

# Evaluation of Embedded AI Through Model Difference Analysis

András Gergely Deé-Lukács<sup>✉</sup>, András Földvári<sup>✉</sup>, András Pataricza<sup>✉</sup>

Budapest University of Technology and Economics

Department of Artificial Intelligence and Systems Engineering

Budapest, Hungary

Email: {andrasgergely.dee-lukacs, andras.foldvari}@edu.bme.hu  
pataricza.andras@vik.bme.hu

**Abstract**—The growing reliance on embedded AI components in critical systems demands robust mechanisms for explainability and reliability. These systems often integrate highly complex, opaque models whose decision-making processes are difficult to interpret, posing significant challenges to debugging and trustworthiness. This paper introduces an approach that allows examining regions identified through model comparisons, specifically focusing on areas where interpretable surrogate models and opaque models diverge or produce inconsistencies. By analyzing these regions, the paper provides actionable insights for identifying edge cases and mitigating risks associated with model inaccuracies.

This paper leverages qualitative abstraction techniques to translate complex model behavior into comprehensible representations, enabling systematic evaluation of discrepancies. By focusing on the intersection of model behavior and system-level impact, the proposed methodologies offer a scalable approach for enhancing both the dependability and interpretability of AI-enabled systems. The findings advance the state of explainable AI and contribute to the development of safer, more transparent applications in critical domains.

**Index Terms**—explainable AI, qualitative reasoning, qualitative model extraction

## I. INTRODUCTION

In the rapidly advancing field of artificial intelligence (AI), integrating explainability into complex systems has emerged as a critical requirement. Embedded AI components are increasingly utilized in systems where their outputs significantly impact decision-making processes, posing challenges such as difficulty in debugging, limited trust due to lack of transparency, and the risk of performance issues in critical scenarios. However, the opacity of these opaque models poses challenges for debugging, trust-building, and performance evaluation. To address this, surrogate models and qualitative reasoning techniques have gained prominence, offering interpretable approximations of complex AI systems.

This paper focuses on analyzing regions identified through model comparison, specifically examining areas where interpretable surrogate models and complex, opaque models converge, diverge, or exhibit inconsistencies better to understand their significance for system-level reliability and improvement. The research explores how these insights can inform system

evaluation, debugging, and improvement by examining these regions—where models agree, disagree, or fail. The primary goal is understanding how these regions can be leveraged to enhance system performance, identify edge cases, and mitigate risks. By leveraging qualitative abstraction and surrogate modeling, this paper introduces a framework for evaluating model differences and understanding model behavior in a systematic and interpretable manner. The methods highlighted in this paper provide actionable strategies for addressing discrepancies.

## II. QUALITATIVE ABSTRACTION

Qualitative Modeling [1] (QM) represents and reasons about continuous aspects (quantities, motion, space, time) in a symbolic, human-like way. Unlike differential equations demanding precise numerics, QM abstracts information to intuitively understand changes without specific numeric knowledge. QM aims to enable symbolic reasoning about continuous aspects of systems, bridging human intuition and formal mathematical models. A key objective is clustering continuous values into discrete categories, making reasoning more intuitive and interpretable.

### A. Clustering Continuous Values

Let  $X \subseteq \mathbb{R}$  be the domain of a continuous feature,  $Q$  the *qualitative domain* of the clustered feature so that  $Q = \{q\}$ ,  $Q \subset \mathbb{Z}$ , and  $I = I_1, I_2, \dots, I_n \subset \mathbb{R}$  real intervals [2]. These intervals represent *partitions* of the feature domain:

$$\forall I_i, I_j : I_i \cap I_j = \emptyset; \bigcup_i I_i = X \quad (1)$$

These partitions are selected in a way that the values they contain represent a *similar behavior* in the system. This means these elements only differ in aspects that are not as relevant. Relevance is defined by the task at hand to be resolved using the qualitative model. There's an *interval membership function* that simply assigns each number to an interval.

$$mb : X \mapsto I : mb(x) = I_i \iff x \in I_i \quad (2)$$

The *cf* clustering function assigns a qualitative value to an element, the value corresponding to a one-on-one mapping with an interval partition of the feature domain.

This research was founded by DigitalTech EDIH DIGITAL-2021-EDIH-01 and DOSS HORIZON-CL3-2022-CS-01 projects.

$$\begin{aligned}
x \in X : cf(x) \in Q \wedge cf(x_1) = cf(x_2) \\
\iff \exists I_i : mb(x_1) = mb(x_2)
\end{aligned} \tag{3}$$

### III. EXPLAINABLE AI

Machine learning models are increasingly complex, leading to opaque decision-making. The need for explainability arises from legal "right to explanation" [3], ethical concerns, trust-building, and model debugging. These factors drive *Explainable AI* (xAI) and *Explanatory Model Analysis* [4], aiming to create verifiable AI systems [5]. Exploratory Model Analysis systematically examines a model's structure, outputs, and behavior. Explanation methods include *interpretable-by-design* models (e.g., decision trees [6], linear/logistic regression, rule-based methods [7], [8]), *model-agnostic* approaches (e.g., LIME [9], SHAP [10]), and *model-specific* techniques (e.g., feature importance in random forests).

#### A. Surrogate Models

A surrogate model is an abstract interpretable-by-design xAI model that approximates the behavior of an advanced, opaque model and describes it using a ruleset. Surrogate modeling is similar to a formal method called *abstract interpretation* [11] that approximates a program's behavior by mapping its concrete states to abstract domains.

1) *Decision Tree*: The decision tree [6] (DT) is one of the simplest and most popular ML algorithms created in the early days. Although it is simple, it is surprisingly accurate and very versatile. It can be ideally used to solve both classification and regression problems. A decision tree (DT) recursively splits a dataset until a stopping condition (e.g., depth limit or a subset with only one label) is met, creating a tree whose vertices represent split conditions, edges represent true/false branches, and leaves represent predictions.

A *node*  $v \in V_{tree}$  represents a point in the tree where a decision is made. Each node is associated with a *feature*  $feature(v)$ , which is an attribute of the dataset used to make the split, and a *value*  $val(v)$ , which determines the threshold for splitting. A *split condition* at node  $v$  is  $splitcondition(v) := feature(v) < val(v)$ . Its children  $v_{true}, v_{false}$  split samples based on whether  $feature(v)$  is less than  $val(v)$ .

A *path condition* of node  $v$  is the conjunction of edge conditions (split condition on true edge, negated otherwise) from the root to  $v$ . A *ruleset* is the disjunction of path conditions of leaves that predict a positive label:

$$ruleset(T) = \bigvee_{\substack{v_i \in V_{tree} \\ deg(v_i)=1}} pathcondition(v_i). \tag{4}$$

2) *Boolean Rules via Column Generation*: The Boolean Rules via Column Generation (BRCG) algorithm [7], implemented in the AIX360 toolkit [12], efficiently generates CNF or DNF rules from a data set by framing Boolean decision rule learning as a linear program. It minimizes the number of false positive and false negative classifications.

Given a training dataset  $D = \{(x, y)^n\}$ , that can be partitioned into  $P \cup Z$ , where  $P$  and  $Z$  contains the indices of the positive and negative samples. Let  $K$  denote the collection of all possible clauses to be used in the ruleset, and  $K_i$  denote the clauses satisfied by sample  $i$ .  $\xi_i \in \{0, 1\}$  denotes if sample  $i$  was misclassified,  $w_k$  denotes whether clause  $k$  was used in the ruleset, and  $c_k$  denotes the complexity of clause  $k$ .  $C$  is a parameter that bounds the complexity of the ruleset to prevent overfitting. The *master integer program* (MIP) minimizes:

$$z_{MIP} = \min \sum_{i \in P} \xi_i + \sum_{i \in Z} \sum_{k \in K_i} w_k \tag{5}$$

subject to constraints ensuring positive sample coverage, bounded complexity, and binary clause usage.

The objective function (5) has two components. The first component it tries to minimize is the number of misclassified positive samples, i.e., the false negative samples. The second component to be minimized represents the number of clauses that satisfy negative samples, in other words, the number of clauses that can lead to false positive predictions.

Since solving the MIP directly is computationally infeasible for real-world datasets, a *column generation* (CG) approach is used. Initially, a *restricted master linear* problem (RMLP), a linear programming (LP) relaxation of the MIP, is solved with a small subset of simple clauses. A *pricing problem* then iteratively identifies and adds clauses with the most *negative reduced cost* to improve the RMLP objective. This process repeats until no clause with a negative reduced cost remains, yielding the optimal ruleset.

#### B. Interpretable Model Differencing

To compare the approximate model with the embedded model, a solution is needed that can reveal the differences between the two models in an understandable and illustrative manner. This is particularly important because, in the case of complex machine learning models, differences often remain hidden, making it challenging to determine where and why their decisions diverge. To address this, the Interpretable Model Differencing (IMD) algorithm [13] enables model comparison by constructing a *Joint Surrogate Tree* (JST), which simultaneously represents both models, visually identifies agreements and disagreements, and presents contradictions in the form of rules.

The process begins by training two models  $M_1, M_2 : X_{data} \mapsto Y_{data}$  with the same dataset. Their outputs are used to label the data as  $Y_{M_1}, Y_{M_2}$ . An IMD instance, a binary classifier  $imd : X_{data} \times Y_{M_1} \times Y_{M_2} \mapsto Y_{imd}$ , is trained to predict whether the models contradict each other for a given sample based on input features and model predictions.

A JST is a decision tree with standard decision nodes and special *OR-nodes*, where the models diverge in their behavior. Decision nodes can be *common* (shared split conditions) or *model-specific*. Subtrees of OR-nodes no longer share conditions. Leaves in the JST hold model predictions, which may be *pure* (single label) or *impure* (mixed labels, resolved by

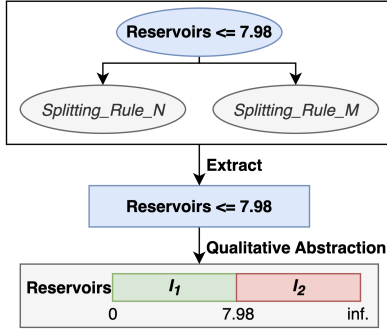


Fig. 1. Example Qualitative Model Extraction from Tree Structures

majority voting). The JST is built recursively. At each node, the algorithm chooses between creating a common node or an OR-node based on impurity measures. An OR-node is created if the following condition fulfills:

$$\text{impurity}(X_{data}, Y_{M_1}) + \text{impurity}(X_{data}, Y_{M_2}) \leq \alpha \cdot \text{impurity}(X_{data}, Y_{M_1}, Y_{M_2}) \quad (6)$$

where  $\alpha \leq 1$  controls the trade-off.

A difference rule can be extracted from a JST by selecting an OR-node, two leaves from its two subtrees with different labels, and conjunction the path conditions of these leaves.

$$\begin{aligned} \text{diffrule} = & \{ \text{pathcondition}(\text{leaf}_1) \wedge \\ & \text{pathcondition}(\text{leaf}_2) : \\ & \text{leaf}_i \in \text{leaves}(v_{OR}), i = 1, 2 \\ & \text{label}(\text{leaf}_1) \neq \text{label}(\text{leaf}_2) \} \end{aligned} \quad (7)$$

### C. Ruleset to Qualitative Model

The core of xAI modeling lies in combining simple rules that accurately approximate the dataset. Our research builds on the observation that widely used models rely on combinations of simple inequalities to partition behavioral domains using logical expressions. Transforming these inequalities into qualitative variables serves as a general principle for deriving qualitative counterpart models. More specifically, the explanations of these models can be represented as a tree graph, where each node corresponds to a decision rule expressed as a split condition (an inequality). These inequalities can be directly converted into a qualitative model.

Each value in these inequalities in a ruleset represents a *qualitative landmark*. This means the continuous domains of the features can be clustered by partitioning them into intervals using interval logic [2] along the split values. After that, each partition is assigned a *qualitative value*, which is named using domain knowledge. This process is called *qualitative abstraction*. Let  $feature \in \mathbb{R}_+$  be a feature and  $R$  be a simple rule containing one split condition:  $R = \{feature < val\}$ . In this case, the domain can be partitioned into  $I_1 = (-\text{inf}, val)$

and  $I_2 = [val, +\text{inf})$ . To these intervals,  $q_1$  and  $q_2$  qualitative values can be assigned.

As an example, in a ruleset built by training a rule-based model on the MetroPT-3 dataset [14], there is a condition:  $[Reservoirs \leq 7.98]$ . After qualitative abstraction using the process described above, we can say that the *Reservoirs* feature could have two qualitative values: *LOW* and *HIGH*.

Converting a ruleset into a qualitative model means we apply this process to all of the variables using all of the conditions in the ruleset. The clustering of overlapping conditions is solved using interval logic.

## IV. MODEL EVALUATION WITH DIFFERENCE MODEL

The aim is to examine the primary model using a surrogate model that matches the data precisely. Next, we employ a model difference method to evaluate the performance of the primary model by comparing its outputs with those of the surrogate model and the labels of the test dataset.

Through this comparison, we can pinpoint the specific regions or conditions in which the primary model underperforms. Identifying these areas of weakness helps us determine which corrective or improvement measures to apply. We also take into account both the magnitude of the errors and the severity of their potential consequences, ensuring that any refinements target the most critical issues effectively.

### A. xAI Surrogate Model

Advanced *opaque models* often dominate in production because they typically outperform simpler, more interpretable models. They excel at handling complexity by learning intricate patterns, including nonlinear relationships in high-dimensional data. However, their complexity makes it hard to understand how decisions are reached and to debug errors when they arise. A classic example is a neural network with fine-tuned parameters trained on clean, standardized data.

Analyzing complex models is challenging due to limited interpretability. We address this by approximating their behavior with a simpler, interpretable surrogate model. This approach replaces expected behavior representations with an abstract surrogate model that covers positive samples in n-dimensional space (Figure 2). This hybrid, model-agnostic method provides a clearer understanding of the original model's behavior.

In our research, we adopt an *interpretable-by-design xAI model* as the surrogate. Apart from interpretability — which allows transformation into a qualitative model — high sensitivity is critical for covering all positive samples. Achieving 100% coverage is typically unrealistic, so techniques like over-sampling, boosting, or cost-sensitive learning are employed to enhance model sensitivity.

### B. Interpretation of Model Differences

As a further step of the analysis, a difference model can be built on top of the surrogate and the primary models. If we evaluate this model while also taking into account the labels of the training dataset, the samples can be divided into

three categories: 1) Certain solutions; 2) Uncertain solutions; 3) False solutions.

*Certain solutions* represent the so called *stable zones*, where both models predict correctly. These scenarios form the basis of the normal operation and the stability of xAI surrogate models. In our research, we have not paid much attention to them. However, they can be used to examine the *characteristic samples* found in regions or clusters where models are performing confidently or consistently well. This analysis can be useful in examining which are the best learned rules.

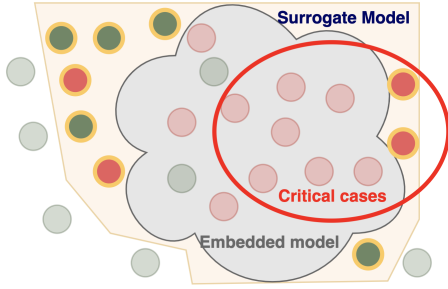


Fig. 2. Surrogate Model and Boundary Region

*Uncertain solutions* contain samples where the models contradict each other. They help identify the edge cases, representing the boundary region between the positive and negative samples (Fig. 2). These can be partitioned into two subcategories: a) *Surrogate Divergence* represents those samples where the surrogate model produces a misprediction and the primary model is correct. They are mostly the consequence of an abstract or overly simplified surrogate model. If the number of these cases is too high, the level of abstraction can be directly tuned by controlling the complexity of the surrogate model. (e.g. setting the depth of a DT). The ideal and most probable scenarios represented by these are *false alarms* (false positive samples), as the goal of the surrogate model is to cover all positive samples. The occurrence of false negative predictions of the surrogate model is very rare. Ideally, there should be none. The existence of these samples is because there is no perfectly sensitive model in practice. However, the question of to what extent a perfectly sensitive surrogate model is needed can be further discussed.

b) *Primary Model Divergence* represents those cases where the primary model mispredicts, and the surrogate model is correct. These cases must be analyzed as these carry the greatest risks in a real application. These scenarios represent the most of the escaping faults, it must be examined whether they are critical. Less often, but it may happen that the real label is disputed, and the embedded model predicts (partly) right. This phenomenon is called *data drift* [15].

*False Solutions* contains samples where both models mispredict. A high number of these cases can often indicate data or feature problems. These cases can also result from model problems such as underfitting, overfitting, or poor generalization. In this case, we assume that the labels in the training and

test datasets are correct. This paper does not address scenarios where errors occur during the labeling process.

### C. Evaluation of the Difference Regions

Our goal of model evaluation using difference models is, firstly, to set up priorities regarding the samples, i.e., the critical cases are given special attention, and secondly, to incorporate feedback into the development cycle. This means an improved cost function that makes the model more accurate to the critical cases, the examination of the underrepresented samples, and the minimization of the data drift. Examining each region is a multi-step process: 1) Identify regions relevant from the perspective of errors or inconsistencies (e.g., JST leaves, data slicing, clustering). 2) Understand why and what types of errors occur in those regions. 3) Assess the risk (evaluate the cost or potential hazard associated with the specific type of error). 4) Develop an action plan (e.g., data cleaning, model fine-tuning, introducing new features, modifying surrogates).

To identify the regions, the difference model can be fitted, and its results compared with the original labels of the test dataset. These results can then be used to segment the dataset.

Various methods can be employed to evaluate the regions, depending on the type of region in question. By fitting qualitative rules to the segmented data points, it is possible to analyze the types of conditions that occur in those regions based on the operational model. This empirical approach helps domain experts gain a general understanding of the region.

Additionally, by comparing the qualitative rules across multiple regions, potential overlaps can be identified. When an operational domain appears in multiple regions, it may indicate an issue with the model or suggest that the engineering model is incomplete.

Once these regions are identified, they can be used for exploratory data analysis (EDA). EDA helps visualize whether the region truly differs from others (e.g., extreme values, rare events, clusters with varying densities). It also allows the detection of *outlier* points or patterns where the model responds incorrectly. This type of anomaly detection contributes to understanding the error profile of the regions (e.g., records typically having missing values or extreme feature values).

During EDA, whether the regions (e.g., leaf nodes in a JST) are sharply separated or exhibit a continuous transition can be examined. This helps assess the artificial boundaries between regions (e.g., created by simple decision rules) and how much they reflect genuine, distinct segments within the data.

Regions critical from a dependability perspective (e.g., extreme values, high-cost errors) can be easily visualized using EDA, allowing domain experts to gain insights quickly. For instance, it can be assessed whether a specific anomalous region truly results from extreme behavior in the actual process or is instead due to measurement or annotation errors.

Error Propagation Analysis (EPA) [16] offers the capability to evaluate the model within the context of a larger system. Since the model's decisions propagate within the system (e.g., supporting specific decisions), it is necessary to assess whether

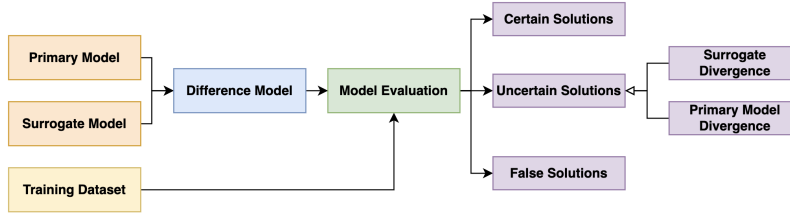


Fig. 3. Solutions of the Model Difference Evaluation

a correct or incorrect decision in a given system state could lead to critical system-wide failures.

EPA provides the opportunity to handle errors with weighted importance. For example, a "False Negative" in a region with few occurrences but high risk (e.g., fraud, hazardous situations) can be more severe than a frequently occurring but low-risk error. Beyond the frequency of errors, the cost or critical impact of the errors can also be incorporated into the analysis.

During the evaluation, the domain expert can assign risk to different regions and operational modes, considering the results of both EDA and EPA. This list can then be used to guide efforts on improving the model and determining which safeguards should be implemented in the system to prevent the model's errors from causing system-wide failures. The insights gained from the evaluation can be used to improve the models by identifying operational regions where performance was suboptimal. Qualitative rules enable impact analysis of the model within its embedding system context, allowing these results to be integrated into the refinement process.

Furthermore, understanding the operational domains creates opportunities to enhance the data collection process or to artificially adjust the existing dataset (e.g., in cases of imbalanced datasets). This ensures that errors are mitigated in subsequent training iterations, reducing the likelihood of incorrect predictions.

## V. EXAMPLE

The example is demonstrated using the MetroPT-3 dataset [14], which contains data from the Air Production Unit (APU) component of a metro train, including attributes such as pressure, temperature, and motor current. The dataset labels indicate whether the component was faulty at a given time.

As a first step, we trained a neural network tasked with predicting errors in the APU. In the second step, we built a surrogate model based on the training data, designed to accurately capture the faulty cases. Finally, we used the IMD algorithm to compare these models and matched the results against the labels in the test dataset. The resulting regions were then evaluated, and qualitative rules were derived using the JST, providing a foundation for further analysis.

### A. Training a Neural Network with Preprocessed Data

As a highly advanced model, a neural network was used as a failure prediction model trained on the MetroPT-3 dataset. This neural net consists of two hidden layers having 64 neurons

and the activation function of *relu* and an output layer of a single neuron using the *sigmoid* activation function. This model was then trained on clean (subsamped and removed outliers) standardized data.

### B. Surrogate Model Creation

Our method to build a sensitive model was to iteratively train a DT with cost-sensitive learning, boost the weight of false negative (FN) samples, and repeat. We trained a DT using class weights of 1/11 for negative samples and 10/11 for positive samples, then multiplied the weight of FN samples by 1.5 and repeated the cycle 10 times.

The resulting model has an extraordinarily high sensitivity, covering all positive samples except one. However, the side effect of low specificity also can be seen as the false positive rate increased by 268%.

### C. Regions from Difference Model

Table I summarizes the number of samples that fall into each solution region. From the table, it is evident that the majority of samples belong to a certain solution category. The remaining samples represent cases where either one or both models made incorrect predictions. In the following, we will examine the data points that fall into the "Primary divergence" region, as these points may indicate operational domains where the primary model needs to perform well to ensure coverage of multiple operational modes, particularly in critical system applications. Additionally, we will analyze the cases where both the surrogate and the primary models produced incorrect predictions.

### D. Evaluation

We fit a RIPPER model to explain those scenarios where the predictions of the primary model diverge with a rule set in CNF form. For instance, one rule describes these system states as:

$$\begin{aligned} & (P_{discharge\ valve} \geq -0.01) \wedge (P_{discharge\ valve} < 1.684) \\ & (T_{oil} \geq 66.6) \wedge (T_{oil} < 70.225) \\ & (P_{pneumatic\ panel} < 6.86) \end{aligned} \quad (8)$$

The qualitative model of this rule is shown in Figure 4.

We used EDA to investigate the False region in order to identify operational domains where the model underperforms. During the analysis, the H1 variable emerged as a notable

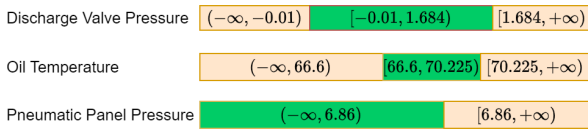


Fig. 4. Qualitative Divergence Rules

TABLE I  
SURROGATE MODEL EVALUATION - REGIONS

Solution type	Primary	Surrogate	Label	# of samples
Certain	0	0	0	292282
	1	1	1	5955
Uncertain	1	0	0	47
	0	1	1	90
Surrogate divergence	0	1	0	3736
	1	0	1	1
False	1	1	0	1279
	0	0	1	0

factor. In Figure 5, the distribution of H1 across the entire dataset is shown in red, while its distribution within the False region is highlighted in blue. It is evident that the blue values are concentrated in the lower range. This pattern was observed for several other variables as well, enabling us to delineate the problematic domain using this method.

## VI. CONCLUSION AND FUTURE WORK

This paper presented a systematic approach for analyzing and evaluating embedded AI components in critical systems through model differencing and surrogate modeling. The approach focuses on identifying and addressing areas of divergence between interpretable surrogate models and opaque primary models, providing actionable insights for improving system performance and dependability. By leveraging qualitative abstraction and explainable AI techniques, the proposed framework enables domain experts to identify edge cases, assess risks, and refine models to mitigate critical issues. The case study utilizing the MetroPT-3 dataset demonstrates the application of the approach.

In future work, we will examine the use of the results from explanatory model analysis [4] for validating and improving the rule set derived from the surrogate model. In this paper, we neglect the ideal requirement for absolute sensitivity (no false

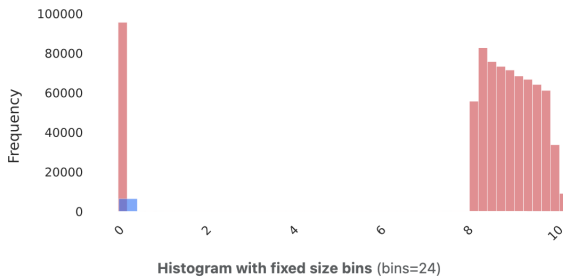


Fig. 5. Evaluation of the H1 Feature

negatives) combined with high specificity (few false alarms) and adopt a best-effort approach, as they do not affect the core algorithm. Another approach under consideration is Inselberg’s nested cavities [17], which can ensure perfect sensitivity—at least for the training set—and then gradually refine specificity. The method employs a convergent series of upper and lower approximations driven by Reed-Mueller normal form logic rather than the usual AND-OR constructs.

## REFERENCES

- [1] K. D. Forbus, “Chapter 9 qualitative modeling,” in *Handbook of Knowledge Representation*, ser. Foundations of Artificial Intelligence, F. van Harmelen, V. Lifschitz, and B. Porter, Eds. Elsevier, 2008, vol. 3, pp. 361–393. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S157465260703009X>
- [2] B. Kubica, *Interval Methods for Solving Nonlinear Constraint Satisfaction, Optimization and Similar Problems: From Inequalities Systems to Game Solutions*, 01 2019.
- [3] B. Goodman and S. Flaxman, “European union regulations on algorithmic decision making and a “right to explanation,”” *AI Magazine*, vol. 38, no. 3, p. 50–57, Sep. 2017. [Online]. Available: <http://dx.doi.org/10.1609/aimag.v38i3.2741>
- [4] P. Biecek and T. Burzykowski, *Explanatory Model Analysis*. Chapman and Hall/CRC, New York, 2021. [Online]. Available: <https://pbiecek.github.io/ema/>
- [5] D. Gunning, E. Vorm, J. Y. Wang, and M. Turek, “Darpa’s explainable ai (xai) program: A retrospective,” *Applied AI Letters*, vol. 2, no. 4, p. e61, 2021. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/aill.2.61>
- [6] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, pp. 81–106, 1986.
- [7] S. Dash, O. Günlük, and D. Wei, “Boolean decision rules via column generation,” 2020.
- [8] W. Cohen, “Fast effective rule induction,” *Twelfth International Conference on Machine Learning: 1995*, vol. 95, 10 2000.
- [9] M. T. Ribeiro, S. Singh, and C. Guestrin, ““why should i trust you?”: Explaining the predictions of any classifier,” 2016. [Online]. Available: <https://arxiv.org/abs/1602.04938>
- [10] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>
- [11] P. Cousot, “Abstract interpretation based formal methods and future challenges,” ser. Lecture Notes in Computer Science, R. Wilhelm, Ed. Springer-Verlag, 2001, pp. 138–156.
- [12] V. Arya, R. K. E. Bellamy, P.-Y. Chen, A. Dhurandhar, M. Hind, S. C. Hoffman, S. Houde, Q. V. Liao, R. Luss, A. Mojsilović, S. Mourad, P. Pedemonte, R. Raghavendra, J. Richards, P. Sattigeri, K. Shanmugam, M. Singh, K. R. Varshney, D. Wei, and Y. Zhang, “One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques,” 2019. [Online]. Available: <https://arxiv.org/abs/1909.03012>
- [13] S. Haldar, D. Saha, D. Wei, R. Nair, and E. M. Daly, “Interpretable differencing of machine learning models,” 2023. [Online]. Available: <https://arxiv.org/abs/2306.06473>
- [14] V. B. R. R. Davari, Narjes and J. Gama, “MetroPT-3 Dataset,” UCI Machine Learning Repository, 2023, DOI: <https://doi.org/10.24432/C5VW3R>.
- [15] S. Ackerman, O. Raz, M. Zalmanovici, and A. Zlotnick, “Automatically detecting data drift in machine learning classifiers,” *arXiv preprint arXiv:2111.05672*, 2021.
- [16] A. Földvári and A. Pataricza, “Handling uncertainty in error propagation analysis,” in *Proceedings of the 30th Minisymposium*. Budapest University of Technology and Economics, 2023, p. 29–32. [Online]. Available: <http://dx.doi.org/10.3311/minisy2023-008>
- [17] A. Inselberg, *Parallel Coordinates: Visualization, Exploration and Classification of High-Dimensional Data*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 643–680. [Online]. Available: [https://doi.org/10.1007/978-3-540-33037-0\\_25](https://doi.org/10.1007/978-3-540-33037-0_25)