Kovalerchuk B., Huber L., Multilayer Machine Learning with Visual Knowledge Discovery, Information Visualization, 2025, https://doi.org/10.1177/14738716251363233

Multilayer Machine Learning with Visual Knowledge Discovery

Boris Kovalerchuk
Department of Computer Science
Central Washington University, USA
Boris.Kovalerchuk@cwu.edu

Lincoln Huber
Department of Computer Science
Central Washington University, USA
Lincoln.Huber@cwu.edu

***Abstract*—**Increasing domain experts' trust in Machine Learning (ML) models requires explainable models. However, employing explainable techniques alone can be insufficient for getting such trust in the model. The reasons include insufficient model accuracy, overfitting, overgeneralization, too complex explanations, and others. Integration of multilayer/multilevel ML and lossless visualization approaches for high-dimensional data is proposed to help address these challenges. Multilayer/multilevel ML approaches have been extensively developed to capture complex relations especially within modern deep learning (DL) approaches. Graph-based lossless visualization approaches for high-dimensional data emerged recently in ML as an opportunity to represent high-dimensional ML information without loss of n-D information as an alternative to lossy dimension reductions methods. These approaches within a Visual Knowledge Discovery (VKD) paradigm use General Line Coordinates (GLC) to create explainable and reversible n-D representations in a visual form with intent to increase user's trust in ML models. However, when viewing vast volumes of data, occlusion affects visual approaches. Both multilayer/multilevel and visualization approaches have challenges of becoming too complex for the user. This paper adapts to these challenges by developing multilayer/multilevel ML approaches based on hyper-rectangles/hyperblocks/hyperboxes and a classifier decomposition at several levels as simple base elements for both multilayer/multilevel ML and visualization. The efficiency of the proposed integrated approach is demonstrated in several cases studies on numerical data and images. Major benefits from this approach are increased generalization and less complex interpretable rules to increase user confidence in each classification.

Keywords—Visual Knowledge Discovery, Explainable Machine Learning, General Line Coordinates, Machine Learning, Visualization, Classification

1. INTRODUCTION

1.1 Motivation

The goal of this paper is *integrating* two categories of productive approaches in Machine Learning (ML): *multilayer/multilevel* machine learning approaches and *lossless visualization* approaches for high-dimensional data to produce interpretable and trustworthy ML models. The motivation for this integration is as follows. The popular Deep Learning (DL) multilayer architecture produces highly accurate machine learning models for difficult tasks, while emerging lossless visualization approaches for multidimensional data appeal to users due to their transparency and trustworthiness. Therefore, integrating these approaches is a promising avenue, which is the goal of this paper.

However, these approaches cannot be integrated as-is due to their deficiencies. Multilayer DL architectures are distrusted by users because of their black-box, uninterpretable nature. We need an alternative multilayer architecture that is transparent to users, as well as an expansion of lossless visualization approaches into multilayer forms. This paper proposes two ML multilayer architectures (1) based on the **hyper-rectangles** (hyperblocks, HBs) that are transparent and interpretable for the users, and (2) lossless visualization based on **decomposition** of a linear and non-linear classifiers. The motivation for selection of the hyperblocks architecture is that it is more general than interpretable decision trees and is at the same level as interpretable logical rules in the First order Logic. This eliminates the need in methods like LIME and SHAP designed to explain black box models. Another advantage of the hyperblocks architecture is that hyperblocks are naturally transparently visualized in Parallel Coordinates and other forms of lossless General Line Coordinates that are easy for the users to understand. The motivation for the decomposition approach is in its transparency and abilities to interpret it with hyperblocks.
Respectively the effort in this paper has three aspects summarized below.

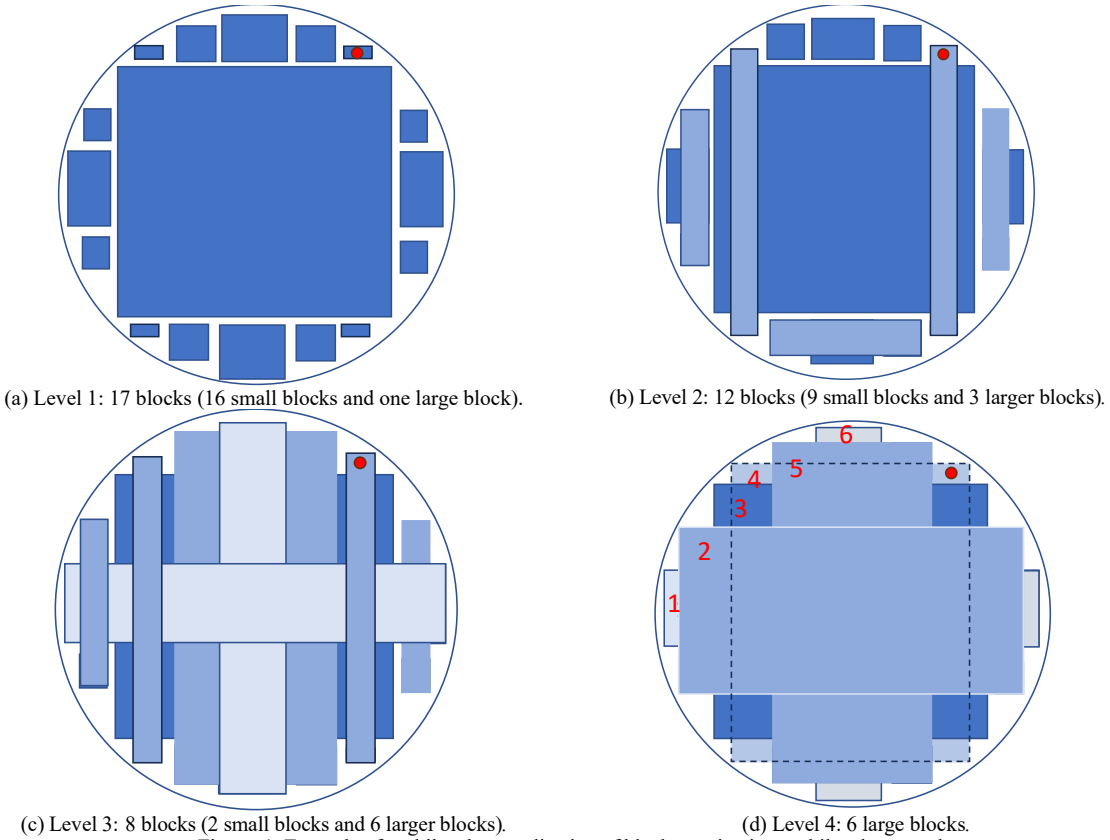
- Merging multilayer Machine Learning and lossless visualization approaches
- For getting interpretable ML models
- By designing and testing hyperblocks-based algorithms and Decomposition-based algorithms.

Multilayer/multilevel ML approaches have been extensively developed to capture complex relations, especially within a Neural Network (NN) framework starting from the multilayer perception and going to the modern deep learning Neural Networks. Commonly terms multilevel and multilayer are used interchangeably in the literature. We will also use both terms interchangeably with a preference of the term used in a source for specific issues.

Lossless visualization approaches for high-dimensional data emerged recently in ML as an opportunity to represent high-dimensional ML information without loss of n-D information in contrast with dimension reductions methods like PCA, MDS, t-SNE, UMAP, and others [Kovalerchuk, 2018]. This allows the discovery of complex n-D relationships in high-dimensional data with more active human-in-the-loop involvement. We review advantages and disadvantages of multilayer ML and lossless n-D data visualization approaches and show the benefits of integrating them. This paper concentrates on two types of multilayer architecture: (1) multilayer architecture based on **hyper-rectangles** and (2) multilayer **decomposition** of classifiers to enhance accuracy and interpretability of ML algorithms. *Multidimensional rectangles,* also known as *hyper-rectangles*, **hyperblocks** and *hyperboxes* play an important role in ML as simple interpretable/explainable concepts with several algorithms proposed for years. Hyperblocks (HBs) are simple **ML models** that are discovered by **Hyperblocks discovery algorithms**. Over 70 hyperblock models are fully visualized in 9 figures in Parallel Coordinates in this paper.

1.2. Research Issues of Hyperblock Models

Next, we outline the motivation and challenges of creating hyperblocks models at various levels, a key contribution of this paper. The illustration is presented with 2-D data in Figure 1. As seen in Figure 1a, the training data for a particular class are distributed equally over each rectangle. It is intuitively obvious that the data form a hypersphere in n-D, space or a circle in 2-D space. The hypersphere, however, lacks an interpretation for heterogeneous data despite having a straightforward mathematical description [Kovalerchuk et al., 2021].



(a) Level 1: 17 blocks (16 small blocks and one large block).

(b) Level 2: 12 blocks (9 small blocks and 3 larger blocks).

(c) Level 3: 8 blocks (2 small blocks and 6 larger blocks).

(d) Level 4: 6 large blocks.

Figure. 1. Example of multilevel generalization of blocks motivating multilevel approach.

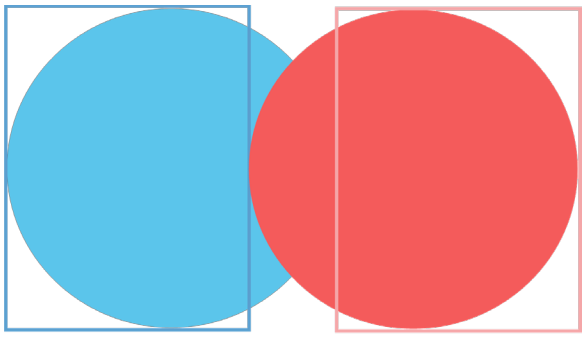
Rectangles, on the other hand, are explainable in the corresponding domains for heterogeneous data, and are comparable to straightforward logical formulae of a conjunction of intervals on individual attributes. To obtain an understandable generalization of the data, rectangular envelopes—referred to as blocks in 2-D—are created around

them. Similarly, several methods have been developed to generate n-D rectangles, which are known as Hyperblocks. There are 17 blocks total, 16 little blocks and one huge block in Figure 1a. When there are 17 blocks in place of a single circle, the explainable HB technique clearly has a flaw.

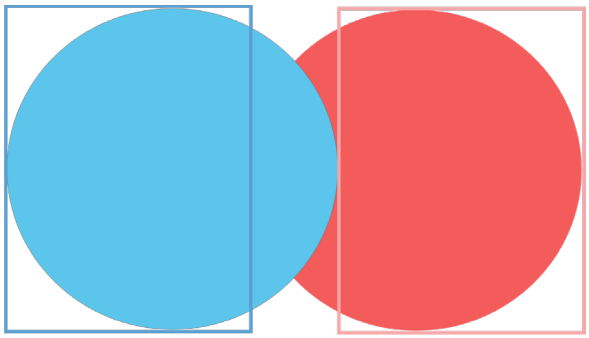
The **low prediction confidence** with **small blocks** is the problem, in addition to the enormous number of blocks. Examine the small block with the new red case that needs to be predicted in the upper right corner of Figure 1a. There are not many training cases from the class in this block, hence the prediction of this class is not highly certain. This is a common instance of **overfitting** data. Figures 1b and 1c, on the other hand, depict this situation as part of a considerably larger block, and Figure 1d as part of a very large block that is delineated by a dotted line. When conveyed to the domain expert or user, these larger blocks significantly increase the level of confidence in the case's classification. Level 1 to level 4 block generalization is illustrated in Figure 1, where 17 level 1 blocks are generalized to 12 level 2 blocks, which are then generalized to 8 level 3 blocks, and eventually to 6 large level 4 blocks.

Although the hyperblock generalization process in 2D space may be seen and carried out naturally with this representation, it cannot be carried out in the same manner for n-D data in n-D space. To visualize n-D data losslessly, more advanced visualization techniques are required. For the HB generalization process, General Line Coordinates (GLC) [Kovalerchuk, 2018] enable such visualizations preserving multidimensional information. Later in this work, the GLC process is explained for a particular approach known as GLC-linear (**GLC-L**).

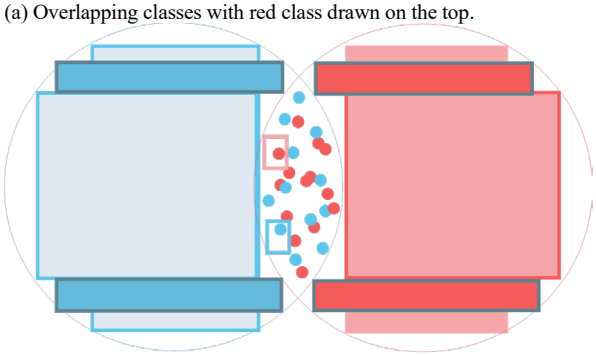
Figure 2 provides the next example where blue and red classes consist of the overlapping circles as shown in Figures 2a and 2b.



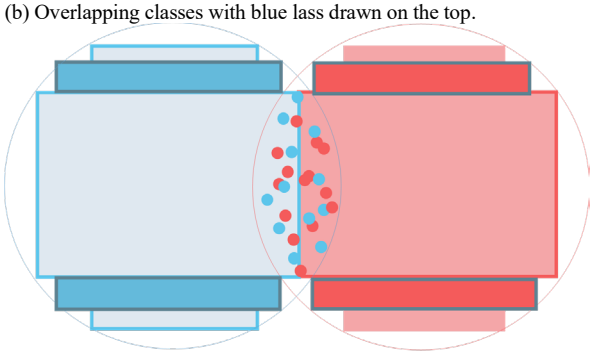
(a) Overlapping classes with red class drawn on the top.



(b) Overlapping classes with blue lass drawn on the top.



(c) Pure blocks/rectangles



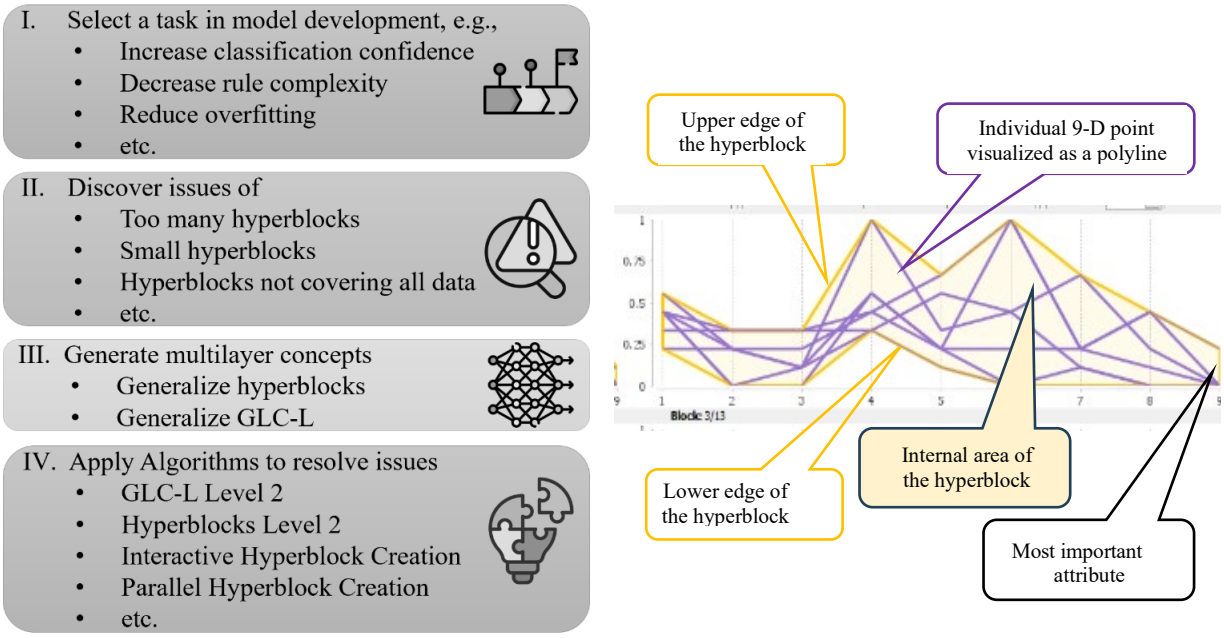
(d) Impure blocks/rectangles with overlap areas included.

Figure 2. Example of multilayer generalization of blocks for two overlapping classes.

We can easily build two large pure rectangles shown as blue and red rectangles in these figures. While these rectangles are simple, they are deficient. They overgeneralize the circles in some areas and do not cover circles in other areas. Here, the large area between these two rectangles is left without any rectangles. This area includes the overlap areas of rectangles, but also small pure areas. Figure 2c and 2d show an attempt to classify cases in that area to make more refined classification. Using the same idea as we demonstrated in Figure 1 we build quite large blue and red rectangles in the pure areas. All attempts to build rectangles in the overlap area end up with very small rectangles with red and blue frames in Figure 2c. This example shows the negative consequences of attempts to continue building pure rectangles in the overlap area. Figure 2d shows an alternative attempt to build impure larger rectangles, which split overlap area between red and blue rectangles. This example demonstrates the need to stop building small HBs and use other approaches. It can be building *impure larger HBs* like shown in Figure 2d or trying to find an alternative classification method with updated cases and features.

Figure 3a outlines the proposed multilayer explainable ML model development process with hyperblocks and function decompositions with GLC-L. It contains four stages: (I) selecting a *task within* model development like reducing overfitting, (II) discovering *issues* like many hyperblocks with only a few cases in them (overfitting issue), (III) generating multilayer solution concepts like generalizing hyperblocks, and (IV) applying algorithm based on (III) to resolve the issues. The key concepts in this approach are Hyperblocks and Decomposition with GLC-L at several levels to build interpretable models, like building hyperblocks of level 2 for a given data. The listed four stages outline the process of discovering issues with these concepts and resolving them. By following this process, highly explainable *human-centered* machine learning models can be developed. These models will increase user confidence in each prediction and maximize the potential for interactive Visual Knowledge Discovery.

The proposed methods for stages (II) - (IV) are described in sections 3 and 4. The examples of these stages for specific tasks are presented in section 5 with five cases studies where multilayer concepts are demonstrated. Case study 1 illustrates the Decomposition approach and case studies 2-5 demonstrate the Hyperblocks approach.



(a) Multilayer explainable ML model development process.

(b) Example of a hyperblock model and data visualization in Parallel Coordinates with its elements commented.

Figure 3. Proposed Multilayer explainable machine learning model development process and an Example of a hyperblock model and data visualization.

Figure 3b presents an example of a hyperblock model visualization in Parallel Coordinates. All cases in HB are visualized as polylines. The yellow lines show the upper and lower limits of the hyperblock in coordinates. The internal area of the hyperblock is colored light yellow, and the attribute with narrowest range is marked as most important.

1.3. Novel Ccontributions

Below we describe the novel contributions of this paper. They consist of the contributions outlined above: (1) Merging Multilayer machine Learning and Lossless visualization approaches. (2) Getting Interpretable machine learning models, and (3) Designing and testing Hyperblock-based algorithms and Decomposition-based algorithms.

Specifically, we propose essentially new concepts and algorithms for levels 2, 3, and beyond with Hyperblocks and decomposition approaches. They are (1) a new GLC-L generalization for levels 2, 3, and beyond, and (2) a series of new Hyperblock algorithms of levels 2,3 and beyond. The new algorithms include Algorithm for Producing Simplified GLC-L, Level n Hyperblock Creation (LnHC) algorithm, Interactive Algorithm for Creation of Pure Intervals, Envelope Case Finder (CRD1-CRD3) algorithms, Decision Tree Hyperblocks (DT-HB) algorithm, Hyperblock creation algorithm (GLC-HBRL), Parallelization algorithm, Parallel Interval Merger Hyperblock Algorithm (PIMHyper).

The details of novel contributions of these algorithms are described in the respective sections of the paper. One of them is the capabilities of the Interactive Algorithm for Creation of Pure Intervals presented in section 4.2. It shows that simple rectangles in a lossless visualization correspond to quite complex areas in the n-D space. This is an important advantage of the proposed lossless visualization. It allows users without advanced knowledge of machine learning to directly manipulate with high-dimensional data and machine learning models.

Most of the Hyprblock algorithms of level 1 were developed before, e.g., General Line Coordinate Hyperblock Rules Linear (GLC-HBRL) [Huber et al., 2024; Recaido et al, 2022, Hayes et al. 2024]. CLC-L level 1 algorithm was defined before in [Kovalerchuk, Dovhalets, 2017].

The rest of this paper is organized as follows. Section 2 reviews state-of-the-art. Section 3 describes a new multilayer architecture based on decomposition and lossless visualization with multilevel architecture. Section 4 describes a new multilayer architecture based on hyperblocks with algorithms for this architecture. Section 5 contains five case studies, which evaluate the proposed architectures and algorithms. Case studies 1 and 2 are devoted to evaluation of algorithms on images and cases studies 3-5 are devoted to evaluation of algorithms of tabular data. Section 6 outlines the features of the DV2 software, and section 7 contains a conclusion with the future work outlined.

2. STATE-OF-THE-ART

This section is devoted to the state-of-the-art in Multilayer Machine Learning, Hyperblock models and Comparison of the proposed approaches with the state-of-the-art approaches.

2.1. Multilayer machine learning vs. single layer machine learning

Many multilayer/multilevel ML approaches have been proposed for years starting from the multilayer perception (MLP). Surveys are available, e.g., [Zhang et al., 2018; Wang et al., 2022]. Below in Table 1 we summarize the advantages and disadvantages of multilayer ML approaches relative to simpler single-layer ML approaches. While the listed advantages make multilayer approaches powerful, the choice between single-layer and multilayer models heavily depends on the specific problem requirements, available computational resources, and the need for model interpretability. The severity of disadvantages also varies depending on the specific problem, dataset, and model architecture [Caruana, Niculescu-Mizil, 2006]. One of the fundamental challenges for selecting multilayer or single layer architectures is in limited abilities to visualize n-D data losslessly that we address in this paper.

Table 1. Advantages and disadvantages of Multilayer machine learning architectures.

Advantages	Disadvantages
A1. *Enhanced Representation Learning* by capturing complex patterns at multiple levels with raw data [Zhou, Feng, 2017].	D1. *Need for Large Datasets* to perform effectively without overfitting in models like deep learning [Zhou, Feng, 2017].
A2. *Interpretability* by using specific multilayer structures of the model design like decision trees [Lundberg, Lee, 2017].	D2. *Reduced Interpretability* due to multiple layers making result explanation more difficult [Lundberg, Lee, 2017].
A3. *Intuitive visualization* for specific multilayer model structures like decision trees [Kovalerchuk et al., 2024].	D3. *Increased visualization challenge* for multilayer model structures like deep networks [Kovalerchuk et al., 2024].
A4. *Flexibility in Feature Representation* by learning supervised and unsupervised representations [Zhu et al., 2019].	D4. *Increased Sensitivity to Hyperparameters* due to their larger number and complex optimization [Ke et al., 2017].
A5. *Improved Generalization* by learning features at different levels of abstraction [Feng et al., 2018].	D5. *Risk of Overfitting* due to increased complexity especially with limited training data [Prokhorenkova et al., 2018].
A6. *Increased Model Capacity* by handling more complex tasks and larger datasets [Chen, Guestrin, 2016].	D6. *Increased Computational Complexity* of training vs. single-layer models [Chen, Guestrin, 2016].
A7. *Handling Mixed Data Types* by using a multilayer design that support numerical and categorical data [Friedman, 2001].	D7. *Higher Risk of Getting Stuck in Local Minima* due to specific multilayer model architectures [Friedman, 2001].
A8. *Robustness to Missing Data* by using structures of the multilayer design adaptable to such scenarios [Loh, 2011].	D8. *Increased Memory Requirements* due to more parameters and intermediate multilevel computations [Loh, 2011].
A9. *Joint Optimization* of all layers, enabling more effective representations and better performance [Ke et al., 2017].	D9. *Vanishing Gradient* in models, like deep networks with slow learning in the earlier layers [Feng et al., 2018].
A10. *Computational Efficiency* by the scalable multilayer model design like decision trees [Prokhorenkova et al., 2018].	D10. *Longer Training Time* with increased model complexity that is undesirable in time-sensitive tasks [Zhu et al., 2019].

2.2. Machine Learning with hyper-rectangles/hyperblocks

Below we present approaches based on hyper-rectangles. Although hyper-rectangles, hyperblocks, and hyperboxes refer to the same underlying concept, they have inspired a variety of distinct methodological approaches. Below we use terms as they are present in the respective sources. The survey of several ML methods based on the hyperboxes

can be found in [Khuat et al., 2021]. These methods open several opportunities for building efficient models in ML with some caveats that we discuss below too.

1. *Interpretability*: Hyper-rectangles are highly interpretable geometric structures in n-D space, making them easy for end-users to understand and visualize [Hayes, Kovalerchuk, 2024].

2. *Abilities to classify mixed and pure hyperblocks*. The Hyper algorithm generalizes decision trees and can discover overlapping or non-overlapping HBs either interactively or automatically [Hayes, Kovalerchuk, 2024].

3. *Visualization*: Hyper-rectangles can be effectively visualized using parallel coordinates, making complex n-D concepts simpler to understand for end users [Huber et al., 2024].

4. *Ensemble Models*: A boosting ensemble-based model called Hyper-Rectangle Base Model (HRBM) uses hyper-rectangles as base models. It follows a gradient boosting approach to produce an ensemble as a weighted linear combination of hyper-rectangles [Konstantinov, Utkin, 2023]. While the base hyper-rectangles are interpretable their ensemble needs to be made interpretable. The authors use a modified SHAP for this. Unfortunately, despite the stated interpretability of the ensemble with SHAP it is not well justified as we discuss in section 2.3. An alternative method for constructing an ensemble of models is described in [Williams, Kovalerchuk, 2025], which avoids relying on a non-interpretable weighted sum of hyperblocks. Instead, it utilizes generalized decision trees based on hyperblocks, ensuring that the ensemble's decision-making process remains fully interpretable.

5. *Scalability*: Hyperbox-based learning algorithms as fundamental representational units, have shown potential for high scalability and online adaptation in dynamically changing environments and streaming data [Khuat et al., 2021].

6. *Flexibility*: The Nested Hyper-Rectangle Learning Model (NHLM) can handle *binary, discrete, or continuous variables*. The nested hyper-rectangles allow *updating a model* with new cases, which belong to another class but are within an existing pure hyper-rectangle. NHLM uses an "*error tolerance*" parameter for continuous variables to determine how close two values must be to be considered "the same" [Chen, 2005; Salzberg, 1991].

An alternative approach that adds flexibility is based on the concept of *membership functions*, which measures the degree to which the input n-D point belongs to the hyperbox [Khuat et al., 2021]. When the n-D point is completely contained within the hyperbox, the degree of membership is 1, and it decreases when the n-D point moves away from the hyperbox. A method to predict the class of the n-D point **x** in this approach is (1) computing maximum of membership values of **x** for all HBs for each class and then (2) classifying an n-D point **x** to the class with the highest membership value to the class [Castillo, Cardenosa, 2012].

The difficulty with both *error tolerance* and *membership functions* are that they need to be learned from the data, which can be limited. For the membership function approach insufficient training data for restoration of a high-dimensional membership function leads to heuristic approximation methods.

7. *Generalization*: A simple form of HBs make generalization easy by using them. However, the hyperbox approach can lead to *overgeneralization* of the cases within the hyperbox. The convex hull around actual cases in n-D space is within their hyperbox, but it is smaller than the hyperbox [Williams, Kovalerchuk, 2025], e.g., three points in 2-D form a triangle as their convex hull but the rectangle around them is larger. It is also visible in the situation when all actual points are in the hypersphere. In this situation the volume of the hypercube around this hypersphere grows exponentially, which is much faster than the volume of the hypercube grows with as the dimension increase.

8. *Abilities to find most informative hyperblocks.* Intuitively each such HB should (a) be pure or *almost pure* (contains majority of the cases of a single class), (b) contain *many cases* of that class, (c) occupy a relatively *small area* in n-D space, and (d) all cases *compactly located* in that area. A HB with all these properties may not exist, therefore a *tradeoff indicator of informativeness* of the HB was proposed in [Liu et al., 2017] called *hyperbox entropy* (*HE*):

$$HE = N \times std \times s$$

where N is the number of data samples in HB, *std* is the data variance in HB to represent the distribution of points in HB, and s is the area of HB to represent the size of HB. In [Liu et al., 2017] the hyperboxes, which include more data with smaller size s and *std* are viewed as better ones. This indicator assumed that all cases in the hyperblock belong to a single class or the number of cases of the other classes is very small. As every trade off indicator, *HE* is deficient. Consider three HBs:

HB_1: small $N = 10$, small *std* = 0.3, small area $s = 1$, $HE_1 = 10*0.3*1 = 3$
HB_2: medium $N = 100$, small *std* = 0.3, small area $s = 1$, $HE_2 = 100*0.3*1 = 30$
HB_3: medium $N = 100$, small *std* = 0.3, large area $s = 100$, $HE_3 = 100*0.3*100 = 3000$

We have HB_2 that is *better* than HB_1 with more cases, the same other two properties, and $HE_2 > HE_1$. In contrast, HB_3 is not necessarily better than HB_2, because HB_3 has larger area, the same two other properties, which can indicate the overgeneralization of data. However, HE_3=3000 is greater than HE_2=30. This example shows that converting 3 measures to a single *HE* number is deficient and a multi-objective optimization with Pareto type of criteria can be a better way to evaluate the quality of the hyperblocks.

2.3. Comparison of proposed approaches with state-of-the-art approaches

The major difference of the hyperblocks approach from popular methods like **LIME** [Ribeiro et al, 2016] and **SHAP** [Lundberg, Lee, 2017] is that the hyperblocks approach creates *intrinsically interpretable ML models* from the beginning. In contrast, LIME and SHAP assume that first a non-interpretable black box model is built and then LIME and SHAP attempt to explain it. Such methods are known as *post-hoc explanation methods*. LIME explains black-box models like Deep Neural Networks or Random Forest by approximating it locally by a linear classifier function. As with any approximation it may not fully represent the underlying model with difficulties explaining it for ML tasks with heterogeneous data. The hyperblocks approach avoids the deficiency of this two-step process by building an interpretable model in the first place. Building intrinsically interpretable machine learning models has been a longstanding goal in the ML field, but efforts have been hampered by the difficulty of achieving accuracy comparable to that of black-box models. This paper shows that the HB approach can build inherently interpretable models with competitive accuracy, which expands the current ML trend to build intrinsically interpretable models bypassing black boxes.

The difference of the proposed **decomposition method** with GLC-L visualization from other decomposition methods like used to explain deep neural networks (DNN) is as follows. The DNN decomposition methods search in DNN a subset of nodes that match to a meaningful concept, like bird head in the bird image. The internals how identified nodes produce a bird head is still a not transparent black box. In essence the black box model is now at the lower level. In contrast we have a full transparent and lossless GLC-L visualization in our decomposition method. It opens an opportunity to apply it to a deep neural network. It will add transparency to the current decomposition methods in DNN. It includes CLG-L visualization of the last layer of Convolutional Neural Networks (CNN) that is closely related to linear classifiers.

The relation of our hyperblock and decomposition approaches with SHAP is more distant. SHAP and similar methods do not approximate a black box model but attempt to extract model's characteristics that help a user to understand it. It is done in the form of the importance of contribution of each attribute to the output prediction. For instance, it can be that the contribution of high blood pressure is 3, and the contribution of Body Mass Index (BMI) is only 0.6 to the total predicted mortality rate 4 for a person. Other factors contribute to the total value of 4 positively or negatively [Lundberg, Lee, 2017]. If a domain expert agrees with these contributions, then it adds trust to the model while the model continues to be a black box for the user. If the domain expert disagrees with contributions produced by SHAP then it is an argument to reject the SHAP results and/or prediction model.

In the example above the contribution of the high blood pressure is 5 times greater than the contribution by BMI. However, for the hypertension induced by obesity, obesity is a cause and should be considered as more important than hypertension. This example shows that computed importance of attributes can be incorrect. Thus, it heavily depends on the method used to identify those most important attributes like SHAP. SHAP introduces assumptions that need to be verified for each given task that is not typically conducted. Those assumptions are actively discussed and criticized in literature. For us one of the most important of them is the assumption of additivity. It came from economics where monetary gain seems naturally additive but in machine learning with heterogeneous data it is not obvious [Kovalerchuk et al. 2021]. Although understanding the contribution of attributes is valuable for model explanation, making irrelevant or unwarranted assumptions about attribute importance can be detrimental.

In contrast, the hyperblocks approach introduces naturally the importance of attribute X_i in each hyperblock H as the **Importance Ratio** (**IR**) of attribute X_i: $L(X_i)/L(H(X_i))$, where $L(X_i)$ is the length (range) of the attribute X_i. and $L(H(X_i))$ is the length (range) of attribute X_i within hyperblock H. For instance, if the length of attribute X_i is 1, and its length within hyperblock H is only 0.25, then IR=4 for attribute X_i. In Figure 3b we marked the attribute with the largest ratio, considered as the most important. The advantage of this concept of importance is in its simplicity and clarity for domain experts, eliminating the need to learn and justify the complex assumptions required by alternative methods such as SHAP.

3. MULTILAYER ARCHITECTURE BASED ON DECOMPOSITION AND VISUALIZATION

This section describes a multilayer architecture that is based on function decomposition and the visualization of decompositions.

3.1. Function Decomposition

Function decomposition methods have a long history in ML [Zupan et al., 1997], signal processing like Fourier series and other domains. Many classifiers like linear discriminant functions, support vector machines (SVM), and others can be represented in an analytical mathematical form like (1) or more generally like (2), which are function decompositions:

$$G(\mathbf{x}) = w_1F_1(\mathbf{x}) + w_2F_2(\mathbf{x}) + \ldots + w_mF_m(\mathbf{x}), \quad (1)$$

$$G(\mathbf{x}) = G(H_1(\mathbf{x_1})), H_2(\mathbf{x_2})), \ldots, H_k(\mathbf{x_k})), \quad (2)$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is an n-D point, $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_k}$ are subsets of $\mathbf{x}$, e.g., $\mathbf{x}_1 = (x_1, x_2)$, $\mathbf{x}_2 = (x_3, x_2)$, w_i are coefficients/weights and F_i and H_1 are functions of the n-D point $\mathbf{x}$ or its subsets.

For instance, consider a version of (1) with 9 functions F_i and four functions H_i (**decomposition level 1**):

$$G(\mathbf{x}) = w_1F_1(\mathbf{x}) + w_2F_2(\mathbf{x}) + \ldots + w_9F_9(\mathbf{x}),$$

$H_1(\mathbf{x}) = w_1F_1(\mathbf{x}) + w_2F_2(\mathbf{x})$, $H_2(\mathbf{x}) = w_3F_1(\mathbf{x}) + w_4F_4(\mathbf{x}) + w_5F_5(\mathbf{x})$, $H_3(\mathbf{x}) = w_6F_6(\mathbf{x}) + w_7F_4(\mathbf{x}) + w_8F_8(\mathbf{x})$, $H_4(\mathbf{x}) = w_9F_9(\mathbf{x})$.

Now we can represent $G(\mathbf{x})$ by its **decomposition level 2**

$$G(\mathbf{x}) = H_1(\mathbf{x}) + H_2(\mathbf{x}) + H_3(\mathbf{x}) + H_4(\mathbf{x})$$

In an example with a linear classifier, we can have $G(\mathbf{x}) = w_1x_1 + w_2x_2 + \ldots + w_9x_9$ at the layer 1. We can group expressions w_ix_i and get layer 2 like: $Q(\mathbf{x}) = H_1(x_1, x_2) + H_2(x_3, x_4, x_5) + H_3(x_6, x_7, x_8) + H_4(x_9)$. This process can continue to build a next **level 3** like

$$G(\mathbf{x}) = Q_1(\mathbf{x_{12}}) + Q_1(\mathbf{x_{34}}), \quad Q_1(\mathbf{x_{12}}) = H_1(\mathbf{x_1}) + H_2(\mathbf{x_2}), \quad Q_2(\mathbf{x_{34}}) = H_3(\mathbf{x_3}) + H_4(\mathbf{x_4}).$$

This is a way to build a multilayer hierarchy of the subfunctions F_i, H_i, Q_i, which together finally produce the same result as the original function G. Neural Networks (NNs) at individual layers also use this form with different functions and weights that are learned at multiple layers. We will call the classifiers that use function $G(\mathbf{x})$ without decomposition **flat classifiers**. The classifiers that use decomposition will be called **multilayer/multilevel classifiers**.

This decomposition process has an important aspect relative to the NNs. In the NNs, the process starts from the postulated multilayer structure and *learn multiple functions*. The decomposition starts from a *learned flat classifier* and then structures it to multiple layers. At first glance decomposition is redundant, because it produces the same result as the flat function. In fact, it is beneficial computationally allowing parallelization of the computation of the output. However, the major benefit that we present in this paper is for visualization, which allows us to reveal hidden patterns as the case studies show in section 4. The next section presents the decomposition method with a visualization approach.

On the mathematical side, the Kolmogorov-Arnold representation theorem shows that decomposition is possible for a wide range of functions, that is every multivariate continuous function F: $[0,1]^n \rightarrow R$ can be represented as a superposition of continuous *single-variable* functions [Braun, Griebel, 2009].

3.2. Lossless visualization with multilevel architecture

Multi-level visualizations of machine learning algorithms can be found for several ML algorithms like decision trees [Kovalerchuk et al., 2024] and neural networks including deep learning networks [Samek et al., 2016; Seifert et al., 2017; Browne et al., 2018; Matveev et al., 2021]. Below we show how to represent Linear Discriminant Functions (LDFs) with the hierarchical multilayer approach. It is within the decomposition approach presented in the section above and is applicable to many other ML algorithms, which includes multilevel visualization for SVM [Huber, Kovalerchuk, 2023]. Later in this paper we will show how an LDF is combined with hyperblocks to provide interpretable models.

GLC-Linear Level 1 and 2. GLC-Linear (GLC-L) is a type of General Line Coordinates capable of discovering linear models and solving supervised learning classification tasks in 2-D [Kovalerchuk, Dovhalets, 2017; Kovalerchuk, 2018]. To produce a GLC-L visualization, coefficients c_i of a linear function $F(\mathbf{x})$ need to be normalized

to produce a new linear function $G(\mathbf{x}) = k_1x_1 + k_2x_2 + \ldots + k_nx_n$ as described below. Let $K = (k_1, k_2, \ldots, k_n)$ and $k_i = c_i / |c_{max}|$, where $|c_{max}| = \max_{i=1:n}|c_i|$. Here all k_i are normalized to the interval [-1, 1]. The following property is true for F and G:

$$F(\mathbf{x}) < T \text{ if and only if } G(\mathbf{x}) < T / |c_{max}|.$$

Figure 4 shows a visualization of GLC-L level 1 for a simple 4-D point **a** = (1, 1, 1, 1). For another point, say, **b** = (0.6, 0.8, 0.7, 0.9) the lengths of the blue vectors in Figure 4 will be adjusted their values in **b**, e.g., $x_1 = b_1 = 0.6$.

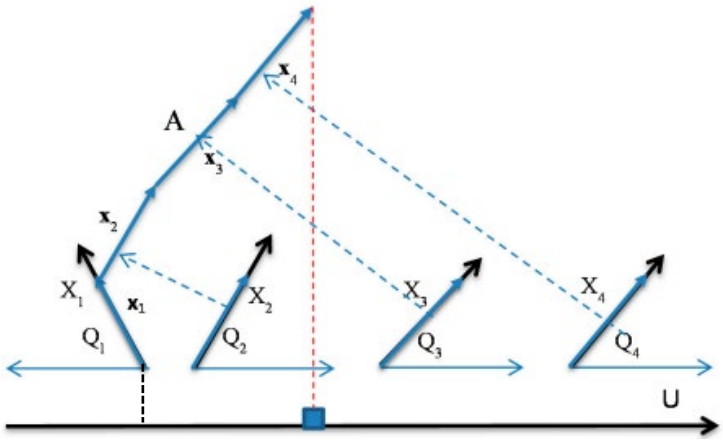
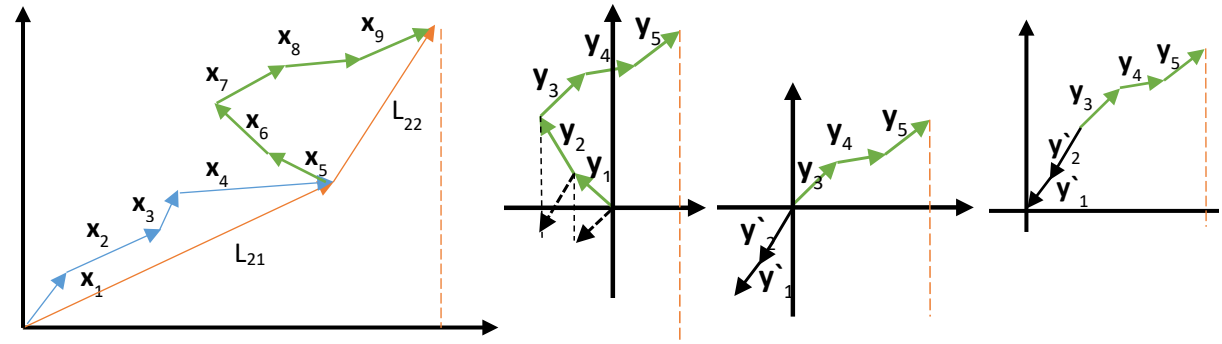


Figure 4. Example of GLC-L level 1. 4-D point **a** = (1, 1, 1, 1) in GLC-L coordinates $X_1 - X_4$ with angles (Q_1, Q_2, Q_3, Q_4). Vectors $\mathbf{x}_i$ are shifted to connect one after another and the end of the last vector is projected to the black line [Kovalerchuk, 2018]. The red projection line ends in the small blue rectangle on the horizontal coordinate U. The length of the segment on U between two dotted projection lines is a value of a linear function of these four valuables.

Below we define a new GLC-L generalization for levels 2, 3, and beyond while GLC-L level 1 was defined before in [Kovalerchuk, Dovhalets, 2017]. Figure 5a presents the concept of a new GLC-L level 2 visualization. In this figure lines L_{21} and L_{22} represent GLC-L level 2 with the blue lines $\mathbf{x}_1$-$\mathbf{x}_4$ forming L_{21} and the green lines $\mathbf{x}_5$-$\mathbf{x}_9$ forming L_{22}. Mathematically L_{21} is a sum of blue vectors and L_{22} is a sum of green vectors:

$$L_{21} = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4, \qquad L_{22} = \mathbf{x}_5 + \mathbf{x}_6 + \mathbf{x}_7 + \mathbf{x}_8 + \mathbf{x}_9,$$

which constitute **vector summation**. The benefit of GLC-L level 2 is in the **generalization** of vectors $\mathbf{x}_i$ of GLC-L level 1. It reduces details and make simpler polylines, producing patterns more visible. It is especially beneficial when many cases are visualized with fewer intersections of their polylines.



(a) Example of GLC-L level 2.

(b) GLC-Lwith negative coefficients for $\mathbf{x}_5$, $\mathbf{x}_6$.

(c) Two GLC-L with positive coefficients for all x_i.

Figure 5. Example of GLC-L visualization level 2 (a) and converting GLC-L for LDF with negative coefficients (b) to GLC-L with positive coefficients (c). In (a) blue and green lines represent level 1 and orange lines represent level 2. Two variables with negative coefficient in (b) are converted to variables with positive coefficients by reversing their vectors.

There are several ways to produce GLC-L level 2, below are some options. The first approach is *summing up* two adjacent level 1 *pairs of vectors*: $L_{21} = \mathbf{x}_1 + \mathbf{x}_2$, $L_{22} = \mathbf{x}_3 + \mathbf{x}_4$, $L_{23} = \mathbf{x}_5 + \mathbf{x}_6$ and so on. Another approach is *interactive* where a user visually groups vectors $\mathbf{x}_i$ like it is shown in Figure 5a. The advantage of an interactive method is that a user, as a domain expert, can do this in a semantically meaningful way for the domain. The next method *orders* all vectors $\mathbf{x}_i$ according to their coefficients in an LDF from the smallest one to the largest one. Typically attributes with the smallest coefficients are *least contributing* to the total value of an LDF and they can be combined into L_{21} resulting

in a relatively significant contribution. Other vectors $\mathbf{x}_i$ with large individual contributions can be kept at level 2 without grouping them: $L_{2k} = \mathbf{x}_i$.

Another advantage of level 2 is for conditions where some coefficients are negative. Here vectors with negative coefficients can be grouped with vectors with positive coefficients to produce the sum level 2 line with a positive coefficient. Line L_{22} in Figure 5a shows it, where $\mathbf{x}_5$ and $\mathbf{x}_6$ have negative coefficients but together with $\mathbf{x}_7$-$\mathbf{x}_9$ they form L_{22} with a positive coefficient. It makes simpler patterns and their analysis. Additionally, level 2 can be generalized to **level 3** and **higher levels**. For example, in Figure 5a vectors L_{21} and L_{22} can be summed up to form a level 3 vector: $L_{31} = L_{21} + L_{22}$.

Algorithm for Producing Simplified GLC-L. When all coefficients of an LDF are positive it simplifies GLC-L visualization making it easier for the user to analyze it. Below, we discuss methods to modify LDFs with some negative coefficients to make all of them positive. Generation of GLC-L level 2 resolves this issue indirectly because the coefficients of level 1 are still negative. Next, we also show explicit methods where LDFs are modified at the same level to make all coefficients positive.

In GLC-L all attributes are normalized to be within interval [0, 1], this interval can be changed to [-1, 0] by multiplying all values of the attribute x_i with a negative coefficient k_i. This will change it to $-k_i$, which is positive. This will reverse the direction of the vector x_i in the GLC-L visualization. See Figures 5b and 5c. Here the vectors were negated to produce vectors: $\mathbf{y}`_5 = -\mathbf{y}_5$, $\mathbf{y}`_6 = -\mathbf{y}_6$ and to make a substitution: $\mathbf{y}_5 + \mathbf{y}_6 + \mathbf{y}_7 + \mathbf{y}_8 + \mathbf{y}_9 = -\mathbf{y}`_5 - \mathbf{y}`_6 + \mathbf{y}_7 + \mathbf{y}_8 + \mathbf{y}_9$.

Figure 5c presents two versions of locating the origin for GLC-L coordinates. The top one separates positive and negative vectors, and the bottom one puts all vectors to the positive quadrant. The advantage of the last one is in allowing us to use the 4th quadrant to show cases of the opposite class as it is done in the GLC-L implementation in the DV2 program [GitHub, 2024]. Technically conversion from Figure 5b to Figure 5c is done by producing negated vectors $\mathbf{y}`$ with mirroring $\mathbf{y}$ vectors, reversing their directions and putting one after another one. This is shown with dotted black lines in Figure 5b.

4. MULTILAYER ARCHITECTURE BASED ON HYPERBLOCKS

4.1. Algorithms for Producing Hyperblocks of Level n

A hyperblock (**n-orthotope**) [Hayes, Kovalerchuk, 2024] is a set of n-D points $\{\mathbf{x} = (x_1, x_2, \ldots, x_n)\}$ with **center** in n-D point $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ and **side lengths** $\boldsymbol{L} = (L_1, L_2, \ldots, L_n)$ such that

$$\forall i \;\; \| x_i - c_i \| \leq L_i/2 \tag{3}$$

Figure 3b above illustrates this concept.

To make HBs of level n, the same HB building algorithms are applied as for level 1. Here we use the HB building algorithm called the General Line Coordinate Hyperblock Rules Linear (GLC-HBRL) [Huber et al., 2024]. The difference between each HB level comes from the data being used in the creation of the HBs. For HBs of level 1, **all cases** are used. For HBs of level n, **representative cases** for each HB of level $n - 1$ are used. Any algorithm capable of gathering these representative cases can be used alongside the Level n Hyperblock Creation Algorithm to create higher level HBs. Algorithm LnHC shows this process.

Level *n* Hyperblock Creation (L*n*HC) Algorithm

1. Create HBs of level n-1.
2. For each HB, select an algorithm to Create a Representative Dataset (CRD algorithm).
3. Create the representative dataset by using CRD algorithm.
4. Use the representative dataset of each HB to create level n HB with the same HB creation algorithm used in Step 1.
5. Repeat steps 2 and 3 until desired level is reached.

To get a representative dataset for the creation of higher level HBs, many algorithms can be used. For a simple representation, one can find cases corresponding to the *envelope* of each HB. This is done in Algorithm CRD1, the steps of which are below. Other methods include finding the *most average case* shown in Algorithm CRD2, *splitting each HB into thirds,* and finding the *most average case* for each HB third shown in Algorithm CRD3.

For this paper all higher level HBs are created with this algorithm. This is because using an envelope representation tended to perform better than the other methods that were tested. However, that is not to say these other data representations are ineffective. For instance, if *explainable* methods will be used instead of *unexplainable*

methods when finding the most average cases in Algorithms CRD2 or CRD3, then it will increase the interpretability of HBs created through such cases.

The benefits and deficiencies of using representative cases for higher level of HB generation are as follows. An important benefit is decreasing the number of HBs, which contain only a few cases each. This likely indicates overfitting. The first deficiency comes from using only representative cases. HBs of level 2 and higher can be less pure than HBs of level 1 because some cases of opposite classes can be between representative cases of a single class.

Defining and finding the most average cases depend on a selected similarity method of cases. While the Euclidean distance is very common, it is **not explainable** for heterogeneous data [Kovalerchuk et al., 2021], which are common in ML. Data attributes in some case studies that are presented in this paper are heterogeneous like the angles, distances, and roots of the moments of the Magic Gamma Telescope (MGT) data. Conversely, one explainable method is the **explainable threshold similarity (ETS) method** with a threshold T or a set of thresholds (T_1, T_2, …, T_n) for two n-D points **x** and **y** such that

$$|x_1 - y_1| < T_1 \ \& \ |x_2 - y_2| < T_2 \ \& \ \dots \ \& \ |x_n - y_n| < T_n. \quad (4)$$

It compares only values of each attribute avoiding uninterpretable summing up and squaring values of different attributes. The Euclidean distance also can be called a **lossy similarity method**. It sums up and squares the similarity values of individual attributes. As a result, the individual differences $|x_i - y_i|$ are lost and not restorable from the Euclidean distance. Many other similarity methods are lossy distance techniques, which may unexplainably compress n-D heterogeneous data into a single value [Huber et al., 2024]. In contrast, the explainable threshold similarity ETS method with a set of thresholds (T_1, T_2, …, T_n) allows to control the upper limit of $|x_i - y_i|$ by T_i.

Algorithm CRD1: Envelope Case Finder

1. Get all cases in each HB.
2. For each attribute of each case, check if the value is equal to the minimum or maximum value of the HB.
   a. If yes, add a given case to the set of envelope cases for the HB and stop looking for cases equal to the current minimum or maximum value.
   b. If not, then continue searching.
3. Return Min-envelope case set for the given HB and Max-envelope case set for the given HB.

Pseudo-code for **Algorithm CRD1: Envelope Case Finder**:

```
Algorithm CRD1: Envelope Case Finder
Input: A set of hyperblock (HBs)
Output: Min_Envelope case set for each HB, Max_Envelope case set for each HB
For each HB in HBs:
  envelope_cases = empty set
    For each case in HB:
    added = False
    For each attribute in case:
      current_value = case[attribute]
      if not min_found[attribute] and current_value == min_values[attribute]:
        Min_envelope_cases.add(case)
        min_found[attribute] = True
        added = True
        break
      elif not max_found[attribute] and current_value == max_values[attribute]:
        Max_envelope_cases.add(case)
        max_found[attribute] = True
        added = True
        break
    if added:
      continue  # Proceed to next case
  return envelope_cases
Example of min case: data = {'temperature': [25, 30, 28], 'humidity': [60, 65, 55]},  result:  {'temperature': 25,  'humidity': 55)}
```

Algorithm CRD2: Average Case Finder

1. For a given HB, create an artificial case using the average value for each attribute.
2. Using some distance technique, find the closest case to the artificial case created in Step 1.
3. Return the case found in Step 2 as a representative set.

Algorithm CRD3: Average Case Finder with Strips

1. Split a given HB into thirds on the range of each attribute.

2. For each HB third, create an artificial case using the average value for each attribute.
3. Using some distance technique, find the closest case to each artificial case.
4. Return the cases found in Step 3 as a representative set.

Additionally, optimizing our HB creation algorithms can also improve results. One way of doing this is by basing the construction of HBs on Decision Trees. Decision Trees are some of the most widely used explainable supervised learning classification techniques. Their construction through recursive partitioning to create the most homogeneous subsets of data can be mirrored in the HB creation process. Algorithm DT-HB shows this process.

Decision Tree Hyperblocks (DT-HB)

1. Create HBs of level *n*.
2. Find the largest HB created and store it.
3. Get all data not in the largest HB.
4. Use the data from Step 3 to create more HBs.
5. Repeat Steps 2-4 until all data are classified.

4.2. Interactive Algorithm for Creation of Pure Intervals

Figure 6 shows the overlap area of the Wisconsin Breast Cancer (WBC) dataset [Wolberg et al., 1996] in GLC-L visualized in the DV2 program [DV2, 2024]. This is the area between the leftmost and rightmost misclassified points within the visualization. The green dotted vertical line indicates the discrimination line of two classes. The yellow vertical lines on the right and the left side indicate limits of the overlap area. Figure 6 shows *pure subareas* colored with their respective classes in the Linear Discriminant Function (LDF) overlap area. These areas can be interactively captured by a user using the GLC Interactive Rules Linear (GLC-IRL) algorithm in the DV2 program [DV2, 2024]. Next a user can analyze these pure areas. If the areas are too narrow from the user's viewpoint, the user can skip them to avoid overfitting. For instance, a user can decide that two magenta areas are too narrow. A user also can relax the requirement of fully pure areas but can search for areas with a purity over some threshold, say, 90%. These methods allow users to interactively create higher level intervals by merging rectangles shown in Figure 6. The areas that the user consider as pure enough will be classified by the **constrained LDF** in the form:

$$\text{If } E_{11} \le F(\mathbf{x}) \le E_{12} \text{ or } E_{21} \le F(\mathbf{x}) \le E_{22} \text{ or } \dots \text{ or } E_{k1} \le F(\mathbf{x}) \le E_{k2} \text{ then } \mathbf{x} \in \text{Class 1},$$

where E_{i1} is the start and E_{i2} is the end of the interval.

These simple rectangles in visualization correspond to quite complex areas in the n-D space. For instance, if a classifier is defined by a linear function $F(\mathbf{x})=5x_1+7x_2+4x_3+x_4+2x_5+3x_6+x_7+4x_8+8x_9$ then a rectangle in GLC-L visualization can represent a quite complex area in 9-D space like $4<5x_1+7x_2+4x_3+x_4+2x_5+3x_6+x_7+4x_8+8x_9<7$. This is an important advantage of the proposed lossless visualization allowing users without advanced knowledge of machine learning to directly manipulate with high-dimensional data and ML models.

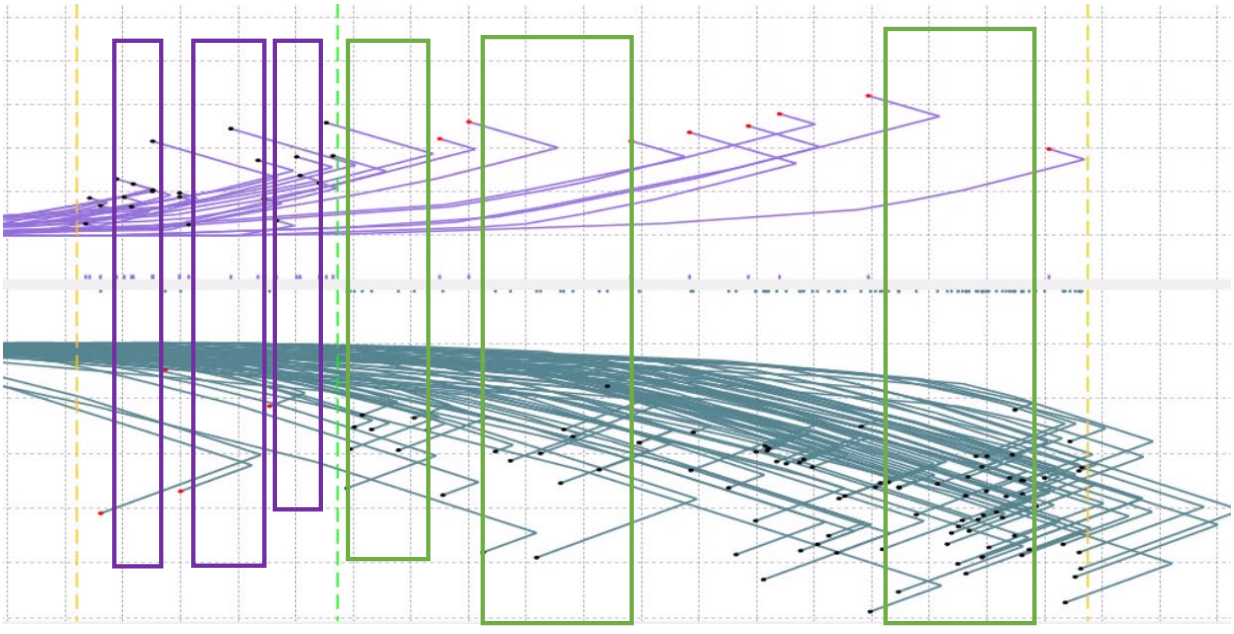


Figure 6. LDF overlap area with pure subareas colored with respective classes visualized in GLC-L. Yellow dotted lines are limits of the overlap area. The green dotted line is the LDF classification threshold. Green rectangles show pure subareas of the green class and blue rectangles show pure subareas in the blue class within the overlap area limited by yellow lines. Small dots just above the middle horizontal line show projections of blue lines (values of LDF). Small dots just below the middle horizontal line show projections of green lines (values of LDF).

4.3. Parallelization Algorithm

It is important to reduce the time spent generating HBs for large datasets. One way to do this is with the parallelization algorithm presented below, where each step can be run in parallel. Figure 7 shows a diagram of this process. The last step 4 in Figure 7 optimizes the set of resulting hyperblocks by deleting duplicated hyperblocks with Parallel Interval Merger Hyperblock Algorithm (PIMHyper).

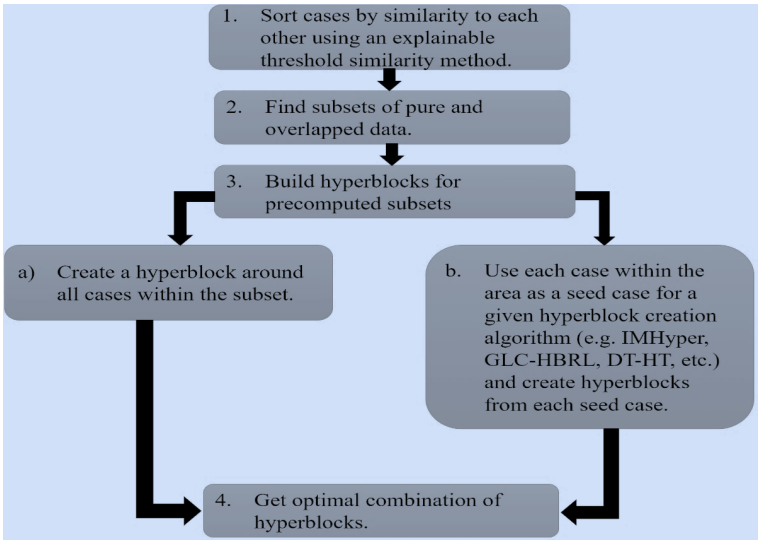


Figure 7. Parallel Hyperblock Creation Algorithm outline: (1) sorting cases, (2) finding pure and overlapped data, (3) building HBs using subsets and (4) optimizing the set of hyperblocks by reviving duplicates (Parallel Interval Merger Hyperblock Algorithm, PIMHyper).

5. CASE STUDIES

This section presents several case studies that illustrate the Multilayer Decomposition and Hyperblock Approaches and Algorithms. The cases studies 1 and 2 are on images (MNIST handwritten digits) and the cases studies 3-5 are on tabular data.

5.1. Case study 1: Multilayer Decomposition Approach --Level 2 GLC-L with MNIST digits 0 and 1

The MNIST dataset is widely used as a benchmark in ML for testing image classification models, particularly due to its simplicity and the accessibility of its grayscale, 28x28 pixel images of handwritten digits (0-9). Its popularity in ML research stems from its large size, high diversity, and relevance to real-world applications which make it an excellent resource for exploring and validating models before applying them to more complex datasets.

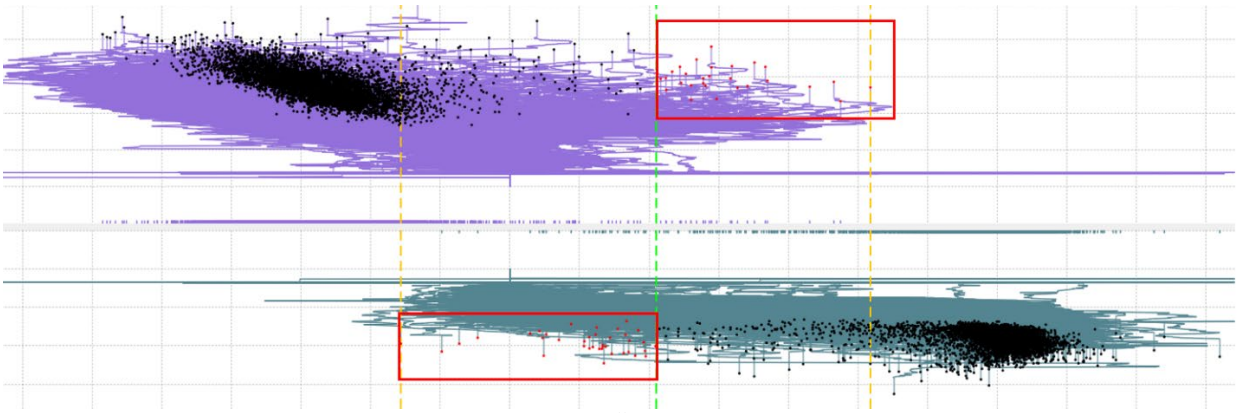
This case study analyzes the MNIST dataset **digits 0 and 1** with 784 features and 12,665 instances of the two classes [MNIST, 2024]. Figure 8a shows a GLC-L visualization of these data with 99.48% accuracy. Table 2 shows a confusion matrix summarizing the performance of this visualization. Here the visualization is weakened by the density of cases, and it is difficult to analyze any specific case. To improve this, the visualization can be focused to only the overlap area. This is the area between the leftmost and rightmost misclassified points within the visualization [Huber et al., 2024] and contains only 559 cases. This removes most confidently classified data, which require less analysis. Figure 8b shows the same GLC-L visualization but with only the overlap area.

Table 2. MNIST digits 0 and 1 confusion matrix for GLC-L.

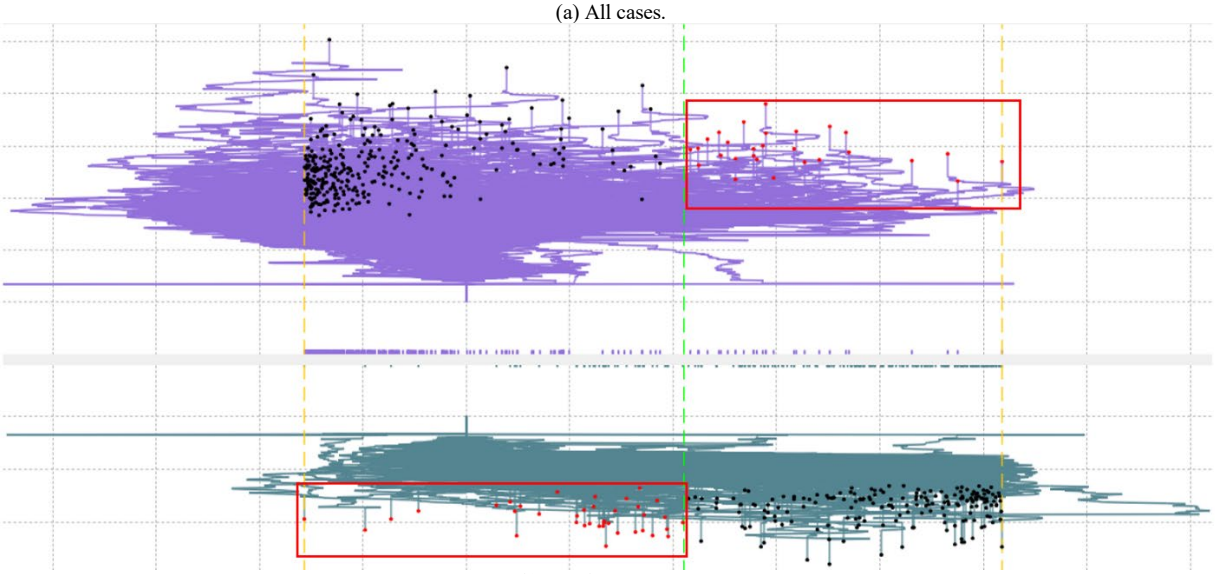
	Predicted Class	
Actual class	0	1
0	5,895	28
1	38	6,704

By visualizing only the overlap area the visualized data can be reduced from 12,665 cases to 559 cases (4.41%). This allows the structure and patterns within the data to become more apparent. For instance, it is visible that the dense

green blob is shorter than the dense purple blob. By moving from the 1-D projection used by the LDF to 2-D rectangle properties, the accuracy of classification can be improved. In Figures 8a and 8b this is captured by red rectangles.



(a) All cases.



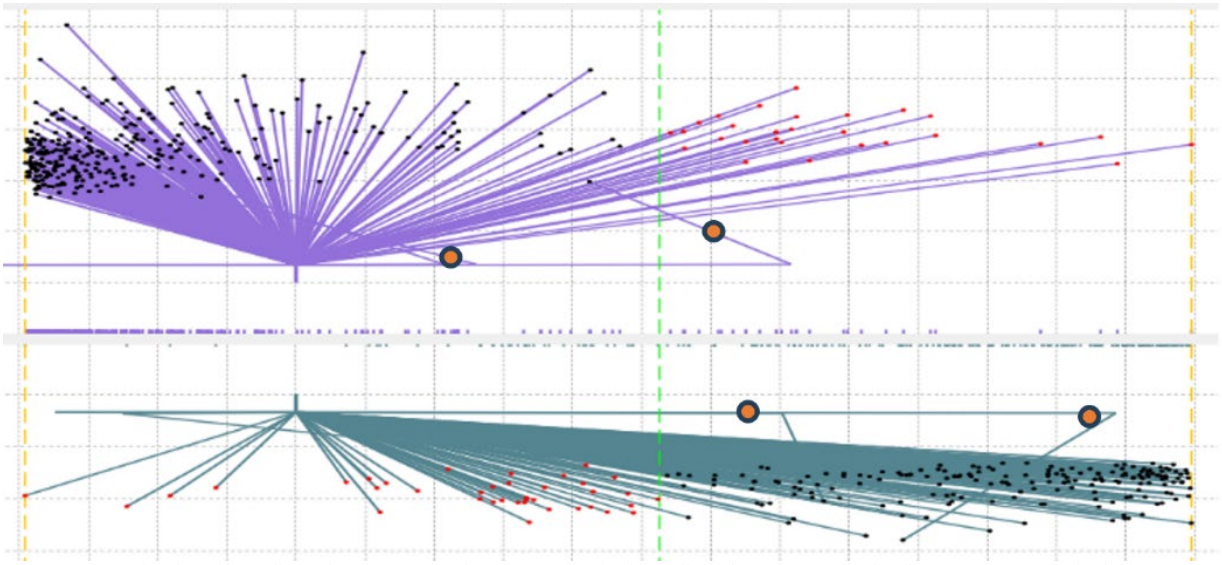
(b) Overlap area cases.

Fig. 8. MNIST dataset digits 0 and 1 visualized in GLC-L. Green line – class discrimination line. Yellow lines – overlap area bounds. Red rectangles show the benefits of improved classification with 2-D rectangles over 1-D projection.

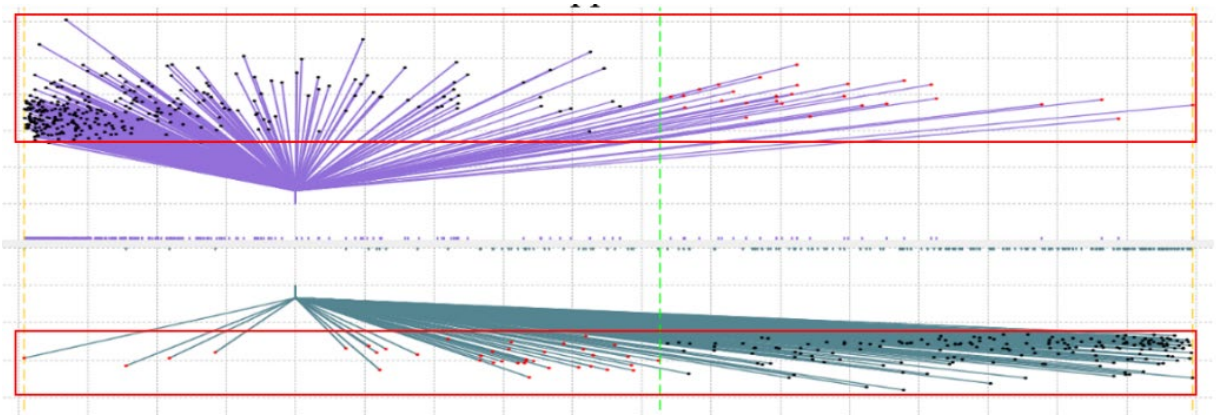
It should be noted that this property is less apparent in Figure 8a with all data visualized. Then, to further simplify the visualization GLC-L level 2 can be applied to the overlap data. Figure 9 shows various examples with GLC-L level 2 with vector summations of *least contributing attributes* as defined in section 2.3. In each visualization the threshold for being a least contributing attribute is loosened to create a more *generalized visualization*.

Figure 8a shows all data in GLC-L and Figure 8b shows only data in the overlap area. With these visualizations, it is difficult to distinguish between individual cases. In Figure 9a, by generalizing all attributes equal to 90°, individual cases become easier to distinguish.

This is a move to GLC-L level 2 visualization, where in each polyline, the segment that has 90° is joined with the adjacent segment in the same way as explained in Figure 5a. This being easier to distinguish remains true for each successive visualization in Figure 9. As more attributes get generalized to a single segment, the easier it becomes to distinguish between individual cases.



(a) GLC-L level 2 with attributes between 88° and 92° generalized. The outliers shown as orange dots became apparent in both classes.



(b) GLC-L level 2 with attributes between 87° and 93°generalized.

Figure 9. MNIST dataset digits 0 and 1 overlap area visualized with GLC-L level 2. Red rectangles show the benefits of improved classification with 2-D rectangles over 1-D projection in Level 2 generalization.

Note that the data can be overgeneralized, e.g., in Figure 9a, many outliers are visible and are marked with orange circles. These outliers are lost and are not visible in Figure 9b, where data are overgeneralized. However, this overgeneralization is useful too, making the difference between the height of the cases clearer as the red rectangles show. Additionally, both classes show that negative and positive attributes are cut out or become less dense and positive. This shows simplification in GLC-L level 2 in Figure 9 that can be useful for the user.

In Figure 9b, the most significant cut out was for the purple class on the left relative to Figure 9a. It allows us to analyze the impact of positive and negative contributions. In Figure 9b, the yellow arrows show that classes are going in opposite directions from their dense areas. Thus, visual exploration helps to find the impact of individual and groups of attributes. In summary this case study shows the benefits of multilayer modeling of LDF with GLC-L visualization.

It allows at the level 1:

- filter out from the further pattern discovery most of the cases that LDF correctly classified and at level 2:
- concentrate analysis on a minority of cases in the overlap area,
- discover difference in 2-D distributions of the cases of two classes in the overlap area,
- discover outliers that are hidden at level 1,
- decrease clutter in visualization with opening abilities to analyze individual cases,
- discover the dominant impact of positive and negative contribution of attributes to each class after individual cases became visible.

Note that visualization at level 2 does not change the mathematical result of the LDF. It produces the same output points, only visualization is simplified allowing us to see the pattern much clearer. In contrast, the generalization of mathematical models typically approximates the function, while GLC-L level 2 fully preserves the function.

5.2. Case Study 2: Multilayer Hyperblock Approach -- Level 2 GLC-L with MNIST digits 2 and 7

In this section, the MNIST dataset has been specifically limited to digits 2 and 7. This choice was driven by the practical consideration that processing the full dataset of 70,000 cases is time intensive. Digits 2 and 7 were chosen due to their visual similarity, which makes them more challenging to differentiate accurately, providing an ideal test for assessing the interpretability and robustness of the proposed models. By focusing on this subset, this paper investigates how well HBs handle larger amounts of data, offering insights that could be extended to even larger, more diverse datasets in future studies.

The MNIST dataset subset of digits 2 and 7 consists of 12,223 cases, with 5,958 instances of digit 2 and 6,265 of digit 7. For model evaluation, these cases were divided into 10 equal folds of 1,222 cases each to perform 10-fold cross-validation, ensuring robust performance assessment. HBs were generated using the Parallel Interval Merger Hyperblock Algorithm (PIMHyper). The largest HB of each class from Fold 1 of Table 3 can be seen in Figure 10.

Table 3: Accuracy of Hyperblocks with 10-fold cross validation for MNIST.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	AVG
Level 1 HBs	99.08	98.25	99.36	99.21	99.37	99.42	99.18	98.12	91.33	99.74	98.31
Level 2 HBs	99.68	99.33	99.61	99.59	99.62	98.64	99.76	99.30	99.48	99.35	99.44
Percent Increase	0.61	1.10	0.25	0.38	0.25	-0.78	0.58	1.20	1.05	-0.39	1.15

The first level of HBs was highly detailed. This added to the classification coverage of the HBs, at the cost of increased model complexity and reduced performance on test data due to overfitting. To reduce model complexity and enhance interpretability and accuracy, a second level of HBs was created by merging and generalizing the initial set of HBs using a representative data subset of the initially generated Level 1 HBs made with the Envelope Case Finder algorithm described above. This Level 2 generalization resulted in a more streamlined and manageable HB structure, making the model both easier to interpret, due to the smaller number of HBs, and less prone to overfitting. Additionally, the increased generalization from Level 2 HBs contributed to an improvement in classification accuracy, demonstrating the effectiveness of HB generalization in achieving a balance between accuracy and interpretability.

Table 3 presents the results from the initial generation of Level 1 and Level 2 HBs, showcasing the impact of increasing generalization on both accuracy and model complexity. Level 1 HBs achieved an average accuracy of **98.31%,** indicating that the fine-grained HBs closely matched the training data. By advancing to Level 2 HBs, this accuracy increased by 1.15%, reaching an average accuracy of **99.44%.** This improvement suggests that the additional generalization achieved by merging and simplifying HBs at Level 2 reduces overfitting, resulting in a model better suited to unseen data.

Beyond accuracy, a key benefit of increasing to Level 2 HBs is the significant reduction in the number of HBs, as detailed in Table 4. On average, the number of HBs decreased by 31.43%, dropping from **304** Level 1 HBs to **209** Level 2 HBs. This reduction in the number of HBs greatly simplifies the model, making it easier for domain experts to interpret and verify each rule. This is especially true when compared to the vast number of parameters typically found in convolutional neural networks (CNNs). For instance, a CNN model like the widely used VGG16 has over 138 million parameters, while ResNet50 has around 23 million parameters [Keras, 2025]. These millions of parameters, distributed across layers of convolutions, filters, and weights, make CNNs highly effective but inherently complex and challenging to interpret. In contrast, the HB model for the MNIST digits 2 and 7 classification task generated only 327,712 rules—significantly fewer than the millions of parameters in CNNs in addition to being in the meaningful numeric attributes immediately interpretably by a domain expert.

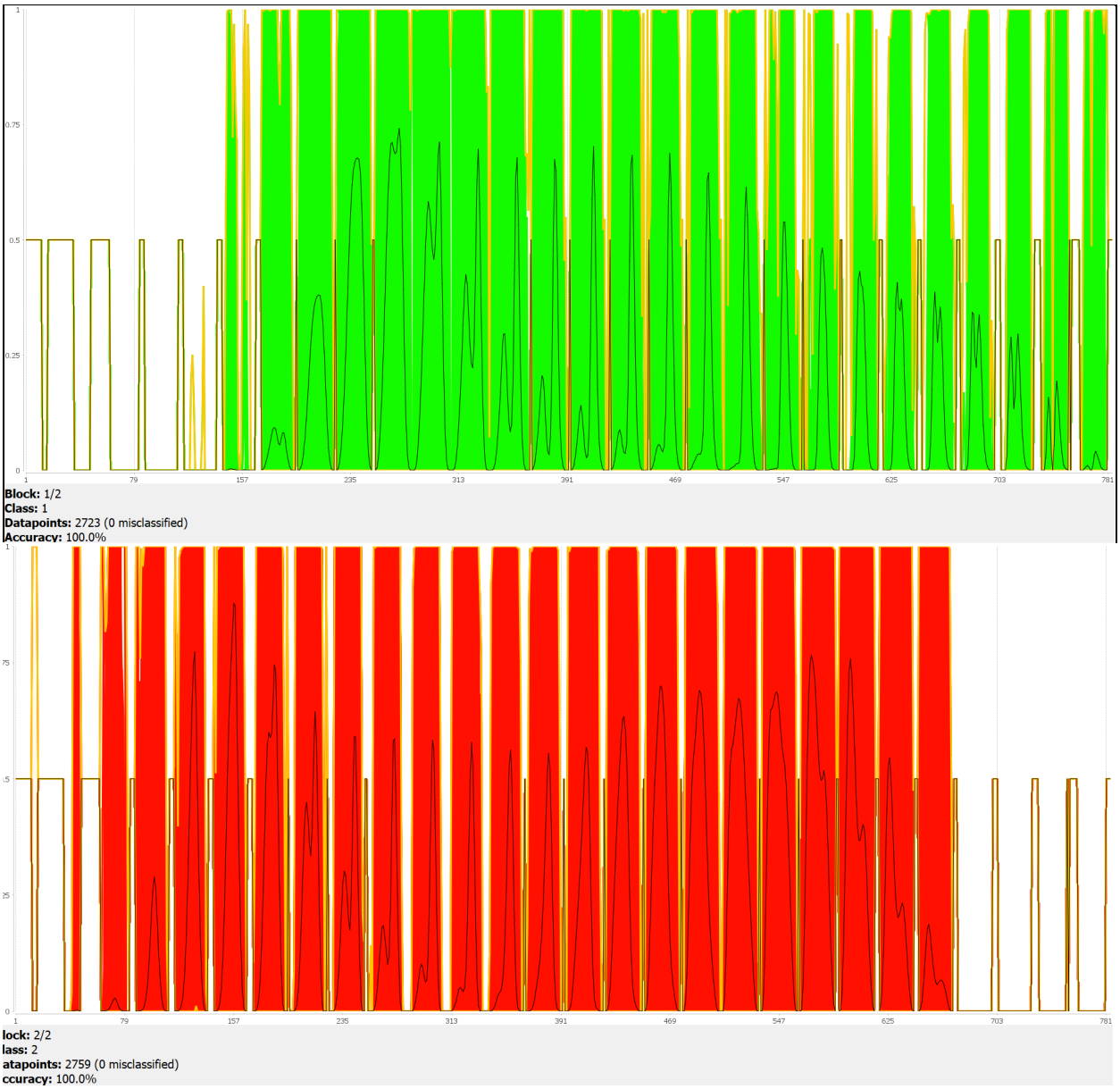


Figure 10. Largest hyperblock of each class for the MNIST digits 2 and 7 dataset visualized in Parallel Coordinates with 100% accuracy. The green hyperblock classifies 2,723 cases as digit 7 (43.46% of all cases of digit 7) and the red hyperblock classifies 2,759 cases as digit 2 (46.31% of all cases of digit 2). There is heavy occlusion in this visualization due to the large number of cases within each hyperblock. Individual cases are indistinguishable from each other. Green lines are cases of digit 7 and red lines are cases of digit 2. The black lines are the average case within each hyperblock.

Table 4. Number of Hyperblocks with 10-fold cross validation for MNIST.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	AVR
Level 1 HBs	276	260	328	337	318	273	327	318	290	315	304.2
Level 2 HBs	219	211	223	188	240	190	199	204	207	205	208.6
Percent Decrease	20.65	18.85	32.01	44.21	24.53	30.40	39.14	35.85	28.62	34.92	31.43

This complexity decrease not only improves interpretability but also decreases the cognitive load to understand the model's decision-making process. Each HB rule directly characterizes a decision boundary, letting domain experts to review and understand classifications without the opaque, layered structure of a CNN. Consequently, the HB approach provides a transparent, manageable alternative for tasks where interpretability is essential, demonstrating that powerful classification models can be both accurate and concise.

The largest decrease was observed in Fold 4 in Table 4, where the number of HBs dropped from 337 at Level 1 to 188 at Level 2—a reduction of 44.21%. In contrast, Fold 2 experienced the smallest decrease, from 260 Level 1

HBs to 211 Level 2 HBs, or an 18.85% reduction. This streamlining is highly beneficial for domain experts who must review each HBs to ensure the interpretability and coherence of the model's rules. The software allows to zoom and pan data in Parallel Coordinates for this. The reduced number of HBs makes it more manageable to confirm that each rule is logical, reducing the cognitive load required to validate the model's classification logic.

It is important to note that, while the HBs achieved high accuracy, they **did not fully capture** all test data points during evaluation. As shown in Table 5, the classification coverage of test data varied between Level 1 and Level 2 HBs, with Level 2 HBs displaying a modest decrease in coverage. Specifically, the Level 1 HBs classified an average of **89.62%** of the test data, whereas Level 2 HBs classified **85.88%**, resulting in a 4.17% decrease in classification coverage. This reduction can be attributed to the higher degree of generalization applied in Level 2, which consolidates multiple smaller HBs into larger, more inclusive ones. While this consolidation improves the model's interpretability and slightly enhances accuracy, it inevitably reduces the number of HBs, limiting the model's capacity to account for every nuance in the test data. In particular, the reduced number of HBs at Level 2 may leave certain test cases outside the decision boundaries, resulting in unclassified points. This trade-off reflects a common balance between model simplicity and coverage, where greater interpretability and generalization are achieved at the expense of exhaustive classification. For domain experts, this outcome means that Level 2 HBs provide a highly interpretable model with robust accuracy, though some outlier data points may fall outside its classifications.

Table 5. MNIST test data classified by hyperblocks with 10-fold cross validation.

Percent Classified	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	AVG
Level 1 HBs	89.85	91.33	88.87	88.79	88.79	88.79	89.77	88.38	91.33	90.26	89.62
Level 2 HBs	86.99	84.21	87.07	87.32	84.12	83.80	87.72	87.97	86.42	83.14	85.88
Percent Decrease	3.18	7.80	2.03	1.66	5.26	5.62	2.28	0.46	5.38	7.89	4.17

To improve the classification rate for outlier data points and ensure a higher percentage of test cases are accurately classified, **k-Nearest Neighbor** (k-NN) HBs were implemented in conjunction with the Explainable Threshold Similarity (ETS) method. The ETS method assesses similarity by measuring the distance between data points according to predefined thresholds, allowing the model to maintain interpretability while classifying ambiguous cases. Using k-NN HBs with ETS ensured that each 10-fold cross validation fold achieved 100% classification coverage, addressing the limitations seen with the standard Level 1 and Level 2 HBs, where some cases were left unclassified.

Table 6. Comparison of k-NN on hyperblocks with standard machine learning classifiers using 10-fold cross validation for MNIST.

Model	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	AVG
DT	98.12	97.71	98.45	98.12	98.12	97.87	98.28	97.71	97.87	98.53	98.08
SGD-L	98.94	98.77	99.10	98.28	98.28	97.71	98.69	97.87	98.53	98.20	98.44
NB	98.12	97.95	97.63	98.20	98.12	97.38	98.20	97.63	97.38	98.45	97.91
SVM	99.51	99.35	99.51	99.43	99.26	99.51	99.43	99.35	99.18	99.43	99.40
k-NN	99.18	98.94	99.43	99.02	99.02	99.35	99.02	98.77	99.26	99.18	99.12
LR	99.35	98.77	99.18	98.77	98.36	98.69	98.36	97.87	98.53	98.77	98.67
LDA	98.36	98.61	98.53	97.71	98.04	98.28	98.28	97.46	98.04	98.28	98.16
MLP	99.51	99.43	99.51	99.26	99.43	99.43	99.43	99.26	99.43	99.43	99.41
RF	99.26	99.10	99.18	99.43	99.10	99.10	99.18	98.77	99.26	99.51	99.19
CNN	99.67	99.43	99.35	99.59	99.18	99.67	99.18	99.59	99.59	99.35	99.46
k-NN HBs	99.01	98.16	99.23	99.11	99.29	99.30	99.09	98.00	98.41	99.64	98.92
Level 2 k-NN HBs	99.61	99.25	99.44	99.47	99.54	98.36	99.68	99.17	99.37	99.05	99.29
St. Dev.	0.005	0.005	0.006	0.006	0.006	0.007	0.005	0.007	0.007	0.005	0.005

Table 6 displays the performance results of the **k-NN HB**s at Levels 1 and 2 alongside nine additional standard ML classifiers from Python's scikit-learn library [Pedregosa et al., 2011] and a CNN from [Preda, 2018]. In this table all interpretable models are highlighted in green. In this case, more models are interpretable because the MNIST dataset has homogenous attributes (pixel intensities). The top four classifiers were the CNN with an average accuracy of **99.46**%, MLP with an average accuracy of **99.41**%, SVM at **99.40**%, and Level 2 k-NN HBs with **99.29**% accuracy. Although the Level 2 k-NN HBs slightly underperformed compared to the CNN, MLP, and SVM, they far exceeded the performance of the next most interpretable model, Decision Trees (DT), which averaged an accuracy of **98.08**%. This result demonstrates that Level 2 k-NN HBs achieve a strong balance between performance and interpretability, closely approaching the accuracy of top-performing models while retaining explainability. For domain experts, this

means k-NN HBs provide a reliable solution that prioritizes interpretability, transparency, and control over each classification, ensuring that even high-performing models can be understood and adjusted as needed.

5.3. Case study 3: Multilayer Hyperblock Approach -- Level 2 hyperblocks for WBC data

Figure 11 shows **26 level 1 HBs** created for the WBC data using the GLC-HBRL HB creation algorithm. These HBs cover all 683 cases with 99.9% accuracy. However, 13 HBs of these 26 HBs contain a single case, this means these HBs are overfit. In contrast, Figure 12 shows the second level of the same HBs created using data from algorithm CRD1: Envelope Case Finder. In the second level, there are **13 HBs instead of 26** with only one HB containing a single case. The accuracy remains similar at 99%.

This shows the potential to minimize overfitting with level 2 HBs. Additionally, for the user, analyzing 13 HBs is much easier than 26. However, of the 683 cases in the WBC dataset, **92 were not picked** up by the level 2 HBs. This means only 86.5% of the WBC data is represented by the level 2 HBs. A simple solution to this is shown in Figure 13 where the 92 missing cases are used to separately create two more level 2 hyperblocks. In this way the 26 level 1 hyperblocks can be reduced to **15 level 2 hyperblocks**.

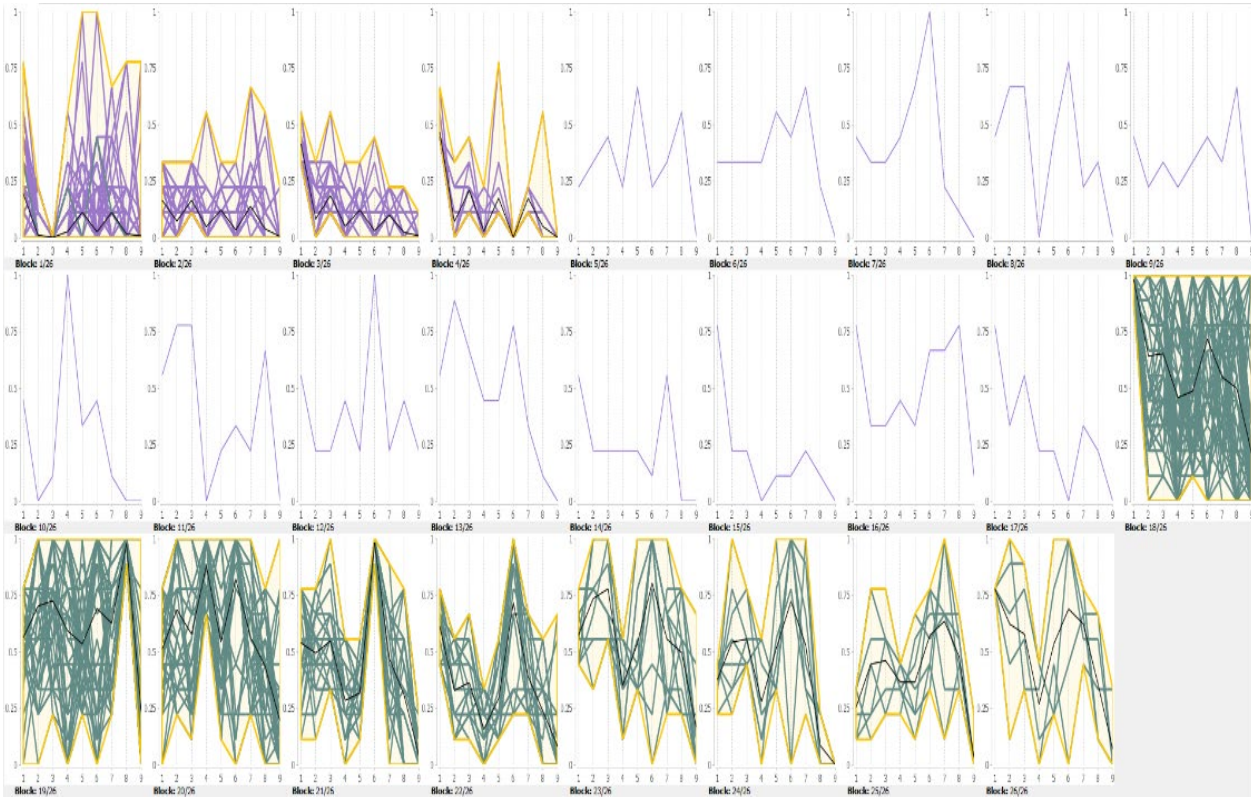
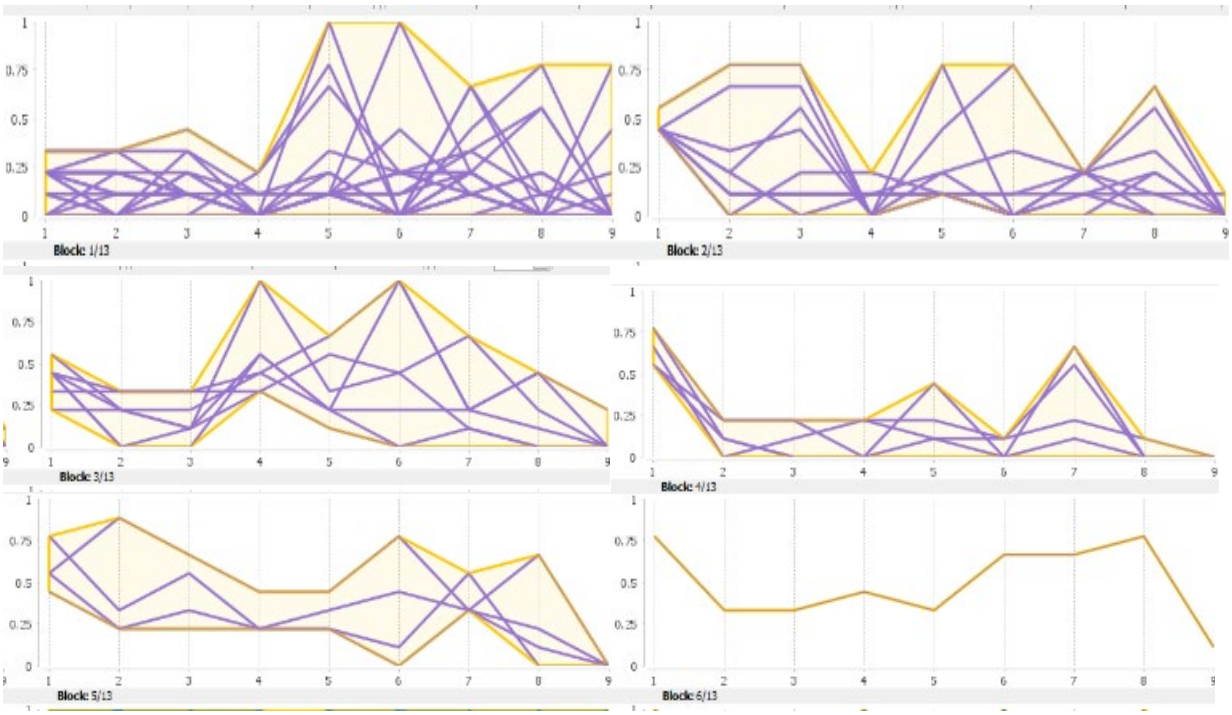
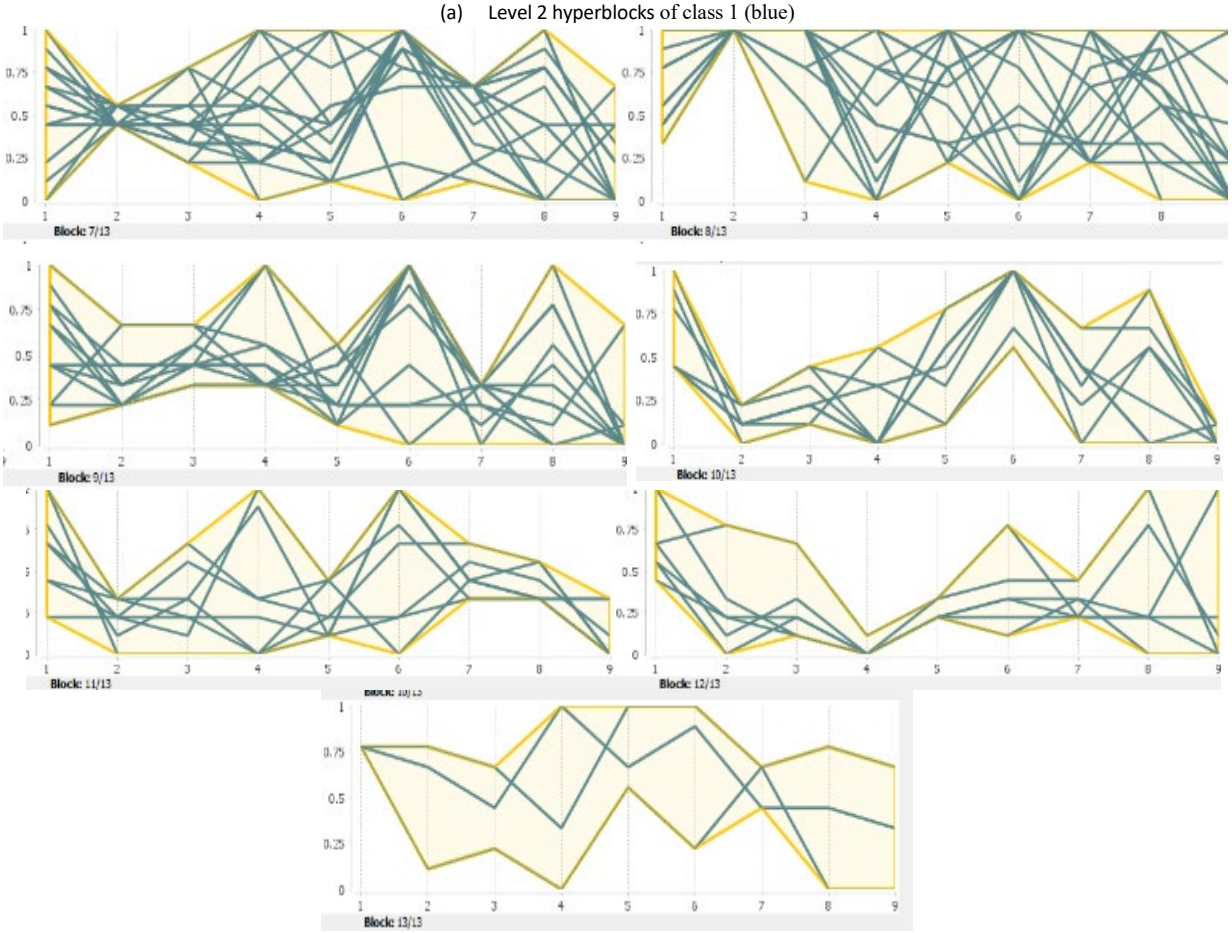


Figure 11. 26 level 1 hyperblocks for the WBC data as a set of interpretable breast cancer classification models.



(a) Level 2 hyperblocks of class 1 (blue)



(b) Level 2 hyperblocks of class 2 (green)

Figure 12. 13 level 2 hyperblocks for the WBC data as a smaller set of interpretable breast cancer classification models relative to level 1.

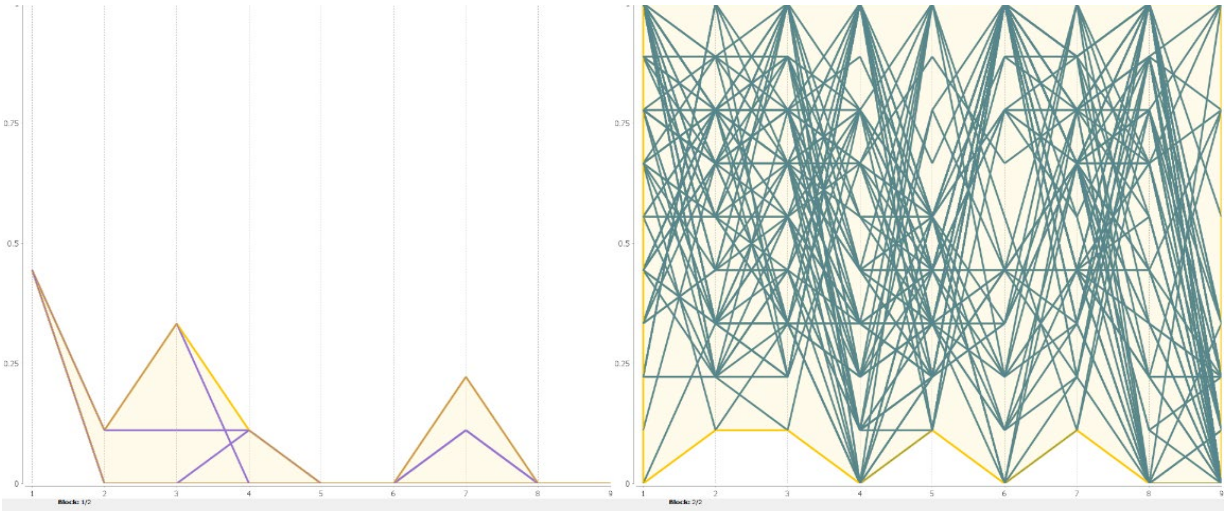


Figure 13. Two hyperblocks created separately by using remaining 92 cases that are not in hyperblocks from Fig. 12 demonstrating abilities to flexible deal with remaining cases.

5.4. Case study 4: Multilayer Hyperblock Approach -- Level 3 hyperblocks for MGT data

This case study analyzes Magic Gamma Telescope (MGT) Dataset [Bock, 2007], with 10 features and 19020 instances of 2 classes. The features are listed in Table 7.

Table 7. Features of Magic Gamma Telescope (MGT) Dataset.

1. fLength: continuous # major axis of ellipse [mm]
2. fWidth: continuous # minor axis of ellipse [mm]
3. fSize: continuous # 10-log of sum of content of all pixels [in #phot]
4. fConc: continuous # ratio of sum of two highest pixels over fSize [ratio]
5. fConc1: continuous # ratio of highest pixel over fSize [ratio]
6. fAsym: continuous # distance from highest pixel to center, projected onto major axis [mm]
7. fM3Long: continuous # 3rd root of third moment along major axis [mm]
8. fM3Trans: continuous # 3rd root of third moment along minor axis [mm]
9. fAlpha: continuous # angle of major axis with vector to origin [deg]
10. fDist: continuous # distance from origin to center of ellipse [mm]
11. class: g,h # gamma (signal), hadron (background)
g = gamma (signal): 12332
h = hadron (background): 6688

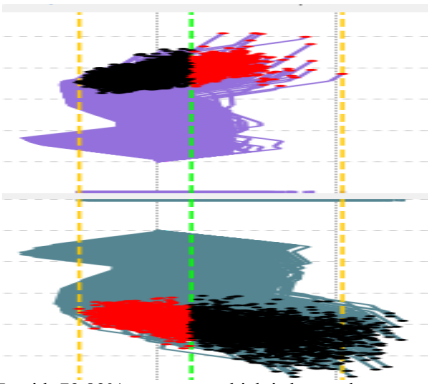


Figure 14. MGT dataset visualized in GLC-L with 79.83% accuracy, which is better than reported in [42[for the decision tree (77%). This interactive visualization allows shifting the green classification line to explore the changes accuracy and overlap areas identified by the yellow lines.

Figure 14 visualizes these data in GLC-L using the DV2 program. Using the GLC-HBRL algorithm to create HBs for the LDF, an explainable set of rules that match the LDF can be created. This is shown in Figure 14. On its own,

LDFs are not explainable for heterogeneous data [Huber et al., 2024; Kovalerchuk et al., 2021]. This is like the Euclidean distance mentioned above. The advantage of this approach is the potential for explainable HB rules with higher accuracy than the LDF. A disadvantage is that attempting to modify HBs to match a LDF can result in many HBs containing only a single case [Huber et al., 2024]. In experiments with the MGT data, 3,787 HBs were generated with accuracy 99.2%. These HBs can be seen in Figure 15. Of these HBs, the largest one contains only 269 cases while *3,253 of these HBs contain only a single case*. Like Case Study 3, it is obvious that level 1 HBs are overfitting the data to explain the given LDF. By creating level 2 HBs, the number of HBs can be reduced from 3787 to 437. These HBs can be seen in Figure 16. This is an 88.5% reduction in the number of HBs. Of these level 2 HBs, the *largest contains 8,449 cases* and *none contains only a single case.* However, the accuracy of these level 2 HBs drops to 83.3%. Nonetheless, this shows that the creation of level 2 HBs can be used to drastically reduce overfitting at the cost of some accuracy. Moreover, it is significantly easier to analyze 437 sets of HB rules in comparison to 3,787.

In Kaggle [Lee, 2025] reported the following accuracy for MGT data: 77% (Decision Tree), 87% (Random Forest), 65% (Naive Bayes), 85% (Neural Network). Extensive Monte Carlo simulation with multiple data spits to training and validation data for Randon Forest shows the average accuracy across all trials **85.2%** [Lee, 2025]**.** This result is stable regardless of splitting the data with a narrow 95% Confidence Interval (84% - 86%). For a wider 95% Confidence Interval (78%- 89%) data variability significantly affects model performance. Our accuracy of 83.3% for Level 2 HBs is consistent with these results. Note that out of four models in [Kotp et al, 2022] only a decision tree is fully interpretable with accuracy of 77%, which is lower than our accuracy of 83.3%. Additionally, the visualization of our model in parallel coordinates (see Figures 15 and 16) is fully accessible to end users, enabling in-depth analysis.

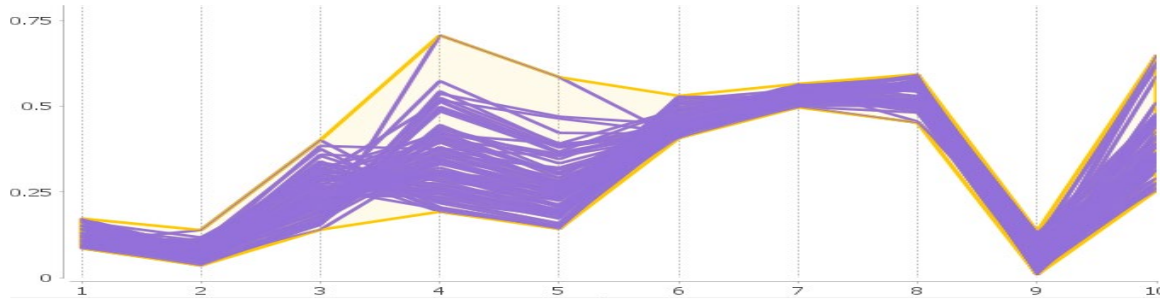


Figure 15. Largest level 1 hyperblock for the MGT dataset.

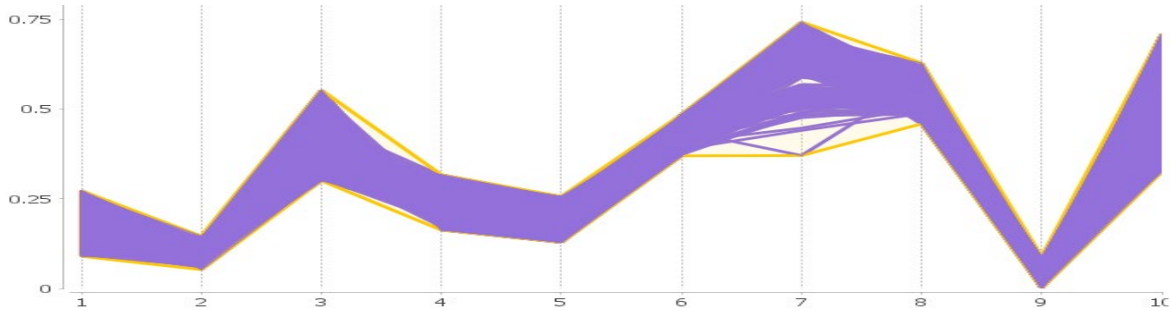
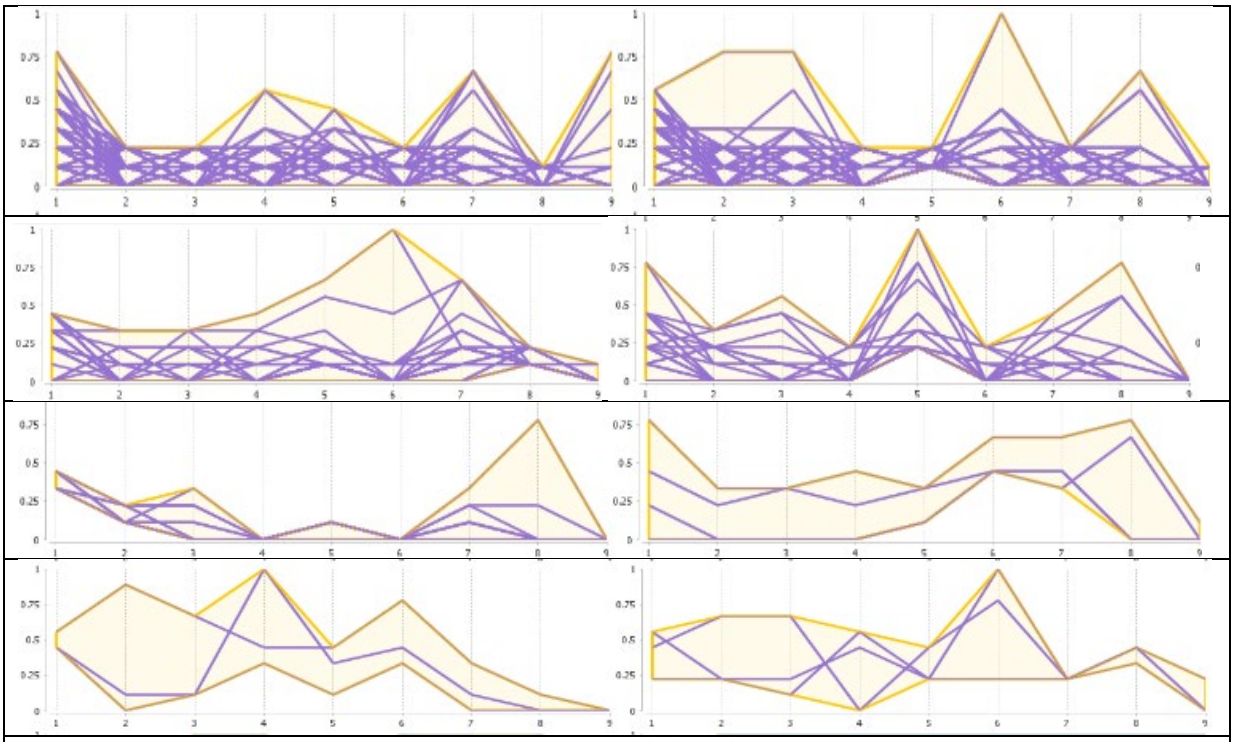


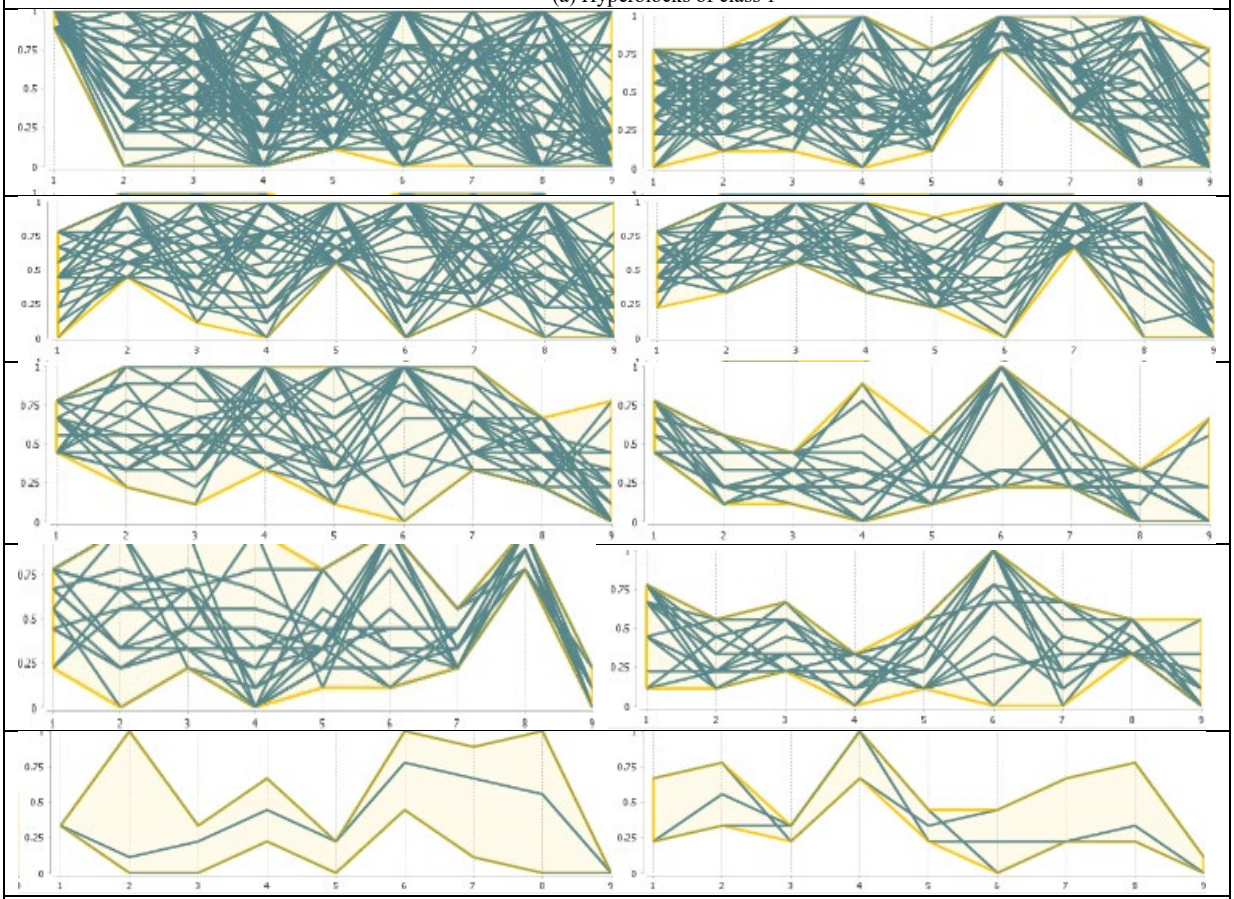
Figure 16. Largest level 2 hyperblock for the MGT dataset with 8,449 cases out of total 19020 cases.

5.5. Case study 5: Multilayer Hyperblock Approach -- DT-HBs for WBC data

This case study shows the potential for alternative HB creation algorithms. In previous case studies, HBs were generated using the GLC-HBRL algorithm. This algorithm builds HBs to explain an LDF. Alternative HB creation algorithms can provide an explainable model directly without referencing an unexplained LDF. For instance, the Interval Merger (IMHyper) [Huber et al. 2024] HB creation algorithm does not use LDFs. For the WBC dataset, using IMHyper produces 18 HBs with 100% accuracy. These HBs are shown in Figure 17. Moreover, by using IMHyper with the DT-HB HB creation algorithm the number of HBs can be reduced from 18 to 13 at the cost of 2.5% accuracy. These HBs are shown in Figure 18. Comparing these 13 HBs with the 15 level 2 HBs created in Case Study 3 shows how changes made to the HB creation algorithm can lead to results equal to or greater than level *n* HB creation algorithms. This shows the multitude of ways in which HB generalization can be developed.

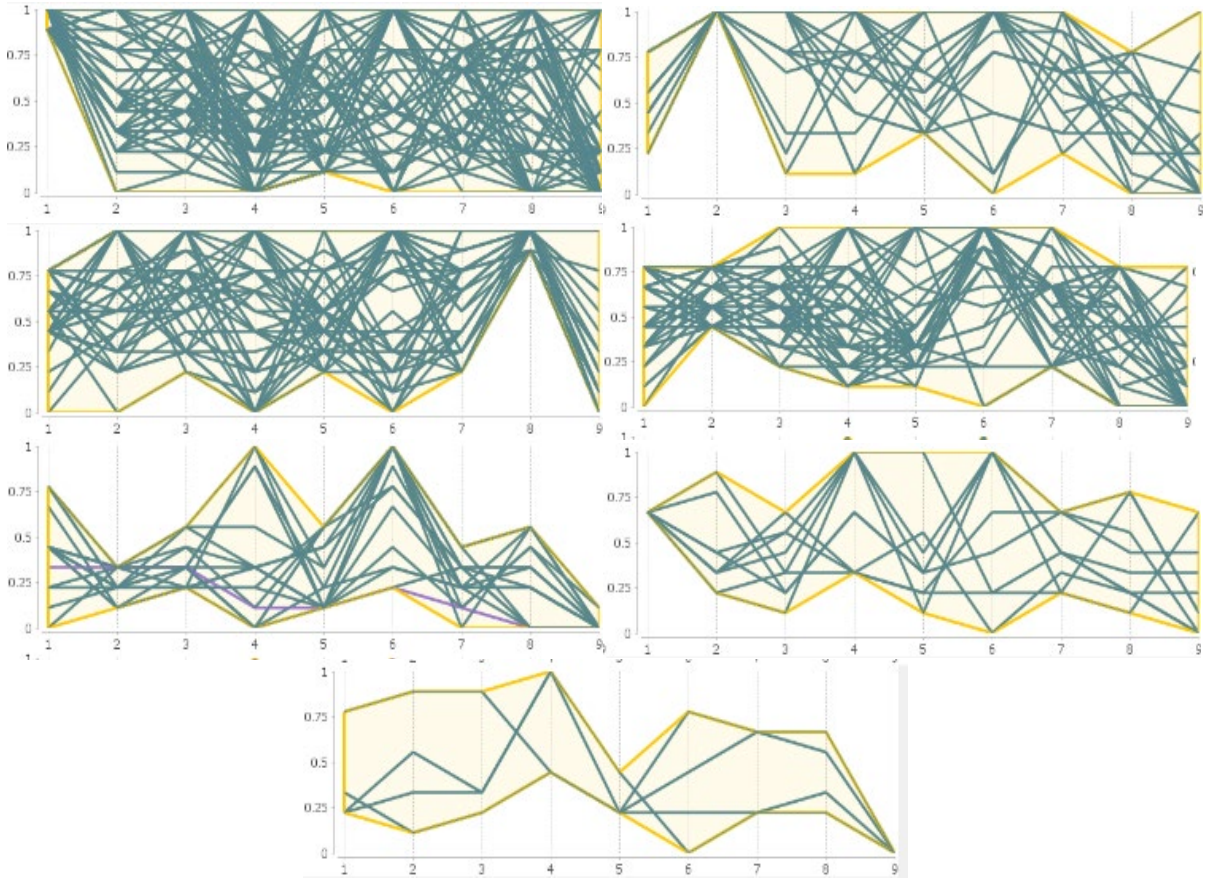


(a) Hyperblocks of class 1

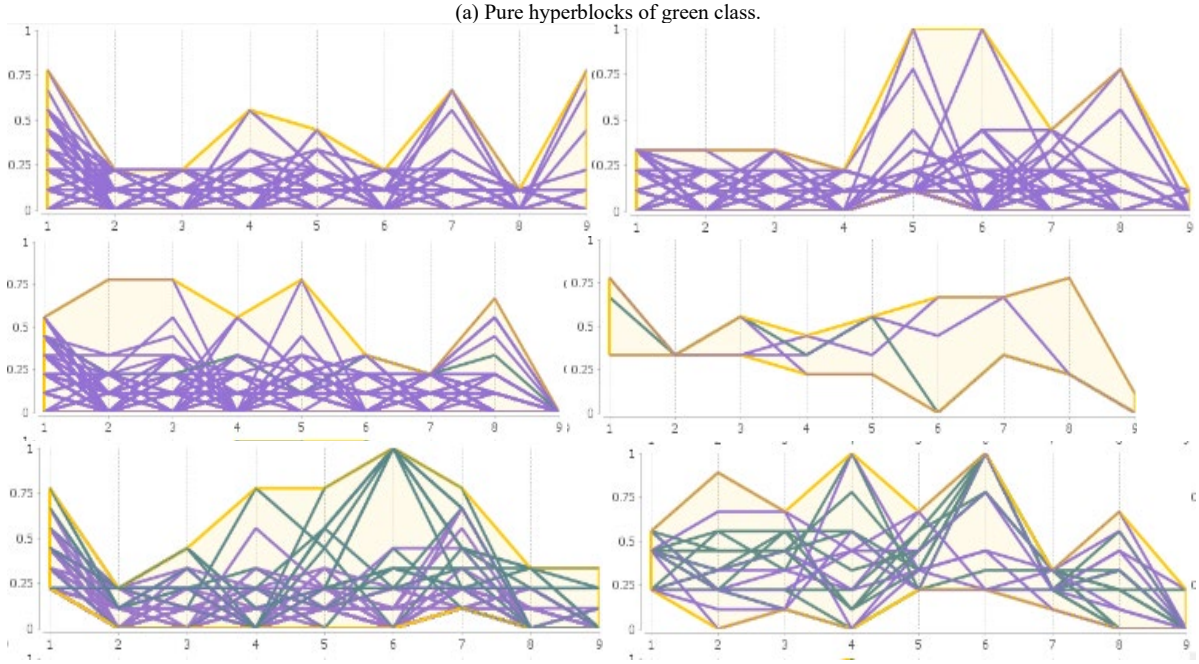


(b)Hyperblocks of class 2.

Figure 17. WBC dataset visualized with 18 hyperblocks with 100% accuracy (magenta- cases of class 1, green -cases of class 2). Each hyperblock is a breast cancer predations model. It can be analyzed by domain experts without needing to be proficient in machine learning.



(a) Pure hyperblocks of green class.



(b) Pure hyperblocks of blue class and mixed impure hyperblocks with cases of both classes.

Figure 18. WBC dataset visualized with 13 DT hyperblocks with 97.5% accuracy (magenta- cases of class 1, green -cases of class 2). Each hyperblock is a breast cancer prediction model. It can be analyzed by domain experts without needing to be proficient in machine learning.

6. FEATURES OF SOFTWARE TOOL

The DV2 program [Huber, Kovalerchuk, 2023] was developed and used for all experiments shown in this paper. Figure 19 shows an example of this program visualizing the WBC dataset in GLC-L. A confusion matrix shows the results of this visual model in the bottom left corner.

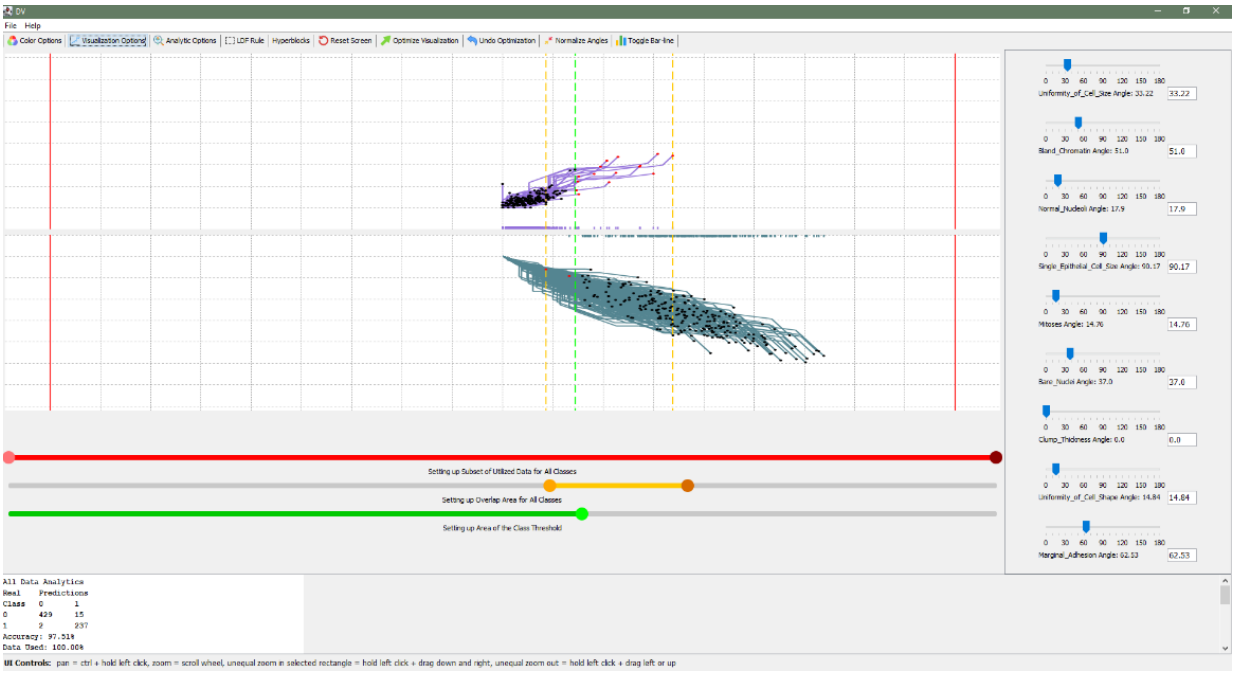


Figure 19. WBC dataset visualized with 98.1% accuracy with GLC-L in DV2 software system.

The option to create HBs is in the toolbar at the top of the window. By selecting this option HBs will be created, and various HB options will appear. Figure 20 shows this HB window. Options include the ability to increase the HB level and visualize in Parallel Coordinates (PC), GLC-L, and Shifted Paired Coordinate (SPC). All these lossless visualization methods for n-D data are defined in [Kovalerchuk, 2018]. Options to visualize HBs together or separately, and the option to visualize overlap data within each HB separately, together, or not at all are also available. Additionally, the outlines and data within each HB can be hidden or visualized.

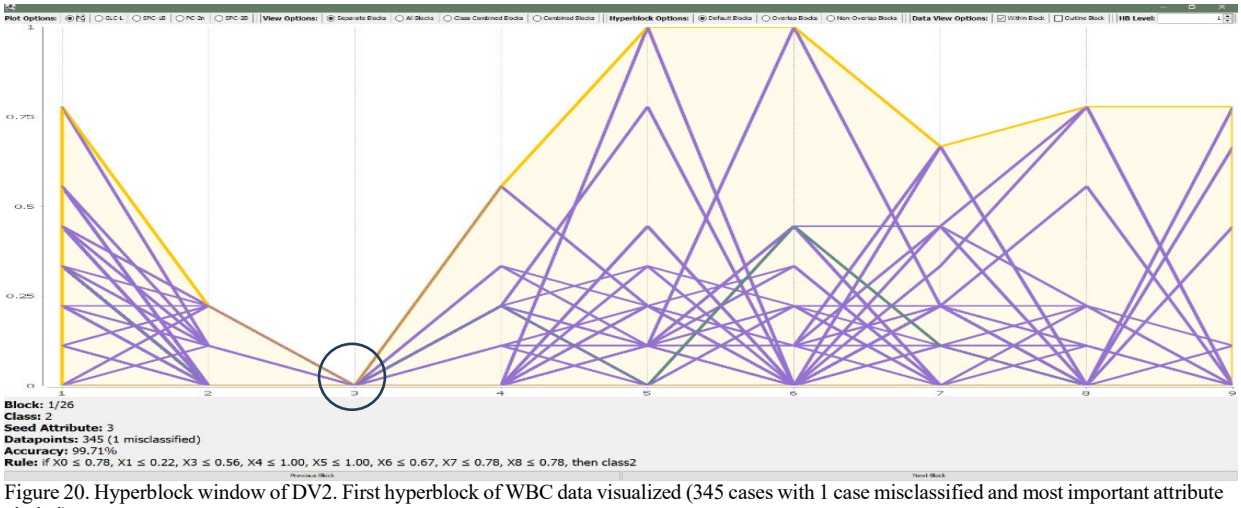


Figure 20. Hyperblock window of DV2. First hyperblock of WBC data visualized (345 cases with 1 case misclassified and most important attribute circled).

Scalability to larger datasets is important to the future of DV2 software system with parallel and performance enhancing features. The parallel HB creation algorithm noted in Figure 7 is implemented in CUDA. This integration of multilayer generalization within GLC visualizations ensures that high-dimensional data remains comprehensible and actionable being simplified for a broader analysis.

The ML methods and visualization techniques are packaged into the DV 2.0 software applications. DV 2.0 was developed using Java and JFreeChart on the Windows 10 operating system. Various ML methods were employed, including Linear Discriminant Analysis (LDA) and Linear Regression (LR) for determining angles and threshold values in General Linear Coordinates Linear (GLC-L) visualizations, and Support Vector Machines (SVM) for identifying support vectors in the General Linear Coordinates Non-Linear (GLC-nL) algorithm. Nine additional standard ML methods implemented using Python's scikit-learn library alongside a Convolutional Neural Network (CNN) from for k-Fold cross-validation analytics: Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), Decision Tree (DT), Stochastic Gradient Descent Linear (SGD-L), Naïve Bayes (NB), k-Nearest Neighbors (k-NN), Logistic Regression (LR), Multilayer Perceptron (MLP), Random Forest (RF), Convolutional Neural Network (CNN). The DV2.0 software is publicly available at GitHub: https://github.com/CWU-VKD-LAB.

7. DISCUSSION

Below we discuss and outline the position of the proposed multilayer approaches with hyperblock classification methods and lossless General Line Coordinates relative to standard machine learning techniques. Both Hyperblock algorithms and standard Decision Tree algorithms produce interpretable models and bypass post-hoc explanation methods like LIME and SHAP. Mathematically a model as a set hyperblocks allows to capture more interpretable properties than a single decision tree. Each branch of the decision tree is a specific hyperblock. These HBs have the common root. This is a constraint of the decision tree that the Hyperblock algorithms do not have. To remove this constraint the algorithms that build a set of decision tree (forest) were proposed including the standard Random Forest algorithm, which is not interpretable. A set of hyperblocks can be equivalent to a forest of decision trees.

Only black-box ML models require explanation methods. Model explanation methods like LIME and SHAP are fundamentally designed to explain black box models as a remedy to inability to build explainable non-black box models in the first place. The current trend in machine Learning if building explainable models in the first place, which can compete in accuracy with black-box models. The proposed multilayer hyperblock approach is in this paradigm.

Visualizations with PCA and t-SNE are lossy and fundamentally bring the risk of losing important information, while different forms of integration of lossy and lossless visualization are useful [Huber at al., 2023]. In contrast, the proposed methods visualize multidimensional data without loss of multidimensional information. Next, dimension reduction methods are designed to visualize only data but not models, while GLC-L method visualizes both and data are visualized losslessly in contrast with dimension reduction methods.

8. CONCLUSION AND FUTURE WORK

8.1 Benefits

For a domain expert to have trust and confidence in an ML model, explainability is necessary and crucial. For this reason, it's essential to maximize both the quantity and generality of the models. Even though hyperblocks are inherently understandable, overfitting and HB multiplicity can nonetheless result in low user confidence. The present study employed Visual Knowledge Discovery techniques in conjunction with level 2 HBs to optimize each HB and boost its explainability, usability, and confidence.

It has been demonstrated that the multilayer technique can generalize data, resulting in fewer HBs. This is particularly relevant when there is only one instance in an overfit HB. The amount of HBs decreased by 50% or more using both the WBC and MGT datasets. This demonstrates how level 2 HBs simplify and improve the explainability of HBs using both small and large datasets. Parallelization then increases the efficiency of this procedure.

The use of GLC-L of level 2 abilities has also been demonstrated to improve explainability and lessen visual occlusion in each GLC-L visualization. In the event of negative coefficients, these advantages can be further enhanced by developing a simplified GLC-L that displays negative coefficients inverted. MNIST data were used to illustrate this.

GLC-L visuals and HB generation can also be combined to produce interactive and understandable higher level HBs using interactive HB creation algorithms. Interactive HB development gives domain experts direct access to the model discovery process, in contrast to previous HB generation techniques. This allows domain experts to add in-depth subject knowledge to a model, improving it in ways that might not be achievable otherwise. With 2-D HB characteristics, a user can also interactively increase the accuracy of linear GLC-L models.

8.2 Limitations

The current approaches and their implementation have several limitations. On of them is the computational complexity of finding hyperblocks (HBs) in large datasets. Our proposed parallelization strategy requires further enhancement.

Another limitation is the large number of hyperblocks and the potential for overfitting. Even after reducing the number of hyperblocks by a factor of eight in one of case studies, approximately 400 hyperblocks remain. This poses a challenge for users who wish to conduct a detailed analysis of the hyperblocks. This issue is also well known for large decision trees, which, despite being interpretable, can become unwieldy. Alternative ways to address this limitation include guiding users to focus on subsets of interest, such as overlap regions, exploring additional levels, or combining hyperblocks based on their importance. While the proposed higher-level HB architectures have reduced overfitting, it remains an ongoing challenge, as is common with many machine learning algorithms

The proposed decomposition method, based on GLC-L, simplifies human analysis of both models and data. However, model interpretation requires two steps: (1) decomposing the model and (2) interpreting the resulting parts using hyperblocks (HBs).While this process is more complex for users than directly building hyperblocks from the outset, it can be applied to explain black-box models.

Another limit arises when visualizing in parallel coordinates as the number of attributes reaches the hundreds. In such high-dimensional settings, the plots become cluttered and difficult to interpret, making it challenging to extract meaningful insights. More efficient, lossless methods are needed to represent both data and hyperblocks in these large dimensions, potentially by exploring alternative coordinate systems. Additionally, the limited number of cases visible within each hyperblock presents another challenge. It would also be desirable to visualize multiple hyperblocks together in a single parallel coordinates plot, rather than side by side as done in this paper, to reduce occlusion and facilitate comparison.

8.3. Future work

In the future, to further increase the interpretability of classifications it is imperative to reduce the number of properties presented in a set of HBs. This can be done by recognizing that many HBs of the same class are very similar and may be identical in every attribute except one or two. Because of this, we can reduce the number of properties by removing duplicate properties from similar HBs and allowing unique properties between similar HBs to be joined with an "**or**". In this way the number of properties can be significantly simplified. These new sets of properties will no longer be called HBs but rather **Disjunctive Forms (DF)**. Disjunctive forms will cover understandable propositional rules. which are convenient for the domain expert to deal with. Also, it would be preferable to remove HBs under a user-specified size rather than only those with a single case.

Moreover, to increase the usability of the Explainable Threshold Similarity (ETS) method it is necessary to create a method that can quickly setup many thresholds for large datasets. This can be done by doing binary search with domain experts using visualizations. First, a threshold T can be arbitrarily setup. Then, a visualization of the data can be shown to a domain expert and asked if the data are similar or different. If similar, increase the threshold. If different, decrease the threshold. Future work will include measuring usability and trust through user studies and surveys. Additionally, a performance analysis will be conducted to more precisely identify the strengths and weaknesses of the proposed approaches.

Additionally, in the case of data that require too many HBs to approximate them highly accurately, it may be impossible to classify them using HBs accurately. To fix this problem, a gradual solution can be applied. For easier data, highly interpretable HBs should be used for classification to provide as much transparency as possible to the domain expert. However, in highly overlapped areas, for the sake of higher accuracy, interpretable explanations can be provided for complex uninterpretable models. In this way some transparency can be preserved while maintaining high accuracy. Finally, further optimizing our approach with hyperblocks, function decomposition and lossless visualization to be more general and more parallelized while maintaining the highest level of accuracy will be advantageous in the future.

REFERENCES

[1] Bock, R.. Magic Gamma Telescope. UCI Machine Learning Repository, 2007, https://doi.org/10.24432/C52C8B.

[2] Braun, J. Griebel, M. On a constructive proof of Kolmogorov's superposition theorem. Constructive Approximation. 30 (3): 653–675, 2009.

[3] Browne K, Swift B, Gardner H. Critical challenges for the visual representation of deep neural networks. Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent. 2018:119-36.

[4] Castillo PRD, Cardenosa J, Fuzzy min-max neural networks for categorical data: application to missing data imputation. Neural Computing and Applications, 2012, 21(6):1349{1362

[5] Caruana, R., Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd International Conference on Machine Learning, 2006, 161-168.

[6] Chen L. Nested Hyper-Rectangle Learning Model for Remote Sensing. Photogrammetric Engineering & Remote Sensing., 2005, 1;71(3):333-340.

[7] Chen, T., Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, 785-794.

[8] DV2.0 software, 2024, GitHub, https://github.com/CWU-VKD-LAB.

[9] Feng, J., Yu, Y., Zhou, Z. H. Multi-layered gradient boosting decision trees. Advances in Neural Information Processing Systems, 2018, 31.
[10] Friedman, J. H. Greedy function approximation: a gradient boosting machine. Annals of Statistics, 2001, 1189-1232.
[11] GitHub: https://github.com/CWU-VKD-LAB, DV2.0, 2024.
[12] Hayes D, Kovalerchuk B. Parallel Coordinates for Discovery of Interpretable Machine Learning Models. In:Artificial Intelligence and Visualization: Advancing Visual Knowledge Discovery 2024, (pp. 125-158), Springer. https://arxiv.org/pdf/2305.18434
[13] Huber L, Kovalerchuk B, Recaido C. Visual knowledge discovery with general line coordinates. In: Artificial Intelligence and Visualization: Advancing Visual Knowledge Discovery 2024, 159-202, Springer, arXiv preprint arXiv:2305.18429. 2023
[14] Huber L, Kovalerchuk B. No-code platform for visual knowledge discovering in general line coordinates: Dv 2.0. In2023 27th International Conference Information Visualisation (IV) 2023, 316-322. IEEE.
[15] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Liu, T. Y. LightGBM: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 2017.
[16] Keras, 2025, https://keras.io/api/applications/
[17] Khuat TT, Ruta D, Gabrys B. Hyperbox-based machine learning algorithms: a comprehensive survey. Soft Computing. 2021 (2):1325-63
[18] Konstantinov AV, Utkin LV. Interpretable ensembles of hyper-rectangles as base models. Neural Computing and Applications. 2023, 5(29):21771-95. arXiv preprint arXiv:2303.08625.
[19] Kotp Y., Farid M., Mohame A., dMAGIC-Gamma-Telescope-Classification, Kaggle, 2022, https://github.com/yousefkotp/MAGIC-Gamma-Telescope-Classification?tab=readme-ov-file#contributers
[20] Kovalerchuk, B., Dovhalets, D. Constructing Interactive Visual classification, clustering, and dimension reduction models for N-D Data. *Informatics*, *4*(3), 2017, 1–27.
[21] Kovalerchuk, B., Visual Knowledge Discovery and Machine Learning, Springer, 2018.
[22] Kovalerchuk B., Ahmad MA, Teredesai A. Survey of explainable machine learning with visual and granular methods beyond quasi-explanations. Interpretable artificial intelligence: A perspective of granular computing. 2021:217-67. https://arxiv.org/pdf/2009.10221
[23] Kovalerchuk B, Dunn A, Worland A, Wagle S. Interactive decision tree creation and enhancement with complete visualization for explainable modeling. In: Artificial Intelligence and Visualization: Advancing Visual Knowledge Discovery 2024, 3-40, Springer.
[24] Lee E., Monte Carlo Simulation for Machine Learning Model Performance Evaluation using the MAGIC Gamma Telescope Dataset, 2025, https://drlee.io/monte-carlo-simulation-for-machine-learning-model-performance-evaluation-using-the-magic-gamma-7d4e6f6e7c73
[25] Liu J, Ma Y, Zhang H, Su H, Xiao G. A modified fuzzy min–max neural network for data clustering and its application on pipeline internal inspection data. Neurocomputing. 2017, 238:56-66.
[26] Loh, W. Y. Classification and regression trees. Wiley Interdiscipl. Reviews: Data Mining and Knowledge Discovery, 1(1), 2011, 14-23.
[27] Lundberg, S. M., Lee, S. I. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30, 2017.
[28] Matveev SA, Oseledets IV, Ponomarev ES, Chertkov AV. Overview of visualization methods for artificial neural networks. Computational Mathematics and Mathematical Physics. 2021 May;61(5):887-99.
[29] MNIST Dataset, https://git-disl.github.io/GTDLBench/datasets/mnist_datasets, 2024.
[30] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Duchesnay, E. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2011, 2825–2830.
[31] Preda, G. (n.d.). Simple introduction to CNN for Mnist (99.37%). Kaggle. 2018. https://www.kaggle.com/code/gpreda/simple-introduction-to-cnn-for-mnist-99-37/notebook
[32] Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems. 2018;31. https://proceedings.neurips.cc/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf
[33] Recaido, C., Kovalerchuk, B., Intpretable Machine Learning for Self-Service High-Risk Decision-Making, in: Proc. of the 26th International Conference on Information Visualisation (IV), IEEE, 2022.
[34] Ribeiro MT, Singh S, Guestrin C. Why should i trust you? Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining, 2016, 1135-1144.
[35] Samek W, Binder A, Montavon G, Lapuschkin S, Müller KR. Evaluating the visualization of what a deep neural network has learned. IEEE transactions on neural networks and learning systems. 2016; 28(11):2660-73.
[36] Salzberg S. A nearest hyperrectangle learning method. Machine learning. 1991;6:251-76.
[37] Seifert C, Aamir A, Balagopalan A, Jain D, Sharma A, Grottel S, Gumhold S. Visualizations of deep neural networks in computer vision: A survey. Transparent data mining for big and small data. 2017:123-44.
[38] Wang X, Liu K, Wang D, Wu L, Fu Y, Xie X. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. In: Proceedings of the ACM web conference 2022, 2098-2108.
[39] Williams A, Kovalerchuk B. Boosting of Classification Models with Human-in-the-Loop Computational Visual Knowledge Discovery. In:International Conference on Human-Computer Interaction 2025, LNAI vol.15822, pp. 391-412. Springer. https://arxiv.org/pdf/2502.07039?
[40] Wolberg,W., Mangasarian,O., Street, N., Street,W. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. https://doi.org/10.24432/C5DW2B.
[41] Zhang Q, Yang LT, Chen Z, Li P. A survey on deep learning for big data. Information Fusion. 2018 Jul 1;42:146-57.
[42] Zhou, Z. H., Feng, J. Deep forest: Towards an alternative to deep neural networks. 2017, arXiv preprint arXiv:1702.08835.
[43] Zhu, J., Shan, Y., Mao, J., Yu, D., Rahmanian, H., & Zhang, Y. Deep embedding forest: Forest-based serving with deep embedding features. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, 1703-1711.
[44] Zupan B, Bohanec M, Bratko I, Demsar J. Machine learning by function decomposition. In: ICML 1997, 421-429.