EFFECTIVE USE OF THE KDD PROCESS AND DATA MINING FOR COMPUTER PERFORMANCE PROFESSIONALS

Susan P. Imberman Ph.D.
College of Staten Island, City University of New York
Imberman@postbox.csi.cuny.edu

Abstract - *The KDD (Knowledge Discovery in Databases) paradigm is a step by step process for finding interesting patterns in large amounts of data. Data mining is one step in the process. This paper defines the KDD process and discusses three data mining algorithms, neural networks, decision trees, and association/dependency rule algorithms. The algorithms' potential as good analytical tools for performance evaluation is shown by looking at results from a computer performance dataset.*

1. Introduction

Knowledge Discovery in Databases (KDD) is the automated discovery of patterns and relationships in large databases. Large databases are not uncommon. Cheaper and larger computer storage capabilities have contributed to the proliferation of such databases in a wide range of fields. Scientific instruments can produce terabytes and petabytes of data at rates reaching gigabytes per hour. Point of sale information, government records, medical records and credit card data, are just a few other sources for this information explosion. Not only are there more large databases, but the databases themselves are getting larger. The number of fields in large databases can approach magnitudes of 10^2 to 10^3. Record numbers in these databases approach magnitudes of 10^9. [7] Computer performance data has also increased in size and dimensionality.

It is much easier to store data than it is to make sense of it. Being able to find relationships in large amounts of stored data can lead to enhanced analysis strategies in fields such as marketing, computer performance analysis, and data analysis in general.

The problem addressed by KDD is to find patterns in these massive datasets. Traditionally data has been analyzed manually, but there are human limits. Large databases offer too much data to analyze in the traditional manner.

The focus of this paper is to first summarize exactly what the KDD process is. This paper attempts to identify parts of the process that are currently being practiced by performance analysts and those parts that are not practiced, but may be useful in analyzing performance data The paper is organized as follows: Section 2 defines the KDD process. Section 3 discusses three data mining algorithms that are not commonly used by performance analysts. The paper concludes with a discussion in section 4.

2. KDD defined

KDD employs methods from various fields such as machine learning, artificial intelligence, pattern recognition, database management and design, statistics, expert systems, and data visualization. It is said to employ a broader model view than statistics and strives to automate the process of data analysis, including the art of hypothesis generation.

KDD has been more formally defined as “the non-trivial process of identifying valid, novel , potentially useful, and ultimately understandable patterns in data.” [7]

The KDD Process is a highly iterative, user involved, multistep process, as can be seen in figure 1.

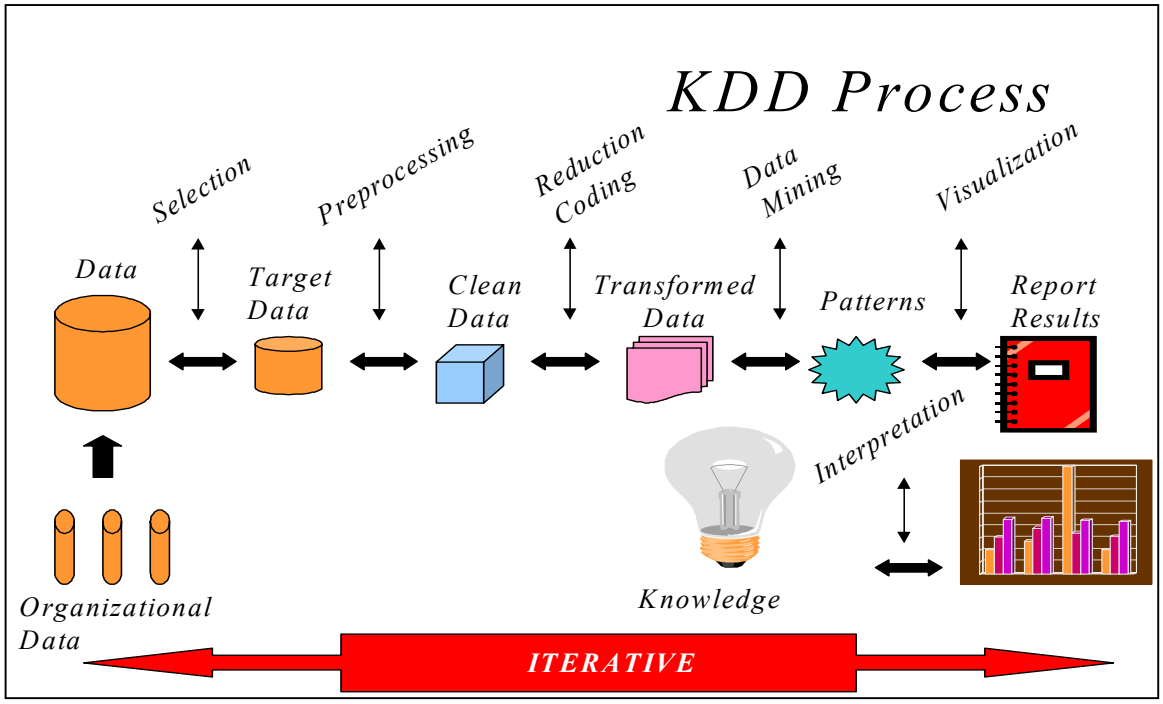

Figure 1 The KDD Process

From figure 1 we see that initially, we have organizational data. This data is the operational data gathered either in one or several locations. All operational data is collected and brought to some central location. These central locations are sometimes called Data Warehouses or Data Marts. Data transformation may be performed on the raw data before it is placed in the data warehouse. Data transformation resolves inconsistencies between the data of one location from that of another. For example, inconsistencies may be differences in data type for the same information and different field names for the same data field. Data warehouses hold both detailed and summary data. Detailed data is used for pattern analysis, where summarized data may hold the results of previous analyses. Data warehouses also contain historical data whereas operational data is usually current. Efficient organization of the data warehouse is essential for efficient data mining.

Once the data is organized, a selection process occurs where some subset of this data becomes the *target data* upon which further analysis is performed. It is important when creating this target data that the data analyst understand the domain, the end user's needs, and what the data mining task might be.

Sometimes data is collected in an ad hoc manner. Data entry mistakes can occur and/or the data may have missing or unknown entries. During the data cleaning and preprocessing stage noise is removed from the data. Outliers and anomalies in the data can pose special problems for the data analyst during the data cleaning process. Since the goal is to find rare patterns in the data, the outliers and anomalies may be representations of these rare patterns. Care must be taken not to remove these types of outliers and anomalies. This step in the process can be the most time consuming.

Data Reduction and Coding step employs transformation techniques that are used to reduce the number of variables in the data by finding useful features with which to represent the data.

Data Mining constitutes one step in the KDD process. The *transformed data* is used in the data mining step. It is in this step that the actual search for patterns of interest is performed. The search for patterns is done within the context of the data mining task and the representational model under which the analysis is being performed. It is important at this stage to decide on the appropriate data mining algorithm (linear/logistic regression, neural networks, association rules, etc.) for the data mining task (classification, database segmentation, rule generation, etc). The data mining task itself can be a classification task, linear regression analysis, rule formation, or cluster analysis.

Patterns generated in the data mining step may not be new or interesting. It is therefore necessary to remove redundant and irrelevant patterns from the set of useful patterns. Once a set of "good" patterns have been discovered, they then have to be reported to the end user. This can be done can be done textually, by

way of reports or using *visualizations* such as graphs, spreadsheets, diagrams, etc.

The interpretation step takes the reported results and interprets this into *knowledge.* Interpretation may require that we resolve possible conflicts with previously discovered knowledge since new knowledge may even be in conflict with knowledge that was believed before the process began. When this is done to user's satisfaction, the knowledge is documented and reported to any interested parties. This again may involve visualization.

It is important to stress that the KDD process is not linear. Results from one phase in the process may be fed back into that phase or into another phase. Current KDD systems have a highly interactive human component. Humans are involved with many if not each step in the KDD process. Hence, the KDD process is highly interactive and iterative.

3. Data Mining

Data mining is one step in the KDD process. It is the most researched part of the process. Data mining algorithms find patterns in large amounts of data by fitting models that are not necessarily statistical models. This is done within computational limits such as time (answers should be found within a finite amount of time) and hardware. Data mining can be used to verify a hypothesis or use a discovery method to find new patterns in the data that can do predictions with new, unseen data.

The term data mining is a somewhat recent one. In the past data mining has been known in the literature as knowledge extraction, information discovery, information harvesting, exploratory data analysis, data archeology, data pattern processing, and functional dependency analysis.

Performance analysts employ many of the same methods that data miners use when analyzing data. These include statistical models such as linear and logarithmic regression, cluster analysis, graphical analysis, and visualizations. There are some algorithms employed by data miners that are not usually used by performance analysts. Decision tree algorithms, neural networks, and association/dependency rule algorithms can be useful analytical tools for the performance analyst. Rather than discuss techniques that are often used by performance analysts, the remainder of this paper will elaborate upon these three algorithms. A performance dataset will be used to describe these algorithms and their use in performance analysis. The dataset contains data collected over a period of several months. A summary dataset was created that aggregates the measurements collected every 15 minutes into daily summary points. Our simple dataset contains 6 metrics:

- Date,
- Total Number of transactions,
- CPU busy %,
- Disk busy %,
- Average Response Time, and
- Average Network Busy %

The reader can observe from cursory data analysis (see the Figure 2 below) that the key relationships inherent in the data is as time moves forward, the number of transactions increases, and network busy percent decreases. Although this is a simplistic dataset in that the patterns in the dataset are quite obvious, because of this, the dataset is well suited for demonstration of the data mining techniques in the following sections.

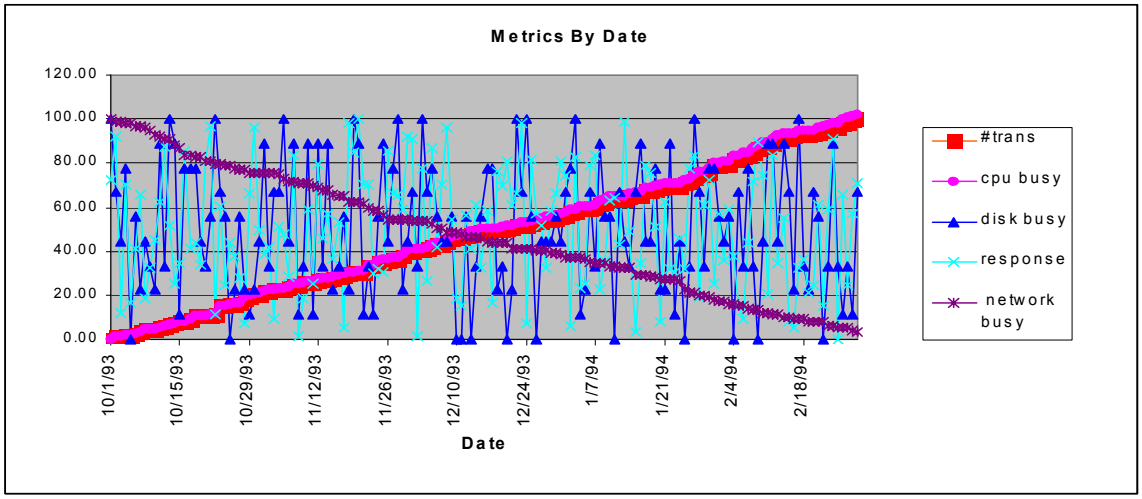

Figure 2 - Data Relationships

3.1 Association/Dependency Rule Algorithms

Dependency modeling uses methods that describe variable dependencies and associations. The results of these algorithm are what is known as association or dependency rules. [1] [2] [3] Association rule algorithms were developed to analyze market basket data. They identified groups of market items that customers tended to buy in association with each other. For example, customers tend to buy soap, shampoo, and hairspray at the same time. Because of their origins, much association rule terminology stems from this domain.

Given a set of items I, (or a set of variables from each transaction or record) an association rule is a probabilistic implication $X \Rightarrow Y$ where X and Y are subsets of I and $X \cap Y = \phi$. Interesting association rules are defined by two metrics, support and confidence. Support is a measure of a rule's significance with respect to the database, and confidence is a measure or the rule's strength.[1]

CPU busy	Network Busy	Response Time High	Network Busy High
1	0	1	0
1	1	0	0
1	1	0	1
1	0	1	1
0	1	0	0
1	0	1	1
1	1	0	0
1	1	1	1
0	1	1	1
1	1	0	1

Table 1 Sample Dataset

More formally, let N be the number of transactions (records) in a database. A set of items is said to satisfy a transaction if that transaction contains those items. For example, given N = 10 as in Table 1, if our item set were {CPU busy = 1, Network busy = 1}, then all records containing values that match the itemset for CPU busy and Network busy would satisfy that set. In Table 1, there are 5 transactions that satisfy the itemset {CPU busy = 1, Network busy = 1}. Given an association rule $X \Rightarrow Y$, let S_1 be the number of transactions that contain or satisfy X. Let S_2 be the number of observations in T that contain or satisfy $X \cup Y$. The support s of $X \Rightarrow Y$ is S_2 / N (or the s% of the transactions/records that contain $X \cup Y$) and the confidence c of $X \Rightarrow Y$ is S_2 / S_1 (the c% of all transactions N that contain X which also contain Y). For example, if we were to look at Table 1, given the association rule CPU busy = 1 $\Rightarrow$ Network Busy = 1, then the support would be 5/10 or 50%. Since there are 8 transactions that contain CPU busy = 1, the confidence for this association rule would be 5/8 or 63%. Note if the association rule we were looking at was Network Busy = 1 $\Rightarrow$ CPU busy = 1, the support would still be 50% but the confidence would be 5/7 or 72% since there are seven transactions with Network Busy = 1. The support of a rule is the rule's statistical significance in the dataset. The confidence is a measure of the rule's strength relative to the dataset. A rule that has a support above a user defined threshold, *minsup*, and confidence above a user defined threshold *minconf*, is an interesting association rule.

The problem of finding interesting association rules was first explored by Agrawal et. al. [1] [2] [3]. Agrawal et. al. decomposed the problem of finding association rules into two parts. The first focuses on finding large itemsets. A large itemset is a set of database items in that have support in the database greater than the user specified minimum (minsup). Once the set of large itemsets is found, the second part of the problem, uses these itemsets to generate rules that have confidence above a user defined minimum (minconf). Finding rules from large itemsets is straightforward. Finding the large itemsets themselves is an exponential problem, and hence computationally expensive to solve. Agrawal et. al. proposed the Apriori algorithm to find large itemsets and the association rules based on these large itemsets. [1] [2] [3].

In order to use Apriori with the computer performance data, we needed to "booleanize" the data. Association rule algorithms evaluate data that has either a zero or one value. This evolved from their origins which was the analysis of market basket data. A customer bought an item (value = 1) or didn't buy the item (value = 0). *Booleanizing* numeric data involves finding a threshold, above which every data point becomes a one, and below which it becomes zero. Since mean has been shown to yield a good threshold, it was used to booleanize the computer performance data. [10] The results given by Apriori with minsup set to 50% and minconf set to 80% on the booleanized computer performance data yielded the following two rules:

transactions $\Rightarrow$ date
(support = 51.0%, confidence = 98.7%)

date $\Rightarrow$ # transactions
(support = 50.3%, confidence = 100.0%)

Note that association rule algorithms only find positive dependencies. In computer performance, negative dependencies are also significant. Silverstein, Brin, and Motwani proposed a modification of Apriori that finds negative dependencies. [14] Domanski, Imberman, and Orchard have proposed the Boolean Analyzer algorithm that also finds these negative dependencies. The resulting rules are ordered based on a probabilistic interestingness measure (PIM). [6] [10] [14] Results from Boolean Analyzer on the

booleanized computer performance data are shown in table2.

Rule	Interpretation
date = 0 ⇒ # transactions = 0	As time decreases, Number of Transactions decreases
date = 1 ⇒ # transactions = 1	As time increases, Number of Transactions increases
date = 0 ⇒ network busy = 1	As time decreases, network busy increases
date = 1 ⇒ network busy = 0	As time increases, network busy decreases
date = 0 ⇒ # transactions = 0 && network busy = 1	As time decreases, Number of Transactions decreases and network busy increases
date = 1 ⇒ # transactions = 1 && network busy = 0	As time increases, Number of Transactions increases and network busy decreases

Table 2 - Results From Boolean Analyzer

Here we not only see the positive dependencies but the negative dependency between the date and network busy.

3.2 Decision Trees

The model representation for decision trees is a tree data structure. As in *figure 3* each node in the tree represents an attribute. Node attribute labels are chosen by a function which can separate the attributes such that values within a node are most similar and between nodes are dissimilar. Different decision tree algorithms use different functions to label each node. The algorithm C4.5 uses *information gain* to decide which attribute labels a node.[11] CART uses the *Gini index*. [4] CHAID uses a chi square statistic to make this determination. The attribute that best classifies the training examples is the one that labels the node. Each attribute value points to a child node. Nodes become leaves when its highest measured attribute classifies all training examples at that node or all attributes have labeled nodes. In Figure 3, A_i stands for an attribute or variable in the dataset, where v_{ij} are the specific values the decision tree algorithm found as good splitting values for this attribute. The input to a decision tree is a set of training examples. Training examples are records from the dataset where one attribute, the target variable, is correctly classified. In Figure 3, training examples were classified with the values yes or no. The tree algorithm learns and generalizes this classification and therefore can classify unseen examples. One of the drawbacks of decision trees is that they can tend to overfit the data. Different algorithms employ different methods of *pruning* the trees to compensate for this.

Each path in the tree represents a conjunction of attributes. The tree is a disjunction of each conjunctive path. There can be many decision trees that can describe a given training set. Usually decision tree algorithms generate multiple trees and recommend the "more accurate" tree as a classifier. Another method for "deciding" between trees is to use these multiple trees as a "committee of experts" for classification.

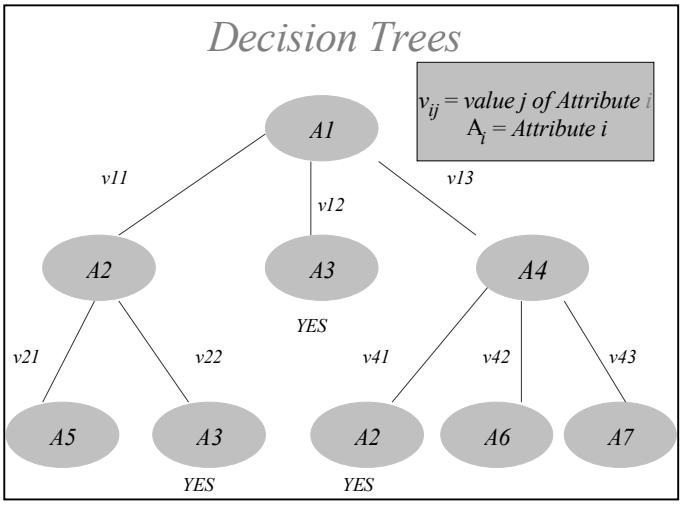

Figure 3 - Decision Trees

C4.5 requires that the target variable be categorical. To use the performance data with C4.5, the target variable, number of transactions, was booleanized. A value of 1 meant a high number of transactions, and 0 described a low number of transactions. For the performance dataset, C4.5 yielded the following results:

Class specified by attribute `num transactions'

Read 152 cases (6 attributes) from 430DATA2.data

Decision tree:

network busy <= 45: 1 (77)
network busy > 45: 0 (75)

Results from the Salford Systems' version of CART on the performance data with a booleanized target variable (number of transactions) is shown in figure 4.

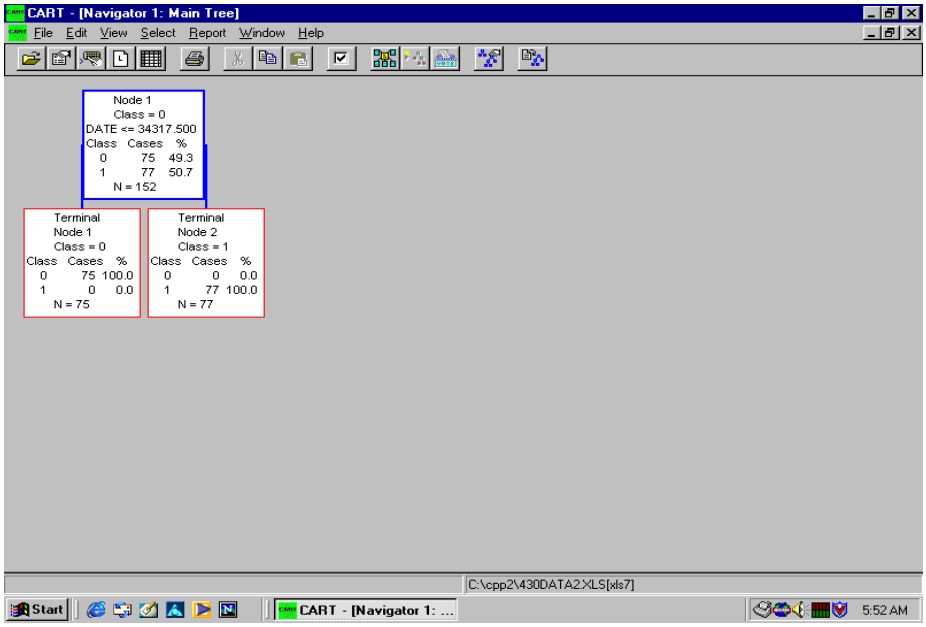

Figure 4 - CART Decision Tree

CART chose date as the splitting attribute as opposed to the network busy attribute chosen by C4.5. Actually both attributes are almost identical in their ability to classify the data with only hundredths of a percent difference between the two. The choice differences displayed by the two methods can be attributed to the different functions used by each algorithm to distinguish between attributes.

CART can also evaluate data using a numeric target variable, thus creating a *regression tree*. The regression tree for the performance data using the original numeric values for number of transactions is shown below.

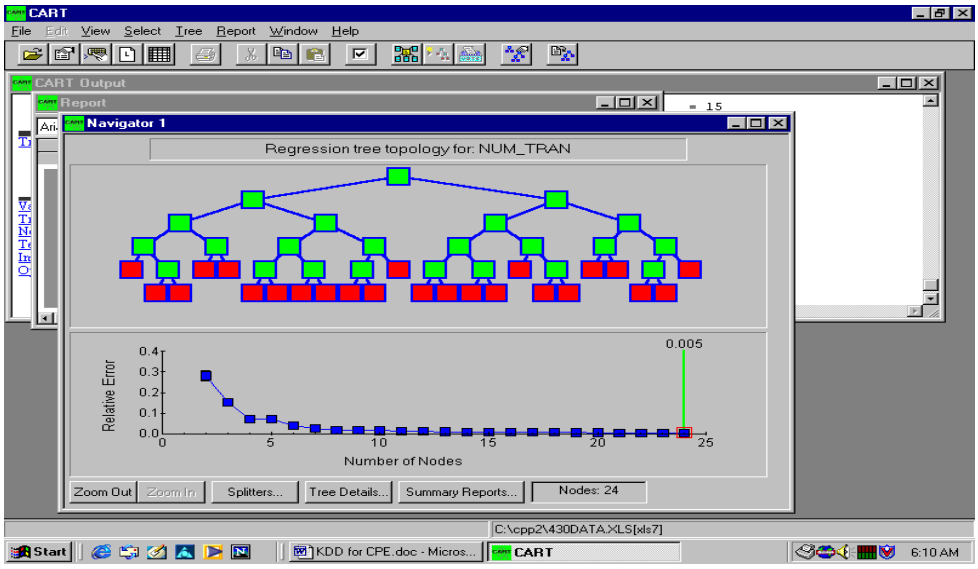

Figure 5 - CART Regression Tree

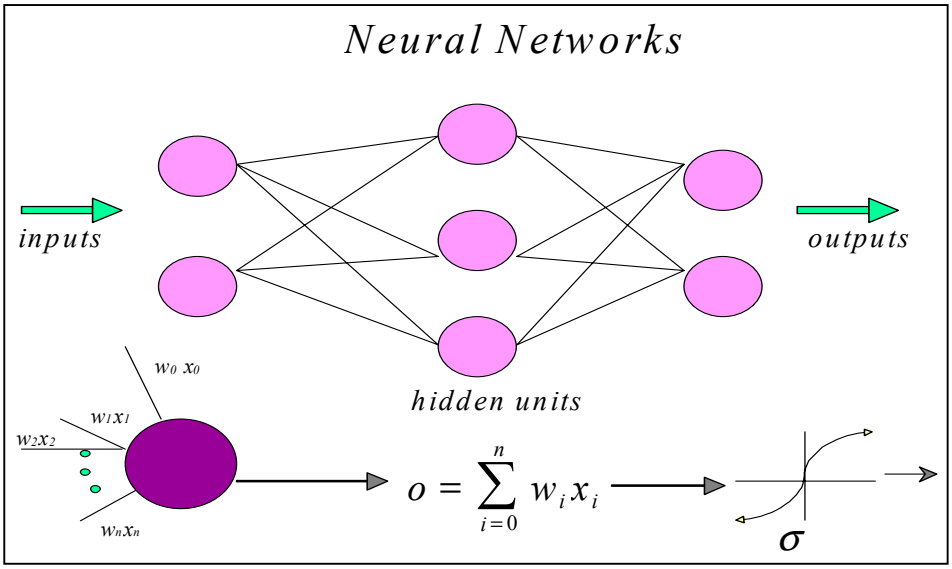

Figure 4 - Neural Networks

Each path in the tree can be interpreted into a rule. For example the far left (first) path yielded the rule:

```
if  (DATE <= 34249.5)
{
   terminalNode = -1;
   mean = 190.571;  //average number of transactions
}
```

The second path yielded the rule:

```
if  (    DATE > 34249.5 &&    DATE <= 34254.5 )
{
   terminalNode = -2;
   mean = 505.4;   //average number of transactions

}
```

Decision trees are good classifiers with easily interpreted results.

3.3 Neural Networks

Neural Networks are a graphical model based on the human nervous system. [11] It consists of a system of connected nodes arranged in multiple layers. In Figure 4 we see that there is an input layer, a hidden layer (there can be several hidden layers), and an output layer. Inputs to the neural network are the same types of training examples that decision trees take as their input. Each node is associated with an input value and a weight. The weights are multiplied by their respective input values and summed. The result of some function of this sum is the output value from the node that is forwarded to the next layer of nodes. [5] [11]. The outputs of a neural network is a set of weights that are the coefficients in a linear regression equation.

One method for determining the network weights is back propagation. Back propagation looks at the error between the networks' output value and the observed value and then adjusts the weights on each node so that this error is reduced. This is repeated for each example fed into the network and repeated multiple times for the entire dataset. Thus it is a highly iterative process. This process is called "training" the neural network.

The neural network itself represents a learned function that can be used to predict on new examples. The representational model of neural networks is difficult to interpret, as opposed to decision trees since we don't always known what is happening in hidden nodes. Neural nets result in a linear regression equation rather than a set of rules. Although neural networks can be slow to train, once trained and implemented, neural networks can classify inputs very quickly.

Nnmodel, a neural net program, was used to evaluate the performance data. [12] For this analysis, 20 randomly selected records were withheld as a test set. The neural net was trained on the remaining 132 records. This is a common practice done when training classifiers. The withheld data is used to benchmark the classifiers ability to predict on unseen data. Figure 5 shows the results of the neural network's predications as compared to the actual values on the withheld test set. Notice that the two are very similar, thus showing that the neural networks did a good job of predicting the number of transactions.

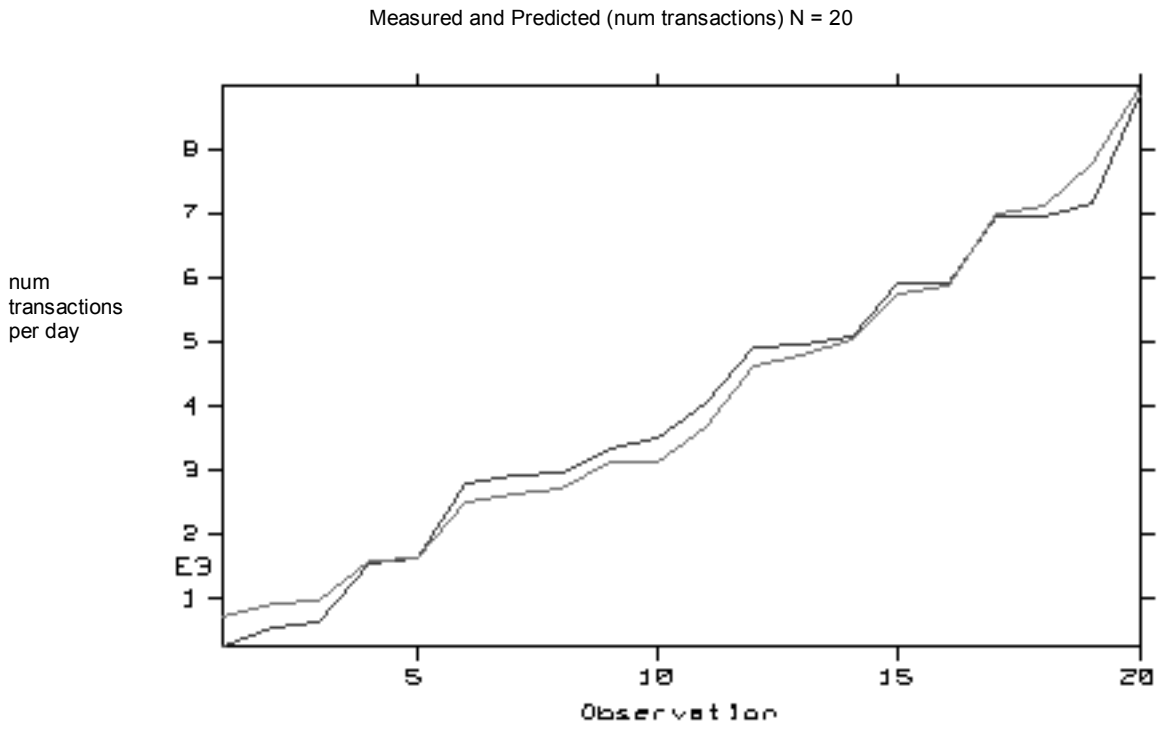

Figure 5 Neural Network Results vs. Observed Results

4. Discussion

The KDD process is a good paradigm for data analysis. One may wonder what the differences between traditional statistical analyses and the KDD process are. Hand [8] discusses these differences. He maintains that methods employed by KDD use statistical approaches that have the same rigor one usually employs in statistics, but many KDD methods tend to be more experimental and less conservative than those used in statistics. Thus they are more suited for a "discovery approach" to data analysis.

Of the data mining techniques discussed in this paper, there are situations that favor one over the other. In addition each method has their drawbacks. Neural networks and decision trees are similar in that they are both classifiers. The classifier model is built and is then used to predict the target class for future unseen data. The advantage that decision trees have over neural networks is that the classification rules are directly readable from the model. Many data analysts don't like the "black box" model built by neural networks. This is because the result of a neural network model is a set of linear regression equations, thus making the reasons for classification difficult to interpret.

Decision tree models are faster to build. It can take a long time to train a neural net. Notwithstanding, once built, the execution of a neural net model can be faster than that of a decision tree. In terms of accuracy, studies have revealed that one method is not necessarily more accurate on the other. Accuracy tends to depend on the actual data involved.

Association/Dependency rule algorithms show concurrence of variables. They tend to generate many more rules than what is seen in decision trees. The advantage is that these algorithms give us the ability to find rare and less obvious patterns in the data. There have been attempts to use the rules generated by association/dependency rule algorithms for classification but using association/dependency rules in this manner has met with mixed results.

It is important that these data mining algorithms not be used in an ad hoc manner. This has come to be known as data dredging. The fear is that doing so, one might discover patterns with no meaning. In fact at one time KDD in the statistics community was considered a “dirty word”. It has been shown that if one looks hard enough in a sufficiently large database, even a randomly generated one, statistically significant patterns can be found.

We also have to be careful as to how we interpret the patterns presented to us by data mining algorithms. One classic data mining legend centers around a major retailer finding the relationship, “Men who buy diapers, buy beer!!” The retailer , might assume that harried dads, going to the store to pick up some diapers, were picking up something for themselves as well!

Notwithstanding, the KDD process, and specifically the data mining step in the process, offer

the performance analyst additional approaches for data analysis.

5. Conclusions

In this paper, a very simplistic computer performance dataset was used to demonstrate three data mining algorithms. The reasons for using such a simplistic set was to give the reader a "taste" for the methods discussed. Usually these algorithms are used on highly dimensional, larger, complex datasets. It was my hope that in demonstrating the functional abilities of association/dependency rules, neural networks, and decision trees on a simple set, the reader would be encouraged to try these methods on their own data.

There are many resources available that implement these algorithms. A nice implementation of Apriori was coded by Christian Borgelt and can be found at:

http://fuzzy.cs.uni-magdeburg.de/~borgelt/apriori/

Along with a discussion of the algorithm, there is also downloadable code. Borgelt's version of Apriori was implemented in SPSS's data mining suite, Clementine.

Another good site is:

http://www.cs.waikato.ac.nz/ml/weka/

This site contains *Weka 3* which is a suite of machine learning and data mining algorithms written in Java with a decent GUI.

http://www.kdnuggets.com/ is a site that has many links to software, both commercial and free, along with other interesting links relating to the field of data mining.

For those interested in reading more on the topic of data mining, [7] is a collection of papers that give a nice overview of the field. [11] is a good text on machine learning. [6] Gives a good introduction to Neural Networks. A recent text by Jiawei Han and Micheline Kamber titled, Data Mining Concepts and Techniques, Academic Press 2001, gives a good introduction to the field in general.

Acknowledgements: I would like to thank Bernie Domanski for his help on this paper and keeping me focused on the needs of the computer performance professional.

6. References:

1. Agrawal, R., T. Imielinsk, and A. Swami. "Mining Association Rules between Sets of Items in Large Databases." *Proceedings of the ACM SIGMOD International Conference on the Management of Data,* 207-216, 1993.

2. Agrawal, R. Mannila, H., Srikant, R. Toivonen, H. and Verkamo, A. I., *Fast Discovery Of Association Rules.* In Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramaswamy Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 307--328. AAAI/MIT Press, 1996.

3. Agrawal, R. and Srikant, R. *Fast Algorithms For Mining Association Rules*. In Proceedings of 20 th Intl. Conf. on Very Large Databases (VLDB'94), pages 487--499, Santiago, Chile, 1994

4. Breiman, L., Friedman, J. H. , Olshen, R. A. , Stone, Charles J. *Classification and Regresion Trees,* Chapman and Hall/CRC, 1984.

5. Domanski, B., *A Neural Net Primer,* Journal of Computer Resource Management, Issue 100, Fall 2000.

6. Domanski, B., 1996. *Discovering the Relationships Between Metrics. The Proceedings of the 1996 Computer Measurement Group.* December 1996, San Diego California, 309 – 313.

7. Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. *From data mining to knowledge discovery: An overview*. In Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramaswamy Uthurusamy, editors, Advances in Knowledge Discovery and Data Mining, pages 1--31. AAAI/MIT Press, 1996.

8. Hand D. J., *Statistics and Data Mining: Interesting Disciplines*, Association for Computers and Machinery SIGKDD Explorations, June 1999 Vol. 1 Issue1, pgs 16 - 19.

9. Imberman, S. 1999. *Comparative Statistical Analyses Of Automated Booleanization Methods For Data Mining Programs* (Doctoral dissertation, City University of New York, 1999). UMI Microform, 9924820.

10. Imberman, S.P., Domanski, B., Orchard, R., 1999. *Using Booleanized Data To Discover Better Relationships Between Metrics.*, CMG99 Proceedings, 1999.

11. Mitchell, T. M., Machine Learning, WCB McGraw-Hill 1997.

12. Nnmodel, http://www.cyberave.com/~carlb/

13. Orchard, R. A. *On the Determination of Relationships Between Computer System State Variables*. *Bell Laboratories Technical Memorandum,* January 15, 1975

14. Silverstein, C., Sergey Brin, and Rajeev Motwani. Beyond Market Baskets: Generalizing Association Rules to Dependence Rules. Data Mining and Knowledge Discovery, 2(1):39-68, 1998