Spring 2022

Interpretable Machine Learning for Self-service High-risk Decision Making

Charles Recaido
Central Washington University, recaidoc@cwu.edu

INTERPRETABLE MACHINE LEARNING FOR SELF-SERVICE HIGH-RISK DECISION MAKING

A Thesis

Presented to

The Graduate Faculty

Central Washington University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Computational Science

---

by

Charles Patrick Recaido

May 2022

CENTRAL WASHINGTON UNIVERSITY

Graduate Studies

We hereby approve the thesis of

Charles Patrick Recaido

Candidate for the degree of Master of Science

APPROVED FOR THE GRADUATE FACULTY

_____ _____

Dr. Boris Kovalerchuk

_____ _____

Dr. Razvan Andonie

_____ _____

Dr. Szilard VAJDA

_____ _____

Dean of Graduate Studies

ABSTRACT

INTERPRETABLE MACHINE LEARNING FOR SELF-SERVICE HIGH-RISK DECISION MAKING

by

Charles Patrick Recaido

May 2022

This research contributes to interpretable machine learning via visual knowledge discovery in General Line Coordinates (GLC). The concepts of hyperblocks as interpretable dataset units and GLC are combined to create a visual self-service machine learning model. Two variants of GLC known as Dynamic Scaffold Coordinates (DSC) are proposed. DSC1 and DSC2 can map in a lossless manner multiple dataset attributes to a single two-dimensional (X, Y) Cartesian plane using a dynamic scaffolding graph construction algorithm.

Hyperblock analysis is used to determine visually appealing dataset attribute orders and to reduce line occlusion. It is shown that hyperblocks can generalize decision tree rules and a series of DSC1 or DSC2 plots can visualize in a lossless manner n-D data in accordance with a decision tree model. For large decision trees with many branches such as MNIST handwritten digits where hyperblock discovery was hampered, dimensionality reduction techniques such as principal component analysis, singular value

decomposition, and t-distributed stochastic neighbor embedding were used to create new attributes of interest for visual class separation.

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. A software package referred to as Dynamic Scaffold Coordinates Visualization System (DSCViz) was created to showcase the DSC1 and DSC2 systems. DSCViz expands the end-user's capabilities by offering several functions such as real-time drag and zoom, scaling techniques, sample clipping, attribute reordering, and the ability to hide classes or change their colors. DSC2 was used to estimate and visualize the worst-case validation splits in the Wisconsin Breast Cancer, Iris, and Seeds dataset. DSC2 was also plotted against MNIST Handwritten digits to determine its feasibility in large datasets. In general, the technique of estimating worst-case validation splits is important for every high-risk application.

ACKNOWLEDGMENTS

First and foremost, I would like to give thanks to Central Washington University and the CWU School of Graduate Studies and Research for giving me the opportunity to pursue higher graduate level education on a short notice. Next, I would like to thank the Computer Science department and faculty for accepting me into the Computational Science program to pursue studies in a field I have great interest in.

With much gratitude I would like to thank the graduate committee for guiding me in the graduate program and teaching solid CS fundamentals which I aspire to use in the industry.

I would like to express my deep gratitude to Dr. Boris Kovalerchuk for taking me under his wing and establishing a research path that I could pursue. Your research is phenomenal and opened my eyes to the world of data visualization. Especially from the uniqueness of General Line Coordinates.

I would like to thank Dr. Razvan Andonie for establishing the program requirements and providing me valuable feedback in the theoretical side of CS, especially in general algorithms and machine learning.

I would like to thank Dr. Szilard Vajda for his brutal operating systems course, but that course greatly improved my foundations at CS, as I was severely lacking due to my background being Physics with little programming experience. I gained the most from that course.

Finally, I would like to thank my wife and my fellow cohort, Stephen, Jason, and Juan for helping me in areas that I got stuck, considering I was green to programming.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

CHAPTER 1

INTRODUCTION

Technological advancements continue to push the role of decision making onto machines. Machines have the raw computational power to analyze datasets with many samples and attributes in a timely manner. The information from the datasets is used to form predictions using a machine learning (ML) model such as k-nearest neighbors, support vector machines, decision trees, neural networks, and more. The results from these predictive models are interpreted either autonomously or by a human for decision making. Many but not all advanced ML models can be complex *black boxes* for the end users. Decision making using a *black box* method requires trust from the end user because of various assumptions [1, 2] that take place during the development process of the model.

Decision making can be a life critical or high-risk process. One dataset this work considers is the Wisconsin Breast Cancer (WBC) dataset. 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 [3]. Other high-risk applications to be considered are missile launches and certain investment strategies.

ML algorithms often rely on random methods of splitting available data into training and validation data. The accuracy of each conducted split as well as the average model accuracy for these splits can be high and considered appropriate dependent on application. However, the accuracy of the *worst-case* split can be significantly lower. For

life critical or high-risk decision making, models with less average accuracy among many splits but an altogether higher accuracy using the estimate of *worst-case* split may be preferable.

The field of data visualization offers many techniques to increase human interpretability of machine learning models and general data analysis [2]. The combination of data visualization and machine learning can even provide self-service models for the end-user [4]. 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.

Our research 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 ML models where the model accuracy can be exaggerated which is unacceptable in applications with high cost of individual errors.

One solution to increase human interpretability of machine learning and provide a basis for a self-service model to the end-user with visual means is General Line Coordinates (GLC). General line coordinates form a multidimensional coordinate system that represents data in the n-dimensional space without data loss based on a unique graph constructing algorithms [5]. Generally, GLC are mapped to three or less dimensions to enhance visual aids. GLC when combined with non-overlapping hyperblocks (HB), regions in an n-dimensional space containing a set of data samples, forms a powerful tool with high level visual clarity. The main advantage of HBs is that they are interpretable,

and all samples of the dataset can be grouped into class - pure HBs known as *atomic* HBs [12]. Therefore, complex datasets can be considered unions of the atomic HBs.

In addition to HBs we explore principal component analysis and t-distributed stochastic neighbor embedding (t-SNE) dimensional reduction techniques. t-SNE components and principal components are used as polyline-origins to influence visual class separation. 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) [6] and genomic sampling [5]. This work considers the complications of t-SNE which is viewed as a *black box* visualization tool and proposes future alternatives.

Our approach involved the development of two *interpretable self-service* ML models with the basis in General Line Coordinates. Both models use a special case of GLC known as Dynamic Scaffolding Coordinates (DSC1, and DSC2). DSC1 has basis in parallel coordinates whilst DSC2 has basis in paired coordinates. DSC1 and DSC2 are lossless data visualization methods that compact the many axes required in DSC1 and DSC2 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 [7]. We adjusted our dynamic scaffolding approach by incorporating hyperblocks and/or 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 application titled DSCViz. 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 (also referred to as the most difficult split) and/or reduce false predictions to improve critical decision making such as tumor diagnosis. This is achieved by analyzing 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 will have zero impact on model performance whereas other overlapped regions will have a great impact when used in the validation set.

More *worst-case* split experiments were conducted on benchmark datasets such as Seeds, Iris, and Wisconsin Breast Cancer [8]. We further investigated our visualization tool on very large datasets such as MNIST handwritten digits [6].

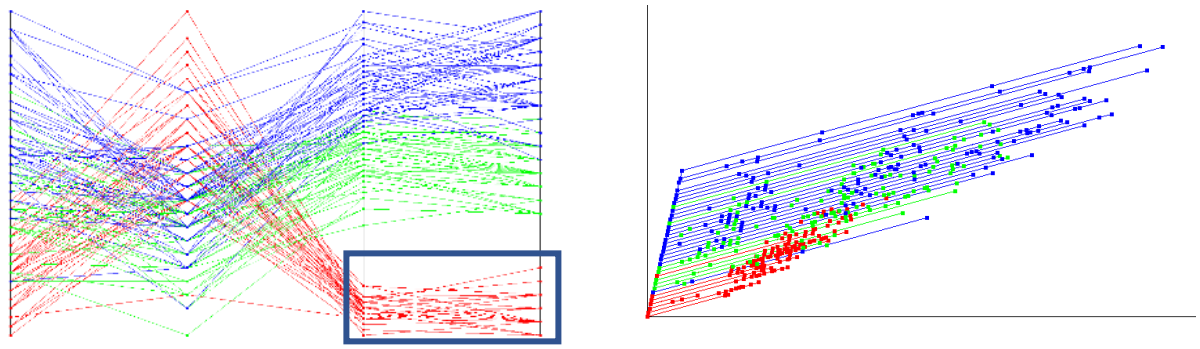
This work explores a novel visualization tool through the combination of a unique general line coordinate system known as Dynamic Scaffolding Coordinates, hyperblocks, and dimensional reduction techniques such as principal component analysis and t-distributed stochastic neighbor embedding. Visualizations of high clarity are produced for multiple benchmark datasets where *worst-case* data splits can be discovered that impact model performance

Related Concepts

There are many forms of GLC visualizations which are dependent on the graph constructing algorithm. GLC generalize multidimensional coordinate systems that use multiple axes such as parallel coordinates (PC), radial (star) coordinates, and shifted paired coordinates (SPC), by transforming them into a line plot with one axis per dimension [8]. This work uses PC and SPC plots to construct two unique GLC visualizations on a two-dimensional plane.

Parallel line coordinates (PC) represent multidimensional data using several one-dimensional axes. Each axis in a PC plot is independent and represents the informational space of one attribute. Figure 1a 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. Parallel line coordinates can be an acceptable tool for decision making. Figure 1a 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 class from each other is more difficult as they exhibit overlap on all four attributes.



(a) Iris dataset in Parallel Coordinates.

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

FIGURE 1: Transformation of a 4-D PC plot to a 2-D GLC plot.

Shifted paired coordinates (SPC) represent multidimensional data using several two-dimensional axes [4]. Each two-dimensional axis holds the informational space of two attributes. Unlike parallel coordinates, SPC can show relationships between two attributes. This leads to a limitation of the SPC plot in which an even number of attributes are required. A dataset with odd number of attributes will require engineering, duplicating, or removing an attribute.

Figure 2 shows the Iris dataset on a shifted paired coordinate 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.

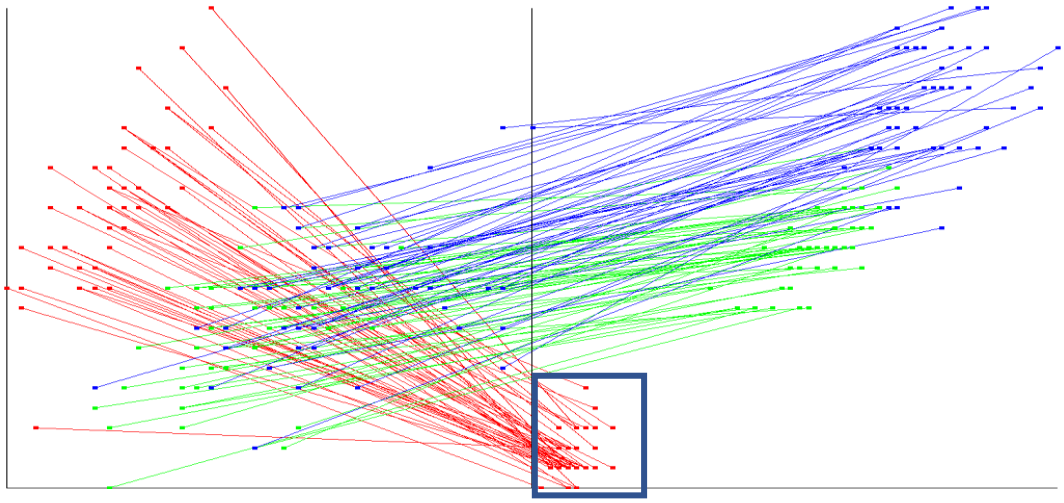


FIGURE 2: Iris dataset on a shifted paired coordinate plot.

Figure 3 is a visually appealing representation of class separation of the Iris dataset in SPC. The red class is completely separated from the red and green class. In Figure 2 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 2 and Figure 3 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 3 was only a few seconds, however, computing permutations of larger multidimensional datasets may not be feasible.

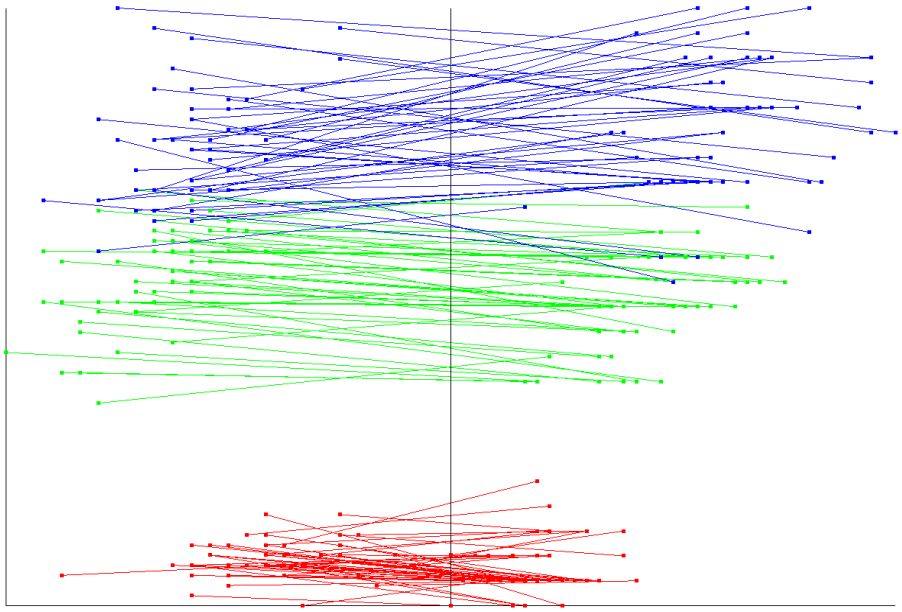


FIGURE 3: Iris dataset on SPC with high visual classification clarity.

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 $\boldsymbol{L} = (L_1, L_2, \dots, L_n)$ [2] 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 may be referred to as a hypercube. HBs 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 dimensions without non-interpretable operations between them. This allows for the separation of different data units that exist among multiple attributes like cell count and cell size for tumor diagnosis during model creation. For example, in k-nearest neighbors, a Euclidean distance search requires a summation of all attributes and the summation of a metric measurement with a count measurement may

not be appropriate to the domain expert. In a previous study hyperblocks were used to generalize decision tree rules [9]. 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 4 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 using decision trees, Merger Hyperblock algorithm [9], and by 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 [9].

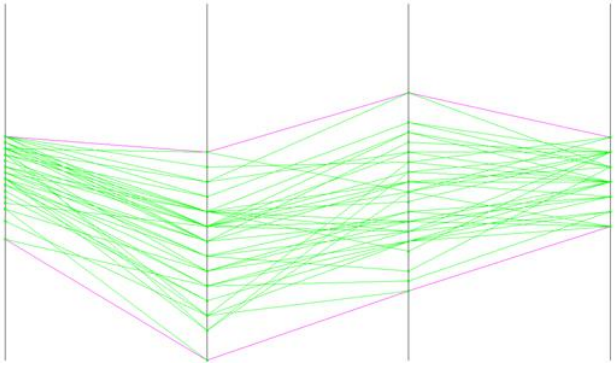


FIGURE 4: Hyperblock in 4-D space with boundary lines (pink) on a PC plot.

Literature Review

In this section we discuss prior works involving general line coordinate systems and their applications. These topics cover interactive visual knowledge discovery in shifted paired coordinates, pareto optimization in general line coordinates, GLC-L coordinate system, and more. We also look at other interpretable machine learning methods such as local interpretable model-agnostic explanations (LIME).

Interactive Visual Self-service Data Classification Approach to Democratize Machine Learning

Wagle and Kovalerchuk., researchers from Central Washington University, were able to increase interpretability of machine learning by introducing the SPCVis software [4]. 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, the researchers developed a genetic algorithm and coordinate order optimizer to find strong attribute permutations that led to class separability. This is especially important for dimensionally rich datasets. Wagle and Kovalerchuk used the SPCVis software and their optimization algorithms 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 other papers.

Data Visualization and Classification of Artificially Created Images

Dovhalets, a researcher from Central Washington University, was able to use interpretable machine learning and General Line Coordinates to transform vectors filled with general attribute information into an image [10]. One important factor to Dovhalets' work is that GLC is a lossless projection from multidimensional data into two dimensions, and the images Dovhalets created contained all the original information of the dataset. Dovhalets' approach allowed him to apply 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 known as dynamic scaffolding, Dovhalets used an alternative GLC system referred to as GLC-L where angles are calculated using a linear function.

Visualizing Multidimensional Data with General Line Coordinates and Pareto Optimization

Brown was able to use GLC-L and Pareto optimization to find a best-case scenario in certain datasets like student performance, and local weather [11] during his research at Central Washington University. 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 CS program. Brown was able to pin down key attributes that would form a basis for his Pareto Subsets, and those subsets would lead to

a Pareto Frontier. Brown was able to condense several parallel coordinate plots into a single GLC-L plot.

"Why Should I Trust You?"

Explaining the Predictions of Any Classifier

Ribeiro *et al*. from the University of Washington developed a model that can explain predictions of arbitrary classifiers by using interpretable models to explain a prediction locally [12]. The researchers refer to this model as Local Interpretable Model-agnostic Explanations (LIME). LIME uses linear surrogate models to explain a single instance by considering the input and the output of any given model, hence "Model-agnostic". LIME uses perturbation around the input to measure changes in the output. LIME has been used in many applications from natural language processing [13] to computer vision [12].

CHAPTER II

DYNAMIC SCAFFOLD COORDINATES

Dynamic Scaffolding Coordinates based on Parallel Coordinates

Dynamic Scaffolding Coordinates based on Parallel Coordinates (DSC1) generalizes 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 multidimensional line. 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 GLC-OS-PC graph construction algorithm (Figure 5) as follows:

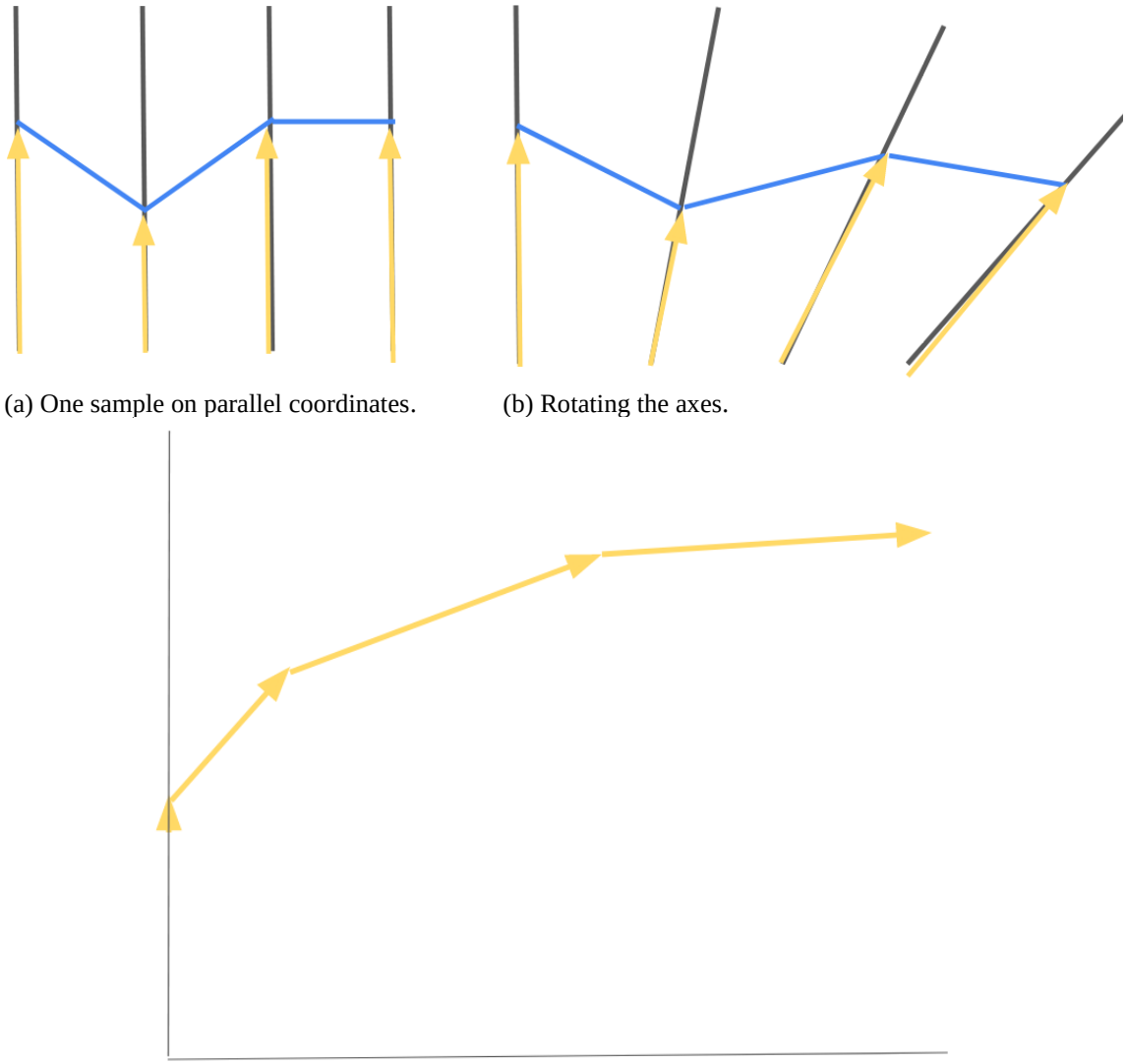
(1) Set up dataset sample coordinates in the same manner as a PC plot.

(2) Apply a rotation transformation for each individual attribute axis with pre-defined angles.

(3) Create a scaffold from the origin to the attribute point for each attribute and 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.



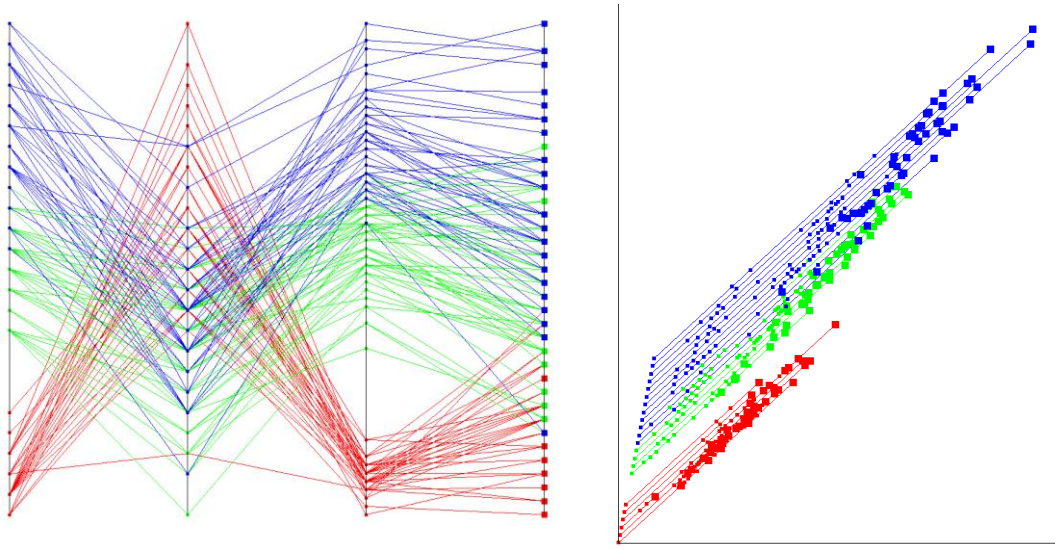
(a) One sample on parallel coordinates.

(b) Rotating the axes.

(c) Connect the scaffolds from tip-to-tail.

FIGURE 5: Visual steps for construction of the DSC1 plot.

Figure 6 shows the final transformation between a parallel coordinate plot and the DSC1 plot on the Iris dataset. An immediate takeaway between the two plots is that DSC1 shows immediate class separation, whereas parallel coordinates require the user to focus on a single attribute.



(a) Iris dataset on PC. (b) Iris dataset on DSC1.

FIGURE 6: Side-by-side comparison of PC 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 known as 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 6), however the possibility exists to change the angle of each attribute as shown in Figure 7.

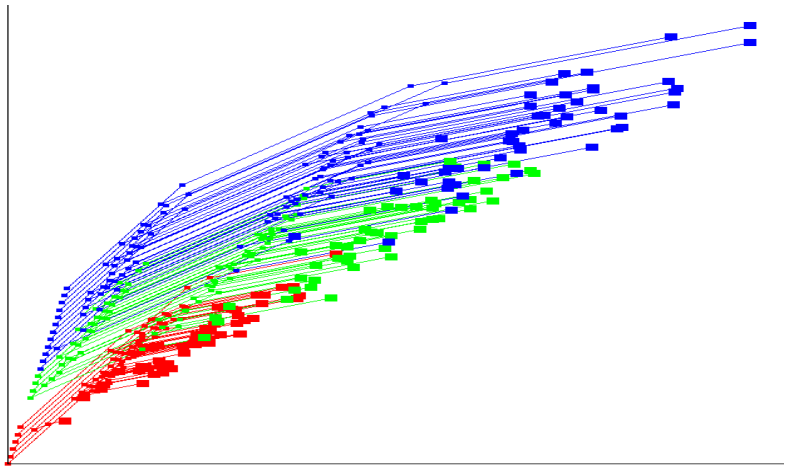


FIGURE 7: DSC1 with a different rotation for each attribute.

Other techniques may be applied to DSC1 such as hiding certain attributes markers and hiding the polylines. Figure 8 is another representation of Figure 6b 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.

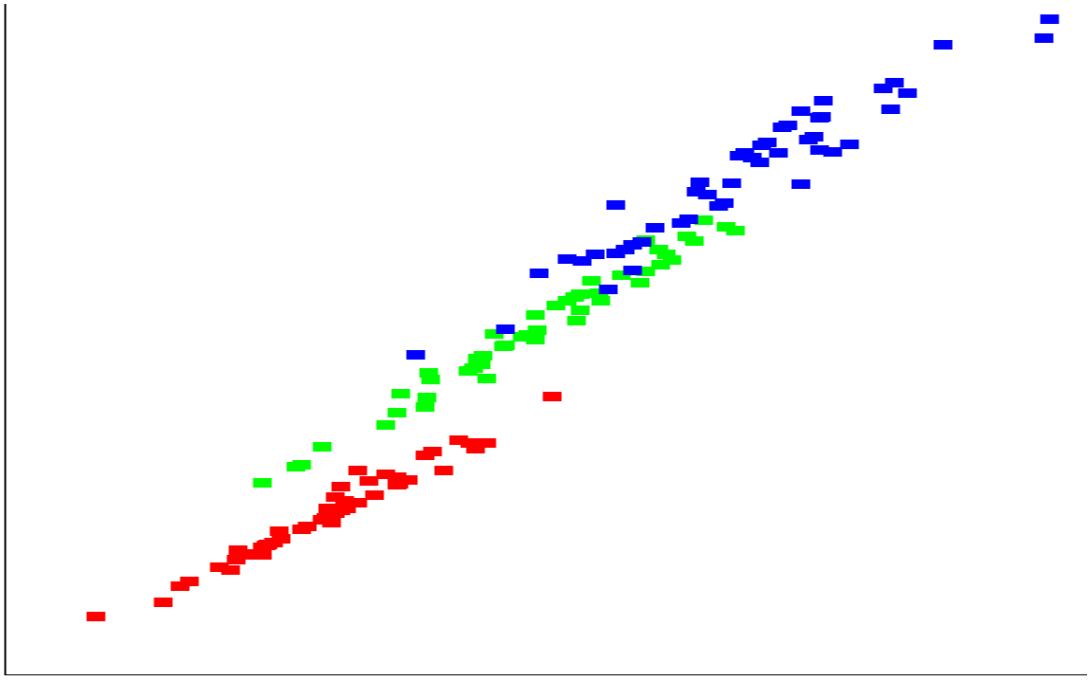
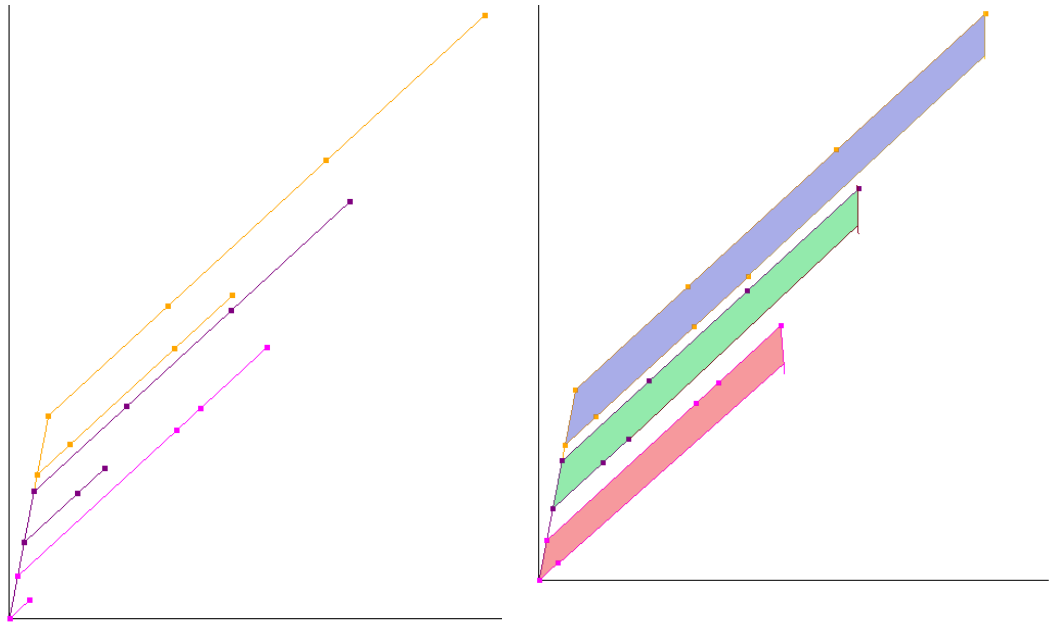


FIGURE 8: Hiding polylines and certain attribute markers on DSC1.

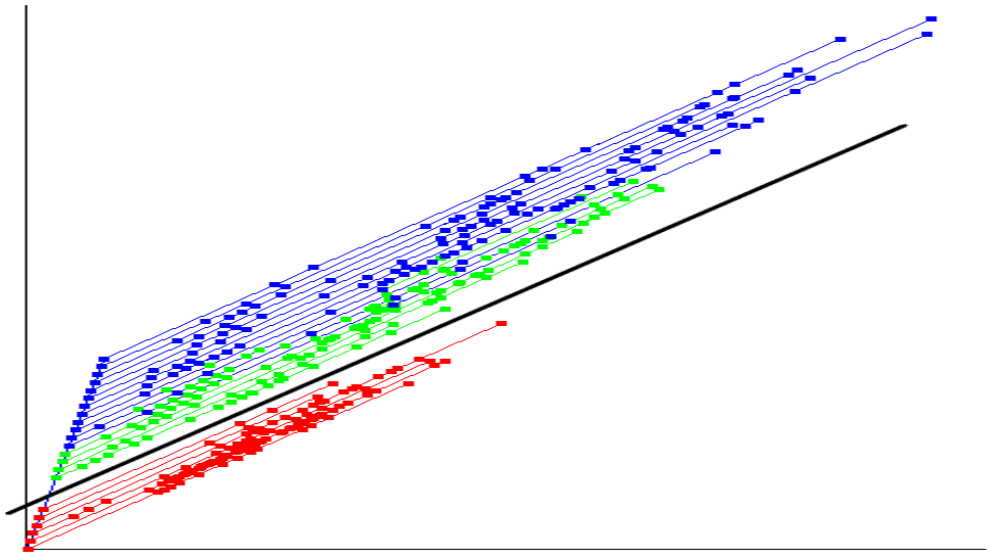
DSC1 grants the user the ability to reduce sample lines to an upper and lower hyperblock boundary line as shown in Figure 9. The alternative visualization reduces line occlusion, which can enhance visual knowledge discovery in sample dense but highly separable datasets. Hyperblocks containing only a few samples relative to the size of dataset do not benefit from this technique.



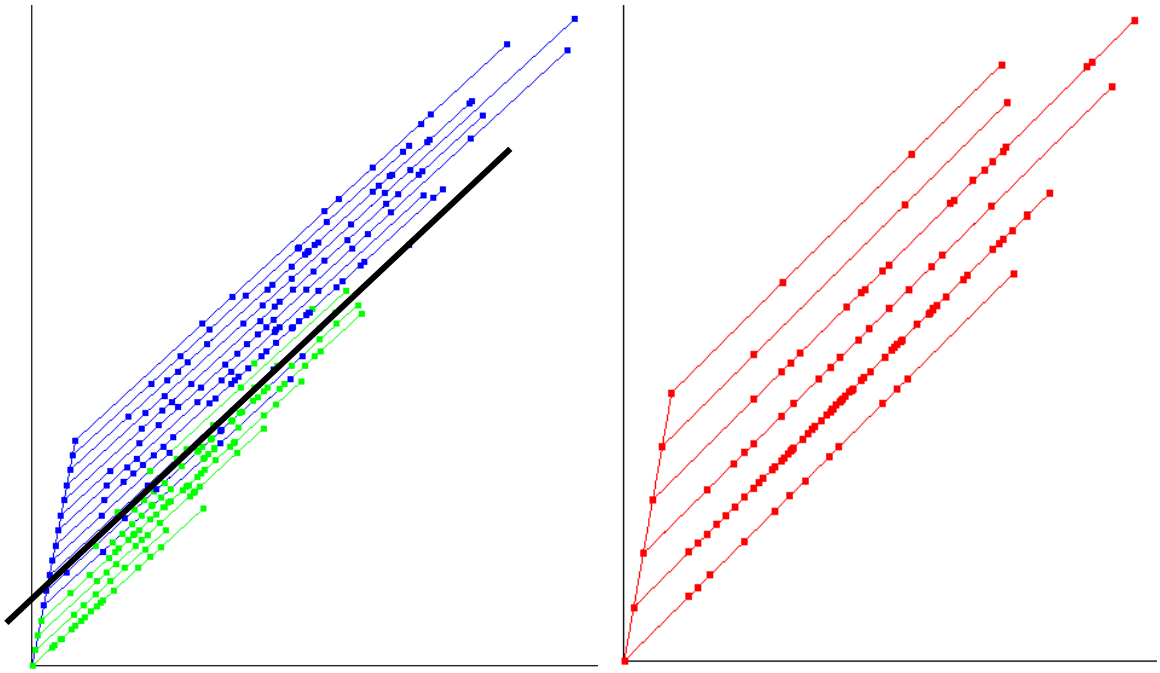
(a) HB boundaries on DSC1. (b) Shaded HBs on DSC1.

FIGURE 9: DSC1 visualized using only HBs.

Dynamic scaffolding coordinates can be used to create visual classifiers. Particularly in DSC1 this can be done via a plot series of DSC1 plots using graphically linear separators as shown in Figure 10.



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



(b) Top split. (c) Bottom split.

FIGURE 10: DSC1 plot series classifier.

Figure 10a shows the graphically linear separator that separates the entire Setosa (red) class from the Virginica (blue) and Versicolor (green) classes. The next step of the classifier would be to separate 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 10b, 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.

Dynamic Scaffolding Coordinates based on Paired Coordinates

Dynamic Scaffolding Coordinates based on Shifted Paired Coordinates (DSC2) generalizes the shifted paired coordinate plot 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.

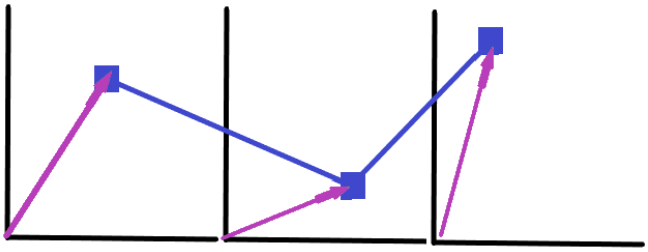
The DSC2 graph construction algorithm (Figure 11) as follows:

(1) Set up dataset sample coordinates in the same manner as a SPC plot.

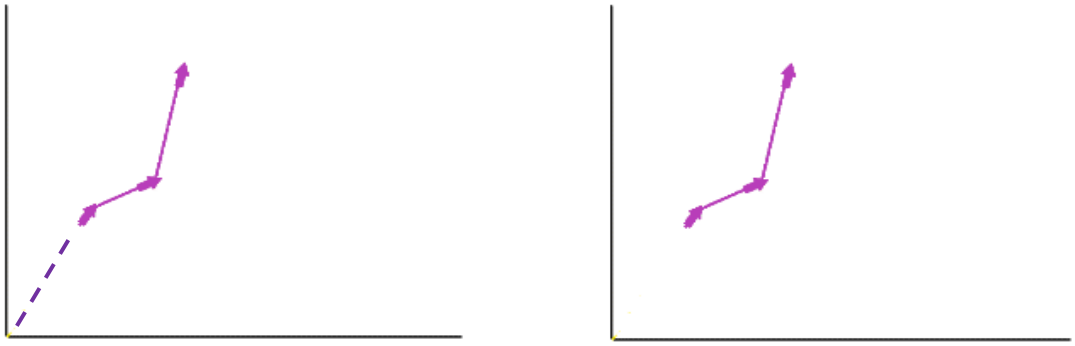
(2) Create a scaffold from the origin to the attribute-pair point for each attribute-pair and for all samples.

(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



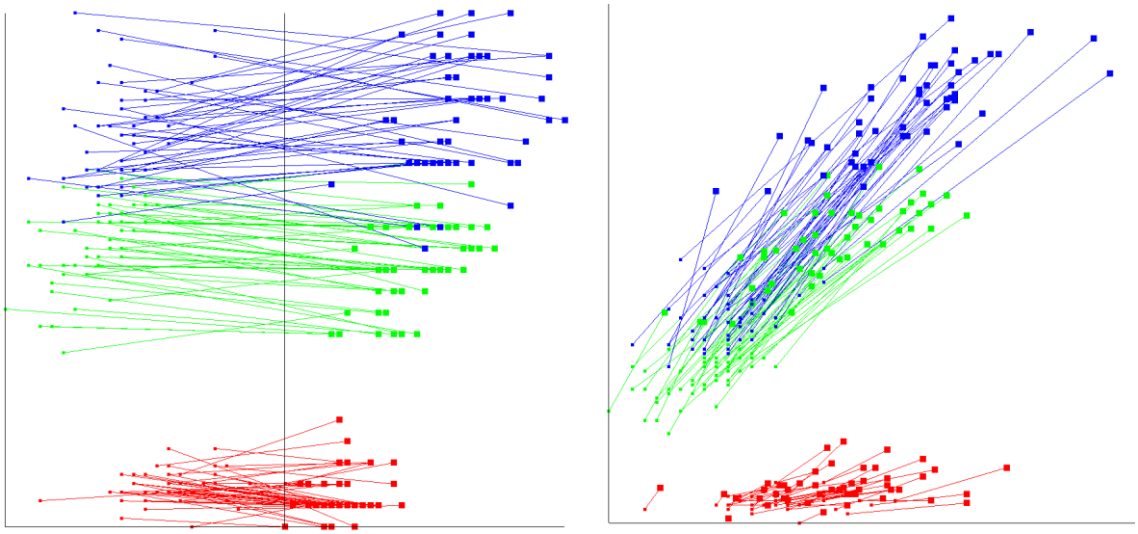
(a) One sample with scaffolds on SPC.



(b) Connecting the scaffolds. (c) Removing the first scaffold.

FIGURE 11: Visual steps for construction of the DSC2 plot.

Figure 12 shows the final transformation between a shifted paired coordinate plot and the DSC2 plot on the Iris dataset.



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

FIGURE 12: Side-by-side comparison of SPC and DSC2 of Iris Dataset.

DSC2 differs from DSC1 in that the hyperblock boundaries are difficult to conceptualize. FIGURE 13 illustrates the boundary boxes from each attribute-pair axes

on shifted paired coordinate and how they condense together to cause overlap within the hyperblock space when transformed into DSC2 coordinates. The Setosa hyperblock 2nd attribute-pair condenses onto the 1st attribute-pair and the 2nd attribute-pair of Versicolor hyperblock condenses onto the 1st attribute-pair of the Virginica hyperblock.

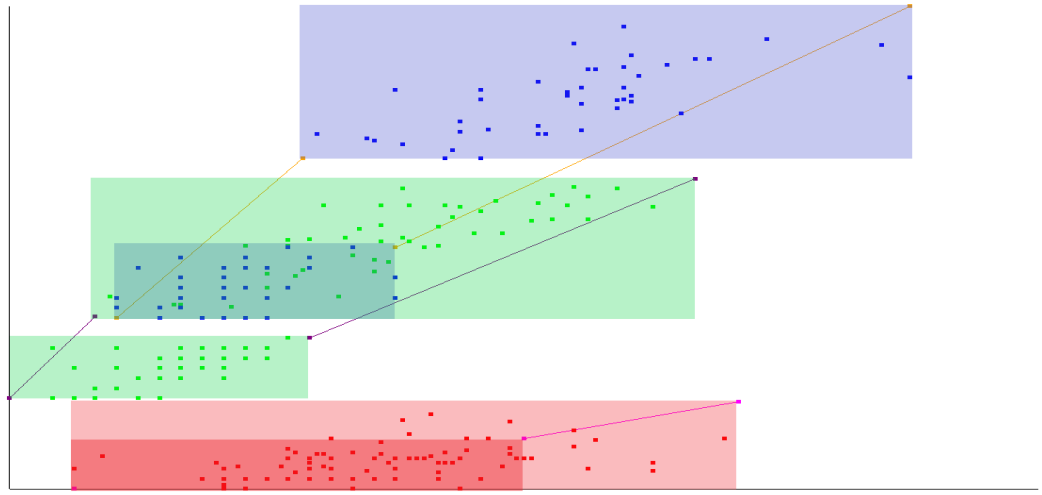


FIGURE 13: Iris hyperblocks on DSC2.

Hyperblock separation can still be identified by looking at the individual boundary boxes made by each attribute-pairing. The 2nd attribute-pair rectangle for all three classes is separated by white space, as well as the 1st attribute-pair rectangle for all classes. In Chapter III we deployed scaling techniques to better separate attribute-pairs on DSC2.

CHAPTER III

METHODS FOR FINDING WORST-CASE SPLITS

In this chapter we estimate the worst-case splits of the Iris dataset, Seeds dataset, and the Wisconsin Breast Cancer (WBC) dataset. 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.

We employed three methods to find these attributes of interest. Hyperblock analysis from decision trees, principal component analysis, and t-distributed stochastic neighbor embedding. These methods are hierarchical in our decision making process, as we prefer to influence class separation on DSC2 using only the data themselves without additional feature engineering or dimensional reduction.

For the Iris dataset we were able to show class separation on DSC2 with only a hyperblock analysis. On the other hand, we were unable to produce quality class separation of the WBC dataset on DSC2 using hyperblock analysis. 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.

Iris Dataset and Hyperblocks

The Iris dataset contains 150 samples of three types of Iris flower (Setosa, Virginica, and Versicolor) [14]. 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. The dataset was obtained from the UCI ML repository [8].

The decision tree (DT) analysis is a simple way to develop HBs for a dataset. The decision tree model used a Classification and Regression Tree algorithm (CART) with Gini impurity criterion, and greedy approach on the best split. Figure 14 forms HBs of the Iris dataset. Each node of the DT represents one HB. To produce non-overlapping HBs from a DT a parent and child node cannot simultaneously be selected as the parent node contains the information of the child, essentially a child is a hyperblock contained inside a parent hyperblock. Going deep into a decision tree to reach 100% purity HBs, where 100 % pure blocks contain only one class, runs the risk of overfitting as HBs with very few or a single sample will be present.

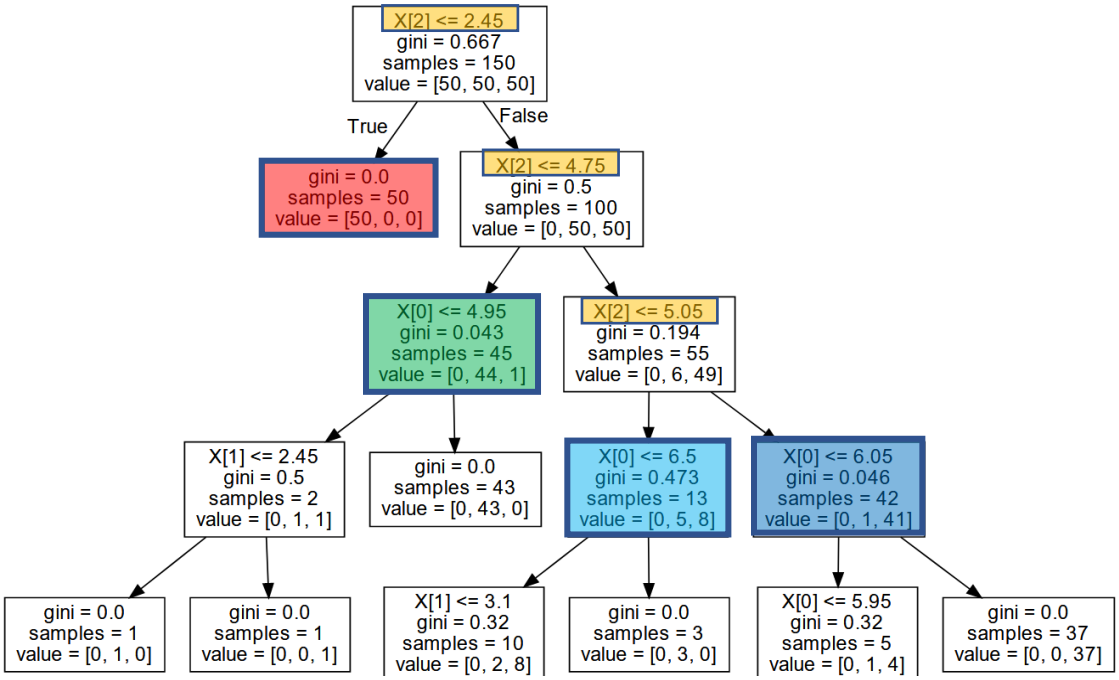
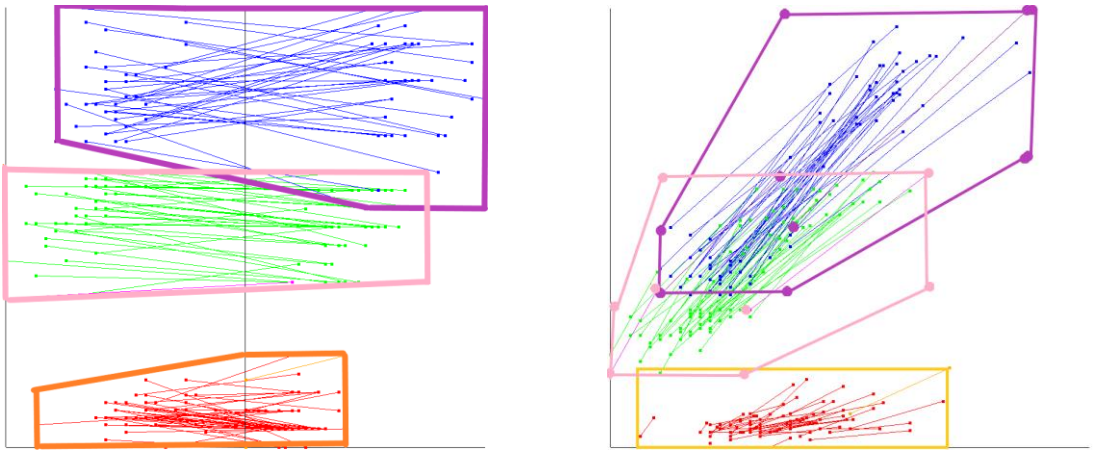


FIGURE 14: Four levels below the root (level 0) of the Iris decision tree.

The red (100% setosa purity), green (97.78% versicolor purity), and blue (97.61% virginica purity) HBs have high purity and contain many samples to reduce the possibility of overfitting. The cyan (61.53% virginica purity) HB is highly impure but still required to keep the information of all dataset samples. The orange highlighted text represent rules that the decision tree made to create these HBs. These DT rules can be applied to multidimensional visualizations when ordering attributes as shown in Figures 15 and 16.



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

FIGURE 15: Side-by-side comparison of SPC and DSC2 of three Iris HBs.

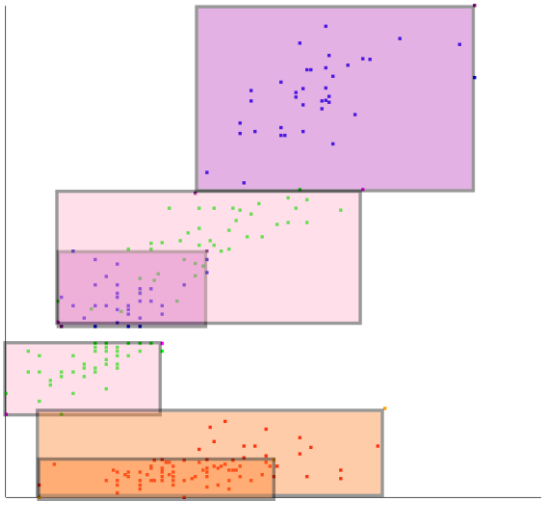


FIGURE 16: Three main class hyperblocks chosen from the decision tree

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 16 where the tips of the green class are nearby the tail of the blue class for some samples. This phenomenon is particularly noticeable when an attribute-pair consists of zero 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 (Figure 17).

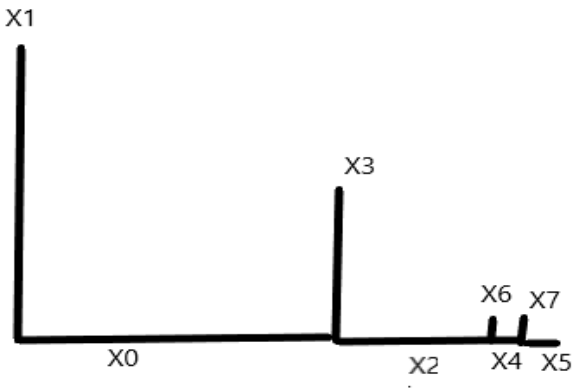
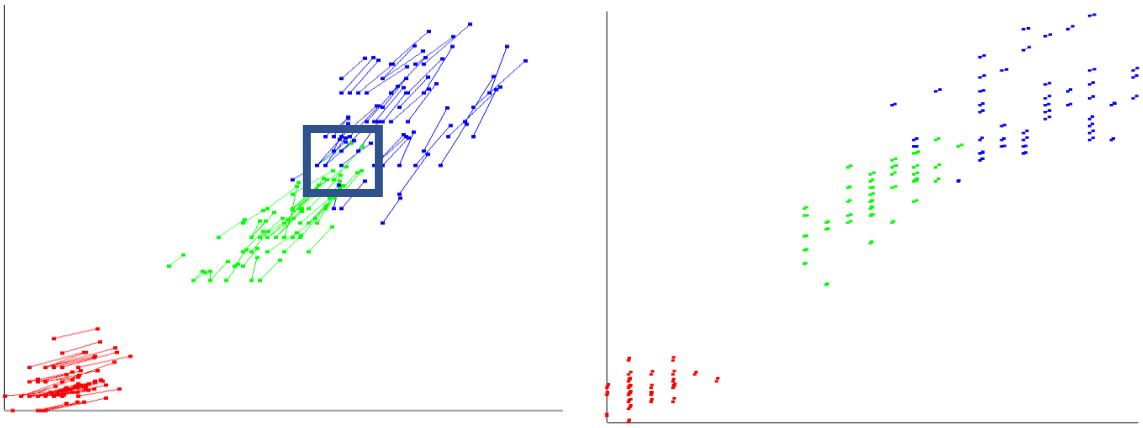


FIGURE 17: Downscaling attribute-pair axes.

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 recommended to carefully select the first couple of attribute-pairs.

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 15b 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 18 it is immediately noticeable which samples to select for a *worst-case* split.



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

FIGURE 18: Emphasizing key attribute-pairs on DSC2.

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 18a. 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 markers rather than polylines.

Figure 19 is used to explain the reason why we tried 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 in classification. 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 trivial 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.

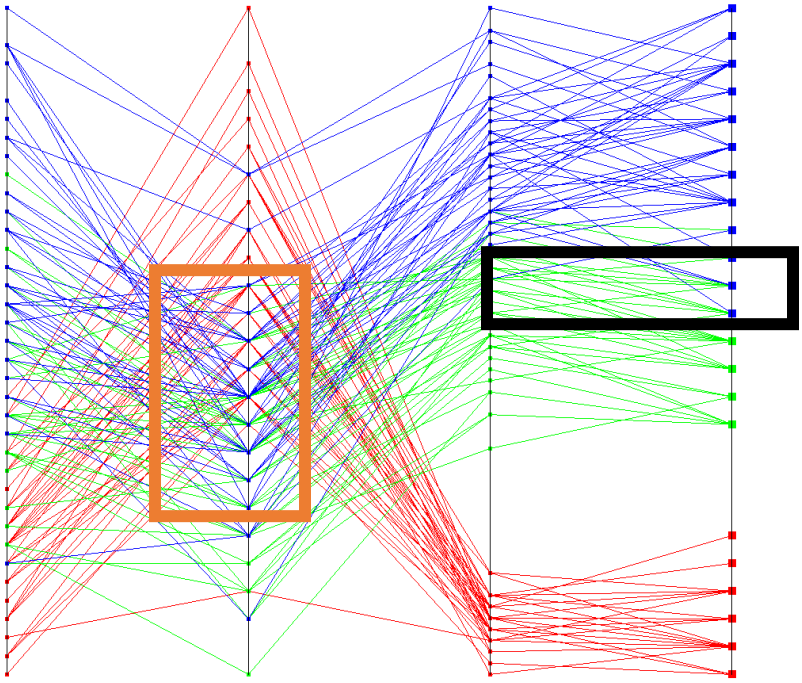


FIGURE 19: Two areas of overlap on the Iris dataset on PC

Another technique known as non-linear scaling [4] can be used to separate Iris data as shown in Figure 20. The decision tree in Figure 17 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.

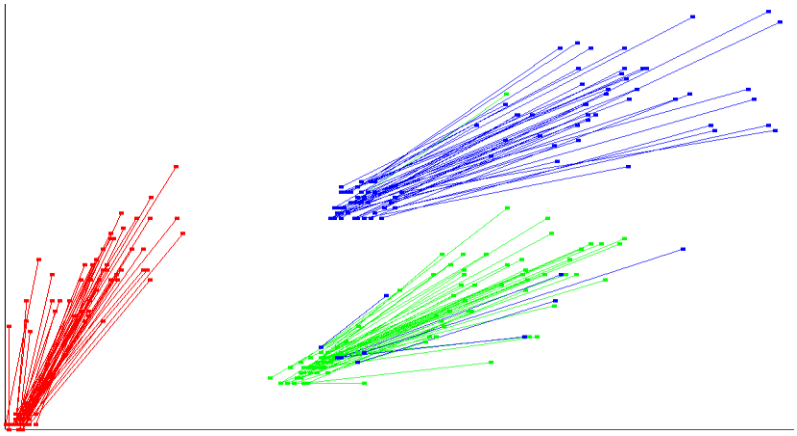


FIGURE 20: Non-linear scaling on certain attributes of the Iris dataset.

Figure 21 demonstrates how non-linear scaling was applied on the first attribute-pair to create Figure 29. 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.

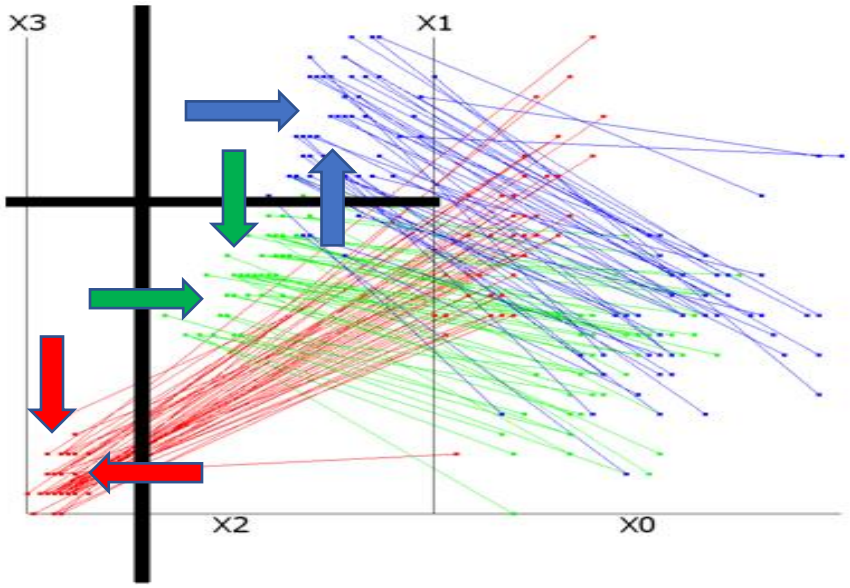
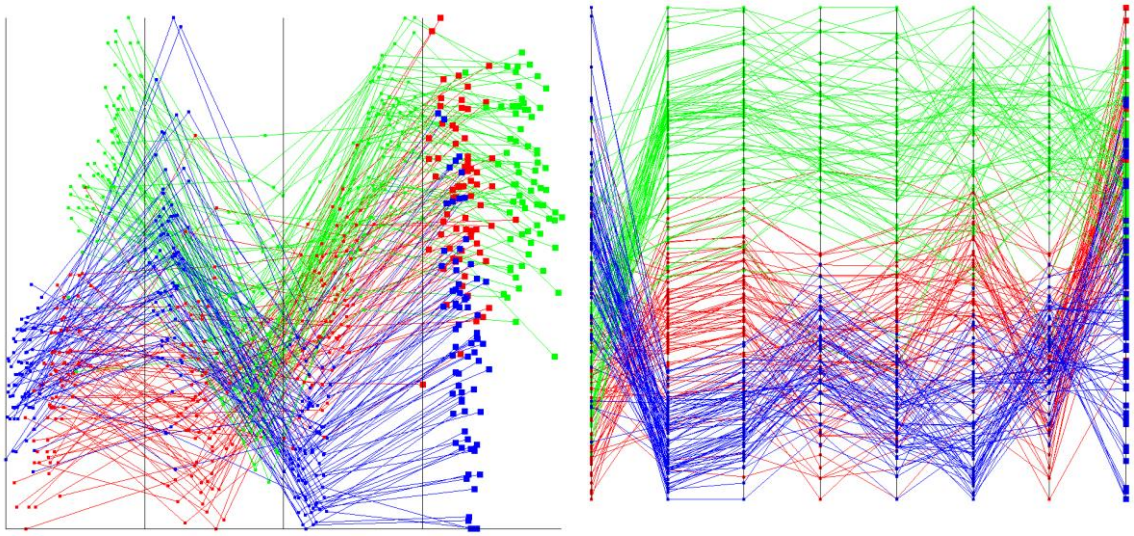


FIGURE 21: Non-linear scaling technique on SPC.

Seeds Dataset And Hyperblocks

The Seeds dataset contains 210 samples of three types of wheat seeds (Kama, Rosa, and Canadian) [15]. 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. The dataset was obtained from the UCI ML repository [8].

Immediate separation of classes is not apparent using a PC or SPC as shown in Figure 22. Note that the Kernel Groove attribute was duplicated to create an even number of attributes for the SPC plot. We decided to use a decision tree analysis (Appendix A.1) to find attributes of interest for DSC2. The attributes of interest happened to be area and kernel groove.



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

FIGURE 22: Seeds dataset on SPC and PC.

From the decision tree analysis in Appendix A.1, we established three hyperblocks, 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. 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. The decision tree model used a Classification and Regression Tree algorithm (CART) with Gini impurity criterion, and greedy approach on the best split.

Using Figure 23 we employed a box clipping algorithm on 21 samples of the Seeds dataset which appeared to be overlapped.

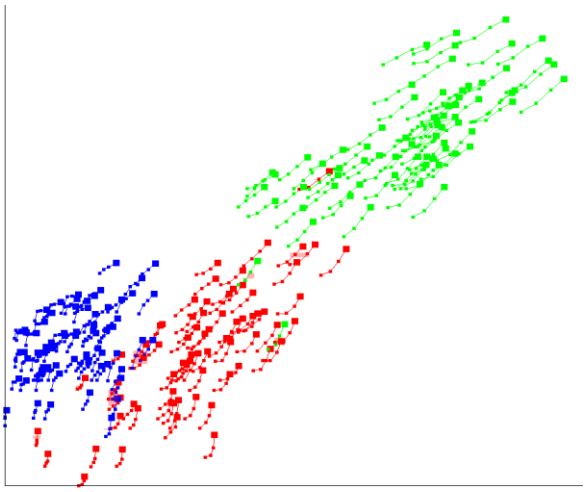


FIGURE 23: Seeds on DSC2 after scaling and HB analysis.

Figure 23 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 24 offers an enhanced view of Figure 23 which shows the samples we clipped into the validation set.

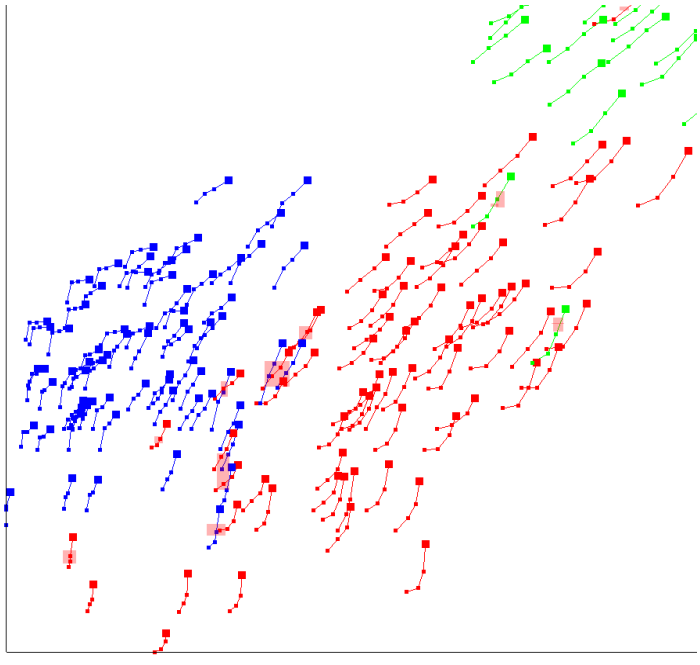


FIGURE 24: Enhanced Version of the Seeds DSC2 Plot.

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 [16]. 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), 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	97.1
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 97.1% 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 all eight models. The best performing model was Support Vector Machine, and the worst performing model was Stochastic Gradient Descent.

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. 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. Using Table 1 we may have chosen SVM as our model, but if we had concern about the worst-case scenario then Random Forest or Decision Tree could be an alternative model.

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

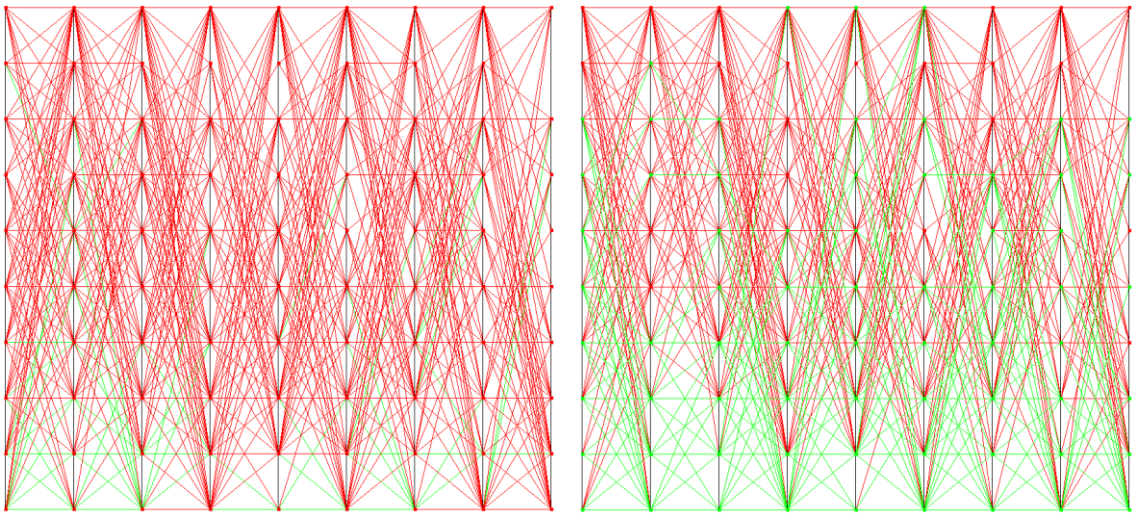
Model	Accuracy
DT	57.14%
SVM	38.10%
RF	61.90%
KNN	23.81%
LR	38.10%
NB	57.14%
SGD	42.86%
MLP	23.81%

Wisconsin Breast Cancer Dataset and Principal Components

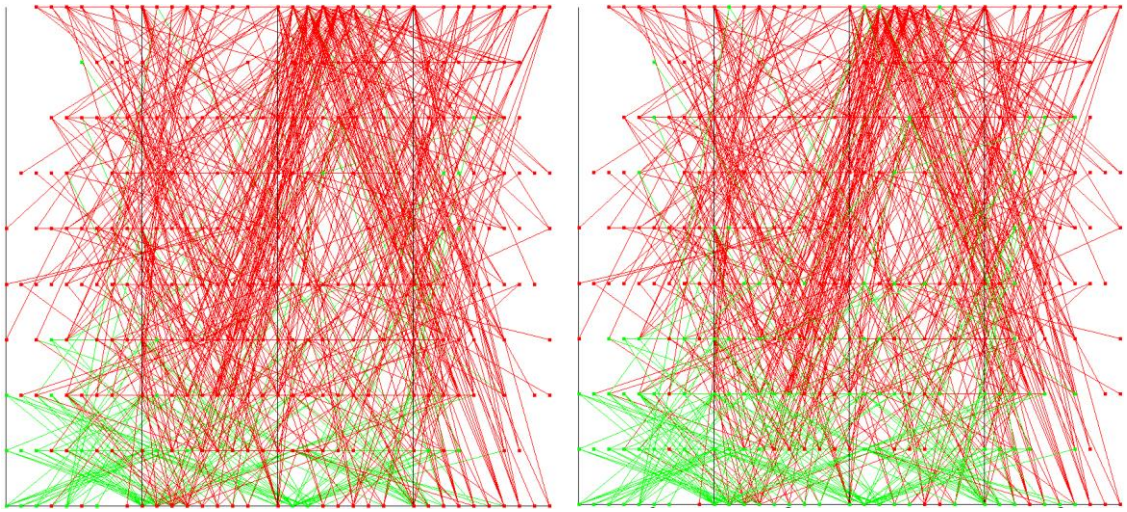
The Wisconsin Breast Cancer dataset contains 699 samples using nine descriptive attributes: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and

mitoses [3]. 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 [11]. The dataset was obtained from the UCI ML repository [8].

Figure 25 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 heavily concentrated towards the bottom of both plots. The benign class has 1.86x more samples than the malignant class (red), but the malignant class consumes more area of the plot.



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

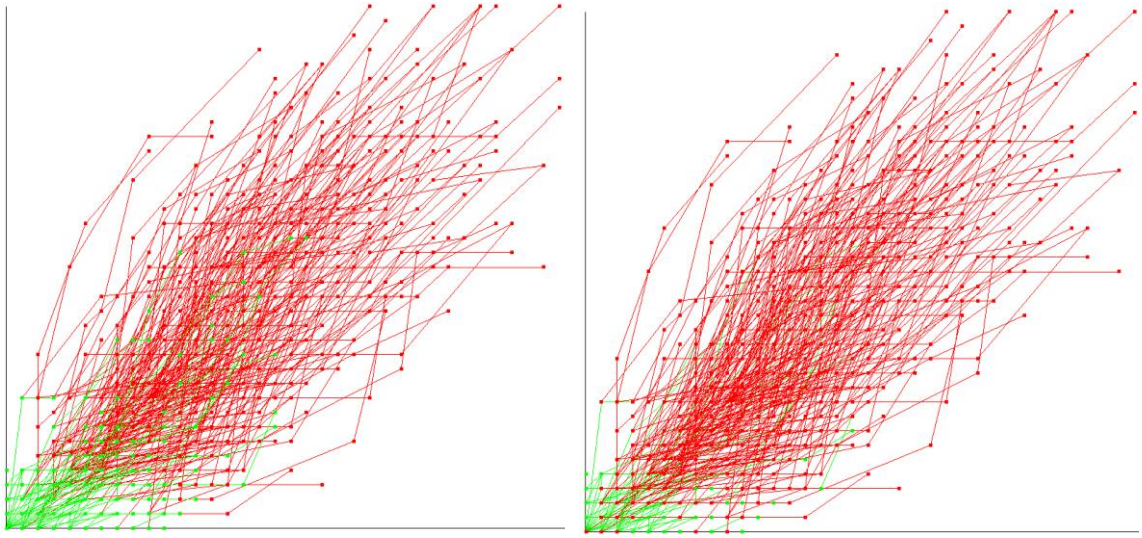


(a) Malignant class on top on SPC (b) Benign class on top on SPC

FIGURE 25: Wisconsin Breast Cancer dataset on PC and SPC.

A decision was made to remove the mitosis attribute for the SPC plot as it requires an even number of attributes. This decision was made from DT analysis (Appendix A) that showed the mitosis feature was not used for complete dataset separation. The decision tree model used a Classification and Regression Tree algorithm (CART) with Gini impurity criterion, and greedy approach on the best split.

Figure 26 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.



(a) Benign class on top on DSC2 (b) Malig. class on top on DSC2

FIGURE 26: DSC2 of the WBC dataset.

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 27). 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 are not ignored in the decision making process. The red squares in Figure 27 represent areas that would clip samples for a validation set.

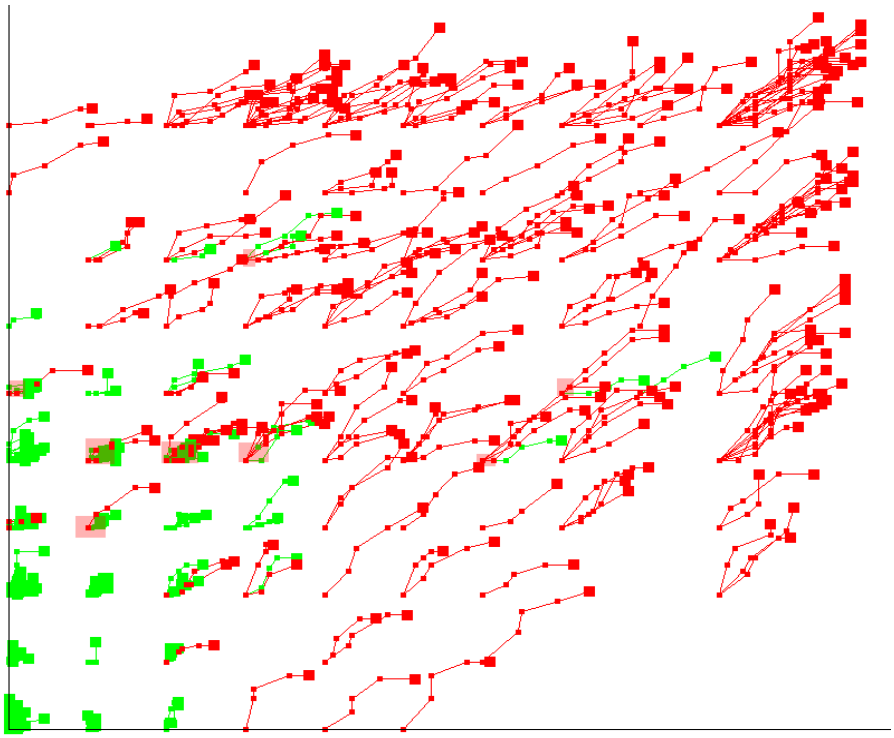


FIGURE 27: WBC dataset on DSC2 after upscaling the 1st attribute-pair.

We decided to further escalate 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 28). Compared to Figure 27 it became easier to identify which samples would result in a *worst-case* split. However, one development that arose from the combination of principal components and real attributes is that we lacked enough samples to fill a validation set. In 10-Fold Cross Validation we would need to select 10% of the dataset, in 3- or 5-Fold Cross Validation even more samples would need to be removed. After removing overlapped samples, we decided to remove samples that were near the boundaries of the two classes.

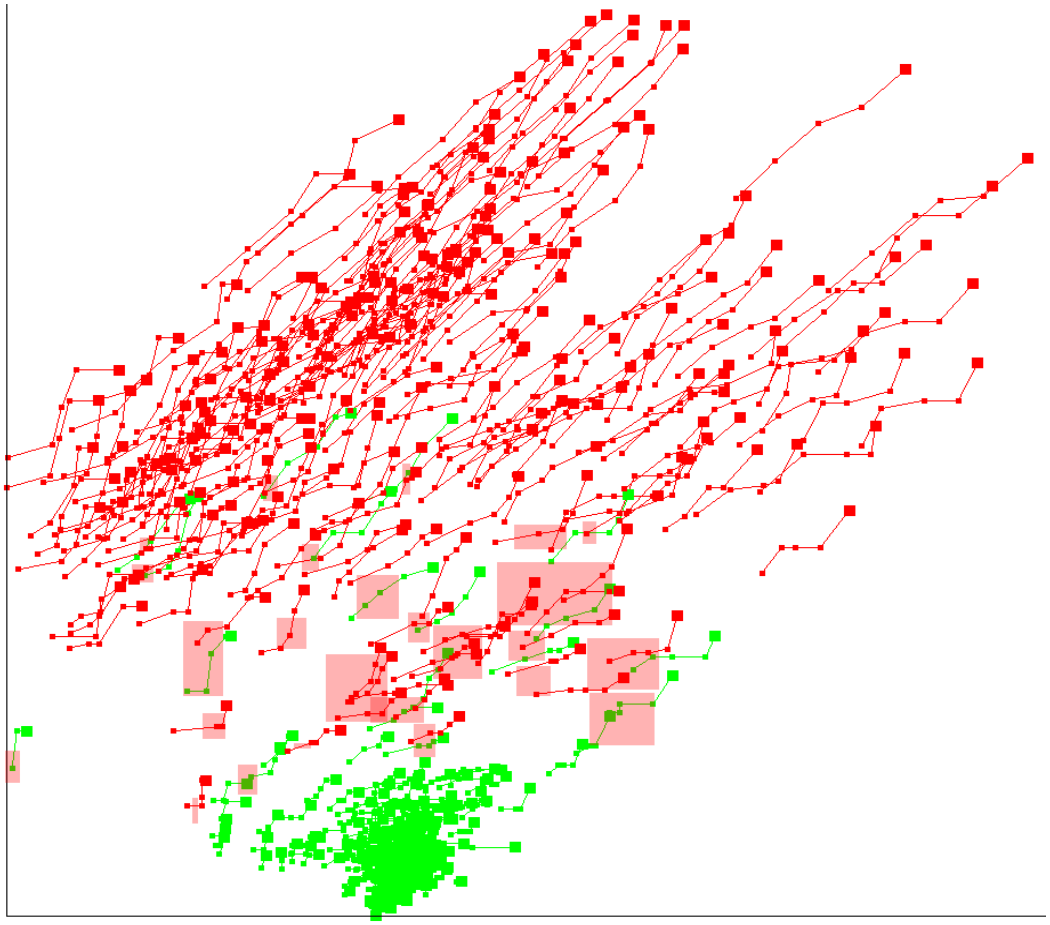


FIGURE 28: WBC dataset with two Principal Components.

One though that occurred to us is that dynamic scaffolding coordinates may not be necessary when looking for worst case splits. That 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 28 that does not show up in Figure 29 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 29 these higher-level insights would not be possible as the points themselves have no differentiation besides location and class color.

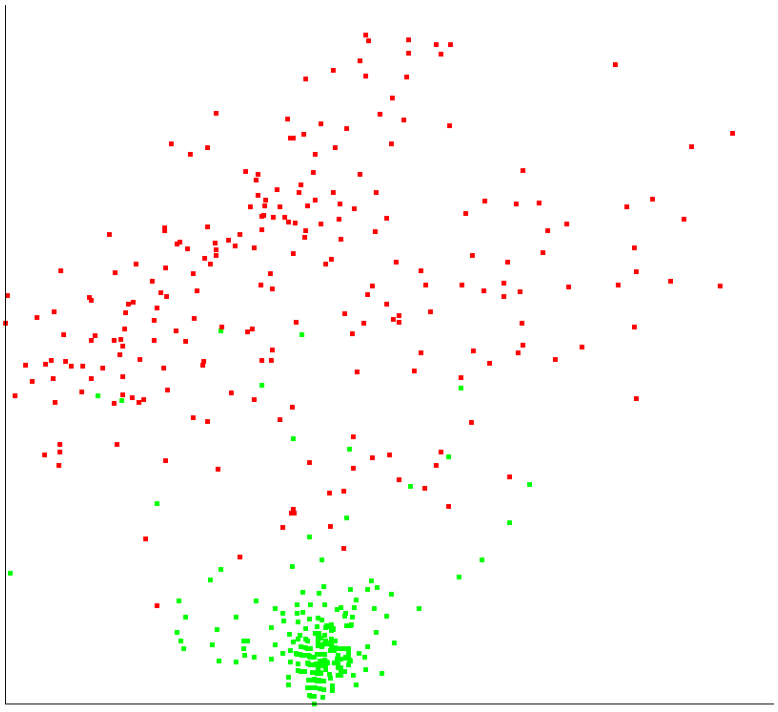


FIGURE 29: WBC Principal Components.

MNIST Handwritten Digits and t-SNE Components

The MNIST Handwritten digits dataset (MNIST) [6] contains 60,000 samples of digits on a 28 x 28-pixel grid in a training set. There is an additional test set containing 10,000 samples. The dataset is available from Kaggle.com [6]. The pixel values are represented in grayscale between 0 and 255. The 28 x 28-pixel grid was converted to a 784-column dataset where each column would represent one attribute. We were unable to produce a SPC or PC plot with the 784 attributes due to the number of axes and plot size required to properly display the attributes without heavy occlusion. We tried to establish decision tree based hyperblocks, but the decision tree accuracy was around 79% and the decision tree was very large, deep branches, and finding attributes of interest in the MNIST dataset was difficult.

Unlike SPC and PC we were able to render a plot that fit inside the application window using DSC2. The plot in Figure 30 gave lack of insight into the dataset as all ten digits were stacked on top of each other and rotating which digit was on top of the stack did not help. The next plan of action was to use a dimensional reduction technique to reduce the number of attributes. In prior work [10] two digits of MNIST were analyzed together using 32 autoencoder features with a high model accuracy. Thus, we decided to reduce the MNIST dataset to 32 autoencoder attributes and plot them on DSC2. Unlike the previous work that compared only two digits, we chose to compare a single digit to all other digits.

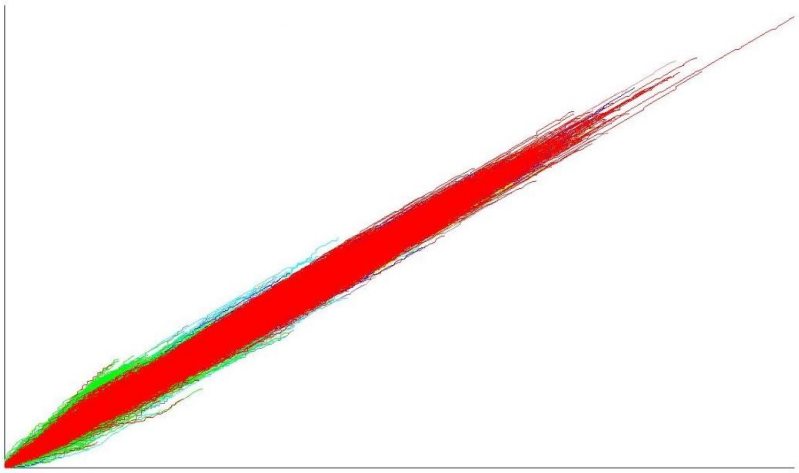


FIGURE 30: 784 Attributes of MNIST on DSC2.

Figure 31 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. From here we decided to use t-SNE components as our attributes of interest, considering that t-SNE has been used to visualize MNIST digits with high clarity.

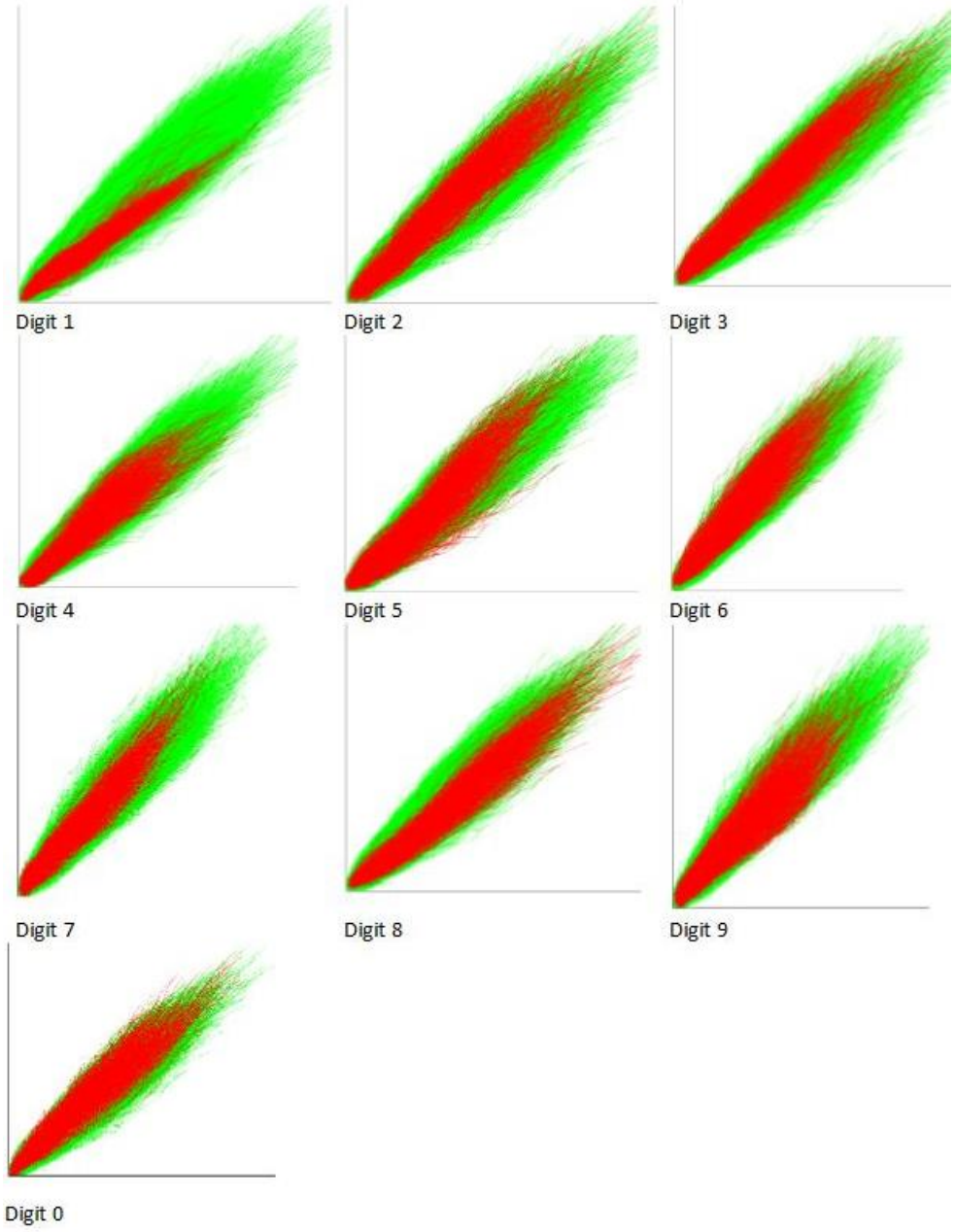


FIGURE 31: Each MNIST Digit using autoencoder features against all other digits.

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 (Figure 32). 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 [17] which leads to t-SNE algorithm being considered as a difficult to interpret or *black box* dimensional reduction technique. Next, t-SNE does not preserve distance between clusters, nor does it preserve cluster density [18]. t-SNE can be simplified to a dimensional reduction technique that does not preserve global structure but preserve local neighborhoods [5]. t-SNE uses a perplexity parameter which [17]. This parameter is related to how many nearby neighbors any point may have.

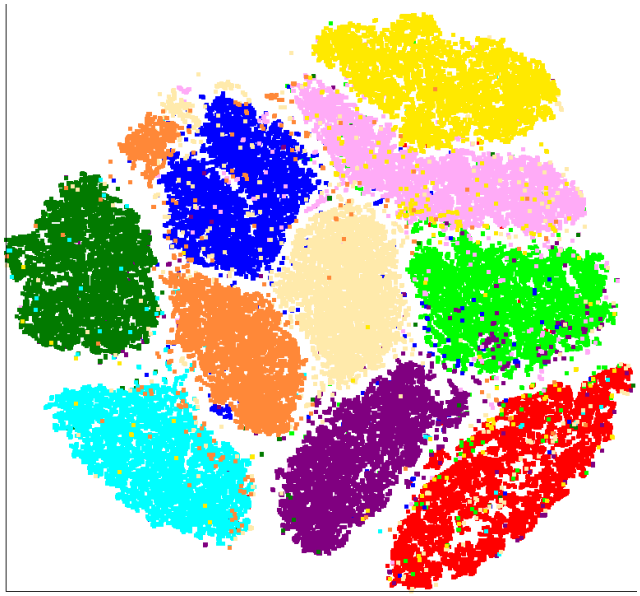


FIGURE 32: t-SNE components for MNIST.

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 32 and Figure 33.

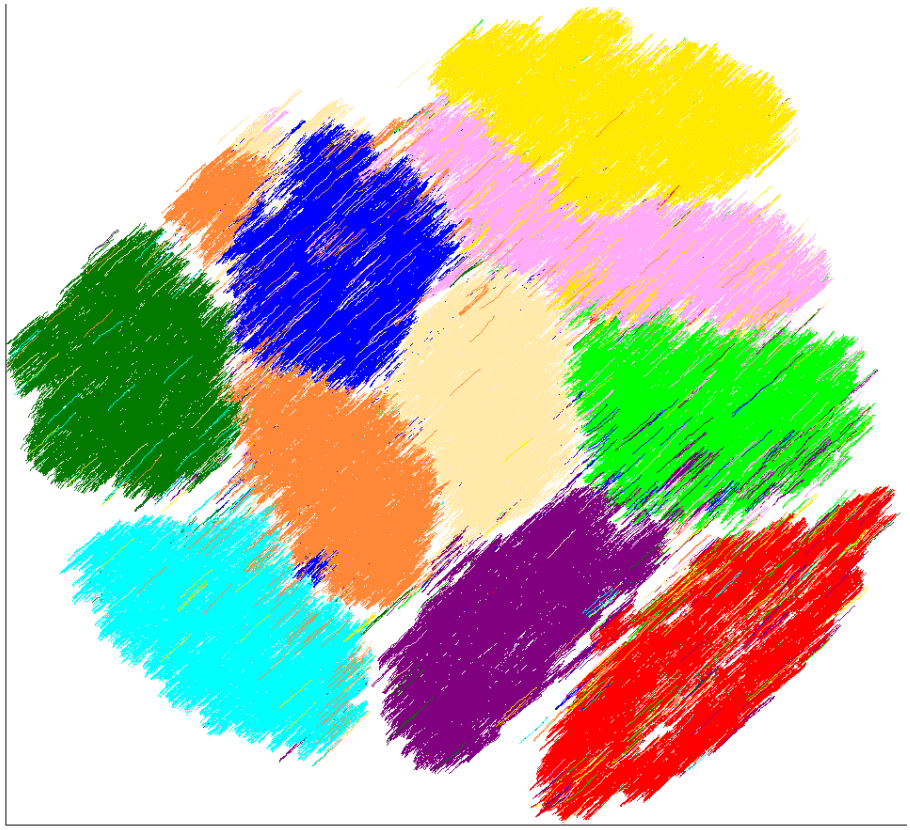
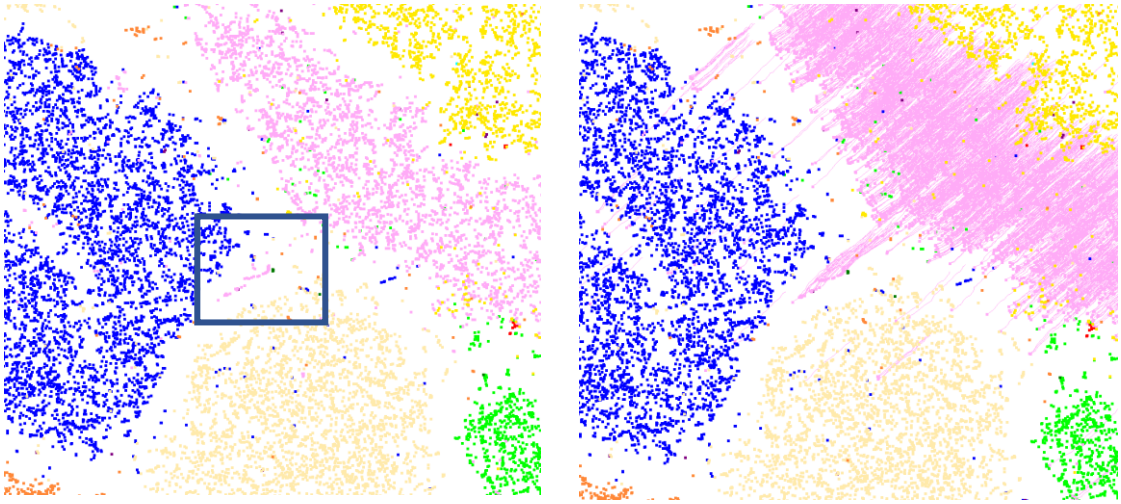


FIGURE 33: t-SNE + MNIST attributes on DSC2.

Figure 34 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 future work we consider other ways at 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.



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

FIGURE 34: t-SNE scatterplot vs DSC2 scaffolds.

CHAPTER IV

EXPERIMENTAL RESULTS AND COMPARISON WITH K-FOLD CROSS VALIDATION ON WBC DATASET

Classification results were obtained from **eight standard ML classifiers** [16] in the sci-kit learn Python library 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 24. 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 we chose 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 shows that standard ML algorithms can classify the WBC dataset within 94 to 97% accuracy without any additional processing, dimensional reduction, or feature engineering. The lowest split was 89.8% and the highest split was 100%. The best performing model was KNN and SVM, and the worst performing model was DT classifier.

TABLE 3: Standard 10-fold Cross Validation model accuracy.

	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 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.

TABLE 4: Upper estimate of the worst split from PCA-DSC2 analysis.

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%

CHAPTER V

CONCLUSIONS

This thesis 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 were 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 were unable to provide adequate attributes of separation, we escalated to various dimensional reduction techniques that allowed for visualizing dimensionally rich datasets like MNIST, with the tradeoff of incorporating lossless attributes.

The results in Table 2 and Table 4 were able to show the lower accuracy of standard ML models for benchmark datasets when looking for difficult dataset splits. Chapter III provides a logical and simple-to-follow 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 dimensional reduction techniques that preserve hyperblock structure or adapting another dimensional reduction technique that

preserves distance such as multidimensional scaling (MDS) or self-organizing maps (SOM).

REFERENCES CITED

[1] 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.

[2] Kovalerchuk B, Ahmad MA, Teredesai A. Survey of explainable machine learning with visual and granular methods beyond quasi-explanations

[3] 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.

[4] Wagle SN, Kovalerchuk B. Interactive visual Self-Service data classification approach to democratize machine learning. In2020 24th International Conference Information Visualisation (IV) 2020 Sep 7 (pp. 280-285). IEEE.

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

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

[7] Kovalerchuk B. Visual knowledge discovery and machine learning. Heidelberg: Springer; 2018 Jan 17.

[8] 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

[9] 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.

[10] Dovhalets D. Data Visualization and Classification of Artificially Created Images. *All Master's Thesis*. 2018. 891. Available at https://digitalcommons.cwu.edu/etd/891

[11] Brown J. Visualizing Multidimensional Data with General Line Coordinates and Pareto Optimization. *All Master's Thesis*. 2017. 898. Available at https://digitalcommons.cwu.edu/etd/898Interpretable artificial intelligence: A perspective of granular computing. 2021:217-67.

[12] Ribeiro MT, Singh S, Guestrin C. " Why should i trust you?" Explaining the predictions of any classifier. InProceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining 2016 Aug 13 (pp. 1135-1144).

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

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

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

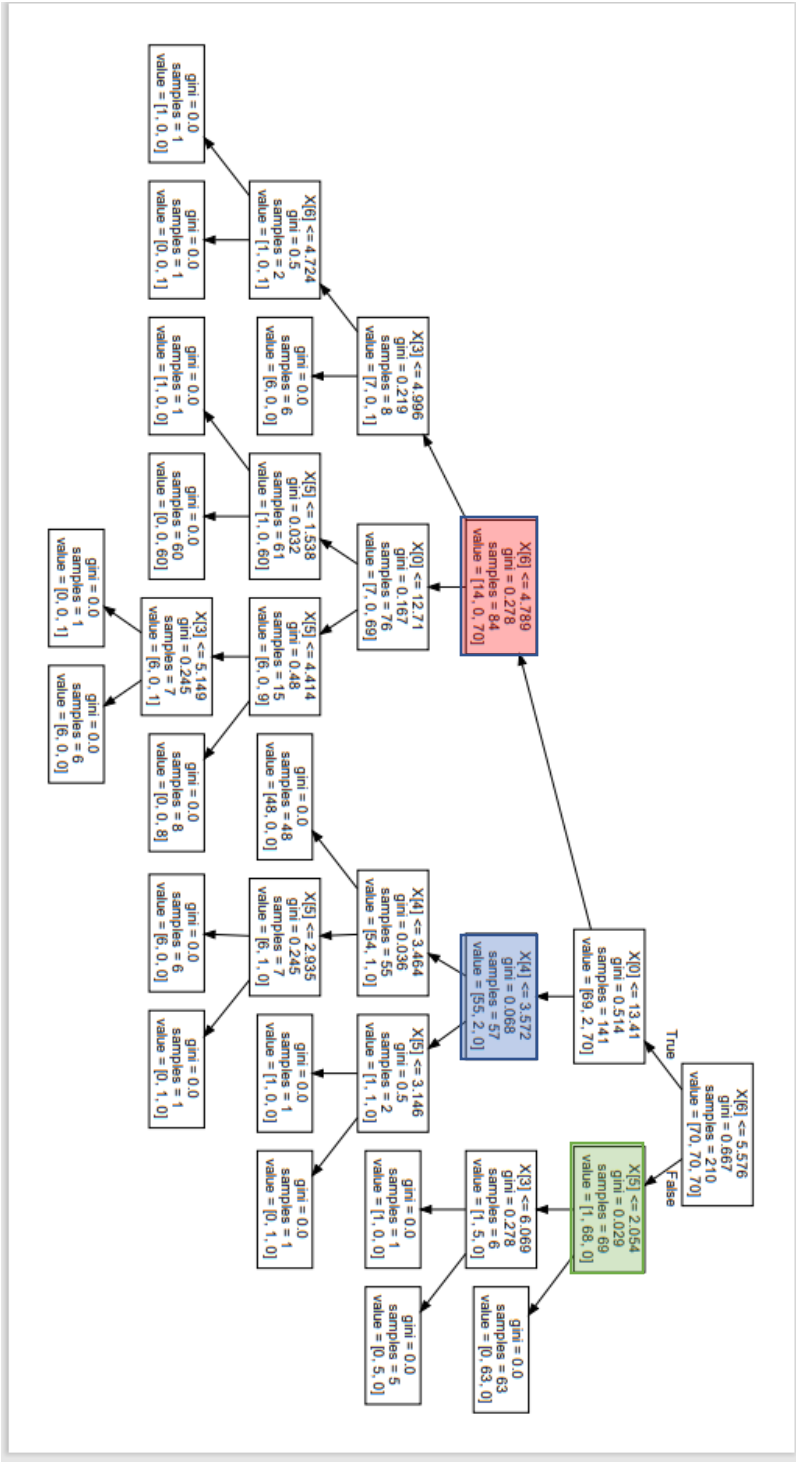
[16] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J. Scikit-learn: Machine learning in Python. the Journal of machine Learning research. 2011 Nov 1;12:2825-30.

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

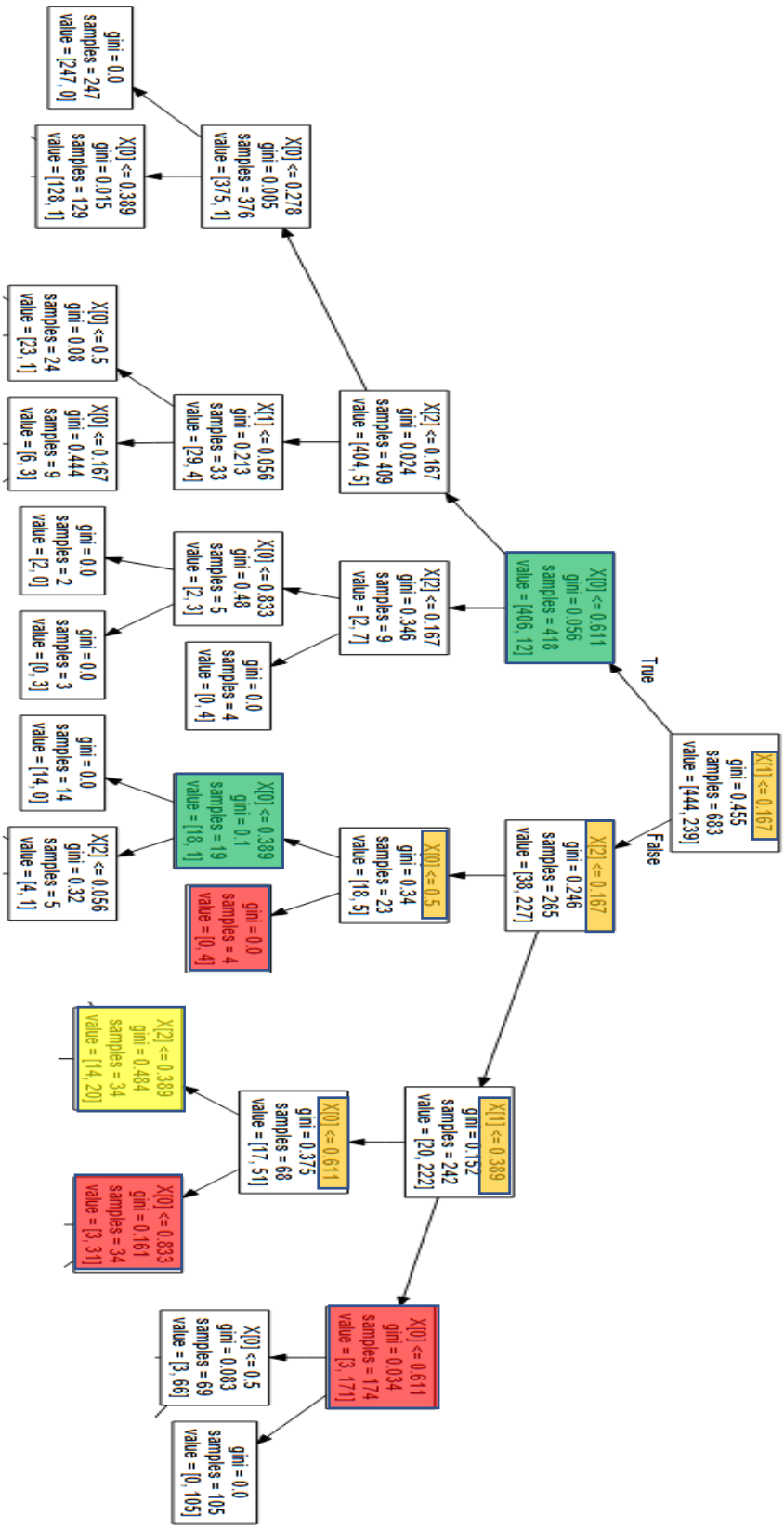
[18] Van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of machine learning research. 2008 Nov 1;9(11).

APPENDIX

Appendix A – Seeds Dataset Decision Tree



Appendix B – Wisconsin Breast Cancer Dataset Decision Tree



Appendix C – DSCViz

DSCViz provides a GUI application to control dynamic scaffolding coordinates, parallel coordinates, and shifted paired coordinate plots (Figure 35). The user can upload a data via the upload button, which brings up the file dialog, however the dataset needs to have some preprocessing done before it can be uploaded into DSCViz. The dataset should have all missing values removed or changed using analytical methods, and dataset attributes should be converted to numeric form rather than a categorical or descriptive string. The class column can remain a string. The class column must be labeled as *class* so that the software knows which column is the labels.

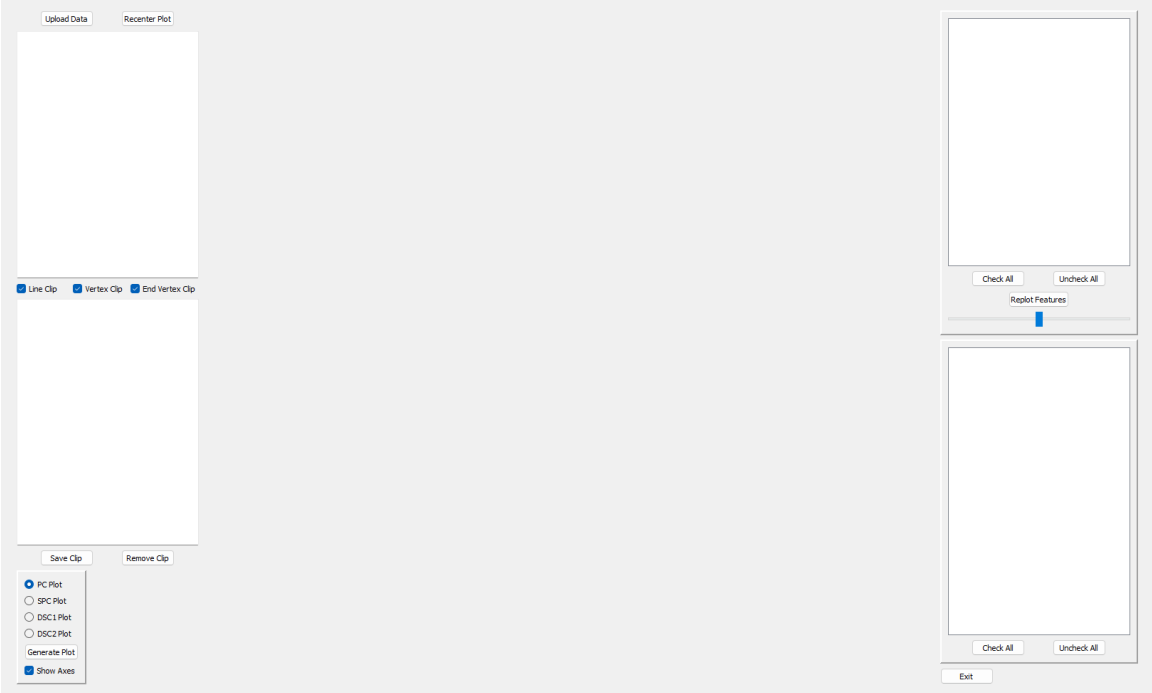


FIGURE 35: DSCViz Application Screen.

After the dataset is uploaded information about the dataset will populate in the top left corner of the window (Figure 36). The user then selects which of the four plots to create using the radio button and clicking generate plot.

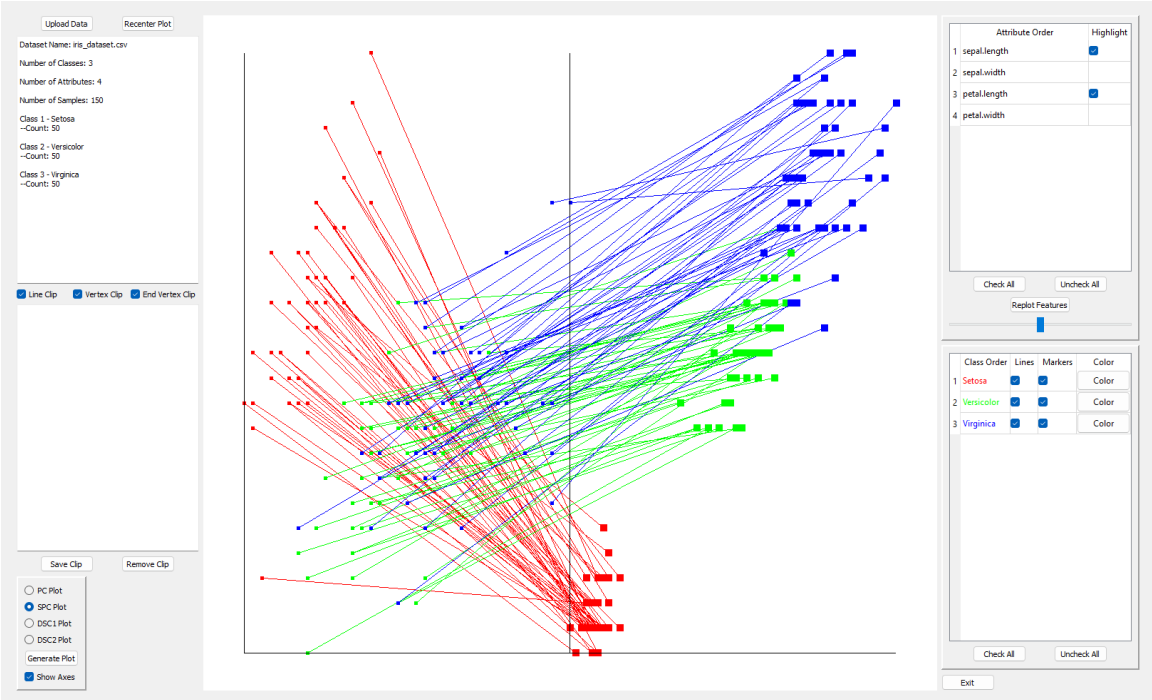


FIGURE 36: DSCViz application running SPC plot.

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 one trick the user may use is 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. Dragging is accomplished by holding down the left click button and moving the mouse. Zooming is accomplished using the mouse scroll wheel.

Attributes in the attribute order table and be dragged and swapped in place. One the user establishes a specific order they click on replot features. This does require running the graph construction algorithm again and can have noticeable lag time on large datasets such as MNIST.

On a similar note, classes can also be reordered with the bottom most class showing on top. 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 (Figure 37). Depending on the plot attribute-pairs may sure a toggle, or attributes may have their own toggle. There is a general slider that controls the transparency of unselected attributes. The slider set at 0 will completely hide the attributes.

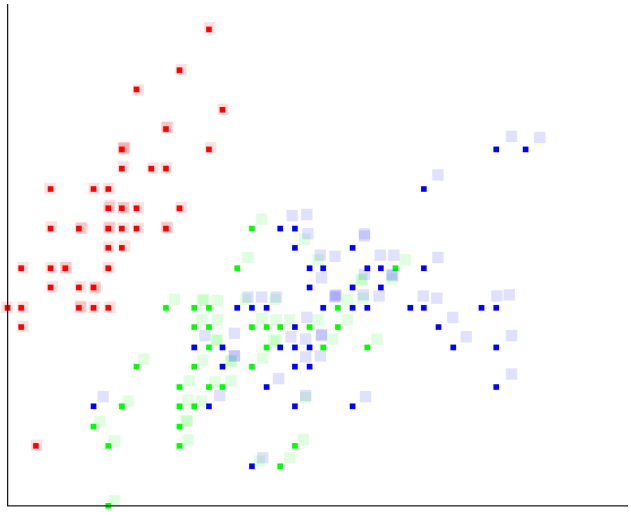


FIGURE 37: Lowering attribute transparency.

DSCViz gives the user the ability to clip samples from the dataset (Figure 38). However, these samples do not disappear off the screen when clipped, and hidden samples will still be clipped if they are in the clipping area. 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. Clicking on Save Clip will save the current clip to a test.csv and train.csv file. These file names **must be changed** when the user is done clipping otherwise, they will be written over when making new clips. Sequential clips can be added and saved any time, but DSCViz does not grant the user the capability to move backwards. The user may remove and reset the clip entirely. All clipping does is remove training samples into the validation set.

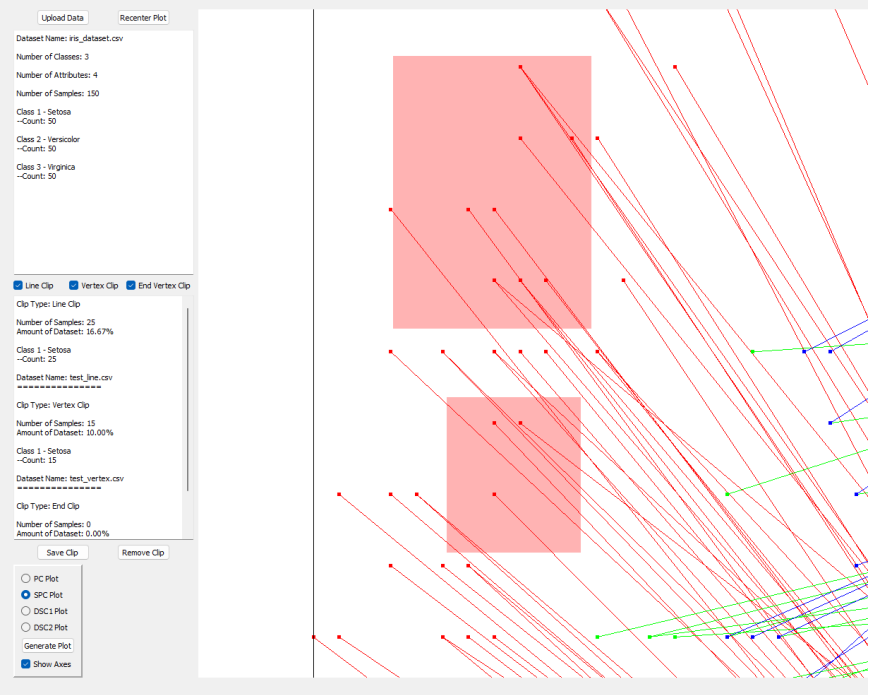


FIGURE 38: Two clipping areas on DSCViz.