Università Di Pisa

Computer science department

Master programme in Data Science and Business Informatics

Interactive visual interfaces for synthetic neighbourhoods and rule based explanations

Supervisor:
Salvatore Rinzivillo

Candidate:
Alessandro Carella

Academic Year 2024/2025

Abstract

As artificial intelligence systems make increasingly consequential decisions in critical domains such as healthcare, finance, and others, AI models need transparent explanations they can understand and trust. The evolving field of eXplainable AI tackles this need, aiming to make those explanations cognitively accessible and confirmable by human users. The objective of this thesis is the development of an interactive visualization that reshapes complex eXplainability methods' outputs into an easier to digest format through intuitive visual interfaces. The proposed system targets local XAI methods that provide explanations through the generation of a neighborhood around the explained instance, and a surrogate model from which the rules are extracted. A scatter plot projection shows how the model organizes instances in the projected 2D space, and a parallel and interconnected decision tree diagram reveals the logical rules underlying predictions to the user. The two visualizations are conceptualized to empower the users to explore the results via an interactive dialogue between spatial intuition and logical reasoning. The system offers multiple tree visualization options tailored to different needs, which is the result of both initial conceptualization and its expansion based on the surveyed literature. The proposed use cases demonstrate how the system supports different iterative exploration workflows where users can adjust neighborhood generation parameters, regenerate explanations, and validate findings across different instances. This work tries to bridge the gap between powerful explanation algorithms and human understanding, providing tools to interpret, validate, and communicate machine learning decisions. The system, in the current state, uses the LORE_{sa} eXplainability method to generate the explanations, but further implementations using other state of the art methods, such as LIME and SHAP, would be easily implementable.

Contents

Chapter 1

Introduction

As artificial intelligence systems become increasingly integrated into critical decision-making processes across healthcare, finance, justice, and other high-stakes domains, the need for transparent and interpretable explanations of machine learning predictions has become indisputable. The challenge lies not merely in the algorithmic generation of explanations, but in their effective communication to human users who must understand, validate, and ultimately trust these systems. This thesis addresses this challenge by developing an interactive visualization framework specifically designed for local eXplainability methods that generate synthetic neighborhoods around the chosen instance and extract rules through surrogate models trained on the aforementioned neighborhood. The system focuses on transforming the complex outputs of those explainability methods into intuitive visual representations that support exploration, confirmation, and understanding of the methods' explanations. The proposed framework enables users to iteratively explore how local decision boundaries are captured by synthetic neighborhoods and how these neighborhoods translate into interpretable decision rules.

The design philosophy underlying this work recognizes that effective explanation communication requires more than static visualization of algorithmic outputs. It demands interactive systems that support diverse analytical workflows, accommodate different cognitive preferences, and enable bidirectional exploration between complementary representational. The proposed system implements this philosophy through parallel presentation of two interconnected views: a two-dimensional spatial projection that reveals the distribution and quality of synthetically generated neighborhood instances, and hierarchical tree visualizations that expose the logical structure of extracted decision rules. The coordination mechanism linking these views enables users to transition between spatial intuition and symbolic reasoning, validating explanation quality through visual confirmation of correspondence between instance distributions and decision boundaries.

The research begins by establishing theoretical foundations in Chapter 2, where

eXplainable Artificial Intelligence concepts are explored through a comprehensive examination of XAI motivations, the distinction between explainability and interpretability, and a detailed analysis of prevalent algorithms like Local Interpretable Model-Agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP). This foundation extends to visualization techniques essential for XAI, including dimensionality reduction methods such as Principal Component Analysis, Multidimensional Scaling, t-distributed Stochastic Neighbor Embedding, and Uniform Manifold Approximation and Projection, alongside decision tree visualization approaches drawn from existing literature. The chapter surveys the landscape of XAI visual analytics tools, categorizing them into comprehensive frameworks, local explanation analysis tools, and surrogate model visualization systems. Building upon this foundation, Chapter 3 articulates the specific research problem through a detailed examination of the $LORE_{sa}$ method, which employs genetic algorithms for neighborhood generation to optimize synthetic instance placement relative to decision boundaries. The chapter analyzes $LORE_{sa}$'s multi-neighborhood ensemble strategy and rule extraction processes while critically identifying fundamental limitations in current textual presentation formats, including lack of spatial context, inability to validate decision boundary correspondence, and absence of mechanisms for exploring alternative decision paths.

The system's development is detailed across Chapters 4, 5, and 6, documenting the design evolution from initial conceptual framework through final implementation and practical demonstration. Chapter 4 traces the conceptual development emphasizing representational alignment between spatial neighborhood analysis and hierarchical rule structure, documenting iterative refinement of three primary visualization components through progressive enhancement of interaction mechanisms and visual encoding strategies. This evolves into the architecture presented in Chapter 5, which details the progressive disclosure workflow encompassing seven configuration stages alongside coordinated visualization systems that link spatial scatter plot projections with three alternative tree layouts through bidirectional coordination mechanisms. The practical utility emerges in Chapter 6 through two detailed use cases: a pedagogical scenario demonstrating XAI concept introduction using the Iris dataset, and a professional workflow involving housing price prediction explanation.

Finally, Chapter 7 synthesizes the system's unique contributions in coordinated spatial-symbolic exploration while identifying future enhancements across four categories: enhanced visual encoding, adaptive interaction mechanisms, alternative analytical perspectives, and integration considerations for maintaining system coherence.

Chapter 2

State of the art

2.1 eXplainable AI (XAI)

Machine learning (Machine Learning) techniques have achieved remarkable performance across many domains, driven by advances in computing power and technologies [1, 2, 3]. Neural networks, as a prominent Machine Learning sub-field, have demonstrated exceptional capability in identifying complex patterns within high-dimensional datasets and delivering high prediction accuracy across numerous applications [1, 4]. However, these sophisticated models present a fundamental challenge: their complex architectures makes them difficult, if not impossible, to interpret and understand directly. Neural networks are commonly referred to as "black-box" models because their internal workings and decision-making mechanisms remain opaque to human observers [1].

2.1.1 Motivations for eXplainable AI

This opacity reveals one of the most critical challenges in modern Machine Learning systems: the need for transparency and explainability [1, 5]. End-users, particularly in sensitive domains such as healthcare, transportation, defense, and finance, require a deep understanding of how classifiers generate predictions, as decision-making in these contexts often carries significant consequences [1]. The ability to explain how predictions are formulated by explaining the underlying mechanisms can substantially enhance trust in Machine Learning models.

To address this need, the field of interpretable algorithms has evolved rapidly, focusing on clarifying the inner workings of black-box models [1, 6]. A pivotal development in this area has been the emergence of **eXplainable Artificial Intelligence (XAI)** [7]. According to the Defense Advanced Research Projects Agency (DARPA) [7], XAI is defined as "a suite of machine learning techniques that

enables human users to understand, appropriately trust, and effectively manage the emerging generation of artificially intelligent partners".

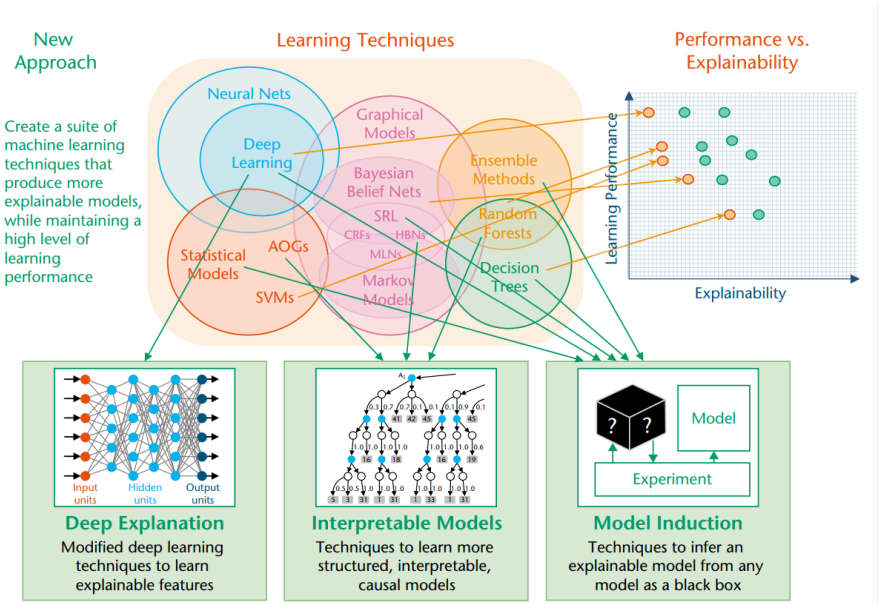


Figure 2.1: Strategies for developing explainable models showing the relationship between learning techniques, performance vs explainability trade-offs, and three main XAI approaches: Deep Explanation, Interpretable Models, and Model Induction.

The DARPA XAI program identifies three primary strategies for developing explainable models, as illustrated in Figure 2.1. These approaches represent different methodological directions for addressing the fundamental trade-off between model performance and explainability.

2.1.2 Defining Explainability vs. Interpretability

While interpretability and explainability are frequently used interchangeably within the Machine Learning community, subtle yet important distinctions exist between these concepts. Biran and Cotton's [8] define interpretability as "the degree to which an observer can understand the cause of a decision". From a Machine Learning perspective, interpretability encompasses understanding how decisions or predictions are generated by ML algorithms through logical reasoning. Explainability, conversely, relates more closely to the internal operational mechanisms of black-box models [1].

XAI aims to reveal the internal functioning of black-box models and the underlying reasoning of their decisions, through various methodological approaches. While domain experts inexperienced in Machine Learning typically seek understanding

Table 2.1: Classification of the most popular XAI techniques in explaining neural networks.

XAI method	Explanation level		Implementation level		Model dependency	
	Global	Local	Intrinsic	Post hoc	Agnostic	Specific
ANCHORS [12]		✓		✓	✓	
LIME [13]	✓	✓		✓	✓	
SHAP [14]		✓		✓	✓	
LRP [15]	✓	✓		✓	✓	
Grad-CAM [16]		✓		✓	✓	
Saliency Maps [17]		✓		✓	✓	
Integrated Gradients [18]		✓		✓	✓	
DeepLIFT [19]		✓		✓	✓	
Bayesian Rule Lists [20]	✓		✓			✓
Distillation [21]	✓			✓	✓	
GAM [22]	✓		✓			✓
Mean Decrease Impurity [23]	✓	✓	✓			✓
CAM [24]		✓		✓	✓	

through reasoning and cause-effect relationships for specific decisions, Machine Learning scientists focus on the internal operational mechanisms of models, attempting to understand how individual components contribute to particular predictions [1].

2.1.3 XAI Taxonomy and Approaches

Despite the growing importance of XAI, no standardized consensus exists regarding its precise definition [9, 10, 11]. This definitional diversity stems from varying domain-specific applications, use-case scenarios, and research objectives [1].

XAI methods can be categorized along three primary dimensions, as shown in Table 2.1:

Explanation Level: This dimension distinguishes between *global explanations*, which focus on the overall model behavior and decision-making mechanisms, and *local explanations*, which untangle model decisions for individual instances or subpopulations [1]. Global methods such as Bayesian Rule Lists (BRL) [20], Generalized Additive Models (GAM), and model distillation techniques provide comprehensive model-wide insights. In contrast, local methods including Local Interpretable Model-Agnostic Explanations (LIME) [13], SHapley Additive exPlanations (SHAP) [14], Gradient-weighted Class Activation Mapping (Grad-CAM), and Deep Learning Important FeaTures (DeepLIFT) focus on instance-specific explanations.

Implementation Level: This categorization distinguishes between *intrinsic* and *post-hoc* explanations [1]. Intrinsic explanations, such as Bayesian Rule Lists and Mean Decrease Impurity (MDI), are provided directly by the model through parameters, decision trees, or rule structures. Post-hoc explanations, on the

other side, analyze pre-trained models to reveal internal functioning and decision mechanisms after the training process completion.

Model Dependency: Methods are classified as either *model-agnostic* (applicable across different model types) or *model-specific* (tailored to particular architectures) [25].

2.1.4 Core Challenges and Goals

The development of eXplainable AI is driven by interconnected objectives that address both technical and human-centered concerns. At its foundation, XAI tires to increase **trust and transparency** in AI systems by providing clear insights into their decision-making processes. This transparency is particularly crucial in high-stakes domains such as healthcare, finance, and criminal justice, where algorithmic decisions can have profound consequences on individuals' lives and where accountability is not merely desirable but legally and ethically necessary.

Beyond increasing trust, XAI must fill the gap between internal mechanisms of complex machine learning models and human comprehension. Effective explanations need to be tailored to diverse audiences, accommodating the needs of technical experts who may require algorithmic insights as well as non-specialist end-users who benefit from intuitive, accessible descriptions. This challenge of achieving multi-level interpretability represents one of the field's most persistent obstacles, as explanation systems must balance technical accuracy with cognitive accessibility.

Moreover, XAI should deliver **actionable insights** that empower users to understand not only *what* decision a model has reached, but *why* it arrived at that particular conclusion. This deeper level of understanding enables informed decision-making, allowing users to appropriately trust, question, or override model predictions based on contextual factors that may not be captured in the training data. Such actionability extends beyond passive understanding to active engagement with AI systems, transforming users from mere consumers of predictions into evaluators of algorithmic reasoning.

From a technical perspective, XAI techniques serve a role in **models improvement**. By revealing the internal logic of models, explanation methods facilitate debugging, bias detection, and performance enhancement, exposing potential weaknesses, unexpected behaviours, or antagonistic correlations that might otherwise remain hidden. This diagnostic capability is essential for developing robust and fair AI systems that can be safely deployed in real-world applications.

The field continues to evolve rapidly, with ongoing research addressing fundamental challenges including explanation consistency, computational efficiency, and the development of standardized evaluation metrics for explanation quality [25]. As XAI matures, the integration of visual analytics techniques with sophisticated

explanation methods promises to deliver more effective human-AI collaborative systems.

2.1.5 Most common XAI Algorithm

The field of eXplainable Artificial Intelligence methods has been largely defined by two groundbreaking approaches: LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations). These techniques target the same fundamental problem, how do we understand what a complex model is thinking when it makes a specific prediction? These methods represent distinct approaches to the task of instance level explainability, each proposing unique mechanisms that have influenced the development of subsequent explanation techniques.

LIME: Local Interpretable Model-Agnostic Explanations

LIME, introduced by Ribeiro et al.[13], operates on the principle that complex models can be explained locally through simple, interpretable surrogate models [13, 25]. The core mechanism of LIME involves generating a synthetic neighborhood, around the instance to be explained, through random perturbation, where samples are drawn both in the vicinity of the target instance and further away[13, 25]. This neighborhood generation exploits a proximity measure $\pi_x(z)$, between an instance z to the explained instance x, to capture locality around x, ensuring that nearby instances receive greater emphasis in the explanation process.

The method samples instances around the target instance x by drawing nonzero elements at random, creating a perturbed dataset $\{z \in \mathbb{R}^d\}$ that is subsequently labeled by the black-box model $b(z)$ [13, 25]. LIME then trains a sparse linear model on these perturbed samples, with the explanation consisting of feature importance weights from this local linear approximation [13, 26].

Visual Affordances: LIME's explanations are typically presented through *bar charts* that display feature importance values, clearly distinguishing between positive and negative contributions to the prediction [1, 13]. These visualizations allow end-users to easily observe both supportive and contradictory feature influences, making the local decision boundary accessible through intuitive graphical representations [1]. One can observe two examples of LIME's explanations in Figure 2.2.

SHAP: SHapley Additive exPlanations

SHAP, developed by Lundberg et. al [14], takes a fundamentally different approach rooted in cooperative game theory. The method computes feature importance through Shapley values, treating features as players in a cooperative game and

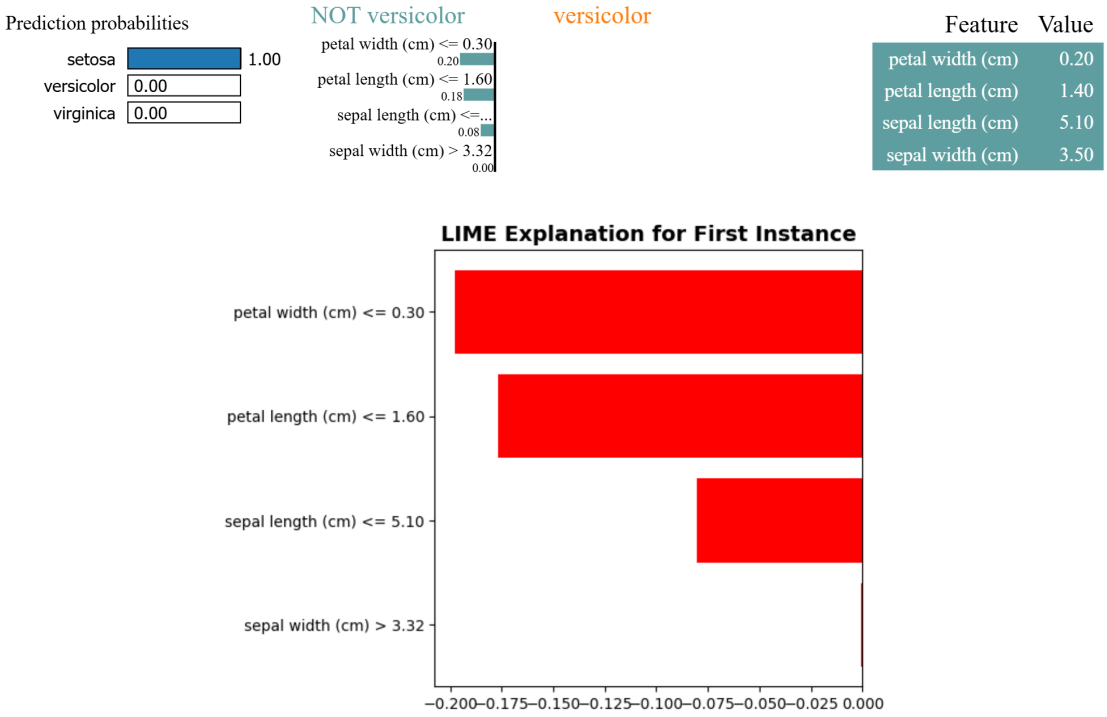


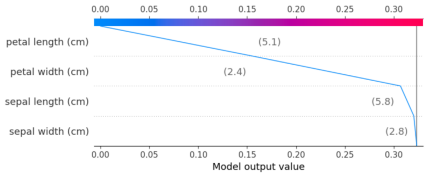
Figure 2.2: Examples of LIME explanations

calculating their marginal contributions to the prediction outcome [1]. SHAP explanations are *additive feature attributions* that satisfy the mathematical formulation: $g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i$, where $z' \in \{0,1\}^M$, $\phi_i \in \mathbb{R}$ represent effects assigned to each feature, and M denotes the number of simplified input features [14, 25].

The method maintains three crucial properties: (i) *local accuracy*, ensuring $f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$, meaning that the explanation model $g(x')$ matches the original model $f(x)$ when $x = h_x(x')$; (ii) *missingness*, where features with $x_i = 0$ have no attributed impact; and (iii) *consistency*, guaranteeing that increased marginal feature contributions result in increased SHAP values [14, 25]. SHAP provides both local explanations for individual instances and global insights through collective SHAP value analysis [25].

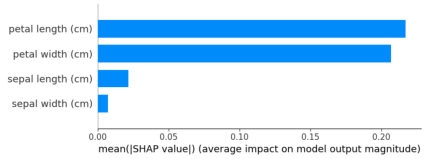
Visual Affordances: SHAP offers diverse visualization techniques including *bar plots* that display feature importance rankings, *beeswarm plots* that combine feature importance with feature effects for each sample, *decision plots* showing the cumulative effect of features on individual predictions, *force plots* that show how each feature pushes the prediction away from a base value, *heatmap plots* for visualizing feature effects across multiple samples, *image plots* designed specifically for computer vision models to highlight relevant pixels, *scatter plots* that reveal the relationship between feature values and their SHAP values, *text plots* for natural language processing models to highlight important words or tokens, *violin summary*

plots that show the distribution of SHAP values for each feature, and *waterfall plots* displaying sequential feature contributions that show how each feature pushes the prediction away from a base value [27]. Some of those visualizations, on the same instance of the iris dataset as the one used in LIME's Figures 2.2, can be observed in Figure 2.3

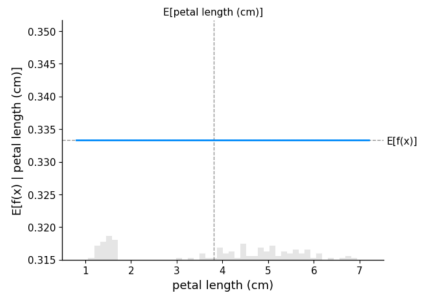


(a) SHAP decision plot showing the cumulative impact of features on model prediction

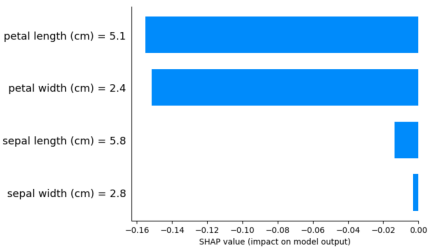
(b) SHAP force plot displaying individual prediction explanation with features contributing positively (red) and negatively (blue) to push the prediction above or below the base value



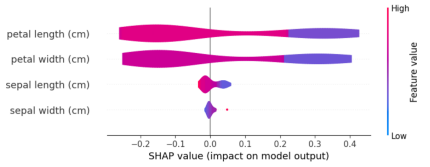
(c) SHAP summary plot showing feature importance and impact distribution



(d) Partial dependence plot showing model output vs feature values



(e) SHAP bar plot showing individual feature contributions



(f) SHAP violin plot showing distribution and impact of feature contributions on model predictions

Figure 2.3: Some of SHAP' visualizations showing different aspects of model interpretability on the iris dataset instance 'sepal length (cm)': 5.8, 'sepal width (cm)': 2.8, 'petal length (cm)': 5.1, 'petal width (cm)': 2.4.

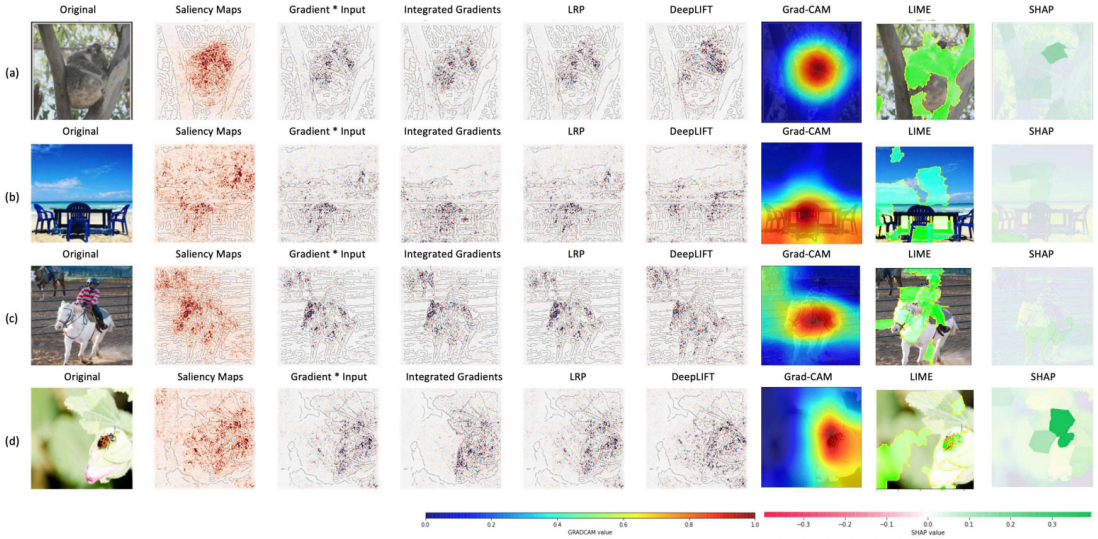


Figure 2.4: Evaluation different gradient-based and perturbation-based techniques in this figure. LIME and SHAP uses segmented superpixels to understand feature importance, while gradient based saliency maps, Integrated Gradients, LRP, DeepLIFT, and Grad-CAM use backpropagation based feature importance in a pixel level [11].

Comparative Analysis of LIME and SHAP

The comparison between LIME and SHAP reveals distinct trade-offs across multiple evaluation dimensions, as demonstrated in Figure 2.4 which shows how both methods perform on real image classification tasks compared to gradient-based approaches:

Fidelity: As shown in Table 2.2, empirical evaluations demonstrate that LIME generally exhibits higher fidelity values compared to SHAP, particularly on adult dataset classifications [25]. LIME achieves fidelity scores above 90% across most model types, while SHAP shows notably lower performance on ensemble models such as CatBoost, with values dropping to 0.777 for CatBoost on adult data and 0.670 on German credit data [25]. This suggests that LIME's linear surrogate models more accurately approximate local black-box behavior in many scenarios.

Faithfulness: Conversely, SHAP demonstrates superior faithfulness compared to LIME across experimental datasets [25]. For the German credit dataset, SHAP achieves faithfulness values of 0.19-0.44, while LIME scores range from 0.16-0.34. On the adult dataset, SHAP consistently outperforms LIME, achieving 0.44 faithfulness with CatBoost compared to LIME's 0.077 [25]. This indicates that SHAP explanations more accurately reflect the true feature influences despite lower fidelity scores.

Interpretability: Both methods offer distinct interpretability advantages. LIME's feature importance vectors are straightforward for domain experts who un-

Table 2.2: Comparison of the fidelity and the faithfulness metrics of different explanation methods from [25]. Bold values indicate the best results. For every evaluation, the mean and the standard deviation over a subset of 50 test set records is reported

Dataset	**Black-Box**	**Fidelity**				**Faithfulness**	
		LIME	SHAP	ANCHOR	LORE	LIME	SHAP
adult	LG	0.979	0.613	**0.989**	0.984	0.099 (0.30)	**0.38** (0.37)
	XGB	0.977	0.877	0.978	**0.982**	0.030 (0.32)	**0.36** (0.49)
	CAT	0.960	0.777	0.988	**0.989**	0.077 (0.32)	**0.44** (0.37)
german	LG	**0.984**	0.910	0.730	0.983	**0.23** (0.60)	0.19 (0.63)
	XGB	**0.999**	0.821	0.802	0.982	0.16 (0.26)	**0.44** (0.21)
	CAT	0.979	0.670	0.620	**0.981**	0.34 (0.33)	**0.43** (0.32)

derstand feature meanings, but may prove challenging for non-technical users when importance values are complex [14, 25]. SHAP's mathematical foundation provides theoretical guarantees but introduces complexity in understanding how base values, output values, and feature importance arrays combine to form explanations [25].

Consistency: Both methods exhibit poor consistency performance, with neither achieving remarkable consistency according to established metrics [25]. This inconsistency represents a shared weakness across many explanation methods, highlighting the need for more robust explanation generation procedures.

2.2 Visualizations for XAI

The following section reviews the literature and techniques that inform the development of the thesis visualization system. Since local explainability methods mostly rely on the same principle of generating a synthetic neighborhood and the training of a surrogate model, this review is structured to address the key components of our interactive explanation system. We begin by examining dimensionality reduction techniques that enable visual exploration of high-dimensional synthetic data in a reduced dimensional space. Subsequently, we explore the literature on decision tree visualization, as decision trees constitute the primary surrogate model employed for rule extraction. Finally, we survey existing visual tools and interactive techniques in eXplainable Artificial Intelligence.

2.2.1 Dimensionality Reduction Techniques

In the context of eXplainable Artificial Intelligence the visualization of high-dimensional data is of high relevance. When an explainability method generates synthetic neighborhoods around the instance to be explained, the generated neighborhoods typically exist in a high-dimensional feature space that is difficult to

interpret and visualize directly. Dimensionality reduction techniques serve as essential tools for projecting these multi-dimensional datasets into two or three-dimensional representations that enable human comprehension and interactive exploration [28].

The primary objective of dimensionality reduction in visual XAI is to preserve meaningful relationships and structures present in the original high-dimensional space while facilitating intuitive visual interpretation. This capability is particularly crucial as it allows users to spatially explore the synthetic neighborhood and understand the distribution of generated instances. The choice of dimensionality reduction technique significantly impacts the quality of explanations. Different methods preserve distinct aspects of the data structure, while some excel at maintaining local neighborhoods, others are very good at preserving global relationships [29].

The effectiveness of different dimensionality reduction methods varies depending on the characteristics of the dataset, the size of the synthetic neighborhood, and the specific visual analytics requirements of the explanation task [30]. The evaluation of dimensionality reduction techniques requires careful consideration of both local and global structure preservation criteria.

The following subsections examine four dimensionality reduction techniques. Each technique exhibits distinct characteristics in preserving data structure. We begin with **Principal Component Analysis (PCA)**, the most widely adopted linear method known for its computational efficiency and interpretability of component meanings. This is followed by **Multidimensional Scaling (MDS)**, a classical approach that focuses on preserving pairwise distances between data points. Subsequently, we explore **t-distributed Stochastic Neighbor Embedding (t-SNE)**, known for its effectiveness in revealing local cluster structures, though with limitations in global structure preservation. Finally, **Uniform Manifold Approximation and Projection (UMAP)**, which has shown superior performance in preserving both local and global structure while maintaining computational efficiency for large-scale synthetic neighborhoods.

The following subsections provide a detailed examination of each technique, followed by a comparative analysis that evaluates their relative performance across key criteria, including structure preservation, computational efficiency, and visual quality for neighborhood exploration.

PCA (Principal Component Analysis)

Principal Component Analysis (PCA) represents one of the, if not the most, widely adopted linear dimensionality reduction technique in data analysis, with historical foundations dating back to the pioneering work of Pearson (1901) and Hotelling (1933) [31]. As a fundamental method in multivariate statistics, PCA aims to

reduce the dimensionality of datasets while preserving as much relevant information as possible through variance maximization [31].

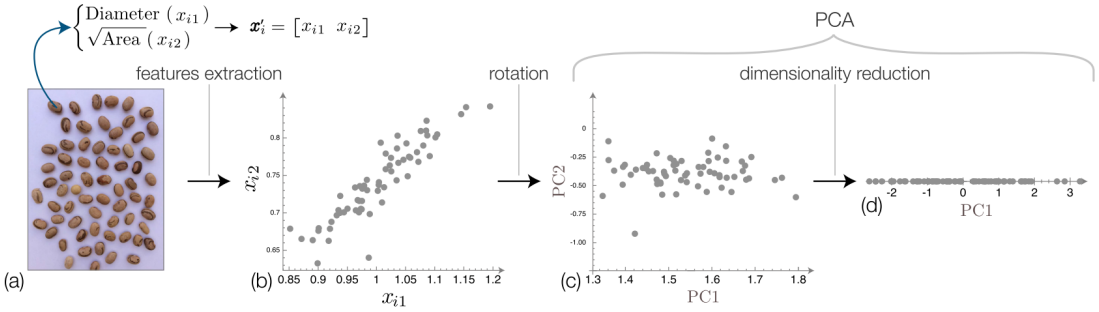


Figure 2.5: Illustration of the PCA process using real-world bean data. bean (a) is characterized in terms of two features: diameter xi1 and square root of area xi2. Though these two features are intrinsically related in a direct fashion, bean shape variations induce a dispersion of the objects when mapped into the features space (b). PCA allows the identification of the orientation of maximum data dispersion (c). As the dispersion in the resulting second axis is relatively small, this axis can be eventually disregarded (d).

The theoretical foundation of PCA rests on the principles of linear transformation, where the algorithm transforms the data to a new coordinate system such that the new variables (principal components) are linear functions of the original variables, are not correlated, and are ordered by the amount of variance that they explain [31]. The transformation operates as a rotation of the coordinate system that aligns the axes with the directions of maximum data variation [32], as illustrated in Figure 2.5. This process can be expressed in matrix form as $Y = XW$, where X represents the original data matrix, W is the transformation matrix composed of eigenvectors, and Y contains the principal component scores.

The algorithm's core mechanism involves computing the covariance matrix of the dataset, followed by eigenvalue decomposition to obtain eigenvectors and eigenvalues sorted in decreasing order [31]. Each eigenvector defines the direction of a principal component, while the corresponding eigenvalue quantifies the amount of variance explained along that direction [32]. An essential property of PCA is that the total data variance is preserved under the transformation: $\sum_{j=1}^{n} \sigma_{x_j}^2 = \sum_{j=1}^{n} \sigma_{y_j}^2$, meaning the variance along each principal component equals the corresponding eigenvalue of the covariance matrix [32].

A distinctive characteristic of PCA is its ability to provide direct interpretability through loadings analysis. The transformation matrix W contains eigenvectors that indicate how the original features contribute to each principal component, allowing users to understand which variables drive the main sources of variation in the data [32]. PCA demonstrates particular effectiveness in scenarios involving correlated features, where the method can achieve substantial dimensionality reduction while

retaining most of the original variance. PCA's deterministic nature provides stability that is particularly valuable in interactive applications where consistent visualizations are required [32].

Particularly significant for eXplainable AI applications is PCA's unique capability to preserve the structure of decision boundaries generated by the local surrogate models. Since PCA applies a linear transformation, the decision boundaries created by linear surrogate model maintain their essential geometric properties when projected into the two-dimensional visualization space. This preservation enables users to observe how the surrogate model rules translate into spatial regions within the spatial neighborhood analysis plot visualization, providing a direct correspondence between the rule-based explanations and the visual representation of the synthetic neighborhood. This characteristic distinguishes PCA from other dimensionality reduction techniques that may significantly distort or obscure the decision boundaries due to their nonlinear transformations.

However, PCA's linear assumption may limit its effectiveness for datasets with complex nonlinear structures. The algorithm's focus on variance maximization means that directions with high variance are prioritized regardless of their relevance for specific analytical tasks [32].

MDS (Multidimensional Scaling)

Multidimensional Scaling (MDS) has theoretical origins tracing back to the pioneering work of Torgerson (1952) [33] in developing systematic methods for scaling multidimensional psychological and perceptual data.

The core principle underlying MDS involves the preservation of pairwise distances between data points across different dimensional representations [33, 34]. The algorithm operates by constructing a low-dimensional embedding where the Euclidean distances between points in the reduced space approximate as closely as possible the original distances (or dissimilarities) measured in the high-dimensional space. This distance-preserving objective distinguishes MDS from techniques that focus primarily on local neighborhood relationships or topological structures.

Classical MDS, also known as Principal **Coordinate** Analysis, employs a three-step procedure as originally formulated by Torgerson [33]. The first step involves obtaining a scale of comparative distances between all pairs of stimuli (any set of objects, items, concepts, or entities in study), analogous to paired comparison methods but applied to distance relationships rather than direct stimulus comparisons. The second step requires estimating an additive constant to convert these comparative distances into absolute distances, addressing the inherent indeterminacy in the zero point of the distance scale. The third step determines the dimensionality of the space necessary to account for these absolute distances and computes the projections of data points onto the axes of this space.

MDS demonstrates particular strength in preserving global structure and long-range distance relationships within data [34]. The algorithm's explicit focus on maintaining the full distance matrix makes it especially suitable for applications where all scales of structure are equally important, and where the preservation of metric relationships takes precedence over local neighborhood fidelity. This global perspective allows MDS to maintain meaningful relative positions between widely separated clusters or groups within the data.

The technique has robust theoretical properties from its foundation in classical metric geometry. Since MDS minimizes the sum of squared errors between original and embedded distances, it provides a principled approach to dimensionality reduction with well-understood optimality properties. The linear nature of classical MDS ensures deterministic results and computational stability, avoiding the convergence issues that can affect iterative nonlinear methods.

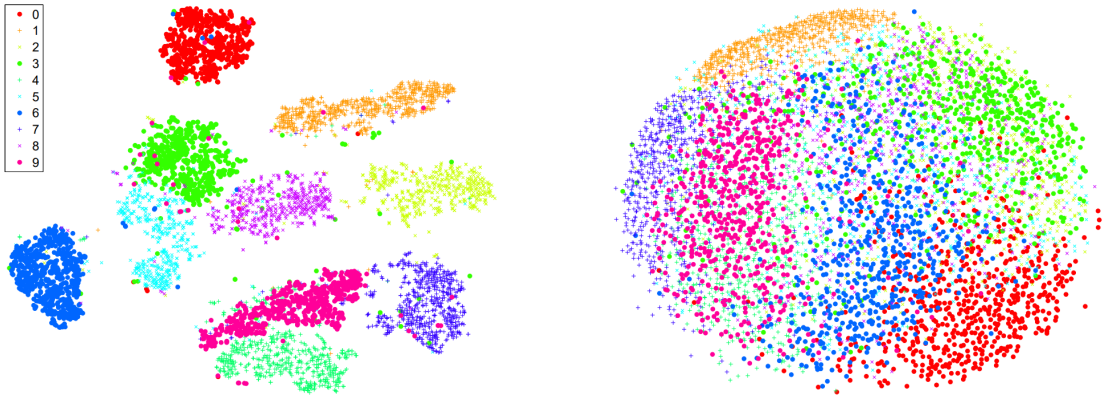
However, MDS faces several inherent limitations that constrain its effectiveness in specific visualization contexts. The algorithm's emphasis on preserving large pairwise distances can come at the expense of local neighborhood structure, potentially causing nearby points in the original space to appear more separated in the reduced dimensional representation than would be ideal for cluster identification [35]. This focus on global distance preservation means that MDS may not effectively capture local manifold structure or fine-grained clustering patterns that are crucial for understanding local decision boundaries in eXplainable AI applications.

Empirical evaluations on transcriptomic datasets reveal that MDS exhibits moderate performance in neighborhood preservation, indicating that local neighborhood relationships are less well preserved compared to methods specifically designed for local structure preservation [28].

The algorithm's linear transformation approach means that MDS may struggle with datasets containing complex nonlinear manifold structures. While this linearity provides interpretability advantages similar to PCA, it can result in poor representations for data that lies on curved or twisted manifolds where Euclidean distances in the original space do not accurately reflect the intrinsic geometric relationships.

Despite these limitations, MDS offers several advantages for specific analytical contexts. The method's deterministic nature ensures reproducible results, which can be valuable for comparative analyses and scientific reporting. The explicit distance preservation objective makes MDS particularly suitable for applications where the maintenance of quantitative distance relationships is more important than the visual separation of clusters or local neighborhoods.

For synthetic neighborhoods visualization applications, MDS presents a mixed proposition. While the algorithm's distance preservation properties might maintain some aspects of the original synthetic neighborhood structure, its tendency to



(a) Visualization by t-SNE. (b) Visualization by Sammon mapping.

Figure 2.6: Comparative visualization of 6,000 handwritten digits from the MNIST dataset using t-SNE (left) and Sammon mapping (right). The t-SNE visualization demonstrates superior cluster separation with clearly distinguishable digit groups (0-9), while Sammon mapping shows significant overlap between different digit classes. This comparison illustrates t-SNE's exceptional capability in preserving local neighborhood structures and creating meaningful cluster formations.

emphasize global relationships over local patterns may not optimally support the identification of local explanation regions that are central to eXplainable AI systems interpretability objectives. Modern variants of MDS, including metric and non-metric forms, have been developed to address some of these limitations, but the classical formulation remains the most commonly implemented approach.

t-SNE (t-distributed Stochastic Neighbor Embedding)

t-distributed Stochastic Neighbor Embedding (t-SNE) represents a nonlinear dimensionality reduction technique designed for data visualization [35]. Developed as an improvement over Stochastic Neighbor Embedding (SNE), t-SNE addresses fundamental challenges in high-dimensional data visualization by employing probabilistic modeling of pairwise similarities and takes advantage of heavy-tailed distributions to overcome the crowding problem in dimension reduction [35].

The algorithmic foundation of t-SNE rests on converting high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. For each data point x_i, the algorithm defines the conditional probability $p_{j|i}$ that x_i would select x_j as its neighbor if neighbors were chosen proportionally to their probability density under a Gaussian centered at x_i [35]. This probability is computed as $p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$, where σ_i represents the variance of the Gaussian kernel around point x_i, determined by the perplexity parameter that controls the effective number of nearest neighbors considered for each point.

A crucial aspect of t-SNE is its use of Cauchy distributions to model similarities in the low-dimensional space, rather than the Gaussian distribution used in the high-dimensional space [35]. This heavy-tailed distribution enables moderate distances in the high-dimensional space to be faithfully represented by much larger distances in the low-dimensional map, effectively eliminating unwanted attractive forces between dissimilar points that would otherwise be crowded together in the visualization.

The algorithm operates by minimizing the Kullback-Leibler divergence between the joint probability distributions in high-dimensional and low-dimensional spaces: $C = \sum_i \sum_j p_{i|j} \log \frac{p_{i|j}}{q_{i|j}}$, where $p_{i|j}$ and $q_{i|j}$ represent the joint probabilities in high and low dimensions respectively [35].

t-SNE demonstrates exceptional capability in preserving local neighborhood structures, making it particularly effective for revealing cluster formations and local patterns within high-dimensional data [28, 35]. As clearly shown in Figures 2.6 the algorithm's focus on modeling local similarities through the perplexity parameter, which allows it to adapt to varying local densities in the data, ensuring that each data point contributes meaningfully to the cost function regardless of its position relative to the global distribution [35]. The perplexity parameter serves as the primary hyperparameter controlling t-SNE's behavior, effectively determining the balance between local and global aspects of the data that are preserved in the embedding [35]. Typical perplexity values range from 5 to 50, with lower values emphasizing very local structures and higher values incorporating more global relationships. The algorithm's performance exhibits robustness across different perplexity settings for the same dataset, though optimal values may vary depending on the intrinsic structure and size of the data.

The algorithm excels at revealing structure at multiple scales simultaneously, making it particularly valuable for complex high-dimensional datasets that contain hierarchical or nested cluster structures [35]. This multi-scale capability emerges from t-SNE's probabilistic framework, which naturally adapts to local data densities while maintaining the ability to reveal broader organizational patterns through the heavy-tailed distribution in the embedding space

However, t-SNE exhibits several inherent limitations that impact its applicability in certain contexts. The algorithm's emphasis on preserving local structure can sometimes come at the expense of global relationships, potentially leading to visualizations where the relative positions of distinct clusters may not accurately reflect their relationships in the original high-dimensional space [35].

For neighborhood visualization applications, t-SNE's strength in local structure preservation makes it particularly valuable for revealing fine-grained patterns and cluster formations within synthetic neighborhoods. The algorithm's ability to create clear visual separation between different regions of the feature space can help

users understand how surrogate models local explanations relate to the underlying data distribution.

UMAP (Uniform Manifold Approximation and Projection)

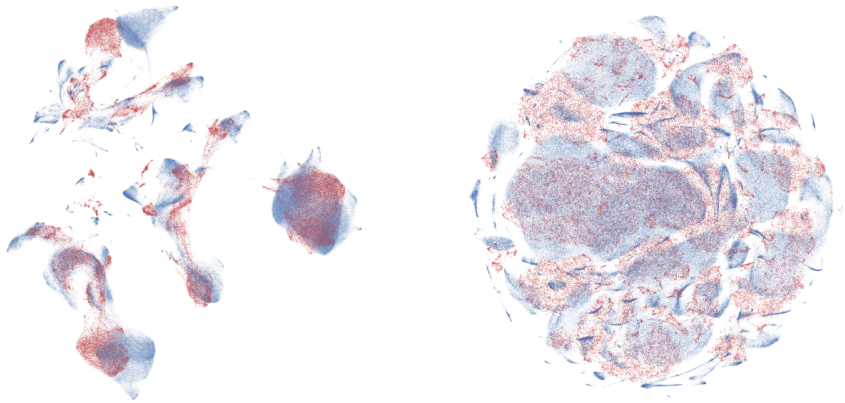
Uniform Manifold Approximation and Projection (UMAP) represents a novel manifold learning technique for dimensionality reduction that differentiates itself through strong theoretical foundations rooted in Riemannian geometry and algebraic topology [34]. Unlike many dimensionality reduction algorithms that rely primarily on empirical design decisions, UMAP's construction is grounded in mathematical theory, particularly drawing from the work of Spivak on categorical approaches to geometric realization of fuzzy simplicial sets [34, 36].

UMAP's theoretical framework begins with the assumption that data lies approximately on a locally connected Riemannian manifold, and that this manifold is equipped with a Riemannian metric with respect to which the data is uniformly distributed [34]. Building on these foundational assumptions, the algorithm operates by constructing local manifold approximations around each data point through two main processes: approximating the manifold on which the data lies, and constructing fuzzy simplicial set representations of the approximated manifold.

The algorithm subsequently patches together their local fuzzy simplicial set representations to build a topological representation of the high-dimensional data [34]. This approach allows UMAP to capture both local neighborhood structures and global topological relationships within the data, with the local connectivity assumption ensuring that each point is connected to its nearest neighbor with maximum membership strength, maintaining topological consistency across the embedding. The algorithm optimizes the low-dimensional embedding layout by minimizing the cross-entropy between the topological representations of the high-dimensional and low-dimensional spaces, effectively preserving the essential geometric and topological properties of the original data manifold.

One of UMAP's most significant advantages lies in its computational efficiency and embedding stability. To demonstrate this stability advantage, Figure 2.7 presents a Procrustes-based alignment comparison between UMAP and t-SNE embeddings using subsampled data from a flow cytometry dataset [37]. The visualization reveals a critical difference in algorithmic robustness: UMAP maintains substantially better structural consistency between the full dataset (blue points) and a 10% subsample (red points), as evidenced by the tight alignment and minimal deviation between the two point sets. In contrast, t-SNE exhibits considerable variation in embedding structure when working with subsampled data, with red and blue points showing significant displacement and structural distortion.

One of UMAP's most significant advantages lies in its computational efficiency and scalability. Empirical evaluations demonstrate that UMAP is approximately



(a) UMAP (b) t-SNE

Figure 2.7: Procrustes-based alignment comparison of UMAP (left) and t-SNE (right) embeddings on flow cytometry data. Red points represent a 10% subsample, blue points represent the full dataset [37].

an order of magnitude faster than t-SNE while maintaining comparable or superior visualization quality. The algorithm exhibits superior scaling performance across multiple dimensions: it scales efficiently with the number of data points, performs exceptionally well with high ambient dimensionality, and maintains computational efficiency when generating embeddings into dimensions higher than two. Unlike t-SNE, which requires global normalization and space trees that scale exponentially with embedding dimension, UMAP represents data as a fuzzy topological structure, allowing it to work without such computationally expensive constructs.

The algorithm's ability to preserve both local and global structure makes it particularly well-suited for interactive visualization applications in eXplainable AI. UMAP successfully maintains local cluster structures that facilitate identification of similar instances in synthetic neighborhoods, while simultaneously preserving global relationships that help users understand the overall data distribution and inter-cluster relationships [29].

UMAP's hyperparameters provide intuitive control over the embedding characteristics. The `n_neighbors` parameter controls the balance between local and global structure preservation, with lower values emphasizing local structure and higher values focusing on global relationships. The `min_dist` parameter controls the minimum distance between points in the low-dimensional representation, affecting the tightness of clusters in the final embedding [34]. These parameters allow users to fine-tune the visualization based on their specific analytical requirements and the characteristics of the synthetic neighborhood being explored.

For synthetic neighborhood visualization requirements, UMAP's combination

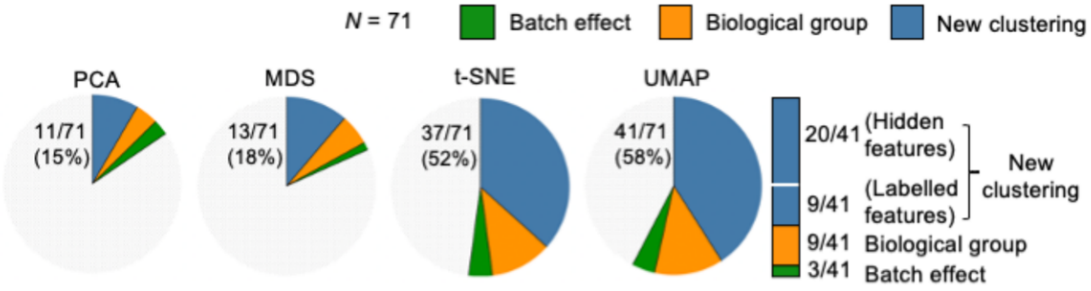


Figure 2.8: Dimensionality reduction techniques comparison results.

of theoretical rigor, computational efficiency, structure preservation, and stability makes it an ideal choice for projecting high-dimensional synthetic neighborhoods into interpretable 2D representations that facilitate effective human-AI interaction and explanation understanding.

Comparative Analysis of Dimensionality Reduction Techniques

Recent empirical research has provided comprehensive comparative evaluations of dimensionality reduction techniques across large-scale datasets, offering valuable insights for their application in eXplainable AI systems. Yang et al.[28] conducted a systematic comparison of the four dimensionality reduction methods previously discussed: UMAP, t-SNE, PCA, and MDS, across 71 bulk transcriptomic datasets, evaluating their performance through both quantitative metrics and qualitative biological interpretability. This comparative analysis provides crucial evidence for selecting appropriate dimensionality reduction techniques.

The comprehensive evaluation framework uses quantitative metrics including **clustering accuracy** measured through Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI), **neighborhood preservation** assessed via Jaccard index, and **computational efficiency** across varying dataset sizes [28].. As shown in Figure 2.8, UMAP identified clustering structures in 41 out of 71 datasets (58%), outperforming both t-SNE (37/71, 52%), MDS (13/71, 18%) and PCA (11/71, 15%).

UMAP vs t-SNE The comparison between UMAP and t-SNE reveals complementary strengths, with UMAP demonstrating superior overall performance in large-scale data analysis. Both methods excel at preserving local neighborhood structures, with UMAP achieving an average neighborhood preservation score of 0.35 ± 0.091 compared to t-SNE's 0.36 ± 0.095, indicating comparable local structure retention capabilities [28]. As shown in Figure 2.9, both techniques substantially outperform linear methods in maintaining local relationships within

Figure 2.9: Dimensionality reduction techniques comparison results on neighborhood-preservation accuracy metric.

high-dimensional data.

However, UMAP demonstrates superior clustering accuracy across multiple evaluation criteria. Figure 2.10 illustrates how UMAP consistently achieved the highest Normalized Mutual Information scores across five different clustering algorithms (k-means, hierarchical clustering, spectral clustering, Gaussian mixture model, and HDBSCAN), while t-SNE ranked second but with notably lower performance.

This enhanced clustering accuracy translates directly to improved separability of meaningful data structures, with UMAP achieving over 90% random forest classification accuracy for batch effect detection compared to t-SNE's slightly lower performance. A critical advantage of UMAP over t-SNE lies in computational efficiency, particularly for large-scale datasets. While both methods perform similarly for smaller datasets (200-500 samples), UMAP gains substantial advantages as dataset size increases. For datasets with 10,000 samples, UMAP requires approximately 3 minutes compared to t-SNE's 1.5 hours, representing more than a 25-fold improvement in computational speed. This efficiency advantage makes UMAP significantly more practical for interactive explanation systems that require responsive visualization updates. Additionally, UMAP exhibits superior global structure preservation compared to t-SNE, arising from its use of Laplacian Eigenmap initialization and cross-entropy objective function rather than t-SNE's random initialization and KL-divergence approach [28].

UMAP vs PCA The comparison between UMAP and PCA highlights fundamental differences between nonlinear manifold learning and linear dimensionality reduction approaches. While PCA demonstrates superior computational efficiency, processing 10,000 samples in approximately 20 seconds compared to UMAP's 3 minutes [28], UMAP provides dramatically superior performance in revealing complex data structures and meaningful clustering patterns. The most striking difference appears in clustering structure identification capabilities. As demonstrated in Figure 2.8, PCA identified clustering structures in only 11 out of 71 datasets (15%), while UMAP revealed structures in 41 datasets (58%). This improvement in structure detection capability is due to UMAP's ability to capture nonlinear relationships

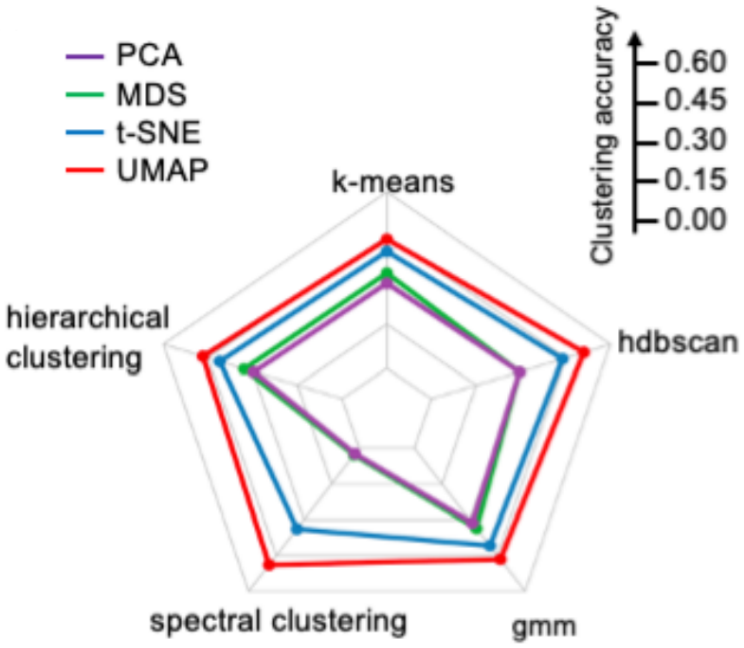


Figure 2.10: Dimensionality reduction techniques comparison results on Normalized Mutual Information scores across five different clustering algorithms.

that PCA's linear transformation cannot represent effectively. Clustering accuracy metrics reveal substantial performance differences between the methods. Figure 2.9 shows UMAP achieving consistently higher NMI scores across all clustering algorithms, while PCA demonstrated the poorest performance among all evaluated methods. This difference extends to neighborhood preservation, where UMAP's local structure retention (0.35 ± 0.091) significantly exceeds PCA's performance (0.19 ± 0.067), indicating that PCA's linear projections fail to maintain the local relationships crucial for interactive data exploration. Despite PCA's computational advantages and interpretability through component loadings, its fundamental limitation lies in the linear assumption that proves inadequate for complex data structures.

UMAP vs MDS The comparison between UMAP and MDS reveals the limitations of classical distance-preserving approaches when applied to complex high-dimensional data. MDS, while theoretically principled in its global distance preservation approach, demonstrates inferior performance across multiple evaluation criteria compared to UMAP's manifold learning methodology. Computational efficiency represents a significant disadvantage for MDS. MDS as the slowest performing method across all dataset sizes, with particularly poor scaling characteristics that make it impractical for large-scale interactive applications. This computational bur-

den severely limits MDS's applicability in explainability systems where responsive visualization updates are essential for effective human-AI interaction. Clustering accuracy metrics reveal substantial performance gaps between the methods. UMAP consistently outperforms MDS across all clustering algorithms evaluated. The neighborhood preservation analysis highlights fundamental differences in the methods' design philosophies. While MDS focuses on preserving global pairwise distances, this emphasis comes at the expense of local neighborhood structure preservation. Figure 2.9 shows MDS achieving only moderate neighborhood preservation (0.26 $\pm$ 0.114) compared to UMAP's superior local structure retention (0.35 $\pm$ 0.091). Structure identification capabilities further demonstrate UMAP's superiority. For synthetic neighborhoods visualization applications, MDS's focus on global distance preservation may theoretically seem appealing for maintaining quantitative relationships within synthetic neighborhoods. However, the empirical evidence demonstrates that UMAP's superior local structure preservation, computational efficiency, and clustering accuracy make it significantly more suitable for interactive explanation systems where users need to explore and understand complex data relationships efficiently.

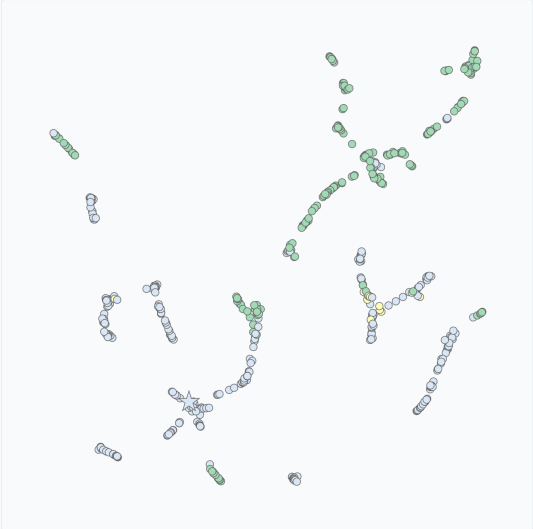
Figure 2.11 illustrates the comparison of the 4 methods on the same generated neighborhood of 3000 samples.

2.2.2 Decision Trees

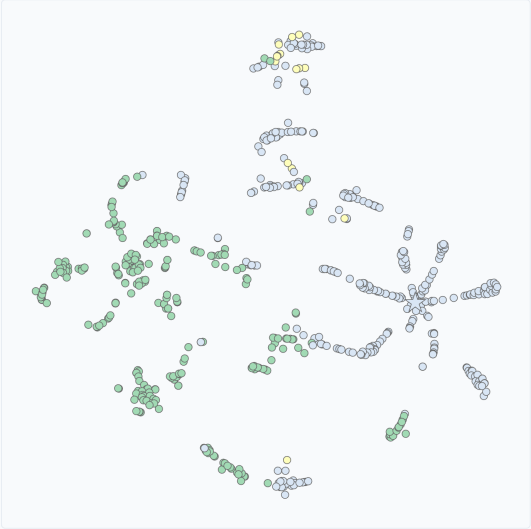
Decision trees have long been recognized as one of the most interpretable machine learning models due to their transparent hierarchical structure and human-readable decision rules [22, 38, 39, 40]. The effective visualization of decision trees for interpretability is an active area of research, with significant implications for eXplainable Artificial Intelligence and human-AI interaction. The challenge extends beyond simply displaying the tree structure to creating interfaces that support comprehensive understanding, analysis, and interaction with both the model and the underlying data.

The literature on decision tree visualization spans from traditional node-link diagrams to sophisticated interactive systems. Hans-Jorg Schulz [41] provides a comprehensive survey identifying over 180 tree visualization techniques, categorizing them by dimensionality (2D, 3D, or hybrid), edge representation (explicit, implicit, or hybrid), and node alignment (radial, axis-parallel, or free). This taxonomic foundation reveals the breadth of approaches available, yet also highlights the fragmented nature of the field, where many techniques have been developed independently without systematic comparison.

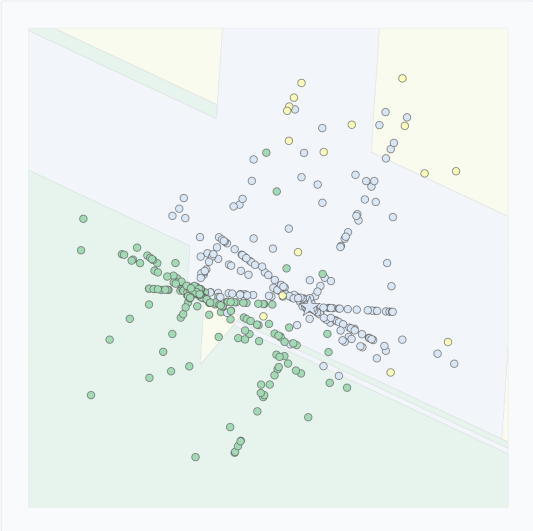
Recent advances in the field emphasize the critical importance of integrating visualization with interaction and algorithmic support. Van den Elzen et al. [42] demonstrate that effective decision tree systems require tight coupling between



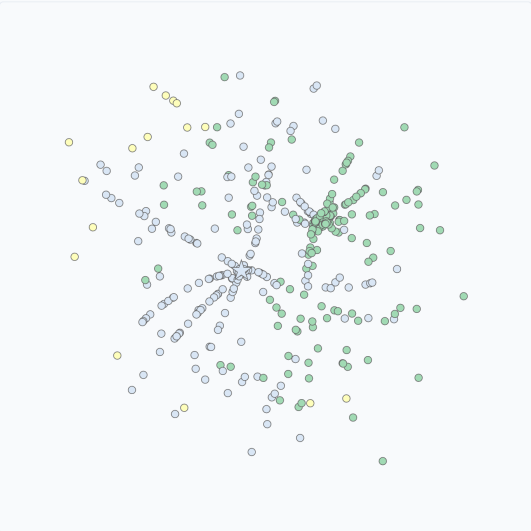
(a) Uniform Manifold Approximation and Projection for dimension reduction



(b) t-distributed Stochastic Neighbor Embedding



(c) Principal Component Analysis



(d) MultiDimensional Scaling

Figure 2.11: Example of the four dimensionality reduction techniques on a LORE$_{sa}$ generated neighborhood of 3000 samples, generated starting from the mean instance of the IRIS dataset

these three components, enabling domain experts to incorporate their knowledge into both tree construction and analysis processes. Their BaobabView system, shown in Figure 2.12, exemplifies this integration by supporting interactive growing, pruning, optimization, and analysis of decision trees through coordinated visual interfaces.

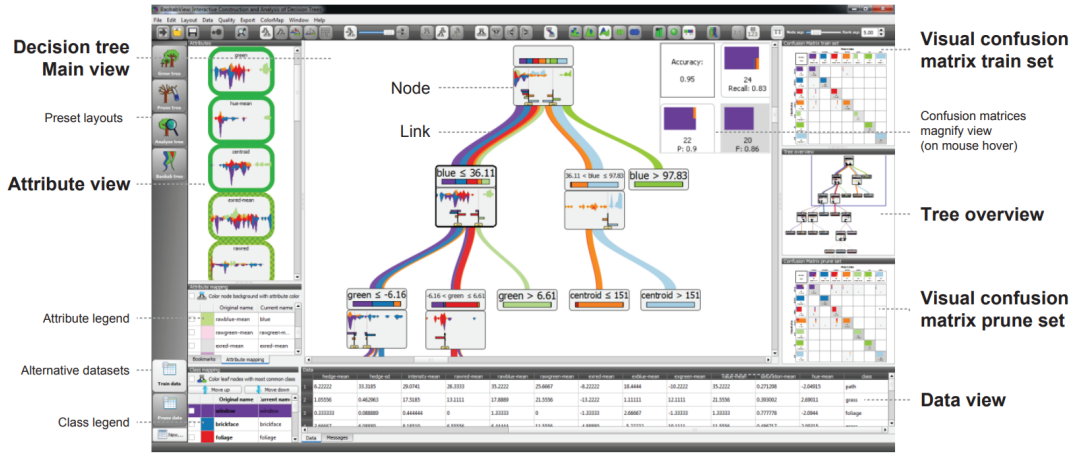
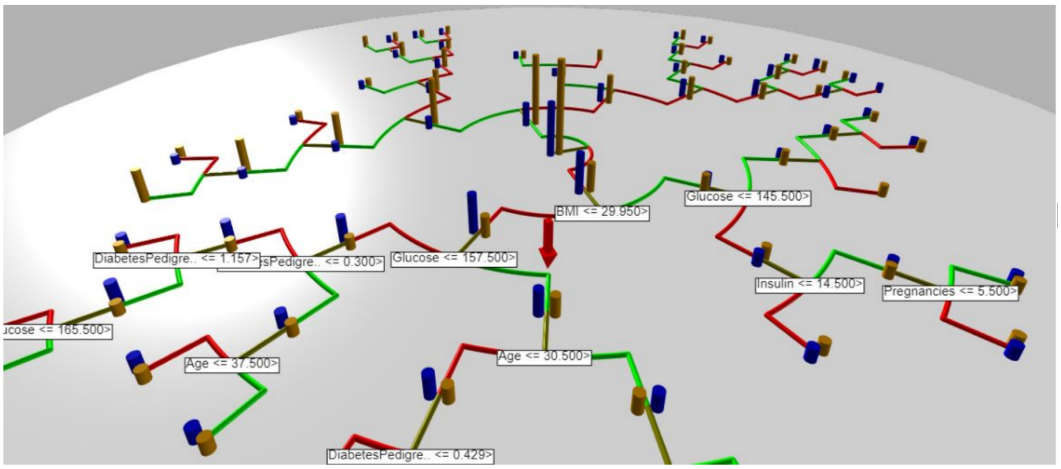


Figure 2.12: Interface of the interactive decision tree construction software BaobabView with the proposed decision tree visualization. Based on adapted node-link diagram, nodes contain important decision tree components while links are visualized as a stream of items flowing between nodes. The interface integrates decision tree main view, attribute view, visual confusion matrices, tree overview, and data view for comprehensive analysis.

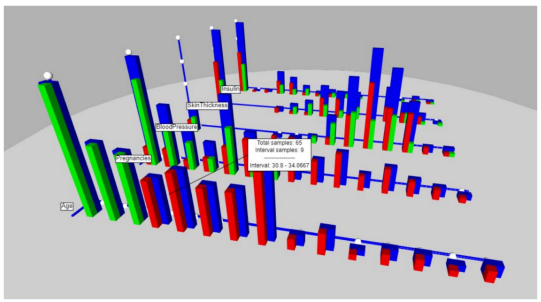
The application domains for decision tree visualization are diverse, with particular emphasis on high-stakes decision-making contexts. In medical applications, Jakub Mrva et al. [43] explore 3D visualization techniques that not only display tree structure but also reveal correlations between attributes and decision impacts, as demonstrated in Figure 2.13.

A significant trend in recent literature is the handling of high-dimensional data and complex model spaces. Szücs et al. [44] deal with the challenge of visualizing decision boundaries in high-dimensional feature spaces through projection strategies, while Wang et al. [38] address the problem of model selection from large sets of equally-performing trees through their innovative Rashomon set visualization shown in Figure 2.14. These works highlight the evolving complexity of decision tree applications and the corresponding need for more sophisticated visualization approaches.

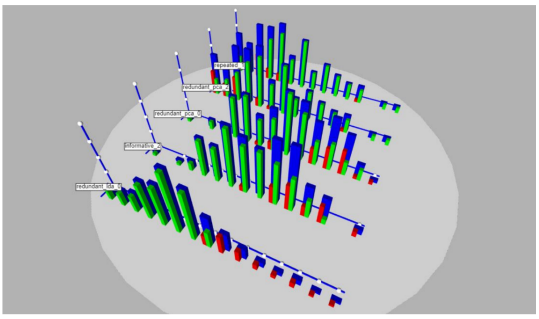
Interactive capabilities have emerged as a central theme across multiple studies. Kovalerchuk et al. [45] demonstrate how interactive threshold modification and dynamic tree construction can enhance model interpretability, while practical tools like dtreeviz [46] focus on providing immediate visual feedback for model validation



(a) Visualization displaying the overall structure of the trained decision tree on diabetes dataset. Focused node is marked by the arrow.



(b) A focused view on one decision tree node. The first set of histograms displays attribute used in the split. Following sets display correlated attributes.



(c) A detailed view on a decision rule in a single node and histograms for other attribute values of both data subsets divided by the node. Histograms in decreasing dissimilarity order highlight most important differences.

Figure 2.13: 3D decision tree visualization for medical data using circular plane layout. Nodes are positioned based on depth from the root, with 3D bar charts and histograms showing class distributions and feature relationships. This approach emphasizes visualization of attribute correlations beyond the primary splitting criterion.

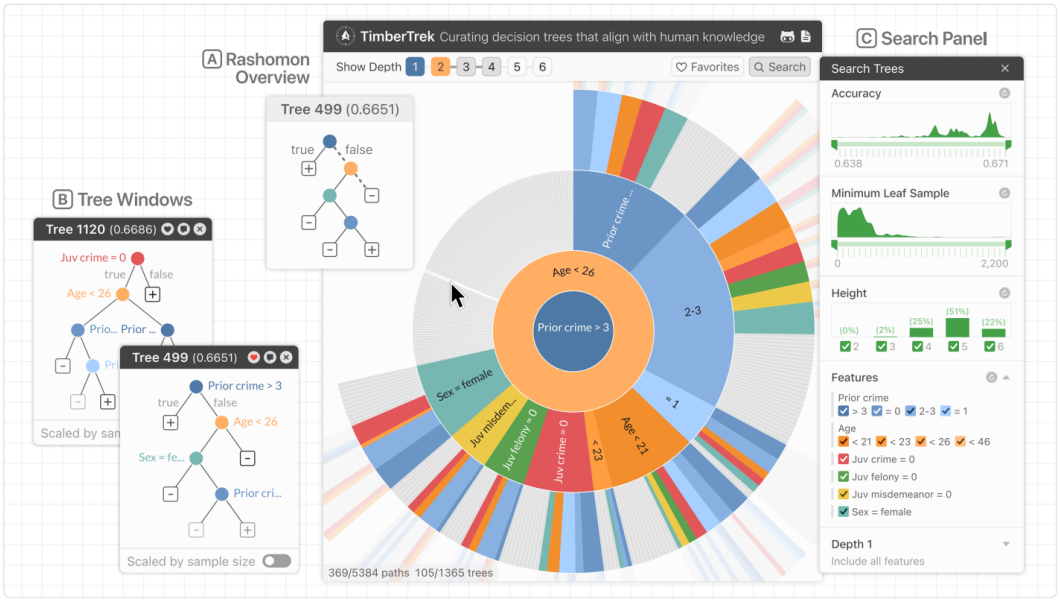


Figure 2.14: TIMBERTREK empowers domain experts and data scientists to easily explore thousands of well-performing decision trees so they can find and collect those trees that best reflect their knowledge and values. Consider the task of predicting whether a criminal is likely to commit a crime in the next two years. (A) The Rashomon Overview visually summarizes all well-performing decision trees by organizing them based on their decision paths, enabling users to seamlessly transition across different model subsets and explore trees with similar prediction patterns. (B) Clicking a tree opens a repositionable Tree Window showing details of a decision tree: multiple windows allow users to compare several model candidates' prediction patterns. (C) The Search Panel provides filtering tools, enabling users to quickly identify decision trees with desired properties, such as accuracy, robustness, simplicity, and used features.

and debugging. The emphasis on interactivity reflects a broader shift from static visualizations toward dynamic exploration tools that support iterative analysis and refinement.

Despite this rich literature, several gaps remain in current approaches. Most existing visualizations treat the tree structure and the underlying data as separate entities, requiring users to mentally connect model decisions with data distributions. Furthermore, there is limited exploration of how spatial neighborhood analysis plot can be integrated with rule-based interfaces to provide spatial context for instance-level interpretations.

Analysis of Existing Literature and Visualization Techniques

The literature reveals a diverse ecosystem of decision tree visualization techniques and tools, which can be categorized based on their visual representation approach,

interaction paradigm, and target use case. Streeb et al. [47] provide one of the most comprehensive analyses of this topic, surveying over 150 publications and categorizing them across 16 distinct tasks, 10 possible visual designs, 16 visual designs of further components, and 16 quality measures. The general results of the evaluation can be observed in Tables 2.3, 2.4 and 2.5, but the consultation of the publication, currently available at the following link https://el-assady.com/publication/2021streebtask/2021streebtask.pdf, is recommended to obtain a full picture of the results and of the reference articles.

Table 2.3: Tasks [47]

Ref.	Concept Introduction	Model Building	Evaluation	Understanding	Diagnosis	Refinement	Comparison	Ensemble Building	Provenance	Reporting	Presentation	Application	Assessment	Monitoring	Decision Modeling	Model Approximation
Total: 152	36	36	34	46	21	8	43	10	3	5	55	6	10	1	11	6

Traditional Layout Approaches The most fundamental category is classical tree layout algorithms. Hans-Jorg Schulz et al. [41] identify several core approaches including **node-link diagrams**, which represent the standard hierarchical visualization with explicit connections between parent and child nodes. The evolution of radial tree visualization approaches is particularly well-documented, as shown in Figure 2.15, which traces the development from natural patterns to interactive systems.

Indentation diagrams provide a text-based alternative where parent-child relationships are conveyed through spatial indentation, though these suffer from poor structural overview for large trees [42].

Coordinate-Based Visualizations A significant body of work explores coordinate system transformations for decision tree representation. Kovalerchuk et al. [45] introduce **General Line Coordinates (GLC)** with two variants: **Bended Coordinates (BC)** that bend at threshold points, and **Shifted Paired Coordinates (SPC)** that visualize n-dimensional points as directed graphs in 2D Cartesian coordinates.

Parallel coordinates integration has been explored by multiple researchers [42, 48], with noting attempts to combine decision trees with parallel coordinate plots, though these suffer from unclear class distribution representation. **Star coordinate plots** have been employed for interactive tree construction, allowing

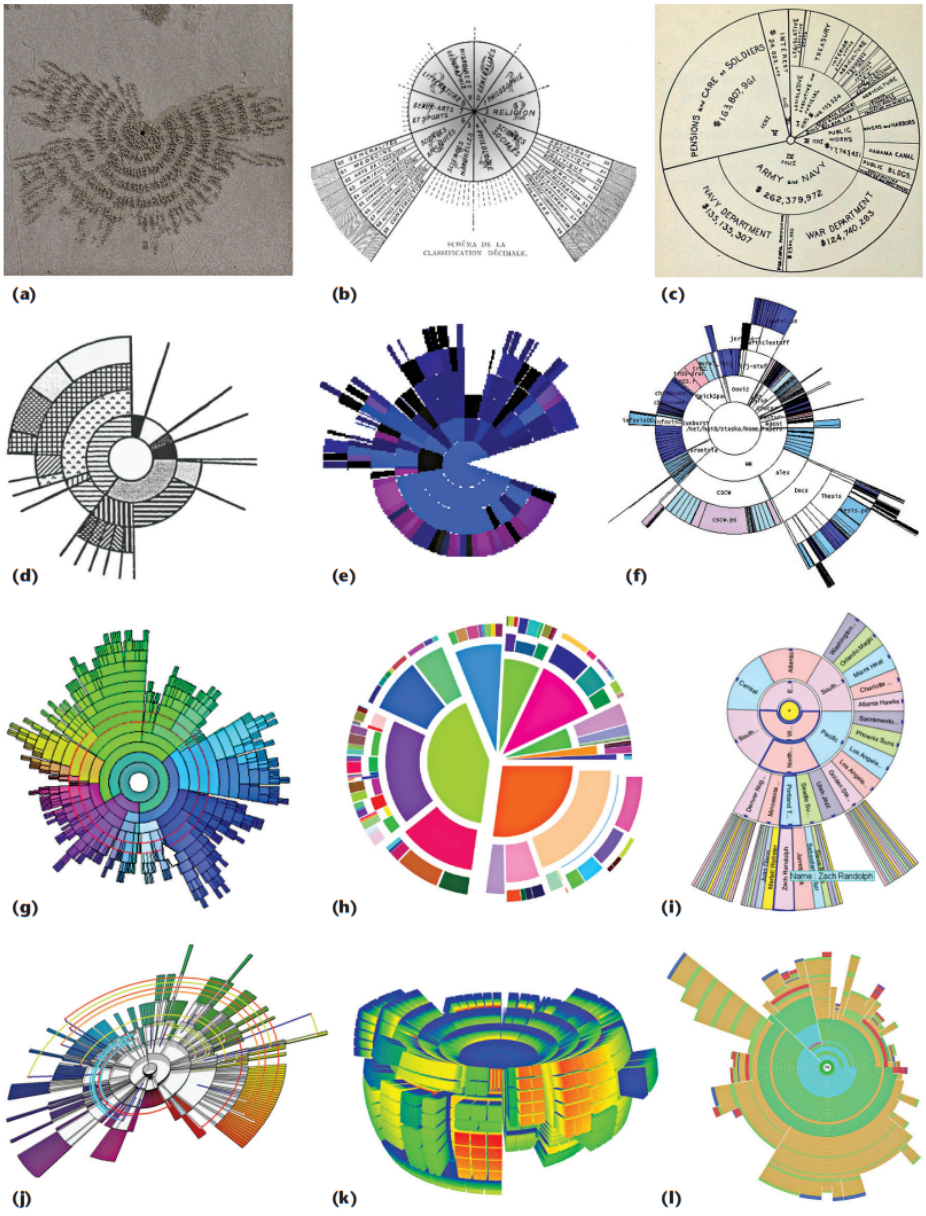


Figure 2.15: The evolution of radially stacked tree visualization. (a) Sand bubbler crab pattern (jkr1812 via Flickr). (b) Universal decimal classification (1905, P. Otlet). (c) Hierarchical sector chart (1921, Am. Soc. Mechanical Engineers). (d) Spoked polar tree map (1993, B. Johnson). (e) Aggregate tree map (1998, M. Chua). (f) Sunburst (2000, J. Stasko). (g) Interring (2002, J. Yang et al.). (h) PieTree (2006, R. O'Donnell et al.). (i) FanLens (2008, X. Lou et al.). (j) Enhanced radial space-filling layout (2009, M. Jia et al.). (k) 3D sunburst wheel (2010, H.-J. Schulz and S. Hadlak). (l) Trevis calling-context tree ring chart (2010, A. Adamoli and M. Hauswirth). The fundamental radial design turns out to be much older than most people think.

Table 2.4: Cross-tabulation of tasks and quality measures displayed in the 152 publications Streeb et al. [47] surveyed. Totals count unique publications in each row/column. Clearly, Accuracy is the most prominently displayed measure of quality. However, compared to the size of our sample quality measures are rarely displayed. There is no relationship apparent between tasks and the quality measures displayed.

Perspective	Column				Row				Marginal/Mixture					n.a.			
Quality Measure / Task	AUC	Recall/Sensitivity	Specificity	Weighted accuracy	Gain-ratio	Gini-index	Information gain	Precision	Accuracy	F1-score	G-means	Lift	χ^2	Frugality	Mean cues used	Other	*Total*
Concept Introduction	1	1	1			1	1					1		1		2	*4*
Model Building	1	2			1	1	4	2	4	1				1		3	*11*
Evaluation	3	4	4	1		1	2	1	11	1	1		1	1	1	4	*18*
Understanding	2	3	1		2	3	2	3	14	1		1	3			6	*24*
Diagnosis		3			1		2	3	4	1			1			2	*7*
Refinement		2			1		2	2	3	1							*4*
Comparison	3	4	3	1			1	1	11		1	1	2	2	1	4	*18*
Ensemble Building	1	1				1	1	1	5			1	1			2	*8*
Provenance	1															1	*1*
Reporting		3	2	1				1	1					1	1		*3*
Presentation		5	3	1		1	1	2	12	1	1		1	1	1	5	*18*
Application		1	1	1		1			1					1	1		*2*
Assessment		2			1	1		2	2				1			2	*5*
Monitoring																	*0*
Decision Modeling									3							1	*4*
Model Approximation		2						2	1								*4*
Total	*6*	*6*	*4*	*1*	*2*	*4*	*5*	*3*	*26*	*1*	*1*	*2*	*3*	*2*	*1*	*12*	

users to paint areas and assign class labels [42, 49, 50]. The StarClass/PaintingClass system workflow is illustrated in Figure 2.16, demonstrating the integration of coordinate-based visualization with interactive model building.

Rule-Based and Matrix Visualizations Ming et al. [51] pioneered the **matrix-based visualization** approach, where each row represents a decision rule and each column represents a feature. This technique transforms decision trees into a standardized rule-based knowledge representation, as illustrated in Figure 2.17, complemented by **stream plots** for continuous features and **stacked bar charts** for categorical features. One can observe the resulting interface in Figure 2.18.

3D and Immersive Approaches Three-dimensional visualization techniques have been explored for enhanced spatial understanding. Mrva et al. [43] present a **3D circular plane layout** where nodes are positioned based on depth from the

Table 2.5: Cross-tabulation of tasks and quality measures displayed in the 152 publications Streeb et al. [47] surveyed. Totals count unique publications in each row/column. Clearly, Accuracy is the most prominently displayed measure of quality. However, compared to the size of the analyzed sample, quality measures are rarely displayed. There is no apparent relationship between tasks and the quality measures displayed.

	Visual design of tree										Visual design of further components																		
Task	Circle packed layout	Flow chart [110]	Icicle plot [40]	(Indented) list	Node-link diagram	Parallel coordinates plot [41]	Pipe diagram [42]	Sunburst diagram [111]	Treemap [18]	Other	Area chart	Bar chart/histogram	Bar chart (stacked)	Box plot [112]	Dot plot [113]	Icon array	Image-like	Kernel density plot	Line chart	Matrix view	Parallel coordinates plot [41]	Pie chart	Pixel-based view [114]	Scatter plot	Star coordinates plot [115]	Stream graph [116]	Other	*Total*
Concept Introduction		1	1	3	35			1	5	3	1	1							1	2	1	2	2	1				*36*
Model Building		1	4	6	20		3	1	2	3	1	4	6		1		1		4	6	3	1	7	10	1	2	3	*36*
Evaluation			5	5	17		4			3		8	6		1	1		1	6	3	1	1	4	12		2	2	*34*
Understanding		2	5	8	28	2	4	1	1	3	1	13	5				1		5	4	3	7	6	15	1	1	4	*46*
Diagnosis		1	2	2	15		5			2	1	6	2		1				2	3	2	2	2	8		2	1	*21*
Refinement		1		1	5		1		1	1	1	2	2	1					2	3		1		1		1	1	*8*
Comparison			4	2	29		3		3	3	2	11	3	1	1	1		1	9	4	3		5	14		1	2	*43*
Ensemble Building			1	2	4		1		1	2	1	4	2						5	1	1	2	1	4		1	2	*10*
Provenance					2								1						1	1	1			1				*3*
Reporting					4		1					1	1			1		1	1	2	1			5				*5*
Presentation		2	1	1	52	1	3		3	1	2	9	3	1		1		1	4	5		2	5	11		1		*55*
Application			1	1	5	1	1			1		1	2			1		1		1		1		2				*6*
Assessment			1	2	3		3				1	4	1						3	1		2	1	2				*10*
Monitoring										1			1															*1*
Decision Modeling					10	1			1			1											1	1				*11*
Model Approximation				1	2		2		1		1	1	1						2	1	1	2		2				*6*
Total	*0*	*5*	*10*	*14*	*110*	*2*	*6*	*2*	*9*	*10*	*4*	*22*	*10*	*1*	*2*	*1*	*1*	*1*	*14*	*13*	*7*	*9*	*16*	*33*	*1*	*2*	*6*	

root, with **3D bar charts** and **3D histograms** showing class distributions and feature relationships. This approach particularly emphasizes the visualization of attribute correlations beyond the primary splitting criterion, as shown in Figure 2.13.

Interactive and Advanced Visualization Systems Several comprehensive systems integrate multiple visualization techniques with interactability. **BaobabView** [42] combines traditional node-link diagrams with integrated confusion matrices, attribute views, and data tables, emphasizing tight integration of visualization, interaction, and algorithmic support. The system's algorithmic support capabilities are detailed in Figure 2.19, which shows how the interface provides visual guidance for split attribute selection.

The data flow visualization capabilities of BaobabView are further illustrated in Figure 2.20, which shows how the system visualizes instance partitioning and highlights misclassifications.

TIMBERTREK [38] introduces innovative **Sunburst diagrams** [52] for visualizing Rashomon sets of equally-performing decision trees, enabling users to explore thousands of model candidates through focus+context [53] interaction techniques, as demonstrated in Figure 2.14.

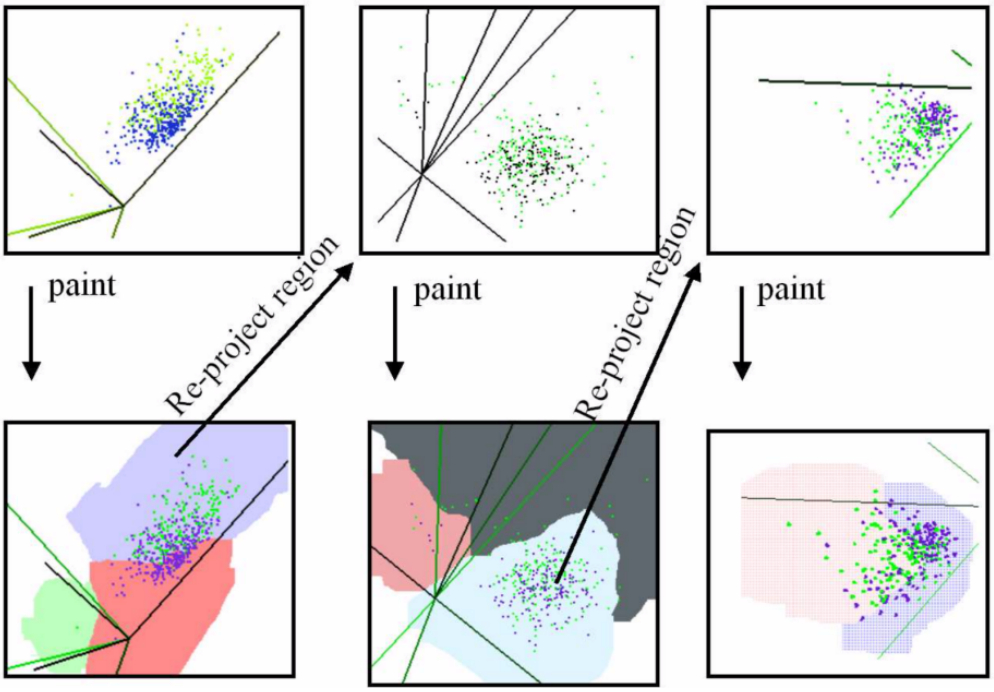


Figure 2.16: Workflow and visualization of the StarClass/PaintingClass system [48, 49]. During model building, analysts use re-projection and painting of regions at different levels of the hierarchy to effectively partition classes in the instance space. The system enables interactive decision boundary creation through direct manipulation of star coordinate projections.

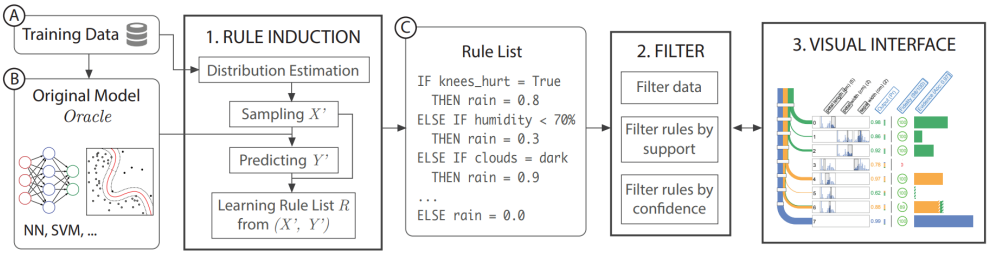


Figure 2.17: The pipeline for creating a rule-based explanation interface. The rule induction step (1) takes (A) the training data and (B) the model to be explained as input, and produces (C) a rule list that approximates the original model. Then the rule list is filtered (2) according to user-specified thresholds of support and confidence. The rule list is visualized as RuleMatrix (3) to help users navigate and analyze the rules.

Practical Tools and Libraries The landscape of practical implementations includes several notable tools. The **dtreeviz library** provides feature-target space visualization with **strip plots**, **scatter plots**, and decision boundary highlighting. Figure 2.21 shows comprehensive comparisons between different tools from the

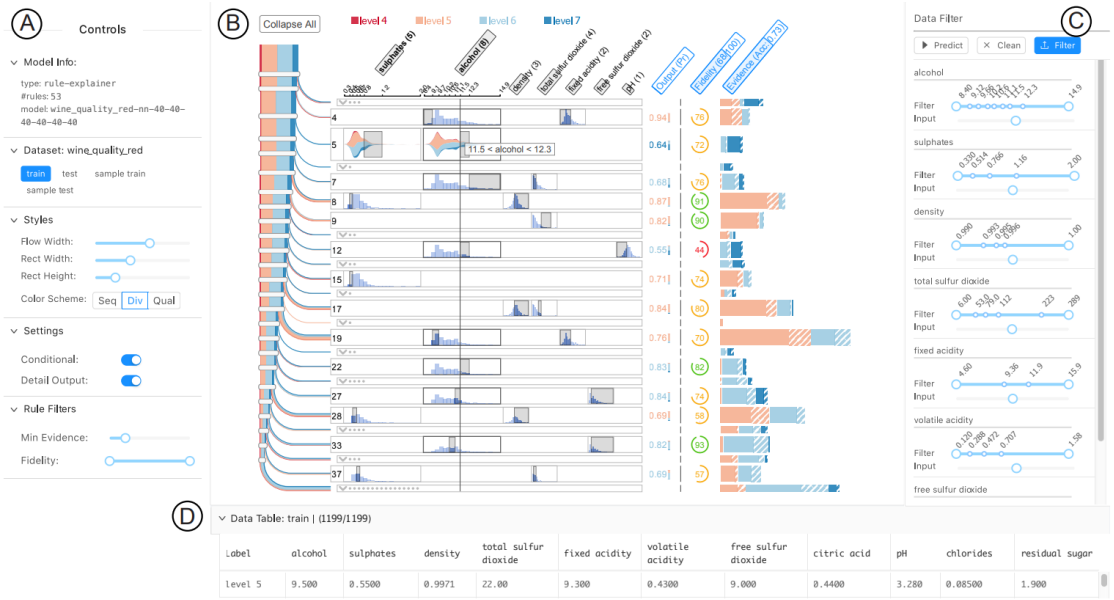


Figure 2.18: Understanding the behavior of a trained neural network using the explanatory visual interface of our proposed technique. The user uses the control panel (A) to specify the detail information to visualize (e.g., level of detail, rule filters). The rule-based explanatory representation is visualized as a matrix (B), where each row represents a rule, and each column is a feature used in the rules. The user can also filter the data or use a customized input in the data filter (C) and navigate the filtered dataset in the data table (D).

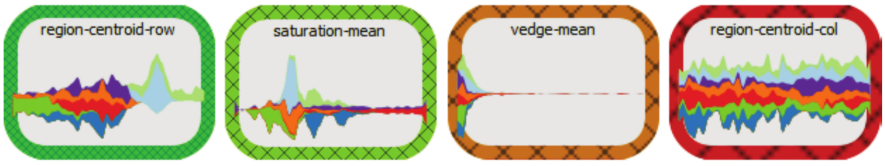


Figure 2.19: The BaobabView system supports analysts with algorithmic support for selecting split attributes and presents suggestions visually. Border color indicates the goodness of the split as measured by the Gain-ratio, enabling users to make informed decisions about tree construction while maintaining control over the modeling process.

library's authors' documentation on "how to visualize decision trees" [46].

Standard machine learning libraries offer basic capabilities: **scikit-learn's plot_tree**, **graphviz integration** for DOT format export, and **text-based representations** [54]. Specialized tools like the **supertree package** add interactive capabilities, including drag-and-zoom, node collapse/expand, and sample flow visualization between nodes [54]. Web-based implementations using **D3.js** enable interactive decision table to tree conversion and real-time traversal [55].

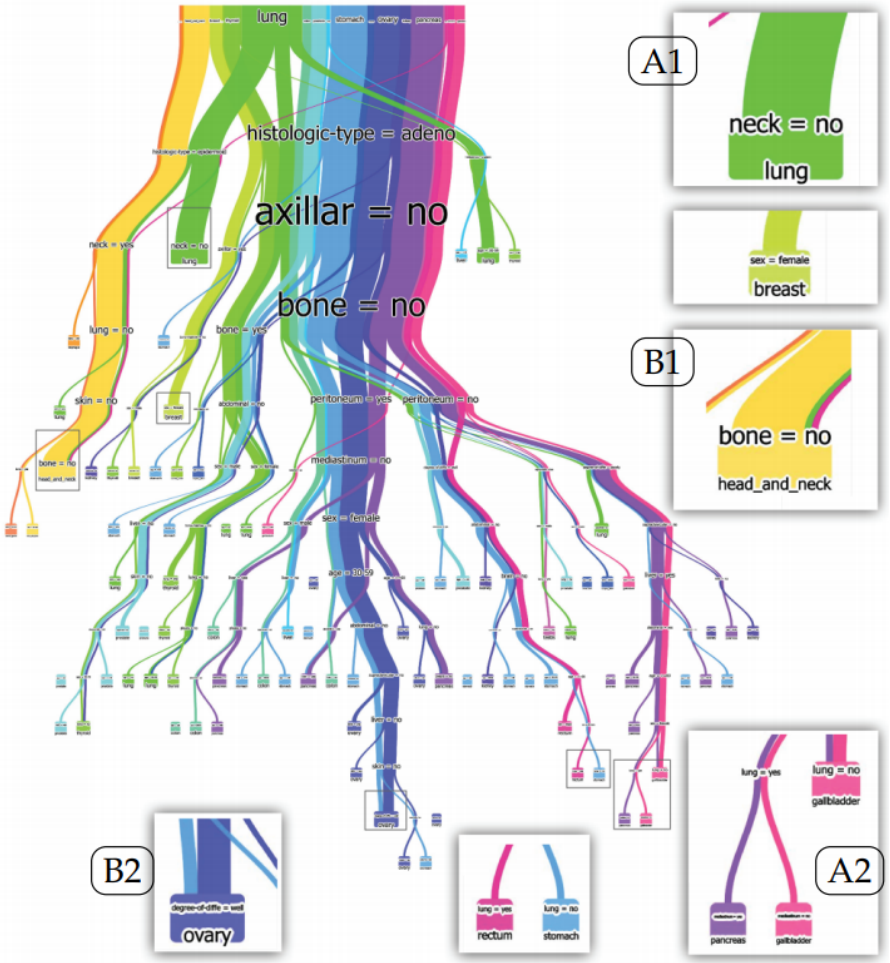
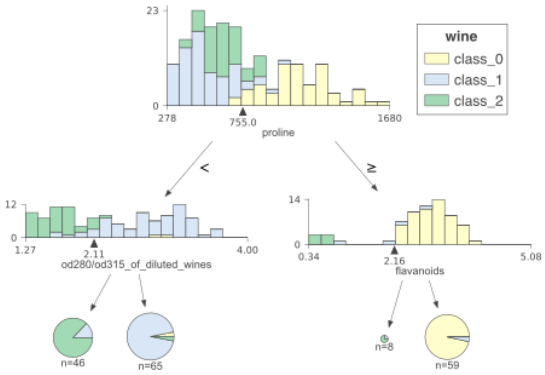


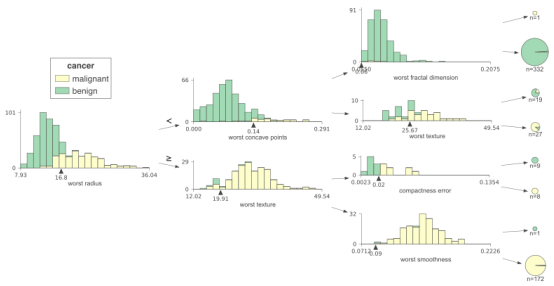
Figure 2.20: BaobabView system showing the partitioning of instances. Correct predictions are visible (A1, A2) and mis-classifications stand out (B1, B2), enabling users to immediately identify areas where the decision tree performs poorly and may require refinement.

Hybrid and Multi-View Approaches An emerging trend involves the combination of multiple visualization paradigms within unified interfaces. Van den Elzen et al. [42] demonstrate the integration of tree structure visualization with data distribution views, confusion matrices, and alternative dataset exploration. Similarly, Fisher et al. [56] discuss the broader principles of coordinated multiple views, including overview and detail patterns and multiform visualizations that could be applied to decision tree contexts.

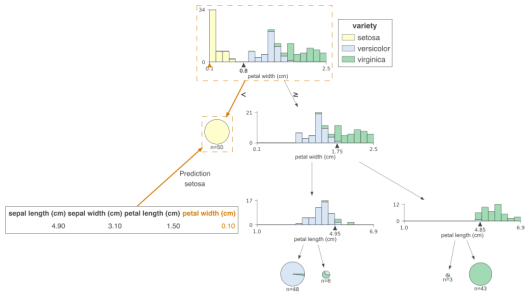
This comprehensive landscape reveals both the richness of available techniques and the fragmentation of approaches, with limited systematic evaluation of when different visualization paradigms are most effective for specific interpretability



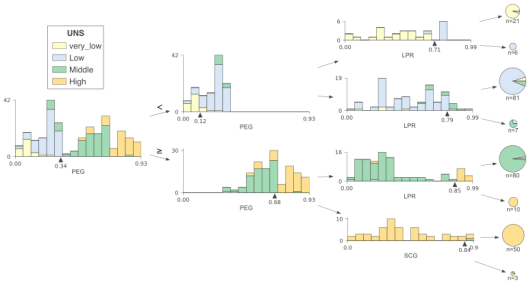
(a) Wine 3-class top-down orientation



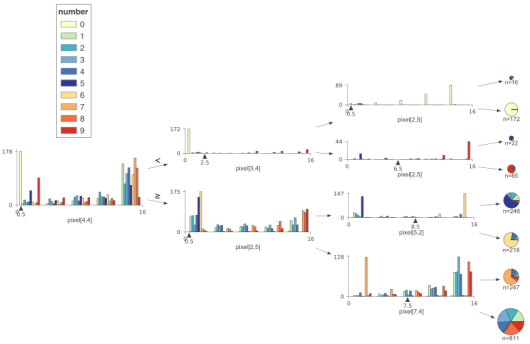
(b) Breast cancer 2-class left-to-right



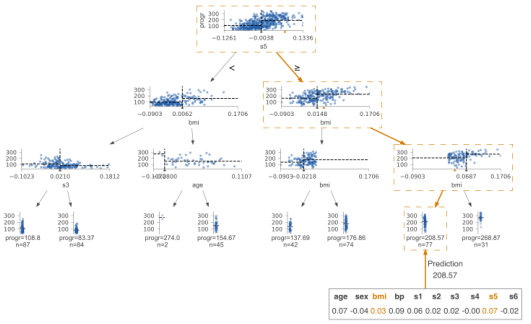
(c) Iris 3-class showing a prediction



(d) User knowledge rating 4-class

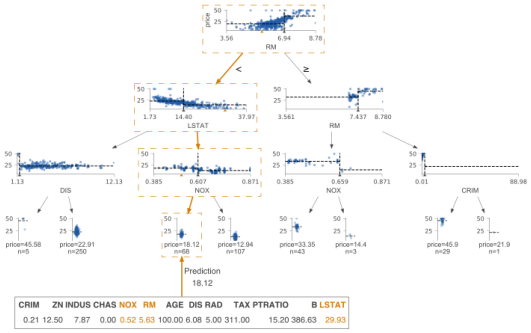


(e) Digits 10-class

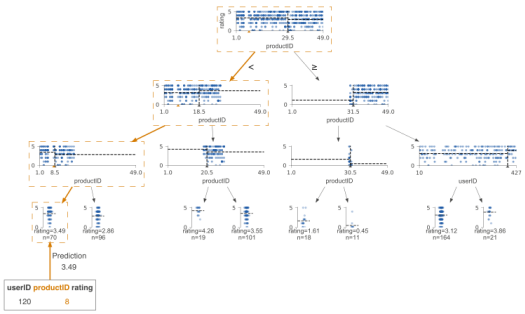


(f) Diabetes showing a prediction

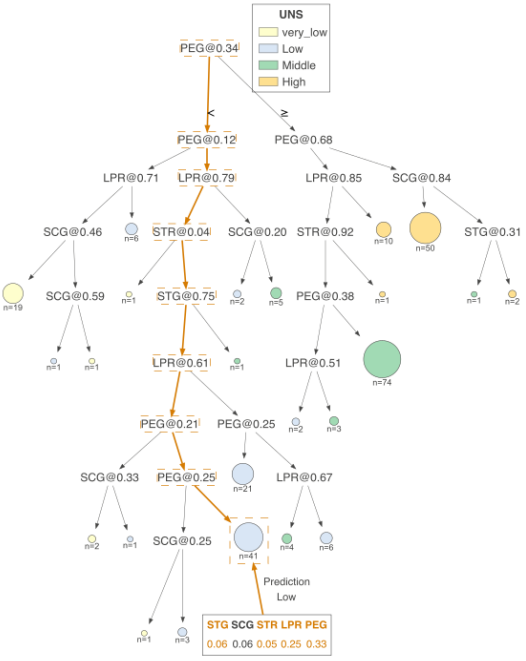
Figure 2.21: Comprehensive comparison of decision tree visualization tools, including scikit-learn, dtreeviz, R packages, SAS, and IBM Watson. The comparison demonstrates how dtreeviz enhances traditional approaches by showing feature distributions as overlapping stacked histograms within decision nodes and using proportional leaf sizes, while maintaining clear decision boundary visualization.



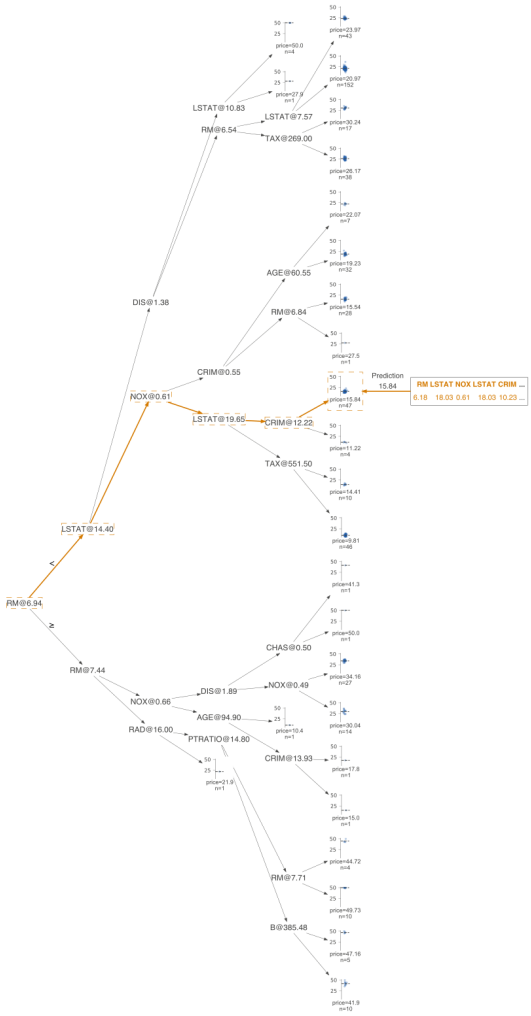
(g) Boston showing a prediction



(h) Sweets showing a prediction



(i) User knowledge rating 4-class non-fancy



(j) Diabetes non-fancy

Figure 2.21 (continued)

goals.

Visualization Design Patterns in Decision Tree Literature

The comprehensive analysis by Streeb et al. [47] reveals clear patterns in how decision trees are visualized across 152 publications. Their systematic categorization, presented in Tables 2.5 and 2.4, demonstrates the dominance of certain approaches in current practice.

Visual Design Preferences The survey reveals a strong preference for traditional node-link diagrams, which appear in 110 of the 152 publications, making them by far the most common approach to representing tree structure. Other tree visualization techniques show much lower adoption rates: treemaps appear in only 9 publications, icicle plots in 10, and pipe diagrams in 6. This concentration suggests that despite advances in hierarchical visualization techniques, the research community largely agrees on the fact that conventional tree representations are more effective.

For additional visual components beyond the core tree structure, the analysis shows frequent use of standard statistical visualizations. Bar charts and histograms appear in more than 20 publications, line charts in almost 15, and scatter plots in more than 30. These findings indicate that researchers commonly augment tree visualizations with familiar chart types.

Node Design and Encoding Strategies A consistent pattern across multiple systems involves encoding sample sizes through node dimensions. Wang et al. [38] demonstrate **funnel-like node representations** where node width corresponds to the percentage of training samples, enabling users to quickly identify important decision points and assess model robustness. Similarly, Wang et al. [38] employ node size variations to represent the significance of different tree components within the Rashomon set [47].

Effective node design integrates multiple information layers without overwhelming users. Elzen et al. [42] showcase nodes containing **class distributions via streamgraphs**, **attribute values**, and **split conditions**, all within compact visual representations, as demonstrated in Figure 2.12. Ming et al. [51] demonstrate how **compact clause representations** can be combined with **data distribution previews** using histograms for continuous features and bar charts for categorical features.

Color Coding Strategies and Accessibility Consistent color coding for target classes emerges as a fundamental pattern. Parr et al. [46] emphasize the use

of **handpicked colorblind-safe palettes**, with distinct schemes for different numbers of target categories (2 through 10). This approach ensures accessibility while maintaining visual coherence across complex trees.

Several systems employ color to highlight important features and decision paths. Elzen et al. [42] use **attribute-based node coloring** to reveal which features are most frequently used in splitting decisions. Parr et al. [46] use **orange wedge highlighting** for decision paths, making the comparison operations easily visible during tree traversal.

Advanced color strategies include the use of transparency and gradients to convey additional information layers. Ming et al. [51] employ **higher opacity for satisfied conditions** and **gradient coloring for confidence intervals**. Mrva et al. [43] use **transparent rendering for filtered nodes**, allowing users to focus on specific classes while maintaining context.

Edge Representation and Data Flow Visualization A recurring best practice involves encoding data flow through edge thickness. Elzen et al. [42] pioneer the use of **color-banded edges with variable width** to visualize the volume of data flowing through each decision branch. This technique provides immediate visual feedback about data distribution across the tree structure, as demonstrated in Figure 2.20.

Multiple systems implement strategies for highlighting active decision paths. Parr et al. [46] use **thicker, colored edges** for paths involved in specific predictions, while Joesquito [55] employs **animated traversal** with color transitions to show decision progression in real-time.

Consistent edge labeling practices include positioning Y/N or condition labels at strategic points along edges. Joesquito [55] demonstrates optimal **midpoint label placement** with appropriate spacing from both source and target nodes.

Layout and Spatial Organization Maintaining consistent spatial relationships across tree levels represents a fundamental design principle. Schulz et al. [41] identify **layer-based organization** as essential for preserving tree topology understanding. Elzen et al. [42] implement **weighted edge algorithms** that optimize readability while minimizing edge crossings.

Advanced systems implement layout algorithms that adapt to tree characteristics. Wang et al. [38] introduce **focus+context techniques** [53] using Sunburst diagrams [52] that allow seamless transitions between overview and detail levels.

Integration of Data Distributions One of the best practices found involves directly integrating data distribution information within tree visualizations. Parr et al. [46] demonstrate the effectiveness of **strip plots and scatter plots** embedded

within decision nodes, showing actual data distributions for split conditions, as illustrated in Figure 2.21. This approach eliminates the cognitive burden of connecting abstract tree structure with the underlying data patterns.

Ming et al. [51] implements **conditional distribution visualization** that shows feature distributions given that previous rules are not satisfied, providing crucial context for understanding rule interactions. Mrva et al. [43] extend this concept with **3D histograms** comparing parent and child node distributions, as shown in Figure 2.13.

Interaction Design Patterns Effective interaction design balances information density with usability through progressive disclosure mechanisms. Ming et al. [51] implement **details-on-demand** through hover interactions and expandable cells, while Wang et al. [38] provide **repositionable tree windows** for comparative analysis, as demonstrated in Figure 2.14.

Multiple systems implement sophisticated filtering capabilities. Ming et al. [51] provide **rule filtering by support and confidence**, while Mrva et al. [43] enable **class-based filtering** with transparency adjustments. Elzen et al. [42] demonstrate **real-time threshold modification** with immediate visual feedback, as shown in Figure 2.19.

Advanced systems integrate tree visualization with complementary views. Elzen et al. [42, 57] coordinate tree views with **confusion matrices**, **attribute distributions**, and **alternative dataset views**, as demonstrated in Figure 2.12. Fisher et al. [56] establish theoretical foundations for such coordination through **brushing and linking** techniques.

Typography and Readability Subtle but important patterns emerge in text presentation. Parr et al. [46] advocate for **gray rather than black text** to reduce eye strain, while maintaining sufficient contrast for accessibility. **Hairline borders** and **outlined shapes** improve element separation without overwhelming visual complexity.

Effective systems organize textual information hierarchically within nodes. Ming et al. [51] demonstrate **matrix organization** where feature order remains consistent across rules, facilitating rapid visual comparison and rule understanding, as illustrated in Figure 2.17.

These patterns collectively establish a foundation for effective surrogate models visualization, emphasizing the importance of integrating multiple information dimensions while maintaining visual clarity and supporting diverse interaction paradigms. The latter analysis of various pieces of literature provides a comprehensive framework for future decision tree visualization development, highlighting both well-established practices and emerging innovative approaches that push the

boundaries of traditional tree representation techniques.

2.2.3 Existing XAI Visual Analytics Tools

The escalation of complex machine learning models across various domains has prompted the development of visual analytics tools designed to make these models more interpretable and trustworthy. While the theoretical foundations of eXplainable Artificial Intelligence have been extensively studied, the practical implementation of these concepts through interactive visual systems represents a critical need to connect the algorithmic explanations to human understanding [58, 59].

As highlighted by Spinner et al.[58], there exists a significant gap between theoretical XAI frameworks and their use in practical systems that support real-world model development and deployment workflows. This gap becomes particularly pronounced when considering the diverse user groups that interact with machine learning models, from domain experts with limited technical background to experienced data scientists in need of algorithmic insights [60].

The landscape of XAI visual analytics tools has evolved to address different aspects of the explainability challenge, ranging from comprehensive frameworks that integrate multiple explanation methods [58] to specialized tools that focus on specific explanation paradigms such as local explanations [61, 62] or surrogate model visualization [63]. However, most existing approaches tend to address either global model understanding or local instance explanations, and in the latter case, they rarely provide integrated views that make the relationship between the generated neighborhood and the resulting explanation obvious.

In the context of this thesis, which focuses on developing an interactive explanation system for explainability methods which involve both a synthetically generated neighborhood and a surrogate model, understanding the current state of visual eXplainable Artificial Intelligence tools is crucial. The integration of dimensionality reduction techniques (such as UMAP projections) with decision tree visualization presents specific design challenges that few existing tools have addressed.

This section examines the current state of the art of XAI visual analytics tools, organizing them into several categories based on their primary focus and methodological approach. We analyze comprehensive frameworks that attempt to unify multiple XAI methods [58, 60], specialized tools for local explanation analysis [61, 62], surrogate model visualization approaches [63], and industrial deployment considerations [59].

Through this analysis, we identify key design patterns, interaction paradigms, and visualization techniques that inform the development of effective interfaces. Particular attention is paid to tools that employ scatter plot visualizations for neighborhood representation and surrogate model interfaces for rule exploration,

as these components are central to the proposed system. Furthermore, we examine how existing tools handle the coordination between different explanation views and support user interaction workflows that allow both exploratory analysis and targeted explanation refinement.

Comprehensive XAI Visual Analytics Frameworks

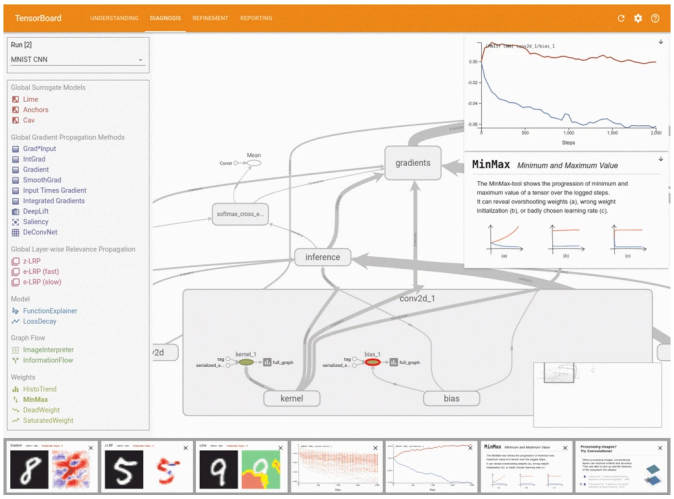
Comprehensive XAI visual analytics frameworks represent an ambitious approach to addressing the XAI field. Unlike specialized tools that focus on particular explanation methods or user scenarios, comprehensive frameworks attempt to integrate multiple XAI techniques within unified platforms that support diverse workflows and user needs. Two notable examples are **explAIner** and **XAutoML**, each addressing different aspects of the machine learning lifecycle while providing comprehensive approaches to model understanding and confirmation.

explAIner: Interactive Deep Learning Model Analysis This framework [58] presents a complete approach to interactive and eXplainable Artificial Intelligence, specifically designed for deep learning models and integrated in the TensorBoard ecosystem. The framework is built around an iterative XAI pipeline that structures the explanation process into three core phases: *understanding*, *diagnosis*, and *refinement*.

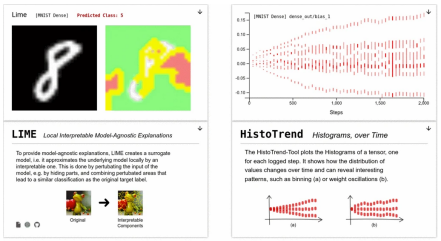
The central concept in this framework are the *explainers*, which serve as a modular building block that take one or more model states as input, applies an XAI method, and outputs either explanations (visualizations, verbalizations, surrogate models) or transition functions for model refinement. The framework distinguishes between single-model explainers that analyze individual model states and multi-model explainers that perform comparative analysis across different model configurations. This modular design enables the integration of diverse explanation methods including LIME [13], LRP [15], SHAP [14], gradient-based methods, and custom implementations.

The framework incorporates eight global monitoring and steering mechanisms that support the overall explanation process, including model quality monitoring, search space exploration, data shift scoring, comparative analytics, provenance tracking, XAI Strategies, knowledge generation, and reporting capabilities. These mechanisms provide essential infrastructure for maintaining context and continuity across extended explanation sessions, allowing users to track their investigation process and build cumulative understanding over time.

From a visualization perspective, explAIner employs a toolbox-based interface, shown in Figure 2.22a, where explainers are arranged by abstraction level, allowing users to progressively examine the results from high-level model understanding to detailed component analysis. The system's integration with TensorBoard's



(a) The diagnosis dashboard. Explainers are arranged in a toolbox-like interface, ordered descending, from high-abstraction to low-abstraction. The graph visualization provides an overview of the full model and allows for node selection. Explanations are shown in the upper toolcard, while information about the explainer is displayed beneath. The provenance bar contains cards from interesting findings



(a) LIME (high-abstraction explainer) (b) HistoTrend (low-abstraction explainer)

(b) Information cards showing results for different explainers (top) with the corresponding descriptions for the explainer itself (bottom).

Figure 2.22: eXplainer interface.

graph visualization provides a familiar foundation for users already working with deep learning models, while overlay cards, represented in Figure 2.22b, present explanatory results and supplementary information based on context.

XAutoML: Transparency for Automated Machine Learning While explAIner focuses on manually developed deep learning models, **XAutoML** [60] addresses the understanding and of automated machine learning (AutoML) systems.

The widespread use of AutoML tools produced high-performing models that do not reveal the processes or design decisions behind the obtained pipelines. XAutoML tackles this challenge through a comprehensive visual analytics approach embedded within JupyterLab, providing four primary views that cover different aspects of the AutoML process. As shown in Figure 2.23, the *optimization overview* provides high-level insights into the AutoML run, including performance trajectories over time, candidate distribution by performance, and ROC curve comparisons for selected models. The *candidate inspection* view enables detailed analysis of individual ML pipelines, through XAI techniques such as surrogate models, feature importance analysis, and partial dependence plots.

Figure 2.24 demonstrates the comprehensive nature of XAutoML's explanation capabilities, showing how the system integrates multiple established XAI techniques

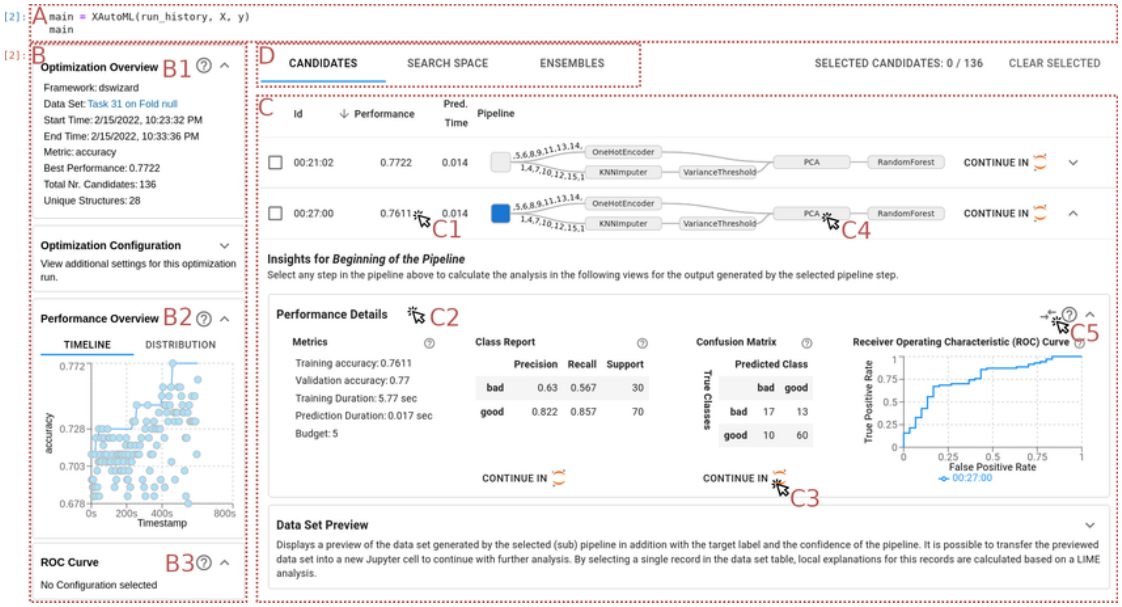
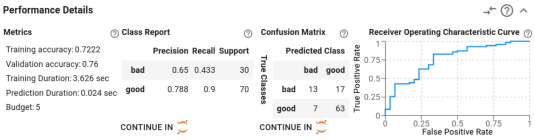


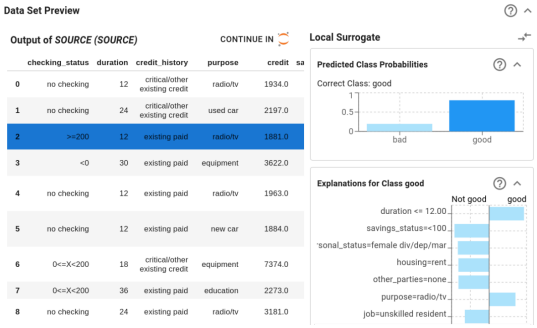
Figure 2.23: Overview of XAutoML. The visualization is integrated with Jupyter and can be accessed with a few lines of code (A). On the left side (B), the optimization overview provides basic statistics about the optimization run (B1), a scatter plot of the accuracy of all candidates over time (B2), and a receiver operating characteristic (ROC) curve of selected candidates (B3, hidden). The leaderboard view (C, partially hidden) provides a comprehensive overview of all evaluated candidates. Users can open single candidates (C1) in an overlay on the right-hand side to reveal detailed information about them (only partially visible). The candidate details contain various boxes grouping related information together. In the performance details view (C2), performance metrics and basic performance visualizations are available. By clicking the Continue in Jupyter button (C3), the according information can be exported to a new Jupyter cell. Users can access the search space and ensemble inspection via the tabs at the top (D).

within a unified interface. The performance details view (a) provides standard validation metrics alongside confusion matrices and ROC curves. The global surrogate view (b) employs decision trees as interpretable approximations of complex models, with user-controllable complexity through a slider interface. The dataset preview and local surrogate view (c) combines data inspection with SHAP-based feature attributions for selected instances. XAutoML is designed to work smoothly with existing data science workflows by embedding into JupyterLab and offering flexible export options, so users can easily move models, datasets, and analysis results back into their coding environments. The system employs a hierarchical information presentation strategy that helps users avoid cognitive overload while revealing technical details step by step. Domain experts can configure simplified views that hide complex ML details, while comprehensive technical information,

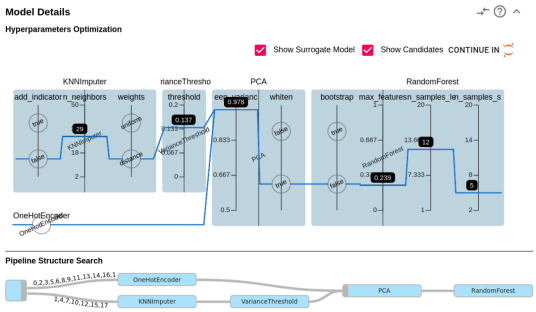


(a) Performance details. From left to right some basic metrics, a class report, a confusion matrix and a ROC curve is displayed.

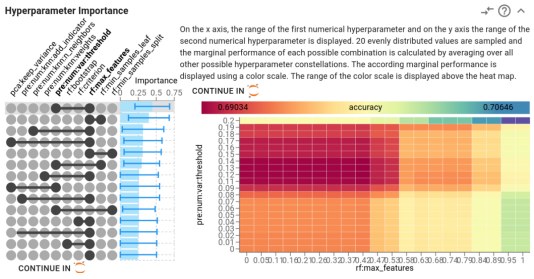
(b) Global surrogate. A decision tree is rendered as a global surrogate. Using the top slider, the tree complexity can be controlled.



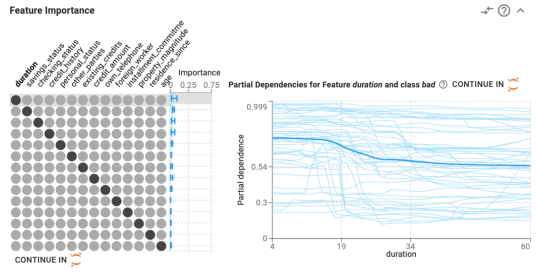
(c) Dataset preview and local surrogate. The left-hand side displays a preview of the input dataset. On the right-hand side feature attributions for the selected sample are displayed.



(d) Model details. At the top, the selected hyperparameter configuration is displayed in a CPC. At the bottom, the aggregated structure search graph up to this candidate is rendered.



(e) Hyperparameter importance with well-performing regions. On the left-hand side hyperparameter (pairs) are ranked by their importance. On the right-hand side a heat map with well-performing regions of the currently selected hyperparameter pair is displayed.



(f) Feature importance with PDP and ICE. On the left-hand side the features are ranked by their importance. On the right-hand side a PDP and ICE plot show the correlation between the feature value and predicted class for the currently selected feature.

Figure 2.24: Overview of visualizations used to explain and validate a single candidate.

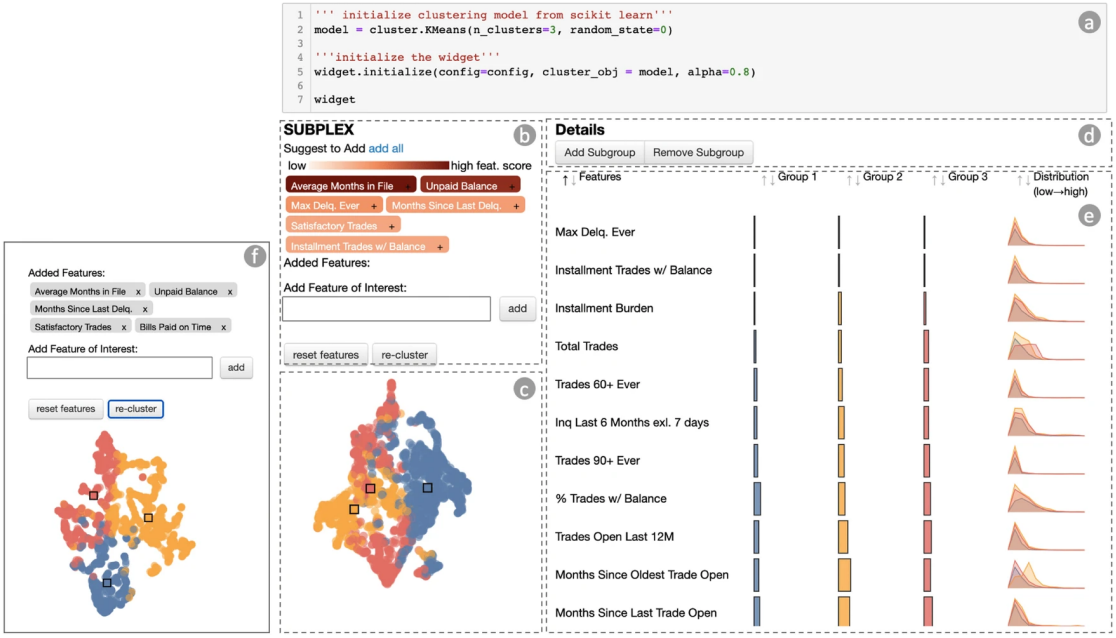


Figure 2.25: SUBPLEX contains five linked views: (a) code block, (b) cluster refinement view, (c) projection view, (d) subpopulation creation panel, (e) local explanation detail view

including hyperparameter configurations, pipeline structure visualizations, and detailed performance metrics, is available to more experienced users.

Local Explanation Analysis Tools

Specialized tools that focus specifically on local explanation analysis have emerged to address the challenge of understanding individual predictions and the underlying decision pattern. Two notable examples of specialized local explanation analysis tools are SUBPLEX and FIPER, each addressing different aspects through innovative visual analytics approaches.

SUBPLEX: Subpopulation-Level Local Explanation Analysis [61] presents a new approach to understanding local model explanations by analyzing them at the subpopulation level rather than treating individual explanations in isolation. The tool addresses a fundamental limitation in traditional local explanation analysis: while methods like LIME [13] and SHAP [14] provide feature importance vectors for individual instances, aggregating these explanations across entire datasets through simple averaging can be misleading, as important patterns may only emerge within specific subgroups of the data.

The core innovation of SUBPLEX lies in its human-in-the-loop framework

that combines automatic clustering and projection techniques with intelligent user guidance to enable pilotable subpopulation analysis. The system addresses the challenge that straightforward application of clustering and projection algorithms to local explanation vectors often fails to reveal meaningful patterns due to the specific characteristics of explanation data, including high dimensionality, sparsity, and varying scales across features.

SUBPLEX implements a three-stage pipeline for subpopulation analysis: *generation*, *exploration*, and *interpretation*. During the generation phase, the system performs automatic clustering of local explanation vectors and provides feature suggestions to guide users toward potentially interesting subpopulations. The clustering process employs K-means for computational efficiency, while UMAP projection is used to visualize the distribution of explanations in a 2D space, enabling users to understand the spatial relationships between different explanation patterns.

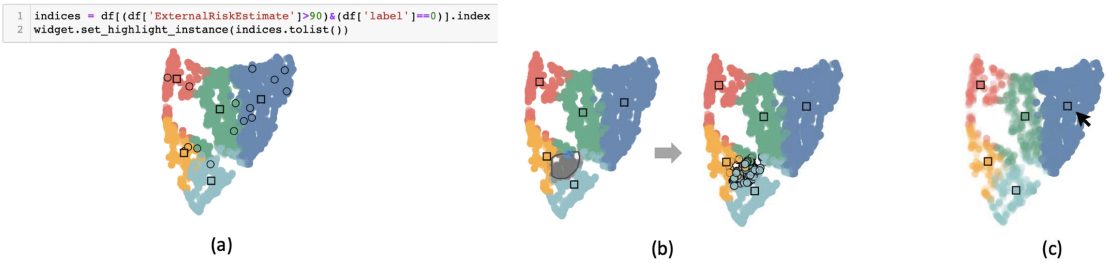


Figure 2.26: Three methods of selecting/creating a subpopulation for inspection. (a) Highlight instances by coding. (b) Select instances by brushing. (c) Highlight a cluster by clicking centroid.

The exploration phase supports both automatic subpopulation discovery and manual subpopulation creation through multiple interaction mechanisms, as illustrated in Figure 2.26. Users can highlight instances by coding, select regions through brushing interactions, or click cluster centroids to define subpopulations of interest. This flexibility acknowledges that domain experts often have specific hypotheses about interesting subgroups that may not be captured by automatic clustering algorithms.

During the interpretation phase, SUBPLEX enables comparative analysis of explanation patterns across subpopulations through bar charts that visualize aggregated local explanations. The system provides export capabilities that allow users to extract intermediate results as variables within Jupyter notebooks, supporting integration with broader data science workflows.

From a visualization perspective, SUBPLEX employs a five-view coordinated interface embedded within Jupyter notebooks, as shown in Figure 2.25. The projection view shows the distribution of local explanations in 2D space using UMAP,

with color encoding to distinguish different subpopulations. The cluster refinement view provides feature suggestions and enables iterative cluster refinement through feature selection. The subpopulation creation panel supports manual manipulation of subgroups, while the local explanation detail view presents aggregated explanation patterns for selected subpopulations.

FIPER: Hybrid Rule and Feature Importance Visualization [62] addresses a different challenge in the analysis of local explanations by combining rule-based explanations with feature importance methods within a merged visual interface. The tool recognizes that rule-based explanations provide comprehensive logical predicates that fully describe the conditions leading to specific predictions, but it can be cognitively demanding to process. Conversely, feature importance methods like SHAP [14] provide lightweight, easily interpretable rankings even though they offer less detailed information about decision boundaries.

FIPER's central contribution is its hybrid approach that uses feature importance rankings to organize and prioritize the presentation of the rule-based explanation. The system employs LORE [26, 64] for rule generation and SHAP [14] for feature importance calculation, though the architecture is designed to accommodate other rule-generating algorithms (such as Anchors [12]) and alternative feature importance methods (such as LIME [13]).

As shown in Figure 2.27, the visualization organizes the explanation space into two coordinated panels. The left panel presents feature importance weights sorted by their absolute values, using color coding to indicate positive (blue) or negative (magenta) contributions to the prediction. The right panel visualizes rule predicates in an order that follows the feature importance ranking, ensuring that the most relevant features are presented first to users.

FIPER employs differentiated visualization strategies based on feature types to maximize information density while maintaining clarity. For categorical features, the system uses stacked bar charts to show the part-of-the-whole relationship of possible values, with diamond markers indicating the observed value for the current instance. For numerical features, box plots provide a compact visualization of data distributions, including quartiles, median, and range information, with highlighted intervals showing the rule predicate ranges.

The system incorporates two forms of interactivity to support the exploration, as illustrated in Figure 2.28. Users can dynamically restrict the view to show only features mentioned in rule predicates, following the principle of providing easy access to relevant information. Additionally, hovering interactions reveal detailed distribution information for specific features, including cardinality for categorical variables and statistical summaries for numerical variables.

A controlled user study comparing FIPER with textual LORE output and

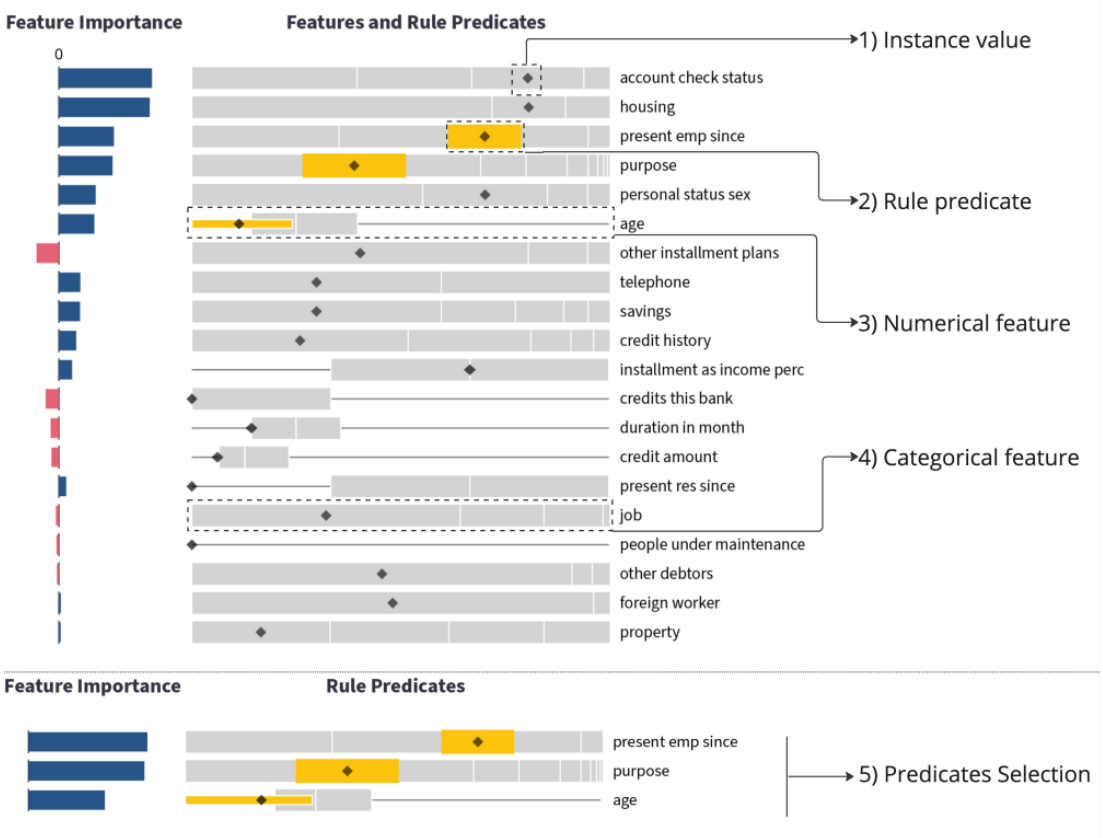


Figure 2.27: FIPER visualization of one instance of the German Credit Risk dataset. Top panel: Attributes are sorted by the absolute value of feature importance (FI). Categorical attributes are represented as stacked bar charts. Numerical values are represented as box plots with quartile information. The intervals contained in the predicates of the rule are highlighted in yellow. Bottom panel: Filtered view of the visualization, showing only the attributes referred to in the rule premise, demonstrating the system's ability to focus user attention on rule-relevant features.

an enhanced XAI library visualization demonstrated that while FIPER required slightly more time for task completion, it substantially improved accuracy across all tasks, even with complex instances. User feedback indicated that FIPER was considered more understandable and valuable, with 10 out of 15 participants selecting it as their preferred visualization approach.

The study also revealed new insights into user preferences. While FIPER demonstrated superior performance for datasets with many features, some participants suggested that simpler visualizations might be more appropriate for datasets with fewer attributes. This finding highlights the importance of adaptive visualization strategies that can scale appropriately with data complexity.

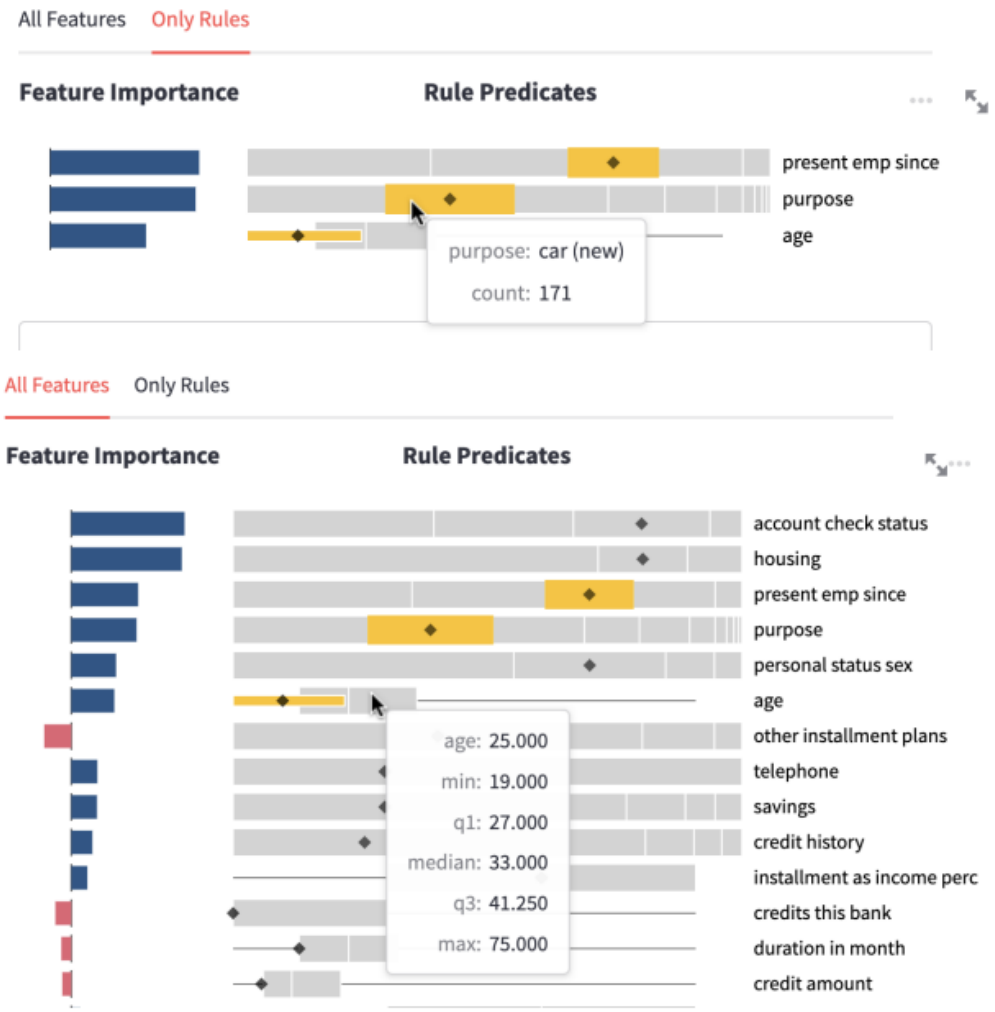


Figure 2.28: Detailed feature information accessible by hovering over specific features: Top panel shows tooltip for a categorical data type, displaying the feature's actual value with its class cardinality information. Bottom panel shows a tooltip for a numerical data type, presenting statistical central values including minimum, maximum, median, first quartile (Q1), and third quartile (Q3).

Surrogate Model Visualization Tools

Surrogate model visualization represents a distinct category within vXAI, focusing specifically on the interpretable models that approximate the behavior of a black box machine learning system. Unlike local explanation analysis tools that examine individual predictions or comprehensive frameworks that integrate multiple explanation modalities, surrogate model visualization tools focus on creating and visualizing simplified models that capture the essential decision-making patterns.

DeforestVis: Decision Stump-Based Surrogate Analysis [63] presents an innovative approach to the visualization of the surrogate model employing decision stumps, decision trees of one level, generated through the AdaBoost [65] algorithm. This approach addresses a critical limitation in traditional surrogate model approaches: while full decision trees can become prohibitively complex when attempting to approximate sophisticated target models, and rule sets can become unwieldy with numerous if-else statements, decision stumps provide a middle ground that maintains interpretability while capturing essential decision patterns.

The system's core lies in its recognition that ensemble methods like AdaBoost naturally decompose complex decision boundaries into collections of simple, weighted decision stumps. Rather than attempting to visualize a single complex surrogate model, DeforestVis takes advantage of this decomposition to provide multiple levels of abstraction: users can examine individual stumps for detailed understanding, analyze feature-based aggregations for intermediate insights, or observe overall model behavior for high-level comprehension.

DeforestVis implements a five-view coordinated interface that supports a comprehensive workflow for surrogate model analysis, as illustrated in Figure 2.29. The *surrogate model selection* view enables users to explore the complexity-fidelity trade-off by incrementally adding decision stumps to the ensemble while monitoring performance metrics. This view is particularly useful for its visualization of the relationship between model complexity (number of stumps) and surrogate fidelity (accuracy in approximating the target model), allowing users to identify optimal points where additional complexity provides diminishing returns. The *behavioral model summarization* view aggregates decision stumps by feature, providing users with feature-level insights into the target model's decision patterns. This aggregation approach addresses the cognitive challenge of interpreting large numbers of individual stumps by organizing them according to the features they split on, enabling users to understand which features the target model considers most important and how their decision boundaries are constructed. The *rule overriding* and *decision comparison* views enable users to manually adjust decision thresholds within individual stumps and immediately observe both local and global impacts of their modifications. The local impact analysis shows how threshold changes affect specific training instances, while global impact analysis demonstrates broader effects on model predictions across the entire dataset. This allows domain experts to fine tune the automatically generated rules, potentially improving both model performance and interpretability. The *test set results* view provides validation capabilities that allow users to assess the effectiveness of their manual modifications on unseen data. This view supports case-by-case analysis, enabling users to understand why specific instances received particular classifications based on the

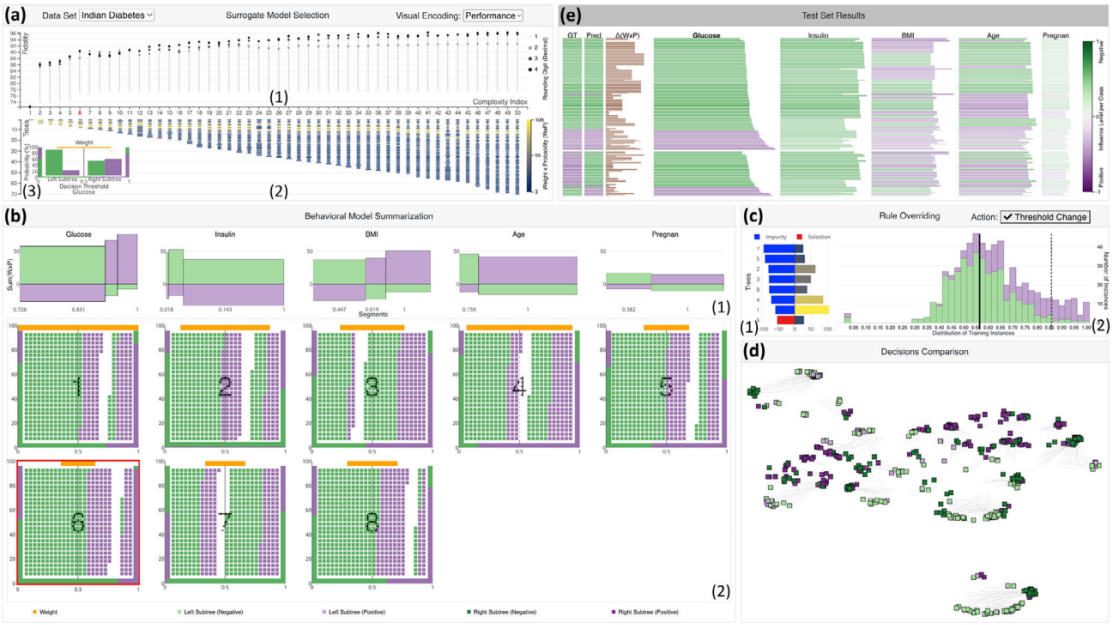


Figure 2.29: Components of DEFORESTVIS: (a.1) lollipop plot shows data-rounding effects in the fidelity score for four different decimal digit precisions; (a.2) dot plot with lines of various widths (unique rules/stumps > original > duplicated) and colors (visual encoding: performance, that is, weighted probability (W × P)) that explains complexity increase as more decision stumps get added and (a.3) selective stump-based explanation; (b.1) segmented bar chart tells the predictive outcome and power of each segment based on automatically computed thresholds and (b.2) detailed stump-based explanation grid; (c.1) bar chart shows the impurity and weighted probability of each decision stump; (c.2) histogram shows the active rule's threshold and distribution of training instances; (d) projection aggregates the global behaviour of instances; colour shows the local behaviour according to the currently selected decision stump; and (e) fragmented bar chart shows the per-feature contribution and influence level for each test case.

contribution of each feature as encoded in the ensemble of decision stumps.

From a technical perspective, DeforestVis employs the Explainable Boosting Machine (EBM) as its target model, chosen specifically because it produces systematically fewer decision rules compared to other ensemble methods like XGBoost or Random Forest. However, the system's workflow remains model-agnostic, as AdaBoost-based surrogate models can approximate the behavior of any machine learning model, making the approach broadly applicable across different modeling contexts. The system's evaluation through expert interviews with data analysts and model developers revealed key insights. Experts particularly appreciated the multi-level transparency that DeforestVis provides, enabling both top-down analysis (starting from overall model behavior) and bottom-up investigation (beginning with individual decision stumps).

Chapter 3

Problem statement

The primary objective of this thesis is to conceptualize and implement an interactive visualization framework designed for the exploration and analysis of the synthetic neighborhood and surrogate model algorithmically generated. Stable and Actionable LOcal Rule-based Explanation method (LORE$_{sa}$) [26], extending LORE [64], was the chosen candidate for the making and testing of the thesis's product. The visualization tool aims to improve the understanding of the user, distancing itself from the library originally provided output regarding rules and counterfactual rules. The interpretable representations are required by domain experts, data scientists, and end-users who seek to understand and confirm machine learning model decisions.

In the following Section 3.1, we will undertake an examination of the mechanisms by which LORE$_{sa}$ generates the aforementioned synthetic neighborhood and constructs the interpretable surrogate model. Furthermore, the implementation choices, design decisions, and representational frameworks that the current version of the library adopts for the representation, organization, and presentation of the generated explanations.

The output produced by LORE$_{sa}$ is fundamentally composed of two complementary components: the extracted rules and the corresponding counterfactual rules, other than the fidelity of the explanation and a feature importance array. The extracted rules are supposed to constitute the logical conditions that characterize the decision patterns and feature interactions that lead to the predicted outcome for the instance under examination. These rules provide an explanation framework, clarifying the specific combinations of feature values and thresholds that support the model's prediction. Contrarily, the counterfactual rules represent the logical negation or alternative conditions that would result in different model predictions, offering a perspective on the decision boundaries and helping users understand what changes would be necessary to alter the predicted outcome.

Building upon this foundation and the identified limitations in current ap-

proaches, Chapter 4 presents the new design proposal for the visualization of the LORE$_{sa}$ outputs. The proposed system will enable users to navigate between different levels of abstraction, from high-level overview representations of the entire explanation space to detailed, instance-specific rule visualizations that facilitate exploration and confirmation of the generated explanations.

3.1 LORE$_{sa}$ stable and actionable LOcal Rule-based Explanations

LOcal Rule-based Explainer (LORE), introduced by Guidotti et al.[64], addresses several fundamental limitations of LIME and SHAP by shifting from feature importance to *symbolic decision rules*. The original LORE employs a genetic algorithm for neighborhood generation, creating a more faithful and dense representation of the local decision space compared to LIME's random sampling approach [25]. LORE$_{sa}$ extends this foundation with enhanced stability and actionability features.

LORE and LORE$_{sa}$'s core mechanism is that given a black-box model b and instance x with prediction $b(x) = y$ the methods generate synthetic neighborhoods through genetic optimization. However, while the original LORE generates a single neighborhood Z and trains one decision tree classifier g, LORE$_{sa}$ generates multiple neighborhoods and employs a bagging-like approach, creating N decision trees that are subsequently merged into a single interpretable predictor. From the resulting decision tree structure, both methods extract (i) a *factual decision rule* corresponding to the path followed by instance x to reach decision y, and (ii) a set of *counterfactual rules* indicating conditions that would alter the prediction outcome. LORE$_{sa}$ enhances the counterfactual extraction with actionability constraints, filtering rules based on user-specified constraints U on immutable features [26, 25].

Algorithm 1: LORE$_{sa}$(x, b, K, U)

Input: x - instance to explain, b - black-box, K - knowledge, U - constr.
Output: e - (counter)factual explanation of x

$D \leftarrow \emptyset$; `// init. empty set of decision trees`
for $i \in \{1, \ldots, N\}$ **do**
  $Z^{(i)}_{=} \leftarrow \text{genetic}(x, \text{fitness}_{x=}, b, K)$; `// neighborhood generation`
  $Z^{(i)}_{\neq} \leftarrow \text{genetic}(x, \text{fitness}_{x\neq}, b, K)$; `// neighborhood generation`
  $Z^{(i)} \leftarrow Z_{=} \cup Z_{\neq}$; `// merge neighborhoods`
  $Y^{(i)} \leftarrow b(Z^{(i)})$; `// apply black-box`
  $d^{(i)} \leftarrow \text{buildDecisionTree}(Z^{(i)}, Y^{(i)})$; `// build decision tree`
  $D \leftarrow D \cup \{d^{(i)}\}$; `// add decision tree to list`
$c \leftarrow \text{mergeDecisionTrees}(D)$; `// merge decision trees`
$r = (p \rightarrow y) \leftarrow \text{extractDecisionRule}(c, x)$; `// factual rule`
$\Phi \leftarrow \text{extractCounterfactuals}(c, r, x, U)$; `// extract actionable counterfactual`
return $e \leftarrow r, \Phi$;

Unlike LIME's random perturbation, LORE$_{sa}$ use genetic algorithms that optimize neighborhood generation through fitness functions (3.1 and 3.2) that minimize distances to the decision boundary [26]. This approach yields neighborhoods that are denser in boundary regions, providing more accurate approximations of local black-box behavior compared to uniform random generation. Figure 3.1 demonstrates that genetic approaches produce neighborhoods with better decision boundary characterization [26].

$$\text{fitness}^{x}_{=}(z) = I_{x\neq} + d(x, z) + l(b_p(x), b_p(z)) \tag{3.1}$$

$$\text{fitness}^{x}_{=}(z) = I_{x\neq} + d(x, z) + (1 - l(b_p(x), b_p(z))) \tag{3.2}$$

3.1.1 Neighborhood Generation

The quality of neighborhood generation is a critical factor in determining the fidelity and reliability of local explanations. The synthetic neighborhood around the explained instance has to accurately capture the local decision boundary of the black-box model while maintaining sufficient density and diversity to allow surrogate model training. LORE$_{sa}$'s approach to neighborhood generation represents a significant improvement from traditional random sampling methods, using a genetic algorithm to optimize the quality and representativeness of the generated synthetic instances.

Genetic Algorithm for Neighborhood Generation

LORE$_{sa}$ employs a genetic algorithm, shown in Alghorithm 2 specifically designed to overcome the limitations of random sampling by optimizing neighborhood quality through evolutionary computation principles [26]. The genetic approach treats neighborhood generation as an optimization problem where the fitness of generated instances is evaluated based on their contribution to accurate local decision boundary approximation.

Algorithm 2: genetic(x, fitness, b, K)

Input: x - instance to explain, fitness - fitness function, b - black-box, K - knowledge base

Parameters: n - population size, g - nbr of generations, p_c - prob crossover, p_m - prob mutation

Output: Z - neighbors of x

$P_0 \leftarrow (x|\forall 1, \ldots, n)$; $i \leftarrow 0$; `// population init.`

while $i < g$ **do**

- $P' \leftarrow \text{crossover}(P_i, p_c)$; `// mix records`
- $P'' \leftarrow \text{mutate}(P', p_m, K)$; `// perform mutations`
- $S \leftarrow \text{evaluate}(P'', \text{fitness}, b)$; `// evaluate population`
- $P_{i+1} \leftarrow \text{select}(P'', S)$; `// select sub-population`
- $i \leftarrow i + 1$; `// update population`

$Z \leftarrow P_i$;

return Z;

Figure 3.1 illustrates the differences between uniform random generation and genetic optimization for a black-box consisting of a random forest model on a bi-dimensional feature space [26]. The genetic approach yields a neighborhood that is denser in the boundary region of the predictor, while random generation produces scattered instances that fail to adequately characterize the decision boundary [26].

The genetic algorithm employs two complementary fitness functions to generate balanced neighborhoods. For instances with the same class as the target ($Z^i_=$), the fitness function $\text{fitness}^x_=(z)$ (3.1) optimizes the proximity to the target instance while maintaining the same classification outcome. Conversely, for instances with different classes ($Z^i_{\neq}$), the fitness function $\text{fitness}^x_{\neq}(z)$ (3.2) seeks instances that are close to the target but receive different predictions from the black-box model [26]. This dual-objective approach ensures that the neighborhood contains both confirmatory instances (supporting the original decision) and contrastive instances (illustrating alternative outcomes).

The genetic algorithm iteratively evolves populations of candidate instances

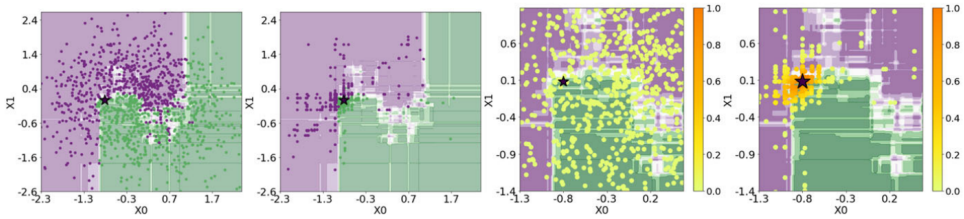


Figure 3.1: Black-box boundary: purple versus green. Starred instance x. UniforMachine Learningy random (1st) and genetic (2nd) neighborhoods. In the (3rd) and (4th) plot is reported the density with levels in the bar

through standard genetic operations, including selection, crossover, and mutation. The selection process favors instances with higher fitness scores, promoting the survival of instances that better serve the explanation objectives. Crossover operations combine successful instances to explore new regions of the feature space, while mutation introduces controlled randomness to prevent premature convergence and maintain diversity [26].

A key innovation in LORE$_{sa}$'s genetic approach involves the integration of domain knowledge through the knowledge base K, which contains information about feature distributions, including "domain of admissible values, mean, variance, probability distribution, etc.". This knowledge base guides the mutation process to ensure that generated instances remain consistent with realistic feature distributions, preventing the creation of implausible or out-of-distribution synthetic instances that would compromise explanation quality [26].

Comparative Analysis of Neighborhood Generation Methods

As previously introduced Figure 3.1 illustrates the differences between uniform random generation and genetic optimization for a black-box consisting of a random forest model on a bi-dimensional feature space [26]. The genetic approach "yields a neighborhood that is denser in the boundary region of the predictor", while random generation produces scattered instances that fail to adequately characterize the decision boundary [26].

The genetic approach produces neighborhoods with superior density characteristics in critical decision regions. "The density of the generated instances is a key factor in extracting correct and faithful local interpretable predictors and explanations" [26]. By concentrating instances near the decision boundary, the genetic algorithm enables more accurate approximation of the black-box model's local behavior, resulting in higher-fidelity surrogate models.

Unlike random sampling, "the genetic approach of LORE$_{sa}$ is driven by minimization of the fitness functions, hence less variable neighborhoods are generated"

[26]. This reduced variability contributes directly to improved explanation consistency, addressing one of the fundamental challenges in local explanation methods. The optimization-driven approach ensures that repeated explanation generation for the same instance produces consistent results, improving user trust and system reliability.

Multi-Neighborhood Ensemble Strategy

LORE$_{sa}$ extends the genetic neighborhood generation approach through a sophisticated ensemble strategy that addresses the consistency challenges inherent in local explanation methods. Rather than relying on a single neighborhood, LORE$_{sa}$ generates multiple independent neighborhoods $Z = \{Z^{(1)}, Z^{(2)}, \ldots, Z^{(N)}\}$ through repeated genetic optimization [26]. The multi-neighborhood strategy draws inspiration from ensemble learning techniques, particularly bagging methods that achieve improved predictive performance and consistency through aggregation. Each neighborhood $Z^{(i)}$ is generated independently using the genetic algorithm, providing diverse perspectives on the local decision space.

For each generated neighborhood $Z^{(i)}$, LORE$_{sa}$ constructs a decision tree classifier $d^{(i)}$ trained on instances labeled with black-box predictions. The algorithm then merges these multiple decision trees into a single interpretable predictor c through a tree merging process [26]. This ensemble approach leverages the consistency benefits of averaging multiple predictors while maintaining the interpretability advantages of tree-based models.

The multi-neighborhood ensemble strategy directly addresses the inconsistency issues that plague single-neighborhood approaches. "Bagging, boosting, and random forests achieve high predictive performances, which, in our context, means high fidelity (accuracy w.r.t. black-box decisions). Moreover, they achieve consistency of predictions by averaging the decisions of several trees" [26]. By adopting those principles, LORE$_{sa}$ achieves both improved fidelity and enhanced consistency in explanation generation.

Technical Implementation and Parameter Optimization

The practical implementation of genetic neighborhood generation involves careful consideration of multiple algorithmic parameters that influence both computational efficiency and explanation quality. LORE$_{sa}$ employs the DEAP (Distributed Evolutionary Algorithms in Python) [66] framework for genetic algorithm implementation, utilizing optimized CART [67, 68] decision tree construction through scikit-learn for efficiency [26].

The genetic algorithm operates with several critical parameters: neighborhood size, crossover probability, mutation probability, and number of generations [26].

These parameters have been empirically confirmed to provide optimal trade-offs between explanation quality and computational efficiency across diverse datasets and model types.

While genetic neighborhood generation requires greater computational resources compared to random sampling, the approach remains practically viable for real-world applications. The investment in computational resources yields substantial returns in explanation quality, fidelity, and consistency.

The genetic neighborhood generation approach in LORE$_{sa}$ represents a fundamental advancement in local explanation methodology, addressing critical limitations of random sampling while providing theoretically motivated and empirically confirmed improvements in explanation quality. The integration of evolutionary optimization principles with domain knowledge and ensemble strategies represents a robust foundation for generating high-quality synthetic neighborhoods that enable accurate and stable local explanations.

3.1.2 Rule Extraction

The transformation of complex decision tree structures into human-interpretable symbolic rules represents the last step in LORE$_{sa}$'s explanation generation process. Once the genetic algorithm has produced high-quality synthetic neighborhoods and multiple decision trees have been merged into a single interpretable predictor, LORE$_{sa}$ employs rule extraction techniques to derive both factual and counterfactual explanations that capture the logic underlying the involved black-box model decisions.

Conceptual Foundation of Symbolic Rule Extraction

The choice of decision trees as the interpretable surrogate model in LORE$_{sa}$ is fundamentally motivated by their natural capacity for symbolic reasoning [26]. Unlike feature importance methods that provide numerical weights, decision trees enable direct extraction of logical rules through their hierarchical structure. "The choice of decision trees as interpretable predictors allows for symbolic reasoning: (i) factual decision rules can readily be derived from the root-to-leaf path in a tree; and, (ii) counterfactual rules can be extracted by symbolic reasoning over a decision tree" [26, 69, 70].

Decision rules provide explanations that are fundamentally closer to human reasoning patterns compared to feature importance approaches [25]. A decision rule r takes the form $p \rightarrow y$, where p represents a premise composed of Boolean conditions on feature values, and y denotes the consequence or predicted class [26]. The premise p consists of a conjunction of split conditions of the form $a_i \in [v_i^{(l)}, v_i^{(u)}]$,

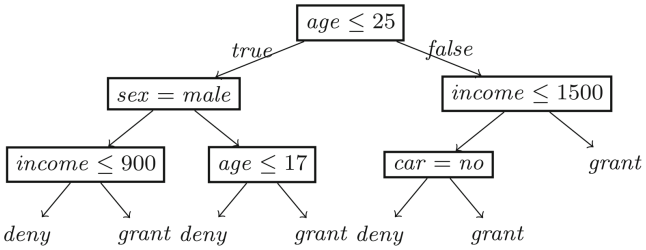


Figure 3.2: Decision tree mimicking the local behavior of a black box classifier. The tree shows split conditions leading to grant/deny loan decisions, illustrating how factual and counterfactual rules can be extracted from root-to-leaf paths.

where a_i represents a feature and $v_i^{(l)}, v_i^{(u)}$ denote lower and upper bound values in the domain of a_i extended with $\pm\infty$ [26].

An instance x satisfies rule r, or r covers x, if every Boolean condition in premise p evaluates to true for x [26]. When the instance to be explained satisfies p, the rule $p \rightarrow y$ represents a candidate explanation of the decision $c(x)$, and if the interpretable predictor accurately mimics the black-box behavior in the neighborhood of x, the rule constitutes a valid local explanation of $b(x) = c(x) = y$ [25].

Factual Rule Extraction Process

The extraction of factual rules from the merged decision tree c follows a systematic path-tracing procedure that captures the logical sequence of decisions leading to the predicted outcome. Given the decision tree c and the instance x to be explained, the factual rule $r = p \rightarrow y$ is constructed by "including in p the split conditions on the path from the root to the leaf satisfied by x, and setting $y = c(x) = b(x)$" [26].

This path-based extraction ensures that the factual rule is both consistent with the decision tree structure and satisfied by the target instance x. The set of split conditions encountered along the root-to-leaf path, also referred to as a *direct reason*, provides a complete logical characterization of why the instance received its particular classification [26]. Unlike minimal explanation approaches that seek to reduce rule complexity, LORE_{sa} prioritizes comprehensiveness, as "experiments show LORE_{sa} returns very small rules" naturally without requiring further minimization procedures [26].

An example of a factual rul is illustrated in Figure 3.2; consider an instance $x = \{\text{age} = 22, \text{sex} = \text{male}, \text{income} = 800, \text{car} = \text{no}\}$ from a credit risk assessment system where LORE_{sa} generates the factual rule: $\{\text{age} \leq 25, \text{sex} = \text{male}, \text{income} \leq 900\} \rightarrow \text{deny}$. This rule explicitly states that the loan denial is attributed to the conjunction of being 25 years old or younger, being male, and having an income of 900 or less. The factual explanation provides transparent insight into exactly

which feature combinations drove the black-box decision [26].

Counterfactual Rule Extraction Algorithm

Counterfactual rule extraction represents a more sophisticated procedure that identifies alternative decision paths within the decision tree that would result in different classification outcomes. LORE$_{sa}$ employs Algorithm 3 to systematically discover and rank counterfactual rules based on minimality criteria [26].

Algorithm 3: extractCounterfactuals(c, r, x, U)

Input: c - decision tree, r - rule, x - instance to explain, U - constraints
Output: Φ - set of counterfactual rules for p
$Q \leftarrow$ getPathsWithDifferentLabel(c, y) ; `// get paths with` $y \neq \hat{y}$
$\Phi \leftarrow \emptyset$; min $\leftarrow +\infty$; `// initialize counterfactual set`
for $q \in Q$ **do**
  if ***not*** $q \rightarrow U|q$ **then**
    continue ; `// skip rule if constraints not satisfied`
  $q_{len} \leftarrow nf(q, x) = |\{sc \in q \mid \neg sc(x)\}|$;
  if $q_{len} <$ min **then**
    $\Phi \leftarrow \{q \rightarrow y\}$; min $\leftarrow q_{len}$;
  else if $q_{len} =$ min **then**
    $\Phi \leftarrow \Phi \cup \{q \rightarrow y\}$;
return Φ;

The counterfactual extraction process operates through the following systematic approach:

Path Identification: The algorithm first identifies all paths in the decision tree c that lead to decisions $y' \neq y$, where y represents the original prediction for instance x. For each such path, the conjunction of split conditions q forms a potential counterfactual premise [26].

Minimality Ranking: Critical to the usefulness of counterfactual explanations is the principle of minimality. The algorithm ranks potential counterfactual rules $q \rightarrow y'$ by computing qlen = nf$(q, x) = |\{sc \in q \mid \neg sc(x)\}|$, which counts the number of split conditions in q that are not satisfied by the original instance x [26]. This metric directly corresponds to the number of feature changes required to achieve the alternative outcome.

Constraint Satisfaction: To ensure actionability, the algorithm filters counterfactual candidates based on user-specified constraints U on features. "Since both the premise q and the constraints U are logic formulae, the test amounts at

checking validity of the implication $q \rightarrow U|_q$" [26]. This constraint satisfaction step eliminates counterfactuals that involve changing immutable features (such as age decreasing or demographic characteristics) or violate domain-specific restrictions.

Continuing the credit risk example shown in Figure 3.2, LORE$_{sa}$ might generate counterfactual rules such as $\{\text{age} > 25, \text{income} > 1500\} \rightarrow$ grant and $\{\text{income} \leq 1500, \text{car} = \text{yes}\} \rightarrow$ grant [26]. These counterfactuals explicitly indicate that either having an income greater than 1500 and being older than 25 or having an income lower than 1500 and owning a car would result in loan approval, providing actionable insights into how the prediction could be altered.

Decision Tree Ensemble Merging for Rule Extraction

A crucial innovation in LORE$_{sa}$ involves the extraction of rules from merged decision tree ensembles rather than individual trees. The multi-neighborhood approach generates multiple decision trees $\{d^{(1)}, d^{(2)}, \ldots, d^{(N)}\}$ that are merged into a single interpretable predictor c before rule extraction can proceed [26].

LORE$_{sa}$ adopts the merging approach introduced by Fan et al.[71], which implements a two-phase procedure for combining multiple decision trees into a unified structure [26]. The first phase merges decision regions using a recursive approach based on condition trees, while the second phase performs pruning to reduce complexity by removing inner nodes with identical leaf classifications. A critical advantage of this merging approach is its lossless nature, "the merging method maintains for every instance the class label assigned by the tree ensembles" [26]. This preservation ensures that rule extraction from the merged tree precisely reflects the collective decision logic of the ensemble.

The ensemble merging process directly contributes to explanation consistency by reducing the effects of randomness in neighborhood generation. "The generalized representation of the knowledge contained in the multiple decision trees helps in reducing the probability that small changes in the data may result in very different explanations" [26]. This consistency improvement represents a significant advantage over single-tree approaches that may be sensitive to variations in neighborhood composition.

Advantages of Rule-Based Explanations

The symbolic rule extraction approach employed by LORE$_{sa}$ offers several compelling advantages over alternative explanation formats, as demonstrated by the performance metrics in Table 3.1:

Human Cognitive Alignment: "Rule-based explanations are considered closer to human reasoning w.r.t. the feature importance-based explanations" [25].

Table 3.1: Aggregated evaluation metrics over experimental datasets and black-boxes showing performance comparison between different explanation methods. LORE_{sa} achieves the best performance in most metrics including fidelity, complexity, and stability measures. Bold value indicates the best perfomance

Method	Silhouette	Fidelity	Complexity	Instability	Instability_{si}
anchor	.116 ± .51	.912 ± .21	4.950 ± 8.20	.174 ± 0.29	.651 ± .949
brl	.019 ± .30	.869 ± .09	1.998 ± 1.23	.889 ± 0.45	n.a.
lime	.444 ± .49	.904 ± .23	9.733 ± 1.47	.787 ± 1.58	.159 ± .142
lore	.408 ± .49	.996 ± .01	4.917 ± 3.69	.123 ± 0.22	.259 ± .847
maple	.127 ± .56	.949 ± .09	29.014 ± 3.25	.651 ± 1.66	n.a.
shap	.463 ± .56	n.a.	6.070 ± 3.84	.608 ± 0.58	**.017 ± .052**
LORE_{sa}	**.569 ± .46**	.992 ± .20	3.986 ± 3.93	**.073 ± 0.07**	.107 ± .081
LORE^{d}_{sa}	**.569 ± .46**	**.999 ± .01**	5.105 ± 4.29	.083 ± 0.08	.107 ± .066

Metric	anchor	LORE	brl	$\textbf{LORE}_{sa}$	$\textbf{LORE}^{d}_{sa}$
coverage	.284 ± .32	.492 ± .27	.344 ± .30	**.742 ± .27**	.485 ± .26
precision	.912 ± .21	.993 ± .07	.732 ± .22	.772 ± .26	**.998 ± .02**
h-mean	.433 ± .25	.657 ± .11	.468 ± .25	**.694 ± .25**	.615 ± .22

Logical rules align naturally with human decision-making processes, making them more intuitive for non-expert users to understand and confirm.

Logical Transparency: Unlike feature importance vectors that require domain expertise to interpret, symbolic rules provide explicit logical conditions that clearly specify decision boundaries. Users can directly verify whether rule conditions apply to their specific circumstances and understand the logical chain leading to predictions.

Actionable Insights: Counterfactual rules provide direct guidance on how to achieve different outcomes by specifying exact feature changes required. This actionability is particularly valuable in applications where users seek to understand how to modify their situation to achieve favorable predictions.

Completeness and Precision: Rule-based explanations capture complete logical conditions for decisions rather than partial information conveyed through feature importance rankings. This completeness ensures that users receive a comprehensive understanding of the decision logic rather than incomplete glimpses into the model behavior.

Superior Performance Metrics: As shown in Table 3.1, LORE_{sa} achieves the best overall performance across multiple evaluation dimensions, with particularly strong results in fidelity (0.992), low complexity (3.986), and great stability (0.073 instability score), significantly outperforming traditional methods like LIME and SHAP.

3.1.3 Current LORE Approach for Showing the Extracted Rules and Counterfactual Rules

The current implementation of LORE$_{sa}$ presents explanations through a structured data format that arranges all relevant information for understanding both factual and counterfactual reasoning in seven primary components that together provide a comprehensive view of the local decision space around the explained instance.

Factual Rule Representation

The factual rule component represents the logical path that leads to the predicted outcome for the target instance. The rule structure follows a premise-consequence format where:

- **Premises**: A list of conditions, each specifying an attribute (`attr`), a threshold value (`val`), and a comparison operator (`op`)
- **Consequence**: The predicted class outcome with its corresponding attribute and equality operator

For example, a factual rule might be represented as:

```
'rule': {
    'premises': [{'attr': 'capital-gain', 'val': 7148.0, 'op': '>'}],
    'consequence': {'attr': 'income', 'val': '>50K', 'op': '='}
}
```

This structure explicitly states that instances with capital gain greater than 7148.0 are predicted to have income greater than 50K, providing the user with direct logical understanding of the decision boundary.

Counterfactual Rule Representation

Counterfactual rules follow an identical structural format but represent alternative decision paths that would yield different classification outcomes. Each counterfactual rule specifies the minimal set of feature changes required to alter the prediction, formatted as:

```
'counterfactuals': [{
    'premises': [
        {'attr': 'capital-gain', 'val': 7148.0, 'op': '<='},
        {'attr': 'capital-loss', 'val': 2728.5, 'op': '<='},
        {'attr': 'age', 'val': 57.0, 'op': '<='},
```

```
        {'attr': 'hours-per-week', 'val': 59.0, 'op': '<='}
    ],
    'consequence': {'attr': 'income', 'val': '<=50K', 'op': '='}
}]
```

The counterfactual representation directly indicates that individuals with capital gain at or below 7148.0, capital loss at or below 2728.5, age at or below 57, and working 59 hours per week or fewer would be predicted to earn 50K or less.

Supporting Information Components

Beyond the core rule structures, LORE$_{sa}$ provides additional components to allow users to better understand the generated explanations:

Fidelity Score: A numerical measure (ranging from 0 to 1) indicating how accurately the surrogate decision tree approximates the black-box model's behavior in the local neighborhood. High fidelity scores (approaching 1.0) indicate reliable explanations.

Delta Specifications: The `deltas` component explicitly identifies the minimal changes required for counterfactual outcomes, formatted as logical conditions such as `[[{'att': 'capital-gain', 'op': '<=', 'thr': 7148.0}]]`.

Counterfactual Samples and Predictions: The `counterfactual_samples` and `counterfactual_predictions` components work in tandem to provide both the synthetic instances generated through the genetic algorithm and their corresponding black-box model predictions. The `counterfactual_samples` array contains concrete examples of alternative scenarios that maintain the same feature structure as the original dataset while representing variations that lead to different predictions. Each synthetic instance in this array has a corresponding entry in the `counterfactual_predictions` array, which lists the black-box model's actual predictions for that instance. This paired structure enables verification of counterfactual logic and assessment of neighborhood quality by allowing direct comparison between the generated instances and their predicted outcomes.

Feature Importance Ranking: A ranked list of features with their corresponding importance scores, providing complementary insight into which attributes most significantly influence local decision boundaries.

Limitations of Current Presentation Format

While the structured data format provides comprehensive information, it presents several limitations for practical explainability that deserve closer examination. Most immediately, there is the considerable **cognitive load** placed on users attempting to parse and understand the textual representation. This challenge is particularly

acute for those without technical backgrounds, who must mentally compose logical conditions to completely grasp decision scenarios, task that demands substantial mental effort and can be overwhelming. Beyond this, users might face the struggle of developing a contextual understanding of what they're seeing without the ability to easily **visualize** how the explained instance relates to the **decision boundaries**. Users are left somewhat in the dark about the density and distribution of the generated synthetic neighborhood that supports the explanation. This lack of perspective makes it difficult to assess whether the explanation represents a typical case or an outlier, and whether the supporting evidence is robust or sparse. The static nature of the current information delivery further complicates the situation. With no mechanism for interactive exploration, users cannot investigate how variations in **specific features might affect both factual and counterfactual rules**. This becomes especially problematic when we consider scalability challenges. As datasets grow more complex, involving higher dimensions or intricate decision boundaries with multiple interacting features, the textual representation becomes increasingly difficult to comprehend.

Chapter 4

Design evolution

4.1 Visualization concept

The development of an effective visualization framework for explanations, shaped in the form of a synthetic neighborhood and a decision tree surrogate model, requires a departure from traditional textual representation approaches. As established in Subsection 3.1.3, current textual formats, as the LORE$_{sa}$'s one, present critical limitations: significant cognitive load for parsing logical conditions, limited contextual understanding of decision boundaries, static information delivery, and scalability challenges for high-dimensional datasets.

Our approach recognizes that the two explanation components are distinct but complementary information artifacts requiring different visualization paradigms: a **synthetic neighborhood** in need of spatial representation, and a **surrogate decision tree** requiring hierarchical visualization of logical structures.

4.1.1 Conceptual Framework

Our framework addresses this fundamental disconnect through an **integrated dual-representation strategy** built on three key principles from visualization and human-computer interaction literature:

Information Integration Reduces Cognitive Load [53]. Coordinated views presenting spatial and symbolic information allows users to directly observe relationships between neighborhood characteristics and rule structure without mental integration across separate interfaces.

Interactive Confirmation Supports Explanation Trust [58]. Bidirectional interaction between neighborhood visualization and rule exploration enables users to confirm explanation quality by examining how decision boundaries correspond to synthetic instance distributions.

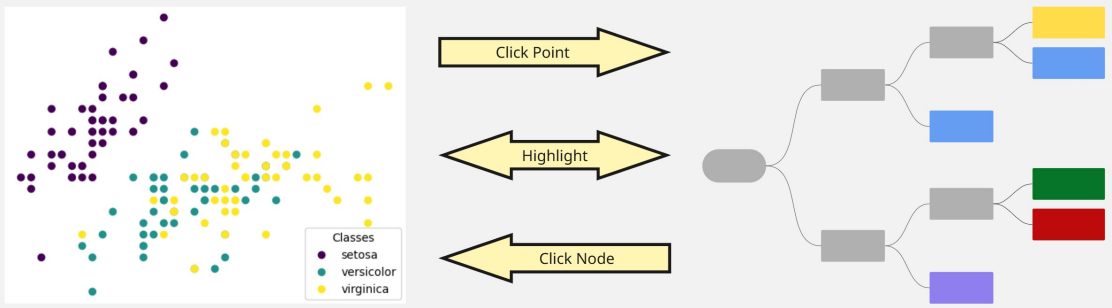


Figure 4.1: Dual representation strategy showing scatter plot visualization of synthetic neighborhood (left), decision tree visualization of rule structure (right), and bidirectional information bridging mechanisms (center arrows)

Multiple Representation Formats Support Diverse User Mental Models [60]. Different visualization formats for the same decision tree structure accommodate diverse cognitive approaches to understanding machine learning explanations.

The framework structures explanation workflow around four sequential but iterative phases: **Configuration** of datasets, models, and explanation parameters through the interface component, **Understanding** synthetic neighborhood quality through scatter plot exploration, **Exploring** the surrogate model structure through interactive decision tree navigation, and **Confirm** extracted rules through cross-referencing between spatial and symbolic representations.

Representational Alignment Philosophy

The conceptual foundation emerges from **representational alignment**, matching visualization modality to the natural structure of underlying information. Rather than forcing mental reconstruction from textual descriptions, our framework makes spatial and logical relationships directly visible and explorable.

This addresses three critical challenges in current XAI visualization approaches [58, 62]: **information fragmentation** addressed through coordinated multiple views, **cognitive overload** mitigated through visual connection mechanisms, and **confirmation difficulties** addressed through interactive exploration mechanisms.

Our implementation focuses on coordinating the visualizations of the spatial neighborhood analysis with the interactive surrogate model interface, as illustrated in Figure 4.1.

Configuration Interface Component

To support the integrated visualization framework, we developed a comprehensive configuration interface that enables users to control all aspects of the LORE$_{sa}$

explanation generation process. This interface serves as the primary entry point for setting up visualization scenarios and was designed initially for testing purposes, but was later reshaped with the ultimate scope of supporting educational applications.

The interface, when loaded in its demo version, provides structured selection mechanisms for datasets and classification algorithms, enabling users to explore explanations across different domains and model types. This supports the pedagogical goal of demonstrating how explanation quality and characteristics vary across different machine learning contexts. Comprehensive parameter configuration enables fine-tuning of both the underlying classifier and the LORE$_{sa}$ explanation generation process. Users can adjust genetic algorithm parameters, and neighborhood generation settings to observe their impact on explanation quality and visualization effectiveness.

Dynamic feature input controls adapt to the selected dataset's characteristics, providing appropriate input mechanisms for numeric, categorical, and ordinal features. This enables users to specify instances for explanation interactively, supporting exploratory analysis workflows where users can investigate how different feature combinations affect both predictions and explanations. Additionally, integrated controls for dimensionality reduction technique selection and visualization parameters enable users to compare how different projection methods affect neighborhood visualization quality.

The interface follows established human-computer interaction principles for complex configuration tasks, employing progressive disclosure to manage complexity and providing immediate visual feedback through the coordinated visualization components below.

Neighborhood 2D Projection Component: Spatial Neighborhood Analysis

The neighborhood 2D projection serves as the primary interface for understanding the spatial characteristics of the synthetic data. High-dimensional synthetic instances are projected into two-dimensional coordinate systems to enable visual exploration.

Our implementation supports multiple dimensionality reduction techniques to accommodate different analytical requirements, with selection controlled through the configuration interface. **Uniform Manifold Approximation and Projection (UMAP)** offers nonlinear manifold learning with superior computational efficiency compared to t-SNE [37, 28], providing intuitive hyperparameter control over local structure preservation and cluster tightness. **Principal Component Analysis (PCA)** provides a deterministic linear transformation preserving linearity of decision boundaries [31], enabling direct projection of surrogate model boundaries into the visualization space.

Color-coded class labeling distinguishes prediction outcomes within synthetic neighborhoods, enabling immediate identification of decision boundary regions and assessment of genetic algorithm instance generation quality.

Surrogate Model Component: Hierarchical Rule Structure

The surrogate model component provides hierarchical visualization of logic extracted from the surrogate model, employing **node-link diagrams** following established best practices [47].

Node Visual Encoding: Split nodes (logical conditions and feature thresholds) employ neutral color schemes emphasizing decision point roles, while leaf nodes receive class-specific coloring corresponding directly to spatial neighborhood analysis plot color mapping. **Edge Visualization** uses variable width encoding to indicate synthetic instance flow through decision branches, with text labels clearly indicating logical conditions and color coding enabling decision path following.

Bidirectional Information Bridging

The main interaction mechanisms lie in the **bidirectional information bridging** that creates explicit connections between spatial and symbolic representations.

User interactions with spatial neighborhood analysis plot points highlight corresponding decision paths in the tree visualization. Clicking synthetic instances traces feature values through the decision tree structure, highlighting complete root-to-leaf classification paths. This enables understanding of how spatial proximity corresponds to logical similarity in decision rules.

On top of that, decision tree node interactions highlight corresponding synthetic instances in spatial neighborhood analysis plots. Leaf node clicks highlight all instances satisfying complete logical paths, while split node clicks highlight instances passing through specific decision points, visualizing the spatial distribution of instances satisfying particular logical conditions.

4.1.2 Interactive Confirmation Model

Our interaction model transforms explanation consumption from passive acceptance into active analytical investigation through **exploratory confirmation** principles. The model begins with comprehensive configuration controls enabling users to establish explanation scenarios, then implements **details-on-demand** through hover interactions, **coordinated highlighting** maintaining visual connections, and **progressive disclosure** enabling incremental exploration.

Confirmation Systems

Sophisticated details-on-demand for both spatial and symbolic elements are provided through the contextual tooltip systems. Spatial neighborhood analysis plot's points tooltips reveal decoded feature values and class prediction information. Decision tree tooltips provide node-specific contextual information, including logical conditions in natural language, sample statistics, and feature importance metrics.

On top of that, **Voronoi diagram overlays** provide explicit decision boundary visualization within spatial neighborhood analysis plot space for linear transformations (PCA). Semi-transparent color encoding matching class-based schemes creates colored polygons representing consistent classification areas, enabling assessment of explanation quality through boundary geometric properties and spatial coherence evaluation.

4.2 Project evolutions

Following the initially proposed visualization, consisiting of just a scatter plot and an interconnected surrogate model visualization, the development helped in refining the design of the interface. The following section discusses the development process and its different stages to provide readers with comprehensive documentation of the project and its evolution. To facilitate better exploration of the project at its various stages, each subsection references the git commit related to the described implementation.

4.2.1 Neighborhood 2D projection

The first version of the spatial neighborhood analysis plot [72] was developed, and, for testing purposes, the iris dataset was used and PCA employed as the dimensionality reduction technique. As previously specified, PCA is a linear transformation method, which enabled us to project the decision boundaries extracted from a mock decision tree classifier trained on the test dataset, as shown in Figure 4.2.

We then developed a subsequent version of the spatial neighborhood analysis plot [73] that used a grid split for representing decision boundaries. As observed in Figure 4.3, this approach reveals subareas of the projected space that were not previously displayed. Additionally, we implemented hover interaction on spatial neighborhood analysis plot points, showing the point's class and additional information.

Regarding the decision boundaries representation implementation, we tested various approaches. The aforementioned grid was our first attempt, followed by an implementation [74] that used Ramer Douglas Peucker path approximation [75, 76]

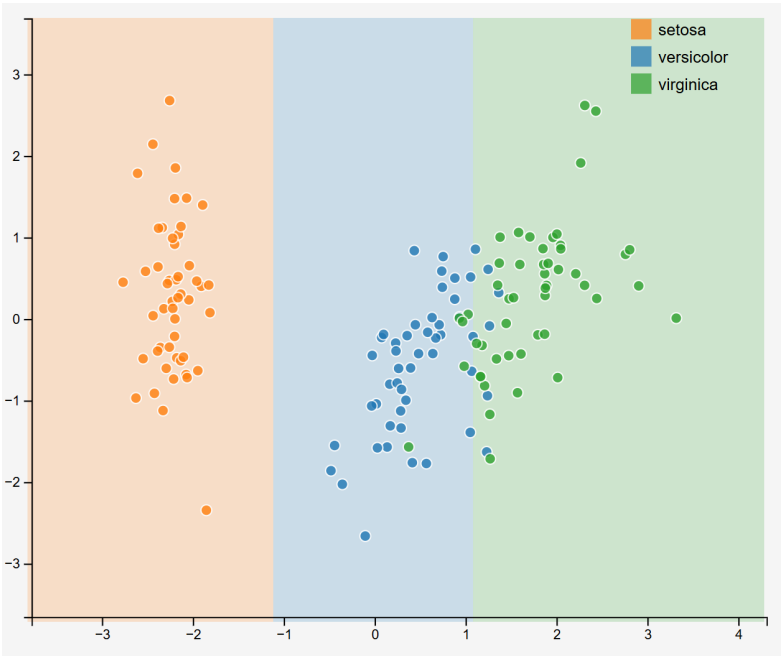


Figure 4.2: Initial spatial neighborhood analysis plot implementation using PCA projection with linear decision boundaries overlay for the iris dataset.

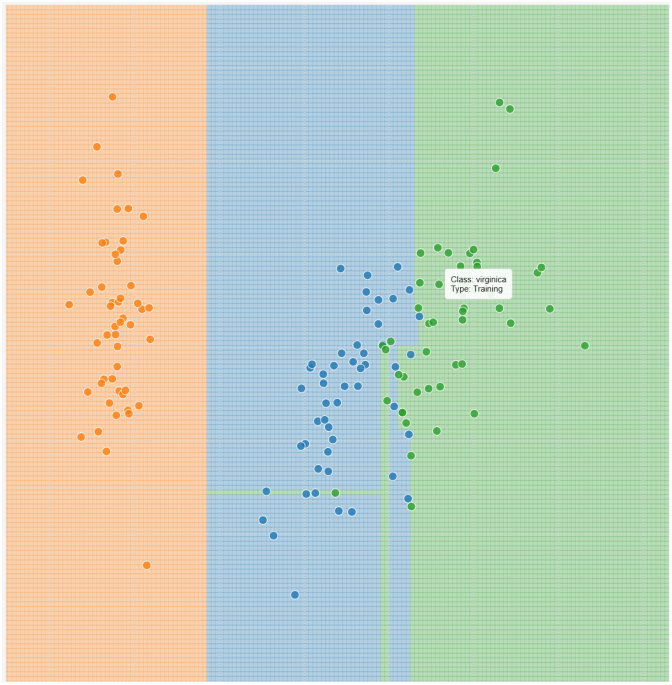


Figure 4.3: Enhanced spatial neighborhood analysis plot with grid-based decision boundary representation and interactive point tooltips.

for simplifying decision paths in the projected space. Our final implementation [77] employs Voronoi tessellation [78] with user settable granularity for space division.

This approach naturally accommodates the irregular decision boundaries produced by decision trees, unlike grid-based heatmaps that may introduce artificial discretization artifacts.

The algorithm begins by constructing a regular mesh grid that densely samples the 2D visualization space. The grid extends slightly beyond the actual data distribution, with boundaries defined as shown in Equation 4.1.

$$\begin{aligned} x_{\min} &= \min(\mathbf{X}_{\text{2D}}[:, 0]) - 1, \quad x_{\max} = \max(\mathbf{X}_{\text{2D}}[:, 0]) + 1 \\ y_{\min} &= \min(\mathbf{X}_{\text{2D}}[:, 1]) - 1, \quad y_{\max} = \max(\mathbf{X}_{\text{2D}}[:, 1]) + 1 \end{aligned} \tag{4.1}$$

where $\mathbf{X}_{\text{2D}}$ represents the PCA-transformed coordinates of all points in the synthetic neighborhood. This margin ensures that decision boundaries extending to the visualization edges are properly captured.

The mesh resolution is controlled by the `step` parameter, which defaults to 0.1 units. This creates a dense sampling with thousands of grid points as illustrated in Equation 4.2.

$$\mathbf{G} = \{(x_i, y_j) \mid x_i \in [x_{\min}, x_{\max}],\ y_j \in [y_{\min}, y_{\max}],\ \text{step} = 0.1\} \tag{4.2}$$

The choice of step size represents a trade-off between boundary smoothness and computational efficiency. Smaller values produce smoother, more detailed boundaries but increase both computation time and the number of Voronoi cells requiring subsequent processing.

For each grid point $\mathbf{g} = (x, y)$ in the 2D visualization space, the surrogate model's prediction is determined. However, the decision tree operates in the original high-dimensional feature space, not in the reduced 2D space. This necessitates an inverse transformation shown in Equation 4.3.

$$\mathbf{g}_{\text{original}} = S^{-1}(P^{-1}(\mathbf{g})) \tag{4.3}$$

where P^{-1} represents the PCA inverse transformation and S^{-1} represents the inverse standardization. PCA's linear nature makes this inverse transformation mathematically well-defined. The inverse PCA transformation reconstructs the original feature representation by multiplying the 2D coordinates by the transpose of the component matrix as illustrated in Equation 4.4.

$$\mathbf{x}_{\text{reconstructed}} = \mathbf{g} \cdot \mathbf{W}^T + \boldsymbol{\mu} \tag{4.4}$$

where $\mathbf{W}$ contains the principal component vectors (eigenvectors of the covariance matrix) and $\boldsymbol{\mu}$ is the mean vector of the original data. The standardization is then inverted by rescaling using the stored standard deviations.

It is important to note that when PCA reduces dimensionality from d dimensions to 2 dimensions, information in the $(d-2)$ discarded components is lost. The inverse transformation therefore reconstructs an approximation of the original point, with the reconstruction lying in the 2-dimensional subspace spanned by the first two principal components.

Once grid points are transformed back to the original feature space, the surrogate model is used to generate the predictions.

This produces a class label for each grid point, effectively creating a discrete classification of the entire 2D visualization space according to how the decision tree would classify points in those regions. The resulting prediction array $\mathbf{Z}$ has the same shape as the original mesh grid, with each element containing the predicted class label for the corresponding spatial location.

The `scipy.spatial.Voronoi` class constructs the initial tessellation by computing the Voronoi diagram for all grid points. Each grid point becomes a Voronoi site, and the diagram partitions the plane into cells such that every location within a cell is closer to that cell's site than to any other site. The Voronoi construction returns:

- **Vertices**: The corner points where Voronoi cell boundaries meet
- **Regions**: Lists of vertex indices defining each Voronoi cell polygon
- **Ridge points**: Pairs of sites that share a boundary

Each Voronoi region is converted to a Shapely Polygon object for geometric manipulation. Infinite regions (those extending to infinity, indicated by vertex index -1) are filtered out since these lie outside the bounded visualization area and cannot be meaningfully rendered.

The raw Voronoi tessellation produces one cell per grid point, resulting in thousands of tiny polygons. Since adjacent grid points often receive the same class prediction from the decision tree, these cells are merged into larger, unified regions using graph-based connectivity analysis.

The undirected graph $G = (V, E)$ is constructed, where: **Vertices** V represent Voronoi regions and **Edges** E connect adjacent regions that share the same predicted class.

Two regions are connected if they share a ridge (common boundary) in the Voronoi diagram and their corresponding grid points received identical class predictions from the decision tree. Using NetworkX, all connected components in this graph are identified. Each connected component represents a maximal set of Voronoi cells that are both spatially adjacent and classified identically. For each component, all constituent polygons are merged into a single unified region using Shapely's `unary_union` operation.

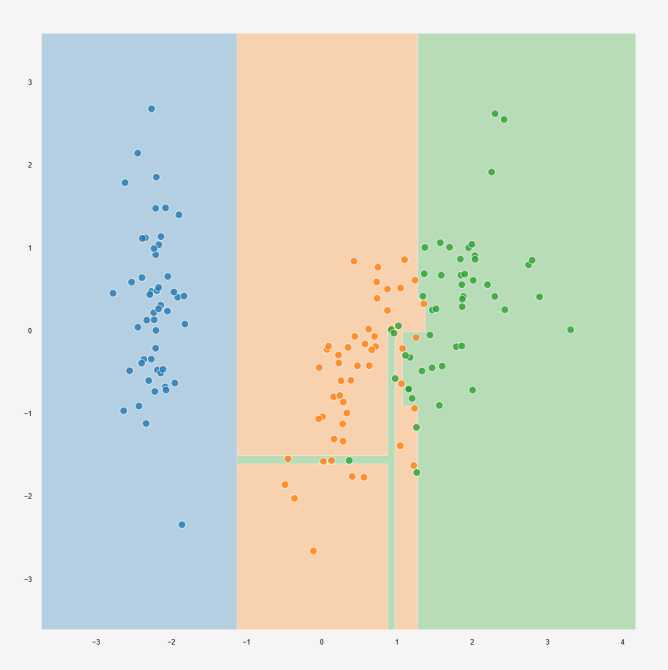


Figure 4.4: Final spatial neighborhood analysis plot implementation featuring Voronoi tessellation-based decision boundaries with user-configurable granularity.

The backend transmits the merged polygon boundaries as arrays of vertex coordinates to the frontend. These polygons are then rendered on the webpage. Each polygon is filled with a color corresponding to its predicted class, maintaining visual consistency with the color encoding used for scatter plot points.

The Voronoi approach provides several advantages for our visualization requirements. It naturally handles irregular decision boundaries without imposing artificial geometric constraints, unlike approaches that force rectangular partitions. The graph-based merging also produces clean, consolidated regions that match the decision tree's actual partitioning structure, avoiding visual clutter from excessive fragmentation. On top of that the method scales efficiently: while grid generation is $O(n^2)$ in the number of grid cells, the Voronoi construction and merging operations remain computationally tractable for interactive use. One can observe an example of the final result in Figure 4.4

Later in the project development, we added the option to switch between the initially default PCA and t-SNE [79], along with UMAP and MDS [80]. To enable users to switch between the four dimensionality reduction techniques, we implemented buttons with related names [81] and added highlighting of split node descendants [82].

Considering the possibility of datasets with more than ten classes, we implemented RGB space projection for color classes [83]. We later replaced this with a

projection in the CIELAB a*-b* (L*=70) color space [84]. The issue with RGB space is that it is three-dimensional, while dimensionality reduction projections are two-dimensional. The relevant implementation does not differ significantly. Additionally, we added the option for showing the original dataset in the projected space. Points from the original dataset are represented with lower opacity to distinguish them from the generated neighborhood ones [85].

All the features just discussed are shown in Figure 4.5. As observed, decision boundaries are not present in the UMAP projection since UMAP, t-SNE, and the used MDS are non-linear projections, and the same results as for PCA cannot be directly achieved. The colors are also the RGB-projected ones, even though the dataset used for showcasing functionality (iris) contains only three classes.

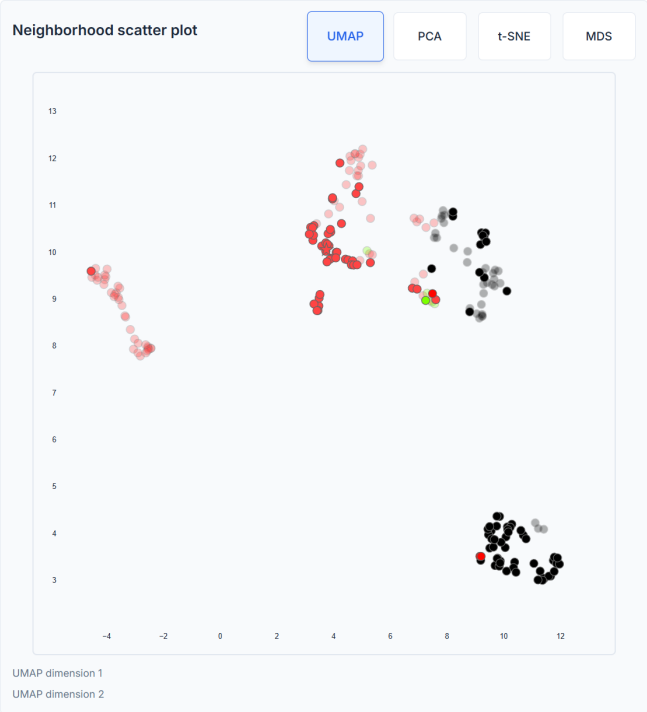


Figure 4.5: Advanced webapp features including dimensionality reduction technique selection (UMAP shown), RGB color space projections and original dataset overlay with opacity differentiation.

Regarding the projection in the CIELAB space, this approach addresses the challenge of generating distinct colors for datasets with numerous classes (more than ten). Unlike predefined color palettes that become inadequate for high-cardinality classification problems, this method dynamically generates colors based on the data distribution.

The algorithm operates on class centroids in the original feature space. For

each class c, the centroid $\boldsymbol{\mu}_c$ is computed as the mean of all samples belonging to that class:

$$\boldsymbol{\mu}_c = \frac{1}{|\mathcal{X}_c|} \sum_{\mathbf{x}_i \in \mathcal{X}_c} \mathbf{x}_i \tag{4.5}$$

where $\mathcal{X}_c$ denotes the set of all samples with class label c.

These high-dimensional centroids are then projected into two-dimensional space using the same dimensionality reduction method and parameters employed for the scatter plot visualization (PCA, t-SNE, UMAP, or MDS). This ensures consistency between the spatial layout of the scatter plot and the color assignments. The projected coordinates are normalized to the $[0, 1]$ range using min-max scaling.

The normalized 2D coordinates $(x, y) \in [0, 1]^2$ are mapped to the chromatic dimensions of the CIELAB color space. CIELAB is a perceptually uniform color space designed to approximate human vision, where the L^* dimension represents lightness, and the a^* and b^* dimensions represent the green-red and blue-yellow color opponents, respectively. The mapping is defined as:

$$\begin{aligned} L^* &= 70 \\ a^* &= ((x - 0.5) \times 2) \times 128 \\ b^* &= ((y - 0.5) \times 2) \times 128 \end{aligned} \tag{4.6}$$

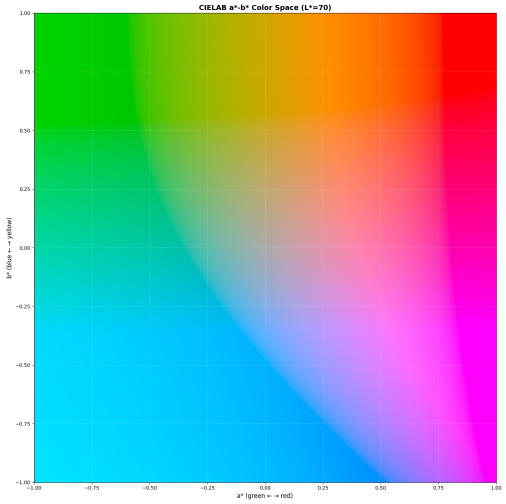
The lightness value is fixed at $L^* = 70$ to ensure consistent brightness across all generated colors, avoiding very dark or very light hues. The (x, y) coordinates are centered at $(0.5, 0.5)$, scaled to $[-1, 1]$, and then multiplied by 128 to span the typical range of CIELAB's chromatic dimensions. The CIELAB coordinates are subsequently converted to hexadecimal color strings for web rendering.

The key advantage of CIELAB over direct RGB projection lies in its perceptual uniformity: equal distances in CIELAB space correspond to approximately equal perceived color differences. This property ensures that classes with similar feature distributions receive visually similar colors, while dissimilar classes are assigned perceptually distinct colors. Additionally, CIELAB's chromatic plane (a^*, b^*) is naturally two-dimensional, making it ideally suited for mapping 2D projections without the dimensional mismatch that occurs with three-dimensional RGB space.

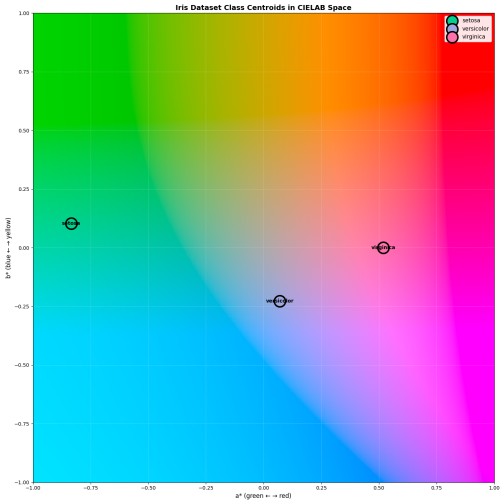
In figure 4.6, one can observe both the CIELAB a*-b* (L*=70) and RGB color spaces, and, for each, an the example projection of the mean instances of each class present in the iris dataset.

4.2.2 Tree layout

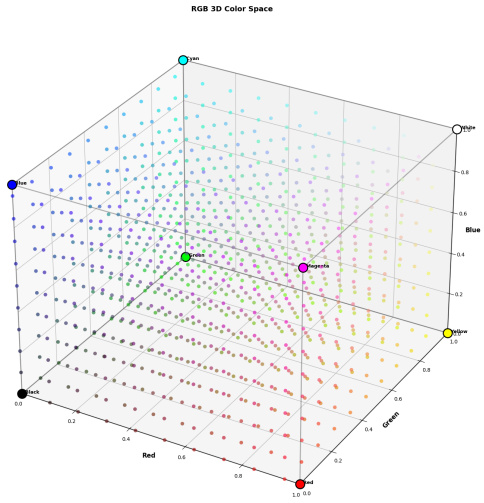
We began the project development with the first implementation of the surrogate model plot [86]. Initially, our main focus centered on creating a visualization that



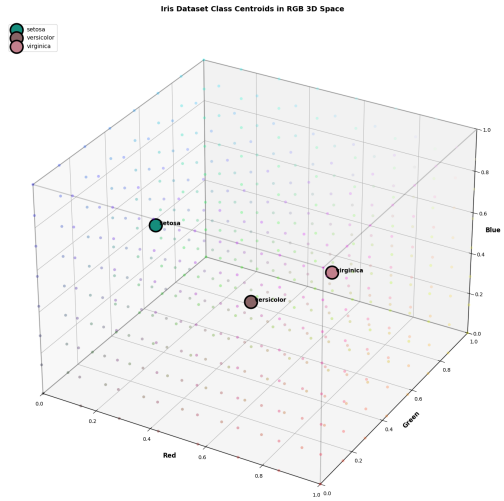
(a) CIELAB a-b Color Space with Luminosity fixed at 70



(b) Iris Dataset Class Centroids projected in the CIELAB a*-b* L=70 Color Space.



(c) RGB 3D Color Space.



(d) Iris Dataset Class Centroids projected in the RGB 3D Color Space

Figure 4.6: Colors projection that were considered during development.

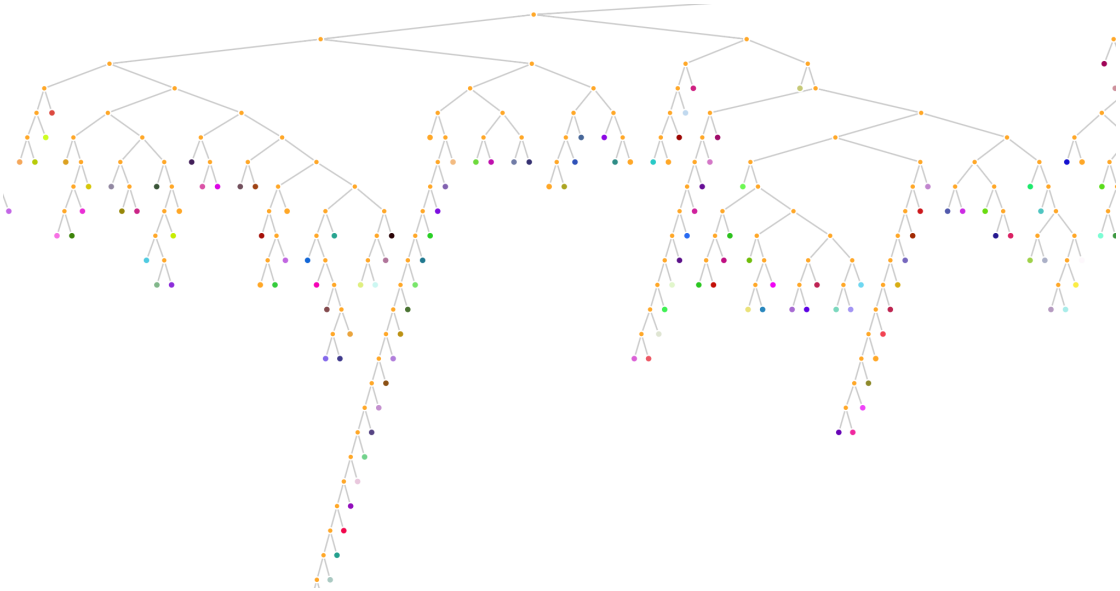


Figure 4.7: Initial surrogate model visualization prototype demonstrating node positioning algorithm for complex tree structures without overlap.

could represent all possible trees in a polished manner.

The primary challenge involved node positioning; using a fixed angle for all splits at different tree heights and with varying numbers of subtrees would have resulted in overlapping nodes.

As observed in Figure 4.7, which represents a section of the complex tree structure used for testing at this development stage, we successfully tested and accomplished node positioning at both labyrinthine and elementary levels of tree complexity.

The node positioning algorithm employs D3's hierarchical tree layout, which implements the Reingold-Tilford algorithm [87] for creating tidy tree drawings. This algorithm ensures optimal node placement without overlaps through three key mechanisms.

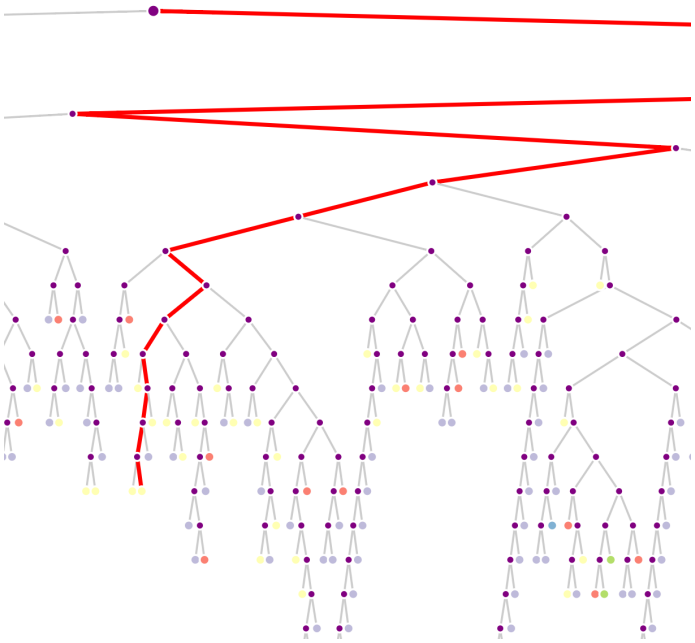
The layout dynamically scales the available space based on tree complexity. Both horizontal and vertical dimensions are calculated proportionally to the total number of nodes: $horizontalSpacing = n_{nodes} \times minSplitWidth$ and $verticalSpacing = n_{nodes} \times minSplitHeight$, where $minSplitWidth = 30$ pixels and $minSplitHeight = 25$ pixels.

This adaptive scaling ensures that larger trees receive proportionally more space, preventing visual congestion. On top of that a separation function enforces consistent inter-node spacing throughout the tree structure. Set to $2 \times minSplitWidth = 60$ pixels, this separation value creates uniform gaps between adjacent nodes at the same depth level, maintaining visual clarity regardless of the

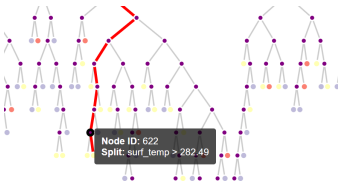
local branching structure.

The layout coordinates also undergo an initial transform calculation that fits the entire tree within the viewport bounds while preserving aspect ratio. This transform computes optimal scale factors for both dimensions and applies appropriate translations to center the tree, enabling users to immediately view the complete structure. The system then applies zoom and pan capabilities, allowing detailed exploration of specific regions while maintaining the overall spatial relationships established by the Reingold-Tilford algorithm.

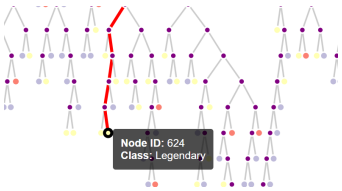
The next implementation step [88] involved implementing click-on-leaf-node interaction for path tracing from the root node to the clicked leaf, as demonstrated in Figure 4.8a. Additionally, we built hover interactions for split and leaf nodes to display either the leaf node class or the split condition based on the node type being hovered, as shown in Figures 4.8b and 4.8c.



(a) Root to leaf node path tracing example.



(b) Tooltip on split node showing the split condition.



(c) Tooltip on leaf node showing its class.

Figure 4.8: Enhanced surrogate model prototype with interactive features: path highlighting and contextual tooltips for different node types.

Subsequently, as shown in Figure 4.9, we added a first version of constant highlighting of the explained instance path in the surrogate model plot [89] and embedded sklearn information regarding split nodes [90] (namely: feature index, impurity, samples, and class distribution).

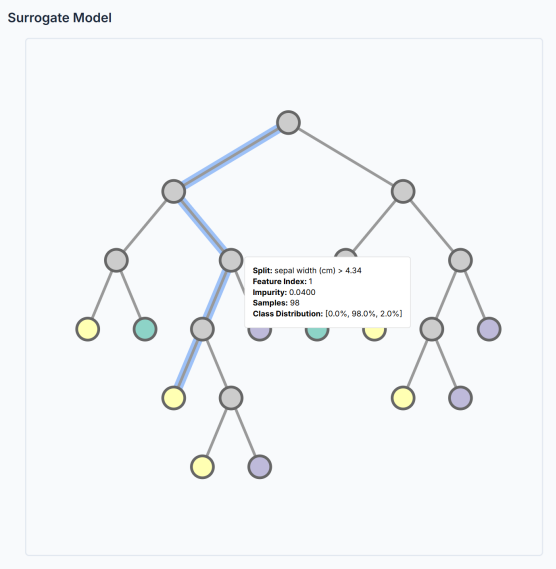


Figure 4.9: Constant highlighting of the explained instance path in the surrogate model and new tooltip

4.2.3 Rule centered approaches

While the classical tree layout successfully visualizes the complete structure of the surrogate decision tree and provides interactive features for path exploration, we recognized that it presents inherent limitations for rule-focused analysis. Although users can trace decision paths by clicking on leaf nodes and examining individual split conditions through tooltips, the hierarchical layout, especially in complex surrogate models, requires significant cognitive effort to identify the generated rule and counterrules. The traditional top-down structure, while effective for understanding the overall tree topology and decision flow, necessitates visual scanning across multiple tree levels and mental aggregation of conditions along a path, a process that becomes increasingly challenging as tree depth and complexity grow.

Moreover, the classical layout allocates equal visual prominence to all branches of the decision tree, regardless of their relevance to the explained instance. This representation, while democratic, does not prioritize the specific rule that explains the instance of interest, nor does it facilitate direct comparison between factual rules and their counterfactual counterparts.

In response to these limitations, we developed two complementary rule-centered visualizations, temporarily named `Tree Spawn` and `Blocks` layouts, and later renamed `Rules Centered` and `Rule Centered and Rule and Counterfactual Centered` respectively. These approaches maintain the interpretability benefits of tree-based hierarchical representations while reorganizing the visual presentation

to prioritize rule comprehension, minimize cognitive load for path interpretation, and facilitate direct exploration of the relationships between factual explanations and counterfactual alternatives. The following subsub-sections detail the design and briefly discuss the implementation of each approach.

Rules Centered layout

The first implementation of the Rules Centered layout [91] can be seen in Figure 4.10, where we represent the explained instance path as rectangles with tooltip information embedded within them instead of using circles, which are used for other branches of the surrogate model tree.

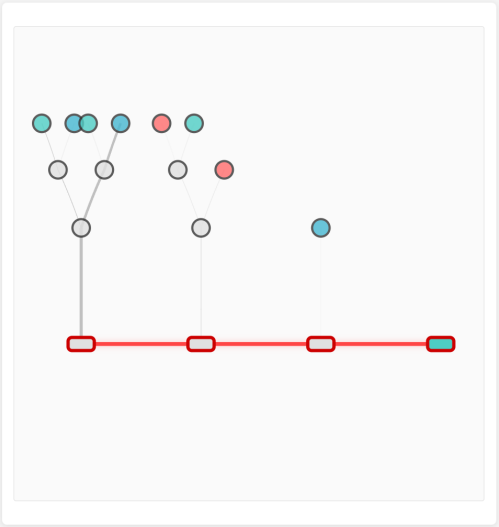
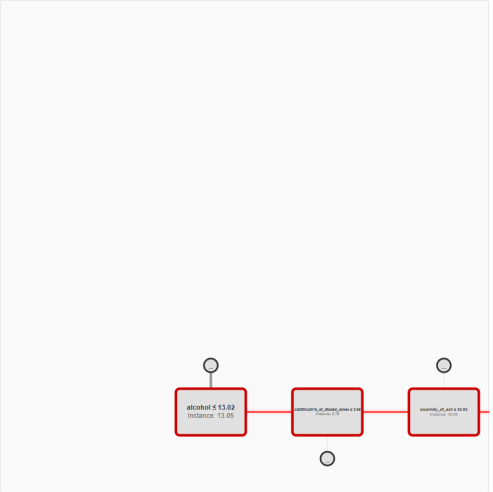


Figure 4.10: Rules Centered layout first implementation with rectangular path representation for explained instance path in surrogate model.

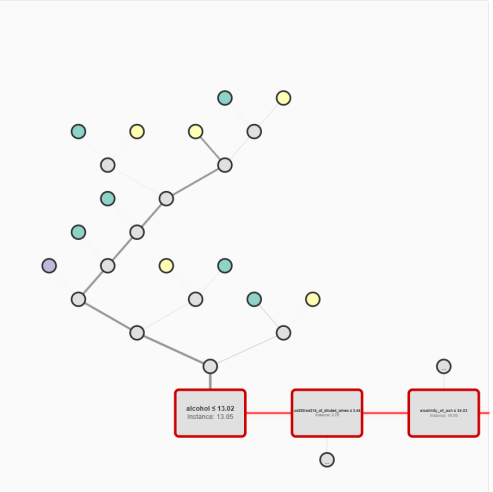
Subsequently, text containing the split or label information was added to the rectangular nodes and set to dynamically change according to the size of the rectangle [84]. We implemented functionality for expanding trees not involved in the explained instance path [92], as one can observe in Figures 4.11a and 4.11b.

Following this, we determined spacing between nodes of the explained instance path based on subtree size [93] and integrated the Rules Centered layout implementation into the main interface [94].

The Rules Centered layout visualization employs a custom linear path layout that fundamentally differs from the hierarchical positioning used in the classic tree visualization. Rather than applying the Reingold-Tilford algorithm to the entire tree structure, the Rules Centered layout visualization uses a hybrid approach that combines linear positioning for the explained instance path with hierarchical layouts for individual subtrees.



(a) Rules Centered visualization mock up with all collapsed subtrees.



(b) Rules Centered visualization mock up with a subtree expanded.

Figure 4.11: Rules Centered layout visualization with collapsible subtree functionality, demonstrating space-efficient exploration of non-explanation paths through right-click expansion.

The layout algorithm operates in three distinct phases: instance path positioning, subtree size analysis, and off-path subtree placement. Initially, the nodes belonging to the instance path are positioned horizontally along a centerline at $y = h_{inner}/2$, where h_{inner} represents the inner height of the visualization canvas. This horizontal arrangement emphasizes the sequential nature of the decision path and creates a clear visual narrative of how the explained instance traverses the surrogate model.

The horizontal spacing between path nodes is calculated adaptively based on the complexity of off-path subtrees. For each node n_i in the instance path, we compute the total size s_i of all subtrees not on the instance path by recursively counting their descendant nodes. The spacing margin m_i between consecutive path nodes is then determined as shown in Equation 4.7.

$$m_i = m_{base} + s_i \times k_{spacing} \tag{4.7}$$

where $m_{base} = 100$ pixels represents the minimum separation and $k_{spacing} = 20$ pixels per node is the spacing multiplier. This adaptive approach ensures that larger, more complex subtrees receive proportionally more horizontal space, preventing visual overlap while maintaining efficient space utilization for simpler branches.

To accommodate viewport constraints, we apply a scaling factor when the required width exceeds available space. Given path nodes $\{n_1, n_2, \ldots, n_k\}$ with calculated margins $\{m_1, m_2, \ldots, m_{k-1}\}$, the total required width is $w_{req} = k \times w_{rect} + \sum_{i=1}^{k-1} m_i$, where $w_{rect} = 150$ pixels is the rectangle width. If $w_{req} > w_{available}$,

we apply the scaling factor shown in Equation 4.8.

$$\alpha = \max\left(0.5, \frac{w_{available} - k \times w_{rect}}{\sum_{i=1}^{k-1} m_i}\right) \tag{4.8}$$

The scaled margins $m'_i = \alpha \times m_i$ are then used to compute final X positions, starting from a centered initial position $x_0 = (w_{available} - w_{final})/2 + w_{rect}/2$, where $w_{final} = k \times w_{rect} + \sum_{i=1}^{k-1} m'_i$.

Off-path subtrees are positioned using the Reingold-Tilford algorithm [87], as previously discussed for the tree layout implementation, independently for each subtree, with their root nodes anchored to their corresponding instance path nodes. The algorithm alternates subtree placement above and below the horizontal path using a global counter, creating a balanced visual distribution. For a subtree rooted at node t_j anchoring to path node n_i at position (x_i, y_c), we first apply D3's tree layout to obtain relative positions. The subtree root is then positioned at (x_i, y_{anchor}), where y_{anchor} is determined as shown in Equation 4.9.

$$y_{anchor} = \begin{cases} y_c - g_{vertical} - d_j & \text{if } j \bmod 2 = 0 \\ y_c + g_{vertical} + d_j & \text{if } j \bmod 2 = 1 \end{cases} \tag{4.9}$$

Here, $g_{vertical} = 100$ pixels represents the vertical gap between the path and subtrees, and d_j is the vertical extent of subtree t_j from its root. All descendant nodes maintain their relative positions from the D3 layout, preserving the hierarchical structure while integrating with the linear path.

The expand/collapse functionality leverages hierarchical visibility management. Each node maintains state flags: `isHidden`, `hasHiddenChildren`, and `isExpanded`. Initially, subtrees are collapsed beyond a code configurable depth ($d_{initial} = 2$ levels), with nodes deeper than this threshold marked as hidden. Right-clicking a collapsed node triggers the `expandSubtree` function, which recursively sets `isHidden = false` for all descendants and updates expansion flags. Conversely, `collapseSubtree` hides all children and resets their expansion states. The visualization filters links and nodes based on these flags, efficiently managing complex trees by displaying only relevant portions while preserving the complete underlying data structure.

This hybrid approach combines the narrative clarity of linear layouts with the structural fidelity of hierarchical algorithms, creating a visualization that effectively balances explanation-centric design with comprehensive tree exploration capabilities.

Rule and Counterfactual Rules Centered layout

Following the Rules Centered layout visualization development, we realized a first implementation of the Rule and Counterfactual Rules Centred layout [95], observable in Figure 4.12a, still using circles for representing the nodes. In this early version, we directed focus toward arranging nodes in the visualization space in the intended way. Subsequently, we developed the first implementation of the Rule and Counterfactual Rules Centered layout using rectangles [96], which also employed different link stroke widths based on the number of samples passing from one node to the next, as observed in Figure 4.12b.

Subsequently, the font size of rectangular nodes containing text was fixed to dynamically change based on rectangle size [84] for both new visualizations.

The blocks tree visualization employs a depth-aligned layout algorithm that fundamentally departs from hierarchical tree layouts by organizing nodes into vertical columns corresponding to their depth level in the decision tree. This column-based approach prioritizes the sequential nature of decision paths while maintaining clear visual alignment across all branches, facilitating direct comparison of split conditions at equivalent depths.

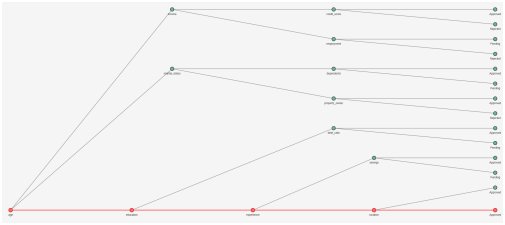
The layout algorithm is composed of a three-phase positioning strategy: depth column establishment, instance path anchoring, and sorted path distribution. First, the algorithm establishes a uniform horizontal spacing for depth levels by partitioning the available width equally among all depths. For a tree with maximum depth d_{max} and available width $w_{available}$, each depth level $d \in \{0, 1, \ldots, d_{max}\}$ receives a fixed X coordinate as shown in Equation 4.10.

$$x_d = m_{left} + d \times \frac{w_{available}}{d_{max}} \tag{4.10}$$

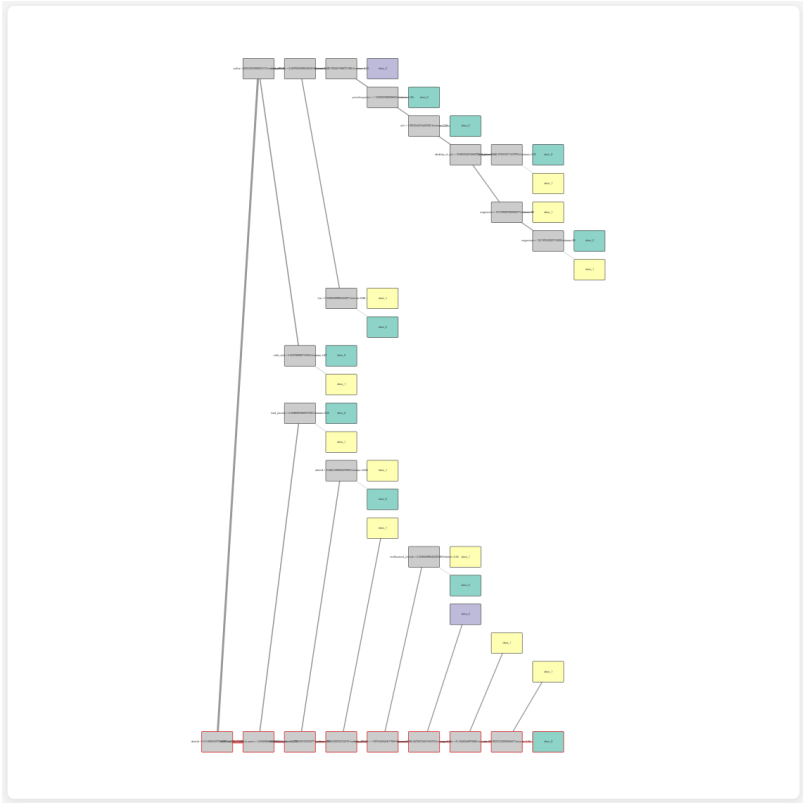
where m_{left} represents the left margin. This uniform spacing ensures that nodes at the same depth align vertically regardless of their specific path, creating a grid-like structure that emphasizes the decision sequence.

The instance path receives privileged positioning at the bottom of the visualization canvas, anchored at a constant Y coordinate $y_{bottom} = h_{effective} - m_{bottom}$, where $h_{effective}$ is the effective canvas height and m_{bottom} is the bottom margin. This bottom-alignment serves dual purposes: it provides immediate visual identification of the explanation path and creates a stable reference baseline for spatial reasoning about alternative decision paths. Each node n_i in the instance path $\{n_0, n_1, \ldots, n_k\}$ is positioned at coordinates (x_i, y_{bottom}), where x_i corresponds to its depth level.

Alternative decision paths are placed above the instance path using a sorting and spacing algorithm that emphasizes their structural relationship to the explanation. The algorithm first identifies all paths not matching the instance path, then sorts these paths by their divergence point from the explained instance. For each



(a) Blocks tree first implementation with focus on nodes arrangement.



(b) Blocks tree first implementation using rectangles.

Figure 4.12: Blocks tree evolution from circular to rectangular nodes with sample-proportional link widths.

alternative path p, its branch point $b(p)$ is computed, defined as the index of the last node shared with the instance path as shown in Equation 4.11.

$$b(p) = \max\{i \mid p[i] = p_{instance}[i],\ 0 \leq i < \min(|p|, |p_{instance}|)\} \tag{4.11}$$

Paths are then sorted in ascending order of branch points, ensuring that paths diverging earlier from the instance path appear higher in the visualization, creating a natural visual narrative of progressive specialization toward the explained decision.

The vertical spacing between alternative paths adapts to both the number of paths and the configured node spacing requirements. Given n_{paths} alternative paths and available vertical space $h_{available} = h_{inner} - 2m_{bottom}$, the spacing between consecutive paths is calculated as in Equation 4.12.

$$s_{path} = \max\left(s_{node}, \frac{h_{available}}{n_{paths}}\right) \tag{4.12}$$

where s_{node} represents the minimum acceptable node spacing. This adaptive spacing ensures visual clarity even for complex trees with numerous branches while maintaining sufficient separation for rectangular node dimensions. Each alternative path p_j at sorted index j receives Y coordinate $y_j = m_{top} + j \times s_{path}$, where m_{top} is the top margin.

The blocks layout scales margins dynamically based on tree complexity through a node scale factor $k_{scale} = \max(1, \sqrt{n_{nodes}/100})$, where n_{nodes} represents the total number of unique nodes. All margin values are multiplied by this factor, ensuring that larger trees receive proportionally more spacing while maintaining visual coherence. The effective canvas dimensions are calculated as $w_{effective} = \max(w_{base}, w_{required})$ and $h_{effective} = \max(h_{base}, h_{required})$, where base dimensions represent minimum canvas size and required dimensions account for scaled spacing requirements.

Links between nodes are rendered as straight line segments connecting rectangular node centers, with stroke width proportional to the sample flow through each connection.

The depth-aligned layout offers several analytical advantages compared to traditional hierarchical layouts. The vertical column structure enables direct comparison of decision criteria across branches at equivalent depths, facilitating identification of feature importance patterns and split threshold variations. The bottom-anchored instance path creates a stable visual reference for understanding how the explained decision relates to alternative outcomes, while the sorted arrangement of alternative paths reveals the decision tree's branching structure in relation to the specific instance being explained.

Chapter 5

Final design

Following the iterative development process outlined in the previous section, the final design of the visualization framework presents an interface that integrates spatial and symbolic representations of LORE$_{sa}$ explanations and that could easily be repurposed to accommodate other explainability methods. The system workflow is separated into two primary components: an **interactive configuration interface** that enables comprehensive control over the explanation generation process, and a **coordinated visualization system** that presents the resulting explanations through two interconnected views.

The final implementation represents the end of our design evolution, incorporating lessons learned from each development phase to create a comprehensive interactive explanation system. The interface employs a **progressive disclosure workflow** [53] that guides users through sequential configuration steps while maintaining access to previously configured options.

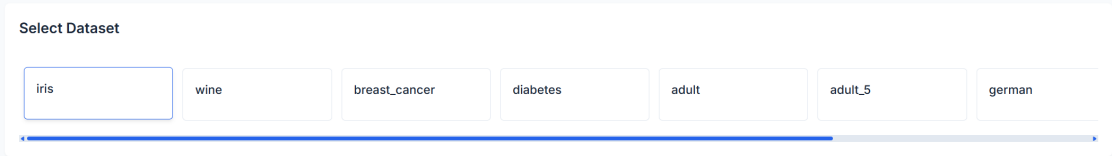
5.1 Configuration Interface Architecture

The configuration interface implements, a **vertically-stacked progressive workflow** that structures the explanation generation process into, in its demo form, seven sequential but revisable stages: dataset selection, classifier selection, classifier parameter configuration, instance specification, explanation parameter settings, dimensionality reduction techniques parameter settings, and visualization selection. This sequential presentation reduces cognitive load by presenting configuration options in logical order while allowing users to revisit previous steps.

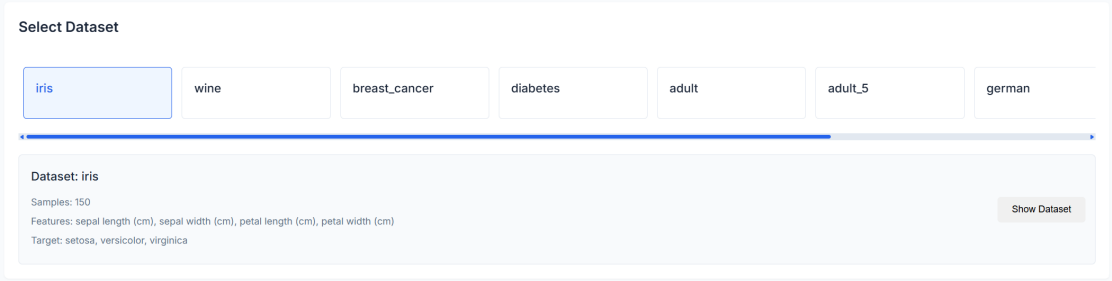
Workflow Progression Design: Each configuration stage employs **scroll-based navigation** with automatic scrolling to newly revealed sections, providing clear visual feedback about workflow progression. Completed sections remain visible above the current configuration step, enabling users to verify previous selections and

modify settings as needed. The interface implements **state preservation** across section transitions, maintaining user selections even when navigating backward through the workflow.

5.1.1 Dataset Selection Interface



(a) Before dataset selection with hover interaction.



(b) After dataset selection.

Figure 5.1: Dataset selection component.

The dataset selection component, shown in Figure 5.1 provides the workflow entry point through a **grid-based card layout** that presents available datasets with contextual information. Each dataset is represented by a distinct card containing the dataset name, basic descriptive information, and selection affordances. Dataset cards employ **hover-based visual feedback** with subtle elevation changes and border highlighting to indicate interactive affordances. The selected dataset receives persistent highlighting through distinct border coloring and background shading, providing clear visual confirmation of the current selection state. Upon dataset selection, an **information panel** appears, revealing dataset metadata including size, feature and target names. A dedicated "Show Dataset" button triggers a **slide-in panel** from the right side of the interface that displays the complete dataset in tabular format, enabling users to inspect actual data values and understand dataset characteristics before proceeding with explanation generation.

The dataset display panel, shown in Figure 5.2 implements **responsive table styling** with horizontal scrolling for wide datasets and header rows. A close button in the top right enables dismissal of the dataset panel without disrupting the main workflow.

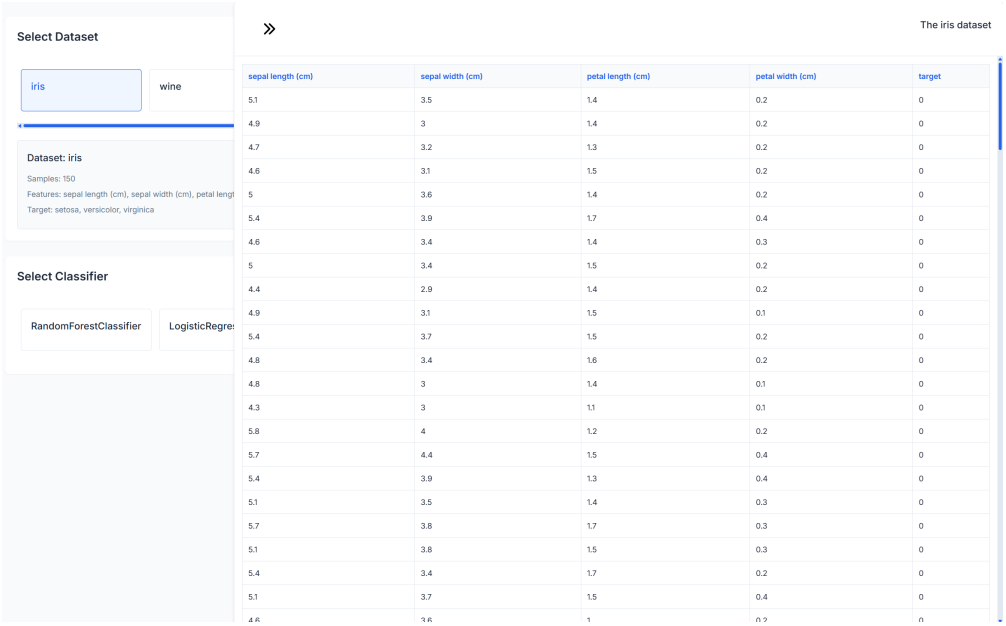


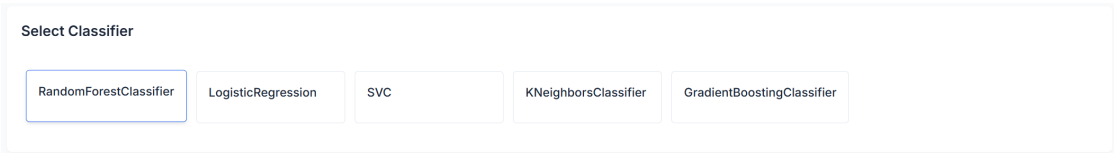
Figure 5.2: Dataset show panel.

5.1.2 Classifier Selection Interface

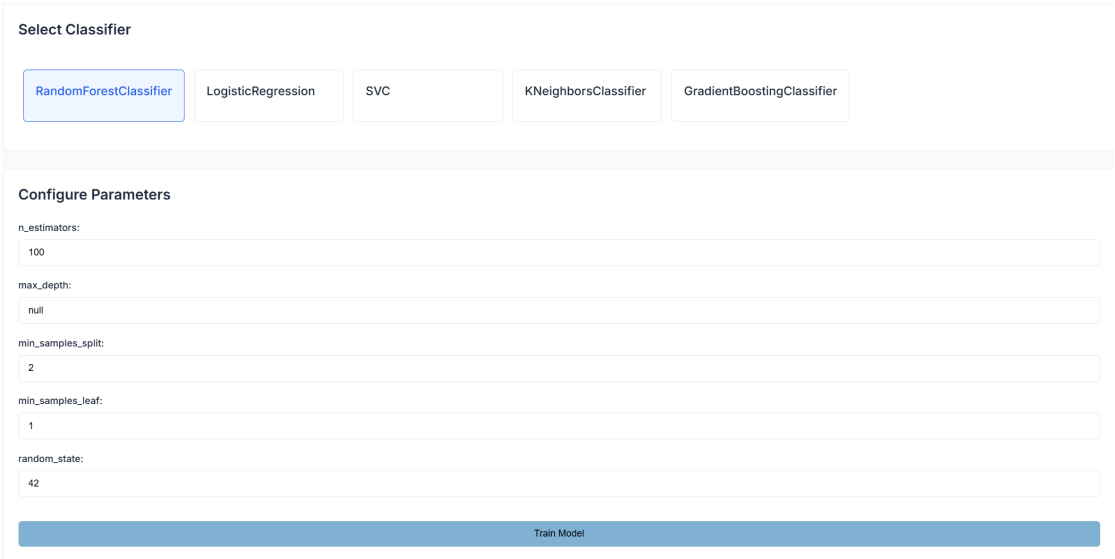
Following dataset selection, the interface reveals the **classifier selection grid**, shown in Figure 5.3 that presents available classification algorithms. The grid layout maintains visual consistency with the dataset selection interface while adapting to the number of available classifiers. Each classifier card displays the algorithm name and maintains the same interaction patterns as dataset cards. The interface supports common classification algorithms, including Random Forest, Support Vector Machines, Gradient Boost Classifier, and others, with the specific set of available classifiers dynamically populated from the backend system. The classifier selection section appears immediately below the dataset information panel through smooth scrolling animation, maintaining visual continuity in the workflow progression. The system preserves dataset selection state, allowing users to modify classifier choice without requiring dataset reselection.

5.1.3 Classifier Parameter Configuration Interface

The parameter configuration section dynamically generates appropriate input controls based on the selected classifier's possible hyperparameters. This **adaptive form generation** ensures that users encounter only relevant parameters while maintaining consistent interaction patterns across different classifiers. Parameters are organized in a **vertical stack layout** with consistent spacing and alignment, employing a clean, modern form design aesthetic. Input fields implement **real-time**



(a) Before classifier selection with hover interaction.



(b) After classifier selection.

Figure 5.3: Classifier selection component.

confirmation with immediate visual feedback for invalid entries, preventing submission of incompatible parameter combinations. At the bottom of the parameter section, the **"Train Model" button** enables workflow progression. Upon activation, the interface displays an **animated loading indicator**, providing visual feedback during potentially lengthy model training operations.

5.1.4 Instance Feature Input Interface

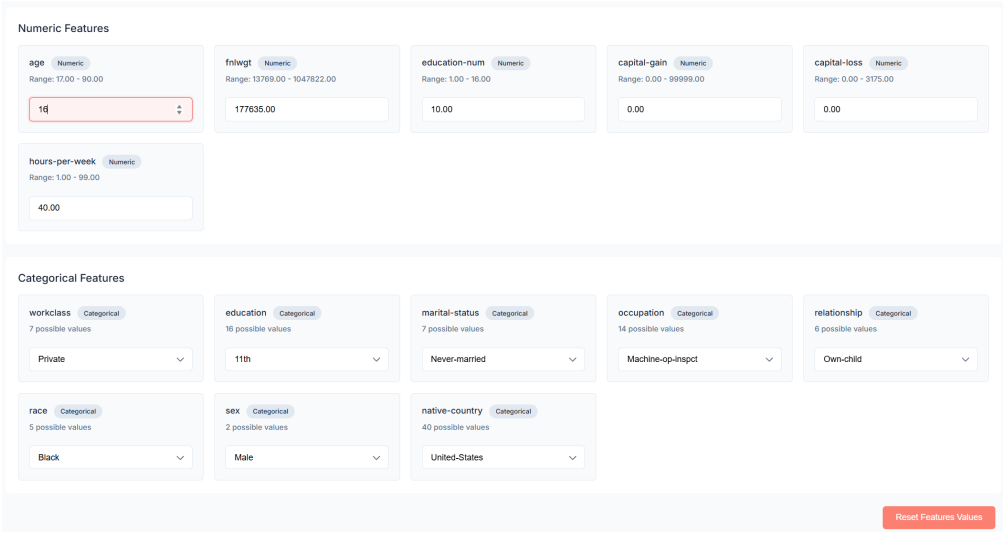


Figure 5.4: Feature input for the adult dataset with both numerical and categorical features and a out of bound input value for the feature "age".

Following successful model training, the interface reveals the **instance feature input section**, shown in Figure 5.4, that enables users to specify the instance for which they seek an explanation. This component implements **adaptive input generation** that accommodates diverse feature types while maintaining intuitive interaction patterns. The interface organizes features into sections grouped by feature type: numeric features and categorical features. This organization reduces visual complexity for datasets with many features while maintaining logical grouping of data characteristics. Numeric features receive **number input fields** with clearly displayed valid range information. Each input shows minimum and maximum acceptable values derived from training data characteristics, but still allows users to enter out-of-bounds values, showing at the same time when the value is out of bounds with a red highlight. The interface displays default values calculated as features median value, providing sensible starting points for exploration. On the other hand, categorical features utilize **dropdown select menus** populated with valid category values. The interface displays category options in a scrollable list. Default selections represent the mode value for the specific feature.

A dedicated **"Reset Features Values" button** positioned above the feature sections enables users to restore all inputs to their default values, mean, and mode for either numerical or categorical features, facilitating exploration of different instance configurations without manual re-adjustment of each field.

5.1.5 Explanation Parameters Settings

Figure 5.5: Explanation Parameters input.

Below the feature input interface, the system presents the **"Explanation Parameters"** section, shown in Figure 5.5, which controls some LORE$_{sa}$ explanation generation parameters and visualization configuration settings.

Neighbourhood Size: The interface provides **numeric input controls** for synthetic neighborhood size, allowing users to specify the number of instances to generate around the explained instance. This parameter critically affects explanation quality, with larger neighborhoods providing more robust rule extraction at the cost of increased computational time.

PCA Scatter plot granularity: For PCA projections, the interface includes a **granularity control** that determines the resolution of Voronoi tessellation used for decision boundary visualization. Lower values create coarser approximations with better performance, while higher values produce more detailed boundary representations at increased computational cost.

Include original dataset in scatter plot: The interface provides a boolean toggle control that enables users to overlay the present original dataset onto the scatter plot visualization alongside the synthetic neighborhood. When enabled, original dataset points are rendered with reduced opacity to distinguish them from synthetic instances while providing crucial context about how the generated neighborhood relates to the broader data distribution. This feature facilitates assessment of neighborhood representativeness by enabling direct visual comparison between synthetic instances and actual training data patterns.

Keep the duplicates in the generated neighborhood: The interface includes thi boolean toggle control that determines whether the genetic algorithm's neighborhood generation process preserves or removes duplicate instances. When enabled, all generated instances are retained even if they have identical feature values, which may be relevant for analyzing the genetic algorithm's convergence behavior, but will increase the computational overhead in visualization rendering.

5.1.6 Dimensionality Reduction techniques Parameters Settings

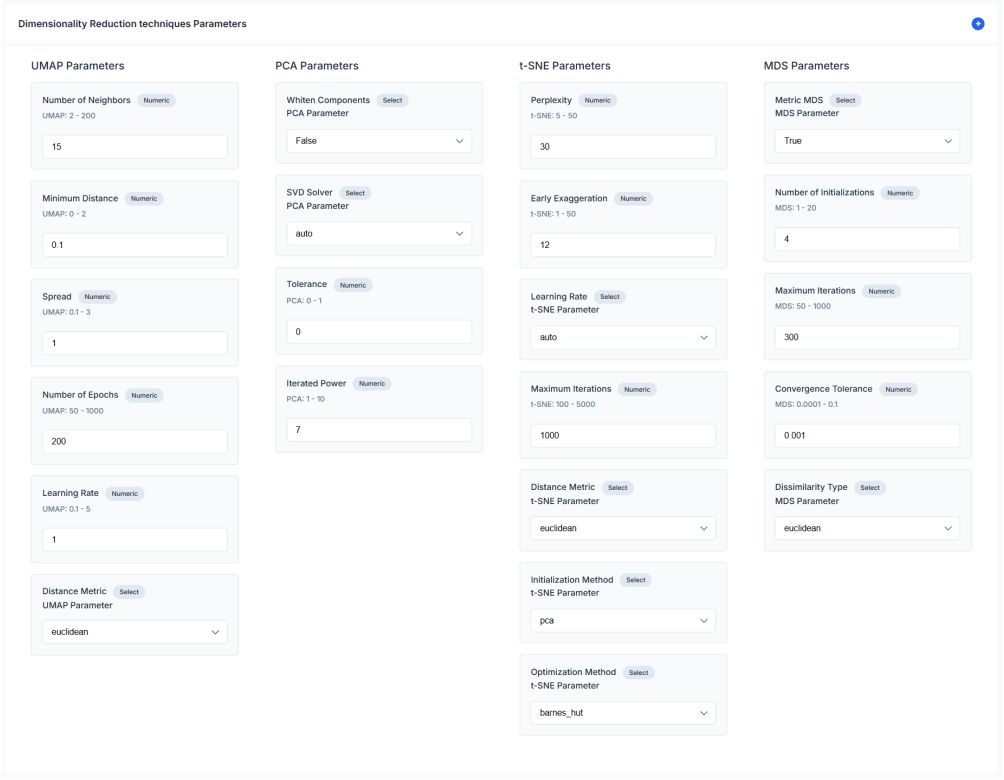


Figure 5.6: Dimensionality Reduction Parameters.

The interface implements **expandable parameter sections** for each supported dimensionality reduction technique. Users can configure UMAP's 'n_neighbors' and 'min_dist' parameters, t-SNE's perplexity and learning rate, PCA's component selection, and MDS metric specifications and many other parameters for each of the dimensionality reduction techniques, as show in Figure 5.6. These advanced controls are presented in collapsible sections to avoid overwhelming novice users while remaining accessible to experts seeking fine-grained control.

5.1.7 Visualization Selection Toggles

A critical component is the **visualization selection interface** that enables users to control which visualization components appear in the final explanation display. The interface presents **four checkbox toggles** corresponding to the Neighborhood 2D projection, Rule and Counterfactual Rules Centred surrogate model, Tree Layout surrogate model, and Rule Centered surrogate model visualizations.

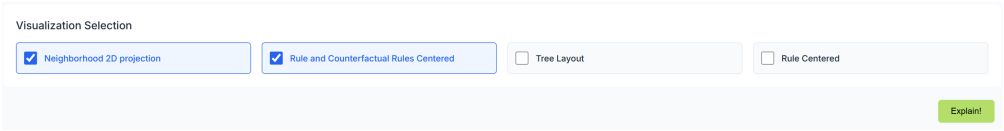


Figure 5.7: Visualization Selection panel, allowing the user to choose which visualizations to render.

Each visualization toggle employs a **checkbox with descriptive label** arranged in a grid layout. The toggles include visual indicators showing which visualizations are currently enabled, with checked boxes receiving enhanced styling through background color changes and border highlighting. The Neighborhood 2D projection is, as previously said, selected by default and can not be deactivated, the user can then choose to render either of the three realized visualizations. The default selection is `Rule and Counterfactual Rules Centered`, and if the user chooses to select one of the other two the selection will behave as with an HTML ratio input.

At the workflow conclusion, a the **"Explain!" button** starts the explanation generation. Upon click, the system displays an **animated loading interface** with progress indicators and descriptive text explaining the explanation generation process. This provides crucial user feedback during potentially lengthy operations involving genetic algorithm-based neighborhood generation and surrogate model training.

5.2 Coordinated Visualization System

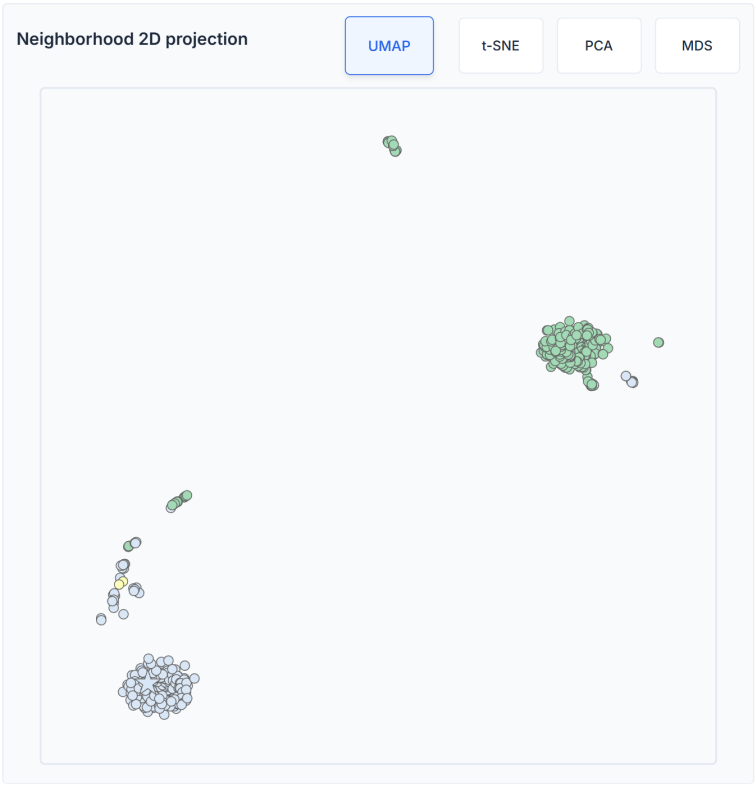
Following successful explanation generation, the interface transitions to present the **coordinated visualization system** that displays the results through the two complementary views. The visualization system architecture centers on **coordinated multiple views** [53], implementing two distinct but interconnected visualization components: the 2 dimensionality-reduced scatter plot for neighborhood exploration, and one of the three surrogate model visualizations that provide different points of view on the same surrogate model structure.

This multi-representation approach addresses the diverse cognitive approaches users may take when interpreting machine learning explanations. The visualization components appear below the configuration interface, maintaining access to configuration controls while presenting explanatory results.

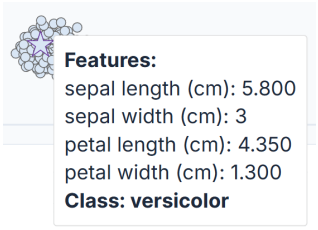
The Bidirectional coordination serves as the fundamental interaction paradigm, enabling information flow between spatial and symbolic representations. Users can initiate interactions from any of the two visualization components, with the system automatically propagating highlighting and selection states across the other view.

This coordination maintains visual consistency through synchronized color schemes and ensures that users can confirm explanations by cross-referencing between different representation modalities.

5.3 2D spatial neighborhood analysis Visualization



(a) Example of generated spatial neighborhood analysis plot.



(b) Example of hover on a spatial neighborhood analysis plot point.

Figure 5.8: Generated spatial neighborhood analysis plot.

The spatial neighborhood analysis plot component, example of which is shown in Figure 5.8a, represents the spatial interface for exploring the synthetic neighborhood, serving as the primary mechanism for understanding the quality and distribution of generated instances in reduced-dimensional space. The implementation supports four distinct dimensionality reduction techniques, each accessible through radio button controls integrated directly into the visualization header.

The interface provides the switching mechanism between **Uniform Manifold Approximation and Projection (UMAP)**, **t-distributed Stochastic Neighbor Embedding (t-SNE)**, **Principal Component Analysis (PCA)**, and **Multidimensional Scaling (MDS)**. Each method offers distinct advantages for

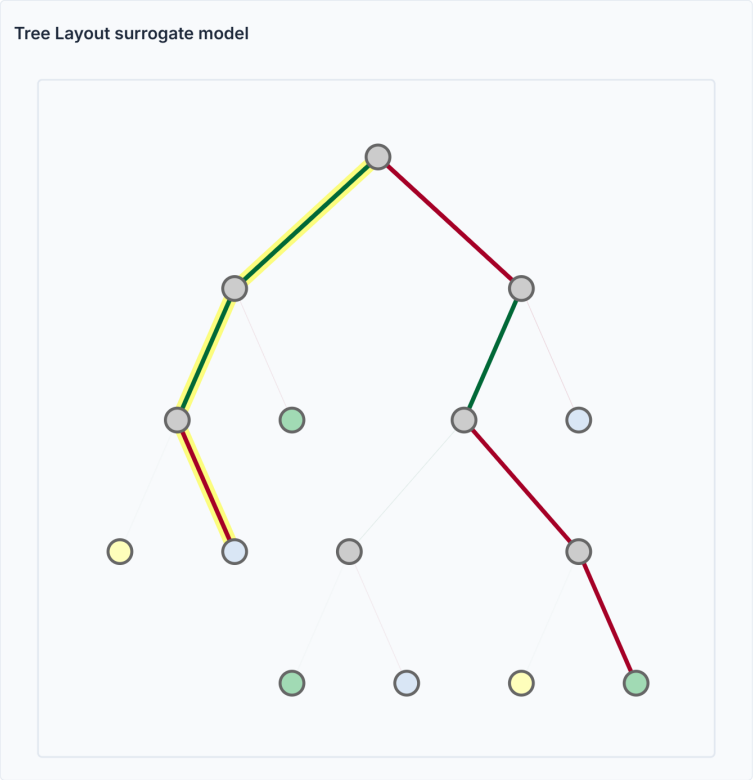
different analytical scenarios. The radio button controls appear immediately above the spatial neighborhood analysis plot, enabling rapid switching between projection methods without requiring navigation away from the visualization context.

The spatial neighborhood analysis plot employs a sophisticated multi-layered visual encoding approach that maximizes information density while maintaining clarity. **Point symbols** distinguish between synthetic neighborhood instances (circles) and the explained instance (star), with symbol size differentiation ensuring the explained instance remains clearly identifiable within the neighborhood context. **Color encoding** employs a perceptually uniform color scheme projected from the CIELAB color space, ensuring consistent class differentiation across varying numbers of target categories. Opacity-based differentiation addresses the challenge of displaying both original dataset points and synthetic neighborhood instances within the same visualization space. Original dataset points are rendered with reduced opacity, creating a contextual background that helps users understand how the synthetic neighborhood relates to the broader data distribution without overwhelming the primary explanation content.

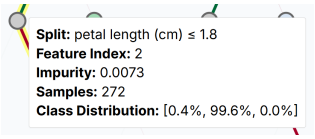
For linear transformations (PCA), the system implements **Voronoi tessellation-based decision boundaries** that provide explicit visualization of classification regions within the projected space. The Voronoi implementation employs user-configurable granularity settings (controlled through the Surrogate Model Parameters section), allowing adjustment of boundary precision based on computational constraints and visualization clarity requirements. Semi-transparent polygon regions use class-consistent coloring to create intuitive decision boundary representations that enable direct assessment of explanation quality through boundary geometric properties.

The spatial neighborhood analysis plot supports comprehensive interaction mechanisms designed to facilitate explanation exploration. **Zoom and pan functionality** enables detailed examination of neighborhood regions, with smooth transitions maintaining spatial orientation. The zoom implementation preserves aspect ratios and provides intuitive mouse wheel or pinch-to-zoom interactions. The hover interactions, example of which is shown in Figure 5.8b, reveal detailed point information through contextual tooltips that display decoded feature values, class predictions, and positional coordinates. The tooltips follow cursor movement smoothly and employ **details-on-demand** principles [53], appearing only when needed to avoid visual clutter. The point selection interactions serve as the primary mechanism for triggering cross-visualization coordination. Clicking any point initiates highlighting of the corresponding decision path in the selected tree visualization, while simultaneously highlighting the selected point with a distinct color. Clicking the same point again deselects it and clears all highlighting, providing intuitive toggle behavior.

5.4 Tree Layout Visualization



(a) Tree Layout visualization.



(b) Tooltip on the Tree Layout visualization for a split node

(c) Tooltip on the Tree Layout visualization for a leaf node

Figure 5.9: Tree Layout visualization and tooltips.

The Tree Layout visualization, shown in Figure 5.9a, provides the foundational hierarchical representation of the surrogate model, implementing established node-link diagram conventions while incorporating interaction capabilities. This visualization serves as a primary reference for understanding logical rule structure and decision flow.

The implementation employs a **traditional top-down hierarchical layout** with systematic node positioning that prevents overlapping while maintaining clear parent-child relationships. **Circular nodes** represent decision points and leaves. The color encoding strategy distinguishes between split nodes and leaf nodes through coordinated class-based coloring. **Split nodes** (representing logical conditions) employ neutral gray coloring to emphasize their decision-making role without competing with class-specific colors. **Leaf nodes** receive coloring that directly corresponds to their predicted class, using the same color scheme employed in the spatial neighborhood analysis visualization to maintain cross-visualization

(a) Click on leaf node interaction.

(b) Click on split node interaction.

Figure 5.10: Classical decision tree click interactions.

consistency. Edge visualization uses variable stroke width encoding to represent the flow of synthetic instances through decision branches. Thicker edges indicate paths with more instances, providing immediate visual feedback about decision tree utilization patterns. The edges also visually encode which split outcome they pursue, with negative ramification being red and green for the opposite case.

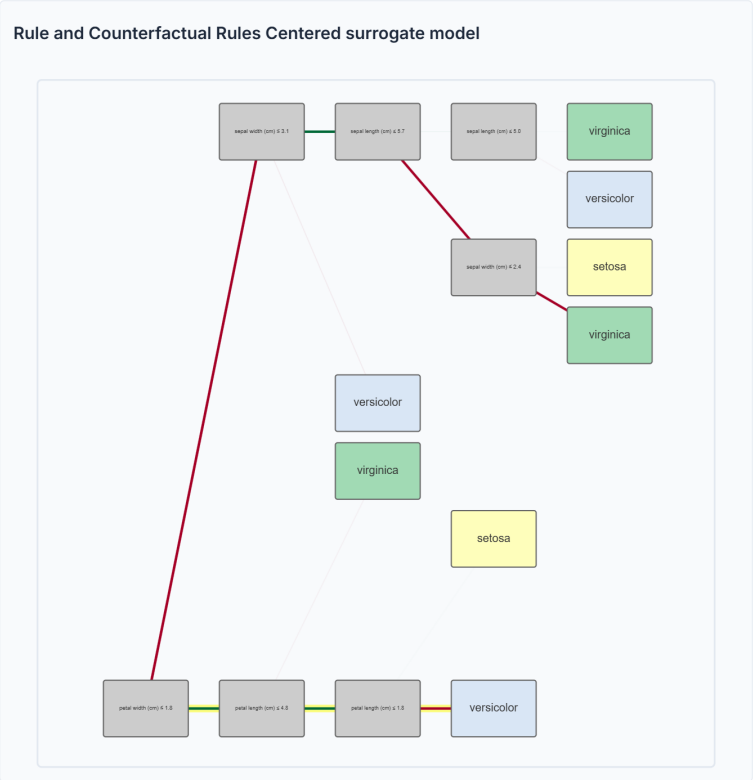
Tooltip information provides comprehensive contextual details through details-on-demand interaction. Split node tooltips, shown in Figure 5.9b, reveal feature names, threshold values, impurity measures (Gini or entropy), sample counts, and class distribution statistics. Leaf node tooltips, shown in Figure 5.9c, display class prediction and sample counts. This layered information approach allows users to access varying levels of detail based on their analytical requirements.

The Tree Layout visualization implements interaction mechanisms that coordinate with all other visualization components. **Node click interactions** trigger highlighting of corresponding instances in the spatial neighborhood analysis plot, with leaf node clicks, showcased in Figure 5.10a, highlighting all instances satisfying the complete logical path, and split node clicks, showcased in Figure 5.10b, highlighting instances passing through specific decision points. Path highlighting provides persistent visual feedback for explained instance trajectories through the tree. The explained instance's path from root to leaf receives continuous highlighting using stroke width enhancement and color differentiation, ensuring users can easily trace the logical conditions that led to the specific prediction.

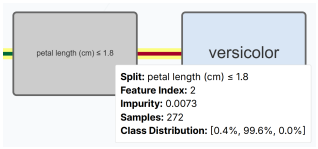
The Tree Layout visualization includes **interactive zoom and pan capa-**

bilities that enable detailed examination of complex tree structures. Users can mouse-wheel zoom or use trackpad gestures to adjust magnification levels, with the system maintaining tree center positioning and providing smooth interpolation between zoom states.

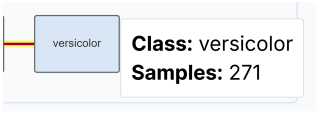
5.5 Rule and Counterfactual Rules Centered Visualization



(a) Rule and Counterfactual Rules Centered plot.



(b) Tooltip on the Rule and Counterfactual Rules Centered visualization for a split node



(c) Tooltip on the Rule and Counterfactual Rules Centered visualization for a leaf node

Figure 5.11: Rule and Counterfactual Rules Centered visualization and tooltips.

The Rule and Counterfactual Rules Centered visualization, shown in Figure 5.11a, represents an innovative approach to hierarchical visualization that addresses layout efficiency challenges inherent in traditional node-link diagrams. This visualization employs **rectangular node representations** with **depth-aligned positioning** to create more compact and readable tree layouts, particularly effective for complex decision structures.

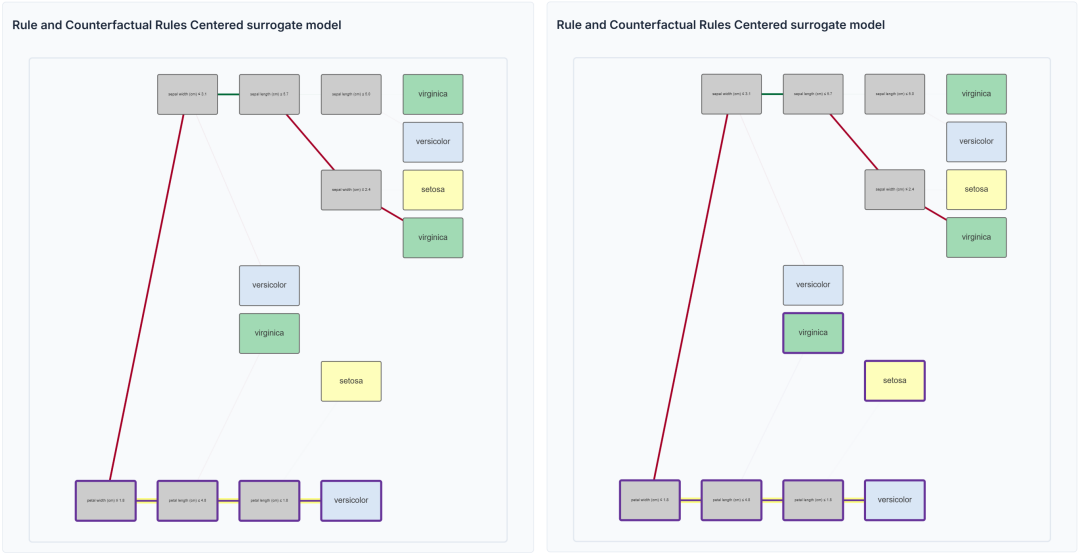
Unlike traditional tree layouts that position nodes based on hierarchical relationships with varying vertical spacing, the Rule and Counterfactual Rules Centered visualization employs **horizontal depth alignment** where all nodes at the same tree depth occupy the same vertical level. This approach creates a more structured, grid-like appearance that facilitates comparison of decision complexity across different tree levels. A key design feature of the Rule and Counterfactual Rules Centered visualization the **guaranteed bottom positioning of the explained instance path**. This path in the tree is always rendered at the bottom of the visualization; therefore users can quickly locate the specific decision path of the generated rule, focusing their attention on the bottom region of the display. This consistent spatial organization reduces visual search time and cognitive load, particularly when exploring alternative decision paths.

Rectangular nodes replace circular representations to maximize information density while improving text readability. Node dimensions adapt dynamically to accommodate textual content, with font sizing automatically adjusting to maintain readability across varying node sizes. Split nodes display decision conditions clearly within the rectangular bounds, while leaf nodes prominently feature class predictions and sample count information. **Straight-line connections** between rectangular nodes provide clear, unambiguous path representation. Variable stroke width encoding continues to represent instance flow, with stroke width proportional to the number of samples traversing each edge. **Color-coded edges** distinguish between "true" (left) and "false" (right) decision paths, using the same consistent color scheme employed across all visualizations.

The blocks tree provides comprehensive contextual details through details-on-demand interaction, matching the Tree Layout visualization. Split node tooltips, shown in Figure 5.11b, reveal feature names, threshold values, impurity measures, sample counts, and class distribution statistics within the rectangular node format. Leaf node tooltips, shown in Figure 5.11c, display class distributions, confidence measures, and sample counts, maintaining consistency with the classic tree's information architecture.

The Rule and Counterfactual Rules Centered visualization implements the same bidirectional coordination mechanisms as the Tree Layout visualization, with node interactions triggering corresponding highlights in the spatial neighborhood analysis plot and other tree visualizations. **Rectangle selection**, demonstrated in Figures 5.12a and 5.12b, provides more generous interaction targets compared to circular nodes, improving usability particularly for complex trees with many small nodes. Leaf node clicks highlight all instances satisfying the complete logical path, while split node clicks highlight instances passing through specific decision points.

The rectangular format enables more effective integration of statistical information within node representations. Sample counts, impurity measures, and class



(a) Click on leaf node interaction.

(b) Click on split node interaction.

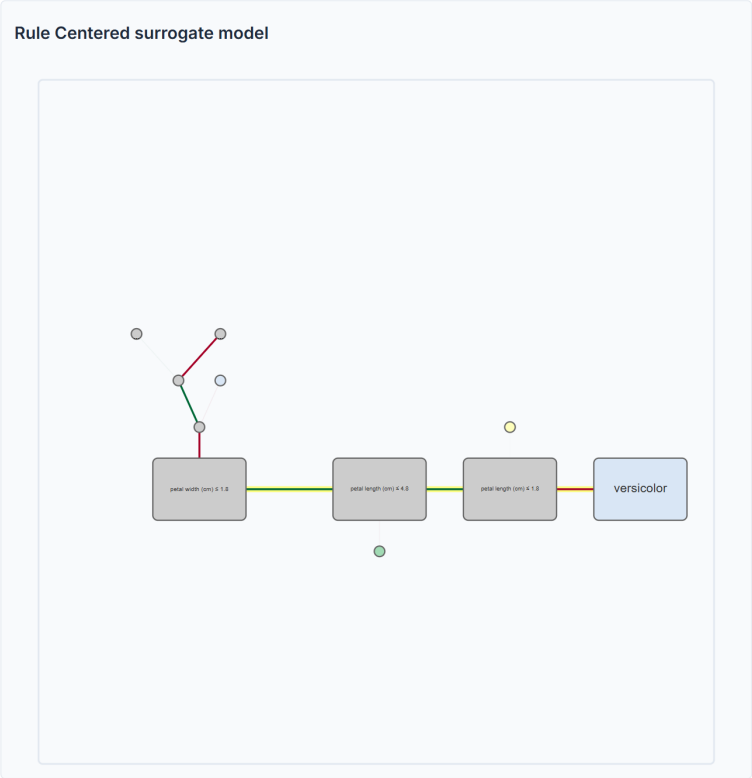
Figure 5.12: Blocks decision tree click interactions.

distributions can be displayed more prominently within rectangular bounds, reducing reliance on tooltip interactions for frequently accessed information. This design decision reflects the principle of making important information directly visible rather than requiring interaction to access it.

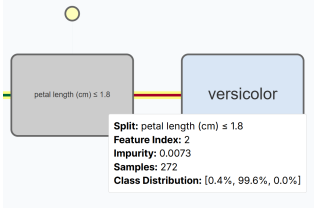
5.6 Rule Centered visualization

The Rule Centered visualization, shown in Figure 5.13a, implements a novel **spawn-based positioning algorithm** that optimizes space utilization while providing enhanced focus on explanation paths. This visualization addresses the challenge of presenting complex tree structures while maintaining clear focus on the instance-specific explanation trajectory. The core innovation lies in the **adaptive spacing mechanism** that adjusts positioning based on subtree complexity. Rather than using uniform spacing between tree levels, the spawn algorithm calculates spacing dynamically based on the size and complexity of subtrees at each level. This approach ensures that complex tree regions receive adequate space while compact regions maintain efficient layouts.

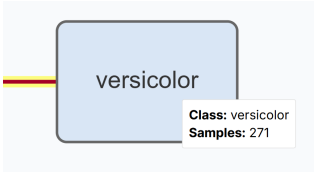
The algorithm specifically analyzes the **size of subtrees not involved in the explained instance path** and adjusts spacing between path nodes accordingly. When a path node has large subtrees branching off, the horizontal spacing to the next path node increases proportionally, preventing overlap and maintaining visual



(a) Rule Centered visualization plot.



(b) Tooltip on the Rule Centered visualization for a split node



(c) Tooltip on the Rule Centered visualization for a leaf node

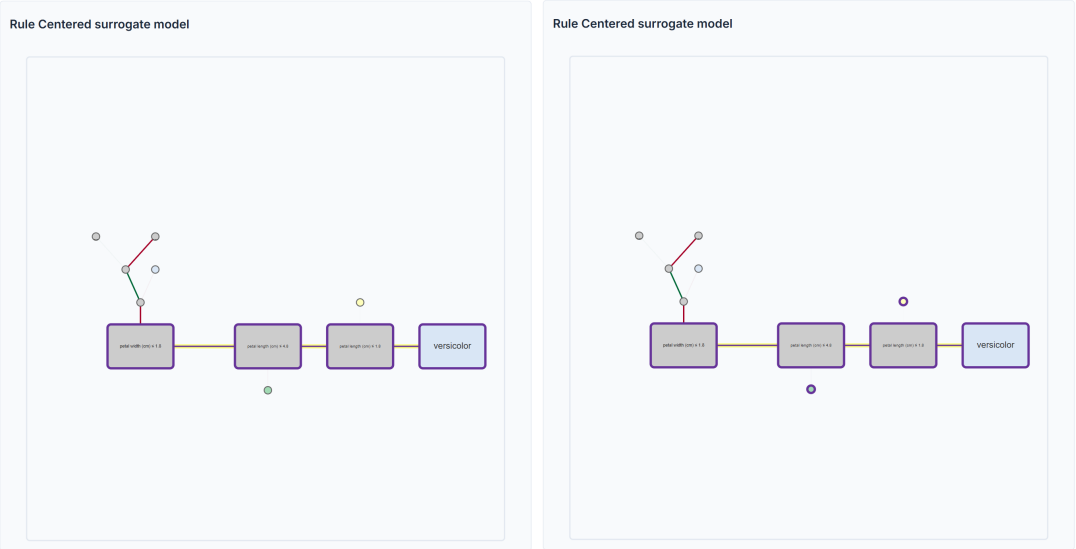
Figure 5.13: Rule Centered visualization and tooltips.

clarity. The Rule Centered visualization prioritizes the **explained instance path** through enhanced visual treatment and positioning optimization. Path nodes receive **rectangular representation** with distinctive styling including thicker borders and brighter colors, while non-path nodes maintain **circular representation** to visually distinguish between explanation-relevant and contextual information.

The Rule Centered visualization provides contextual details consistent with the other tree visualizations. Split node tooltips, shown in Figure 5.13b, display feature names, threshold values, impurity measures, sample counts, and class distributions. Leaf node tooltips, shown in Figure 5.13c, present predicted class and sample counts, maintaining the established information architecture across all tree visualizations.

A key innovation in the Rule Centered visualization is the implementation of **interactive subtree collapse/expand functionality**, illustrated in Figure 5.15. Subtrees not involved in the explained instance path can be collapsed to reduce visual complexity, allowing users to focus on explanation-relevant information while maintaining the option to explore alternative decision paths when needed.

Collapsed subtrees are represented by **simplified placeholder nodes**, shown in Figure 5.15b, which appear as circles with special visual marking to indicate



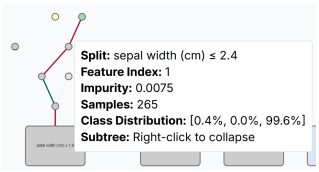
(a) Click on leaf node interaction.

(b) Click on split node interaction.

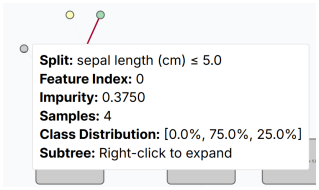
Figure 5.14: Rule Centered visualization click interactions.

hidden structure. When users hover over these placeholder nodes, the system provides the **information** about the collapsed subtree.

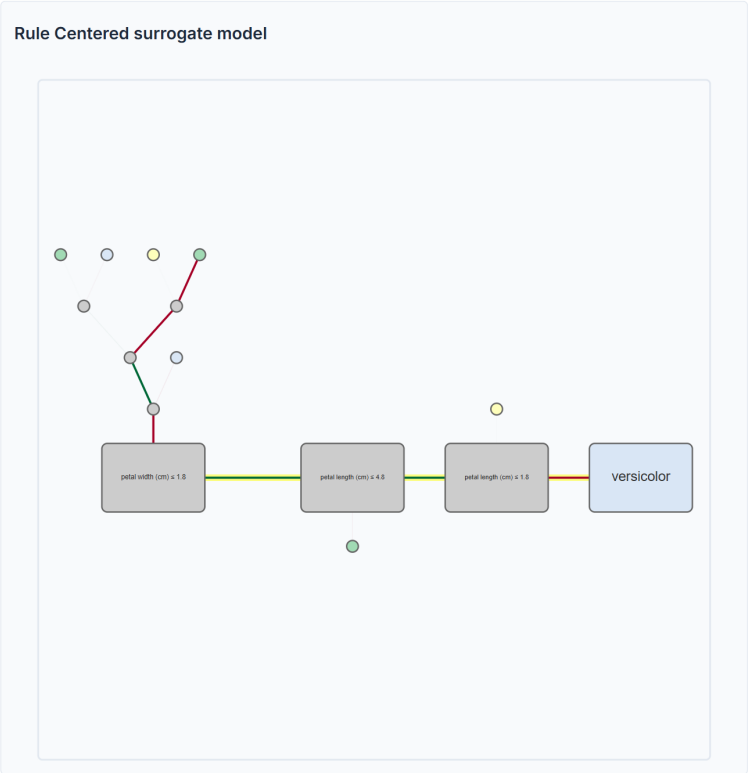
Right-click **context menu interactions** provide intuitive access to subtree expansion controls. When users right-click on a collapsed subtree placeholder node, a context menu appears with options to "Expand Subtree", enabling progressive disclosure of tree complexity. Figure 5.15c demonstrates the result of subtree expansion, revealing the complete decision structure that was previously hidden. Expanded subtrees, as shown in Figure 5.15a, can similarly be collapsed through right-click interactions, providing bidirectional control over information density. This interaction pattern supports **exploratory analysis workflows** where users can progressively reveal complexity as needed rather than confronting the full tree structure immediately. The spawn tree employs **visual hierarchy differentiation** to guide user attention effectively. Explained instance path nodes receive enhanced visual treatment through increased size, distinctive rectangular shapes with rounded corners, and prominent borders. The collapsed and expanded states, illustrated in Figure 5.15, is communicated through smaller circles with special marking, maintain clear visual distinction between explanation-relevant paths and contextual information. Beyond standard click and hover interactions demonstrated in Figures 5.14a and 5.14b, the spawn tree supports **contextual interaction modes** that adapt based on node type and state. Path nodes provide detailed explanation information through tooltips showing feature conditions, sample statistics, and impurity measures. Split node clicks trigger the same cross-visualization coordination as



(a) Hover interaction on an expanded subtree node



(b) Hover interaction on a collapsed subtree node



(c) Fully expanded subtree visualization showing additional decision paths

Figure 5.15: Rule Centered visualization subtree collapse/expand interactions and states.

other tree types, highlighting corresponding instances in the spatial neighborhood analysis plot.

5.7 Integrated Coordination System

The final design's greatest strength lies in its **comprehensive coordination architecture** that seamlessly connects all visualization components through bidirectional interaction mechanisms.

Interaction Flow: When a user clicks a point in the spatial neighborhood analysis plot, as shown in Figure 5.16, the coordinator identifies the corresponding feature vector, traces the decision path through the rendered tree structure, and propagates highlighting to it. Conversely, when a user clicks a tree node, as shown in Figure 5.17 the coordinator identifies all synthetic neighborhood instances that satisfy the node's conditions and highlights the corresponding points in the spatial neighborhood analysis plot.

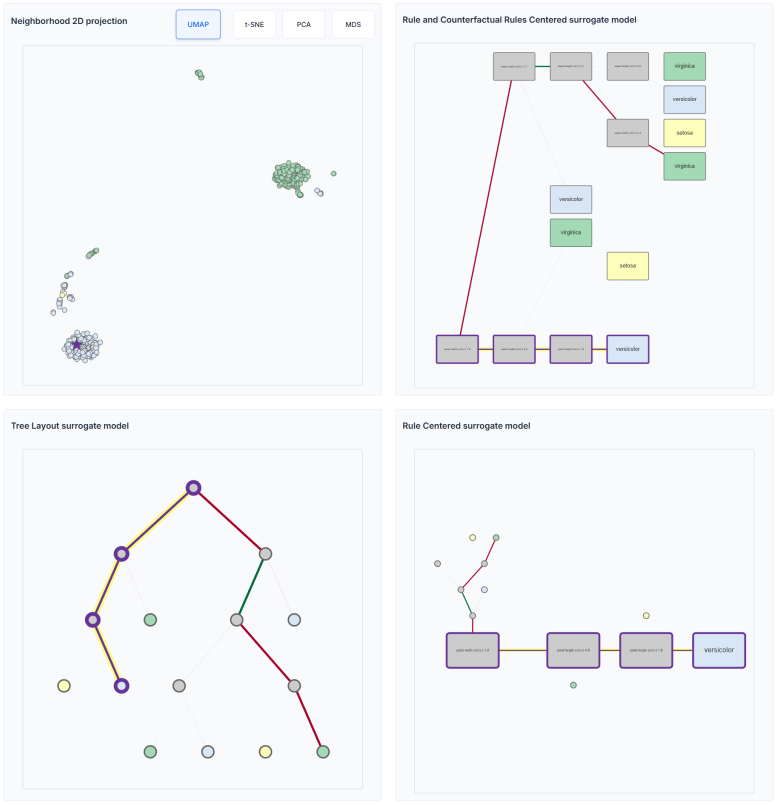


Figure 5.16: Click on spatial neighborhood analysis plot point interaction.

Color Scheme Consistency: All visualizations employ a unified color scheme, either from the treeviz library [46], shown in Table 5.1, or, when the number of the dataset classes is higer than 10, the mean points of each class of the dataset are projected in the CIELAB color space with L*=70, ensuring perceptual uniformity across different numbers of target classes. This approach addresses the challenge of maintaining distinguishable colors for datasets with varying numbers of classes (from 2 to 10+) while ensuring consistent visual experience across all visualization components and, up until 10 classes, color-blindness friendliness.

The final design successfully integrates the spatial and symbolic representation paradigms through coordinated multiple views, providing users with comprehensive tools to confirm, explore, and understand the generated explanations. The configuration interface enables flexible control over the explanation generation process, while the multi-tree approach accommodates diverse cognitive preferences, and the spatial neighborhood analysis plot provides essential spatial context for understanding neighborhood quality and decision boundary characteristics.

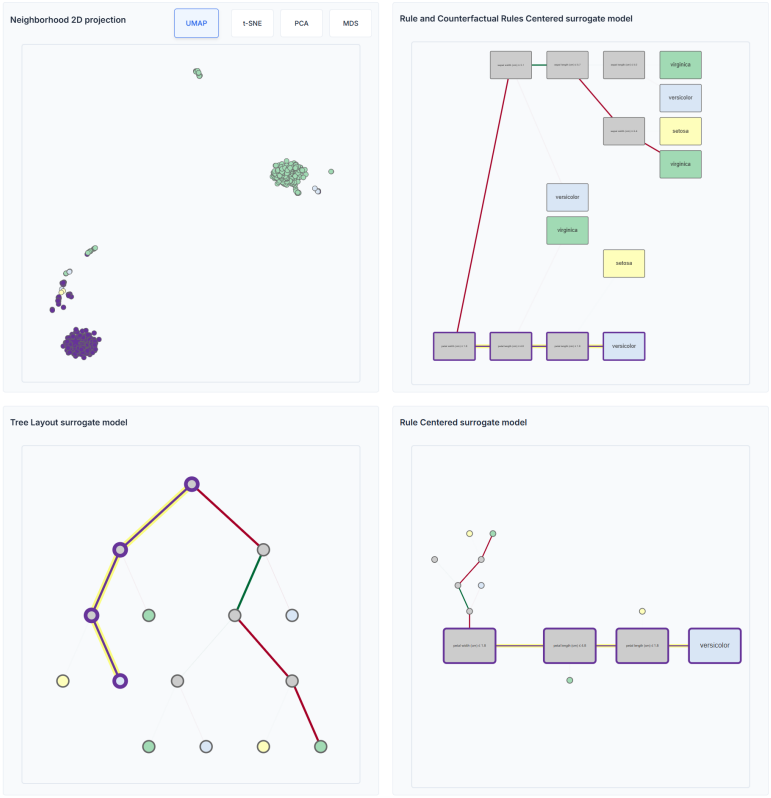


Figure 5.17: Click on decision tree plot interaction.

Table 5.1: Color blind friendly color palettes.

Number of Classes	Color 1	Color 2	Color 3	Color 4	Color 5	Color 6	Color 7	Color 8	Color 9	Color 10
2	#FEFEBB	#a1dab4								
3	#FEFEBB	#D9E6F5	#a1dab4							
4	#FEFEBB	#D9E6F5	#a1dab4	#fee090						
5	#FEFEBB	#D9E6F5	#a1dab4	#41b6c4	#fee090					
6	#FEFEBB	#c7e9b4	#41b6c4	#2c7fb8	#fee090	#f46d43				
7	#FEFEBB	#c7e9b4	#7fcdbb	#41b6c4	#225ea8	#fdae61	#f46d43			
8	#FEFEBB	#edf8b1	#c7e9b4	#7fcdbb	#1d91c0	#225ea8	#fdae61	#f46d43		
9	#FEFEBB	#c7e9b4	#41b6c4	#74add1	#4575b4	#313695	#fee090	#fdae61	#f46d43	
10	#FEFEBB	#c7e9b4	#41b6c4	#74add1	#4575b4	#313695	#fee090	#fdae61	#f46d43	#d73027

Chapter 6

Use cases

Following the design bestow, the following chapter aims at presenting possible scenarios in which the realized system, with its integrated and interconnected visualizations, might aid different kinds of users. We analyze the use of the interface both through the lens of a professor explaining how eXplainability methods work or a student who is approaching the subject of eXplainable Artificial Intelligence for the first time – but has some previous knowledge regarding common techniques, such as commonly used datasets, commonly used classifiers, and dimensionality reduction techniques – both through the lens of a more experienced data scientist or someone with deeper field knowledge that could use the system to analyze it's own classifier reliability and explore how the chosen eXplainability method (LORE_{sa}) explains the model.

Through those very distant lenses, we aim to provide the reader with a comprehensive understanding of how the system can be interacted with and how the users who might fall in between the two exposed cases might interact with the system.

6.1 Use case: teaching

The following section proposes a teaching scenario in which the interactive subjects are a teacher and a classroom, where the proposed tool is used to explain eXplainable Artificial Intelligence concepts.

The teacher prepares for an advanced machine learning seminar where they will introduce students to eXplainable AI methods, specifically focusing on local explanation techniques that generate synthetic neighborhoods and extract symbolic decision rules and their explaniability method of choice is LORE_{sa}. They recognize that previous approaches to teaching these concepts through the textual rule representations presented significant pedagogical challenges. When students examined traditional explanation outputs consisting of nested dictionaries contain-

ing premises, consequences, and counterfactual conditions, they struggled to build an intuitive understanding of how these logical structures related to the actual decision-making process. The cognitive load of parsing multiple rule conditions while simultaneously trying to understand their interpretation in feature space created a barrier in learning.

The teacher decides to use the interactive visualization system and the Iris dataset for the demonstration. The Iris dataset offers intuitive semantics that help students focus on understanding explanation mechanisms rather than wrestling with domain complexity.

6.1.1 Initial Configuration and Conceptual Foundation

The teacher navigates to the demonstration interface and begins by selecting the Iris dataset. As they click the dataset card, the system reveals the classifier selection interface, and they explain to students why a Random Forest classifier with default parameters is chosen: Random forests operate as black-box ensembles, making their individual predictions difficult to interpret despite their strong performance. This opacity motivates the need for explanation methods that can reveal the decision logic for specific instances.

On top of that, the interface allows the teacher to showcase the dataset in its entirety by the click of the `Show Dataset` button. Allowing the students to observe the actual values of the dataset in use, as shown in Figure 6.1.

Figure 6.1: Iris dataset snippet shown at the click of the `Show Dataset` button

The interface automatically displays the feature input section, organizing the four Iris features into their numerical group. The teacher observes the median default values populated in each field: sepal length (5.80 cm), sepal width (3.00 cm), petal length (4.35 cm), and petal width (1.30 cm), and uses this moment to introduce a fundamental concept: These median values represent a typical instance from the dataset. When we explain a prediction for this instance, we are learning about how the classifier behaves in a common region of the feature space, not at an extreme edge case.

The teacher scrolls to the Explanation Parameters section and highlights the neighborhood size control, currently set to 500 instances. This parameter determines how many synthetic instances the explanation method generates around our target instance. This parameter is controlling the size of our local investigation, answering the question: How does the classifier behave in this specific neighborhood of feature space? They lower the default value, noting that for the relatively simple Iris dataset, 200 instances provide more than adequate coverage of the local decision space, over the 150 total samples of the dataset.

The teacher leaves the dimensionality reduction parameters at their defaults and enables the original dataset overlay option, explaining that by including the original training data in the visualization, they can observe how the synthetic neighborhood relates to the actual data distribution. This helps in assessing whether the local explanation captures realistic regions of the feature space.

They select the Rule and Counterfactual Rules Centered visualization from the tree layout options, as shown in Figure 6.2, reasoning that its rectangular node structure and guaranteed bottom positioning of the explained instance path provides the clearest initial introduction to decision tree structure. With configuration complete, they click the `Explain!` button and the system later displays the results of the explanation.

6.1.2 First Encounter: Understanding Spatial Neighborhoods

When the visualizations appear, the teacher directs students' attention first to the 2D spatial neighborhood projection, displayed with UMAP dimensionality reduction as shown in Figure 6.3a. The plot reveals four distinct clusters of points, color-coded in #FEFEBB (setosa), #D9E6F5 (versicolor), #a1dab4 (virginica), and a mixed one with #D9E6F5 (versicolor), #a1dab4 (virginica), with the explained instance marked by the star symbol within the versicolor cluster.

The teacher focuses the attention of the students on how the dimensionality reduction has organized the four-dimensional Iris feature space into this two-dimensional projection. The teacher points out that the spatial separation between

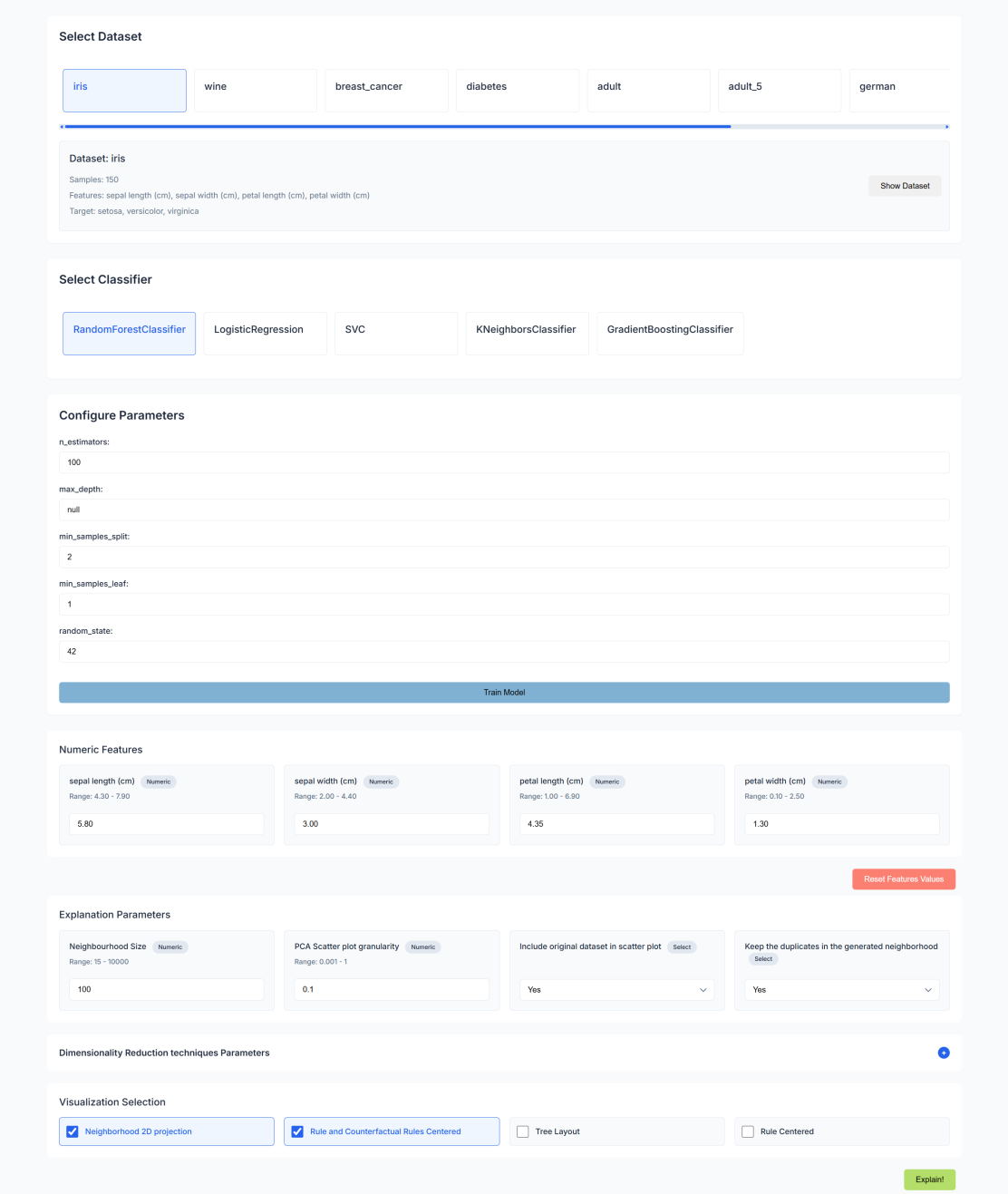
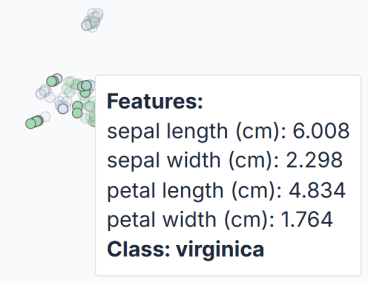
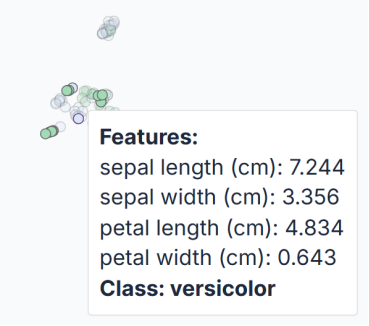


Figure 6.2: Complete configuration of the system before generating the explanation for the iris dataset's instance.

(a) Neighborhood 2D projection



(b) Virginica class instance near the decision boundary.



(c) Versicolor class instance near the decision boundary.

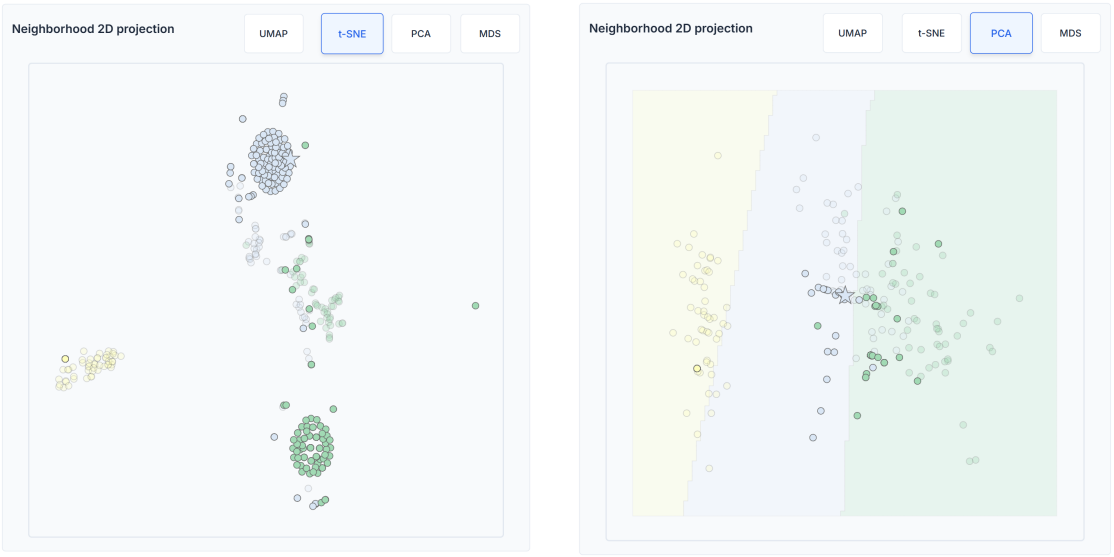
Figure 6.3: Initial scatter plot on the Iris dataset using UMAP dimensionality reduction.

clusters immediately reveals something important about the classification task: setosa flowers are clearly distinct from the other two species, while versicolor and virginica show some overlap in their decision boundaries.

The teacher hovers over several points in the scatter plot, and students observe the tooltip displaying feature values and class predictions. The teacher highlights how the synthetic instances are concentrated more densely near the explained instance, and notes that this is not a random occurrence. The genetic algorithm that generated these instances specifically optimizes for neighborhood quality, creating some instances, shown in Figures 6.3b and 6.3c, near decision boundaries where the classification behavior is most interesting. This observation provides the first opportunity to contrast with the limitations of previous approaches: If we were looking at the traditional textual output, we would see a list of counterfactual samples with their feature values, but we would have no intuitive sense of how these instances are distributed in space. The visualization makes this distribution immediately comprehensible.

The teacher clicks the UMAP radio button off and selects t-SNE instead, as shown in Figure 6.4a. The projection reorganizes, maintaining the three-cluster

structure but with different spatial relationships. For teaching purposes, being able to switch between these projections helps understanding that there is no single 'correct' way to visualize high-dimensional data and that each technique offers different insights. The teacher later switches to PCA, noting how the linear projection creates a different perspective where the explained instance sits at the boundary between clusters rather than clearly within one. The decision boundaries generated by the linear surrogate model are now visible on the Neighborhood 2D projection, and the students have a glimpse of how the decision space is divided by the model, as shown in Figure 6.4b.



(a) t-SNE projection of the generated neighborhood.

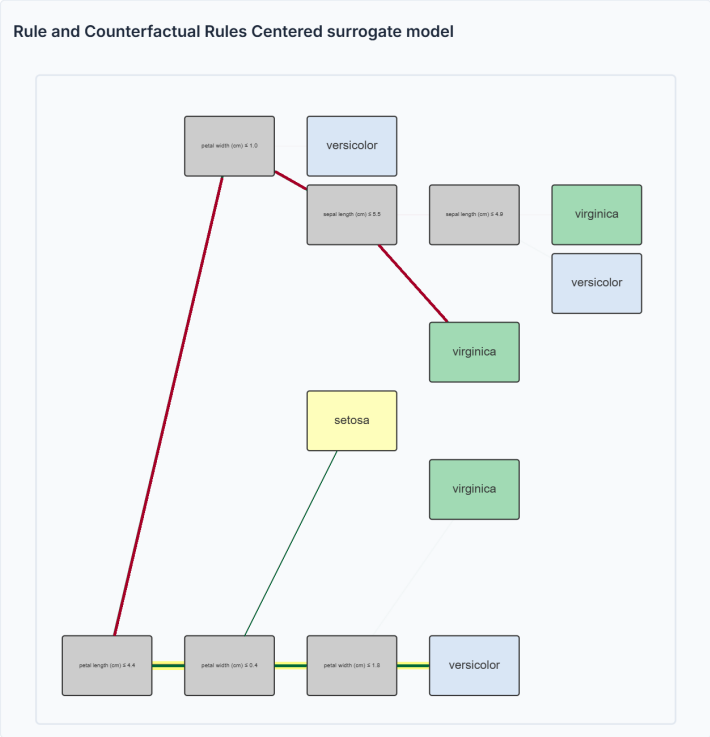
(b) PCA projection of the generated neighborhood, with decision boundaries displayed.

Figure 6.4: t-SNE and PCA projection of the generated neighborhood.

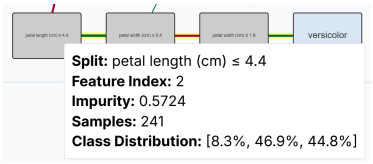
6.1.3 Navigating Decision Tree Structure

Having established spatial intuition about the neighborhood, the teacher shifts focus to the Rule and Counterfactual Rules Centered visualization displayed next to the scatter plot. The tree structure, shown in Figure 6.5a, presents rectangular nodes arranged in depth-aligned rows, with the explained instance path highlighted at the bottom through enhanced rectangular nodes connected by thick edges.

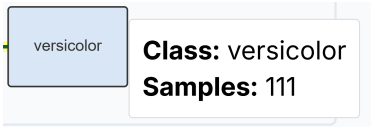
The shown surrogate model approximates how the Random Forest behaves in the neighborhood they just examined. They trace their cursor along the highlighted path from root to leaf: The instance starts at the root node, where the tree tests whether petal length is less than or equal to 4.4 centimeters. Since our instance



(a) Rule and Counterfactual Rules Centered surrogate model.



(b) Root node tooltip for the generated Rule and Counterfactual Rules Centered surrogate model.



(c) Predicted class hover tooltip.

Figure 6.5: Use case's Rule and Counterfactual Rules Centered surrogate model and details.

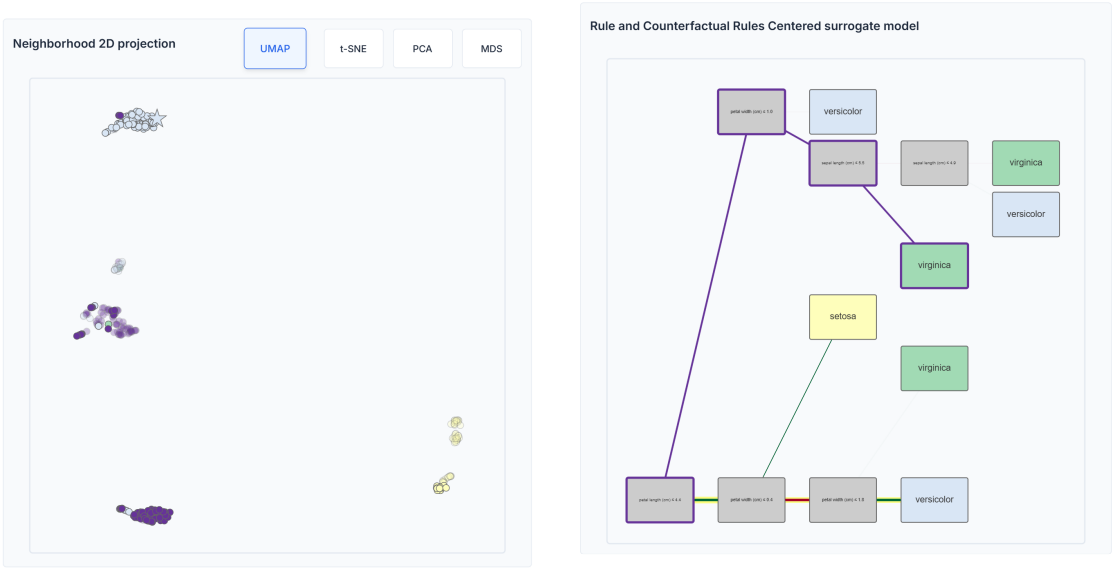
has a petal length of 4.35 cm, it takes the true branch. The teacher hovers over the root node, and students see the tooltip reveal detailed statistics as shown in Figure 6.5b: the split condition, impurity measure, sample count, and class distribution. Later the teacher also hovers the leaf node related to the prediction to show the number of samples relative to the predicted class, as shown in Figure 6.5c.

The teacher continues tracing the path and shows how, at the next level, the tree splits on the petal width feature twice. The chosen instance is lower than the threshold in the first instance, taking the true branch, and later it is higher, taking the false branch. The highlighted path provides the user with the factual rule, otherwise said, the conjunction of conditions that led to the versicolor prediction for this specific instance.

The teacher points to the virginica leaf nodes displayed throughout the tree above the highlighted path and talks about how the alternative paths represent counterfactual rules, otherwise said, different combinations of conditions that would lead to different predictions. For instance, if our flower had petal length less than 4.4 cm, petal width lower than 1 cm, sepal length lower than 5.5 it would have been classified as virginica regardless of other features. The rectangular nodes clearly

display the predicted class in each leaf, with color-coding matching the scatter plot for visual consistency.

The teacher clicks on the Counterfactual alternative leaf node, as shown in Figure 6.6. The system immediately highlights the corresponding points in the scatter plot above, mainly the cluster of green instances representing the virginica class at the bottom left of the scatter plot, and some points, originating from the dataset, near the decision boundary contested by virginica and versicolor instances. The teacher notes how this bidirectional coordination is crucial for building understanding, as they can see that these virginica instances occupy a specific region of the projected space, giving geometric intuition about the counterfactual scenario.



(a) Neighborhood 2D projection.

(b) Rule and Counterfactual Rules Centered surrogate model.

Figure 6.6: Click on counterfactual rule from the Rule and Counterfactual Rules Centered surrogate model visualization.

6.1.4 Exploring Interaction Patterns and Coordination

To deepen students' understanding of the coordination mechanism, the teacher demonstrates the reverse interaction flow. They click a purple point in the scatter plot's setosa cluster. The tree visualization immediately responds by highlighting the leftmost path from the root to the setosa leaf node. When any instance from the scatter plot is selected, the system automatically shows the decision path that the instance follows through the tree. This coordination helps in understanding the relationship between spatial position and logical conditions.

The teacher selects several points along the boundary between versicolor and virginica clusters, clicking each in succession. Students observe how the highlighted paths change, sometimes following similar routes but diverging at different split nodes. The teacher also highlights how instances near the decision boundary in the scatter plot correspond to paths that split at deeper levels in the tree. This showcases how the classifier needs more feature tests to distinguish between similar cases. The latter interactions are shown in Figure 6.7.

The teacher hovers over a split node in the middle of the tree, and the tooltip reveals the number of samples that pass through the node and its value of impurity. The impurity measures shows how mixed the classes are at each decision point. High impurity means the genetic algorithm generated instances from multiple classes that satisfy the conditions leading to this node, which is exactly what is expected near decision boundaries. The teacher also points out how the traditional output included a fidelity score indicating how well the surrogate matches the black-box, but provided no insight into where and why the approximation might be imperfect. These per-node statistics give much finer-grained understanding of local explanation quality.

6.1.5 Comparing Tree Layout Alternatives

Having established foundational understanding with the Rule and Counterfactual Rules Centered visualization, the teacher decides to demonstrate how different tree layouts can provide complementary insights. They change the visualization selection to Tree Layout, the visualization in Figure 6.8a appears.

The teacher explains that the traditional node-link tree layout uses circular nodes and emphasizes hierarchical relationships through vertical positioning. Students immediately notice the differences: nodes are positioned based on their depth and sibling relationships rather than aligned in horizontal rows, and the explained instance path uses only edge highlighting rather than also using the node positioning to draw attention.

The teacher clicks on a leaf node in the Tree Layout, observing the same bidirectional coordination with the scatter plot as the coordination mechanism remains consistent across different tree layouts. This consistency is important since the users of the interface can choose the layout that best fits the analytical goals without losing access to the same interaction capabilities.

Later, the teacher switches to the Rule Centered visualization, shown in Figure 6.8b. Students notice immediately that only the explained instance path appears as rectangular nodes, while the alternative branches are represented as collapsed circular nodes with special markings to allow the maintenance of awareness of alternative scenarios.

The teacher right-clicks on one of the collapsed subtree nodes, revealing a

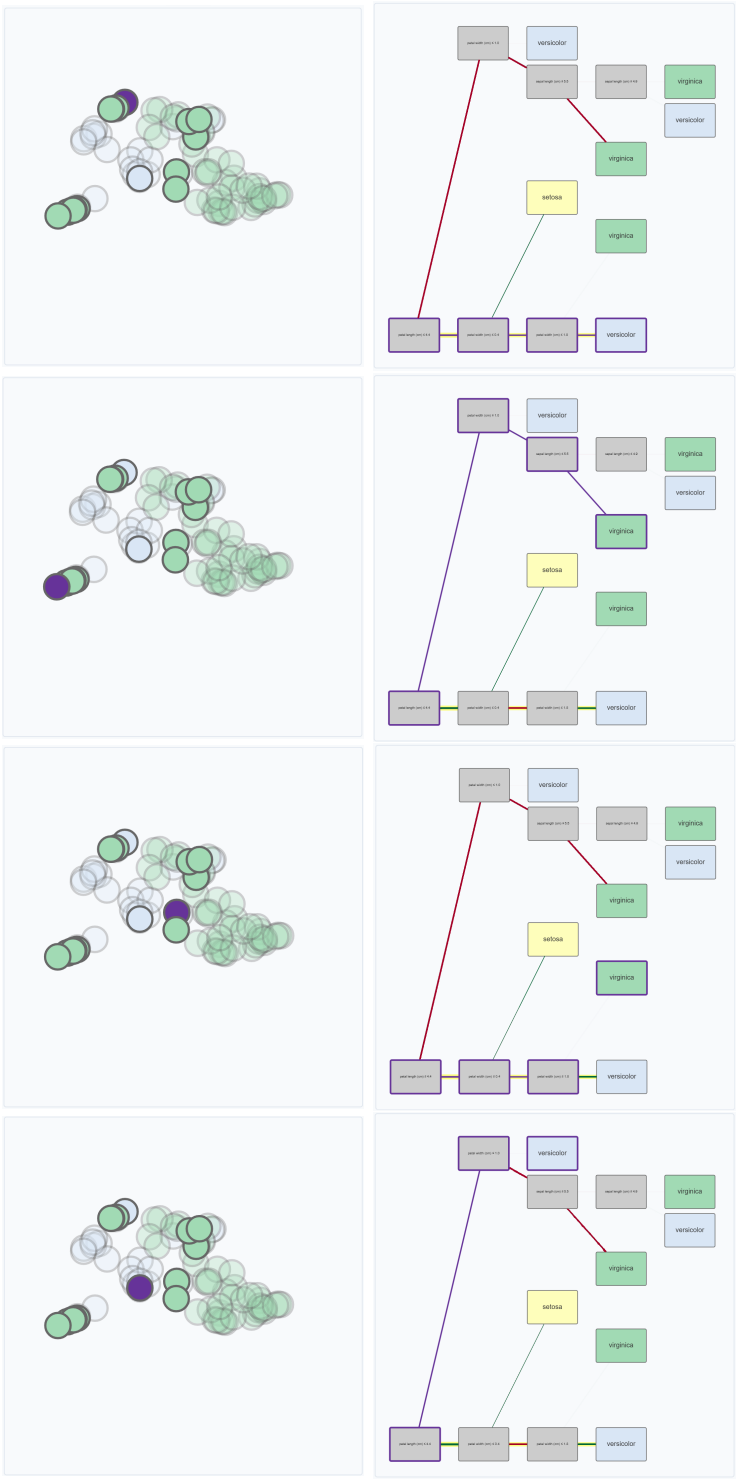
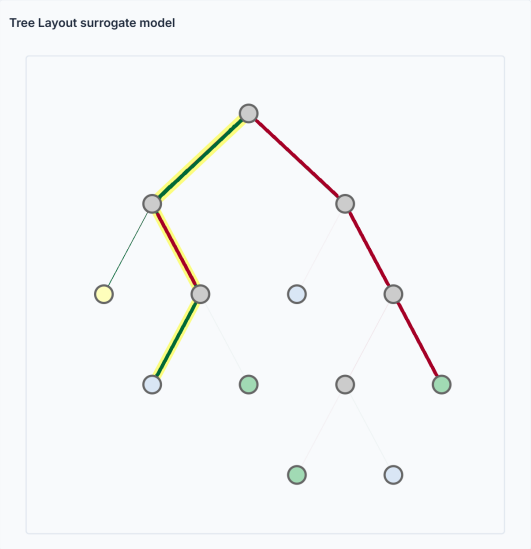
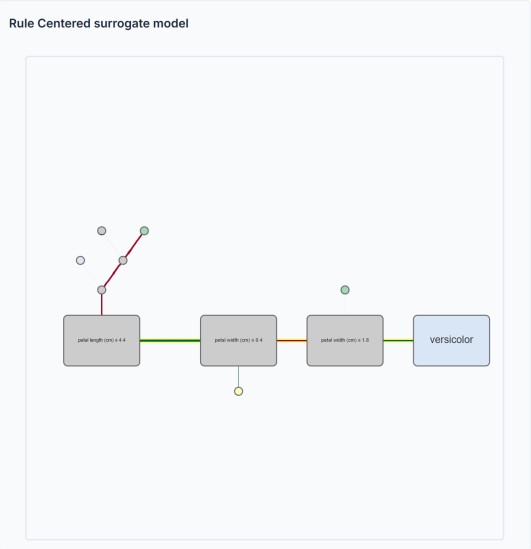


Figure 6.7: Different interactions at the selection of points near the decision boundary.

context menu with an `Expand Subtree` option. When they select it, the hidden branches unfurl, revealing the complete decision structure. The teacher explains how this progressive disclosure supports different depths of investigation. The teacher explains to the class that they can focus on the main explanation path and selectively explore alternatives only when needed, reducing visual complexity while maintaining access to complete information.



(a) Tree layout surrogate model for the same tree previously analyzed by the teacher and class.



(b) Rule centered surrogate model visualization for the same tree previously analyzed by the teacher and class.

Figure 6.8: Alternative visualizations shown in the teacher use case.

6.2 Use case: data scientist

A data scientist working on housing price prediction is trying to understand their trained classifier's decision patterns for specific predictions. They have developed a custom Random Forest model on the California Housing dataset and need to explain predictions for stakeholder communication. Since they were already working in the Jupyter Notebook environment, the data scientist opts to use the interface through that tool, which allows integration of their pre-trained classifier directly into the explanation workflow.

6.2.1 Dataset Preparation and Model Configuration

The data scientist begins by preparing the California Housing dataset for their analysis task. The original dataset contains median house values as continuous targets. They transform this regression problem into a five-class classification task by dividing the target into equal-frequency percentile bands. Each band represents twenty percent of the data distribution: percentile_0 (lowest prices, $15,000-$107,200), percentile_1 ($107,200-$157,300), percentile_2 ($157,300-$209,400), percentile_3 ($209,400-$290,000), and percentile_4 (highest prices, $290,000-$500,000).

This discretization choice serves multiple analytical purposes. It converts a complex regression problem into interpretable price categories that align with real estate market segments. The equal-frequency binning ensures balanced class representation across the dataset, avoiding issues with sparse high-value regions. The resulting five-class structure provides sufficient granularity to distinguish meaningful price tiers while maintaining interpretability.

The dataset features eight attributes characterizing housing blocks in California: median income (MedInc), house age (HouseAge), average rooms (AveRooms), average bedrooms (AveBedrms), population, average occupancy (AveOccup), latitude, and longitude. The data scientist recognizes that geographic coordinates (latitude and longitude) carry particular importance in housing price prediction, as location fundamentally determines property values in real estate markets.

They construct a preprocessing pipeline using scikit-learn's ColumnTransformer, applying StandardScaler to all numeric features. Without standardization, features with larger numeric ranges would disproportionately influence neighborhood generation.

For the classifier, the data scientist selects Random Forest and performs systematic hyperparameter optimization. They conduct grid search over 1,152 parameter combinations using ten percent of the data to identify optimal hyperparameters efficiently. The search evaluates n_estimators (50, 100, 200, 300), max_depth (10, 20, 30, None), min_samples_split (2, 5, 10), min_samples_leaf (1, 2, 4), max_features (sqrt, log2), and bootstrap options (True, False). Ten-fold cross-validation on the sample ensures robust parameter selection.

The optimization process identifies the following configuration: 200 trees, maximum depth 30, minimum samples split 5, minimum samples leaf 1, sqrt max features, and bootstrap enabled. The data scientist trains the final model on the complete dataset using these parameters with a 70-30 train-test split. Cross-validation on the full training set yields 0.8156 accuracy, while the held-out test set achieves 0.8098 accuracy. The Random Forest produces well-calibrated predictions across all five price percentile classes, with particularly strong performance in the extreme percentiles (percentile_0 and percentile_4) where feature patterns prove most distinctive.

6.2.2 Instance Selection and Interface Initialization

The data scientist selects an instance for explanation: a property with median income $830,140, house age 21 years, 6.24 average rooms, 0.97 average bedrooms, population 2,401, average occupancy 2.11, located at latitude 37.86 and longitude -122.22. The Random Forest predicts this instance as percentile_4, the highest price category. These coordinates place the property in the East Bay region of the San Francisco Bay Area, a location where high property values align with expectations given the elevated median income.

They initialize the Jupyter interface by creating a `TabularGeneticGeneratorLore` object, passing their trained classifier (wrapped as a black-box predictor) and the dataset descriptor. The data scientist calls the `interactive_explanation` method with the selected instance, setting `inJupyter=False` to launch the full browser-based interface rather than an embedded notebook view. This choice provides access to the complete visualization system with full interaction capabilities.

The interface opens in the browser, bypassing the dataset selection, classifier selection, and model training sections since the classifier and instance were already provided. The interface proceeds directly to the explanation parameter configuration, where the data scientist can adjust neighborhood size, dimensionality reduction settings, and visualization preferences.

6.2.3 Initial Explanation with Default Parameters

The data scientist begins their analysis with the default neighborhood size of 500 synthetic instances. They accept the default UMAP and PCA dimensionality reduction parameters. They initiate explanation generation by clicking the `Explain!` button.

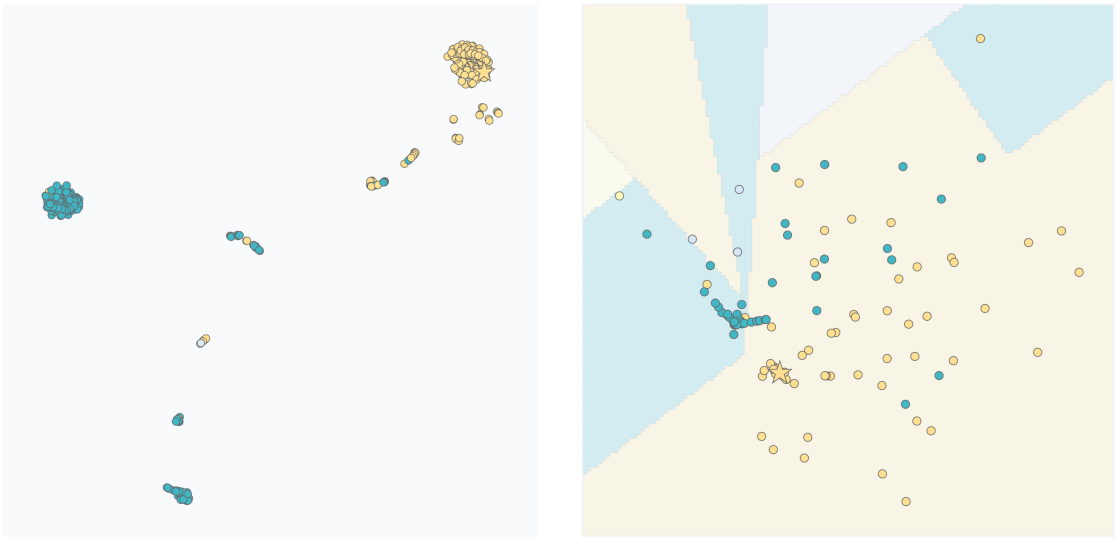
The system generates 500 synthetic neighbors around the target instance using the genetic algorithm, trains the surrogate decision tree on this neighborhood, and extracts both the factual and counterfactual rules. The visualization components render simultaneously: the 2D scatter plot showing the synthetic neighborhood projected via UMAP, shown in Figure 6.9a, and the tree visualization of the surrogate model, shown in Figure 6.10.

The data scientist examines the UMAP projection first. The visualization reveals distinct spatial clusters corresponding to different predicted price percentiles. A dense cluster of percentile_4 instances (#fee090) appears in the middle right quadrant, including the explained instance. Scattered percentile_3 instances (#41b6c4) form smaller groups, some intermixed with percentile_4 instances along cluster boundaries, others forming an isolated cluster in the bottom left. The spatial separation between percentile_4 and lower percentiles appears relatively

clear, though some percentile_3 instances lie close to the percentile_4 cluster boundary.

The data scientist notices that the synthetic neighborhood exhibits uneven class distribution. The majority of generated instances receive percentile_4 predictions, matching the explained instance's class. Fewer percentile_3 instances appear, and percentile_0, percentile_1, and percentile_2 instances are nearly absent from this particular neighborhood generation. This absence reflects the genetic algorithm's characteristic behavior: it focuses exploration on the local decision space around the explained instance. Since the explained instance belongs to percentile_4 with a confident prediction, the algorithm generates synthetic neighbors primarily within and immediately adjacent to this high-value region. Exploring distant feature value combinations corresponding to percentile_0, percentile_1, or percentile_2 would require traversing substantial feature space distances from the explained instance. The genetic algorithm's fitness function prioritizes generating instances that maintain proximity to the target instance while exploring local decision boundaries, resulting in concentrated sampling within the percentile_4 region and its immediate neighbors.

They switch to the PCA projection, shown in Figure 6.9b, to examine decision boundaries more directly. The PCA method enables Voronoi tessellation-based boundary visualization, which approximates the classifier's decision regions using a grid of test points.



(a) UMAP projection (b) PCA projection with decision boundaries

Figure 6.9: Spatial neighborhood visualizations for 500-instance synthetic neighborhood: UMAP projection showing cluster structure (a) and PCA projection with Voronoi tessellation-based decision boundary visualization (b)

The PCA projection displays decision boundaries rendered as colored background regions. The percentile_4 region (light #fee090) dominates the visualization space, occupying most of the bottom right and central portions. A percentile_3 region (light #41b6c4) appears in the middle left quadrant. The explained instance (star marker) sits within the percentile_4 region. The boundary visualization confirms the classifier's confident prediction for this instance. The nearest boundary lies several principal component units away, suggesting that substantial feature modifications would be required to alter the prediction.

6.2.4 Surrogate Model Analysis and Rule Extraction

The data scientist examines the surrogate model to understand the extracted rules. They select the Rule and Counterfactual Rules Centered tree visualization, which positions the explained instance's path at the bottom for easy identification.

Figure 6.10 displays the decision tree structure. The root node splits on MedInc. The explained instance follows the true branch (MedInc $>$ 7.2), leading directly to a leaf node predicting percentile_4. This factual rule is very simple: "If MedInc $>$ 7.2, then predict percentile_4." The single-condition rule indicates that median income above $720,000 serves as the primary determinant for high-value property classification in this local decision region.

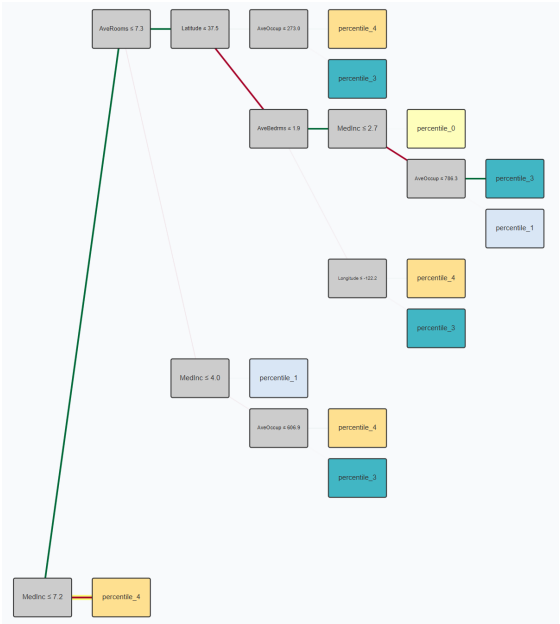


Figure 6.10: Rule and Counterfactual Rules Centered visualization of surrogate model trained on 500-instance neighborhood, showing simple tree structure with factual rule path (MedInc $>$ 7.2 $\rightarrow$ percentile_4) at bottom and counterfactual path to percentile_3

The simplicity of this factual rule provides immediate insight into the classifier's behavior. Properties with median household incomes exceeding $720,000 receive confident percentile_4 predictions regardless of other feature values. This threshold aligns with real estate market understanding: neighborhoods with such high median incomes typically command premium property prices. However, the data scientist recognizes that this single-split rule, while interpretable, may be oversimplifying the decision boundary. The rule captures the dominant pattern but potentially obscures secondary factors that influence predictions for edge cases.

The surrogate model also reveals counterfactual rules indicating alternative prediction outcomes. The meaningful counterfactual rule extracted by the explainability method follows the false branch from the root: MedInc $\leq 7.2 \rightarrow$ AveRooms $\leq 7.3 \rightarrow$ Latitude $> 37.5 \rightarrow$ AveBedrms $\leq 1.9 \rightarrow$ MedInc $> 2.7 \rightarrow$ AveOccup $\leq 786.3 \rightarrow$ percentile_3. This counterfactual rule specifies the conditions under which the classifier predicts percentile_3 rather than percentile_4. The rule reveals a hierarchical decision structure: first, median income must fall below $720,000; then, geographic location (latitude above 37.5°, corresponding to Bay Area regions) combined with moderate occupancy patterns and income above $270,000 leads to percentile_3 predictions.

The counterfactual rule provides actionable insight. To shift a prediction from percentile_4 to percentile_3, the primary change involves reducing median income below the $720,000 threshold. However, the subsequent conditions indicate that even with lower income, properties maintain relatively high value (percentile_3 rather than lower percentiles) when located in premium geographic areas (Latitude > 37.5) with reasonable occupancy characteristics. This pattern reflects the real estate principle that location preserves value even when income indicators decline.

The data scientist observes that only one meaningful counterfactual rule emerges from the 500-instance neighborhood. The tree structure shows relatively few alternative paths, with most instances following either the direct percentile_4 path or the single percentile_3 counterfactual path. This limited counterfactual diversity suggests that the 500-instance neighborhood may not fully characterize the decision space, particularly for predictions substantially different from percentile_4.

At this stage, the data scientist recognizes the value of the visual interface compared to the textual rule representations described in Section 3.1. The tree visualization immediately reveals the hierarchical rule structure through spatial arrangement. Following the explained instance's path from root to leaf requires simple visual scanning rather than parsing nested logical conditions. The counterfactual rule, while complex with six conditions, becomes more easily comprehensible through the visual path representation.

6.2.5 Recognizing Need for Expanded Analysis

As previously said, while the data scientist explored the tree structure and examined the individual neighborhood instances, they recognize a potential limitation in the current analysis. They reason that increasing the neighborhood size should generate a more comprehensive sample of the local decision space, potentially uncovering additional counterfactual patterns and providing more robust rule extraction. A larger neighborhood may reveal how the classifier handles edge cases near decision boundaries and may produce additional counterfactual rules or refine the existing rules with more detailed conditions.

The data scientist decides to regenerate the explanation with a substantially larger neighborhood. They select 3,000 instances as the new neighborhood size, a number that should provide adequate representation of alternative decision paths while remaining computationally tractable.

6.2.6 Regenerating Explanation with Expanded Neighborhood

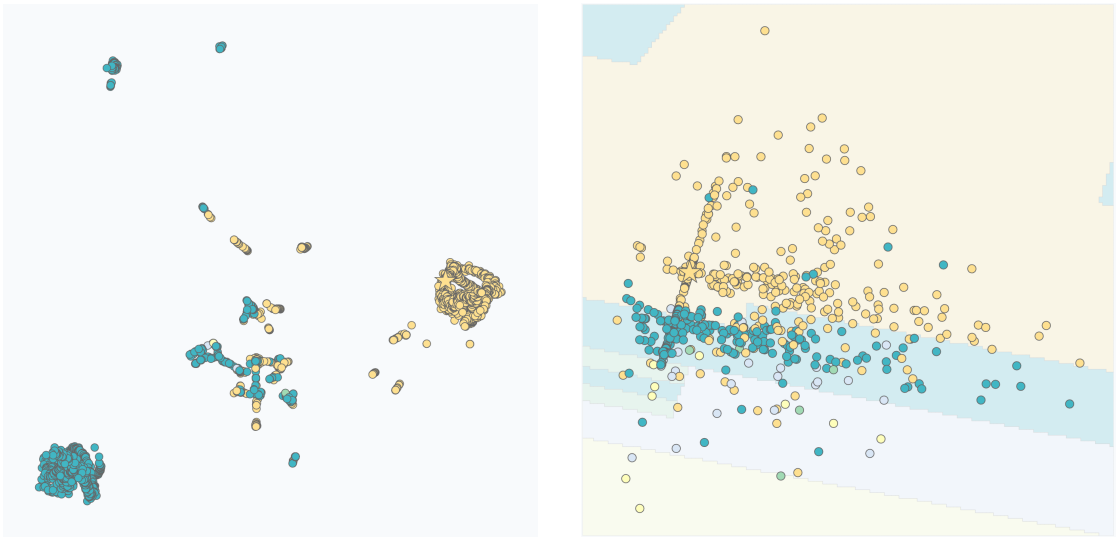
The data scientist returns to the parameter configuration interface and modifies the neighborhood size field from 500 to 3000. They retain all other settings to enable direct comparison with the previous results. They click the `Explain!` button to initiate the new explanation generation. The computational time increases proportionally with the neighborhood size, as the genetic algorithm must generate and evaluate six times as many synthetic instances, and the decision tree training operates on a larger dataset.

When the visualizations appear, the data scientist immediately observes notable differences from the 500-instance case. The UMAP projection, shown in Figure 6.11a, displays substantially increased instance density across all regions of the plot, especially regarding the decision boundary between percentile_3 and percentile_4.

The expanded neighborhood reveals a more nuanced class distribution. While percentile_4 instances still dominate (as expected given the explained instance's confident classification), the data scientist now observes substantially more percentile_3 instances interspersed throughout the visualization. The cluster arrangement remains similar to the 500-instance case, but the increased density provides finer-grained characterization of the transition zones. In fact, the new extracted rule is more complex but the root split is constant with the previous explanation. The new extracted rule is the following: MedInc $> 7.4 \rightarrow$ Latitude $\leq 41.5 \rightarrow$ MedInc $\leq 12.6 \rightarrow$ Population $\leq 18771.9 \rightarrow \text{percentile}_4$.

The increased neighborhood size provides better characterization of the transition zones between classes. The data scientist notices gradual transitions from percentile_4 to percentile_3 regions, with intermediate instances forming visible

gradients in the UMAP projection. This continuous representation helps them understand how the classifier's confidence degrades as feature values shift away from the explained instance's configuration.



(a) UMAP projection (b) PCA projection with decision boundaries

Figure 6.11: Spatial neighborhood visualizations for 3000-instance synthetic neighborhood: UMAP projection showing increased density and clearer transition zones (a) and PCA projection revealing presence of all five price percentile classes with band-like decision regions (b)

The data scientist switches to the PCA projection to examine how the expanded neighborhood affects decision boundary visualization. Figure 6.11b shows the result.

The PCA boundary visualization with 3,000 instances displays decision region geometry similar in overall structure to the 500-instance case. The percentile_4 region still occupies the majority of the space, but the data scientist now observes a distinct percentile_3 region. The class boundaries appear to split the decision space into bands where each class occupies its own region, especially evident for the percentile_3, percentile_1, and percentile_0 classes which form roughly parallel strips.

More significantly, the visualization now reveals clearer regions corresponding to percentile_2, percentile_1, and percentile_0 predictions. These regions, especially the percentile_2 region, occupy small spaces. Their presence confirms that the expanded neighborhood successfully generated synthetic instances exploring more distant regions of the feature space, uncovering decision patterns that were not relevant in the smaller neighborhood's focused local exploration.

The data scientist examines the surrogate tree trained on the 3,000-instance

neighborhood, shown in Figure 6.12. The tree exhibits increased depth and structural complexity compared to the 500-instance tree. The root split still occurs on MedInc, but the threshold shifts slightly to 7.4 (from 7.2 in the smaller neighborhood). The factual rule path now extends through multiple splits, incorporating secondary conditions on Latitude, a second MedInc constraint establishing an upper bound, and Population. This refined rule structure provides more detailed characterization of the decision logic while maintaining consistency with the simpler 500-instance rule in its fundamental pattern.

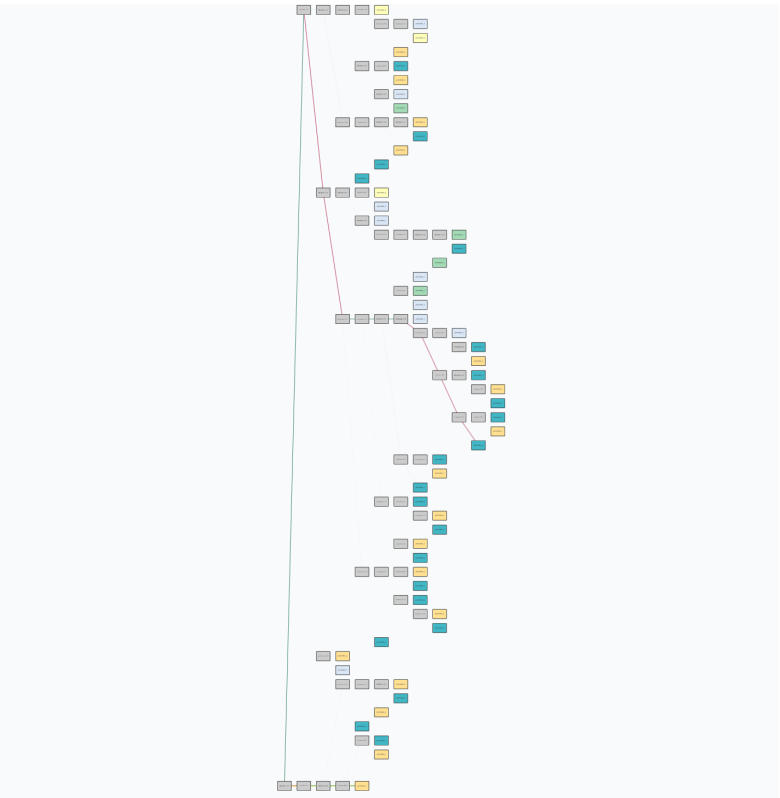
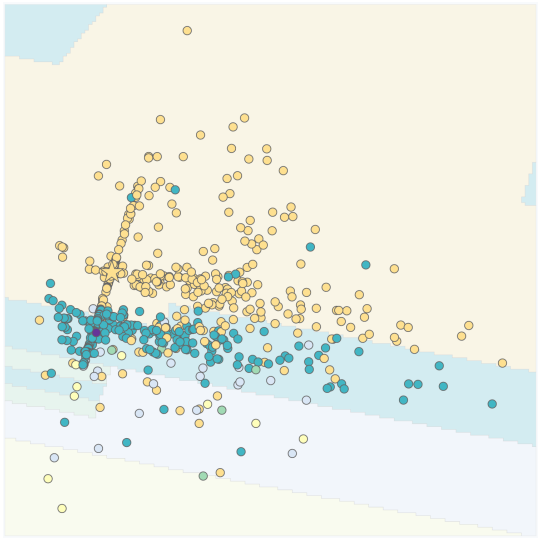


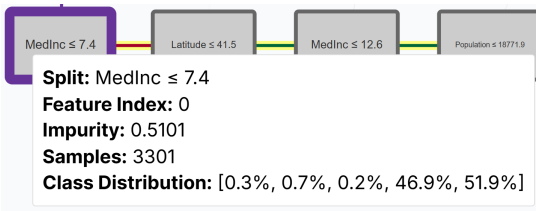
Figure 6.12: Rule and Counterfactual Rules Centered visualization of surrogate model trained on 3000-instance neighborhood, showing increased tree depth and refined factual rule with dual MedInc conditions bounding the typical high-value income range

6.2.7 Detailed Interaction-Driven Exploration

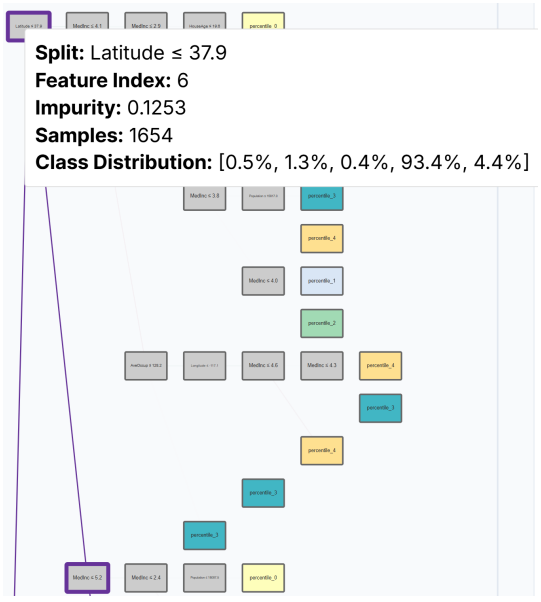
The data scientist makes use of the coordinated visualizations to explore the relationship between neighborhood instances and their decision paths. They begin by selecting a percentile_3 instance near the class boundary in the PCA scatter plot (Figure 6.11b).



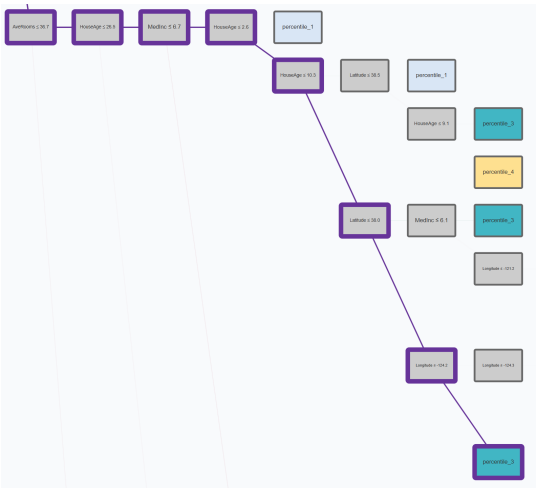
(a) PCA scatter plot with highlighted percentile_3 instance



(b) Tree visualization top section showing root split



(c) Tree visualization middle section with highlighted path



(d) Tree visualization bottom section showing path to and percentile_3 leaf node

Figure 6.13: Coordinated visualization interaction: clicking a percentile_3 instance in the PCA scatter plot (a) triggers path highlighting across the full tree visualization (b-d), demonstrating bidirectional coordination between spatial and symbolic representations with detailed view of split conditions

Figure 6.13 captures this coordinated highlighting. The selected point in the scatter plot appears in the highlight color, and emphasis is put on the tree path from root to the corresponding leaf. This immediate visual feedback confirms the logical conditions that led to this specific instance's classification.

The data scientist hovers over the percentile_3 leaf node in the tree to examine detailed statistics. The tooltip reveals that around 1400 instances from the 3,000-instance neighborhood follow this path. This substantial sample count indicates a robust, stable pattern rather than an isolated edge case.

The data scientist systematically explores instances across different regions of the scatter plots. They click on a percentile_0 instance located in the peripheral region of the PCA projection, interaction shown in Figure 6.11b. The tree highlighting reveals a long path through multiple splits, ultimately reaching a small leaf node with only a few instances. This scarce population contrasts sharply with the number of instances for the percentile_3 counterfactual rule leaf and the heavily populated percentile_4 rule leaf. The data scientist recognizes that rules based on such small sample counts carry lower confidence—they may represent genuine but rare decision patterns.

They switch between the three tree visualization layouts (Tree Layout, Rule and Counterfactual Rules Centered, Rule Centered) to examine the same highlighted path from different visual perspectives. The Tree Layout provides an overview of the entire decision structure. The Rule and Counterfactual Rules Centered view positions the explained instance's path at the bottom, facilitating comparison between the factual rule and alternative paths. The Rule Centered view collapses irrelevant subtrees, focusing attention on the paths most relevant to understanding the explained instance. Each layout offers distinct advantages for different analytical tasks.

Throughout this interactive exploration, the data scientist builds comprehensive understanding by iteratively forming hypotheses, testing them through instance selection, and observing the corresponding tree paths and tooltip information. The bidirectional coordination proves essential: scatter plot interactions reveal rule structures, while tree interactions reveal instance distributions.

6.2.8 Validating Generality: Middle-Class Instance Analysis

After completing the analysis of the percentile_4 instance, the data scientist reflects on whether the instance selection influenced the genetic algorithm's exploration behavior. The initial instance belonged to percentile_4, representing properties at the upper boundary of the dataset's price distribution. The data scientist hypothesizes that this extreme position might have constrained the genetic algorithm's ability to explore diverse regions of the feature space, biasing the observed patterns and consequential understanding of the model toward only high-value property

characteristics.

To validate whether the explanation patterns generalize across different regions of the decision space, they select a new instance from percentile_2, the middle class of the five-category distribution. This instance represents properties with moderate prices ($157,300-$209,400), occupying the central region of the price distribution rather than an extreme. The selected percentile_2 instance has median income $338,410, house age 29 years, 4.8 average rooms, 1.002 average bedrooms, population 1919, average occupancy 2.7, located at latitude 37.69 and longitude -121.76 (East Bay area, near Livermore).

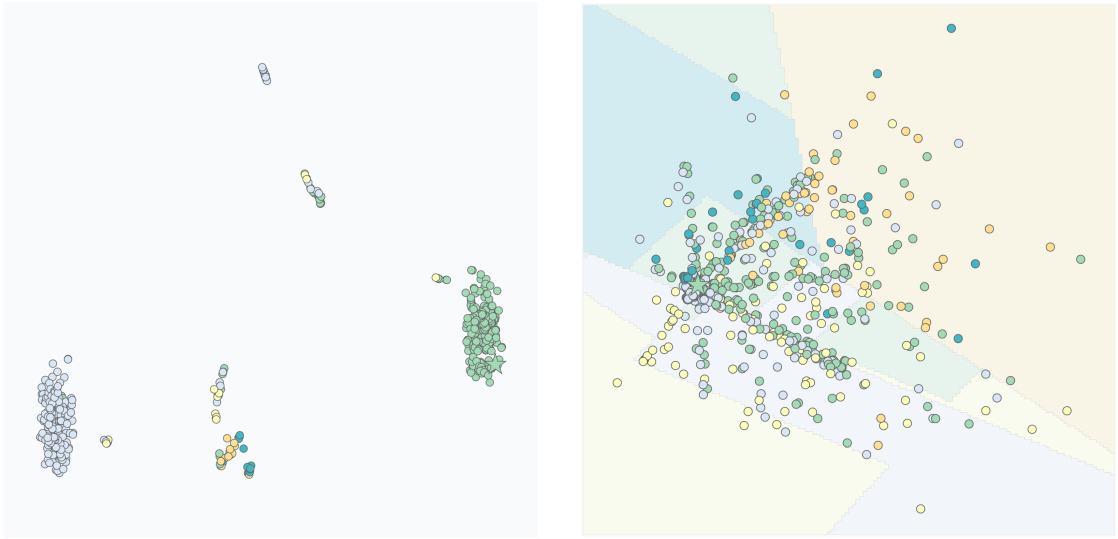
The data scientist generates explanations for this percentile_2 instance using both 500 and 3,000 synthetic neighbors to examine whether the neighborhood size effects observed for the percentile_4 instance hold for instances in different regions of the decision space.

The analysis reveals patterns remarkably consistent with the percentile_4 case. The UMAP projection for the 500-instance neighborhood, shown in Figure 6.14a, displays a clusters of percentile_2 (#a1dab4) instances, surrounding the explained instance, and another cluster of percentile_1 (#D9E6F5) instances. Other percentiles appear throughout the projection, and overall the spatial distribution patterns are similar to those observed in the percentile_4 analysis. The genetic algorithm again focuses exploration on the local decision space around the explained instance, generating synthetic neighbors primarily within the predicted class region and its immediate boundaries.

The surrogate tree structure maintains the same fundamental pattern observed previously: one clear factual rule path leading to the predicted class (percentile_2), and one meaningful counterfactual rule. The counterfactual rule in this case leads to percentile_1 rather than percentile_3, reflecting the downward rather than upward class transition relevant for a middle-range instance.

When the data scientist regenerates the explanation with 3,000 synthetic instances, the patterns again parallel the previous analysis. The PCA projection with decision boundaries, shown in Figure 6.14b, reveals increased instance density and clearer boundary visualization. The decision space appears slightly more mixed compared to the percentile_4 case. This reflects the genuine decision landscape for middle-range properties, which naturally have adjacent class boundaries on both sides rather than the more isolated position of extreme-value properties.

The expanded neighborhood yields a refined factual rule with additional conditions, maintaining the same refinement pattern observed in the percentile_4 analysis. This validation analysis confirms that the explanation methodology produces consistent, stable results across different regions of the decision space. The slightly more mixed decision space for the percentile_2 instance reflects genuine classifier behavior rather than algorithm limitation, accurately characterizing the



(a) UMAP projection of 500-instance synthetic neighborhood showing cluster structures similar to percentile_4 analysis with dominant predicted class and counterfactual rule cluster, plus scattered adjacent class instances

(b) PCA projection with decision boundaries for 3000-instance neighborhood showing slightly more mixed decision space with adjacent class regions visible on multiple sides of the predicted class region

Figure 6.14: Comparison of UMAP and PCA projections for percentile_2 instance neighborhoods at different scales

more complex boundary structure surrounding middle-range properties.

6.2.9 Insights and Methodological Implications

Through this analysis, the data scientist derives several key insights about the Random Forest classifier's behavior on the California Housing task. They confirm that geographic location (Latitude in particular) serves as a secondary decision factor, with the approximate boundary at 37.5 degrees latitude corresponding to the transition from Bay Area premium pricing to lower-cost regions. They identify MedInc as the most critical feature, with thresholds around $730,000 separating high-value properties from lower-value ones. The dual MedInc conditions in the refined 3000 samples rule further reveal that typical high-value properties occupy a specific income range ($740,000-$1,260,000) rather than simply exceeding a lower threshold.

The data scientist recognizes that neighborhood size critically affects explanation detail and confidence. The 500-instance default proves adequate for understanding the explained instance's immediate classification and captures the primary counterfactual scenario (percentile_3). The simpler rule extracted from the 500-instance neighborhood provides clear, interpretable logic that stakeholders can readily under-

stand. The 3,000-instance neighborhood provides substantially more comprehensive coverage, revealing additional counterfactual rules for all five price percentiles and enabling extraction of refined rules with multiple conditions that capture secondary decision factors. The more complex rules from the larger neighborhood offer deeper insight but require more careful interpretation.

This experience informs their approach to future model explanation tasks. They establish a workflow principle: begin with default parameters for rapid initial exploration, but systematically increase neighborhood size when dealing with multi-class problems or when initial explanations appear incomplete. They also recognize the value of comparing multiple neighborhood sizes to validate explanation consistency and identify potential artifacts of insufficient sampling.

The interactive visualization proves essential for this analytical workflow. The ability to rapidly switch between different dimensionality reduction methods (UMAP for cluster structure, PCA for boundaries), explore tree structures at multiple levels of detail through three alternative layouts, and confirm patterns through coordinated highlighting provides flexibility that enables thorough analysis. The coordinated interaction between scatter plots and tree visualizations allows the data scientist to continuously validate rules against actual instance distributions, building confidence in the extracted explanations.

Chapter 7

Conclusions

This thesis addresses a critical gap in explainable artificial intelligence by developing an interactive visualization framework that transforms how users understand and validate eXplainability methods outputs. While existing XAI visual analytics tools tend to address either global model understanding or local instance explanations independently, this work introduces a coordinated framework that makes the relationship between synthetically generated neighborhoods and resulting rule-based explanations explicitly visible and explorable. The research demonstrates that effective explanation communication requires more than algorithmic sophistication—it demands carefully designed interactive visual systems that bridge computational outputs with human understanding. The primary contribution lies in the development of the dual-representation strategy that integrates the spatial and the symbolic views through coordinated multiple visualizations. The framework implements scatter plot projections using multiple dimensionality reduction techniques to reveal the spatial distribution of synthetic instances, on top of that the users can choose between three alternative decision tree layouts based on their cognitive preference, and sophisticated bidirectional coordination mechanisms that maintain visual consistency via the synchronized highlighting. This approach addresses the limitations in current XAI presentation formats, particularly the lack of spatial context, inability to validate decision boundary correspondence, and absence of mechanisms for exploring alternative decision paths. By allowing users to transition fluidly between spatial intuition and logical reasoning, the system transforms explanation consumption from passive acceptance into active analytical investigation.

Two innovative decision tree visualization paradigms advance the state of the art specifically for explanation scenarios. The Rule and Counterfactual Rules Centered visualization employs a depth-aligned layout with rectangular nodes organized in vertical columns, creating stable visual references that allow for direct comparison of decision criteria across branches at equivalent depths. The Rule

Centered visualization implements an adaptive expansion-based positioning that dynamically adjusts spacing based on subtree complexity, with interactive subtree collapse/expand functionality enabling progressive disclosure of surrogate model complexity. These visualizations depart from traditional hierarchical tree layouts by prioritizing explanation clarity over comprehensive structural display, while sophisticated interaction patterns support explanation validation through visual confirmation.

Through the surrounding interface users can observe how variations in neighborhood size, diversity, and quality affect explanation characteristics, while toggles for different dimensionality reduction techniques with customizable parameters support methodological experimentation. The iterative development process reveals important design patterns including the evolution from grid-based to Voronoi tessellation for decision boundary representation and the refinement of color schemes to CIELAB color space projection for perceptual uniformity. These methodological insights validate that effective XAI visualization requires addressing information fragmentation through coordinated views, cognitive overload through visual connection mechanisms, and confirmation difficulties through interactive exploration.

The documented use cases demonstrate the system's effectiveness across diverse scenarios. The pedagogical application using the Iris dataset shows how coordinated visualizations support conceptual introduction to XAI methods, with progressive disclosure of functionality supporting scaffolded learning where each interaction builds upon previously established understanding. The availability of three alternative tree layouts enables teachers to demonstrate how different visual representations of identical logical structures reveal complementary insights. On the other end, the professional workflow analyzing California Housing dataset predictions illustrates iterative refinement of explanations through parameter adjustment, with the simulated data scientist, beginning with the default neighborhood size, recognizes limitations, regenerating with expanded neighborhood sizes, and conducting a detailed exploration of refined rules. The coordinated visualizations enable validation of explanation quality through cross-referencing between spatial and symbolic representations, building confidence in the surrogate model's fidelity.

Dynamic color schemes scale from 2 to 10+ classes while maintaining colorblindness friendliness, support for multiple dimensionality reduction techniques provides methodological flexibility, and Voronoi tessellation with user-settable granularity represents a sophisticated approach to visualizing decision surfaces in reduced-dimensional projections. These technical achievements create a robust foundation that validates key design principles: information integration reduces cognitive load through coordinated views, interactive confirmation supports explanation trust through bidirectional coordination mechanisms, and multiple representation formats support diverse user mental models through alternative tree visualizations.

Even though the system represents a solid starting point for this new development in visual analytics for eXplainable AI, some additional features, nuanced and borrowed from the surveyed systems, could be implemented to further improve its capabilities and address identified limitations. These potential enhancements span diverse aspects of the system's design, from visual encoding strategies to interaction paradigms, each offering opportunities to deepen the analytical power of the framework while maintaining its core principle of bidirectional spatial-symbolic coordination.

One possible direction involves enriching the **visual encoding of decision tree edges through multi-colored branch encoding**, following the pattern established by BaobabView [42]. Currently, edges employ variable widths to encode sample flow with uniform colors indicating split outcomes, but this approach limits the immediate visibility of class distribution information within branches, which is still encapsulated in the ending node tooltip. By displaying color-coded bands proportional to class distributions within each branch, where each band's width represents the percentage of instances belonging to a specific class, users could immediately assess class composition without requiring tooltip interactions. However, this enhancement introduces trade-offs between information density and visual simplicity, requiring careful visual hierarchy management to ensure that critical elements like the explained instance path and leaf node predictions maintain appropriate visual prominence despite increased edge representation complexity.

Beyond edge encoding refinements, an **alternative spatial organization** through horizontal partition chart layouts could reshape how users perceive distributional importance. Current node-link representations allocate space proportionally to tree structure rather than data importance, treating branches containing thousands of instances identically to branches containing only a few edge cases. A horizontal partition layout, such as the one shown in Figure 7.1, would represent the root node as a horizontal bar spanning full width, with each decision split partitioning the parent bar into vertically stacked child segments proportional to instance flow percentages. Tree depth would progress horizontally from left to right, making path importance immediately visible through bar height: paths containing 60% of neighborhood instances would occupy 60% of the vertical space, while paths with only 2% would appear as narrow slivers. However, this alternative layout introduces significant limitations alongside its distributional clarity benefits: path highlighting becomes fundamentally challenging as individual instances contribute fractionally to bar segment populations rather than corresponding to discrete visual elements, and counterfactual path comparison would face similar challenges when paths appear as vertically stacked segments rather than horizontally aligned rectangular nodes. The reduced hierarchical clarity represents another trade-off, as the cognitive load for understanding tree topology would likely increase compared

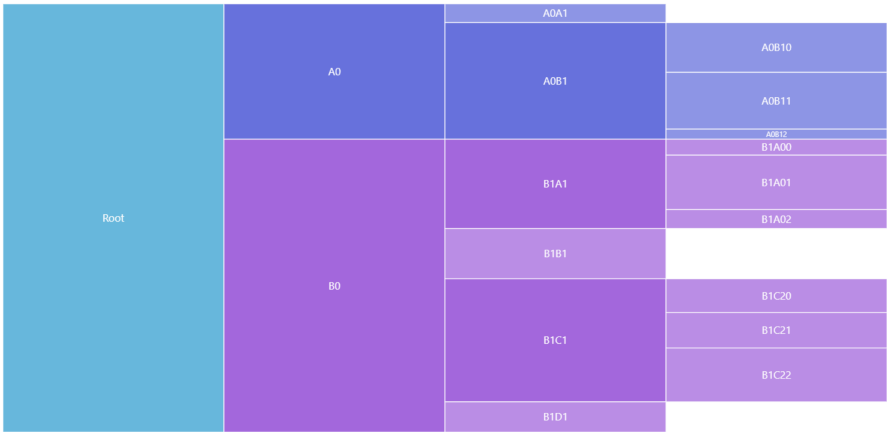


Figure 7.1: Concept of the alternative tree layout proposed, extracted from [97]

to explicit node-link connections. This enhancement might work most effectively as an initial overview tool where users examine overall neighborhood distribution patterns before switching to node-link layouts for detailed path examination and counterfactual comparison.

Moving beyond static visual representations, adaptive interaction mechanisms could provide more nuanced control over information complexity. Currently, the system requires complete layout switching to **transition between Rule Centered's focused representation and Rule and Counterfactual Rules Centered's** comparison capabilities. A **hybrid visualization** mode would enable interactive transformation within the Rule Centered layout, allowing users to progressively convert individual counterfactual paths from circular representations to rectangular depth-aligned representations on demand. Users could right-click specific counterfactual leaves of interest, perhaps after examining the scatter plot and noticing a cluster of instances with different class predictions, and select "Convert to rectangular path," progressively building a comparison view customized to their specific analytical interests rather than being presented with all possible counterfactuals simultaneously. This progressive transformation approach would support natural analytical workflows where users start with focused examination and gradually expand scope based on discovered patterns, managing cognitive load by enabling selective counterfactual exploration. Complementing this progressive disclosure approach, **threshold-based leaf filtering** would introduce a user-adjustable slider control enabling dynamic hiding of leaves below a specified sample percentage threshold. As users adjust the slider, leaves containing fewer instances than the threshold would disappear, creating "stub" branches with visual indicators communicating that some children are hidden.

The current system's exclusive reliance on class-based coloring, while effective for supporting the primary analytical goal of verifying that spatial proximity correlates

with classification consistency, limits exploration of alternative organizational patterns that may exist within synthetic neighborhoods. **Clustering-based alternative coloring** would implement a toggle enabling users to switch between class-based and clustering-based coloring throughout the interface, with colors assigned based on clustering algorithms such as k-means, DBSCAN, or hierarchical clustering applied to neighborhoods in either original feature space or projected space. This enhancement would address multiple analytical use cases: cluster-class correspondence verification would reveal whether surrogate decision boundaries align with intrinsic neighborhood structure, with strong correspondence suggesting high-quality neighborhood generation and substantial misalignment indicating that neighborhoods span multiple decision regions or that the surrogate model introduces artificial complexity. Within-class pattern discovery would enable the identification of subgroups among instances receiving identical predictions, revealing that predictions may actually encompass multiple distinct feature configurations that cluster separately despite receiving the same class prediction. Projected space artifact detection would reveal whether dimensionality reduction creates artificial groupings by comparing clustering results between the original high-dimensional feature space and the two-dimensional projected space, helping users assess projection quality and avoid drawing incorrect conclusions based on projection artifacts. The ability to toggle between class-based and cluster-based coloring while simultaneously switching between UMAP, PCA, t-SNE, and MDS projections would create a rich exploration space for investigating how different analytical perspectives and dimensionality reduction techniques interact.

Finally, bridging the gap between interactive visual exploration and traditional communication formats, **textual rule export functionality** would address situations where users need to communicate explanation analysis results through channels that do not support interactive visualizations—written reports, academic manuscripts, or presentation slides. Right-click context menu operations on leaf nodes would generate formatted textual representations of complete decision paths in multiple output formats: natural language format for non-technical audiences, Python-style conditional format for code integration, formal logic notation for academic manuscripts, and LaTeXtable row format for direct insertion into tabular presentations.

Future development prioritization would benefit from empirical evaluation comparing enhancement alternatives through controlled user studies examining whether multi-colored edges improve decision boundary assessment accuracy, task completion time measurements comparing hybrid visualization mode against full layout switching workflows, or A/B testing of clustering-based coloring to determine whether it reveals actionable insights or introduces analytical confusion. Such evaluations would provide evidence-based guidance for implementation pri-

oritization, ensuring that development effort focuses on enhancements delivering measurable user experience improvements. The iterative design process that shaped the current thesis system, progressing from initial prototypes through comparative evaluation and use case validation to the final implemented designs, establishes a methodological template for enhancement evaluation that future work could employ.

Bibliography

[1] Gulsum Alicioglu and Bo Sun. "A survey of visual analytics for Explainable Artificial Intelligence methods". In: *Computers & Graphics* 102 (2022), pp. 502–520. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2021.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0097849321001886.

[2] Minsuk Kahng et al. "ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 88–97. DOI: 10.1109/TVCG.2017.2744718.

[3] A. Chatzimparmpas et al. "The State of the Art in Enhancing Trust in Machine Learning Models with the Use of Visualizations". In: *Computer Graphics Forum* 39.3 (June 2020), pp. 713–756. ISSN: 1467-8659. DOI: 10.1111/cgf.14034. URL: http://dx.doi.org/10.1111/cgf.14034.

[4] Christina B. Azodi, Jiliang Tang, and Shin-Han Shiu. "Opening the Black Box: Interpretable Machine Learning for Geneticists". In: *Trends in Genetics* 36.6 (2020), pp. 442–455. ISSN: 0168-9525. DOI: https://doi.org/10.1016/j.tig.2020.03.005. URL: https://www.sciencedirect.com/science/article/pii/S016895252030069X.

[5] Evren Dağlarli. "Explainable Artificial Intelligence (xAI) Approaches and Deep Meta-Learning Models". In: *Advances and Applications in Deep Learning*. Ed. by Marco Antonio Aceves-Fernandez. Rijeka: IntechOpen, 2020. Chap. 5. DOI: 10.5772/intechopen.92172. URL: https://doi.org/10.5772/intechopen.92172.

[6] Mengchen Liu et al. "Towards Better Analysis of Deep Convolutional Neural Networks". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 91–100. DOI: 10.1109/TVCG.2016.2598831.

[7] David Gunning and David W. Aha. "DARPA's Explainable Artificial Intelligence (XAI) Program". In: *AI Mag.* 40 (2019), pp. 44–58. URL: https://api.semanticscholar.org/CorpusID:67773377.

[8] Or Biran and Courtenay V. Cotton. *Explanation and Justification in Machine Learning : A Survey Or*. 2017. URL: https://api.semanticscholar.org/CorpusID:3911355.

[9] Frank Emmert-Streib, Olli Yli-Harja, and Matthias Dehmer. "Explainable artificial intelligence and machine learning: A reality rooted perspective". In: *WIREs Data Mining and Knowledge Discovery* 10.6 (June 2020). ISSN: 1942-4795. DOI: 10.1002/widm.1368. URL: http://dx.doi.org/10.1002/widm.1368.

[10] Amina Adadi and Mohammed Berrada. "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052.

[11] Arun Das and Paul Rad. *Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey*. 2020. arXiv: 2006.11371 [cs.CV]. URL: https://arxiv.org/abs/2006.11371.

[12] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: high-precision model-agnostic explanations". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*. 2016. arXiv: 1602.04938 [cs.LG]. URL: https://arxiv.org/abs/1602.04938.

[14] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. arXiv: 1705.07874 [cs.AI]. URL: https://arxiv.org/abs/1705.07874.

[15] Sebastian Bach et al. "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLoS ONE* 10 (2015). URL: https://api.semanticscholar.org/CorpusID:9327892.

[16] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[17] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034 [cs.CV]. URL: https://arxiv.org/abs/1312.6034.

[18] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703.01365 [cs.LG]. URL: https://arxiv.org/abs/1703.01365.

[19] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. *Learning Important Features Through Propagating Activation Differences*. 2019. arXiv: 1704.02685 [cs.CV]. URL: https://arxiv.org/abs/1704.02685.

[20] Benjamin Letham et al. "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model". In: *The Annals of Applied Statistics* 9.3 (Sept. 2015). ISSN: 1932-6157. DOI: 10.1214/15-aoas848. URL: http://dx.doi.org/10.1214/15-AOAS848.

[21] Sarah Tan et al. "Distill-and-Compare: Auditing Black-Box Models Using Transparent Model Distillation". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '18. New Orleans, LA, USA: Association for Computing Machinery, 2018, pp. 303–310. ISBN: 9781450360128. DOI: 10.1145/3278721.3278725. URL: https://doi.org/10.1145/3278721.3278725.

[22] Rich Caruana et al. "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '15. Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1721–1730. ISBN: 9781450336642. DOI: 10.1145/2783258.2788613. URL: https://doi.org/10.1145/2783258.2788613.

[23] Leo Breiman. *Manual on Setting Up, Using, and Understanding Random Forests v3.1*. Tech. rep. 4(1):29. University of California, Berkeley, 2002. URL: https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf.

[24] Bolei Zhou et al. "Learning Deep Features for Discriminative Localization". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2921–2929. DOI: 10.1109/CVPR.2016.319.

[25] Francesco Bodria et al. *Benchmarking and Survey of Explanation Methods for Black Box Models*. 2021. arXiv: 2102.13076 [cs.AI]. URL: https://arxiv.org/abs/2102.13076.

[26] Riccardo Guidotti et al. "Stable and actionable explanations of black-box models through factual and counterfactual rules". In: *Data Min. Knowl. Discov.* 38.5 (Nov. 2022), pp. 2825–2862. ISSN: 1384-5810. DOI: 10.1007/s10618-022-00878-5. URL: https://doi.org/10.1007/s10618-022-00878-5.

[27] Scott Lundberg. *SHAP online documentation*. Observable. 2022. URL: https://shap.readthedocs.io/en/latest/api_examples.html#plots.

[28] Yang Yang et al. "Dimensionality reduction by UMAP reinforces sample heterogeneity analysis in bulk transcriptomic data". In: *Cell Reports* 36.4 (2021), p. 109442. ISSN: 2211-1247. DOI: https://doi.org/10.1016/j.celrep.2021.109442. URL: https://www.sciencedirect.com/science/article/pii/S2211124721008597.

[29] Etienne Becht et al. "Dimensionality reduction for visualizing single-cell data using UMAP". In: *Nature Biotechnology* 37 (2018), pp. 38–44. URL: https://api.semanticscholar.org/CorpusID:54473203.

[30] Yingfan Wang et al. *Understanding How Dimension Reduction Tools Work: An Empirical Approach to Deciphering t-SNE, UMAP, TriMAP, and PaCMAP for Data Visualization*. 2021. arXiv: 2012.04456 [cs.LG]. URL: https://arxiv.org/abs/2012.04456.

[31] Martin Sewell. *Principal Component Analysis*. Tech. rep. University College London, Jan. 2008. URL: http://www.stats.org.uk/pca/pca.pdf.

[32] Felipe L. Gewers et al. "Principal Component Analysis: A Natural Approach to Data Exploration". In: *ACM Computing Surveys* 54.4 (May 2021), pp. 1–34. ISSN: 1557-7341. DOI: 10.1145/3447755. URL: http://dx.doi.org/10.1145/3447755.

[33] Warren S. Torgerson. "Multidimensional scaling: I. Theory and method". In: *Psychometrika* 17 (1952), pp. 401–419. URL: https://api.semanticscholar.org/CorpusID:120849755.

[34] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426.

[35] Laurens van der Maaten and Geoffrey E. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: https://api.semanticscholar.org/CorpusID:5855042.

[36] David I. Spivak. *METRIC REALIZATION OF FUZZY SIMPLICIAL SETS*. 2009. URL: https://api.semanticscholar.org/CorpusID:35883069.

[37] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML]. URL: https://arxiv.org/abs/1802.03426.

[38] Zijie J. Wang et al. *TimberTrek: Exploring and Curating Sparse Decision Trees with Interactive Visualization*. Oct. 2022. DOI: 10.1109/vis54862.2022.00021. URL: http://dx.doi.org/10.1109/VIS54862.2022.00021.

[39] Jimmy Lin et al. *Generalized and Scalable Optimal Sparse Decision Trees*. 2022. arXiv: 2006.08690 [cs.LG]. URL: https://arxiv.org/abs/2006.08690.

[40] Berk Ustun and Cynthia Rudin. *Learning Optimized Risk Scores*. 2019. arXiv: 1610.00168 [stat.ML]. URL: https://arxiv.org/abs/1610.00168.

[41] Hans-Jörg Schulz. "Treevis.net: A Tree Visualization Reference". In: *IEEE Computer Graphics and Applications* 31 (2011), pp. 11–15. URL: https://api.semanticscholar.org/CorpusID:8225504.

[42] Stef van den Elzen and Jarke J. van Wijk. "BaobabView: Interactive construction and analysis of decision trees". In: *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*. 2011, pp. 151–160. DOI: 10.1109/VAST.2011.6102453.

[43] Jakub Mrva et al. *Decision Support in Medical Data Using 3D Decision Tree Visualisation*. 2019. URL: https://api.semanticscholar.org/CorpusID:210970801.

[44] Dora Szücs and Florian Schmidt. *Decision Tree Visualization for High-Dimensional Numerical Data*. Oct. 2018. DOI: 10.1109/SNAMS.2018.8554961.

[45] Boris Kovalerchuk Andrew Dunn, Alex Worland, and Sridevi Wagle. *Interactive Decision Tree Creation and Enhancement with Complete Visualization for Explainable Modeling*. 2023. arXiv: 2305.18432 [cs.LG]. URL: https://arxiv.org/abs/2305.18432.

[46] T. Parr and P. Grover. *How to visualize decision trees*. explained.ai. 2019. URL: https://explained.ai/decision-tree-viz/.

[47] Dirk Streeb et al. "Task-Based Visual Interactive Modeling: Decision Trees and Rule-Based Classifiers". In: *IEEE Transactions on Visualization and Computer Graphics* 28 (2021), pp. 3307–3323. URL: https://api.semanticscholar.org/CorpusID:231604534.

[48] Yonghong Xu et al. "Parallel Filter: A Visual Classifier Based on Parallel Coordinates and Multivariate Data Analysis". In: *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*. Ed. by De-Shuang Huang, Laurent Heutte, and Marco Loog. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1172–1183. ISBN: 978-3-540-74205-0.

[49] Soon Tee Teoh and Kwan-Liu Ma. "PaintingClass: interactive construction, visualization and exploration of decision trees". In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. Washington, D.C.: Association for Computing Machinery, 2003, pp. 667–672. ISBN: 1581137370. DOI: 10.1145/956750.956837. URL: https://doi.org/10.1145/956750.956837.

[50] Soon Tee Teoh and Kwan-Liu Ma. "StarClass: Interactive Visual Classification using Star Coordinates". In: *SDM*. 2003. URL: https://api.semanticscholar.org/CorpusID:15241816.

[51] Yao Ming, Huamin Qu, and Enrico Bertini. *RuleMatrix: Visualizing and Understanding Classifiers with Rules*. 2018. arXiv: 1807.06228 [cs.LG]. URL: https://arxiv.org/abs/1807.06228.

[52] J. Stasko and E. Zhang. "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations". In: *IEEE Symposium on Information Visualization 2000. INFOVIS 2000. Proceedings*. 2000, pp. 57–65. DOI: 10.1109/INFVIS.2000.885091.

[53] Stuart Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision To Think*. Academic Press, Jan. 1999. ISBN: 978-1-55860-533-6.

[54] P. Płoński. *How to visualize Decision Tree in Python (scikit-learn)*. MLJAR Blog. 2021. URL: https://mljar.com/blog/visualize-decision-tree/.

[55] joesquito. *Decision Tables*. Observable. 2024. URL: https://observablehq.com/@joesquito/decision-table.

[56] Danyel Fisher and Miriah Meyer. "Multiple and Coordinated Views". In: *Making Data Visual: A Practical Guide to Using Visualization for Insight*. O'Reilly Media, 2012. Chap. 6. URL: https://www.oreilly.com/library/view/making-data-visual/9781491960493/.

[57] Usama Fayyad, Georges G. Grinstein, and Andreas Wierse. *Information visualization in data mining and knowledge discovery*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. ISBN: 1558606890.

[58] Thilo Spinner et al. "explAIner: A Visual Analytics Framework for Interactive and Explainable Machine Learning". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 1064–1074. DOI: 10.1109/TVCG.2019.2934629.

[59] Kostiantyn Kucher, Elmira Zohrevandi, and Carl A. L. Westin. "Towards Visual Analytics for Explainable AI in Industrial Applications". In: *Analytics* 4.1 (2025). ISSN: 2813-2203. DOI: 10.3390/analytics4010007. URL: https://www.mdpi.com/2813-2203/4/1/7.

[60] Marc-André Zöller et al. "XAutoML: A Visual Analytics Tool for Understanding and Validating Automated Machine Learning". In: *ACM Transactions on Interactive Intelligent Systems* 13.4 (Dec. 2023), pp. 1–39. ISSN: 2160-6463. DOI: 10.1145/3625240. URL: http://dx.doi.org/10.1145/3625240.

[61] Jun Yuan et al. "SUBPLEX: A Visual Analytics Approach to Understand Local Model Explanations at the Subpopulation Level". In: *IEEE Computer Graphics and Applications* 42.6 (2022), pp. 24–36. DOI: 10.1109/MCG.2022.3199727.

[62] Eleonora Cappuccio et al. *Fiper: a Visual-based Explanation Combining Rules and Feature Importance*. 2024. arXiv: 2404.16903 [cs.HC]. URL: https://arxiv.org/abs/2404.16903.

[63] Angelos Chatzimparmpas et al. "DeforestVis: Behaviour Analysis of Machine Learning Models with Surrogate Decision Stumps". In: *Computer Graphics Forum* 43 (2023). URL: https://api.semanticscholar.org/CorpusID:257912686.

[64] Riccardo Guidotti et al. "Factual and Counterfactual Explanations for Black Box Decision Making". In: *IEEE Intelligent Systems* 34 (2019), pp. 14–23. URL: https://api.semanticscholar.org/CorpusID:210931542.

[65] Yoav Freund and Robert E Schapire. "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. URL: https://www.sciencedirect.com/science/article/pii/S002200009791504X.

[66] Félix-Antoine Fortin et al. "DEAP: evolutionary algorithms made easy". In: *J. Mach. Learn. Res.* 13 (2012), pp. 2171–2175. URL: https://api.semanticscholar.org/CorpusID:15629107.

[67] Wei-Yin Loh. "Classification and Regression Trees". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1 (Jan. 2011), pp. 14–23. DOI: 10.1002/widm.8.

[68] Tan et al. *Introduction to Data Mining*. Addison Wesley, May 2005.

[69] L. Breiman et al. "Classification and Regression Trees". In: *Biometrics* 40 (1984), p. 874. URL: https://api.semanticscholar.org/CorpusID:29458883.

[70] Kacper Sokol and Peter Flach. "Desiderata for interpretability: explaining decision tree predictions with counterfactuals". In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA: AAAI Press, 2019. ISBN: 978-1-57735-809-1. DOI: 10.1609/aaai.v33i01.330110035. URL: https://doi.org/10.1609/aaai.v33i01.330110035.

[71] Chenglin Fan and P. Li. "Classification Acceleration via Merging Decision Trees". In: *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference* (2020). URL: https://api.semanticscholar.org/CorpusID:224805052.

[72] Alessandro Carella. *Git commit: Scatter plot first version.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/31aebe09426c54603ce64c833cce69895eb9bac9.

[73] Alessandro Carella. *Git commit: Tooltip and click on point functionality for scatter plot.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/75d6b0071d124a3980afed0c6e8d59b69caf3f07.

[74] Alessandro Carella. *Git commit: Implemented boundary paths instead of grid for the split of the projected boundaries space using ramer douglas peucker for path approximation.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/e89162e58471838a0536dc9e113aacf8a3371b24.

[75] Urs Ramer. "An iterative procedure for the polygonal approximation of plane curves". In: *Computer Graphics and Image Processing* 1.3 (1972), pp. 244–256. ISSN: 0146-664X. DOI: https://doi.org/10.1016/S0146-664X(72)80017-0. URL: https://www.sciencedirect.com/science/article/pii/S0146664X72800170.

[76] DAVID H DOUGLAS and THOMAS K PEUCKER. "ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE". In: *Cartographica* 10.2 (1973), pp. 112–122. DOI: 10.3138/FM57-6770-U75U-7727. eprint: https://doi.org/10.3138/FM57-6770-U75U-7727. URL: https://doi.org/10.3138/FM57-6770-U75U-7727.

[77] Alessandro Carella. *Git commit: Voronoi tessellation for scatter plot decision boundaries.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/2e491d4facc812e34fb9fbdd61d7e2116918d671.

[78] Wojciech Pokojski and Paulina Pokojska. "Voronoi diagrams – inventor, method, applications". In: *Polish Cartographical Review* 50.3 (2018), pp. 141–150. DOI: 10.2478/pcr-2018-0009. URL: https://doi.org/10.2478/pcr-2018-0009.

[79] Alessandro Carella. *Git commit: Added t-SNE as an option in the scatter plot.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/0b40ca0b97c33ca23834e633cfafc5e505ecf380.

[80] Alessandro Carella. *Git commit: Added UMAP and MDS as an option in the scatter plot.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/e0bf5907e567ed3bf24d4e97a7f79dd7a348eeb5.

[81] Alessandro Carella. *Git commit: Added switch for different dimension reducing techniques in the scatter plot.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/12aec6a64db3d208a633c36b8e01933cb7abf7e4.

[82] Alessandro Carella. *Git commit: Split node highlight.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/23f078473cf832c9fa7069e045d43c3f64d3a914.

[83] Alessandro Carella. *Git commit: Dynamic colors for classes in RGB space.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/79ddcd8630729f96ed8e109ce78232fb7ec5b7c5.

[84] Alessandro Carella. *Git commit: Dynamic font for the text in the rectangles.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/4151678797efc48ddf7086ed9833d621076b6242.

[85] Alessandro Carella. *Git commit: Added option to see the dataset points in the scatter plot.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/af1f79008ca8ea8d38e8903be520e6f452311939.

[86] Alessandro Carella. *Git commit: Early protoype of decision tree visualization.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/50ecf96dafc8c9a1955ada30135a54dfa84eaa65.

[87] E.M. Reingold and J.S. Tilford. "Tidier Drawings of Trees". In: *IEEE Transactions on Software Engineering* SE-7.2 (1981), pp. 223–228. DOI: 10.1109/TSE.1981.234519.

[88] Alessandro Carella. *Git commit: New colors palette.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/f1fa71cd70a6e4497f0aa353ba3632395bd75b4d.

[89] Alessandro Carella. *Git commit: Constant highlight for the explained instance in the decision tree plot.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/0469c76105d9d975959984df9cdbdd83dd2d6008.

[90] Alessandro Carella. *Git commit: Improved tooltip with sklearn-like description.* Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/bafb21589f70fca670b7aa8c63c76ed23989ed59.

[91] Alessandro Carella. *Git commit: First implementation of the tree spawn visualization*. Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/f8e8d0e9ea249aa4035f1330535e98a70c817988.

[92] Alessandro Carella. *Git commit: Implemented the expansion of the subtrees in the spawn tree visualization*. Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/202c78cce6144ac3e39bbd5b1f73927bb7e9c452.

[93] Alessandro Carella. *Git commit: Improved spacing in the tree spawn visualization*. Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/12ef4628f0a478a4e4fe3a2574b9253b24787d5d.

[94] Alessandro Carella. *Git commit: Integrated spawn tree visualization in the webapp*. Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/d7b0d4e30e3809be348d2a4b04a3f87d8b8142aa.

[95] Alessandro Carella. *Git commit: First implementation of the blocks tree visualization*. Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/5e711a416bf16f381cedbed764a5a6fb7ad70269.

[96] Alessandro Carella. *Git commit: Implementation of the blocks tree visualization using rectangles for the nodes*. Observable. 2025. URL: https://github.com/AlessandroCarella/Master-thesis/tree/da7896cfd46ad0f250bf1b54c4f010c45b454676.

[97] amcharts. *Horizontal Partition Chart - amCharts*. Observable. 2025. URL: https://www.amcharts.com/demos/horizontal-partition-chart/.