Autonomous Visualization

Khalid El-Arini **Andrew W. Moore** **Ting Liu**

June 2006
CMU-CS-06-137

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This is an expanded version of:

Khalid El-Arini, Andrew W. Moore and Ting Liu. Autonomous Visualization. In *Proc. European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2006)*, September 2006, Berlin, Germany.

Keywords: visualization, data mining, classification, interpretability

Abstract

Many classification algorithms suffer from a lack of human interpretability. Using such classifiers to solve real world problems often requires blind faith in the given model. In this paper we present a novel approach to classification that takes into account interpretability and visualization of the results. We attempt to efficiently discover the most relevant snapshot of the data, in the form of a two-dimensional scatter plot with easily understandable axes. We then use this plot as the basis for a classification algorithm. Furthermore, we investigate the trade-off between classification accuracy and interpretability by comparing the performance of our classifier on real data with that of several traditional classifiers. Upon evaluating our algorithm on a wide range of canonical data sets we find that, in most cases, it is possible to obtain additional interpretability with little or no loss in classification accuracy.

1 Introduction

Many modern day problems involve copious amounts of data, and experts across many fields are interested in efficiently extracting meaning from this raw input. Unfortunately, machine learning techniques remain largely inaccessible to many of these domain experts, due to the often unwieldy results produced by conventional classification algorithms, especially when dealing with massive, high-dimensional data sets. Usually, the only output of such classifiers is a symbolic answer (e.g., "Galaxy" or "Star"), or a probability (e.g., "77% chance of cancer"). These algorithms lack any sort of human-interpretable explanation as to why they make the decisions that they do. For instance, while it is possible to examine the weights to and from the hidden layer in a feed-forward neural network in order to obtain some insight regarding the output decision, this information is hard to combine into a coherent explanation.

In this paper we present a classification algorithm that takes into account human interpretability of the results. We attempt to find the most relevant snapshot of a data set, in the form of a two-dimensional scatter plot with easily understandable axes. The result of this search procedure is a transformation of our original attributes into two new features, which not only results in a potentially useful visualization of the data, but can be used as the basis for a simple classifier. Using this new classifier, we can begin to investigate the trade-off between classification accuracy and interpretability. We call this entire process *Autonomous Visualization* (AV).

1.1 Related Work.

Previously, there has been much work on the visualization of large data sets, which usually involves projecting several dimensions onto a two-dimensional plot that is easy for humans to comprehend. de Oliveira and Levkowitz provide a recent survey of the field [5].

Many have tackled the problem of adding interpretability to clustering. For instance, the H-BLOB clustering algorithm displays a hierarchical clustering of data using implicit surfaces, including an interactive feature allowing users to specify the visible graph cuts [16]. Pelleg and Moore trade off expressiveness of their clustering model for aesthetics by learning mixtures of rectangles rather than mixtures of Gaussians [15]. Agrawal et al. design their clustering algorithm to generate output in the form of a DNF expression, for ease of interpretability [1]. Our work differs from these in that we are dealing with the problem of classification, rather than clustering.

There has been significant work—both early and recent—on data-driven scientific discovery. For instance, Langley's BACON system is able to interpret scientific data sets and automatically discover a variety of numerical laws, including the ideal gas law and Kepler's third law of planetary motion [12]. The ABACUS system is similar, but performs its task more efficiently [7]. More recently, Galkin et al. attempt to flag relevant plasmagrams received from the Radio Plasma Imager (RPI) aboard the IMAGE spacecraft, since the high volume of data makes it impossible for NASA scientists to manually examine each one individually [9]. We go further by not limiting our techniques to any specific application area.

Goldberger et al. describe a dimensionality reduction technique that can be used for producing visualizations of high-dimensional data, in the process of learning a Mahalanobis distance measure to be used in nearest neighbor learning [10]. While their approach of finding an optimal

transformation of the data to a lower-dimensional space before running nearest neighbor is similar in nature to what we are attempting, one key difference is that the scatter plots they produce do not have easily interpretable axes. This is also true for other techniques that produce features that are linear combinations of the inputs, such as principal components analysis and projection pursuit [8], [14].

Weiss et al. propose a heuristic search over production rules in order to find the best single fixed-length rule for classification [17]. They use a rule-based system because they believe that both rules and decision trees are more interpretable to humans than other classification techniques. Holte shows that very simple classification rules perform remarkably well on canonical machine learning data sets [11]. Caruana et al. laud the interpretability of case-based methods, such as k-nearest neighbor, since these methods tend to provide related exemplars from the data set that can help explain a classification result [4]. Our goal in this work has a similar spirit, in that we wish to produce classifiers that are easily interpretable to the domain experts that would use them. We differ in that we do not rely on the format of a rule set or nearest neighbor to make our classifier interpretable, but rather on direct visualization.

2 Methodology

Our approach in designing a visualizable, interpretable classification algorithm assumes that the data consists of real-valued input attributes and a single, discrete output. At a high level, our technique consists of three steps:

1. We search over two-dimensional scatter plots of the data, and select the most relevant plot (where *relevance* in this context is defined below).

2. Given the most relevant plot, we transform the data into the two dimensions defined by its axes, and then train a simple classifier in this transformed space

3. We classify future data points by transforming them into this two-dimensional space, and applying the classifier trained in the previous step

2.1 Scatter Plots

In order to find the most relevant snapshot of the data, we search through the space of possible two-dimensional scatter plots. To ensure that the features plotted on the two axes are understandable to humans, as well as to make our search tractable, we place limits on the types of scatter plots we consider.

First, each of the two axes in a scatter plot represents an arithmetic expression of only one or two input attributes. Expressions of more than two attributes begin losing their ease of interpretability. Furthermore, we limit the possible arithmetic expressions to ones that contain commonly understood operators. These include addition, subtraction, multiplication, division, logarithms, exponentials, square roots, and combinations thereof. Finally, rather than plugging in the attributes directly as operands, we first scale and translate them appropriately, in order to produce

a more compelling visualization. For instance, the sum of two attributes is a more informative expression if both attributes range over the same interval, so we transform each attribute to be between zero and one before computing the sum. However, for the sake of visual clarity, we omit displaying these transformation constants in the axes labels of the plots found in this paper.

For example, Figure 2 shows an AV scatter plot for a breast cancer data set.[1] Note that the axes consist of simple expressions (e.g., log[(cell size uniformity) + (bare nucleus)]), that might be easily interpretable by a domain expert, and that the two classes are well separated. For the remainder of this paper, we refer to the pair of expressions that define a scatter plot as a *pairexp*.

2.2 Relevance

The single most important aspect of the Autonomous Visualization process is determining which scatter plots are better than others. This relies on defining a relevance metric that allows us to score pairs of expressions on how likely they are to produce interesting visualizations of the data. One characteristic of visually appealing scatter plots is that points from the same class tend to be grouped together. We developed a metric based on this intuition, as well as an efficient algorithm to compute these scores.

We consider the input data to have been generated by a set of two-dimensional Gaussian distributions, one per class. Thus, we transform the data into the two dimensions defined by the current pairexp, and compute for each class k the maximum likelihood Gaussian, with mean μ_k and covariance $\mathbf{\Sigma}_k$ [6]. We then compute the number of misclassifications that would occur if we used the Gaussians to classify the points in our data set. We define the score of the current pairexp to be this *training set error*. Intuitively, a plot that has well separated classes should obtain a lower training set error, and thus a better score, than a plot whose classes are not well separated. We use training set error rather than validation set error on a held out set because it is precisely the points in the training set that we wish to visualize.

Our score can now be written as $\sum_i I(c_i \neq \hat{c}_i)$, where $I(\cdot)$ is the indicator function, c_i is the correct output class for point $\mathbf{x_i}$ and $\hat{c}_i$ is the class predicted by the Gaussian Bayes classifier:

$$\hat{c}_i = \text{argmax}_k P(c_i = k|\mathbf{x_i}) = \text{argmax}_k P(\mathbf{x_i}|c_i = k)P(c_i = k) \quad (1)$$

where $P(\mathbf{x_i}|c_i = k)$ is the Gaussian density function for class k, and $P(c_i = k)$ is the prior probability of class k, which we compute empirically using the sample proportion.

2.3 Classification

Once we find the best pair of expressions, we have two human interpretable axes on which to plot our data. However, we also now have transformed the original m-dimensional input space to a two-dimensional space that can be used for Gaussian Bayes classification. We classify new data points using the maximum likelihood Gaussians learned for this new transformed space. Specifically, we compute the predicted class $\hat{c}_i$ for each point to be classified, as described previously.

[1] Please note that all scatter plots in this paper are best viewed in color.

Notice that the goal of this work is not to find a transformation of the data that preserves distances (a problem addressed by, e.g., Multidimensional Scaling [3]), but rather to find one that facilitates classification. For example, suppose that given three input variables x, y and z, there might be a very simple linear classification boundary when $1/x$ is plotted against y. This would make for a desirable plot, even though proximity in the plot would not be strongly related to the original space. This is due to, first, the nonlinear transform of x, and second, the intentional exclusion of the irrelevant variable z.

3 Acceleration

3.1 Accelerating Score Computation

Naïvely, we can compute the relevance score for a given pairexp by iterating through every point in the data set and checking its Gaussian Bayes classification against the true class label. This approach clearly takes linear time in the number of data points, per pairexp. However, by exploiting the spatial structure of our data, we developed a more efficient algorithm for computing the score using kd-trees.

A kd-tree is a binary tree that stores a finite number of k-dimensional points at its nodes [2]. Each node in the tree represents a set of data points, bounded by a k-dimensional hyperrectangle. The root represents all the points in the data set. The widest dimension of the root's bounding box is split, thus assigning nodes to each of its two children. This occurs all the way down the tree until some small threshold is reached (e.g., 20 points per leaf), and the splitting terminates.

Thus, the first step in efficiently computing the score for a given pairexp of up to four attributes (i.e., up to two per expression) is constructing a kd-tree with these attributes as its dimensions. For each node, in addition to storing its associated points, we store the number of points that belong to each output class. Note that although kd-tree construction is an $O(n \log n)$ operation, since we are constructing the kd-tree with the raw attribute values (i.e., no expression-specific information) we only need to construct one tree for all possible pairexps of these four attributes. With our current set of legal arithmetic expressions, this amounts to one tree constructed per nearly 3,000 pairexps.

We traverse the tree linearly, and at a given kd-tree node, we attempt to calculate the number of misclassified points belonging to that node without actually iterating through all of them. Given the bounding hyperrectangle for the node, we check to see if any of the class Gaussians are dominant. In other words, for every geometric location $\mathbf{x}$ in the bounding box, does there exist a class k such that $P(c_{\mathbf{x}} = k|\mathbf{x}) > P(c_{\mathbf{x}} = l|\mathbf{x})$ for all classes l? If so, we say that class k dominates this node, and we can prune our search. This is because all points of other classes belonging to this node will be misclassified. If no class dominates the node, we sum the results of recursive calls to the node's two children. We only ever iterate through the points of a node if we are at a leaf node that is not dominated by any single class.

Determining whether a class is dominant in a kd-tree node involves a pairwise comparison among all the classes. This operation is quadratic in the number of classes, but since, at least in some domains, data sets seldom have more than two or three classes, this has not been a serious

limitation thus far. Class k dominates class l in node $\mathcal{N}$ if and only if

$$\min_{\mathbf{x}\in\mathcal{N}} \frac{P(c_{\mathbf{x}}=k|\mathbf{x})}{P(c_{\mathbf{x}}=l|\mathbf{x})} = \min_{\mathbf{x}\in\mathcal{N}} \frac{P(\mathbf{x}|c_{\mathbf{x}}=k)P(c_{\mathbf{x}}=k)}{P(\mathbf{x}|c_{\mathbf{x}}=l)P(c_{\mathbf{x}}=l)} > 1. \tag{2}$$

If, for a given class k, we assume that $\mu_k = [\mu_{k1} \quad \mu_{k2}]^T$, $\mathbf{\Sigma}_k = \text{diag}(\sigma_{k1}^2, \sigma_{k2}^2)$, and $\mathbf{x} = [x_1 \quad x_2]^T$, then we can simplify our expression as follows:

$$K_{kl} \exp\left[-\frac{1}{2}\left(\frac{(x_1-\mu_{k1})^2}{\sigma_{k1}^2} - \frac{(x_1-\mu_{l1})^2}{\sigma_{l1}^2}\right) - \frac{1}{2}\left(\frac{(x_2-\mu_{k2})^2}{\sigma_{k2}^2} - \frac{(x_2-\mu_{l2})^2}{\sigma_{l2}^2}\right)\right] \tag{3}$$

where $K_{kl} = \frac{P(k)\sigma_{l1}\sigma_{l2}}{P(l)\sigma_{k1}\sigma_{k2}}$. If the minimum of this expression across the entire area of the node's bounding box is greater than one, then class k dominates class l. Since we are limiting our Gaussians to be diagonal, it means that we can minimize with respect to x_1 and x_2 separately, and use these values to solve for the minimum of the entire expression.

Finally, it is important to note that due to the monotonicity of our scoring metric, we can perform *early termination* pruning, which is a significant source of computational gain. As we traverse the kd-tree, we can terminate immediately after computing a misclassification score that is greater than the best one so far, even if we have only looked at a tiny portion of the tree. This allows this algorithm to run much faster than the naïve one in practice.

3.2 Accelerating Pairexp Search

Finding the optimal scatter plot that represents the best scoring pairexp involves a hefty combinatorial search. The search can be divided into a two-stage hierarchy. At an outer level, we iterate over tuples of attributes. In an inner loop, given a tuple of attributes, we iterate over possible pairexps that can be generated from those attributes. If both of these stages are exhaustive—i.e., we iterate over all possible pairs, triples, and quadruples of the input attributes, and for each tuple, we exhaustively generate all possible pairexps—while we will indeed find the optimal scatter plot under our scoring metric, it is computationally infeasible for high dimensional data sets, since we would have to consider $\mathrm{O}(m^4)$ tuples, where m is the number of dimensions.

Rather than exhaustively considering all possible 2-,3-, and 4-tuples of the input attributes, we instead perform a greedy search, followed by one step of hill-climbing:

1. We exhaustively consider all pairs of attributes, remembering which pair produced the best scoring pairexp (e.g., if the best scoring two attribute pairexp was [x, $\log y$], then we remember (x, y) as the best pair).

2. Given the best pair, we consider all triples that contain the best pair.

3. Given the best triple, we consider all quadruples that contain the best triple.

4. Starting from the best quadruple, we consider all other quadruples that can be obtained by changing each of the four attributes, one at a time.

Note that the greediness only applies to the outer stage; our inner loop search over pairexps of a given tuple is exhaustive. This heuristic search now only has a quadratic dependency on m, since steps 2-4 are now linear. Furthermore, this quadratic dependency is ameliorated by the fact that, in contrast to quadruples, the number of pairexps that can be generated from a pair of attributes is quite limited.

4 Experimental Results

We evaluated both the naïve and kd-tree based implementations on twelve different data sets. Eleven come from the UC Irvine repository, while EDSGC is a subset of the Edinburgh/Durham Southern Galaxy Catalogue.

4.1 Interpretability

We produced a scatter plot for each of the twelve data sets. A representative sample can be found in Figures 2 to 13. We see that our algorithm succeeds in producing some separation of the classes. Furthermore, the axes represent simple expressions involving attributes that are well known to the domain experts that would be interested in analyzing the data. This stands in contrast to the alternative of plotting the data as projected onto its top two principal components, since in this case, the axes are no longer directly interpretable.

In order to verify that our algorithm was producing interesting and interpretable visualizations of the data, we consulted a domain expert in astronomy, who provided us with real data from the Sloan Digital Sky Survey (SDSS). Given ten real input attributes, and a 50,000 record subset of the data corresponding to a region of the sky deemed interesting by the astronomer, AV produced the star/galaxy plot seen in Figure 14. The domain expert confirmed that the plot was precisely the kind of plot he would expect from this data. This anecdotal evidence supports our claim that AV indeed produces useful visualizations.

Furthermore, in order to determine whether the complexity of our binary arithmetic expressions was warranted, we ran a version of our algorithm that considered only pairexps with unary expressions on the two axes. When applied to the SDSS data, this version produces the rather uninspiring plot shown in Figure 15. This shows that there indeed are data sets that are best visualized by combining multiple attributes on a single axis.

4.2 Classification Accuracy

We computed five-fold cross validation accuracies after running our AV classifier on all the data sets. For the sake of comparison, we ran nine canonical classifiers on the same data [18]. It should be noted that, for the sake of expediency, we limited all larger data sets to 5,000 randomly selected records.

When we compare the classification accuracies of our classifier with those of the canonical algorithms, we find that the AV classifier is competitive. In Figure 1, we see a pairwise comparison

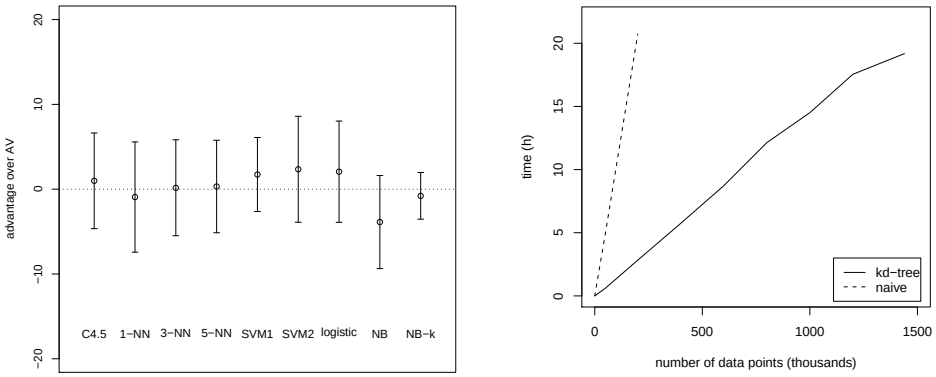

Figure 1: *Left:* A pairwise comparison between AV and canonical classifiers, showing the difference in classification accuracy, averaged over all data sets. *Right:* Timing analysis of AV on expanded EDSGC data set, as number of records is increased

between our classifier and the nine canonical classifiers. On average, the classification accuracy of AV is only 0.168 percentage points lower than the other algorithms. In fact, AV outperforms 1-nearest neighbor and naïve Bayes. On these data sets, the best performing algorithm was a support vector machine with quadratic kernels, and AV is only 2.348 percentage points less accurate on average. In Table 1, we see that AV outperforms the *average* canonical classifier on all data sets except iris, realmpg, sonar and vehicle. However, although close at times, AV is never the *most accurate* classifier on a given data set.

4.3 Efficiency

Table 1 shows timing results for running AV with both the kd-tree and non-kd-tree implementations of the algorithm. We see that by implementing our algorithm using kd-trees, we make noticeable gains in efficiency over the naïve implementation, especially for larger data sets. The adult data set, for instance, sees a 4.12 fold speed-up, and the EDSGC data set sees a speed-up of 8.12. It is important to note, however, that the vehicle data set performs slightly more slowly under the kd-tree algorithm than the naïve one. This is perhaps due to the relatively small size of the data set preventing the algorithm from overcoming the six pairwise class comparisons it must do per kd-tree node, a problem not encountered in the 50,000 record forest data set, despite its seven output classes.

Furthermore, we timed the kd-tree implementation on the EDSGC data set as we varied the number of points from 50,000 to the full 1.4 million record data set. The resulting plot, as shown in Figure 1, is mostly linear, but begins to dip slightly as we approach the size of the full data set. When compared to the plot for the naïve implementation, we see that the overhead of building the kd-trees is a small price to pay for the gain in efficiency.

It is important to note that the timing costs reported here are only incurred during the training phase of the classifier. Once a relevant pairexp has been found during training, future data points

Table 1: Classification accuracy and timing results

			Accuracy (%)			Timing (sec)	
Data set	Records	Attributes	best	average	AV	kd-tree	Naïve
Abalone	4178	8	55.47	53.28	53.84	308	339
Adult	48844	6	83.92	80.87	82.02	502	2069
Breast-w	701	9	97.42	96.15	96.19	22	59
Diabetes	770	8	77.47	74.51	77.34	22	55
EDSGC	50000	22	99.46	95.16	99.16	2221	18037
Forest	50000	10	69.82	67.40	68.12	4856	8445
Heart-statlog	272	13	85.19	80.16	81.48	21	46
Ionosphere	357	34	91.74	87.59	91.17	69	183
Iris	155	4	96.67	95.78	93.33	2.0	7.2
Realmpg	394	7	82.91	72.79	70.66	2.0	6.8
Sonar	210	60	87.02	77.51	74.04	140	227
Vehicle	851	18	80.50	69.35	60.52	279	268

can be classified simply by transforming them to the two-dimensional space and applying the Gaussian Bayes classifier. Alternatively, for a large data set, time can be saved by training only on a subset of the data in order to obtain the best pairexp, but then transforming the entire data set for visualization purposes.

5 Discussion

There is much flexibility inherent to the AV process, which provides ample opportunity for exploration during algorithm design. For instance, before settling on the Gaussian Bayes misclassification score, we implemented several others for comparison. The first attempt was a pairwise count that penalized points of differing classes that were in close proximity to each other. We found that this algorithm generally produced poorer visualizations, and relied on several parameters that we found difficult to set. (Leban et al. propose a similar method, which they use to successfully analyze and visualize gene expression data [13].) In another attempt, we tried maximizing the Gaussian log likelihood of the data, and found that while we produced visualizations that were at least as good as our current results, it was difficult to optimize the algorithm in terms of efficiency, which we believe is the key to making it tractable to search such an intense diversity of scatter plots. We faced a similar situation when we tried a Gaussian misclassification score that allowed full covariance matrices, rather than the diagonal matrices that we use in our current algorithm. In fact, since our relevance score is simply a training set error, conceivably any classifier can be substituted for the Gaussian Bayes classifier that we use, either for computing the score or for classifying new data.

Future work will focus on improving the scalability of our techniques to allow for interactive speeds on very large data sets, as well as extending AV to other forms of data, such as time series.

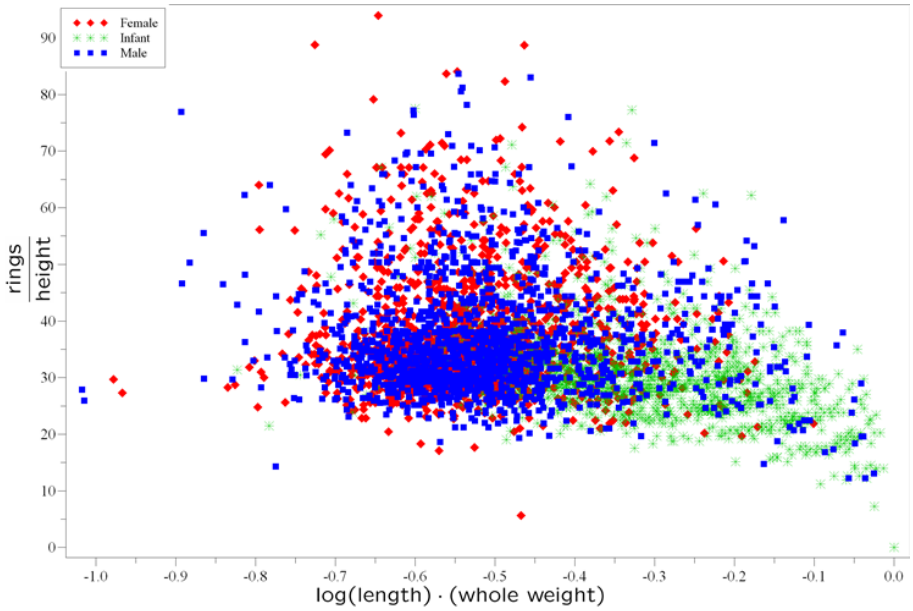

Figure 2: The AV plot for the Abalone data set manages to separate the males and females from the infants, but cannot distinguish between males and females

6 Conclusion

We have described a novel approach to classification that takes into account interpretability of the results. We have detailed an algorithm that produces relevant scatter plots of real-valued data sets, and evaluated it in terms of interpretability, classification accuracy, and efficiency. For most problems we considered, our algorithm was competitive with state-of-the-art classifiers. We provided compelling evidence that it is possible to obtain additional interpretability with little or no loss in classification accuracy. Furthermore, we showed that our algorithm is efficient, making it feasible for use on large, real-world data sets. Finally, our results demonstrate the potential for further investigation into the opportunities and challenges of integrating visualization with classification.

Acknowledgments

We would like to thank Karen Chen and Kevin Killourhy for their help in running some of the experiments, and Dr. Andrew Connolly for his evaluation of the results.

References

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceed-*

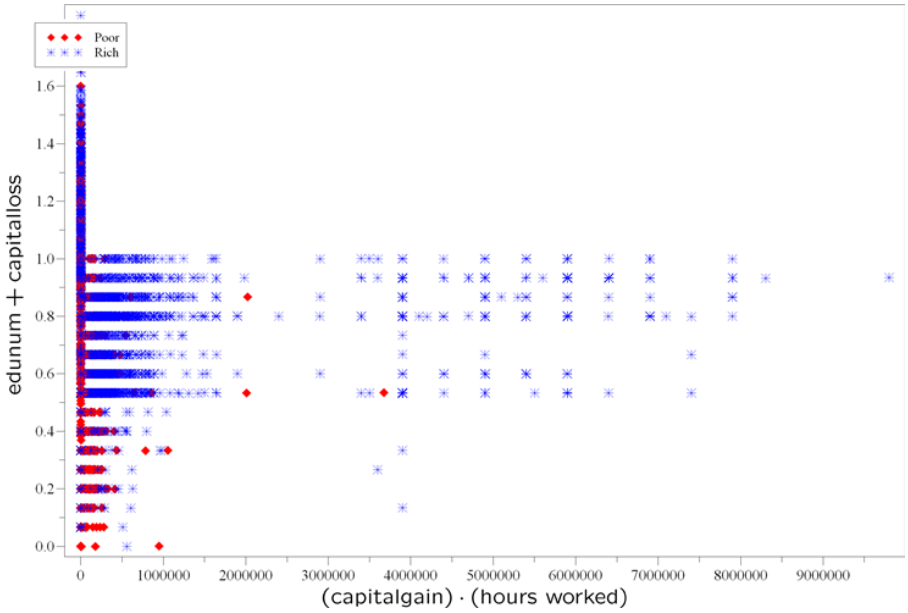

Figure 3: The AV plot for the Adult data set uses a combination of the “capital gain” and “hours worked” attributes to separate the rich from the poor

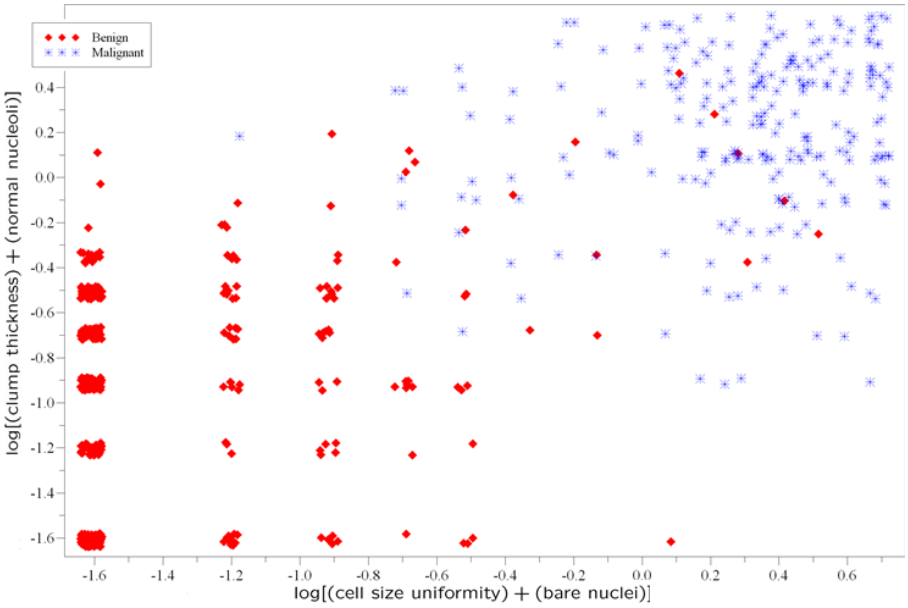

Figure 4: The AV plot for a breast cancer data set (Breast-w) separates the benign and malignant cases along a diagonal decision boundary

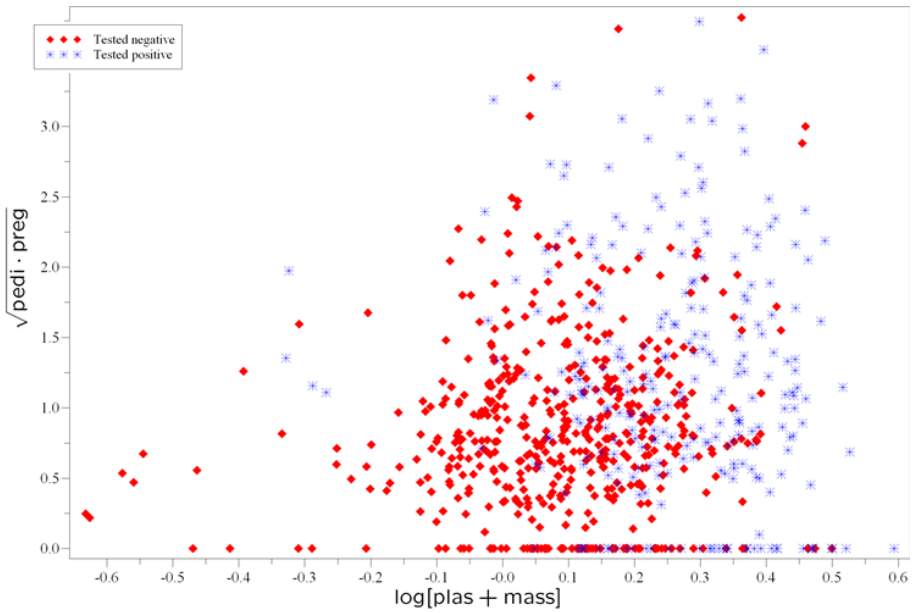

Figure 5: AV separates along the horizontal axis patients who test positive for diabetes from patients who test negative

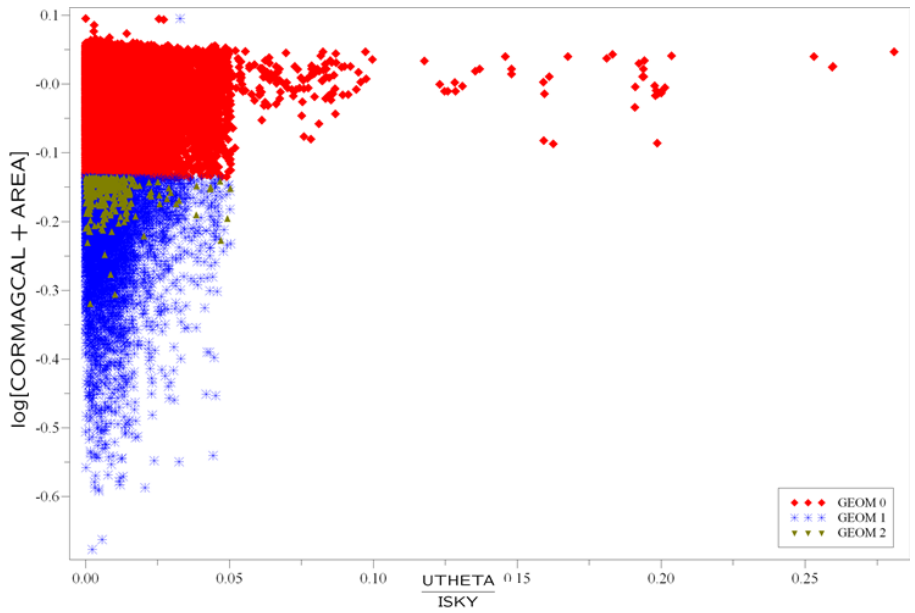

Figure 6: The AV plot for the EDSGC astronomy data set chooses an expression for the vertical axis that separates the data in order to predict the geometric parameter

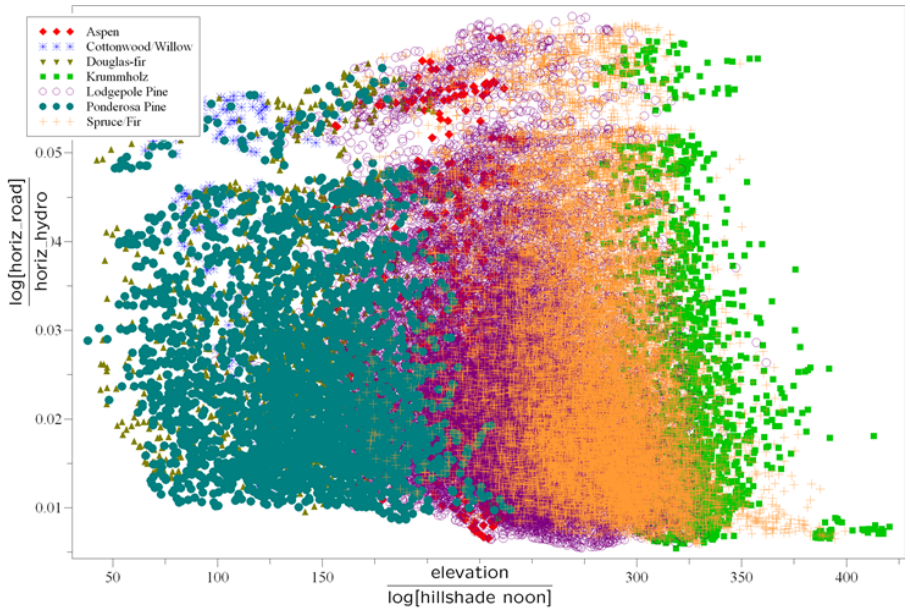

Figure 7: The AV plot for the forest data set separates the various cover types along the horizontal axis

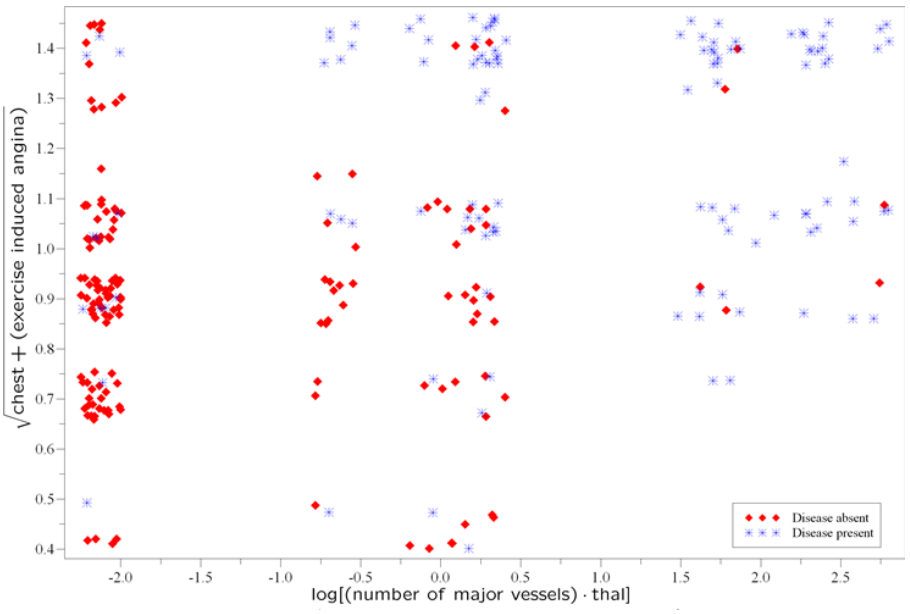

Figure 8: AV produces a diagonal decision boundary for classifying the presence or absence of heart disease in the Heart Statlog data set

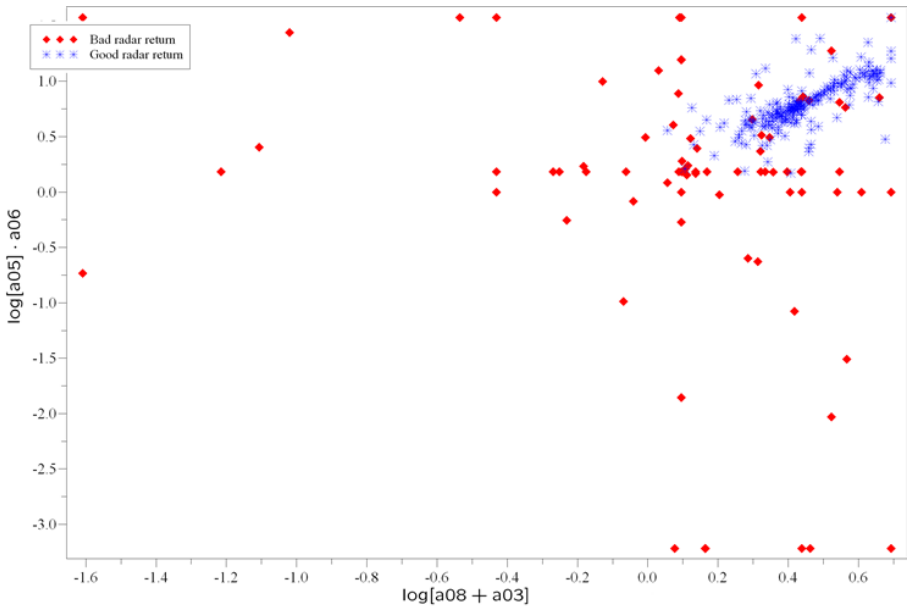

Figure 9: The AV plot on the Ionosphere data set forces good radar returns into the top right corner

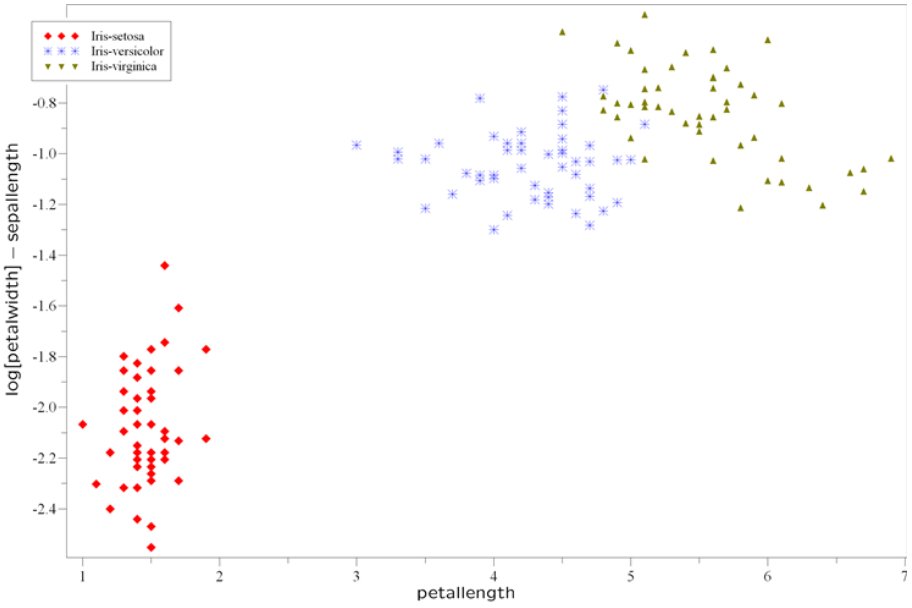

Figure 10: This canonical Iris data set is easily separated using very simple expressions on the axes

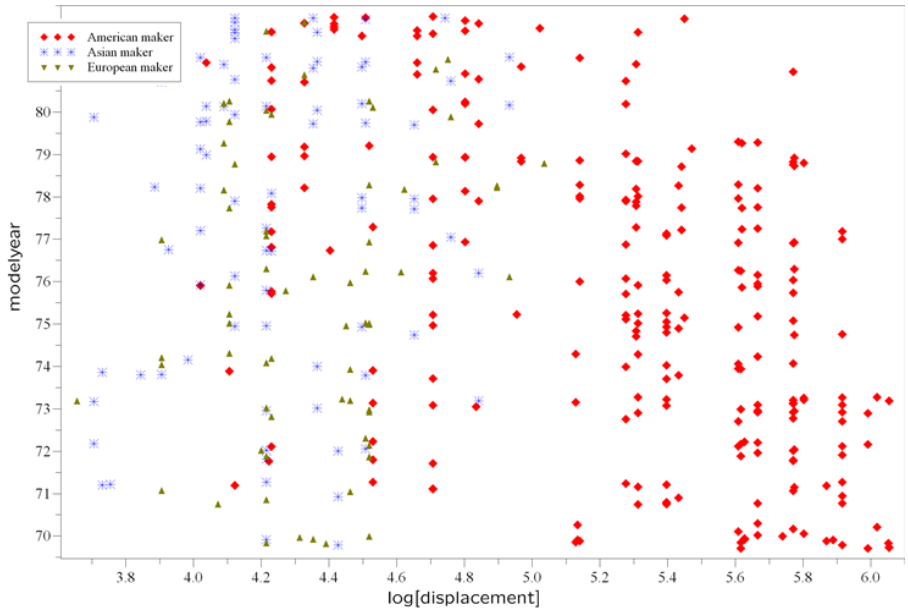

Figure 11: On the Realmpg data set, we see that American auto makers are identified by the high displacement of their automobiles, while it is much harder to distinguish between European and Asian makers

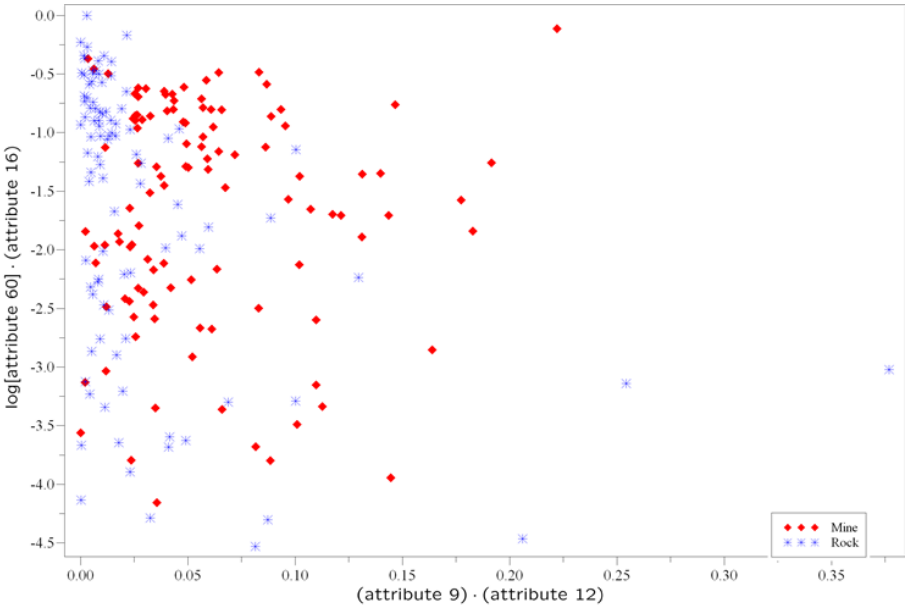

Figure 12: Mine and rock in this Sonar data set are separated by AV along the horizontal axis.

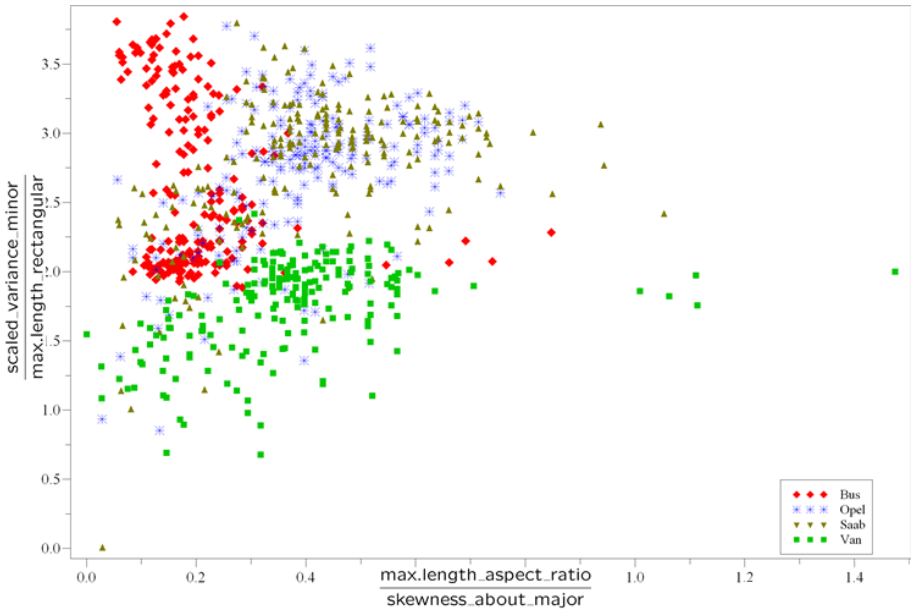

Figure 13: In the Vehicle data set, AV separates busses from vans from passenger cars, but cannot distinguish between Opels and Saabs

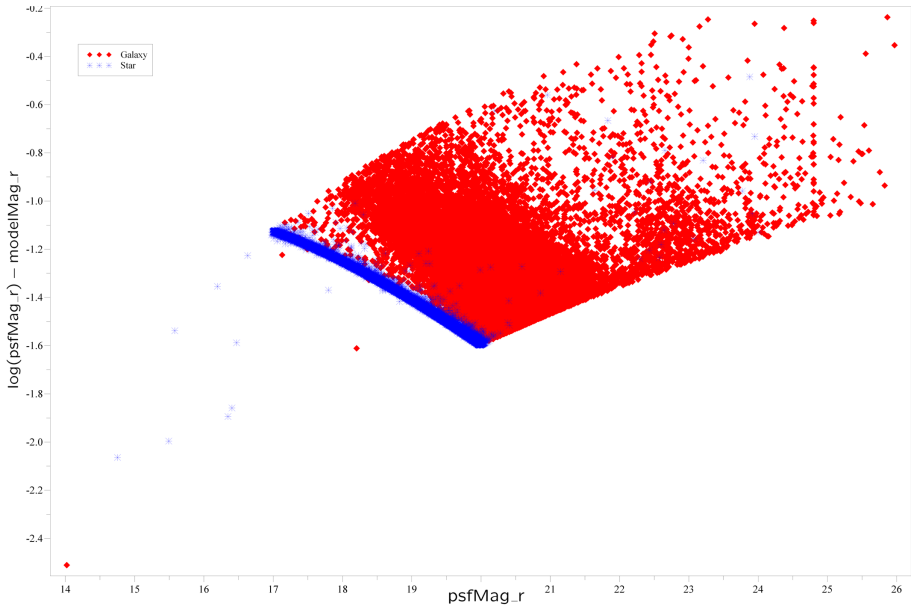

Figure 14: This SDSS star/galaxy plot produced by AV clearly separates the two classes while maintaining simple, interpretable axes

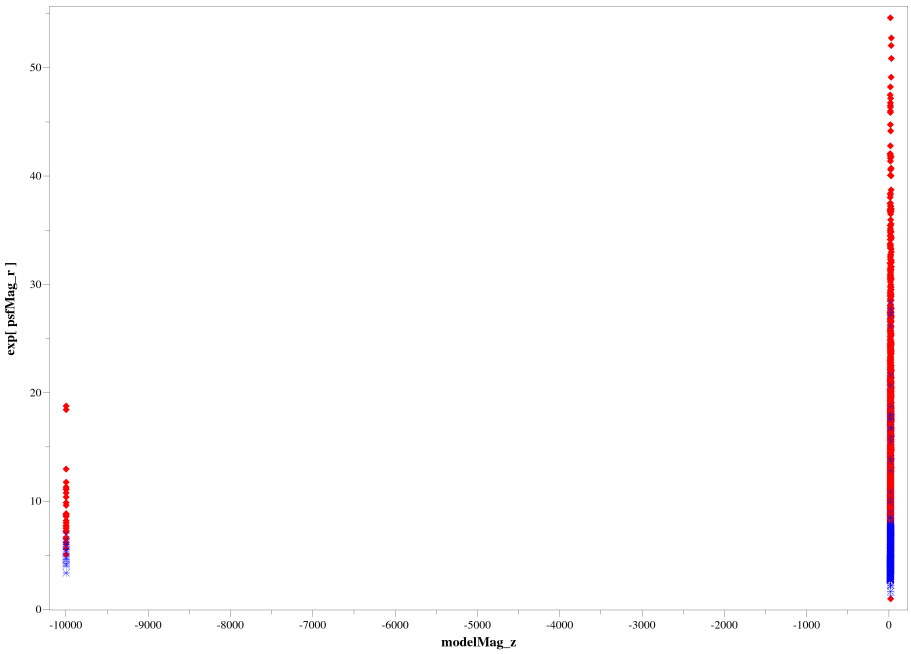

Figure 15: This SDSS star/galaxy plot was produced with AV limited to only unary expressions on each axis, underlining the necessity of the more complicated binary expressions

ings of the ACM SIGMOD International Conference on Management of Data, pages 94–105. ACM Press, Seattle, WA, 1998.

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[3] I. Borg and P. Groenen. *Modern Multidimensional Scaling*. Springer-Verlag, 1997.

[4] R. Caruana, H. Kangarloo, J. David, N. Dionisio, U. Sinha, and D. Johnson. Case-based explanation of non-case-based learning methods. In *Proceedings of the American Medical Informatics Association Symposium*, pages 212–215, 1999.

[5] Maria Cristina Ferreira de Oliveira and Haim Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3), July-September 2003.

[6] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley and Sons, New York, NY, 2nd edition, 2001.

[7] Brian C. Falkenhainer and Ryszard S. Michalski. Integrating quantitative and qualitative discovery: The ABACUS system. *Machine Learning*, 1(4):367–401, 1986.

[8] J. H. Friedman and J. W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Computers*, C-23(9):881–889, 1974.

[9] I.A. Galkin, B.W. Reinisch, X. Huang, R.F. Benson, and S.F. Fung. Automated diagnostics for resonance signature recognition on IMAGE/RPI plasmagrams. *Radio Science*, 2004.

[10] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems*. MIT Press, 2005.

[11] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.

[12] P. Langley. Data-driven discovery of physical laws. *Cognitive Science*, 5(1):31–54, 1981.

[13] G. Leban, M. Mramor, I. Bratko, and B. Zupan. Simple and effective visual models for gene expression cancer diagnostics. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 167–176, 2005.

[14] E. Lee, D. Cook, S. Klinke, and T. Lumley. Projection pursuit for exploratory supervised classification. *Comp. and Graphical Statistics*, 14(4):831–846, 2005.

[15] Dan Pelleg and Andrew Moore. Mixtures of rectangles: Interpretable soft clustering. In Carla Brodley and Andrea Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning*, pages 401–408. Morgan Kaufmann, San Francisco, CA, 2001.

[16] T. C. Sprenger, R. Brunella, and M. H. Gross. H-BLOB: A hierarchical visual clustering method using implicit surfaces. In *Proceedings of the Conference on Visualization '00*, pages 61–68, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[17] Sholom M. Weiss, Robert S. Galen, and Prasad V. Tadepalli. Maximizing the predictive value of production rules. *Artificial Intelligence*, 45(1-2):47–71, 1990.

[18] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, CA, 2000.