VISUAL EXPLAINABLE MACHINE LEARNING FOR HIGH-STAKES DECISION-MAKING WITH WORST CASE ESTIMATES

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

Abstract - A major motivation for explaining and rigorous evaluating Machine Learning (ML) models is coming from high-stake decision-making tasks like cancer diagnostics, self-driving cars, and others with possible catastrophic consequences of wrong decisions. This chapter shows that visual knowledge discovery (VKD) methods, based on the General Line Coordinates (GLC) recently developed, can significantly contribute to solving this problem. The concept of hyperblocks (n-D rectangles) as interpretable dataset units and GLC are combined to create visual self-service machine learning models. Two variants of Dynamic Scaffold Coordinates (DSC) are proposed. It allows losslessly mapping high-dimensional datasets to a single two-dimensional Cartesian plane and building interactively an ML predictive model in this 2-D visualization space. Major benefits of DSC1 and DSC2 is their highly interpretable nature. They allow domain experts to control or establish new machine learning models through visual pattern discovery. It opens a visually appealing opportunity for domain experts, who are not ML experts, to build ML models as a self-service bringing the domain expertise to the model discovery, which increases model explainability and trust for the end user. DSC were used to find, visualize, and estimate the worst-case validation splits in several benchmark datasets, which is important for high-risk application. For large datasets DSC is combined with dimensionality reduction techniques such as principal component analysis, singular value decomposition, and t-distributed stochastic neighbor embedding. A software package referred to as Dynamic Scaffold Coordinates Visualization System (DSCViz) was created to showcase the DSC1 and DSC2 systems.

Keywords: explainable machine learning, visualization, hyperblock, multidimensional coordinate system, self-service model.

1 Introduction

1.1 Motivation and approach overview

Many Machine Learning (ML) models are complex *black boxes* for the end users, which creates difficulties for trusted decision making using them [11, 15] due to various and often hidden assumptions made in the model discovery process. Decision-making can be a life critical or high-risk process. One dataset this work considers is the Wisconsin Breast Cancer (WBC) dataset [6,19]. This is a benchmark dataset used to classify tumors as benign or malignant. A misdiagnosis of a malignant tumor as benign could be a fatal decision for a patient [19]. Other high-risk applications include self-driving cars, missile launches, certain investment strategies and others. A large part of the current impressive successes of deep learning models is not in the high-risk decisions. For many high-risk decisions, the user still reluctant to rely on machine learning models due multiple reason including the *lack of model interpretability* and *uncertainty on model accuracy*.

ML algorithms often rely on *random splitting* available data into training and validation data. The accuracy of each conducted split as well as the *average model accuracy* for splits like 10-fold cross validation can be high and considered appropriate dependent on application. However, the accuracy of the *worst-case* split can be significantly lower than the average, where most difficult cases are put to the validation set to test the

model in the most stressful situations. For life-critical or high-risk decision-making, models with lower average accuracy among many random splits but higher accuracy using the estimate of *worst-case* split may be preferable [9]. This approach is known as a *minmax strategy* based on the *Shannon function* [9] with the goal of finding the model that produces the highest accuracy on most difficult validation data. While finding the exact solution of this problem is computationally challenging the visual knowledge discovery approach allows successfully to find an *upper estimate* of it, as this chapter shows below.

This work considers two main problems: (1) the **interpretability** of ML models where many advanced ML models are difficult to understand, respectively called *black boxes*, and (2) the **reliability of accuracy** of ML models, where the model accuracy can be exaggerated, which is unacceptable in applications with high cost of individual errors.

Many techniques to increase human interpretability of machine learning models exist including data visualization, e.g., [15-18, 24]. The combination of data visualization and machine learning can provide self-service models for the end-user [18]. Self-service visualization models allow an end-user to apply their domain knowledge to tweak the model rules for better decision-making. Human interpretability also provides benefits in data transparency, data fairness, and the development of new models.

Multiple approaches can be found in the literature to address model interpretability and reliability of accuracy problems. This chapter follows the approaches presented in [9,11] based on the visual means. Practically all interpretability studies one way or another use visualization, but mostly to present the results of model explainability studies. The approach that we follow in this chapter has a fundamental advantage of having *lossless visual means* as a major part of both ML model discovery and explanation. It provides a basis for a *self-service model* discovery and interpretation by the end-user/domain expert.

Unfortunately, not every known visualization technique fit well this goal. Machine learning data are fundamentally multidimensional, but popular existing methods to visualize multidimensional data like principal component analysis (PCA), multidimensional scaling (MDS), t-distributed stochastic neighbor embedding (t-SNE) and RadVis are lossy one way or another in representing multidimensional data. It means that attempts to discover multidimensional pattern in these visualizations have a fundamental flaw. In addition, artificially created new coordinates like PCA and t-SNE components do not have natural domain interpretations and can be viewed as black-box *dimensionality reduction* (DR) techniques.

The patterns which exist in n-D space can be lost or corrupted in the process of conversion of n-D data to these visualizations in 2-D/3-D before we will try to discover patterns in these visualization spaces. For instance, two different n-D points can be mapped to a single point in the visualization space due to much smaller size of 2-D/3-D space than n-D space. It is well illustrated by the *Johnson-Lindenstrauss lemma* [22].

The corruption of n-D patterns is illustrated in Figure 1 for t-SNE for Wisconsin Breast Cancer (WBC) data. Each n-D rectangle (hyperblock, HB) is a continuous part of the n-D space, but in t-SNE some of those hyperblocks are not continuous. Some cases of the same hyperblock are far away in t-SNE visualization with multiple cases of other hyperblocks in between.

Next, all n-D points in the hyperblock are within the distances $d_1,d_2,\ldots,d_n$ from the center n-D point of the HB in the respective coordinates. This is an interpretable similarity in contrast with, say, similarity in Euclidian distance often used in multiple kernels, that can be non-interpretable for heterogenous data with heterogeneous attributes like, blood pressure, pulse, temperature and so on. Figure 1 shows 648 9-D cases mapped to the first two t-SNE components with 8 malignant hyperblocks and 4 benign hyperblocks. The red arrow on the top points to the blue case from benign HB B2 and the red arrow in the center of this picture points to another blue case from the same benign hyperblock B2, which are far away from the first blue case with multiple cases on other HBs in between. This picture shows other cases with the same issue. It also shows multiple cases from benign and malignant classes that are near each other. They are good candidates to be checked for the worst-case validation set. See cases between two black lines as an example.

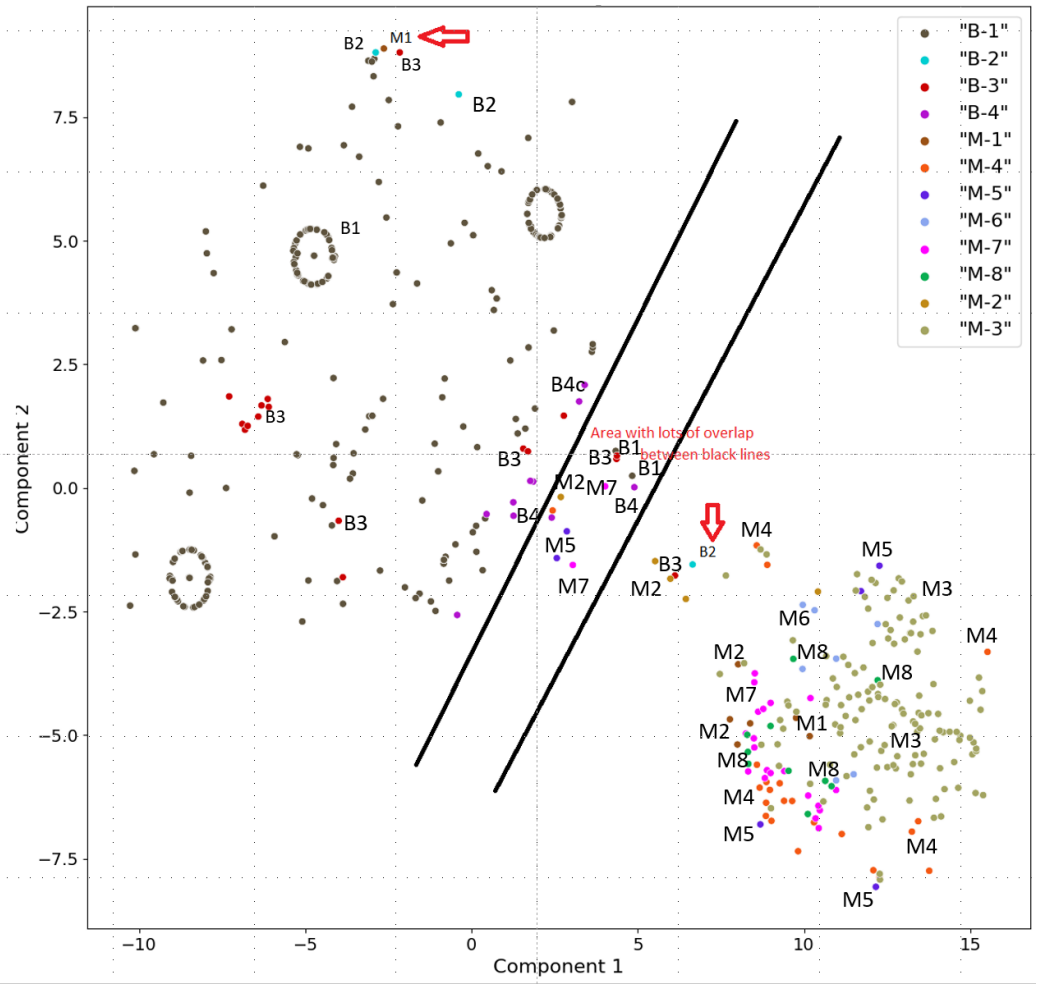


Figure 1. Corruption of interpretable patterns of 9-D points in t-SNE in WBC data.

Therefore, we rely on General Line Coordinates (GLC), which losslessly visualize n-D data with unique graph constructing algorithms [10]. Generally, multidimensional GLC are mapped to graphs in 3-D/2-D space to enhance visual aids. In this chapter to form a powerful tool with high level visual clarity GLC we combine them with non-overlapping hyperblocks (HB. The main advantage of HBs over other methods like many kernels is that HBs *are interpretable*, and all samples of the dataset can be grouped forming pure HBs known as *atomic* HBs [12]. Therefore, complex datasets can be considered unions of the atomic HBs.

While PCA, t-SNE and other lossy methods can corrupt data as was shown above for t-SNE, if the corruption is minor or can be controlled, these methods are very useful. t-SNE is a non-linear unsupervised dimensionality reduction technique that has been applied to high dimensional datasets such as MNIST handwritten digits (784 dimensions) [3] and genomic sampling [8]. Therefore, we explore capabilities of lossy methods in this chapter in combination with General Line Coordinates for visual class separation.

This work explores a new visualization method through the combination of a novel General Line Coordinates system called Dynamic Scaffolding Coordinates (DSC) along with hyperblocks, and dimensional reduction techniques such as principal component analysis and t-distributed stochastic neighbor embedding. Visualizations of high clarity are produced for multiple benchmark datasets where *worst-case* data splits can be discovered that impact model performance. These new methods are called **Dynamic Scaffolding**

Coordinates (**DSC1**, and **DSC2**). DSC1 has basis in parallel coordinates whilst DSC2 has basis in Shitted Paired Coordinates [10]. DSC1 and DSC2 are lossless data visualization methods, which compact multiple input coordinates to only two axes.

One immediate consequence of our visual approach is that large datasets hamper visual knowledge discovery (VKD) due to line occlusion and permutation complexity [10]. We adjusted our dynamic scaffolding approach by incorporating hyperblocks and/or methods like PCA and t-SNE to reduce the number of lines and lay a foundation for our dynamic scaffolds to grow from.

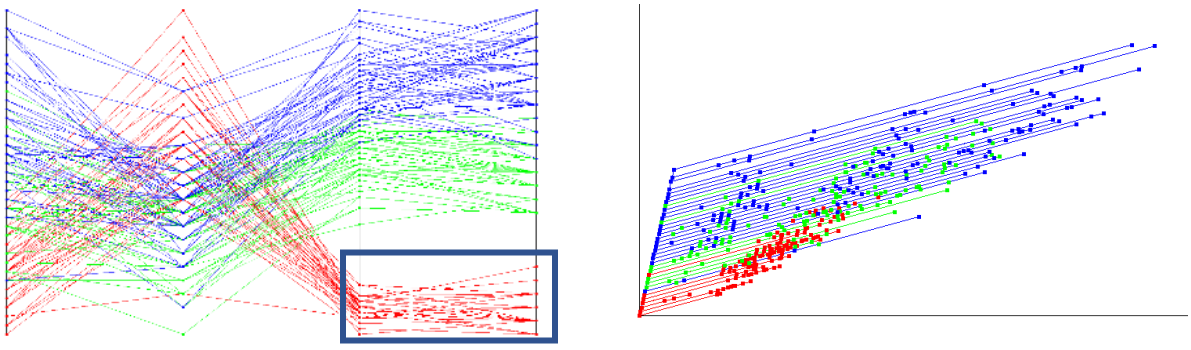
We packaged DSC1 and DSC2 into a software toolkit DSCViz [25]. DSCViz grants the user interactive capabilities such as changing attribute order, emphasizing or deemphasizing classes and/or attributes, visualizing the coordinate system via polylines, markers, or both, zoom and drag capabilities, and clipping/bounding algorithms for sample selection. In addition, DSCViz includes parallel coordinate and shifted paired coordinate plots with the same functionalities listed above. All functionalities are purposefully built to enhance visual knowledge discovery for the end user. DSCViz was developed using Python as the default programming language, QTCreator for the GUI interface, and OpenGL for plot rendering.

We applied the DSCViz software to a real problem, which is using visual knowledge discovery to find the upper estimate of the worst dataset split to training and validation sets (also referred to as the most difficult or **worst-case split**) and/or reduce false predictions to improve critical decision-making such as tumor diagnosis. This is achieved by analyzing visually the DSC1 and DSC2 plots for regions of class overlap and selecting these samples for a validation set. This validation set is expected to reduce general model accuracy when compared to standard k-fold cross validation. One challenge of this VKD approach we found was that not all overlapped regions are equal when discriminating classes from each other, some overlapped regions practically do not impact model performance whereas other overlapped regions have a great impact when used in the validation set.

Several *worst-case* split experiments were conducted on benchmark datasets such as Seeds, Iris, and Wisconsin Breast Cancer [6]. We further investigated our visualization tool on very large datasets such as MNIST handwritten digits [3].

1.2 Forms of General Line Coordinates

There are many forms of GLC visualizations which are dependent on the graph constructing algorithm. GLC generalize multidimensional coordinate systems with multiple axes such as parallel coordinates (PC), radial (star) coordinates, shifted paired coordinates (SPC) and others, by allowing more flexibility in location of coordinates [10]. This work uses PC and SPC to construct two new GLC visualizations on a 2-D plane.



(a) Iris dataset in Parallel Coordinates.

(b) Iris dataset in one form of General Line Coordinates.

Figure 2. Transformation of a 4-D PC plot to a 2-D GLC plot.

Parallel line coordinates (PC) represent multidimensional data using several one-dimensional axes. Parallel line coordinates have been used extensively for visual knowledge discovery and decision-making [21]. Each axis in a PC plot is independent and represents the informational space of one attribute. For instance, a 4-D point **x**=(1,3,2,5) is represented by a polyline (directed graph) with 4 nodes located on respective 4 vertical axes at the respective positions (1,3,2,5). Figure 2a shows the Iris dataset on a parallel coordinate plot. Each axis is responsible for one of four attributes: petal width, petal length, sepal width, and sepal length. The minimum and maximum values of each axis correspond to the range of each attribute. Increasing the range of one attribute will have no effect on the other attributes. Figure 2a shows a clear classification area for the red class on the bottom right-hand corner across two attributes. Any new samples that would plot in the bottom right-hand corner would be classified as red class. Establishing a rule to classify blue and green cases from each other is more difficult as they exhibit overlap on all four attributes. Figure 2b shows the Iris dataset in another form of General Line Coordinates.

Shifted paired coordinates (SPC) represent multidimensional data using several two-dimensional axes [10, 18]. Each two-dimensional axis holds the informational space of two attributes. SPC show relationships between pairs of attributes. For instance, a 4-D point **x**=(1,3,2,5) is represented by a polyline (directed graph) which connects node/point (1,3) of the first pair of coordinates in one Cartesian plot with node/point (2,5) of the second pair of coordinates in another Cartesian plot. This leads to a limitation of the SPC plot in which an even number of attributes are required. A dataset with an odd number of attributes will require engineering, duplicating, or removing an attribute. Duplicating of an attribute is most desirable because it preserves information of the dataset.

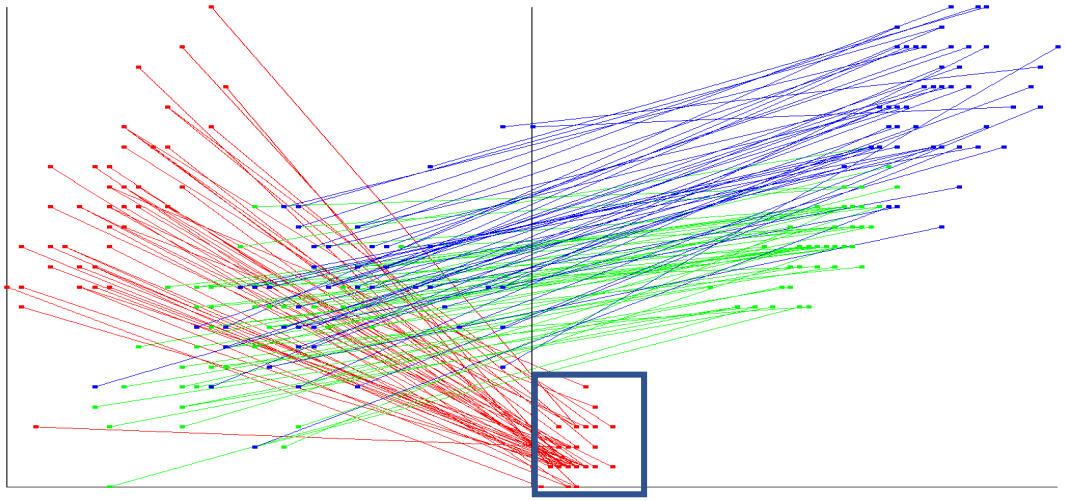


Figure 3. Iris dataset on a shifted paired coordinate plot.

Figure 3 shows the lossless Iris dataset on a shifted paired coordinates plot. The first attribute-pair shows a relationship between sepal width (vertical) and sepal length (horizontal), and the second attribute-pair shows a relationship between petal width (vertical) and petal length (horizontal). Separation of the red class is shown in the blue rectangle. Green and blue classes are *highly overlapped*. The order of attributes is important when making a SPC plot due to the unique relationship between any two attributes, which leads to visually different plots. This introduces a new hurdle in developing multidimensional visualizations, which is the exponential time complexity of plotting every attribute order permutation and choosing the best one.

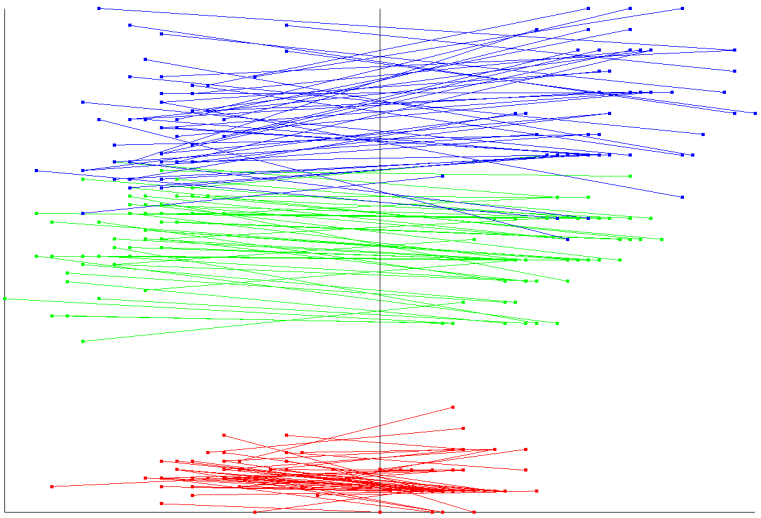


Figure 4. Iris dataset on SPC with high visual classification clarity.

Figure 4 is a visually appealing representation of class separation of the Iris dataset in SPC constructed using an alternative pairing of the same 4 coordinates. The red class is completely separated from the red and green class. In Figure 3 the green class and blue class had large amounts of overlap, but for this permutation of attributes the blue and green classes overlap in a *much smaller area*. The difference between Figure 3 and Figure 4 is quite drastic in readability to the user. The time to produce every attribute permutation of the Iris dataset and choose the plot for Figure 4 was only a few seconds, however, computing permutations of larger multidimensional datasets may not be feasible.

A **hyperblock** (HB) is a multidimensional "rectangle" (n-orthotope) with set of multidimensional points $\{\boldsymbol{x} = (x_1, x_2, \dots, x_n)\}$ with center $\boldsymbol{c} = (c_1, c_2, \dots, c_n)$ and side lengths $\boldsymbol{L} = (L_1, L_2, \dots, L_n)$ [12, 20] such that,

$$\forall i \in N, |x_i - c_i| \leq \frac{L_i}{2} \tag{1}$$

If a HB has equal length sides, then it is known as a *hypercube*. HBs can represent a group of samples with similar attribute values and can be used to condense the number of lines needed for visualization. HBs are highly interpretable data units as a combination of individual coordinates without imposing non-interpretable operations between coordinates. This allows for the separation of different data units that exist among multiple attributes during model creation. In contrast, some other kernels known in machine learning are not interpretable. For example, in k-nearest neighbors, a Euclidean distance search requires a summation of squared differences for all attributes and computing square roots of them. This may have no meaning in the domain, like cancer diagnostics, and not appropriate to the domain expert. In a previous study hyperblocks were used to generalize decision tree rules [12]. Class *purity ratings* are given to HBs, with the purest HBs containing samples from only one class, and the *least pure* HBs contain equal number of samples from all classes.

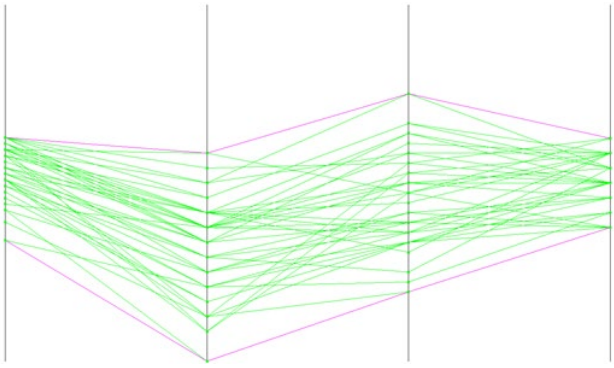


Figure 5. Hyperblock in 4-D space with boundary lines (pink) on a PC plot.

Figure 5 shows a HB in 4-D space. The boundary lines (pink) contain several green samples. The green lines can be omitted, and the pink boundary lines will still contain the information space of all those samples. Methods like principal component analysis can produce HBs but those HBs can be non-interpretable. Non-overlapping interpretable hyperblocks from a specific dataset can be found by using decision trees, Merger Hyperblock algorithm [12], and other methods. For this research we considered the decision tree method of creating hyperblocks due to the smaller time complexity of running a decision tree compared to Merger Hyperblock algorithm.

1.3 Related studies

In this section we discuss prior studies involving General Line Coordinates (GLC) systems and other approaches like Local Interpretable Model-agnostic Explanations (LIME) and their applications.

The LIME method was proposed as means to approximate less interpretable models such as neural networks [14, 23]. It approximates a learning model at the local level by performing perturbation around a particular input and mapping the output through linear models. LIME has been used in many applications from natural language processing [4] to computer vision [14]. However, LIME and similar methods have a *limited interpretability* for heterogeneous data [11] in contrast with decision trees and logical models, which we exploit in this study with methods based on the GLC.

For GLC these studies include interactive visual knowledge discovery in shifted paired coordinates (SPC), Pareto optimization in GLC, GLC-L coordinate system, and more [10]. In [18] interpretability of machine learning models was increased by developing interactive shifted paired coordinates system in the SPCVis software. SPVis introduces a plethora of features that adapt the shifted paired coordinate system such as non-linear scaling, non-orthogonal displays, and serpent parallel coordinates. In addition, in [18] a genetic algorithm and coordinate order optimizer are used to find strong attribute permutations that led to class separability, which is especially important for high-dimensional datasets. The SPCVis optimization algorithms allow to produce a multi-stage visual classifier. Accuracy results using SPCVis on Seeds, Wisconsin Breast Cancer, and Iris datasets are comparable to ML algorithms from the literature. Our study also uses SPC, not for discovering classification models directly, but by using SPC to design new lossless visualizations DSC2.

In [5,10] General Line Coordinates are used to transform non-image data (vectors filled with general numeric attribute information) into an image. One important factor of it is that GLC is a **lossless** projection from multidimensional data into two dimensions, and the produced image contains all the original information of the dataset. This approach allowed to apply to any dataset that is not image such as Wisconsin Breast Cancer

dataset and input that data into a Convolutional Neural Network. While our research uses a GLC system with dynamic scaffolding, [5] used an alternative GLC system referred to as GLC-L where angles are calculated by optimizing a linear discrimination function. There is a close link between DCS1 and GLC-L in design of the visualization, but with a significant difference in their use to discover visual patterns.

The first difference between GLC-L and DSC2 is in the ways of assigning angles and ordering of attributes. In GLC-L the order of the attributes (segments of the polyline) is not critical, but in scaffolding it is optimized. In DSC we derive the angles and order of attributes from external sources like location of values of the attributes in parallel coordinates, shifted paired coordinates, decision trees and other possible sources. It is based on the idea that if some attributes separate classes better than other in external sources, then it can be exploited to assign angles and order of attributes in DSC to improve visualization. Next, DSC allows adding/engineering additional attributes to improve visualization too.

In [1,10] GLC-L and Pareto optimization are used to find a best-case scenario in certain datasets like student performance, and local weather. As an example, one might want to know which month is best to go hiking, or which pre-major classes lead to better students in preparation for getting accepted into a Computer Science program. It is demonstrated in [1,10] how to pin down key attributes that would form a basis for the Pareto subsets, and those subsets would lead to a Pareto Frontier. This work was able to condense several parallel coordinate plots into a single GLC-L plot. This work also uses GLC-L visualization method, which is similar DSC methods we develop in this study, but for setting up priorities among alternatives in the Pareto set, not for discovering classification models.

In [12] Parallel Coordinates are used as a basis for discovering interpretable classification rules. In this work we propose to use the parallel coordinate to design an alternative dynamic visualization DSC1 for better visual discrimination of classes. Parallel coordinates are static, where the location of the next coordinate does not depend on the location of the previous coordinates. The dependence using in DSC allows to capture more complex patters than in static parallel coordinates.

2 Dynamic Scaffold Coordinates with Hyperblocks

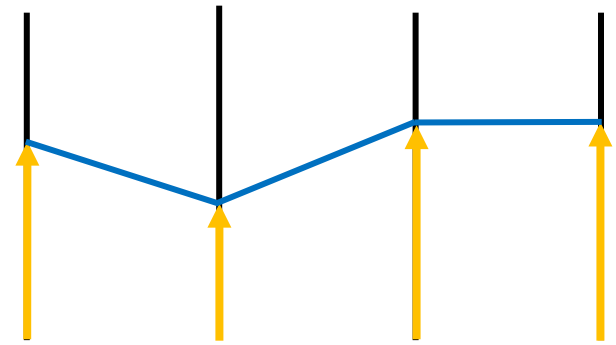
2.1 Dynamic Scaffolding Coordinates based on Parallel Coordinates

Dynamic Scaffolding Coordinates based on Parallel Coordinates (DSC1) generalize the parallel coordinate plot by creating a series of origin-to-attribute scaffolds. Each attribute axis is given a certain angle and the scaffolds connected tip-to-tail to form a polyline to represent losslessly n-D data. The axis tilt can be user-defined or found analytically through optimization. The axis tilt is required to better visualize data trends across two dimensions. Without the axis tilt the line components would stack vertically in one dimension.

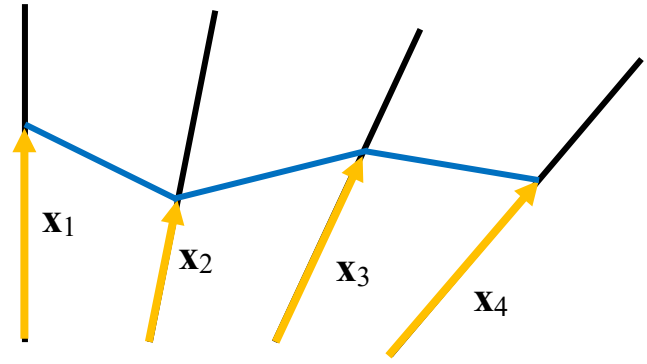
The DSC1 graph construction algorithm (Figure 6) as follows:

(1) Set up dataset sample coordinates in the same manner as a Parallel Coordinates plot.
(2) Apply a rotation transformation for each individual attribute axis with pre-defined angles.
(3) Create a vector from the origin to the attribute point for each attribute. Each of these vectors is called a **scaffold.** The scaffolds are created for all samples.
(4) The first attribute scaffold position is left untouched; however, the tail of the first attribute scaffold is removed, making the tips of the first attribute the "origin" of the polyline.
(5) Translate the remaining scaffolds to the tip of the preceding scaffold.

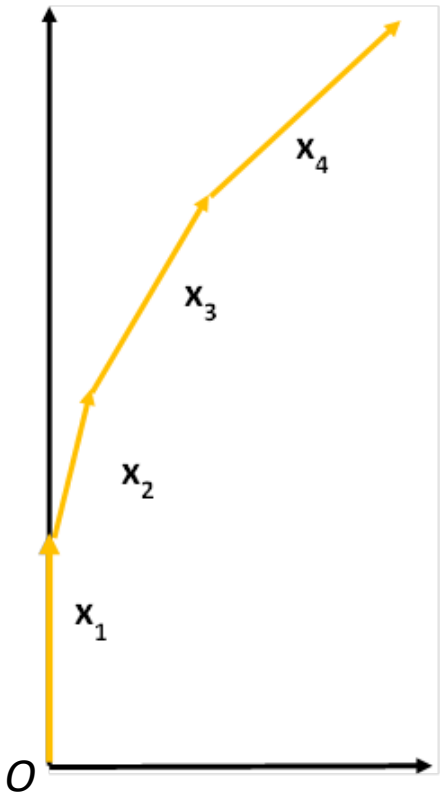
Mathematically this process can be described as follows. Consider n-D point $X= (x_1,x_2,..,x_n)$, then we produce rotated vectors $\mathbf{x}_1,\mathbf{x}_2,..,\mathbf{x}_n$ from X, such as shown in Figure 6b, with lengths equal to respective values of x_i, $||\mathbf{x}_i||=x_i$. $i=1:n$. Then we add these vectors to the origin point O: $V_n = O + \mathbf{x}_1+\mathbf{x}_2+\ldots+\mathbf{x}_n$ to produce a *directed graph* like shown in Figure 6c. Here, in contrast with classical summation of vectors with the n-D point, which will produce a single n-D point V_n, we preserve results of all consecutive sums $V_i = O+\mathbf{x}_1+\mathbf{x}_2+\ldots\mathbf{x}_i$. These points $(V_1,V_2,\ldots, V_n)$ are vertices of the directed graph like shown in Figure 6c.



(a) One sample on parallel coordinates.

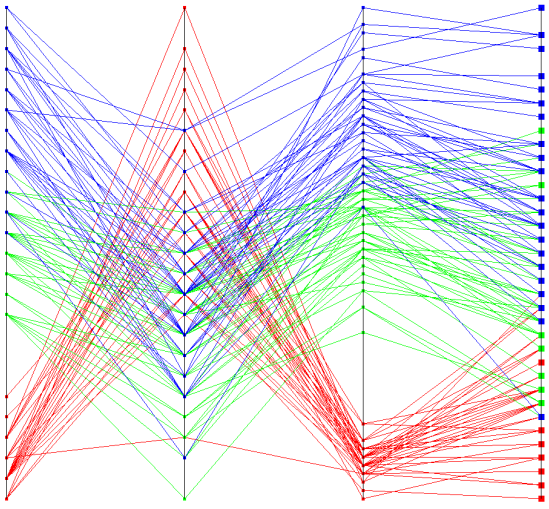


(b) Rotating the axes.

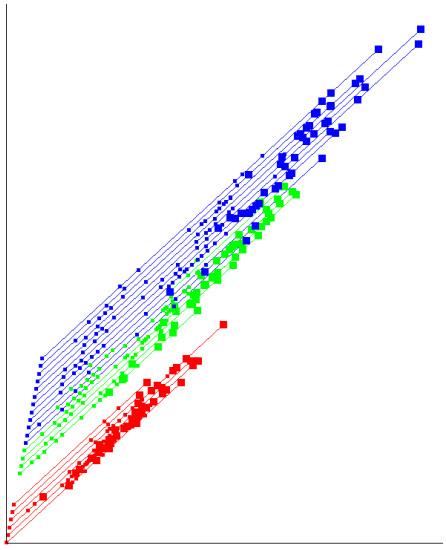


(c) Rotating the axes.

Figure 6. Visual steps for construction of the DSC1 plot.



(a) Iris dataset on Parallel Coordinates.



(b) Iris dataset on DSC1.

Figure 7. Side-by-side comparison of Parallel Coordinates and DSC1 of Iris dataset.

The angles in the DSC1 graph construction algorithm are chosen to visually show separation of classes that separate on one attribute called the *attribute of separation*. The attribute of separation is placed first in the

order of attributes and given the steepest angle to emphasize its importance and the order for the remaining attributes sharing the same angle (Figure 7b), however the possibility exists to change the angle of each attribute as shown in Figure 8. The comparison of Figure 7a with Iris data in parallel coordinates with the same data in DSC1 in Figures 7b and 8 shows the pure advantages of DSC1 over the parallel coordinates for this dataset, while both methods losslessly represent all these four-dimensional data in 2-D visualization space.

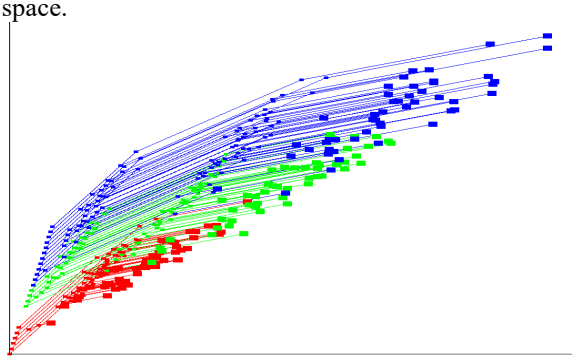


Figure 8. DSC1 with a different rotation for each attribute.

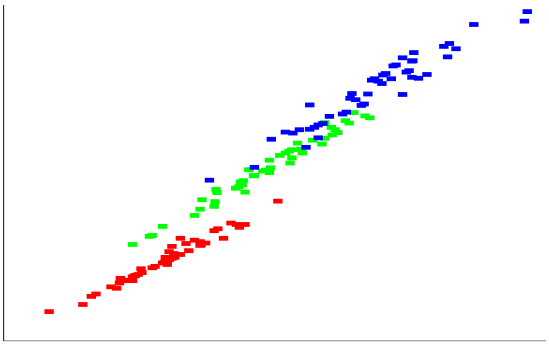
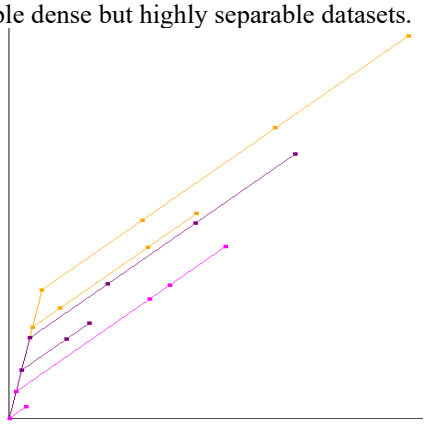


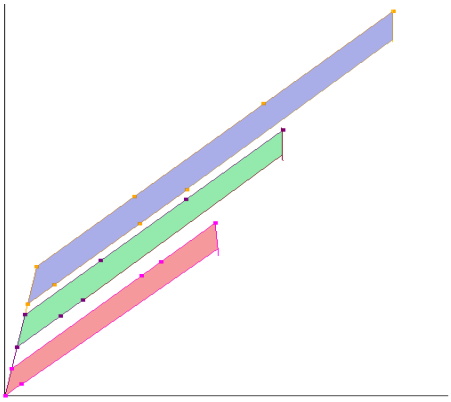
Figure 9. Hiding polylines and certain attribute markers on DSC1.

Other techniques may be applied to DSC1 such as hiding certain attributes markers and hiding the polylines. Figure 9 is another representation of Figure 7b where the polylines are hidden as well as the first three attributes. These self-service techniques can be deployed by the end-user to highlight certain attributes or regions of the dataset that may be of interest.

Figures 7b, 8 and 9 clearly shows that blue and green classes overlap. We can find subsets of those classes in the form of hyperblocks, which do not overlap as Figure 10 shows. Next, DSC1 software grants the user the ability to reduce graphs of samples to an upper and lower hyperblock boundary line as shown in Figure 10. This alternative visualization reduces line occlusion, which can enhance visual knowledge discovery in sample dense but highly separable datasets.



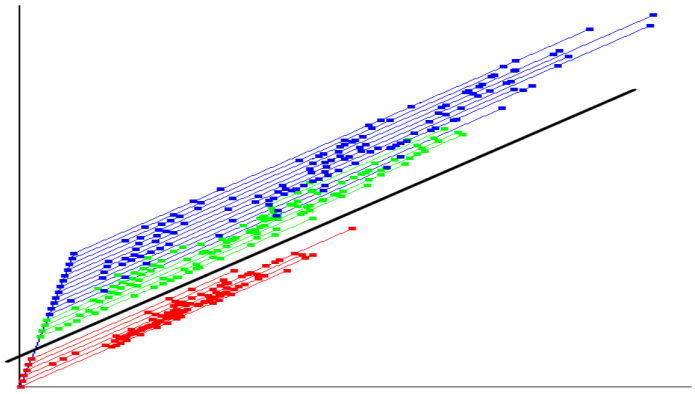
(a) HB boundaries on DSC1.



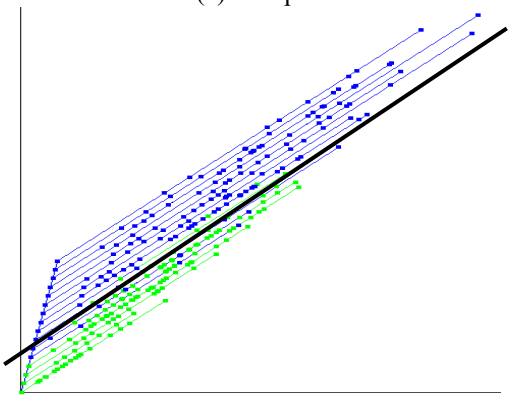
(b) Shaded HBs on DSC1.

Figure 10. DSC1 visualized using only HBs.

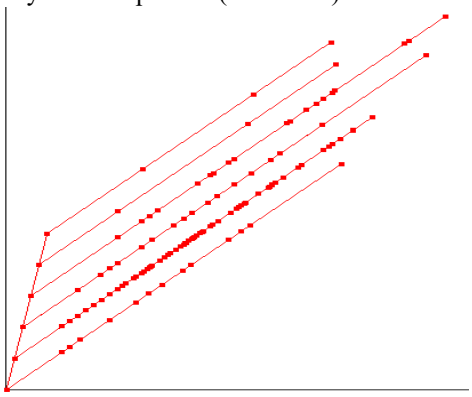
Dynamic scaffolding coordinates can be used to create visual classifiers. Particularly in DSC1 this can be done via a series of DSC1 plots using graphical linear separators. Figure 11a shows the graphical linear separator that separates the entire Setosa (red) class from the Virginica (blue) and Versicolor (green) classes. The next step of the classifier is separating the Virginica and Versicolor classes. There does not exist a spot that can completely divide the two classes without misclassifying some samples as shown in Figure 11b, thus one must analyze the best spot for a graphically linear separator. This methodology is very similar to rule establishment in a decision tree where clear divides between classes may not exist.



(a) Complete Iris dataset with graphically linear separator (black line).



(b) Top split.



(c) Bottom split.

Figure 11. DSC1 plot series classifier.

2.2 Dynamic Scaffolding Coordinates based on Shifted Paired Coordinates

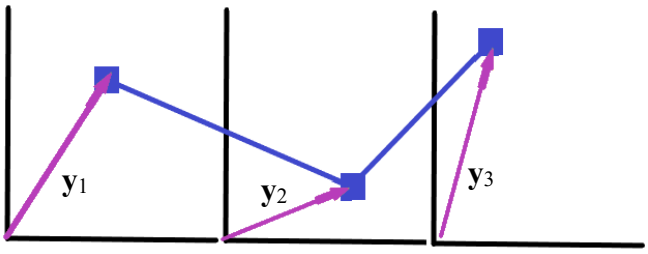
Dynamic Scaffolding Coordinates (DSC2) based on Shifted Paired Coordinates generalize the Shifted Paired Coordinates by creating a series of origin-to-pair scaffolds. Each scaffold is connected tip-to-tail; however, the tail of the first scaffold line is removed as the first attribute pair is the starting point of the multidimensional line.

The DSC2 graph construction algorithm illustrated in Figure 12 is as follows:

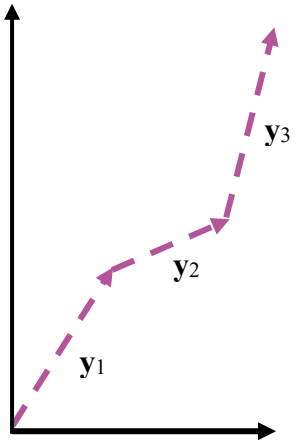
(1) Set up dataset sample coordinates in the same manner as a SPC plot.
(2) Create a scaffold from the origin to the attribute-pair point for each attribute-pair and for all samples.
(3) The first attribute-pair scaffold position is left untouched; however, the tail of the first scaffold is removed, making the tips of the first attribute-pair the "origin" of the polyline.
(4) Translate the remaining scaffolds, to the tip of the preceding scaffold

The general mathematical description of the DSC2 process is the same as for DSC1. The only difference is that vectors are produced by using SPC not by using parallel coordinates. Here we create a set of 2-D vectors $\mathbf{y}_i$ from n-D point $X= (x_1,x_2,..,x_n)$ as follows, $\mathbf{y}_1=(x_1,x_2)$, $\mathbf{y}_2=(x_3,x_4)$,… $\mathbf{y}_{n/2}=(x_{n-1},x_n)$ for even n. If the dimension n is odd then we repeat the last x_n getting $\mathbf{y}_{(n+1)/2}=(x_n,x_n)$. Then we add these vectors $\mathbf{y}_i$ to the origin point O: $V_n = O + \mathbf{y}_1+\mathbf{y}_2+\ldots+\mathbf{y}_{n/2}$ to produce a *directed graph* like shown in Figure 12. Here, again, in contrast with classical summation of vectors with the n-D point, which will produce a single n-D point V_n, we preserve results of all consecutive sums $V_i = O+\mathbf{y}_1+\mathbf{y}_2+\ldots\mathbf{y}_i$. These points $(V_1,V_2,\ldots, V_n)$ are vertices of the directed graph. The DSC2 process is also lossless in the same way as DSC1 process without any loss of n-dimensional information. In contrast, with DSC1 it contains two times less nodes for even n. Figure 13 allows to compare Iris data in SPC and DSC2. Here in contrast with DSC1 vs. parallel coordinates, DSC2 did not produced

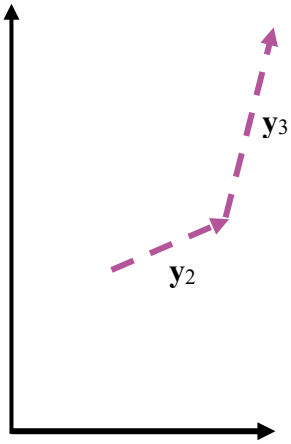
better separation of the classes than SPC. In the next section we will show a way how DSC2 is enhanced to make more appealing on these data for finding the area where the classes overlap.



(a) One sample with scaffolds on SPC.

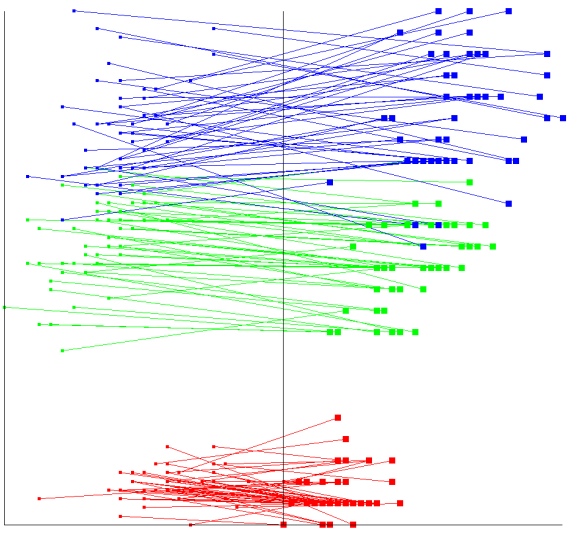


(b) Connecting the scaffolds.

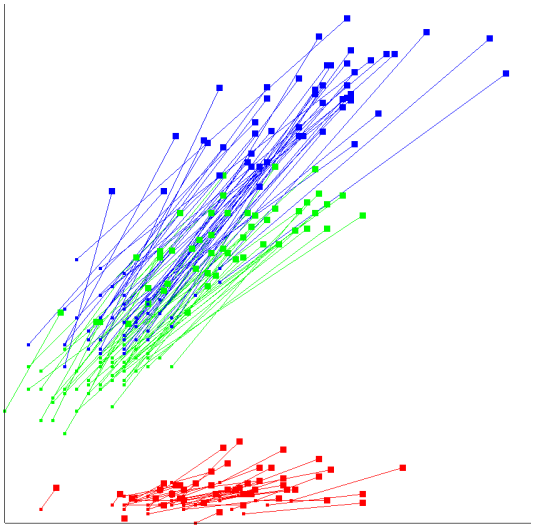


(c) Removing the first scaffold.

Figure 12. Visual steps for construction of the DSC2 plot.



(a) Iris dataset on SPC.



(b) Iris dataset on DSC2.

Figure 13. Side-by-side comparison of SPC and DSC2 of Iris Dataset.

3 Methods and Experimental Results

In this section we describe the methods for visualizing hyperblocks and finding the worst-case splits of data to training and validation sets in DSC. These methods are illustrated on Iris, Seeds, and Wisconsin Breast Cancer (WBC) datasets. In addition, we explored how our visualization performs on large datasets such as MNIST Handwritten Digits. Attributes of interest are developed to produce class separation on the DSC2 plot. Attributes of interest are important because testing every attribute permutation on a DSC2 plot, a factorial time complexity problem, is not feasible for large datasets of a high dimensionality.

3.1 Overview of Methods

We employed three methods to find the **attributes of interest**: (1) hyperblock analysis from decision trees, (2) principal component analysis, and (3) t-distributed stochastic neighbor embedding (t-SNE).

As was pointed out above **hyperblocks** play an important role in visualization of multidimensional data of multiple classes. While in general, the distribution of data of multiple classes can be very complex in multidimensional space, hyperblocks are quite simple and interpretable. Therefore, there is a great interest to visualize hyperblocks, which do not overlap in the n-D space, also non-overlapping in the 2-D visualization space. If this goal will be reached, then data with more complex distributions in a high-dimensional space can be represented as combinations of several hyperblocks. This is especially beneficial when those hyperblocks are pure, i.e., contain only cases of given class.

More formally and specifically this task can be formulated as follows. Consider m non-overlapping HB_i, i=1:m in the n-D space. It means that for each pair HB_i HB_j exists a coordinate $X_{k(i,j)}$ where these HBs do not overlap. The goal is finding and using those separating coordinates $X_{k(i,j)}$ to visualize HBs without-overlap in 2-D.

If the number of these separating coordinates is relatively small, then the chances to visualize those hyperblocks non-overlapping in the 2-D visualization space is much higher. However, it is possible that all $X_{k(i,j)}$ differ for all pairs of these HBs. The total number of these pairs is the number of pairs combinations. Thus, the attempts to visualize many non-overlapping hyperblocks in DSC1 without overlap can fail.

For three HBs it can be done successfully as the Iris example above demonstrates. Therefore, a sequential process is feasible, where first data are split to three hyperblocks. These initial HBs can be impure in a high-dimensional space. Then for each of these HBs, the process continues to split them to 2-3 hyperblocks. With more iterations, the HBs can be made purer and purer. As we see, this is very similar to the decision tree process, while it does not require a binary split. It can operate with three HBs at each iteration. In the actual examples below, we used the decision trees to find the hyperblocks as a computationally efficient process.

Additional attributes can be very beneficial in the situation when hyperblocks are not sufficient to resolve the issue being too simple relative to the actual distribution of the cases, which may require too many hyperblocks. The main idea of using additional attributes is as follows. We compute first two principal components of the dataset, add them to the dataset as two additional attributes and use in visualization along with original attributes. Similarly, we can use t-SNE or Multidimensional Scaling (MDS) components.

As was shown in section 2.1 for DSC1 and below for DSC2 for the Iris dataset hyperblock analysis was successful for class separation. On the other hand, to produce quality class separation of a more complex WBC dataset on DSC2 using hyperblock analysis was not sufficient. To visualize WBC class separation, we escalated to using two principal components in addition to the real dataset attributes. Likewise, for MNIST we were unable to produce class separation using hyperblock analysis or principal components and escalated to using t-distributed stochastic neighbor embedding components in addition to principal components.

3.2 DT Hyperblocks with Iris Dataset

The Iris dataset contains 150 samples of three types of Iris flower (Setosa, Virginica, and Versicolor) [6,7]. The number of samples are balanced between the classes meaning there is 50 samples per Iris flower. The dataset has four attributes relating to petal width, petal length, sepal length, and sepal width.

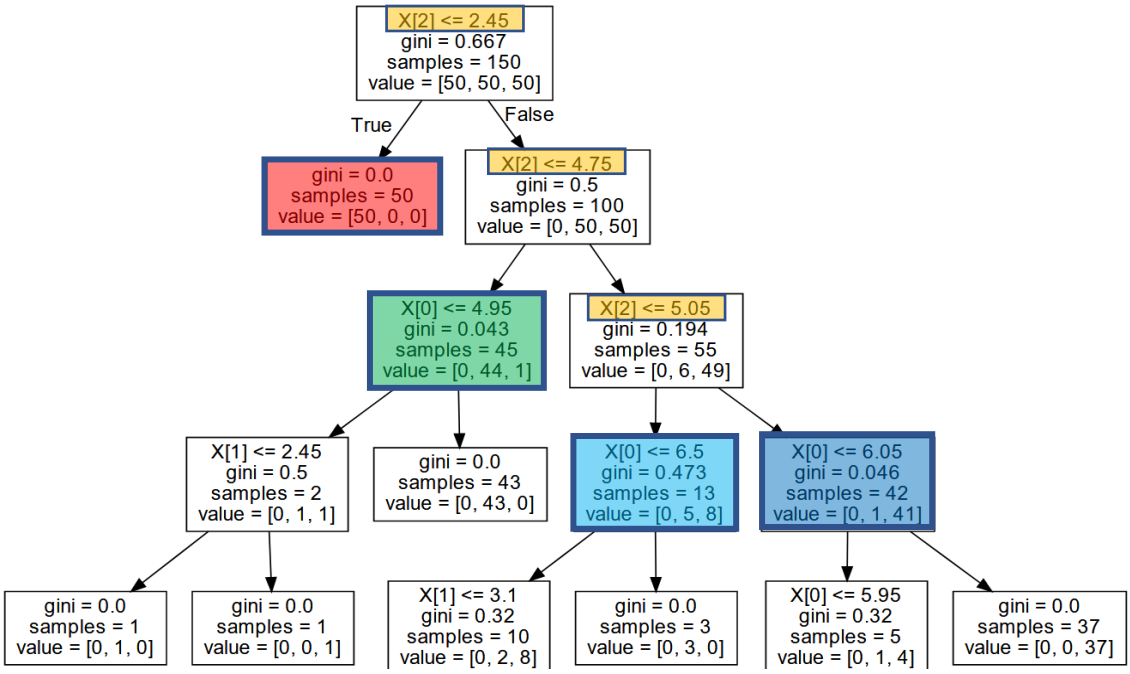
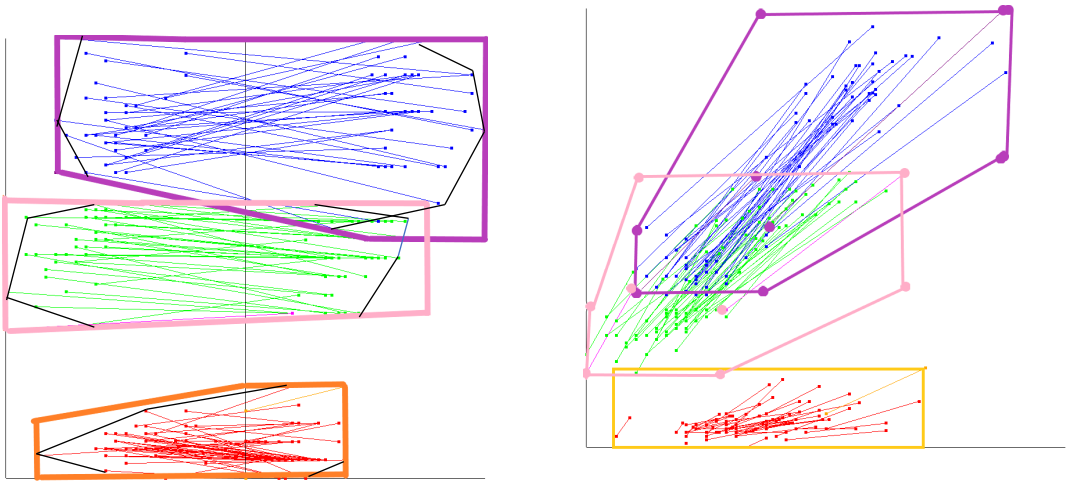


Figure 15. Four levels below the root (level 0) of the Iris decision tree.

The decision tree (DT) analysis is a simple way to develop HBs for this dataset. Figure 15 shows a decision tree model built using the Classification and Regression Tree algorithm (CART) with Gini impurity criterion, and greedy approach on the best split. This figure forms HBs of the Iris dataset. Each sub-branch of the DT represents one HB. To produce non-overlapping HBs from a DT a parent and child node cannot simultaneously be selected because the parent node contains the information of the child, essentially a child is a sub-hyperblock contained inside a parent hyperblock. Going deep into a decision tree a branch can form 100% pure HBs, which contain cases of only one class. However, it runs the risk of overfitting as HBs with very few or a single sample will be present, as some full branches ended in the terminal nodes of the DT show in Figure 15 with a single case.

The HB ended in the red node has 100% Setosa purity, the HB ended in the green node has 97.78% versicolor purity, and the HB ended in blue node has 97.61% virginica purity. These HBs have high purity and contain many samples to reduce the possibility of overfitting. The HB ended in cyan node has only 61.53% virginica purity. It is highly impure. The nodes with the orange highlighted text illustrate the decision tree rules which create such HBs. These DT rules can be applied to multidimensional visualizations when ordering attributes as shown in Figure 16. Figure 17 is a modified Figure 16b where connecting lines are omitted. In Figure 17 the rectangle 1.1 includes the start points of the cases of Setosa class (class 1), the rectangle 1.2 includes the end points of cases of this class. Similarly, rectangles 2.1, 2.2, 3.1 and 3.2 show in DSC2 these points of cases from remaining two classes.



(a) Iris dataset on SPC (b) Iris dataset on DSC2

Figure 16. Side-by-side comparison of SPC and DSC2 of three Iris HBs with outlined classes.

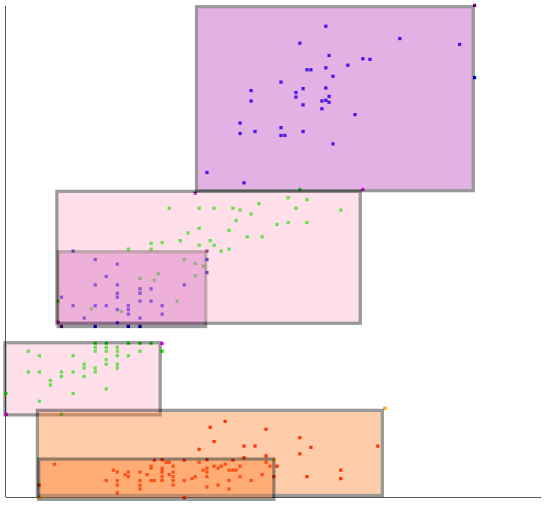


Figure 17. Three main class hyperblocks chosen from the decision tree.

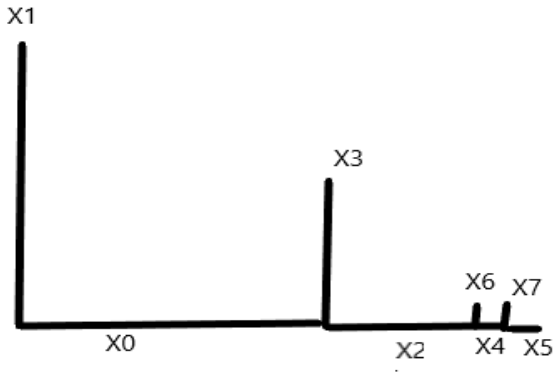
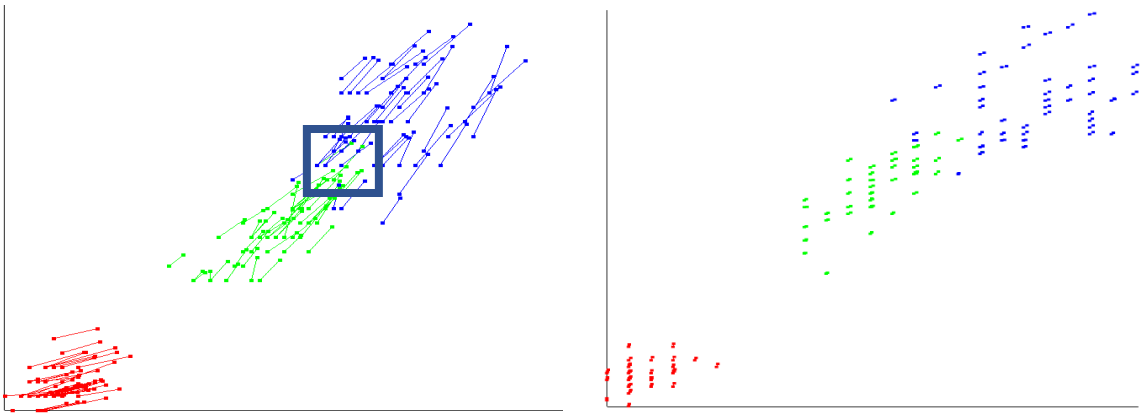


Figure 18. Downscaling attribute-pair axes.

One caveat to the transformation of SPC to DSC2 is that the succeeding attribute-pairs can condense onto the preceding attribute-pair as shown in Figure 17 where the tips of the green class are nearby the tail of the blue class for some samples. See rectangles 2.2 and 3.1 in Figure 17. It is particularly noticeable when an attribute-pair consist of zeros or very small values. One method we deployed to combat this phenomenon is to scale certain attributes-pairs to be larger than the succeeding attribute-pairs as shown in Figure 18.

This effect has diminishing returns on attribute-pairs that come last due to polyline growth always being in the positive up and right directions, thus it is beneficial to carefully select the first couple of attribute-pairs.



(a) Scaling 2nd attribute pair to 10% (b) Scaling 2nd attribute pair to 99%

Figure 19. Emphasizing key attribute-pairs on DSC2.

By decreasing the size of the succeeding attribute pairs or increasing the size of the preceding attribute pairs we were able to highlight better separability between classes. In Figure 16b there was higher visual class overlap, and it was difficult to select samples for a validation set that would result in a *worst-case* split, however in Figure 19 it is immediately noticeable which samples to select for a *worst-case* split.

The Setosa (red) class is completely separated and can be omitted from the worst-case decision analysis as ML models would not struggle to classify Setosa from the other two classes. However, Virginica (blue) and Versicolor (green) have overlap. This leaves 100 samples between the two classes and a 90%-10% test-validation split would require 10 samples to be selected which is shown in Figure 19a. DSCViz features a clipping function that employs the Cohen-Sutherland line clipping algorithm to find samples that are clipped by a user defined rectangle. There also exists vertex bounding where samples are selected if any vertex of the sample's polyline is contained within the box. This can be useful when viewing the dataset using marker points rather than polylines.

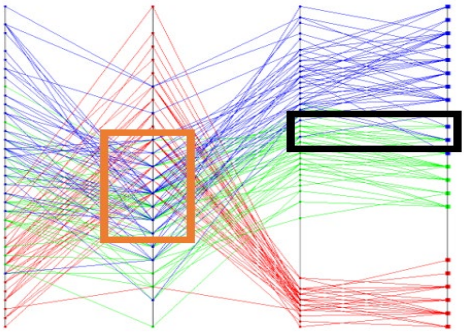


Figure 20. Two areas of overlap on the Iris dataset on PC.

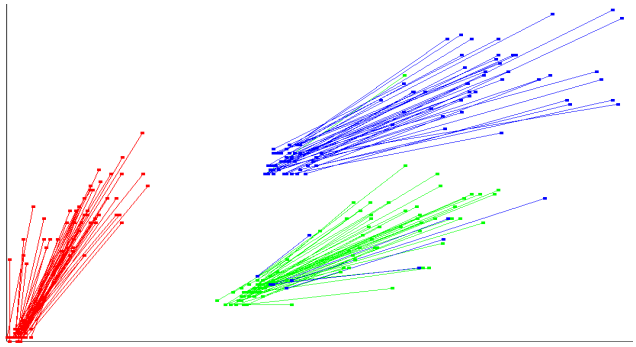


Figure 21. Non-linear scaling on certain attributes of the Iris dataset in DSC2 after non-linear rescaling.

Figure 20 is used to explain the reason why we need to reduce class overlap as much as possible before choosing samples for a *worst-case* validation split. Whilst the orange rectangle is certainly over an area that appears to be highly overlapped, picking those samples may not lead to a bad split because those attributes are likely to not be used by the classification algorithms. Clearly a model such as DT or SVM would make a classification decision using the third or fourth attributes and ignore the first two attributes. By reducing overlap as much as possible we can determine which areas of a dataset that a ML model might use in the decision-making process rather than ignore. The black rectangle is an excellent spot to choose samples for a

bad split. However, the Iris dataset is a simple dataset, and it is clear where ML models may decide to create rules. On larger datasets it is more difficult to determine what areas of overlap are forced into a ML model's rule creation rather than ignored.

This analysis highlights the close relations between (1) building a visualization with the *smallest overlap* of cases of different classes and (2) finding the *worst validation* set in this visualization space. In essence, while these problems may seem unrelated, but they are really two sides of the same coin. We want to get visualization where the classes are separated as much as possible and then pick up the area where they still overlap to be used as a worst validation set.

Another technique known as non-linear scaling [10, 18] can be used to separate Iris data as shown in Figure 20. The decision tree in Figure 15 for the Iris dataset made a rule that separated majority of the Setosa class (red) at a normalized value of 0.17 for petal length. Another decision tree was used to pick up separation using the petal width attribute. The Virginica class (blue) was separated from the Versicolor class (green) 0.67 for petal length. Samples with a petal length attribute greater than 0.17 scaled closer to a normalized value of one, while values less than 0.17 were scaled closer to a normalized value of zero. Likewise for petal width we used 0.67 as the indicator to push attributes values towards 0 or 1. Non-linear scaling exaggerations is also applied. A high exaggeration would squish attribute values at the limits of 0 and 1 whilst a low exaggeration will retain more of the original data.

Figure 22 demonstrates how non-linear scaling was applied on the first attribute-pair to create Figure 21. The classes are pushed in the direction of the corresponding color arrows. The Virginica class (blue) is pushed up because it is above the black horizontal line and the Versicolor class (green) and Setosa class (red) are pushed down as they are below the black horizontal line. The red class is pushed to the left because it is on the left side of the black vertical line whilst the green and blue classes are pushed right as they are right of the black vertical line.

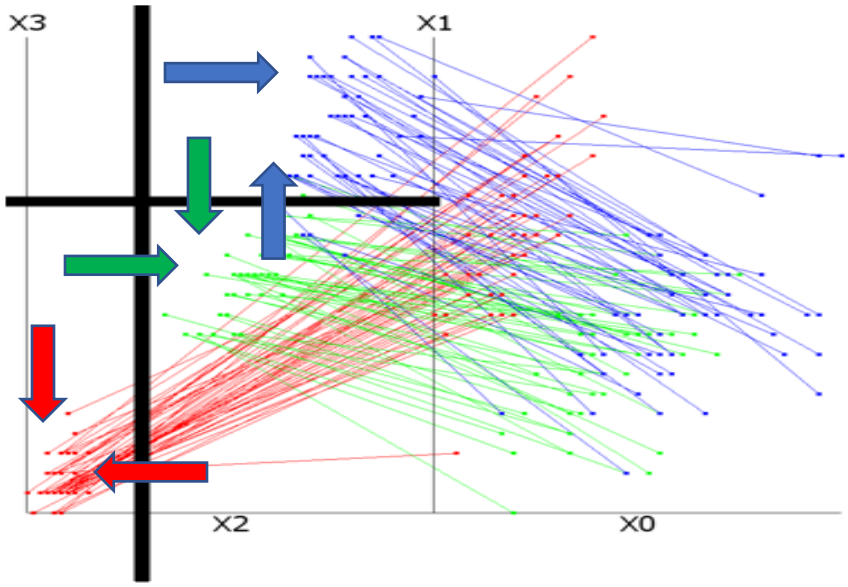
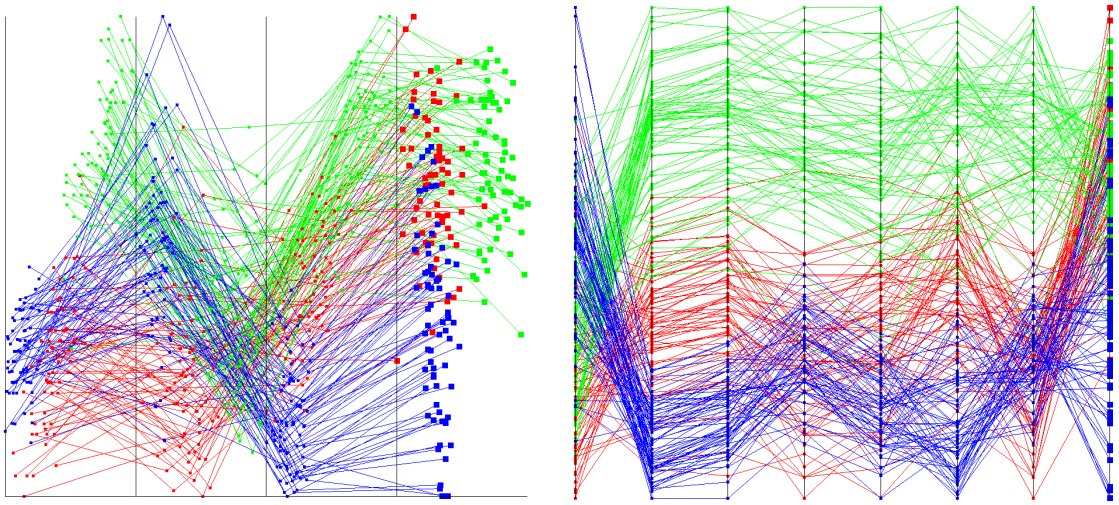


Figure 22. Non-linear scaling technique on SPC.

3.3 DT Hyperblocks with Seeds Dataset

The Seeds dataset contains 210 samples of three types of wheat seeds (Kama, Rosa, and Canadian) [2,6]. The number of samples are balanced between the classes meaning there is 70 samples per wheat seed. The dataset has seven attributes relating to area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, and length of kernel groove.



(a) Seeds dataset on Shifted Paired Coordinates (b) Seeds dataset on Parallel Coordinates

Figure 23. Seeds dataset on SPC and PC.

Immediate separation of classes is not apparent using a PC or SPC as shown in Figure 23. Note that the Kernel Groove attribute was duplicated to create an even number of attributes for the SPC plot. We used a decision tree analysis to find attributes of interest for DSC2 in the process like what described above for the iris dataset. This decision tree is in Appendix A. The attributes of interest happened to be area and kernel groove. The resulting visualization in DSC2 is shown in Figure 24, which demonstrates its advantages over visualization of the same Seeds data in Shifted Paired Coordinates and Parallel Coordinates shown in Figure 22. The three classes of seeds are much better separated in Figure 2 3.

Analyzing the decision tree in Appendix A, we established three hyperblocks with relatively high purity: the green HB contained 68 samples of the green class and 1 sample from the red class, whilst the blue HB contained 70 samples of the red class and 14 samples of the red class, and finally, the red HB contained 55 samples of the red class and 2 samples of the green class. The green HB was separated from the red and blue HBs at a kernel groove value of 5.576 whilst the red and blue HBs were further separated from each other at an area value of 13.410. This decision tree model is also a CART model with Gini impurity criterion, and greedy approach on the best split.

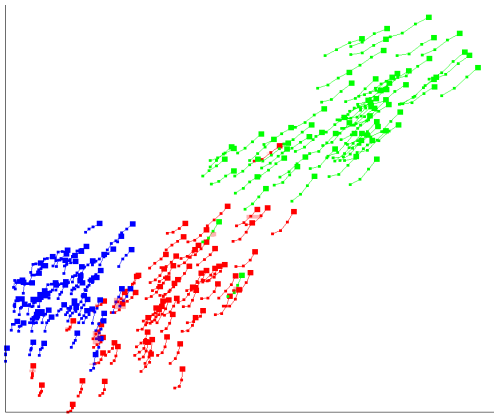


Figure 24. Seeds on DSC2 after scaling and HB analysis.

Using Figure 24 we employed a box clipping algorithm on 21 samples of the Seeds dataset which appeared to be overlapped. Figure 24 is the full dataset and is difficult to see the boxes we used to clip samples as we

plucked individual samples at a time. Figure 25 offers an enhanced view of Figure 24 which shows the samples we clipped into the validation set.

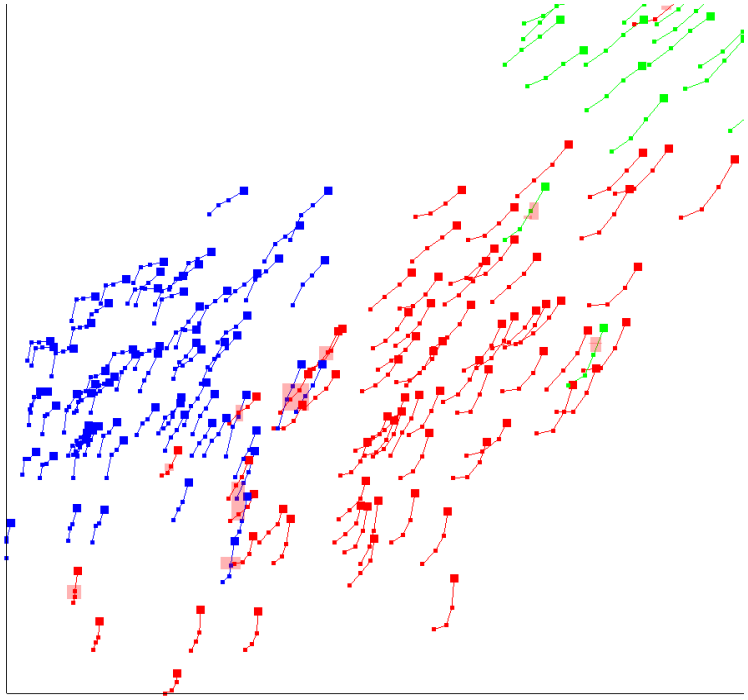


Figure 25. Enhanced Version of the Seeds DSC2 Plot.

After selecting 21 samples (10% of the Seeds dataset) and removing the duplicated kernel groove attribute we compared our validation split to a standard 10-Fold Cross Validation (CV) package in the scikit-learn library [13]. The 10 splits were reused for each model of the eight contained in Table 1. For each ML model all parameters were kept as default. The classifiers are as follows: Decision Tree (DT) using CART algorithm and greedy approach, Support Vector Machine (SVM) with linear approach and l2 penalty, Random Forests (RF) with 100 estimators, K-Nearest Neighbors (KNN) with 5 neighbors and uniform weights, Logistic Regression (LR) with l2 penalty, Gaussian Naïve Bayes (NB), Stochastic Gradient Descent (SGD) linear classifier, and Multilayer Perceptron (MLP) with one hidden layer of 100 hidden units.

Table 1. Standard 10-fold Cross Validation model accuracy on Seeds dataset.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	AVG
DT	95.2	95.2	90.5	95.2	100.0	81.0	100.0	95.2	76.2	81.0	91.0
SVM	100.0	95.2	90.5	100.0	100.0	85.7	100.0	90.5	71.4	66.7	90.0
RF	85.7	95.2	95.2	95.2	100.0	95.2	100.0	95.2	71.4	85.7	91.9
KNN	95.2	95.2	90.5	85.7	100.0	81.0	95.2	90.5	76.2	76.2	88.6
LR	100.0	95.2	95.2	100.0	100.0	81.0	95.2	90.5	81.0	76.2	91.4
NB	90.5	90.5	95.2	90.5	100.0	90.5	100.0	95.2	61.9	76.2	89.0
SGD	81.0	85.7	81.0	90.5	81.0	85.7	95.2	81.0	71.4	90.5	84.3
MLP	95.2	90.5	81.0	95.2	100.0	81.0	95.2	90.5	85.7	81.0	89.5

Table 1 shows that standard ML algorithms can classify the Seeds dataset within 84.3% to 91.9% average accuracy without any additional processing, dimensional reduction, or feature engineering. The lowest split was 66.7% in SVM, and the highest split was 100% across seven models. The best performing model was Random Forest, and the worst performing model was Stochastic Gradient Descent.

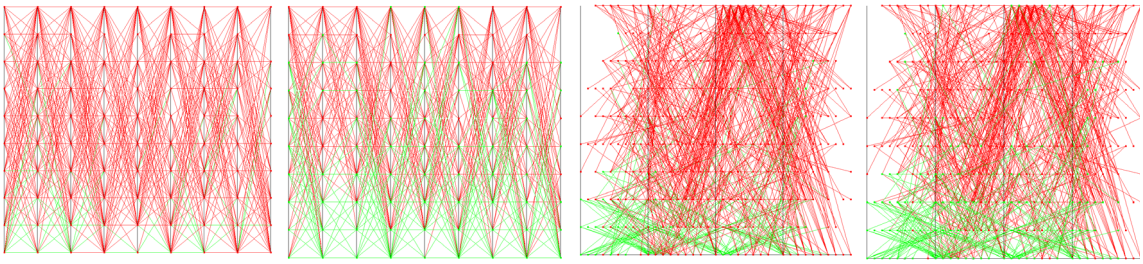
Table 2. Estimate of the worst-case split on Seeds dataset.

Model	Accuracy
DT	57.14%
SVM	38.10%
RF	61.90%
KNN	23.81%
LR	38.10%
NB	57.14%
SGD	42.86%
MLP	23.81%

Table 2 shows that despite the strong model accuracies obtained in Table 1, all eight classifiers had an accuracy between 23.81% and 61.9% in the upper estimate of the worst-case split. Again Random Forest was the best performing algorithm whilst KNN had the lowest performance. Intuitively this makes sense that KNN would be the lowest because the samples we took from the DSC2 plot had neighbors from a different class.

3.3 DT hyperblocks and Principal Components with Wisconsin Breast Cancer Dataset

The Wisconsin Breast Cancer dataset contains 699 samples using nine descriptive attributes: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses [6,19]. We removed 16 samples which have missing values leaving a total of 683 samples. Those 683 samples include 444 benign cases and 239 malignant cases. We chose the WBC dataset due the high-risk nature of cancer tumor diagnosis. A misdiagnosis of a malignant tumor as a benign tumor could prove fatal for the patient [19].

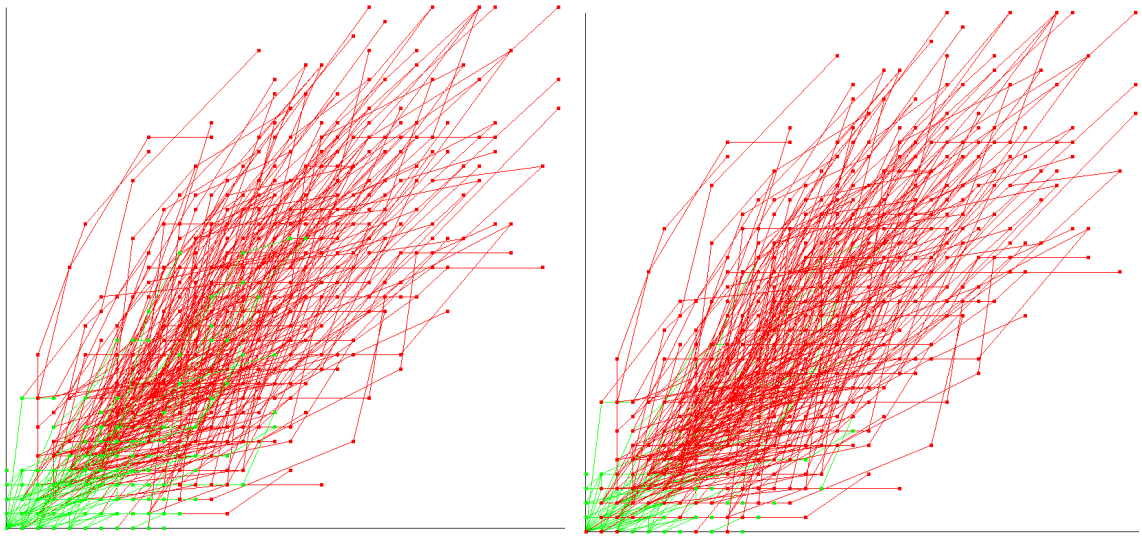


(a) Malignant class on top on PC (b) Benign class on top on PC (c) Malignant class on top on SPC (d) Benign class on top on SPC

Figure 26. Wisconsin Breast Cancer dataset on PC and SPC.

Figure 26 shows the WBC dataset on PC and SPC plots. In both types of plots the dataset samples are heavily overlapped, however, the benign class (green) is concentrated towards the bottom of both plots. The benign class has 1.86 times more samples than the malignant class (red), but the malignant class consumes more area of the plot.

We remove the mitosis attribute for the SPC plot as it requires an even number of attributes. This decision was made from analysis of DT (see Appendix B), that showed the mitosis feature was not used for complete set separation in DT. As in the previous datasets the decision tree model also used CART with Gini impurity criterion, and greedy approach on the best split.



(a) Benign class on top on DSC2 (b) Malignant class on top on DSC2

Figure 27. DSC2 of the WBC dataset.

Figure 27 shows the WBC dataset on DSC2. It is shown that the benign class (green) exists mostly in the bottom left corner and has short polyline growth whilst the malignant class (red) also starts in the bottom left corner but has long polyline growth. To reduce the area of overlap to a manageable level we employed attribute scaling to find regions of overlap that would have meaning to a ML model.

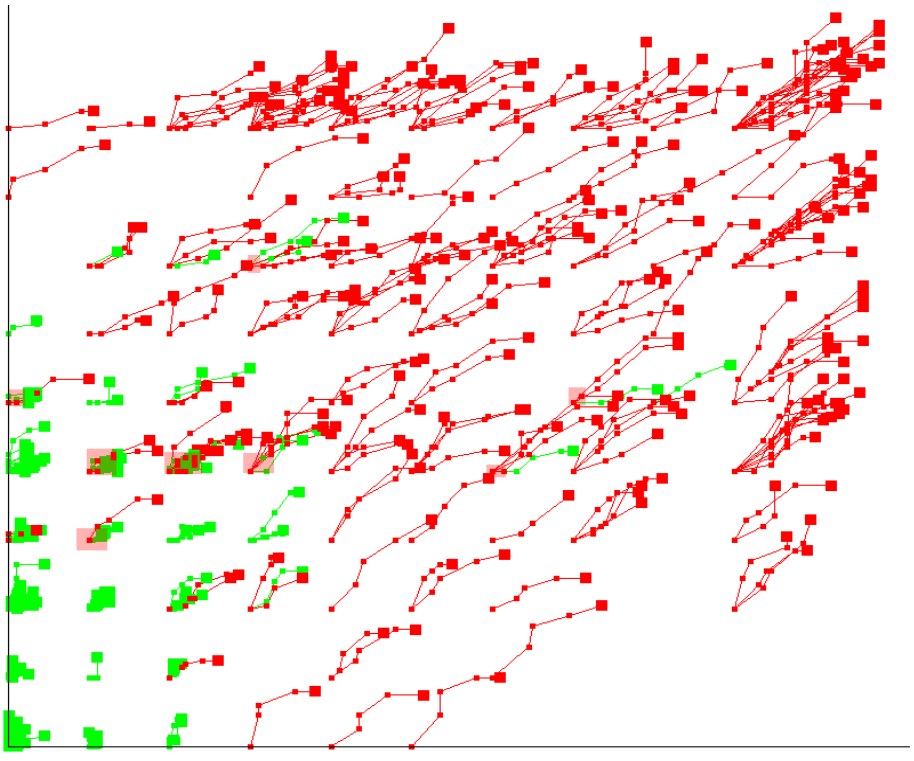


Figure 28. WBC dataset on DSC2 after upscaling the 1st attribute-pair.

Using the DT analysis in Appendix B, it was difficult to decide attributes of interest because of the many branches and deep level of the decision tree. Unlike the Iris dataset the WBC dataset requires multiple attributes to separate the two classes. It was noticed that the uniformity of cell size and uniformity of cell shape attributes were a reoccurring attribute that the decision tree used to create rules. Thus, we made these two attributes the attribute of interest and placed them as the first attribute-pair on DSC2 (Figure 28). After upscaling, the 1st attribute-pair we were able to develop a clearer picture of where proper overlap can exist for a *worst-case* split. Proper overlap refers to areas of overlap that likely will not be ignored by the classier algorithms. The red squares in Figure 28 represent areas that would clip samples for a validation set.

Next we further escalated WBC data visualization by utilizing principal component analysis as an attempt to preserve global structure in only two dimensions. These two principal components would become our attributes of interest and accordingly placed as the first attribute-pair (Figure 29). Compared to Figure 28 it became easier to identify which samples would result in a *worst-case* split. To have enough samples to fill a validation set we used overlapped samples and samples that were near the boundaries of the two classes.

For some datasets dynamic scaffolding coordinates may not be necessary when looking for worst case splits. The two attributes of interest (in this case the principal components) in the form of a scatterplot would be just as effective. However, when combining the WBC attributes with the two principal components we were able to capture the general structure of the samples. One obvious trend in Figure 29 that does not show up in Figure 30 is that malignant cases tend to have large values across many attributes whereas benign cases tend to have attribute values close to zero. The end-user can gain a better insight into which samples are likely malignant and which ones are benign. It could also be useful in determining edge cases of benign tumors that may transform into a malignant tumor later. Of course, this line of logic would require going through the scientific process to determine if benign cases with long polylines are more likely to turn into malignant cases. With Figure 30 these higher-level insights would not be possible as the points themselves have no differentiation besides location and class color.

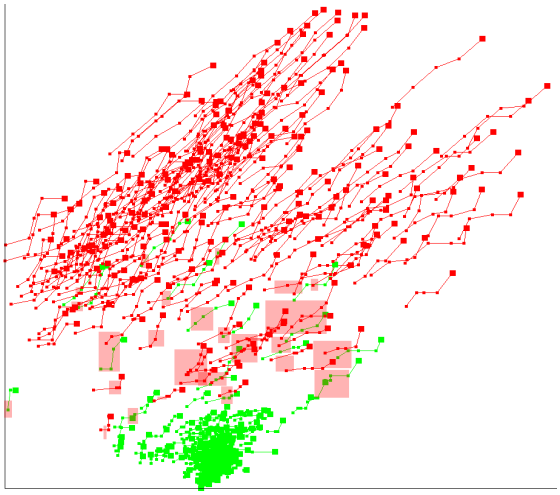


Figure 29. WBC dataset with two Principal Components.

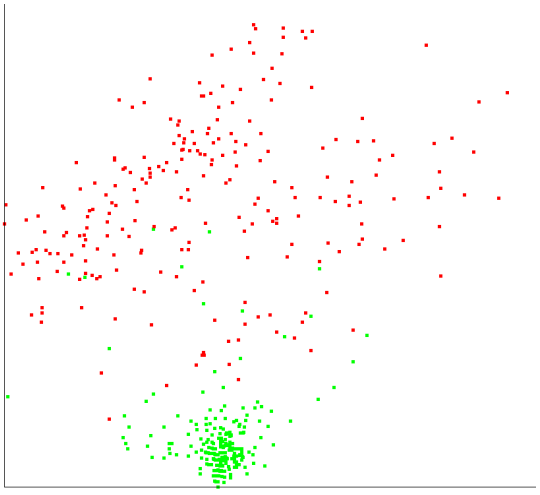


Figure 30. WBC Principal Components.

3.4 MNIST Handwritten Digits with Autoencoder and t-SNE Components

The datasets we experimented in previous sections were relatively small in both dimensions, and the number of cases. The goal of this section is to explore capabilities of DSC visualization for a much larger dataset. For this purpose, we selected the MNIST Handwritten digits dataset (MNIST) [3], which contains 60,000 samples of digits on a 28 × 28-pixel grid in a training set. There is an additional test set containing 10,000 samples. The dataset is available from Kaggle.com. The pixel values are represented in grayscale between 0

and 255. The 28 × 28-pixel grid was converted to a 784-dimensional dataset. In the previous sections, we were able to visualize smaller datasets in SPC and parallel coordinates and have shown advantages of DSC over them. For larger MNIST dataset, a SPC or PC our implementations did not handle the 784 attributes. These methods require special treatment to deal with heavy occlusion. The decision three approach, which was successful for smaller datasets above, has a limited applicability to the MNIST dataset, because the produced DT was very large, deep and with accuracy only around 79%. Respectively, finding attributes of interest in the MNIST DT was difficult.

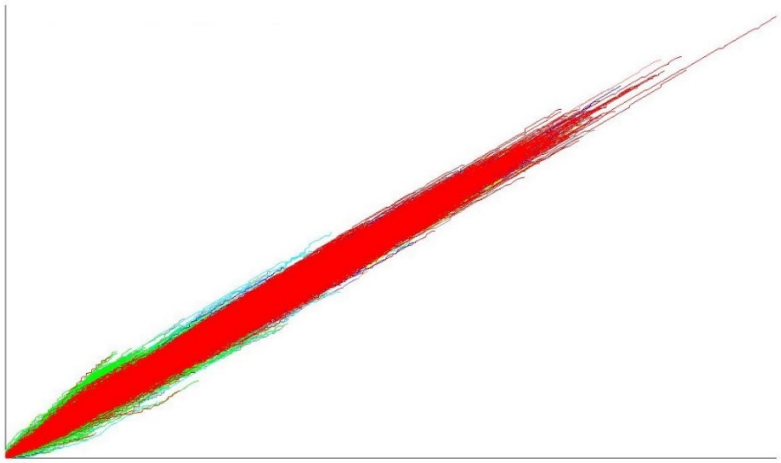


Figure 31. 784 Attributes of MNIST on DSC2.

Unlike SPC and PC we were able to render a plot that fit inside the application window using DSC2. The plot in Figure 31 gave lack of insight into the dataset as all ten digits were stacked on top of each other. The dimension reduction technique using Autoencoder was more successful. In prior work [5] two digits of MNIST were analyzed together using 32 autoencoder features with a high model accuracy. Therefore, we reduced the MNIST dataset to 32 autoencoder attributes and plot them on DSC2. Unlike [5] that compared only two digits, we chose to compare a single digit to all other digits.

Figure 32 provided interesting digit patterns of the MNIST dataset on DSC2. Each digit had a slightly different polyline growth pattern and density. However, there is large amounts of overlap between the ten-digit classes, and it was difficult to visually separate the classes from each other. The next attempt was to use t-SNE components as attributes of interest (Figure 33), considering that t-SNE has been used to visualize MNIST digits with high clarity.

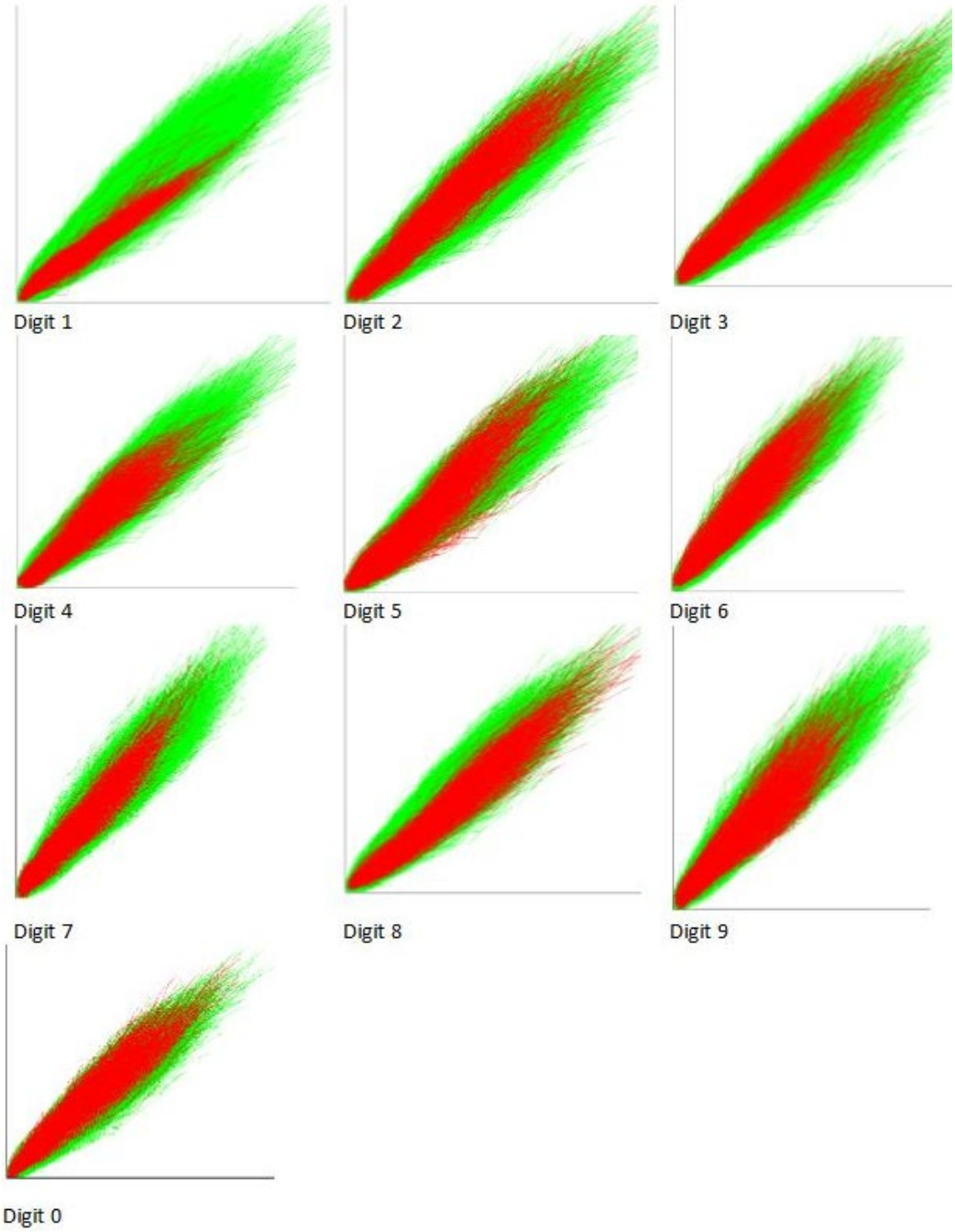


Figure 32. Each MNIST Digit using autoencoder features against all other digits on DSC2.

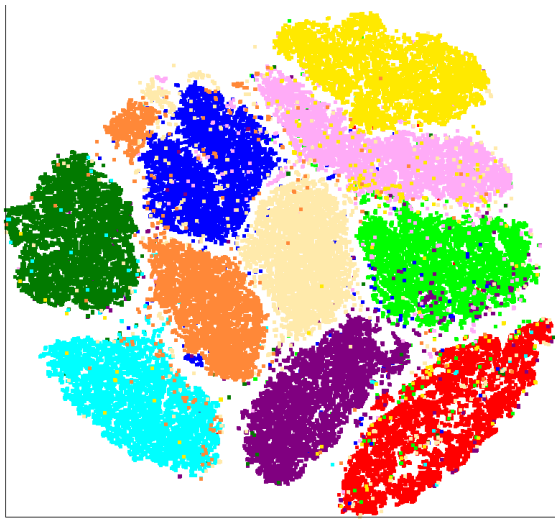


Figure 33. t-SNE components for MNIST.

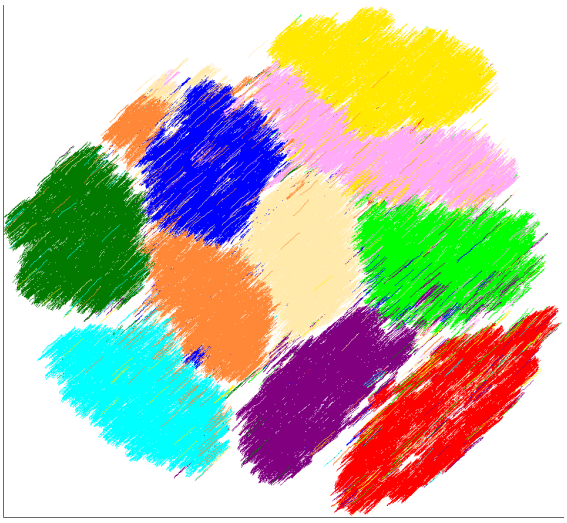
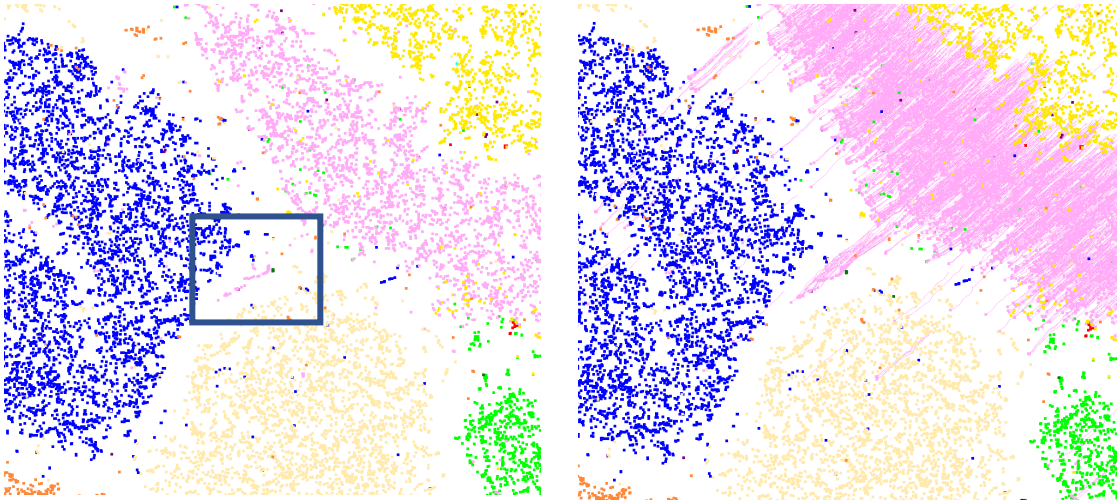


Figure 34. t-SNE + MNIST attributes on DSC2.

We reduced the MNIST dataset from 784 attributes to only 50 truncated Singular Value Decomposition (SVD) components. SVD describes a linear transformation through rotations and scaling. It is essentially a map between different sized vector spaces. We used Truncated SVD which is beneficial when data is sparse. MNIST can be considered sparse as many pixels in a single digit pixel-grid are without ink. With these 50 SVD components we generated two t-SNE components. However, t-SNE has several drawbacks. t-SNE components cannot be applied to unseen data without applying them simultaneously with the training data due to the lack of a parametric embedding [16], which leads to t-SNE algorithm being considered as a difficult to interpret or *black box* dimension reduction technique. Next, t-SNE does not preserve distance between clusters, nor does it preserve cluster density [13]. It is also very visible in our Figure 1 for WBC data as we discussed in section 1. t-SNE can be simplified to a dimensional reduction technique that does not preserve global structure but preserves local neighborhoods [8]. t-SNE uses a perplexity parameter which we set to 30. The creator of t-SNE recommends a perplexity value between 5 and 50 [13]. This parameter is related to how many nearby neighbors any point may have.



(a) t-SNE only scatterplot. (b) t-SNE + DSC2 scaffolds.

Figure 35, t-SNE scatterplot vs DSC2 scaffolds.

Like the WBC with PCA in DSC2 plot from the previous section, we grew our sample polylines from the two t-SNE components which became our attributes of interest. The MNIST SVD components were downscaled by a factor of 0.003 to keep the polylines from growing on top of other clusters. Figure 32 and Figure 33 look very similar when the entire plot is in frame. However, zooming in on the DSC2 plot can reveal important differences between Figure 33 and Figure 34.

Figure 35 reveals that certain points can end up growing into the main cluster of their class when using DSC2 scaffolding on top of t-SNE. There is a risk to this analysis however, since t-SNE does not preserve distances, whilst the scaffolds from the SVD components do preserve distance in SVD. In future work we will consider other ways of visualizing MNIST dataset without having to rely on black box techniques such as t-SNE, but for our current research we found t-SNE and scaffolding to produce class clusters with high clarity. While MNIST dataset is not a high-risk application, if we were to choose samples for *worst-case* analysis, we would pick samples that lie in a class cluster but are not part of that class.

3.5 Comparison of worst and average validation sets for different classifiers

This section is devoted to comparison of performance on average and worst validation sets, where average validation sets are obtained by using k-fold cross validation and the worst validation sets are obtained by visual methods described in the previous sections. It is important to know how big the difference is between accuracy for the average and worst validation sets for different classifiers to be able to select better ones.

Classification results were obtained from **eight standard ML classifiers** in the sci-kit learn Python library [13] using 10-fold cross validation. The single 10-fold cross validation tested against eight models was compared to the validation split found using several box-clipping areas in Figure 29. The samples in the multiple boxes were clipped using Cohen Sutherland algorithm into a validation set of 67 samples which is 9.81% of the entire dataset. The PCA components were removed, and the mitosis attribute was added back to both the training and validation set. The goal of our DSC2 visualization was to select samples from the original dataset that may lead to an upper estimate of the *worst-case* split. We refer to this validation split as an *upper estimate*, as this methodology does not guarantee the *absolute worst-case* split.

For each ML model all parameters were kept as default, except for the multilayer perceptron which was given an additional hidden layer from the default of one hidden layer. The classifiers are as follows: Decision Tree (DT) using CART algorithm and greedy approach, Support Vector Machine (SVM) with linear approach and l2 penalty, Random Forests (RF) with 100 estimators, K-Nearest Neighbors (KNN) with 5 neighbors and uniform weights, Logistic Regression (LR) with l2 penalty, Gaussian Naïve Bayes (NB), Stochastic Gradient Descent (SGD), and Multilayer Perceptron (MLP) with one hidden layer of 100 hidden units.

Table 3. Standard 10-fold Cross Validation model accuracy on WPC data.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10	AVG
DT	95.7	91.3	95.7	92.6	95.6	91.1	94.1	98.5	97.1	95.6	94.7
SVM	92.8	98.6	95.7	94.1	98.5	97.1	97.1	100	98.5	98.5	97.1
RF	92.8	94.2	95.7	94.1	98.5	97.1	98.5	98.5	98.5	98.5	96.6
KNN	91.3	98.6	95.6	94.1	100	97.1	98.5	100	98.5	98.5	97.2
LR	92.7	97.1	94.2	94.1	100	97.1	95.5	100	97.1	100	96.8
NB	92.7	95.6	94.2	94.1	98.5	95.6	97.1	97.1	98.5	97.1	96.1
SGD	95.7	92.8	95.7	94.1	100	91.2	95.6	98.5	98.5	100	96.2
MLP	89.8	89.9	94.2	92.6	100	94.1	97.1	100	98.5	98.5	95.5

Table 3 shows that standard ML algorithms can classify the WBC dataset within 94% to 97% average accuracy without any additional processing, dimensional reduction, or feature engineering. The lowest split

is resulted in 89.8% and the highest split is resulted in 100%. The best performing model was KNN and SVM, and the worst performing model was DT classifier.

Table 4. Upper estimate of the worst split from PCA-DSC2 analysis for WBC data.

Model	Accuracy
DT	62.12%
SVM	62.12%
RF	65.15%
KNN	60.60%
LR	63.63%
NB	72.72%
SGD	57.58%
MLP	60.60%

Table 4 shows that despite the strong model accuracies obtained in Table 3, all eight classifiers had an accuracy between 57% and 72% in the upper estimate of the **worst-case scenario.** In life-critical and other high-risk applications knowing the worst performance of a model can influence reliance on the model and the possibilities of incorporating additional models into decision-making as a safeguard. In this case 10-fold cross validation accuracy suggests using KNN or SVM classifiers, but the model that performed the best on the estimate of the most difficult split was Naïve Bayes by a margin of 7.57% of the next best model Random Forest.

4. DSCViz Software

DSCViz provides a GUI application to control dynamic scaffolding coordinates, parallel coordinates, and shifted paired coordinates plots. It is available upon request for not commercial use. After the dataset is uploaded information about the dataset will populate in the top left corner of the window. The user then selects which of the four plots to create using the radio button and clicking generate plot.

From here the user has the capabilities to drag and zoom in real time using GPU rendering by making various OpenGL calls. The dataset vertices are stored in the GPU memory for fast access when doing matrix operations.

DSCViz has been used on datasets such as MNIST, which can include 40 million data points, of course many are overlapped and do not render. If the dragging and zooming is slow a user may use hiding all the classes and markers, dragging the plot, and then reactivating. There are no immediate slowdowns for small datasets such as WBC and Iris, or MNIST after dimensional reduction techniques are applied.

A user can establishe a specific order of features in the UI. Similarly, classes can also be reordered. Individual class markers, class polylines, as well as the plot axes all have toggles to hide. The attribute markers can be controlled at the attribute level by toggling them in the highlight column. There is a general slider that controls the transparency of unselected attributes. The slider set at 0 will completely hide the attributes.

DSCViz gives the user the ability to clip samples from the dataset. The line clipping algorithm is Cohen-Sutherland. There is also the option to clip samples by any vertex or end vertices. Unlike the line clipping method, the vertex methods only clip samples that have a vertex inside the clipping area. Sequential clips can be added and saved any time. The user may remove and reset the clip entirely. All clipping samples are moved into the validation set. Figures 36-38 illustrate DSCViz.

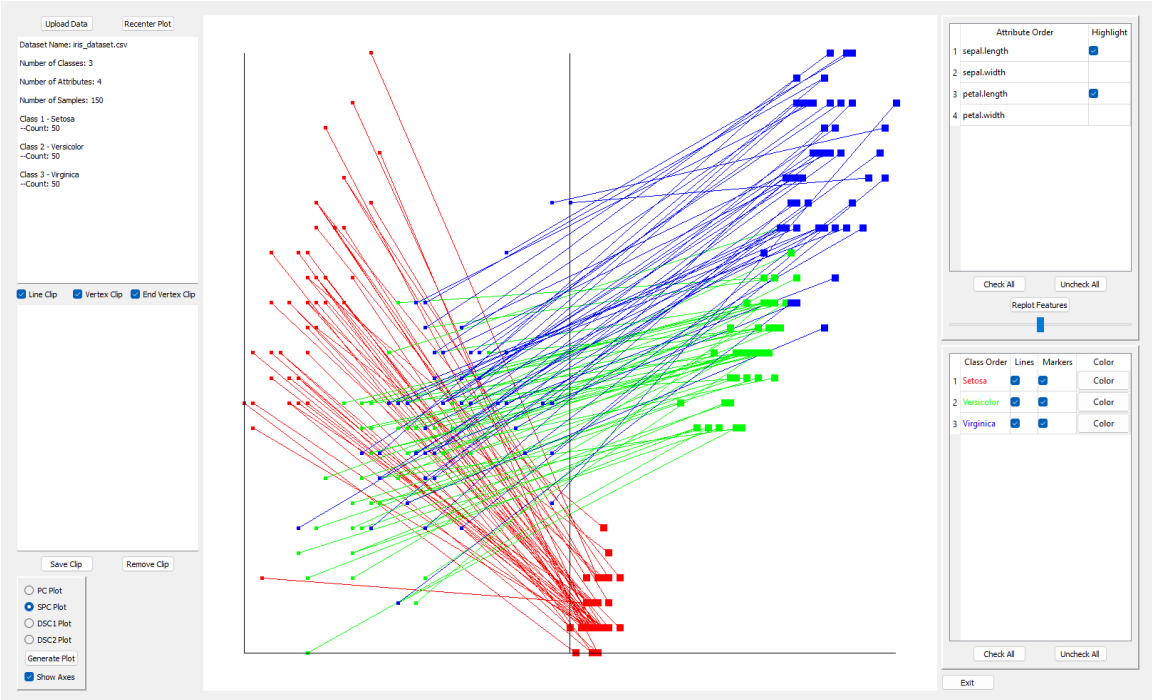


Figure 36. DSCViz application running SPC plot.

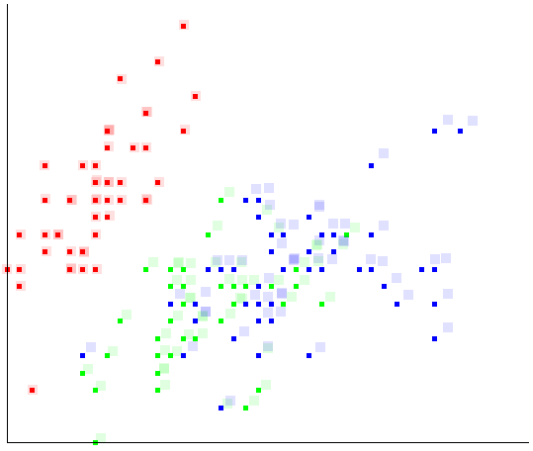


Figure 37. Lowering attribute transparency.

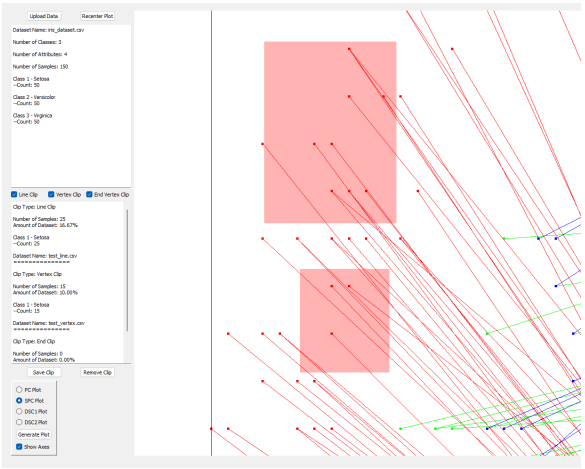


Figure 38. Two clipping areas on DSCViz.

5 Conclusions

This study contributes to visual and interpretable machine learning methods by developing DSC1 and DSC2 systems that can be used for multidimensional visualization, analysis, and classification. DSC1 and DSC2 have self-service components that allow domain experts to change, add, or remove attributes and select regions of highly condensed samples for model selection.

Hyperblocks as interpretable data units are used to highlight attributes of separation within a dataset as a computationally efficient alternative to genetic or brute force algorithms for attribute order permutation selection. When HBs are unable to provide adequate attributes of separation, we escalated to various dimension reduction techniques that allowed for visualizing dimensionally rich datasets like MNIST.

The results in Table 2 and Table 4 show the lower accuracy of standard ML models for benchmark datasets when looking for difficult dataset splits. Section 3 provides a framework of visualizing these difficult splits on DSC2. One area that we would like to improve on is visualizing difficult splits in a dataset without reliance on dimensional reduction techniques to provide plot clarity. In future work we look towards developing dimension reduction techniques that preserve hyperblock structure or adapting other dimension reduction techniques like multidimensional scaling and self-organizing maps.

References

[1] Brown J. Visualizing Multidimensional Data with General Line Coordinates and Pareto Optimization. Central Washington University, All *Master's Theses.* 2017. 898. Available at https://digitalcommons.cwu.edu/etd/898

[2] Charytanowicz M, Niewczas J, Kulczycki P, Kowalski PA, Łukasik S, Żak S. Complete gradient clustering algorithm for features analysis of x-ray images. In: Information technologies in biomedicine 2010 (pp. 15-24). Springer.

[3] Deng L. The MNIST database of handwritten digit images for machine learning research. IEEE signal processing magazine. 2012 Oct 18;29(6):141-2.

[4] Di Cicco V, Firmani D, Kouras N, Merialdo P, Srivastava D. Interpreting deep learning models for entity resolution: an experience report using LIME. In: Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management 2019 Jul 5 (pp. 1-4).

[5] Dovhalets D, Kovalerchuk B, Vajda S, Andonie R. Deep learning of 2-D images representing n-D data in general line coordinates. In: International Symposium on Affective Science and Engineering ISASE2018 2018 (pp. 1-6). Japan Society of Kansei Engineering, https://www.jstage.jst.go.jp/article/isase/ISASE2018/0/ISASE2018_1_18/_pdf

[6] Dua, D. and Graff, C. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science, 2019

[7] Fisher RA. The use of multiple measurements in taxonomic problems. Annals of eugenics. 1936 Sep;7(2):179-88.

[8] Kobak D, Berens P. The art of using t-SNE for single-cell transcriptomics. Nature communications. 2019 Nov 28;10(1):1-4.

[9] Kovalerchuk B. Enhancement of cross validation using hybrid visual and analytical means with Shannon function. In: Beyond Traditional Probabilistic Data Processing Techniques: Interval, Fuzzy etc. Methods and Their Applications 2020 (pp. 517-543). Springer, Cham

[10] Kovalerchuk B. Visual knowledge discovery and machine learning. Springer; 2018.

[11] 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.

[12] Kovalerchuk B, Hayes D. Discovering Interpretable Machine Learning Models in Parallel Coordinates. In2021 25th International Conference Information Visualisation (IV) 2021 Jul 5 (pp. 181-188). IEEE.

[13] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J. Scikit-learn: Machine learning in Python. Journal of machine Learning research. 2011 Nov 1;12:2825-30.

[14] 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 Aug 13 (pp. 1135-1144).

[15] Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence. 2019 May;1(5):206-15.

[16] Van Der Maaten L. Learning a parametric embedding by preserving local structure. In: Artificial intelligence and statistics 2009 Apr 15 (pp. 384-391). PMLR.

[17] Van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008 Nov 1;9(11).

[18] Wagle SN, Kovalerchuk B. Self-service Data Classification Using Interactive Visualization and Interpretable Machine Learning. In: Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery 2022 (pp. 101-139). Springer, Cham.

[19] Wolberg WH, Street WN, Mangasarian OL. Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates. Cancer letters. 1994 Mar 15;77(2-3):163-71.

[20] –Coxeter, H. S. M. Regular Polytopes, 3rd ed. New York: Dover, pp. 122-123, 1973.

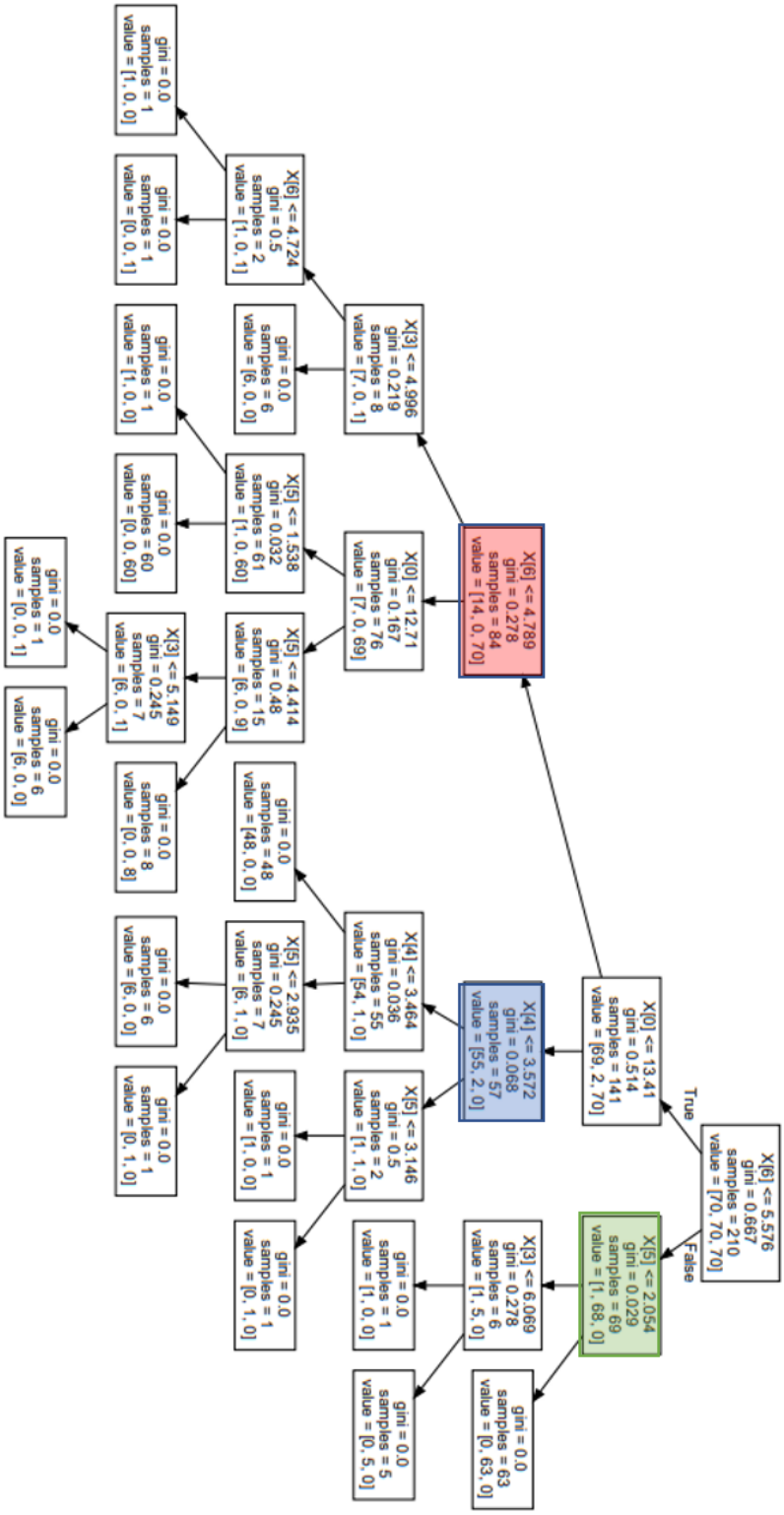
[21] Inselberg, A., Parallel Coordinates: Visual Multidimensional Geometry and its Applications, Springer, 2009

[22] Johnson WB, Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. Contemp. Math., 1984;26:189-206.

[23] Di Cicco V, Firmani D, Koudas N, Merialdo P, Srivastava D. Interpreting deep learning models for entity resolution: an experience report using LIME. In: Proc. of the 2nd Inter. Workshop on Exploiting Artificial Intelligence Techniques for Data Management 2019 Jul 5 (pp. 1-4).

[24] Kovalerchuk B, Andonie R, Datia N, Nazemi K, Banissi E. Visual Knowledge Discovery with Artificial Intelligence: Challenges and Future Directions. In: Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery 2022 (pp. 1-27). Springer, Cham.

Appendix A – Seeds Dataset Decision Tree



Appendix B – Wisconsin Breast Cancer Dataset Decision Tree