INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

COMPARATIVE STATISTICAL ANALYSES OF AUTOMATED BOOLEANIZATION METHODS FOR DATA MINING PROGRAMS

by

SUSAN P. IMBERMAN

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1999

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/21/99
Date

Chair of Examining Committee

4/27/99
Date

Executive Officer

Dr. Bernard Domanski

Dr. Charles Giardina

Dr. Joesph Hellerstein
Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

COMPARATIVE STATISTICAL ANALYSES OF AUTOMATED BOOLEANIZATION METHODS FOR DATA MINING PROGRAMS

by

Susan P. Imberman

Advisor: Professor Michael Kress, Professor Bernard Domanski

KDD (Knowledge Discovery in Databases) is the automated discovery of patterns and relationships in large databases. Data mining is one step in the KDD process. Many data mining algorithms and methods find data patterns using techniques such as neural networks, decision trees, statistical analysis, deviation detection, etc. The Boolean Analyzer is a data mining method that finds dependency rules of the form $X \Rightarrow Y$. Data is Booleanized with regard to values in a threshold set. That is each data transaction/observation is transformed into a vector of 0's and 1's. Each vector defines a state for that transaction/observation. Vector states can be organized into a state occurrence matrix. From this matrix we can compute a measure of event dependency. A new matrix can be formed called a state linkage matrix that consists of the measures of the event dependency represented by a row and a column in the state occurrence matrix. The values of the state linkage matrix can be used to find the measures of more complex relationships. In turn complex relationships can define rule systems where the measure of a rule system is the value of the complex rule from which the rule system was derived. Rule systems can be more easily implemented that the complex rule.

A significant step in the above process is the Booleanization step. It is this step that most directly affects the statistical significance and strength of the rules generated by

the algorithm. In the past expert in the domain where the data analysis was being done determined threshold sets. There is a basis for automating the formation of the threshold set. Rules generated by the Boolean Analyzer algorithm using an expert threshold set were compared with rules generated with threshold sets formed using the mean, mode, median, and clustering of variable values. This yields the conclusion that threshold automation using mean and median values was a valid alternative to threshold formation by an expert. Subsequent analysis using mean and median thresholds on synthetic data with known relationships produced excellent results.

Acknowledgements

This doctoral thesis is dedicated to:

My husband Steven. I thank you for encouraging me to complete the process known as a doctorate. You are my love, my life, my soul mate. I love you.

Dr. Bernie (Bernie Domanski). There are few people who would extend themselves to invest the time and energy needed to help a friend in the way that you helped me. Words can't express the gratitude I feel towards you. I look forward to our continuing friendship and professional collaboration.

Mike Kress. In every project there is one person who lays the path for the others to follow and keeps them focused on that path. I am glad you were the person to focus me. Thank you so much for stepping in and helping me finish my doctorate.

Charlie Giardina. Your confidence in me was greatly appreciated. I considered it high praise. Thank you for making sure my mathematics was sound.

Joe Hellerstein, thank you for taking the time to be on my doctoral committee.

Scott, Matthew, and Adam, my three sons. Thank you for not complaining too much when I didn't have the time to do everything you wanted me to do. In spite of what you may have thought at times, you guys were always my first priority. I love you all.

My parents, who have always been supportive of whatever I do. Thank you for your encouragement, love, and support. You are the best.

My husband's parents, who are no longer with us. We miss you.

My friends, colleagues, and mentors of the Computer Science Department, College of Staten Island. Thank you for your encouragement and support. You taught me everything I needed to know.

My friends and family. I took my inspiration from all of you.

Table of Contents

List of Tables

List of Figures

Chapter 1

Knowledge Discovery in Databases

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. 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. For example, in marketing, retail data can yield insight into relationships between the sale of specific items and certain demographic groups.

Not only are the numbers of large databases growing, but also 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. The problem is trying to find a pattern in the data, given millions of records with tens or hundreds of fields. Traditionally data has been analyzed manually, but there are human limits. Large databases offer too much data to analyze in any conventional manner.

SQL queries can be inadequate when it comes to handling information from large databases. The type of information needed from these databases can also be hard to

formulate in SQL. For example, it is not uncommon to query the database for answers to questions such as, find me all calling card irregularities, find me a stock that will do well, or what are the buying preferences of people who buy Barbie dolls. KDD methods are attempts to handle these types of problems.

1.1 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 can be more formally defined as "the non-trivial process of identifying valid, novel , potentially useful, and ultimately understandable patterns in data." (Fayyad, U. M., G. Piaetsky-Shapiro, and P. Smyth, 1996)

Given the above definition:

- Let **F** be a set of facts or data
- **E** is an expression in some language **L**
- Given $F_g \subseteq F$
- Then **E** is a pattern if it is simpler then F_g

Patterns are valid if they fall within certain bounds of certainty. Let certainty be some function **C** that maps expressions **E** in **L** to a partially or totally ordered set of measurements M_c where each element **c** of M_c can be defined by $c = C(E,F)$.

Now let us define what is meant by a novel pattern. When we speak of novel patterns, novelty refers to the system in which we are currently working, not the world at large. Therefore, define a function **N(E,F).** Novelty is some function **N** that maps expressions **E** in **L** to a partially or totally ordered set of measurements $\mathbf{M}_n$ where each element **n** of $\mathbf{M}_n$ can be defined by **n = N(E,F).** Essentially, if you haven't already discovered the pattern, then it is novel.

For a pattern to be "potentially useful", it must have the potential of resulting in some useful action. Let **U** be a utility function that measures this potential. **U** maps expressions **E** in **L** to a partially or totally ordered set of measurements $\mathbf{M}_u$ where each element **u** of $\mathbf{M}_u$ can be defined by **u = U(E,F).**

An "ultimately understandable pattern", is one that is understandable to the end user. Understandability is a hard parameter to measure. Let's assume a pattern is understandable if it is "simple". Then we can define a simplicity function **S** where **S** maps expressions **E** in **L** to a partially or totally ordered set of measurements $\mathbf{M}_s$, where each element **s** of $\mathbf{M}_s$ can be defined by **s = S (E,F).** Simplicity is a user-defined function. What might appear simple and understandable to one individual may not be simple and understandable to another.

It is not enough to generate understandable, useful, novel, valid patterns. We may not be *interested* in all the patterns. Hence we define the concept of "interestingness" as the ability to measure how interesting a pattern may be. Let **I** be an interestingness function that maps expressions **E** in **L** to a partially or totally ordered set of measurements $\mathbf{M}_I$ where each **i** of $\mathbf{M}_I$ can be defined as **i = I(E,F,C,N,U,S)** . This

means that interestingness is some function of expressions, facts, certainty, novelty, usefulness, and simplicity. Interestingness can be either measure explicitly or implicitly.

Our goal is to ultimately gain knowledge. Given a pattern $E \in L$, E is knowledge if for some user specified interestingness measure $i \in M_I$, $I(E,F,C,N,U,S) > i$. This is not an absolute measure and very much user defined. For example, as an instantiation of this we might have some thresholds $c \in M_c$, $s \in M_s$ and $u \in M_u$, then a pattern E is knowledge IFF $C(E,F) > c$ and $S(E,F) > s$ and $U(S,F) > u$.

1-2 The KDD Process

The KDD Process is a highly iterative, user involved, multistep process.

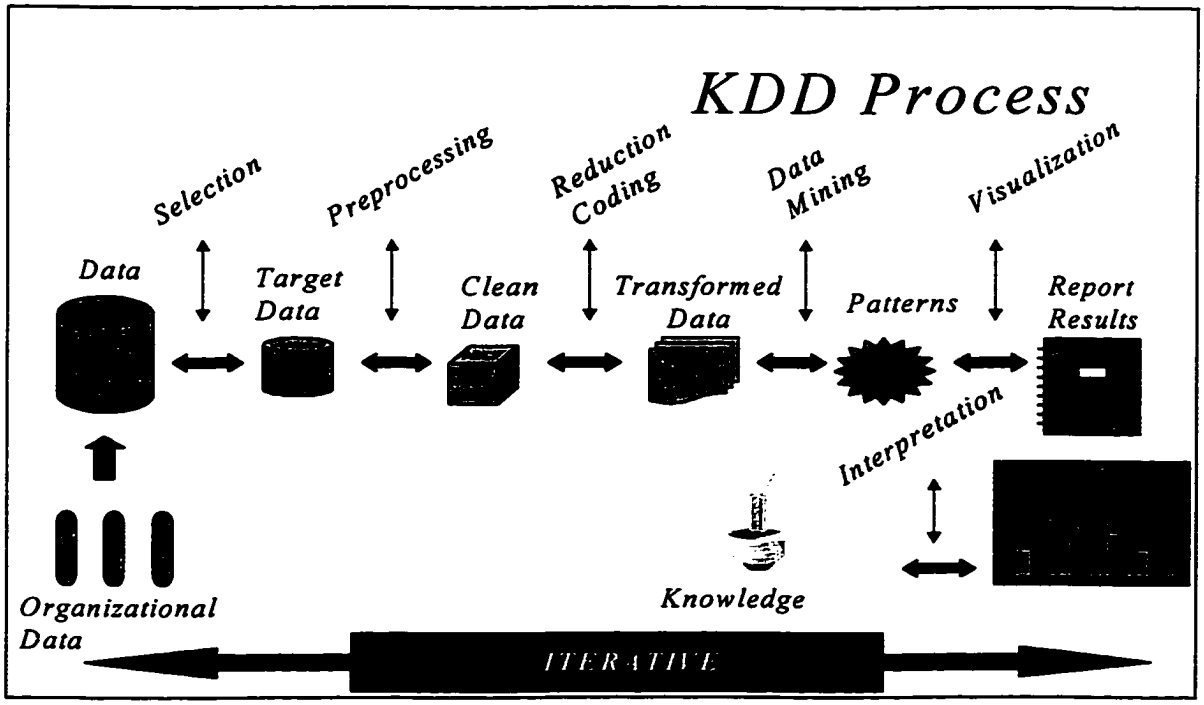


Figure 1 The KDD Process

1. Organizational Data - Initially, we have organizational data. This data is the operational data gathered either in one or in 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. There may be inconsistencies between the data of one location from that of another. These inconsistencies may consist of 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 more pattern analysis. Summarized data may hold the results of previous analyses so that work need not be redone. They also contain historical data whereas operational data is usually current.
2. Selection - Once the data is organized, a selection process occurs where some subset of this data becomes the *target data* and further analysis is performed. It is important when creating this target data, the data analyst understand the domain, the end user's needs, and what the data mining task might be.
3. Data Cleaning and Preprocessing – Sometimes data is collected in an ad hoc manner. Data entry mistakes can occur and/or the data may have missing or unknown entries. In this stage noise is removed from the data. Outliers and anomalies in the data pose special problems for the data analyst during this 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. Time sequence changes are also handled in this phase.

4. Data Reduction and Coding – Transformation techniques are used to reduce the number of variables in the data by finding useful features to use to represent the data. Time stamped data is difficult to store in databases. Time dependent data may need to be transformed to be made usable
5. Data Mining – 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 for the data mining task. The data mining task itself can be a classification task, linear regression analysis, rule formation, or cluster analysis. This is the most researched step in the process.
6. Visualization – Patterns generated in the data mining step may not be new or interesting, therefore these redundant and irrelevant patterns are removed 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 visually using graphs, spreadsheets, diagrams, etc.
7. Interpretation – Reported results are interpreted into *knowledge.* Interpretation may require that we resolve possible conflicts with previously discovered knowledge. New knowledge may even be in conflict with knowledge that was believed before the process began. When this is done to user satisfaction, the knowledge is documented and reported to any interested parties. This again may involve visualization. These parties then have the option of making use of the discovered knowledge.

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, KDD is highly interactive and iterative. One controversy in the KDD community centers around how much human interaction there should be in the process. On one side researchers argue that human involvement can either implicitly or explicitly add domain knowledge into the process. Others argue that needed domain knowledge may be implicit in the data. To date there are no totally autonomous systems for KDD.

Chapter 2

Data Mining

Data mining is one step in the KDD process. It is the most researched phase of the process. Data mining can be thought of as "a search through a space of possibilities". The more possibilities, the more chance of finding some useful information. Hence, data mining is synonymous with big databases.

More formally given a set of facts **F**, data mining results in an enumeration $\mathbf{E}_j$ of expressions, using a set of data mining algorithms. Data mining determines patterns from large amounts of data by fitting models, not necessarily statistical models, within computational limits such as time (answers should be found within a finite amount of time) and hardware. This is a goal defined process. Data mining can be used to verify a hypothesis or use some discovery method to find new patterns in the data that can be used to predict on new data.

The term data mining is a somewhat recent one. Other terminology for data mining in the literature include knowledge extraction, information discovery, information harvesting, exploratory data analysis, data archeology, data pattern processing, and functional dependency analysis.

2.1 Data Mining Models and Methods

Each data mining algorithm can be thought of as having three components, model representation, model evaluation, and some search methodology. Model representation is the language **L** use to represent the expressions(patterns) **E** in. It is related to the type of

information that is being discovered. The language can also dictate the types of patterns discovered. The importance here is to choose the correct representation. If too descriptive a language is chosen there is a danger of over fitting the data. Notwithstanding, the model has to be complex enough to explain the data but restrained enough to be able to generalize over new data. Model evaluation refers to the scoring methods used to see how well a pattern or model fits into the KDD process. Search methods include algorithms such as greedy search, gradient descent, etc. that are used during the KDD process.

Data mining algorithms can be thought of as falling into two categories, predictive and descriptive algorithms. Predictive data mining algorithms result in the ability to predict future events from historical data. Algorithms that fall into this category include classification algorithms such as decision trees, neural networks, and inductive logic programming (ILP) along with regression algorithms. Descriptive algorithms find some human interpretable rules, relationships, and/or patterns. These include algorithms that do deviation detection, clustering, database segmentation, summarization and visualization, and dependency modeling

2.1.1 Predictive Algorithms

The model representation for decision trees is a tree data structure. As in *diagram 2* each node in the tree represents an attribute. Node attribute labels are chosen by some entropy/information gain function. 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. An attribute will appear only once along any path.

Each path in the tree represents a conjunction of attributes. The tree is a disjunction of each conjunctive path. Cross validation, where examples are randomly and repeatedly partitioned into test and train sets, and significance tests such as Chi Square are used as evaluation methods for decision trees. Search in decision trees is a greedy hill climb through the space of all possible decision trees. There can be many decision trees that can describe a given training set.

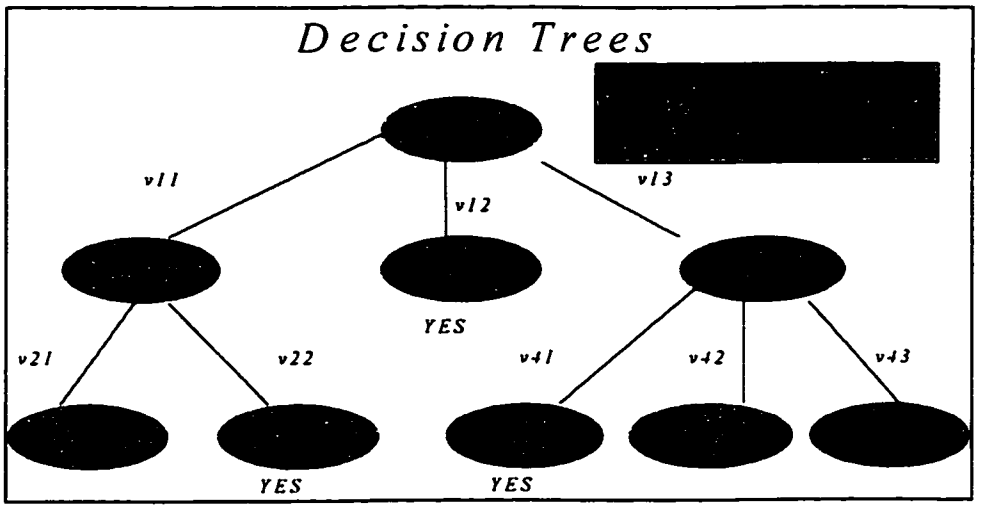


Figure 2 - Decision Trees

Different decision tree algorithms differ as to the functions used to in determining how to assign attributes. They also differ as to how to split the values. ID3 splits values as above whereas GID3 allows each child to have a subset of values. One of the drawbacks of decision trees is that they can tend to overfit the data. Different algorithms employ different methods of pruning trees.

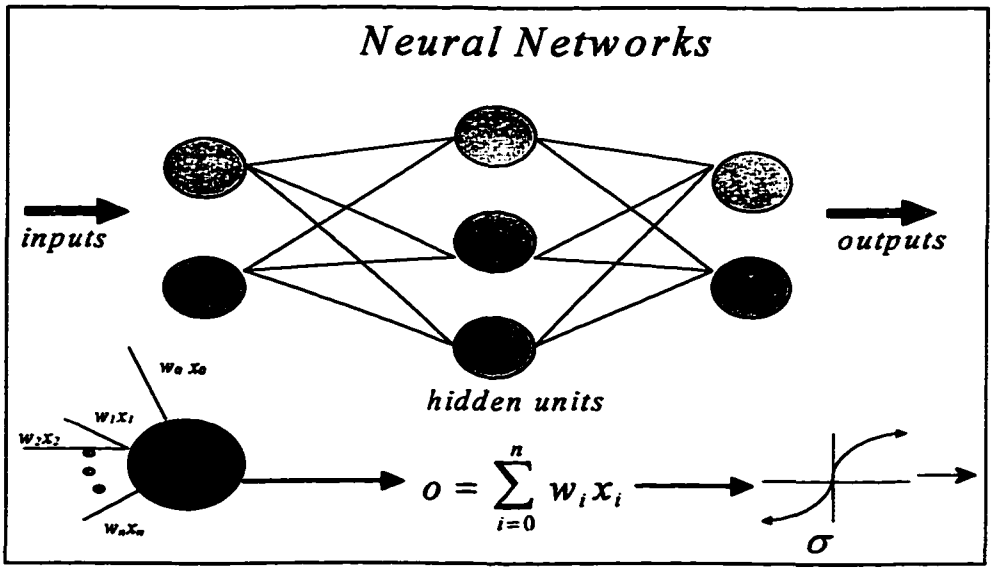


Figure 3 - Neural Networks

Neural Networks are a graphical model based on the human neural system. It consists of a system of connected nodes that are analogous to human neurons. Each input is associated with a weight. The result of some function of these weights is the output value from that is forwarded to the next layer of nodes. Backpropagation algorithms allow for the adjusting of weights in an iterative process. The result is that 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 since we don't always known what is happening in hidden nodes. Search is a gradient descent through the space of all possible functions that can be represented using weights applied to networks of interconnected nodes. Neural networks are evaluated using standard squared error functions and cross entropy loss functions.

The representational model for inductive logic programming (ILP) is first order logic. ILP algorithms learn concepts that describe patterns rather than just identifying

them. Concepts are learned from examples and domain knowledge, which is represented in predicate form. Learned concepts are also predicates. ILP algorithms that use empirical methods will find multiple or single predicate rules using large numbers of examples and domain knowledge. Interactive ILP systems are supervised learners that can learn from small numbers of examples and queries to the expert involved. Examples of empirical ILP programs are FOIL and GOLEM. Interactive ILP programs include MIS and CLINT. ILP algorithms use highly computational search methods. Evaluation is done through logical methods. This usually entails evaluation of how the learned rules predict over new data.

Regression analysis results in some function **F** that will take a data item **d** and map it to some real valued number. Regression algorithms result in a function. Time series applications are viewed as a special type of regression problem.

2.1.2 Descriptive Algorithms

Given a set of data, deviation detection determines if significant changes have occurred from previously measured values of the data. It concentrates on abnormal data changes. Methods used by these types of algorithms are significance testing, standard deviation, and means analysis.

Clustering is a computer automated method by which the data is partitioned into subsets such that within each partition the data is most similar, and between partitions the data is most dissimilar. Partitions form clusters which may or may not overlap depending on the algorithm. Within each cluster, similarity of data points may be with respect to more than one variable. Both Euclidean distance measures and Bayesian techniques have

been employed. The automated nature of this methodology classifies it as an unsupervised learner. Database segmentation is sometimes viewed as a special case of clustering. The database is partitioned into subsets. Data items are grouped according to a single feature.

Human pattern recognition abilities exceed that of any known automated method. Summarization and visualization techniques make use of these abilities. In an effort to couple machine analysis with human pattern recognition, data is encoded with respect to color, position, size, etc. In essence, the data is condensed to a simpler form. Summarization and visualization may offer different views of the data allowing for drill down into different segments of the data. Results are provided in a way that may lead to knowledge discovery.

Dependency modeling are methods that describe variable dependencies. The results of these algorithm are what is known as association rules or dependency rules. An association rule can be defined as being of the form $\mathbf{X} \Rightarrow \mathbf{Y}$ where **X** and **Y** are disjoint sets of items.[Agrawal, R., T. Imielinsk, and A. Swami, 1993] Rules have a true or false value associated with them, along with an associated confidence measure. In this definition, variables are connected with conjunction and no negations were allowed. More recently, the concept of an association rule was extended to include negations. [Silverstein, C., Sergey Brin, and Rajeev Motwani, 1998]

2.1.3 Evaluation

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 generate 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 beer, buy diapers!!" The retailer , might assumed that harried dads, going to the store to pick up some diapers, were picking up something for themselves as well. To capitalize on this, store stock was rearranged so that beer and diapers were near each other. This resulted in increased sales of both beer and diapers. In another example, Wal-Mart found a relationship that people who bought Barbie dolls also bought a certain candy bar. Suggested ways of capitalizing on this relationship ranged from shaping the candy like a Barbie doll to placing food coupons for the candy bar inside Barbie Doll packages.

2.2 Data Mining Applications

Data Mining applications fall into two categories. They are either generic or domain specific. Generic tools can be used across multiple, possibly unrelated, domains. The goal is to be able to use the tool without too much domain specific customization. Generic single task tools are used mainly in the mining step of the KDD process. These tools usually require a great deal of pre and post processing. Generic multitask tools may combine a variety of data mining methods and may even incorporate some visualization techniques. Domain specific applications do discovery in a single domain. The user

doesn't need to know about the discovery process. All data mining systems are different. To date there is not one program that addresses knowledge discovery for all domains.

2.2.1 SKICAT

SKICAT , Sky Image Cataloging and Analysis Tool, was used by astronomers to classify and catalog sky objects from astronomical data.[Fayyad, U. M., S. G. Djorgovski, and N. Weir, 1996] It was used on images obtained from the Second Palomar Observatory Sky Survey (POSS-II). Three thousand digitized images of 23,040 X 23,040 16 bit pixels comprising a total of 3 terabytes were collected in this survey. The object was to catalog sky objects such as galaxies, and stellar objects including quasars. The survey covered the entire northern sky. The main goal of the project was to more accurately catalog sky objects in an automated , timely manner than has previously been done in former surveys. As a subgoal, the project tried to detect images from these photographic plates that were too faint for visual recognition. The objects detected by the survey were one order of magnitude fainter than in previous surveys. The size of the dataset and the difficulty of visual interpretation necessitated some automated means for the interpretation of the data.

SKICAT used decision tree supervised learning. Astronomers using high resolution data from telescopic CCD images classified faint objects. CCD images have a higher resolution and a higher signal to noise ratio than the photographic images. This allowed the recognition and classification of faint objects by astronomers. This dataset only coincided with a small subset of the data obtained with POSS II. The labeled low

resolution images of the classified objects was used to train the decision tree. Multiple decision trees were generated on randomly generated test and train sample sets.

Additional classification learning algorithms such as neural nets and unsupervised Bayesian learning (AutoClass) were also investigated. Results with neural nets were almost as good as that with decision trees. When AutoClass was used as the learner, 10 new high red shift quasars were discovered. These were found to correspond to rare clusters in the data. Research still continues with regard to AutoClass.

RULER was a subsystem that was used to obtain a minimal set of rules that cover the examples from the rules generated by the decision trees. In RULER, the decision trees were pruned with regard to a statistical measure of a rule's conditions, correlated with the class by chance. Rules with probabilities above a certain threshold (.01) were pruned from the tree. A greedy covering algorithm was then applied to find a minimal set of rules that cover all the training examples. The result was a set of rules that was smaller than any of the rules in the decision trees that created this set.

SKICAT was able to find objects that were one order of magnitude fainter than previous with above 90% accuracy. The accuracy of SKICAT's algorithm was evaluated in two ways. The first was to compare classifications made by SKICAT on photographic images to those classified by astronomers on CCD images. The same images may appear in more than one plate. Another measure of accuracy was to measure the number of times the classification of these multiple type images conflicted.

2.2.2 Health Kefir

Health-KEFIR (Key Findings Reporter) analyzes healthcare observations in an effort to control costs and improve quality.[Matheus, C. J., G. Piatsky-Shapiro, and D. McNeil, 1996] KEFIR looks for deviations in the database from previous measurements or from the accepted norm for a particular attribute. These deviations indicate interesting patterns in the data since they differ from what is expected. Deviations are ranked according to a measure of interestingness. In the domain of health-care analysis, interestingness is measured by the impact of a prescribed response. This is the benefit, such as financial gain, obtained by a set course of action. A number of user defined key deviations are explained relative to their related deviations, and in some instances a recommended action is offered. In healthcare key deviations are those that indicate problems that can be remedied.

The search space in Health-KEFIR is divided into three components. The population group is the first component. This group is defined by characteristics such as employee group, geographical region, employee status, etc. The second component is the medical study area. This consists of the medical problem along with treatments associated with this problem. A set of relevant measures comprise the third area. These measures are independent of the population group, but related to the study area. For example, the medical admissions study area can have measurements describing the length of stay. Deviations are obtained for intersections of populations and study areas (defined as a sector). The top sector is the sector containing all instances of that category. The computed deviations define subsectors, which in turn have their deviations computed. This recursive process continues until the search space is exhausted or the sectors become

too small for further splitting. Deviations, along with sector information, and related deviations are stored in a structure called a finding. Findings are ordered according to the interestingness measure and the top N, where N is user defined, are denoted as key findings. These are reported by KEFIR along with any recommendations. Recommendations are coded into KEFIR as production rules. These recommendations are obtained from domain experts. Reports are presented in HTML and based on a template that mimics natural language. They include all findings and relevant and related deviations. The findings can be shown in graph form.

KEFIR runs on a Sun SPARCstation 20 with an Informix DBMS. KEFIR took the equivalent of two *F*ull *T*ime-*E*mployee man (*FTE*) years to complete. An additional 6 month FTE was required to build KEFIR's knowledge base.

KEFIR was field tested. No formal data was presented as to its effectiveness. Statements were made indicating that user feedback was such that KEFIR reported the same deviations as those reported by health care analysts, along with some findings that were not determined by these analysts. KEFIR is being modified to work in other domains such as market analysis.

2.2.3 JARtool

JARtool, Jet Propulsion Laboratory Adaptive Recognition tool, was used to detect small volcanoes on the planet Venus[Burl, M. C, et. al. 1998]. Data came from the Magellan spacecraft, which over a period of five years, mapped the surface of Venus. The data set used was a set of 30,000 images of 1,000 X 1,000 pixels each. Large volcanoes were mapped manually. It was found that humans got tired after cataloging

50 – 100 images over a period of a few days. Because of this, it was estimated that small volcano cataloging would take 10 – 20 man years. There was a need for an automated cataloging system.

Feature selection was a major problem in the classification process. It was hard to distinguish volcanoes from the background. There were no clear set of feature vectors. JARtool used principal component analysis to reduce the dimensionality of the problem. The reduced feature vector was used to train the classifier. The system developed a classifier using a subset of approximately 30 – 40 images on which planetary geologists labeled small volcanoes. JARtool was able to match the scientists' performance with regard to classifying certain types of volcanoes (high probability volcanoes vs. those scientists are not sure of). It was more successful on homogeneous image sets, (sets of images that are in close proximity to each other), than on heterogeneous image sets, (images taken from random areas of the planet).

JARtool is an example of a train by example approach. This approach lends itself to data mining in the image processing range since it allows for the discovery of patterns that answer the query, "Find me stuff that looks like this".

Data mining in the scientific domain offers different problems than those of the business domain. Domain knowledge is used to a greater extent since it is usually well documented and available in papers, texts, etc. On the down side, in scientific domains, there is a need for higher levels of accuracy than in business. There are issues of scalability since there can be large amounts of sensor data generated. This might necessitate the use of highly parallel supercomputers in data analysis. As opposed to the business domain where the search is for frequently occurring events, in the scientific

domain we may have to be able to detect low probability classes such as those found in SKICAT. What we want may to find are rare occurrences of some phenomena.

2.2.4 ILP applications

GOLEM and CLAUDIEN were two ILP applications that was used to look at biological classification of river quality[Dzeroski, S., 1996]. The presence of different family classifications of benthic macro-invertebrates in rivers, is indicative of the extent of pollution found in these rivers. Different families of these invertebrates are sensitive to different pollutants including biodegradable pollutants such as sewage, and toxic pollutants such as heavy metals and pesticides. In this study, CLAUDIEN and GOLEM learned rules that would classify water quality based on the presence of these types of benthic macro-invertebrates. Training examples were gotten from a database of 292 field samples collected from British rivers and classified by an expert. An expert in the field evaluated the generated rules. GOLEM found 35 rules, 25 of which were found to be acceptable by the expert. CLAUDIEN found 79 rules two thirds of which were found to be of acceptable by the expert.

GOLEM has also been used to predict the secondary structure of proteins. The secondary structure of proteins is where in three dimensional space you would find a particular amino acid.. The sequence of amino acids is the primary structure of a protein. The problem was simplified to finding if an amino acid resided an alpha helix protein structure. GOLEM was able to induce rules that were 78% accurate on the training set and 81% accurate on the test set.

2.2.5 Market Analysis

Retailers, to predict the buying patterns of their customers, use data mining applications in this domain. Wal-Mart created one of the largest databases in the world using transaction/observation[1] data. Customer databases are analyzed and searched for customer buying patterns and preferences. These applications use techniques such as segmentation, interactive querying, and predictive modeling. The results allow retailers to select customers in a more precise and targeted manner. Rules generated by these systems help market analysts make better marketing decisions.

Two examples of marketing programs are Coverstory and Spotlight [Matheus, C. J., G. Piatsky-Shapiro, and D. McNeil, 1996]. They analyzed supermarket sales data. Patterns were found relating changes in product volume and share. These answered questions of the type, "Where do we ship those cans of beans?"

Opportunity Explorer was another application that found business relationships for sales representatives of consumer packaged goods with their retailers. The results were presented as advantageous protocols for retailers. They were advised with regard to stocking of additional products or the running of special promotions to enhance sales.

Market basket analysis is a significant area where data mining has taken place. Market basket analysis uses point of sale information to describe relationships between retail stock movement. The information is used to determine shelf space allocation, store layout, product location and promotions. Most data mining algorithms that do market basket analysis use some type of dependency modeling. Some examples of data mining

---

[1] On transactions and observations: the data mining community defines a transaction as a measurement, while the computer science community defines such a measurement to be an observation. In either case,

tools in this domain are IBM Data Miner, Lucent Technology's Niche Works, and a future version of KEFIR.

2.2.6 Investment Forecasting

Many investment companies use data mining to manage stock portfolios. The methods employed by these companies tend to be proprietary and not usually described in the literature. Investment companies are competitive and don't tend to publish their methods. Most packages in this category use regression, neural networks.

Fidelity Stock Selector used a neural network to select investments. Results are presented to a fund manager who makes final decisions. The system did well up to a point. It is uncertain whether the system was at fault or the human, and fidelity is not telling who is responsible.

Other examples of data mining in the financial markets include LBS Capital Management which manages funds worth $600 million. It uses a system of expert systems, neural networks and genetic algorithms. Since its inception in 1993, it has outperformed the stock market. Carlberg & Associates use a neural network for predicting Standard and Poor's 500 Index. It used interest rates, oil prices, earnings, dividends, and the dollar index as inputs and was able to explain 96% of the variation in S&P from 1986 to 1995. [Brachman, R. J., et. al. 1994]

---

this is simply a row in a table of a database. For the purpose of consistency, we will adopt the term observation for this thesis.

2.2.7 Other Applications

PRISM and FALCON are programs that detect credit card fraud. FALCON uses a neural network to detect suspicious credit card observations. Banks use these programs.

FAIS [Brachman, R. J., et. al. 1994] is used by the U.S. Treasury Financial Crimes Enforcement Network to detect money laundering. Information to this system comes from government forms. FAIS is a combination of off the shelf components with some customization. One of the problems it had to contend with was that much of the data it used was in the form of handwritten notes.

AT&T detects calling fraud by using visualization techniques to display call activity in a way that picks up on unusual patterns [Cox, K. C., S. G. Eick, G. J. Wills, and R. J. Brachman, 1997]. Clonedetector by GTE finds cellular phone clones by using deviation detection to see how calling patterns differ from a customer's calling profile.

In addition, the IRS has developed a pilot system to be used for selecting returns for audit. Problem detection and prediction in Boeing 737's are analyzed by CASSIOPE. MERGE PURGE was used to eliminate duplicate claims in the Washington State Welfare System.

Even the sports world uses data mining. IBM's ADVANCED SCOUT analyzes data from NBA games. It helps coaches to organize and interpret this data. SCOUT finds patterns of play and has been used successfully by the Seattle Supersonics in the 1996 NBA finals. As can be seen there are a wide breadth of applications with new applications being brought into the market each day.

2.3 How to compare KDD systems

With the wide variety of KDD systems and the equally diverse number of domain applications, it is difficult to compare systems. Nonetheless, there are some guidelines that can be used when deciding between systems.

1. Who is the user of the KDD systems?
2. What types of tasks are supported by the KDD systems?
3. What tools are associated with each supported task?
4. Are tools integrated with each other? Are there various steps in the process? Do these address user needs?
5. In what manner does the system allow for incorporation of background knowledge?
6. How is discovered information presented to the end user?
7. Are the results of the KDD system able to be used in some applicable way by a user other then the data engineer, i.e. some businessperson looking for a market trend, etc.?

These are questions that are relevant to all KDD systems.

2.4 Ethical Issues

It is important to address the ethical issues raised by data mining. Invasion of privacy issues are at stake. Government and business databases contain a lot of personal information. These problems have been addressed more directly in Europe. The Organization for Economic Cooperation and Development (OECD) addressed these issues for the European Union nations. OECD has said that data analysis on living individuals should not be done without their consent. There are movements in this

country for legislation that follows in the same vein. Much of the argument against regulation contends that, especially in the medical domain, getting individual consent might not be feasible, resulting in the loss of potential knowledge advances. On the other hand, most data mining deals with discovering patterns with regard to groups and not individuals. As long as the results don't point to any one individual, regulations are not necessary. Alternately, this can be a problem in small databases where combinations of group patterns can point to individuals. There are definite Orwellian overtones to the KDD process. As individuals, it is important to be aware of the potential abuses of this process and to temper that with its potential benefits.

Chapter 3

Algorithms That Generate Association Rules

Association rules are implications of the form $X \Rightarrow Y$. The thick arrow is used to differentiate the probabilistically defined implications of these rules from logical implication. Constraints on which operators, the numbers of variables that can be placed in X or Y, and the values for these variables have changed as algorithms became better able to generate more complex rules. Initially, association rules were defined in the following way [Agrawal, R., T. Imielinsk, and A. Swami, 1993]:

Let $I = I_1, I_2, \ldots I_m$ a set of binary attributes.

Given X is some subset of I and T a database which is a set of observations t.

Each observation in $t \in T$ can be thought of as a binary vector of 0's and 1's.

A tuple t is said to satisfy X if for each value $t(k)$ in tuple t, $t(k) = 1$ for all attributes I_k in X.

An association rule is an implication $X \Rightarrow I_j$ where I_j is an item in I that is not in X.

Note, here the constraints on the antecedent of the association rule are such that the only Boolean operation allowed to connect attributes is conjunction. Attributes in the

antecedent are only allowed a value of one, hence no negations are allowed. The consequent can contain only one attribute. Also, attributes of values of the consequent are only allowed a value of one. I shall define rules of this genre as Type I rules.

In later algorithms by the above authors, the constraints on the antecedent were relaxed to allow for more than one attribute. These algorithms defined an association rule as an implication $X \Rightarrow Y$ where X and Y are subsets of I and $X \cap Y = \phi$. Constraints still existed on the attribute values. Attributes could not be negated and could only be connected with conjunction. Define these rules as Type II rules.

Constraints on association rules were further relaxed and these types of rules were redefined as dependency rules.(Silverstein, C., Sergey Brin, and Rajeev Motwani, 1998) Dependency rules are defined as an implication $X \Rightarrow Y$ where X and Y are subsets of I, the attributes of set X are dependent to the attributes of set Y, and $X \cap Y = \phi$. Instantiations of X and Y such that any attribute $I_x \in X$ and any attribute $I_y \in Y$ can have values of one or zero, denoting presence or absence in the observation. The constraint whereby all attributes can only be connected by conjunction still holds. Define these as Type III rules.

Orchard's algorithm, Boolean Analyzer (Orchard, Robert A., 1975) - later extended by Domanski (Domanski, B., 1996), is also based on dependence. In the hill-climbing phase of the algorithm, complex dependency rules are determined. These rules are similar to those defined by Silverstein, Brin, and Motwani, but also allow for attributes to be connected with the disjunction in addition to conjunction. We define these as Type IV rules.

3.1 AIS

AIS (Agrawal, R., T. Imielinsk, and A. Swami, 1993) is an algorithm that finds association rules of Type I. Rules are generated within boundaries set by the confidence of a rule and by its support. The parameters of confidence and support denote the strength and statistical significance of a rule respectively.

Given a rule $X \Rightarrow Y$

Let N be the number of observations in T.

Let S_1 be the number of observations in T that satisfy X.

Let S_2 be the number of observations in T that satisfy the vector $X \cup Y$ where

X∩Y is null

Then the confidence c of a rule is S_2 / S_1

The support s of a rule is S_2 / N

Note if we divide the support of a rule, S_2 / N, by the support of the antecedent, S_1 / N, we can also obtain its confidence.

The goal of AIS is to find all combinations of attributes that are above a threshold value for support, defined as *minsupport*. Combinations of attributes whose support is above minsupport are called large *itemsets*. From these large itemsets, AIS generates rules that are above a threshold value for confidence, defined as minconf.

Once large itemsets are identified, rules can be generated. Rules are generated by looking at various combinations of attributes for the consequent and antecedent of the rule $X \Rightarrow Y$ such that X and Y are subsets of the large itemset and $X \cap Y = \phi$. If the ratio r of the support of $X \cup Y$ divided by the support of X (the actual confidence of the rule

$X \Rightarrow Y$) is greater than minconf then that rule holds with at least a confidence of minconf.

The major portion of the algorithm finds large itemsets. To do this the algorithm makes many passes over the data. In a particular pass the algorithm creates candidate itemsets from observations in the data and from itemsets contained in a frontier set, created from previous passes. The frontier set consists of itemsets that have been extended during a pass. Observations are compared to the frontier itemsets. If extending a frontier set can create an itemset, it is added to the list of candidate itemsets. For example, if a frontier set contains items I_1 $I_2 I_3$ and the observation codes for $I_1 I_2 I_3 I_7 I_9$ then candidate itemsets would be I_1 I_2 I_3 I_7, I_1 I_2 I_3 I_9, I_1 I_2 I_3 I_7 I_9. Candidate itemsets are associated with counters that keep track of their support. As the algorithm passes over the data an estimation of support is calculated. If the estimated support is larger than minsupport the itemset can be extended further. Those that are expected to be small are not extended. So candidate itemset I_1 I_2 I_3 I_7 would not be extended to I_1 I_2 I_3 I_7 I_9 if I_1 I_2 I_3 I_7 were expected to be small. The frontier set contains those itemsets that were expected to be small but in actuality were large. These are placed in the frontier set since their extensions were not considered in previous passes. The algorithm ends when the frontier set is empty. Large itemsets are maintained and used for rule generation.

3.2 Apriori, AprioriTid, and AprioriHybrid

Apriori, AprioriTid, and AprioriHybrid algorithms improved upon the above algorithm[Agrawal, R., H. et. al., 1996]. The problem with AIS was that it would generate and count too many small itemsets. In Apriori, AprioriTid, and AprioriHybrid,

the observations in the database were not used for candidate itemset calculation. Rather, the large itemsets found in a previous pass were used to generate new candidate itemsets. AprioriTid used an encoding that captured information as to which observations corresponded to that itemset.

In Apriori, the first pass through the data finds the support for itemsets consisting of one attribute. Those itemsets that are above minsupport, form the set of large itemsets. The $k-1$ sized candidate itemsets are used in the next pass to find large itemsets for the *kth* pass. This is done by forming the union of two $k-1$ itemsets such that these itemsets share $k-2$ items. Candidate itemsets are pruned from this set if any $k-1$ subset of the k candidate itemset in question was not a previously found to be a large $k-1$ itemset. Once the new candidate sets for pass k are formed they are checked against the data to determine their support, and in consequence which are above minsupport. These large itemsets form the set of large itemsets for pass k. This is repeated until the set of large itemsets for a pass becomes null.

AprioriTid is similar to Apriori. Candidate itemset creation is the same but instead of determining support from the database, support is calculated from a new set, C'_k which contains a observation's ID number associated with the set of all large k itemsets that are supported by that observation. Observations that don't match large itemsets are not contained in this set.

For small sized itemsets, C'_k has the potential of being larger than the data itself. When itemsets become large, C'_k will be much smaller. AprioriTid performed better than Apriori when C'_k was able to fit in memory. AprioriHybrid uses the Apriori algorithm

until C'_k fits into memory and then switches to the AprioriTid algorithm. AprioriHybrid was shown to perform better than Apriori and AprioriTid.

3.3 Dependency Rule Algorithm

It was argued [Silverstein, C., Sergey Brin, and Rajeev Motwani, 1998] that the support/confidence measured association rules that Agrawal et. al. described could not discover rules that described negative dependencies. For example, AIS, and the Apriori algorithms would not discover rules such as "People who buy diapers do not buy Geritol." Therefore association rules were not adequate for domains that need to mine these types of rules.

Silverstein et al differentiate between variable dependency and event dependency. A dataset will have a set of variables or attributes. An event is an instantiation of these variables. Each variable can have a value of zero or one. A value of zero indicates the absence of the variable, and a value of one indicates its presence in the event. This algorithm tests for dependency between variables, and then looks at instantiations of the events that are defined by the variables to see which events define strong rules. For example, given that I_1 and I_2 are two variables from the set of attributes $I = I_1, I_2, \ldots I_m$, then the events associated with these two variables are $I_1 I_2$, $I_1 I_2'$, $I_1' I_2$, $I_1' I_2'$.

The algorithm tries to find itemsets of variables that are minimally dependent. A minimally dependent itemset is a dependent itemset whose subsets are not dependent. The significance of a minimally dependent itemset stems from the property of variable dependency as being upward-closed in the lattice of all itemsets. If a set I from the set of all itemsets is upward-closed in property P, then each superset of I in this set of itemsets

contains property *P*. A set *I* is downward-closed for property *P* if each subset of *I* has property *P*. All the itemsets that are minimally dependent form a border in the itemset lattice, which defines a boundary for the space of all dependent itemsets.

Dependency was measured using a Chi-Square statistic. A contingency table was created where each cell in the table is labeled with the Cartesian product of the variables and their negations. The values in the cells were the counts of the observations in the database that matched the cell label. A sample contingency table for variables I_1 and I_7 is as shown.

	I_7	I_7'	row-sum
I_1	4	2	6
I_1'	6	8	14
col-sum	11	7	38

Table 1 - Contingency Table For Event Dependencies

The above table indicates that there are 4 observations with I_1 and I_7 together, 6 observations of observations where I_1 was present and I_7 was not, etc. The Chi-Square statistic was given as:

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E(r))^2}{E(r)}$$

Where O(r) stands for the observed value for a variable and E(r) is the expected value. For the contingency table above the Chi-Square value would be:

$$\frac{(4-6*11/38)^2}{6*11/38}+\frac{(2-6*7/38)^2}{6*7/38}+\frac{(6-14*11/38)^2}{14*11/38}+\frac{(8-14*7/38)^2}{14*7/38}$$

This equals 16.002. There is one degree of freedom since values are Boolean. A Chi-Square value of 3.84 denotes a p value of 0.05. This that the above variables are independent with at the 95% confidence level.

To find which cells of the contingency table were most dependent, a measure of interestingness was defined. This measure for the two events denoted by a cell *r* in the contingency table was defined as follows:

$$I(r)=\frac{p(xy)}{p(x)p(y)}$$

Maximizing | *I(r)* – 1| finds the cell in the contingency table that most contributes to the Chi-Square value of the contingency table. Values of interestingness although comparable within a contingency table, are not comparable between contingency tables.

Given a contingency table for a set of items S. The contingency table support (CT-support) is the value *p*, where *p* is the percentage of cells in the contingency table having a support value of *s*. Significant itemsets are those that are CT-supported and minimally dependent.

The algorithm for finding the dependency border, first defines a minimum Chi-Square significance level, a minimum support, and a minimum CT-support. Each item in the database is counted to find their support. Those items whose support in the dataset is larger than minimum support are paired with each other. Item pairs are added to a candidate set for dependency analysis. Each itemset in the candidate has their contingency table created. Those itemsets whose CT-support is greater than the defined

minimum are evaluated for their Chi-Square values. Those that have significant Chi-Square values are placed in a set of significant itemsets. Those that are not are added to the set of insignificant itemsets. The significant and insignificant sets were initially set to null. All itemsets in the candidate set are tested in this manner. Once this is finished a new candidate set is formed from the insignificant set. The new sets in the candidate set have cardinality S such that each |S-1| subset is in the insignificant set. The algorithm ends when the candidate set is empty. The result in the significant set is a list of all minimally dependent itemsets.

Chapter 4

Boolean Analyzer

Boolean Analyzer is an algorithm developed by Dr. Robert Orchard (Orchard, Robert A., 1975) and expanded upon by Dr. Bernard Domanski (Domanski, B., 1996). Boolean Analyzer (BA) takes data, Booleanizes the data, then calculates a dependency measure indicating the strength of the events in a dependency relationship. Positive and negative values of this measure indicate if the relationship is direct or indirect, respectively.

4.1 The Algorithm

Given a database T, define an activity matrix as a set of variables with their corresponding values, possibly over time.

The values are obtained from a dataset T where each observation $t \in T$ is one instantiation of the set $X = \{X_1, ..., X_n\}$ where $X_1, ..., X_n$ represent the variable attributes in T. Values for $X_1, ..., X_n$ can be numeric, categorical or Boolean. There are no data type restrictions.

Define a threshold set $H = \{ h_1, ..., h_n\}$ such that each h_i represents a Boolean boundary on the variables of X.

We Booleanize the matrix of n variables, $X_1, ..., X_n$ as follows.

$$X_i = \begin{cases} 0 & \text{if } Value(X_i) < h_i \\ 1 & \text{if } Value(X_i) \geq h_i \end{cases}$$

The threshold set *H* is determined using the domain knowledge of an expert. Once each value is changed to its Boolean equivalent, we can form a Boolean activity matrix.

THE BOOLEAN ACTIVITY MATRIX

	X_1	X_2	X_3	X_4	X_5	X_6
observation 1	0	1	1	1	0	1
observation 2	1	0	1	1	1	1
observation 3	1	1	1	0	1	0
•			•			
•			•			
•			•			

Figure 4 - The Boolean Activity Matrix

Rows represent the state of each variable during a single observation. Each row of this Boolean activity matrix can be in any of 2^n possible states. Each of these states defines an equivalence class on the set of observations in the activity matrix. Based on these equivalence classes we can create from the Boolean activity matrix another matrix called the State Occurrence matrix.

The 2^n possible states can be organized into a State Occurrence matrix (SO matrix) by partitioning the variables X_1 to X_n into two disjoint groups, X_1 to X_j and X_{j+1} to X_n

Label each row with j-tuples, $<X_1, ..., X_j>$ such that each j-tuple forms a distinct class. The total number of row classes are thus 2^j.

Label each column with (n-j)-tuples, $<X_{j+1}, ..., X_n>$ such that each (n-j)-tuple forms a distinct class. The total number of column classes are thus 2^{n-j}.

Let X_i represent an unprimed (value 1) variable.

Let X_i' represent a primed (value 0) variable

Then each row or column class can be labeled using this notation.

With no loss of generality, consider the following example

Assume we have dataset *T* of sample size, $| t | = n = 100$. Then let :

$X_1 = 1$ CPU Active	$X_1 = 0$ CPU Idle
$X_2 = 1$ System state	$X_2 = 0$ User State
$X_3 = 1$ Online workload	$X_3 = 0$ Batch workload
$X_4 = 1$ Tape activity significant	$X_4 = 0$ Tape not significant
$X_5 = 1$ Disk activity significant	$X_5 = 0$ Disk not significant
$X_6 = 1$ High Paging	$X_6 = 0$ Expected Paging

Further assume that the State Occurrence Matrix in *Table 2* was calculated from the activity matrix:

	$X_4X_5X_6$	$X_4X_5X_6'$	$X_4X_5'X_6$	$X_4X_5'X_6'$	$X_4'X_5X_6$	$X_4'X_5X_6'$	$X_4'X_5'X_6$	$X_4'X_5'X_6'$
$X_1X_2X_3$	0	1	0	0	4	2	1	2
$X_1X_2X_3'$	2	0	1	0	3	0	2	2
$X_1X_2'X_3$	1	0	0	1	5	1	0	2
$X_1X_2'X_3'$	3	0	0	0	3	5	2	2
$X_1'X_2X_3$	1	0	1	1	3	1	2	1
$X_1'X_2X_3'$	0	2	0	1	0	3	5	4
$X_1'X_2'X_3$	0	0	0	1	5	3	4	2
$X_1'X_2'X_3'$	0	1	2	1	0	5	4	2

Table 2 - The State Occurrence Matrix

Note, the intersection between a row and a column forms an equivalence class on the activity matrix. Matrix values indicate the support for that state in the activity matrix as defined by the Boolean threshold set *H*. For example, given state $X_1'X_2'X_3X_4'X_5X_6$, this implies that

- the CPU is idle
- the system is in user state
- running an online workload
- with low tape activity, and high paging activity

This state occurs 5 times in the activity matrix. This corresponds to the event 001011.

Just by inspecting the State Occurrence matrix we can observe some interesting relationships.

1. The CPU was active in system state processing online workload 10 times. Of these, 5 times there was a high paging rate.
2. The next-to-rightmost column, (X_4'X_5'X_6) shows that high paging activity, no disk or tape activity occurred 20 times. Of these, the CPU was idle in 15 of them.
3. To find out the support for CPU activity and tape activity (X_1X_4) we look at the upper left quadrant of the State Occurrence matrix and sum the values in each cell that is labeled with X_1 and X_4. This is shown in *Table 3*.

	$X_4X_5X_6$	$X_4X_5X_6$'	X_4X_5'X_6	X_4X_5'X_6'	X_4'X_5X_6	X_4'X_5X_6'	X_4'X_5'X_6	X_4'X_5'X_6'
$X_1X_2X_3$	0	1	0	0	4	2	1	2
$X_1X_2X_3$'	2	0	1	0	3	0	2	2
X_1X_2'X_3	1	0	0	1	5	1	0	2
X_1X_2'X_3'	3	0	0	0	3	5	2	2
X_1'X_2X_3	1	0	1	1	3	1	2	1
X_1'X_2X_3'	0	2	0	1	0	3	5	4
X_1'X_2'X_3	0	0	0	1	5	3	4	2
X_1'X_2'X_3'	0	1	2	1	0	5	4	2

Table 3 - State Occurrence Matrix Showing $X_1 : X_4$

4.1.1 Relationships in Matrix Form

We can think of dependency relationships between two variables as being in one of four possible event states, X_i and X_j are both high, X_i is high when X_j is low, X_i is low when X_j is high, and both X_i and X_j are low. If we define ":" to stand for a the dependency relationship of an event, then we have:

$$X_i : X_j$$

$$X_i : X_j'$$

$$X_i' : X_j$$

$$X_i' : X_j'$$

The above can be represented by a 2 x 2 contingency table.

Using $X_1 : X_4$ as an example we have:

	X_4	X_4'
X_1	9	36
X_1'	11	44

We define these types of contingency tables as a relationship matrix. Each cell is computed by finding their support from the State Occurrence matrix

We can represent more complex dependency relationships as well.

X_1X_3:X_4X_6 (CPU active processing the online workload vs. tape and high paging activity).

The entries in the State Occurrence matrix corresponding to this relationship are:

	$X_4X_5X_6$	$X_4X_5X_6'$	$X_4X_5'X_6$	$X_4X_5'X_6'$	$X_4'X_5X_6$	$X_4'X_5X_6'$	$X_4'X_5'X_6$	$X_4'X_5'X_6$
$X_1X_2X_3$	0	1	0	0	4	2	1	2
$X_1X_2X_3'$	2	0	1	0	3	0	2	2
$X_1X_2'X_3$	1	0	0	1	5	1	0	2
$X_1X_2'X_3'$	3	0	0	0	3	5	2	2
$X_1'X_2X_3$	1	0	1	1	3	1	2	1
$X_1'X_2X_3'$	0	2	0	1	0	3	5	4
$X_1'X_2'X_3$	0	0	0	1	5	3	4	2
$X_1'X_2'X_3'$	0	1	2	1	0	5	4	2

Table 4 - State Occurrence Matrix showing X_1X_3:X_4X_6

Looking at the relationship matrix we find:

	X_4X_6	$(X_4X_6)'$
X_1X_3	1	19
$(X_1X_3)'$	10	70

Values in each cell are calculated as in the previous example. The value in the upper left cell is the result of summing all values found in the State Occurrence matrix that correspond to entries where the rows contain both X_1X_3 and the columns contain X_4X_6. The lower left cell is calculated by summing values in the State Occurrence matrix for rows that contain either X_1' or X_3' along with X_4 and X_6.

4.1.2 Meaningful Dependency Relationships

The number of ways that we can partition the rows into two disjoint sets is equal to the recurrence relation S(n,k), which is the number of partitions of a set S of cardinality n into k partitions where:

$S(n,1) = 1 \quad S(n,n) = 1$

$S(n,k) = S(n-1,k-1) + k(S(n-1), k)\ 2 \leq k \leq n-1$

We are looking for dependencies between sets of column variables to sets of row variables. The number of dependency relations is S(n,k) + 1 for the rows, multiplied by S(n,k) + 1 for the columns. We add one to the number of 2-partitions since the 1-partition containing all the rows or all the columns is a valid consideration. For our example this equals:

$(S(8,2) + 1)\ (S(8,2) + 1) = 128 * 128 = 16{,}384$ possible dependency relationships

.

Define a PMSR as a possibly meaningful state dependency relationship. Our problem is to determine which of these 16,384 possible state dependency relationships are meaningful. Let's say we are given a 2 x 2 matrix where X and X' stand for a group of rows and their complement respectively, and Y and Y' stand for a group of columns and their complement respectively. Assume that X and Y are independent. Then let the probability of event X be p(X).

Therefore the probability of X' is 1 - p(X).

Let the probability of event Y be p(Y).

Therefore the probability of Y' is 1 - p(Y).

Since X and Y are independent then the number of times X occurs with respect to Y is $p(X \wedge Y) = p(X)p(Y)$. We also have:

$p(X \wedge Y') = p(X)\,(1 - p(Y))$

$p(X' \wedge Y) = (1 - p(X))\,p(Y)$

$p(X' \wedge Y') = (1 - p(X))\,(1 - p(Y))$

Representing the above in a contingency table

	Y	Y'
X	$p(X)p(Y)$	$p(X)\,(1 - p(Y))$
X'	$(1 - p(X))\,p(Y)$	$(1 - p(X))\,(1 - p(Y))$

We see that the column ratio of the number of times X occurs with respect to Y to the number of times X' occurs with respect to Y is equal to the ratio of the number of times X occurs with respect to Y' to the number of times X' occurs with Y'.

$$\frac{p(X)p(Y)}{(1 - p(X))\,p(Y)} = \frac{p(X)\,(1 - p(Y))}{(1 - p(X))\,(1 - p(Y))}$$

We find the same if we look at the row ratio.

Let a, b, c and d represent the values in each quadrant of the above contingency table. The values for a, b, c, and d are the occurrences of the joint events, which can be read from the State Occurrence Matrix.

	Y	Y'
X	a	b
X'	c	d

Then we have:

$$\frac{a}{c} = \frac{b}{d}$$

Therefore, when a, b, c and d are independent we find

$$\mathbf{0 = ad - bc}$$

Define events that are not independent as being dependent. Then using the above equation we can write **m = ad – bc** where m is a measure on the type and extent of the dependency of the event X to Y. Large negative values show that X has a strong inverse dependency relationship to Y. Large positive values indicate a strong direct dependency relationship. Values close to or equal to zero indicate that the variables are independent and not related. For example, the measure of $X_1 : X_4$ is m = 0, $X_1X_3 : X_4X_6$ m = -120.

4.1.3 The State Linkage Matrix

Generalizing from the State Occurrence matrix, the relationship matrix defined by a single row i and a single column j would be:

	column j	(column j)'
row i	a_{ij}	$r_i - a_{ij}$
(row i)'	$c_j - a_{ij}$	$N - r_i - c_j + a_{ij}$

where:

a_{ij} = entry at row I, column j of the SO matrix

N = total sample size of the activity matrix

r_i = sum of the entries in row i of the SO matrix

c_j = sum of the entries in column j of the SO matrix

Taking the measure and combining terms we get:

$m_{ij} = a_{ij} N - r_i c_j$

We can use the above measure to form the state linkage matrix whose entries show the measure value for the simple dependency relationship of a single row with a single column.

	$X_4X_5X_6$	$X_4X_5X_6'$	$X_4X_5'X_6$	$X_4X_5'X_6'$	$X_4'X_5X_6$	$X_4'X_5X_6'$	$X_4'X_5'X_6$	$X_4'X_5'X_6'$
$X_1X_2X_3$	-70	60	-40	-50	170	0	-100	30
$X_1X_2X_3'$	130	-40	60	-50	70	-200	0	30
$X_1X_2'X_3$	30	-40	-40	50	270	-100	-200	30
$X_1X_2'X_3'$	195	-60	-60	-75	-45	200	-100	-55
$X_1'X_2X_3$	30	-40	60	50	70	-100	0	-70
$X_1'X_2X_3'$	-105	140	-60	25	-345	0	200	145
$X_1'X_2'X_3$	-105	-60	-60	25	155	0	100	-55
$X_1'X_2'X_3'$	-105	40	140	25	-345	200	100	-55

Table 5 - The State Linkage Matrix

From the state linkage matrix (SL matrix) we can determine the measures of more complex relationships. To calculate the measure for a relationship X vs. Y, where X represents a set of rows and Y a set of columns, sum the measures of the entries that correspond to those rows and columns combined

Example X_1X_3 vs. X_4X_6

	$X_4X_5X_6$	$X_4X_5X_6'$	$X_4X_5'X_6$	$X_4X_5'X_6'$	$X_4'X_5X_6$	$X_4'X_5X_6'$	$X_4'X_5'X_6$	$X_4'X_5'X_6'$
$X_1X_2X_3$	- 70	60	- 40	- 50	170	0	- 100	30
$X_1X_2X_3'$	130	- 40	60	- 50	70	- 200	0	30
$X_1X_2'X_3$	30	- 40	- 40	50	270	- 100	- 200	30
$X_1X_2'X_3'$	195	- 60	-60	- 75	- 45	200	-100	- 55
$X_1'X_2X_3$	30	- 40	60	50	70	- 100	0	- 70
$X_1'X_2X_3'$	- 105	140	- 60	25	- 345	0	200	145
$X_1'X_2'X_3$	-105	-60	- 60	25	155	0	100	- 55
$X_1'X_2'X_3'$	-105	40	140	25	- 345	200	100	- 5

Table 6 - State Linkage Matrix Showing m = -70 + 30 -40 -40 = -120

4.1.4 The meaning of the State Linkage Matrix

By looking at the State Linkage matrix we can see the dominant values for relationships defined by all row variables vs. all column variables. Those relationships with high negative or high positive values are dominant.

We can use the State Linkage matrix to find the highest measure relationship for any one variable such as disk activity(X_5) and the row states. To do this we calculate m(row: X_5)

$m(X_1X_2X_3 : X_5) =$ -70 + 60 + 170 + 0 = 160

$m(X_1X_2X_3' : X_5) =$ 130 - 40 + 70 – 200 = -40

$m(X_1X_2'X_3 : X_5) =$ 30 - 40 + 270 - 100 = 160

$m(X_1X_2'X_3' : X_5) =$ 195 - 60 - 45 + 200 = 290

$m(X_1'X_2X_3 : X_5) =$ 30 - 40 +70 - 100 = -40

$m(X_1'X_2X_3' : X_5) =$ -105 + 140 - 345 + 0 = -310

$m(X_1'X_2'X_3 : X_5) =$ -105 - 60 + 155 + 0 = -10

$m(X_1'X_2'X_3' : X_5) =$ -105 + 40 - 345 + 200 = -210

From the above we find that $X_1X_2'X_3' : X_5$ is the system state that contributes most to disk activity.

To find the relationship for the set of system states that, in general contribute to disk activity we merely combine with disjunction the states that have a positive measure to form a composite system state. $X_1X_2X_3$ v $X_1X_2'X_3$ v $X_1X_2'X_3'$ which reduces to $(X_1 (X_2' \text{ v } X_3))$. The measure of the more complex dependency

relationship $(X_1 (X_2' \vee X_3)) : X_5$ is 610. This is found by summing the measures of each component dependency relationship.

$m(X_1X_2X_3 : X_5) + m(X_1X_2'X_3 : X_5) + m(X_1X_2'X_3' : X_5) = 160 + 160 + 290 = 610.$

Taking a look at the values for this relationship from our original State Occurrence matrix we get:

	X_5	X_5'
$X_1 (X_2' \vee X_3)$	25	10
$(X_1 (X_2' \vee X_3))'$	29	36

This confirms our measure of 610. Notice that 5 out of 7 times the CPU is active and we are not in system state processing batch, there is disk activity. This seems to verify our previous conclusion.

4.1.5 Finding High Measure Dependency Relationships

We found from the previous section that by combining the variables in rows that contained positive measures for variable X_5 with disjunction, we were able to find a complex dependency relationship with a higher measure than the semisimple dependency relationships for X_5. We can use the same methodology to formulate a hill-climbing technique that can find complex relationships for the entire system.

Define a significant row or column in the State Linkage Matrix as a row or column whose measure is positive. From the previous section we have the rows denoted by $X_1X_2X_3$, $X_1X_2'X_3$, and $X_1X_2'X_3'$ as significant rows. If we sum, by column, the measures for each significant row we get *Table 7.*

	$X_1X_2X_3$	$X_1X_2X_3'$	$X_1X_2'X_3$	$X_1X_2'X_3'$	$X_1'X_2X_3$	$X_1'X_2X_3'$	$X_1'X_2'X_3$	$X_1'X_2'X_3'$	
$X_1X_2X_3$	-70	60	-40	-50	170	0	-100	30	
$X_1X_2X_3'$	130	-40	60	-50	70	-200	0	30	
$X_1X_2'X_3$	30	-40	-40	50	270	-100	-200	30	
$X_1X_2'X_3'$	195	-60	-60	-75	-45	200	-100	-55	
$X_1'X_2X_3$	30	-40	60	50	70	-100	0	-70	
$X_1'X_2X_3'$	-105	140	-60	25	-345	0	200	145	
$X_1'X_2'X_3$	-105	-60	-60	25	155	0	100	-55	
$X_1'X_2'X_3'$	-105	40	140	25	-345	200	100	-5	
									PMSR
Sum of pmsr's of significant rows		-40	-140	-75			-400		655

Table 7 - Hill Climb Showing Significant Rows

The sum of the positive measures for each column gives us the measure of the complex dependency relationship $X_1X_2X_3$ v $X_1X_2'X_3$ v $X_1X_2'X_3'$: $X_4X_5X_6$ v $X_4'X_5X_6$ v $X_4'X_5X_6'$ v $X_4'X_5'X_6'$ which equals 655. This relationship has a higher measure than previous.

Next, continuing in the same vein, sum the measures for each significant column as is shown in *Table 8.*

		$X_4X_5X_6$'	X_4X_5'X_6	X_4X_5'X_6'			X_4'X_5'X_6		Sum of pmsr's of significant
$X_1X_2X_3$		60	- 40	- 50			- 100		130
$X_1X_2X_3$'		- 40	60	- 50			0		30
X_1X_2'X_3		- 40	- 40	50			- 200		230
X_1X_2'X_3'		- 60	-60	- 75			-100		295
X_1'X_2X_3		- 40	60	50			0		-70
X_1'X_2X_3'		140	- 60	25			200		-305
X_1'X_2'X_3		-60	- 60	25			100		-5
X_1'X_2'X_3'		40	140	25			100		-305
								PMSR	685

Table 8 - Hill Climb Showing Significant Columns

The sum of the measures is 685. The positive valued rows denote the significant rows for the next calculation. If we continue, we find the complex relationship X_1 v X_3 : X_5X_6 with a measure of 900.

System State	*Direction of Derivation*	*I/O State*	*PMSR*
$X_1(X_2$'v$X_3)$	←	X_5	610
$X_1(X_2$'v$X_3)$	→	$(X_5X_6$ v X_4'X_6')	655
X_1	←	$(X_5X_6$ v X_4'X_6')	685
X_1	→	$(X_5X_6$ v X_4'X_5'X_6')	785
X_1 v X_2X_3	←	$(X_5X_6$ v X_4'X_5'X_6')	815
X_1 v X_2X_3	→	X_5X_6	850
X_1 v X_3	←	X_5X_6	900
X_1 v X_3	→	X_5X_6	900

Table 9 - The Hill Climb to Highest PMSR

Chapter 5

Extensions to the Boolean Analyzer Algorithm

Up to now the algorithm only identified strong dependency relationships for dependent events. One major contribution of this thesis is to extend the Boolean Analyzer algorithm to do rule generation. In the process, a rule system is defined that has the potential of being stronger than the complex rule from which it was derived.

5-1 Rule Types

Define a simple relationship X : Y as the relationship defined by the intersection of a row and a column of the State Linkage matrix where X is the j - tuple $<X_1, ..., X_j>$ for that row and Y is the (n-j)-tuple, $<X_{j+1}, ..., X_n>$ for that column. Ex. $X_1X_2X_3' : X_4 X_5 X_6$. Define a semisimple relationship as a relationship X : Y where X is the conjunction of any set of row attributes such that $X \subseteq \{X_1, ..., X_j\}$ and Y is the conjunction of any set of column attributes such that $Y \subseteq \{X_{j+1}, ..., X_n\}$. Note simple relationships are also semisimple. Ex. $X_1 X_2X_3' : X_4 X_5 X_6$, $X_1X_2X_3' : X_5$ Define a complex relationship as a relationship X : Y where X is the conjunction and/or disjunction of any set of row attributes such that $X \subseteq \{X_1, ..., X_j\}$ and Y is the conjunction and/or disjunction of any set of column attributes such that $Y \subseteq \{X_{j+1}, ..., X_n\}$. Note simple and semisimple relationships are complex.

Ex. $X_1X_2X_3' : X_4 X_5 X_6$, $X_1X_2X_3' : X_5$, $X_1 (X_2' \vee X_3) : X_5X_6$

Although the complex relationship X_1 v X_3 : X_5X_6 is easy to interpret, in the previous example we were only working with six variables. Real word datasets, such as market basket data, can have upwards of 1000 variables. Most researchers have shied away from complex relationships because they can be hard to interpret.

We can decompose the complex relationship into a system of several semisimple relationships. For example we can decompose X_1 v X_3 : X_5X_6 into the following set of semisimple relationships.

$$X_1 : X_5X_6 \; m = 750$$

$$X_3 : X_5X_6 \; m = 550$$

Notice that the measure of each individual semisimple relationship is smaller than the measure of the set of semisimple relationships combined. We also can decompose the complex relationship into a set of simple ones. For the example this set would be:

$X_1X_2X_3 : X_4X_5X_6$	m = -70
$X_1X_2X_3' : X_4X_5X_6$	m = 130
$X_1X_2'X_3 : X_4X_5X_6$	m = 30
$X_1X_2'X_3' : X_4'X_5X_6$	m = 195
$X_1X_2X_3 : X_4'X_5X_6$	m = 170
$X_1X_2X_3' : X_4'X_5X_6$	m = 70
$X_1X_2'X_3 : X_4'X_5X_6$	m = 270
$X_1X_2'X_3' : X_4'X_5X_6$	m = -45
$X_1' X_2 X_3 : X_4'X_5X_6$	m = 70
$X_1' X_2 ' X_3 : X_4'X_5X_6$	m = 155
$X_1' X_2 ' X_3 : X_4 X_5X_6$	m = -105
$X_1' X_2 X_3 : X_4 X_5X_6$	m = 30

This set of simple relationships have a combined event dependency which is higher than a single simple relationship. In fact the hill climb phase of the Boolean Analyzer algorithm gave us a set of easily interpretable relationships that are higher in measure than the individual rules that make up the complex relationship on which they are based.

5.2 Rule Generation

The measure of a relationship is bi-directional in that the measure of X : Y is equal to the measure of Y : X. We define a dependency rule as the probabilistic

implication X $\Rightarrow$ Y where X is some subset of row or column attributes and if X is a proper subset of row attributes then Y is a proper subset of column attributes also if X is a proper subset of column attributes then Y is a proper subset of row attributes.

According to this definition, each relationship matrix represents two possible dependency rules, X $\Rightarrow$ Y and, Y $\Rightarrow$ X. Obviously, X $\Rightarrow$ Y is not the same as Y $\Rightarrow$ X. For example, the strength of the statement, "People who buy Barbie dolls are likely to buy candy bar brand X.", tells us nothing about people who buy candy bar brand X and their affinity for Barbie dolls. We can use the confidence measure defined in chapter 3 to see which of the two rules is stronger. In actuality in order to decide which possible dependency rule is best we need only look at the support of the antecedent and use the rule with the smaller value since the rule support is the same for both implications.

We also have to be mindful of the support for the generated rule. Rules that have the same dependency measure might have different levels of support in the dataset. Define an interesting dependency rule as one that has support above a threshold s, confidence above a threshold c, and dependency above measure m.

For the results of the hill climb, which yields a rule that can be decomposed into a rule system of dependency relationships, we can also find the support and confidence of these rule systems. Rule system support is equal to the support of the complex relationship from which the system is derived. Given R_1 ... R_n represents a system relationship, where R_i represents rule i, we can calculate the support of the system by finding the sum of the support of all the rules, making sure we count their intersection only once.

$$S(R_1 \ldots R_n) = \sum_{1 \le i \le n} S(R_i) - \sum_{1 \le i < j \le n} S(R_i \cap R_j) +$$

$$\sum_{1 \le i < j < k \le n} S(R_i \cap R_j \cap R_k) - \ldots + (-1)^{n+1} (S(R_1 \cap R_2 \cap \ldots \cap R_n)$$

The question that remains is when we decompose complex relationships, which of the two implications do we use, X $\Rightarrow$ Y or Y $\Rightarrow$ X, for each rule. A complex relationship defines a rule system where all row variables are either in the rule antecedent or the rule consequent for all rules in the system. We can therefore define the system confidence as the support of all the rules divided by the support of all the rule antecedents.

$$C(R_1 \ldots R_n) = \frac{S(R_1 \ldots Rn)}{S(R_1 \ldots R_j)}$$

For a system, choose X $\Rightarrow$ Y such that system confidence is maximized. Or where rule system antecedent support is minimal.

This may not be the maximum confidence for all the possible rule combinations derivable from the complex relationship. If we choose R_i = X $\Rightarrow$ Y such that confidence is maximized for each R_i then we can obtain a rule system which has the potential of having higher confidence than the system defined by the complex rule. The following example shows a complex rule that can be decomposed into a rule system whose confidence is greater than the complex rule from which it was derived.

Given the complex dependency relationship : $x_1x_2' \vee x_1x_3 : x_5x_6 \vee x_4'x_5'x_6'$

From the state occurrence matrix, the confidence of $x_1x_2' \vee x_1x_3 \Rightarrow x_5x_6 \vee x_4'x_5'x_6'$ is 22/35 or .6286 and the confidence of $x_5x_6 \vee x_4'x_5'x_6' \Rightarrow x_1x_2' \vee x_1x_3$ is 22/47 or .4681. Therefore the more confident rule for this dependency relation is:

$x_1x_2' \vee x_1x_3 \Rightarrow x_5x_6 \vee x_4'x_5'x_6'$

We can define a rule system based on this rule as follows:

$x_1x_2' \Rightarrow x_5x_6$

$x_1x_2' \Rightarrow x_4'x_5'x_6$

$x_1x_3 \Rightarrow x_5x_6$

$x_1x_3 \Rightarrow x_4'x_5'x_6$

This rule system has the same support and confidence as the complex rule.

List each implication and its inverse with their confidence values.

$X_1x_2' \Rightarrow x_5x_6$	$C(R_1) = S(R_1) / 25$
$x_5x_6 \Rightarrow x_1x_2'$	$C(R_1) = S(R_1)/30$
$x_1x_2' \Rightarrow x_4'x_5'x_6$	$C(R_2) = S(R_2) /25$
$x_4'x_5'x_6 \Rightarrow x_1x_2'$	$C(R_2) = S(R_2)/17$
$x_1x_3 \Rightarrow x_5x_6$	$C(R_3) = S(R_3)/20$
$x_5x_6 \Rightarrow x_1x_3$	$C(R_3) = S(R_3)/30$
$x_1x_3 \Rightarrow x_4'x_5'x_6$	$C(R_4) = S(R_4)/20$
$x_4'x_5'x_6 \Rightarrow x_1x_3$	$C(R_4) = S(R_4)/17$

The highlighted rules are the one's with the higher confidence for the rule pair. We can now form a new rule system that has stronger rules than the previous system.

$X_1x_2' \Rightarrow x_5x_6$

$x_4'x_5'x_6 \Rightarrow x_1x_2'$

$x_1x_3 \Rightarrow x_5x_6$

$x_4'x_5'x_6 \Rightarrow x_1x_3$

5.3 Discussion

Most association or dependency rule algorithms do a greedy search through the hypothesis space of all rules. Boolean Analyzer does both greedy and exhaustive search through a reduced space of all rules. One major advantage to Boolean Analyzer is that it uses the domain knowledge of an expert to prune the hypothesis space. The incorporation of expert knowledge into Boolean Analyzer is unique among this class of algorithms. Domain knowledge introduction occurs during two separate steps of the algorithm. The partition step explicitly uses the domain knowledge of the expert to split the set of variables into column variables and row variables. By doing this we reduce our search space considerably. To get a sense of how reduced our search space becomes, for the example discussed, the number of possible variable partitions is equal to the C(6,3) + C(6,2) + C(6,1) = 41, where C(n,k) is the combination of n items taken k at a time. This represents the creation of 3 X 3, 2 X 4, and 5 X 2 partitions respectively. By recognizing that this dataset has one significant partition, we have reduced our search space for simple rules by a factor of 40.

Domain knowledge is implicitly introduced in the Booleanization step. This step is very significant in determining the strength of the rules produced by this algorithm. What is also significant about the Booleanization process is that it allows us to mine rules notwithstanding different variable data types.

One major drawback to this algorithm also lies in the partition step. Although this works well with datasets where there is a finite, small number of possible partitions, in datasets where there is no delineation, it is necessary to look at all partitions. This can be a computationally expensive procedure. An improvement to the algorithm might lie in finding some method that searches possible partitions to identify those that might yield high measure dependent relationships.

Boolean Analyzer makes only one pass through the dataset since Booleanization and the creation of the State Occurrence matrix can be done in one step. Subsequent search is done by looking at the State Occurrence matrix and the State Linkage matrix. Therefore, Boolean Analyzer is not bound by the size of the dataset, but by its dimensionality. Most other algorithms of this type are bound by both the size of the dataset and its dimensionality since they need to make several passes over the data. In Boolean Analyzer there is a tradeoff between size and dimensionality. If the number of variables is n, then Boolean Analyzer is most efficient when the number of records in the dataset is larger than 2^n.

In conclusion, Boolean Analyzer is an exponential algorithm. It gains efficiency by pruning the search space by using domain knowledge to partition the variable set into row variables and column variables. It can also find rule systems that can have a higher dependency, support, and confidence than any single rule that

is in the rule system. A rule system may even be more confident than any of the semisimple rules that are commonly generated by association rule/dependency rule algorithms.

Chapter 6

Boolean Thresholds

The next major focus of this thesis will be on the Booleanization step of the algorithm. As was pointed out in Chapter 4, this is a very significant step in the algorithm. It is this step that most directly affects the statistical significance and strength of the rules generated by the algorithm. By using an expert to create the threshold set H we implicitly input expert knowledge into the algorithm. This is very much desired, but there are problems with using an expert.

1. It is well documented that experts do not always agree.
2. The domain knowledge we seek may already be implicit in the data.
3. Experts may be able to set thresholds for the world at large, which may not accurately reflect the world of the data.
4. Given a dataset with high dimensionality, determining thresholds by hand can be time consuming.
5. The system end user may not be an expert.
6. There may not be an expert for the domain that we are looking at.
7. Experts may be fuzzy on the what the exact Boolean boundary may be, making it difficult to elicit Boolean thresholds from the expert.

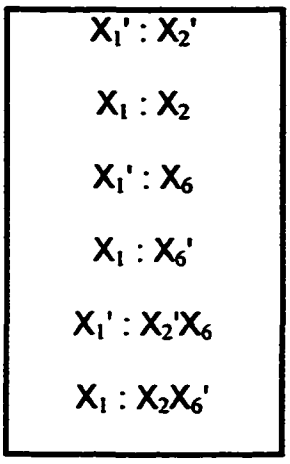
In light of all these drawbacks, automated methods of Booleanization are a viable alternative or an adjunct to Booleanization by an expert.

There are many different ways of Booleanizing data. The focus will be on determining the threshold set using the mean of the values of each data attribute, the

median of these values, the mode, and by partitioning the values into two groups using clustering techniques. Values above the threshold will take on a Boolean value of 1, and below it a value of 0. The rules generated by automated methods will be compared to the rules generated by Boolean Analyzer using an expert defined threshold set. An analysis of the difference between rule measures will be done using standard statistical analysis.

6.1 Testing on Synthetic data

Once mean, median, mode and clustering were identified as possible good methods of automated Boolean threshold determination mean, mean, and mode were looked at to see how well they performed on a dataset where there was no expert, and the relationships within the dataset were known. A synthetic dataset was created consisting of 152 records and 6 variables. Values for variable1 ascending values of dates (mm-dd-yy). Variable2 were numeric and also increased over time. Variable6 values decreased over time. Variables three, four, and five, were numeric and random. Partitioning was done between variable one and the rest of the variables. (How do the variables change over time?) The known dependency relationships can be expressed as follows:



From the synthetic dataset a random sample of 20 records was withheld. Mean was used to find Boolean thresholds in the remaining 132 record dataset. This was done so that each trial would be done using a slightly different dataset with different valued thresholds. This data was Booleanized and the fed into the Boolean Analyzer algorithm. The 20 record sample was used to see if the generated rules would predict well over "new" data. This was repeated 10 times. In 10 out of 10 trials the first six rules found by the Boolean Analyzer algorithm were the six known relationships and the rules were able to predict relationships in the test sample data. The same process was repeated using the median for the Boolean thresholds. Again, in 10 out of 10 trials the first six rules found by the Boolean Analyzer algorithm were the six known relationships and the rules were able to predict relationships in the test sample data.. Mode failed in this dataset as an automated method. Each record of variable1 contained a unique date value. Therefore no frequency was able to be found. The conclusion reached is that in the absence of an expert, mean and median Boolean

thresholds can find known relationships. There are problems with mode as an automated method.

6.2 Oil Futures Data

Once it was determined how automated methods work in a dataset where there was no expert, these methods were then used to compare the simple rules generated in a dataset where there was an expert available. The data used for this project was real data obtained from Hudson Capital Group[1], a Wall Street futures firm. The data describes crude oil prices, trading volume, and crude oil stockpiles. Data selection applied to this dataset, resulted in a subset of the original data which was selected according to criteria defined by the expert. Although the original data had entries for crude oil for several named months, the data that was significant in relating stock volume to price were the front crude oil future and the prompt/second month. According to the expert, the interest for the front month future is greater than that for prompt/second month The exception was for those times when the second month interest was greater than that of the first. In this case the second month is in actuality the front month and the third is the prompt/second month. Using this rule, the target data consisted of all front and prompt/second month observations.

---

[1] Permission granted for use of the company name Hudson Capital Group was granted by Mr. Todd Gross, company president.

Interest	Settlement Price	Volume	PADD 1 TOT CRUDE INV	PADD 2 TOT CRUDE INV	PADD 3 TOT CRUDE INV
33093	12.38	14346	17283	70552	151514
29961	12.26	10029	17283	70552	151514
30225	11.99	12165	17283	70552	151514
33050	11.96	13188	17283	70552	151514
29223	11.19	10052	17589	71204	151391
31621	11.16	12340	17589	71204	151391
27327	11.22	13011	17589	71204	151391
33379	11.28	13458	17589	71204	151391
27993	10.77	12070	17589	71204	151391

Table 10 - Sample Crude Oil Data

After data selection the next step was to clean the data. Although the data was in an Excel spreadsheet, all date related data was not entered in standard Excel format. In addition, the spreadsheet was not organized to facilitate its use with Boolean Analyzer. Each week a report is issued documenting stock quantities of crude oil at various locations throughout the country. The data on crude oil prices and volume traded for each commodity is reported daily. Boolean Analyzer needs all attributes, with their associated values, to be present in each observation. This necessitated associating the correct report entry with the appropriate crude oil entry.

The Boolean Analyzer algorithm itself needed to be updated. The algorithm previously did not incorporate the support and confidence measures outlined in Chapters 2 and 3. Current versions of the algorithm were prototype versions and only able to handle a small number of variables. The Boolean Analyzer program needed to be able to address these deficiencies. In addition, current versions of the Boolean Analyzer program do not do rule generation. The rule generation algorithm outlined in chapter 5 were incorporated as well.

6.3 Methods for Finding Boolean Thresholds

The decision to use mean, median, mode and clustering for finding the threshold set was based on the inclination to use methods that are easily calculated, and are fairly standard and accepted means of partitioning data. Also, packages that do these are widely available.

The idea behind automation was to try to partition the variables' values in a way that most closely matched the way an expert might partition them. Mean had intuitive appeal. Most people generalize by using an estimate of average value. In fact when the oil futures expert was asked to define a threshold set, his verbalizations indicated that he was thinking in terms of an average value. The disadvantage to using a mean for the threshold set is that mean is sensitive to outliers. In statistics, one way around the effects of outliers on the mean, is to consider them as errors and to disregard them. In data mining this is sometimes not desirable. While looking for rare patterns in the data, outliers may be the very information that leads to these patterns. Hence, the median offered a solution to this problem. The median is an accepted method of determining an average value, and is less sensitive to outliers. Mode offers a different intuitive appeal. People tend to remember things that are repetitive. If a particular value repeats itself, then it is this value that might be expressed by an expert when defining a threshold set. Mean, median, and mode all operate under the assumption that data is normally distributed. If the data is not normally distributed, and possibly multimodal, then clustering algorithms can capture this data distribution.

6.4 Statistical Methods

Boolean Analyzer attaches to each discovered rule a measure of dependency. This measure imposes an order on the set of rules. We can use this order to compare the measures of the rules generated by Boolean Analyzer using an expert defined threshold set, and the rules generated by Boolean Analyzer using rules generated by automated means. The measures associated with each rule are contiguous within the interval - $(n/2)^2 < 0 < (n/2)^2$, where n is the number of observations in the database. We cannot make any assumptions about the form of this contiguous distribution. Therefore the type of statistic used should be non-parametric.

Below is a sample of the top positively valued simple rules where expert thresholds were used for Booleanization.

IPV : P1'P2'P3'	614734
I'P'V' : P1P2P3'	460722
IP'V : P1'P2P3'	382098
IPV' : P1'P2'P3'	256496
I'PV' : P1P2'P3'	213114
I'P'V' : P1P2'P3'	202268
IP'V' : P1'P2P3'	146260
I'PV : P1'P2P3'	116230
IP'V : P1P2P3	108212
I'PV' : P1'P2'P3'	86874
IP'V : P1'P2P3	71904

Table 11 - Rules and Rule Measures

In the above table each symbol stands for the following:

I - Open Interest high	I' - Open Interest low
P - Price high	P' - Price low
V - Volume high	V' - Volume low
P1 - Padd 1 Inventory high	P1' - Padd 1 Inventory low
P2 - Padd 2 Inventory high	P2' - Padd 2 Inventory high
P3 - Padd 3 Inventory high	P3' - Padd 3 Inventory high

For example the first rule, IPV : P1'P2'P3', can be interpreted as saying, "High Interest, High Price, and High Volume has a direct dependency to Padd 1 Low Inventory, Padd 2 low Inventory and Padd3 Low Inventory.

Two different non-parametric statistical tests were used, the Spearman Rank-Order Correlation Coefficient and the Kendall Tau b Correlation Coefficient. The Spearman Rank-Order Correlation Coefficient uses the magnitude of the differences between one ranking to another to get a measure on the association between the two rankings. The Kendall Rank-Order Correlation Coefficient uses the number of agreements and disagreements in the ordering of the two variables' rankings to find a measure of association between the two. These two correlation coefficients can quantify the differences between the variables. Each of these tests enables us to have different insights into the differences between each of the rankings.

Since, theoretically, we can create an infinite set of rules from one dataset, which subset of these rules should be included in the statistical analysis? Consider first rules that are not simple or semisimple, and that there could be an inginite set of rules that fall into this class. But due to the underlying principle that these rules are created by partitioning the state occurrence matrix and subsequently applying Boolean reduction, this set is finite. It was shown that these types of rules define a rule system of semisimple rules. Therefore semisimple rules are the building blocks of these more complex rules. In turn, rules that are semisimple but are not simple are equivalent to rule systems of simple rules. Hence, the simple rules are the building blocks of the other types of rules. In comparing the rules generated by Boolean Analyzer, we need only compare the simple rules. There is also a statistical basis for limiting the analysis to this set. The simple rules represent an independent set of rules

whose measurements are not dependent on the measurement of other rules in this set. The measures of more complex rules are dependent on the values of these simple rules.

The simple rules are the rules that are defined by the intersection of a row and column of the state linkage matrix. Deciding on one or more partitions of the variable set creates the state linkage matrix. The oil futures dataset had a natural partition between the reported oil stock data and the crude oil price/volume data. It was not necessary to look at all the simple rules since it is only the ones generated by this one partition that was considered significant by the expert. The resulting dataset had 5,782 entries and 6 variables.

6.5 Correlation Results

Boolean thresholds elicited from the expert were applied to the target data resulting in a Booleanized data set. The same was done for mean thresholds, median thresholds and thresholds determined by mode. When the mode for open interest variable was found, the mode value was zero. Since open interest shouldn't have a zero value, errors in the data were suspected. This suspicion was confirmed with the domain expert. Hence, mode is sensitive to errors in the data, making it less attractive for threshold automation than methods using mean and median. In addition, when the value for each occurrence of a variable is unique, such as date values, mode fails as an automated method. There were four records having this erroneous data. These were deleted from the dataset, with the resulting dataset having 5,878 records, and all further analyses were done on this data.

In order to cluster the data, each value was transformed into its z- score. The transformed data was clustered using the FASTCLUS procedure in the *SAS®* statistical package. FASTCLUS uses a K-means algorithm to do clustering. Clusters corresponding to high values in the original target data were assigned a value of one, clusters with low values a zero.

The Kendall Tau b and Spearman rank order correlation coefficients were calculated, using the *SAS®* CORR procedure on the simple rules generated by the above methods of Booleanization. The results were as follows:

Spearman Corr Coefficient		Kendall Tau b	
	Expert		Expert
MEAN	0.71832	MEAN	0.52948
p value mean	0.0001	p value mean	0.0001
MEDIAN	0.70336	MEDIAN	0.5085
p value median	0.0001	p value median	0.0001
MODE	0.66937	MODE	0.48752
p value mode	0.0001	p value mode	0.0001
CLUSTER	0.21197	CLUSTER	0.14786
p value cluster	0.0927	p value cluster	0.086

Table 13 - Results of Expert vs. Mean, Mode, and Clustering

Looking at the Spearman correlation coefficient we see a 72% and 70% correlation for mean and median respectively. Although not very strong, there is a good correlation to the Expert's rules. The values for the Kendall Tau b where not high enough to show even good correlation. One possible explanation for these results

could be attributed to a poor Boolean threshold set by the expert for the variable associated with padd3 crude oil stockpiles. For the other five variables the expert was able to give concrete Boolean thresholds. For padd3, the expert was not sure what an exact boundary should be. The next step was to test to see if this fuzzy boundary affected the correlation coefficients between the expert rules and the automated ones. The variable for padd3 was removed from the dataset and the same analysis was repeated on the remaining five variables. The results of the five variable analysis were as follows:

Spearman Correlation Coefficients	Expert	Kendall Tau b Correlation	Expert
MEAN	0.96188	MEAN	0.85484
p value mean	0.0001	p value mean	0.0001
MEDIAN	0.97984	MEDIAN	0.89516
p value median	0.0001	p value median	0.0001
MODE	0.84824	MODE	0.69758
p value mode	0.0001	p value mode	0.0001
CLUSTER	0.3794	CLUSTER	0.2621
p value cluster	0.0322	p value cluster	0.035

Table 14 - Results of 5 Variable Comparison of Expert vs. Mean, Median, Mode, and Clustering

Results improved tremendously with the removal of this variable. Both correlation coefficients showed high correlation between the expert and the automated methods of mean and median, a lower correlation between the expert and mode, and a poor correlation between the expert and the clustering technique. Results were

encouraging, but the question remained, did things get better because the problem was reduced to five variables or was there really a significant correlation between the expert and these automated methods. The variable for padd3 was reintroduced into the dataset. Since mean seemed to be good method of automation, the expert Boolean threshold for that variable was replaced with the mean Boolean threshold. The analysis were repeated. The same was done with replacement of the expert Boolean threshold with the median Boolean threshold.

The results were as follows:

Spearman Correlation Coefficent	Expert variable replaced with mean threshold	Expert variable replaced with median threshold
MEAN	0.98118	0.97276
p value mean	0.0001	0.0001
MEDIAN	0.97088	0.95861
p value median	0.0001	0.0001
MODE	0.86809	0.86012
p value mode	0.0001	0.0001
CLUSTER	0.27647	0.30865
p value cluster	0.027	0.0131

Kendall Tau b Correlation Coefficient	Expert variable replaced with mean threshold	Expert variable replaced with median threshold
MEAN	0.91369	0.875
p value mean	0.0001	0.0001
MEDIAN	0.85417	0.82738
p value median	0.0001	0.0001
MODE	0.69444	0.68155
p value median	0.0001	0.0001
CLUSTER	0.19841	0.22718
p value cluster	0.0205	0.008

Table 15 - Six Variable Results with Fuzzy Threshold Replaced

The above correlation coefficients indicate that the previous high coefficients were not due to the reduction in the number of variables. In fact, the substitution of a mean or median Boolean boundary for a fuzzy boundary given by an expert can yield good

results. Mean and Median look like good methods of automated Boolean threshold determination, with mode not as strong a method, and clustering a poor one.

6.6 Conclusions

1. The rules generated using Boolean thresholds elicited from an expert correlate well with the rules generated using automated methods of mean and median. The expert rules correlate less significantly with rules found by Boolean thresholds using mode, and poorly with those found using clustering.
2. Mean and median thresholds can be used in combination with expert defined thresholds when an expert is fuzzy about a specific variable's threshold.
3. Mean and median thresholds yield excellent results in the absence of an expert.
4. Mode is not as good an automated method since
    a) it is more sensitive to errors
    b) it can't capture thresholds for unique data values such as date values
    c) it doesn't correlate as well to an expert's knowledge as mean and median.

6.7 Future Research

1. Each method of finding rules produced a somewhat different rule set. Further investigation can be done to see which method of Booleanization, expert, mean, mode, or clustering, produces acceptable rules to the domain expert.
2. Mean and Mode seem to be acceptable automated Booleanization methods in the oil futures domain, but would they hold in other domains? This question will be

investigated using a dataset of clinical opthamology data provided by a clinical study done by Dr. Irene Ludwig, Louisiana State University School of Medicine.

3. Boolean Analyzer works well when there are a small number of identifiable partitions on the variable set. When there is no logical partitioning of the variable set, Boolean Analyzer becomes computationally expensive. Here, research can center on finding an algorithm that will identify partitions that can lead to high measure dependency relationships.
4. Boolean Analyzer can not work with missing values. Rectifying this deficiency can be another avenue of research.
5. The hill climb step of the Boolean Analyzer algorithm yields one complex relationship that defines the highest measure system dependency relation. If we use a genetic algorithm to investigate multiple starting points in this hill climb, and then look at the complex rules generated at the penultimate step in the climb. This may generate a set of complex rules that have higher measure than the set of semi-simple rules.

Appendix A - Source Code for Boolean Analyzer Program

```
//The following code will Booleanize an N row dataset according to given
//boundaries.
// It will then find the dependency relationships in the Booleanized dataset


#include <iostream.h>
#include <math.h>
#include <stdlib.h>
#include <fstream.h>

#define N 5778 //# of rows in data file
#define M 5    //# of columns in data file (number of variables
#define RV 3
#define CV 2
#define R 8
#define C 4

#define MAXSTACK 32
#define MAXRULES 32



        int exp2(int);

        void generaterowtable (int tbl[R][RV],int r,int rv);
        void generatecoltable (int tbl[C][CV], int col, int cv);
        void nextrowbinval(int rowvl [RV], int, int);
        void nextcolbinval(int colvl [RV], int, int);
        void boolzrpt ();



int
main (int)
{

        float actmtrx [N] [M];// activity matrix,
                                                // data reduced to their binary mappings

        float SO [R] [C];  // State occurence matrix
        float SL [R] [C]; //State Linkage matrix
        unsigned long int r = 0, c = 0;
        int i, j, k,countr, countc, count = 0;
```

```
float ri= 0, cj = 0, aij;
float SRTARRAY [MAXRULES];
unsigned int rules[MAXRULES][2];

int l,m,n,p,q,s;
int rowmatch, colmatch;

int rowtable [R][RV];
int coltable [C][CV];

int rowvar[RV];
int colvar[CV];

int rowvalue[RV];
int colvalue[CV];
float measure;


//boolzrpt();
fstream InStream("boolcl50.prn", ios::in);

if (InStream.fail())
{
        cerr << "File that failed to open was" << "boolxprt.txt"
                        << "\n\n";
};

fstream OutStream("output.out", ios::out);

if (OutStream.fail())
{
        cerr << "File that failed to open was" << "output.out"
                        << "\n\n";
};

//Input Boolean data
for (i = 0;i < N;++i)
        for (j = 0;j < M; ++j)
                InStream >> actmtrx [i] [j];


//print activity matrix
    for (i = 0; i < N; ++i)
                {
                        for (j = 0; j < M; ++j)
                        {
```

```
                        cout<< actmtrx [i][j] << ' ';
                        OutStream << actmtrx [i][j] << ' ';
                }
                cout<<'\n';
                OutStream << '\n';
        }


//Initialize SO and SL matrice
for (i = 0; i < R; ++i)
        for (j = 0; j < C; ++j)
                SO[i][j] = 0;

for (i = 0; i < R; ++i)
        for (j = 0; j < C; ++j)
                SL[i][j] = 0;

//Calculate SO martrix

countr = RV;
countc = M - countr;
for(i = 0;i < N; ++i)
        {
                for (j = 0; j < M; ++j)
                {
                if (count < RV)
                 {
                        //      r <<= 1;
                                if (actmtrx[i][j] == 1)
                                        r = r + exp2(countr - 1);
                                countr--;
                        }
                        else
                        {
                        //              c <<= 1;
                                if (actmtrx[i][j] == 1) c = c + exp2(countc -1) ;
                                countc--;
                        }

                 ++count;
                }

                SO[r][c] = SO[r][c] + 1;
                r = 0;
                c = 0;
                count = 0;
```

```
                countr = RV;
                countc = M  - countr;
        }


    //Calculate SL matrix
        for (i = 0; i < R; ++i)
        {

                for (k = 0; k < C; ++k)
                                ri += SO [i][k];

                for (j = 0; j < C; ++j)
                {
                                aij = SO[i][j];

                                for (k = 0;k < R; ++k)
                                        cj += SO[k][j];

                                SL[i][j] = (aij * N) - (ri * cj);
                                cj = 0;
                }
                ri = 0;

        }
//Print State Occurence Matix


    cout<<'\n'<<"State Occurence Matrix";
    cout <<'\n';
    OutStream << '\n' << "State Occurence Matrix" << '\n';

    for (i = 0; i < R; ++i)
                {
                        for (j = 0; j < C; ++j)
                        {
                                cout<< SO [i][j] << ' ';
                                OutStream << SO [i][j] << '\t';
                        }
                        cout<<'\n';
                        OutStream << '\n';
                }

 //Print SL matrix
```

```
cout <<'\n'<<"State Linkage Matrix"<<'\n';
OutStream <<'\n'<<"State Linkage Matrix"<<'\n';
for (i = 0; i < R; ++i)
                {
                        for (j = 0; j < C; ++j)
                        {
                                cout.setf(ios::left);
                                cout.width(8);
                                OutStream.setf(ios::left);
                                OutStream.width(8);
                                cout << SL [i][j];
                                OutStream <<  SL [i][j];
                        }
                        cout<<'\n';
                        OutStream << '\n';
                }

cout<<'\n';
    OutStream << '\n';

k = 0;
    for (i = 0; i < R; ++i)
    {
            for (j = 0; j < C; ++j)
            {
                    rules [k] [0] = i;
                    rules [k] [1] = j;
                    SRTARRAY [k] = SL[i] [j];
                    ++k;
            }

    }

    //generate truth table for pattern matching and for choosing variables
    generaterowtable (rowtable, R, RV);
    generatecoltable (coltable, C, CV);



    countr = 0;
    countc = 0;
    k = 0;
    l = 0;


    //find row half of relationship
```

```
for (i = (R-1); i < R; ++i)
{
        while (k < RV)
        {
                if (rowtable[i][k] == 1)
                {
                        rowvar[countr] = k + 1;
                        ++countr;
                }

                ++k;
        }

        //find column half of relationship
        for (j = (C-1); j < C; ++j)
        {
                while (l < CV)
                {
                        if (coltable[j][l] == 1)
                        {
                                colvar[countc] = l + 1;
                                ++countc;
                        }

                        ++l;
                }

                //Now rowvar has numbers of which variables are in column
                //half of relationship. colvar has column variables.
                //Now assign values to these variables.

                //Find measures for all Boolean combinations of these variables

                //intitialize first Boolean value for row and column



                for (q = 0; q < (exp2(countr)); ++q)
                {
                        nextrowbinval(rowvalue, q, countr);

                        for (s = 0; s < (exp2(countc)); ++s)
                        {
                                nextcolbinval(colvalue, s, countc);
                                measure = 0;
```

```
//print out relation name
for (m = 0;m < countr; ++m)
{
        OutStream << 'X' << rowvar[m];
        if (rowvalue[m] == 0) OutStream << '\';
}

OutStream << " : ";

for (m = 0;m < countc; ++m)
{
        OutStream << 'X' << colvar[m]+RV;
        if (colvalue[m] == 0) OutStream << '\';
}

OutStream << "\t\t";
//find rows in SL that match row half of relation
rowmatch = 1;
colmatch = 1;
for (m = 0; m < R; ++m)
{
        for (p = 0; p < countr; ++p)

                if (rowvalue[p] != rowtable[m][rowvar[p]-1])
                                rowmatch = 0;



        //if we find a matching row, then find all columns that
        //   match
        //and add the SL entry to measure.
                if (rowmatch)
                {
                        for (n = 0; n < C; ++n)
                        {
                                for (p = 0; p < countc; ++p)
                                if (colvalue[p] !=
                                  coltable[n][colvar[p]-1])
                                                colmatch = 0;

                                if (colmatch) measure +=
                                                SL[m][n];
                                colmatch = 1;
                        }//endn
                }//endif
```

```
                                        rowmatch = 1;
                        }//endm
                        OutStream << "\t\t"<< measure<<'\n';
                        measure = 0;
                }//endr
        }//endq
        l = 0;
        countc = 0;
        } //end j loop
                k = 0;
                countr = 0;
        } // end i loop




        InStream.close();
        OutStream.close();
        return(0);

        }


 void
generaterowtable ( int tbl[R][RV], int r, int rv)
{
        int i, value, num;

        unsigned c, displayMask = 1 << (rv - 1);

        value = 0;
        num = 0;
        for (i = 0;i < r; ++i)
        {
                for (c = 1; c <= rv; c++)
                {
                        tbl[i][c-1] = (value & displayMask ? 1 : 0);
                        value <<= 1;

                }
                ++num;
                value = num;
        }
```

```
}

void
generatecoltable ( int tbl[C][CV], int col, int cv)
{
        int i, value, num;

        unsigned c, displayMask = 1 << (cv - 1);

        value = 0;
        num = 0;
        for (i = 0;i < col; ++i)
        {
                for (c = 1; c <= cv; c++)
                {
                        tbl[i][c-1]  = (value & displayMask ? 1 : 0);
                        value <<= 1;

                }
                ++num;
                value = num;
        }



}

void
nextrowbinval(int rowvl[RV], int rowbin, int count)
{
        unsigned displayMask = 1;
        int c, value;

        displayMask = 1;// << count;
        value = rowbin;

        for (c = (count-1); c >=0; c--)
        {
                rowvl[c] = (value & displayMask ? 1 : 0);
                value >>= 1;
        }

}

void
```

```
nextcolbinval(int colvl[RV], int colbin, int count)
{
        unsigned displayMask = 1;
        int c, value;

        displayMask = 1; //<< count;
        value = colbin;

        for (c = (count - 1); c >= 0; c--)
        {
                colvl[c] = (value & displayMask ? 1 : 0);
                value >>= 1;
        }


}


 int exp2 ( int x)

 {
         int i, result = 1;


         if (x == 0) return 1;
         else for (i = 1; i <= x; i++)
                 result = result * 2;

         return result;

 }

 void boolzrpt ()
 {
         int i;

        float crude[M];  // holds one record of the crude oil data

        //boundaries for interest, settlement price, volume, padd1,
        //   padd2, padd3, padd4, padd5, total crude
        float boundary [] = {66830,19.42,29808,15308, 72927};

        // open input and output files
        fstream Incrude("crud5no0.prn", ios::in);

        if (Incrude.fail())
```

```
{
        cerr << "File that failed to open was " << "crudrpt.prn"
                        << "\n\n";
        exit(1);
};


fstream OutStream("x6160.txt", ios::out);

if (OutStream.fail())
{
        cerr << "File that failed to open was" << "crudbool.out"
                        << "\n\n";
};


i = 1;
Incrude >> crude[0];



while (!Incrude.eof() )
{
        // if value is lower than boundary value is replaced by 0 else 1

        if (crude[0] < boundary[0])
                                OutStream << '0' << '\t';
        else OutStream << '1' << '\t';


                while (!Incrude.eof() && (i < 5))
                {
                        Incrude >> crude[i];
                        if (crude[i] <= boundary[i])
                                OutStream << '0' << '\t';
                        else OutStream << '1' << '\t';
                        ++i;
                }

        //      Incrude >> crude[9];
                OutStream << endl;

                i = 1;

                Incrude >> crude[0];
```

```
        }

        Incrude.close():
        OutStream.close();
        return;

}
```

Appendix B - *SAS* ® code for finding Correlation Coefficients.

```
data thesis;
        infile 'c:\datathes\rl6.prn';
        input rulename $ 1-30
                expert
                mean
                mode
                median
                cluster;

proc print data = thesis;
        title 'Thesis';
         var rulename
                expert
                mean
                mode
                median
                cluster;



proc corr data = thesis spearman kendall;
        var expert;

        with    mean
                mode
                median
                cluster;
run;
```

Appendix C - Synthetic Dataset

```
10/1/93,28,3,9,71,99
10/2/93,140,80,6,90,98
10/3/93,156,87,4,12,98
10/4/93,187,95,7,69,97
10/5/93,226,40,0,16,97
10/6/93,288,16,5,40,96
10/7/93,309,10,2,64,96
10/8/93,449,84,4,18,95
10/9/93,453,89,3,32,94
10/10/93,481,77,2,44,92
10/11/93,535,23,8,61,91
10/12/93,609,37,3,86,90
10/13/93,658,58,9,51,90
10/14/93,739,33,8,25,87
10/15/93,776,25,1,34,86
10/16/93,807,0,7,84,83
10/17/93,896,27,7,40,83
10/18/93,1030,81,7,43,82
10/19/93,1044,7,4,33,82
10/20/93,1063,71,3,79,80
10/21/93,1073,15,5,95,80
10/22/93,1154,24,9,11,79
10/23/93,1499,98,6,59,79
10/24/93,1503,90,5,24,78
10/25/93,1556,86,0,43,78
10/26/93,1617,75,2,37,77
10/27/93,1628,39,5,27,76
10/28/93,1860,58,2,7,76
10/29/93,1919,89,1,65,75
10/30/93,2026,90,2,94,75
10/31/93,2060,84,4,49,75
11/1/93,2167,76,8,38,75
11/2/93,2177,19,3,41,75
11/3/93,2232,15,6,9,75
11/4/93,2237,20,6,50,74
11/5/93,2277,18,9,47,72
11/6/93,2352,0,4,28,71
11/7/93,2438,75,8,82,71
11/8/93,2439,17,1,1,70
11/9/93,2546,15,3,18,70
11/10/93,2566,97,8,58,70
11/11/93,2572,72,1,25,69
11/12/93,2637,2,8,78,68
11/13/93,2697,79,3,45,67
11/14/93,2729,55,8,56,67
11/15/93,2782,20,2,36,65
11/16/93,2844,57,3,52,64
11/17/93,2896,42,5,5,64
11/18/93,2957,54,2,96,62
11/19/93,3009,93,9,83,62
11/20/93,3019,40,8,98,62
11/21/93,3024,25,1,69,61
11/22/93,3297,1,3,69,59
11/23/93,3316,11,1,29,58
```

11/24/93,3477,39,5,31,58
11/25/93,3534,29,8,30,56
11/26/93,3565,37,4,84,54
11/27/93,3634,40,7,65,54
11/28/93,3649,40,9,64,54
11/29/93,3735,51,2,58,54
11/30/93,3958,74,4,90,54
12/1/93,3989,28,6,89,53
12/2/93,4018,28,3,1,53
12/3/93,4019,29,9,76,53
12/4/93,4100,58,6,26,53
12/5/93,4229,50,7,85,52
12/6/93,4275,9,5,41,50
12/7/93,4284,7,4,69,50
12/8/93,4438,49,4,94,48
12/9/93,4542,98,5,53,48
12/10/93,4596,61,0,18,48
12/11/93,4635,28,0,15,47
12/12/93,4687,42,5,55,46
12/13/93,4780,78,0,40,46
12/14/93,4799,5,3,60,46
12/15/93,4898,51,5,32,45
12/16/93,4921,90,7,57,44
12/17/93,4943,43,7,16,43
12/18/93,4952,5,2,75,43
12/19/93,4970,95,3,68,43
12/20/93,5033,79,0,79,43
12/21/93,5060,48,2,60,41
12/22/93,5137,90,9,65,41
12/23/93,5172,1,6,96,41
12/24/93,5180,0,9,7,41
12/25/93,5248,29,5,80,41
12/26/93,5330,4,0,56,40
12/27/93,5382,31,4,51,40
12/28/93,5420,73,4,32,40
12/29/93,5429,7,4,57,39
12/30/93,5476,76,5,65,39
12/31/93,5575,80,4,79,38
1/1/94,5610,72,6,73,37
1/2/94,5751,19,7,6,37
1/3/94,5795,24,9,81,37
1/4/94,5886,21,1,23,37
1/5/94,5891,92,2,25,36
1/6/94,5897,71,6,77,35
1/7/94,5960,75,3,82,34
1/8/94,6139,71,8,22,34
1/9/94,6226,35,5,35,34
1/10/94,6317,20,5,62,33
1/11/94,6331,13,0,63,33
1/12/94,6343,83,6,42,32
1/13/94,6345,18,4,97,32
1/14/94,6459,76,3,48,32
1/15/94,6570,54,6,3,29
1/16/94,6572,64,8,34,29
1/17/94,6724,11,4,77,29
1/18/94,6761,35,4,74,28

```
1/19/94,6779,43,7,50,28
1/20/94,6840,17,2,8,27
1/21/94,6929,97,2,61,27
1/22/94,6951,30,8,31,27
1/23/94,6952,79,1,30,27
1/24/94,6961,7,4,45,26
1/25/94,7055,59,0,31,22
1/26/94,7127,13,3,76,21
1/27/94,7344,62,9,82,21
1/28/94,7460,55,6,19,20
1/29/94,7463,90,3,60,20
1/30/94,7560,59,7,71,19
1/31/94,7844,68,7,25,18
2/1/94,7852,54,5,56,17
2/2/94,7901,28,4,35,17
2/3/94,7931,97,5,58,16
2/4/94,8144,46,0,37,16
2/5/94,8146,13,6,15,15
2/6/94,8274,19,3,9,15
2/7/94,8298,61,7,43,14
2/8/94,8472,42,3,70,13
2/9/94,8626,21,0,88,13
2/10/94,8714,4,4,73,12
2/11/94,8725,75,8,20,12
2/12/94,8866,46,8,82,11
2/13/94,9057,41,4,34,11
2/14/94,9109,24,8,54,10
2/15/94,9137,53,6,8,10
2/16/94,9161,71,2,5,9
2/17/94,9234,27,9,32,9
2/18/94,9276,64,3,36,9
2/19/94,9278,32,2,21,8
2/20/94,9287,42,6,24,8
2/21/94,9385,68,5,60,8
2/22/94,9449,7,0,15,8
2/23/94,9495,24,3,58,6
2/24/94,9592,28,8,89,6
2/25/94,9604,52,3,0,5
2/26/94,9798,25,1,64,5
2/27/94,9853,98,3,24,5
2/28/94,9937,87,1,56,4
3/1/94,9967,60,6,69,3
```

Bibliography

Adriaans, P. and D. Zantinge, 1996. *Data Mining*, Addison Wesley Longman Limited

Agrawal, R., H. Mannila, S. Ramakrishnan, H. Toivonen, and I. Verkamo, 1996 Fast Discovery of Association Rules. In *Advances In Knowledge Discovery and Data Mining*. AAAI/MIT Press, Cambridge Mass

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

Biggs, N. L. 1989. *Discrete Mathematics*, Oxford University Press

Brachman, R. J., and T. Anand, 1996. The Process of Knowledge Discovery in Databases. In *Advances In Knowledge Discovery and Data Mining*. AAAI/MIT Press, Cambridge Mass.

Brachman, R. J., T. Khabaza, W. Kloesgen, G. Piatetsky-Shapiro, E. Simoudis, 1994 Mining Business Databases. *Communications of the ACM* 39(11) : 42-48.

Buntine, W. 1996. Graphical Models for Discovering Knowledge. In *Advances In Knowledge Discovery and Data Mining*. AAAI/MIT Press, Cambridge Mass.

Burl, M. C., L. Asker, P. Smyth, U. M. Fayyad, P. Perona, L. Crumpler, and J. Aubele 1998, Learning to Recognize Volcanoes on Venus. *Machine Learning*, 30(2/3):165-192.

Cheeseman, P., and J. Stutz, 1996. Bayesian Classification (AutoClass); Theory and Results. In *Advances In Knowledge Discovery and Data Mining*. AAAI/MIT Press, Cambridge Mass

Cody, R. P., and J. K. Smith 1997, *Applied Statistics and the SAS Programming Language*, Prentice Hall

Cox, K. C., S. G. Eick, G. J. Wills, and R. J. Brachman, 1997. Visual Data Mining: Recognizing Telephone Calling Fraud. *Data mining and Knowledge Discovery*. 1(2):225-231.

Domanski, B., 1996. Clustering Still the Right Technique for Studying Workloads, *ICCM Capacity Management Review*, 24(4) : 1-5.

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

Dubois, D. And H. Prade, 1980. *Fuzzy Sets and Systems Theory and Applications,* Academic Press, Inc..

Dzeroski, S., 1996. Inductive Logic Programming and Knowledge Discovery in Databases. In *Advances In Knowledge Discovery and Data Mining.* AAAI/MIT Press, Cambridge Mass.

Elder, J.F. IV, and D. Pregibon, 1996. Statistical Perspective on Knowledge Discovery in Databases, In *Advances In Knowledge Discovery and Data Mining.* AAAI/MIT Press, Cambridge Mass.

Fayyad, U. M., D. Haussler, and P. Stolorz, 1994. Mining Scientific Data. *Communications of the ACM* 39(11) : 27-34.

Fayyad, U. M., G. Piaetsky-Shapiro, and P. Smyth, 1994. The KDD Process for Extracting Useful Knowledge form Volumes of Data. *Communications of the ACM* 39(11):27-34.

Fayyad, U. M., G. Piaetsky-Shapiro, and P. Smyth, 1996. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37 – 54.

Fayyad, U. M., G. Piaetsky-Shapiro, and P. Smyth, 1996. From Data Mining to Knowledge Discovery: An Overview. In *Advances In Knowledge Discovery and Data Mining.* AAAI/MIT Press, Cambridge Mass.

Fayyad, U. M., S. G. Djorgovski, and N. Weir, 1996. Automating the Analysis and Cataloging of Sky Surveys. In *Advances In Knowledge Discovery and Data Mining.* AAAI/MIT Press, Cambridge Mass.

Freund, J. E., and G. A. Simon, 1997. *Modern Elementary Statistics* Prentice Hall Inc

Glymour, D. M.; Pregibon, D.; Smyth P.; Statistical Inference and Data Mining. *Communications of the ACM* 39(11):27-34.

Inmon, W.H., 1994. The Data Warehouse and Data Mining. *Communications of the ACM* 39(11):49-50.

Klir, G., and B. Yuan 1995. *Fuzzy Sets and Fuzzy Logic* Prentice Hall PTR

Klir, G, U. H. St. Clair, and B. Yuan, 1997. *Fuzzy Set Theory, Foundations and Applications,* Prentice Hall PTR

Langley, P. 1996. *Elements of Machine Learning,* Morgan Kaufman Publishers Inc.

Matheus, C. J., G. Piatsky-Shapiro, and D. McNeil, 1996. Selecting and Reporting What is Intersting:The KEFIR Applications to Healthcare Data. In *Advances In Knowledge Discovery and Data Mining*. AAAI/MIT Press, Cambridge Mass.

Mitchell, T. 1997. *Machine Learning*, McGraw Hill Publishers Inc.

Noether, G. E, 1967. *Elements of Nonparametric Statistics* John Wiley & Sons Inc.

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

Rosen, K. H. 1995. *Discrete Mathematics and Its Applications*, McGraw-Hill, Inc.

SAS Institute Inc, 1989. *SAS/STAT® User's Guide, Version 6, Fourth Edition Volumes 1 and 2*, SAS Institute Inc.

Schlotzhauer, S. D., and R. C. Litell 1997. *SAS® System for Elementary Statistical Analysis*, SAS Institute Inc.

Siegel, S., and J. N. Castellan, 1988. *Nonparametric Statistics For the Behavioral Sciences*, McGraw Hill Inc.

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

Weiss, S. M., and N. Indurkhya ,1998. *Predictive Data Mining A Practical Guide*, Morgan Kaufman Publishers Inc.

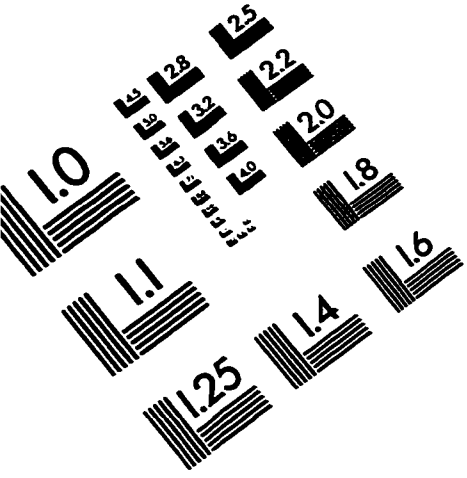
Yule, G. U., and M.G. Kendall, 1968. *An Introduction to the Theory of Statistics*, Hafner Publishing Co.

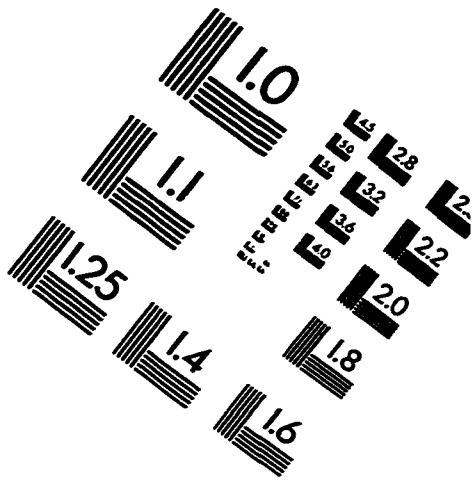
Internet Sites Used in the Preparation of this Thesis

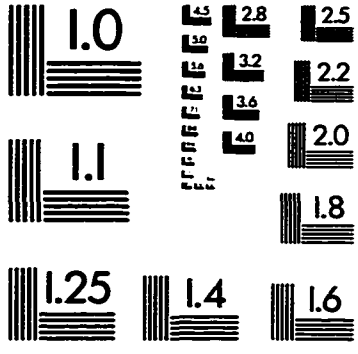
http://www.ics.uci.edu/AI/ML/Machine-Learning.html
UCI Machine Learning, University of California Irvine

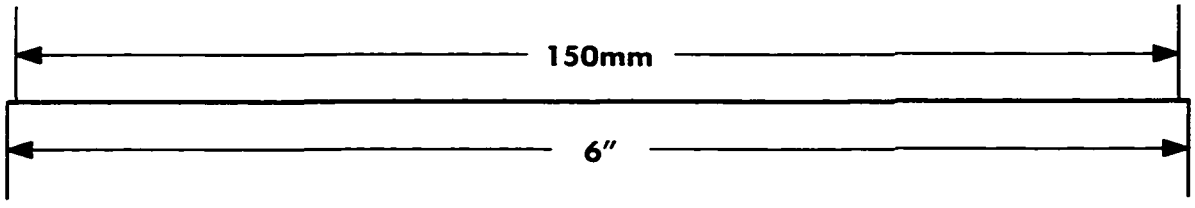
http://www.kdnuggets.com/index.html
KDNuggets™ Directory: Data Mining and Knowledge Discovery Resources

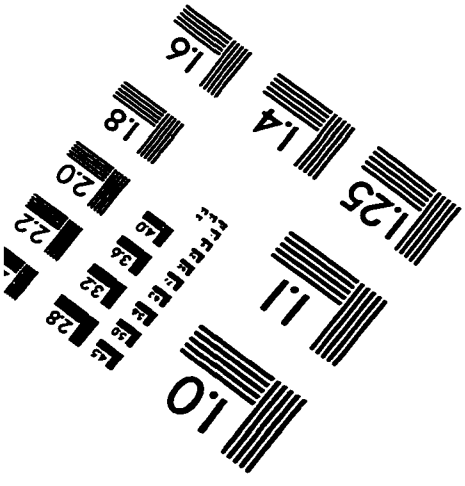
IMAGE EVALUATION TEST TARGET (QA-3)

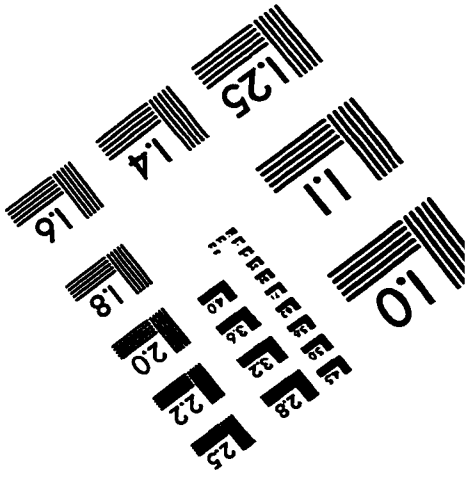












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989