Predicate Induction and Visual Analytics: Bridging Human and Automated Reasoning

A dissertation

submitted by

Brian Montambault

In partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Computer Science

TUFTS UNIVERSITY

February 2026

ADVISOR: Remco Chang

Abstract

Predicate Induction and Visual Analytics: Bridging Human and Automated Reasoning

Brian Montambault

ADVISOR: Remco Chang

While data-driven decision making has garnered increased interest across a wide range of domains, real-world analysis tasks continue to pose significant challenges. These tasks are often characterized by loosely defined goals and shaped by information that is incomplete, ambiguous, or dependent on domain-specific context. Visual Analytics (VA) seeks to address these challenges by integrating automated data analysis and human reasoning through interactive visual interfaces. However, this approach can fall short when users struggle to align analysis results with their own understanding of the data and task. In this thesis, I introduce predicate induction for visual analytics as a technique to help bridge the gap between human and automated reasoning. Predicate induction produces concise, interpretable statements describing relevant subsets of the data. By representing both concrete subsets and abstract logical concepts, predicates serve as a shared medium for automated and human reasoning. I demonstrate the flexibility and effectiveness of this approach through the design of three systems for three distinct analysis use-cases: PIXAL for anomaly reasoning, DimBridge for explainable dimensionality reduction, and VIGOR for graph visualization recommendation. For each system, I demonstrate how the predicate induction engine and visual interface are designed to address the nuances of the particular analysis use-case. Then, I discuss our initial steps to-

wards generalizing this approach beyond our three use-cases, presenting a formal framework for predicate induction in visual analytics.

For Jamie

Acknowledgments

This work could not have been completed without the support of my family, friends, and labmates.

To my wife, Jamie, thank you for being there for me through every late night and setback. I would not have made it to the end without your constant support and encouragement.

To my parents, David, and Debbi, thank you for instilling in me the value of pursuing my interests. To my brother Kevin, sister Kayla, and brother in-law Ben, I could not ask for kinder and more supportive siblings.

To my labmates Ashley, Camelia, Gabe, Jen, Mingwei, Sjeord, Alexandra, Sasha, Manling, Brian, and Susie, thank you all for the invaluable contributions you have made to my development as a researcher.

To Remco, thank you for your guidance throughout this process and for helping me to make my research my own. I could not imagine having done this with any other advisor.

To my committee Rob Jacob, Elaine Short, Jonathan Lamontagne, and Jeff Gilbert, thank you for the valuable feedback and guidance throughout this process.

Brian Montambault

TUFTS UNIVERSITY

February 2026

Contents

List of Tables

List of Figures

Summary of Papers

Portions of this dissertation are based on the following papers:

- Brian Montambault, Camelia D. Brumar, Michael Behrisch, and Remco Chang. Pixal: Anomaly reasoning with visual analytics, 2022
- Brian Montambault, Gabriel Appleby, Jen Rogers, Camelia D Brumar, Mingwei Li, and Remco Chang. Dimbridge: Interactive explanation of visual patterns in dimensionality reductions with predicate logic. *IEEE Transactions on Visualization and Computer Graphics*, 2024
- Sjoerd Vink, Brian Montambault, Mingwei Li, Remco Chang, and Michael Behrisch. Graph visualization guidelines as learnable predicates. *International Conference on Computer Graphics, Interaction, Visualization Theory and Applications*, 2026 (accepted)

Chapter 1

Introduction

Recent advances in data management and analysis technologies have made data-driven decision making increasingly accessible across a wide range of domains. As the value of extracting insights from data continues to become more widely recognized, a growing proportion of analysis tasks are characterized by loosely defined hypotheses, domain-specific objectives, and incomplete information. For example, anomaly detection, the problem of identifying data points that deviate significantly from expected patterns, has become widely accessible through both open source libraries [PVG+11] and commercial tools [tab]. However, leveraging anomaly detection results for data-driven decision making remains a challenge [CBK09]. What constitutes an "anomaly" is highly dependent on context, making it difficult to apply a uniform approach across diverse use-cases. Furthermore, the process of extracting insights that are applicable to decision making is similarly context dependent, requiring an additional layer of reasoning beyond simply identifying unusual data points. At the same time, the size and complexity of datasets available for analysis continue to grow rapidly, fueled by advances in infrastructure for data collection and storage [KKKG14]. These trends underscore the need for new analysis tools capable of supporting users in navigating complex, ill-defined problems, while processing and extracting insights from vast quantities of data at scale.

Visual Analytics (VA) addresses this need by combining the computational power of automated analysis techniques with the flexibility of human reasoning

through the use of interactive visual interfaces [TC06, KMS+08]. While purely automated methods require strict assumptions and a rigid interpretation of results, data visualizations are flexible representations that, when combined with users' background knowledge, can provide insights into multiple facets of the data. The power of data visualization is exemplified by Anscombe's Quartet, which describes four datasets characterized by dramatically different relationships that share identical summary statistics [Ans73]. While these distinctions are immediately recognizable when the data is visualized, they are obscured by the fixed assumptions imposed by the choice of summary statistics. Interaction provides further benefits beyond static data visualizations, allowing users to filter data, adjust the parameters of underlying algorithms, and manipulate visual encodings to focus on specific aspects of the broader analysis task. This process facilitates a sense-making loop that allows users to iteratively explore, refine, and test hypotheses [PC05]. Over the course of an analysis, users are able to sharpen their understanding of both the data and the analysis problem, while incorporating domain-specific knowledge that may not be apparent from the raw data alone. While VA offers advantages over purely automated approaches, users may still struggle to interpret analysis results. Outputs from complex algorithms can be difficult to relate to domain-specific concepts, making it unclear how the results support decision-making. Moreover, the process by which raw data is transformed into results is frequently opaque, leaving users uncertain about why a particular output was produced, and which facets of the data were the most influential. This is especially true for tools geared towards domain experts, who bring rich domain knowledge to the analysis but may lack expertise in data science and statistics. This disconnect can break the sense-making loop, leaving users frustrated and unable to extract further insights.

If VA systems are to be widely adopted as a tool for large-scale data analysis, their underlying analysis processes must produce results that are interpretable and transparent to users from diverse backgrounds and varying levels of technical expertise. In this thesis, I introduce predicate induction as one such automated analysis technique that can be broadly applied in VA systems. Predicate induction

is an automated analysis technique that produces predicates - concise logical statements that describe subsets of the data relevant given a particular use-case. For example, consider a tabular sales dataset where each row represents a transaction by an online retailer, and each column corresponds to attributes such as customer demographics, product category, and financial metrics. Suppose the retailer seeks to identify customer segments where total sales are lower than forecasted. Using predicate induction, a sales analyst can identify subsets of the data that fit this criteria - e.g., "customerAge $>$ 65 AND Region = Midwest", describing only the sales records where the customer was over 65 years old and located in the Midwest region. While other automated analysis techniques may produce results that are difficult for users to conceptualize in the relevant domain context, predicates are directly interpretable. A predicate is composed of a set of conditions that each specify a data attribute and a set of values that can be interpreted independently of the data (e.g., customers in a particular age bracket of from a particular region). Predicates are understood not just as results derived from the data, but as a meaningful concept within the context of their use-case. Furthermore, data points that are relevant to the result and those that are not are defined explicitly, leaving no ambiguity as to how that result was derived from the data. By representing both an abstract explanation and a concrete subset of the data, predicates address the two major limitations of results generated by other automated analysis techniques: understanding the meaning of results in terms of their use-case, and understanding the reasoning process leading from the raw data to the results.

In addition to facilitating interpretation of results generated automatically, predicates are inherently composable and editable, providing a strong foundation for user interaction. Because predicates are built from individual conditions, users can modify them easily to explore related hypotheses or apply their domain knowledge. For instance, an analyst might suspect that a similar pattern exists for this age group for the Southwest region, and can immediately test their hypothesis by modifying the condition "Region = Midwest" to Region = Southwest". Similarly an analyst might wish to further refine or expand a predicate by adding or remov-

ing conditions. For example, an analyst might understand from their background knowledge that older customers are more likely to purchase a certain type of product (e.g., Home Goods), and choose to add an additional clause "productCategory = Home Goods". In this way, predicates function not only as outputs of analysis, but as manipulable artifacts that drive further inquiry. In addition to modifying automatically generated predicates, users can simply generate their own predicates from scratch that can be directly compared to those produced by the algorithm. By producing results that are easily interpreted and manipulated by users, predicate induction provides a flexible and transparent bridge between automated insight and human reasoning. By utilizing predicate induction, VA systems can avoid the frustration often experienced by users in reconciling results produced by automated analysis techniques with assumptions derived by their own reasoning process.

In this thesis, I provide evidence for the effectiveness of predicate induction in VA through three diverse analysis use-cases: anomaly reasoning (AR), explainable dimensionality reduction (XDR), and graph visualization recommendation (GVR). For each use-case, I present a system designed to bridge the gap between automated insight and human reasoning through the use of 1) a predicate induction engine and 2) an interactive visual interface.

First, I present PIXAL, a VA system designed for AR. While anomaly detection algorithms are widespread and accessible, AR, the process of understanding and communicating the broader underlying phenomenon, is often necessary for extracting meaningful insights. PIXAL uses predicate induction to identify collections of related anomalies, using a Bayesian t-test to compare anomaly scores within a collection to the rest of the data. The resulting Bayes factor quantifies the strength of evidence supporting the hypothesis that a data points in a particular collection have significantly higher anomaly scores. Importantly, PIXAL's induction engine is capable of generating multiple plausible explanations for a given anomalous collection. PIXAL's visual interface allows users to directly compare and manipulate these explanations, facilitating the synthesis of automatically generated results with their domain expertise while verifying and building trust in the Bayes factor as a

reliable metric. Furthermore, PIXAL's interface allows users to visualize anomalous collections in the context of similar, but normal, collections, providing the necessary context for interpreting anomalies in terms of domain-relevant attributes. Findings from these comparisons are output as straightforward text descriptions; results that can be communicated to stakeholders without needing to provide complex technical background.

Second, I present DimBridge, a VA system designed for XDR. While dimensionality reduction (DR) allows users to identify the existence of patterns in high-dimensional data, understanding their meaning in terms of the original dimensions requires further reasoning. DimBridge uses predicate induction to find subsets of data in the high-dimensional space that match a particular pattern in the DR results, identified by the user through the visual interface. By defining predicates in terms of the original dimensions, DimBridge's induction engine defines the smallest subspace needed to define a given pattern. While visualizing all dimensions would be a major challenge, this smaller subspace is easy to visualize in DimBridge's interface, allowing users to reason about patterns in terms of relationships found existing in the original dimensions. To facilitate a smooth sense-making loop, DimBridge's induction engine prioritizes speed over accuracy. DimBridge's visual interface compensates for potential inaccuracies by displaying the pattern defined by the automatically generated predicate alongside the original pattern selected by the user. By making these inaccuracies explicit, DimBridge's visual interface allows users to identify and correct for misleading or spurious results through further exploration and interaction.

Finally, I present VIGOR, a system designed for graph visualization recommendation (GVR). Graph datasets represent not only data points, but relationships between data points, providing rich structure for analysis not found in tabular datasets. However, this added complexity makes graph visualization a major challenge, especially for users with limited experience using available visualization techniques. VIGOR uses predicate induction to generate interpretable guidelines from labeled examples, describing when particular visualization techniques are appro-

priate based on statistics generated from the graph (e.g., number of nodes/edges, density). In addition to serving as guidelines generated automatically by the induction engine, predicates defined over graph statistics can be generated directly from the literature, providing a framework that combines expert knowledge with data-driven insights. Integrated into a visual interface supporting multiple graph visualization techniques, VIGOR recommends suitable visualizations while helping users develop a generalizable understanding of relationships between graph properties and the effectiveness of each visualization technique.

Following the use-cases, I propose a formal framework for applying predicate induction to VA. I begin by presenting a generalized understanding of predicate induction gained through experience with diverse use-cases, specifying what distinguishes predicate induction from other automated analysis techniques, when predicate induction is useful (and when it is not), and detailing the tradeoff between accuracy, interpretability, and speed. Then, I discuss how an interactive visual interface can be used to manage this tradeoff, while providing additional context for understanding subsets of data and their corresponding explanations. Finally, I formally define these components of the induction engine and visual interface in terms within a discrete design space, providing a tool that researchers can use to apply this technique to future work across a wide range of possible use-cases.

In the next chapter, I begin by reviewing related work on visual analytics and predicate induction. In Chapter 3, I introduce PIXAL, a VA system for AR, and demonstrate its effectiveness through interviews with industry data scientists and analysts in healthcare and roadside assistance domains. In Chapter 4, I introduce DimBridge, a VA system for XDR, and demonstrate its effectiveness through a materials science case study with domain scientists. In Chapter 5, I introduce VIGOR, a system for GVR, and demonstrate its effectiveness in generating guidelines from an expert-labeled graph dataset. In Chapter 6 I propose a formal framework for applying predicate induction in visual analytics, defining a path forward for future applications across diverse use-cases. In Chapter 7 I reflect on the advantages of this approach and implications for broader questions in the VA community. Finally,

in Chapter 8 I explain how the systems and formal framework presented in this thesis contribute towards the overall goal of bridging the gap between human and automated reasoning in visual analytics.

Chapter 2

Related Work

This chapter reviews prior work related to the application of human and automated reasoning in complex data analysis use-cases, characterized by loosely defined objectives, hypotheses, and relevant context. I organize this work into three sections. First, I discuss foundational work in visual analytics that defines the complementary roles of human and automated reasoning within an analytic dialogue. Second, I discuss interaction, visualization, and explainable artificial intelligence (XAI) as mechanisms designed to support communication between the human analyst and the automated system within this dialogue. Finally, I discuss systems that use predicate logic as a shared, flexible representation of both human and automated reasoning, combined with visualization to address a wide range of analysis use-cases.

2.1 Visual Analytics

Thomas and Cook (2005) define the process of visual analytics as an active discourse between the analyst and the system [CT05]. Within this discourse, the system uses complex computations to extract relevant patterns from the data, while the analyst uses their background knowledge, experience, and flexible reasoning abilities to relate patterns to real-world challenges. Kiem et al. (2006) identify active discourse as the distinguishing factor between visual analytics and traditional information seeking [KMSZ06]. This iterative process allows the analyst to continuously refine

their understanding of the problem, and the system to improve how it presents the data to best support the analyst. Later work incorporated these theories into formal models [KMS+08, SSS+14]. Kiem et al. (2008) propose a model that defines visual analytics as the process of transforming data into insights [KMS+08]. In this model, the system applies transformations to the data that yield hypotheses and visualizations, while the analyst interacts with the system to modify or generate new hypotheses and visualizations. Over the course of the analysis, hypotheses and visualizations are continuously refined, supporting the analyst in discovering increasingly relevant insights. Sacha et al. (2014) propose a model of knowledge generation in visual analytics that elaborates on this view [SSS+14]. In their model, the automated system applies transformations and generates visualizations from the data, while the analyst interacts with the system in three loops. In the exploration loop, analysts interpret patterns to generate hypotheses and guide further analysis. In the verification loop, analysts test and refine hypotheses using evidence observed in the visualization. In the knowledge generation loop, analysts synthesize newly acquired insights with their prior knowledge.

Visual analytics has been applied across a wide range of critical domains, including healthcare [CG15], finance [CGK+], and cybersecurity [GS09]. The integration of flexible human reasoning makes this approach particularly effective for ill-defined or ambiguous use-cases where domain-specific context plays a central role and cannot be derived from data alone. Two notable examples include anomaly detection and high-dimensional data analysis.

Visual Analytics for Anomaly Detection: The visual analytics community has a rich history of research and development of systems for anomaly detection, dating back to the original visual analytics research agendas by Thomas and Cook [TC06] and Keim et al. [KMS+08]. Such systems can be found in nearly any domain where anomalies represent critical failures, including fraud detection [LGM+17, LGM+18, CGK+], network security monitoring [MKN+07, GS09, XGH06], and machine learning [WPB+19, CL18, HKPC18]. For example, the Situ system [GRS+18] utilizes unsupervised anomaly detection algorithms to detect potential suspicious data and

provide analysts with an interactive visual interface to explore and make sense of the found anomalies. Similarly, visual analytics systems in the domains of space-time and trajectories analysis (e.g., [AAG03, AMST11, LYC10, MRH+09]), network security (e.g., [XWY+20, ZCS19, XXM19] and surveys by Shiravi et al. [SSG11] and Zhang et al. [ZWL+17]), and systems such as Voila [CLZ+17] used in urban development scenarios and TargetVue [CSL+16] for social media analysis help analysts make sense of anomalous and suspicious behaviors in complex, dynamic, and heterogeneous data. Other approaches focus on relationships between anomalies, such as the higher-order correlation graphs proposed by Yan et al. [YST+18, TSZ+18].

Visual Analytics for High-Dimensional data analysis: High-dimensional data poses a significant challenge for analysts, concealing important patterns across a large number of data attributes. A number of visual analytic solutions have been proposed that leverage novel visualization and interaction techniques. One of the most common families of methods is multiline graphs [And72, Ins85], which plot several to many features either overlaid or stacked vertically over another dimension. This visual technique does not reduce the dimensionality of the data in terms of losing information or combining dimensions. An example of this is the parallel coordinates plot [Ins85, HW13], which puts each dimension on a separate axis and draws a line connecting these axes for each instance. Parallel coordinates are often used in concert with sophisticated interaction techniques for selecting groups of data items, e.g., angular brushing of dimensions [HLD02], multi-way brushing for high dimensional pattern discovery [RLS+19], as well as augmented designs that better convey relationships between dimensions [BKS20]. The multiline approach allows for the visualization of all dimensions but can become cluttered and hard to interpret as the number of dimensions increases. Visual analytic systems designed for high-dimensional data often utilize dimensionality reduction (DR) algorithms such as PCA [Pea01], t-SNE [vdMH08], and UMAP [MHSG18]. These algorithms aim to project the high-dimensional space onto two dimensions that can then be visualized using traditional techniques (e.g., scatterplots) while preserving the original structure.

In these use-cases, human reasoning can provide key context and background knowledge that cannot be derived from automated analysis methods alone. However, visual analytics can fall short if users are unable to interpret the reasoning behind results presented through the visual interface, or cannot sufficiently express their own reasoning through interaction. In the following section, I discuss methods for communicating human reasoning through interaction, and automated reasoning through visualization and XAI.

2.2 Interaction, Visualization, and Explainable AI

Within the visual analytics process, analysts communicate abstract knowledge, assumptions, and hypotheses by interacting with the system, while the system communicates concrete patterns in the data through visualization. In this section, I discuss theories and methods that help clarify the role and value of interaction and visualization in communicating between between analyst and system. Additionally, I discuss methods for explaining the complex, automated reasoning of machine learning models from the field of XAI.

2.2.1 Interaction

Interaction is the primary means by which analysts communicate their reasoning to automated systems [YKSJ07]. Prior work has investigated the relationship between analysts' high-level reasoning and their interactions with the system using taxonomies and typologies, machine learning models, and formal languages. For example, Yi et al. (2007) propose a taxonomy that aligns interactions with user intent, including categories for select, explore, reconfigure, encode, abstract/elaborate, and filter interactions [YKSJ07]. Brehmer and Munzner (2013) propose a multi-level typology that bridges high-level analytic tasks (e.g., discover, present, enjoy), mid-level actions (produce, search, query), and low-level interactions (encode, manipulate, introduce) [BM13].

Rather than classifying interactions into fixed categories, other work has used

machine learning to infer latent properties of the analyst's reasoning from potentially noisy interaction signals. This approach is particularly well suited for capturing forms of knowledge that are difficult to articulate directly, such as the appropriate distance function for a two-dimensional projection [BLBC12], the relevance of words or semantic concepts in text collections [BNH14], or which visual features are most important for a given analysis task [BKSS14]. In addition to expert knowledge, this approach has been applied to learning other relevant properties of the analyst's reasoning process such as bias [GSC16] and exploration strategy [MGO21].

Other approaches model interaction using more expressive formal languages. Kandel et al. (2011) introduce Wrangler, a system that enables users to perform data transformations that are difficult to specify directly by automatically inferring the underlying operations from examples [KPHH11]. Similarly, Gulwani et al. (2012) propose a method for learning regular expressions from user-provided examples in Excel [GHS12]. Instead of inferring the analyst's reasoning from noisy interactions, other systems use formal languages that allow analysts to explicitly specify their intent. In HomeFinder, user interactions are translated into SQL queries that filter homes based on preferences along dimensions such as cost, number of bedrooms, and distance to points of interest [WS92]. Similar to SQL, logical predicates have been used as a flexible, expressive, and transparent representation of the analyst's reasoning. For example, Koch et al. (2009) present an interactive system that enables analysts to construct first-order predicates to identify relevant patents based on combinations of attributes and relationships [KBGE09]. The three systems presented in the following chapters, DimBridge, PIXAL, and VIGOR, build on this work, allowing analysts to communicate with an automated system by generating and manipulating logical predicates using a visual interface. In the next subsection, I discuss use of visualization to communicate patterns and insights to the analyst.

2.2.2 Visualization

The system's role in the analytic discourse is to communicate relevant patterns to the analyst. When the analyst's goals are ambiguous, visualization provides a

flexible representation that allows them to consider many hypotheses at once [Tuk80, Ans73]. Subsequent theories and methods have further elaborated the incorporation of concrete patterns and evidence found in the data into the analyst's abstract domain knowledge and expertise through visualization. According to the model proposed by Van Wijk (2005), the value of a visualization can be quantified in terms of the net value of the knowledge gained by the analyst relative to the cost of generating and interpreting the visualization. Ziemkiewicz and Kosara (2008) focus more explicitly on how concrete visual patterns are mapped to abstract concepts through visual metaphors [ZK08]. They argue that visualizations communicate meaning by leveraging metaphorical correspondences between visual structure and conceptual structure, and that successful interpretation depends on the analyst's ability to internalize these metaphors.

Other work has defined concrete patterns and insights that analysts search for in visualizations to support their reasoning. In a recent literature review, Battle and Ottley (2023) identify a number of concrete patterns and observations that have been defined as "insights" [BO23], including correlations [CL+15, ZZZK18], distributions [CL+15, ZZZK18, SND05], and trends or outliers [CL+15]. Other categories include more abstract insights like hypotheses and generalizations [LH14, GGZL15], and groupings or comparisons [CL+15, SND05]. Similar work has investigated how specific insights are communicated through visualization. For example, Gleicher (2011) proposes a taxonomy of methods for visual comparison that includes juxtaposition, superposition, and explicit encoding, and generalizes across a variety of use-cases and specific visual elements [GAW+11]. In the previous section, I discussed the use of logical predicates as a means for the analyst to communicate their own reasoning to the system through interaction. Weaver (2009) explicitly links interactions represented as predicates in conjunctive normal form and a set of visual comparisons defined in conjunctive visual form [Wea]. In the following chapters, I describe three systems that build on this work, using the visual comparison of subsets as a technique for communicating a wide range of patterns and insights relevant across three distinct use-cases.

2.2.3 XAI

While visualization is a powerful tool for handling ambiguity, it is not always necessary when analysis tasks are well defined [KMSZ06]. These types of problems are more suited for a purely automated approach, where the system learns the parameters of a well-specified model without the analyst's intervention. However, recent work in interpretable machine learning and XAI has demonstrated the need to communicate some aspects of the automated reasoning process, particularly in cases where fairness, privacy, reliability, trust are important criteria [DVK17]. In this subsection, I review XAI methods designed to explain complex patterns in the data to human users. In particular, I identify rule-based approaches as a flexible, interpretable representation of complex automated reasoning that is consistent with predicate-based representations of human reasoning.

Doshi-Velez et al. (2017) identify a number of qualities that distinguish explanations across different methods and tasks [DVK17]. For example, the cognitive chunks that form the basic unit of explanations and their level of compositionality vary across different methods, while the nature of user expertise varies across tasks. Murdoch et al. (2019) introduce a framework that evaluates explanations by their ability to make accurate predictions and describe the behavior of the model, while being relevant to the target audience [MSK$^+$19]. Model sparsity, simulatability, and modularity are all qualities that can improve these metrics.

A number of taxonomies have been proposed that categorize XAI techniques by the method of obtaining the explanation and how it is represented [CPC19, LPK20, MCB20, ADRDS$^+$20]. Explanations can be *intrinsic* to a model that is inherently interpretable. For example, non-zero coefficients in a sparse linear model [RC18a], the if-then structure of a decision tree [BFSO84], or explicit rules [YRS17] are all intrinsic properties of the model that explain their behavior. *Post-hoc* explanations, in contrast, are derived after model training and seek to approximate or summarize the behavior "black-box" models [RSG16, LL17, RSG18]. Whether intrinsic or post-hoc, to prominent categories of interpretable model are *feature-based*

and *rule-based*. Feature-based models provide explanations in terms of coefficients or importance weights assigned to each data attribute. For example, LIME uses a local, linear surrogate model that can be explained in terms of its coefficients to help interpret the decision making process of a black-box model on a single data point [RSG16]. SHAP values provide a similar explanation, generating explanations based on feature importances derived using a game-theoretical algorithm [LL17]. While feature-based explanations that are compositional, simulatable, and sparse, it is not always clear how feature weights presented in an explanation correspond to relevant patterns in the data [ADRDS+20].

On the other hand, rule-based explanations describe model behavior in terms of logical predicates. For example, ANCHORS [RSG18] uses predicates to define post-hoc explanations, while Dash et al. propose a rule-based classifier that is inherently interpretable [DGW18]. Furthermore, predicates have been used as part of counterfactual explanations [WMR17, DCL+18] to demonstrate how alternate scenarios would have resulted in a different outcomes. Like feature-based explanations, rule-based explanations are compositional, simulatable, and sparse. However, while feature weights may require additional interpretation, predicate logic is directly compatible with representations of human reasoning [Wea]. In the following chapters, I describe three systems that use rule-based explanations to describe patterns automatically identified by the system across a variety of analysis use-cases.

2.3 Predicate Induction for Visual Analytics

The previous section provides background on the use of interaction, visualization, and XAI as means of communication between human analyst's and automated systems. In particular, predicate logic is utilized throughout this work to represent human reasoning through interaction, and automated reasoning through the visual comparison of subsets and rule-based explanations. A number of visual analytic systems have been designed that use predicates in combination with interaction, visual comparison, and automated explanations in a wide variety of analysis use-cases.

Wu et al. (2013) introduce, Scorpion, a system designed to assist analysts in identifying anomalies in aggregate database queries [WM13]. The user begins by visualizing the results of an aggregate query, comparing groups to identify those with an unusually high or low aggregate value. The user selects suspected anomalies and indicates their direction (high or low), and the system automatically generates a set of predicates describing input tuples that most influence the aggregate anomalies in terms of the remaining data attributes. Predicates generated by Scorpion allow users to work backwards from aggregate outliers observed in the visualization to common properties of their individual data points. Roy and Suciu (2014) propose a similar system for explaining aggregate anomalies that can be applied to SQL queries with joins [RS14]. This system uses a novel algorithm for predicate induction that uses the degree of explanation by intervention based on causality relations defined by foreign keys. Similar to Scorpion, this system describes visual interface where the analyst identifies anomalies in the results of the query. DBSherlock uses a similar workflow where the analyst visualizes and selects aggregate anomalies and the system generates predicate explanations [YNM16]. DBSherlock incorporates an additional interaction, allowing the analyst to provide feedback to a causal model that is used to improve future explanations. Other systems apply a similar approach, but do not include visualization or interaction in their workflow. Roy et al. (2015) propose PerfAugur, a system for explaining anomalies in streaming cloud services logs [RKDK15]. Bailis (2017) propose Macrobase, a system for identifying and presenting relevant aggregates to analysts. These systems appear frequently in the database community (see Dadvar (2022) for a recent survey [DGS22]). However, many use automated predicate induction alone, without the inclusion of interaction and visualization.

Other systems have been developed for investigating model performance. Ming et al. (2019) propose RuleMatrix, a system for explaining the performance of black-box machine learning models [MQB19]. Chung et al. (2019) propose Slice Finder, a system for identifying problematic subsets that are not explained well by a given model [CKP$^+$20]. While similar to the goals of XAI methods described

in the previous sections, this work takes a visual analytic approach and is able to incorporate human reasoning and promote insight through analytic dialogue. In the following chapters, I discuss three systems that extend this approach to three distinct analysis use-cases. Additionally, I propose a formal framework that defines these systems within a common framework of predicate induction, interaction, and visualization.

Chapter 3

PIXAL: Anomaly Reasoning with Visual Analytics

3.1 Problem Statement

Anomaly detection is crucial for promptly addressing unexpected observations in data, especially in domains where such anomalies can have significant impact, such as healthcare, finance, and cybersecurity [AMI16, PP07, KLSH04]. Much of the work that has gone towards supporting anomaly detection is concerned with identifying **point anomalies** – individual data points that significantly deviate from the norm or expected behavior. Point anomalies can be critical for identifying extreme cases that need immediate attention.

However, point anomalies often fail to reveal more complex patterns present in the data, which instead emerge in collections of anomalies. For example, a healthcare analyst might flag individual claims as anomalies when routine hospital visits are billed higher than expected. While a few isolated cases may be unrelated, multiple anomalies from the same group of providers and procedures likely indicate an underlying issue, such as a training gap for new providers, a policy change by a major insurer, or coordinated fraud. We refer to this process of identifying and explaining collections of anomalies as **anomaly reasoning** in this chapter.

While anomaly reasoning provides a number of benefits beyond anomaly detection, it is challenging due to the large number of point anomalies and potential relationships among them. Relationships between anomalies can have multiple interpretations, with a single group of anomalies potentially explained by different underlying phenomena. For example, one analyst might interpret fraudulent healthcare claims involving two specific procedures as two distinct issues. Another analyst might note that both groups of claims originate from the same provider, linking them to a single underlying cause. A number of tools have been developed that help users understand relationships between point anomalies by providing additional context on demand [GRS+18, CLZ+17, CSL+16, YST+18]. However, none of these tools directly address the problem of identifying and explaining collections of anomalies.

In this chapter, we first clarify the challenges and tasks of anomaly reasoning by interviewing five data professionals. Then, we present PIXAL, a visual analytics system designed to address these needs. Specifically, PIXAL represents collections of anomalies as first-order predicates, allowing analysts to work directly with patterns of anomalies rather than a potentially large number of individual point anomalies. PIXAL consists of two components: the RPI algorithm, which automatically generates predicates using a recursive algorithm, and the PIXAL interface, which allows users to manipulate, evaluate, contextualize, and interpret predicates. Together, the PIXAL system provides a framework for analysts to solve the challenging problem of anomaly reasoning, including identifying, evaluating, contextualizing, and communicating collections of anomalies.

We evaluate PIXAL with a usage scenario and follow-up interviews with data professionals. The usage scenario demonstrates that PIXAL facilitates anomaly reasoning by giving analysts the tools to identify and explain relationships in collections of anomalies. In our follow-up interviews, data professionals indicate that PIXAL would be a significant improvement over current anomaly reasoning workflows.

3.2 Anomaly Reasoning

In this section, we further distinguish anomaly reasoning from anomaly detection. Anomaly detection and anomaly reasoning both involve an $n \times m$ dataset, $\mathcal{D}$, and n anomaly scores, $\mathcal{S}$. The goal of anomaly detection is to identify a set of data points (d_i) that are considered unusual compared to the rest of the data. After calculating the anomaly scores, $\mathcal{S}$, data points with scores (s_i) above a threshold (τ) are flagged as potential anomalies. Often, flagged anomalies undergo manual review by analysts who confirm or reject them based on their domain knowledge, K. Let H_i represent the hypothesis that the i-th data point is an anomaly. We define **anomaly detection** (AD) as the process that identifies all data points where H_i is sufficiently likely given the anomaly scores, the original data, and the analyst's background knowledge:

$$AD(\mathcal{D}, \mathcal{S}) = \{d_i \in \mathcal{D} \mid \Pr(H_i | s_i, d_i, K) \geq \tau\}$$

While anomaly detection focuses on identifying individual anomalies, **anomaly reasoning** (AR) involves identifying and explaining collections of anomalies. Let $\mathcal{D}_c$ denote a collection of data points, and let $\mathbb{P}_{>1}(\mathcal{D})$ denote the powerset of $\mathcal{D}$ that includes all collections with at least two data points. We use e to denote an explanation of the relationship between points in a collection, and E_c to denote the set of all possible explanations for a given collection. Let H_e represent the hypothesis that the explanation e describes a relationship among anomalies. We define anomaly reasoning as the process that identifies all collections and explanations where H_e is sufficiently likely:

$$AR(\mathcal{D}, \mathcal{S}) = \{\langle \mathcal{D}_c, e \rangle | \mathcal{D}_c \in \mathbb{P}_{>1}(\mathcal{D}), e \in E_c, \Pr(H_e | \mathcal{S}_c, \mathcal{D}_c, K) \geq \tau\}$$

Similar to anomaly detection, a collection and explanation are considered anomalous if the likelihood of H_e meets or exceeds the threshold τ. However, unlike

H_i, which depends on the anomaly score and attributes of a single data point, H_e depends on the scores and attributes of multiple data points, introducing several technical challenges:

Many possible collections: Anomaly reasoning involves evaluating collections of data points rather than single points. For a dataset of size n, there are $2^n - n - 1$ possible collections of two or more data points (i.e., size of $\mathbb{P}_{>1}(\mathcal{D})$). This exponential increase in the number of collections presents a major challenge, as the space of possible anomalies to assess is far larger than in traditional anomaly detection.

Many possible explanations: Each collection of anomalies ($\mathcal{D}_c$) can potentially have multiple explanations (e) describing the relationships between its data points, with some explanations overlapping or being equally plausible. This creates ambiguity and significantly complicates the evaluation of explanations, requiring careful consideration of multiple possible relationships for each collection.

Likelihood depends on multiple observations: In anomaly detection, the likelihood of a hypothesis H_i depends on the data point d_i and its anomaly score s_i, meaning that evaluating $\Pr(H_i|s_i, d_i, K)$ requires considering at most $m + 1$ values for a dataset with m attributes. For anomaly reasoning, however, the likelihood of a hypothesis H_e depends on a collection of data points $\mathcal{D}_c$ and a collection of anomaly scores $\mathcal{S}_c$. Thus, evaluating $\Pr(H_e|\mathcal{S}_c, \mathcal{D}_c, K)$ involves considering up to $(m + 1) \times |\mathcal{D}_c|$ values for a collection with $|\mathcal{D}_c|$ data points, making hypothesis evaluation significantly more complex than in anomaly detection.

3.3 Interview Study and Task Analysis

We conducted interviews with five data professionals to understand how practitioners handle the challenges of anomaly reasoning. Our participants included four professionals from the automotive insurance sector - two data analysts and two data scientists - and one data scientist from a healthcare technology company. Each participant confirmed that identifying and explaining patterns of anomalies was part of their job responsibilities.

The interviews were conducted individually via video conferencing. During each session, we asked participants to describe their typical anomaly reasoning process and to share relevant real-world examples. Specifically, we explored the steps they followed, the tools they used at each stage, and any obstacles they encountered.

From these discussions, we identified four three key tasks involved in anomaly reasoning: **hypothesize**, **evaluate** and **contextualize**. Additionally, we noted a fourth task, **communicate**, which participants commonly performed but which falls outside our formal definition of anomaly reasoning. For each task, we summarize participants' processes, including the tools they used and the main challenges they faced. We also formally define the inputs and outputs for each task and connect them to our formal definition of anomaly reasoning.

3.3.1 Hypothesize (T1)

All five participants gave similar accounts of how they begin a new anomaly reasoning process. After detecting anomalies using either off-the-shelf or custom algorithms, they described a process involving the generation of a small set of plausible hypotheses, each outlining a potential relationship among anomalies.

Methods: Describing their process for generating hypotheses, analyst's described using a combination of domain expertise, intuition, and collaboration with stakeholders. For instance, an automotive analyst might hypothesize that a sudden decline in response rates for roadside assistance requests is linked to weather disruptions, while a healthcare analyst may suspect changes in payer policies when unusual billing patterns are observed. In both cases, the analyst generates hypotheses to explain the observed anomalies by drawing from past experiences with similar phenomena.

Formalization: We formally describe these analysts' process of forming hypotheses by extending our earlier notion of anomaly detection and reasoning. We use $\Pr(H_e|K)$ to denote the subjective likelihood that H_e is true, given the background knowledge of the user K. We use A_0 to denote the space of all possible collections

and explanations:

$$A_0 = \{\langle \mathcal{D}_c, e\rangle | \mathcal{D}_c \in \mathbb{P}_{\geq 1}(\mathcal{D}), e \in E_c\} \tag{3.1}$$

The analysts' goal is to identify hypotheses that are sufficiently plausible, determined by the threshold τ_1:

$$A_1 = \{\langle \mathcal{D}_c, e\rangle \in A_0 | \Pr(H_e|K) \geq \tau_1\} \tag{3.2}$$

Problem: While all participants found it necessary to focus on a narrow set of plausible hypotheses, some described cases where this led to overlooking relevant hypotheses or adopting ones that later turned out to be false, particularly when operating under time constraints. For example, the healthcare analyst noted that while their knowledge of billing processes was valuable, it sometimes caused them to focus on familiar explanations and miss less obvious but relevant factors, such as diagnosis codes, charge modifiers, and information about procedure duration. In the automotive domain, analysts experienced similar challenges, where their intuition about provider performance could sometimes overshadow alternative explanations.

Requirement: An anomaly reasoning system must assist analysts in identifying plausible hypotheses that explain collections of anomalies. To reduce manual effort and avoid oversights, the system must automatically generate a diverse range of hypotheses (**R1**). Furthermore, it should allow analysts to leverage their domain expertise, considering hypotheses that were not generated by the system or excluding those that they consider implausible (**R2**).

3.3.2 Evaluate (T2)

All five participants described a process for evaluating a collection of data points to determine whether they were indeed anomalous. While the previous task relied heavily on the analysts' domain expertise, this task required them to closely observe the data to assess the *anomalousness* of specific points within a given collection.

Methods: Most participants utilized anomaly detection algorithms to assess the

anomalousness of data points in a given collection. After generating anomaly scores, they used a combination of summary statistics and data visualization to analyze the scores within a collection, often comparing to scores from data points outside of the collection. For instance, the healthcare analyst used an algorithm that assigned high anomaly scores to claims that, based on historical data, should not have been denied but were nonetheless flagged. They then calculated the average anomaly score for each payer and assessed whether any had a significantly higher average score than the rest. Similarly, most automotive analysts applied anomaly detection algorithms, with one analyst employing the isolation forest algorithm to identify unusual observations based on various key performance indicators (KPIs). This analyst also utilized descriptive statistics, calculating average scores over time, across regions, and among clients. They experimented with several data visualizations, such as histograms, to gain deeper insights into the distribution of anomaly scores.

Formalization: We use $\Pr(\mathcal{S}|H_e)$ to denote the likelihood of observing the anomaly scores $\mathcal{S}$, given that H_e is true. Starting with data collections and their associated explanations (A_1), generated from the previous *Hypothesize* task, the analysts' goal in this task is to identify collections and explanations whose high anomaly scores yield sufficient evidence for H_e, determined by the threshold $\tau_{\in}$:

$$A_2 = \{\langle \mathcal{D}_c, e\rangle \in A_1 | \Pr(\mathcal{S}|H_e) \geq \tau_2\}$$

Problem: Several participants described the challenge of evaluating collections based on multiple anomaly scores. For example, the healthcare analyst explained that they were uncomfortable relying solely on the average anomaly score to assess payers. This was particularly true when dealing with smaller collections of claims, where a high average score could be due to normal variation rather than indicating a consistent underlying issue. In the automotive domain, one analyst shared how subjective interpretations of visualized score distributions often led to inconsistent conclusions. These challenges underscore the difficulty of evaluating collections of potential anomalies, especially in cases where the data was limited or the patterns

were ambiguous.

Requirement: An anomaly reasoning system must assist users in determining if a given collection is anomalous **(R3)**. Given the inherent uncertainty in evaluating anomaly collections, these tools should be robust and deliver standardized, interpretable results across different collections and users. These evaluation tools should be integrated into the anomaly reasoning workflow, allowing users to refine their hypothesized explanations based on evaluation feedback **(R4)**.

3.3.3 Contextualize (T3)

All five participants described the need to contextualize anomalies in terms of relevant data attributes. Unlike the *Evaluate* task, where analysts rely on anomaly scores to determine if a collection is anomalous, this task involves using the original data attributes to identify specific, domain-relevant factors that make a collection unusual. While most participants found anomaly detection algorithms helpful in identifying potentially anomalous collections, none felt confident presenting results based solely on these scores. Instead, they were motivated by the need to make sense of anomalies within the context of business-relevant attributes.

Methods: Participants used a combination of methods from previous tasks to contextualize collections of anomalies. Similar to the *Evaluate* task, they applied summary statistics and data visualization, but instead of focusing on anomaly scores, these techniques were applied to the original data attributes. To identify the most relevant attributes, participants relied primarily on their domain expertise, as seen in the *Hypothesize* task. For example, the healthcare analyst leveraged their understanding of healthcare billing to determine that the denial rate was the most relevant statistic for contextualizing anomalies. In the automotive domain, one analyst, after identifying a highly anomalous collection within a specific region and time range, used a line chart to plot the values of a number of key performance indicators (KPIs) for that region over time. By identifying specific KPIs that showed notable spikes or dips during the anomalous period they were able to pinpoint specific operational

issues contributing to the anomalies.

Formalization: We use $\Pr(\mathcal{D}|H_e)$ to denote the likelihood of observing the data $\mathcal{D}$, given that H_e is true. The analysts' goal in this task is to identify collections whose original data attributes yield sufficient evidence for H_e, determined by the threshold τ_3:

$$A_3 = \{\langle \mathcal{D}_c, e\rangle \in A_2 | \Pr(\mathcal{D}|H_e) \geq \tau_3\}$$

Problem: Participants reported encountering significant difficulties completing this task, primarily related to the challenge of identifying which data attributes were most relevant for understanding anomalies. For example, the healthcare analyst explained that while they relied on their domain knowledge to select attributes like denial rates, the complexity of healthcare billing made it difficult to account for all potential factors, such as changes in coding practices or payer policies. This often led to time-consuming trial and error, where different attributes were tested without clear guidance. In the automotive domain, an analyst described the difficulty in isolating which KPIs were truly responsible for the anomaly within a complex operational environment, where multiple factors could contribute to a single event. This challenge was compounded by noisy data, where normal fluctuations could mask or exaggerate certain trends, making it difficult to discern the real cause of anomalies.

Requirement: An anomaly reasoning system must help analysts contextualize explanations of anomalous collections, demonstrating the unusual behavior in terms of domain-relevant attributes **(R5)**. An anomaly reasoning system must provide robust visualization tools and summary statistics that highlight how these attributes deviate from normal patterns in a given context, enabling easy comparison between anomalous collections and typical data. These tools should integrate seamlessly into the anomaly reasoning workflow, allowing insights gained from contextualizing an explanation to support the development of alternative explanations **(R6)**.

3.3.4 Communicate (T4)

All five analysts emphasized the need to effectively communicate their findings to stakeholders and decision-makers. Participants stressed the need to avoid technical jargon and detailed statistical explanations, opting instead for intuitive visualizations and "natural language" descriptions that clearly convey key insights. For example, the healthcare analyst noted that instead of presenting anomaly detection scores, they would explain the findings in terms of increased claim denials, using simple bar charts to illustrate trends. Similarly, an automotive analyst highlighted the importance of framing anomalies in terms of operational impact, such as delays in service response, using line graphs to show deviations from expected performance.

Methods: Participants described using a variety of methods to communicate their findings to decision-makers, with a strong focus on creating clear and actionable reports. This typically involved manually curating and annotating results, often relying on tools like PowerPoint, Excel, or visualization software such as Tableau. Analysts reported spending considerable time creating slides, graphs, and other visual aids, translating complex data into simplified, digestible formats that non-technical stakeholders could easily understand. They also emphasized the importance of adding context to the data by using narrative descriptions that frame the results in a way that aligns with business goals, often customizing each report to the specific needs and concerns of their audience.

Formalization: The *Hypothesize*, *Evaluate*, and *Contextualize* tasks yield a set of collections and explanations, denoted A_3, that are sufficiently plausible, and support the observed anomaly scores ($\mathcal{S}$), and data ($\mathcal{D}$). For each explanation (e), the analysts' goal in the *Communicate* task is to generate a report (e.g., a combination of visualizations and natural-language descriptions), r_e. The result of this task is such a report for each explanation in A_3:

$$A_4 = \{r_e | \langle \mathcal{D}_c, H_e \rangle \in A_3\}$$

Problem: Participants noted the final communication step in particular as time-consuming and labor-intensive, with much of the effort going into reformatting graphs and tweaking reports to make them interpretable for decision-makers. Additionally, the lack of automation means that analysts have to recreate similar reports multiple times, making it inefficient, especially when working under tight deadlines. This process also introduces the risk of miscommunication, as simplifying complex insights for non-technical users can sometimes result in oversimplified or misinterpreted conclusions, ultimately affecting the quality and actionability of the decisions made based on the reports.

Requirement: To address these challenges, an anomaly reasoning system must assist users in communicating their findings in the form of natural language explanations accompanied by intuitive visual aids that can be easily understood by non-technical stakeholders. This system must automate the generation of clear, concise narratives that explain the key insights without relying on technical jargon **(R7)**.

3.4 PIXAL System Overview

We designed PIXAL to address analysts needs in performing anomaly reasoning. Specifically, PIXAL consists of two components. First, a novel algorithm, recursive predicate induction (RPI) takes the $n \times m$ data $\mathcal{D}$ and n anomaly scores $\mathcal{S}$ as inputs and generates a set of conjunctive predicates as output, each representing both a collection of anomalies (**R3**) and an explanation (**R4**). Then, users explore, evaluate, and contextualize the resulting predicates in an interactive visual interface (Figure 3.2).

In the *predicate view*, users can explore as well as modify the collections and explanations generated by the RPI algorithm (**R1, R2**), and evaluate collections by visualizing their distributions of anomaly scores and comparing a number of quantitative measures (**R5**). Among these measures, the *Bayes factor* provides an interpretable representation of the strength of evidence that a collection is particu-

larly anomalous compared to the rest of the data, taking uncertainty into account (**R6**). Predicates can be edited directly from the predicate view, allowing users to adjust hypothesized explanations in response to evaluation results (**R7**). In the *data view*, users can select a predicate and PIXAL will automatically generate a set of small multiples visualizing the anomalies in the context of the data attributes (**R8**, **R9**), along with natural language descriptions (**R9**). In the *focus view*, users select a small multiple and modify the visualization to best communicate the explanation to stakeholders (**R11**). Modifying the visualization also updates the corresponding predicate in the predicate view, allowing users to incorporate insights gained by contextualizing anomalies back into the hypothesis generation process (**R10**).

3.5 RPI Algorithm

PIXAL uses conjunctive predicates to represent both collections and explanations. Taking a hypothesis testing approach, PIXAL uses a Bayesian t-test to compare the anomaly scores of data points in the collection to those of the rest of the data. Taking advantage of the compositional structure of conjunctive predicates, PIXAL utilizes a recursive predicate induction algorithm (RPI) to automatically generate predicates with high Bayes factors, indicating high anomaly scores relative to the rest of the data.

3.5.1 Predicates for Anomaly Reasoning

A predicate is defined by a set of attributes of $\mathcal{D}$ and a set of values for each attribute, and describes a subset of data points. Each attribute and set of values defines a clause, denoted ϕ_v^a, that defines a subset of data points based on the attribute, a, and values, v. For example, the clause $\phi_{[500,1000]}^{sales}$ yields true for all data points in $\mathcal{D}$ whose value for the attribute "sales" is between 500 and 1000. A conjunctive predicate, Φ, defines a subset of $\mathcal{D}$ that includes data points that match the conditions defined by all clauses:

$$\Phi(\mathcal{D}) = \{d \in \mathcal{D} | \phi(d); \forall \phi \in \Phi\}$$

In addition to defining a collection of data points, each predicate provides an interpretable explanation. For example, viewing the clause $\phi^{sales}_{[500,1000]}$ informs us that all data points in the collection will have values of the sales attribute between 500 and 1000, without needing to inspect any points in the collection. This property allows us to define the space of possible collections and explanations, A_0 (see Equation 3.1), as the space of all possible conjunctive predicates given the attributes of $\mathcal{D}$, denoted $\mathbf{\Phi_0}$. We then define the set of plausible predicates, approximating A_1 (see Equation 3.2):

$$A_1 \approx \mathbf{\Phi_1} = \{\Phi \in \mathbf{\Phi_0} | \Pr(H_\Phi | k) \geq \tau_1\}$$

where H_Φ denotes the hypothesis that the data explained by Φ are anomalous, the set of predicates that are evaluated to be anomalous according to their anomaly scores:

$$A_2 \approx \mathbf{\Phi_2} = \{\Phi \in \mathbf{\Phi_1} | \Pr(\mathcal{S} | H_\Phi) \geq \tau_2\}$$

and finally, the set of predicates that are also determined to be anomalous in the context of the data attributes:

$$A_3 \approx \mathbf{\Phi_3} = \{\Phi \in \mathbf{\Phi_2} | \Pr(\mathcal{D} | H_\Phi, k) \geq \tau_3\}$$

While $\Pr(H_\Phi | k)$ and $\Pr(\mathcal{D} | H_\Phi, k)$ depend on the analysts' background domain knowledge k, $\Pr(\mathcal{S} | H_\Phi)$ only depends on the predicate Φ and the anomaly scores $\mathcal{S}$. We frame the problem of evaluating $\Pr(\mathcal{S} | H_\Phi) \geq \tau_2$ as a hypothesis test, comparing the null hypothesis H^0_Φ (i.e., collection described by Φ **are not** anomalies) to the alternative hypothesis H^1_Φ (i.e., collection described by Φ **are** anomalies). Using a Bayesian approach, we calculate the ratio of marginal likelihoods for each hypothesis, known as the Bayes factor (BF):

$$BF = \frac{Pr(\mathcal{S} | H^1_\Phi)}{Pr(\mathcal{S} | H^0_\Phi)}$$

The Bayes factor is interpreted as the degree of evidence for an alternative hypothesis

over the null hypothesis. Like frequentist p-values, Bayes factors take into account the average, size, and variance of a samples defined by $\mathcal{S}_\Phi$ and $\mathcal{S}_{\neg\Phi}$. While p-values only measure the likelihood of the data assuming the null hypothesis is true, the Bayes factor can also be used to measures evidence for the alternative hypothesis. This is particularly relevant for anomaly reasoning, where the alternative hypothesis represents the existence of an anomalous phenomenon. Bayes factors have a standard interpretation relative to the strength of evidence, with Bayes factors 3.2 to 10 being "substantial", 10 to 100 "strong", and greater than 100 "decisive"[KR95].

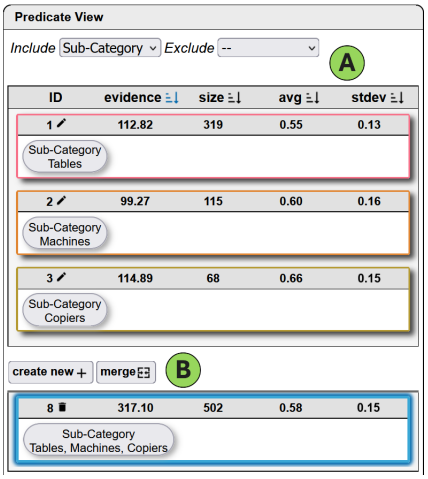
PIXAL defines the marginal likelihoods for each hypothesis according to the Bayesian t-test introduced by Rouder et al.[RSS+09]. The models for both the null and alternative hypotheses are concerned with the effect size $\delta = \frac{\bar{X}-\bar{Y}}{\sigma}$, where X and Y are two samples and σ is their pooled standard deviation. Under H^0_Φ, δ is set to 0, encoding the assumption that their is no difference between the two samples. Under H^1_Φ, δ is allowed to take a value drawn from a Cauchy distribution, allowing it to take values greater than or less than 0, assuming there is a difference between the two samples.

The sample means and variances, μ and σ^2, are given the same treatment for both models, and are given noninformative priors as a result. μ is given the uniform prior $Pr(\mu) = 1$, while σ^2 is given the Jeffreys prior, $Pr(\sigma^2) = 1/\sigma^2$. The Bayes factor resulting from this model is known as the *JZS Bayes factor*, which has been implemented as a library available in R. For the full definition of the JZS Bayes factor as a Bayesian hypothesis test, we refer to Equation 1 in the work by Rouder et al.[RSS+09] (p. 231).

3.5.2 Recursive Predicate Induction

Given a dataset's individual point anomaly scores, PIXAL uses the RPI algorithm to automatically generate plausible hypotheses. Taking advantage of the compositional structure of conjunctive predicates, we can define all those that (a) have one more clause than Φ and (b) have a higher Bayes factor than both Φ and the new clause:

Figure 3.1: A user filters the Predicate View to only include those containing the attribute 'Sub-Category" (A), merges the three predicates (Sub-Category: ['Tables'], Sub-Category: ['Machines'], and Sub-Category: ['Copiers']) into a new predicate: Sub-Categories: ['Tables', 'Machines', 'Copiers'].



$$\mathbf{\Phi}_{+1} = \{\Phi \wedge \phi' | BF_{\Phi \wedge \phi'} > max(BF_{\Phi}, BF_{\phi'})\}$$

where BF_{Φ} is the JZS Bayes factor for a given predicate and ϕ' is a clause defined with an attribute not already in Φ. For example, consider a dataset defined by the attributes $\mathcal{F}_{\mathcal{D}} = \{sales, profit, state\}$, and the predicate $\Phi = \{\phi^{sales}_{[100,200]}\}$, representing all transactions with sale amounts between 100 and 200. This predicate can be refined by adding a clause defined with either of the remaining attributes profit and state. This results in the new predicates $\{\phi^{sales}_{[100,200]}, \phi^{profit}_{[a,b]}\}$ and $\{\phi^{sales}_{[100,200]}, \phi^{state}_{\{v_1,\ldots,v_k\}}\}$, where $a \leq b$ are possible values of the profit attribute and v_i are possible values of the state attribute. Φ_{+1} then includes all predicates where the new Bayes factor is greater than that of both the original predicate and the new clause. Any of the new predicates can be further refined by adding a clause with the remaining attribute (e.g., $\phi^{sales}_{[100,200]}, \phi^{profit}_{[a,b]}, \phi^{state}_{\{v_1,\ldots,v_k\}}$).

The RPI algorithm begins by defining a set of single-clause base predicates for each attribute. For discrete attributes, each predicate corresponds to one possible value:

$$\boldsymbol{\phi}_f = \{f \in [v] | v \in \{d^f_i | d_i \in \mathcal{D}\}\}$$

For continuous attributes, the range of possible values is discretized into b bins, and each predicate corresponds to a range of values:

$$\boldsymbol{\phi}_f = \{f \in [f_{min} + (i)f_{size}, f_{min} + (i+1)f_{size} | 0 \leq i \leq b\}\}$$

where f_{min} is the minimum value of f, and $f_{size} = (f_{max} - f_{min})/b$ is the size of each bin. At each step, the RPI algorithm yields a set of predicates, starting with a single "null" predicate that consists of zero clauses and contains every point in $\mathcal{D}$. On subsequent steps, each predicate in the set is refined by adding one additional clause as described above. If a predicate cannot be refined (i.e., there are no predicates with one additional clause and a higher Bayes factor) then this operation yields a set containing only the original predicate:

$$refine(\Phi) = \begin{cases} \boldsymbol{\Phi}_{+1}, & \text{if } |\boldsymbol{\Phi}_{+\mathbf{1}}| > 0 \\ \{\Phi\}, & \text{otherwise} \end{cases} \tag{3.3}$$

At step i, the RPI algorithm yields:

$$RPI_i = \begin{cases} \{\varnothing\}, & \text{if } i = 0 \\ \cup\{refine(\Phi) | \Phi \in RPI_{i-1}\}, & \text{otherwise} \end{cases} \tag{3.4}$$

where $\varnothing$ denotes the null predicate. The algorithm terminates when none of the predicates in RPI_i can be further refined (i.e., $RPI_{i+1} = RPI_i$).

The RPI algorithm employs a greedy approach, constructing optimal predicates incrementally by adding one attribute at a time. To mitigate the risk of becoming trapped in a local optimum, the algorithm begins its search from multiple initial base predicates, effectively creating multiple "starting points". However, this strategy can lead to redundancy, as the same predicate may be generated from different starting points, resulting in unnecessary computations. To address this, the RPI algorithm incorporates a pruning phase. During this phase, it checks whether a newly generated predicate has already been derived from another starting point,

and discards any duplicates that are identified.

3.6 Visual Interface

PIXAL is a VA system designed to support anomaly reasoning, helping users hypothesize, evaluate, contextualize, and communicate findings about collections of anomalies. PIXAL consists of three integrated views: The *Predicate View* serves as the starting point, displaying predicates generated by the RPI algorithm. Each predicate represents a hypothesis, described by a set of attribute values, and defines a collection of data points using statistics of its anomaly scores. Users can explore hypotheses **(T1)** by filtering, modifying, or creating new predicates, and evaluate collections of anomalies **(T2)** using the provided statistics. The *Score View* allows users to build trust in the statistics and evaluate anomaly scores in more granular detail as needed, visualizing the full distribution of anomaly scores for a given collection **(T2)**. Users can compare the distribution of anomaly scores for a collection with those of the entire dataset or other collections, allowing users to further evaluate a collection based on its relative anomalousness.

The *Attribute View* supports contextualizing anomalies in terms of domain-relevant attributes **(T3)**. Selecting a predicate populates this view with a grid of small multiples, each illustrating how the anomalous collection differs from normal data. Each plot highlights these differences and is accompanied by a natural language description to enhance interpretability **(T4)**. Interactions with PIXAL are dynamically propagated through all three views, ensuring consistency and integration throughout the anomaly reasoning workflow, and allowing analysts to move fluidly between hypothesizing, evaluating, contextualizing, and communicating their findings.

3.7 Predicate View

To begin, the *Predicate View* (Figure 3.2A) presents users with a number of plausible hypotheses generated by the RPI algorithm **(R1)** (see subsection 3.5.2). Users can

Figure 3.2: The PIXAL visual interface. The *Predicate View* displays subsets of the data with high anomaly scores. Users can filter and sort subsets generated by the RPI algorithm (A), or define new subsets from scratch or by editing and merging existing predicates (B). In the *Score View*, users compare the distribution of anomaly scores between different subsets and the full dataset (C). In the *Attribute View*, users visualize anomalies in the context of the original data dimensions with a set of small multiples (D). PIXAL assists users in communicating their results to stakeholders with natural language descriptions (E).

leverage their domain expertise to refine the space of plausible hypotheses by only showing predicates that include or exclude a given attribute. Users can create a new predicate, or modify an existing one by manually adjusting the values included in each clause (Figure 3.2B), allowing them to consider alternative predicates in addition to those generated by the RPI algorithm **(R2)**.

Once the user has identified a predicate representing a plausible hypothesis, they can immediately evaluate it by inspecting its statistics **(R3)**. The Bayes factor, calculated by the RPI algorithm, is reported as the *evidence* for the hypothesis that the anomaly scores in the collection are higher than those in the rest of the dataset. The evidence depends on the *size* of the collection, its average (*avg*) anomaly score, and the standard deviation (*stdev*) of its anomaly scores. When the user modifies a predicate, its statistics are dynamically recalculated, allowing them to rapidly transition between generating and evaluating hypotheses **(R4)**.

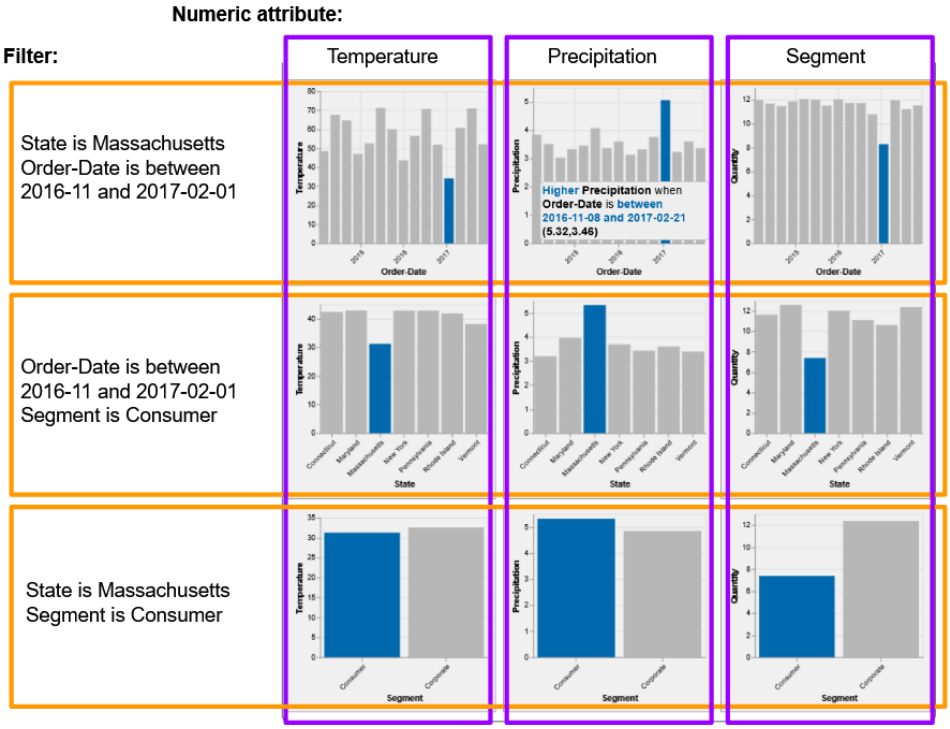


Figure 3.3: The Attribute View allows the user to contextualize a collection of anomalies. For a predicate with three clauses, each row is filtered by two attributes (e.g., State and Order-Date). The last attribute is encoded on the x-axis. Each row encodes a numeric attribute on the y-axis.

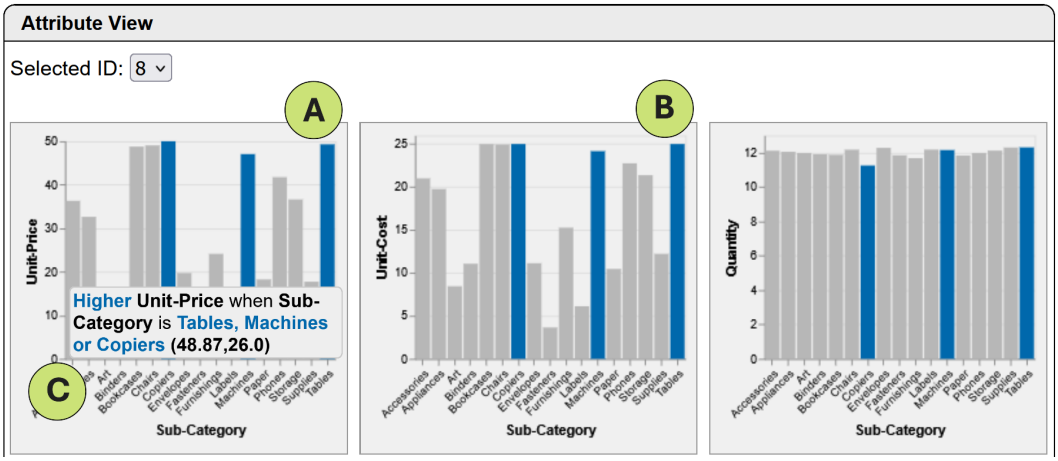


Figure 3.4: The analyst visualizes the new predicate in the Attribute View. The bar charts indicate that average Unit-Price and Unit-Cost are higher for Tables, Machines, and Copiers (A, B). PIXAL provides a natural language description for these relationships that the analyst accesses by hovering over each chart (C). In this example, PIXAL informs the analyst that the Unit Price of items in the Sub-Category of 'Tables', 'Machines', and 'Copiers' are higher (48.87) than that of items in other categories (26.0).

3.7.1 Score View

While the Predicate View provides summary statistics, the *Score View* allows the user to evaluate each collection at a more granular level (Figure 3.2C). The Score View includes a distribution for each predicate visible in the Predicate View. Distributions for selected predicates are highlighted by increasing their opacity. Additionally, the Score View includes the distribution of anomaly scores for the full dataset, allowing the user to assess each collection's relative anomalousness by comparing their scores to those of the entire dataset or another collection **(R3)**. The distributions update dynamically whenever a predicate is modified, maintaining tight integration between hypothesis generation and evaluation **(R4)**.

3.7.2 Attribute View

Once the user has confirmed that a collection is anomalous, the *Attribute View* (Figure 3.2D) allows them to explore the anomalies in the context of the data attributes. When the user selects a predicate, the Attribute View is populated by a grid of bar charts. Each row corresponds to one of the attributes included

in the selected predicate. Each bar encodes a possible value of that attribute (or one of 15 bins for continuous attributes), and its height encodes the average of one of the remaining numeric attributes of the dataset. The data visualized in each row are filtered by each attribute in the predicate, except the one encoded by the x-axis. If a value is included in the predicate, its bar is highlighted with a blue fill; otherwise, the fill is gray. Each plot highlights the difference between the anomalous collection and a normal collection based on the average value of an interpretable, domain-relevant attribute, rather than opaque anomaly scores (**R5**).

To facilitate interpretation, users can hover over each small multiple to reveal a natural language description. The description states the dependent variable, whether its average increases or decreases in the anomalous region, the average values for each region, the independent variable defining the anomalous region, and the specific values that define it. For example, "Average *Unit-Price* **increases** from **21.11** to **48.87**" (**R7**). The user can modify the filters or the anomalous region, and the small multiples will update. Additionally, this will create a new predicate reflecting the updated values. Observing the statistics in the Predicate View and the full anomaly score distribution in the Score View, allowing for tight integration between contextualizing the collection, refining the hypothesis, reevaluating anomaly scores, and returning to contextualization (**R6**).

3.8 Usage Scenario

We describe our tool with a usage scenario, where an analyst uses PIXAL to reason about anomalies in a sales dataset. The dataset used in this scenario is modified from the *Superstore Sales* dataset included with Tableau Desktop (version 2020.4). Each row in the dataset includes the `Sub-Category` of product sold (e.g. Furniture, Office Supplies), the `State` it was sold in, the customer `Segment` (e.g. Consumer, Home Office), `Order-Date`, the `Quantity` of the product being sold, and the `Unit-Price` and `Unit-Cost` of the product being sold. Additionally, the data includes the average `Temperature` and `Precipitation` on the `Order-Date`. In this scenario, an analyst

has been tasked by Superstore executives with identifying and reporting anomalous transactions. Specifically, the analyst is expected to identify transactions with an unusual `Quantity`, `Unit-Price`, or `Unit-Cost`. The analyst begins by generating anomaly scores for the dataset using the isolation forest algorithm[LTZ08] from Scikit-learn[PVG+11]. Next, the analyst loads the data and anomaly scores into the RPI algorithm, which generates 7 predicates:

1. Sub–Category: ['Tables']
2. Sub–Category: ['Machines']
3. Sub–Category: ['Copiers']
4. Order–Date: [2016−11−8, 2017−02−21] &
   State: ['Vermont']
5. State: ['Vermont'] & Segment: ['Corporate']
6. State: ['Massachusetts'] &
   Segment: ['Consumer'] &
   Order–Date: [2016−11−8, 2017−02−21]
7. Precipitation: [2.74, 4.59] & Temperature:
   [−0.09, 36.36] & Segment: ['Consumer']

3.8.1 Discovering Business Logic

The analyst notes that the top three ranked predicates all include a clause with the "Sub-Category" attribute (see Figure 3.2A). They select all three, and compare the distribution of anomaly scores in each collection to the full dataset (see Figure 3.2B). This confirms that transactions involving products with certain sub-categories (Copiers, Tables, and Machines) are consistently anomalous. However, the analyst is unable to discern from existing business logic why certain sub-categories would tend to be more anomalous than others. To investigate further, they filter the Predicate View to show only predicates that include the attribute `Sub-Category` (Figure 3.1A), and create a new predicate by merging them (Figure 3.1B):

8. Sub–Category: ['Copiers', 'Machines', 'Tables']

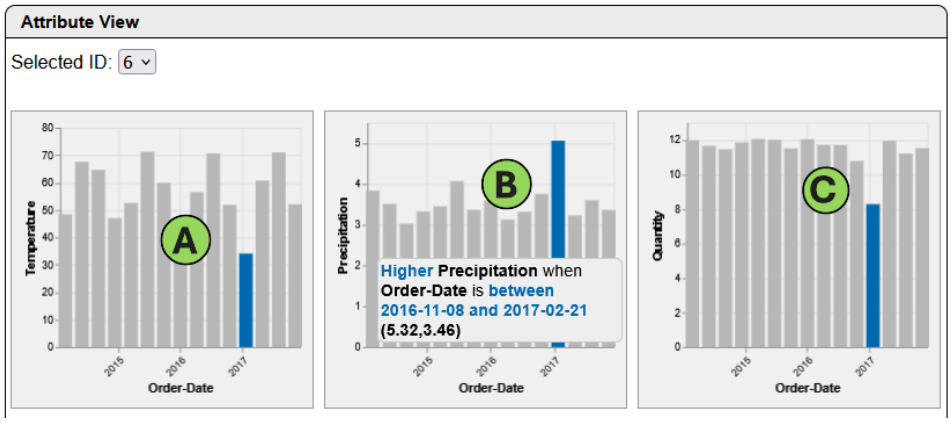


Figure 3.5: The Attribute View allows the user to contextualize a collection of anomalies. In this case, they observe that, between 2016-12-21 and 2017-01-15 and for the consumer segment, transactions in Massachusetts have lower `Temperature` (A), lower `Precipitation` (B), and higher `Quantity` (C).

In the Attribute View, they observe that average `Unit-Cost` and `Unit-price` are higher for Tables, Machines, and Copiers compared to other sub-categories, while `Quantity` is similar across all sub-categories (see Figure 3.1C). Hovering over each small multiple, PIXAL provides a natural language description of these relationships (see Figure 3.4). While this confirms that transactions involving Tables, Machines, and Copiers are indeed "*data outliers*" when viewed from the point of view of business operation, the fact that these items have higher production costs (`Unit-Cost`) and therefore higher prices (`Unit-Price`) is not surprising. Recognizing that these anomalies reflect an artifact of the anomaly detection algorithm (treating high `Unit-Price` as anomalous even when justified by proportional `Unit-Cost`) the analyst concludes these findings are not actionable and decides not to report them to stakeholders. Instead, they refine their investigation by excluding predicates with the Sub-Category attribute in the Predicate View, and continue their investigation.

3.8.2 Discovering Unexpected Outliers

Continuing their analysis, they select the next highest ranked predicate and confirm that this collection is highly anomalous in the Score View. In the Attribute View, they observe that, between 2016-12-21 and 2017-01-15 and for the `Sub-Category`

"Customer", transactions in Massachusetts have lower `Quantity`, lower `Temperature`, and higher `Precipitation` (see Figure 3.5). The analyst recognizes that lower sale quantities in Massachusetts during this period could indicate operational issues, potentially linked to adverse weather conditions. To confirm that these patterns are consistent across consumer segments, they copy the predicate and modify the `Segment` filter to include both Consumer and Corporate transactions (see Figure 3.7B). Predictably, the analyst finds that the unusual weather pattern does not depend on the `Segment`. However, the anomaly scores of the new collection are not particularly anomalous as visualized in the Score View (Figure 3.6). This suggests that, while the unusual weather event impacted Consumer sales, the same effect was not seen in Corporate sales. Anticipating that stakeholders will want to compare their findings with existing monthly reports, they align the date filter to include each full month (from 2016-12-01 to 2017-01-31) (see Figure 3.7A). After confirming that the expanded collection is still anomalous, they refer back to the natural language descriptions generated by PIXAL and prepare to present their results to stakeholders.

3.9 Evaluation: Interview Study

We conducted an interview study with three professional data analysts to evaluate our anomaly reasoning framework and the effectiveness of PIXAL in solving an anomaly reasoning task.

Participants: We recruited three professional data analysts to participate in the interview study. As part of the recruitment process, all participants confirmed that they had experience with anomaly reasoning as part of their job duties. Two of the analysts worked as data scientists for healthcare companies, and the third analyst was a data scientist at an academic institution. One of the participants in the healthcare domain had previously participated in the initial requirement gathering interview.

Procedure: We walked each participant through a scenario in which an analyst

uses PIXAL to perform anomaly reasoning on the *Superstore sales* dataset and report their findings to decision makers. After introducing the dataset, we demonstrated the use of PIXAL in the scenario described in Section 3.8. Participants were encouraged to ask questions and direct the exploration process in the style of pair-analytics [AHKF11]. Each participant was interviewed following the demonstration of PIXAL. For each task outlined in section 3.3 – *Hypothesize*, *Evaluate*, *Contextualize*, and *Communicate* – participants were asked to evaluate the effectiveness of PIXAL in relation to their current workflow. Interviews were structured around four open-ended questions to facilitate discussion:

1. Is this task necessary for anomaly reasoning?
2. What capabilities would a system need to include in order to support the task?
3. How does your typical workflow support these capabilities?
4. Would using PIXAL be an improvement over your current workflow?

Each interview lasted about 60 minutes, with 30 minutes dedicated to the analysis of the *Superstore* data using PIXAL and 30 minutes for the follow-up questions and discussions. All interviews were conducted over video conference.

3.10 Study Results

In our analysis of each interview, we recorded whether the participant considered each task relevant for anomaly reasoning, and instances where the participant noted a positive or negative difference between PIXAL and their current workflow. All three participants agreed that the tasks **hypothesize**, **evaluate**, **contextualize**, and **communicate** accurately represent their experience with real-world anomaly reasoning tasks, confirming the results of our initial interview study. In this section, we elaborate on insights from participants for each task, focusing on the comparison between PIXAL and their current workflow. Overall, their feedback suggests that PIXAL would be a significant improvement over current anomaly reasoning workflows.

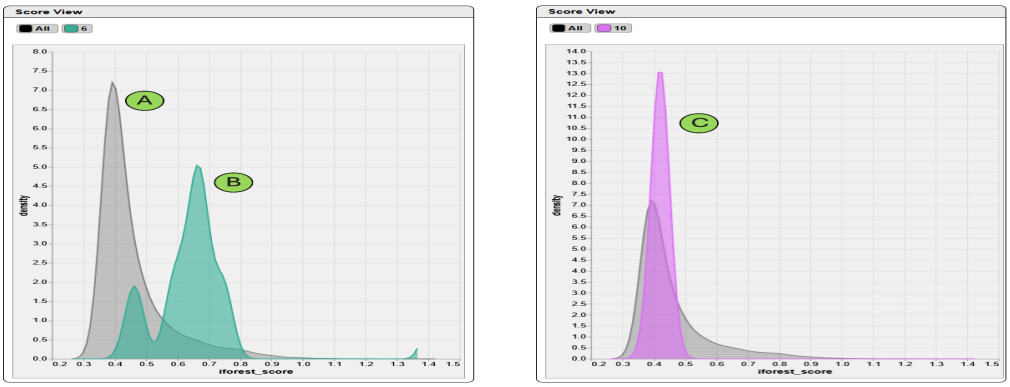


Figure 3.6: Interacting with a predicate will modify the Score View in real time. In this case, the user modifies a predicate by changing the "Consumer" `Segment` to "Corporate". (see Figure 3.7B). In the first panel, the Score View displays the distribution of anomaly scores of the original predicate compared to the full dataset, with the x-axis encoding anomaly score and the y-axis encoding density. Comparing the two distributions shows that data points in the predicate (B) are relatively anomalous compared to the full dataset (A) (i.e., distribution B is to the right of the distribution A). In the second panel, the Score View shows a new predicate that the user created by changing the `Segment` from "Consumer" to "Corporate" (C). Compared to the full distribution, the new predicate does not appear to have particularly high anomaly scores. This demonstrates that Consumer sales were impacted by anomalies while Corporate sales were not.

3.10.1 Hypothesize

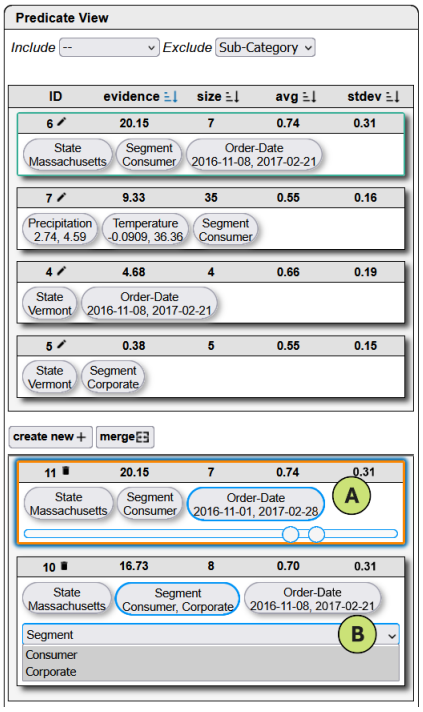
All three participants stated that identifying hypotheses that could explain collections of anomalies was a crucial first step in their anomaly reasoning process. Considering the benefits of PIXAL over their current workflows, participants emphasized the use of predicates to represent collections of anomalies and hypotheses. One, focusing on the benefit of working with predicates over individual anomalies, noted: "*Understanding the data as predicates instead of the raw data makes it a lot easier to consume, even for someone who knows what they're looking for.*" Another participant emphasized the interpretability of predicates: "*The predicates seem flexible and easy to understand. You're really just talking about grouping with different features.*"

All participants also highlighted the generation of predicates supported by PIXAL as a significant improvement over their current workflow. One participant noted the usefulness of approaching potential anomalies from multiple perspectives with automatically generated predicates: "*Having the ability to cut across the data in many different ways is always useful.*" Another participant focused on the utility of user-defined predicates, noting "*Analysts need to be able to play with their data and experiment with different hypotheses that might not necessarily be presented by the system. These tools seem like they would help investigate alternatives.*" The remaining participant noted the benefits of generating multiple predicates in cases where the data can contain groups of anomalies coming from distinct underlying phenomena: "*When we're trying to explain a group of anomalies, we test a collection of hypotheses. Having multiple predicates lets you compare multiple different classes of anomaly types.*"

3.10.2 Evaluate

All three participants stated that evaluating collections of data points to determine if they are, as a whole, anomalous, was a critical part of anomaly reasoning. In their assessment of PIXAL, all three participants emphasized trust-building as a major

Figure 3.7: PIXAL allows the user to test alternative hypotheses by editing existing predicates. The user can edit the minimum and maximum values for a continuous attribute by adjusting a slider (A), and the values for a discrete attribute by adding or removing from a drop-down list (B). These interactions will update the distribution view in real-time (see Figure 3.6) such that the user can receive immediate feedback from testing different predicates.



improvement over their current workflow. One participant specifically noted the ease of confirming the anomalousness of a collection using PIXAL, stating: "*The predicates generated by PIXAL seem easy to verify. I trust them more knowing that I can easily check whether those data points are actually anomalous or not.*" Another participant noted the interactive investigation of predicates as a way of building trust: "*The more interactions a user has access to, the more they will trust the system. PIXAL seems like it provides a complete set of tools for interacting with and building trust in predicates.*" The remaining participant also emphasized PIXAL's interactive capabilities during the evaluation task: "*A lot of times people already have an idea of what they're expecting to see. The only way they'll trust a system that's telling them something that goes against their intuition is to confirm it themselves.*"

3.10.3 Contextualize

All three participants stated that contextualizing anomalies in terms of domain-relevant attributes was a critical step in their typical anomaly reasoning workflow. Both participants from the healthcare domain noted that decisions made based on anomaly scores alone with no additional context would not be acceptable in most cases. One participant emphasized explicit regulatory constraints, stating: "*At this moment there is no accepted mechanism from a regulatory perspective enabling a system to determine what the anomaly is*," while the other participant focused on the complexity of the anomalies themselves: "*We can determine from the output of an algorithm whether there is [abnormality in the data], but determining what exactly makes it abnormal must be done by a human*"

Additionally, two participants commented on improvements to their workflow supported by the small multiples view. One emphasized the usefulness of the plots in contextualizing anomalies: "*The plots help you quickly identify patterns and understand which features might be contributing to anomalies.*", while the other participant praised the combination of text descriptions and plots: "*The text and plots work well together. You can quickly compare anomalies to normal data by reading the text and get a better sense of magnitude and overall pattern by looking at the plot.*"

3.10.4 Communicate

All three participants stated that communicating results to stakeholders was a critical final step in their typical anomaly reasoning process. Two participants stated that the integrated communication tools included in PIXAL would be an improvement over their typical workflow. Specifically, one participant noted that the plots automatically generated by PIXAL could be directly communicated to stakeholders: "*An analyst might not be satisfied by a statistically fragile summary like average quantity, but usually [the images generated by PIXAL] would be sufficient for reporting purposes*". The other participant noted the ability to easily adjust results to

align with stakeholder expectations would be a major improvement to their current workflow: "*The date ranges in the predicates generated by PIXAL do not necessarily align with time periods that are meaningful for the business. In this case, it would be helpful to modify the predicate so that date ranges align with financial quarters or some other meaningful period.*"

3.11 Discussion and Limitations

In this chapter, we proposed a formal definition of anomaly reasoning and introduced PIXAL, a visual analytics system designed for this task. In this section, we reflect on limitations both in our formal definition and design of PIXAL, and discuss opportunities for future work.

3.11.1 Anomaly Reasoning

Future work will continue to develop our formal definition of anomaly reasoning. In particular, future work will extend this formalization to take the role of business stakeholders in anomaly reasoning into account. Additionally, we note that the algorithm and visual interface that make up the core of PIXAL can be configured for use cases other than collections of anomalies. Future work will use our formalization as a starting point to investigate similar reasoning tasks.

Formalizing the Stakeholder: Our interviews revealed that anomaly reasoning requires close collaboration between data analysts and business stakeholders. While our formalization focused on analysts' roles, future work will explore stakeholders' responsibilities, such as defining normal vs. anomalous behavior, assessing business impact of a collection of anomalies, and determining corrective actions. In particular, we expect to find that the communication of business knowledge to data analysts plays an important role in anomaly reasoning, similar to the communication of results from data analysts to stakeholders.

Exploring Similar Reasoning Tasks: The primary goal of anomaly reasoning is to identify collections of data points that are highly anomalous. As such, the RPI

algorithm employed by PIXAL generates predicates with higher anomaly scores than the rest of the data (i.e., high Bayes factor). Similarly, PIXAL's visual interface is designed with the assumption that each predicate represents a collection of anomalies. Specifically, visualizations are designed to help analysts evaluate whether collections are anomalous, and contextualize the collection alongside normal data, while interactions are designed to help analysts explore and filter hypotheses. While PIXAL maximizes Bayes factor, the RPI algorithm can be configured to optimize any aggregate measure defining a collection. Additionally, the interpretability of predicates allows visual analytic systems to support diverse visualizations and interactions. Future work will explore applying this approach to other reasoning tasks beyond anomaly detection.

3.11.2 Predicate Induction

PIXAL's use of a predicate induction algorithm to generate multiple interpretable groupings of anomalies was found to be useful by professional data analysts in our interview study. However, we note that PIXAL can be further improved, especially with regard to its computational efficiency when analyzing large datasets. In PIXAL, the ability to generate multiple overlapping groupings requires a recursive approach to searching through the space of possible predicates instead of greedily maximizing Bayes factor. While the RPI algorithm only takes a few seconds to run on small datasets (10k rows and 10 dimensions), larger datasets (100k rows, 30 dimensions) can take 20+ minutes, while really big datasets (1 million rows, 100 dimensions) can take hours to complete. In PIXAL, this is mitigated by generating predicates offline as a preprocessing step. However, improving the performance of the predicate induction algorithm could lead to a more interactive, user-guided predicate generation process.

Dimensionality Reduction: One possible way to optimize the RPI algorithm is to first perform dimension reduction, such as PCA, on a dataset with a large number of dimensions. The downside to this approach is that this will make the resulting

predicates difficult to interpret. However, in return, this approach reduces run time while also creating more precise predicates because the choices of the attributes are no longer limited to axis-aligned dimensions. The trade-off between interpretability and performance depends on many factors such as data size, access to computation resources, etc. Future designers of anomaly reasoning systems similar to PIXAL will need to take these factors into consideration when making the appropriate trade-off decision.

Alternative Algorithms: Another approach to improving the speed of predicate induction is to explore alternative algorithms. For example, DimBridge [MAR+24] finds predicates by optimizing a differentiable loss function. This results in faster performance, at the cost of generating fewer predicates. Future work will consider alternatives to the greedy RPI algorithm, taking into account the trade-off between speed and accuracy.

3.12 Conclusion

In this chapter, we develop a formal definition of anomaly reasoning based on interviews with data professionals, and present PIXAL, a visual analytics system for anomaly reasoning. By representing collections of anomalies as first-order predicates, PIXAL allows analysts to work directly with patterns of anomalies rather than individual point anomalies. To aid analysts in the **Hypothesize** task, PIXAL automatically generates predicates using a recursive algorithm. In the *Predicate View*, analysts can explore predicate generated by the algorithm or create new predicates. The *Score View* allows analysts to **Evaluate** collections by comparing their distributions of anomaly scores The *Attribute View* allows analysts to **Contextualize** anomalies by visualizing them alongside normal data. PIXAL provides natural language descriptions, helping analysts to **Communicate** their findings to stakeholders. We demonstrate the effectiveness of PIXAL in a usage scenario and follow-up interviews with data professionals.

Chapter 4

DimBridge

4.1 Problem Statement

The demand for interactive visual exploration techniques for high-dimensional data has significantly grown due to the substantial increase in data generation over the last decade [LMW+17]. Among these techniques, dimensionality reduction (DR) is one of the most prominent methods for visualizing high-dimensional data. Well-known DR techniques such as MDS [KW78], PCA [Pea01], t-SNE [vdMH08], and UMAP [MHSG18] generate low-dimensional projections of data, where the spatial proximity of points encodes a measure of similarity between data items [NA19, EMK+21]. These techniques offer 2D representations of high-dimensional data, facilitating users in identifying data patterns in the high-dimensional space more readily.

However, despite the utility of DR methods in understanding *what* data items are similar, few provide insight into *why* the data are similar. As a result, users often need to rely on patterns that they perceive in the projection, e.g., clusters of points, spatial density, or distinctive shapes, to infer characteristics of the original high-dimensional data [SG17, BSIM14, SMT13, KEV+18]. While some of these patterns may indeed reflect underlying data characteristics, they can also be influenced by noise, artifacts stemming from the DR process [MCMT14, NA19, KB19], or even the propensity of humans to see visual patterns

where none exist [WVJ16, Ell18, WFG+12]. Modern visualization tools and interaction techniques offer some assistance in understanding projections, such as helping users discern DR distortions [Aup07, LA11, SDMT15], examining or contrasting perceived clusters [LWCC18, MJEG21, EHA+23], or generating new plausible data points in the projection [dBD+12, AVMC+15, EAS+23].

In this chapter, we present DimBridge, building on existing work to offer a new way to explore and understand perceived visual patterns in a DR projection. The workflow for DimBridge is designed for simplicity. As shown in Figure 3.2, a user highlights a pattern in the projection space, and DimBridge (1) identifies relevant data dimensions and (2) determines value intervals for each dimension to explain the selected visual pattern. The resulting subspace, defined by the selected data dimensions and intervals, is visualized using a scatterplot matrix (SPLOM). This visualization enables a user to observe the selected visual pattern in a reduced context of the original high-dimensional data space – known as *"Mapping Synthesized [Data] to Original Dimensions"* in DR task taxonomies [BSIM14, NA19]. By linking visual patterns in the projection space to the original data space, DimBridge helps users understand the meaning of these patterns in a familiar context.

DimBridge supports several user interactions (select, select and contrast, select and draw) for identifying visual patterns in a projection, including clusters, outliers, and shapes. Once a pattern is selected, DimBridge automatically identifies *predicates* by optimizing a subset of data dimensions and intervals that best explain the visual pattern. In addition to highlighting exactly where the pattern occurs within this subspace, this approach yields a reduced subspace of dimensions that is practical to visualize using a SPLOM and allows users to further contextualize observed patterns within the original data space. DimBridge is equipped with two predicate induction algorithms, a novel Predicate Regression algorithm tailored for speed and scalability, complemented by the PIXAL algorithm for accuracy[MBBC22b]. Both algorithms allow for a smoothness constraint, which enables DimBridge to support a novel select and draw interaction. Within this framework, predicates serve as a conceptual bridge connecting the low-dimensional

(i.e., the projection) and the high-dimensional (i.e., the original data) spaces.

To demonstrate DimBridge's efficacy, we showcase the system on various datasets, present its utility in a case study, and iteratively evaluate its functionality with domain experts. With these examples, we also hope to demonstrate that DimBridge provides a means for users to explain projections using the original data space, solving key challenges in interpretability associated with conventional projection-based high-dimensional data visualization.

In summary, our work makes the following contributions:

- We introduce DimBridge, a system that uses predicate logic to *bridge* the projection and data spaces by simultaneously identifying dimensions and intervals that explain a given pattern, helping users to make sense of 2D projections of high-dimensional data.
- We introduce a new interaction design for selecting visual patterns within DR projections and a new algorithm for generating predicates with smoothness constraints given a selected visual pattern (Predicate Regression).
- We evaluate DimBridge through several showcases with different domain applications, a case study, and evaluations with researchers from materials science and pharmaceutical drug discovery.

4.2 Interpreting DR Results: Design Goal and Tasks

Patterns identified in conventional DR visualizations are challenging to interpret because they lose the underlying semantics of the original dimensions, such as the context for why points are considered similar. Our work addresses these challenges by bridging the projection and underlying data space. Specifically, DimBridge facilitates the interpretation of patterns found in DR results through interactive querying of the projection, enabling retrieval and visualization of relevant subspaces of the original dimensions.

We break down the process of identifying and interpreting patterns into three high-level tasks, developed while working closely with domain experts. These tasks

are the foundational requirements used to develop the DimBridge system.

(T1) Identify and query visual patterns in a projection

Dimensionality reduction (DR) transforms high-dimensional data into low-dimensional forms to facilitate visualization and analysis while preserving important structures and relationships. A common approach is to reduce the data to two dimensions and visualize the resulting patterns using a scatterplot. DimBridge should support the user in identifying potential patterns of interest and facilitate direct engagement with the projection, enabling users to explore by selecting patterns that capture their interest. There are multiple patterns that a user can identify within a projection that have unique approaches to selection and interaction. DimBridge must allow the user to identify and select each of the following patterns:

Clusters: Identifying clusters is a key application of DR projections [SZS+16], as it allows a way to classify similar and dissimilar points based on attributes. Although it can be intuitive to identify distinct groups of data points, it is not directly clear why they formed or how they differ by visualizing the projection alone.

Outliers: Identifying outliers is closely linked to recognizing clusters, yet determining if outliers in a 2D projection accurately reflect those in the high-dimensional data is challenging based solely on the projection. While outliers may be identified based on distance in the projection, it is not clear how this distance translates to the original dimensions.

Spatial Density: Identifying clusters is not always straightforward, especially with varied point densities [HE11, SHGF16]. Unlike clear-cut clusters, density across a projection can change gradually, complicating the task of selecting a group of points for detailed analysis. For example, points just outside a selected dense area might still belong to that group, reflecting the challenge of making precise selections.

Shapes: DR projections are also used to uncover hidden low-dimensional structures in data [SZS+16], such as identifying a low-dimensional manifold that explains the data distribution. These structures often appear as specific shapes within the pro-

jection, such as a curve slicing through a group of points.

(T2) Retrieve and visualize relevant subspaces of the original data

Understanding visual patterns in the DR results requires explaining them in the context of the data's original dimensions. Traditional visualization techniques, however, struggle with handling high-dimensional data effectively. Given a visual pattern in the DR results queried by the users, DimBridge must be able to retrieve relevant subspaces of the original data. For this subspace to be visualized effectively, it must have a relatively small number of dimensions.

(T3) Evaluate the visual pattern in context

While the goal is to explain a visual pattern in the original dimensions, distortions introduced by the DR algorithm can result in patterns in the DR results that do not correspond to patterns in the original dimensions. Beyond simply retrieving relevant subspaces, DimBridge must help users evaluate a selected pattern given a retrieved subspace.

4.3 Example Use Case and System Overview

We illustrate DimBridge's design goals with a simple use case. Figure 3.2 shows a set of animal images labeled with 14 numeric attributes (e.g., furry, whiskers). After applying UMAP for dimensionality reduction and visualizing the results, the user identifies a cluster of data points forming an unusual curved shape in the *Projection View*.

The user investigates this pattern by selecting points at the top of the cluster with a bounding box and dragging it to the bottom, following the cluster's curved shape (T1). DimBridge then identifies the key dimensions and intervals that explain the visual pattern (T2). These dimensions are visualized in a scatterplot matrix in the *SPLOM View*, with the intervals shown in the *Predicate View* (T3). With these visualizations, the user observes:

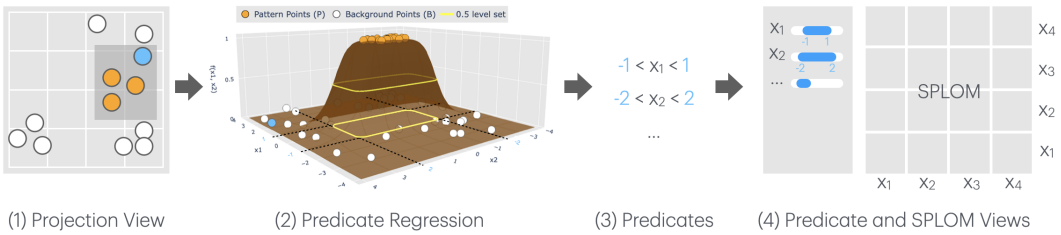


Figure 4.1: System Overview. (1) User makes a brush selection in the projection view. Note that the user intends to select the cluster of orange points, but includes another data point (shown in blue) by mistake due to DR distortions. (2) DimBridge fits a multi-dimensional "bump" function (Equation 4.5) to encapsulate the user selected points (orange). Notice that the algorithm automatically removes the blue point from consideration. (3) The "bump" function can be described using a reduced set of the original data dimensions and their respective intervals. These, in turn, can be represented as predicates. (4) The predicates are used to generate the predicate and the SPLOM views, which show the intervals and the selected data in the original data dimension, respectively.

- The selected shape can be explained using just 4 of the original 14 dimensions.
- The shape most closely correlates with the "furry" and "whiskers" dimensions.
- The top of the shape represents very furry animals with whiskers and big ears. Towards the bottom, the animals become larger, less furry, and have small ears and no whiskers.
- There is a downward shift in the "big ears" interval where the shape curves right, indicating a tipping point where the animals' ears become smaller.
- The shape represents a progression from foxes to wolves, wolf-like dogs, large dogs, and finally smaller puppies, which the user can see in the returned data.

To support this workflow, DimBridge models the user's interaction as a set of continuous brushes. In Figure 3.2, the user's drawn shape is discretized into 12 brushes, shown as 12 intervals for each dimension in the Predicate View. DimBridge uses a novel algorithm, Predicate Regression, to identify the dimensions and intervals that best explain these brushes by minimizing a differentiable loss function (see Figure 4.1 (2)) and enforcing a **smoothness constraint** to ensure consistency between the 12 brushes.

In section 4.4 we describe our predicate induction engine. Section 4.5 covers

the design of the visualization interface and interaction techniques. We demonstrate DimBridge's utility with three examples and a use case in drug discovery.

4.4 Predicate Induction Engine

The central motivation of this work is to support users' interpretation of patterns found in the DR results in the context of the original data dimensions. The predicate induction engine facilitates this by generating predicates that best explain, in the original dimensions, a visual pattern identified by the user.

4.4.1 Bridging DR and Data Patterns with Predicates

DimBridge uses first-order predicate logic as a "bridge" between patterns in the DR results and relevant subspaces of the original dimensions. This allows users to capitalize on the strengths of both spaces: patterns can be identified visually in the DR results, and their semantics can be recovered in the original dimensions.

A first-order predicate, Φ, is defined as a conjunction of one or more clauses, each consisting of a data dimension and a minimum and maximum value. The clause ϕ_j is defined for the j-th dimension, with minimum and maximum values denoted ϕ_j^{min} and ϕ_j^{max}. A predicate contains all data points with values that fall within the intervals defined by each clause. Alternatively, a predicate can be thought of as a function that takes a data point, x_i, as input and outputs a binary label: 1 if the data point falls within the intervals defined by each clause, and 0 otherwise:

$$\Phi(x_i) = \mathbb{1}[\phi_j^{min} \leq x_{ij} \leq \phi_j^{max},\ \forall \phi_j \in \Phi] \tag{4.1}$$

4.4.2 Generating Predicates from User Interactions

The predicate induction engine enables DimBridge to leverage predicates as a bridge between patterns in the DR results and original dimensions by generating predicates that closely match the user's selection. Unlike existing approaches that identify a subspace of relevant dimensions, DimBridge seeks a compact description of the

selected pattern in the original dimensions. Our predicate induction approach jointly identifies both dimensions and intervals that describe the selected pattern.

To achieve this, we first derive a set of background points, B, and pattern points, P, from a user's interaction. Second, we define the combined dataset $X = B \cup P$ and associated binary labels, $Y = \{\mathbb{1}[x_i \in P] | x_i \in X\}$. Finally, we use a predicate induction algorithm to identify predicates that contain a set of data points, represented by the binary labels $\Phi(X) = \{\Phi(x_i) | x_i \in X\}$, that closely match the data points selected by the user, represented by Y.

In our implementation of DimBridge, we consider two predicate induction algorithms. The first is the recursive predicate induction (RPI) algorithm used by the PIXAL system [MBBC22b]. This algorithm constructs predicates from the ground up, starting with a set of single clause predicates that are iteratively refined. A predicate is scored at each step and continuously refined until the score stops improving. For DimBridge, we score predicates using the F1 score calculated between $\Phi(X)$ and Y, balancing false positives (points that are in the predicate but not the user's selection) and false negatives (points that are in the user's selection but not the predicate). This results in multiple, overlapping predicates, each representing a candidate subspace. While this may be suitable for offline applications, this is too slow for an interactive system. The second is a novel algorithm, Predicate Regression that works by approximating high dimensional bounding cuboids via a differentiable proxy function.

4.4.3 Predicate Regression

Rather than generating and evaluating predicates iteratively, the Predicate Regression algorithm defines a differentiable loss function based on a reparameterization of first-order predicates, which it then minimizes to find the "best" predicate given the user-defined X and Y.

When a predicate clause fully covers the data extent along a dimension, it always returns true. Therefore, even if a predicate defines a subspace including only the clause dimensions, we can expand it to all dimensions with the interval being

strictly greater than the data extent:

$$\Phi(x_i) = \mathbb{1}[\phi_j^{min} \leq x_{ij} \leq \phi_j^{max},\ \forall j = 1 \dots M] \tag{4.2}$$

with $\phi_j^{max} \geq \max_i\{x_{ij}\}$ and $\phi_j^{min} \leq \min_i\{x_{ij}\}$ when $\phi_j \notin \Phi$.

This expansion allows us to optimize all predicate clauses simultaneously. Moreover, we can define a predicate with the parameters $\boldsymbol{\mu}$ and $\mathbf{r}$, denoting a midpoint and range for each dimension (j) respectively:

$$\mu_j = \frac{\phi_j^{max} + \phi_j^{min}}{2},\ \ r_j = \frac{\phi_j^{max} - \phi_j^{min}}{2} \tag{4.3}$$

This reparameterization allows us to consider whether a data point is contained by a predicate not only as a binary label, but a continuous probability:

$$Pr(\Phi(x_i) = 1|\mathbf{r}, \boldsymbol{\mu}, b) := \frac{1}{1 + \sum_{j=1}^{M} |\frac{1}{r_j} \cdot (x_{ij} - \mu_j)|^b} \tag{4.4}$$

Generating a Predicate for a Single Brush: Geometrically, the probability gives a rounded bump function (see Figure 4.1 (2)) of x_i centering at $\boldsymbol{\mu}$, where b is a fixed parameter controlling the steepness of the bump. Moreover, the 0.5 level set is enclosed by the predicate bounding box, $\prod_{j=1}^{M}[\mu_j - r_j, \mu_j + r_j]$, along each data feature dimension j. To find the optimal predicate via optimizing the parameters, we rewrite $1/r_j$ as a_j and view the probability as a differentiable function of a_j and μ_j:

$$f(x_i|\mathbf{a}, \boldsymbol{\mu}, b) = \frac{1}{1 + \sum_{j=1}^{M} |a_j \cdot (x_{ij} - \mu_j)|^b} \tag{4.5}$$

Given an X and Y defined by a user's selection, the loss function, binary cross entropy (BCE), is defined as:

$$\mathcal{L}_{bce}(\mathbf{a}, \boldsymbol{\mu}|X, Y) = \frac{1}{N}\sum_{i=1}^{N} y_i \log(f(x_i)) + (1 - y_i)\log(1 - f(x_i))$$

Recall that a_j is the inverse of the range r_j. Enlarging the range beyond data extent (i.e. forcing $a_j \to 0$) effectively eliminates the corresponding clause in the predicate conjunction. Therefore, selection of features in the predicate can be achieved through an L_1 regularization, $||\mathbf{a}||_1$. Eventually, minimizing loss functions over $\mathbf{a}$ and $\boldsymbol{\mu}$ gives a predicate in the form $\prod_{j=1}^{M}[\mu_j^* - 1/a_j^*, \mu_j^* + 1/a_j^*]$, where

$$\mathbf{a}^*, \boldsymbol{\mu}^* = \arg\min_{\mathbf{a},\boldsymbol{\mu}} \mathcal{L}_{bce}(\mathbf{a}, \boldsymbol{\mu}) + \gamma_1 \cdot ||\mathbf{a}||_1 \tag{4.6}$$

and γ_1 controls the strength of the predicate sparsity.

Generating Predicates for Contrasting Two or Multiple Continuous Brushes: A key requirement for a predicate induction algorithm in the interactive nature of DimBridge is the consistency and smoothness of results from consecutive interactions. For example, one may expect smoothly changing clauses when fine-tuning, contrasting queries, or brushing curves over a region. For these cases, a constraint can be added to the objective function that encourages smoothness in the resulting predicates. The draw interaction results in a discretized sequence of selections derived from the user's gesture. Given a sequence X_t, Y_t for $t = 1...T$, optimizing

$$\sum_{t=1}^{T} \mathcal{L}_{bce}(\mathbf{a_t}, \boldsymbol{\mu_t} | X_t, Y_t) \tag{4.7}$$

gives a sequence of predicates independent to one another. To compare two regions, we let $T = 2$; to brush curves over a region, we let $T = 12$.

To encourage consistency and continuity between consecutive predicates, a smoothness loss function is introduced:

$$\mathcal{L}_{smooth} = \sum_{t=2}^{T} \gamma_a \cdot ||\mathbf{a_t} - \mathbf{a_{t-1}}||^2 + \gamma_\mu \cdot ||\boldsymbol{\mu_t} - \boldsymbol{\mu_{t-1}}||^2 \tag{4.8}$$

where γ_a and γ_μ control the strength of consistency and continuity between consecutive predicates.

In entirety, the loss function becomes

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_{bce}(\mathbf{a_t}, \boldsymbol{\mu_t} | X_t, Y_t) + \sum_{t=1}^{T} \gamma_1 \cdot ||\mathbf{a_t}||_1 \tag{4.9}$$

$$+ \sum_{t=2}^{T} \gamma_a \cdot ||\mathbf{a_t} - \mathbf{a_{t-1}}||^2 + \sum_{t=2}^{T} \gamma_\mu \cdot ||\boldsymbol{\mu_t} - \boldsymbol{\mu_{t-1}}||^2 \tag{4.10}$$

4.5 Visual Interface

The DimBridge interface interface comprises three coordinated views: the *Projection View*, the *Predicate View*, and the *SPLOM View*. These views enable users to query patterns in the DR results, view and modify predicates, visualize relevant subspaces of the original dimensions, and evaluate the queried patterns in context. By coordinating these views, DimBridge allows for comprehensive exploration and understanding of patterns in high-dimensional data, linking DR results to original dimensions through interactive visualizations.

4.5.1 Projection View

The DR results are visualized with a scatterplot in the Projection View. A user can interact directly with the scatterplot through multiple brushing techniques to explore perceived patterns.

Select: A user can select a group of data points (P) distinct from the rest of the dataset (B) using the bounding box or lasso. The Predicate Induction Engine will then generate predicates to distinguish P from B in the original dimensions. This selection helps identify unique **clusters** based on shape or position, global **outliers** spatially distant from the rest of the data, and regions with varying **spatial density**.

Select and Contrast: Instead of comparing a cluster to the entire dataset, a user can use the bounding box or lasso to select another group of data points (B) for comparison. This interaction helps explain differences between two clusters, local outliers and their neighbors, or variations in spatial density within a specific region.

Select and Draw: If a group of data points forms a distinct, continuous **shape**,

a user can use the bounding box or lasso to select the starting point. Then, they can drag to the shape's endpoint, selecting points along the way. The Predicate Induction Engine will then generate predicates to distinguish the selected points (P) at multiple intervals along the shape.

4.5.2 Predicate View

The Predicate View bridges patterns in the DR results with the original dimensions, highlighting points in both the Projection and SPLOM Views. Each dimension in a predicate generated by the induction engine is listed along with its associated interval. Users can modify predicates by adjusting the intervals or adding/removing dimensions to see the impact on the other views. Intervals can be represented in several ways based on user interaction:

Select: Each dimension is displayed with a horizontal bar representing its full range of values. A highlighted segment indicates the predicate's defined interval (Fig. 4.2). Users can modify the predicate by dragging the endpoints of the highlighted segment (Fig. 4.6, 4.7).

Select and Contrast: Instead of one interval, two intervals (one for each selection) are displayed for each dimension (Fig. 4.3).

Select and Draw: A user's drawn gesture is converted into discrete selections, displaying multiple intervals vertically to save space (Fig. 3.2).

4.5.3 SPLOM View

The SPLOM View visualizes subspaces of the original dimensions in a scatterplot matrix, focusing on dimensions relevant to the predicates. Only including dimensions involved in the predicates reduces clutter and illustrates the relationship between these dimensions and the data points within the predicate.

4.5.4 Visualizing Predicate Accuracy

The Predicate Induction Engine aims to match a user's selection as closely as possible, but distortions from the DR algorithm can cause mismatches. These mismatches are illustrated using color in the Projection and SPLOM Views. Similar to prior work [JAL+22], this provides transparency on whether a cluster can be explained by a first-order predicate in the original dimensions.

Points in the Projection and SPLOM Views are color-coded based on their relation to the predicate. True positives (points in both the user's selection and the predicate) are shown in purple. False positives (points in the predicate but not in the selection) are shown in red, and false negatives (points in the selection but not in the predicate) are shown in blue. True negatives (points in neither the selection nor the predicate) are shown in grey.

4.5.5 Implementation and Performance

Our implementation of DimBridge consists of a web-based front-end and a Flask server[1]. The front-end visualization is implemented in Observable, rendering the scatterplots in the projection view and the SPLOM using WebGL.

The server-side Predicate Regression algorithm is implemented using PyTorch, which allows for GPU acceleration and distributed computing for improved performance. In our exploration with expert users, we maintained interactivity (typically less than 3 seconds) with datasets up to 2,703 dimensions and 22,000 instances on an Nvidia T4 GPU. However, the latency can be reduced with additional GPU support or by reducing the number of iterations in the optimization (Equations 4.6 and 4.8).

[1]The front-end and server source code are available on Observable (`https://observablehq.com/@tiga1231/dimbridge`) and Github (`https://github.com/tiga1231/dim-bridge`).

4.6 Showcases

In this section, we provide three showcases using DimBridge to analyze image data, motion-capture data, and scientific data. Each of these showcases highlights user tasks in exploring and making sense of high-dimensional data.

4.6.1 Understanding Model-Generated Images

We use DimBridge to analyze the output of a generative vision model, demonstrating the range of patterns in DR projections and how DimBridge helps make sense of them. We consider the StyleGAN3 [KAL+21] image generation model, which generates images of animals. We define 14 textual phrases as attributes describing visual appearances (e.g., "furry," "size") and behaviors (e.g., "excited," "suspicious looking"). Using CLIP [RKH+21], we score each attribute against a generated image. This set of continuously-valued attributes forms our high-dimensional space, alongside the generated images, to verify findings.

Explaining a Cluster: DimBridge demonstrates cluster analysis in a UMAP projection, highlighting discernible groups within the data, as seen in the scatterplot (Fig. 4.2). Upon brushing one of the clusters, DimBridge derives the predicate that best distinguishes those data items from the rest of the dataset. The subsequent predicate shows cheetahs as uniquely furry and spotted within the dataset (Fig. 4.2.3-4). There are a few false positives (colored in red), such as animals within the predicate but outside the brushed region. These are mostly cheetahs, indicating the brushed cluster was slightly imprecise.

As a baseline for comparison, we chose four dimensions and plotted the user's selection in the SPLOM (Fig. 4.2-4). Plotting these dimensions hardly contextualizes the user's selection against the dataset, whereas the algorithmically defined predicates allow reasoning about *why* this cluster is discriminative, highlighted in the scatterplot views (Fig. 4.2-3).

Contrasting Two Regions in Context: Beyond comparing a cluster with the overall dataset, users can also compare one cluster to another. In Fig. 4.3, DimBridge

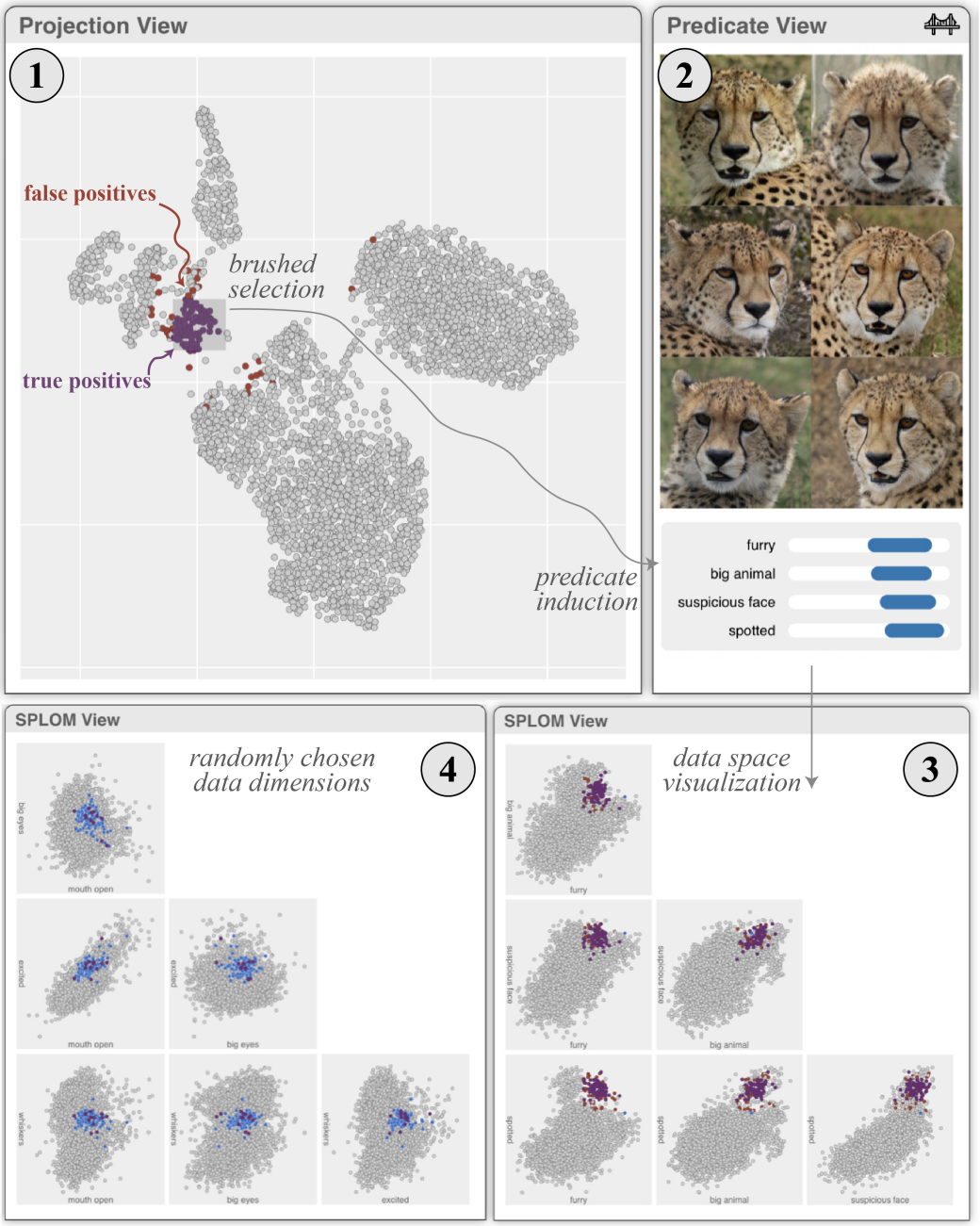


Figure 4.2: DimBridge allows one to better understand a potential cluster within the output space of a generative vision model. Upon performing a brush in the scatterplot (1), DimBridge finds a predicate comprised of 4 attributes (2) that, combined, help distinguish cheetahs from other animals (3), e.g. a big animal with spotted features. In comparison, highlighting brushed data points in randomly chosen four features (4) does not help in distinguishing key features of cheetahs from other animals.

Figure 4.3: DimBridge allows one to contrast one region of the dimensionality reduction plot from another. Upon brushing two regions, DimBridge finds a predicate that explains the two regions from the rest of the data points. DimBridge finds that while both kittens (blue) and puppies (orange) are not big animals, kittens have whiskers, and puppies have bigger ears.

highlights the differences between kitten and puppy clusters in the DR projection, distinguishing them by features such as "whiskers" and "big ears" and noting they are both categorized as smaller than other animals in the dataset.

4.6.2 Understanding Progression in Motion Captures

Biomechanical data, such as cyclic motion recordings, are crucial in orthopedic, rehabilitation, and sports research [HSMHW16]. The Multivariate Gait Dataset [HHW22, SPRHW08, HHHWP11, HSMHW16] captures human walking motion through 6 joint angles (left, right $\times$ ankle, knee, hip) over 101 timestamps and 10 repetitions, involving 10 subjects under 3 bracing conditions (unbraced, knee brace, ankle brace). For illustration, we selected 2 subjects from the dataset.

The DR plot is generated via UMAP using six angular features, excluding timestamps, repetitions, subject IDs, and bracing conditions. In the DR projection scatterplot, (Fig. 4.4), we identified three groups of loops corresponding to the three bracing conditions, each containing two overlapping repetition bundles. The difference between the two bundles is due to the subject, as shown in Fig. 4.4. Coloring the DR plot by attributes aids in understanding data one attribute at

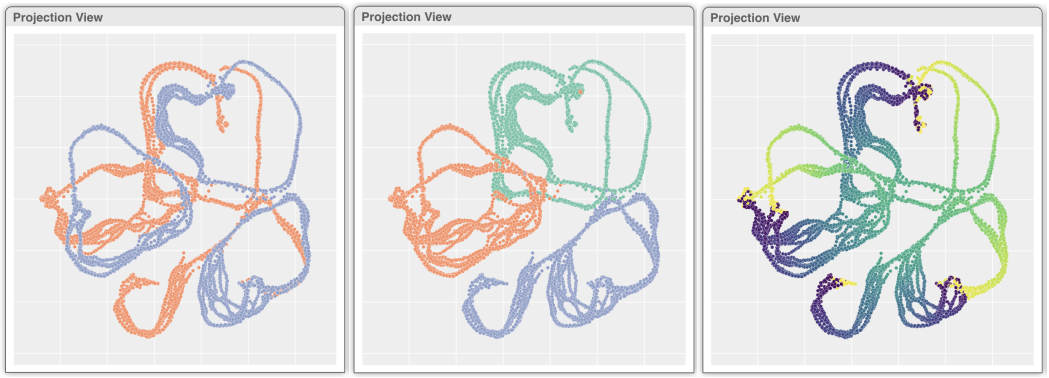


Figure 4.4: DR plot of the Motion Capture dataset. **Left:** color by subject. **Middle:** color by bracing conditions. **Right:** color by time.

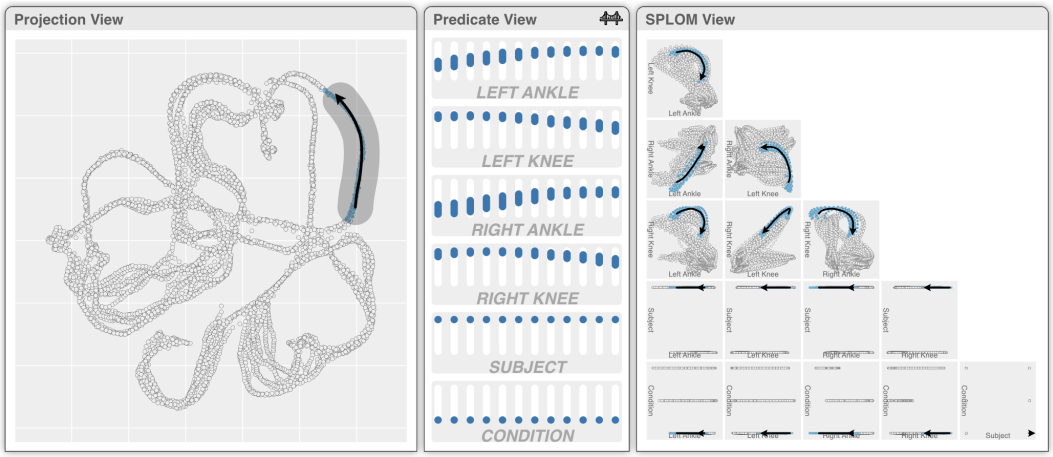


Figure 4.5: DimBridge shows that the curve following the flow of time in the figure captures only one subject and condition, and the segment represents a period with increased angles on left and right ankles and slightly decreased angles on left and right knees.

a time, but analyzing multiple continuous attributes simultaneously is challenging. Displaying all six joint angles in a SPLOM creates cognitive load due to the numerous subplots. However, DimBridge generates predicates that highlight a few key views, significantly reducing the visual workload in exploratory data analysis.

Understanding Progression Over a Curve: Brushing over a segment in the direction of time flow, the predicate induction algorithm identifies it as belonging to a single subject under a specific bracing condition (Fig. 4.5). It summarizes the progression as an increase in left and right ankle angles with a slight decrease in left and right knee angles. The SPLOM view confirms that this summary distinguishes the brushed segment from the rest of the data.

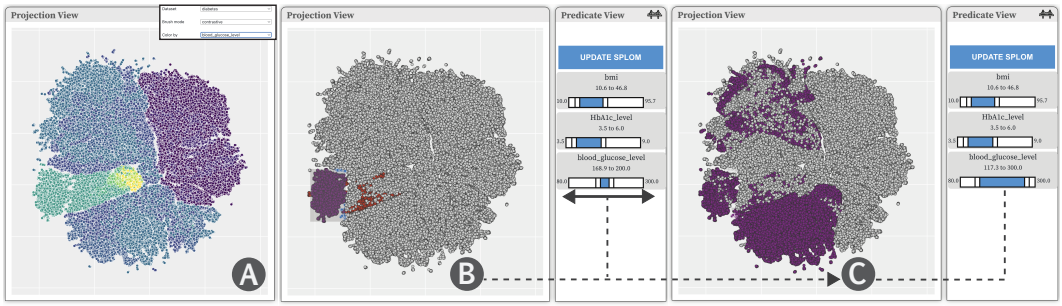


Figure 4.6: A system screenshot showing the projection and predicate components: (A) Users start by coloring the projection according to a feature, here Blood Glucose level. (B) A domain practitioner explores a data subset, creating a predicate based on their selection. (C) They then adjust the Blood Glucose range to a meaningful one, observing changes in the point distribution.

4.6.3 Examining Populations within a Diabetes Study

Healthcare remains one of the fastest-growing industries in the U.S. [BoLS] rapidly modernizing and producing data at an increasing rate [RC18b]. New techniques are needed to make sense of this influx of data. In this use case, we examine a diabetes dataset with eight features and a binary class label. The features are gender, age, hypertension, heart disease, smoking history, body mass index, HbA1c level, and blood glucose level [Mus23].

Specifying a Known Population: We demonstrate how a practitioner with domain knowledge can use DimBridge to manually adjust predicate ranges to find specific populations within the projection. The user starts by adding relevant features to the projection and SPLOM in the data-space visualization. In this case, the user is interested in BMI, HbA1c level, blood glucose level, and age. They can color the projection by blood glucose values (Fig. 4.6-A). Brushing a subset of interest in the lower left of the projection, they can generate predicates informed by the value range of this feature (Fig. 4.6-B).

Interactive Predicate Refinement: The user can then adjust the blood glucose level ranges in the predicate view to focus on particular states of the condition (Fig.4.6-B). By manipulating these ranges, the system dynamically updates the projection view, highlighting the data points that fall within the newly specified thresholds. This process is visualized in Fig.4.6-C, where the adjusted range causes

a redistribution of the highlighted points, providing immediate visual feedback.

The ability to update these ranges is not only important for identifying distinct diabetic populations but also for exploring the complex interplay between glucose levels and other biomarkers. Once ranges are updated, users can click the "Update SPLOM" button to examine pairwise relationships between blood glucose levels and other variables like BMI, HbA1c, and age. This refined analysis can reveal subtle correlations or patterns that might be missed with a static approach.

4.7 Case Study: Investigating Properties of the Mn1-xGexTe Alloy

The case study demonstrates the use of DimBridge in materials science to understand the structure of alloys that are hard to model with traditional methods. Our collaborators applied advanced sampling techniques [NNG+23] and density functional theory (DFT), a method to calculate material properties based on electron behavior, to explore the Mn1-xGexTe alloy structure. This resulted in about 500 theoretical models with different combinations of manganese (Mn) and germanium (Ge).

4.7.1 Theoretical Models and Empirical Validation: Current Analysis Methods

Characterizing theoretical models is crucial for understanding how atoms bond and arrange themselves and the local structure, which directly relates to the material properties. For non-periodic materials like alloys, this characterization combines theoretical and empirical methods to simulate atom distributions. The standard analysis involves comparing computational predictions with experimental data.

In this case study, empirical data validating the theoretical models were obtained from the Spallation Neutron Source at Oak Ridge National Lab. Researchers analyze data using computational simulations, graphical representations, and empirical data comparisons to understand and compare the local atomic structure of

different alloy compositions to experimental data.

Signals in the data, referred to as "peaks", are discussed in subsequent sections. Although generating lower-dimensional representations is straightforward, linking these to the underlying science driving the clustering remains challenging in our collaborators' current workflow.

4.7.2 Analyzing the Mn1-xGexTe Alloy with Dimbridge

Our collaborators used DimBridge as an exploratory analysis tool to understand the shared characteristics of computationally generated theoretical models for the Mn1-xGexTe alloy that align with experimental data from Oak Ridge National Lab. They specifically asked: **(1)** How does changing the proportion of manganese (Mn) affect the alloy's structure as it shifts from a slanted, diamond-like shape to a straight-edged, cube-like shape? **(2)** Are the shapes formed by Mn atoms consistent across the alloy space? For instance, do shapes with 6 out of 12 manganese atoms in their outer layer look the same across different Mn-Ge mixtures?

One collaborator, a graduate student, studies how varying the Mn and Ge ratio in alloy samples affects crystal structures to understand the relationship between local atomic coordination, global crystal structure, and material properties. Familiar with the data, she used DimBridge to generate hypotheses, find areas of interest, and refine her understanding and investigative strategies for subsequent data runs. During our collaborator's use of DimBridge, it became clear that the system is not just a tool for bridging dimensional spaces but also a catalyst for testing hypotheses and identifying areas for exploration.

Her exploration progressed through three distinct iterations, each using a revised dataset tailored to a new investigative strategy. All iterations used datasets from the five subsets but differed in the properties of these subsets. Subsets varied in Mn to Ge combinations: $Mn_{0.125}Ge_{0.875}Te$, $Mn_{0.2}Ge_{0.8}Te$, $Mn_{0.25}Ge_{0.75}Te$, $Mn_{0.3}Ge_{0.7}Te$, $Mn_{0.375}Ge_{0.625}Te$.

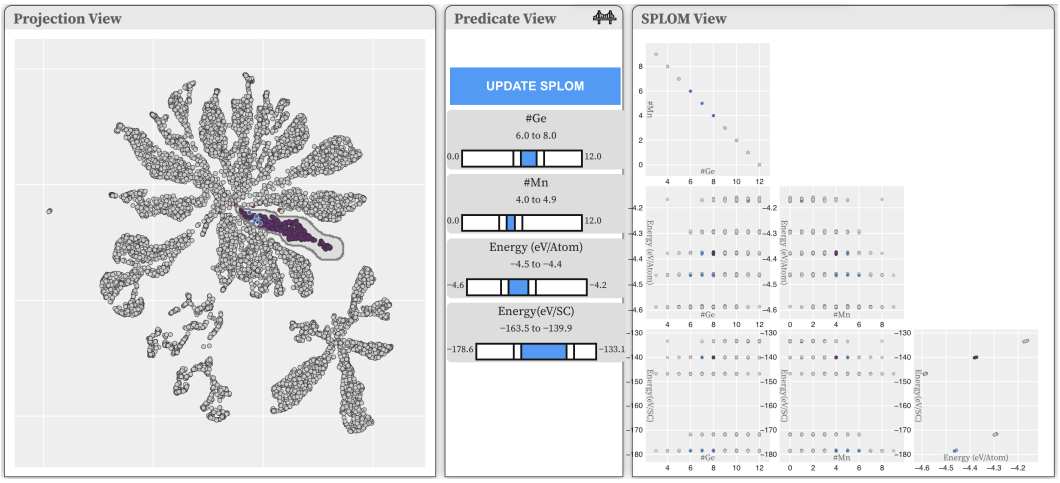


Figure 4.7: UI of DimBridge with cluster selected, showing the division of points into clean groups in the SPLOM. This indicates to our collaborator the strong influence Mn has on the clustering of data.

4.7.3 Phase one: Investigating Mn's Influence on Separation

Initial exploration involved a broad examination of all five subsets. The graduate student experimented with various hyperparameters to assess their impact on data separation, including adjusting and removing attributes to test the significance of the Mn coordination shell.

After removing attributes, she would recolor the projection to confirm expected separations, such as distinct clusters within each subset. She then brushed clusters of interest to check for correlations between attributes or to verify if the clusters were due to uninteresting factors. For example, some clusters were separated by the proportion of Mn and Ge, producing results that were not informative for her research, as shown in Figure 4.7.

4.7.4 Phase two: Investigating Temperature Dependencies

Her findings showed that the number of Mn atoms significantly influenced data separation in lower-dimensional spaces, prompting her to adjust her analysis to focus on the relationships between different attributes and her experimental data. She then investigated how temperature affects each alloy combination (Fig. 4.8), focusing on attributes related to the Pair Distribution Function (PDF) plots. These

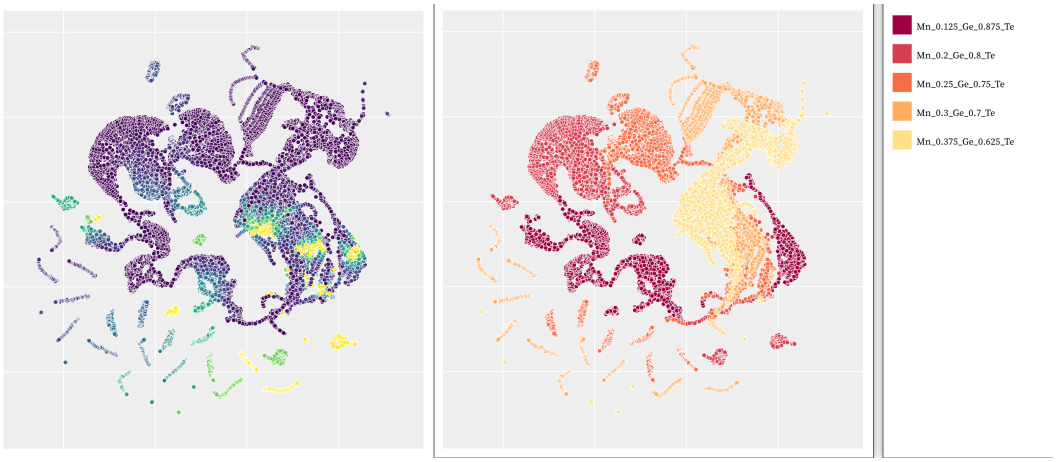


Figure 4.8: Projection view showing temperature dependencies color-coded by attributes. The left is color-coded by temperature and the right is color-coded by the data subset indicating the varying combinations of Mn and Ge.

plots helped her link local atomic structures, seen as peaks in the PDF, to the properties of various alloys. By examining the projection's top curve to observe value changes within interesting loop shapes, she confirmed that with increasing temperature, the data points tend to converge (Fig. 4.9-A).

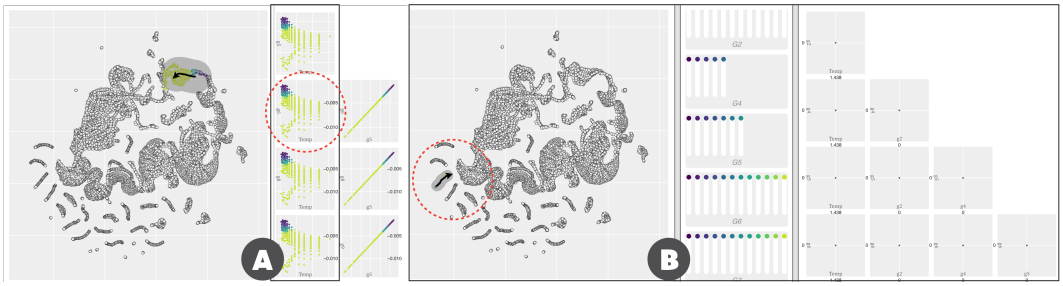


Figure 4.9: (A) DimBridge explains a trail within the alloy temperature prediction dataset. The SPLOM in the data view shows curves merging as the temperature increases. (B) The view of the data subset for the drawn shape selection indicates that the selected cluster contains only one temperature value.

She also made note of the small, isolated worm-like clusters outside of the larger shape. Drawing a shape over one of the small clusters, she saw that these small clusters are composed of a single temperature (Fig. 4.9-B). She noted, "something is definitely happening here! For the [$Mn_{0.3}Ge_{0.7}Te$] datasets, it looks like it's making worms grouped by temperature for a bunch of different POSCARs. I assume that's the case for the [$Mn_{0.25}Ge_{0.75}Te$], and [$Mn_{0.2}Ge_{0.}Te$], worms as well." Color coding the projection by combination, she confirmed her assumption on the data subsets.

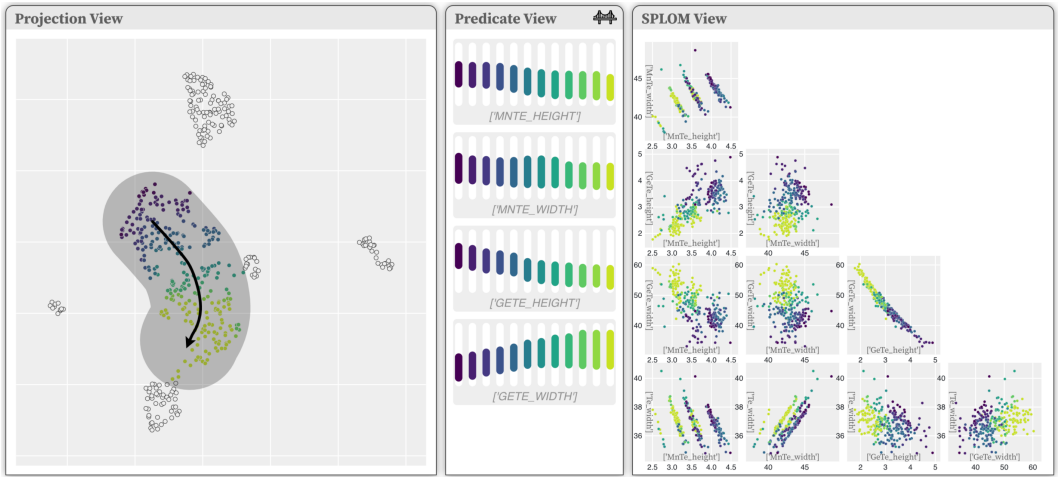


Figure 4.10: DimBridge views showing the attributes of the drawn shape selection of the larger cluster. In the SPLOM, you can see the correlation between Te height and Mn height.

4.7.5 Phase Three: Investigating Subsets of Interest

In the next stage of exploration, our collaborator created a dataset focusing on the outputs of her existing model rather than temperature dependencies, aligning it more closely with her experiments. This smaller, targeted dataset highlights subsets that previously formed unique, worm-like clusters. Building on her analysis suggesting Mn influences Te positions, she colored the projection by $MnGe_r$.

After identifying an interesting cluster, she examined its predicate ranges and potential relationships. Observing correlations, she drew a shape down the larger cluster (Fig. 4.10), making a key observation: the height and shape of MnTe peaks significantly impact Te peak locations and heights, showing a stronger correlation with MnTe peaks than with GeTe/Te peaks. This consistency suggests that manganese (Mn) plays a pivotal role in altering Te atom positions, while germanium (Ge) has little effect. "I really love seeing this because it's showing me how influential the Mn is in the PDF patterns."

Revisiting the questions posed in the previous section: **How does changing the proportion of manganese (Mn) affect the way the alloy's structure shifts from a slanted, diamond-like shape to a straight-edged, cube-like shape?** Our collaborator found that Mn significantly influences the alloy's structure, showing a clear correlation between the intensities of Mn-Te and Te-Te peaks

across the alloy space. This helps understand how Ge atoms cause local distortions, especially in low Mn concentrations, where Ge shifts the structure towards a more rhombohedral form. **Are the shapes formed by manganese (Mn) atoms similar across the alloy space?** In the initial exploration, projections showed clustering based on the number of Mn atoms in the coordination shell, regardless of composition. This suggests that the proximity of Mn atoms within their shells is more crucial than the overall elemental mix.

4.7.6 Observations

During our collaborator's interaction with DimBridge, we observed several notable aspects. Her process was highly iterative, moving from broad to focused analysis, often switching between datasets. The adaptability of DimBridge was a significant asset, allowing her to switch datasets, add and remove columns, and get immediate feedback on selections. The ability to dynamically update projections and data spaces enabled rapid iteration on hypotheses and analysis direction, even accounting for unexpected clusters defined by unusual attributes.

As well, our collaborator found DimBridge's functionality to draw a shape was extremely valuable for understanding changes in properties between one data subset to another, especially during the second phase of the analysis of temperature dependencies as seen in Figure 4.9-A.

4.8 Discussions and Limitations

DimBridge is designed to support the user in making sense of visual patterns in dimensionality reduction projections. In this section, we reflect on emergent insights, implications of DimBridge's value beyond its original design goals, limitations, and opportunities for future work.

4.8.1 DimBridge Design Considerations: Why SPLOM?

We use a SPLOM due to its efficacy for visualizing patterns in high dimensional space. This could involve the characterization of subspaces or the relationships between attribute values, which are both important tasks for our collaborators. Additionally, the SPLOM view allows a user to visualize the predicate value ranges of the relevant attributes in all possible pairings. For these reasons, we chose SPLOM over alternative high dimensional techniques. However, future work will explore other methods of visualization to make DimBridge more flexible for a wider variety of datasets and tasks.

4.8.2 The Value of Flexibility

Along with creating a flexible analysis environment, it is also important to consider how we can enhance DimBridge to be more adaptable in terms of its composition and features. Below, we outline several opportunities for doing so.

Beyond Projection Visualizations: Not all data and tasks benefit from the exploration of a projection. The projection view can be substituted for other methods of visualizing an overview or summary of the dataset, depending on the analysis tasks. Considering an example in the context of material science, imagine researchers are seeking to develop a new alloy with high strength and corrosion resistance for aerospace applications. They use a connectivity matrix to explore potential candidates, where the matrix includes a variety of known alloys, each characterized by their mechanical and chemical properties. Selecting cells or submatrices are entry points for understanding the properties of nearest neighbors of a selected cell, or defining ranges of desired properties to highlight relevant cells in the matrix. Or in a more general context, one could also imagine a scenario where the projection view might contain geospatial data, and patterns of location could be explored across relevant dimensions. Future work will explore the use of predicates as a bridge between high-dimensional data and alternative low-dimensional representations, beyond just projections.

Beyond Axis-Aligned Dimensions: DimBridge assumes the original (axis-aligned) data dimensions in the data-space visualization as they are the most interpretable. However, this design requirement can be lifted for advanced users who can understand complex data dimensions. Although we illustrated DimBridge using the original data dimensions in Section 4.6, we observe that the predicate induction could also have been performed using the principal components with the SPLOM showing the data with principal components as the data dimensions. The resulting SPLOM visualization will be less interpretable, but the use of principle components (and possibly other non-linear dimensions) can result in predicates that better fit the user-selected data.

Adapting Predicate Induction: DimBridge's predicate induction engine uses the RPI and Predicate Regression algorithms, representing two extremes of the accuracy/scalability tradeoff. Future work exploring predicate induction algorithms that take a more balanced approach to this tradeoff could be beneficial. Additionally, the predicate induction algorithm can be further tuned to the task of understanding DR results by explicitly accounting for distortions introduced by the DR algorithm. "Distortion aware" predicate induction could more effectively identify patterns in the original dimensions by adjusting for distortions in regions brushed by a user.

4.8.3 Limitations

The induction engine can generate predicates from any tabular dataset with continuous dimensions and is agnostic to the DR algorithm used for projection. However, we consider the following limitations:

textbfContinuous Dimensions: Both implemented predicate induction algorithms require continuous dimensions. Off-the-shelf feature-engineering techniques can be used to incorporate ordinal and nominal dimensions by numerically encoding them before using the predicate induction engine.

Meaningful Dimensions: Using the original data dimensions in the SPLOM for explanation assumes that these dimensions are semantically meaningful to the user.

For data dimensions that are intrinsically not meaningful (e.g., if each pixel of an image represents a dimension), additional processing to extract semantics, such as the use of CLIP [RKH+21] in Figure 3.2 can be effective for explanation.

Quality of DR Results: The quality of the induction engine's explanations depends on the quality of the DR projection. While similar points in the projection are expected to be similar in the original space, DR algorithms can introduce artifacts that disrupt this correspondence. If an artifact is mistaken for a pattern and selected, the induction engine will still return the best matching predicate, even if no good match exists. As discussed in section 4.5.4, DimBridge addresses this by highlighting selected data points as false positives or false negatives that the predicate fails to match. These situations should only occur if the DR projection produces misleading visual patterns.

Multiple Predicates: Our current implementation of the predicate induction engine returns only the top predicate for visualization by DimBridge. While a given pattern may have multiple plausible explanations in the original dimensions, addressing this involves considerations beyond this work, such as visualizing multiple predicates and maintaining interactivity. Exploring the potential to explain patterns using multiple predicates is an exciting avenue for future work.

4.9 Conclusion

In this chapter we present DimBridge, a system that *bridge*s a projection space with the original data space using first-order predicate logic. DimBridge connects patterns observed in the projection space to the original data space, helping users understand the pattern within the familiar data space. This decreases the likelihood of false discoveries resulting from spurious structure within the projection. DimBridge is agnostic to the projection algorithm, the visualization technique used within the data space, and the predicate induction algorithm itself. We illustrate three showcases of DimBridge within scientific data, motion-capture data, and imagery data. Finally, we evaluated the utility of DimBridge with a domain expert

who found the design to be helpful in her workflow.

Chapter 5

VIGOR: Graph Visualization Design Guidelines as Learnable Predicates

5.1 Problem Statement

Graphs are a powerful way to represent complex, interconnected data across domains ranging from biology [BSRG06] and social science [Zin10] to finance [DLM14] and cybersecurity [NHT+16]. Yet, while graphs themselves are flexible and expressive, effectively visualizing them remains a persistent challenge [LAB+23]. Practitioners often rely on node-link diagrams rather than visualizations that may be more effective for a given task [KEC06]. In the remainder of this paper, we use "graph" to mean multivariate graphs unless otherwise specified.

To move beyond default choices like node-link diagrams, researchers have explored scenarios in which alternative visualization techniques are more effective. Ghoniem et al. [GFC04] found that node-link diagrams are more effective for smaller, sparser graphs, while adjacency matrices are more effective for larger, denser graphs. Similarly, Wang et al. [WHW+24] found that chord diagrams are well-suited for graphs with a medium number of nodes (albeit loosely defined) and

moderate edge densities, providing a good balance between overview and readability. Nobre et al. [NMSL19] took a step toward consolidating these findings and graph visualization guidelines into recommendations for practical use. However, these recommendations are presented in a static table that is not easily extensible, and do not cover all visualization designs (e.g., chord diagrams).

The limitations of current approaches based on fragmented design guidelines pose challenges for both researchers and practitioners. For researchers, the lack of a shared representation means that empirical findings are rarely incorporated into a common corpus, leaving valuable results isolated rather than building toward cumulative knowledge. For designers, the lack of a common foundation makes graph visualization design difficult. Current guidelines do not capture factors such as domain standards, preferences, and graphs with varying complexity and statistical properties. Moreover, it is often unclear whether a guideline derived for purely structural graphs generalizes to richer settings, such as multivariate graphs, weighted graphs, or multi-graphs. Without a formal foundation, existing design knowledge remains difficult to apply consistently and even harder to extend, leaving research and practice disjointed.

We make two contributions to bridge this gap: (i) a predicate-based representation of graph visualization design guidelines, and (ii) an algorithm for learning these predicates from data. Predicates mirror the structure of existing empirical rules and guidelines (see section 5.2 and section 5.4 for a more detailed explanation). To derive such rules, we propose a predicate induction algorithm that learns from a labeled dataset, where each graph is represented as a vector of graph statistics and the label indicates the appropriate visualization type, based either on established guidelines or user preferences. This algorithm not only learns but can also optimize and adapt predicates as new data or user feedback becomes available. This predicate-based representation captures empirical findings as explicit, human-readable rules that can be inspected, compared, and accumulated across studies. It also supports systematic refinement, since predicates can be evaluated, optimized, and updated as new evidence or feedback becomes available. Finally, the approach

remains extensible, allowing new rules, graph features, and visualization types to be incorporated as the field evolves.

We evaluate our predicate-based representation and induction algorithm through two experiments, each corresponding to the usage scenarios introduced in section 5.5. First, we test the system's ability to recover known rules by examining whether predicates learned from a labeled dataset reproduce expert-defined guidelines from the literature. Following Nobre et al.'s reference table [NMSL19], we label synthetically generated graphs according to their most appropriate visualization type and evaluate how closely the recovered predicates match these expert rules. Second, we evaluate personalization and recommendation by simulating different user preference profiles and testing how well the induced predicates generalize to held-out graphs. We generate labels that vary in their adherence to literature-derived guidelines and assess whether the learned predicates both adapt to these preferences and correctly recommend visualizations. Across both experiments, our approach successfully reconstructs expert rules, adapts to diverse usage patterns, and generalizes well on held-out data, demonstrating the potential of predicates as a foundation for accumulating and operationalizing graph visualization design knowledge.

5.2 Design Guidelines as Predicates

Rule-based and machine-learning approaches have both advanced automated visualization design, but each is limited for graph data: rule-based systems are interpretable yet brittle and hard to extend, while machine-learning systems adapt flexibly but often act as opaque black boxes. We address this gap by expressing graph visualization guidelines as bounded, interpretable predicate conditions over graph statistics. This section focuses on the representational role of predicates and how they consolidate design knowledge. section 5.3 then describes how they can be learned from labeled data.

5.2.1 Background: Predicates

A predicate is a logical condition that evaluates to `true` or `false` for a given object. Such conditions are typically expressed as functions that take an input and return a Boolean value; i.e., for a number x, IsEven(x) returns `true` if x is even and `false` otherwise.

In our setting, predicates are built from clauses over graph statistics, where each statistic can be *continuous*, *binary*, or *categorical*. A clause constrains a given graph statistic x according to its type:

- **Continuous**: an interval constraint that evaluates to `true` for a graph G when $a < x < b$, and to `false` otherwise, where $a \in \mathbb{R}$ and $b \in \mathbb{R}$ are the parameters of the predicate.
- **Binary**: an equality constraint that evaluates to `true` when $x = a$, and `false` otherwise, where $a \in \{0, 1\}$ is the parameter of the predicate.
- **Categorical**: a membership constraint that evaluates to `true` if $x \in a$ where a is a set of values and the parameter of the predicate.

For example, a continuous clause might specify that the graph's density lies within a given interval ($\textit{density} \in [0, 0.1]$). A binary clause could state that the graph is directed ($\textit{is-directed} = 1$). A categorical clause might require that the graph type belongs to a subset of categories (*graph-type=tree*). A predicate is then the conjunction of one or more such clauses, and it evaluates to true if and only if all of its clauses evaluate to true. For instance, $(\textit{density} \in [0, 0.1] \land \textit{is-directed} = 1 \land \textit{graph-type=tree})$, which evaluates to true only for graphs that are simultaneously sparse, directed, and a tree, in other words, a hierarchy.

This abstraction provides a natural way to capture design guidelines. Guidelines can be understood as statements about which kinds of graphs call for which design choices, and predicates supply the formal structure to express such statements in a clear and extensible manner. By encoding guidelines as explicit conditions over graph statistics, predicates can be evaluated to determine which guideline

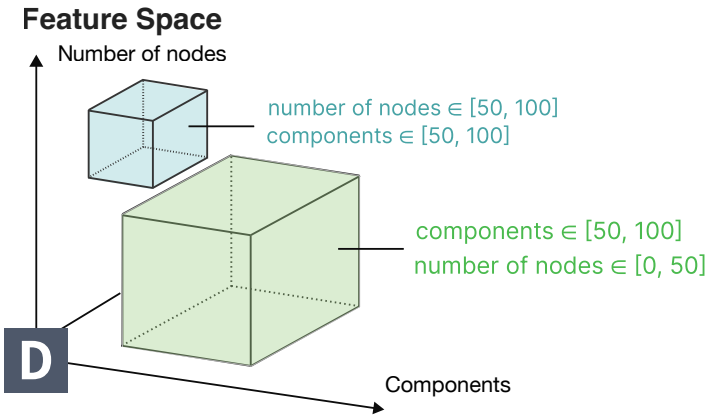


Figure 5.1: From predicates to feature space: each box corresponds to a bounded region of graph statistics that encodes a visualization guideline.

best fits a given graph. At the same time, the representation is modular, since new guidelines or visualization techniques can be incorporated by adding or adjusting predicate clauses rather than reworking the entire rule set. Finally, predicates preserve transparency, since each outcome can be traced back to explicit conditions, allowing researchers and practitioners to inspect and refine the reasoning behind design choices.

5.2.2 Graphs and Predicates in Feature Space

Each graph G can be described by a vector of statistics $s = (s_1, s_2, \ldots, s_M)$, where each s_j represents a graph property such as its density, number of nodes, or clustering coefficient. This vector embeds G as a point in $\mathbb{R}^M$, which we call the *feature space* of graphs. The feature space provides a common coordinate system in which graphs can be compared, forming the basis for expressing design guidelines as predicates and for learning and refining them from data.

In this space, a predicate clause restricts one coordinate of the feature vector to an allowed set of values, while a conjunction of predicate clauses defines an axis-aligned region of the space. Visualization guidelines can then be expressed as such regions, since they specify conditions on which kinds of graphs are suited to which design choices. For example, when considering modularity and clustering coefficient,

each graph is represented as a point in the two-dimensional plane. The guideline "*use NodeTrix for highly clustered and modular graphs*" then corresponds to a rectangle bounded by (modularity $\in [0.6, 1.0] \wedge$ clustering coefficient $\in [0.5, 1.0]$).

More elaborate guidelines can be expressed as unions or intersections of such regions, capturing conditions that overlap or exclude each other. Representing predicates in feature space makes the structure of design knowledge explicit. Guidelines that overlap correspond to alternative but compatible design choices, while disjoint regions signal potential conflicts. This geometric view also supports consolidation, because rules derived from different studies can be compared by examining how their regions align, and newly proposed rules can be integrated by adding regions to the space.

At the same time, the rigidity of hard cut-offs becomes apparent in this representation. Two graphs with nearly identical properties may fall on opposite sides of a boundary. For example, a hierarchical graph with 10 layers may satisfy a sunburst guideline, while one with 11 layers does not, even though they are nearly indistinguishable in practice. Such discontinuities underscore the limitations of hard, static boundaries and motivate a probabilistic extension. In section 5.3, we build on the feature-space representation to relax these hard cut-offs, allowing predicate boundaries to be learned and adapted from data.

5.3 Predicate Induction Algorithm

While the predicate-based representation from the previous section makes design guidelines interpretable, manually curating predicates as design rules is insufficient. The landscape of graph visualization evolves with new findings, diverse datasets, and shifting user preferences. To remain useful, predicates must therefore be *induced* from data rather than only prescribed in advance. Predicate induction addresses this by making the axis-aligned regions learnable.

Learning Predicate Regions: Instead of fixing regions a priori, we optimize them to align with labeled training examples. Building on the approach of Montambault

et al. [MAR+24], we formalize predicate induction through two components: a probability function, which evaluates how well a graph satisfies a candidate predicate, and a loss function, which guides the adjustment of predicate intervals. Together, these elements define an optimization procedure that refines predicates into accurate and generalizable design rules.

The Probability Function: For each visualization design guideline v, we define a candidate predicate Φ_v as a conjunction of bounded conditions over graph statistics. Formally, let $\Phi_v = (\phi_1, \phi_2, \ldots, \phi_M)$ be the parameters of learnable predicate clauses for M graph statistics. Each clause $\phi_j := (\mu_j, r_j)$ specifies a learnable region for the j-th statistic. We write $s_i = (s_{i1}, s_{i2}, \ldots, s_{iM})$ for the statistics vector of a given graph i, with s_{ij} denoting its value on the j-th statistic. The induction algorithm then evaluates how well s_i satisfies Φ_v through a probability function:

$$\Pr(\Phi_v, s_i) := \frac{1}{1 + \sum_{j=1}^{M} \mathcal{L}(\phi_j, s_{ij})} \tag{5.1}$$

Loss Functions by Data Type: The loss function $\mathcal{L}(\phi_j, s_{ij})$ specifies the mismatch between the value s_{ij} and the clause ϕ_j. We define a loss function for each data type represented in the graph statistics: continuous, binary, and categorical. For continuous statistics, e.g., *number of nodes or edges*, we define a differentiable bump function:

$$\mathcal{L}_{continuous}(\phi_j, s_{ij}) = |\frac{1}{r_j} \cdot (s_{ij} - \mu_j)|^b$$

where b controls the steepness and roundness of the bump (empirically, we set $b = 3$). This function gives low loss (high probability) to values near the midpoint and penalizes those farther away.

For binary statistics, e.g., graph statistics such as *is-directed* or *is-multi-graph*, we denote the learnable parameter, sensitivity for the binary variable, as $\phi_j := (w_j)$. The loss is defined as:

$$\mathcal{L}_{binary}(\phi_j, s_{ij}) = \exp(-s_{ij} \cdot w_j)$$

where $s_{ij} \in \{-1, 1\}$ is the binary value, and w_j is a learned weight. A positive w_j favors $s_{ij} = 1$ in that it reduces the loss, while a negative w_j favors $s_{ij} = -1$.

For categorical statistics, e.g., *graph-type* $\in$ [Tree, Sparse], the learnable parameters for the clause is list of $\phi_j := (w_{j,c_1}, w_{j,c_2}, \dots)$. We define loss as the average binary loss over all categorical values:

$$\mathcal{L}_{categorical}(\phi_j, s_{ij}) = \exp(-\sum_{c \in C_j} s_{ij,c} \cdot w_j)$$

where C_j is the set of all possible values of the j-th statistic, and $|C_j|$ its cardinality. $w_{j,c}$ is a learned weight for a given value c in C_j. $s_{ij,c}$ is used to denote a binary value, where $s_{ij,c} = 1$ if $s_{ij} = c$ and $s_{ij,c} = -1$ otherwise. In this way, categorical variables are treated similarly to numbers, preserving consistency in our probabilistic framework.

Optimization Objective: Let S denote the set of graph statistics vectors over a set of graph instances. Y_v is a vector of length $|S|$ representing the labels of a visualization design v (e.g., v = node-link) such that $y_{v,i}$ is a binary value denoting if graph s_i is labeled as v. Given the graphs S and the labels Y_v, we compute an optimal predicate for the visualization v by minimizing the binary cross entropy:

$$\begin{aligned}\mathcal{L}(\Phi_v \mid S, Y_v) = \frac{1}{N} \sum_{i=1}^{N} & y_{v,i} \log(\Pr(\Phi_v, s_{ij})) \\ & + (1 - y_{v,i}) \log\big(1 - \Pr(\Phi_v, s_{ij})\big)\end{aligned} \tag{5.2}$$

This optimization adjusts the centers and ranges of each predicate's intervals in continuous graph statistics, as well as their sensitivity in binary and categorical cases, to ensure that the probability scores align with the labeled examples. Each optimized region in feature space then corresponds to a learned visualization guideline. For instance, if the training data consistently labels sparse graphs with fewer than 200 nodes as "node-link," the induction algorithm will converge on a predicate approximating (*density* $\in [0, 0.1] \wedge$ *node count* $\in [0, 200]$).

By learning predicates from labeled examples, our approach moves beyond

fixed, hand-crafted rules. The induction algorithm produces guidelines that remain interpretable as ranges on graph statistics, but that also adapt to new evidence, user preferences, or domain-specific contexts. This flexibility makes predicates both a stable representation of guidelines and a mechanism for continuous refinement.

5.4 Predicates Applied to Graph Visualization

We now apply predicates to graph visualization by defining the graph statistics that form the input space and examining how predicate-based guidelines behave on graphs. This illustrates how the earlier abstract representation becomes concrete in practice.

5.4.1 Input Space

To determine graph statistics that comprise the input space, we began with the 168 references compiled by Nobre et al. [NMSL19] and extended this corpus with papers published in the past five years in TVCG and CHI. Our initial search yielded 1,680 papers in TVCG and 2,200 in CHI containing the keyword "graph." From this set, we retained only those that explicitly articulated design guidelines or employed a graph statistic as a central element in the design or evaluation of a visualization technique. After applying these criteria, the corpus comprised 193 papers. From this refined set, we systematically extracted the graph statistics identified by authors as influential for visualization design and effectiveness. We also incorporated a small set of additional graph metrics based on the authors' domain knowledge, ensuring coverage of statistics commonly used in practice but underrepresented in the surveyed literature.

The resulting input space (see Table 5.1) reflects the statistics most often used to inform visualization choices in the literature and in practice. For clarity, we organize these statistics into four broad categories that frequently recur in the literature: general properties, connectivity measures, cohesion measures, and elements. These statistics serve as the features over which our induction algorithm

learns predicate regions, enabling guidelines to be expressed, refined, and compared systematically.

Category	Measure	Data Type
General	Graph type	`[Tree, Cycle, ...]`
	Directed	`Boolean`
	Spatial	`Boolean`
	Planar	`Boolean`
	Hypergraph	`Boolean`
	Layer count	`Numeric (integer)`
Connectivity	Self-loops	`Numeric (integer)`
	Parallel edges	`Numeric (integer)`
	Size (nodes and edges)	`Numeric (integer)`
	Components	`Numeric (integer)`
	Connectivity / paths	`Numeric (float)`
	Diameter	`Numeric (integer)`
	Average path length	`Numeric (float)`
Cohesion	Cut ratio	`Numeric (0, 1)`
	Separability	`Numeric (0, 1)`
	Density	`Numeric (0, 1)`
	Cluster density	`Numeric (0, 1)`
	Clustering coefficient	`Numeric (0, 1)`
	Triangle count	`Numeric (float)`
	Modularity	`Numeric (0, 1)`
	Average degree	`Numeric (float)`
	Centrality measures	`Numeric (float)`
	Communities	`Numeric (integer)`
	Clusters	`Numeric (integer)`
Elements	Node/Edge types	`Numeric (integer)`
	Node/Edge attributes	`Numeric (integer)`

Table 5.1: Graph statistics that define the input feature space.

5.4.2 Uncertainty and Trade-offs in Guidelines

Visualization design guidelines are rarely absolute. Graphs often fall near the boundaries of statistical ranges, and different guidelines may overlap or even conflict. Moreover, empirical studies frequently disagree on the exact thresholds, underscoring the need for a representation that tolerates uncertainty.

For example, consider three graphs with density values of 0.05, 0.1, and 0.15. A deterministic predicate such as $\textit{density} \in [0, 0.1]$ applied to these three groups would evaluate to true, true, and false, respectively. This draws a hard boundary between those inside and outside the range, but misses important nuance. The first graph lies in the center of the interval and strongly satisfies the guideline, the second sits right on the boundary, and the third is just outside the range and often nearly

indistinguishable from the former in practice (note: the range of *density* is from 0 to 1). By interpreting predicate membership probabilistically (see Equation 5.1), these cases are differentiated: the first graph receives the highest probability, the second moderate, and the third minimal, but non-zero, probability, despite having a graph statistics that lie outside of the predicate's interval.

Applied to graph visualization, this treatment of uncertainty has several implications. First, it clarifies trade-offs in overlapping regions of the feature space where a single graph meets the predicates of more than one visualization guideline. That is, when its statistics satisfy multiple conjunctive conditions simultaneously, making it eligible for several visualization types at once. Second, it reduces the brittleness of fixed thresholds. For example, values in guidelines like "*greater than 200 nodes*" or "*density below 0.1*" become zones of gradual transition rather than hard cut-offs. Third, it provides a basis for comparing guidelines across studies, as overlapping regions reveal consensus while divergences highlight conflicts or gaps. Finally, it creates opportunities for adaptive or personalized visualization. When multiple guidelines receive partial support, their relative probabilities can guide recommendations while keeping conditions interpretable. In this way, predicates encode graph visualization guidelines while also revealing their uncertainty, overlaps, and points of contention.

5.5 Usage Scenarios

We now illustrate how predicates can be used in practice through two usage scenarios: recovering implicit rules from labeled data and adapting predicates to user preferences. section 5.6 evaluates these scenarios by testing rule recovery and assessing how well learned predicates adapt to user preferences and generalize to unseen graphs.

5.5.1 Rule Recovery

A major advantage of the predicate-based representation is its ability to recover design rules directly from labeled data rather than relying on manually encoded

guidelines, as in systems like Draco [MWN+18]. Given a dataset of graphs labeled with their designated visualization type, each graph is mapped to a feature vector of statistics, and the learning process induces a predicate for each visualization. Each clause specifies a constraint on a statistic, and optimization adjusts these parameters so that $\Pr(\Phi_v, s_i)$ (Equation 5.1) aligns with the observed labels, converging on regions of the feature space that best separate visualization choices. For example, if graphs with high modularity and clustering coefficient are consistently visualized with NodeTrix diagrams, the learning process would recover a predicate such as *modularity* $\in$ *[0.6,1.0]* $\wedge$ *clustering coefficient* $\in$ *[0.5,1.0]*, capturing conditions under which hybrid visualizations are preferred.

5.5.2 Personalization and Recommendation

Predicates not only recover rules from data but also support personalization and recommendation. Because each guideline is expressed as a conjunction of clauses $\Phi_v = (\phi_1, \ldots, \phi_M)$, user or domain preferences can be incorporated by adjusting clause intervals: positive feedback expands relevant ranges, while negative feedback contracts them. This yields interpretable refinements, in contrast to prior personalized systems [GZ09], [MVT16], [OYC15], [QRD+22], and allows separate predicate sets $\{\Phi_v\}$ to reflect different user groups or contexts. The same structure enables visualization recommendation for new graphs. Mapping a graph G to a feature vector s and evaluating $\Pr(\Phi_v, s)$ produces a probability distribution over visualization types that reflects how well the graph satisfies each guideline. Graphs deep inside a predicate's region receive strong support, while those near boundaries may partially satisfy several guidelines, making trade-offs explicit. Because recommendations derive directly from clause contributions (e.g., size, density, clustering), they remain fully interpretable. Treating personalization and recommendation as two uses of the same probabilistic evaluation unifies user adaptation and design guidance in a transparent, extensible model.

5.6 Evaluation

We evaluate our predicate representation and induction algorithm through two structured experiments:

1. **Rule recovery** – testing whether the induction algorithm can reconstruct known guidelines when only labeled data is provided.

2. **Personalization and recommendation** – evaluating how predicates adapt to user-specific preferences and how well these personalized predicates generalize to unseen graphs.

Evaluation Data We generated a dataset of 1200 sample graphs, consisting of 80% sparse and dense graphs, 10% cycles, and 10% trees. Graph sizes range from 2 to 200 nodes, and include both simple and multivariate graphs. Graph statistics were extracted for all graphs to produce feature vectors. We use 1000 graphs for training (rule recovery and personalization) and reserve 200 graphs for evaluating generalization.

5.6.1 Recovering Rules from Labeled Data

This experiment corresponds to the *rule recovery* scenario. Here, the induction algorithm is trained solely on labeled graph–visualization pairs, without being given any predefined predicates or expert rules. The aim is to test whether the algorithm can infer meaningful predicates directly from the data, mimicking a situation where only empirical evidence is available. If the learned predicates match the original ones used to generate the labels, we can conclude that the induction algorithm has successfully recovered the underlying design rules.

Generating Ground-truth: To establish an impartial ground-truth dataset that maps each generated graph to a visualization recommendation, we refer to Nobre et al. [NMSL19]'s reference table. In their scheme, a score of zero indicates no support and a score of three indicates strong support. To make these recommendations operational, we expressed each "full support" condition (score = 3) as a predicate

clause over the corresponding statistic. When multiple statistics were mentioned together, we combined them conjunctively into a single predicate. Applying these predicates to our generated graphs allowed us to assign each graph the visualization type it best supports. This procedure produced the ground-truth labels for our feature vectors. Notably, their guidelines seldom, if ever, recommend the use of adjacency matrices, treemaps, and sunbursts. The predicates are shown in Table 5.2.

Visualization	**Predicate**
Node-Link	`n_nodes` $\in [0, 100]$, `graph_type` $\in [sparse, k-partite, tree]$, `node_types` $\in [1, 1]$, and `edge_types` $\in [1, 1]$
Attribute-driven positioning	`n_nodes` $\in [0, 100]$, `graph_type` $\in [sparse, k-partite]$, `node_attributes` $\in [0, 5]$, and `node_types` $\in [1, 5]$
Attribute-driven faceting	`n_nodes` $\in [0, 100]$, `graph_type` $\in [sparse]$, `node_attributes` $\in [0, 5]$, and `node_types` $\in [1, 1]$
Adjacency Matrix	`n_nodes` $\in [0, 100]$, `graph_type` $\in [dense]$, `node_attributes` $\in [5, 10]$, `node_types` $\in [1, 1]$, `edge_attributes` $\in [0, 3]$, and `edge_types` $\in [1, 1]$
Quilts	`n_nodes` $\in [0, 100]$, `graph_type` $\in [sparse, k-partite, tree]$, `node_attributes` $\in [0, 10]$, `node_types` $\in [1, 5]$, `edge_attributes` $\in [0, 10]$, and `edge_types` $\in [1, 1]$
BioFabric	`n_nodes` $\in [0, 100]$, `graph_type` $\in [sparse, dense]$, `node_attributes` $\in [0, 10]$, `node_types` $\in [1, 5]$, `edge_attributes` $\in [0, 10]$, and `edge_types` $\in [1, 5]$
Treemap	`graph_type` $\in [sparse, tree]$, `node_attributes` $\in [0, 5]$, and `node_types` $\in [1, 1]$
Sunburst	`graph_type` $\in [sparse, tree]$, `node_attributes` $\in [0, 5]$, and `node_types` $\in [1, 1]$

Table 5.2: Nobre et al.'s multivariate graph visualization design guidelines [NMSL19] converted to predicates.

Reconstructed Predicates To assess the ability to recreate the guidelines as predicates, we compare the learned predicates with the Nobre et al. [NMSL19] guidelines by evaluating the degree of overlap for each predicate clause, measured using their intersection over union (IOU). The values of IOU range from 0 to 1, where 0 represents low accuracy where there is no overlap between the ground-truth and the learned predicates, and 1 means high accuracy with complete overlap between the two. Average IOU for each visualization (Table 5.3) indicate that, in most cases, the predicates learned closely match the ground truth. The predicates defined for the adjacency matrix and sunburst visualization types could not be recovered, since Nobre et al.'s guidelines never recommended these techniques when only considering scores of three, and therefore they did not appear in the labeled dataset. Overall, the general match between the ground-truth and the learned predicates demonstrates that the induction algorithm can induce visualization guidelines from data.

Visualization	Average IOU (higher better)
Node-Link	0.83
Attribute-driven positioning	0.55
Attribute-driven faceting	0.79
Quilts	0.65
BioFabric	0.31
TreeMap	0.66

Table 5.3: Average IOU for each visualization defined by Nobre et al. [NMSL19].

5.6.2 Personalization and Recommendation

This experiment corresponds to the *personalization and recommendation* scenario. Here, we evaluate whether the system can adapt to user-specific preferences and generalize the learned predicate boundaries to unseen graphs. We test whether the algorithm can flexibly adapt when users deviate from given rules. By simulating different adherence levels, we assess the robustness of the induction to noisy or inconsistent feedback.

We base these labels on a set of predicates derived from prior literature, which captures common guidelines. While not exhaustive, this collection spans the major visualization types studied in the literature and provides a representative baseline of design rules for our evaluation. To simulate different behaviors, we generate three user profiles: an *informed user* who follows expert predicates consistently, a *semi-informed user* who deviates 25% of the time, and an *uninformed user* who deviates 50% of the time. These profiles produce three versions of training labels. Running predicate induction on each labeled dataset produces one learned predicate for every visualization type. For a given graph, the algorithm assigns a score to each visualization type based on how well the graph satisfies its corresponding predicate. The visualization type with the highest score is recommended as the label. If two types receive the same score, we break the tie by choosing the type that appeared more frequently in the training labels. After inducing predicates from each user's labeled data, we evaluate recommendation performance on the 200 held-out graphs. A prediction is considered correct when the visualization with the highest probability

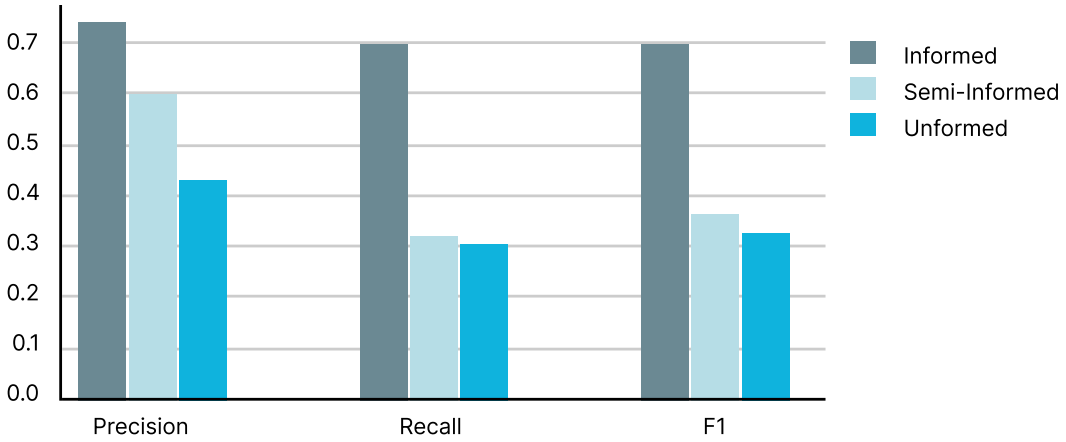


Figure 5.2: Generalization performance across user types.

matches the simulated user's label.

As shown in Figure 5.2, predicates learned from consistent user behavior generalize well to unseen data, achieving an F1-score of 0.70. Performance declines as user behavior becomes noisier, but the system still captures broad patterns in preference and applies them to new graphs. These results highlight that personalization and recommendation emerge naturally from the same predicate-based mechanism.

5.7 Discussion

While the predicate-based representation and the mechanism for learning predicates provide a foundation for consolidating graph visualization design knowledge, several limitations highlight opportunities for future work. At the same time, these contributions open pathways toward more systematic, interpretable, and cumulative progress in graph visualization research and practice.

of the Predicate Representation: In the present formulation, each predicate is treated as an independent, conjunctive rule defining a single bounded region of the feature space. This structure improves interpretability but limits expressiveness. Dependencies between conditions, such as density thresholds that differ for directed versus undirected graphs, cannot be modeled. Likewise, many visualization types may be appropriate under multiple disjoint conditions, which a single predicate region cannot capture. Extending the representation to support conditional

or disjunctive rules would make the model more expressive. Additionally, our approach maps graphs to visualization types, maintaining tractability but omitting encoding-level considerations such as layout aesthetics, perceptual constraints, and interaction techniques. Extending predicates to capture these lower-level guidelines would expand their applicability and better support design decisions. Applying this approach to tabular data, is more difficult, since tabular visualizations often rely on transformations like grouping or binning, which cannot be reduced to simple bounded conditions.

Integrating Expert Knowledge with Learned Rules: The current system focuses on learning predicates from labeled data. Incorporating expert-derived guidelines as priors, then refining them through feedback, would combine the stability of established knowledge with the adaptability of data-driven refinement. Such integration would support cumulative knowledge building rather than treating expert rules and learned rules as separate resources.

Balancing Multiple Valid Explanations: For many visualization decisions, more than one predicate may offer a valid explanation. The present approach converges on a single predicate per visualization type, simplifying outcomes but not fully reflecting the diversity of possible design rationales. Future work could incorporate weighting or complexity penalties to balance multiple competing explanations, favor simpler predicates, or highlight alternative interpretations when several explanations hold.

Toward Systematic and Cumulative Design Knowledge: A key challenge for the field is the absence of systematic methods to compare graph-visualization techniques, assess their effectiveness across graph types and tasks, or determine when new designs are warranted. A shared repository of graph–visualization mappings would support synthesis, reproducibility, and large-scale learning, while clarifying the relative importance of the graph statistics used in practice. Identifying which features best predict visualization effectiveness would advance evidence-based, generalizable guidelines. By expressing such guidelines as predicates and refining them through learned adjustments, this work provides a structured framework that unifies

interpretability, adaptability, and transparency, which helps consolidate dispersed design knowledge into a cumulative, coherent foundation.

5.8 Conclusion

This paper offers two contributions: a predicate-based representation that formalizes visualization guidelines as bounded, interpretable conditions over graph statistics, and a learning mechanism that refines these predicates from labeled data or user feedback. Together, these components enable systematic comparison, extension, and consolidation of design knowledge. Looking forward, we will expand the predicate space to include additional statistics, visualization types, and conditional logic. With such advances, predicates could evolve into a unifying resource that supports both visualization research and evidence-based practice.

Chapter 6

A Predicate Framework for Visual Analytics

In the previous chapters, I discussed three systems designed for three separate analysis use-cases. While these use-cases have little in common on their surface, our solutions converged on VA systems with predicate induction as their primary component for automated analysis. I have argued in this thesis that, in general, predicate induction as an automated analysis technique within a VA system can serve as a bridge between human and automated reasoning, facilitating the iterative sense-making loop that characterizes effective VA, and is necessary for applying data-driven decision making to complex, ill-defined, and data-intensive use-cases. As the scope of target use-cases continues to expand (i.e., there is increasing interest in data-driven decision making across diverse use-cases), flexible tools for building VA systems become increasingly important. For systems based on predicate induction for VA, this means tools that facilitate designers and domain experts in applying this approach to a given use-case. In this chapter, I propose a formal framework for predicate induction in VA - a tool that can be used designers and domain experts alike to assess the applicability of this approach for novel use-cases, evaluate similarities with existing systems, and specify appropriate technical roles for the induction engine, visual interface, and user interactions.

This Chapter is structured into two section. In the first section, I reflect on successes and challenges encountered in implementing predicate induction for VA in our three use-cases - anomaly reasoning (AR), explainable dimensionality reduction (XDR) and graph visualization recommendation (GVR) - and discuss the general properties of predicate induction that make it a powerful tool for VA in certain use-cases. I begin by defining predicate induction formally, highlighting the logical structure of conjunctive first-order predicates and the use-case specific objective function as its core components. I discuss the general characteristics of the induction algorithms implemented for our use-cases, and compare them to similar algorithms from the literature, including those that may be used as alternative methods for predicate induction and those that, while producing similar results, are insufficient for predicate induction. Then, I return to the three use-cases discussed in this thesis, identifying two general requirements that characterize appropriate applications for predicate induction: 1) identifying relevant subsets within a dataset, and 2) providing meaningful explanations. Finally, I discuss the limitations of predicate induction in VA, highlighting the inherent tradeoff between accuracy, interpretability, and speed. I discuss how these limitations can be managed through the design of the induction engine and visual interface.

In the second section, I present a formal framework that provides a scaffolding for defining the structure and role of the induction engine, visual interface, and user for a given system. I begin by motivating a formal framework as a useful tool for applying predicate induction to VA, highlighting the need to a) determine if predicate induction is appropriate for a given use-case, b) identify similar use-cases and systems, and c) specify roles for the induction engine, visual interface, and user that, given the nuances of a particular use-case, leverage the strengths of predicate induction while managing its limitations. Then, I present the framework itself, formally defining the role of the induction engine, visual interface, and user in identifying relevant subsets and meaningful explanations that generalizes across use-cases. I define the systems we designed for our three use-cases - PIXAL, DimBridge, and VIGOR - within this framework, allowing for a formal comparison despite the

apparent dissimilarity across use-cases. Finally, I discuss future work that is needed in evaluating this framework, suggesting how we might measure our success in a) evaluating existing systems, and b) applying predicate induction for VA to novel use-cases.

6.1 Why Predicate Induction?

Throughout this thesis, I describe how predicate induction helps facilitate the sense-making loop characteristic of VA systems by bridging human and automated reasoning. In this section, I reflect on the the particular benefits and limitations of predicate induction for VA in practice. To begin to answer "why predicate induction?", I start by answering "what is predicate induction?", first defining the basis of predicate induction - predicates themselves - as logical abstractions that reference concrete subsets of a dataset, then defining the use-case specific objective function used to automatically generate relevant predicates. I further define predicate induction in terms of the algorithms we implemented in our use-cases, comparing to similar algorithms to underscore what constitutes predicate induction and what does not.

I then move on to the more practical question, "what use-cases benefit from predicate induction?", revisiting the three seemingly unrelated use-cases presented in previous chapters to identify generalizable characteristics. I find that the underlying analysis goal of all three use-cases can be conceptualized as identifying relevant, concrete subsets and assigning them meaningful, abstract explanations. While the notion of relevance and meaningfulness vary significantly across use-cases, I demonstrate the flexibility of predicate induction as an approach, allowing designers to define an appropriate objective function that captures relevance and logical structure capturing meaning.

Then, I discuss the drawbacks of predicate induction, highlighting potential limitations to accuracy, interpretability, and speed. While conjunctive predicates are generally straightforward to interpret, they are limited to representing subsets as

axis-aligned bounding boxes. This is often insufficient in practice for capturing the complexity of patterns that actually exist in the data. Furthermore, while predicate induction allows for flexibility in defining an objective function, it is limited to representing relevance as a single value. In practice, we find that the relevance of a subset is often multi-faceted or ill-defined, and cannot be sufficiently represented by a single numeric value. While accuracy limitations can be mitigated by allowing for more complex logical structures, I demonstrate the inherent tradeoff between with interpretability and speed.

Finally, I discuss how these limitations can be managed through the careful design of the induction engine and visual interface. For the induction engine, I discuss how limitations can be managed by generating a greater number, more precise, or more logically complex predicates, and by including data pre-processing steps such as dimensionality reduction or subspace sampling. For the visual interface, I discuss how accuracy limitations can be managed by providing users with rich visual representations of subsets that facilitate a deeper evaluation of relevance than is assumed by the induction engine's one-dimensional objective function. I discuss how interpretability limitations can be similarly managed by contextualizing the logical structure of predicates. Visualization is a flexible tool for both scenarios, providing context through the representation of a predicate's subset and logical structure, or through comparison with others. Moreover, interaction allows users to define predicates directly, allowing for the exploration and verification of those generated by the induction engine, those represented concepts present in the users background knowledge but not the data itself, and more expressive logical structures to capture more complex problems.

6.1.1 What is Predicate Induction?

To define predicate induction, I begin by defining its core representation, the predicate, in the context of data analysis. In general, a predicate refers to a logical statement that resolves to either "true" or "false" when applied to an individual object. A predicate is first-order if these objects are not predicates themselves. In

this work, we use "predicate" to refer to first-order predicates where objects are a data point, or row, from a dataset consisting of n rows and m columns. Furthermore, this work primarily deals with *conjunctive* predicates, which are composed of a set of conditions, and resolve to "true" only if all conditions resolve to "true". In this context, a predicate defines both a subset of the data (all rows that resolve to "true") and a concise description of that subset in terms of one or more of its m columns.

6.1.1.1 Predicates

What is a predicate in the context of VA, and what makes it a useful representation? Formally, a predicate Q defines a subset of a dataset D, where D consists of n_D rows and m_D attributes, denoted $Attr_D$. The predicate Q is composed of a set of $m_Q \leq m_D$ conditions, each defined over a subset of attributes denoted $Attr_Q \subseteq Attr_D$. q_{attr} denotes a condition defined by a particular attribute, and defines a set of values that for which the condition will resolve to "true". A given data point, $d \in D$, resolves to "true" for a given condition if its corresponding attribute, d_{attr}, is contained in q_{attr}. For continuous attributes (e.g., sales, duration, date/time) q_{attr} takes the form of an interval (e.g., $q_{sales} := [100, 200]$), while for binary or categorical attributes (e.g., region, productCategory, isDirected) it takes the form of a discrete set (e.g., $q_{region} := \{SouthEast, SouthWest\}$). The predicate Q resolves to "true" for d only if it satisfies all conditions:

$$\forall attr \in Attr_Q, d_{attr} \in q_{attr}$$

In addition to defining a truth value for each data point in D, Q defines a function that takes the full dataset as input and outputs a subset where each data point resolves to "true":

$$Q(D) = D_Q \subseteq D$$

Crucially, with this formalization we can represent both a concrete subset of the

data, and an abstract explanation of the subset whose logical structure is interpretable independent of the data. Moreover, the logical structure of Q implicitly defines related predicates. Specifically, $\neg Q$ defines the compliment of Q, defining the subset $D \setminus D_Q$, and $Q_{\neg attr}$ defines a predicate that is otherwise identical to Q, but resolves to the opposite truth value for the condition q_{attr}. In this way, a predicate defines not only an explanation and a subset, but also a set of relationships to other predicates. It is users' ability to reason about all three of these components that makes predicates a uniquely powerful tool for bridging human and automated reasoning.

6.1.1.2 Induction Engine

What does "induction" mean in the context of predicates? Predicate induction describes the process of generating a predicate from D, such that the logical structure of Q or the subset it defines have some property that is relevant for a given analysis use-case. Relevance is encoded as a function, f, that takes a predicate as input and outputs a numeric value. Formally, E denotes a "predicate induction engine" that takes the dataset D and the relevance function f as input, and outputs a set of relevant predicates:

$$\mathbf{Q} = E(D, f)$$

The choice of relevance function plays the principle role in determining the behavior of the induction engine, and must be carefully considered for each use-case.

How did we choose a relevance function for each of our three use-cases (AR, XDR, GVR)? In the case of AR, a predicate is relevant if it defines a subset that is highly anomalous compared to the rest of the data. This is represented formally in PIXAL's induction engine as a Bayesian t-test that compares the anomaly scores in D_Q against those in the rest of the data, $D_{\neg Q}$. This outputs a Bayes factor quantifies the evidence for the hypothesis stating that the subset is anomalous. For XDR, a predicate defined in the original dimensions is relevant if its subset matches

the pattern defined by the user in the DR results. Formally, DimBridge's induction engine uses the binary cross entropy (BCE) to quantify the degree of match. For GVR, a predicate is relevant if it defines a subset that closely matches a set of points in a graph visualization dataset labeled by a given visualization technique. Similar to DimBridge, VIGOR's induction engine uses BCE to quantify this match.

6.1.1.3 Algorithms

Given a dataset and a relevance function, how do we generate predicates in practice? In this work, we consider any algorithm that takes a dataset, a relevance function, and possibly additional parameters (e.g., the scale factor of the Bayesian t-test, a pattern specified by the user) and returns one or more predicates to be a predicate induction algorithm. These algorithms vary in how they search the space of candidate predicates and, as we will see in following sections, how they balance the tradeoff between accuracy, interpretability, and speed. In the systems presented in this thesis, predicate induction is implemented with two algorithms: recursive predicate induction (RPI), and predicate regression (PR).

The RPI algorithm incrementally builds predicates by recursively adding new conditions to an existing set of base predicates. The process continues until no further refinement improves the relevance score. This algorithm was designed specifically for the AR use-case, where 1) an unspecified number of predicates are required to accurately capture an unknown number of anomalous collections, 2) overlapping predicates providing multiple plausible explanations are desired, and 3) induction can be run independently as a preprocessing step, making speed a minor consideration.

PR frames predicate learning as a numeric optimization problem, representing a fixed number of predicates as a set of parameters that can be optimized by minimizing a loss function using stochastic gradient descent. This loss function is based on predicate relevance, with relevant predicates resulting in a smaller loss. This algorithm was initially designed for the XDR use-case, where 1) the number of predicates was known in advance (determined by the user interaction), 2) induc-

tion occurs between interactions and speed is a major consideration, and 3) data attributes are continuous, allowing for the straightforward definition of a differentiable loss function. This algorithm was extended for the GVR use-case by designing a differentiable loss function that, in addition to continuous attributes, is compatible with binary and categorical attributes such as those represented by graph statistics.

While the above algorithms were explicitly designed for predicate induction in the context of our three use-cases, several related families of algorithm could be adapted for similar purposes. Decision trees partition a dataset according according to a set of conditions, allowing each path from root to leaf to be interpreted as a predicate [BFSO84]. While fast to train and naturally interpretable, decision trees are typically optimized for predictive power on the entire dataset rather than for relevance of a particular subset. However, future work could modify the objective function used to determine splits to optimize relevance of subsets defined by each node. Algorithms like RIPPER [C$^+$96] or Bayesian Rule Lists [YRS17] generate compact sets of interpretable rules that resemble predicates. Similar to decision tree learning, they are designed to optimize classification accuracy, but future work could apply similar approaches that are optimized based on relevance. Association rule mining is an approach that finds frequent co-occurrences of attribute-value pairs [HGN00]. While a set of attribute-value pairs is readily interpreted as a conjunctive predicate, these rules are most commonly evaluated in terms of how frequently they occur in the dataset, measured by metrics like support and confidence. Future work could adapt these algorithms for predicate induction by replacing these metrics with a use-case specific measure of relevance.

6.1.1.4 Similar Algorithms

A number of automated analysis methods exist that, similar to predicate induction, identify relevant subsets, but do not provide explanations:

Clustering algorithms: (e.g., K-means [Llo82], DBSCAN [EKSX96]) group data into clusters based on similarity, but the resulting clusters are often opaque. While centroids or densities can offer some intuition, the clusters are not defined by explicit

conditions that describe why a point belongs to one group rather than another.

Classification models: (e.g., support vector machines [HDO+98]) assign class labels to data points, effectively partitioning the dataset into subsets. However, these models often prioritize predictive accuracy over interpretability. A predicted class label does not explain why the prediction was made or what defines the group.

Random forests: [Bre01] define subsets of data at each level. However, the complexity of these models often makes individual paths uninterpretable. Ensuring high accuracy results in many paths that overlap or conflict, making it impossible to extract a coherent explanation for any one subset.

Some use-cases require meaningful explanations, but defined over the whole dataset rather than specific relevant subsets. A number of automated analysis methods exist that, similar to predicate induction, provide explanations, but do not distinguish between relevant and irrelevant subsets:

Explainable AI (XAI): techniques like SHAP [LL17] or LIME [RSG16] provide explanations for individual predictions, attributing importance to input features. While useful for understanding model behavior, these explanations are local and do not define subsets that generalize across multiple data points.

Sparse linear models: [RC18a] offer global explanations in the form of coefficients, indicating which features contribute positively or negatively to the prediction. But these models describe relationships, not subsets-they do not tell us which data points are similar or relevant for a particular finding.

Finally, some use-cases require neither relevant subsets or meaningful explanations. A number of automated analysis techniques exist that are completely distinct from predicate induction:

Deep learning models: like those commonly used for image [LLY+21] and text [YSHZ19] data, are typically treated as black boxes. While they may achieve high performance on predictive tasks, their internal representations are rarely interpretable, and they do not produce understandable subsets of data.

Ensemble models: [SR18] that combine many weak learners often trade interpretability for accuracy, producing predictions that are difficult to trace back to a

coherent subset or rationale.

In short, while other automated analysis methods exist that may produce relevant subsets or meaningful explanations, predicate induction produces both. While predicate induction is well suited for use-cases that require both, other techniques may be more appropriate for other use-cases. In the next section, I detail the drawbacks that may be encountered in using predicate induction - specifically, managing the tradeoff between accuracy, interpretability, and speed.

6.1.2 What Use-Cases Benefit from Predicate Induction?

In general, predicate induction is an effective automated analysis technique for VA because predicates represent both a concrete subset of the data that can be inspected by users, and an abstract explanation of the subset whose logical structure can be interpreted by users independently of the data. Additionally, predicates can be directly manipulated by users. When faced with ill-posed problems, or when results generated by the induction engine do not align with a user's expectations, predicates offer a way to explore alternatives or adjust assumptions. With these advantages, predicates serve as a bridge between human and automated reasoning, allowing users to make use of the the flexibility and context-awareness of the former while leveraging the scalability of the latter. Furthermore, in defining predicate induction, we observe that the notion of predicate "relevance" is flexible, and can be defined for diverse use-cases, and that the exact implementation of the induction algorithm is similarly flexible, allowing designers to take into account diverse, use-case specific constraints. While these characteristics suggest strong advantages in general, they do not necessarily indicate whether a predicate induction will be the optimal choice for a given use-case. Specifically, why was predicate induction effective in our three use-cases? Generally, what characteristics can we observe from a use-case to determine if predicate induction will be useful?

In AR, the goal is to identify collections of anomalies and explain the underlying phenomenon in terms that can be interpreted by non-technical stakeholders. In XDR, the goal is to explain a pattern observed in a low-dimensional projection

in terms of the original dimensions. In GVR, the goal is to identify a subset of graphs that can be visualized using a particular technique, and explain that subset in terms of guidelines based on interpretable graph statistics. In each of these use-cases, the underlying analysis goal can be described as identifying **relevant subsets** and **meaningful explanations**. For these use-cases, we can therefore determine that a predicate represents a valid insight if it is relevant and meaningful.

Relevant: The subset defined by a predicate must be *relevant* given a particular analysis use-case. In AR, relevance is based on anomaly scores - each point has a numeric score, and relevant subsets contain many high-scoring points. In XDR and GVR, relevance is binary: a point either belongs to the pattern (XDR) or is labeled as appropriate for a visualization (GVR). In general, a relevant predicate selects a subset consisting predominantly of relevant data points.

Meaningful: In addition to representing a relevant subset, the logical structure of a predicate must represent some *meaningful* concept within the context of the use-case. In AR, this means the explanation must be understandable to domain experts. In XDR, explanations must be expressed in terms of the original dimensions. In GVR, the explanations must represent guidelines defined based on graph properties. In general, a predicate is meaningful when its logical structure maps to an understandable concept within the problem domain.

While these use-cases appear different, on their surface, the underlying analysis goal can be described generally as identifying relevant and meaningful predicates. These examples point to a broader generalization: predicate induction is well-suited to problems where the goal can be expressed as finding and explaining relevant subsets of data.

6.1.3 Limitations

The simple logical structure of conjunctive first-order predicates makes a powerful tool for VA, applicable to a range of diverse use-cases. However, this structure also lends itself to a number of limitations. Most acutely, the simple structure of predicates can lead to **inaccuracies** in representing patterns in the data. This can

lead to poor analysis outcomes - the user may fail to identify a critical insight, or be mislead by false discoveries. Additionally, while **interpretability** is generally a strength of predicate induction, there are cases where limitations may be encountered. Finally, while predicates are generally efficient to generate due to their simple structure, there are cases where **speed** may prove to be a limitation.

6.1.3.1 Accuracy

Because predicate induction is limited to constructing simple, axis-aligned subsets, it naturally struggles to capture more complex patterns in data. Specifically, accuracy limitations arise in three cases:

1. when patterns cannot be represented as a subset of rows
2. when a pattern can be represented as a subset, but the induction engine's objective function fails to identify it as relevant
3. when a subset is relevant, but does not fit in an axis-aligned bounding box

Patterns Other Than Subsets: Predicates describe patterns that correspond to a subset of the data - that is, a set of related rows. More complex relationships between rows, such as similarity, cannot be represented by predicates. Furthermore, predicates cannot capture relationships between columns, such as correlation or linear relationships.

Misaligned Objective Function: The relevance of a predicate is determined by the induction engine's objective function, which assigns each a numeric value. However, in practice relevance is often ill-defined or multi-dimensional, and the objective function must be simplified or rely on heuristics. In these cases, predicate induction may fail to identify subsets whose relevance is not fully captured by the objective function. For example, PIXAL's induction engine uses a Bayesian t-test on anomaly scores as its objective function. While a high Bayes factor signifies a high degree of evidence, this interpretation relies on the assumption that anomaly

scores are normally distributed. If this assumption fails to hold up, the Bayes factor could prove to be a poor measure of relevance.

Subsets Not Axis-Aligned: Predicate induction will yield inaccurate results on relevant subsets that cannot be fully captured by an axis-aligned bounding box - if a subset follows a diagonal, curved, or otherwise non-orthogonal shape, any predicate composed of simple conjunctions over individual attribute sets will only approximate this shape. The mismatch between the predicate's structure and the true geometry of the subset leads to two types of inaccuracies: false positives and false negatives. False positives occur when a predicate includes data points that irrelevant to an otherwise relevant pattern. For example, suppose an analyst is exploring a dataset where an unusual cluster of customers lies along a narrow diagonal in a 2D projection (e.g., where higher income correlates precisely with higher spending). A rectangular predicate attempting to enclose this diagonal will likely also include many points above or below the diagonal that do not exhibit the same spending-to-income behavior. False negatives occur when a predicate fails to include points that are actually part of a relevant subset. Continuing the diagonal cluster example, if the bounding box is too conservative, it may exclude valid instances along the edges of the pattern. Both types of inaccuracies can result in poor analysis outcomes, and erode trust in the system.

6.1.3.2 Interpretability

While predicates are generally interpretable, there are important limitations that emerge in practice. Interpretabilitty limitations arise in two cases:

1. when the logical structure of the predicate becomes too complex
2. when the data attributes themselves are not meaningful

Predicate Complexity: The compositional structure of conjunctive first-order predicates makes them simple to interpret. As long as each condition is itself interpretable, users can interpret the full predicate by considering all predicates at once.

However, users may fail to interpret a predicate as a whole if it is composed of more conditions than can be represented in working memory.

Data Attributes Not Meaningful: For a predicate to serve as an effective explanation, its logical structure must map to semantic concepts that users can understand and reason about. Otherwise, the explanation may be formally correct but practically useless in the context of the analysis. For instance, predicates defined over arbitrary numeric embeddings may be syntactically valid, but offer no insight into what the predicate actually means.

6.1.3.3 Speed

The space of candidate predicates is combinatorially large. To maintain tractability, induction algorithms must rely on some greedy heuristics or mechanism to constrain the search spaces. This is critical in VA applications, where the induction engine must be responsive to user interactions. The minimum acceptable speed for the induction algorithm is completely dependent on its particular role in the system. In PIXAL, predicate induction is run after anomaly scores are generated, but before the user interacts directly with the system. Because slow predicate generation cannot interfere with user interactions, PIXAL's induction engine can afford to run for longer and generates more candidate predicates as a result. In contrast, DimBridge's induction engine generates predicates in response to patterns identified by the user through interaction. Failing to generate results quickly could result in the analysis stalling or being abandoned completely.

6.1.3.4 Managing Limitations

Sufficient accuracy, interpretability, and speed are all needed for a system to be effective for a given use-case. The limitations of predicate induction can be managed through careful design decisions made for both the induction engine and the visual interface. For example, we can directly improve the accuracy of the induction engine by generating a greater number, more precise, or logically complex predicates, or

by applying data transformations that allow subsets to better match axis-aligned assumptions. To improve interpretability and speed, we can promote sparse predicates, initialize the algorithm to focus on a more narrow search space, or reduce the number of attributes that may be included in a predicate through dimensionality reduction or subspace sampling. In the visual interface, both accuracy and interpretability can be facilitated through visualization. By visualizing a subset, users can determine for themselves if a predicate sufficiently matches their idea of relevant. By visualizing an explanation, users can contextualize the logical structure of a predicate to better interpret its meaning. I discuss these approaches in more detail and provide examples from our use-cases in the following sections.

6.1.4 Managing Limitations with the Induction Engine

Limitations of the accuracy, interpretability, and speed of predicate induction can be managed through the design of the induction engine. Accuracy can be improved by generating a larger number of predicates, making predicates more precise, increasing logical complexity, or transforming the data to better conform to axis-aligned assumptions. Interpretability can be preserved by favoring sparse predicates or limiting logical complexity. Speed can be improved by reducing the number or complexity of predicates, lowering the dimensionality of the data, or by intelligently initializing the search space.

Generating More Predicates: Increasing the number of predicates can reduce the risk of false negatives, identifying a larger number of potentially relevant subsets. This is particularly valuable in use-cases where there are many potential relevant subsets, or where multiple plausible explanations exist for the same subset. Because both conditions are true for AR, a major design requirement of PIXAL's induction engine was to generate multiple, overlapping predicates. However, more predicates increases the risk of false positives, as they may capture spurious or overlapping subsets. Additionally, generating and evaluating more predicates increases computational cost. In the next section, I discuss how these tradeoffs are managed through PIXAL's visual interface.

Generating More Precise Predicates: Precision can be increased by refining predicates to define tighter boundaries around the relevant subset. This reduces false positives by excluding irrelevant data, improving the quality of the explanation. However, more precise predicates may also exclude valid points near the boundary of the pattern, increasing the risk of false negatives. In addition, precise predicates tend to contain more conditions, which can make them harder to interpret and slower to generate.

Using More Complex Logical Structures: Predicates can be made more accurate by increasing their logical complexity. For example, including disjunctions or negations allows predicates to capture subsets that are not strictly defined by axis-aligned bounding boxes. However, even a small increase in logical complexity results in a large increase in the induction engine's search space, making this approach computationally expensive. Additionally, more complex predicates will be more difficult for users to interpret.

Applying Dimensionality Reduction: Transforming the data using techniques like PCA [Pea01] can make previously non-axis-aligned patterns become axis-aligned in the new space. Dimensionality reduction also improves speed, shrinking the induction engine's search space by reducing the number of available attributes. However, the transformed dimensions typically lose their original semantic meaning, making the resulting predicates difficult to interpret.

Promoting Sparse Predicates: Sparse predicates (those with fewer conditions) are easier to interpret and faster to compute. This is particularly valuable for use-cases with a high number of attributes, such as XDR. In DimBridge, we promote sparsity by adding a regularization term to the objective function. However, sparsity often comes at the cost of accuracy: with fewer conditions, the predicate may capture a broader, less precise subset of the data, increasing false positives or failing to fully isolate the relevant subset. In the next section, I discuss how reduced accuracy is managed through DimBridge's visual interface.

Subspace Sampling: Subspace sampling involves randomly selecting a subset of attributes for each run of the induction engine. Similar to promoting sparsity, this

approach can reduce computational cost and enhance interpretability by limiting the number of available attributes. However, important attribute combinations may be missed, leading to lower accuracy, and in some cases reducing interpretability.

Defining Initial Predicates: An alternative approach to reducing the induction engine's search space is to initialize the algorithm with a curated set of predicates. If the initial predicates are well-informed, the search will converge on more promising regions and yield faster and more accurate results. However, if the initialization is poor, the algorithm may converge on suboptimal solutions or fail to discover relevant subsets altogether.

Each of these techniques can be used to manage the limitations of predicate induction, either by increasing accuracy, interpretability, or speed. While this list is not exhaustive, it underscores the inherent tradeoff between these factors, which must each be weighed according the nuances of each specific use-case. However, it is not always possible to find a sufficient balance through the induction engine alone. In the next section, I describe how limitations can be further mitigated through the design of the visual interface.

6.1.5 Managing Limitations with the Visual Interface

Limitations of predicate induction can be mitigated to some extent through the design of the induction algorithm. However, the inherent tradeoff between accuracy, interpretability, and speed makes it difficult to manage these limitations through the induction engine alone. In these cases, visualization and interaction can be used in conjunction with the induction engine to help manage these tradeoffs. By visualizing data subsets directly, users can evaluate the relevance of induced predicates beyond what is captured by a single score, and identify and correct inaccuracies. Visualizing explanations (i.e., the logical structure of predicates) can facilitate interpretation by contextualizing logical conditions in terms of the overall structure of the data. In the systems presented in this thesis, visualizations of both subsets and explanations take a number of forms. While a visual encoding of a single subset or explanation can be useful by itself, the power of visualization becomes even more apparent when

leveraged to compare multiple predicates.

Furthermore, interaction allows users to define, modify, or filter predicates to directly augment the induction engine. This is especially important for use-cases with ambiguous objectives, where relevance is difficult to formalize. Interaction also facilitates interpretation: by directly engaging with the logical structure of a predicate - adjusting values, changing operators, or composing new conditions - users can better align explanations with their knowledge of the data and domain.

6.1.5.1 Visualization Helps Verify Relevance and Accuracy

Knowing that predicates generated by the induction engine are unlikely to perfectly capture the true structure of relevant subsets in the data, visualization plays a critical role in helping users bridge the gap between an induced predicate and the underlying pattern it approximates. While the induction engine's objective function provides a single numeric metric of relevance for a subset, visualizing the subset provides a more flexible representation. Furthermore, restricting subsets to fit in axis-aligned bounding boxes will inevitably result in some relevant data points being excluded, and some relevant points being included. Visualization can be used to manage these inaccuracies, helping users understand which aspects of a pattern were faithfully captured and which aspects were not. In some cases, this can be achieved by visualizing a subset alone, providing users with a richer understanding of its characteristics beyond what is represented by a strictly-specified objective function. In other cases, reasoning can be further facilitated by allowing comparison to another subset, or a larger group of similar subsets.

Visualizing a Subset: While the induction engine's objective function quantifies the relevance of a predicate as a single numeric value, directly visualizing the subset represented by the predicate provides a richer understanding of its relevant characteristics. For example, PIXAL uses the Bayesian t-test factor as its objective function, striking a balance between the size of a subset, and the average and standard deviation of its anomaly scores. However, a user may wish to weigh some of these elements more than others. Furthermore, the Bayesian t-test assumes that

anomaly scores are normally distributed, while this may not always be true in practice. In the Score View, users get a more complete picture of anomaly scores in a subset by visualizing the full distribution, revealing details not apparent by the Bayes factor alone.

Visualizing two Subsets: In addition to visualizing a the anomaly score distribution of a single subset, PIXAL's Score View allows the user to compare the distributions of multiple subsets. By default, the Score View contrasts the anomaly score distribution of a given predicate with that of the rest of the data, allowing users to evaluate scores in an anomalous collection relative to anomaly scores of the normal data. Furthermore, the user can select multiple predicates to visualize in the Score View at once, allowing users to consider a collection's anomalousness relative other anomalous collections, in addition to the normal data. Similarly, PIXAL's Attribute View helps users determine if a collection is anomalous by comparing it to a similar, but normal, subset. Specifically, the normal subset is defined by a nearly identical predicate, but with one condition negated. The anomalous and normal subsets are compared in terms of domain relevant attributes, allowing users to evaluate a collection without needing to rely on potentially opaque anomaly scores. In DimBridge, the induction engine uses BCE as a metric for relevance, summarizing the match between the predicate and a selected pattern as a single value. However, this value does not convey which components of a pattern were accurately represented by a predicate, and which were not. DimBridge's Projection View allows users to evaluate relevance for each data point by highlighting false positives and false negatives, comparing the generated predicate to the subset defined by the user. Furthermore, while the BCE can represent the match between a user selected pattern and a subset in the high-dimensional data, it does not indicate whether that subset represents an interesting pattern. In the SPLOM View, the subset is visualized in the context of the rest of the data (within a smaller, relevant subspace), allowing users to reason about the broader pattern.

Visualizing many Subsets: While pairwise comparisons offer valuable context, visualizing multiple subsets can allow users to evaluate a predicate relative to a set

of predicates that share some meaningful relationship. PIXAL's Predicate View displays statistics (Bayes factor, average and standard deviation of anomaly scores, subset size) for many predicates simultaneously. Users can assess not just whether a predicate is anomalous compared to the rest of the data, but how it ranks compared to the other predicates on multiple factors related to anomalousness. When the Predicate View is filtered to display only a related group of predicates (i.e., all predicates including or excluding certain attributes), the Score View and Predicate View allow for comparison along specific dimensions of interest (e.g., among all predicates where "Region = East"). In DimBridge, the SPLOM View supports utilizes comparison of multiple subsets in cases where the pattern identified by the user is a continuous shape. The path of the shape, discretized into k subsets, is traced in terms a relevant set of the original dimensions.

6.1.5.2 Visualization Facilitates Meaningful Interpretation

Predicates are naturally interpretable, describing a subset by a set of human readable conditions defined over data attributes. However, for an explanation to be meaningful, these conditions must correspond to concepts that are understood within the semantics of the use-case, beyond numeric ranges or discrete sets. Visualization can help facilitate meaningful interpretation by contextualizing these conditions in terms of the dataset as a whole. Beyond adding context to a single predicate, visualization can further facilitate reasoning by comparing the logical structure of one predicate to a similar or contrasting predicate, or by visualizing a group of predicates within the same context.

Visualizing an Explanation: The logical structure of a predicate can be visualized by orienting the intervals or sets of values defined for each condition relative to full domain defined by the dataset as a whole. Users can then interpret the meaning of a particular set of values defined by a condition with the full dataset as a point of reference. In PIXAL's, the Predicate View the conditions of each predicate is visualized by listing all each condition as an attribute and set of values, displayed as text. Selecting a condition displays an interactive element: For those defined

over discrete attributes, a dropdown menu displays all values found in the data, with those included in the condition highlighted. For those defined over continuous attributes, a range slider displays the upper and lower bounds of the data as its limits, while the positions of the left and right sliders represent the range defined by the condition. A similar approach is used in the DimBridge interface, where the Predicate View displays the range of values allowed for a given attribute within the minimum and maximum values found in the full dataset.

Explanation vs. Explanation: Similar to visualizing subsets, further context can be provided by visualizing a pair of explanations. In DimBridge, a pair of predicates generated by the "Select and Contrast" interaction have both sets of conditions displayed side-by-side in the Predicate View, grouped by attribute. This layout allows users to compare the value ranges between two distinct predicates and evaluate their meaning in terms of their similarities and differences.

Explanation vs. Many Explanations: Beyond comparing two explanations, visualizations can compare a set of explanations defined by a particular relationship. In PIXAL's Predicate View, users can filter predicates to only show those that include or exclude a specific set of attributes, allowing them to consider the meaning of an explanation relative to a group of similar predicates. In DimBridge, when the user selects a continuous pattern, the Predicate View compares the conditions of the k discretized predicates, grouped by attribute. By visualizing many explanations, users can reason about how the logical structure defining the pattern in high-dimensions varies continuously with the shape of the pattern observed in the DR results.

6.1.5.3 Interaction Supports Generation of Relevant Predicates

While visualization helps users evaluate and verify the accuracy of predicates generated by the induction engine, interaction facilitates more direct intervention, allowing users to define, modify, or filter predicates themselves. This intervention is critical in cases where the induction algorithm produces incomplete or imprecise explanations due to the inherent limitations of conjunctive predicates, a mismatched

objective function, or as the result of computational tradeoffs. Through interaction, users can incorporate domain knowledge, introduce additional logical complexity, or fine-tune predicates that more accurately represent relevant subsets.

In PIXAL, the induction engine generates a large number of overlapping predicates, many of which may contain false positives or otherwise imperfectly describe relevant patterns. If a user determines that a predicate is not actually relevant after viewing some combination of the Predicate, Score, and Attribute Views, they can choose to filter it out by interacting with the Predicate View. In addition to filtering, PIXAL's Predicate View allows users to modify predicates generated by the induction engine. Specifically, users can modify the set of values for a given condition by adjusting the associated dropdown menu or range slider, add, remove, or negate a condition, or refine the predicate by defining additional conditions.

Furthermore, PIXAL's Predicate View supports the creation of user-defined predicates from scratch. This is essential in cases where the induction engine fails to identify relevant subsets, due to it not being sufficiently favored by the objective function or too structurally complex to express with simple conjunctive predicates. User-defined predicates afford more complexity, allowing negation at the predicate or condition level, and conditions that specify multiple discrete attribute values. In DimBridge, user-defined predicates play a fundamental role in defining the induction engine's objective function. In XDR, relevance is defined relative to patterns observed in the DR results. In DimBridge, users specify a pattern by defining one or more predicates in terms of the two dimensions representing the DR results.

6.1.5.4 Interaction Supports Interpretation

Beyond allowing users to directly identify relevant subsets, interaction can also play a crucial role in facilitating meaningful interpretation of explanations. In many cases, users possess domain knowledge that is not available to to the induction engine. Interaction enables users to directly encode this knowledge by creating new predicates or modifying existing ones to reflect concepts that are meaningful within the context of the use-case. For example, a sales analyst may know that particular

store locations have historically underperformed due to supply chain issues. While these issues may not be represented in the data itself, the analyst can create a predicate representing the effected locations in order to encode this important concept. Similarly, an analyst might adjust the time interval of a predicate to generated by the induction engine to align with known fiscal quarters or business cycles, ensuring that explanations are semantically meaningful in the context of the use-case. Furthermore, interaction can allow users to filter out predicates representing explanations that, while structurally valid, may not represent meaningful concepts. In PIXAL, the induction engine to generates a large number of overlapping predicates, often representing many alternative explanations for a similar collection of anomalies. The Predicate View allows users to filter out predicates representing implausible or overly complex explanations.

6.2 Framework

In the previous section, I discuss the benefits and limitations of predicate induction in detail. In this section, I define this approach within a formal framework. I begin by motivating the need for such a framework. While I have demonstrated that predicate induction in VA is a flexible approach that can be applied to diverse use-cases, this flexibility comes at a cost. The exact role of the induction engine, visual interface, and user vary widely across use-cases, making it difficult to provide concrete design guidelines for a particular use-case. The proposed framework represents each component - the induction engine, visual interface, and user interactions formally. Together, these components define a unified framework describing systems that produce a set of relevant and meaningful predicates, given a dataset and a user. The space of predicates that can be generated by the induction engine or through user interactions is defined by a context-free grammar that allows for varying levels of logical complexity. The induction engine is defined by an objective function, and a variable number of predicates returned by the engine. The visual interface is formalized as a set of views, each defined as a function that maps a candidate

predicate to some degree of evidence for either relevance or meaning.

I conclude by discussing observations gained from applying this framework to the three systems and use-cases discussed in this thesis, noting contrasting degrees of complexity between generated and user-defined predicates, varying levels of complexity among induction engines and visual interfaces, and, particularly, the value of the Predicate View in anchoring multiple visualizations of the same subset. Additionally, I discuss how future work can be directed towards evaluating and refining this framework to represent more diverse use-cases and systems, and towards building practical tools for applying this framework in the design of future systems.

6.2.1 Why a Formal Framework?

As argued in this thesis, predicate induction is a powerful technique for VA, and can be applied to a diverse range of analysis use-cases. By representing both abstract explanations and concrete subsets of data, predicates can bridge the gap between human and automated reasoning to facilitate the iterative sense-making loop that characterizes effective VA. This approach is highly flexible, with the potential to be applied to a wide range of diverse use-cases. The behavior of the induction engine is determined primarily by the choice of objective function, whose only requirement is that it outputs a numeric value. Furthermore, predicate induction can be implemented using a variety of algorithms, providing further flexibility in meeting the constraints of a particular use-case. Similarly, the visual interface can support reasoning about the relevance and meaning of predicates using a wide variety of visual encodings, as well as incorporating human reasoning directly through user interaction.

While flexibility is a core strength of predicate induction in VA, the cost is a lack of specificity in applying this approach to new systems and evaluating existing systems. In the previous section, I argue that predicate induction for VA is an effective approach for use-cases that require identifying relevant subsets and meaningful explanations. However, both concepts are loosely defined, relying on the designer to operationalize these concepts in a way that aligns with the semantics and analysis

goals of a particular use-case. While this conceptual framing can be broadly applied, there are no guarantees that a system can be designed to sufficiently represent a given notion of relevance and meaning. For the systems described in this thesis, we took inspiration from a number of existing systems that use predicate induction and VA for similar to solve similar use-cases. However, while these systems may share a similar high-level approach, they differ significantly in their implementation details. Without a formal framework, it is difficult to articulate the relationship between these systems, compare their designs meaningfully, or determine whether an existing approach can be adapted to a new context.

The formal framework presented in this section attempts to provide the structure needed to effectively compare existing systems that use predicate induction and VA, and apply predicate induction and VA to new use-cases.

6.2.2 Predicate Induction for Visual Analytics

The fundamental goal of the systems described in this thesis is to identify a set of predicates, each defining a relevant subset and a meaningful explanation. Such a system, denoted S, is composed of a predicate induction engine (E), and a visual interface (V). The user's goal is to identify a set of relevant and meaningful predicates ($\mathbf{Q}$) given a particular dataset (D). The predicate induction engine (E) operates over D and a relevance function (f) to produce a set of predicates, $\mathbf{Q}_E$, from the set of all valid predicates, $\mathbf{Q}_E^*$:

$$\mathbf{Q}_E \subseteq \mathbf{Q}_E^* = E(D, f)$$

The user interacts with the visual interface to generate a set of predicates, $\mathbf{Q}_{U,V}$, from the set of all valid predicates, $\mathbf{Q}_{U,V}^*$. In addition to generating user-defined predicates, the visual interface consists of two sets of views, V_a and V_b, which help the user determine if a given predicate is relevant and meaningful respectively. Given the predicates produced by the induction engine ($\mathbf{Q}_E$) and the user ($\mathbf{Q}_{U,V}$), the user's goal is to identify those that are sufficiently likely to be both relevant and

meaningful. Using a_Q and b_Q to denote the hypotheses that Q is relevant and meaningful respectively, $\mathbf{Q}$ is defined as the subset of candidate predicates that the user determines are sufficiently likely to be both relevant and meaningful:

$$\mathbf{Q} = \{Q \in \mathbf{Q}_E \cup \mathbf{Q}_{U,V} \mid \mathcal{L}_U(a_Q|V_a)\mathcal{L}_U(b_Q|V_b) \geq \tau\}$$

where $\mathcal{L}_U(x_Q|V_x)$ denotes the likelihood assigned by a particular user to a hypothesis given a set of observed views.

Within this framework, a given system is defined by the induction engine (E), the visual interface (V), and the space of valid predicates that can be produced by the induction engine ($\mathbf{Q}^*_E$) or through user interaction ($\mathbf{Q}^*_{U,V}$). In the following sections, I define each of these components in more detail.

System	$\mathbf{pre}_{Q,E}$	$\mathbf{pre}_{q,E}$	$\mathbf{set}_E$	$\mathbf{n}_Q$	$\mathbf{f}$
PIXAL	ϵ	ϵ	$set_{cont} \rightarrow [ak, bk)$ $set_{cont} \rightarrow \{a\}$	≥ 1	BTT(Q, ¬Q, r, $\text{attr}_{\text{anom}}$)
DimBridge	ϵ	ϵ	$set_{attr \notin \{dim_1, dim_2\}} \rightarrow [a, b]$	$\text{n}_{\text{select}} := 1$ $\text{n}_{\text{contrast}} := 2$ $\text{n}_{\text{shape}} := \text{k}$	$\text{f}_{\text{select}} :=$ BCE(Q, $\mathbf{Q}_{U,V}$, b, λ_1) $\text{f}_{\text{contrast}} :=$ BCE(Q_a, Q_b, $\mathbf{Q}_{U,V}$, b, γ_1, γ_2) $\text{f}_{\text{shape}} :=$ BCE($\mathbf{Q}$, $\mathbf{Q}_{U,V}$, b, λ_1, λ_2)
VIGOR	ϵ	ϵ	$set_{num} \rightarrow [a, b]$, $set_{cat} \rightarrow \{a\}$	|Vtypes|	BCE(Q, Q_{vtype}, b)

Table 6.1: Predicate induction engine specifications for PIXAL, DimBridge, and VIGOR.

System	$\mathbf{pre}_{Q,U}$	$\mathbf{pre}_{q,U}$	$\mathbf{set}_U$	$\mathbf{V}_\rho$	$\mathbf{V}_\mu$
PIXAL	$\epsilon \mid \neg$	$\epsilon \mid \neg$	$set_{cont'} \rightarrow [a, b]$ $set_{disc'} \rightarrow A$	predViewStats(Q, $\mathbf{Q}_{\text{filtered}}$) scoreView(Q, $\mathbf{Q}_{\text{selected}}$) attrView(Q, $\text{Q}_{\neg\text{attr}}$)	predView(Q, $\mathbf{Q}_{\text{filtered}}$)
DimBridge	ϵ	ϵ	$set_{attr \in \{dim_1, dim_2\}} \rightarrow [a, b]$	projView(Q, $\mathbf{Q}_{U,V}$) splomView(Q, ¬Q)	predView(Q)
VIGOR	$\epsilon \mid \neg$	$\epsilon \mid \neg$	$set_{num} \rightarrow [a, b]$, $set_{cat} \rightarrow \{a\}$	–	–

Table 6.2: Visual interface specifications for PIXAL, DimBridge, and VIGOR.

6.2.3 Predicates

The set of valid predicates that can be generated by either the induction engine or through user interaction varies across use-cases. In this framework, I limit valid predicates to conjunctions prefixed by a unary operator (e.g., negation). Each condition defines an attribute and a set of values for with the condition resolves to "true",

its own unary operator, and some additional compositional structure. A predicate defined over a dataset with m_D attributes consisting of $n_Q \leq m_D$ conditions can be defined by a context-free grammar (CFG) with the following production rules, with nonterminal symbols shown in blue and terminal symbols shown in red:

$$\langle Q \rangle \rightarrow \langle pre_Q \rangle \ (\ \langle q_{attr_1} \rangle \wedge ... \wedge \langle q_{attr_{n_Q}} \rangle \ )$$

$$\langle q_{attr} \rangle \rightarrow \langle pre_q \rangle \ (attr \in \langle set \rangle)$$

With this formalization, a set of valid predicates can be defined by specifying the production rules for the predicate-level prefix pre_Q, the condition-level prefix pre_q, and the set set. For example, the set of conjunctive predicates that allows discrete sets or intervals (open or closed), disjunctions, and negation of the whole predicate or individual conditions is defined by specifying:

$$\langle pre_Q \rangle \rightarrow \epsilon \mid \neg$$

$$\langle pre_q \rangle \rightarrow \epsilon \mid \neg$$

$$\langle set \rangle \rightarrow A \mid (\ (\mid [\ ) \ a, b \ (\ ) \mid] \ )$$

where $a, b \in D_{attr}, a < b$, and $A \subseteq D_{attr}$. More generally, I use G to denote a function that maps the specification of these production rules to a particular CFG that defines a set of valid predicates, $\mathbf{Q}^*$:

$$G : pre_Q \times pre_q \times set \rightarrow \mathbf{Q}^*$$

This formalization is used to define valid predicates generated by the induction engine ($\mathbf{Q}^*_E$) or user ($\mathbf{Q}^*_{U,V}$) for PIXAL, DimBridge, and VIGOR as follows:

6.2.3.1 PIXAL

PIXAL's induction engine done not allow for either predicate or condition-level prefixes, or any additional compositional structure. Sets are defined as an interval

for continuous attributes, and a set containing one value for discrete attributes. Formally, the set of valid predicates for PIXAL's induction engine is defined as:

$$\mathbf{Q}_E^* := G(\epsilon, \epsilon, set_{cont} | set_{disc}, \epsilon)$$

where ϵ denotes the empty string, while set_{cont} and set_{disc} define set for continuous and discrete attributes respectively. Specifically, set_{cont} yields $[a_k, b_k)$, where a_k and b_k are one of the endpoints defined by discretizing continuous attributes into k bins, and $a_k < b_k$. set_{disc} yields $\{a\}$, where $a \in D_{attr}$. In PIXAL's Predicate View, users modify or create their own predicates with more complex logical structure. Predicates generated by the user can include both predicate and condition-level negations. Furthermore, multiple values can be defined for discrete attributes, while intervals defined for continuous attributes can vary beyond the endpoints generated by the discretization process. Formally, the space of predicates that can be generated by interacting with PIXAL's visual interface is defined as:

$$\mathbf{Q}_{U,V}^* := G(\epsilon \mid \neg, \epsilon \mid \neg, set_{cont'} | set_{disc'}, \epsilon)$$

where $set_{cont'}$ yeilds $[a, b)$, with $a, b \in [\min(D_{attr}), \max(D_{attr}]$, and $set_{disc'}$ yields A, where $A \subseteq D_{attr}$.

6.2.3.2 DimBridge

For DimBridge, we assume that the original dimensions are all numeric. Formally, the space of predicates that can be generated by DimBridge's induction engine is defined as:

$$\mathbf{Q}_E^* := G(\epsilon, \epsilon, set_{attr \notin \{dim_1, dim_2\}}, \epsilon)$$

where set_{attr} yields $[a, b]$, and $a, b \in [\min(D_{attr}), \max(D_{attr})]$. dim_1 and dim_2 denote the dimensions produced by the DR algorithm, and are excluded from the predicate generated by the induction algorithm. In DimBridge's visual interface, the user

defines predicates by selecting bounding boxes in the Projection View. Formally, the space of predicates that can be generated by interacting with DimBridge's visual interface is defined as:

$$\mathbf{Q}^*_{U,V} := G(\epsilon, \epsilon, set_{attr \in \{dim_1, dim_2\}}, \epsilon)$$

6.2.3.3 VIGOR

VIGOR's induction engine generates predicates over numeric, and categorical (including binary) graph statistics. Formally:

$$\mathbf{Q}^*_{E} := G(\epsilon, \epsilon, set_{num}, |set_{cat}, \epsilon)$$

where set_{num} yields $[a, b]$ and set_{cat} yields A, with $a, b \in [\min(D_{attr}), \max(D_{attr})]$ and $A \in D_{attr}$. While users do not directly create predicates in the VIGOR system, we demonstrated how experts can extract predicates from the literature. These take a similar logical form as those generated by the induction algorithm. However, similar to the case with PIXAL, additional logical structure can be afforded:

$$\mathbf{Q}^*_{U,V} := G(\epsilon \mid \neg, \epsilon \mid \neg, set_{num} | set_{cat}, \epsilon)$$

6.2.4 Induction Engine

The induction engine takes the data, D, and objective function, f, as input, and generates a set of n_Q predicates, $\mathbf{Q} \in \mathbf{Q}^*_E$. Specifically, f, is a function that takes a predicate, Q, and a set of other parameters, Θ_f, and outputs a numeric score:

$$f : Q \times \Theta_f \rightarrow \mathbb{R}$$

Formally, the induction engine is defined by the objective function, and the number of predicates that are generated. I define the induction engine of PIXAL, DimBridge, and VIGOR using this method.

6.2.4.1 PIXAL

Given users' need to compare multiple plausible explanations for anomalous collections, PIXAL's induction engine did not specify a fixed number of predicates to generate:

$$n_Q \geq 1$$

To quantify the relative anomalousness of a subset, PIXAL uses a Bayesian t-test that compares anomaly scores in a subset to those in the rest of the data:

$$f := BTT(Q, \neg Q, r, attr_{anom})$$

and depends on the scale parameter, r, and the attribute designated as the anomaly score, $attr_{anom}$.

6.2.4.2 DimBridge

The number of predicates generated by DimBridge's induction engine depends on the pattern type selected by the user:

$$n_Q := \{n_{select}, n_{contrast}, n_{shape}\}$$

Specifically, selecting a single cluster requires generating one predicate ($n_{select} := 1$), selecting two contrasting clusters requires generating two predicates ($n_{contrast} := 2$), and selecting a continuous shape requires generating k predicates, where k is the number of bins used to discretize the user's selection ($n_{shape} := k$). For its objective function, DimBridge uses the binary cross entropy between the predicate and the pattern selected by the user. Similarly, its exact formulization depends on the pattern selected by the user:

$$f := \{f_{select}, f_{contrast}, f_{shape}\}$$

When selecting a single cluster:

$$f_{select} := BCE(Q, \mathbf{Q}_{U,V}, b, \lambda_1)$$

where γ_1 controls the strength of the sparsity term, and b the steepness of the differentiable bump function. For contrasting clusters:

$$f_{contrast} := BCE(Q_a, Q_b, \mathbf{Q}_{U,V}, b, \gamma_1, \gamma_2)$$

where γ_2 controls for the smoothness constraint between Q_a and Q_b. Finally, for continuous shapes:

$$f_{shape} := BCE(\mathbf{Q}, \mathbf{Q}_{U,V}, b, \lambda_1, \lambda_2)$$

6.2.4.3 VIGOR

VIGOR's induction engine one predicate for each visualization type:

$$n_Q := |Vtypes|$$

where $Vtypes$ is the set of graph visualization techniques that appear in the dataset D. Similar to DimBridge, VIGOR's induction engine uses binary cross entropy as its objective function. Specifically, the BCE between the predicate and the subset of rows labeled by a given visualization type, denoted by the predicate Q_{vtype}:

$$f := BCE(Q, Q_{vtype})$$

6.2.5 Visual Interface

Formally, V denotes a set of views making up a system's visual interface. Each view has one of two roles in terms of the user's analysis: to help determine if the subset defined by a predicate is relevant, or help determine if its logical structure is meaningful:

$$V = \langle V_a, V_b \rangle$$

where V_a is the set of views used to determine relevance, and V_b is the set of views used to determine meaning.

6.2.5.1 PIXAL

PIXAL's visual interface consists of three views, the Predicate View, Score View, and Attribute View. The Predicate View displays anomaly score statistics for each predicate, which can be compared across the rest of the filtered predicates:

$$predViewStats(Q, \mathbf{Q}_{filtered})$$

In the Score View, users compare the distribution of anomaly scores between a set of predicates selected from the Predicate View:

$$scoreView(Q, \mathbf{Q}_{selected})$$

In the Attribute View, users compare a predicate to a similar subset, defined by a predicate identical to the target predicate other than one of it's clauses (defined by the attribute *attr*) is negated:

$$attrView(Q, Q_{\neg attr})$$

Together, these views are used to help determine if a given predicate is relevant:

$$V_a = \{predViewStats(Q, \mathbf{Q}_{filtered}), scoreView(Q, \mathbf{Q}_{selected}), attrView(Q, Q_{\neg attr})\}$$

In the Predicate View, users interpret the meaning of a predicate's logical structure by visualizing value ranges and sets in the context of the data, and comparing them to other filtered predicates:

$$V_b := \{predView(Q, \mathbf{Q}_{filtered})\}$$

6.2.5.2 DimBridge

Like PIXAL, DimBridge includes a Predicate View to help users interpret the conditions of a predicate in context:

$$V_b := \{predView(Q)\}$$

In the Projection View, users evaluate the accuracy of the pattern identified by a predicate compared to the pattern selected by the user. In the SPLOM View, users contextualize a pattern in terms of relevant dimensions:

$$V_a := \{projView(Q, \mathbf{Q}_{U,V}), splomView(Q, \neg Q)\}$$

6.2.5.3 VIGOR

While the goal of GVR is to generate graph visualizations, VIGOR does not include any visualizations of the predicates themselves. Similarly, VIGOR does not include any visualizations of subsets of the graph dataset, as this data is generally not of interest to the user who is only trying to visualize their own graph:

$$V_a, V_b := \{\}$$

6.2.6 Discussion

The purpose of the formal framework presented in this chapter is twofold: to provide a structured lens for analyzing current systems that use predicate induction in visual analytics, and to offer a foundation for applying predicate induction in the design of VA systems.

It is important to note that the framework outlined here is not the only way to formalize predicate induction in VA. Specifically, we have focused on a framework that captures the three systems presented in this thesis: PIXAL, DimBridge, and VIGOR. However, extending this framework to represent other systems may require introducing additional complexity. For example, although we observe that

the behavior of a predicate induction engine is largely determined by the number of predicates generated and the objective function, we also note that the exact implementation of the induction algorithm can vary significantly between systems. This framework could benefit from specifying additional parameters to distinguish between implementations (e.g., overlapping vs. distinct, sparsity) or to capture the additional parameters of the objective function (e.g., scale or regularization factors). Similarly, while we model views in the visual interface as functions that output evidence for a predicate's relevance or meaningfulness, this representation omits any definition of the visual encoding itself. A more comprehensive framework might incorporate an expressive model of visual encodings (e.g. grammar of graphics [Wil11]) to better account for how visual representation influences interpretation and interaction. Finally, while our grammar-based representation of valid predicates is sufficient for our use-cases, more expressive logical models could allow future frameworks to describe a richer space of possible explanations.

Nonetheless, this framework allows for several observations about the systems and use-cases presented in this thesis:

1. Efficient predicate induction requires limiting the complexity of the search space. For each of our systems, we disallow negations and compositional conditions, and restrict discrete sets to contain only one value.

2. User-generated predicates, in contrast, can afford more expressivity. Users can define negated terms, construct disjunctions, and describe discrete sets containing multiple values.

3. Predicate induction can support objective functions with varying levels of complexity. While the objective function of PIXAL and VIGOR include one additional parameter (the scale factors r and b respectively), DimBridge's objective function includes two additional regularization factors. Additionally, DimBridge can utilize multiple possible objective functions depending on the type of pattern selected by the user.

4. Across systems, we generally observe a diverse set of visualization techniques tailored to each use-case. A notable exception is the Predicate View, which has a similar implementation in both PIXAL and DimBridge. In both systems, the Predicate View is situated between two views focusing on different aspects of the same subset. In this way, the Predicate View can act as a bridge between different views of the same subset, with the explanation acting as an interpretable anchor.

Chapter 7

Discussion

In this thesis, I discuss the use of predicate induction in visual analytics as a powerful tool for bridging human and automated reasoning. In particular, I discuss the application of this approach to three use-cases: anomaly reasoning, explainable dimensionality reduction, and graph visualization recommendation. While these use-cases appear to have little in common on their surface, they can be conceptualized in terms of two critical requirements: (1) the need to identify relevant subsets of the data and (2) the need to explain those subsets in a way that is meaningful to the user.

In general, I discuss the application of predicate induction and visual analytics for use-cases that can be interpreted within this conceptual framing, as well as the inherent limitations of this approach. While the work presented in this thesis focuses on one specific approach applied to a set of conceptually similar use-cases, this work has been heavily influenced by broader questions posed by the visual analytics community. Conversely, my experience designing systems that emphasize the union between human and automated reasoning has shaped my view on these questions. In this chapter, I reflect on how the approach taken in this thesis relates to broader questions in visual analytics, and outline key directions for future work that arise from these insights.

7.1 Human vs. Automated Reasoning

All three systems described in this thesis demonstrate how predicate induction can serve as a bridge between automated and human reasoning. At the same time, we see significant variation in the division of labor between the induction engine and the human analyst across use-cases and designs. This relates to a broader question central to the field of visual analytics: which parts of the analytic process should be handled by automation, and which require human reasoning? Predicates offer a useful abstraction for exploring this question. They can be defined either by the system (through the induction engine) or by the user (through interaction with the visual interface), while preserving the same semantics regardless of their origin.

Both approaches to generating predicates impose their own constraints. Predicate induction defines a large search space of possible logical conditions and evaluates them algorithmically using a well-specified objective function. In contrast, predicates defined by user can be afforded more complexity and are not limited by the choice of objective function, but must be generated manually rather than automatically. For systems based on predicates (or similarly flexible abstractions) it is possible that the role of human vs. automated reasoning does not need to be strictly defined. Designer may instead specify a solution in terms of predicates, while deferring choice of how those predicates will be generated to a later implementation stage. By defining the underlying solution in abstract terms, designers can evaluate multiple systems with varying configurations of human and automated reasoning without altering their fundamental approach. Beyond optimizing performance the performance on a particular use-case, adopting this approach may reveal more general insights into the relative strengths of human and automated reasoning.

7.2 One-off Systems vs. Solution Spaces

The goal of visual analytics research is often oriented towards producing standalone tools designed to solve specific analysis tasks. While focus this has resulted in many effective tools across a wide range of use-cases, it comes with a two major

drawbacks. First, these systems often fail to generalize, even to analysis tasks that are structurally similar. This is true for the three systems presented in this thesis. While all three were developed for tasks related to identifying and explaining relevant subsets of the data, none can be applied outside of the specific use-case they were designed for. Second, the emphasis on the design of specific systems can obscure the underlying analysis process required for a given use-case. The motivation for design decisions related to visual encodings, layout, or interaction mechanics is often left implicit. For example, PIXAL's Score View was designed to provide visibility into anomaly scores for collections that may not be faithfully represented by the Bayesian t-test. While this objective could potentially be satisfied through a number of alternate visual encodings, interactions, or automated analysis techniques, the existence of these alternatives are not made explicit. This lack of transparency can have implications both for users, who may misunderstand the purpose of a component of the system in the context of their analysis, or other designers, who may not be able to extract useful generalizations from a given system.

Instead of aiming towards the implementation of a specific system as a single solution to an analysis use-case, researchers can instead work towards identifying broader solution spaces. Unlike one-off systems, solution spaces have the potential to represent generalizable strategies that support multiple possible system implementations for a given use-case. The combination of predicate induction and visual analytics as a general approach could support this paradigm, by allowing researchers to define solutions abstractly as a set of predicates, while the specifics of how predicates are generated and evaluated can vary across implementations.

7.3 VA Systems vs. Analysis Environments

Visual analytic systems are commonly designed as complex, multi-view applications within a self-contained environment. This approach can simplify the design and evaluation of one-off systems, but breaks from the more fluid analysis environments typically used by both analysts and designers. For analysts, a system that can only

be interacted with outside of their typical analysis environment is unlikely to be adopted. For designers, the transition from working in a fluid analysis environment for data exploration to designing a self-contained application can result in key insights being lost in translation. Recent work has sought to bridge this divide using modular visual and interactive components embedded in interactive notebook environments.

Predicates may offer a promising avenue for future work in this area, representing abstract units of reasoning that can be applied both in an exploratory analysis environment and a visual interface, while being directly interpretable and programmable.

Chapter 8

Conclusion

This thesis aims to present predicate induction and visual analytics as a powerful, flexible technique that can address the challenges posed by massive data volumes and ill-defined analysis goals by bridging human and automated reasoning. I contribute three novel systems towards applying this approach to specific use-cases, as well as a formal framework for predicate induction in visual analytics.

First, we design PIXAL, a system for anomaly reasoning. PIXAL uses predicate induction to automatically identify collections of related anomalies, providing multiple plausible explanations for users to consider. With PIXAL's visual interface, users can directly evaluate, compare, and manipulate these explanations, providing the domain context required for effective anomaly reasoning. Then, we design DimBridge, a system for explainable dimensionality reduction. DimBridge allows users to identify patterns in 2D dimensionality reduction results by interacting with a visual interface, and the induction engine generates predicates that match the identified pattern in the original dimensions. Users can then visualize the selected pattern in a manageable subspace of the original dimensions, consisting only of those that are relevant to that particular pattern. The last system we design is VIGOR, a system for graph visualization recommendation. VIGOR uses predicate induction to learn interpretable guidelines for graph visualization techniques from a labeled dataset. Furthermore, we demonstrate how guidelines can be generated directly from the literature, providing a unifying framework for expert and data-driven guidelines.

Finally, I propose a formal framework for predicate induction in visual analytics. Taking a step back from our three systems, I define predicate induction in general terms, describing its key characteristics, use-cases where it can be applied effectively, and limitations. I define the components this type of system within a formal framework, providing a tool for a) understanding existing systems and b) applying this approach to future work. These contributions help address limitations in current visual analytic systems, presenting an approach that uses predicates to bridge human and automated reasoning.

Bibliography

[AAG03] Natalia Andrienko, Gennady Andrienko, and Peter Gatalsky. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, 14(6):503–541, 2003.

[ADRDS+20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[AHKF11] Richard Arias-Hernandez, Linda T Kaastra, and Brian Fisher. Joint action theory and pair analytics: In-vivo studies of cognition and social interaction in collaborative visual analytics. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[AMI16] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md. Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Gener. Comput. Syst.*, 55(C):278–288, feb 2016.

[AMST11] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of time-oriented data*. Springer Science & Business Media, 2011.

[And72] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 28(1):125–136, 1972.

[Ans73] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[Aup07] Michaël Aupetit. Visualizing distortions and recovering topology in continuous projection techniques. *Neurocomputing*, 70(7-9):1304–1330, 2007.

[AVMC+15] Elisa Amorim, Emilio Vital Brazil, Jesús Mena-Chalco, Luiz Velho, Luis Gustavo Nonato, Faramarz Samavati, and Mario Costa Sousa. Facing the high-dimensions: Inverse projection with radial basis functions. *Comput. Graphics*, 48:35–47, 2015.

[BFSO84] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[BKS20] Jinwook Bok, Bohyoung Kim, and Jinwook Seo. Augmenting parallel coordinates plots with color-coded stacked histograms. *IEEE TVCG*, 28(7):2563–2576, 2020.

[BKSS14] Michael Behrisch, Fatih Korkmaz, Lin Shao, and Tobias Schreck. Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In Min Chen, David S. Ebert, and Chris North, editors, *9th IEEE Conference on Visual Analytics Science and Technology, IEEE VAST 2014, Paris, France, October 25-31, 2014*, pages 43–52. IEEE Computer Society, 2014.

[BLBC12] Eli T Brown, Jingjing Liu, Carla E Brodley, and Remco Chang. Disfunction: Learning distance functions interactively. In *2012 IEEE conference on visual analytics science and technology (VAST)*, pages 83–92. IEEE, 2012.

[BM13] Matthew Brehmer and Tamara Munzner. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics*, 19(12):2376–2385, 2013.

[BNH14] Lauren Bradel, Chris North, and Leanna House. Multi-model semantic interaction for text analytics. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 163–172. IEEE, 2014.

[BO23] Leilani Battle and Alvitta Ottley. What exactly is an insight? a literature review. *2023 IEEE Visualization and Visual Analytics (VIS)*, pages 91–95, 2023.

[BoLS] Bureau of Labor Statistics. Healthcare Occupations : Occupational Outlook Handbook: : U.S. Bureau of Labor Statistics — bls.gov. `https://www.bls.gov/ooh/healthcare/home.htm`. [Accessed 29-Mar-2023].

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BSIM14] Matthew Brehmer, Michael Sedlmair, Stephen Ingram, and Tamara Munzner. Visualizing dimensionally-reduced data: Interviews with analysts and a characterization of task sequences. In *Proc. BELIV*, pages 1–8, New York, 2014. ACM.

[BSRG06] Michael Baitaluk, Mayya Sedova, Animesh Ray, and Amarnath Gupta. Biologicalnetworks: visualization and analysis tool for systems biology. *Nucleic acids research*, 34(suppl_2):W466–W471, 2006.

[C+96] William W Cohen et al. Learning rules that classify e-mail. In *AAAI spring symposium on machine learning in information access*, volume 18, page 25. California, 1996.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[CG15] Jesus J Caban and David Gotz. Visual analytics in healthcare–opportunities and research challenges. *Journal of the American Medical Informatics Association*, 22(2):260–262, 2015.

[CGK+] Remco Chang, Mohammad Ghoniem, Robert Kosara, William Ribarsky, Jing Yang, Evan Suma, Caroline Ziemkiewicz, Daniel Kern, and Agus Sudjianto. Wirevis: Visualization of categorical, time-varying data from financial transactions. In *2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 155–162. IEEE.

[CKP+20] Yeounoh Chung, T. Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Automated data slicing for model validation: A big data - ai integration approach. *IEEE Transactions on Knowledge and Data Engineering*, 32:2284–2296, 2020.

[CL+15] Eun Kyoung Choe, Bongshin Lee, et al. Characterizing visualization insights from quantified selfers' personal data presentations. *IEEE computer graphics and applications*, 35(4):28–37, 2015.

[CL18] Jaegul Choo and Shixia Liu. Visual analytics for explainable deep learning. *IEEE computer graphics and applications*, 38(4):84–92, 2018.

[CLZ+17] Nan Cao, Chaoguang Lin, Qiuhan Zhu, Yu-Ru Lin, Xian Teng, and Xidao Wen. Voila: Visual anomaly detection and monitoring with streaming spatiotemporal data. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2017.

[CPC19] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8), 2019.

[CSL+16] N. Cao, C. Shi, S. Lin, J. Lu, Y. Lin, and C. Lin. Targetvue: Visual analysis of anomalous user behaviors in online communication sys-

tems. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):280–289, 2016.

[CT05] Kristin A Cook and James J Thomas. Illuminating the Path: The Research and Development Agenda for Visual Analytics. *National Visualization and Analytics Ctr*, 2005.

[dBD+12] E. P. dos Santos Amorim, E. V. Brazil, J. Daniels, P. Joia, L. G. Nonato, and M. C. Sousa. ilamp: Exploring high-dimensional spacing through backward multidimensional projection. In *Proc. VAST*, pages 53–62, New York, 2012. IEEE.

[DCL+18] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Pai-Shun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Neural Information Processing Systems*, 2018.

[DGS22] Vargha Dadvar, Lukasz Golab, and Divesh Srivastava. Exploring data using patterns: A survey. *Information Systems*, 108:101985, 2022.

[DGW18] Sanjeeb Dash, Oktay Günlük, and Dennis Wei. Boolean decision rules via column generation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4660–4670, Red Hook, NY, USA, 2018. Curran Associates Inc.

[DLM14] Walter Didimo, Giuseppe Liotta, and Fabrizio Montecchiani. Network visualization for financial crime detection. *Journal of Visual Languages & Computing*, 25(4):433–451, 2014.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*, 2017.

[EAS+23] Mateus Espadoto, Gabriel Appleby, Ashley Suh, Dylan Cashman, Mingwei Li, Carlos Scheidegger, Erik W. Anderson, Remco

Chang, and Alexandru C. Telea. Unprojection: Leveraging inverse-projections for visual analytics of high-dimensional data. *IEEE TVCG*, 29(2):1559–1572, 2023.

[EHA+23] Klaus Eckelt, Andreas Hinterreiter, Patrick Adelberger, Conny Walchshofer, Vaishali Dhanoa, Christina Humer, Moritz Heckmann, Christian Steinparz, and Marc Streit. Visual exploration of relationships and structure in low-dimensional embeddings. *IEEE TVCG*, 29(7):3312–3326, 2023.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 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, page 226–231. AAAI Press, 1996.

[Ell18] Geoffrey Ellis. *Cognitive biases in visualizations*. Springer, 2018.

[EMK+21] Mateus Espadoto, Rafael M. Martins, Andreas Kerren, Nina S. T. Hirata, and Alexandru C. Telea. Toward a quantitative survey of dimension reduction techniques. *IEEE TVCG*, 27(3):2153–2173, 2021.

[GAW+11] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D Hansen, and Jonathan C Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[GFC04] Mohammad Ghoniem, J-D Fekete, and Philippe Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE symposium on information visualization*, pages 17–24. Ieee, 2004.

[GGZL15] Hua Guo, Steven R Gomez, Caroline Ziemkiewicz, and David H Laidlaw. A case study using visualization interaction logs and insight met-

rics to understand how analysts arrive at insights. *IEEE transactions on visualization and computer graphics*, 22(1):51–60, 2015.

[GHS12] Sumit Gulwani, William R Harris, and Rishabh Singh. Spreadsheet data manipulation using examples. *Communications of the ACM*, 55(8):97–105, 2012.

[GRS+18] John Goodall, Eric Ragan, Chad Steed, Joel Reed, G. Richardson, Kelly Huffer, Robert Bridges, and Jason Laska. Situ: Identifying and explaining suspicious behavior in networks. *IEEE TVCG*, PP:1–1, 08 2018.

[GS09] John R Goodall and Mark Sowul. Viassist: Visual analytics for cyber defense. In *2009 IEEE conference on technologies for homeland security*, pages 143–150. IEEE, 2009.

[GSC16] David Gotz, Shun Sun, and Nan Cao. Adaptive contextualization: Combating bias during high-dimensional visualization and data selection. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 85–95, 2016.

[GZ09] David Gotz and Michelle X Zhou. Characterizing users' visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, 2009.

[HDO+98] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[HE11] Christopher Healey and James Enns. Attention and visual memory in visualization and computer graphics. *IEEE TVCG*, 18(7):1170–1188, 2011.

[HGN00] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter*, 2(1):58–64, 2000.

[HHHWP11] Nathaniel E Helwig, Sungjin Hong, Elizabeth T Hsiao-Wecksler, and John D Polk. Methods to temporally align gait cycle data. *J. Biomech.*, 44(3):561–566, 2011.

[HHW22] Nathaniel Helwig and Elizabeth Hsiao-Wecksler. Multivariate Gait Data. UCI Machine Learning Repository, 2022.

[HKPC18] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693, 2018.

[HLD02] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. In *Proc. InfoVis*, pages 127–130, New York, 2002. IEEE.

[HSMHW16] Nathaniel E Helwig, K Alex Shorter, Ping Ma, and Elizabeth T Hsiao-Wecksler. Smoothing spline analysis of variance models: A new tool for the analysis of cyclic biomechanical data. *J. Biomech.*, 49(14):3216–3222, 2016.

[HW13] Julian Heinrich and Daniel Weiskopf. State of the Art of Parallel Coordinates. In M. Sbert and L. Szirmay-Kalos, editors, *Eurographics 2013 - State of the Art Reports*, pages 95–116. The Eurographics Association, 2013.

[Ins85] Alfred Inselberg. The plane with parallel coordinates. *Visual Comput.*, 1(2):69–91, 1985.

[JAL+22] Hyeon Jeon, Michael Aupetit, Soohyun Lee, Hyung-Kwon Ko, Youngtaek Kim, and Jinwook Seo. Distortion-aware brushing for interac-

tive cluster analysis in multidimensional projections. *arXiv preprint arXiv:2201.06379*, 2022.

[KAL+21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, pages 852–863, Red Hook, NY, 2021. Curran Associates Inc.

[KB19] Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *Nat. Commun.*, 10(5416), 2019.

[KBGE09] Steffen Koch, Harald Bosch, Mark Giereth, and Thomas Ertl. Iterative integration of visual insights during patent search and analysis. In *2009 IEEE Symposium on Visual Analytics Science and Technology*, pages 203–210, 2009.

[KEC06] René Keller, Claudia M Eckert, and P John Clarkson. Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? *Information Visualization*, 5(1):62–76, 2006.

[KEV+18] Bum Chul Kwon, Ben Eysenbach, Janu Verma, Kenney Ng, Christopher De Filippi, Walter F. Stewart, and Adam Perer. Clustervision: Visual supervision of unsupervised clustering. *IEEE TVCG*, 24(1):142–151, 2018.

[KKKG14] Karthik Kambatla, Giorgos Kollias, Vipin Kumar, and Ananth Grama. Trends in big data analytics. *Journal of Parallel and Distributed Computing*, 74(7):2561–2573, 2014. Special Issue on Perspectives on Parallel and Distributed Processing.

[KLSH04] Yufeng Kou, Chang-Tien Lu, S. Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *IEEE International Con-*

ference on Networking, Sensing and Control, 2004, volume 2, pages 749–754 Vol.2, 2004.

[KMS+08] Daniel A Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. Visual analytics: Scope and challenges. In *Visual data mining*, pages 76–90. Springer, 2008.

[KMSZ06] D.A. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Tenth International Conference on Information Visualisation (IV'06)*, pages 9–16, 2006.

[KPHH11] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372, 2011.

[KR95] Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995.

[KW78] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Sage, Thousand Oaks, CA, 1978.

[LA11] Sylvain Lespinats and Michaël Aupetit. Checkviz: Sanity check and topological clues for linear and non-linear mappings. *CGF*, 30(1):113–125, 2011.

[LAB+23] Harry Li, Gabriel Appleby, Camelia Daniela Brumar, Remco Chang, and Ashley Suh. Knowledge graphs in practice: Characterizing their users, challenges, and visualization opportunities. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[LGM+17] Roger A Leite, Theresia Gschwandtner, Silvia Miksch, Simone Kriglstein, Margit Pohl, Erich Gstrein, and Johannes Kuntner. Eva: Visual analytics to identify fraudulent events. *IEEE transactions on visualization and computer graphics*, 24(1):330–339, 2017.

[LGM+18] Roger A Leite, Theresia Gschwandtner, Silvia Miksch, Erich Gstrein, and Johannes Kuntner. Visual analytics for event detection: Focusing on fraud. *Visual Informatics*, 2(4):198–212, 2018.

[LH14] Zhicheng Liu and Jeffrey Heer. The effects of interactive latency on exploratory visual analysis. *IEEE TVCG*, 20(12):2122–2131, 2014.

[LL17] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NIPS*, 2017.

[Llo82] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LLY+21] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[LMW+17] S. Liu, D. Maljovec, B. Wang, P. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE TVCG*, 23(3):1249–1268, 2017.

[LPK20] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[LWCC18] Hongsen Liao, Yingcai Wu, Li Chen, and Wei Chen. Cluster-based visual abstraction for multivariate scatterplots. *IEEE TVCG*, 24(9):2531–2545, 2018.

[LYC10] Zicheng Liao, Yizhou Yu, and Baoquan Chen. Anomaly detection in gps data based on visual analytics. In *2010 IEEE Symposium on Visual Analytics Science and Technology*, pages 51–58. IEEE, 2010.

[MAR+24] Brian Montambault, Gabriel Appleby, Jen Rogers, Camelia D Brumar, Mingwei Li, and Remco Chang. Dimbridge: Interactive explanation of visual patterns in dimensionality reductions with predicate logic. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[MBBC22a] Brian Montambault, Camelia D. Brumar, Michael Behrisch, and Remco Chang. Pixal: Anomaly reasoning with visual analytics, 2022.

[MBBC22b] Brian Montambault, Camelia D. Brumar, Michael Behrisch, and Remco Chang. Pixal: Anomaly reasoning with visual analytics, 2022.

[MCB20] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 417–431. Springer, 2020.

[MCMT14] Rafael Messias Martins, Danilo Barbosa Coimbra, Rosane Minghim, and A.C. Telea. Visual analysis of dimensionality reduction quality for parameterized projections. *Comput. Graphics*, 41:26–42, 2014.

[MGO21] Shayan Monadjemi, Roman Garnett, and Alvitta Ottley. Competing models: Inferring exploration patterns and information relevance via bayesian model selection. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):412–421, 2021.

[MHSG18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *JOSS*, 3(29):861, 2018.

[MJEG21] Wilson E. Marcílio-Jr, Danilo M. Eler, and Rogério E. Garcia. Contrastive analysis for scatterplot-based representations of dimensionality reduction. *Comput. Graphics*, 101:46–58, 2021.

[MKN+07] Florian Mansmann, Daniel A Keim, Stephen C North, Brian Rexroad, and Daniel Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1105–1112, 2007.

[MQB19] Yao Ming, H. Qu, and E. Bertini. Rulematrix: Visualizing and understanding classifiers with rules. *IEEE Transactions on Visualization and Computer Graphics*, 25:342–352, 2019.

[MRH+09] Ross Maciejewski, Stephen Rudolph, Ryan Hafen, Ahmad Abusalah, Mohamed Yakout, Mourad Ouzzani, William S Cleveland, Shaun J Grannis, and David S Ebert. A visual analytics approach to understanding spatiotemporal hotspots. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):205–220, 2009.

[MSK+19] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *ArXiv*, abs/1901.04592, 2019.

[Mus23] M. Mustafa. Diabetes prediction dataset. `https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset`, 2023. Accessed: March 19 2024.

[MVT16] Belgin Mutlu, Eduardo Veas, and Christoph Trattner. Vizrec: Recommending personalized visualizations. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(4):1–39, 2016.

[MWN+18] Dominik Moritz, Chenglong Wang, Greg L Nelson, Halden Lin, Adam M Smith, Bill Howe, and Jeffrey Heer. Formalizing visual-

ization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics*, 25(1):438–448, 2018.

[NA19] Luis Gustavo Nonato and Michaël Aupetit. Multidimensional projection for visual analytics: Linking techniques with distortions, tasks, and layout enrichment. *IEEE TVCG*, 25(8):2650–2673, 2019.

[NHT+16] Steven Noel, Eric Harley, Kam Him Tam, Michael Limiero, and Matthew Share. Cygraph: graph-based analytics and visualization for cybersecurity. In *Handbook of statistics*, volume 35, pages 117–167. Elsevier, 2016.

[NMSL19] Carolina Nobre, Miriah Meyer, Marc Streit, and Alexander Lex. The state of the art in visualizing multivariate networks. In *Computer Graphics Forum*, volume 38, pages 807–832. Wiley Online Lib., 2019.

[NNG+23] Andrew Novick, Quan Nguyen, Roman Garnett, Eric Toberer, and Vladan Stevanović. Simulating high-entropy alloys at finite temperatures: An uncertainty-based approach. *Phys. Rev. Mater.*, 7(6):063801, 2023.

[OYC15] Alvitta Ottley, Huahai Yang, and Remco Chang. Personality as a predictor of user strategy: How locus of control affects search strategies on tree visualizations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3251–3254, 2015.

[PC05] Peter Pirolli and Stuart Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of international conference on intelligence analysis*, volume 5, pages 2–4. McLean, VA, USA, 2005.

[Pea01] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[PP07] Animesh Patcha and Jung-Min Jerry Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Networks*, 51:3448–3470, 2007.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QRD+22] Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Nesreen K Ahmed. Personalized visualization recommendation. *ACM Transactions on the Web (TWEB)*, 16(3):1–47, 2022.

[RC18a] Jonas Ranstam and Jonathan A Cook. Lasso regression. *Journal of British Surgery*, 105(10):1348–1348, 2018.

[RC18b] Blagoj Ristevski and Ming Chen. Big data analytics in medicine and healthcare. *Journal of Integrative Bioinformatics*, 15(3):20170030, 2018.

[RKDK15] Sudip Roy, Arnd Christian König, Igor Dvorkin, and Manish Kumar. Perfaugur: Robust diagnostics for performance anomalies in cloud services. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1167–1178, 2015.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell,

Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proc. ICML*, pages 8748–8763. PMLR, 2021.

[RLS+19] Richard C. Roberts, Robert S. Laramee, Gary A. Smith, Paul Brookes, and Tony D'Cruze. Smart brushing for parallel coordinates. *IEEE TVCG*, 25(3):1575–1590, 2019.

[RS14] Sudeepa Roy and Dan Suciu. A formal approach to finding explanations for database queries. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, page 1579–1590, New York, NY, USA, 2014. Association for Computing Machinery.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

[RSS+09] Jeffrey Rouder, Paul Speckman, Dongchu Sun, Richard Morey, and G.J. Iverson. Bayesian t test for accepting and rejecting the null hypothesis. *Psychonomic bulletin & review*, 16:225–37, 05 2009.

[SDMT15] Julian Stahnke, Marian Dörk, Boris Müller, and Andreas Thom. Probing projections: Interaction techniques for interpreting arrangements and errors of dimensionality reductions. *IEEE TVCG*, 22(1):629–638, 2015.

[SG17] Alper Sarikaya and Michael Gleicher. Scatterplots: Tasks, data, and designs. *IEEE TVCG*, 24(1):402–412, 2017.

[SHGF16] Danielle Albers Szafir, Steve Haroz, Michael Gleicher, and Steven Franconeri. Four types of ensemble coding in data visualizations. *JOV*, 16(5):11–11, 2016.

[SMT13] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE TVCG*, 9:2634–2643, 2013.

[SND05] Purvi Saraiya, Chris North, and Karen Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE transactions on visualization and computer graphics*, 11(4):443–456, 2005.

[SPRHW08] K Alex Shorter, John D Polk, Karl S Rosengren, and Elizabeth T Hsiao-Wecksler. A new approach to detecting asymmetries in gait. *Clin. Biomech.*, 23(4):459–467, 2008.

[SR18] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 8(4):e1249, 2018.

[SSG11] Hadi Shiravi, Ali Shiravi, and Ali A Ghorbani. A survey of visualization systems for network security. *IEEE Transactions on visualization and computer graphics*, 18(8):1313–1329, 2011.

[SSS+14] Dominik Sacha, Andreas Stoffel, Florian Stoffel, Bum Chul Kwon, Geoffrey Ellis, and Daniel A. Keim. Knowledge generation model for visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1604–1613, 2014.

[SZS+16] Dominik Sacha, Leishi Zhang, Michael Sedlmair, John A Lee, Jaakko Peltonen, Daniel Weiskopf, Stephen C North, and Daniel A Keim.

Visual interaction with dimensionality reduction: A structured literature analysis. *IEEE TVCG*, 23(1):241–250, 2016.

[tab] Anomaly Detection. `https://exchange.tableau.com/products/1066`. Accessed: 2025-07-17.

[TC06] James J Thomas and Kristin A Cook. A visual analytics agenda. *IEEE computer graphics and applications*, 26(1):10–13, 2006.

[TSZ+18] Jun Tao, Lei Shi, Zhou Zhuang, Congcong Huang, Rulei Yu, Purui Su, Chaoli Wang, and Yang Chen. Visual analysis of collective anomalies through high-order correlation graph. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 150–159. IEEE, 2018.

[Tuk80] John W. Tukey. We need both exploratory and confirmatory. *The American Statistician*, 34(1):23–25, 1980.

[vdMH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9(86):2579–2605, 2008.

[VML+ed] Sjoerd Vink, Brian Montambault, Mingwei Li, Remco Chang, and Michael Behrisch. Graph visualization guidelines as learnable predicates. *International Conference on Computer Graphics, Interaction, Visualization Theory and Applications*, 2026 (accepted).

[Wea] Chris Weaver. Conjunctive Visual Forms. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), nov.

[WFG+12] Johan Wagemans, Jacob Feldman, Sergei Gepshtein, Ruth Kimchi, James R Pomerantz, Peter A Van der Helm, and Cees Van Leeuwen. A century of gestalt psychology in visual perception: Ii. conceptual and theoretical foundations. *Psychological Bulletin*, 138(6):1218–1252, 2012.

[WHW+24] Kai Wang, Shuqi He, Wenlu Wang, Jinbei Yu, Yu Liu, and Lingyun Yu. Chordination: Evaluating visual design choices in chord diagrams for network data. *arXiv preprint arXiv:2408.02268*, 2024.

[Wil11] Leland Wilkinson. The grammar of graphics. In *Handbook of computational statistics: Concepts and methods*, pages 375–414. Springer, 2011.

[WM13] Eugene Wu and Samuel Madden. Scorpion: Explaining away outliers in aggregate queries. *Proc. VLDB Endow.*, 6(8):553–564, June 2013.

[WMR17] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[WPB+19] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.

[WS92] Christopher Williamson and Ben Shneiderman. The dynamic homefinder: Evaluating dynamic queries in a real-estate information exploration system. In *Proc. of the 15th ACM SIGIR Conf. on Research and Development in IR*, pages 338–346, 1992.

[WVJ16] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. *Distill*, 2016.

[XGH06] Ling Xiao, John Gerth, and Pat Hanrahan. Enhancing visual analysis of network traffic using a knowledge representation. In *2006 IEEE symposium on visual analytics science and technology*, pages 107–114. IEEE, 2006.

[XWY+20] K. Xu, Y. Wang, L. Yang, Y. Wang, B. Qiao, S. Qin, Y. Xu, H. Zhang, and H. Qu. Clouddet: Interactive visual analysis of anomalous per-

formances in cloud computing systems. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1107–1117, 2020.

[XXM19] C. Xie, W. Xu, and K. Mueller. A visual analytics framework for the detection of anomalous call stack trees in high performance computing applications. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):215–224, 2019.

[YKSJ07] Ji Soo Yi, Youn ah Kang, John Stasko, and J.A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[YNM16] Dong Young Yoon, Ning Niu, and Barzan Mozafari. Dbsherlock: A performance diagnostic tool for transactional databases. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 1599–1614, New York, NY, USA, 2016. Association for Computing Machinery.

[YRS17] Hongyu Yang, C. Rudin, and Margo I. Seltzer. Scalable bayesian rule lists. In *ICML*, 2017.

[YSHZ19] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[YST+18] Jia Yan, Lei Shi, Jun Tao, Xiaolong Yu, Zhou Zhuang, Congcong Huang, Rulei Yu, Purui Su, Chaoli Wang, and Yang Chen. Visual analysis of collective anomalies using faceted high-order correlation graphs. *IEEE transactions on visualization and computer graphics*, 26(7):2517–2534, 2018.

[ZCS19] Wei Zong, Yang-Wai Chow, and Willy Susilo. Interactive three-dimensional visualization of network intrusion detection data for machine learning. *Future Generation Computer Systems*, 102, 08 2019.

[Zin10] Andrei Zinovyev. Data visualization in political and social sciences. *arXiv preprint arXiv:1008.1188*, 2010.

[ZK08] Caroline Ziemkiewicz and Robert Kosara. The shaping of information by visual metaphors. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1269–1276, 2008.

[ZWL+17] Tianye Zhang, Xumeng Wang, Zongzhuang Li, Fangzhou Guo, Yuxin Ma, and Wei Chen. A survey of network anomaly visualization. *Science China Information Sciences*, 60(12):121101, 2017.

[ZZZK18] Emanuel Zgraggen, Zheguang Zhao, Robert Zeleznik, and Tim Kraska. Investigating the effect of the multiple comparisons problem in visual analysis. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–12, 2018.