A Clustering Rule Based Approach for Classification Problems

by

Philicity Kapryelle Williams

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 9, 2010

Keywords: Data Mining, Classification, Rule Based Classifiers, Data Clustering

Approved by

Juan E. Gilbert, Chair, Professor of Computer Science and Software Engineering
N. Hari Narayanan, Professor of Computer Science and Software Engineering
Cheryl D. Seals, Associate Professor of Computer Science and Software Engineering

Abstract

Today's data storage and collection abilities have allowed the accumulation of enormous amounts of data. Data mining can be a useful tool in transforming these large amounts of raw data into useful information. Predictive modeling is a very popular area in data mining. The results of these type tasks can contain helpful information that can be used in decision making. Problems arise when the data sets that are used to build these models are not as complete (e.g. erroneous/missing values) as the data used to evaluate the model. Rule based classifiers are widely used and accepted type of predictive model. We present a method to reduce the severity of the effects of missing data on the performance of rule base classifiers using divisive data clustering. The Clustering Rule based Approach (CRA) clusters the original training data and builds a separate rule based model on the cluster wise data. The individual models are combined into a larger model and evaluated against test data. We evaluate the effects of the missing attribute information for ordered and unordered rule sets. We experimentally show that the collective model is less affected by missing attribute information when the test data has missing attribute values.

Acknowledgements

First and foremost, I would like to thank God for he is my strength and through him I can do a1l things. I would like to thank Dr. Juan Gilbert for all his support, encouragement, and patience. The many times when I did not know how I was going to make it, or didn't think I could do it, he was there to guide me. Thank you to my family for their love, support, and encouragement. They have been very supportive an encouraging throughout my academic tenure here at Auburn. Last but not least I would like to say thank you to all of the members of the Human Centered Computing Lab. There support, encouragement, and just being there will forever be appreciated!!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Advances in data collection and storage capabilities have enabled organizations, companies, institutions, etc to collect immense amounts of data. The challenge lies in trying to turn all of that raw data into useful information. One way to address this problem is with the use of data mining.

Data Mining can be defined as "sorting through data to identify patterns and establish relationships" (TechTarget, 2008). Data mining blends traditional data analysis methods with sophisticated algorithms for processing large amounts of data. These algorithms search large data repositories in order to find new and useful patterns that might have otherwise been unidentified. Data mining methods can and are being used in many domains to address a variety of tasks.

One of the more popular uses of data mining techniques is predictive modeling or classification. Certain types (i.e. decision trees, neural networks, etc.) of data mining algorithms have the ability to predict future outcomes using the attributes in the data set. For example, let's assume that a long distance company wants to promote its new long distance plan. The company can apply predictive modeling algorithms to specifically target customers who are more likely to sign up for the plan based on their past behavior.

As one could imagine, the accuracy of predictive modeling algorithms is very important. Their results contain information that can be very valuable and have the ability to help make decisions and drive change. Rule based algorithms are widely used and accepted to perform classification of tasks due to their ease of interpretability and understanding. These classifiers build a model or rule set from training data as a set of high-quality rules, which can be used to predict the class labels of unlabeled instances (Wang & Karypis, 2003). There have been many rule based classification systems studied and proposed which yield adequate classification accuracy in a variety of applications such as RIPPER (Cohen, 1995), FOIL (Quinlan & Gaetz, 1993), and CPAR (Yin & Han, 2003) . However, if the test data used to evaluate the model is not as complete as the training data used to build the model the classification accuracy suffers as the model tends to follow the training data too closely. This is problematic as in most real world applications data is often incorrect and or missing parts. Thus, it is advantageous to have a rule set or model that is robust and can make reasonably accurate predictions when the test data is incomplete. This research explores the notion of adding robustness to rule based classifiers via divisive clustering.

1.2 Problem Description and Background

Rule based classifiers are used widely because of the ease of interpretability of the models or set of rules they generate. These classifiers perform exceptionally well on complete data sets. Meaning, the data is clean, correct, and does not have missing attribute values. To generate the model or train the classifier the training data uses attribute and values relative to each other to segregate the data to generate a rule relative to a particular class. This poses a problem as real world data sets are not clean, correct and often have missing attribute values. The models produced by traditional rule based classifiers are sensitive to missing attribute values

in new/unseen data. For example, Table 1 and Figure 1 depict a popular data set and the resulting decision tree (Quinlan J. , 1993).

Day	Outlook	Temperature	Humidity	Wind	Play Tennis?
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	Normal	Weak	Yes
D5	Rain	Cool	High	Strong	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Strong	Yes
D9	Sunny	Cool	Normal	Weak	No
D10	Rain	Mild	Normal	Weak	No

Table 1Tennis Training Data

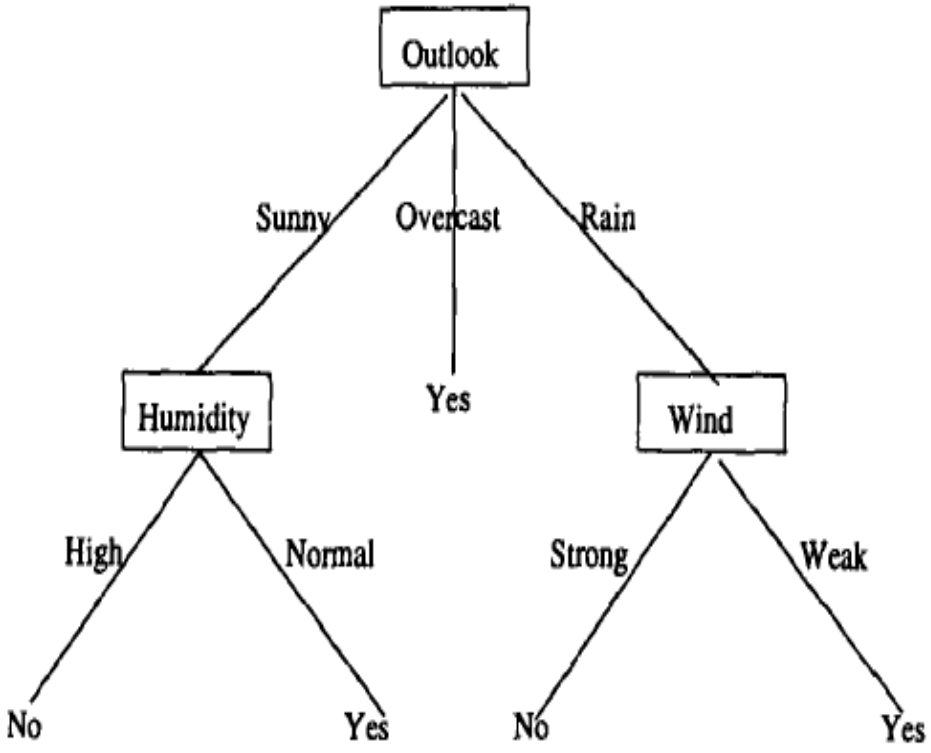


Figure 1 ID3 Decision Tree

The rules derived from this tree are:

If outlook is sunny and humidity is high, then do not play tennis.
If outlook is sunny and humidity is normal, then play tennis.
If outlook is overcast, then play tennis.
If outlook is rain and wind is strong, then do not play tennis.
If outlook is rain and wind is weak, then play tennis.

Notice that all of the rules derived contain the outlook attribute. Suppose there is some test data in which the outlook information is missing. This model is not robust as none of the rules generated would be able to make predictions. In fact, as the number of attributes within a data set that have missing data increases the classifiers performance/accuracy decreases. The model that the classifier generates is unable to classify all of the instances new/unseen test data. For the purposes of this research we call this a hard classification problem.

This research proposes a new method for adding robustness to rule based classifiers using divisive clustering for hard classification problems. With this approach the training data set is broken into clusters based on holistic similarities such that items within a particular cluster are more alike than those outside of the cluster. Each cluster generates a model trained with smaller homogeneous data sets. Multiple models are created using the clustered data and then combined into a single model (the experiment). The hypothesis for this research is that the resulting combined rule base will be more resilient against missing attribute information. Resiliency refers to minimizing any decreases in classification accuracy as the number of missing attributes increases versus the traditional (the control) model. It is believed that the collective model is more resilient than the traditional model because it will have an added level of robustness and ability to classify new/unseen data instances that would not have been classified with a traditional classifier. The collective model contains rules that otherwise would not have been

created because of the alternate views of the data set used to train the multiple individual classifiers.

1.3 Organization

In the following chapters the research agenda will be examined. Chapter 2 provides an overview of data mining techniques and practices, more specifically, predictive modeling/classification and data clustering, and discusses any work pertinent to this research. Chapter 3 provides conceptual view of the approach studied in this research. Chapters 4 and 5 describe the design of the experiment conducted to study the validity of the approach and the results of that experiment respectively. Finally, Chapter 6 provides a summary and conclusions and discusses areas of future work.

Chapter 2

Literature Review

2.1 Data Mining Overview

Data Mining is the process of discovering useful information in large amounts of data. It is fundamentally built on the principles of algorithms used in statistics, artificial intelligence, pattern recognition, and machine learning. From statistics, data mining incorporates characteristics such as sampling, estimation, and hypothesis testing. From artificial intelligence, pattern recognition, and machine learning, data mining incorporates the following: search algorithms, modeling techniques, and learning theories. Figure 2 conveys data mining's relationship to other areas (Tan, Steinbach, & Kumar, 2006)

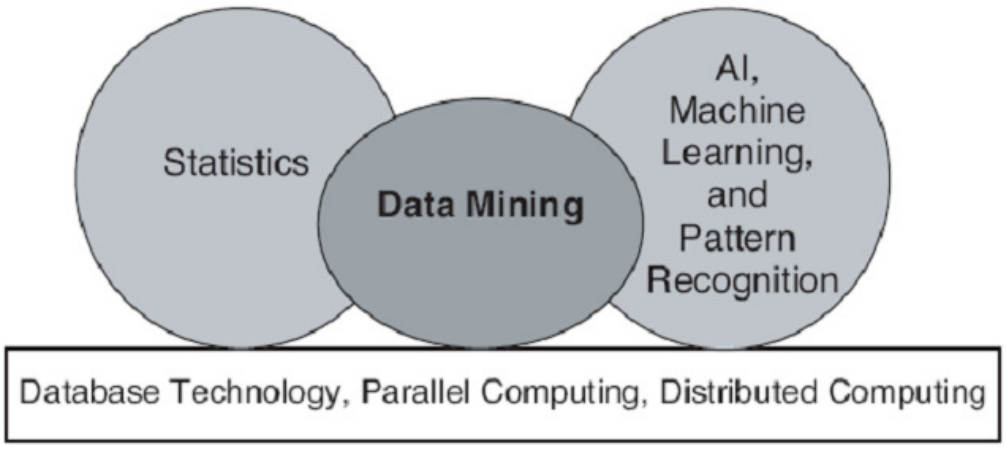


Figure 2 Data mining's relationship to other disciplines

Data mining is being used in many domains in a variety of ways. Due to this, a CRross Industry Standard for Data Mining Projects (CRISP-DM) was developed. This standard is tool, industry, and application neutral. The life cycle according to the standard has six phases:

Business/Research Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. Figure 3 depicts CRISP-DM (CRISP-DM, 2000).

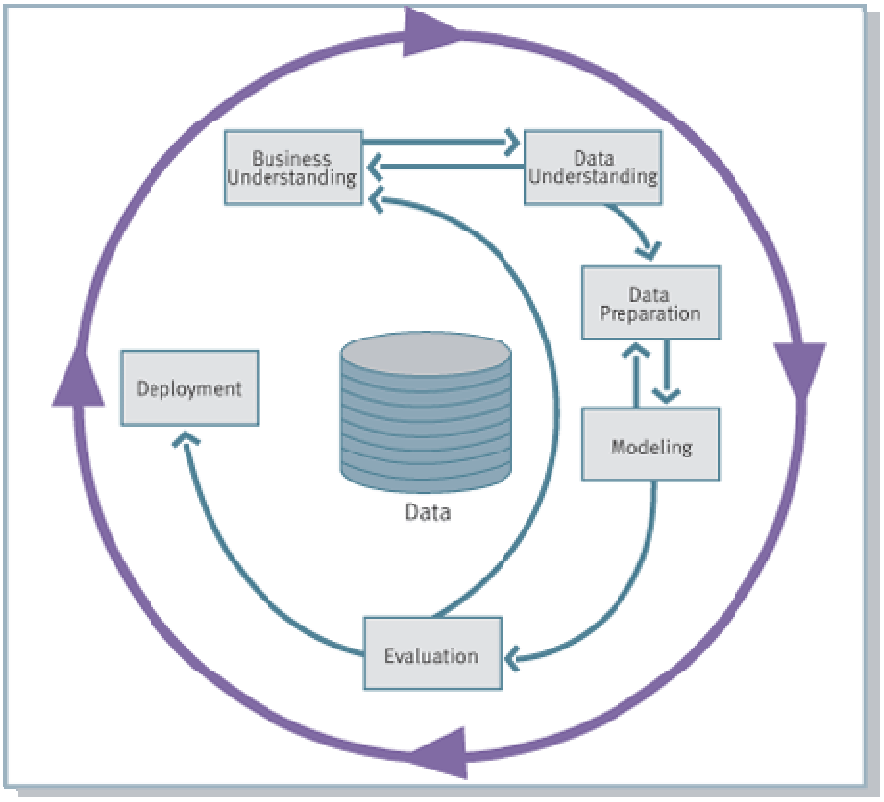
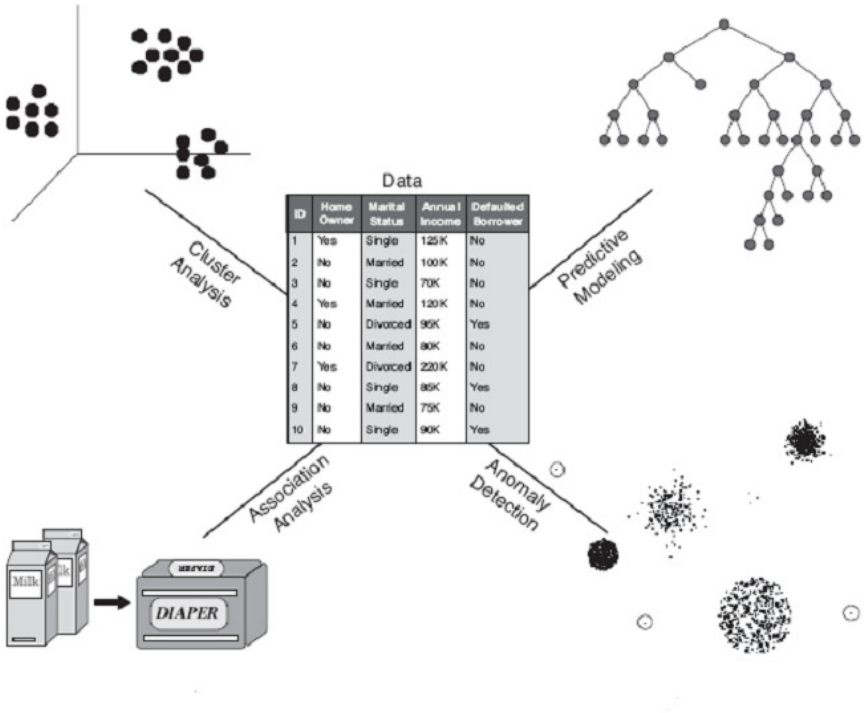


Figure 3 CRISP-DM Model

Most data mining tasks are usually either predictive or descriptive in nature. Predictive tasks are ones in which the goal is to predict the value of a particular attribute based on the values of other attributes. The attribute that is being predicted is frequently referred to as the target or dependent variable. The other attributes that are used for making the prediction are

referred to as the explanatory or independent variables. Descriptive tasks are ones in which the goal is to derive patterns (i.e. correlations, clusters, trends) that will provide a summary of the underlying relationships in the data. These tasks are usually exploratory and often require some post-processing to confirm as well as explain the results.

There are four tasks that are at the heart of data mining. They are Predictive Modeling, Cluster Analysis, Association Analysis, and Anomaly Detection (see Figure 4) (Tan, Steinbach, & Kumar, 2006)



ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	80K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 4 Four Core Data Mining Tasks

2.2 Predictive Modeling

Predictive modeling is one of the most popular subfields in data mining. It is the process of using the patterns found in the data set to predict future outcomes. Algorithms used for

predictive modeling build a model for the dependent variable as a function of the independent variables (Berson, Smith, & Thearling, 1999). There are two types of predictive modeling tasks: classification and regression. Classification tasks are used when the target or dependent variable is discrete. Regression tasks are used when the target variable is continuous (Tan, Steinbach, & Kumar, 2006)

Predictive modeling problems are comprised of four things: a dependent variable, independent variables, a learning/training data set, and a test data set (Lewis, 2000). The learning/training data set contains values for both the dependent and independent variables, and is used to build the model. This model is then applied to the test set for evaluation. The test set is a subset of the training/learning data set. The performance of the model is based on the counts of the test records that are correctly or incorrectly predicted or classified.

There are several predictive modeling techniques. The next section provides an overview of some of them.

2.2.1 Rule based Classifiers

A rule-based classifier is a method used for classification using a set of "if…then" rules. A rule can be expressed as r_i: (*Condition*) → y_i. The left hand side of the rule is the antecedent or condition. The right hand side is the consequent. A rule covers an instance if the condition matches the attributes of that instance. An example of classification using a rule based method is depicted below. Figure 5 shows the training data, Figure 6 displays the rule set (model) generated and Table 2 contains the unlabeled instances for the vertebrate classification problem. The goal of this model is to determine what type of vertebrate the instance will be classified as. There are five possible outcomes: mammal, reptile, fishes, amphibians, and birds. (Tan, Steinbach, & Kumar, 2006)

R1 covers the hawk instance. A hawk does not give birth and can fly and will be classified as a bird. R3 covers the grizzly bear instance. A grizzly bear gives birth and is warm blooded and will be classified as a mammal.

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

Figure 5 Vertebrate Training Data (Labeled/Known Outcomes)

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Figure 6 Vertebrate Rule Set/Model

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

Table 2 Vertebrate Unlabeled/Unseen Data

Rule based classification algorithms tend to fall into the two categories: decision trees and covering algorithms (Li, Topor, & Shen, 2002) (Hu & Li, 2005) (Tan, Steinbach, & Kumar, 2006) (Witten & Frank, 2005).

2.2.1.1 Decision Trees

A decision tree is a predictive model in which the results are displayed as a tree type structure (Berson, Smith, & Thearling, 1999). A decision tree consists of a collection of decision nodes, which are connected by branches, descending from the root node until coming to an end at the leaf nodes. Each branch of the tree is a classification question and the leaves are the partitions or segments of the dataset with the classification or decision (Larose, 2005) (Quinlan J. R., 1986).

The first step in the decision tree building process is to "grow" the tree. The goal is to create a tree that works as close to perfect as possible with the data provided. When growing the tree, the main task is to find the best question to ask at each branch or split point in the tree. This task is repeated until there is either only one record in the segment, each of the records in the segment are the same, or there is not any significant gain in making a split. When one of these conditions are met the tree stops growing (Berson, Smith, & Thearling, 1999). Figure 7 conveys a simple decision tree example (DMS, 2008).

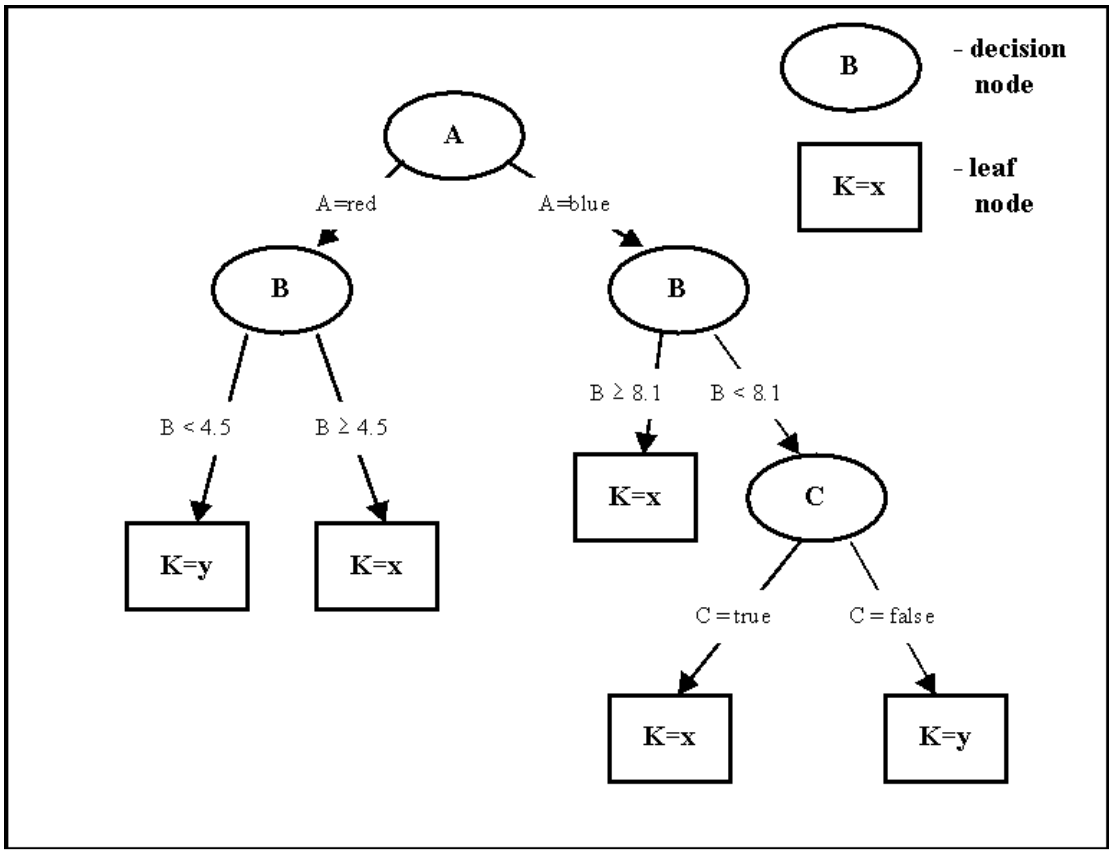


Figure 7 Simple Decision Tree

There are certain requirements that must be adhered to in order to apply a decision tree algorithm. A training data set must be provided. This training data should be varied in nature and provide the algorithm with a robust sampling of the types of records that will need classification in the future. This is very important as decision trees learn by example. If the algorithm is provided bad data then the ability to correctly classify future data will greatly diminish. Also, the target variable must be discrete (Larose, 2005).

Decision trees are a part of data mining technology that has existed in some form for decades. Originally, these techniques were created for statisticians to automate the process of determining which attributes were useful or in correlation with the problem they were attempting

to solve or understand. They are also particularly adept at handling raw data with little or no pre-processing (Berson, Smith, & Thearling, 1999).

Perhaps the most appealing aspect of decision trees is how easy they are to interpret. This is especially evident when deriving decision rules. Rules can be derived by simply traversing the tree from the root to any leaf node and come in the form of *if antecedent, then consequent*. The antecedent contains the attribute values from the branches taken by a particular path throughout the tree. The consequent contains the classification value for the dependent variable given by that particular leaf node (Larose, 2005) (Tan, Steinbach, & Kumar, 2006) (Witten & Frank, 2005). Using the simple decision tree in Figure 7 an example of a decision rule would be if A = red and B < 4.5 then K = y. There are several algorithms that are used to produce decision trees. These include: ID3, C4.5, Classification and Regression Trees (CART), and Chi-Square Automatic Interaction Detector (CHAID). The next sections describe each.

ID3

Developed in the 1970's, ID3 was one of the first decision tree algorithms. Initially, the algorithm was used for things like learning good game play strategies for chess end games. In chess, the end game is the portion of the game where there are only a few pieces left on the board (Quinlan & Gaetz, 1993) (Chess Endgame, 2008). However, ID3 has been used for a variety of tasks in various domains and has been modified and improved many times.

ID3 works by choosing the predictors and their corresponding split values based on the gain in information that the split(s) will make available. Gain is the difference in the amount of information that will be needed to properly make an accurate prediction before and after a split(s) is made (Berson, Smith, & Thearling, 1999).

C4.5

C4.5 is an enhancement of the ID3 algorithm. It improved ID3 algorithm by allowing predictors with missing or continuous values to be used, introducing tree pruning, and enabling rule derivation. The trees produced by this algorithm are also not restricted to binary splits. These trees are more variable in shape (Berson, Smith, & Thearling, 1999) (Larose, 2005) (Quinlan J. , 1993).

Classification and Regression Trees

Classification and Regression Trees (CART) was developed in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone (Breiman, Freidman, Olshen, & Stone, 1984). Many of the C4.5 techniques appear in CART. The trees that are produced by the CART algorithm are strictly binary. This means that each node in the tree can only have two branches. CART analysis is a form of recursive partitioning. The algorithm recursively partitions the data in the training set into subsets of records with similar values for the target or dependent variable. To grow the tree, CART performs an exhaustive search of available variables to pick the best or most optimal split variable. The fully grown tree that is initially produced will yield the lowest error rate when compared against the training data. However, this model may be too complex and usually results in overfitting. An overfit model is one that too closely follows all of the traits of the training data and is not general enough to represent the overall population (Berson, Smith, & Thearling, 1999) (Larose, 2005) (Lewis, 2000).

CART uses a cross validation approach and pruning to combat overfitting. Cross validation, also known as leave-one-out training, is a method for confirming a procedure for building a model that is computationally intensive (Lewis, 2000). In cross-validation, the training data is divided randomly into N segments, stratified by the variable of interest. One of the

segments is set aside for use as an independent test data set. The other (N-1) segments are combined for use as the training data set. The entire model building process is repeated N times, using a different segment of data as the test data each time. N different models are produced, each of which that can be tested against the independent test data. Based on the results of the cross-validation the initial complex tree is pruned to produce the most optimal general tree. The most complex tree rarely performs best on the test data. By using cross validation, the tree that is most likely to perform well on new data is produced (Lewis, 2000).

Chi-Square Automatic Interaction Detector

Chi-Square Automatic Interaction Detector (CHAID) was published in 1980 by Gordon V. Kass (Kass, 1980). CHAID is very similar to CART but differs in the way splits are determined. The algorithm uses the chi square test to choose the best split. The chi square test is used in contingency tables to determine which categorical predictor is the furthest from independence with the prediction values. All predictors must either be categorical or forced into a categorical form, because of CHAID's reliance on contingency tables, to shape its test of significance for each predictor (Berson, Smith, & Thearling, 1999) (CHAID).

2.2.1.2 Covering Algorithms

Rules can be extracted directly from the training data using a sequential covering algorithm. These algorithms learn a set of classification rules one rule at a time and use a sequential covering method to remove instances in the training data that are covered by each new rule found (Wang & Karypis, 2003) (Han & Kamber, 2006). This process repeats until there are no instances left in the training data set. There are many types of covering algorithms such as AQ15 (Michalski, Mozetic, Hong, & Lavrac, 1986), CN2 (Clark & Niblett, 1989), PRISM (Witten & Frank, 2005) and RIPPER (Cohen, 1995). A basic sequential covering algorithm is

depicted in Figure 8. The algorithm begins with an empty rule list. The best rule for class *y* that will cover the training records is mined. During this process all the training records that are covered by class *y* are deemed positive examples while those belonging to other classes are negative. A rule is of interest if it covers the majority of the positive and none or a minute few of the negative examples. The newly found rule is added to the bottom of the rule list and the training instances it covers are removed. This process repeats until some stopping criteria is met and proceeds to generate rules for the remaining classes (Tan, Steinbach, & Kumar, 2006).

Let E be the training records and A be the set of attribute-value pairs, $\{(A_j, v_j)\}$.
Let Y_o be an ordered set of classes $\{y_1, y_2, \ldots y_k\}$.
Let $R = \{\}$ be the initial rule list.
for each class $y \in Y_o - \{y_k\}$ **do**
    while stopping condition is not met **do**
        $r \leftarrow$ Learn-One-Rule (E, A, y).
        Remove training records from E that are covered by r.
        Add r to the bottom of the rule list: $R \rightarrow R \vee r$.
    end while
end for
Insert the default rule, $\{\} \rightarrow y_k$, to the bottom of the rule list R.

Figure 8 Basic Sequential Covering Algorithm

There are two commonly used methods to "grow" a rule: general to specific or specific to general. The general to specific strategy starts by creating a rule r: $\{\} \rightarrow y$. The left side of the rule is an empty set and the right side contains the target classification. This initial rule has poor quality as it covers all of the instances in the training data for that class. New conditions or conjuncts are added to improve the quality of the rule. For example, Figure 9 (a) depicts the general to specific rule growing method. The condition (conjunct), Body Temperature=warm-blooded is added initially to the rule. The algorithm will then explore all possibilities and greedily chooses the next condition to add. The specific to general method one of the positive examples is chosen at random as the seed to grow the rule. There is a refinement step in which

the rule is generalized by removing a condition (conjunct) to allow it to cover more positive examples. Figure 9(b) depicts this process (Tan, Steinbach, & Kumar, 2006).

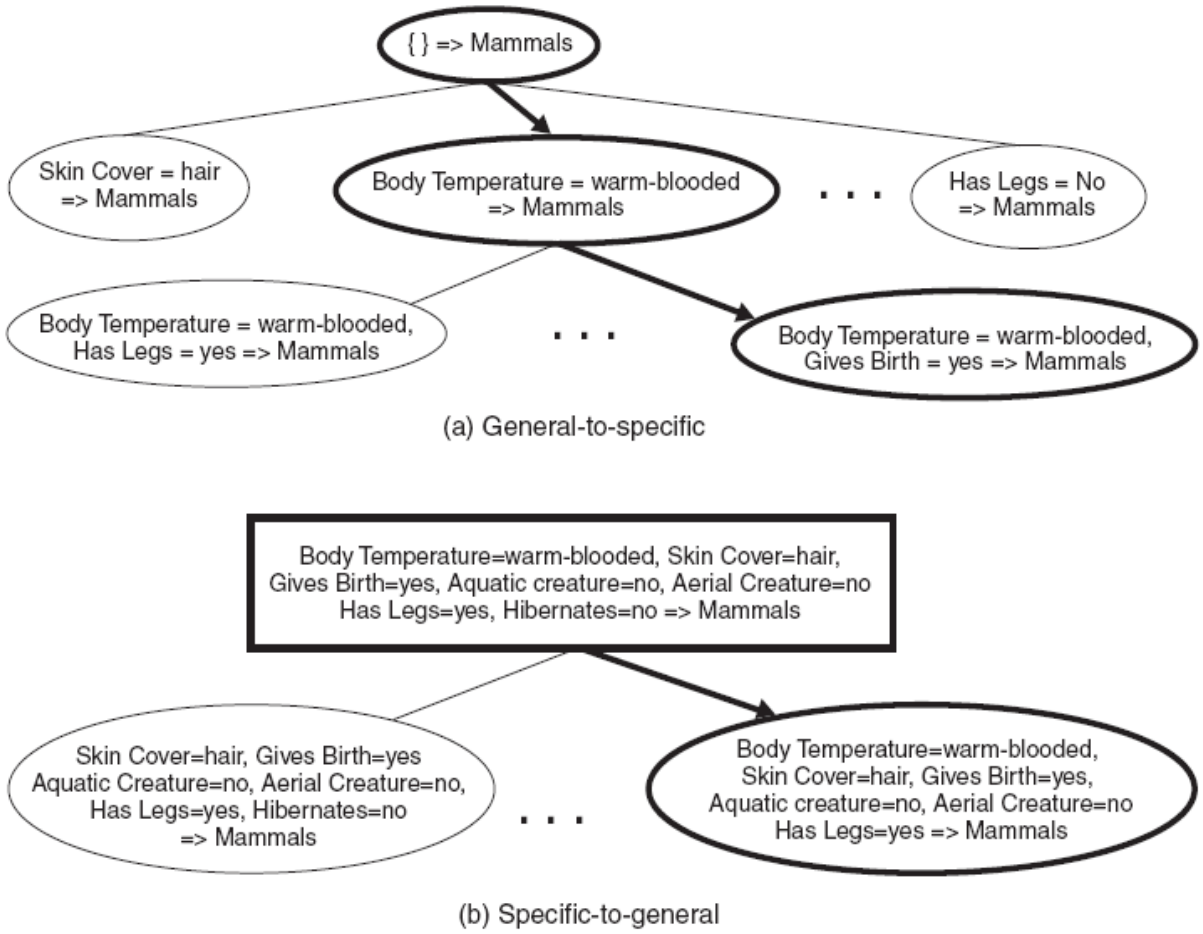


Figure 9 Rule Growing Strategies

Some evaluation metric (i.e. accuracy, coverage, etc.) is used to determine which conjuncts to add or remove to improve rule quality during the process of growing a rule.

2.2.1.3 Ordered and Unordered Rule Sets

Rule based-classifier rule sets are either ordered or unordered. The rules within an ordered rule set are arranged in an order based on some metric (i.e. accuracy) in decreasing order of

priority. When there is a new unlabeled instance, the first rule that matches makes the prediction. An ordered rule set is also known as a decision list. These classifiers usually have a "default class" which is used when no classification can be made (no rule matches) (Tan, Steinbach, & Kumar, 2006) (Hu & Li, 2005). C4.5rules (Quinlan J. , 1993) and CBA (Liu, Hsu, & Ma, 1998) both use this method.

Un-ordered models do not arrange the rules in any type of sequence or order. All of the rules that match are used to make the prediction (i.e. voting). The class that receives the most votes wins and is used as the prediction. The vote for each class may be weighted by the rules accuracy in some instances. There are both advantages and disadvantages when using an unordered rule base. Unordered rule sets are not as vulnerable to misclassifications as a result of choosing the wrong rule. Building the model is also less expensive as the rules do not have to be kept in any particular order. However, classifying a new test instance can be expensive as the all of the attributes of that instance must be compared against every rule in the model. (Tan, Steinbach, & Kumar, 2006) (Witten & Frank, 2005)

2.2.1 Other Classifiers

Naïve Bayesian Classifiers

Bayesian classifiers are based on the Bayes theorem. This theorem is a simple mathematical formula that is used for calculating conditional probabilities. The naïve Bayes classifier learns the conditional probability for each attribute given a particular class label. For example, if A and B are two random events then P (A|B) is the conditional probability of A occurring given B. Classification is performed by using the Bayes theorem to calculate the probability of a class given a particular instance. The class with the highest posterior probability is then used as the prediction. Using the previous example, P (A|B) is the posterior probability.

(Friedman & Goldszmidt, Building classifiers using Bayesian networks, 1996) (Langley & Sage, 1994) (Friedman, Geiger, & Goldszmidt, Bayesian Network Classifiers, 1997)

Neural Networks

An example of an authentic neural network would be the human brain. The brain can recognize patterns, make predictions, and learn. Artificial neural networks seek to emulate these capabilities. Neural networks in data mining are essentially computer programs applying pattern recognition and machine learning algorithms to build predictive models (Berson, Smith, & Thearling, 1999).

There are two main structures in a neural network: nodes and links. Nodes are artificial neurons and links are the connections between them. To make a prediction, the neural network accepts values for the independent variables or predictors at the input nodes. The values of these nodes are then multiplied by values stored in the links. These values are added together at the output node, after which some threshold function is used and the resulting number is the prediction. Figure 10 (Berson, Smith, & Thearling, 1999) is an example of a simple neural network. In this example Age and Income are the input nodes and Default is the output node and predicts if a person will default on a bank loan.

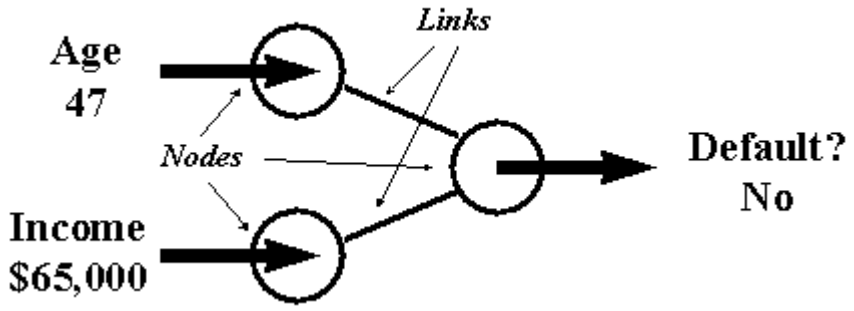


Figure 10 Simple Neural Network

Most neural networks are not as simple as depicted above. There is usually hidden layer of nodes between the input and output nodes. They are deemed "hidden" because their contents are not made known to the end user. It is also possible to have more than one hidden layer, thus making the network very complex.

2.3 Association Analysis

Association analysis is useful for finding interesting relationships that are hidden in large data sets. The goal of this type of analysis is to uncover rules (or associations) for quantifying the relationship between two or more attributes (Larose, 2005). These relationships are displayed in the form of an association rule. An association rule is an implication expression of the form: $X \rightarrow Y : \sup_X^Y, conf_X^Y$, where X and Y are disjoint itemsets (i.e., $X \cap Y = \emptyset$), $\sup_X^Y$ is the support, and $conf_X^Y$ is the confidence.

The strength or goodness a rule is measured by its support and confidence. Support defines how many times a rule is pertinent to a particular data set. Confidence defines how often item in Y appear in instances that contain X. For example, a store may find that out of 100 customers shopping on a Tuesday night, 20 bought diapers, and of the 20 who bought diapers, 5 bought beer. The association rule for this example would be if buy diapers then buy beer. This rule would have a support of 5/100 = 5% and a confidence of 5/20 = 25% (diapers → beer: 0.05, 0.25) (Tan, Steinbach, & Kumar, 2006)

Mining association rules from large data repositories involves a two step process: frequent itemset generation and rule generation. First, all of the frequent itemsets must be found. Frequent itemsets are those that satisfy some minimum support threshold. Then, from the frequent item sets found in the first step, association rules are generated that satisfy the minimum

support and confidence conditions. In association analysis an itemset is a collection of zero or more items.

Frequent itemset generation is generally more expensive than rule generation. A dataset that has *k* items could produce up to 2^k -1 itemsets. Because *k* can be quite large, the search space of itemsets that needs to be investigated is exponentially large. To enumerate the list of all possible itemsets a lattice structure can be used (Figure 11). The brute force method of finding frequent itemsets involves determining the support count for each candidate itemset in the lattice structure. There are ways of reducing the computational complexity of frequent itemset generation. One of these is to reduce the number of candidate itemsets using the Apriori principle. The Apriori principle states that if an itemset is frequent, then all of its subsets must also be frequent (Figure 12). On the other hand, if an itemset is infrequent then all of its supersets must be infrequent too. For example in Figure 13, because {a, b} is an infrequent itemset the entire sub graph containing the supersets {a, b} can be pruned. This method of reducing the exponential search space based on the support measure is known as support-based pruning. Apriori was the first rule mining algorithm which pioneered the use of support based pruning (Tan, Steinbach, & Kumar, 2006). This algorithm uses support based pruning to control the exponential growth of candidate itemsets. Apriori uses a bottom up technique in which the frequent item sets are extended one at a time. This is known as candidate itemset generation. This process continues until no further successful extensions are found (Larose, 2005).

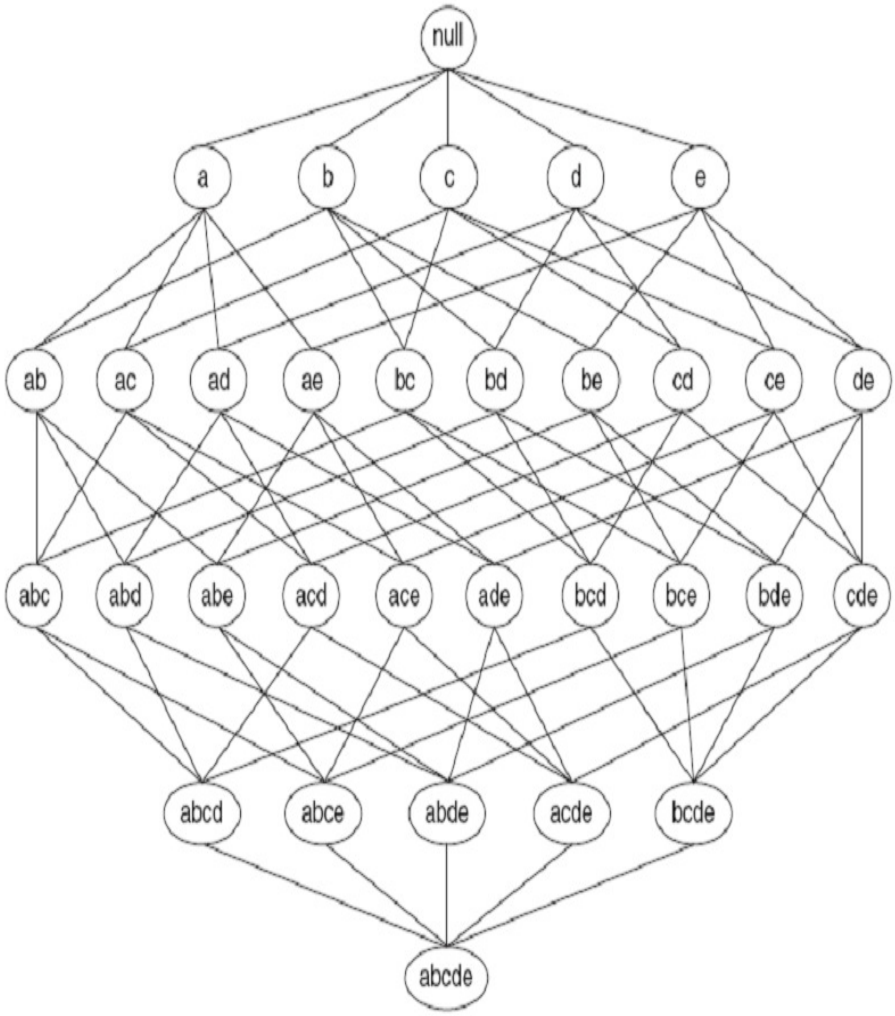


Figure 11 Lattice Structure

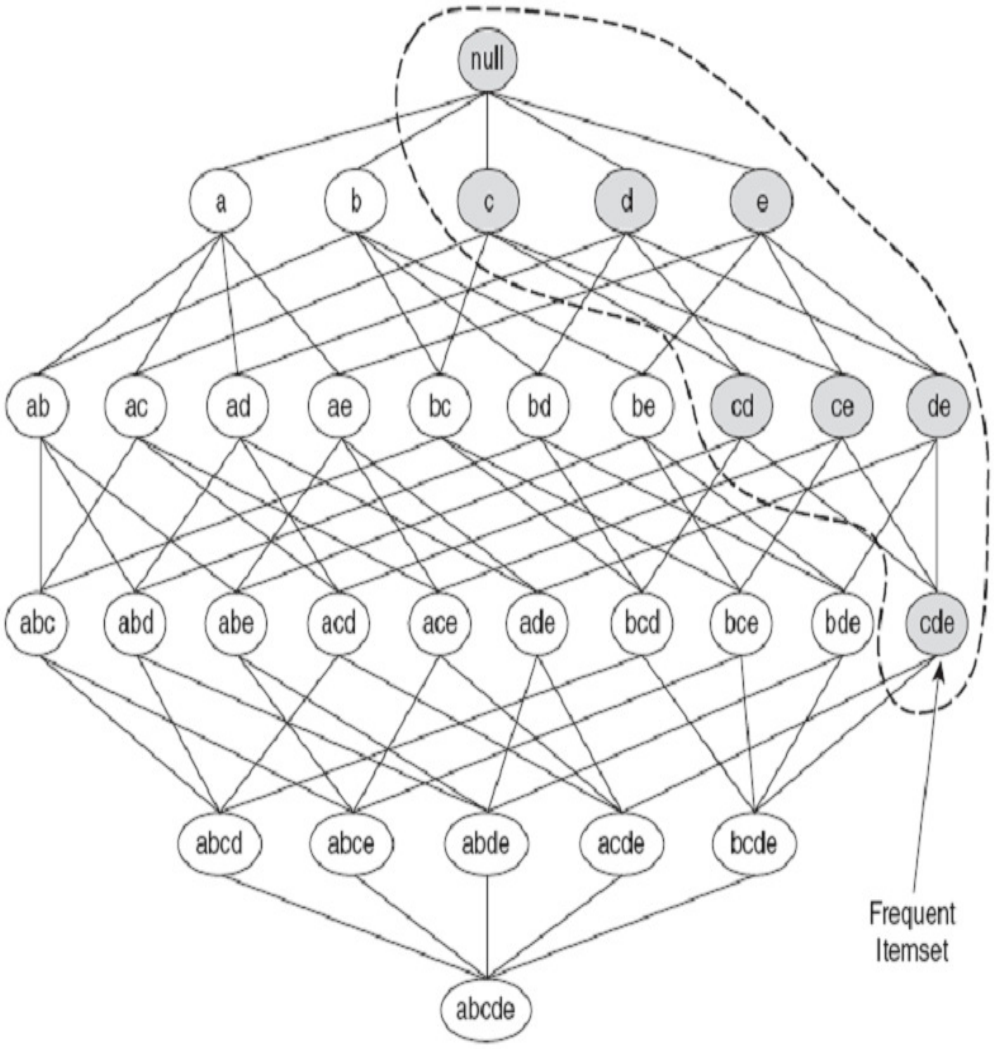


Figure 12 Apriori Principle

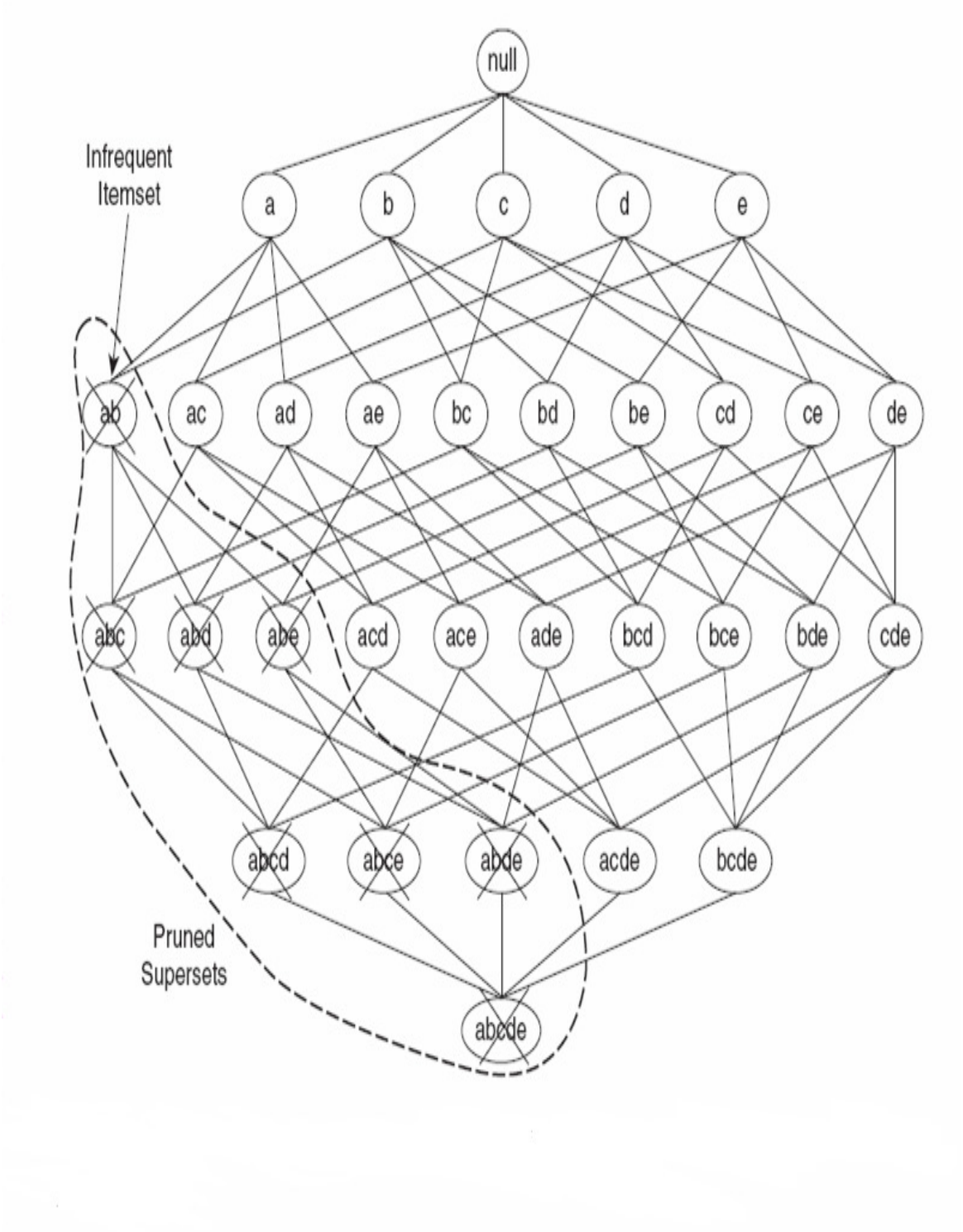


Figure 13 Support Based Pruning

2.4 Cluster Analysis

Clustering can be defined as "a division of data into groups of similar objects (Berkhin, 2002)". Instances within these groups or clusters are more similar to each other than instances belonging to other clusters. Clustering and classification are different from one another in that there is no target or dependent variable in clustering. Clustering does not attempt to classify or predict the value of the target variable. These algorithms attempt to segment the whole data set into homogenous clusters. The more similarity within a cluster and the bigger the difference between clusters the better the clustering. For the best possible performance, clustering algorithms require that the data be normalized so that any one attribute or variable will not control the analysis. (Jain, Murty, & Flynn) (Anderberg, 1973) There are two main types of clustering algorithms: hierarchical and partitional. The next section describes each.

2.4.1 Hierarchical Clustering

Hierarchical clustering creates a tree of clusters known as a dendrogram. With this type of clustering, the smallest clusters in the tree join together to create the next level of clusters. At this level, the clusters then join together to create the next level of clusters. The top or root of this tree is the cluster that contains all the records. There are two types of hierarchical clustering: agglomerative and divisive (Berkhin, 2002) (Berson, Smith, & Thearling, 1999).

Agglomerative clustering algorithms begin with having as many clusters as there are records. Each cluster will contain one record. Then the clusters that are closest to one another, based on some distance, are joined together to create the next largest cluster. This process is continued until the hierarchy is built with a single cluster, which contains all records, at the top. Figures 14 and 15 describe the agglomerative clustering process (Cluster Analysis, 2008).

Figure 14 contains a set of records that will be clustered. Figure 15 shows the cluster hierarchy after the agglomerative approach has been applied.

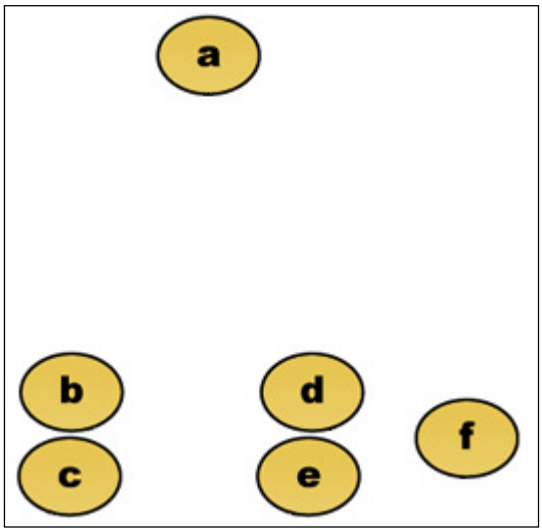


Figure 14 Records to be clustered

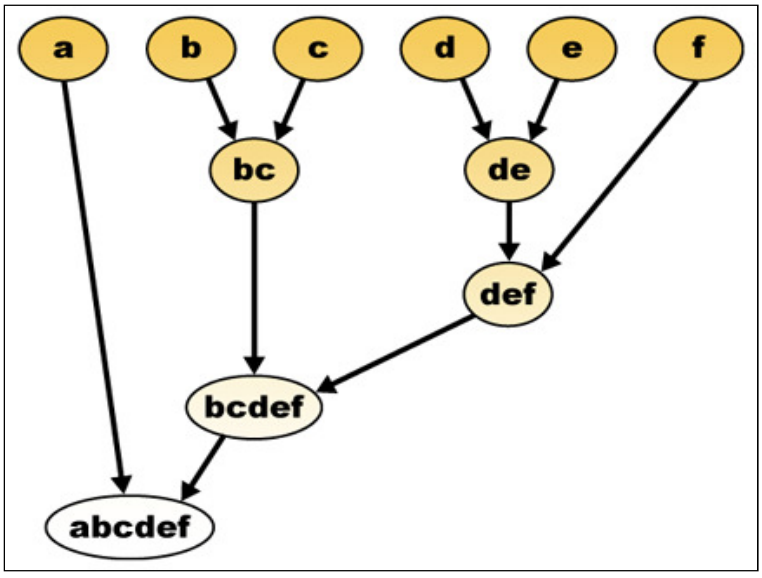


Figure 15 Cluster Hierarchy

Divisive clustering algorithms apply an opposite approach. These types of algorithms start will all of the records in one cluster and then recursively splits the most suitable cluster. This process continues until some stopping criteria are met.

How appropriate a cluster or clusters for merging or splitting depends on how similar or dissimilar items in each cluster are. The distance between individual records has to be generalized to the distance between clusters in order for splitting or merging to occur. This proximity measure is called a linkage metric. The major linkage metrics include: Single Linkage, Complete Linkage, and Average Linkage (see Figure 16) (Tan, Steinbach, & Kumar, 2006).

Single linkage (nearest neighbor) is based on the minimum distance between any record in one cluster and any record in another cluster. Cluster similarity is based on the similarity of the most similar members from each cluster.

Complete linkage (farthest neighbor) is based on the maximum distance of any record in one cluster and any record in another cluster. Cluster similarity is based on the similarity of the most dissimilar members from each cluster.

Average linkage was designed to decrease the dependence of the cluster linkage criteria on extreme values. The criteria here is the average distance of all the records in one cluster from all of the records in another cluster.

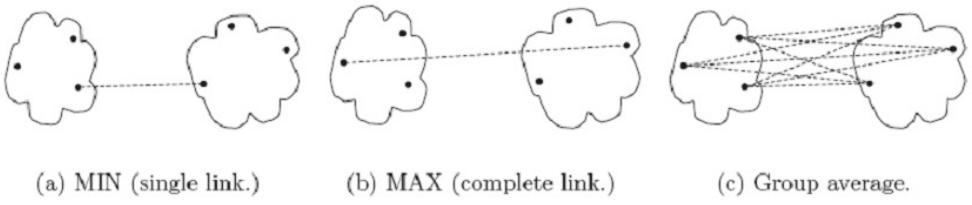


Figure 16 Linkage Examples

2.4.2 Partional Clustering

Partitional clustering is dividing the data set in such a way that each record belongs to one and only one cluster. These algorithms don't produce dendrogram as in hierarchical clustering. A single partition of the data is produced. Partitional clustering algorithms begin with a randomly picked or user defined number of clusters. The algorithms then optimize each cluster based on some validity measure. There are several partitioning clustering approaches. The next section discusses two of these: K-Means and Expectation Maximization (EM) (Berkhin, 2002).

K-Means

K-means is one of the oldest and most widely used clustering techniques. The name K-means comes from how each of the *K* clusters is represented by the mean of the points within that cluster. This point is called the centroid. The basic K-means algorithm is very simple and straight forward. Figure 17 describes this algorithm (Tan, Steinbach, & Kumar, 2006).

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

Figure 17 K-means Algorithm

The first step is to pick *K* initial centroids. *K* is the number of clusters wanted and is a user specified parameter. Next each data instance or point is assigned to the centroid it is closest to. This collection of points is a cluster. The centroid of each of these clusters is updated based on the points assigned to the cluster. These steps are repeated until there is no change in the centroids.

Instances are assigned to a centroid based on a proximity measure that quantifies closeness for a particular data set. There are several types of proximity measures such as Euclidean distance and Cosine similarity (Berkhin, 2002).

Expectation Maximization (EM)

The EM technique is very similar to the k-means approach. They are both iterative in nature and start with a random guess. The difference lies in the idea of hard versus soft cluster membership. In k-means a hard membership approach is adopted. This means that an instance belongs to one and only one cluster. In EM a soft membership approach is used, which means that membership of an instance can be spread amongst many clusters.

The EM algorithm is a two step process: Expectation and Maximization. The expectation portion is the first step and involves calculating cluster probabilities (the expected class values). The maximization step involves calculating the distribution parameters. This step is the maximization of the possibilities of the distribution given the data. These steps are repeated until a "log-likelihood convergence is achieved." (Berkhin, 2002) (Witten & Frank, 2005).

2.5 Anomaly Detection

The main objective in anomaly detection is to locate instances that are different from most of the other instances. These abnormal objects are known as outliers and have attribute values that deviate considerably from the norm or expected values. Some areas where this type

of analysis is very important are fraud detection, intrusion detection, public health and medicine. Some common causes of anomalies are things such as data from different types, natural variation and data measurement or collection errors (Tan, Steinbach, & Kumar, 2006)

There are three fundamental approaches to anomaly detection: supervised, unsupervised, and semi-supervised. The difference in these techniques is the degree to which known outcomes or classifications (class labels) are available for at least some portion of the data (Tan, Steinbach, & Kumar, 2006)

The supervised anomaly detection approach requires a training set that contains both normal and abnormal instances. Unsupervised anomaly detection techniques seek to assign a score to each instance that reflects how abnormal that instance is. With semi-supervised anomaly detection techniques the goal is to find an anomaly score (label) for a set or group of instances using information from labeled normal instances.

2.6 Ensemble Methods

As one could imagine the accuracy of predictive modeling algorithms is quite important. Ensemble methods seek to improve the accuracy of classifiers (such as decision trees) by combining the predictions of multiple classifiers (Figure 18). These methods construct a set of base classifiers using the training data and make classifications for new instances by voting on the prediction each base classifier makes. (Tan, Steinbach, &Kumar, 2006) Research has shown that ensembles have a tendency to perform better than single classifiers in the ensemble (Opitz & Macline, 1999) (Quinlan J. , 2006) (Freund & Schapire, 1999) (Berk, 2004). Bagging, boosting, and stacking are well known ensemble techniques.

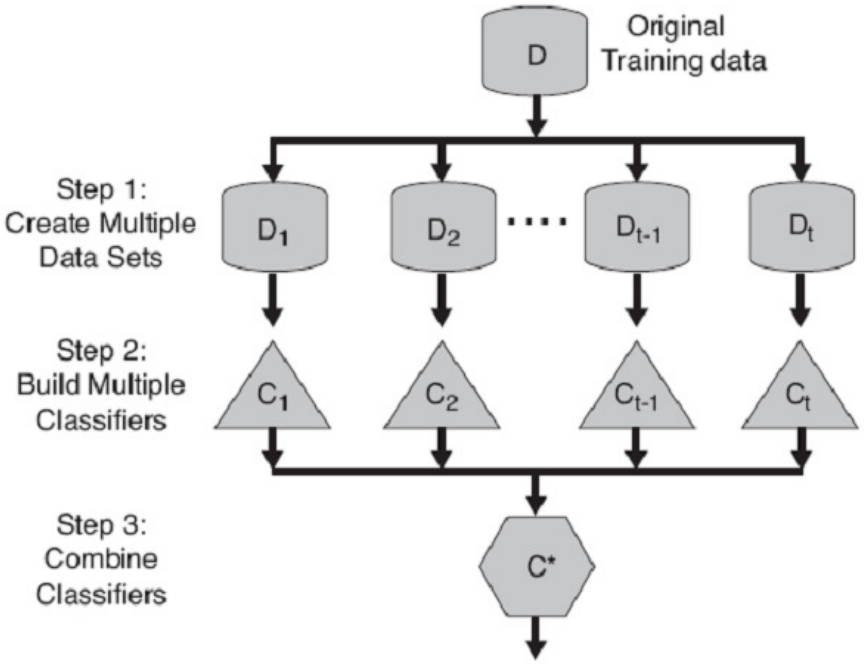


Figure 18 Ensemble Method Process

2.6.1 Bagging

Bagging or bootstrap aggregation trains each base classifier with a random redistribution of the training (Opitz & Macline, 1999). Each of the training sets for the base classifiers are generated by randomly selecting, with replacement, N examples, where N is the size of the original training set. Since sampling is done with replacement, some of the instances may possibly appear many times within the same training set. Likewise, some instances may be omitted from the training set (see Figure 19).

As previously stated, bagging involves combining multiple models (for example decision trees). These models can be combined by having each model vote on each new instance. The class that receives the most votes is deemed to be the correct one. When the target variable is numeric the average is used. The predictions or classifications made by voting become more

dependable as more votes are considered. Researchers have determined that bagging is effective on the learning algorithms of which small changes in the training data can produce big changes in predictions (Tan, Steinbach, & Kumar, 2006) (Witten & Frank, 2005) (Opitz & Macline, 1999).

2.6.2 Boosting

This ensemble method technique gets its name from its capacity to take a "weak learning algorithm" and "boosting" it into a "strong" learning algorithm (Freund & Schapire, 1999). A weak learning algorithm is one that performs slightly better than random guessing. Like bagging, boosting uses voting to combine the output of multiple models of the same type (for example decision trees). Boosting, however, is iterative, unlike bagging in which each base classifier is built separately; each new model is influenced by the performance of the models built before it. New models are pushed to become experts for instances mishandled by earlier models. Boosting weights a model's contribution by its performance instead of giving equal weight to all as in bagging (Witten & Frank, 2005). For example in Figure 19 (Opitz & Macline, 1999), assume instance 1 is an outlier and hard to classify correctly. This instance appears more in later training sets because boosting will focus (increase its weight) more on correctly classifying it.

A sample of a single classifier on an imaginary set of data.	
	(Original) Training Set
Training-set-1:	**1**, 2, 3, 4, 5, 6, 7, 8

A sample of Bagging on the same data.	
	(Resampled) Training Set
Training-set-**1**:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	7, 8, 5, 6, 4, 2, 7, 1
Training-set-3:	3, 6, 2, 7, 5, 6, 2, 2
Training-set-4:	4, 5, 1, 4, 6, 4, 3, 8

A sample of Boosting on the same data.	
	(Resampled) Training Set
Training-set-**1**:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	**1**, 4, 5, 4, 1, 5, 6, 4
Training-set-3:	7, 1, 5, 8, 1, 8, **1**, 4
Training-set-4:	**1**, 1, 6, **1**, 1, 3, **1**, 5

Figure 19 Bagging and Boosting Example

2.6.3 Stacking

Stacking or Stacked Generalization differs from bagging and boosting in that it involves combining classifiers of different types (e.g., decision trees and neural networks). It is less commonly used because it can be difficult to analyze and there is no standard or accepted best practice. Stacking eliminates voting by introducing the idea of a metalearner. Stacking attempts to "learn" which base classifiers are the most reliable by using the metalearner to determine how best to combine the output of the base classifiers (Witten & Frank, 2005).

The inputs to the metalearner (level 1 model) are the predictions of the base classifiers (level 0 model). Each level 1 instance will have as many attributes as there are level 0 learners.

When classifying, the new instance is given to each of the level 0 learners and those results are given to the level 1 learner. The level 1 learner then "learns" the best way to combine the predictions to make the final prediction (Witten & Frank, 2005).

2.7 Other Related Information and Work

Discretization

Many data of the data mining algorithms used for classification tasks can't tolerate numeric data. For these algorithms the data must be transformed into a categorical format. This transformation is known as discretization. Discretization has two steps:

- Decide how many categories to create
- Determine how to map the values to the categories

There are many ways to discretize continuous attributes (Dougherty, Kohavi, & Sahami, 1995). One of the simplest ways is to make use of the standard deviation to create categories. For example, suppose attribute *A* is a nominal attribute. First obtain the minimum, maximum and standard deviation values for *A*. Let min=1, max=5 and stddev=0.25. The initial category would be 1 –1.25 (min <= *A* < (min + stddev)). Any values greater than or equal to 1 and less than 1.25 would be updated to reflect this category in the dataset. The next category would be 1.25 – 1.50 ((min + stddev) <= *A* < (min + stddev + stddev)). Again, any values that fall within this range would be reflected in the dataset. This process continues until the range is less than or equal the max value.

Association Based Classification

There have been efforts made to combine association and traditional rule mining techniques. (Liu, Hsu, & Ma, 1998), showed that this integration could be performed efficiently without loss in classification accuracy. The combination focuses on the association rules where

the right side contains the target class attribute. These rules are identified as class association rules. Data mining with this associative classification framework has 3 basic steps:

- Discretize any continuous attributes
- Generate all class association rules
- Build classifier bases on the rules generated

All association based algorithms e.g. Apriori (Agrawal & Srikant, 1994), can be adapted to use for classification purposes. CBA (Liu, Hsu, & Ma, 1998) , CMAR (Li, Han, & Pei, 2001), and HARMONY (Wang & Karypis, 2003) are examples of applications that utilize some form of this technique. (Hu & Li, 2005)

Data Clustering and Predictive Modeling

For some classification or regression problems it is often advantageous to divide the data into relatively homogenous groups or clusters and build a model for each group. The advantages of building multiple models in this manner are improved accuracy, reliability and interpretability. (Baumann & Germond, 1993), (Djukanovic, Babic, Sobajic, & Pao, 1993), (Oh & Han, 2001) and (Lokmic & Smith, 2000) all explored this two step method in their research.

(Deodhar & Ghosh, 2007), investigated notion of infusing data clustering process with the creation of predictive models. Their work focused on problems in which the independent variables could be naturally sectioned into two or more groups and makes use of co-clustering. The authors partitioned the independent variables into groups with respect to their modes then instantaneously clustered along each mode and built a predictive model for each co-cluster. Co-clustering is a method that simultaneously clusters data along multiple axes. (Cheng & Church, 2000) (Cho, Dhillon, Guan, & Sra, 2004) This technique is usually applied to a matrix of data points and takes advantage of the duality between two axes to improve upon traditional single sided clustering. The rows in the matrix are data points and the columns are features.

The aforementioned approaches are all methods that have successfully combined some aspect of data clustering and predictive modeling/classification. However, none of them address the issue of attempting to make classifications when the data that is used to evaluate the model is not as complete as the data used to build the model.

Robust Rule based Classification

(Hu & Li, 2005) and (Li, Topor, & Shen, 2002) focus on building a more robust classifier. The authors proposed using association rules as a way to make rule base classifiers more robust. The idea is that a traditional rule based classifier can be improved by adding a certain level of redundancy to the rule set. This robust classifier contains more rules than the generalized model and combats possible classification error due to missing and/or incomplete data in the new/unseen instances. In terms of missing values, robustness involves the following:

- Ability to tolerate missing values in test data
- Ability to tolerate missing values in training data

Missing data is a well known issue in the field of data mining. There has been research performed in regards to handling missing values in training data such as work presented by (Clark & Niblett, 1989), (Mingers, 1989) and (Batista & Monard, 2003). General methods for managing missing values involve pre-process substitution via estimations utilizing an approach such as nearest neighbors. The work presented by (Hu & Li, 2005) and (Li, Topor, & Shen, 2002) differs from these existing methods in that no estimations or substitutions are made for any missing values. This research employs a larger rule set to make the classifier more resilient against missing test data. The authors designed, implemented and evaluated a practical robust rule based classifier for 0, 1, 2, 3 and 4 missing attributes using an ordered rule set using association classification rules. The rules were ordered by rule accuracy in descending order. A

rule set was generated using the training data and evaluated against test data with added missing values. The missing values were added by randomly omitting some values in the test data. Each test record in the test data had the same number of missing attributes on average. The results of the experiment were positive and proved that their proposed method of adding redundancy via association rules was indeed more resilient against missing attributes in the test data.

The work by (Hu & Li, 2005) is similar to the research presented in this dissertation as both are focused on creating models that are a resilient against missing attribute information. However, this research differs in that we present a method to add resiliency via divisive clustering and rule based models created using a sequential covering algorithm. Multiple rule sets are generated, combined and used as a collective model. The creation of multiple models on different segments of the data allows for the creation of rules that may not have been created using a more traditional approach which aids in adding resiliency against missing data.

2.8 Data Mining Tools

There are many data mining tools available. This section gives a brief description of some of them.

Microsoft SQL Server

Microsoft SQL Server is a "comprehensive, integrated data management and analysis software that enables organizations to reliably manage mission-critical information and confidently run today's increasingly complex business applications (Microsoft, 2007)". SQL Server 2005 is the platform leader in a variety of areas including business intelligence and database management systems.

Rapid Miner

"Rapid Miner (formerly YALE) is the world-leading open-source system for knowledge discovery and data mining (Rapid-i, 2008)". This tool is not only available as a stand-alone application for data analysis but also as a data mining engine that can be integrated into your own products.

Weka

Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software that is written in Java. The software was developed at the University of Waikato and is free under the GNU General Public License (Witten & Frank, 2005).

ORACLE Data Mining

Oracle Data Mining (ODM) is an option of Oracle Database 10*g* Enterprise Edition. This tool provides the ability to "produce actionable predictive information and build integrated business intelligence applications (ORACLE, 2007)".

Chapter 3

Research Details

3.1 Conceptual Overview

This section describes the approach taken to test the hypothesis discussed in section 1.2. The objective of this research is to develop a new method for classification problems in which the test data may not be as complete as data used to build a model that is still able to achieve fairly accurate results. The problem is more formally expressed here.

Let $A = \{a_1, a_2, \ldots, a_m\}$ be a set of distinct attributes found in a table, D, where $D = \{d_1, d_2, \ldots, d_n\}$ indicates a set of all instances within a table. Attributes capture the basic characteristics of an instance. Each attribute a_i contains a set of distinct values $V_i = \{v_1, v_2, \ldots, v_n\}$. A pattern is a set of one or more attribute value equivalent constraints that are conjunctively joined ($a_i = v_{ij}$ and $a_{t=}v_{tk\ldots}$). A rule, r, is an implication in the form $P \rightarrow c$, where P is a pattern *(the condition)* and c *(the consequence)*, such that c is a member of A where $c = a_k$ and c is not a member of P. A traditional rule based classifier would build a rule set $R = \{r_1, r_2, \ldots, r_n\}$.Given a test instance T, a rule r covers T if *condition(r)* $\subseteq T$. A rule can make predictions on the instances it covers, $r(T) \rightarrow$ *consequence(r)*. If the *consequence(r)* is the actual class of T then the rule has made the correct prediction. If the class is not the actual class it is an incorrect prediction. Given a test instance T', with one or more missing attribute values from A, where V' (unknown attribute values) is not a subset of V (known attribute values), denoted $V' \not\subset V$; r cannot make predictions for T'. Therefore T' is unclassifiable by R.

Using the proposed approach, multiple rule sets would be generated via divisive data clustering. A cluster, $clust_j$ ($1 \leq j \leq k$), where $clust_j \subseteq D$, such that each instance is a member of one and only one cluster, $Clust = \{clust_1, clust_2,\ldots,clust_k\}$,where k is the designated number of initial clusters. An instance is assigned to a cluster, $clust_j$, based on holistic similarities such that instances within a cluster are more similar than those outside the cluster. A rule set, R_j, is generated for each cluster. The multiple rule sets are combined into a super rule set, $R' = \{R_1 \cup R_2 \ldots \cup R_k\}$. R' is more resilient against missing attribute information as there exists a k of which the attributes in R' are sufficiently diverse. Meaning, that there exists two or more rule sets within R' that are not proper subsets with respect to their individual As, where ($1 \leq j \leq k$ and $1 \leq i \leq k$ and $i \neq j$) of other rule sets within R'.

Clustering Rule Based Approach

As previously mentioned the premise of this research is to add robustness and resiliency to rule based classification via divisive clustering. The idea is that there exists a k that will create a model that will be more resilient against missing attributes. We present the Clustering Rule based Approach (CRA) for classification. In CRA, the data is first clustered into k clusters using divisive clustering. The clustered data is then used to train individual rule base classifiers using a sequential covering algorithm. The rules are then merged into a combined rule set. Finally, each test instances is evaluated against the combined rule set, predicting its output. Figure 20 depicts this process. It is important to note that prior to the cluster creation any continuous attributes are discretized.

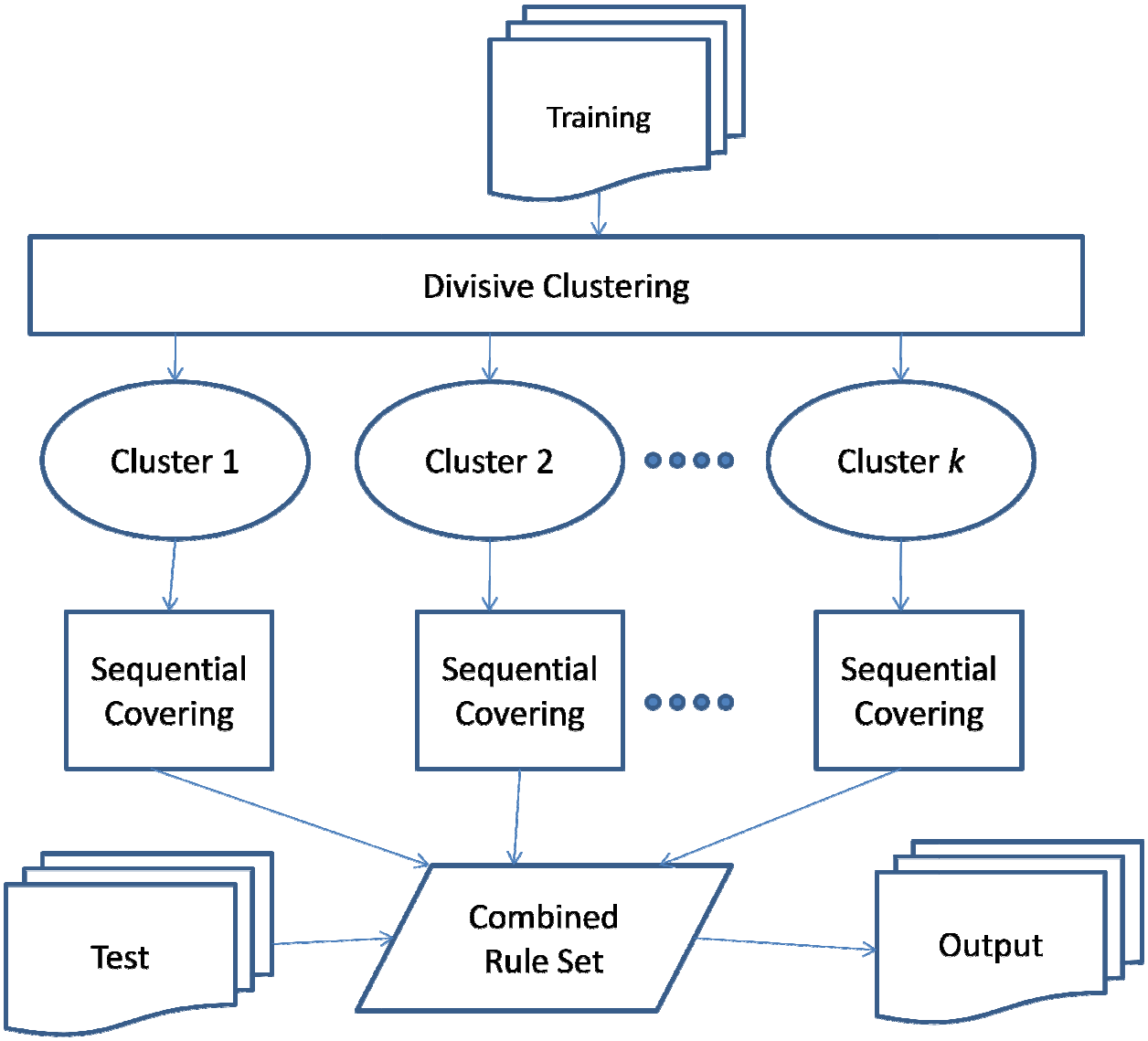


Figure 20 CRA Flow

Chapter 4

Experiment Design

We conducted an experiment to test the validity of the CRA process. This section describes the design of that experiment.

4.1 Data

The data for this experiment was obtained from the UCI Repository (Blake & Merz, 1998). These data sets varied in size, attributes, and attribute value data types (e.g. continuous, nominal). Table 3 gives a description of each data set used.

Data Set	Number of Records	Number of Attributes	Number of Classifications
Annealing	898	38	5
Thyroid	3,164	25	2
Mushroom	8,124	22	2
Tic-Tac-Toe	958	9	2
Congressional Voting	435	16	2

Table 3 Data Sets

4.2 Materials and Tools

All of the coding used to create and test the CRA process was completed utilizing Java. All data was stored and manipulated using MYSQL database tables and Microsoft Excel. The sequential covering algorithm used in this experiment was the PRISM algorithm. The divisive clustering algorithm used is the approach employed by Applications Quest (AppsQuest).

4.2.1 PRISM

PRISM is a simple sequential covering algorithm introduced by (Witten & Frank, 2005). The algorithm follows the process of the basic covering algorithm presented in chapter 2. The pseudocode for the PRISM algorithm is shown in figure 21. PRISM generates only correct or perfect rules. The success of a rule is measured by the accuracy formula *p/t,* where *t* is all of the instances covered by the new rule of which *p* are positive examples of the class. Rules with less than 100% accuracy are incorrect, meaning cases are assigned to the class that are not do not actually belong to the class in question. PRISM continues to add conjuncts (conditions) to the rule until it is perfect or there are no more attributes available for use.

```
For each class C
    Initialize E to the instance set
    While E contains instances in class C
        Create a rule R with an empty left-hand side that predicts class C
        Until R is perfect (or there are no more attributes to use) do
            For each attribute A not mentioned in R, and each value v,
            Consider adding the condition A=v to the left-hand side of R
            Select A and v to maximize the accuracy p/t
            (break ties by choosing the condition with the largest p)
        Add A=v to R
    Remove the instances covered by R from E.
```

Figure 21 PRISM Pseudocode

4.2.2 Applications Quest™

Applications Quest™ (AppsQuest) is a data mining tool developed by Juan E. Gilbert to aid institutions, agencies and businesses in the admissions/hiring process using data clustering. (Gilbert J. E., Applications Quest: A Case Study on Holistic Admissions, 2008) (Gilbert J. E., Applications Quest: Computing Diversity, 2006) The goal of the system is to recommend either

the most diverse or most similar group of applicants depending upon the needs of the organization by comparing applicants in a holistic way. There are many elements of an application that can be easily compared. Take GPA (grade point average), for example. If one student has a GPA of 3.5 and another has 2.7 the GPA of 3.5 is clearly higher. This comparison is not as straightforward when there are non numeric elements of the application to consider. AppsQuest differs from other divisive clustering applications in its treatment of nominal attributes (non-numeric). Traditionally, data clustering applications require the nominal attributes be assigned a value that allows them to be compared (e.g. 0 if the values are different and 1 if the values are the same). City, state, race, and gender are examples of nominal attributes. To provide a more accurate measure to compare nominal attributes the Nominal Population Metric (NPM) was created. The NPM process begins by identifying the nominal attributes. For clarification purposes, let's use Race as an example. There are 20 applicants in the pool where Race is an attribute. This attribute has the following unique values: B-black, W-white, H-hispanic, A-Asian, and O-other. Using the AppsQuest approach every application is compared to every other application in its entirety. For 20 applications compared 2 at a time there would be 190 comparisons. These results were obtained using the formula $n\ C\ r = n!/[(n\text{-}r)!\ r!]$. The NPM processes the nominal attributes in the following manner (Gilbert J. E., 2008):

1. Calculate the total number of combinations for all applications using $n\ C\ r$. Continuing the example, the total number of 190 combinations and place into matrix of combination and application pairs. This matrix is called the similarity matrix (see Table 4).

Application 1	**Application 2**	**Difference Measure**
1	2	20%
1	3	15%

1	4	75%

Table 4 Similarity Matrix with Difference Measures Example

2. Calculate the number of unique attributes values. For this example, the Race attribute has 5.

3. Calculate the number of combinations for the unique nominal attributes using *n C r*. The result is 10 combinations for the Race attribute.

4. For the 10 combinations of the nominal attribute value pairs, compute the coverage percentage within the application similarity matrix. This equates to the number of rows in the application similarity matrix have a value for Race pairs W-B, W-H, B-A, etc. These figures are placed into a nominal population matrix. (see Table 5)

Race	**Coverage**
W-B	30%
W-H	15%
B-A	5%

Table 5 Example Nominal Population Matrix

5. The nominal population matrix contains the attribute pair coverage across all comparisons. This is an accurate measure of the impact of the nominal attribute value pairs based on their existence within the data set.

6. If necessary the Coverage values are adjusted. This is the objective when the application is when measuring difference vs. similarity. The W-B Race pair had the greatest coverage in the nominal population matrix and is therefore the most common attribute value pair. The B-A value pair is the least common or most novel. The B-A pair should be receive more "credit" as it is the most novel. The credit is calculated proportionately in relation to other attribute value pairs. To accomplish this, the Coverage values are subtracted from 100% or 1.00 (see Table 6). This inverts the Coverage values such that B-A is given more credit compared to W-B relative to their actual coverage within application similarity matrix.

Race	**Coverage**
W-B	70%
W-H	85%

B-A	95%

Table 6 Nominal Population Matrix Example

7. The Coverage values in the nominal population matrix are now the Nominal Population Metrics which can be used in clustering algorithms as a method to accurately compare nominal attributes. The question, "What's the difference between a Black and Asian along the Race attribute?" can be answered. The NPM value is 95% in this example.

An alternative approach is to use the distribution of the values for each attribute. Using the previous example, Table 7 depicts the distribution of race within the data set (e.g. 70% of the instances have a W for race).

Race	Distribution
B	10%
W	70%
A	15%
H	5%

Table 7 Example Nominal Population Matrix

Race	Distribution
B	90%
W	30%
A	85%
H	95%

Table 8 Example Nominal Population Matrix Adjusted

1. Compute the distribution for each nominal attribute. (see Table 7)

2. Use the distribution value for each attribute at time of comparison. For example if calculating the difference between B and A along the Race attribute .1 and .15 would be used, respectively, in the calculation.

3. Compute the Nominal difference by identifying the two smallest distribution values. In this example this corresponds to H and B with values .05 and .1 respectively. This identifies the 2 most novel attributes based on their frequency of occurrence in the data set.

4. Calculate the sum of the 2 most novel attributes to determine their combined frequency (.05 + .1=.15). This frequency becomes the baseline for normalization as these are the most novel pair-wise nominal attribute values (for Race in this example).

5. Calculate the normalization by subtracting the combined frequency from 1. In this example, 1 - .15 = .85. This value becomes the normalization factor. All pair-wise comparisons are divided by this value. The H-B comparison yields a difference value of 1 (.85/.85). The W-H comparison yields a difference of .294 ([1-(.70 + .05)]/.85 =.294). The difference between W-H is .294 vs. 1.00 for H-B.

In general, the NPM covers the use of the distribution of nominal attribute values in efforts to determine the similarity or difference between nominal attribute values.

4.3 Procedure

The data was loaded into MYSQL database tables. The data was discretized using the standard deviation to create the corresponding categories for each continuous attribute (if any). The data was then split into training and test data sets. 80% of the data was used for training and 20% of the data was used for testing purposes. Proportionate samples for each class were used. The training data was clustered using AppsQuest divisive clustering algorithm. PRISM was then run on the clustered data. The resulting rule sets were then merged into a collective model. It is important to note that during the merging process any duplicate rules were removed. Contradictory rules were also resolved. Contradictory rules are rules in which the conditions for two or more rules are the same but the resulting classes are different. If this occurred the rule with the highest coverage was kept and the other rules removed. We examined the effects of the clustering rule based approach using both ordered and unordered rule sets for 0, 1, 2, 3, and 4 missing attributes. The attributes determined to be "missing" were chosen at random and modified in test data prior to each iteration of the experiment. (Hu & Li, 2005) The accuracy of each rule was determined using the calculation *p/t.* In this instance the *t* corresponds to the total number of instances covered in relation to the entire training data set instead of a particular class

and *p* is the number of instances that are positive examples of the target classification. A voting scheme was used in the unordered rule set using the accuracy of each rule that matched the test instance. The class with the highest sum of accuracies was used. The rule with the highest coverage was used to make the prediction if a tie occurred. In the ordered rule set the rules were arranged in decreasing order of accuracy. The first rule that matched was used to make the prediction. The results reported are based on an average of ten runs per dataset. Meaning for each data set the process was executed with 10 training and 10 individual test sets.

4.4 Expected Outcomes

It is expected that for some *k*, the CRA process will produce a collective model that will be more tolerant of missing attribute information in the new unseen test instances. The collective model will be larger and have shorter simpler rules which often achieve high classification accuracy and aid in adding robustness. (Witten & Frank, 2005) The rule length refers to the number of conditions (conjuncts) the rule contains. Longer rules are more specific and shorter rules are more general.

Chapter 5

Research Findings

5.1 Results

The goal of this research is to introduce a new method for adding robustness to rule based classifiers using divisive clustering. The resulting model would be more resilient against missing attribute information in new/unseen data. This section describes the results of the experiment conducted to test the validity of this method.

k values

Given the number of different datasets used in the experiment, there are no guarantees that the best performing *k* on one dataset will translate to the best performing *k* on other datasets. As such, the performance of the CRA process is measured against a set of *k* values: 1, 3, 5, 7, 9, 11, 13, and 15; however, it is possible to build the model using *k* values across a larger range, i.e. 1, 2, 3, … N-1, but that approach was not necessary. The results for *k* value 1 reflect the performance of the model using the traditional methods (control group).

Results for Congressional Voting Data

Tables 9 through 13 display the results for the tests performed using the Congressional Voting dataset. The traditional classifier performed very well on the complete data set as shown in Table 4 achieving an accuracy of 97% and 99% for the unordered and ordered models respectively. The CRA process performed well and was able to match the performance of the

traditional unordered classifier with an accuracy of 97%. For 1, 2, 3 and 4 missing attributes there existed a *k* that produced a collective model that obtained higher classification accuracy for either type of model. These values are highlighted in green. Instances in which the CRA process "tied" with the traditional model are highlighted in yellow. **(Note: Highlighting scheme green for wins, yellow for ties will be consistent for remaining results.)** Figures 22 and 23 depict a graphical representation of the classification accuracy for each *k.* As the number of missing attributes increased, the performance of the traditional method (*k* value 1 for ordered and unordered models) suffered while the CRA process performed more consistently. Figure 24 depicts the rule average rule lengths for the models. The rule lengths were smaller for models built using the CRA method. On average, the rule lengths were 2.81 and 1.84 for the traditional and CRA methods respectively. Additionally, the size of the CRA generated models were larger (contained more rules) than the traditional models. The CRA models contained, on average, approximately 25 rules while the traditional model contained 22. Also, there was a decrease in the number of instances not classified when the CRA method was used.

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.97	.99	1	23.9	33.7	2.83
3	.97	.98	0	24.4	30.2	2.22
5	.95	.95	0	26.8	31.5	2.01
7	.95	.96	0	26.8	30.6	1.94
9	.93	.95	0	26	29	1.77
11	.91	.97	0	24.8	28.3	1.69
13	.92	.95	0	24.8	28.6	1.71
15	.91	.94	0	24.9	28.5	1.68

Table 9 Congressional Voting Results for Complete Data (No Missing Attributes)

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.92	.96	1.75	21.6	31.5	2.78
3	.94	.97	1	22.8	29	2.17
5	.94	.95	0	24.9	29.3	1.98
7	.93	.95	0	24.9	28.6	1.91
9	.91	.91	0	23.3	25.7	1.69
11	.91	.92	0	22.7	25.6	1.63
13	.9	.93	0	22.6	25.4	1.65
15	.92	.93	0	23.9	25.8	1.59

Table 10 Congressional Voting 1 Missing Attribute

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.87	.89	4	21.45	30.18	2.68
3	.89	.91	1.36	23.18	30.45	2.29
5	.9	.92	0	25.18	31.64	2.1
7	.89	.92	0	25.91	20.45	1.96
9	.9	.92	0	25.91	29	1.76
11	.87	.91	0	25.82	28.73	1.73
13	.87	.9	0	24.36	27.36	1.68
15	.89	.91	0	24.91	27.27	1.63

Table 11 Congressional Data Results 2 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.85	.86	6.64	21.27	32.09	2.86
3	.92	.92	.82	24.09	29.82	2.05
5	.92	.93	0	25.73	30.18	2.06
7	.91	.91	0	26.27	29.45	1.96
9	.88	.91	0	25.73	28.27	1.83
11	.85	.89	0	24.82	26.36	1.67
13	.87	.91	0	24.91	26.64	1.59
15	.87	.9	0	25	25.91	1.57

Table 12 Congressional Voting Data Results 3 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.77	.79	9.18	24.81	34.18	2.9
3	.86	.88	2	25.82	32.27	2.3
5	.89	.9	0	25.55	29.55	1.92
7	.89	.9	0	27.18	30.64	1.91
9	.85	.89	0	26.45	28.55	1.76
11	.86	.89	0	24.73	27.45	1.75
13	.87	.9	0	24.73	27.36	1.7
15	.88	.9	0	25.36	26.73	1.56

Table 13 Congressional Voting Results 4 Missing Attributes

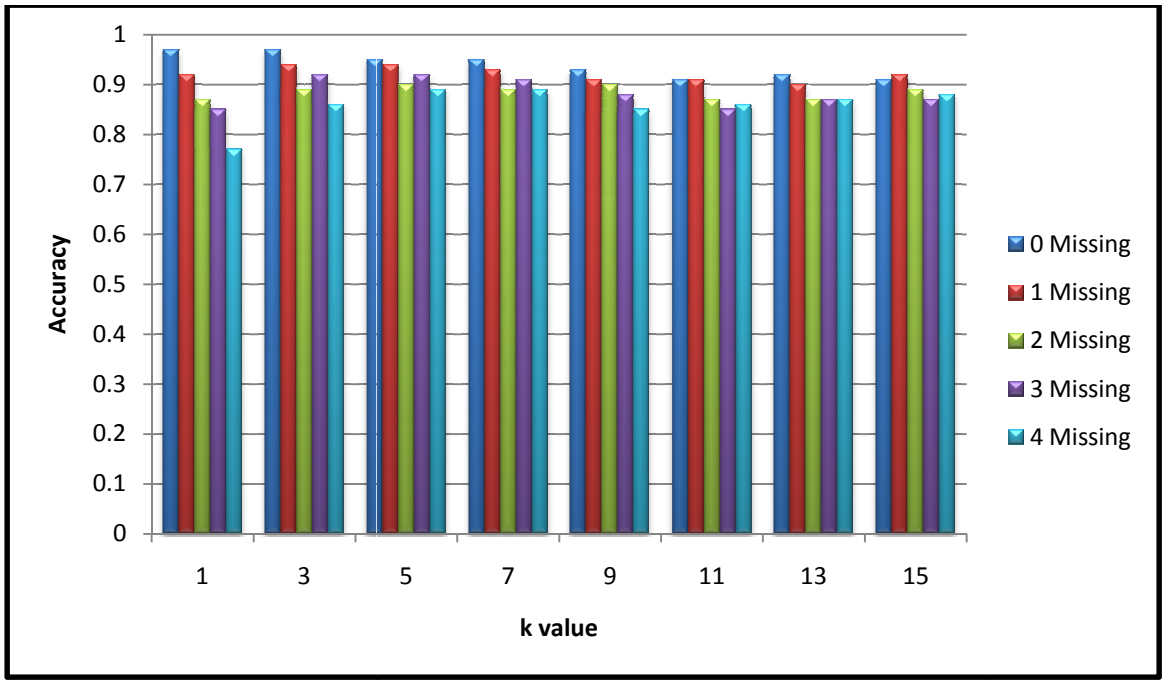


Figure 22 Congressional Voting Accuracies Unordered Model

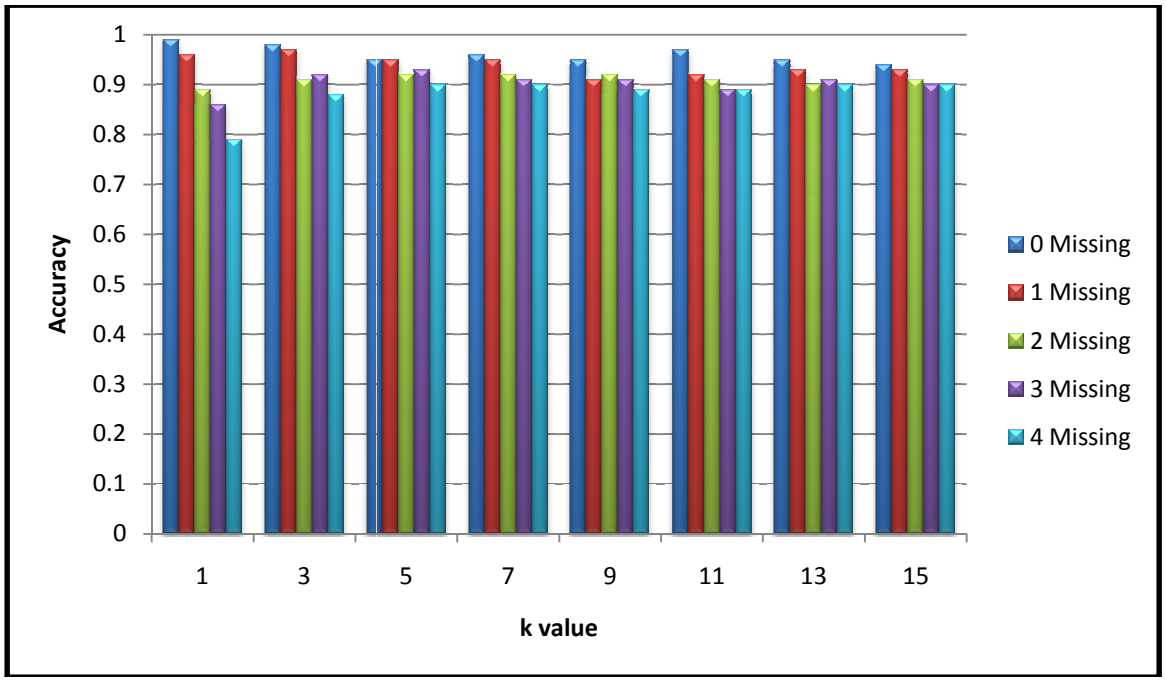


Figure 23 Congressional Voting Accuracies Ordered Model

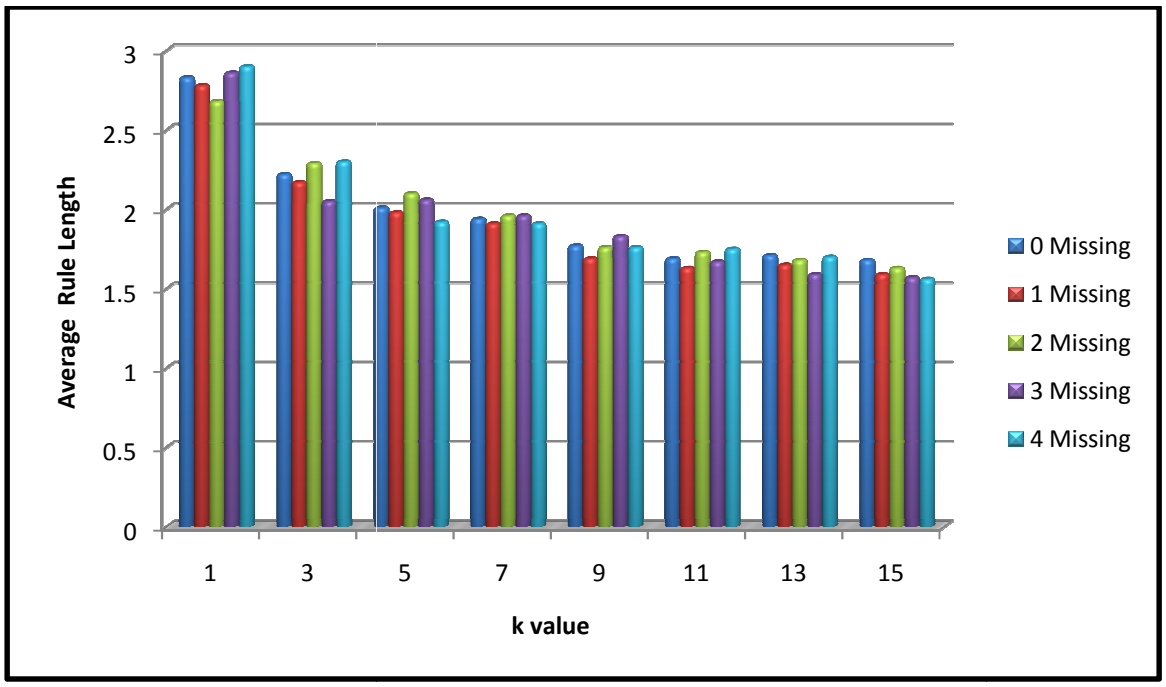


Figure 24 Congressional Voting Average Rule Lengths

	Cluster	Rules	Accuracy	Coverage
1	0	If phys_fee_freeze=`n` and adopt_budget_res=`y` then class=`democrat`	1	173
2	0	If phys_fee_freeze=`n` and crime=`n` then class=`democrat`	1	126
3	0	If syn_corp_cuts=`y` and immigration=`n` and water_proj_cost_shar=`n` and handicaped_infants=`n` and exp_admin_safrica=`n` then class=`democrat`	1	2
4	0	If phys_fee_freeze=`n` and syn_corp_cuts=`y` then class=`democrat`	1	94
5	0	If el_salvador_aid=`n` and syn_corp_cuts=`y` and immigration=`n` then class=`democrat`	1	35
6	0	If phys_fee_freeze=`?` and education_spending=`n` then class=`democrat`	1	6
7	0	If exp_admin_safrica=`?` and adopt_budget_res=`y` and aid_nic_contras=`n` then class=`democrat`	1	6
8	0	If education_spending=`n` and exp_admin_safrica=`n` and crime=`y` then class=`democrat`	1	5
9	0	If phys_fee_freeze=`n` and education_spending=`y` then class=`democrat`	1	23
10	0	If superfund_right_to_sue=`?` and syn_corp_cuts=`y` then class=`democrat`	1	7
11	0	If superfund_right_to_sue=`?` and exp_admin_safrica=`y` then class=`democrat`	1	7
12	0	If phys_fee_freeze=`n` and exp_admin_safrica=`?` then class=`democrat`	1	60
13	0	If adopt_budget_res=`?` and mx_missile=`y` then class=`democrat`	1	5
14	0	If syn_corp_cuts=`y` and superfund_right_to_sue=`n` and adopt_budget_res=`n` then class=`democrat`	1	8
15	0	If exp_admin_safrica=`?` and handicaped_infants=`y` and adopt_budget_res=`n` then class=`democrat`	1	3
16	0	If adopt_budget_res=`n` and education_spending=`y` and handicaped_infants=`n` then class=`republican`	0.92391	92
17	0	If relig_grp_schools=`y` and syn_corp_cuts=`n` then class=`republican`	0.75781	128
18	0	If adopt_budget_res=`n` and relig_grp_schools=`n` then class=`republican`	0.71429	14
19	0	If superfund_right_to_sue=`y` and exp_admin_safrica=`y` and syn_corp_cuts=`n` and immigration=`y` and mx_missile=`y` then class=`republican`	0.46154	13
20	0	If crime=`y` and handicaped_infants=`y` and syn_corp_cuts=`y` and relig_grp_schools=`n` then class=`republican`	0.25	4

21	0	If el_salvador_aid=`y` and immigration=`n` and mx_missile=`n` and handicaped_infants=`y` and duty_free_exports=`n` and relig_grp_schools=`y` then class=`republican`	0.66667	12
22	0	If el_salvador_aid=`y` and water_proj_cost_shar=`n` and education_spending=`y` and adopt_budget_res=`y` then class=`republican`	0.85714	7
23	0	If duty_free_exports=`?` and handicaped_infants=`n` and exp_admin_safrica=`?` and relig_grp_schools=`y` then class=`republican`	0.75	4
24	0	If handicaped_infants=`n` and relig_grp_schools=`n` and immigration=`y` and water_proj_cost_shar=`n` and duty_free_exports=`y` and exp_admin_safrica=`y` and education_spending=`n` then class=`republican`	0.2	5
25	0	If water_proj_cost_shar=`y` then class=`republican`	0.39264	163
26	0	If exp_admin_safrica=`?` then class=`republican`	0.17284	81

Table 14 Congressional Voting Example Traditional Rule Set

Table 14 displays one of the rule sets generated in the traditional manner using the Congressional Voting data. Table 15 displays one of rule sets generated with the CRA process. The rule set in Table 15 is a collective model derived from 7 individual models (7 clusters).

	Cluster	Rules	Accuracy	Coverage
1	0	If water_proj_cost_shar=`n` and duty_free_exports=`n` and handicaped_infants=`n` and exp_admin_safrica=`n` then class=`democrat`	0.125	16
2	0	If phys_fee_freeze=`n` and education_spending=`y` then class=`democrat`	1	23
3	0	If adopt_budget_res=`y` and duty_free_exports=`n` then class=`democrat`	0.85915	71
4	0	If education_spending=`n` and duty_free_exports=`y` then class=`democrat`	0.9619	105
5	0	If superfund_right_to_sue=`n` and anti_sat_test_ban=`n` then class=`democrat`	0.75	20
6	0	If education_spending=`n` and exp_admin_safrica=`?` then class=`democrat`	0.96364	55
7	0	If el_salvador_aid=`?` then class=`democrat`	0.91667	12
8	0	If mx_missile=`y` then class=`democrat`	0.9125	160
9	0	If water_proj_cost_shar=`y` and exp_admin_safrica=`y` and duty_free_exports=`n` then class=`republican`	0.54386	57
10	0	If phys_fee_freeze=`y` and adopt_budget_res=`n` and superfund_right_to_sue=`y` then class=`republican`	0.97143	105
12	0	If phys_fee_freeze=`y` and adopt_budget_res=`n` then	0.95041	121

		class=`republican`		
13	0	If exp_admin_safrica=`y` and anti_sat_test_ban=`n` and education_spending=`y` then class=`republican`	0.86275	51
15	1	If adopt_budget_res=`y` then class=`republican`	0.07576	198
16	2	If crime=`n` then class=`democrat`	0.98485	132
17	2	If duty_free_exports=`y` then class=`democrat`	0.91729	133
18	2	If handicaped_infants=`n` then class=`democrat`	0.42932	191
19	2	If immigration=`n` then class=`democrat`	0.65143	175
20	2	If phys_fee_freeze=`n` and adopt_budget_res=`y` then class=`democrat`	1	173
21	2	If immigration=`y` and duty_free_exports=`n` and superfund_right_to_sue=`n` and mx_missile=`y` and anti_sat_test_ban=`y` and water_proj_cost_shar=`y` and syn_corp_cuts=`y` then class=`republican`	0.33333	3
22	2	If crime=`y` and syn_corp_cuts=`n` and duty_free_exports=`n` and water_proj_cost_shar=`n` then class=`republican`	0.85417	48
23	3	If phys_fee_freeze=`n` then class=`democrat`	0.9898	196
24	3	If crime=`y` then class=`republican`	0.62871	202
26	4	If aid_nic_contras=`y` then class=`democrat`	0.91935	186
27	4	If adopt_budget_res=`?` then class=`democrat`	0.77778	9
28	5	If superfund_right_to_sue=`y` then class=`democrat`	0.36047	172
29	5	If adopt_budget_res=`n` then class=`democrat`	0.17021	141
30	5	If handicaped_infants=`n` and education_spending=`n` and el_salvador_aid=`y` and adopt_budget_res=`y` and mx_missile=`n` and relig_grp_schools=`y` and water_proj_cost_shar=`y` then class=`republican`	0.2	5
31	6	If aid_nic_contras=`n` then class=`democrat`	0.26351	148
32	6	If aid_nic_contras=`y` and el_salvador_aid=`y` then class=`republican`	0.47619	21
33	6	If anti_sat_test_ban=`y` and education_spending=`n` and immigration=`y` then class=`republican`	0.125	72

Table 15 Congressional Voting Example Collective Rule Set

Results for Anneal data

Tables 16 through 20 contain the results for the performed using the Anneal data set. The traditional classifiers both performed well on the complete data set achieving a 90% and 94% for the unordered and ordered models respectively. The CRA process achieved a higher accuracy

than the traditional model for tests with 2, 3 and 4 missing attributes (highlighted in green). Figures 25 and 26 depict a graphical representation of the classification accuracy for each *k*. There was a decrease in the number of instances not classified when the CRA method was utilized with the occurrence of missing attributes.

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.9	.94	0	28	46.4	3.7
3	.82	.9	0	46.4	52.6	2.75
5	.78	.86	0	52.3	53.7	2.44
7	.76	.86	0	51.4	52.9	2.54
9	.76	.85	0	52.8	51.6	2.21
11	.76	.86	0	53.4	52.6	2.17
13	.76	.85	0	54.4	52.7	2.07
15	.76	.85	0	52.5	49.8	1.93

Table 16 Anneal Results Complete Data (No Missing Attributes)

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.87	.91	.6	31	46.3	3.46
3	.83	.9	0	44.6	51.7	2.79
5	.77	.87	0	51.7	55.7	2.58
7	.76	.86	0	52	53	2.38
9	.76	.86	0	50.8	50.2	2.23
11	.76	.84	0	49.4	50.7	2.1
13	.77	.84	0	50.8	49.5	1.9
15	.76	.84	0	51.6	50.2	1.86

Table 17 Anneal Results 1 Missing Attribute

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.85	.87	2.1	33.6	50.7	3.88
3	.8	.88	0	49.3	55.4	3
5	.78	.86	0	55.7	55.7	2.64
7	.78	.86	0	54.4	54.9	2.62
9	.77	.87	0	53.8	51.7	2.36

11	.76	.87	0	53.5	50	2.13
13	.76	.85	0	52.7	48.2	1.98
15	.76	.85	0	52.3	47.1	1.81

Table 18 Anneal Results 2 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.85	.88	2.2	30.6	47.8	3.49
3	.82	.9	0	44.2	52.8	2.97
5	.78	.87	0	51.7	52.6	2.48
7	.76	.86	0	53.5	54.6	2.41
9	.76	.86	0	53.8	53.4	2.27
11	.76	.84	0	52.5	51.6	2.16
13	.76	.85	0	54.9	51.3	1.98
15	.76	.84		53	48.8	1.9

Table 19 Anneal Results 3 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.82	.84	4.8	33.3	48.3	3.61
3	.81	.87	0	47.3	56.7	3.19
5	.78	.84	0	46.6	54.7	2.76
7	.77	.84	0	50.3	51.4	2.41
9	.76	.83	0	52.7	51.5	2.23
11	.76	.83	0	54	51	2.12
13	.76	.83	0	56	51.8	2
15	.76	.83	0	51.6	48.4	1.89

Table 20 Anneal Results 4 Missing Attributes

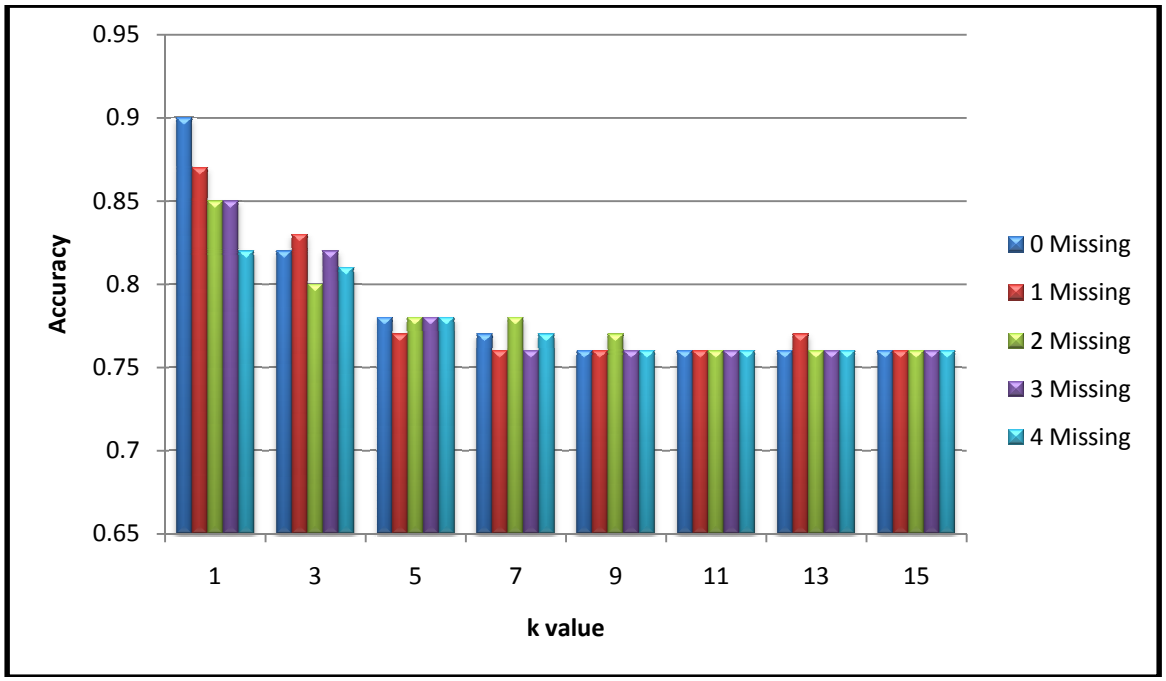


Figure 25 Anneal Accuracies Unordered Model

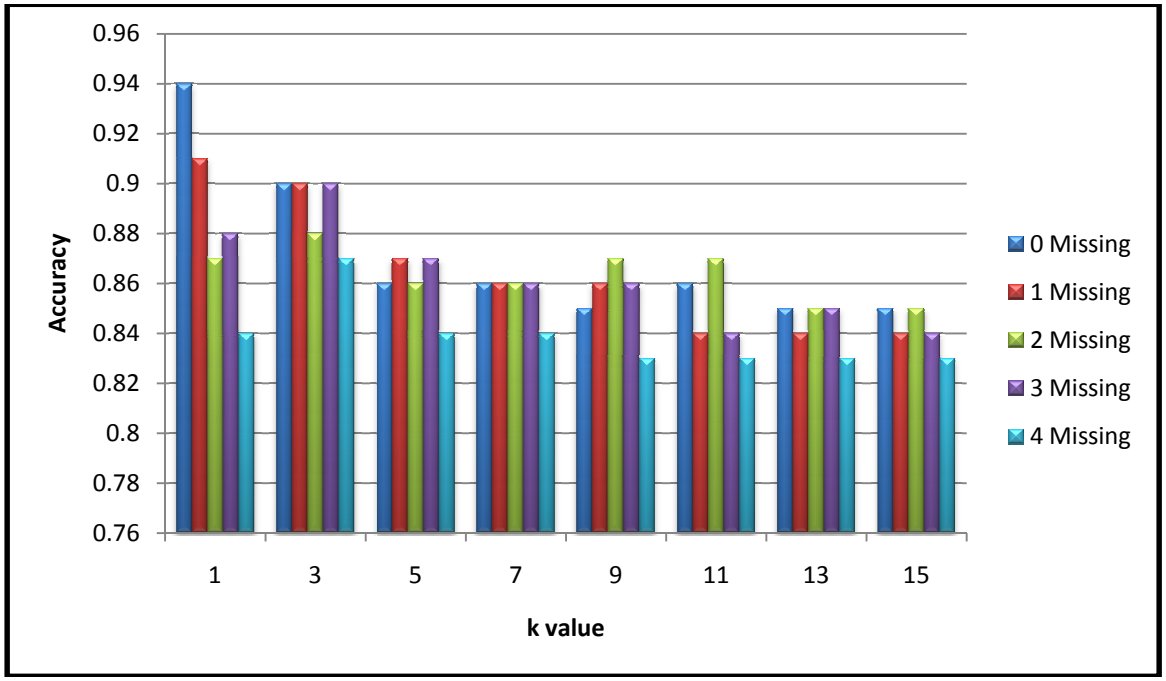


Figure 26 Anneal Accuracies Ordered Model

Figure 27 depicts the average rule length for the models created using the Anneal data set. The CRA method produced shorter rules. The rule lengths for the collective and traditional methods were 2.32 and 3.63 on average. Also, the models created via the CRA method were

generally larger than those created with the traditional method. The collective model contained 52 rules, on average, while the traditional model contained 31. Tables 21 and 22 display examples of the traditional and collective models created.

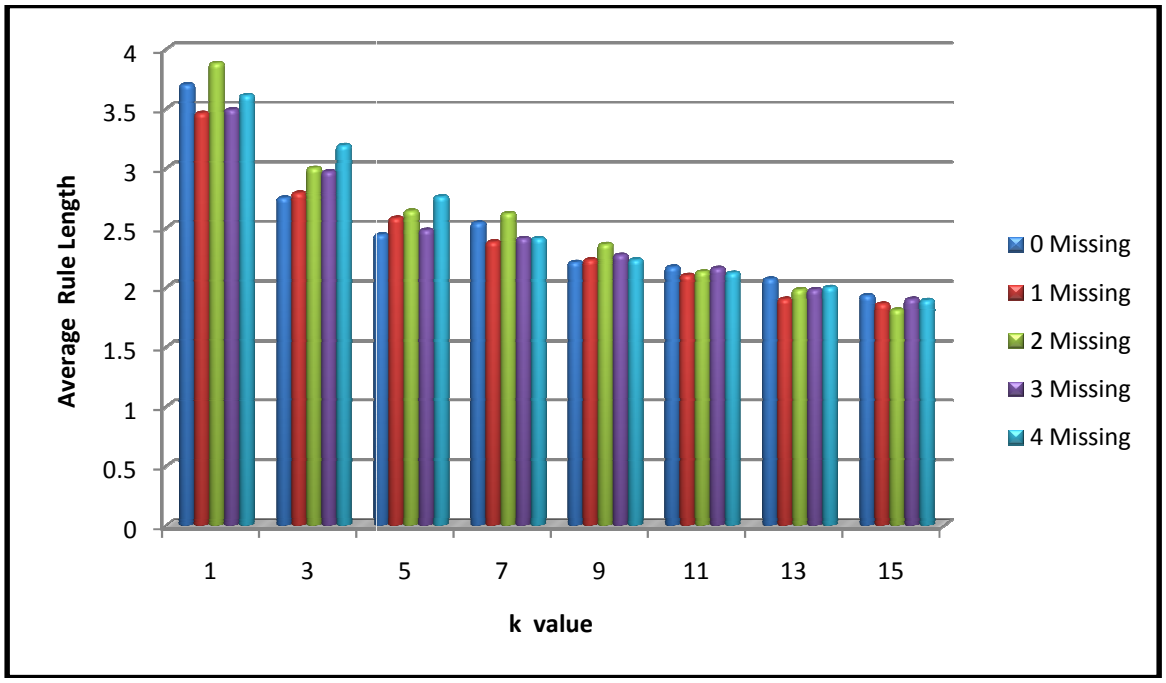


Figure 27 Anneal Average Rule Lengths

	Cluster	Rules	Accuracy	Coverage
1	0	If steel=`S` and carbon=`0-13.72` then class=`1`	1	4
2	0	If squality=`?` and bf=`Y` and steel=`A` then class=`1`	1	1
3	0	If exptl=`Y` then class=`1`	1	1
4	0	If squality=`?` and family=`?` and carbon=`0-13.72` and strength=`0-114.95` then class=`2`	0.81481	81
5	0	If con=`?` and temper_rolling=`?` and carbon=`0-13.72` and family=`?` and squality=`?` then class=`2`	0.42857	14
6	0	If width=`810.61-1215.92` and carbon=`0-13.72` and temper_rolling=`?` and con=`?` then class=`2`	0.71429	7
7	0	If strength=`574.76-689.71` then class=`2`	1	8
8	0	If formal_ability=`2` and len=`0-1870.92` and enamelability=`?` then class=`3`	0.90476	210
9	0	If bl=`Y` then class=`3`	0.85124	121
10	0	If bf=`Y` then class=`3`	0.95192	104

11	0	If con=`S` and thick=`0.24-1.11` then class=`3`	0.91228	285
12	0	If family=`?` and strength=`0-114.95` and width=`810.61-1215.92` and hardness=`0-24.75` then class=`3`	0.86111	36
13	0	If family=`?` and width=`405.31-810.61` and temper_rolling=`?` and strength=`0-114.95` then class=`3`	0.88333	240
14	0	If family=`?` and width=`1215.92-1525` and oil=`?` then class=`3`	0.8253	166
15	0	If steel=`R` and exptl=`?` then class=`3`	0.71362	213
16	0	If squality=`G` and thick=`0.24-1.11` and len=`0-1870.92` then class=`3`	0.93548	93
17	0	If thick=`1.98-2.85` and blue_bright_varn_clean=`?` then class=`3`	0.80488	41
18	0	If formal_ability=`?` and bt=`?` and oil=`?` and width=`405.31-810.61` and bw_me=`?` then class=`3`	0.85507	69
19	0	If width=`0-405.31` and family=`?` then class=`3`	0.82143	84
20	0	If ferro=`?` and chrom=`?` and hardness=`0-24.75` and phos=`?` and width=`405.31-810.61` and bw_me=`?` then class=`3`	0.84153	183
21	0	If con=`S` then class=`3`	0.84966	439
22	0	If hardness=`49.49-74.24` then class=`3`	1	66
23	0	If formal_ability=`2` then class=`3`	0.86139	303
24	0	If con=`?` and steel=`A` and bt=`?` and width=`405.31-810.61` and lustre=`?` and oil=`?` and bf=`?` then class=`5`	0.51429	35
25	0	If con=`?` and temper_rolling=`?` and carbon=`0-13.72` and bt=`?` and sfinish=`?` and bore=`0000` and bf=`?` then class=`5`	0.57143	70
26	0	If steel=`?` and squality=`?` and blue_bright_varn_clean=`?` and len=`0-1870.92` then class=`5`	0.92308	13
27	0	If thick=`0.24-1.11` then class=`5`	0.06865	437
28	0	If con=`?` and squality=`G` and bt=`?` and bl=`?` and lustre=`?` and bore=`0000` and bw_me=`?` and bf=`?` then class=`U`	0.31373	51
29	0	If bl=`?` then class=`U`	0.05369	596

Table 21 Example Anneal Traditional Rule Set

RuleID	Cluster	Rules	Accuracy	Coverage
2	0	If squality=`?` and steel=`R` and con=`S` then class=`2`	1	61
3	0	If sfinish=`P` then class=`2`	1	6

4	0	If enamelability=`2` then class=`2`	1	5
5	0	If shape=`COIL` then class=`3`	0.82121	330
6	0	If thick=`0.24-1.11` and family=`?` and width=`405.31-810.61` and enamelability=`?` then class=`3`	0.8984	187
7	0	If family=`?` then class=`3`	0.82736	614
8	0	If con=`S` then class=`3`	0.84966	439
9	0	If thick=`1.11-1.98` and bl=`?` and width=`1215.92-1525` and bf=`?` and enamelability=`?` and strength=`0-114.95` then class=`5`	0.31818	22
10	0	If non_ageing=`N` and thick=`1.11-1.98` and formal_ability=`3` and len=`0-1870.92` and shape=`SHEET` then class=`5`	0.83333	12
11	0	If non_ageing=`N` and bw_me=`B` then class=`5`	0.53333	15
12	0	If steel=`A` and bw_me=`?` then class=`5`	0.12075	265
13	0	If thick=`0.24-1.11` then class=`5`	0.06865	437
14	0	If bc=`?` then class=`U`	0.04469	716
15	1	If steel=`S` and carbon=`0-13.72` then class=`1`	1	4
16	1	If exptl=`Y` then class=`1`	1	1
17	1	If squality=`?` and carbon=`0-13.72` and family=`?` and width=`0-405.31` then class=`2`	0.75	12
18	1	If steel=`R` and squality=`?` and bt=`?` and shape=`COIL` then class=`2`	0.64286	14
19	1	If con=`?` and carbon=`0-13.72` and bore=`0000` and family=`?` and squality=`?` and bt=`?` and width=`405.31-810.61` then class=`2`	0.18182	11
20	1	If strength=`574.76-689.71` then class=`2`	1	8
21	1	If family=`?` and temper_rolling=`?` and width=`405.31-810.61` and formal_ability=`?` then class=`3`	0.90625	64
22	1	If con=`S` and thick=`0.24-1.11` then class=`3`	0.91228	285
23	1	If thick=`1.98-2.85` and blue_bright_varn_clean=`?` then class=`3`	0.80488	41
24	1	If formal_ability=`2` then class=`3`	0.86139	303
25	1	If strength=`459.81-574.76` then class=`3`	0.61538	13
26	1	If family=`?` and thick=`0.24-1.11` then class=`3`	0.8973	370
27	1	If bt=`Y` and hardness=`0-24.75` then class=`3`	0.76923	39
28	1	If thick=`1.11-1.98` then class=`3`	0.55621	169
29	1	If carbon=`41.15-54.86` then class=`3`	1	13
30	1	If con=`A` then class=`5`	0.46667	30
31	1	If thick=`0.24-1.11` and carbon=`0-13.72` and strength=`0-114.95` and hardness=`0-24.75` then	0.09772	307

		class=`5`		
32	1	If squality=`G` and strength=`0-114.95` then class=`U`	0.14483	145
33	1	If width=`1215.92-1525` then class=`U`	0.03286	213
34	2	If squality=`?` and steel=`A` and bf=`Y` then class=`1`	1	1
35	2	If squality=`?` and family=`?` and carbon=`0-13.72` and bc=`?` and bl=`?` and bore=`0000` and bt=`?` and bw_me=`?` and cbond=`?` and chrom=`?` and corr=`?` and enamelability=`?` and exptl=`?` and ferro=`?` and hardness=`0-24.75` and jurofm=`?` and lustre=`?` and marvi=`?` and non_ageing=`?` and oil=`?` and packing=`?` and phos=`?` and product_type=`C` and len=`0-1870.92` and blue_bright_varn_clean=`?` and sfinish=`?` and strength=`0-114.95` and bf=`Y` and width=`405.31-810.61` then class=`2`	0.5	2
36	2	If steel=`M` then class=`2`	0.5	16
37	2	If blue_bright_varn_clean=`B` then class=`2`	1	3
38	2	If con=`S` and shape=`SHEET` then class=`2`	0.19244	291
39	2	If width=`0-405.31` and ferro=`?` and hardness=`0-24.75` then class=`3`	0.75676	74
40	2	If thick=`0.24-1.11` and family=`?` then class=`3`	0.8973	370
41	2	If bt=`?` and formal_ability=`?` and oil=`?` and shape=`SHEET` and sfinish=`?` then class=`3`	0.98485	66
42	2	If thick=`1.98-2.85` and con=`?` then class=`3`	0.88571	35
43	2	If ferro=`?` and hardness=`0-24.75` and phos=`?` and thick=`1.11-1.98` then class=`3`	0.60769	130
44	2	If hardness=`49.49-74.24` then class=`3`	1	66
45	2	If shape=`SHEET` and strength=`0-114.95` and bt=`?` and len=`0-1870.92` and bw_me=`?` and oil=`?` and carbon=`0-13.72` then class=`5`	0.176	125
46	2	If shape=`COIL` and bf=`?` and lustre=`?` and bt=`?` then class=`5`	0.08511	235
47	2	If steel=`A` and bf=`?` and bore=`0000` and lustre=`?` then class=`U`	0.10425	259

Table 22 Example Anneal Collective Rule Set

Results for Thyroid Data

Tables 23 through 28 display the results of the test performed using the Thyroid data set. The traditional models performed very well achieving accuracies of 97% and 98% for the unordered and ordered models respectively. A collective model was created that was able to match the performance of the traditional model in the presence of 0, 1, 2, and 3 missing attributes (highlighted in yellow). The CRA process created a model that achieved a higher accuracy than the traditional method in the presence of 4 missing attributes. Figures 28 and 29 provide a graphical representation of the accuracies achieved for each *k*.

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.97	.98	.1	52.3	59.7	2.22
3	.96	.98	0	59	64.9	2.13
5	.95	.97	0	51.7	51.4	2.01
7	.95	.97	0	50.3	49	1.78
9	.95	.96	0	48.2	46.1	1.7
11	.95	.96	0	47.2	46	1.72
13	.95	.96	0	47.7	45.7	1.7
15	.95	.96	0	47.9	45.8	1.62

Table 23 Thyroid Results Complete Data (No Missing Attributes)

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.97	.97	1.7	53.4	60.1	2.29
3	.95	.97	.3	58.4	64.2	2.18
5	.95	.97	.1	50.2	49.5	1.92
7	.95	.96	0	47.1	46.2	1.81
9	.95	.96	0	49	46.8	1.71
11	.95	.96	0	47.4	44.5	1.67
13	.95	.96	0	47.4	43.8	1.66
15	.96	.96	0	46.3	42.2	1.63

Table 24 Thyroid Results 1 Missing Attribute

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.96	.97	3.4	57.9	59.8	2.3
3	.96	.97	.8	62.7	61	2.17
5	.95	.97	0	49.3	49.2	2.12
7	.95	.96	0	44.7	44.2	1.72
9	.95	.96	0	44.9	42.1	1.64
11	.95	.96	0	43.7	40.4	1.63
13	.95	.96	0	43.1	40	1.61
15	.95	.96	0	41.4	38.1	1.56

Table 25 Thyroid Results 2 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.95	.97	1.9	45.6	52.8	2.34
3	.95	.97	.8	48	54.6	2.26
5	.95	.96	.4	56.4	55.7	2.07
7	.95	.96	.2	52.7	50.6	1.87
9	.95	.96	0	53.6	50.6	1.81
11	.95	.96	0	54.5	49.9	1.77
13	.95	.96	0	52.9	46.6	1.72
15	.95	.96	0	49	43.8	1.65

Table 26 Thyroid Results 3 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.94	.95	11.9	55.1	59.3	2.34
3	.95	.97	.5	61.5	63.4	2.39
5	.95	.97	.1	56.1	53.3	1.89
7	.95	.96	0	55	51.2	1.84
9	.95	.96	0	52.2	48.2	1.76
11	.95	.96	0	47.7	46.1	1.74
13	.95	.96	0	46.4	43.2	1.67
15	.95	.96	0	43.3	41.5	1.64

Table 27 Thyroid Results 4 Missing Attributes

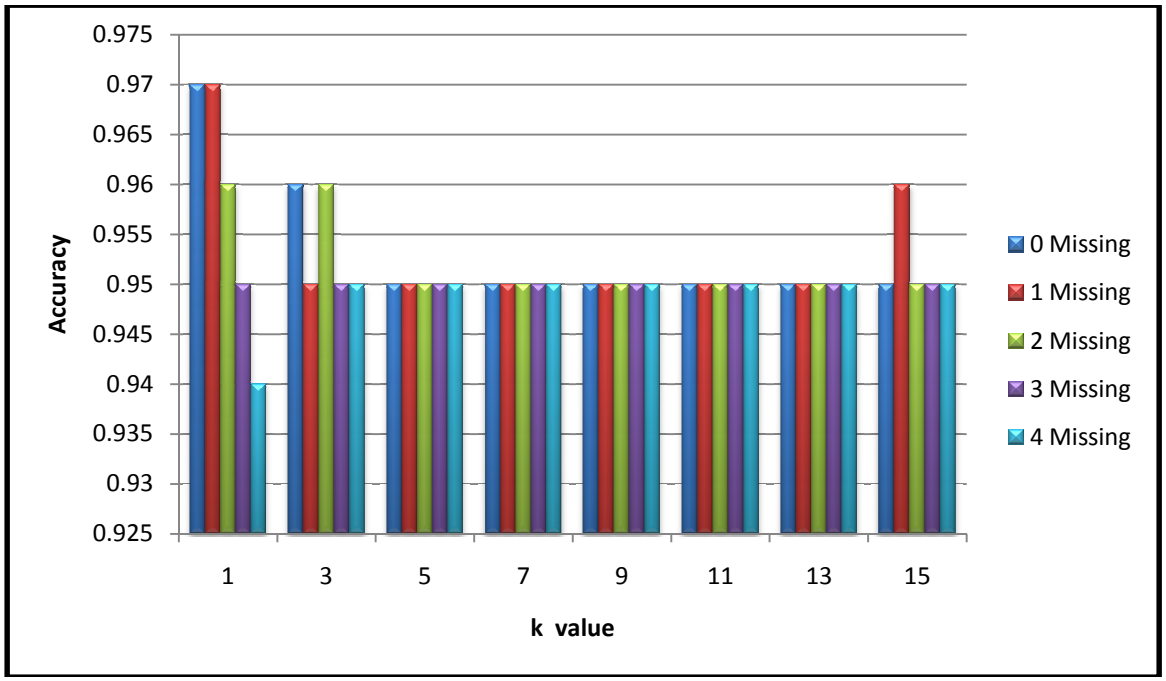


Figure 28 Thyroid Accuracies Unordered Models

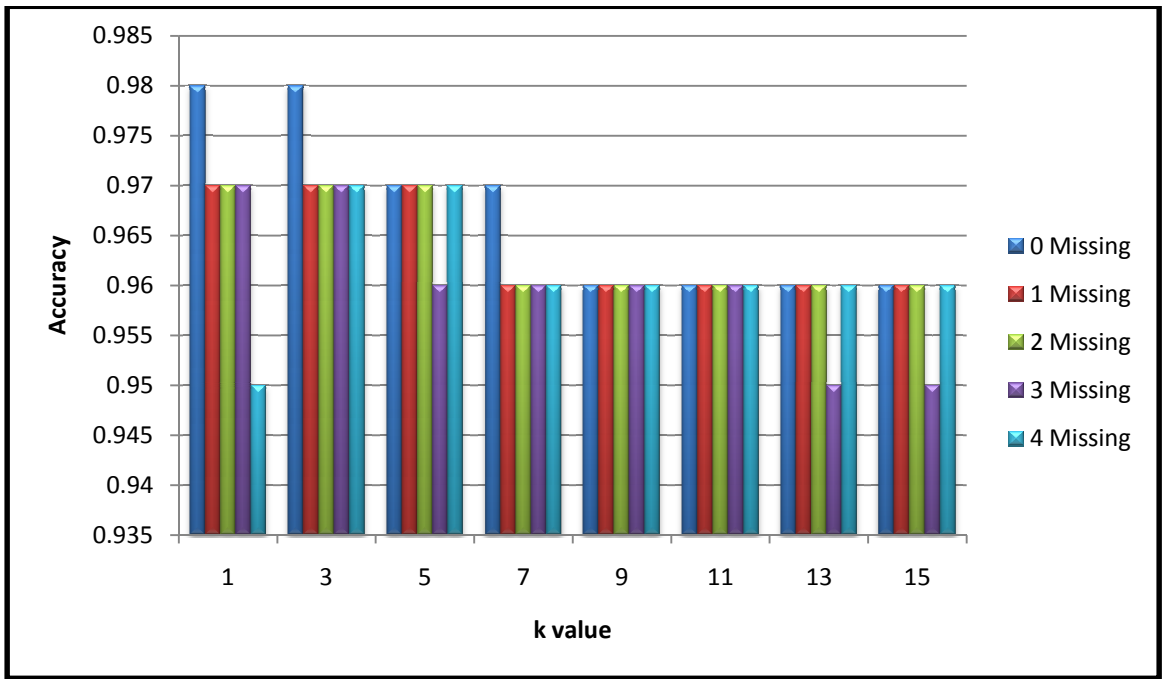


Figure 29 Thyroid Accuracies Ordered Models

Figure 30 displays the average rule lengths of the models created using the Thyroid data set. The rules that were generated via the CRA method were shorter. The rule lengths were 2.3

and 1.8, on average, for the traditional and collective models respectively. The models created by the traditional method were larger. On average, the traditional method models contained 52 rules while the collective models contained 50. Tables 28 and 29 are examples of models generated for this data set via the traditional and CRA methods.

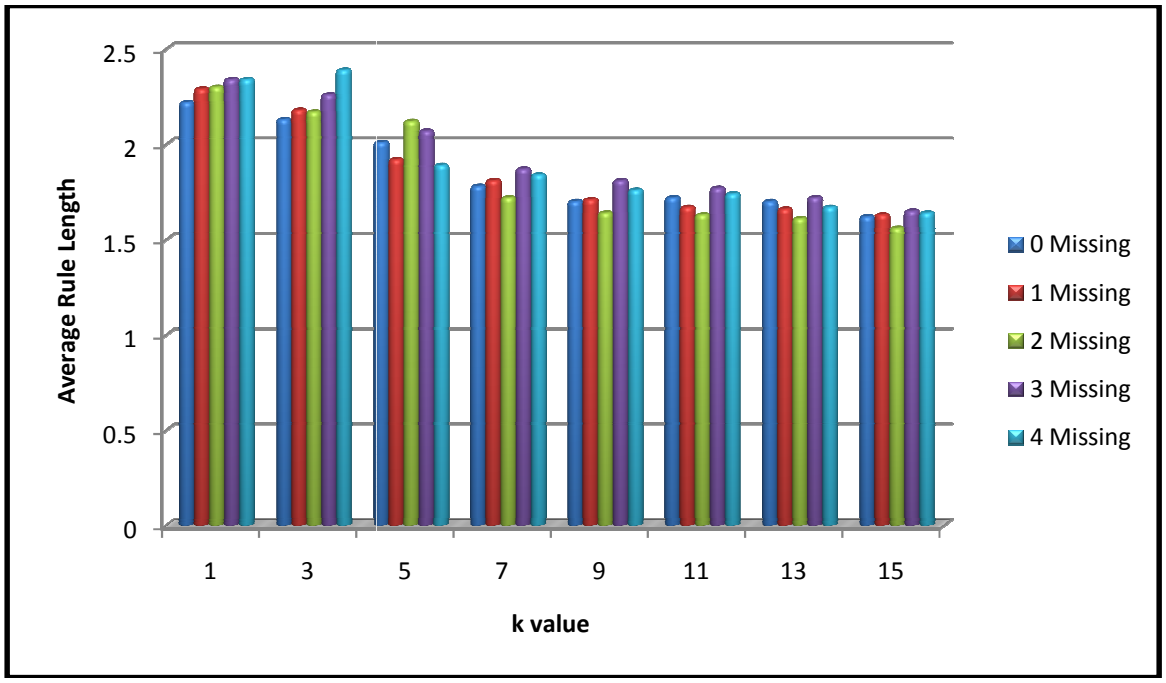


Figure 30 Thyroid Average Rule Lengths

	Cluster	Rules	Accuracy	Coverage
1	0	If TSH=`164` then class=`hypothyroid`	1	1
2	0	If FTI=`0-60.23` and TSH=`23.9-47.79` and T3=`1.99-2.99` then class=`hypothyroid`	1	6
3	0	If FTI=`0-60.23` and age=`58.87-78.16` and T4U=`1.36-1.59` and tumor=`f` then class=`hypothyroid`	1	4
4	0	If TSH=`150` then class=`hypothyroid`	1	2
5	0	If TT4=`5.30` then class=`hypothyroid`	1	1
6	0	If TSH=`47.79-71.69` and T3=`0-1` and query_hypothyroid=`f` then class=`hypothyroid`	1	13
7	0	If FTI=`0-60.23` and TSH=`23.9-47.79` and T3=`0-1` then class=`hypothyroid`	1	13
8	0	If FTI=`0-60.23` and age=`78.16-97.45` and on_thyroxine=`f` then class=`hypothyroid`	1	6
9	0	If thyroid_surgery=`t` and age=`1-20.29` and T3=`1-1.99` then class=`hypothyroid`	1	3

10	0	If query_hypothyroid=`t` and age=`78.16-97.45` and T3=`1.99-2.99` and FTI=`60.23-99` then class=`hypothyroid`	1	2
11	0	If TSH=`235` then class=`hypothyroid`	1	1
12	0	If TSH=`216` then class=`hypothyroid`	1	1
13	0	If FTI=`0-60.23` and age=`58.87-78.16` and TT4=`10-55.48` and T3=`1-1.99` then class=`hypothyroid`	1	10
14	0	If TSH=`213` then class=`hypothyroid`	1	1
15	0	If T4U=`0.91-1.13` and on_thyroxine=`f` and query_hypothyroid=`t` and TT4=`55.48-99` and T3=`1-1.99` and age=`58.87-78.16` and sex=`F` then class=`hypothyroid`	0.5	2
16	0	If T4U=`0.91-1.13` and age=`39.58-58.87` and query_hypothyroid=`t` and TT4=`55.48-99` and T3=`1.99-2.99` then class=`hypothyroid`	1	1
17	0	If TSH=`100` and on_thyroxine=`f` then class=`hypothyroid`	1	3
18	0	If TSH=`47.79-71.69` and FTI=`0-60.23` and on_antithyroid_medication=`f` and query_hypothyroid=`f` then class=`hypothyroid`	1	20
19	0	If FTI=`0-60.23` and TSH=`23.9-47.79` and age=`20.29-39.58` then class=`hypothyroid`	1	7
20	0	If TSH=`288` then class=`hypothyroid`	1	1
21	0	If TSH=`145` then class=`hypothyroid`	1	3
22	0	If TSH=`71.69-95.58` and age=`58.87-78.16` then class=`hypothyroid`	1	1
23	0	If TT4=`6` then class=`hypothyroid`	1	2
24	0	If FTI=`0-60.23` and TSH_measured=`y` and age=`39.58-58.87` and T4U=`0.91-1.13` and TT4=`10-55.48` and on_antithyroid_medication=`f` then class=`hypothyroid`	0.72727	11
25	0	If thyroid_surgery=`t` and age=`1-20.29` and FTI=`60.23-99` then class=`hypothyroid`	0.66667	3
26	0	If TT4=`230` and age=`20.29-39.58` and query_hypothyroid=`f` then class=`hypothyroid`	1	1
27	0	If TT4=`4` and age=`20.29-39.58` then class=`hypothyroid`	1	2
28	0	If TSH=`95.58-96` then class=`hypothyroid`	1	1
29	0	If FTI=`0-60.23` and TSH_measured=`y` and on_thyroxine=`f` and sick=`f` and sex=`M` and age=`39.58-58.87` and query_hyperthyroid=`f` then class=`hypothyroid`	0.75	8
30	0	If TSH=`109` then class=`hypothyroid`	1	1
31	0	If T4U=`1.13-1.36` and TT4=`55.48-99` and age=`58.87-78.16` and T3=`1.99-2.99` then class=`hypothyroid`	0.5	2

32	0	If TT4=`10-55.48` and T3=`0-1` and TSH=`23.9-47.79` then class=`hypothyroid`	1	12
33	0	If FTI=`0-60.23` and T4U=`0.45-0.68` and sick=`f` then class=`hypothyroid`	1	2
34	0	If age=`?` then class=`hypothyroid`	0.02924	342
35	0	If TT4=`7.50` then class=`hypothyroid`	1	1
36	0	If TSH=`143` then class=`hypothyroid`	1	1
37	0	If FTI=`0-60.23` and T4U=`1.13-1.36` and sex=`M` then class=`hypothyroid`	1	5
38	0	If TT4=`10-55.48` and T3=`0-1` and sex=`F` and T4U=`0.68-0.91` and age=`58.87-78.16` then class=`hypothyroid`	0.66667	3
39	0	If FTI=`0-60.23` and thyroid_surgery=`t` and age=`20.29-39.58` then class=`hypothyroid`	1	1
40	0	If TT4=`10-55.48` and thyroid_surgery=`t` then class=`hypothyroid`	0.85714	7
41	0	If FTI=`60.23-99` and age=`20.29-39.58` and sex=`M` and T3=`1-1.99` and T4U=`0.91-1.13` then class=`hypothyroid`	0.14286	7
42	0	If goitre=`t` and TT4=`55.48-99` and T3=`?` and age=`39.58-58.87` and TSH=`0-23.9` then class=`hypothyroid`	1	1
43	0	If TSH=`530` then class=`hypothyroid`	1	1
44	0	If FTI=`0-60.23` and thyroid_surgery=`t` then class=`hypothyroid`	0.75	8
45	0	If TSH=`165` then class=`hypothyroid`	1	1
46	0	If query_hypothyroid=`t` and T4U=`0.68-0.91` and T3=`1.99-2.99` and age=`58.87-78.16` then class=`hypothyroid`	1	2
47	0	If TT4=`109` and TSH=`23.9-47.79` then class=`hypothyroid`	1	1
48	0	If TSH=`117` then class=`hypothyroid`	1	1
49	0	If TSH=`71.69-95.58` and age=`1-20.29` then class=`hypothyroid`	1	1
50	0	If T3=`3.99-4.98` then class=`negative`	1	43
51	0	If TSH=`0-23.9` and age=`?` and sex=`M` then class=`negative`	1	78
52	0	If FTI=`121` then class=`negative`	1	30
53	0	If TSH=`0-23.9` then class=`negative`	0.98279	2034
54	0	If TSH=`?` then class=`negative`	1	371
55	0	If T3=`1-1.99` and TSH=`23.9-47.79` and query_hypothyroid=`f` and on_thyroxine=`f` and age=`?` then class=`negative`	1	2

56	0	If T3=`1-1.99` and TSH=`23.9-47.79` and query_hypothyroid=`f` and age=`58.87-78.16` and sex=`F` then class=`negative`	0.8	5
57	0	If TSH=`23.9-47.79` and T3=`1-1.99` and query_hypothyroid=`f` and on_thyroxine=`f` and thyroid_surgery=`f` then class=`negative`	0.76923	13
58	0	If query_hyperthyroid=`f` then class=`negative`	0.9503	2334

Table 28 Thyroid Example Traditional Model

	Cluster	Rules	Accuracy	Coverage
1	0	If TSH=`164` then class=`hypothyroid`	1	1
2	0	If FTI=`0-60.23` and TSH=`23.9-47.79` and T3=`1.99-2.99` then class=`hypothyroid`	1	6
3	0	If FTI=`0-60.23` and age=`58.87-78.16` and T4U=`1.36-1.59` and tumor=`f` then class=`hypothyroid`	1	4
4	0	If TSH=`150` then class=`hypothyroid`	1	2
5	0	If TT4=`5.30` then class=`hypothyroid`	1	1
6	0	If TSH=`47.79-71.69` and T3=`0-1` and query_hypothyroid=`f` then class=`hypothyroid`	1	13
7	0	If FTI=`0-60.23` and TSH=`23.9-47.79` and T3=`0-1` then class=`hypothyroid`	1	13
8	0	If FTI=`0-60.23` and age=`78.16-97.45` and on_thyroxine=`f` then class=`hypothyroid`	1	6
9	0	If thyroid_surgery=`t` and age=`1-20.29` and T3=`1-1.99` then class=`hypothyroid`	1	3
10	0	If query_hypothyroid=`t` and age=`78.16-97.45` and T3=`1.99-2.99` and FTI=`60.23-99` then class=`hypothyroid`	1	2
11	0	If TSH=`235` then class=`hypothyroid`	1	1
12	0	If TSH=`216` then class=`hypothyroid`	1	1
13	0	If FTI=`0-60.23` and age=`58.87-78.16` and TT4=`10-55.48` and T3=`1-1.99` then class=`hypothyroid`	1	10
14	0	If TSH=`213` then class=`hypothyroid`	1	1
15	0	If T4U=`0.91-1.13` and on_thyroxine=`f` and query_hypothyroid=`t` and TT4=`55.48-99` and T3=`1-1.99` and age=`58.87-78.16` and sex=`F` then class=`hypothyroid`	0.5	2
16	0	If T4U=`0.91-1.13` and age=`39.58-58.87` and query_hypothyroid=`t` and TT4=`55.48-99` and T3=`1.99-2.99` then class=`hypothyroid`	1	1
17	0	If TSH=`100` and on_thyroxine=`f` then class=`hypothyroid`	1	3

18	0	If TSH=`47.79-71.69` and FTI=`0-60.23` and on_antithyroid_medication=`f` and query_hypothyroid=`f` then class=`hypothyroid`	1	20
19	0	If FTI=`0-60.23` and TSH=`23.9-47.79` and age=`20.29-39.58` then class=`hypothyroid`	1	7
20	0	If TSH=`288` then class=`hypothyroid`	1	1
21	0	If TSH=`145` then class=`hypothyroid`	1	3
22	0	If TSH=`71.69-95.58` and age=`58.87-78.16` then class=`hypothyroid`	1	1
23	0	If TT4=`6` then class=`hypothyroid`	1	2
24	0	If FTI=`0-60.23` and TSH_measured=`y` and age=`39.58-58.87` and T4U=`0.91-1.13` and TT4=`10-55.48` and on_antithyroid_medication=`f` then class=`hypothyroid`	0.72727	11
25	0	If thyroid_surgery=`t` and age=`1-20.29` then class=`hypothyroid`	0.66667	6
26	0	If TT4=`230` and age=`20.29-39.58` and query_hypothyroid=`f` then class=`hypothyroid`	1	1
27	0	If TT4=`4` and age=`20.29-39.58` then class=`hypothyroid`	1	2
28	0	If TSH=`95.58-96` then class=`hypothyroid`	1	1
29	0	If FTI=`0-60.23` and TSH_measured=`y` and on_thyroxine=`f` and sick=`f` and sex=`M` and age=`39.58-58.87` and query_hyperthyroid=`f` then class=`hypothyroid`	0.75	8
30	0	If TSH=`109` then class=`hypothyroid`	1	1
31	0	If T4U=`1.13-1.36` and TT4=`55.48-99` and age=`58.87-78.16` and T3=`1.99-2.99` then class=`hypothyroid`	0.5	2
32	0	If TT4=`10-55.48` and T3=`0-1` and TSH=`23.9-47.79` then class=`hypothyroid`	1	12
33	0	If FTI=`0-60.23` and T4U=`0.45-0.68` and sick=`f` then class=`hypothyroid`	1	2
34	0	If age=`?` then class=`hypothyroid`	0.02924	342
35	0	If TT4=`7.50` then class=`hypothyroid`	1	1
36	0	If TSH=`143` then class=`hypothyroid`	1	1
37	0	If FTI=`0-60.23` and T4U=`1.13-1.36` and sex=`M` then class=`hypothyroid`	1	5
38	0	If TT4=`10-55.48` and T3=`0-1` and sex=`F` and T4U=`0.68-0.91` and age=`58.87-78.16` then class=`hypothyroid`	0.66667	3
39	0	If FTI=`0-60.23` and thyroid_surgery=`t` and age=`20.29-39.58` then class=`hypothyroid`	1	1
40	0	If TT4=`10-55.48` and thyroid_surgery=`t` then class=`hypothyroid`	0.85714	7

41	0	If FTI=`60.23-99` and age=`20.29-39.58` and sex=`M` and T3=`1-1.99` and T4U=`0.91-1.13` then class=`hypothyroid`	0.14286	7
42	0	If goitre=`t` and T3=`?` and TT4=`55.48-99` and age=`39.58-58.87` then class=`hypothyroid`	0.5	2
43	0	If TSH=`530` then class=`hypothyroid`	1	1
44	0	If FTI=`0-60.23` and thyroid_surgery=`t` then class=`hypothyroid`	0.75	8
45	0	If TSH=`165` then class=`hypothyroid`	1	1
46	0	If query_hypothyroid=`t` and T4U=`0.68-0.91` and T3=`1.99-2.99` and age=`58.87-78.16` then class=`hypothyroid`	1	2
47	0	If TT4=`109` and TSH=`23.9-47.79` then class=`hypothyroid`	1	1
48	0	If TSH=`117` then class=`hypothyroid`	1	1
49	0	If TSH=`71.69-95.58` and age=`1-20.29` then class=`hypothyroid`	1	1
50	0	If FTI=`124` then class=`negative`	1	26
51	0	If FTI=`109` then class=`negative`	1	31
52	0	If TSH=`0-23.9` and T4U=`0.68-0.91` and sex=`M` then class=`negative`	1	259
53	0	If T3=`3.99-4.98` then class=`negative`	1	43
54	0	If TSH=`0-23.9` and age=`?` and sex=`M` then class=`negative`	1	78
55	0	If FTI=`121` then class=`negative`	1	30
56	0	If TSH=`0-23.9` then class=`negative`	0.98279	2034
57	0	If TSH=`?` then class=`negative`	1	371
58	0	If T3=`1-1.99` and TSH=`23.9-47.79` and query_hypothyroid=`f` and on_thyroxine=`f` and age=`?` then class=`negative`	1	2
59	0	If T3=`1-1.99` and TSH=`23.9-47.79` and query_hypothyroid=`f` and age=`58.87-78.16` and sex=`F` then class=`negative`	0.8	5
60	0	If TSH=`23.9-47.79` and T3=`1-1.99` and query_hypothyroid=`f` and on_thyroxine=`f` and thyroid_surgery=`f` then class=`negative`	0.76923	13
61	0	If query_hyperthyroid=`f` then class=`negative`	0.9503	2334
62	1	If age=`58.87-78.16` then class=`hypothyroid`	0.06417	748
63	1	If FTI_measured=`y` then class=`negative`	0.9482	2336
64	2	If age=`20.29-39.58` then class=`hypothyroid`	0.03978	553
65	2	If lithium=`f` then class=`negative`	0.95217	2530

Table 29 Thyroid Example Collective Model

Results for Mushroom Data

Tables 30 through 34 display the results of the test performed using the Mushroom data set. The traditional models achieved accuracies of 89% and 100% for the unordered and ordered models respectively. The CRA process generated a collective model that achieved a higher accuracy than the traditional method with the occurrence of 2, 3, and 4 missing attributes. Figures 31 and 32 provide a graphical representation of the accuracies achieved for each *k*. Additionally, the CRA method matched the performance of the traditional method with the occurrence of 0 and 3 missing attributes.

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.89	1	0	23.5	33.4	2.06
3	.89	.9	0	28.4	32.6	1.54
5	.88	.91	0	26.4	29.7	1.41
7	.87	.89	0	25.8	28.2	1.36
9	.85	.9	0	25.2	27.9	1.36
11	.85	.91	0	24.5	27.1	1.31
13	.8	.91	0	25.2	27.7	1.31
15	.76	.9	0	25.8	28.1	1.35

Table 30 Mushroom Results Complete Data (No Missing Attributes)

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.89	.96	40.8	23	33.4	2.01
3	.87	.92	.3	26.7	32.1	1.56
5	.86	.89	.1	25.2	29.3	1.45
7	.82	.87	0	25.3	28.7	1.41
9	.77	.87	0	23.8	27.3	1.36
11	.81	.9	0	23.3	25.5	1.31
13	.81	.9	0	23.6	26.3	1.31
15	.8	.9	0	24.4	26.9	1.35

Table 31 Mushroom Results 1 Missing Attribute

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.86	.92	75.1	22.2	31.3	1.95
3	.87	.88	2.3	28.9	36	1.73
5	.8	.87	.2	25.4	30.2	1.47
7	.78	.87	0	24.1	27	1.37
9	.79	.86	0	22.8	25	1.28
11	.77	.88	0	23.2	25.4	1.32
13	.76	.87	0	24.8	26.7	1.32
15	.76	.87	0	24.4	27.1	1.36

Table 32 Mushroom Results 2 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.8	.86	157	21.3	30.2	1.92
3	.82	.85	1.7	28	34.1	1.69
5	.8	.88	.7	25.9	29.8	1.49
7	.76	.81	0	25.6	28.7	1.38
9	.76	.86	0	24.6	27.1	1.28
11	.76	.86	.2	25.2	27.4	1.29
13	.76	.86	0	26.5	28.4	1.31
15	.77	.86	.1	25.9	27.9	1.29

Table 33 Mushroom Results 3 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.77	.82	201.7	24.3	35.2	2.06
3	.82	.87	2.4	25.1	29.9	1.52
5	.8	.86	1	28.4	32.5	1.59
7	.75	.87	.2	25.8	26.9	1.36
9	.74	.84	.3	23.6	26.8	1.36
11	.75	.85	.5	22.6	25.7	1.4
13	.71	.85	.2	23.5	27	1.39
15	.7	.85	.1	24.6	28	1.39

Table 34 Mushroom Results 4 Missing Attributes

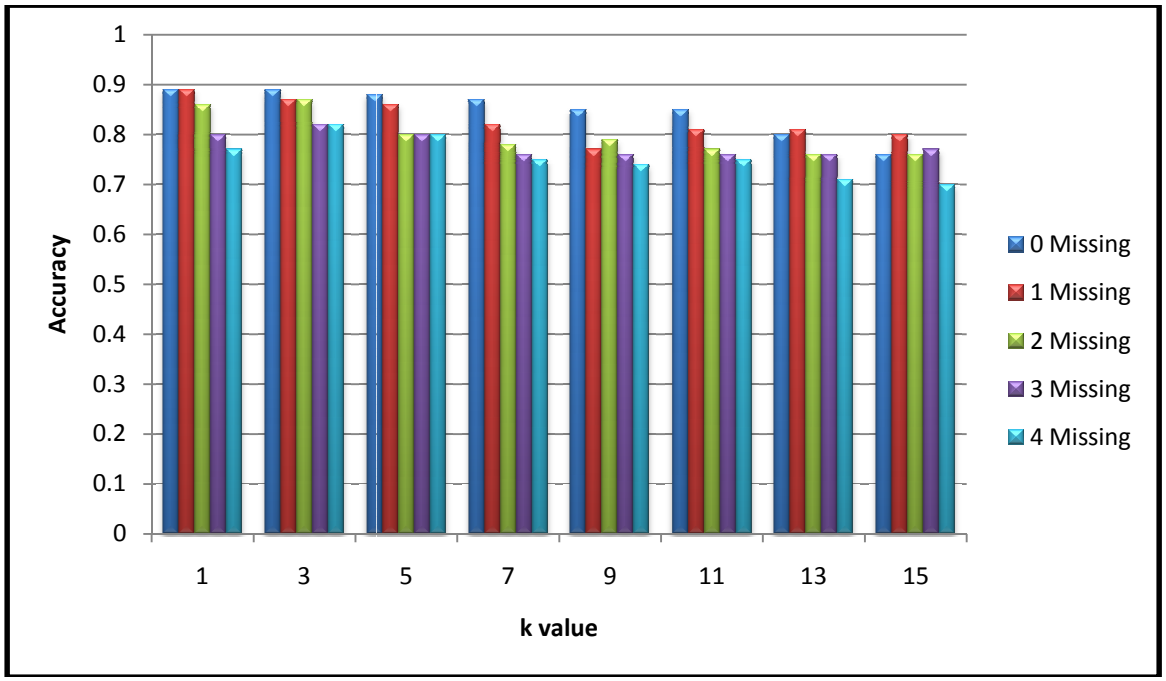


Figure 31Mushroom Accuracies Unordered Models

Figure 33 depicts the average rule lengths of the models generated. The collective models created with the CRA process contained smaller rules. On average, the rule lengths were 2 and 1.4 for the traditional and collective models respectively. The models created via the CRA method were generally larger than the traditional models. The collective models contained 25 rules while the traditional model contained 22 on average. Tables 35 and 36 display examples of the traditional and collective models created using this data set.

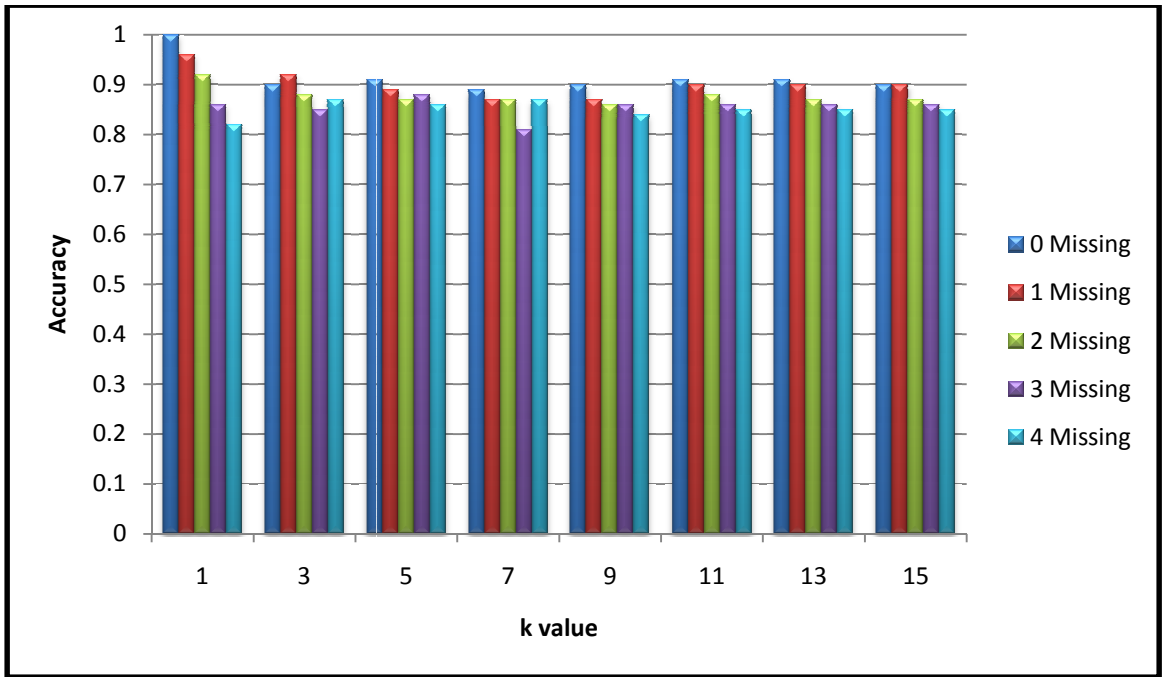


Figure 32 Mushroom Accuracies Ordered Models

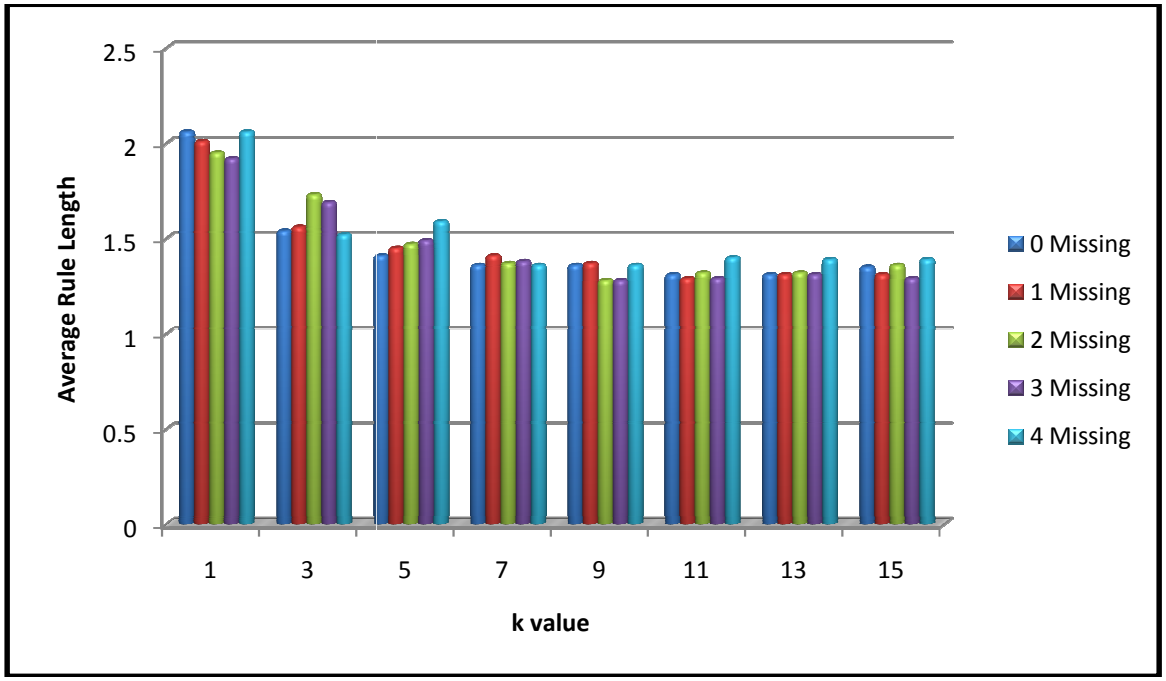


Figure 33 Mushroom Average Rule Lengths

	Cluster	Rules	Accuracy	Coverage
1	0	If odor=`a` then class=`e`	1	319
2	0	If odor=`n` and stalk_shape=`t` then class=`e`	1	2020
3	0	If habitat=`w` then class=`e`	1	145

4	0	If odor=`l` then class=`e`	1	317
5	0	If stalk_color_above_ring=`o` then class=`e`	1	143
6	0	If ring_number=`t` and spore_print_color=`w` then class=`e`	1	417
7	0	If odor=`n` and stalk_surface_below_ring=`f` then class=`e`	1	362
8	0	If cap_shape=`s` then class=`e`	1	25
9	0	If odor=`n` and habitat=`u` then class=`e`	1	72
10	0	If cap_color=`c` and gill_size=`n` then class=`e`	1	21
11	0	If gill_spacing=`w` and cap_color=`n` then class=`e`	1	223
12	0	If cap_color=`y` then class=`p`	0.63392	855
13	0	If cap_surface=`s` and gill_spacing=`c` and gill_attachment=`f` and ring_number=`o` then class=`p`	0.8406	1261
14	0	If gill_size=`n` and stalk_shape=`t` then class=`p`	0.9455	1468
15	0	If habitat=`p` and ring_number=`o` then class=`p`	0.91551	864
16	0	If stalk_shape=`e` and population=`y` and habitat=`g` and gill_color=`p` then class=`p`	0.81081	74
17	0	If stalk_shape=`e` and ring_number=`o` and habitat=`g` then class=`p`	0.6118	729
18	0	If habitat=`m` and gill_color=`w` then class=`p`	0.15789	57
19	0	If population=`v` and stalk_color_above_ring=`w` and ring_type=`p` and cap_shape=`x` and ring_number=`o` and gill_attachment=`f` and stalk_surface_above_ring=`s` and stalk_surface_below_ring=`s` and veil_color=`w` and veil_type=`p` and cap_surface=`f` and habitat=`d` and stalk_root=`b` and gill_color=`u` and spore_print_color=`k` then class=`p`	0.41667	12
20	0	If stalk_root=`b` and cap_shape=`x` then class=`p`	0.50772	1554
21	0	If cap_color=`g` and bruises=`f` and ring_number=`o` then class=`p`	0.69004	813
22	0	If population=`v` and stalk_color_below_ring=`w` and stalk_color_above_ring=`w` and bruises=`t` then class=`p`	0.60682	440
23	0	If population=`s` and stalk_surface_below_ring=`s` and stalk_root=`e` then class=`p`	0.40892	269
24	0	If habitat=`d` then class=`p`	0.39825	2521
25	0	If ring_number=`o` and gill_spacing=`w` and stalk_surface_below_ring=`s` and stalk_surface_above_ring=`s` then class=`p`	0.23939	330

Table 35 Mushroom Example Traditional Model

	Cluster	Rules	Accuracy	Coverage
1	0	If odor=`n` and stalk_shape=`t` then class=`e`	1	2020
2	0	If population=`y` then class=`e`	0.62127	1373
3	0	If odor=`n` and cap_color=`g` then class=`e`	1	834
4	0	If habitat=`w` then class=`e`	1	145
5	0	If stalk_color_below_ring=`n` and gill_spacing=`w` then class=`e`	1	39
6	0	If odor=`n` and stalk_surface_below_ring=`k` then class=`e`	1	120
7	0	If cap_shape=`s` then class=`e`	1	25
8	0	If odor=`l` then class=`e`	1	317
9	0	If odor=`n` and cap_shape=`x` then class=`e`	0.99442	1255
10	0	If odor=`a` then class=`e`	1	319
11	0	If ring_type=`f` then class=`e`	1	39
12	0	If ring_number=`t` and spore_print_color=`w` then class=`e`	1	417
13	0	If odor=`n` and habitat=`u` then class=`e`	1	72
14	0	If stalk_surface_below_ring=`k` then class=`p`	0.9346	1835
15	0	If population=`v` and stalk_color_below_ring=`w` and cap_color=`e` then class=`p`	0.82641	409
16	0	If stalk_color_below_ring=`p` and stalk_color_above_ring=`p` and cap_color=`e` then class=`p`	0.78112	233
17	0	If stalk_color_above_ring=`w` and gill_spacing=`c` then class=`p`	0.50811	2527
18	0	If gill_size=`n` and bruises=`f` and cap_shape=`x` then class=`p`	0.91828	673
19	0	If ring_type=`e` then class=`p`	0.63653	2234
20	0	If gill_color=`w` then class=`p`	0.20329	974
21	1	If ring_type=`p` then class=`e`	0.79388	3168
22	1	If veil_color=`w` then class=`p`	0.49228	6348
23	1	If bruises=`f` then class=`p`	0.69286	3793
24	2	If ring_number=`o` then class=`e`	0.49175	5997
25	2	If spore_print_color=`w` then class=`e`	0.23862	1911
26	2	If bruises=`t` then class=`p`	0.18632	2705

Table 36 Mushroom Example Collective Model

Results for Tic-Tac-Toe Data

Tables 37 through 41 display the results of the test performed using the Tic-Tac-Toe data set. The traditional models achieved accuracies of 82% and 99% for the unordered and ordered models respectively. The CRA process generated a collective model that achieved a higher accuracy than the traditional method with the occurrence of 1, 2, 3, and 4 missing attributes. Additionally, the number of instances not classified decreased when using the collective models created with the CRA process. Figures 34 and 35 provide a graphical representation of the accuracies achieved for each *k* while Figure 36 displays the average rule lengths. The rules created using the CRA method were shorter. On average the rule lengths were 3.35 and 2.38 for the traditional and collective models respectively. The CRA models were larger than the traditional models. The collective model contained 97 rules while the traditional model contained 52 rules on average. Tables 42 and 43 provide examples of the traditional and collective models generated.

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.82	.99	.1	54.6	26.8	3.4
3	.79	.96	0	78.9	27	2.73
5	.76	.94	0	89.7	27	2.54
7	.78	.9	0	101.2	26.9	2.42
9	.76	.86	0	107.5	27	2.3
11	.75	.85	0	106.8	27	2.27
13	.75	.84	0	104	27	2.2
15	.74	.82	0	101.9	27	2.13

Table 37 Tic-Tac-Toe Results Complete Data (No Missing Attributes)

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.75	.84	11.5	52.1	26.6	3.33

3	.75	.85	.1	80.9	26.9	2.72
5	.74	.83	0	88.3	27	2.6
7	.74	.81	0	105	27	2.42
9	.74	.8	0	106	27	2.32
11	.73	.8	0	99.9	26.9	2.23
13	.71	.78	0	98.5	26.9	2.16
15	.71	.78	0	98.3	27	2.12

Table 38 Tic-Tac-Toe Results 1 Missing Attribute

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.68	.73	21.3	49.7	26.7	3.31
3	.72	.79	.6	78.2	27	2.76
5	.72	.78	0	87.9	27	2.58
7	.72	.77	0	98.1	27	2.42
9	.72	.75	0	106.1	27	2.34
11	.71	.75	0	105.7	27	2.26
13	.7	.74	0	105.1	27	2.18
15	.69	.73	0	102.3	27	2.14

Table 39 Tic-Tac-Toe Results 2 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.57	.58	46.6	49.7	26.5	3.36
3	.68	.72	3.7	80.9	27	2.77
5	.7	.73	.5	86.8	27	2.62
7	.69	.72	.2	96.6	27	2.43
9	.69	.72	0	103.9	27	2.35
11	.69	.72	0	104.2	27	2.28
13	.67	.73	0	99.7	27	2.16
15	.66	.71	0	100.8	27	2.1

Table 40 Tic-Tac-Toe Results 3 Missing Attributes

Number Clusters	Unordered Accuracy	Ordered Accuracy	Average Number Not Classified	Average Number of Rules	Average Number Unique Attributes	Average Rule Length
1	.48	.49	67.6	52.2	26.8	3.34
3	.66	.68	8	77	26.8	2.75

5	.68	.7	2.4	88.3	27	2.57
7	.68	.71	.3	102.6	27	2.42
9	.69	.71	0	109.8	27	2.32
11	.68	.7	0	109.1	27	2.28
13	.67	.7	0	102.1	26.9	2.2
15	.66	.7	0	99.6	27	2.14

Table 41 Tic-Tac-Toe Results 4 Missing Attributes

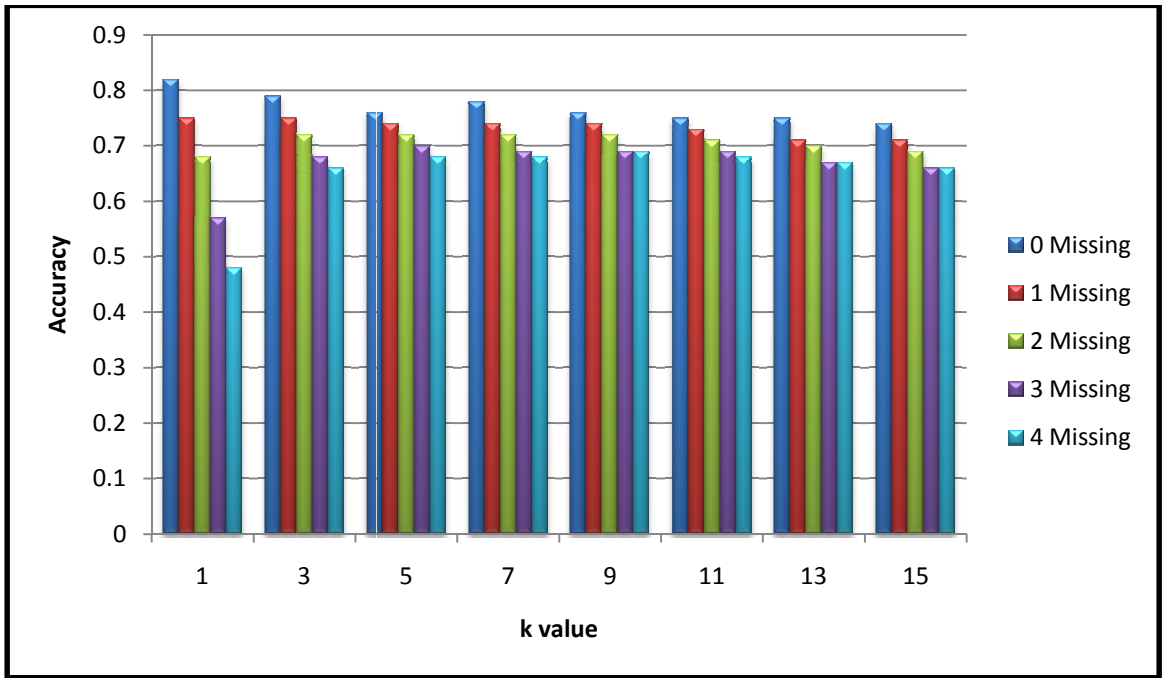


Figure 34 Tic-Tac-Toe Accuracies Unordered Models

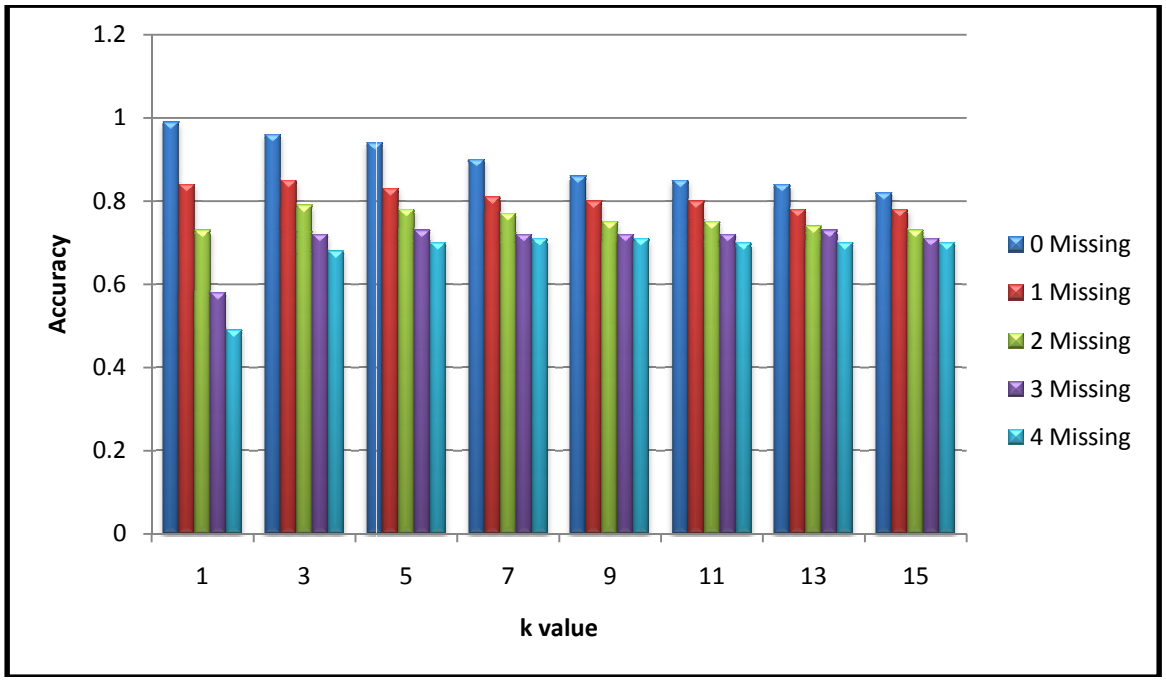


Figure 35 Tic-Tac-Toe Accuracies Ordered Models

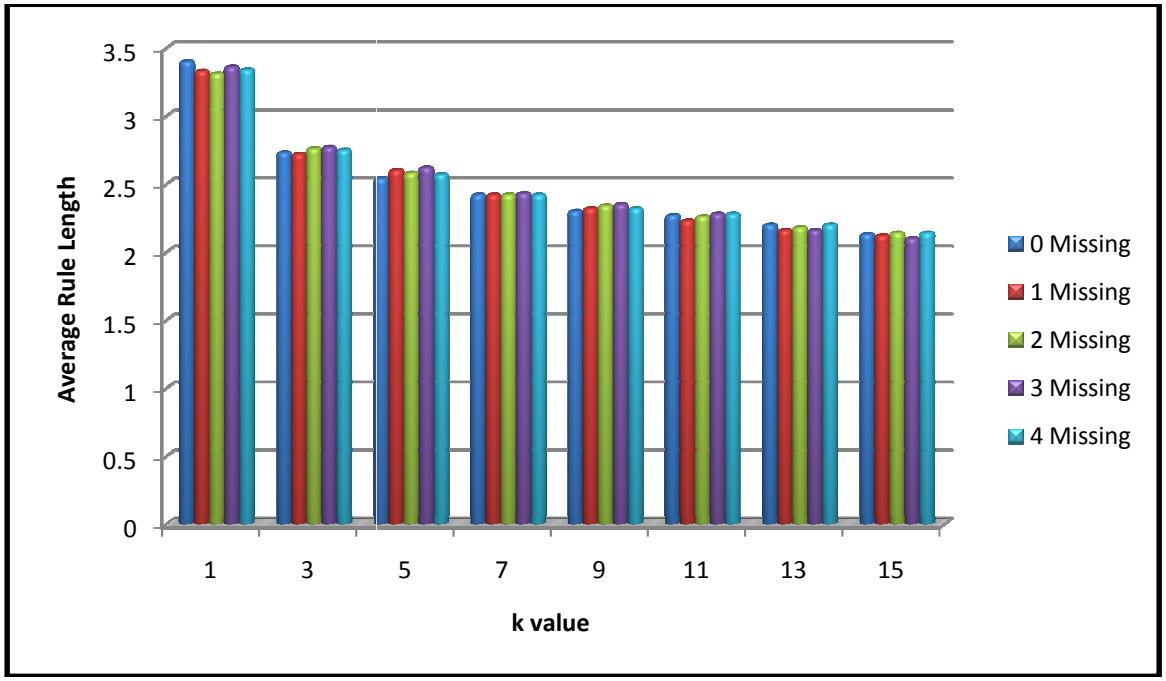


Figure 36 Tic-Tac-Toe Average Rule Lengths

	Cluster	Rules	Accuracy	Coverage
1	0	If bottom_left_square=`o` and top_left_square=`o` and middle_left_square=`o` then class=`negative`	1	32

2	0	If middle_middle_square=`o` and bottom_middle_square=`o` and top_middle_square=`o` then class=`negative`	1	32
3	0	If middle_middle_square=`o` and bottom_left_square=`o` and top_right_square=`o` then class=`negative`	1	38
4	0	If bottom_right_square=`o` and bottom_left_square=`o` and bottom_middle_square=`o` then class=`negative`	1	30
5	0	If top_left_square=`o` and top_right_square=`o` and top_middle_square=`o` then class=`negative`	1	28
6	0	If bottom_right_square=`o` and top_right_square=`o` and middle_right_square=`o` then class=`negative`	1	28
7	0	If bottom_right_square=`o` and bottom_middle_square=`x` and top_right_square=`x` and middle_left_square=`x` and top_middle_square=`o` and bottom_left_square=`o` and top_left_square=`x` then class=`negative`	1	2
8	0	If bottom_right_square=`o` and bottom_middle_square=`x` and middle_right_square=`x` and top_right_square=`x` and top_middle_square=`o` and top_left_square=`x` then class=`negative`	1	3
9	0	If middle_middle_square=`o` and top_left_square=`o` and bottom_right_square=`o` then class=`negative`	1	39
10	0	If middle_right_square=`o` and bottom_middle_square=`x` and top_right_square=`x` and bottom_right_square=`x` and middle_left_square=`x` then class=`negative`	1	2
11	0	If middle_middle_square=`o` and middle_right_square=`o` and middle_left_square=`o` then class=`negative`	1	27
12	0	If top_right_square=`o` and middle_left_square=`o` and bottom_right_square=`o` and bottom_left_square=`x` then class=`negative`	1	5
13	0	If top_right_square=`o` and bottom_middle_square=`o` and middle_middle_square=`o` and bottom_right_square=`x` then class=`negative`	1	7
14	0	If top_right_square=`o` and top_left_square=`o` and bottom_middle_square=`o` and middle_left_square=`o` then class=`negative`	1	1
15	0	If middle_middle_square=`b` and middle_right_square=`o` then class=`positive`	0.8	50
16	0	If middle_middle_square=`x` and bottom_left_square=`x` and bottom_middle_square=`o` then class=`positive`	0.93103	58

17	0	If middle_middle_square=`x` and bottom_left_square=`b` then class=`positive`	0.84146	82
18	0	If bottom_middle_square=`b` and top_middle_square=`o` then class=`positive`	0.82432	74
19	0	If bottom_right_square=`x` and bottom_middle_square=`x` and top_middle_square=`o` and middle_middle_square=`o` then class=`positive`	0.80952	21
20	0	If middle_middle_square=`x` and bottom_left_square=`x` and top_left_square=`o` and bottom_right_square=`o` then class=`positive`	0.95652	23
21	0	If bottom_left_square=`x` and top_right_square=`o` and middle_left_square=`x` and bottom_right_square=`b` and middle_middle_square=`o` then class=`positive`	0.875	8
22	0	If middle_middle_square=`b` and bottom_left_square=`x` and middle_right_square=`b` then class=`positive`	0.89474	19
23	0	If middle_middle_square=`x` and bottom_right_square=`b` and top_right_square=`o` then class=`positive`	0.74074	27
24	0	If bottom_right_square=`x` and middle_middle_square=`b` and bottom_left_square=`b` then class=`positive`	0.76923	13
25	0	If bottom_right_square=`x` and middle_middle_square=`b` and middle_right_square=`x` then class=`positive`	0.84375	32
26	0	If middle_middle_square=`x` and top_left_square=`b` and middle_left_square=`o` then class=`positive`	0.96154	26
27	0	If middle_middle_square=`x` and top_right_square=`b` and middle_left_square=`x` then class=`positive`	0.91667	36
28	0	If top_middle_square=`x` and bottom_middle_square=`o` then class=`positive`	0.6383	94
29	0	If middle_middle_square=`x` and middle_left_square=`b` and top_right_square=`b` then class=`positive`	0.84211	19
30	0	If bottom_right_square=`x` and middle_left_square=`b` and bottom_middle_square=`x` then class=`positive`	0.79487	39
31	0	If bottom_right_square=`x` and top_right_square=`o` and bottom_middle_square=`x` and top_middle_square=`x` and top_left_square=`o` then class=`positive`	0.77778	9
32	0	If top_left_square=`x` and middle_middle_square=`b` then class=`positive`	0.82609	69
33	0	If middle_left_square=`b` and bottom_left_square=`o` then class=`positive`	0.67123	73

34	0	If middle_middle_square=`x` and bottom_left_square=`x` then class=`positive`	0.85606	132
35	0	If middle_middle_square=`x` and top_right_square=`o` and middle_left_square=`x` and bottom_right_square=`x` then class=`positive`	0.84211	19
36	0	If bottom_middle_square=`b` then class=`positive`	0.69697	198
37	0	If top_middle_square=`b` and bottom_right_square=`x` and middle_right_square=`x` and middle_middle_square=`o` and top_left_square=`o` then class=`positive`	0.88889	9
38	0	If top_right_square=`o` and bottom_left_square=`x` then class=`positive`	0.64655	116
39	0	If middle_middle_square=`x` and top_left_square=`x` then class=`positive`	0.82353	136
40	0	If top_left_square=`x` and bottom_right_square=`o` and top_middle_square=`x` then class=`positive`	0.64407	59
41	0	If top_left_square=`b` and bottom_left_square=`o` and bottom_right_square=`x` then class=`positive`	0.8	25
42	0	If middle_left_square=`x` and bottom_middle_square=`x` and top_left_square=`b` then class=`positive`	0.57895	19
43	0	If top_middle_square=`b` and bottom_right_square=`x` and middle_middle_square=`o` and middle_right_square=`x` and middle_left_square=`o` then class=`positive`	0.88889	9
44	0	If top_middle_square=`b` and bottom_middle_square=`o` then class=`positive`	0.85366	82
45	0	If middle_right_square=`o` and middle_left_square=`x` and top_left_square=`x` and bottom_middle_square=`x` then class=`positive`	0.52941	17
46	0	If bottom_middle_square=`x` and top_left_square=`o` and top_right_square=`x` and bottom_left_square=`o` and bottom_right_square=`o` and top_middle_square=`x` then class=`positive`	0.33333	3
47	0	If middle_right_square=`b` and top_right_square=`b` and middle_left_square=`o` then class=`positive`	0.83333	18
48	0	If middle_middle_square=`x` then class=`positive`	0.79109	359
49	0	If bottom_right_square=`x` and top_left_square=`o` and top_right_square=`b` and bottom_middle_square=`x` then class=`positive`	0.81818	22
50	0	If bottom_middle_square=`x` and middle_middle_square=`o` and middle_left_square=`o` and top_left_square=`o` then class=`positive`	0.42857	14

51	0	If bottom_middle_square=`x` then class=`positive`	0.58863	299

Table 42 Tic-Tac-Toe Example Traditional Model

RuleID	Cluster	Rules	Accuracy	Coverage
1	0	If top_right_square=`o` and bottom_right_square=`o` and bottom_left_square=`b` then class=`negative`	0.57895	19
2	0	If bottom_middle_square=`o` then class=`positive`	0.69259	270
3	0	If middle_right_square=`o` then class=`positive`	0.70722	263
5	1	If bottom_right_square=`o` and bottom_middle_square=`o` then class=`negative`	0.45977	87
6	1	If middle_middle_square=`o` and bottom_middle_square=`o` then class=`negative`	0.5974	77
7	1	If bottom_right_square=`o` and top_left_square=`x` and bottom_middle_square=`x` then class=`negative`	0.5	36
8	1	If top_left_square=`o` and top_middle_square=`o` and top_right_square=`o` then class=`negative`	1	28
9	1	If middle_middle_square=`x` then class=`positive`	0.79109	359
10	1	If bottom_right_square=`x` then class=`positive`	0.71554	341
11	2	If middle_left_square=`o` and bottom_right_square=`b` then class=`negative`	0.38182	55
12	2	If top_middle_square=`x` and middle_middle_square=`o` then class=`negative`	0.6036	111
13	2	If bottom_middle_square=`x` and top_middle_square=`b` and bottom_left_square=`o` and bottom_right_square=`b` then class=`negative`	0.85714	7
14	2	If bottom_left_square=`b` and bottom_middle_square=`x` then class=`negative`	0.51724	58
15	2	If middle_middle_square=`b` and bottom_right_square=`o` then class=`negative`	0.44444	45
16	2	If bottom_right_square=`o` and top_right_square=`x` then class=`negative`	0.37879	132
17	2	If top_left_square=`x` and top_right_square=`x` then class=`negative`	0.23256	129
18	2	If top_left_square=`x` and bottom_left_square=`o` and middle_middle_square=`o` then class=`negative`	0.71111	45
19	2	If bottom_right_square=`o` and middle_right_square=`o` then class=`negative`	0.45783	83
20	2	If middle_right_square=`b` and top_left_square=`o` then class=`negative`	0.51471	68
21	2	If top_middle_square=`o` and	0.62963	135

		middle_right_square=`x` then class=`positive`		
22	2	If middle_right_square=`x` and top_middle_square=`b` and top_right_square=`x` and bottom_left_square=`o` and middle_middle_square=`o` then class=`positive`	0.83333	6
23	2	If bottom_right_square=`x` and top_left_square=`o` and middle_left_square=`x` then class=`positive`	0.52778	36
24	2	If top_left_square=`b` and middle_left_square=`b` and bottom_right_square=`x` then class=`positive`	0.80952	21
25	2	If middle_middle_square=`x` and bottom_left_square=`o` then class=`positive`	0.70345	145
26	2	If bottom_right_square=`b` and top_right_square=`o` then class=`positive`	0.5614	57
27	2	If bottom_middle_square=`b` and bottom_right_square=`o` then class=`positive`	0.72603	73
28	2	If bottom_left_square=`x` and top_middle_square=`o` then class=`positive`	0.72078	154
29	2	If bottom_left_square=`x` and top_middle_square=`b` and middle_middle_square=`o` then class=`positive`	0.70455	44
30	3	If middle_right_square=`o` and top_right_square=`o` and bottom_right_square=`o` then class=`negative`	1	28
31	3	If middle_left_square=`o` and bottom_left_square=`x` and bottom_middle_square=`x` then class=`negative`	0.24561	57
32	3	If middle_middle_square=`b` then class=`negative`	0.28467	137
33	3	If bottom_middle_square=`o` and bottom_left_square=`o` then class=`negative`	0.45349	86
34	3	If middle_middle_square=`o` and top_left_square=`o` and bottom_right_square=`o` then class=`negative`	1	39
35	3	If bottom_right_square=`x` and middle_left_square=`o` then class=`negative`	0.26923	156
36	3	If bottom_middle_square=`b` then class=`positive`	0.69697	198
37	3	If middle_right_square=`b` and bottom_left_square=`b` then class=`positive`	0.625	32
38	3	If top_right_square=`b` and bottom_middle_square=`x` then class=`positive`	0.73418	79
39	3	If middle_middle_square=`x` and middle_left_square=`b` then class=`positive`	0.80208	96
40	3	If middle_right_square=`b` and top_left_square=`x` then class=`positive`	0.79	100
41	3	If top_right_square=`o` and	0.34568	81

		middle_middle_square=`o` then class=`positive`		
42	3	If bottom_left_square=`x` then class=`positive`	0.727	337
43	4	If middle_left_square=`x` and middle_right_square=`b` then class=`negative`	0.49333	75
44	4	If top_middle_square=`o` and bottom_right_square=`b` then class=`negative`	0.43103	58
45	4	If top_left_square=`o` and middle_left_square=`o` and bottom_left_square=`o` then class=`negative`	1	32
46	4	If top_middle_square=`o` and bottom_left_square=`b` then class=`negative`	0.40984	61
47	4	If middle_left_square=`x` and top_left_square=`b` then class=`negative`	0.44776	67
48	4	If top_left_square=`x` then class=`positive`	0.70088	341
49	4	If bottom_middle_square=`x` then class=`positive`	0.58863	299
50	4	If top_middle_square=`x` then class=`positive`	0.5914	279
51	5	If bottom_left_square=`o` and middle_middle_square=`b` and bottom_right_square=`o` then class=`negative`	0.75	16
52	5	If bottom_left_square=`o` and middle_right_square=`o` and bottom_right_square=`o` then class=`negative`	0.5	14
53	5	If bottom_left_square=`b` and middle_right_square=`o` then class=`negative`	0.41818	55
54	5	If bottom_left_square=`o` and top_right_square=`o` and bottom_right_square=`o` then class=`negative`	0.6	15
55	5	If bottom_left_square=`b` and top_middle_square=`o` and middle_middle_square=`o` then class=`negative`	0.77778	18
56	5	If bottom_left_square=`o` and middle_right_square=`b` and bottom_right_square=`o` and bottom_middle_square=`o` then class=`negative`	1	11
57	5	If middle_middle_square=`o` then class=`positive`	0.43911	271
58	5	If middle_right_square=`b` then class=`positive`	0.66834	199
59	5	If top_right_square=`b` then class=`positive`	0.69143	175
60	5	If top_middle_square=`o` then class=`positive`	0.68132	273
61	5	If middle_left_square=`x` and middle_middle_square=`b` then class=`positive`	0.61404	57
62	5	If middle_right_square=`x` then class=`positive`	0.59672	305
63	6	If bottom_left_square=`o` and middle_left_square=`o` then class=`negative`	0.45783	83
64	6	If middle_middle_square=`o` and bottom_right_square=`b` and bottom_middle_square=`o` then class=`negative`	0.5	20

65	6	If top_right_square=`o` and bottom_middle_square=`b` and bottom_right_square=`x` then class=`negative`	0.45946	37
66	6	If top_right_square=`o` and middle_left_square=`b` and top_middle_square=`o` then class=`negative`	0.58824	17
67	6	If middle_middle_square=`o` and bottom_middle_square=`o` and top_middle_square=`o` then class=`negative`	1	32
68	6	If top_middle_square=`x` and bottom_left_square=`o` then class=`negative`	0.46715	137
69	6	If top_right_square=`o` and middle_right_square=`o` then class=`negative`	0.45122	82
70	6	If middle_right_square=`o` and bottom_left_square=`o` then class=`negative`	0.3	60
71	6	If top_middle_square=`x` and top_right_square=`b` then class=`negative`	0.45161	62
72	6	If top_right_square=`o` and top_left_square=`o` and top_middle_square=`o` then class=`negative`	1	28
73	6	If top_left_square=`x` and top_middle_square=`b` then class=`negative`	0.33333	93
74	6	If bottom_right_square=`o` and middle_middle_square=`o` then class=`negative`	0.66667	81
75	6	If top_middle_square=`x` and middle_right_square=`o` then class=`negative`	0.34454	119
76	6	If bottom_left_square=`o` and middle_middle_square=`o` then class=`negative`	0.7013	77
77	6	If bottom_left_square=`o` and bottom_right_square=`o` then class=`negative`	0.5974	77
78	6	If top_middle_square=`x` and bottom_right_square=`x` and middle_left_square=`x` then class=`negative`	0.66667	12
79	6	If top_right_square=`x` and top_left_square=`o` and bottom_left_square=`x` then class=`positive`	0.79032	62
80	6	If middle_middle_square=`b` and bottom_right_square=`x` then class=`positive`	0.83582	67
81	6	If middle_middle_square=`x` and bottom_left_square=`x` and middle_left_square=`o` then class=`positive`	0.8871	62
82	6	If top_right_square=`b` and top_left_square=`o` then class=`positive`	0.62121	66
83	6	If middle_middle_square=`x` and middle_left_square=`x` then class=`positive`	0.784	125
84	6	If top_left_square=`b` and middle_middle_square=`x` and	0.81818	22

		middle_left_square=`b` then class=`positive`		
85	6	If bottom_right_square=`x` and middle_left_square=`o` and middle_right_square=`x` then class=`positive`	0.77419	62
86	6	If bottom_right_square=`x` and bottom_left_square=`b` and middle_right_square=`x` then class=`positive`	0.79487	39
87	6	If middle_right_square=`b` and bottom_right_square=`x` then class=`positive`	0.65476	84
88	6	If middle_middle_square=`b` and bottom_left_square=`x` and middle_right_square=`x` then class=`positive`	0.76923	13
89	6	If middle_left_square=`b` and middle_right_square=`x` then class=`positive`	0.5	70
90	6	If top_right_square=`o` then class=`positive`	0.5625	256
91	7	If bottom_middle_square=`b` and middle_right_square=`o` and top_middle_square=`x` then class=`negative`	0.42424	33
92	7	If bottom_left_square=`o` then class=`negative`	0.44403	268
93	7	If middle_middle_square=`x` and middle_right_square=`o` and bottom_right_square=`o` then class=`negative`	0.42	50
94	7	If middle_middle_square=`x` and bottom_right_square=`x` then class=`negative`	0.15	140
95	7	If middle_middle_square=`o` and bottom_right_square=`o` then class=`negative`	0.66667	81
96	7	If top_left_square=`o` and bottom_left_square=`x` and bottom_right_square=`x` then class=`positive`	0.83636	55
97	7	If bottom_right_square=`o` and middle_left_square=`x` then class=`positive`	0.5473	148
98	8	If bottom_left_square=`o` and top_right_square=`b` and top_left_square=`o` then class=`negative`	0.52381	21
99	8	If top_right_square=`o` and bottom_middle_square=`b` then class=`negative`	0.48611	72
100	8	If bottom_left_square=`o` and top_left_square=`o` and middle_left_square=`o` then class=`negative`	1	32
101	8	If top_left_square=`x` and middle_middle_square=`o` then class=`negative`	0.48529	136
102	8	If bottom_middle_square=`o` and top_right_square=`o` then class=`negative`	0.2963	54
103	8	If bottom_left_square=`b` then class=`positive`	0.66049	162
104	8	If top_right_square=`x` and middle_right_square=`x` then class=`positive`	0.69466	131

105	8	If middle_right_square=`b` and bottom_middle_square=`b` then class=`positive`	0.73913	46

Table 43 Tic-Tac-Toe Example Collective Model

While the collective models generated using the CRA process were not able to achieve higher accuracies than the traditional models in all cases. As the number of missing attributes increased the performance of the traditional method (ordered and unordered model) suffered the most. The collective models were less affected by the introduction of missing attributes. Take the Anneal data set for example. The highest accuracies achieved by the traditional and collective models (ordered) were 94% and 90% respectively under the best conditions (0 missing attributes). These models achieved 84% and 87% accuracy under the worst conditions (4 missing attributes). This equates to a 10% drop in accuracy for the traditional model compared to only a 3% drop in accuracy for the CRA model. Even though the models produced by the CRA process achieved lower accuracies in some instances these models were less affected by the missing attribute information. Table 44 provides the best performance of each data set under the best (no missing attributes) and worst conditions (4 missing attributes) and the drop in accuracy for each dataset for the ordered models. Table 45 displays this information for the unordered models. Figures 37 and 38 illustrate the drop in accuracy for both types of models.

Data Set	Traditional Best	Collective Best	Traditional Worst	Collective Worst	Traditional Drop	Collective Drop
Congressional Voting	99%	98%	79%	90%	20%	8%
Anneal	94%	90%	84%	87%	10%	3%
Thyroid	98%	98%	95%	97%	3%	1%
Mushroom	100%	90%	82%	87%	18%	3%
Tic-Tac-Toe	99%	96%	49%	71%	50%	28%

Table 44 Best Performance for Best and Worst Conditions Ordered Models

Data Set	Traditional Best	Collective Best	Traditional Worst	Collective Worst	Traditional Drop	Collective Drop
Congressional Voting	97%	97%	77%	89%	20%	8%
Anneal	90%	82%	82%	81%	8%	1%
Thyroid	97%	96%	94%	95%	3%	1%
Mushroom	89%	89%	77%	82%	12%	7%
Tic-Tac-Toe	82%	79%	48%	69%	34%	10%

Table 45 Best Performance for Best and Worst Conditions Unordered Models

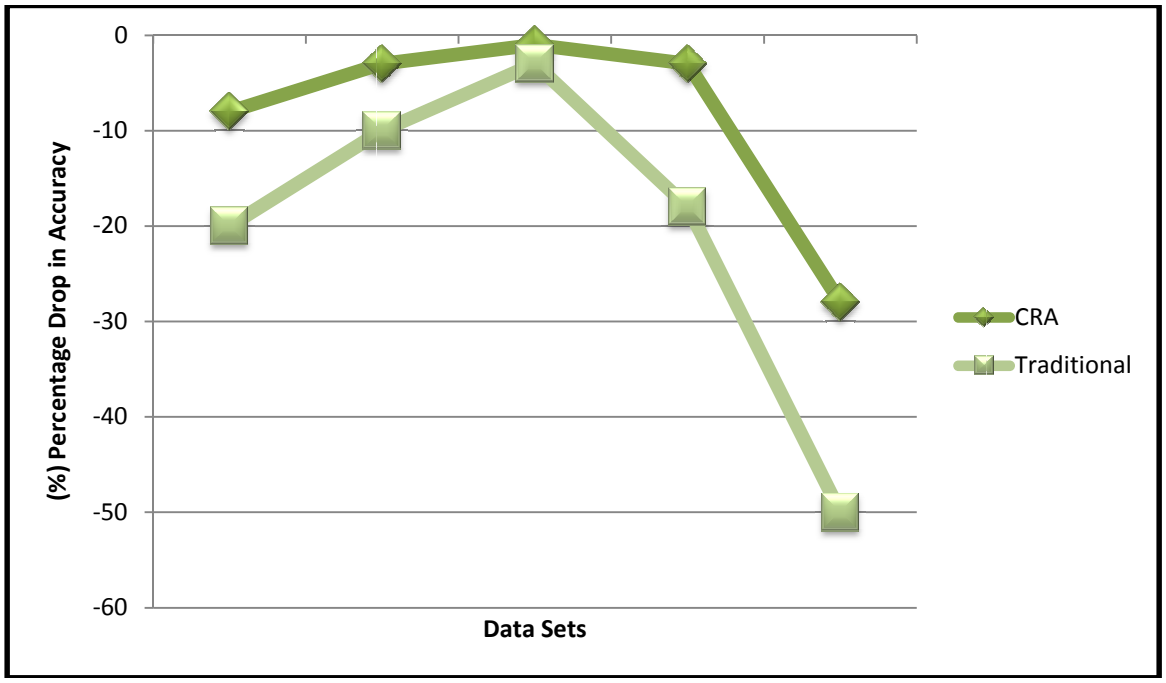


Figure 37 Percentage Drop in Accuracy Ordered Models

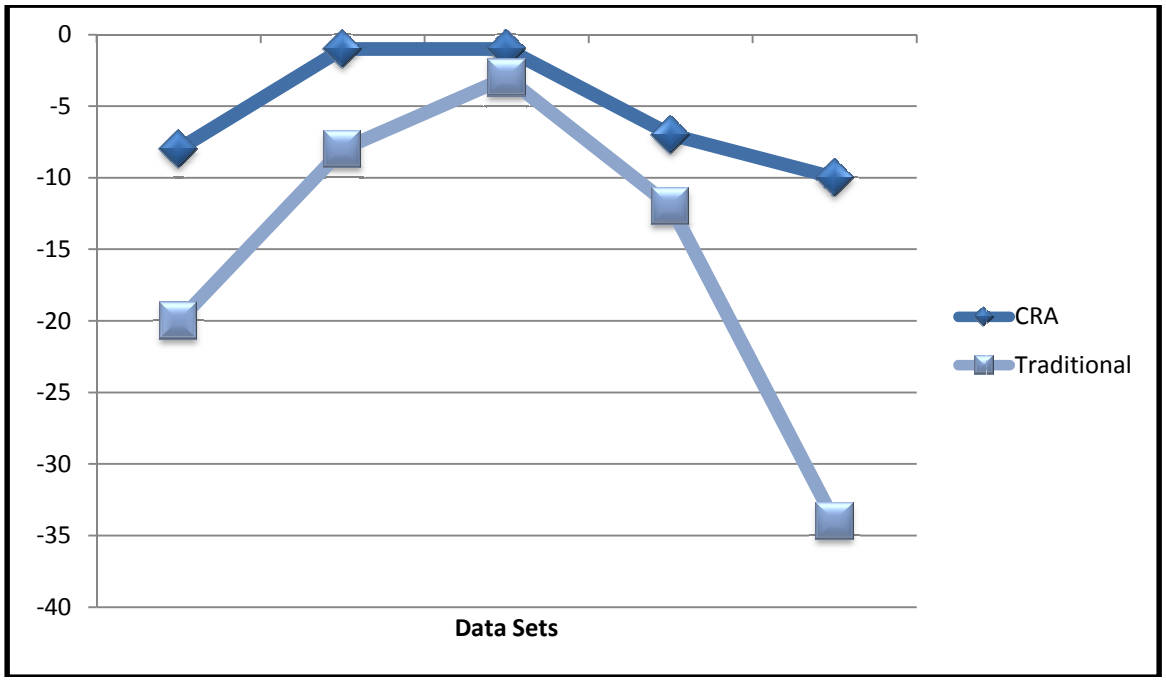


Figure 38 Percentage Drop in Accuracy Unordered Models

The aforementioned results of the drop in accuracy were evaluated using a Welch's Two Sample t-test to determine if they were statistically significant. Tables 46 and 47 (ordered and unordered models) display the results of this analysis for each data set. The null hypothesis was H_0: No difference in means for Traditional vs. CRA drop in classification accuracy for this test. For each data set evaluated, the p-value was less than .05. Therefore we can reject the null hypothesis. Meaning, that there is evidence that supports the notion that the collective models created using the CRA process are more resilient against the affects of missing attribute information.

Data Set	t	Degrees of Freedom	p-value
Congressional Voting	5.8755	17.889	0.00001
Anneal	2.6111	17.731	0.01783

Thyroid	2.5697	12.488	0.02392
Mushroom	5.9832	16.013	0.00001
Tic-Tac-Toe	12.6789	11.794	0.00000003

Table 46 Results of Two Sample Welch t-test (Ordered Models)

Data Set	t	Degrees of Freedom	p-value
Congressional Voting	5.1854	17.91	0.00006
Anneal	3.6934	15.388	0.002087
Thyroid	3.678	10.998	0.003640
Mushroom	2.1514	17.916	0.04534
Tic-Tac-Toe	9.268	13.954	0.0000002

Table 47 Results Two Sample Welch t-test (Unordered Models)

The hypothesis of this research was that the model created with the CRA process would be more resilient against missing attribute information in the test data. Resiliency refers to minimizing any decreases in classification accuracy as the number of missing attributes increases. For each data set the collective models created with the CRA process suffered less of a drop in classification accuracy as compared to the traditional models (see Figures 37 and 38). These findings are statistically significant (see tables 46 and 47) and support the hypothesis as the collective model maintained the ability to make fairly accurate predictions when faced with less than desirable conditions. This resiliency is a result of the collective model being generally larger than the traditional model and containing shorter more general rules as described in the analysis for each data set. This directly relates to the manner in which the collective models

were created. The CRA process of building individual models on smaller homogenous subsets of the data produced collective models for all data sets that contained rules that were on average shorter than the rules in the traditional model. Also, the collective models created for 4 out the 5 data sets contained more rules on average than the traditional model.

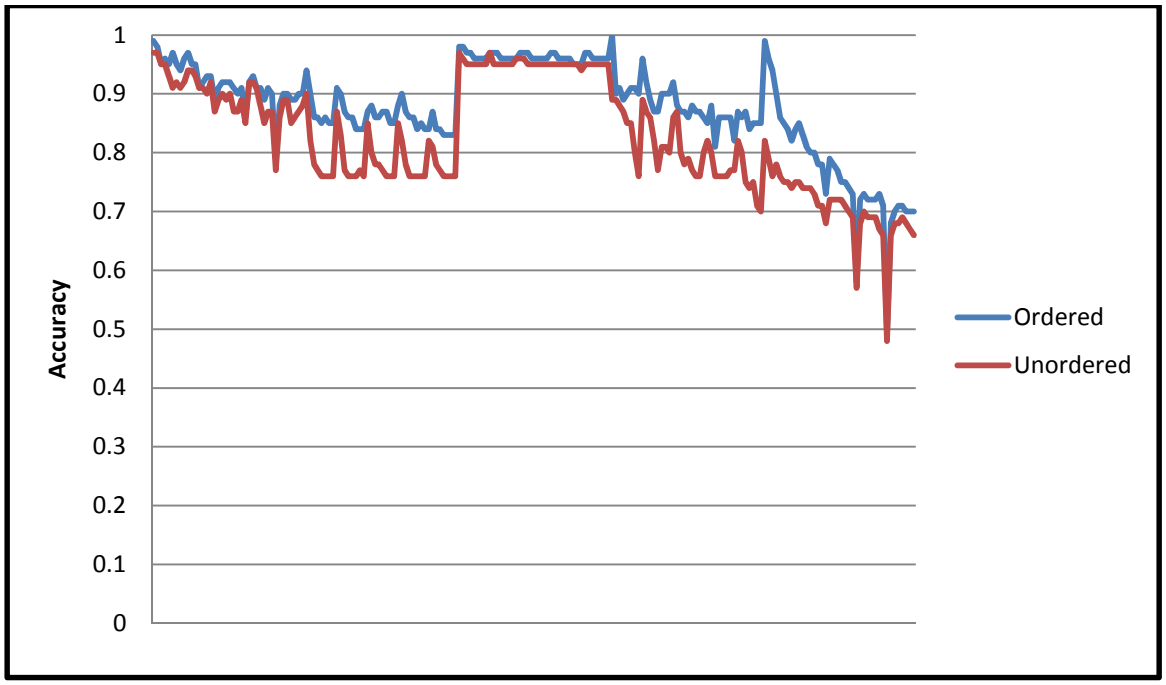


Figure 39 Ordered vs. Unordered Accuracies Overall

Figure 39 displays the overall performance of the unordered and ordered models. As seen in the figure, the ordered models performed better than the unordered models from an overall aspect. However, the difference in performance was negligible in some instances

Chapter 6

Summary

6.1 Summary and Conclusion

Data mining is a powerful and useful tool that can be used to extract information from large data repositories. These techniques are used in a variety of domains to perform an array of different tasks. Predictive modeling/classification is one of the most thoroughly studied and researched subfield in data mining. The goal of these types of tasks is to predict some future outcome based on historical information. Rule based classifiers are one of the most often used types of predictive model. A rule based classifier builds a model from training data as a set of high quality rules. The models that theses algorithms generate are easy to understand and interpret and widely accepted. There have been many rule based classification systems designed, studied and implemented that perform adequately in many applications. However, problems arise when the data that is used to assess the predictive ability of the model is not as comprehensive as the data used to create the model. When faced with missing attribute information it is desirable to have a model that maintains the ability to make fairly accurate predictions. Missing attributes are a well known issue in the field of data mining. (Batista & Monard, 2003) (Witten & Frank, 2005) (Tan, Steinbach, & Kumar, 2006) The majority of the research performed focuses on handling missing attribute information in the training data. There has been little in the way of studying the effects of missing information in the test data. We present a process which results in a model that is more resilient against the effects of missing attribute information in the

new/unseen data. The clustering rule-based approach (CRA) seeks to added robustness using data clustering. The training data is clustered and multiple models are built using the cluster wise data. The models are then combined into a collective model and evaluated against test data modified to contain missing attribute information. The experimental results support the hypothesis that the collective model is more resilient against the effects of missing attribute information. The collective model incurred a smaller loss in classification accuracy when faced with missing attributes. Also, the collective model displayed the ability to classify instances unable to be classified using traditional methods. These findings are encouraging as most real world data sets are often not as complete as practitioners would like. The work presented and evaluated here provides a new method of ensuring the model created will have the ability to make fairly accurate predictions when faced with missing/incomplete data.

6.2 Future Work

There are many paths the direction of this research can take. We evaluated two methods of combining the rules from the multiple models: sort in decreasing order of accuracy and a voting scheme. Other approaches for combining the rules should be examined. Also, a wider variety of data sets should be used to evaluate the CRA process. The effects of a different set of *k* values should also be examined. Different clustering algorithms may also be studied, as well as, combining this technique with other process proven to improve classification accuracy such as ensemble methods.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *Proceedings of teh Twentieth International Conference on Very Large Databases*, (pp. 487-499). Santiago, Chile.

Anderberg, M. R. (1973). *Cluters Analysis for Applications.* Academic Press.
Batista, G. E., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence 17(5-6)* , 519-533.

Baumann, T., & Germond, A. (1993). Application of the kohonen network to short-term forecasting. *In ANNPS*, (pp. 407-412).

Berk, R. (2004). An Introduction to Ensemble Methods for Data Analysis. *California Center for Population Research. Online Working Paper Series.*

Berkhin, P. (2002). *Survey of Clustering Data Mining Techniques.* San Jose, CA: Acrue Software.

Berson, A., Smith, S., & Thearling, K. (1999). *Building Data Mining Applications for CRM.* McGraw-Hill Companies.

Blake, E. K., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Retrieved from http://www.ics.uci.edu/~mlearn/MLRepository.html

Breiman, L., Freidman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group.

CHAID. (n.d.). Retrieved February 15, 2008, from http://www.wikipedia.org/wiki/CHAID
Cheng, Y., & Church, G. (2000). Biclustering of experession data. *ICMB* , 93-103.

Chess Endgame. (2008). Retrieved April 3, 2008, from http://www.wikipedia.org/wiki/Endgame

Cho, H., Dhillon, I. S., Guan, Y., & Sra, S. (2004). Minimum sum squared residue co-clustering of gene expression data. *SDM* .

Clark, P., & Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning 3(4)* , 261 - 283.

Cluster Analysis. (2008). Retrieved March 6, 2008, from Wikipedia: http://en.wikipedia.org/wiki/Data_clusterinCohen, W. (1995). Fast Effective Rule Induction. *Procedings of 12th International Conference on Machine Learning*, (pp. 115-123). Tahoe City, CA.

CRISP-DM. (2000). Retrieved February 16, 2008, from CRISP-DM: Cross Industry Standard Process for Data Mining: http://www.crisp-dm.org/Process/index.htm

Deodhar, M., & Ghosh, J. (2007). A Framework for Simultaneous Co-clustering and Learning from Complex Data. *KDD'07*, (pp. 250-259). San Jose, California.

Djukanovic, B., Babic, B., Sobajic, D., & Pao, Y. (1993). Unsupervised/supervised learning concept for 24-house load forecasting. *IEE Proceedings-Generation, Transmission and Distribution*, (pp. 140:311-318).

DMS. (2008). Retrieved March 16, 2008, from Decision Trees: http://dms.irb.hr/tutorial/tut_dtrees.php

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continous features. *ICML.*

Freund, Y., & Schapire, R. (1999). A short introduction to boosting. *Japanese Society for Artificial Intelligence* , 771-780.

Friedman, N., & Goldszmidt, M. (1996). Building classifiers using Bayesian networks. *National Conference on Artificial Intelligence* (pp. 1277-1284). Menlo Park, CA: AAAI Press.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian Network Classifiers. *Machine Learning 29* , 131-163.

Gilbert, J. E. (2006). Applicaitons Quest: Computing Diversity. *Commmunications of the ACM, 49, 3* , pp. 99-104.

Gilbert, J. E. (2008). Applications Quest: A Case Study on Holistic Admissions. *Journal of College Admission* , 12 - 18.

Gilbert, J. E. (2006). Applications Quest: Computing Diversity. *Communications of the ACM, 49, 3, ACM* , 99-104.

Gilbert, J. E. (2008). *Patent No. US 2008/10183731 A1.* USA.

Han, J., & Kamber, M. (2006). *Data Mining Techniques and Concepts.* Morgan Kaufmann.

Hu, H., & Li, J. (2005). Using Association Rules to Make Rule-based Classifiers Robust.

Proceedings of the Sixteenth Austrailian Database Conference (ADC 2005), (pp. 47-54). Newcastle, Australia.

Jain, A. K., Murty, M. M., & Flynn, P. J. (n.d.). Data Clustering: A Review. *ACM Computing Surveys 31, 3* , 264-323.

Kass, G. V. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Applied Statitics,Vol 29, No 2* , 119-127.

Langley, P., & Sage, S. (1994). Induction of Selective Bayesian Classifiers. *Conference on Uncertainty in Artificial Intelligence* (pp. 399-406). Seattle, WA: Morgan Kaufmann.

Larose, D. (2005). *Discovering Knowledge in Data: An Introduction to Data Mining.* Hoboken, NJ: John Wiley & Sons.

Lewis, R. (2000). Indroduction to Classification and Regression Trees (CART). *Annual Meeting of the Society for Academic Emergency Medicine.* San Francisco, CA.

Li, J., Topor, R., & Shen, H. (2002). Construct robust rule sets for classification. *Proceedings of the Eigth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 564-569). Edmonton, Alberta, Canada.

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. *Proceedings 2001 IEEE International Conference on Data Mining (ICDM 2001)* (pp. 369-376). IEEE Computer Society Press.

Liu, B., Hsu, J., & Ma, Y. (1998). Integrating classification and assoiciation rule mining. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, (pp. 27-31).

Lokmic, L., & Smith, K. (2000). Cash flow forecasting using supervised and unsupervised neural netoworks. *IJCNN* , 06:6343.

Michalski, R., Mozetic, I., Hong, J., & Lavrac, N. (1986). The AQ15 inductive learning system: an overview and experiments. *Proceedings of IMAL 1986'.* Universite de Paris-Sud, Orsay.

Microsoft. (2007). *Microsoft SQL Server 2005*. Retrieved March 15, 2008, from Microsoft Corporation: http://www.microsoft.com/sql/prodinfo/overview/default.mspx

Mingers, J. (1989). An emprical comparison of selection measures for decision tree induction. *Machine Learning vol 3* , 319-342.

Oh, K., & Han, I. (2001). An intelligent clustering forecasting system based on change-point detection and artificial neural networks: Application to financial economics. *HICSS-34 volume 3* , 3011.

Opitz, D., & Macline, R. (1999). Popular ensemble methods: An empirical study. *Artificial Intelligence Research* , 169-198.
ORACLE. (2007). *ORACLE*. Retrieved March 15, 2008, from Oracle Data Mining: http://www.oracle.com/technology/products/bi/odm/index.html

Quinlan, J. (2006). *Bagging,Boosting, and C4.5*. Retrieved January 29, 2008, from RuleQuest: http://www.rulequest.com/Personal/q.aaai96.ps

Quinlan, J. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.
Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning vol. 1* , 81-106.

Quinlan, J., & Gaetz, M. W. (1993). FOIL: A Midterm Report. *Proceedings of the 1993 European Conference on Machine Learning*, (pp. 3-20). Vienna, Austria.

Rapid-i. (2008). *Rapid Miner*. Retrieved March 16, 2008, from http://rapid-i.com

Tan, P.-N., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining.* Boston, MA: Addison Wesley.

TechTarget. (2008). Retrieved January 29, 2008, from What is data mining?: http://searchsqlserver.techtarget.com/sDefinition/0,,sid87_gci211901,00.html

Wang, J., & Karypis, G. (2003). HARMONY: Efficiently Mining the Best Rules for Classifcation. *Proceedings of the 2003 SIAM International Conference on Data Mining*, (pp. 205-216). Newport Beach, CA.

Witten, I. H., & Frank, E. (2005). *Data Mining. Practical Machine Learning Tools and Techniques.* San Francisco, CA: Morgan Kaufmann.

Yin, X., & Han, J. (2003). CPAR: Classification based on Predictive Associaiton Rules. *Proceedings of the 2003 SIAM International Conference on Data Mining*, (pp. 331-335). San Francisco, CA.