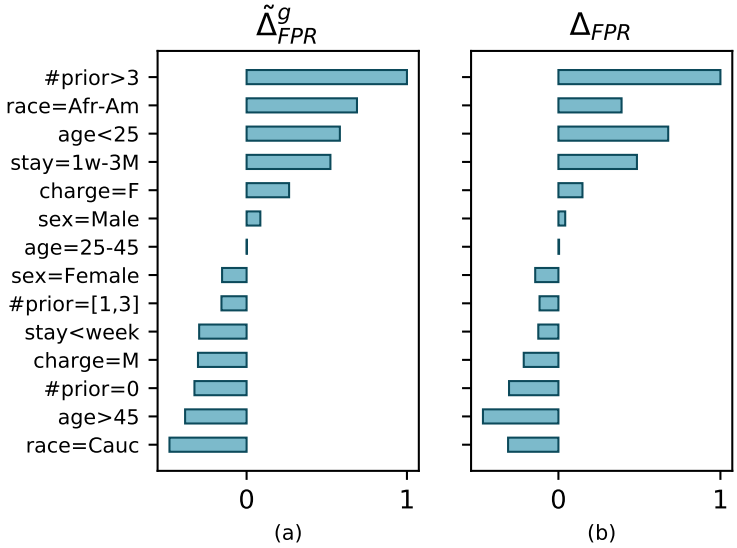


Figure 6.5: Relative magnitudes of global Shapley value and individual item divergence, for false-positive rate in the *COMPAS* dataset with $s = 0.1$

outcome rate of interest h, the algorithm returns the divergence coefficient for all frequent itemsets. The algorithm requires the definition of the minimum support threshold s as its (single) input parameter.

Algorithm 2: The DivExplorer algorithm.

Input: D, u, v, o, h, s
Output: FP divergence FP_Δ

```
1  o(D) = computeOutcomeFunction(D, o, h);
2  T̂_D, F̂_D, ⊥̂_D = OutcomeOneHotEncoding(o(D));
3  FI_withOutcomes = [];
4  for step_i in Frequent Pattern Mining steps do
5  |   I_step_i = extractItemsets(D, step_i);
6  |   for I in I_step_i do
7  |   |   T_I, F_I, ⊥_I = cardinalityOutcomesI(I, (T̂_D, F̂_D, ⊥̂_D));
8  |   |   if (T_I + F_I + ⊥_I)/len(D) ≥ s then
9  |   |   |   FI_withOutcomes.append(I, (T_I, F_I, ⊥_I));
10 |   |   end
11 |   end
12 end
13 FP_h = evaluateFunctionh(FI_withOutcomes, h);
14 FP_Δh = divergence(FP_h, h(D));
15 return FP_Δh
```

The first step (Line 1) of Algorithm 2 computes the outcome function on all instances $x \in D$ (see Section 6.1.1). The outcome function results are then the inputs to the *OutcomeOneHotEncoding* function that maps each outcome to a one-hot representation. More specifically, for each instance $x \in D$, T_x, F_x and $\bot_x$ are estimated, with T_x equal to 1 if $o(x) = \mathrm{T}$ and 0 otherwise (F_x and $\bot_x$ are computed analogously). This representation enables us to tally the outcome function values simply by adding the one-hot representations. The results are $\hat{\mathrm{T}}_D$, $\hat{\mathrm{F}}_D$ and $\hat{\bot}_D$ one-hot representations of outcome function $o(x)$ for dataset D.

Next, for each $step_i$ of a generic FPM technique, itemsets are extracted (Line 5). The general function *extractItemsets* extracts the itemsets to be evaluated for support threshold at $step_i$ and varies depending on the FPM algorithm of choice. For example, $step_i$ could be level i iteration in level-wise approaches such as Apriori [3], or the recursive step performed by FP-growth [70] on the FP-tree compressed representation. We implemented both an Apriori-based and an FP-growth-based version of DivExplorer.

The cardinalities T_I, F_I, $\bot_I$ of each itemset I extracted at $step_i$ are then estimated, with $\mathrm{T}_I = |\{x \mid x \models I \wedge o(x) = \mathrm{T}\}|$, $\mathrm{F}_I = |\{x \mid x \models I \wedge o(x) = \mathrm{F}\}|$ and $\bot_I = |\{x \mid x \models I \wedge o(x) = \bot\}|$. Note that function *cardinalityOutcomesI* does not require access to the dataset D, because it is integrated into the FPM algorithm. T_I, F_I and $\bot_I$ are computed as the sum of the $\hat{\mathrm{T}}_D$, $\hat{\mathrm{F}}_D$ and $\hat{\bot}_D$ terms that satisfy I. The sum of T_I, F_I and $\bot_I$ represents the support count of itemset I, i.e. $\mid D(I) \mid$.

Hence, it is exploited to estimate if I is frequent, i.e. with a support greater or equal than s (Line 8). Frequent itemsets and their cardinality outcomes are stored in $FI\_withOutcomes$ (Line 9).

Once all frequent itemsets are extracted, the outcome rate of outcome function h is estimated for all frequent itemsets with *evaluateFunctionh* (Line 13). Finally, the h-divergence (Equation 6.1) of all frequent itemsets is computed (Line 14) and returned (Line 15). The extracted frequent itemsets can be ranked according to many different metrics, such as their statistical significance, support, or h-divergence. In this work, we rank itemsets according to h-divergence, to identify subgroups where the classification behavior diverges strongly. Users can choose their preferred ranking according to the problem, and to their desired analysis goals.

It is straightforward to extend Algorithm 2 to efficiently compute the h-divergence of multiple outcome functions simultaneously.

Theorem 3. (*Soundness and completeness*) Algorithm 2, called with minimum support s, is sound and complete:

- *Sound:* If Algorithm 2 outputs an itemset I along with h-divergence $\Delta_h(I)$, then there is an itemset I in the dataset with support above s and with divergence $\Delta_h(I)$.
- *Complete:* If there is an itemset I with support above s and with divergence $\Delta_h(I)$, the itemset I along with its divergence will be part of the output.

We note that completeness does not hold for Slice Finder, since the search for problematic itemsets is pruned whenever sufficiently problematic itemsets are found, so that longer (more specific) itemsets, even if more problematic, can be missed. This will be illustrated later in Section 6.4.5.

6.4 Experimental results

dataset	$\|D\|$	$\|A\|$	$\|A\|_{cont}$	$\|A\|_{cat}$
adult	45,222	11	4	7
bank	11,162	15	6	9
COMPAS	6,172	6	2	4
german	1,000	21	7	14
heart	296	13	5	8
artificial	50,000	10	0	10

Table 6.4: Dataset characteristics. A_{cont} is the set of continuous attributes, A_{cat} of categorical ones.

We present here results on the running time of DivExplorer, on its ability to extract and summarize divergence information on real-world datasets, and on the visualizations and explorations that can be created on the basis of its output. We also outline the main differences between our approach and Slice Finder [33].

The main features of the datasets used in our experiments are reported in Table 6.4. The cardinalities are reported after standard preprocessing steps (e.g., removing instances with missing values). For most of our experiments, we used the *COMPAS* dataset [8], already introduced in Section 6.1.5, and the *adult* dataset [96]. The *adult* dataset includes census data and the prediction of individual incomes, divided into two classes "≤50K" and "'>50K". In our analysis, we used the age, workclass, education, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week features. For the performance experiments, we also used the *German Credit Data*, *Bank Marketing*, and *heart* datasets [96]. The *German Credit Data* (*german*) dataset is devoted to the prediction of an individual's credit risk using loan application data, according to attributes as age, sex[1], checking_account, credit_amount, duration, purpose, etc. The *Bank Marketing* (*bank*) dataset contains information related to a direct marketing campaign of a Portuguese banking institution. The *heart* dataset contains data to detect the presence of a heart disease in patients. Its features describe the demographical and health information (as serum cholesterol, resting blood pressure) of patients. Finally, we also used the *artificial* dataset already described in Section 6.2.5.

DivExplorer has been developed in Python. The source code and all the datasets used in our experiments are available [2], together with the description of all performed preprocessing steps. In all the reported experiments, DivExplorer is coupled with FP-growth (illustrated in Section 3.2) as frequent pattern mining technique to extract frequent itemsets [70].

6.4.1 Performance analysis

We evaluated the efficiency of DivExplorer by measuring the execution time required to (i) extract all frequent itemset and (ii) estimate their divergence and statistical significance. We repeated each experiment 5 times and reported the average execution time. The experiments were performed on a PC with Ubuntu 16.04.1 LTS 64 bit, 16 GB RAM, 2.40GHz×4 Intel Core i7. For all the datasets, except *COMPAS* and the *artificial* dataset (for which the class label is already provided), we used a random forest classifier with default parameters to provide the classification outcome u.

[1] From the the original features, we derived "sex" and "civil-status" from the "personal-status" attribute.

[2] https://divexplorer.github.io/

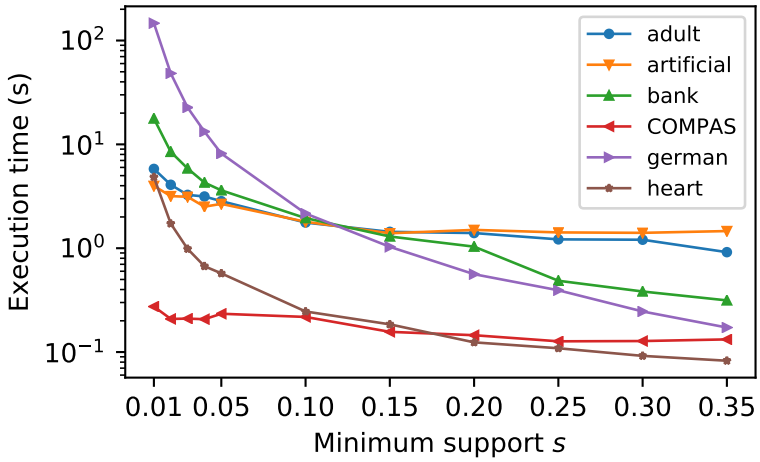


Figure 6.6: DivExplorer execution time when varying the minimum support threshold.

Figure 6.6 shows DivExplorer execution time as a function of the support threshold. The higher the support threshold, the lower the running time. Note that the execution time depends on the FPM algorithms used for the extraction of the itemsets (FP-growth in the reported experiments). For all considered datasets, except *german*, the execution time is below 20s, even for minimum support thresholds as low as 0.01. For the *german* dataset the worst-case execution time is anyway lower than 150s. The execution time required to compute itemset divergence and statistical significance is negligible (<7%) compared to the time required for itemset extraction.

The number of frequent itemsets extracted by DivExplorer when varying the minimum support is reported in Figure 6.7. For low support thresholds, the number of extracted patterns for the *german* dataset is very high, thus impacting the execution time, as shown in Figure 6.6. For this dataset, a support threshold equal to 0.01 is rather low, as it corresponds to 10 records only (see Table 6.4). Nevertheless, the ability of DivExplorer to find divergent itemsets with very low levels of support enables the analysis of under-represented group behavior in the dataset.

6.4.2 Exploring dataset divergence

In this section, we demonstrate the capability of DivExplorer to (a) detect the itemsets that mostly contribute to misclassifications, (b) provide a "drill-down" analysis to highlight most influential items in an itemset divergence, and (c) explore the global contribution of single items to divergence. We focus on the *adult* dataset, as similar results for *COMPAS* have been presented throughout the chapter. A

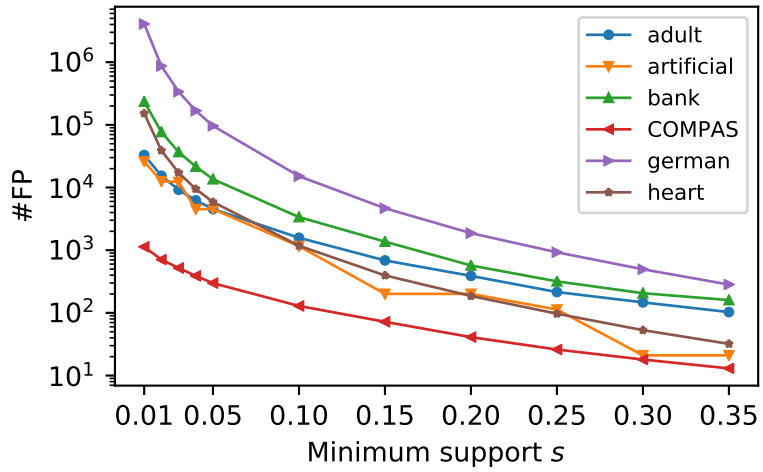


Figure 6.7: Number of frequent itemsets when varying the minimum support threshold.

complete report of the experimental outcome for all the datasets under analysis is available [124].

Itemset	Sup	Δ_{FPR}	t
gain=0, status=Married, occup=Prof, race=White	0.05	0.469	25.8
gain=0, loss=0, status=Married, occup=Prof	0.05	0.462	26.6
loss=0, status=Married, occup=Prof, race=White	0.06	0.458	25.3
	Sup	Δ_{FNR}	t
age≤28, gain=0, hoursXW≤40, status=Unmarried	0.17	0.61	21.8
gain=0, loss=0, edu=HS, hoursXW≤40, status=Unmarried	0.14	0.61	28.2
gain=0, loss=0, status=Unmarried, relation=Own-child	0.12	0.61	18.9

Table 6.5: Top-3 divergent itemsets for FPR and FNR. *adult* dataset, $s = 0.05$.

Table 6.5 shows the top divergent itemsets for *adult*, both for the FPR and FNR rate, with $s = 0.05$. The reported itemsets show some degree of overlap, which will be discussed in Section 6.4.3. Figure 6.8 reports the item contributions to the top divergent itemsets of Table 6.5. Figure 6.8(a) shows that the most relevant items which contribute to the higher-than-overall misclassification rate for the high-income class are *being married* and *working as a professional*. The items *gain=0* (capital gain) and *race=White* have instead a very small influence. For the top FNR itemset (Figure 6.8(b)), we observe that *age* $\leq$ *28*, *capital gain* $=$ *0*, and

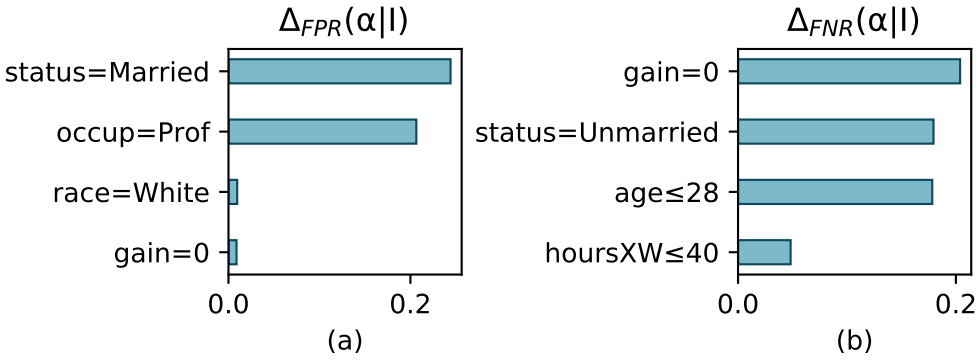


Figure 6.8: Contributions of individual items to the divergence of the adult frequent patterns having greatest FPR (Line 1 of Table 6.5) and FNR (Line 4 of Table 6.5) divergence.

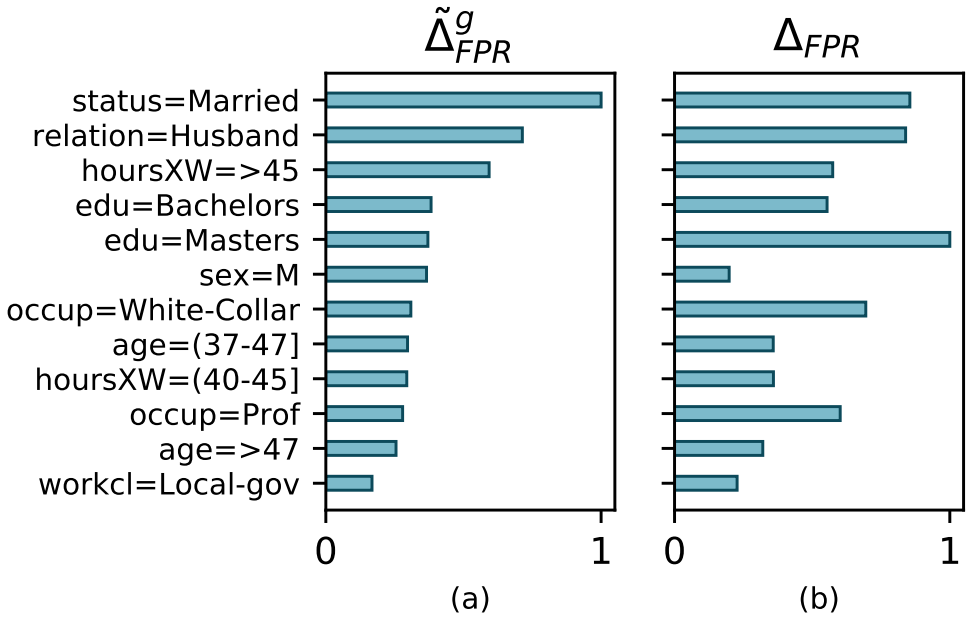


Figure 6.9: Relative magnitude of global Shapley value (a) and individual item divergence (b), for FPR, adult dataset, $s = 0.05$. Top 12 global item positive contributions are reported.

being unmarried are the most important items, while *number of hours per week* $\leq$ *40* provides a limited contribution.

Figure 6.9 shows the relative magnitude of global and individual item contribution to FPR divergence, again for *adult*; for conciseness, only the 12 items with the largest positive contribution are shown. Consider the item *edu = Masters*. While its

individual divergence is the highest overall, its global divergence is markedly lower, indicating its limited role in giving rise to divergence via association (in longer itemsets). Indeed, *edu = Masters* does not appear in the top divergent itemsets of Table 6.5.

6.4.3 Summarizing divergent itemsets

Itemset	Sup	Δ_{FPR}	t
status=Married, occup=Prof	0.07	0.434	26.1
occup=Prof, relation=Husband	0.06	0.423	23.4
edu=Bachelors, status=Married	0.09	0.413	29

Table 6.6: Top-3 divergent itemsets for FPR with redundancy pruning. *adult* dataset, $\epsilon = 0.05$, $s = 0.05$.

As seen in Table 6.5, the top divergent itemsets often include some level of redundancy. DivExplorer can reduce such redundancy via the heuristic pruning approach discussed in Section 6.1.4, which eliminates itemsets that are not significantly more divergent than their shorter subsets. We report in Table 6.6 the top FPR-divergent itemsets for the *adult* dataset when applying a redundancy threshold $\epsilon = 0.05$. Comparing this result with Table 6.5, we note how pruning helps in presenting more diverse, and thus relevant, information. The most FPR-divergent itemset in Table 6.6 is *status=Married, occup=Prof* (occupation=Professional), with a slightly lower divergence and similar statistical significance. The importance of these two items was already shown by Figure 6.8(a), in which they were providing the most relevant contribution to the itemset divergence. On a global scale, for the FPR-divergence, the total number of extracted itemsets drops from 4534 to just 40.

Figure 6.10(a) and 6.10(b) report a quantitative evaluation of the impact of the pruning parameter ϵ, and minimum support s on the number of divergent itemsets returned, for FPR-divergence in *COMPAS* and *adult*. We see how the heuristic post-pruning, even with relatively small values of ϵ, leads to an effective summarization of divergent itemsets.

6.4.4 Lattice visual exploration

DivExplorer allows the interactive exploration of divergent patterns by means of a visual representation of the itemset lattice. In this lattice, nodes correspond to frequent itemsets and edges to subset relationships between itemsets. Given a divergent pattern of interest I, the itemset lattice shows all its subsets and their

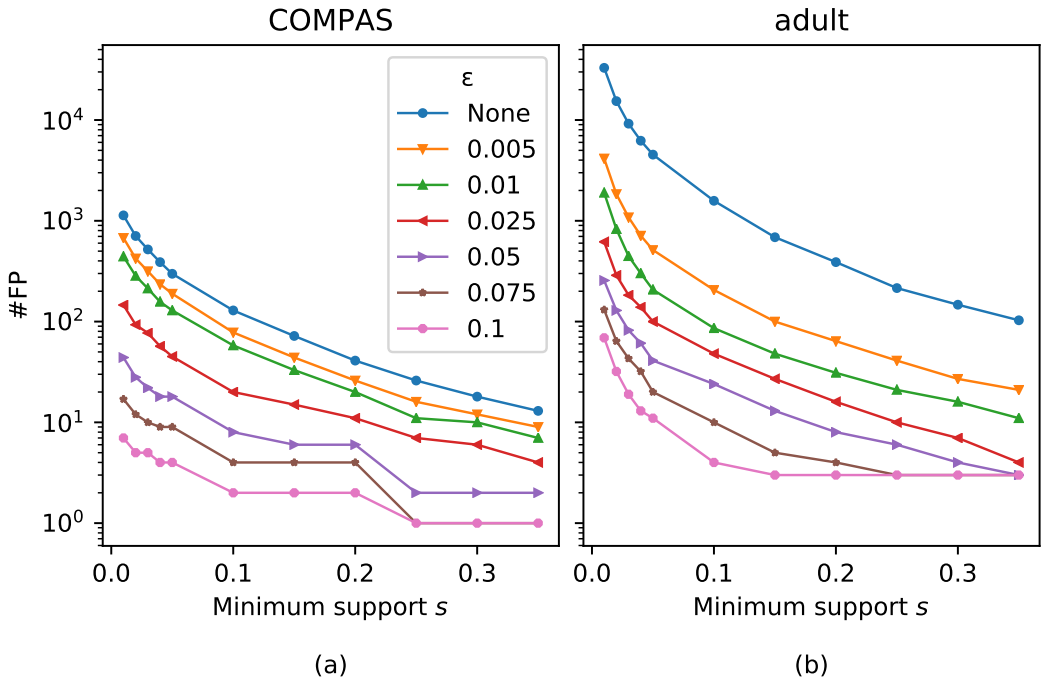


Figure 6.10: Number of frequent itemsets varying redundancy pruning threshold ϵ for FPR divergence of *COMPAS* and *adult* datasets.

divergence coefficient. The root represents the empty subset (with Δ_h=0 by definition) and the last level the pattern I itself. Figure 6.11 reports a portion of the lattice for the *adult* dataset. The itemset lattice may be actively navigated. The visualization allows the identification of the items driving divergence increases, i.e., items that, when added to a subset, increase the divergence. Furthermore, the user may interactively select a divergence threshold T. The lattice nodes with divergence coefficients larger than the threshold are highlighted.

The itemset lattice can also be exploited to explore corrective behaviors. The visualization highlights the subsets (i.e., nodes in the lattice) in which a corrective phenomenon is observable. An example of corrective phenomenon visualization is reported in Figure 6.11. The example shows the lattice for the FNR divergence of itemset I_x=(*loss=0, workclass=Private, edu=Bachelors, gain=0*) in the *adult* dataset. Item *edu=Bachelors* is a corrective item for pattern I_y=(*loss=0, workclass=Private, gain=0*). The FNR divergence drops from 0.17 for itemset I_y to -0.03 for itemset I_x when the item *edu=Bachelors* is included. Besides pattern I_x, item *edu=Bachelors* introduces a corrective effect for all the itemsets including it in the lattice. Hence, the exploration of the itemset lattice also allows a deeper and more comprehensive analysis of corrective behaviors.

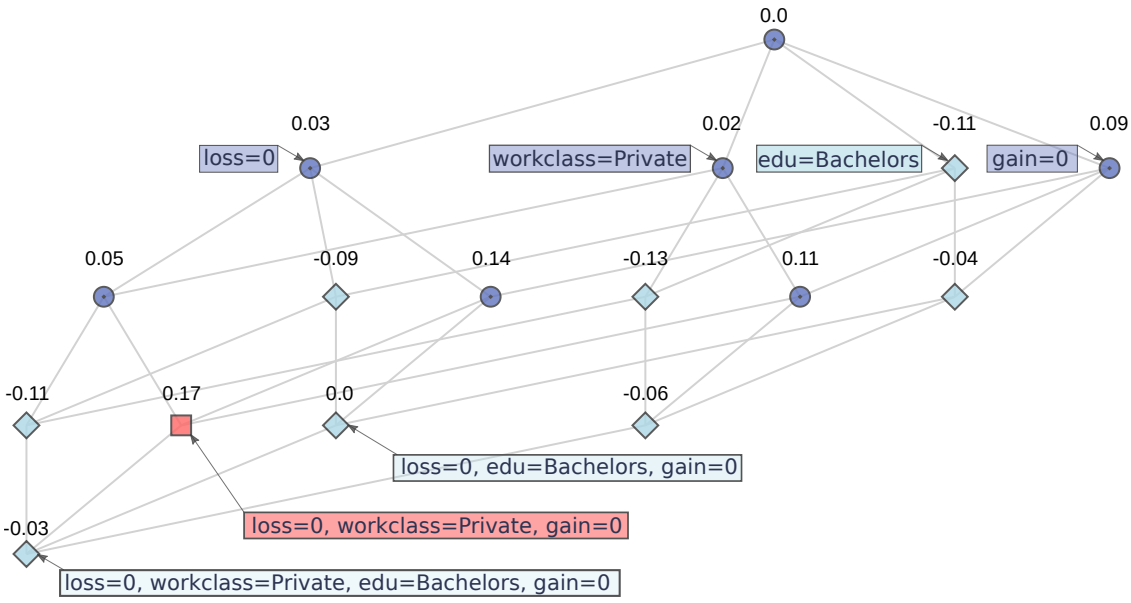


Figure 6.11: Lattice showing a corrective phenomenon for FNR divergence on the *adult* dataset. Nodes showing a corrective phenomenon appear as rhombus in light blue. Nodes with FNR divergence $\geq T = 0.15$ are squares in magenta.

6.4.5 Comparison with Slice Finder

Slice Finder [33] is close to our approach DivExplorer. It identifies slices of the data, denoted by conjunctions of literals, i.e., itemsets, in which the model performs poorly. Slice Finder measures how "problematic" a slice is by comparing the classifier loss on the slice, and on the *remainder* of the dataset. This notion is similar to our notion of divergence, with two differences. First, Slice Finder measures classifier loss, while DivExplorer is based on an outcome function that encodes metrics such as FPR and FNR. Second, Slice Finder measures the difference between an itemset and its complement, while DivExplorer measures the difference between the itemset and the *whole* dataset. The main difference between Slice Finder and our approach, however, is that Slice Finder's search is not exhaustive: the exploration of an itemset is stopped (no larger itemsets are considered) when a sufficiently large deviation is found, and the overall exploration stops once a prescribed number of itemsets has been found. We can afford to perform an exhaustive search due to our reliance on efficient frequent pattern mining algorithms. Our exhaustive search allows us to study item contributions to individual itemsets and global divergence. The exhaustive search also enables the identification of corrective items.

It is difficult to provide a comparison of Slice Finder and DivExplorer on a general dataset, because the two tools drive their exploration differently (effect size and bound on result size for Slice Finder, support size, and divergence for

DivExplorer). For this reason, we use an artificial dataset for validation purposes. We use the *artificial* dataset of Section 6.2.5, where the divergent itemsets $a = b = c = 0$ and $a = b = c = 1$ are well characterized and drive both explorations equally. Unless differently specified, we executed Slice Finder with its default parameters. We use the predicted and class labels as inputs to DivExplorer. We used a Random Forest classifier with default parameters to provide the loss function required by Slice Finder. DivExplorer minimum support is set to 0.01. For Slice Finder, we set *degree* to 3 to obtain itemsets of length 3.

Since DivExplorer does not enforce parallel execution, for a fair comparison we turned it off in Slice Finder. In this case, DivExplorer mean execution time is 4s, 4.5 times faster than Slice Finder. If parallel execution is turned on (max-workers=4), DivExplorer is 3.5 times faster than Slice Finder.

DivExplorer successfully identifies $(a{=}0,b{=}0,c{=}0)$ and $(a{=}1,b{=}1,c{=}1)$ as the itemsets with the highest FPR divergence. Slice Finder finds all 6 subsets of length 2 of $(a{=}0,b{=}0,c{=}0)$ and $(a{=}1,b{=}1,c{=}1)$. These subsets are already highly "problematic" (in our terms, they have high divergence). Hence, Slice Finder's search stops. The stopping criterion based on "problematicity" fails to identify the two itemsets that are the true source of divergence, returning their many subsets instead. If the threshold of the effect size is increased to 1.65, Slice Finder identifies the true source of divergence. This search requires 18s with 1 worker.

6.4.6 User study

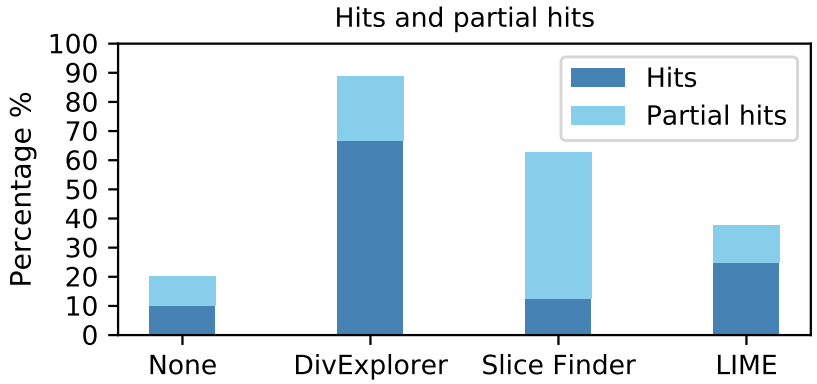


Figure 6.12: User study results. Percentage of hits for the injected bias according to the provided information.

We conducted a user study to assess how useful is the information provided by DivExplorer in helping users identify data subgroups with anomalous behavior. The study compared the information provided by DivExplorer, Slice Finder, and LIME [138], as a relevant representative method from the Explainable AI

domain. The latter can be considered as a semi-supervised approach. The human-in-the-loop identification is aided by the information of the behavior of individual predictions. For the study, we considered the *COMPAS* dataset. We performed a controlled experiment in which we artificially injected bias in a subgroup, and we measured how well the information provided by the different tools allowed users to identify the injected bias. Specifically, in the training set, we injected bias in the subgroup characterized by the pattern {age_cat>45, charge_degree=M}, changing all outcomes to recidivate, and we trained a (biased) multi-layer perceptron neural network on such modified dataset. We then analyzed the misclassifications of such a biased classifier on the (unmodified) testing dataset with DivExplorer, Slice Finder, and LIME.

The study involved 35 undergraduate computer science students, who had some knowledge of the notions of classifiers, and false positive and negative errors. We divided the users into four groups. Group 1 was shown examples of correctly and misclassified instances drawn uniformly at random. The other groups were shown the same information as group 1, and in addition:

- Group 2: the top 6 itemsets and their Δ_{FPR} computed by DivExplorer with $s = 0.05$, and the global item divergence.
- Group 3: the itemsets computed by Slice Finder and their impact factors, with *degree=3* and default parameters.
- Group 4: LIME explanations for 8 correctly classified and 8 mis-classified instances drawn uniformly at random.

The amount of information received by groups 2, 3, and 4, was similar, amounting to a couple of pages in PDF format. We asked the participants to select the top 5 itemsets that are most affected by errors. We consider the following metrics in evaluating the user answers: *hit* and *partial hit*. The metric *hit* $\in \{0,1\}$ is 1 if the user included the injected bias itemset {age_cat>45, charge_degree=M}, and 0 otherwise. The metric *partial hit* $\in \{0,1\}$ is 1 if the user included the items {age_cat>45} or {charge_degree=M}, and 0 otherwise.

Figure 6.12 summarizes the percentage of hits and partial hits for each user group. The information provided by DivExplorer was the one that led the users most directly to identify the injected bias, with a combined hit rate of 88.89%. In group 1, 20% of the users completely or partially identified the biased subgroups by carefully inspecting the misclassified instances. In group 3 (Slice Finder), most of the users only partially selected the biased itemset. Slice Finder with default parameters identifies the two items composing the itemset as already highly 'problematic', and prunes the search. Finally, in group 4 the explanations provided by LIME led to a combined hit rate of 37.5%. Interestingly, the supervised approach using the explanation provided by LIME had more full hits than Slice Finder, in

spite of LIME's goal being to provide classification explanations, rather than identifying critical subgroups.

6.5 Identifying divergent subgroups in scoring and rankings

DivExplorer allows to understanding peculiar behaviors of classification models in data subgroups. In this context, DivExplorer is a post-hoc model agnostic explainability technique to enhance the interpretability of classification models at the subgroup level. The notion of divergence and the DivExplorer approach can be extended to other critical scenarios beyond classification purposes [125]. In the following section, we propose a generalization of DivExplorer for scoring functions, ranking systems, as well as general quantitative prediction functions as in regression problems.

6.5.1 Generalization of divergence

We are interested in identifying subgroups of data that behave differently, compared to the overall dataset, with respect to statistical measures. In rankings, we are interested in data subgroups where the average rank deviates from the global one. In scoring, we want to identify the subgroups with an average score that differ from the average one for the entire data.

The definition of divergence proposed in Definition 6.1 is already generally defined for any generic function $h : 2^D \mapsto \mathbb{R}$ defined over subsets of the dataset. However, for classification performance measures, we specify h as the outcome rate of a boolean outcome function.

We propose a redefinition of the outcome function introduced in Section 6.1. The redefined *outcome function* captures the statistic of interest.

An *outcome function* $o : X \mapsto \{\bot\} \cup \mathbb{R}$ associates with each instance either a do-not-consider value $\bot$, or a real number. For an itemset I, we define the *outcome* $h_o(I)$ *on* I via:

$$h_o(I) = E\{o(x) \mid x \models I, o(x) \neq \bot\} \ . \tag{6.11}$$

If I is empty, $h_0(\emptyset)$ is the outcome of the complete dataset. We use $h_o(I)$ with respect to outcome function o to define the divergence as in 6.1:

$$\Delta_{h_o}(I) = h_o(I) - h_o(\emptyset) \ . \tag{6.12}$$

The divergence of an itemset captures the difference in behavior between the itemset, and the entire dataset, with respect to the outcome function under consideration.

With this redefinition of the outcome function, we can apply all the explorations and analyses previously described. The generalization involves only two minor adaptations of the DivExplorer approach. The first regards the DivExplorer algorithm for the extraction of patterns and their computation of divergence described in Section 6.3. For boolean outcome function we still leverage Algorithm 2. For the other definitions, the implementation is very straightforward and consists of substituting the one-hot encoding of the boolean outcome function to consider the $o(x) \neq \perp$ terms. The second adaptation regards the statistical significance computation. For non-boolean outcome functions, we directly use Welch's t-test.

6.5.2 Scenarios of pattern divergence

The definition of the outcome function of an instance $o(x)$ varies considering the task of interest. In the following, we show how different outcome functions enable the analysis of the divergence for scoring function and ranking systems.

As a running example to better illustrate the scenarios and the corresponding outcome functions we use the *law school* dataset. The Law School Admission Council conducted a survey across 163 law schools in the United States in 1998 [168]. The resulting Law School Dataset contains information on 21790 law students such as their entrance exam scores (LSAT), their grade-point average (GPA) collected prior to law school, and their normalized first-year average grade (ZFYA), in addition to their race and sex. We use the dataset as prepared by [88]. In this dataset, we study how the average ZFYA score, and the average rank of students after the first year, vary across subgroups. In the experiments, we set the minimum support threshold s to 0.005 and it corresponds to about 100 students. The extraction of frequent patterns and the estimation of divergence take than a second (0.3s).

Score divergence.

In numerous scenarios, the outcome of an instance can be a quantitative attribute associated with the instance itself. The quantitative value can be derived in multiple ways. It can be a quantitative intrinsic characteristic of the instance itself. For example, in the Law School Dataset, the outcome can be the normalized first-year average of each student. It consists in taking $o(x) = ZFYA(x)$.

More generally, the outcome can be derived by a generic function $w : A \mapsto \mathbb{R}$ that assigns a score for each instance given the attribute space of input. The score can also be an integer or a binary attribute.

Scoring functions are adopted in a wide range of scenarios as online job marketplaces, school admissions, risk assessment, and quality evaluation. The understanding of subgroups of data in which the scores differ from the overall behavior is particularly relevant considering the fairness concerns. Scoring functions are often applied to rate and rank individuals. For example, students are rated for

school admissions or potential employees are ranked in online job marketplaces. In these contexts, it is compelling the assessment if divergent behaviors occur in data subgroups. Several existing works directly study group fairness over protected attributes. The fairness evaluation is performed by using predefined groups [152], by considering single attributes [31, 71] or by automatically partitioning on the protected attributes allowing the identification of groups over any combinations of protected attributes [52]. Differently, we do not leverage predefined sensitive attributes and we conduct the exploration for all the attributes in input. The domain experts can decide to provide as input a selection of the attributes to focus only on the attribute they consider of interest.

Itemset	Sup	Δ_{ZFYA}	t
LSAT>41.0, UGPA>3.5, race=White, sex=Female	0.03	0.4115	11.1
LSAT>41.0, UGPA>3.5, race=White	0.07	0.4063	16.8
LSAT>41.0, UGPA>3.5, race=White, sex=Male	0.04	0.4025	13.0
LSAT>41.0, UGPA>3.5, sex=Male	0.04	0.3812	12.6
LSAT>41.0, UGPA>3.5	0.08	0.3761	15.9
LSAT≤33.0, race=Black, sex=Male	0.02	-1.0257	21.2
LSAT≤33.0, UGPA≤3.0, race=Black, sex=Male	0.01	-1.0049	17.5
LSAT≤33.0, race=Black	0.05	-0.9787	33.3
LSAT≤33.0, UGPA≤3.0, race=Black	0.03	-0.9757	27.2
race=Black, sex=Male	0.02	-0.9665	22.3

Table 6.7: Top-5 itemsets with highest and lowest ZFYA divergence for the Law School Dataset. The support threshold is $s = 0.005$.

Consider the Law School Dataset and the normalized first-year average of each student as outcome function $o(x)$. Table 6.7 lists the five itemsets with the greatest positive and negative divergence, among those with support at least $s = 0.005$, which corresponds to about 100 students. The table reports also the t-value of the divergence, computed according to Welch's t-test. From the results, we see that the itemset with the greatest positive divergence is {LSAT>41.0, UGPA>3.5, race=White, sex=Female}, for which the ZFYA-divergence is 0.41. The itemset with the greatest negative divergence is {LSAT≤33.0, race=Black, sex=Male}, for which the ZFYA score is on average lower by 1.03 compared with the dataset average.

Figure 6.13 reports the contribution of each item to the divergence for the itemset with the greatest negative divergence. We see that race=Black is the predominant factor, with LSAT≤33.0 giving a minor contribution, and sex=Male a negligible one. The predominant role of race stands out as a warning signal, indicating that this negative score divergence merits further investigation.

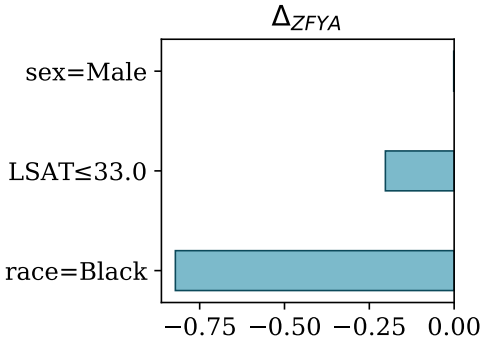


Figure 6.13: Contributions of individual items to the divergence of the frequent patterns having lowest ZFYA divergence for the *law* dataset ($s = 0.005$).

Ranking divergence.

A ranking is a permutation of the instances [174]. In a ranking system, every instance x has a *rank* $i(x) \in \mathbb{N}_{>0}$, where $i = 1$ is the top rank. Often rankings are derived from scores and scoring functions. In this scenario, we refer to them as score-based rankings [174]. Alternative, rankings may be derived using supervised learning methods [174].

A natural and intuitive way to define the outcome function o for rankings is via a *rank valuation function* $\gamma : \mathbb{N}_{>0} \mapsto \mathbb{R}$, where $\gamma(i)$ represents the value, to an instance, of being ranked in position i. We define the outcome of instance $x \in X$ via:

$$o(x) = \gamma(r(x)) \tag{6.13}$$

The outcome $h_o(I)$ of an itemset I then corresponds to the average value an instance in I receives from being ranked. The use of rank-value function γ allows the applicability also in contexts where the score or value associated with the ranking is not available and only the instance permutation is known.

We can consider, as an example, the admission process to university. Often applicants are ranked according to a certain scoring function and the top k are admitted. A possible rank evaluation function can be $\gamma(i) = 1$ for $i \leq k$ and $\gamma(i) = 0$ otherwise. The outcome $h_o(I)$ corresponds to the admission rate of I, that is, the fraction of applicants in I that are admitted. The divergence $\Delta_{h_o}(I)$ then represents how more, or less, likely applicants in I are to be admitted, compared with the general population.

Another rank evaluation function to model the relation between rank and benefit is with $\gamma(i) = i^{\alpha}$. This function is common in search applications, with $\alpha = -0.1$.

Consider again the Law School Dataset and consider for example internship applications. Student internship applications may be scored using the first-year average grade (ZFYA) of the students. The applications are then displayed to

Itemset	Sup	Δ_γ	t
LSAT>41.0, UGPA>3.5, race=White, sex=Female	0.03	0.0206	8.7
LSAT>41.0, UGPA>3.5, race=White	0.07	0.0196	13.0
LSAT>41.0, UGPA>3.5, race=White, sex=Male	0.04	0.0189	9.9
LSAT>41.0, UGPA>3.5, sex=Female	0.03	0.0185	8.3
LSAT>41.0, UGPA>3.5	0.08	0.0183	12.6
LSAT<=33.0, race=Black, sex=Male	0.02	-0.0283	25.6
LSAT<=33.0, UGPA<=3.0, race=Black, sex=Male	0.01	-0.0280	21.0
LSAT<=33.0, UGPA<=3.0, race=Black	0.03	-0.0278	31.4
LSAT<=33.0, race=Black	0.05	-0.0277	37.1
LSAT<=33.0, UGPA<=3.0, race=Black, sex=Female	0.02	-0.0276	24.6

Table 6.8: Top-5 itemsets with highest and lowest divergence for the Law School Dataset, for the internship example, with $\gamma(i) = i^{-0.1}$. The support threshold is $s = 0.005$.

internship hosts sorted according to their rank. This scenario is common for job applications, document and search relevance, and product rating.

Table 6.8 shows the itemsets with top and bottom divergence with respect to the rank evaluation function $\gamma(i) = i^{-0.1}$ where student applications are ranked according to the ZFYA. We see that the itemset that derives the most benefit out of internships would be {LSAT>41.0, UGPA>3.5, race=White, sex=Female}, and the one deriving the least benefit would be {LSAT≤33.0, race=Black, sex=Male}.

We can use the global Shapley value defined in 6.10 to inspect which are the most influencing terms to the γ-divergence. The results are reported in Figure 6.14. The most influencing terms to a higher than average ranking are the high scores of LSAT and UGPA. Correspondingly, a low LSAT score has a negative influence on high positions in the rankings. Worryingly, all ethnicities that differ from the Caucasian one ("race=White" in the dataset) have negative contributions. These ethnicities have generally a lower than average position in the rankings. The global Shapley value confirms the concerns firstly aroused by the local Shapley value (Figure 6.13) and the need for further assessments on the predominant role of race on the ranking.

The divergence analysis can reveal if the ranking impacts differently subgroups identified by protected attributes values. The automatic identification can hence be a tool for the evaluation of group fairness in rankings.

We remark that the generalization of divergence still includes classification tasks. We propose a redefinition of the boolean outcome function in Equation 6.3 so that it can be applied in the Definition 6.11 of outcome of an itemset I $h_o(I)$.

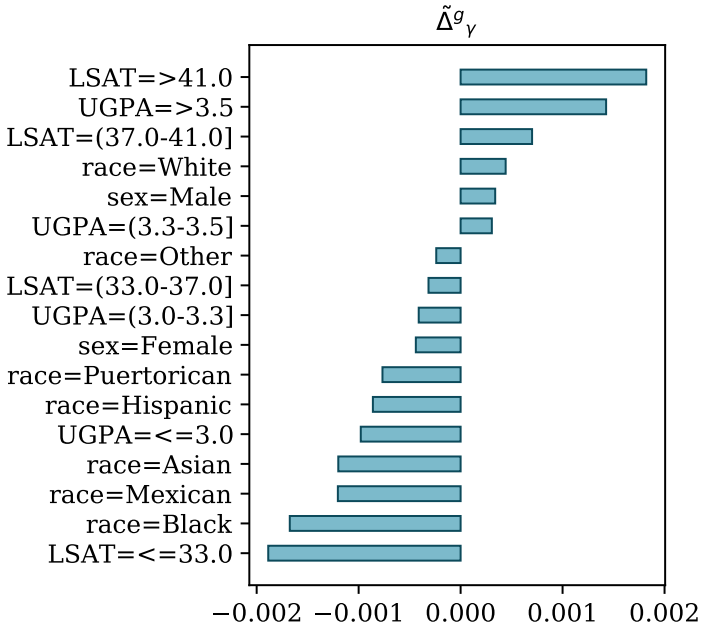


Figure 6.14: Global contributions via global Shapley values of items to γ-divergence for the Law School Dataset ranked with respect to the ZFYA score, with $\gamma = i^{-0.1}$ ($s = 0.005$).

Given a classifier, let $p(x) \in \{\text{T}, \text{F}\}$ be the predicted value for an instance x, and let $t(x)$ be the true value (ground truth). To capture the divergence of the false positive rate, we redefine the outcome function:

$$o(x) = \begin{cases} \perp & \text{if } t(x) = \text{T} \\ 0 & \text{if } t(x) = \text{F} \text{ and } p(x) = \text{F} \\ 1 & \text{if } t(x) = \text{F} \text{ and } p(x) = \text{T} \end{cases}$$

for $x \in X$. The outcome $\perp$ is used to exclude from the statistic computation the true positives. In this way, the outcome $h_o(I)$ of an itemset I is its false positive rate. Outcome functions for capturing classification performance as accuracy, false negative rate, true positive rate, can be similarly defined.

6.6 Remarks and conclusions

In this chapter, we propose the notion of divergence over itemsets as a measure of different classification behaviors in subsets of a given dataset. A solid theoretical foundation, based on Shapley values, is proposed to quantify divergence contributions, both for pattern and dataset. The concept of divergence also allows capturing

interesting item behaviors, for example, corrective items. An efficient algorithm for divergence computation is provided and an extended experimental evaluation shows the effectiveness and efficiency of the proposed approach. Finally, we show how the notion of divergence can be generalized to multiple contexts as scoring and ranking systems.

Given the generality of the divergence notion, as future work, we plan to study its extension to other data science tasks, including, e.g., the preprocessing tasks.

Chapter 7

DivExplorer interactive system

This chapter illustrates the DivExplorer interactive system for the interactive inspection of classifiers behavior in data subgroups and its functionalities [130]. The interactive tool leverages the DivExplorer algorithm for classification tasks, presented in Chapter 6. The tool enables users to analyze the behavior of a classifier over a dataset, and to identify the portions of the dataset on which the classifier behaves in an anomalous fashion, for example, by exhibiting particularly high false-positive or false-negative rates. If the underlying data has ethical implications, DivExplorer tool can be used to identify subgroups of data for which the behavior of the classifier is problematic, and to analyze the factors that contribute to the problematic performance.

The chapter is organized as follows. Section 7.1 specifies the tool implementation details and illustrates the analysis setup. Section 7.2 presents the tool functionalities. Finally, Section 7.3 draws the conclusions.

7.1 The DivExplorer tool

In this section, we outline the tool specification and the analysis setup.

7.1.1 Tool Implementation

DivExplorer is implemented as a web app, and it can be deployed on any cloud that provides services for running containerized web services. Our hosting relies on Google Appengine [1]. The back-end, which implements the data access layers and analysis algorithms, is written in Python, and relies on the py4web web framework [133]. The dataset analysis operations are implemented on top of the Pandas library for dataset processing [109], and the scikit-learn library for data

[1]Source code available in our project page http://divexplorer.github.io

mining [131]. The front-end is written using the vue.js Javascript framework, which enables dynamic visualizations and explorations of the dataset. Data is stored in a cloud SQL database. In particular, we use Google Cloud Mysql and Google Cloud Storage.

7.1.2 Analysis Setup

DivExplorer analyzes datasets in tabular form. Rows represent dataset *instances*, and columns are the *attributes*. The DivExplorer tool assumes that continuous attributes are *discretized*. Hence, attributes that are continuous in nature, such as age, or income, need to be discretized into fixed ranges prior to the dataset being input to DivExplorer. However, the classifier itself can use the original, non-discretized data to build its model. Discretization is used only for data exploration, not for classification. We are considering, as future work, integrating the discretization step directly in DivExplorer. Finally, DivExplorer requires users to identify two columns: the *true* class of each instance, and the *predicted* class of each instance as output by the classifier.

Before analyzing a dataset with the DivExplorer system, the dataset is uploaded and the two attributes (columns) corresponding to the predicted class and the ground-truth class are identified. The only parameter to be configured is the support threshold. DivExplorer analyzes the behavior of the classifier on all itemsets whose support is above a specified support threshold s. As discussed in Section 6.1.3, specifying a support threshold has two related purposes. First, it excludes from the analysis itemsets with few instances, on which the analysis would be affected by statistical fluctuations. Second, it reduces the number of itemsets under consideration, leading to a tool output of manageable size.

As running example, we use as in Chapter 6 the *COMPAS* dataset [8]. *COMPAS* contains demographic information and criminal history of defendants. We recall that for each defendant, the dataset provides a score (called the *COMPAS* score), produced by a classifier, and intended to reflect the likelihood that the defendant will commit crimes in the future (recidivate). The dataset also contains ground truth information on which defendants actually recidivated. For the following example, we set the minimum support threshold to 0.05. Thus, only itemsets supported by 5% or more of the dataset instances are considered.

7.2 DivExplorer tool functionalities

The DivExplorer interactive features allow users to dynamically explore the behavior of a classifier on an arbitrary dataset. The overall goal of the implementation of the UI is to allow users to dynamically explore the behavior of the classifier on the dataset.

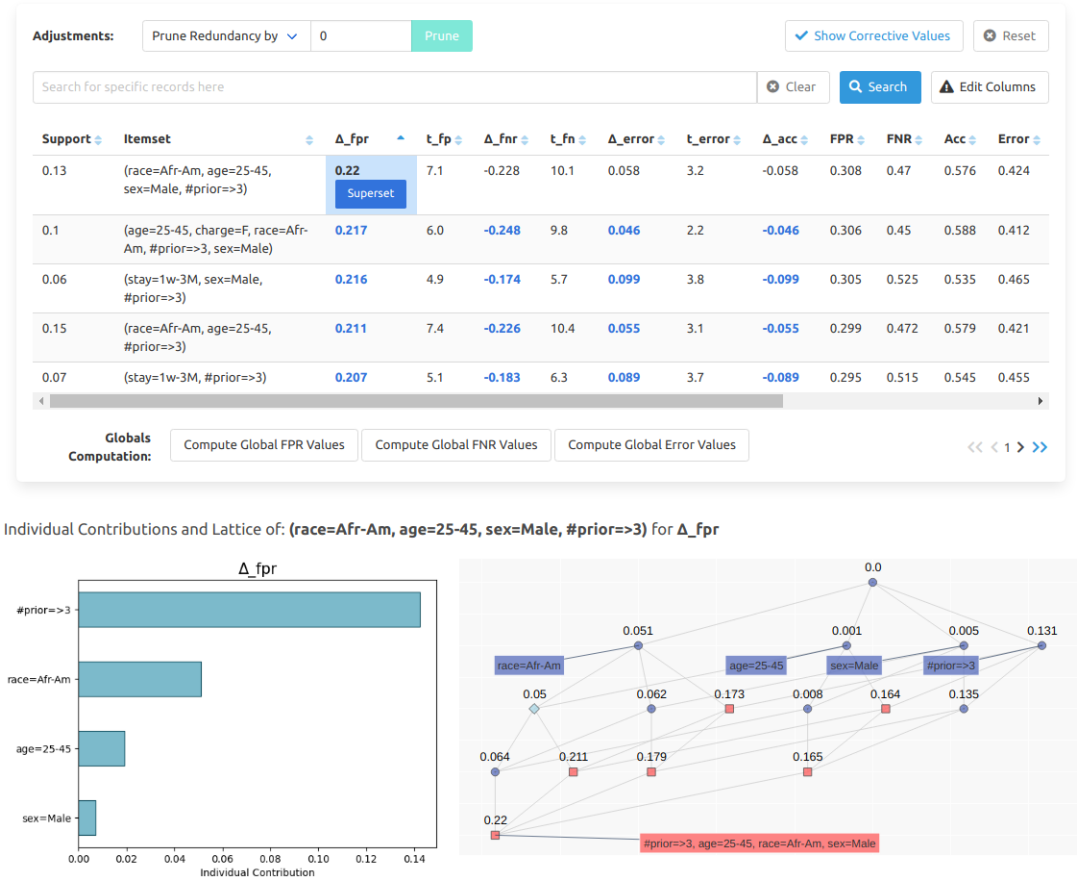


Figure 7.1: DivExplorer main UI.

DivExplorer first provides global information, such as the list of itemsets where the classifier behavior most deviates from the average (Section 7.2.1). Next, users can drill into specific regions of data — specific itemsets, and explore the roles of individual attributes in affecting the classifier behavior, both considering specific itemsets (Section 7.2.2) and on a global scale over the dataset (Section 7.2.3).

7.2.1 Divergent itemsets

DivExplorer analyzes the dataset and computes the *divergence* of itemsets with respect to error metrics such as false-positive, false-negative, and error rates [126]. DivExplorer computes the divergences for all itemsets above the specified support size and displays them in a table that can be sorted according to any characteristics, as depicted in Figure 7.1.

Many of the most divergent itemsets in Figure 7.1 are quite similar. To obtain a more insightful summarization, DivExplorer enables users to prune (redundant) itemsets by specifying a threshold ϵ. It leverages the heuristic pruning approach

defined in 6.1.4. If two itemsets I and $I \cup \{A\}$ have divergences closer than ϵ, only the smaller itemset I is included in the output. In this example, summarizing with $\epsilon = 0.02$, the total number of patterns is reduced from 313 to 51.

7.2.2 Measuring the Role of Items in Itemset Divergence

On DivExplorer divergence table, the users can select the individual divergence values for detailed inspection. In Figure 7.1, the user has selected Δ_{FPR} for the top itemset *{age=25-45, #prior>3, race=Afr-Am, sex=Male}*. Thus, the details for the FPR-divergence for this itemset are displayed at the bottom.

On the bottom left is a bar graph indicating the extent to which the individual items contributed to the divergence of the itemset. The contributions are derived using the notion of Shapley values, as defined in Section 6.2.1. Most of the divergence of this itemset (most of the excess false-positive rate) is due to having more than 3 prior offenses, with *race=Afr-Am* as the second most important factor.

On the bottom right, similar information is conveyed via the lattice of subsets of the selected itemset. Each subset is labeled with its own divergence, and by hovering on it, the user can see which of the itemset's items appear in the subset. The lattice view is particularly useful when items are correlated. In this case, the Shapley value splits the contribution among correlated items. The itemset lattice shows the precise effect of each item in each context.

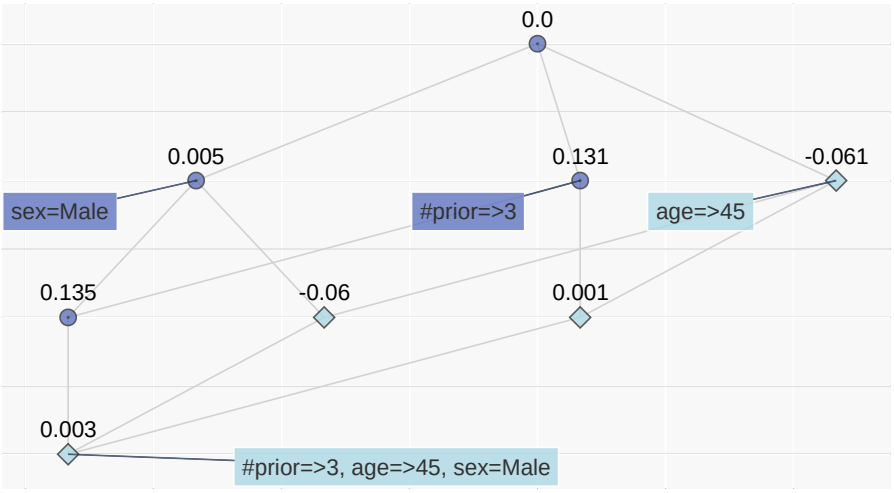


Figure 7.2: Lattice showing a corrective phenomenon for FNR divergence on the *COMPAS* dataset (rhombus nodes).

The lattice visualization is also useful for studying the role of *corrective items,* which are items that, when added to an itemset, *reduce* the divergence. For instance, Figure 7.2 shows that adding the *age>45* item to the *{#prior>3, sex=Male}* itemset *reduces* the false-negative divergence from 0.13 to 0.003. This indicates that

the bias corresponding to *{#prior>3, sex=Male}*, upon closer inspection, affects only people below 45.

7.2.3 Item Influence on the Entire Dataset

The DivExplorer system also enables users to examine the influence of each item on the divergence of the entire dataset. This can be done in two different ways, both illustrated in Figure 7.3. The simplest measure, named the *individual divergence* of an item, is simply the divergence of an item in isolation (Section 6.2.1). The *global divergence* of an item is a generalization of the Shapley value to the entire set of all items (Section 6.2.4). Intuitively, the global divergence summarizes how much an item increases the divergence when added to any itemset with which it is compatible (i.e., with which it does not share any attribute). From Figure 7.3, we see that the top item for global false-positive divergence, *#prior>3*, is part of the itemset that has overall greatest divergence.

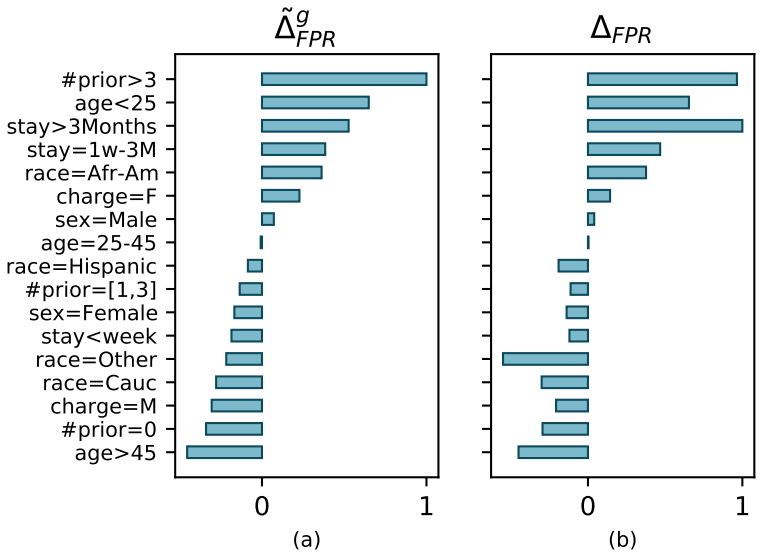


Figure 7.3: Relative magnitudes of global Shapley value and individual item divergence, for false-positive rate in the *COMPAS* dataset with s=0.05.

7.2.4 Drill-down and interactive searches

The global Shapley value reveal that *age<25* and *stay>3Months* have a high contribution to the false-positive divergence. Interestingly, these two items do not belong to the most divergent itemsets (reported in Figure 7.1). To examine more

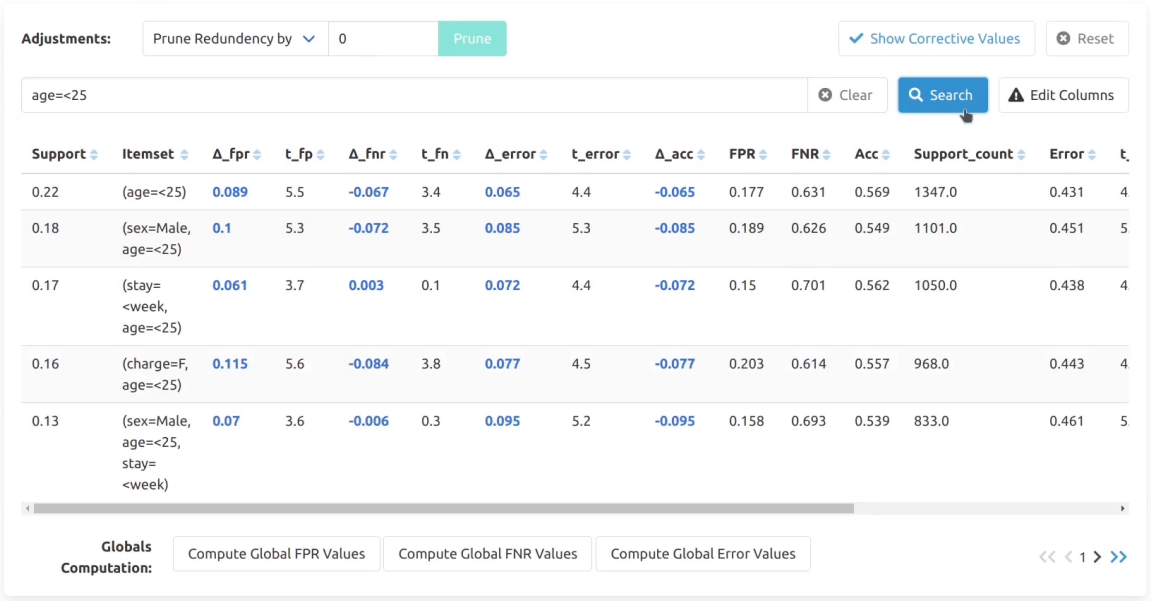


Figure 7.4: Search functionality of DivExplorer tool.

closely their role, the user can use the *search function* in DivExplorer and examine the itemsets in which these items appear. Figure 7.4 shows the results of the search for the 1-length pattern *age<25*.

The DivExplorer tool offers the possibility *of searching over supersets* of an itemset of interest I. The system automatically selects and shows all the supersets (i.e. itemsets that contain I) and their relative divergence score and statistics. Figure 7.1 shows the button search for supersets for the most divergent itemset.

7.3 Remarks and conclusion

In this chapter, we present the DivExplorer web app, a tool that enables users to explore datasets and find subgroups of data for which a classifier behaves in an anomalous manner. These subgroups, denoted as divergent subgroups, may exhibit, for example, higher-than-normal false positive or negative rates. The web application leverages on the DivExplorer [126] algorithm described in Chapter 6.

Users can upload their labeled dataset coupled with the predictions made by a generic classifier. Hence, the approach is model agnostic. The DivExplorer system automatically runs the identification of the divergent patterns. Users can interactively inspect the identified subgroups by analyzing the local contribution via Shapley values, visualizing the lattice graph, selecting supersets, and searching specific subgroups.

DivExplorer can be used to analyze and debug classifiers. If the data has

ethical or social implications, DivExplorer can be also used to identify bias in classifiers. As future work, we plan to extend the support for multiple scenarios as scoring ranking and systems and preprocessing tasks.

Chapter 8

Conclusions and future work

The research activity described in this thesis addressed the lack of transparency of classification models. Black-box models are adopted in multiple high-stake applications despite their inability to disclose their inner workings. Especially in high-stake areas as health care, criminal justice, and insurance, understanding the model behavior is essential. Through model explainability, data scientists and domain experts can assess fundamental properties of the model as its trustworthiness and its fair and not-discriminatory decision-making process. We propose post-hoc explainability techniques to enhance the interpretability of classifiers from both the individual and subgroup perspectives. The proposed techniques leverage the notion of patterns. Patterns are conjunctions of attribute-value value pairs intrinsically interpretable which allow us to capture relevant associations of attribute values and identify subgroups in the attribute domain. The proposed methods focus on structure data and are model agnostic, applicable to a generic classification model.

In Chapter 1, we debate the need for explainability and its related concepts and desiderata. Chapter 2 outlines the related work while Chapter 3 illustrates basic background notions and algorithms that are used through the thesis.

Chapter 4 focuses on enhancing model interpretability from the individual instance perspective. We propose LACE [129] an explanation method to explain the reasons behind individual predictions. The approach exploits a removal-based technique to quantify the influence on the prediction of individual feature values and relevant subgroups of feature values. We use the notion of prediction difference to compute the prediction change when one or more feature values are omitted. The relevant subsets of attribute values are determined using a local surrogate model. The local model is an associative classifier learned in the neighborhood of the individual prediction to explain. It captures relevant association of attribute values in form of local rules that provide a qualitative understanding of the reasons behind the prediction. The local model allows overcoming the exponential time complexity of the enumeration of all possible subsets of feature values for the removal-based

estimation. We quantify the relevance through the prediction difference only of the patterns, i.e. subsets of attribute values, extracted by the local rules. Experimental results show the ability of LACE to capture the reasons behind individual predictions, identifying the feature values and relevant patterns that determine them.

Interactivity is one of the goals and desiderata of explainable AI research [12]. The interaction between the model and the users allows them to investigate their assumptions on the model behavior. In Chapter 5, we propose X-PLAIN [127] an interactive tool for human-in-the-loop inspections of the reasons behind model predictions. The tool, which leverages on LACE as explanation method, allows users to actively speculate on the model behavior, to perform what-if analysis and it extracts explanation metadata characterizing the model behavior.

Chapter 6 addresses the understanding a model peculiar behavior from the subgroup perspective. We propose DIVEXPLORER [126], a novel approach to identify and characterize subgroups of data in which a classification model behaves differently. The identification of these critical subgroups is relevant in many applications as model validation and debugging and the assessment of the model fairness. Considering the use of machine learning models in high-stake tasks involving individuals, increasing concerns arise on the potential discriminating behavior of models over data subgroups. The analysis of peculiar behavior in subgroups may reveal if they are characterized by sensitive attributes as ethnicity and gender. We introduce the notion of divergence to capture the different behavior of the model on subgroups with respect to the overall behavior. We propose an automatic identification of all subgroups with adequate representation in the data. The exploration leverages frequent pattern mining techniques for the complete analysis of subgroups with adequate frequency. We use the notion of Shapley value to characterize each data subgroup and understand the feature values most influencing the pattern divergence. We also propose a generalization of the Shapley value for the global computation of divergence. Global item divergence indicates the feature values that mostly contribute to the divergent behavior of the classifier. Theoretical analyses and experimental evaluations show the effectiveness of DIVEXPLORER in revealing the peculiar behavior of classification models in data subgroups. We also show how the notion of the divergence and DIVEXPLORER can be generalized to other contexts beyond classification tasks as scoring and rankings.

Considering again the desiderata of interactivity, in Chapter 7 we propose a web app that leverages DIVEXPLORER [124] for the interactive exploration of classifier behavior in data subgroups. The interactive system supports drill-down operations, interactive searches, and human-in-the-loop inspection of subgroups of data for which a classifier behaves in an anomalous manner.

8.1 Future work

This section outlines enhancements and opportunities for future work.

The individual explanations provided by our explanation approach highlight the single features and subsets of attribute values that are relevant for the prediction. The explanations are hence provided both in form of feature importance and local rules. Another interesting explanation form is represented by counterfactual explanations which express small changes in the attribute values of individual instances that change the prediction. We plan to exploit the inner computations of the prediction difference based on feature omission to derive changes in the prediction outcome and hence deriving counterfactual explanations.

The proposed local pattern explanations reveal the statistical associations for the observed instance and the class label learned by the classier. However, associations do not reveal the causal dependencies among the observed variables and the class. In multiple contexts, the assessment of causality is a mandatory requirement. Several approaches have hence been proposed to reveal causal relationships and structures among the data as for discrimination discovery [24, 177]. We are thus interested in exploring the link between the concept of pattern explanations and causal reasoning to identify causality.

We then consider extending the approach to deal with very large data sets for big data applications. In this domain, considering the intrinsic sparsity of the data, the local nature of the approach could be suitable in highlighting local feature values and specific associations.

The automatic approach for the identification of divergent subgroups leverage discretized structured data. We firstly plan to propose an automatic method to derive a proper discretization of continuous attributes that consider divergence criteria. The goal of the discretization algorithm is to highlight the attribute slices that produce the highest and lowest divergence, to inspect both most and less problematic behaviors.

We then envision to propose a novel identification approach that can directly handle continuous attributes. We also consider alternatives to data slicing in the attribute domain as by exploring the notion of examples as descriptors of subgroups.

DivExplorer is currently designed for a single node and in memory. To improve scalability, we plan to extend the algorithm for an efficient distributed parallelization in a cluster of nodes.

We show how the notion of divergence can be generalized to multiple applications of machine learning beyond classification as rankings. We will continue to work in these contexts and we plan to specifically consider ethical concerns and algorithmic bias.

The results provided by our approach DivExplorer provide useful insights on the model behavior at the subgroup level and its potential discriminating behavior.

The automatic identification of divergent data subgroups and their characterization can in fact reveal if a different model behavior is observed for sensitive attributes. We are firstly interested in studying the relations of the notion of divergence with existing metrics for fairness assessment. We then plan to work on the design of mitigation strategies of algorithm bias using divergence analysis and exploiting DivExplorer results.

Finally, we plan to extend the notion of divergence and its analysis for unstructured data as images and text. Specifically for text, as the first extension, we plan to apply our technique to textual datasets by considering words as target items. We then plan to propose to develop approaches that can cope with the sparsity nature of text data. Another interesting line of work consists in including the notion of semantic and taxonomy. The idea is to firstly capture high-level abstractions of the data. We then leverage data abstractions to group the data and identify divergent subgroups.

Bibliography

[1] Carlo Abrate and Francesco Bonchi. "Counterfactual Graphs for Explainable Classification of Brain Networks". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD '21. Virtual Event, Singapore: Association for Computing Machinery, 2021, pp. 2495–2504. ISBN: 9781450383325. DOI: 10.1145/3447548.3467154.

[2] Bruno Agard and Andrew Kusiak. "Data mining for subassembly selection". In: *J. Manuf. Sci. Eng.* 126.3 (2004), pp. 627–631. DOI: 10.1115/1.1763182.

[3] Rakesh Agrawal and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules in Large Databases". In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. ISBN: 1558601538.

[4] Elvio Amparore, Alan Perotti, and Paolo Bajardi. "To trust or not to trust an explanation: using LEAF to evaluate local linear XAI methods". In: *PeerJ Computer Science* 7 (2021), e479. DOI: 10.7717/peerj-cs.479.

[5] TensorFlow Model Analysis. *Introducing TensorFlow Model Analysis: Scaleable, Sliced, and Full-Pass Metrics*. https://medium.com/tensorflow/introducing-tensorflow-model-analysis-scaleable-sliced-and-full-pass-metrics-5cde7baf0b7b. 2018.

[6] Robert Andrews, Joachim Diederich, and Alan B Tickle. "Survey and critique of techniques for extracting rules from trained artificial neural networks". In: *Knowledge-based systems* 8.6 (1995), pp. 373–389. DOI: 10.1016/0950-7051(96)81920-4.

[7] Elaine Angelino et al. "Learning Certifiably Optimal Rule Lists for Categorical Data". In: *Journal of Machine Learning Research* 18 (2018), pp. 1–78.

[8] Julia Angwin et al. *Machine Bias*. May 2016. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[9] Mihael Ankerst et al. “OPTICS: Ordering points to identify the clustering structure”. In: *ACM Sigmod record* 28.2 (1999), pp. 49–60. DOI: 10.1145/304181.304187.

[10] Daniele Apiletti and Eliana Pastor. “Correlating Espresso Quality with Coffee-Machine Parameters by Means of Association Rule Mining”. In: *Electronics* 9.1 (2020), p. 100. DOI: 10.3390/electronics9010100.

[11] Leila Arras et al. “Explaining Recurrent Neural Network Predictions in Sentiment Analysis”. In: *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. 2017, pp. 159–168.

[12] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information Fusion* 58 (2020), pp. 82–115. DOI: 10.1016/j.inffus.2019.12.012.

[13] Irit Askira-Gelman. “Knowledge discovery: comprehensibility of the results”. In: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*. Vol. 5. 1998, 247–255 vol.5. DOI: 10.1109/HICSS.1998.648319.

[14] Abolfazl Asudeh, Zhongjun Jin, and H.V. Jagadish. “Assessing and Remedying Coverage for a Given Dataset”. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, pp. 554–565. DOI: 10.1109/ICDE.2019.00056.

[15] Giuseppe Attanasio. *g8a9/l3wrapper: Add parallelization and rule sets modifiers*. Version v0.7.0. July 2020. DOI: 10.5281/zenodo.3935385.

[16] Nahla Barakat and Andrew P. Bradley. “Rule extraction from support vector machines: a review”. In: *Neurocomputing* 74.1-3 (2010), pp. 178–190. DOI: 10.1016/j.neucom.2010.02.016.

[17] Elena Baralis, Silvia Chiusano, and Paolo Garza. “A lazy approach to associative classification”. In: *IEEE Transactions on Knowledge and Data Engineering* 20.2 (2008), pp. 156–171. DOI: 10.1109/TKDE.2007.190677.

[18] Solon Barocas and Andrew D. Selbst. “Big Data’s Disparate Impact”. In: *California Lab Review* 104 (2016), p. 671. DOI: 10.2139/ssrn.2477899.

[19] Denis Baylor et al. “TFX: A TensorFlow-Based Production-Scale Machine Learning Platform”. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’17. Halifax, NS, Canada: Association for Computing Machinery, 2017, pp. 1387–1395. ISBN: 9781450348874. DOI: 10.1145/3097983.3098021.

[20] Barry Becker, Ron Kohavi, and Dan Sommerfield. "Visualizing the simple Bayesian classifier". In: *Information visualization in data mining and knowledge discovery* 18 (2001), pp. 237–249.

[21] Rachel K. E. Bellamy et al. "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias". In: *IBM Journal of Research and Development* 63.4/5 (2019), 4:1–4:15. DOI: 10.1147/JRD.2019.2942287.

[22] Adrien Bibal and Benoit Frenay. "Interpretability of Machine Learning Models and Representations: an Introduction". In: *24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 2016, pp. 77–82.

[23] Or Biran and Courtenay Cotton. "Explanation and justification in machine learning: A survey". In: *IJCAI-17 workshop on explainable AI (XAI)*. Vol. 8. 1. 2017, pp. 8–13.

[24] Francesco Bonchi et al. "Exposing the probabilistic causal structure of discrimination". In: *International Journal of Data Science and Analytics* 3.1 (2017), pp. 1–21. DOI: 10.1007/s41060-016-0040-z.

[25] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[26] Ángel Alexander Cabrera et al. "FairVis: Visual analytics for discovering intersectional bias in machine learning". In: *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE. 2019, pp. 46–56. DOI: 10.1109/VAST47406.2019.8986948.

[27] Doina Caragea, Dianne Cook, and Vasant G. Honavar. "Gaining Insights into Support Vector Machine Pattern Classifiers Using Projection-based Tour Methods". In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '01. San Francisco, California: ACM, 2001, pp. 251–256. ISBN: 1-58113-391-X. DOI: 10.1145/502512.502547.

[28] Rich Caruana et al. "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: ACM, 2015, pp. 1721–1730. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2788613.

[29] L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. "Ranking with fairness constraints". In: *arXiv preprint* (2018). arXiv: 1704.06840 [cs.DS].

[30] Jadzia Cendrowska. "PRISM: An algorithm for inducing modular rules". In: *International Journal of Man-Machine Studies* 27.4 (1987), pp. 349–370. DOI: 10.1016/S0020-7373(87)80003-2.

[31] Le Chen et al. "Investigating the Impact of Gender on Rank in Resume Search Engines". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–14. ISBN: 9781450356206. DOI: 10.1145/3173574.3174225.

[32] Wei-Chou Chen, Shian-Shyong Tseng, and Ching-Yao Wang. "A novel manufacturing defect detection method using association rule mining techniques". In: *Expert systems with applications* 29.4 (2005), pp. 807–815. DOI: 10.1007/978-3-540-24677-0_9.

[33] Yeounoh Chung et al. "Automated Data Slicing for Model Validation: A Big data - AI Integration Approach". In: *IEEE Transactions on Knowledge and Data Engineering* (2019). DOI: 10.1109/TKDE.2019.2916074.

[34] Yeounoh Chung et al. "Slice Finder: Automated Data Slicing for Model Validation". In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, pp. 1550–1553. DOI: 10.1109/ICDE.2019.00139.

[35] Peter Clark and Tim Niblett. "The CN2 induction algorithm". In: *Machine learning* 3.4 (1989), pp. 261–283. DOI: 10.1007/BF00116835.

[36] William W Cohen. "Fast effective rule induction". In: *Machine learning proceedings 1995*. Elsevier, 1995, pp. 115–123. DOI: 10.1016/B978-1-55860-377-6.50023-2.

[37] Ian Covert, Scott Lundberg, and Su-In Lee. "Explaining by Removing: A Unified Framework for Model Explanation". In: *arXiv preprint* (2020). arXiv: 2011.14878 [cs.LG].

[38] Mark W. Craven and Jude W. Shavlik. "Extracting Tree-structured Representations of Trained Networks". In: *Proceedings of the 8th International Conference on Neural Information Processing Systems*. NIPS'95. Denver, Colorado: MIT Press, 1995, pp. 24–30. URL: http://dl.acm.org/citation.cfm?id=2998828.2998832.

[39] Catherine Da Cunha, Bruno Agard, and Andrew Kusiak. "Data mining for improvement of product quality". In: *International journal of production research* 44.18-19 (2006), pp. 4027–4041. DOI: 10.1080/00207540600678904.

[40] Piotr Dabkowski and Yarin Gal. "Real Time Image Saliency for Black Box Classifiers". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6970–6979. ISBN: 9781510860964.

[41] Susanne Dandl et al. "Multi-objective counterfactual explanations". In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2020, pp. 448–469. DOI: 10.1007/978-3-030-58112-1_31.

[42] Houtao Deng. "Interpreting tree ensembles with intrees". In: *International Journal of Data Science and Analytics* 7.4 (2019), pp. 277–287. DOI: 10.1007/s41060-018-0144-8.

[43] Vincenzo Di Cicco et al. "Interpreting Deep Learning Models for Entity Resolution: An Experience Report Using LIME". In: *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. aiDM '19. Amsterdam, Netherlands: Association for Computing Machinery, 2019. ISBN: 9781450368025. DOI: 10.1145/3329859.3329878.

[44] Evelina Di Corso, Tania Cerquitelli, and Daniele Apiletti. "Metatech: Meteorological data analysis for thermal energy characterization by means of self-learning transparent models". In: *Energies* 11.6 (2018), p. 1336. DOI: 10.3390/en11061336.

[45] Collins Dictionaries. "Collins English Dictionary - Complete & Unabridged 10th Edition". In: (Aug. 2016).

[46] Webster's New World College Dictionaries. "Webster's New World College Dictionary, 4th Edition". In: (2010).

[47] Cambridge dictionary. "Cambridge dictionary, 4th Edition". In: (2013).

[48] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: *arXiv* (2017). URL: https://arxiv.org/abs/1702.08608.

[49] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[50] Cynthia Dwork and Christina Ilvento. "Group fairness under composition". In: *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency (FAT* 2018)*. 2018.

[51] Lilian Edwards and Michael Veale. "Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for". In: *Duke L. & Tech. Rev.* 16 (2017), p. 18. DOI: 10.2139/ssrn.2972855.

[52] Shady Elbassuoni, Sihem Amer-Yahia, and Ahmad Ghizzawi. "Fairness of Scoring in Online Job Marketplaces". In: *ACM Transactions on Data Science* 1.4 (2020), pp. 1–30. DOI: 10.1145/3402883.

[53] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[54] Council of the European Union and European Parliament. "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)". In: vol. L119. 2016.

[55] Cao Feng and Donald Michie. "Machine learning of rules and trees". In: *Machine learning, neural and statistical classification* (1994), pp. 50–83.

[56] Ruth Fong and Andrea Vedaldi. "Interpretable Explanations of Black Boxes by Meaningful Perturbation". In: *CoRR* abs/1704.03296 (2017). arXiv: 1704.03296. URL: http://arxiv.org/abs/1704.03296.

[57] James R Foulds et al. "An intersectional definition of fairness". In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE. 2020, pp. 1918–1921. DOI: 10.1109/ICDE48307.2020.00203.

[58] William J Frawley, Gregory Piatetsky-Shapiro, and Christopher J Matheus. "Knowledge discovery in databases: An overview". In: *AI magazine* 13.3 (1992), pp. 57–57. DOI: 10.1609/aimag.v13i3.1011.

[59] Alex A. Freitas. "Comprehensible Classification Models: A Position Paper". In: *SIGKDD Explor. Newsl.* 15.1 (Mar. 2014), pp. 1–10. ISSN: 1931-0145. DOI: 10.1145/2594473.2594475.

[60] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[61] Glenn Fung, Sathyakama Sandilya, and R Bharat Rao. "Rule extraction from linear support vector machines". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 32–40. DOI: 10.1145/1081870.1081878.

[62] Alex Goldstein et al. "Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation". In: *Journal of Computational and Graphical Statistics* 24.1 (2015), pp. 44–65. DOI: 10.1080/10618600.2014.907095.

[63] Matteo Golfarelli and Stefano Rizzi. *Data warehouse design: Modern principles and methodologies*. McGraw-Hill, Inc., 2009.

[64] Bryce Goodman and Seth Flaxman. "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"". In: *AI Magazine* 38.3 (2017), pp. 50–57. DOI: 10.1609/aimag.v38i3.2741.

[65] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[66] Riccardo Guidotti. "Evaluating local explanation methods on ground truth". In: *Artificial Intelligence* 291 (2021), p. 103428. DOI: 10.1016/j.artint.2020.103428.

[67] Riccardo Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (Aug. 2018). ISSN: 0360-0300. DOI: 10.1145/3236009.

[68] Riccardo Guidotti et al. "Local Rule-Based Explanations of Black Box Decision Systems". In: *CoRR* abs/1805.10820 (2018). arXiv: 1805.10820. URL: http://arxiv.org/abs/1805.10820.

[69] Patrick Hall and Navdeep Gill. *Introduction to Machine Learning Interpretability*. O'Reilly Media, Incorporated, 2018.

[70] Jiawei Han, Jian Pei, and Yiwen Yin. "Mining Frequent Patterns without Candidate Generation". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*. Ed. by Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein. ACM, 2000, pp. 1–12. DOI: 10.1145/342009.335372.

[71] Anikó Hannák et al. "Bias in Online Freelance Marketplaces: Evidence from TaskRabbit and Fiverr". In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. CSCW '17. Portland, Oregon, USA: Association for Computing Machinery, 2017, pp. 1914–1933. ISBN: 9781450343350. DOI: 10.1145/2998181.2998327.

[72] Satoshi Hara and Kohei Hayashi. "Making tree ensembles interpretable". In: (2016). arXiv: 1606.05390 [stat.ML].

[73] Maaike Harbers, Karel van den Bosch, and John-Jules Meyer. "Design and evaluation of explainable BDI agents". In: *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 2. IEEE. 2010, pp. 125–132. DOI: 10.1109/WI-IAT.2010.115.

[74] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[75] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7.

[76] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7.

[77] Johan Huysmans et al. “An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models”. In: *Decision Support Systems* 51.1 (2011), pp. 141–154. DOI: 10.1016/j.dss.2010.12.003.

[78] Aleks Jakulin et al. “Nomograms for Visualizing Support Vector Machines”. In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD ’05. Chicago, Illinois, USA: ACM, 2005, pp. 108–117. ISBN: 1-59593-135-X. DOI: 10.1145/1081870.1081886. URL: http://doi.acm.org/10.1145/1081870.1081886.

[79] Dominik Janzing, Lenon Minorics, and Patrick Bloebaum. “Feature relevance quantification in explainable AI: A causal problem”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2907–2916. URL: https://proceedings.mlr.press/v108/janzing20a.html.

[80] Yunzhe Jia et al. “Exploiting patterns to explain individual predictions”. In: *Knowledge and Information Systems* (2019), pp. 1–24. DOI: 10.1007/s10115-019-01368-9.

[81] Yunzhe Jia et al. “Improving the Quality of Explanations with Local Embedding Perturbations”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 875–884. ISBN: 9781450362016. DOI: 10.1145/3292500.3330930.

[82] Shengyi Jiang et al. “An improved K-nearest-neighbor algorithm for text categorization”. In: *Expert Systems with Applications* 39.1 (2012), pp. 1503–1509. DOI: 10.1016/j.eswa.2011.08.040.

[83] Zhongjun Jin et al. “MithraCoverage: A System for Investigating Population Bias for Intersectional Fairness”. In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 2721–2724. ISBN: 9781450367356. DOI: 10.1145/3318464.3384689.

[84] M. Joglekar, H. Garcia-Molina, and A. Parameswaran. “Interactive Data Exploration with Smart Drill-Down”. In: *IEEE Transactions on Knowledge & Data Engineering* 31.01 (2019), pp. 46–60. ISSN: 1558-2191. DOI: 10.1109/TKDE.2017.2685998.

[85] Minsuk Kahng, Dezhi Fang, and Duen Horng (Polo) Chau. “Visual Exploration of Machine Learning Results Using Data Cube Analysis”. In: *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. HILDA ’16. San Francisco, California: Association for Computing Machinery, 2016. ISBN: 9781450342070. DOI: 10.1145/2939502.2939503.

[86] Amir-Hossein Karimi et al. "Model-agnostic counterfactual explanations for consequential decisions". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 895–905. URL: https://proceedings.mlr.press/v108/karimi20a.html.

[87] Michael J. Kearns et al. "Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2569–2577. URL: http://proceedings.mlr.press/v80/kearns18a.html.

[88] Matt Kusner et al. "Counterfactual Fairness". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4069–4079. ISBN: 9781510860964.

[89] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. "Interpretable decision sets: A joint framework for description and prediction". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1675–1684. DOI: 10.1145/2939672.2939874.

[90] Pat Langley et al. "Explainable Agency for Intelligent Autonomous Systems". In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*. Ed. by Satinder P. Singh and Shaul Markovitch. AAAI Press, 2017, pp. 4762–4764. URL: http://aaai.org/ocs/index.php/IAAI/IAAI17/paper/view/15046.

[91] Thibault Laugel et al. "Defining Locality for Surrogates in Post-hoc Interpretablity". In: *Workshop on Human Interpretability for Machine Learning (WHI)-International Conference on Machine Learning (ICML)*. 2018.

[92] Nada Lavrač. "Selected techniques for data mining in medicine". In: *Artificial intelligence in medicine* 16.1 (1999), pp. 3–23. DOI: 10.1016/S0933-3657(98)00062-1.

[93] Jing Lei et al. "Distribution-Free Predictive Inference for Regression". In: *Journal of the American Statistical Association* 113.523 (2018), pp. 1094–1111. DOI: 10.1080/01621459.2017.1307116.

[94] V. Lemaire, R. Feraud, and N. Voisine. "Contact personalization using a score understanding method". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. June 2008, pp. 649–654. DOI: 10.1109/IJCNN.2008.4633863.

[95] Wenmin Li, Jiawei Han, and Jian Pei. "CMAR: Accurate and efficient classification based on multiple class-association rules". In: *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE. 2001, pp. 369–376. DOI: 10.1109/ICDM.2001.989541.

[96] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[97] Jimmy Lin et al. "Generalized and Scalable Optimal Sparse Decision Trees". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6150–6160. URL: https://proceedings.mlr.press/v119/lin20g.html.

[98] Zachary C Lipton. "The mythos of model interpretability". In: *arXiv preprint* (2017). arXiv: 1606.03490 [cs.LG].

[99] Bing Liu, Wynne Hsu, and Yiming Ma. "Integrating Classification and Association Rule Mining". In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. KDD'98. New York, NY: AAAI Press, 1998, pp. 80–86. URL: http://dl.acm.org/citation.cfm?id=3000292.3000305.

[100] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[101] Yin Lou, Rich Caruana, and Johannes Gehrke. "Intelligible Models for Classification and Regression". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. Beijing, China: Association for Computing Machinery, 2012, pp. 150–158. ISBN: 9781450314626. DOI: 10.1145/2339530.2339556.

[102] Yin Lou et al. "Accurate intelligible models with pairwise interactions". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 623–631. DOI: 10.1145/2487575.2487579.

[103] Jacobus Lubsen, J Pool, and E Van der Does. "A practical device for the application of a diagnostic or prognostic function". In: *Methods of information in medicine* 17.02 (1978), pp. 127–129.

[104] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[105] Scott M Lundberg et al. "From local explanations to global understanding with explainable AI for trees". In: *Nature machine intelligence* 2.1 (2020), pp. 56–67. DOI: 10.1038/s42256-019-0138-9.

[106] Yuxin Ma et al. "EasySVM: A visual analysis approach for open-box support vector machines". In: *Computational Visual Media* 3.2 (2017), pp. 161–175.

[107] David Martens, BB Baesens, and Tony Van Gestel. "Decompositional rule extraction from support vector machines by active learning". In: *IEEE Transactions on Knowledge and Data Engineering* 21.2 (2009), pp. 178–191. DOI: 10.1109/TKDE.2008.131.

[108] Morteza Mashayekhi and Robin Gras. "Rule Extraction from Random Forest: the RF+HC Methods". In: *Advances in Artificial Intelligence*. Ed. by Denilson Barbosa and Evangelos Milios. Cham: Springer International Publishing, 2015, pp. 223–237. ISBN: 978-3-319-18356-5. DOI: 10.1007/978-3-319-18356-5_20.

[109] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[110] Ryszard S. Michalski et al. "The Multi-Purpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains". In: AAAI'86 (1986), pp. 1041–1045.

[111] George A Miller. "The magical number seven, plus or minus two: Some limits on our capacity for processing information." In: *Psychological review* 63.2 (1956), p. 81. DOI: 10.1037/h0043158.

[112] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial intelligence* 267 (2019), pp. 1–38. DOI: 10.1016/j.artint.2018.07.007.

[113] Graziano Mita et al. "LIBRE: Learning Interpretable Boolean Rule Ensembles". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 245–255. URL: https://proceedings.mlr.press/v108/mita20a.html.

[114] Christoph Molnar. "Interpretable Machine Learning. A Guide for Making Black Box Models Explainable". In: (2019). https://christophm.github.io/interpretable-ml-book/.

[115] Giulio Morina et al. "Auditing and Achieving Intersectional Fairness in Classification Problems". In: *arXiv preprint* (2019). arXiv: 1911.01468 [cs.LG].

[116] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. "Explaining machine learning classifiers through diverse counterfactual explanations". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 607–617. DOI: 10.1145/3351095.3372850.

[117] Martin Možina et al. "Nomograms for Visualization of Naive Bayesian Classifier". In: *Knowledge Discovery in Databases: PKDD 2004*. Ed. by Jean-François Boulicaut et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 337–348. ISBN: 978-3-540-30116-5. DOI: 10.1007/978-3-540-30116-5_32.

[118] John Ashworth Nelder and Robert WM Wedderburn. "Generalized linear models". In: *Journal of the Royal Statistical Society: Series A (General)* 135.3 (1972), pp. 370–384. DOI: doi.org/10.2307/2344614.

[119] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks". In: *arXiv preprint* (2016). arXiv: 1602.03616 [cs.NE].

[120] Harsha Nori et al. "Interpretml: A unified framework for machine learning interpretability". In: *arXiv preprint* (2019). arXiv: 1909.09223 [cs.LG].

[121] Haydemar Núñez, Cecilio Angulo, and Andreu Català. "Rule extraction from support vector machines." In: *Esann*. 2002, pp. 107–112.

[122] Fred C Pampel. *Logistic regression: A primer*. Vol. 132. Sage publications, 2020.

[123] Eliana Pastor. "Deriving Local Internal Logic for Black Box Models". In: *Proceedings of the 26th Italian Symposium on Advanced Database Systems, Castellaneta Marina (Taranto), Italy, June 24-27, 2018*. Ed. by Sonia Bergamaschi, Tommaso Di Noia, and Andrea Maurino. Vol. 2161. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2161/paper47.pdf.

[124] Eliana Pastor, Luca de Alfaro, and Elena Baralis. *DivExplorer project page*. 2021. URL: https://divexplorer.github.io/.

[125] Eliana Pastor, Luca de Alfaro, and Elena Baralis. "Identifying Biased Subgroups in Ranking and Classification". In: *arXiv preprint* (2021). arXiv: 2108.07450 [cs.LG].

[126] Eliana Pastor, Luca de Alfaro, and Elena Baralis. "Looking for Trouble: Analyzing Classifier Behavior via Pattern Divergence". In: *Proceedings of the 2021 International Conference on Management of Data*. SIGMOD/PODS '21. Virtual Event, China: Association for Computing Machinery, 2021, pp. 1400–1412. ISBN: 9781450383431. DOI: 10.1145/3448016.3457284.

[127] Eliana Pastor and Elena Baralis. "Bring Your Own Data to X-PLAIN". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 2805–2808. ISBN: 9781450367356. DOI: 10.1145/3318464.3384710.

[128] Eliana Pastor and Elena Baralis. "Enhancing Interpretability of Black Box Models by means of Local Rules". In: *6th ACM Celebration of Women in Computing: womENcourage 2019*. 2019.

[129] Eliana Pastor and Elena Baralis. "Explaining Black Box Models by Means of Local Rules". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. SAC '19. Limassol, Cyprus: Association for Computing Machinery, 2019, pp. 510–517. ISBN: 9781450359337. DOI: 10.1145/3297280.3297328.

[130] Eliana Pastor et al. "How Divergent Is Your Data?" In: *Proc. VLDB Endow.* VLDB '21 14.12 (2021), pp. 2835–2838. DOI: 10.14778/3476311.3476357.

[131] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[132] Vitali Petsiuk, Abir Das, and Kate Saenko. "RISE: Randomized input sampling for explanation of black-box models". In: *arXiv preprint arXiv:1806.07421* (2018).

[133] Massimo Di Pierro. *Py4Web*. 2021. URL: https://py4web.com.

[134] Alun Preece et al. "Stakeholders in explainable AI". In: *arXiv preprint* (2018). arXiv: 1810.00184 [cs.AI].

[135] Pearl Pu and Li Chen. "Trust Building with Explanation Interfaces". In: *Proceedings of the 11th International Conference on Intelligent User Interfaces*. IUI '06. Sydney, Australia: Association for Computing Machinery, 2006, pp. 93–100. ISBN: 1595932879. DOI: 10.1145/1111449.1111475.

[136] J Ross Quinlan and R Mike Cameron-Jones. "FOIL: A midterm report". In: *European conference on machine learning*. Springer. 1993, pp. 1–20. DOI: 10.1007/3-540-56602-3_124.

[137] *React - A JavaScript library for building user interfaces*. http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm.

[138] Marco Túlio 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*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 1135–1144. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778.

[139] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-precision model-agnostic explanations". In: *AAAI Conference on Artificial Intelligence*. 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982.

[140] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Model-agnostic interpretability of machine learning". In: *arXiv preprint* (2016). arXiv: 1606.05386 [stat.ML].

[141] M. Robnik-Šikonja and I. Kononenko. "Explaining Classifications For Individual Instances". In: *IEEE Transactions on Knowledge and Data Engineering* 20.5 (May 2008), pp. 589–600. DOI: 10.1109/TKDE.2007.190734.

[142] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x.

[143] Cynthia Rudin and Şeyda Ertekin. "Learning customized and optimized lists of rules with mathematical programming". In: *Mathematical Programming Computation* 10.4 (2018), pp. 659–702. DOI: 10.1007/s12532-018-0143-8.

[144] Cynthia Rudin et al. "Interpretable machine learning: Fundamental principles and 10 grand challenges". In: *arXiv preprint* (2021). arXiv: 2103.11251 [cs.LG].

[145] Svetlana Sagadeeva and Matthias Boehm. "SliceLine: Fast, Linear-Algebra-Based Slice Finding for ML Model Debugging". In: *Proceedings of the 2021 International Conference on Management of Data*. SIGMOD/PODS '21. Virtual Event, China: Association for Computing Machinery, 2021, pp. 2290–2299. ISBN: 9781450383431. DOI: 10.1145/3448016.3457323.

[146] Pedro Saleiro et al. "Aequitas: A Bias and Fairness Audit Toolkit". In: *arXiv preprint* (2019). arXiv: 1811.05577 [cs.LG].

[147] Patrick Schwab and Walter Karlen. "CXPlain: Causal Explanations for Model Interpretation under Uncertainty". In: 32 (2019). Ed. by H. Wallach et al. URL: https://proceedings.neurips.cc/paper/2019/file/3ab6be46e1d6b21d59a3c3a0b9d0f6ef-Paper.pdf.

[148] Andrew Selbst and Julia Powles. ""Meaningful Information" and the Right to Explanation". In: *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. Ed. by Sorelle A. Friedler and Christo Wilson. Vol. 81. Proceedings of Machine Learning Research. PMLR, 2018, pp. 48–48. URL: https://proceedings.mlr.press/v81/selbst18a.html.

[149] Rudy Setiono and Huan Liu. "Understanding neural networks via rule extraction". In: *IJCAI*. Vol. 1. 1995, pp. 480–485.

[150] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[151] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint* (2014). arXiv: 1312.6034 [cs.CV].

[152] Ashudeep Singh and Thorsten Joachims. "Fairness of exposure in rankings". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2219–2228. DOI: 10.1145/3219819.3220088.

[153] E. Štrumbelj and I. Kononenko. "An Efficient Explanation of Individual Classifications Using Game Theory". In: *J. Mach. Learn. Res.* 11 (Mar. 2010), pp. 1–18. ISSN: 1532-4435. URL: http://jmlr.org/papers/v11/strumbelj10a.html.

[154] E. Štrumbelj and I. Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and Information Systems* 41.3 (Dec. 2014), pp. 647–665. ISSN: 0219-3116. DOI: 10.1007/s10115-013-0679-x.

[155] E. Štrumbelj, I. Kononenko, and M. Robnik Šikonja. "Explaining Instance Classifications with Interactions of Subsets of Feature Values". In: *Data Knowl. Eng.* 68.10 (Oct. 2009), pp. 886–904. ISSN: 0169-023X. DOI: 10.1016/j.datak.2009.01.004.

[156] Pang-Ning Tan et al. *Introduction to Data Mining (2nd Edition)*. 2nd. Pearson, 2018. ISBN: 0133128903.

[157] Fadi Abdeljaber Thabtah. "A review of associative classification mining". In: *Knowledge Engineering Review* 22.1 (2007), pp. 37–65. DOI: 10.1017/S0269888907001026.

[158] SL Ting et al. "Mining logistics data to assure the quality in a sustainable food supply chain: A case in the red wine industry". In: *International Journal of Production Economics* 152 (2014), pp. 200–209. DOI: 10.1016/j.ijpe.2013.12.010.

[159] Geoffrey G Towell and Jude W Shavlik. "Extracting refined rules from knowledge-based neural networks". In: *Machine learning* 13.1 (1993), pp. 71–101.

[160] F.-Y. Tzeng and K.-L. Ma. "Opening the black box - data driven visualization of neural networks". In: *VIS 05. IEEE Visualization, 2005.* 2005, pp. 383–390. DOI: 10.1109/VISUAL.2005.1532820.

[161] Berk Ustun and Cynthia Rudin. "Learning Optimized Risk Scores". In: *Journal of Machine Learning Research* 20.150 (2019), pp. 1–75. URL: http://jmlr.org/papers/v20/18-615.html.

[162] Francesco Ventura and Tania Cerquitelli. "What's in the box? Explaining the black-box model through an evaluation of its interpretable features". In: *arXiv preprint* (2019). arXiv: 1908.04348 [cs.CV].

[163] Francesco Ventura et al. "Explaining the Deep Natural Language Processing by Mining Textual Interpretable Features". In: *arXiv preprint* (2021). arXiv: 2106.06697 [cs.CL].

[164] Sahil Verma, John Dickerson, and Keegan Hines. "Counterfactual Explanations for Machine Learning: A Review". In: *arXiv preprint* (2020). arXiv: 2010.10596 [cs.LG].

[165] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. "Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation". In: *International Data Privacy Law* 7.2 (June 2017), pp. 76–99. ISSN: 2044-3994. DOI: 10.1093/idpl/ipx005.

[166] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR". In: *Harv. JL & Tech.* 31 (2017), p. 841.

[167] Claus Weihs and UM Sondhauss. "Combining mental fit and data fit for classification rule selection". In: *Exploratory Data Analysis in Empirical Research*. Springer, 2003, pp. 188–203. DOI: 10.1007/978-3-642-55721-7_21.

[168] Linda F. Wightman. In: *LSAC research report series*. 1998. URL: https://eric.ed.gov/?id=ED469370.

[169] Tongshuang Wu et al. "Errudite: Scalable, Reproducible, and Testable Error Analysis". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 747–763. DOI: 10.18653/v1/P19-1073.

[170] Ke Yang and Julia Stoyanovich. "Measuring Fairness in Ranked Outputs". In: *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. SSDBM '17. Chicago, IL, USA: Association for Computing Machinery, 2017. ISBN: 9781450352826. DOI: 10.1145/3085504.3085526.

[171] Xiaoxin Yin and Jiawei Han. "CPAR: Classification based on predictive association rules". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM. 2003, pp. 331–335. DOI: 10.1137/1.9781611972733.40.

[172] Mohammed J Zaki et al. "Parallel algorithms for discovery of association rules". In: *Data mining and knowledge discovery* 1.4 (1997), pp. 343–373. DOI: 10.1023/A:1009773317876.

[173] Meike Zehlike and Carlos Castillo. "Reducing Disparate Exposure in Ranking: A Learning To Rank Approach". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 2849–2855. ISBN: 9781450370233. DOI: 10.1145/3366424.3380048.

[174] Meike Zehlike, Ke Yang, and Julia Stoyanovich. "Fairness in Ranking: A Survey". In: *arXiv preprint* (2021). arXiv: 2103.14000 [cs.IR].

[175] Meike Zehlike et al. "FA*IR: A Fair Top-k Ranking Algorithm". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1569–1578. ISBN: 9781450349185. DOI: 10.1145/3132847.3132938.

[176] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_53.

[177] Junzhe Zhang and Elias Bareinboim. "Fairness in decision-making—the causal explanation formula". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16949.

[178] Xiuzhen Zhang, Guozhu Dong, and Kotagiri Ramamohanarao. "Information-Based Classification by Aggregating Emerging Patterns". In: *Proceedings of the Second International Conference on Intelligent Data Engineering and Automated Learning, Data Mining, Financial Engineering, and Intelligent Agents*. IDEAL '00. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 48–53. ISBN: 3540414509. DOI: 10.1007/3-540-44491-2_8.

[179] Luisa M Zintgraf et al. "Visualizing deep neural network decisions: Prediction difference analysis". In: *arXiv preprint* (2017). arXiv: 1702.04595 [cs.CV].

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.