May 2015

Data Preprocessing for Advanced Analytics

Andrew Robertson McKay
Worcester Polytechnic Institute

Patrick Dennis Murphy
Worcester Polytechnic Institute

Data Preprocessing for Advanced Analytics

A Major Qualifying Project Submitted to the Faculty of

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the

Degree in Bachelor of Science

In Computer Science

By **Andrew McKay**

And **Patrick Murphy**

Sponsoring Organization:

QueBIT

Project Advisor:

Professor Carolina Ruiz

Abstract

The goal of this project is to improve the attribute selection aspect of data preprocessing. This is done by working on two techniques, attribute combination and clustering. Attribute combination generates new attributes by combining pairs of numeric attributes using arithmetic operations. Attribute clustering discovers groups of categorical attributes based on similarity as determined by the Minimum Description Length (MDL). The attribute combinations generated by this project frequently have significantly increased correlation to the target attribute than those of the original attributes. The clusters of categorical attributes allow analysts to select a subset of the attributes in the original dataset that produce approximately the same classification accuracy as the full set of attributes when used in Logistic Regression, Naïve Bayes, and Decision Trees while significantly reducing the dimensionality of the data. Additionally the clustering can be used to identify patterns in frequently occurring attribute value combinations for attributes in the same cluster. Both the attribute combination and the attribute clustering techniques investigated provide data analysts with additional insight into a dataset and exhibit a reasonable time performance.

Executive Summary

Problem Statement

The problem for this Major Qualifying Project was provided by QueBIT, our project sponsor. QueBIT is a business analytics company which helps other organizations improve their ability to make valuable decisions based on data. One of the largest expenditures of time QueBIT analysts face in data mining is data preprocessing. In particular, attribute selection is a time intensive and indispensable part of this preprocessing. The high dimensionality of the real-world datasets that QueBIT works with makes it often infeasible and undesirable to perform predictive modeling over the full data. Therefore the attributes of the dataset must be carefully analyzed in order to select a subset and/or extract new attributes that preserve the information contained in the original dataset while significantly reducing its dimensionality.

Goal

The goal of this project is to develop techniques that will help with attribute selection and attribute extraction by discovering relationships among original dataset attributes.

Objectives

The objectives of this project are to devise the aforementioned attribute selection and extraction techniques, implement them in the *R* programming language, and test them on a variety of datasets to evaluate their strengths and weaknesses. Furthermore, to create an *R* package with these implementations that QueBIT can use in IBM's SPSS modeler.

Methodology

Two techniques were developed to reach the goal. The first technique was attribute combination. Each combination created contains a pair of the original numeric attributes in the dataset, A and B. These attributes are combined using six different arithmetic operations: A+B, A-B, A*B, A/B, A*B/(A+B), and (A+B)/(A*B). These six operations are calculated for every pair of numeric attributes that are not the

target attribute. The combinations are ordered by their correlation to the target attribute. Because of the approach followed to generate attribute combinations, this attribute extraction technique is applicable in regression tasks, where the prediction target is a numeric attribute.

The second technique is attribute clustering. These clusters are created over categorical attributes using hierarchical clustering with Minimum Description Length (MDL) as the distance measure between attributes. In addition to calculating the clustering with lowest MDL, our code provides supplementary useful information about these clusters such as frequently occurring attribute value combinations. Because of the approach followed to cluster attributes, the evaluation of this technique was restricted to classification tasks, where the prediction target is a categorical attribute.

Results

The attribute combinations generated by this project frequently have significantly increased correlation to the target attribute compared to those of the attributes used to create these combinations. In some cases the correlation of the combination is more than 30% higher than the correlation of the original two attributes. Although many of the combinations generated do not improve on their original components, these are filtered out and only the combinations with the highest gain in correlation are kept. Our attribute combination technique was tested on over eleven datasets of varying sizes to evaluate its efficiency and time performance. All datasets tested produced a non-trivial amount of combinations resulting in improved accuracy. The arithmetic formulas that produced the best combinations varied among datasets, and every formula was seen to produce significantly improved correlation on several occasions.

The results for clustering methods were that the clusters generated tended to significantly reduce the dimensionality of a dataset while still achieving a classification accuracy within 1% of that of the original dataset when using a variety of classification techniques such as Linear Regression, Naïve Bayes, and Decision Trees. We tested our clustering over nine categorical and mixed-type datasets to evaluate

efficiency and time performance. We were also able to identify types of datasets that are and are not well suited to the application of this technique.

Conclusion

The techniques presented in this project provide data analysts with additional functionality during data preprocessing. These techniques are capable of providing valuable insight about the dataset that would otherwise be difficult to discover.

Table of Contents

List of Figures

List of Tables

1 Introduction

The application of Knowledge Discovery in Databases (KDD) -techniques has become an increasingly ubiquitous practice in modern industry, economics, and academia. Where historically data analysis was investigated by hand on a relatively small dataset, today database sizes are measured in Gigabytes, Terabytes, and even Petabytes, rendering such historical techniques insufficient. The use of KDD has allowed analysts who would otherwise be overwhelmed by the volume of information to inspect and refine the data available to them to see the patterns that are hiding in the veritable mountains of information.

Although KDD techniques have been the subject of considerable study, automation, and refinement, these techniques still frequently require the expertise of a human analyst to achieve meaningful results. Our sponsor for this project is QueBIT, which is a business analytics strategy company that helps organizations improve their ability to make decisions that create value. Analysts at QueBIT have significant experience applying KDD techniques in business environments to discover answers and solutions in data to various problems on the behalf of many contracting institutions. They frequently are called upon to use these techniques to improve modeling of future results, as well as to discover patterns that explain current trends. One of the primary tools used by QueBIT's analysts is IBM's SPSS Modeler, a data mining and text analysis tool. The tool is able to use software packages from several different programming languages to automate different KDD techniques, including the open source programming language *R*, which is designed for use in statistical computing.

In applying the principles of KDD, one of the most challenging tasks analysts are currently faced with is refining the input data through *data preprocessing*. Many of the techniques that can be applied to a dataset rely on the quality of data on which they are being performed. Real data is frequently incomplete, noisy, and inconsistent. The presence of these problems can significantly reduce the accuracy of models built on the data, and can obfuscate or even alter the patterns that exist. Additionally, datasets are frequently too large to be suitable even for KDD techniques designed for large-scale data. One of the goals of data preprocessing in KDD is not only to refine the data available but also to accurately

summarize it to increase the range of techniques that may be applied to it. In practice, meeting these goals is an often time-consuming effort performed by data analysts.

QueBIT's analysts have sponsored our work at Worcester Polytechnic Institute (WPI) to provide them with an *R* package that will provide analysts with tools to assist in the Data Preprocessing stage of KDD. In collaboration with our project liaison from QueBIT, Keith McCormick, we identified specific areas of data preprocessing that our work is intended to improve. Reducing the size of an input dataset via attribute selection is one of the most time intensive steps of an analyst's role in the KDD process, and there are doubtless many possible methods that could improve the efficiency of this step through automation of existing, and time intensive, techniques and summarization of the data. The problem faced in our work is how to provide the analyst with information to reduce the time required for selecting attributes without compromising the accuracy of the final model or reducing confidence in any patterns discovered.

The goal of our project is to devise techniques, and implement them in an *R* package, that can assist analysts in attribute selection by providing them functions that can discover information about attribute relationships. The creation of an *R* package allows QueBIT analysts to use these techniques in SPSS Modeler as well as their normal tools. We intended for the package we produce to improve the efficiency of time usage for attribute selection as well as providing information that would otherwise be difficult to discover manually, instead of replacing any work done by the analysts. We accomplished this by focusing on two facets of attribute selection and extraction: possible mathematical combinations of numeric attributes, and a summary of the attributes that fulfill a similar role in the dataset in the form of a set of clusters.

In order to discover information about possible combinations of attributes we wanted to test different methods of evaluating mathematical combinations of numeric attributes. Initially we investigated the possibility of using Genetic Algorithms to explore the space of possible mathematical relationships; however these did not produce useful combinations. Instead we decided to exhaustively search the possible combinations of attributes based on predetermined algebraic formulas. The motivation

for this idea was to automate the discovery of relationships such as that between body weight and height as Body Mass Index. Once we have evaluated the possible combinations we report those that have produced the most relevant results to the analyst, as well as store the full results of all possible combinations to allow further analysis.

Additionally we have incorporated an algorithm for clustering attributes in categorical datasets along with Factor Analysis into a function that will group the attributes in a dataset based on their similarity in distribution and predictive value. Categorical attributes are initially clustered based on the algorithm described in Section 3.2 Attribute Clustering. These clusters are used as the baseline for further refinement based on reverting the hierarchical clustering performed incrementally. These different possible clusterings are compared to both a basic predictive model constructed over the same dataset and to each other to determine the best clustering produced. All information produced in this process is reported back to the analyst, and more information can be obtained from the clusterings by the analyst with the additional analysis functions.

2 Background

Real world data is often incomplete, inconsistent, or lacking in details. The process to make real world data more useful to work with is usually called data preprocessing. There are many steps and pieces to data preprocessing. These steps include data cleaning, data integration, data transformation, data reduction, and data discretization. Data cleaning is when data is cleansed through by filling in missing values, smoothing noisy data, or resolving irregularities in the data. Data integration is when data with different representations are combined, and conflicts within the data are fixed. Data transformation is when data is aggregated, normalized, and generalized. Data reduction is making a smaller reduced representation of an entire dataset. Data discretization involves reducing the number of values for an attribute into intervals. These techniques are not always used for every dataset, but combinations of these techniques used by an expert can lead to a much more manageable and useful dataset to use. The focus of this MQP is to help with data transformation and reduction so that the experts at Quebit can be more efficient by not having to spend as much time on these steps of data preprocessing.

2.1 Correlation

One of our methods to help with data transformation and reduction bases its analysis on correlation. The correlation coefficient is a measure of the degree to which changes to the value of a variable predict change to the value of another variable (Rouse, 2013). This coefficient ranges from minus one to positive one representing a negative correlation to a positive correlation.

The correlation function in R allows for three slightly different calculations of correlation. The three methods are the Pearson, Kendall, and Spearman calculations. The Pearson coefficient is the default method as it is the most commonly used. The calculation of the Pearson coefficient is the covariance of the x and y variables divided by the product of the standard deviation of x and standard deviation of y (Lane, 2013) (Lund & Lund, 2015). The Kendall and Spearman correlations would work better for nominal or ordinal attributes, so Pearson was the method that was chosen to be used over numeric attributes.

2.2 Cluster Analysis

One of the techniques for discerning relationships and patterns in data, and an area in which much work has already been done, is Cluster Analysis. In KDD Cluster Analysis is best described as the task of grouping objects together such that objects in a group are more similar, or at least less dissimilar, to themselves as they are to objects in different groups. The technique itself does not represent a single algorithm but rather describes a wide group of algorithms that evaluate objects on different concepts of similarity using different methods. The different approaches to Cluster Analysis can be generally grouped into three categories: Joining, Two-way Joining, and K-means Clustering (Clustering Approaches). Joining attempts to group objects into increasingly large clusters using some measure of similarity or distance. Two-way Joining is similar to Joining except that attributes and instances are both clustered simultaneously with the intent of possibly discovering patterns across both attributes and instances. Finally K-means Clustering is based on existing knowledge of a likely amount of clusters, and leverages that knowledge to separate the data into the predetermined amount of clusters. Our work specifically relies upon a type of Joining clustering called Hierarchical Clustering wherein every point is originally in a cluster by itself. These clusters are then combined one by one based on the smallest difference between the clusters until only one cluster remains.

2.2.1 Iterative Clustering

In addition to the aforementioned methods of evaluating attribute relationships, Michael Mampacy and Jiles Vreeken of the Department of Computer Science at the University of Antwerp in Belgium (Mampacy & Vreeken, 2013) have developed a method for clustering categorical attributes using the mathematical concept of Minimum Description Length (MDL). The approach presented in their paper uses Hierarchical Clustering in concert with MDL to group attributes based on mutual information contained in these attributes in the dataset. The authors of that paper suggest that “data mining is

essentially an iterative process" (Mampacy & Vreeken, 2013) and that the algorithm they have developed should serve as a first-iteration summary of the relationships between attributes that loosely identifies "which attributes are most correlated, and in what value configurations they occur" (Mampacy & Vreeken, 2013). To the best of both their knowledge and ours there are not any comparable methods for summarizing attributes in a dataset, or for discerning high-level information about their relationships.

2.2.2 Minimum Description Length

The attribute clustering method described in (Mampacy & Vreeken, 2013) is based on the mathematical concept of MDL. This concept may be described by the principle that given a set of models *S*, the best model *M* in *S* is the one that minimizes $L(M) + L(D \mid M)$ where $L(M)$ is the length, in bits, of the description of *M* and $L(D \mid M)$ is the length, in bits, of the description of the data encoded with *M*. The algorithm described by Mampacy and Vreeken uses the Bell number to encode information to a produce model-to-data code (Vereshchagin N, 2004). The algorithm determines at each level of the hierarchical clustering the description length of each of the next possible attribute clusters and selects the next cluster based on this information. It continues until all the attributes have been grouped into a single cluster and then selects the clustering with the smallest description length of those selected in any of its iterations.

2.2.3 Attribute Clustering Algorithm

The algorithm used to determine the clustering is provided here:

Algorithm 1: ATTRIBUTECLUSTERING(D)

input : Categorical dataset D over a set of attributes $\mathcal{A}$.
output : Attribute clustering $\mathcal{C} = \{A_1, \ldots, A_k\}$.

$\mathcal{C} \leftarrow \{\{a\} \mid a \in \mathcal{A}\}$
$\mathcal{C}_{\min} \leftarrow \mathcal{C}$
Compute and store $CS_D(A_i, A_j)$ for all $i \neq j$
while $|\mathcal{C}| > 1$ **do**
    $A_i, A_j \leftarrow \arg\max_{i,j} CS_D(A_i, A_j)$
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{A_i, A_j\} \bigcup \{A_i \cup A_j\}$
    Compute and store $CS_D(A_{ij}, A_l)$ for all $l \neq ij$
    if $L(\mathcal{C}, D) < L(\mathcal{C}_{\min}, D)$ **then**
        $\mathcal{C}_{\min} \leftarrow \mathcal{C}$
return $\mathcal{C}_{\min}$

Figure 1: Attribute Clustering Algorithm
Taken from (Mampacy & Vreeken, 2013) – This is the algorithm used for attribute clustering.

For the purposes of the algorithm above, an attribute clustering C = {A1, . . . , Ak} of a set of attributes A is a partition of A such that

1. Every attribute belongs to a cluster: $\bigcup_{i=1}^{k} Ai$ = A,
2. All clusters are pairwise disjoint: ∀ i != j : Ai ∩ Aj = ∅ ,
3. There are no empty clusters: ∀Ai ∈ C : Ai != ∅ .

The algorithm begins by treating each attribute as its own cluster and initializes the minimum length clustering to be the same. The Cluster Similarity (CS) for the dataset D is then calculated for all possible pairs of clusters and stored as a table for reference. CS is calculated using the formula:

$$CS_D(\text{Ai , Aj }) = |\text{D}| * \text{I(Ai , Aj)} + \Delta\text{L(CT)}$$

where *I* is the mutual information between clusters Ai and Aj is the sum of their individual entropies less the sum of the entropy from a cluster that would result from combining the original two. It is given by the formula

$$I(A_i, A_j) = H(A_i) + H(A_j) - H(A_{ij}).$$

Additionally the term ΔL(CT) represents the difference in Code Table length that would result from the combination of the two given clusters. The Code Table is the bitwise encoding of all combinations of attribute values present in the dataset given as a parameter based on their frequency of

appearance. The formula to calculate the length of a given code table for a specific clustering of attributes Ai is given by:

$$L(CT_i) = \sum_{\substack{v \in dom(A_i) \\ fr(A_i=v)\neq 0}} \log(|dom(A_i)|) + \log\log|D| - \log fr(A_i = v)$$

Once the Code Similarities have been calculated for every possible pair of attribute clusters Ai, Aj they are stored at index (i, j) in a matrix and are used as the 'fitness' measure by which the attributes are clustered hierarchically. From the baseline of each attribute being in a single attribute cluster until all attributes have been clustered together, the algorithm determines the pair of clusters with the highest Code Similarity and combines them into a single cluster. The Code Similarity is updated to reflect the merging of the two clusters into one and is stored once again in a Code Similarity table. The algorithm then evaluates if description length of the clustering, L(C, D), is less than the description length of the previous shortest length clustering L(Cmin, D), the equations for which are given below. These equations leverage the use of the Bell number, B*n*, which represents the number of possible partitions of a set of *n* attributes.

Definition 2 The description length of a categorical dataset D using an attribute clustering $\mathcal{C} = \{A_1, \ldots, A_k\}$ of size k is defined as

$$L(\mathcal{C}, D) = L(\mathcal{C}) + L(D \mid \mathcal{C}) ,$$

where

$$\begin{cases} L(D \mid \mathcal{C}) = |D| \sum_{i=1}^{k} H(A_i) \\ L(\mathcal{C}) \quad = \log B_n + \sum_{i=1}^{k} L(CT_i) \\ L(CT_i) \quad = \sum\limits_{\substack{v \in dom(A_i) \\ fr(A_i=v)\neq 0}} \log|dom(A_i)| + \log\log|D| - \log fr(A_i = v) \end{cases}$$

Figure 2: Attribute Clustering Definitions
Taken from (Mampacy & Vreeken, 2013) – These are several of the equations used in the algorithm described in Figure 1.

If the current clustering results in a shorter description length it is recorded as the best clustering, Cmin, and the algorithm continues. Once all clusters have been combined into one the algorithm reports the best clustering discovered and finishes. The clusters are returned by our implementation of the

algorithm as a list of lists of numbers where each number represents the attribute that occurs in that position in the dataset.

2.3 The R Programming Language

One of the primary requirements for our project was the creation of an R language package for use in SPSS Modeler by QueBIT. R is a programming language for statistical computing and visualization (R Project). It is a GNU Project based on the S language developed at Bell Labs by John Chambers and collaborators. The project is maintained and developed by volunteer programmers and provides a suite of software facilities for data evaluation, calculation, and information display. It allows developers to independently create software packages for distribution and use.

3 Methodology

We have designed two main methods for data preprocessing. One is called attribute combinations and the other is called attribute clustering. The attribute combinations method uses only numeric attributes while attribute clustering focuses on nominal attribute, with an additional factor analysis for numeric attributes.

In this section, we describe these methods and provide an input/ouput description of the implementation of each of the methods in R. Examples of the output generated by these methods are provided in Section 4 Results.

3.1 Attribute Combinations

3.1.1 Combination Creation

3.1.1.1 Overview

The objective of the combinations method is to combine numeric attributes using arithmetic operations aiming to find better prediction attributes for a specific target attribute. The combinations we create contain two attributes which we will refer to as A and B. The arithmetic operations that are used include A+B, A-B, A*B, A/B, A*B/(A+B), and (A+B)/(A*B). These six operations are calculated for every pair of numeric attributes that are not the target attribute. The target attribute is the attribute that is trying to be predicted using other attributes.

3.1.1.2 Ranking by correlation

Using correlation we determine which combinations are most effective for predicting the target attribute. The combination correlations are compared with correlation from the individual attributes that make up the combination in order to determine if the combination is better than the individual attributes. These resulting combination correlations are ordered by the highest correlation.

3.1.1.3 Ranking by correlation gain

An alternative way of ranking the combinations is to rank them by their gain in correlation. The gain in correlation is the amount of improvement in correlation based on the correlations of the individual attributes that make up the combination. The equation is the following:

$$corr\ gain = |corr(combination(A,B),Target)| - max(|corr(A,Target)|, |corr(B,Target)|)$$

For example, if attribute A had correlation of .1, B had a correlation of .2, and the combined A*B correlation was .8, the gain in correlation would be be .8 minus .2 which equals .6.

3.1.1.4 Function Overview

In this section we describe the R function implemented for attribute combination.

combinations(data, target, num_top, list_by);

Table 1: Inputs of the combinations function

Input Parameter	Parameter Description
data: data.frame	The dataset read into R as a *data.frame*
target: integer	The index number of the target attribute in the dataset
num_top: integer	The number of combinations the user wants displayed. The function will display the top results ranked by one of two values which is decided in *list_by*
list_by: integer	The order by which the combinations are ranked. 1 = Correlation, 2 = Correlation Gain

Table 2: Outputs of the combinations function

Output Item	Item Description
list_top: data.frame	This is the returned value from the function. This data.frame contains the top combination results as customized by the input parameters (number of results shown, ordered by correlation or correlation gain). This data.frame can be stored to a variable to be used by the *eval_combinations* function.
out_top_combined.txt	A text file which contains the top combination results (this text file is the same as what is printed out or stored to a variable)

	This file is delimited with "#" and can be opened with a spreadsheet editor like excel
out_all_combined.txt	A text file which contains all of the combination results. This file is delimited with "#" and can be opened with a spreadsheet editor like excel
out_data.txt	A text file which is a normalized version of the dataset where the numeric data is scaled from 0 to 1. This file is delimited with "#" and can be opened with a spreadsheet editor like excel.

3.1.2 Combination Evaluation

3.1.2.1 Overview

An additional analysis method was created for the combinations results. This method consists of two functions, called *eval_combinations* and *eval_combination*. These functions are nearly identical in nature but a couple of key differences. *eval_combinations* evaluates the top results that you get when running the *combinations* functions. The *eval_combination* function lets you pick a specific combination to use for evaluation. Both functions run linear regressions tests.

3.1.2.2 eval_combinations Function

The *eval_combinations* function takes the top results that were created with the *combinations* function and runs linear regressions tests using k-fold crossvalidation to compare if the dataset with these new combinations is a better predictor of the target attribute.

eval_combinations(data,target,top_n_result,num_fold)

Table 3: Inputs of the eval_combinations function

Input Parameter	Parameter Description
data: data.frame	The dataset read into R as a *data.frame*
target: integer	The index number of the target attribute in the dataset
top_n_result: data.frame	The *data.frame* that was returned by the *combinations* function which the user wishes to evaluate. This *data.frame* is expected to be the top n combinations based on what the user chose in the *combinations* function.

num_fold: integer	The number of k-fold cross validations to be run with linear regression

Table 4: Outputs of the eval_combinations function

Output Item	Item Description
result: data.frame	This is the returned value from the function. This data.frame will be printed out unless stored as a variable. This data.frame contains the MeanSquareError values for the linear regression models of the original dataset and the original datasetl with the top combinations added.
out_numeric_data_origin.txt	A text file which contains the original dataset input to the function. This file is delimited with "#" and can be opened with a spreadsheet editor like excel.
out_numeric_data_add.txt	A text file which contains the dataset input to the function plus the top combinations added to the dataset. This file is delimited with "#" and can be opened with a spreadsheet editor like excel.

3.1.2.3 eval_combination Function

The *eval_combination* function allows the user to constructone specific attribute combination, and to run linear regressions tests to compare if the dataset with this new combination added is, or the dataset with this new combination replacing its parent attributes.

eval_combination(data,target,indexA,indexB,func_option,num_fold)

Table 5: Inputs of the eval_combination function

Input Parameter	Parameter Description
data: data.frame	The dataset read into R as a *data.frame*
target: integer	The index number of the target attribute in the dataset
indexA: integer	The index number of attribute A for the combination the user wants to construct
indexB: integer	The index number of attribute B for the combination the user wants to construct
func_option: integer	The function operation number which ranges from 1 to 6 based on the six different possible arithmetic combinations: 1:=A+B, 2:=A-B,

	3:=A*B, 4:=A/B, 5:=A*B/(A+B), 6:=(A+B)/(A*B)
num_fold: integer	The number of k-fold cross validations ran with linear regression

Table 6: Outputs of the eval_combinations function

Output Item	Item Description
result: data.frame	This is the returned value from the function. This data.frame will be printed out unless stored as a variable. This data.frame contains the MeanSquareError values for the linear regression models of the original dataset, the original datasetl with the combination added, and the original dataset with combination added with the combinations parent attributes removed.
out_numeric_data_origin.txt	A text file which contains the original dataset input to the function. This file is delimited with "#" and can be opened with a spreadsheet editor like excel.
out_numeric_data_add.txt	A text file which contains the dataset input to the function plus the combination added to the dataset. This file is delimited with "#" and can be opened with a spreadsheet editor like excel.
out_numeric_data_replace.txt	A text file which contains the dataset input to the function minus the attributes that make up the combination plus the one combination added to the dataset. This file is delimited with "#" and can be opened with a spreadsheet editor like excel.

3.2 Attribute Clustering

3.2.1 Overview

In order to discover how attributes interact and allow analysts to identify attributes that fulfill redundant roles in class prediction we decided to investigate the use of clustering on attributes. Note that the typical use of clustering in data mining deals with the clustering of data instances. Here we deal with clustering of data attributes instead. By clustering attributes based on their mutual information we are able

to identify which attributes contain more unique information for prediction than others and allow analysts to see which attributes are most closely related.

3.2.2 Attribute Clustering Function

The foundation of the approach we have developed to attribute clustering is given by the work of Mampacy and Vreeken for clustering categorical attributes using Minimum Description Length (Mampacy & Vreeken, 2013). Their algorithm is described in detailed in Section 2.2.3 Attribute Clustering Algorithm. In addition to the clusters generated for categorical attributes we evaluate the predictive accuracy of the clusters with the numeric attributes included. This provides a basic measure of the relationships between attributes that we refine further by analyzing predictive accuracy of models constructed using attributes selected based on the clusters. Additionally the clusters are also compared to those that result from removing the poorest fitting attribute from its cluster several times iteratively. The information collected this way is reported to the analyst. The major steps of this process are designed as separate functions to be run in sequence as needed by analysts. This is packaged in *R* with the attribute combination analysis and will be used by QueBIT in SPSS Modeler.

3.2.2.1 Function Overview

AttributeClustering(dataset);

Table 7: Inputs of the AttributeClustering function

Input Parameter	Parameter Description
dataset: **data.frame**	The dataset read into R as a *data.frame*. Categorical attributes must be identified as such in *R* for datasets with mixed attribute types. Any mixed datasets stored in formats that represent categorical attributes with integers will require additional preprocessing to identify which attributes are categorical. This is done primarily by using a good data storage format. Specifically any format for storing the data that represents categorical variables as an integer will not work correctly when read into *R*. If such a format is used the columns in the data.frame containing

	categorical attributes will need to be manually identified as such. The algorithm expects the last attribute to be the target.

Table 8: Outputs of the AttributeClustering function

Output Item	Item Description
clusters: **list of list of integers**	This is the returned value from the function. This list itself contains lists that represent each cluster. The list representing a cluster contains an integer to represent each member attribute. These integers reflect that attribute's column number in the original dataset. Each attribute is a member of only one cluster. The order of the attributes in the list is a result of the hierarchical clustering used. This returned list should be stored in a variable to use with the other functions.

3.2.3 AttributeClusteringReport

In addition to the attribute clustering described in (Mampacy & Vreeken, 2013), we also implemented several other functions related to these clusters to help analysts identify patterns in the clusters. The first such function, AttributeClusteringReport(), takes as arguments an attribute clustering in the format output by AttributeClustering()and a dataset over which the clusters are to be evaluated. This function converts the raw clusters into a more human-readable form and also provides a list of the most frequently occurring attribute value combinations for the attributes in a cluster, sorted by their classification accuracy.

3.2.3.1 Function Overview

AttributeClusteringReport(clusters, dataset);

Table 9: Inputs of the AttributeClusteringReport function

Input Parameter	Parameter Description
dataset: **data.frame**	The dataset read into R as a *data.frame*. Categorical attributes must be identified as such in *R* for datasets with mixed attribute types. Any mixed datasets stored in formats that represent categorical attributes with integers will require additional preprocessing to identify which attributes are categorical. The

	algorithm expects the last attribute to be the target.
clusters: **list of lists of integers**	The clusters are the result of the *AttributeClustering()* function.

Table 10: Outputs of the AttributeClusteringReport function

Output Item	Item Description
cluster information: **terminal output**	This is the returned value from the function. Each attribute in a cluster is translated from raw form (the *clusters* input) to the attribute names given in the original dataset. and a cluster number and is printed to the terminal. Additionally the most frequently occurring attribute value combinations for each cluster are printed to the terminal along with their classification accuracy.

3.2.4 Attribute Cluster Analysis

We provide also an AttributeClusterAnalysis() function that takes an attribute clustering and the dataset it was generated from as arguments and performs additional analysis to refine the information represented by the clusters. This function ranks each attribute within a cluster based on that attribute's predictive accuracy with respect to the target attribute. This function was primarily used to evaluate the clustering algorithm.

3.2.4.1 Function Overview

AttributeClusterAnalysis(clusters, dataset);

Table 11: Inputs of the AttributeClusterAnalysis function

Input Parameter	Parameter Description
dataset: **data.frame**	The dataset read into R as a *data.frame*. Categorical attributes must be identified as such in *R* for datasets with mixed attribute types. Any mixed datasets stored in formats that represent categorical attributes with integers will require additional preprocessing to identify which attributes are categorical. The algorithm expects the last attribute to be the target.

clusters: **list of lists of integers**	The clusters are the result of the *AttributeClustering()* function and are the main focus of evaluation.

Table 12: Outputs of the AttributeClusterAnalysis function

Output Item	Item Description
clusters: **list of lists of pairs of integers and classification accuracy**	This is the returned value from the function. This list itself contains lists that represent each cluster. The list representing a cluster contains an integer to represent each member attribute paired with that attributes classification accuracy. These integers reflect that attribute's column number in the original dataset. Each attribute is a member of only one cluster. The order of the lists is sorted in decreasing order by classification accuracy.
classification accuracy information: **printed output on terminal**	The classification accuracy of the attribute clusters is printed to the terminal for three different classification techniques based on three different subsets of the original dataset's attributes. The classification techniques used are Logistic Regression, Naïve Bayes, and Decision Trees. The subsets of attributes used are: • the best attribute from each cluster, • the best attribute from each cluster and each numeric attribute, and • all attributes.

4 Results

This results section presents overall results of the experimentation performed to evaluate our attribute combinations and attribute clustering techniques. Also, it provides a few illustrative examples of results obtained from specific datasets. Additional detailed results can be viewed in Appendix J: Combination over previous combination results.

4.1 Attribute Combinations

4.1.1 Overview

The two combination functions were run on eleven datasets to evaluate the functions on their usefulness and time performance.

We experimented also with applying our attribute combination function over already combined attributes. That is, running our function over the result of the output of a previous run. Appendix J: Combination over previous combination results shows the results we obtained in one such experiment.

Dataset	Attributes	Instances	Time to Run	# of combinations
Adult Income http://archive.ics.uci.edu/ml/datasets/Adult	14 (5 numeric)	32561	2 seconds	60
Restaurant Consumer Data http://archive.ics.uci.edu/ml/datasets/Restaurant+%26+consumer+data	19(5 numeric)	138	1 second	60
Automobile http://archive.ics.uci.edu/ml/datasets/Automobile	26 (16 numeric)	205	2 seconds	720
CoverType http://archive.ics.uci.edu/ml/datasets/Covertype	54	581012	1.96 hours	8586
Music http://archive.ics.uci.edu/ml/datasets/YearPredictionMSD	90	515345	4.3 hours	24030
Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime	128 (126 numeric)	1994	4 - 6 minutes	47250
Urban Land Cover http://archive.ics.uci.edu/ml/datasets/Urban+Land+Cover	148 (147 numeric)	168	4 - 6 minutes	64386
Musk http://archive.ics.uci.edu/ml/datasets/Musk+%28Version+2%29	168	6598	9.95 minutes	84168
LSVT_Voice_Rehabilitation http://archive.ics.uci.edu/ml/datasets/LSVT+Voice+Rehabilitation	309	126	2.67 hours	285516
UJIIndoorLoc http://archive.ics.uci.edu/ml/datasets/UJIIndoorLoc	529	21048	9.33 hours	837936
ISOLET http://archive.ics.uci.edu/ml/datasets/ISOLET	617	7797	15.43 hours	1140216

Table 13: Combination Creation Statistics for all Datasets

This table contains the datasets that were used for experiments on the combinations functions as well as some statistics about their creation.

4.1.2 Combination Creation

4.1.2.1 Overview

The *combinations* function was ran on eleven datasets multiple times for different target attributes. For a couple of datasets there was more involved testing if there were more target attributes known to compare the top 5 or top 10 results for testing the *eval_combinations* function. To help understand the output of the *combinations* function, descriptions of the columns of the output are in Table 14.

Table 14: Combination Output Column Description

Column Title	Description
Correlation	The correlation of the combination with the target attribute
Function	The arithmetic operation used for the combination
Attribute A	The name of attribute A in the combination
Attribute B	The name of attribute B in the combination
Corr A Target	The correlation of attribute A with the target attribute
Corr B target	The correlation of attribute B with the target attribute
Gain	The gain in correlation from the highest individual correlation (corr A target & corr B taget) to the combination correlation
Index A	The index number of attribute A within the dataset
Index B	The index number of attribute B within the dataset
Function Option	The arithmetic operation used for the combination represented as a number. (Used primarily for *eval_combination*) 1:=A+B, 2:=A-B, 3:=A*B, 4:=A/B, 5:=A*B/(A+B), 6:=(A+B)/(A*B)
Time taken to run	The amount of seconds it takes for the function to fully execute which includes creating the combinations and ranking them.

The following subsection present the printout that the users would see when they run the *combinations*() function.

4.1.2.2 Experiments Output 1 - Sorted by correlation

Table 15and Table 16present the outputs from running the *combinations()* function with the same parameters except the *list_by* parameter.

This first result (Table 15) is sorted by the correlation of the combinations.

Function call: *combinations(communitiesData,139,10,1)*
Using dataset: Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

	correlation	function	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	Index A	Index B	Function Option
1	-0.8245	A-B	larcPerPop.	nonViolPerPop	0.6039	0.7936	0.0308	141	147	2
2	0.8134	A*B/ (A+B)	medRentpctHousInc.	nonViolPerPop	0.3490	0.7936	0.0197	94	147	5
3	0.8123	A*B	medRentpctHousInc.	nonViolPerPop	0.3490	0.7936	0.0186	94	147	3
4	0.8078	A+B	violentPerPop.	nonViolPerPop	0.6764	0.7936	0.0141	146	147	1
5	0.8078	A+B	assaultPerPop.	nonViolPerPop	0.6211	0.7936	0.0141	137	147	1
6	0.8007	A*B/ (A+B)	pctFemDivorc.	nonViolPerPop	0.5624	0.7936	0.0070	46	147	5
7	0.7986	A+B	robbbPerPop.	nonViolPerPop	0.6446	0.7936	0.0049	135	147	1
8	0.7968	A+B	pctKidsBornNevrMarr	nonViolPerPop	0.6199	0.7936	0.0032	56	147	1
9	0.7963	A*B/ (A+B)	pctAllDivorc.	nonViolPerPop	0.5626	0.7936	0.0026	47	147	5
10	-0.7957	A-B	pctKids2Par.	nonViolPerPop	-0.687	0.7936	0.0020	50	147	2
11	Time taken to run:				389.98	seconds				

Table 15: *combinations* output sorted by combination correlation
This is the top 10 results sorted by combination correlation.
Target Attribute is 139 which is assaultPerPop: number of assaults per 100K population

When looking at Table 15 it can be noted that for every combination of the top ten combinations, attribute B was always the nonViolPerPop attribute. It is also interesting that the simple arithmetic combination of A-B is the best result.

4.1.2.3 Experiments Output 2 - Sorted by gain

Now we will take a look at the *combinations()* output for the same target but instead sorting by the correlation gain of the combinations (Table 16).

Function with the parameters to obtain the following results: *combinations(communitiesData,139,10,2)*
Using dataset: Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.5782	A*B/(A+B)	NAperCap.	burglaries.	-0.0670	0.2378	0.3404	29	138	5
2	0.4847	A*B/(A+B)	NAperCap.	assaults.	-0.0670	0.1534	0.3313	29	136	5
3	0.5421	A-B	persPerFam.	persPerOwnOccup.	0.0381	-0.2110	0.3311	48	71	2
4	-0.4632	(A+B)/(A*B)	pctUrban.	murders.	0.0148	0.1350	0.3281	17	130	6
5	0.4587	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.0670	0.1342	0.3245	29	55	5
6	-0.4531	(A+B)/(A*B)	fold.	murders.	-0.0240	0.1350	0.3180	5	130	6
7	0.4093	A/B	NAperCap.	pctPolicAsian.	-0.0670	0.0948	0.3144	29	116	4
8	0.4166	A*B/(A+B)	NAperCap.	robberies.	-0.0670	0.1037	0.3129	29	134	5
9	0.4584	A*B/(A+B)	NAperCap.	persPoverty.	-0.0670	0.1471	0.3113	29	33	5
10	0.5161	A/B	persPerFam.	persPerOwnOccup.	0.0381	-0.2110	0.3051	48	71	4
11	Time taken to run:				385.42	seconds				

Table 16: *combinations* output sorted by combination correlation gain
Target Attribute is 139 which is assaultPerPop: number of assaults per 100K population
Any interesting piece of information from this result is that it several different arithmetic combinations are in the top 10 results.

It can be noted that none of the same attributes are seen in Table 15 and Table 16. Additionally, although the combined correlations of Table 16 are all lower than those in Table 15, the best gain (.3404) is eleven times better than the gain of the best correlation (0.0308). This information may be useful for an analyst to know, so he/she may want to run both of these combinations to cover different aspects of predicting the target. The evaluations of the predictive power of the attribute combinations in Table 15 and Table 16 can be seen in Table 17 and Table 18.

4.1.2.4 Experiments - Time Performance

The execution time of the *combinations()* function over each of the eleven datasets was collected and graphed in Figure 3. These results show that typically the function took longer with datasets that have a higher number of attributes as expected, however if there was a substantial number of data instances in a dataset it may take longer than a different dataset with more attributes. These tests were run with a computer of the following specifications. 6 GB RAM, Intel i7 CPU 960 @3.20GHz.

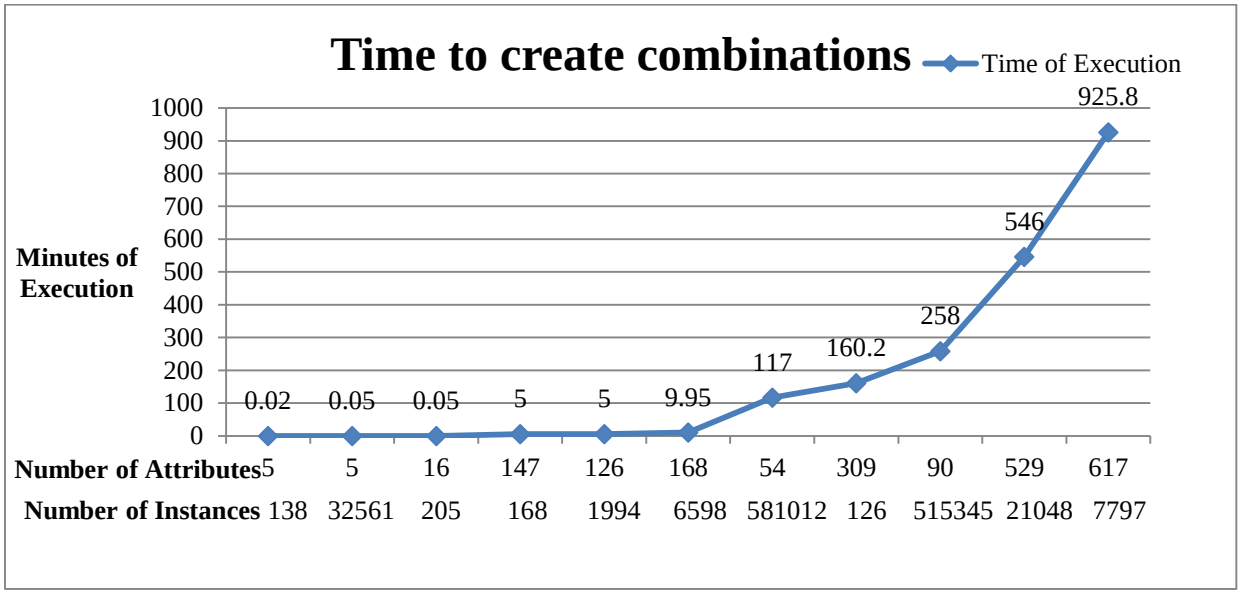

Figure 3: Time to create combinations
Graph showing the time in minutes to run the combination creation function (combinations).

4.1.3 Combination Evaluation

4.1.3.1 Overview

The *eval_combinations()* and *eval_combination()* functions were ran on the eleven datasets in Table 13 multiple times for different target attributes. The *eval_combinations()* function used the results from the *combinations()* experiments as it takes in a data.frame of results.

The following outputs present the printout that the user would see when they run the function. All evaluation tests use linear regression with 10-cross fold validation.

4.1.3.2 Experiments Output 1 - Evaluation of top n attribute combinations sorted by correlation

The *eval_combinations()* function returns the MeanSquareError and Rsquared value for the original dataset model and for the added combinations model. The added combinations model is the original dataset model plus the top n combinations made from the *combinations* function. The lower the MeanSquareError value is, the better. For RSquared, the higher value is better. Table 17 contains the result from using the top 10 attribute combinations sorted by correlation from Table 15.

Function call:*eval_combinations(communitiesData,139,crimeia,10)*
Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

Model	MeanSquareError	RSquared
Original Set of Attributes	0.002098	0.878800113249694
Adding Top n Combinations by correlation to set of attributes	0.000871	0.979875754547201
Time taken: 5.43 seconds		

Table 17 : *eval_combinations* results of top 10 sorted by correlation
The results of running eval_combinations with the top 10 results ordered by correlation from Figure 1.

The added combinations model has a lower MeanSquareError than the original dataset model which means that using the combinations improved the accuracy. The next section will look at the evaluation of the top 10 results ordered by correlation gain to see which top 10 results reduced the prediction error the most.

4.1.3.3 Experiments Output 2 - Evaluation of attribute combinations sorted by correlation gain

In contrast with Table 17, Table 18 shows the results of *eval_combinations()* when using the top 10 results from correlation gain that come from Table 16.

Function call: *eval_combinations(communitiesData,139,crimeib,10)*
Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

Model	MeanSquareError	RSquared
Original Set of Attributes	0.002098	0.878800113249694
Adding Top n Combination by gain to set of attributes	0.00119	0.897802053667757
Time taken: 5.49 seconds		

Table 18: *eval_combinations* results of top 10 sorted by correlation gain
The results of running eval_combinations with the top 10 results ordered by correlation gain from Figure 1.

Once again the added combination set of attributes performed better than the original set of attributes. However comparing the MeanSquareError of the added combination sets of both Table 17 and Table 18 (0.000871 and 0.00119 respectively), the MeanSquareError of Table 17 is better which shows that the top 10 results sorted by correlation worked better than the top 10 results sorted by correlation gain for this dataset.

4.1.3.4 Experiments Top 10 vs. Top 5 - Comparison to original model

The Communities and Crime dataset had 18 different possible target attributes. This allowed for a way to test what ways of using the *combinations()* function provides better results. For this experiment, the *combinations()* function was ran for each of the 18 target attributes, with 4 different sets of customization. The four sets used were the top 10 correlation sorted results, the top 10 correlation gain sorted results, the top 5 correlation sorted results, and the top 5 correlation gain results. All of those different sets of combinations were ran through the *eval_combinations()* function. The results of *eval*_combinations were read to count how often the four different add models were better than the original model (Figure 4).

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

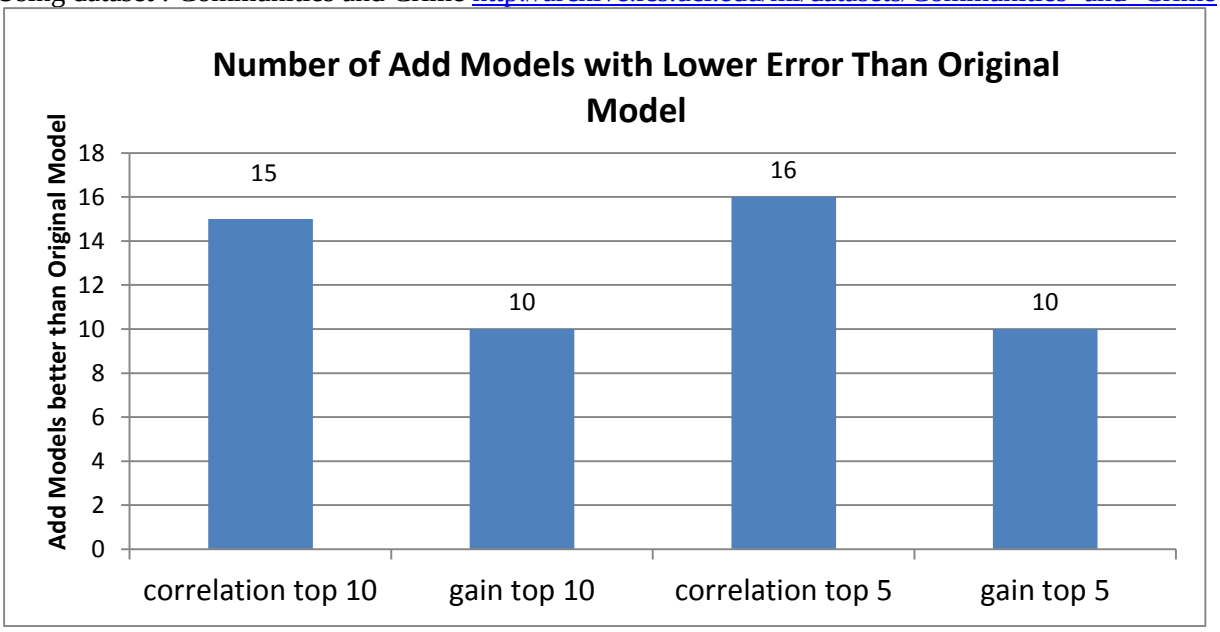

Figure 4: Number of Add Models with Lower Error than Original Model

The add model is the original dataset plus the top set of combinations created from the *combinations function.* This graph shows how many times the different top results performed better than the original model. The total amount of tests was 18.

Overall correlation performed better than correlation gain, however it was interesting to note that the differences between using the top 10 or the top 5 did not seem to make a significant difference in the results in terms of the number of times the datasets with the added combinations are better than the original. We decided to then see whether is better to use the Top 10 or the Top 5 attribute combinations for those experiments in which adding top n attribute combinations produced a reduction in prediction error in comparison with the prediction error of the original attributes.

4.1.3.5 Experiments Top 10 vs. Top 5 Comparison

The next test was to find how often the top 5 results outperformed the the top 10 results when both sets of results performed better than the original dataset. There were 14 (out of 18) experiments in which adding the top n attribute combinations sorted by correlation performed better than the original attributes for both top 10 and top 5.There were 9 (out of 18) experiments in which adding the top n attribute combinations sorted by correlation gain performed better than the original attributes for both top 10 and top 5. The distribution of top 5 performing better than top 10 when both are better is even distributed for correlation(Figure 5)while this distribution was not even when looking at correlation gain. The top 10 results performed better than the top 5 for correlation gain in these cases (Figure 5).

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

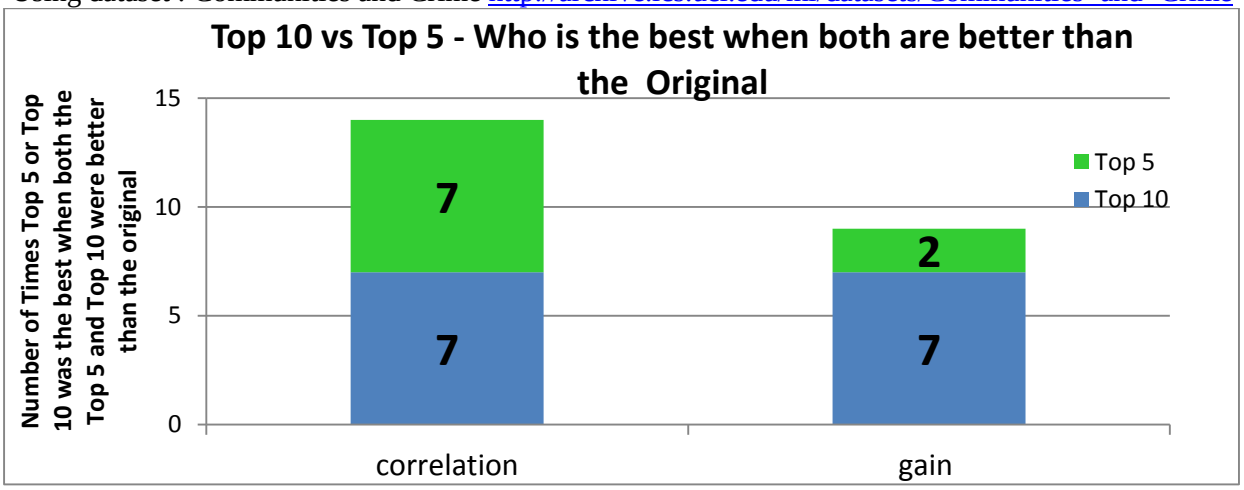

Figure 5: Top 10 vs Top 5 combinations evaluation results
Top 5 and top 10 performed equally for correlation but favored the top 10 when using correlation gain. The order in which the top 5 and the top 10 results are stacked is irrelevant.

4.1.3.6 Experiments Replacing/Adding Top Attribute Combination

Similar tests to those described in Section 4.1.3.2 Experiments Output 1 - Evaluation of top n attribute combinations sorted by correlation were done with the 18 different target attributes, however in this case the *eval_combination()* function was used instead of *eval_combinations()*. The *eval_combination()* function returns the MeanSquareError value for the original dataset model, the added combinations model, and the replace model. The added combinations model in *eval_combination()* is the original dataset model plus the combination made through the parameters given to the *eval_combination()* function. The replace model is the original dataset model plus the combination made through the parameters and the removal of the attributes that make up the combination. The lower the MeanSquareError value is, the better. For RSquared, the higher value is better.. For these experiments only the best combination is used from both correlation and correlation gain. Figure 6 shows that using just the top correlation combination or just the top correlation gain combination led to a fairly frequent

improvement over the original model. The top correlation attribute combination performed better than the top correlation gain attribute combination in our tests.

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

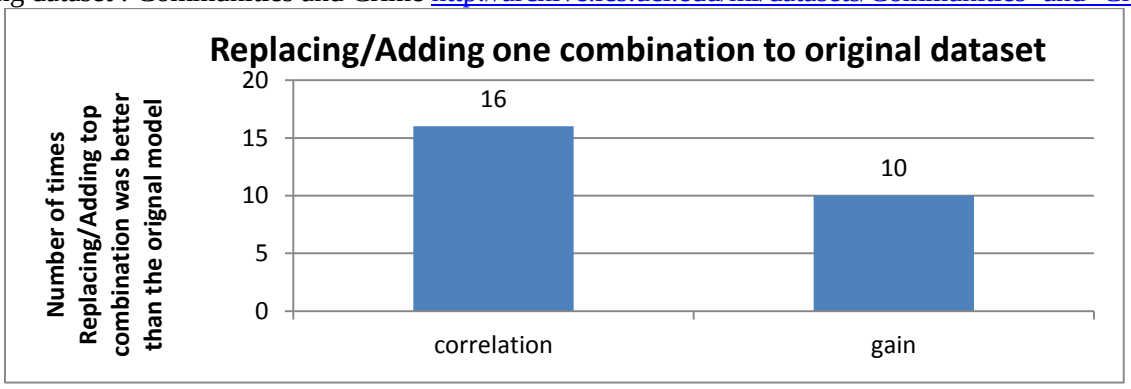

Figure 6: Replacing/Adding one combination
This graph shows how either replacing or adding the best combination from the *combinations* function performed versus the original. There were 18 tests in total.

4.1.3.7 Experiments Replacing Vs Adding - Correlation

The next test was to see how often the add and replace models with the best correlation combination were better than the original model. As shown in Figure 6 we know that correlation overall performed better than the original model 16 times. However, when looking at the replacing and adding when using the best correlation combination (Figure 7) , both the add model and replace model were better than the original model 14 times (out of 16 either of the models were better than the original) .

Using dataset: Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

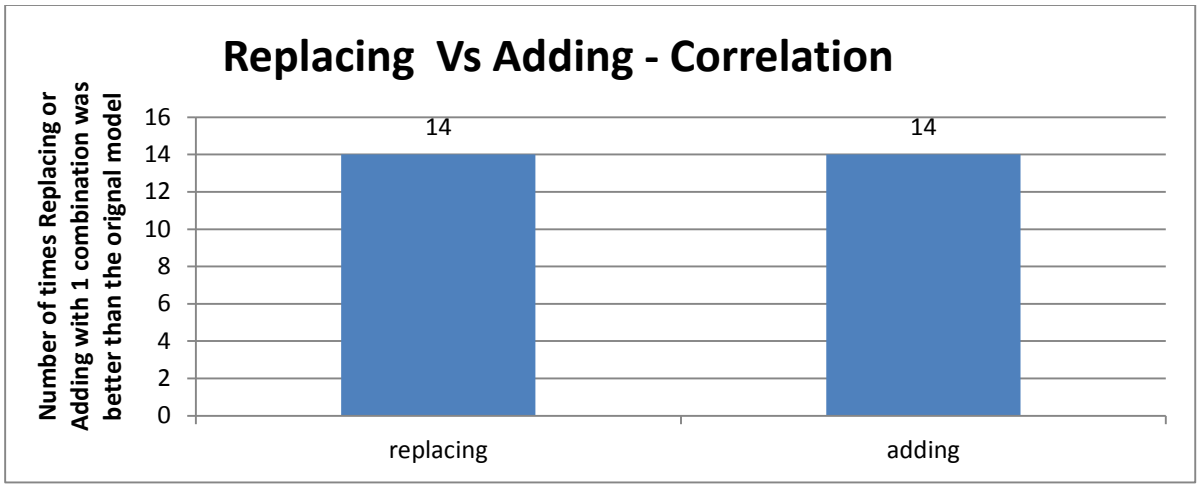

Figure 7: Replacing and Adding using the best correlation combination
This graph shows that replacing and adding using the best correlation combination had the same number of times they were better than the original.

4.1.3.8 Experiments Replacing Vs Adding - Correlation Gain

Similar to section 4.1.3.7 Experiments Replacing Vs Adding - Correlation above, this experiment was done to see how often the add and replace models with the best correlation gain combination were better than the original model. From Figure 6 it was determined that correlation gain performed better than the original 10 times. However, when looking at the replacing and adding when using the best correlation gain combination (Figure 8), replacing performed better than adding. Replacing was better than the original model 9 times (out of the 10 timeseither of the models were better than the original) while adding was better than the original model only 6. Additionaly, there was only one case where replacing did not perform better than the original in the cases where correlation gain was better. This may give indication that when using the singular best result from correlation gain, replacing works better than adding the combination to the dataset.

Using dataset: Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

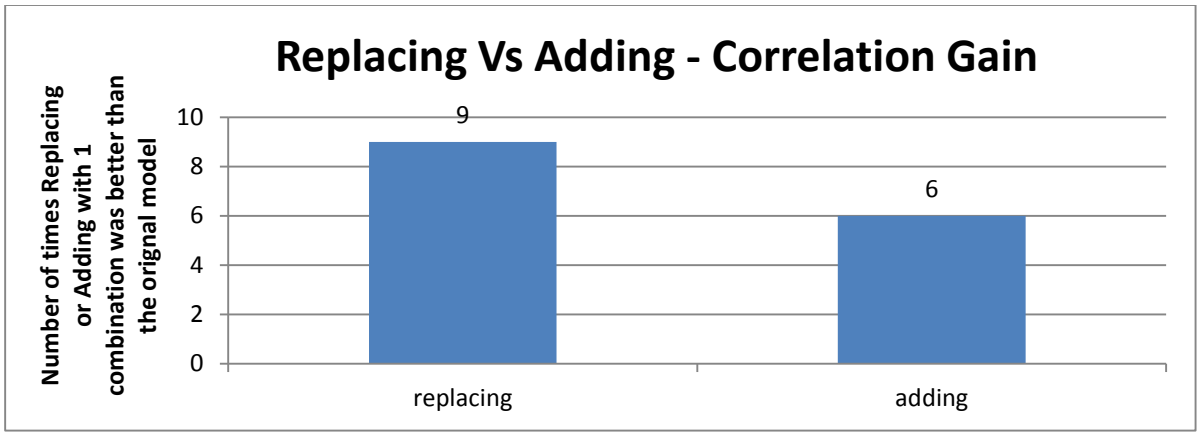

Figure 8: Replacing and Adding using the best correlation gain combination
This graph shows replacing and adding using the best correlation gain combination. Replacing was better than the original 3 more times than adding was.

4.2 Attribute Clustering

4.2.1 Overview

The clustering algorithm was run on a series of categorical and mixed datasets containing both categorical and other types of attributes to evaluate its performance and utility. Results from these experiments are included in the table below. Generally classification techniques such as Logistic Regression, Naïve Bayes, and Decision Trees perform comparably when given the entire dataset and when given only the best attributes from each cluster (in addition to any numeric attributes not considered in the clustering) as input. The resulting loss in accuracy is typically between 1% and 0.5%, and usually results in a reduction in the number of attributes considered by a factor of two. Additionally, as shown in Figure 9: Runtime of AttributeClustering vs Attributes in a Dataset the runtime of the algorithm seems to scale polynomially based on the number of attributes in the dataset. In some cases the clusters produced by our algorithm resulted in an increase in predictive accuracy. Specifically interesting are the results for the Chess and Tic-Tac-Toe datasets. These datasets represent all possible new positions a move could generate at certain points in the game (which point changes between the datasets) and what the outcome would be of those moves. For these types of data sets we see that the use of Decision Trees, Naïve Bayes, and Logistic Regression combined with our clusters results in a significant drop in classification accuracy.

This suggests that the clustering approach is not well suited for this type of dataset, which follows the intuition that when evaluating all positions attempting to group those positions based on which row or column they fall in is unlikely to be very productive for chess and tic-tac-toe.

4.2.2 Attribute Clustering Results

Table 19: Clustering Results Column Descriptions

Column Title	Description
Dataset (type)	The name of the dataset and the type(s) of attributes that are present in it
Number of Attributes & Instances	The number of attributes and instances present in the dataset
Number of Clusters	The number of clusters produced when evaluated with *AttributeClustering()*
Time to Run	The amount of seconds it takes for *AttributeClustering()* to fully execute
Cluster Classification Accuracy	The classification accuracy of the indicated technique using only the best attributes from each cluster
Cluster + Numeric Classification Accuracy	The classification accuracy of the indicated technique using the best attributes from each cluster and all numeric attributes
All Attribute Classification Accuracy	The classification accuracy of the indicated technique using all the attributes from the orginal dataset
Random Cluster Accuracy	The classification accuracy of the indicated technique using a number of randomly selected attributes equal to the number of clusters

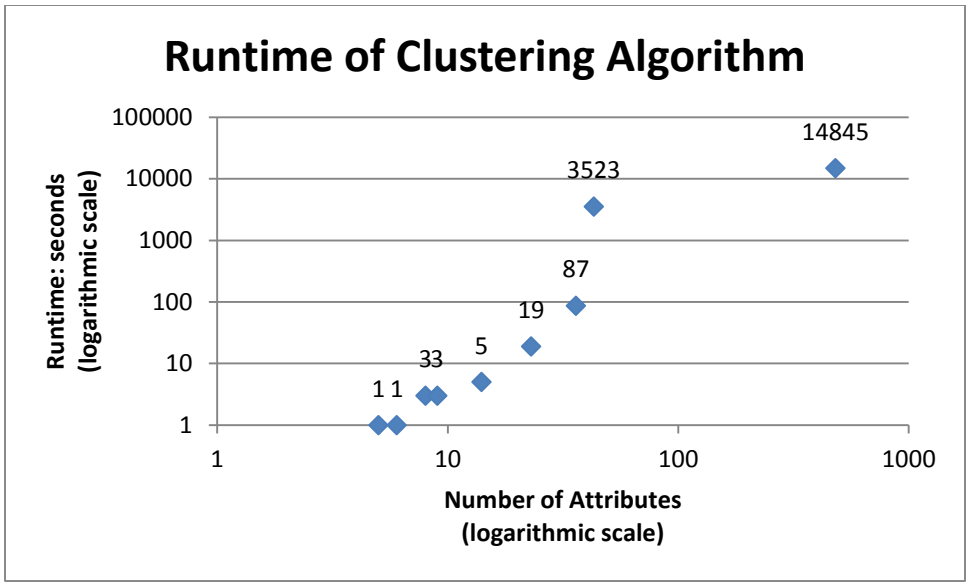

Figure 9: Runtime of AttributeClustering vs Attributes in a Dataset
The x axis contains the number of attributes in the dataset and the y axis represents the run time (in seconds). The runtime above represents the total runtime of the clustering algorithm taken on a Windows 7 desktop running an Intel Core i7-3960X CPU @ 3.30 GHz with 16 GB of Ram.

The full runtime of the clustering algorithm is determined primarily by the number of attributes. The algorithm itself suggests that the runtime will be O(n^3m) bounded, where *n* is the number of attributes in a given dataset and *m* is the number of data instances. The more computationally complex pieces of the algorithm such as calculation of the Bell number and generation of the code table have been optimized, resulting in an increase in performance from the naïve implementation. The most extreme such improvement in runtime is the change from a runtime of approximately one hour for the Census Income dataset (14 attributes, 32000 instances) to around 5 seconds. In addition to the time complexity shown, there are several boundaries on the size of datasets that our algorithm has been capable of processing. Several datasets with more than 1000 attributes and more than 200,000 instances were experimented on using the algorithm in R Studio however due to the size of these data sets the amount of memory allocated to the process governing R Studio was exceeded, causing the evaluation to fail.

Dataset (type)	Number of Attributes & Instances	Number of Clusters	Time to Run
Mushroom (categorical) https://archive.ics.uci.edu/ml/datasets/Mushroom	23 Attributes, 8124 Instances	4	19 seconds
Census Income (mixed) http://archive.ics.uci.edu/ml/datasets/Adult	14 Attributes, 32561 Instances	3	5 seconds
Connect (categorical) https://archive.ics.uci.edu/ml/datasets/Connect-4	43 Attributes, 67557 Instances	7	3523 seconds
Nursery (categorical) https://archive.ics.uci.edu/ml/datasets/Nursery	8 Attributes, 12960 Instances	8	3 seconds
Cup-98 (mixed) (non-classification dataset) https://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html	481 Attributes, 95412 Instances	20	14845 seconds
Car (categorical) https://archive.ics.uci.edu/ml/datasets/Car+Evaluation	6 Attributes, 1728 Instances	6	1 second
Chess (categorical) https://archive.ics.uci.edu/ml/datasets/Chess+%28King-Rook+vs.+King-Pawn%29	36 Attributes, 3196 Instances	8	87 seconds
Hayes-Roth (categorical) https://archive.ics.uci.edu/ml/datasets/Hayes-Roth	5 Attributes, 160 Instances	5	1 second
Tic-Tac-Toe (categorical) https://archive.ics.uci.edu/ml/datasets/Tic-Tac-Toe+Endgame	9 Attributes, 958 Instances	6	3 seconds

Table 20: Attribute Clustering Resluts: Datasets and Runtimes

This figure contains information for the attribute clustering algorithm's runtime and results for different datsets as well as some information about that dataset such as what types of attributes are present, where it can be found, and the number of clusters generated.

The Cup-98 dataset is a non-classification dataset and does not have a designated target attribute. It was evaluated here as a test for scalability but has no meaningful results to report for classification accuracy and therefore is not included in the figures below.

Dataset (type)	Cluster Classification Accuracy	Cluster + Numeric Classification Accuracy	All Attribute Classification Accuracy	Random Cluster Accuracy
Mushroom (categorical)	98.6%	92.6%	100%	92.6%
Census Income (mixed)	78.2%	81.4%	82%	76.9%
Connect (categorical)	65.8%	65.8%	65.8%	65.8%
Nursery (categorical)	85.9%	85.9%	85.9%	85.9%
Car (categorical)	93.5%	93.5%	93.5%	93.5%
Chess (categorical)	77.6%	77.6%	97.7%	63.3%
Hayes-Roth (categorical)	100%	100%	100%	100%
Tic-Tac-Toe (categorical)	74.2%	74.2%	82.5%	65.3%

Table 21: Attribute Clustering Results: Decision Tree Classification Accuracy
This figure contains information from the evaluation of the classification accuracy of decision trees using the clustering described in 3.2 Attribute Clustering.

Classification accuracy for the Cup-98 dataset was not applicable because the decision tree creation algorithms implemented in *R* that were used were incapable of processing several of the attributes in the dataset due to the number of levels present.

Dataset (type)	Cluster Classification Accuracy	Cluster + Numeric Classification Accuracy	All Attribute Classification Accuracy
Mushroom (categorical)	N/A	N/A	N/A
Census Income (mixed)	79.2%	82.1%	83.2%
Connect (categorical)	95.5%	95.5%	90.5%
Nursery (categorical)	100%	100%	100%
Car (categorical)	88.5%	88.5%	88.5%
Chess (categorical)	77.3%	77.3%	97.8%
Hayes-Roth (categorical)	N/A	N/A	N/A
Tic-Tac-Toe (categorical)	72.8%	72.8%	98.3%

Table 22: Attribute Clustering Results: Logistic Regression Classification Accuracy
This figure contains information from the evaluation of the classification accuracy of logistic regression using the clustering described in 3.2 Attribute Clustering.

The datasets designated N/A were unable to be evaluated using Logistic Regression because of the number of factor levels of one or more attributes. These datasets were left unevaluated because transforming them to use Logistic Regression for evaluation would change the dataset from the original and render such evaluation meaningless as a measure of the original dataset.

Dataset (type)	Cluster Classification Accuracy	Cluster + Numeric Classification Accuracy	All Attribute Classification Accuracy
Mushroom (categorical)	98.5%	98.5%	94.4%
Census Income (mixed)	79.2%	80.0%	82.9%
Connect (categorical)	0%	0%	0%
Nursery (categorical)	33.3%	33.3%	33.3%
Car (categorical)	16.7%	16.7%	16.7%
Chess (categorical)	77.3%	77.3%	88.2%
Hayes-Roth (categorical)	N/A	N/A	N/A
Tic-Tac-Toe (categorical)	71.3%	71.3%	69.8%

Table 23: Attribute Clustering Results: Naïve Bayes Classification Accuracy
This figure contains information from the evaluation of the classification accuracy of naïve bayes using the clustering described in 3.2 Attribute Clustering.

The results here show that in some cases the attribute clustering used resulted in improved performace with the Naïve Bayes classification technique. Also shown is that in the cases where Naïve Bayes performed very poorly the clustering was unable to overcome what is apparently an inherently bad fit for Naïve Bayes. In general the clustering yielded either superior or comparable results for the Naïve Bayes classification technique.

4.2.3 Attribute Clustering Report Output

The *AttributeClusteringReport* function is capable of returning both a verbose mode containing the frequently occurring attribute value combinations for the cluster and of just listing the attributes in each cluster. Table 24: Attribute Clustering Report Verbose Output shows the

output for the function with the verbose flag set to TRUE. This provides analysts with the predictive accuracy of the most frequently occurring attribute value combinations for the attributes in each cluster with more than a single attribute.

The output below is taken from the results for the Mushroom dataset. This dataset is available online at the UCI Machine Learning Repository. The dataset describes hypothetical samples of 23 different species of gilled mushrooms in the Agaricus and Lepiota Family. The samples are all classified as either safely edible or unsafe to eat, where unsafe to eat means either that the mushroom is definitively poisonous or that its edibility is unknown and therefore the mushroom is unsafe to eat for risk of accidentally ingesting a poisonous mushroom.

Cluster 2: capsurface capcolor gillattachment stalkcolorabovering veilcolor population

capsurface	capcolor	gillattachment	stalkcolorabovering	veilcolor	population	freq	pred.accuracy
y	n	f	w	w	v	318	0.8050314
y	e	f	p	w	v	264	0.8181818
y	e	f	w	w	v	264	0.8181818
y	n	f	p	w	v	264	0.8181818
s	w	f	w	w	s	260	0.6307692

Cluster 3: bruises odor gillspacing gillsize gillcolor stalkshape stalkroot stalksurfaceabovering stalksurfacebelowring stalkcolorbelowring ringnumber ringtype sporeprintcolor habitat

bruises	odor	gillspacing	gillsize	gillcolor	stalkshape	stalkroot	stalksurfaceabovering
t	n	c	b	n	t	b	s
t	n	c	b	n	t	b	s
t	n	c	b	n	t	b	s
t	n	c	b	n	t	b	s
t	n	c	b	n	t	b	s

stalksurfacebelowring	stalkcolorbelowring	ringnumber	ringtype	sporeprintcolor	habitat	freq	pred.accuracy
s	g	o	p	k	d	72	1
s	g	o	p	n	d	72	1
s	p	o	p	k	d	72	1
s	p	o	p	n	d	72	1
s	w	o	p	k	d	72	1

Table 24: Attribute Clustering Report Verbose Output
The output displayed above contains the 5 most frequently occurring attribute value combinations in the clusters indicated as well as the predictive accuracy of those attribute values when all present together.

In addition to the verbose setting the *AttributeClusteirngReport* function also has the option to only output the names of the attributes in each cluster. This is intended to let the analyst to more easily work with the attributes in each cluster than the normal *AttributeClustering* output would allow.

Attributes in Dataset: capshape, capsurface, capcolor, gillattachment, stalkcolorabovering, veilcolor, population, bruises, odor, gillspacing, gillsize, gillcolor, stalkshape, stalkroot, stalksurfaceabovering, stalksurfacebelowring, stalkcolorbelowring, ringnumber, ringtype, sporeprintcolor, habitat, veiltype

Cluster 1: capshape

Cluster 2: capsurface, capcolor, gillattachment, stalkcolorabovering, veilcolor, population

Cluster 3: bruises, odor, gillspacing, gillsize, gillcolor, stalkshape, stalkroot, stalksurfaceabovering, stalksurfacebelowring, stalkcolorbelowring, ringnumber, ringtype, sporeprintcolor, habitat

Cluster 4: veiltype

4.2.4 Clustering on Datasets with Categorical Attributes and Other Attribute Types

We also evaluated the use of Factor Analysis to group numeric attributes in a manner roughly conceptually comparable to the clustering performed on categorical attributes in our algorithm. In evaluating Factor Analysis we had explored the possibility of using all factors produced as their own cluster as well as the notion of each factor representing a single attribute in the cluster of numeric attributes. In practice neither of these results created particularly meaningful improvements. Of the mixed datasets we tested with, the chi squared goodness-of-fit test rejected all possible numbers of factors that could be generated over those datasets' numeric attributes as "statistically insignificant", indicating the fit was poor. This test for goodness-of-fit was reported by using the summary function on the logistic regression model generated by the function in *R*. The values of this test were in the range of 10^{-23} to 10^{-71}, indicating an extremely poor fit in the cases tested. Additionally, if this value was ingnored and the factors were used with the clusters to predict the target regardless the accuracy gained by this approach was less than 1%. Because of this indication that Factor Analysis was often not a suitable tool to group numeric attributes on the datasets we tested, we decided not to pursue this approach any further and tofocus primarily on the clustering of the categorical attributes.

5 Conclusion

The techniques presented in this paper represent a significant addition to the tools available to a data analyst. Although our techniques are not capable of replacing any of the traditional and more established techniques, they are capable of providing valuable insight about the dataset that would otherwise be unavailable or infeasible to discover. Our attribute combinations techniques allow for an automatic method of trying many possible combinations which will help analysts from having to try and test these combinations on their own. In addition to the value provided by examining combinations, our R package allows analysts to investigate the relationships between attributes by clustering categorical attributes in the dataset. These clusters are relatively computationally inexpensive to generate and provide insight about which attributes are similar predictors of the target class in the data set. Additionally, when using the clusters generated as a basis for future experiments the classification accuracy of techniques such as Logistic Regression and Decision Tree generation is typically preserved to within one percent of the classification accuracy of performing these techniques on the original dataset, while also significantly reducing the dimensional complexity of the dataset. This clustering also allows frequently occurring value combinations of attributes that fulfill a similar predictive role to be discovered, which can serve to identify patterns accross similar attributes in the data set. Overall we expect our package to provide additional insight to analysts for evaluating datasets and discovering patterns therein.

Bibliography

Inc, S. (2004). *Cluster Analysis*. Retrieved March 2015, from University of Texas Arlington: http://www.uta.edu/faculty/sawasthi/Statistics/stcluan.html

Janssen, C. (n.d.). *Data Preprocessing*. Retrieved March 2015, from techopedia: http://www.techopedia.com/definition/14650/data-preprocessing

Lane, D. M. (2013, October). *Values of the Pearson Correlation*. Retrieved March 2015, from OnlineStatBook: http://onlinestatbook.com/2/describing_bivariate_data/pearson.html

Lund, A., & Lund, M. (2015). *Pearson Product-Moment Correlation - When you should run this test, the range of values the coefficienct can take and how to measure strength of association*. Retrieved March 2015, from Laerd statistics: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php

Mampacy, M., & Vreeken, J. (2013). Summarizing Categorical Data by Clustering Attributes. *Data Mining and Knowledge Discovery Journal*, 130-173.

McCormick, K., Abbott, D., Brown, M. S., Khabaza, T., & Mutchler, S. R. (2013). *IBM SPSS Modeler Cookboo.* Packt Publishing Ltd.

N, V., & Vitanyi, P. (2004). Kolmogorov's structure functions and model selection. In *IEEE Transactions on Information Theory* (pp. 3265-3290).

R Project. (n.d.). Retrieved from R Project: http://www.r-project.org/

Rouse, M. (2013, February). *Correlation Coefficient*. Retrieved March 2015, from WhatIs.com: http://whatis.techtarget.com/definition/correlation-coefficient

Vereshchagin N, V. P. (2004). Kolmogorov's structure functions and model selection. *Transactions on Information Theory*, 50(12):3265-3290. IEEE.

Appendix A: Combinations Creation Function Manual

Appendicies A through F follow the format used in (McCormick, Abbott, Brown, Khabaza, & Mutchler, 2013).

Running the combinations creation function

The combination creation function will create combinations of attributes and rank them based on a target attribute. It will report back a data.frame containing details of the top combinations. An example of this data.frame can be seen in Table 15.

Getting ready

The *combinations* function requires a data.frame of the dataset the user wishes to analyze, as well as knowing what the index of the target attribute is. The user should also be aware of whether or not they want to just have the results print out or if the user wants to store the results in a variable to use for *eval_combinations*.

How to do it

1. The user will need to fill in the parameters of the *combinations* function. The parameters are as follows (data,target,num_top,list_by). We will address each one individually.
2. data - The data parameter needs to be a data.frame. data.frames can be created in many different ways such as using a function to read in other datasets as a data.frame. Some common examples are reading in an excel file, a CSV file, or an SPSS model. Each method has parameters that allow for reading the top row of the dataset as column titles, which is necessary for using the combinations function.

a. Excel file:

```
> library(gdata)              # load gdata package
> help(read.xls)              # documentation
> mydata = read.xls("mydata.xls",1)  # read from first sheet
```

b. CSV File:

```
> mydata = read.csv("mydata.csv",1)  # read csv file
```

c. SPSS Model:

```
> library(foreign)            # load the foreign package
> help(read.spss)             # documentation
> mydata = read.spss("myfile", to.data.frame=TRUE)
```

3. Target - This parameter is the index number which corresponds to what column number the target attribute is.
4. num_top - This parameter is a number that represents the number of combinations the user wants displayed. The function will display the top results ranked by one of two values which is decided in *list_by*
5. list_by - This parameter is either the number 1 or 2 which represents the method in which the combinations are ranked. 1 = Correlation, 2 = Correlation Gain
6. Once you have all of the parameters set and you've read in the dataset, you call the function. We recommend setting a variable equal to the result of the function so the user can refer back to the result easily as well as it is necessary for using the *eval_combinations* method.

    a. Setting the result of the function equal to a variable:

    ```
    >result = combinations(data,target,num_top,1ist_by)
    ```

    b. Running the function without setting the results to a variable.

    ```
    >combinations(data, target, num_top, list_by)
    ```

7. View the results
    a. If you set a variable to the result of the function, it will not initially print the results. You can view the top results simply by entering the name of the variable. In the case of example 6a you would just type: >result
    b. If you ran the function without setting the results to a variable the top results will print automatically when the function ends

How it works

The combinations we create contain two attributes which we will refer to as A and B. The arithmetic operations that are used include A+B, A-B, A*B, A/B, A*B/(A+B), and (A+B)/(A*B). These six operations are calculated for every pair of numeric attributes that are not the target attribute.

The combinations are then ranked by their correlation to the target attribute, or the correlation gain. The correlation gain is calculated using the following formula

corr gain = |corr (combination(A,B),Target)| - max(|corr (A,Target)|, | corr (B,Target)|)

The function then returns the combinations sorted with the ranking chosen by the user with the parameter list_by.

There's more...

There are three files that are written every time the combinations function is ran.

out_top_combined.txt	A text file which contains the top combination results (this text file is the same as what is printed out or stored to a variable) This file can be opened in excel and delimited with "#".
out_all_combined.txt	A text file which contains all of the combination results. This file can be opened in excel and delimited with "#".
out_data.txt	A text file which is the normalized version of the dataset which was input to the function. This file can be opened in excel and delimited with "#".

Appendix B: Combinations Evaluations Function 1 Manual

Running the combinations evaluations function #1

The first combination evaluation function will evaluation combinations of attributes based on the data.frame created by the *combinations* function. It will report back a data.frame of results of the linear regression tests comparing the original dataset and the original dataset with the addition of the top combinations given to the dataset. The data.frame will look like Table 17.

Getting ready

The *eval_combinations* function requires a data.frame of the dataset the user wishes to analyze, as well as knowing what the index of the target attribute is. Most importantly, the *eval_combinations* function requires the data.frame created by the *combinations* function. This data.frame contains the top N combinations depending on how many the user chose when running the combinations function.

How to do it

1. The user will need to fill in the parameters of the *eval_combinations* function. The parameters are as follows (data,target,top_n_result,num_fold). We will address each one individually.
2. data - The data parameter needs to be a data.frame. data.frames can be created in many different ways such as using a function to read in other datasets as a data.frame. Some common examples are reading in an excel file, a CSV file, or an SPSS model. Each method has parameters that allow for reading the top row of the dataset as column titles, which is necessary for using the combinations function.
    a. Excel file:

```
> library(gdata)              # load gdata package
> help(read.xls)              # documentation
> mydata = read.xls("mydata.xls",1)  # read from first sheet
```

    b. CSV File:

> mydata = read.csv("mydata.csv",1) # read csv file

c. SPSS Model:

> library(foreign) # load the foreign package

> help(read.spss) # documentation

> mydata = read.spss("myfile", to.data.frame=TRUE)

3. Target - This parameter is the index number which corresponds to what column number the target attribute is.
4. top_n_result - This parameter is a data.frame that was returned by the combinations function which you wish to evaluate. This data.frame is the top n combinations based on what the user chose in the combinations function
5. num_fold - This parameter is a number that will be number of k-fold cross validations ran with linear regression
6. Once you have all of the parameters set and you've read in the dataset, you call the function. We recommend setting a variable equal to the result of the function so the user can refer back to the result easily.
    a. Setting the result of the function equal to a variable:

    > Eval_Result = eval_combinations(data,target,top_n_result,num_fold)

    b. Running the function without setting the results to a variable.

    > eval_combinations(data,target,top_n_result,num_fold)

7. View the results
    a. If you set a variable to the result of the function, it will not initially print the results. You can view the top results simply by entering the name of the variable. In the case of example 6a you would just type: > Eval_Result
    b. If you ran the function without setting the results to a variable the top results will print automatically when the function ends

How it works

The *eval_combinations* function takes the top results that were created with the *combinations* (An example result of the function can be seen in Table 15) function and runs linear regressions tests to compare if the dataset with these new combinations is a better predictor of the target attribute. The linear regression test was already created and usable in R.

There's more...

There are two files that are written every time the combinations function is ran.

out_numeric_data_origin.txt	A text file which contains the original dataset input to the function.This file can be opened in excel and delimited with "#".
out_numeric_data_add.txt	A text file which contains the dataset input to the function plus the top combinations added to the dataset. This file can be opened in excel and delimited with "#".

Appendix C: Combinations Evaluations Function 2 Manual

Running the combinations evaluations function #2

The second combination evaluation function will evaluation a single combination of attributes input specifically into the function parameters. It will report back a data.frame of results of the linear regression tests comparing the original dataset, the original dataset with the addition of the this input combination, and the original dataset plus this input combination and the removal of the two attributes that are used to create the combination.

Getting ready

The *eval_combination* function requires a data.frame of the dataset the user wishes to analyze, as well as knowing what the index of the target attribute is. It also requires the user to have a combination in mind to use for this test.

How to do it

1. The user will need to fill in the parameters of the *eval_combination* function. The parameters are as follows (data,target,indexA,indexB,func_option,num_fold). We will address each one individually.
2. data - The data parameter needs to be a data.frame. data.frames can be created in many different ways such as using a function to read in other datasets as a data.frame. Some common examples are reading in an excel file, a CSV file, or an SPSS model. Each method has parameters that allow for reading the top row of the dataset as column titles, which is necessary for using the combinations function.
    a. Excel file:

```
> library(gdata)              # load gdata package
> help(read.xls)              # documentation
> mydata = read.xls("mydata.xls",1)  # read from first sheet
```

b. CSV File:

```
> mydata = read.csv("mydata.csv",1)  # read csv file
```

c. SPSS Model:

```
> library(foreign)              # load the foreign package
> help(read.spss)               # documentation
> mydata = read.spss("myfile", to.data.frame=TRUE)
```

3. Target - This parameter is the index number which corresponds to what column number the target attribute is.
4. indexA - This parameter is the index number of attribute A for the combination the user will be making.
5. indexB - This parameter is the index number of attribute B for the combination the user will be making.
6. func_option - This parameter is a number number which ranges from one to six based on the six different possible arithmetic combinations. This number is shown in output from the combinations function as well as in several pieces of documentation

   1:=A+B, 2:=A-B, 3:=A*B, 4:=A/B, 5:=A*B/(A+B), 6:=(A+B)/(A*B)
7. num_fold - This parameter is a number that will be number of k-fold cross validations ran with linear regression
8. Once you have all of the parameters set and you've read in the dataset, you call the function. We recommend setting a variable equal to the result of the function so the user can refer back to the result easily.

   a. Setting the result of the function equal to a variable:

   ```
   > Eval_Result = eval_combination(data,target,indexA,indexB,func_option,num_fold)
   ```

   b. Running the function without setting the results to a variable.

   ```
   > eval_combination(data,target,indexA,indexB,func_option,num_fold)
   ```

9. View the results

a. If you set a variable to the result of the function, it will not initially print the results. You can view the top results simply by entering the name of the variable. In the case of example 8a you would just type: > Eval_Result
b. If you ran the function without setting the results to a variable the top results will print automatically when the function ends

How it works

The *eval_combination* function takes the parameters given to create one combination and runs linear regressions tests to compare if the dataset with this new combination is a better predictor of the target attribute. It runs tests on these three models: the original dataset, the original dataset with the addition of the this input combination, and the original dataset plus this input combination and the removal of the two attributes that are used to create the combination. The linear regression test was already created and usable in R.

There's more...

There are three files that are written every time the combinations function is ran.

out_numeric_data_origin.txt	A text file which contains the original dataset input to the function.This file can be opened in excel and delimited with "#".
out_numeric_data_add.txt	A text file which contains the dataset input to the function plus the combination added to the dataset. This file can be opened in excel and delimited with "#".
out_numeric_data_replace.txt	A text file which contains the dataset input to the function minus the attributes that make up the combination plus the one combination added to the dataset. This file can be opened in excel and delimited with "#".

Appendix D: Attribute Clustering Function Manual

Running the attribute clustering function

The attribute clustering function will create clusters for the categorical attributes in a dataset and will return these clusters as a list of lists of integers with the integers indicating the attribute corresponding to that column number in the original dataset.

Getting ready

The *AttributeClustering* function requires a data.frame of the dataset the user wishes to analyze. Categorical attributes must be designated as such in the data frame for datasets with multiple types of attributes. The target attribute is assumed to be the last one in the data.frame.

How to do it

1. The user will need to fill in the parameters of the *AttributeClustering* function. The only parameter is (dataset) and is the data.frame containing the dataset to be analyzed. This data.frame can be populated in a number of ways.
    a. Excel file:

        ```
        > library(gdata)              # load gdata package
        > help(read.xls)              # documentation
        > mydata = read.xls("mydata.xls",1)  # read from first sheet
        ```

    b. CSV File:

        ```
        > mydata = read.csv("mydata.csv",1)  # read csv file
        ```

    c. SPSS Model:

        ```
        > library(foreign)            # load the foreign package
        > help(read.spss)             # documentation
        > mydata = read.spss("myfile", to.data.frame=TRUE)
        ```

2. Once you have all of the parameters set and you've read in the dataset, you call the function. We recommend setting a variable equal to the result of the function so the user can refer back to the result easily as well as it is necessary for using the *AttirbuteClusteringReport* and *AttributeClusterAnalysis* methods. This can be done with the code

   >clusters = *AttributeClustering*(dataset)

3. To view the results we recommend running the *AttributeClusteringReport* function with the verbose parameter set to FALSE.

How it works

The clustering function See Section **Error! Reference source not found.** for more details.

AttributeClustering(dataset);

Input Parameter	Parameter Description
dataset: **data.frame**	The dataset read into R as a *data.frame*. Categorical attributes must be identified as such in *R* for datasets with mixed attribute types. Any mixed datasets stored in formats that represent categorical attributes with integers will require additional preprocessing to identify which attributes are categorical. The algorithm expects the last attribute to be the target.

Output Item	Item Description
clusters: **list of list of integers**	This is the returned value from the function. This list itself contains lists that represent each cluster. The list representing a cluster contains an integer to represent each member attribute. These integers reflect that attribute's column number in the original dataset. Each attribute is a member of only one cluster. The order of the attributes in the list is a result of the hierarchical clustering used. This returned list should be stored in a variable to use with the other functions.

Appendix E: Attribute Clustering Report Function Manual

Running the attribute clustering report function

The attribute clustering report function will report information on the attribute clusters created by the *AttributeClustering* function. There are two settings for this function, verbose = TRUE and verbose = FALSE. The output is printed to the terminal.

Getting ready

The *AttributeClusteringReport* function requires a data.frame of the dataset the user wishes to analyze. Additionally it requires clusters associated with the dataset in the format returned by *AttributeClustering*.

How to do it

1. The user will need to fill in the parameters of the *AttributeClustering* function. The first parameter is (clusters) and is the attribute clustering associated with the dataset that is being reported on. This uses the same format as the return value from the *AttributeClustering* function.
2. The second parameter is (dataset) and is the data.frame containing the dataset to be analyzed. This data.frame can be populated in a number of ways.
    a. Excel file:

        ```
        > library(gdata)              # load gdata package
        > help(read.xls)              # documentation
        > mydata = read.xls("mydata.xls",1)  # read from first sheet
        ```

    b. CSV File:

        ```
        > mydata = read.csv("mydata.csv",1)  # read csv file
        ```

    c. SPSS Model:

```
> library(foreign)          # load the foreign package
> help(read.spss)           # documentation
> mydata = read.spss("myfile", to.data.frame=TRUE)
```

3. The final parameter is (verbose). This parameter accepts TRUE or FALSE and indicates whether or not additional information about the frequently occurring attribute value combinations for each cluster is desired.
4. Once you have all of the parameters set and you've read in the dataset, you call the function. This can be done with the code

>*AttributeClusteringReport*(clusters, dataset, verbose)

How it works

The function iterates over the attributes in each cluster to convert them into a more human-readable format.

AttributeClusteringReport(clusters, dataset);

Input Parameter	Parameter Description
dataset: **data.frame**	The dataset read into R as a *data.frame*. Categorical attributes must be identified as such in *R* for datasets with mixed attribute types. Any mixed datasets stored in formats that represent categorical attributes with integers will require additional preprocessing to identify which attributes are categorical. The algorithm expects the last attribute to be the target.
clusters: **list of lists of integers**	The clusters are the result of the *AttributeClustering()* function.

Output Item	Item Description
cluster information: **terminal output**	This is the returned value from the function. Each attribute in a cluster is translated from raw form (the *clusters* input) to the attribute names given in the original dataset. and a cluster number and is printed to the terminal. Additionally the most frequently occurring

	attribute value combinations for each cluster are printed to the terminal along with their classification accuracy.

Appendix F: Attribute Cluster Analysis Function Manual

Running the attribute cluster analysis function

The attribute cluster analysis function will order the attributes in clusters based on their predictive accuracy as well as evaluate their classification accuracy using Decision Trees , Naïve Bayes, and Logistic Regression. The classification accuracy output is printed to the terminal and the ordered clusters are returned as a list of lists of pairs of integers and floating point numbers representing the attribute corresponding to the column indicated by the integer in the orginal dataset and the predictive accuracy of that attribute. The data.frame required for this function is subject to the same constraints as in the *AttributeClustering* function.

Getting ready

The *AttributeClusterAnalysis* function requires a data.frame of the dataset the user wishes to analyze. Additionally it requires clusters associated with the dataset in the format returned by *AttributeClustering*.

How to do it

1. The user will need to fill in the parameters of the *AttributeClustering* function. The first parameter is (clusters) and is the attribute clustering associated with the dataset that is being reported on. This uses the same format as the return value from the *AttributeClustering* function.
2. The second parameter is (dataset) and is the data.frame containing the dataset to be analyzed. This data.frame can be populated in a number of ways.
    a. Excel file:

        ```
        > library(gdata)              # load gdata package
        > help(read.xls)              # documentation
        > mydata = read.xls("mydata.xls",1)  # read from first sheet
        ```

    b. CSV File:

> mydata = read.csv("mydata.csv",1) # read csv file

c. SPSS Model:

> library(foreign) # load the foreign package

> help(read.spss) # documentation

> mydata = read.spss("myfile", to.data.frame=TRUE)

3. Once you have all of the parameters set and you've read in the dataset, you call the function. We recommend storing the clusters returned in a variable for possible future use.This can be done with the code

>orderedclusters = *AttributeClusterReport*(clusters, dataset, verbose)

How it works

The function iterates over each cluster to evaluate it and order the attributes contained within.

Input Parameter	Parameter Description
dataset: **data.frame**	The dataset read into R as a *data.frame*. Categorical attributes must be identified as such in *R* for datasets with mixed attribute types. Any mixed datasets stored in formats that represent categorical attributes with integers will require additional preprocessing to identify which attributes are categorical. The algorithm expects the last attribute to be the target.
clusters: **list of lists of integers**	The clusters are the result of the *AttributeClustering()* function and are the main focus of evaluation.

Output Item	Item Description
clusters: **list of lists of pairs of integers and classification accuracy**	This is the returned value from the function. This list itself contains lists that represent each cluster. The list representing a cluster contains an integer to represent each member attribute paired with that attributes classification accuracy. These integers reflect that attribute's column number in the original dataset. Each attribute is a member of only one cluster. The

	order of the lists is sorted in decreasing order by classification accuracy.
classification accuracy information: **printed output on terminal**	The classification accuracy of the attribute clusters is printed to the terminal for three different classification techniques based on three different subsets of the original dataset's attributes. The classification techniques used are Logistic Regression, Naïve Bayes, and Decision Trees. The subsets of attributes used are: • the best attribute from each cluster, • the best attribute from each cluster and each numeric attribute, and • all attributes.

Appendix G: Combination Creation Additional Tests

Combination Creation Additional Tests - Creating top 10 and top 5 correlation and correlation gain combinations for 18 different target attributes

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

To help easy the readability of these results, note that the naming of the variables matches the following format.

crimeLetter1Letter2

Letter 1 is the set of test. This letter ranges from a to r for the 18 different target attributes.

Letter2 is for what type of combination creation is made and ranges from a to d.

a = top 10 sorted by correlation

b = top 10 sorted by correlation gain

c = top 5 sorted by correlation

d = top 5 sorted by correlation gain

The command executive will be displayed above each table:

crimera = combinations(dat4,130,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9989704	A*B	pop.	murdPerPop.	0.9592 944	0.33562 7919	0.0396 76054	6	131	3

2	0.9989445	A*B	persUrban.	murdPerPop.	0.9574842	0.33562 7919	0.0414 60335	16	131	3
3	0.9840821 7	A/B	persPoverty.	gangUnit.	0.9761283	-0.02095 7945	0.0079 53825	33	127	4
4	0.9838696 2	A*B/(A+B)	kidsBornNe vrMarr.	burglaries.	0.9781384	0.95612 5436	0.0057 31205	55	138	5
5	0.9827527 7	A*B	kidsBornNe vrMarr.	pctFgnImmig.10.	0.9781384	0.08891 5445	0.0046 14359	55	61	3
6	0.9827446 3	A+B	kidsBornNe vrMarr.	autoTheft.	0.9781384	0.97070 5928	0.0046 06214	55	142	1
7	0.9824222 8	A*B/(A+B)	kidsBornNe vrMarr.	autoTheft.	0.9781384	0.97070 5928	0.0042 83871	55	142	5
8	0.9823489	A*B	persPoverty.	murdPerPop.	0.9761283	0.33562 7919	0.0062 20556	33	131	3
9	0.9819455 1	A*B/(A+B)	kidsBornNe vrMarr.	larcenies.	0.9781384	0.93376 323	0.0038 07102	55	140	5
10	0.9818452 5	A*B/(A+B)	kidsBornNe vrMarr.	policCarsAvail.	0.9781384	0.87321 731	0.0037 06839	55	124	5
11	Time	taken	to	run:	402.82	seconds				

crimerb = combinations(dat4,130,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9593858 3	A*B	landArea.	popDensity.	0.1841122	0.20917 7109	0.7502 08721	121	122	3
2	0.6915079 6	A*B/(A+B)	landArea.	popDensity.	0.1841122	0.20917 7109	0.4823 30849	121	122	5
3	0.7950695 4	A*B	landArea.	pctUsePubTrans.	0.1841122	0.33381 822	0.4612 51324	121	123	3

4	0.6141727	A*B	pctNotSpeakEng.	landArea.	0.1108129	0.18411 2249	0.4300 60525	67	121	3
5	0.6025545 3	A*B/(A+B)	pctNotSpeakEng.	landArea.	0.1108129	0.18411 2249	0.4184 42281	67	121	5
6	0.5925523 9	A*B	pctImmig.5.	landArea.	0.1195084	0.18411 2249	0.4084 4014	63	121	3
7	0.5878469 1	A*B	pctImmig.3.	landArea.	0.1160784	0.18411 2249	0.4037 34664	62	121	3
8	0.5871708 5	A*B	pctImmig.8.	landArea.	0.1238542	0.18411 2249	0.4030 58599	64	121	3
9	0.5847594 7	A*B	pctImmig.10.	landArea.	0.1270777	0.18411 2249	0.4006 4722	65	121	3
10	0.5817050 4	A*B/(A+B)	pctImmig.3.	landArea.	0.1160784	0.18411 2249	0.3975 92787	62	121	5
11	Time	taken	to	run:	400.31	seconds				

crimerc = combinations(dat4,130,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9989704 4	A*B	pop.	murdPerPop.	0.9592944	0.33562 7919	0.0396 76054	6	131	3
2	0.9989445	A*B	persUrban.	murdPerPop.	0.9574842	0.33562 7919	0.0414 60335	16	131	3
3	0.9840821 7	A/B	persPoverty.	gangUnit.	0.9761283	-0.02095 7945	0.0079 53825	33	127	4
4	0.9838696 2	A*B/(A+B)	kidsBornNevrMarr.	burglaries.	0.9781384	0.95612 5436	0.0057 31205	55	138	5
5	0.9827527 7	A*B	kidsBornNevrMarr.	pctFgnImmig.10.	0.9781384	0.08891 5445	0.0046 14359	55	61	3
6	Time	taken	to	run:	399.64	seconds				

crimerd = combinations(dat4,130,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	function Option
1	0.9593858 3	A*B	landArea.	popDensity.	0.1841 122	0.20917 7109	0.7502 08721	121	122	3
2	0.6915079 6	A*B/(A+B)	landArea.	popDensity.	0.1841 122	0.20917 7109	0.4823 30849	121	122	5
3	0.7950695 4	A*B	landArea.	pctUsePubTrans.	0.1841 122	0.33381 822	0.4612 51324	121	123	3
4	0.6141727 7	A*B	pctNotSpea kEng.	landArea.	0.1108 129	0.18411 2249	0.4300 60525	67	121	3
5	0.6025545 3	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1108 129	0.18411 2249	0.4184 42281	67	121	5

crimeaa = combinations(dat4,131,10,1)crimeaa

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7399238 4	A*B/(A+B)	pctPoverty.	robbbPerPop.	0.4829 53	0.68993 4075	0.0499 89763	34	135	5
2	0.7336653 7	A+B	pctBlack.	robbbPerPop.	0.6516 878	0.68993 4075	0.0437 31298	8	135	1
3	0.7327529 5	A*B/(A+B)	pctPubAsst.	robbbPerPop.	0.5207 87	0.68993 4075	0.0428 18879	23	135	5
4	0.7315356 8	A*B/(A+B)	pctKidsBor nNevrMarr.	robbbPerPop.	0.6796 068	0.68993 4075	0.0416 01603	56	135	5
5	0.7300411	A*B/(A+B)	pctVacantB oarded.	robbbPerPop.	0.5547 624	0.68993 4075	0.0401 07026	80	135	5

6	0.7299845 1	A+B	pctPolicBla ck.	robbbPerPop.	0.5424 678	0.68993 4075	0.0400 50438	114	135	1
7	0.7283930 8	A+B	pctKidsBor nNevrMarr.	robbbPerPop.	0.6796 068	0.68993 4075	0.0384 59002	56	135	1
8	0.7278835 7	A+B	pctKidsBor nNevrMarr.	pctPolicBlack.	0.6796 068	0.54246 7772	0.0482 76757	56	114	1
9	0.7273030 7	A*B/(A+B)	pctUnemplo y.	robbbPerPop.	0.4653 863	0.68993 4075	0.0373 68997	38	135	5
10	-0.72649	A-B	pctWhite.	robbbPerPop.	-0.6660 04	0.68993 4075	0.0365 55909	9	135	2
11	Time	taken	to	run:	518.76	seconds				

crimeab = combinations(dat4,131,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7047045 5	A*B/(A+B)	NAperCap.	murders.	-0.0479 8	0.33562 7919	0.3690 76631	29	130	5
2	-0.6329585	A-B	perCapInc.	whitePerCap.	-0.2787 83	-0.15117 3247	0.3541 75093	26	27	2
3	0.5751095 3	A*B/(A+B)	NAperCap.	robberies.	-0.0479 8	0.22729 6181	0.3478 13354	29	134	5
4	0.6771034 6	A/B	murders.	burglaries.	0.3356 279	0.33576 1957	0.3413 41504	130	138	4
5	0.6710403 6	A*B/(A+B)	blackPerCa p.	murders.	-0.1801 89	0.33562 7919	0.3354 12437	28	130	5

6	0.6196094	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.04798	0.28758 8545	0.3320 20895	29	55	5
7	0.6672093 8	A*B/(A+B)	otherPerCap.	murders.	-0.107666	0.33562 7919	0.3315 81457	31	130	5
8	0.5092178 5	A-B	persPerFam.	persPerOwnOccup.	-0.1797863	-0.05593 4471	0.3294 31529	48	71	2
9	0.6573624 8	A/B	murders.	larcenies.	0.3356279	0.30690 6649	0.3217 3456	130	140	4
10	0.5999284 3	A*B/(A+B)	otherPerCap.	kidsBornNevrMarr.	-0.107666	0.28758 8545	0.3123 39885	31	55	5
11	Time	taken	to	run:	386.75	seconds				

crimeac = combinations(dat4,131,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7399238 4	A*B/(A+B)	pctPoverty.	robbbPerPop.	0.482953	0.68993 4075	0.0499 89763	34	135	5
2	0.7336653 7	A+B	pctBlack.	robbbPerPop.	0.6516878	0.68993 4075	0.0437 31298	8	135	1
3	0.7327529 5	A*B/(A+B)	pctPubAsst.	robbbPerPop.	0.520787	0.68993 4075	0.0428 18879	23	135	5
4	0.7315356 8	A*B/(A+B)	pctKidsBornNevrMarr.	robbbPerPop.	0.6796068	0.68993 4075	0.0416 01603	56	135	5
5	0.7300411	A*B/(A+B)	pctVacantBoarded.	robbbPerPop.	0.5547624	0.68993 4075	0.0401 07026	80	135	5
6	Time	taken	to	run:	382.36	seconds				

crimead = combinations(dat4,131,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7047045	A*B/(A+B)	NAperCap.	murders.	-0.04798	0.33562 7919	0.3690 76631	29	130	5
2	-0.6329585	A-B	perCapInc.	whitePerCap.	-0.27883	-0.15117 3247	0.3541 75093	26	27	2
3	0.5751095 3	A*B/(A+B)	NAperCap.	robberies.	-0.04798	0.22729 6181	0.3478 13354	29	134	5
4	0.6771034 6	A/B	murders.	burglaries.	0.3356 279	0.33576 1957	0.3413 41504	130	138	4
5	0.6710403 6	A*B/(A+B)	blackPerCa p.	murders.	-0.18018 9	0.33562 7919	0.3354 12437	28	130	5
6	Time	taken	to	run:	397.97	seconds				

crimeba = combinations(dat4,132,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9769854 9	A*B	pop.	rapesPerPop.	0.8744 752	0.28601 2337	0.1025 10274	6	133	3
2	0.9765266	A*B	persUrban.	rapesPerPop.	0.8742 584	0.28601 2337	0.1022 68233	16	133	3
3	0.9572625 4	A*B	persPoverty.	rapesPerPop.	0.8526 566	0.28601 2337	0.1046 05893	33	133	3
4	0.9527582	A*B	rapesPerPop .	autoTheft.	0.2860 123	0.86582 6862	0.0869 31338	133	142	3

5	0.9426076	A*B	rapesPerPop.	burglaries.	0.2860123	0.893675218	0.048932382	133	138	3
6	0.93996247	A*B	rapesPerPop.	larcenies.	0.2860123	0.882667592	0.057294879	133	140	3
7	0.93341378	A*B/(A+B)	pop.	violentPerPop.	0.8744752	0.328420876	0.058938572	6	146	5
8	0.93086355	A*B/(A+B)	persPoverty.	violentPerPop.	0.8526566	0.328420876	0.078206902	33	146	5
9	0.92645015	A*B/(A+B)	persUrban.	violentPerPop.	0.8742584	0.328420876	0.052191788	16	146	5
10	0.920186	A*B	communityCode.	burglaries.	-0.010734	0.893675218	0.026510785	4	138	3
11	Time	taken	to	run:	384.25	seconds				

crimebb = combinations(dat4,132,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
1	0.87421309	A*B	landArea.	popDensity.	0.2649455	0.176885699	0.609267559	121	122	3
2	0.78307773	A*B/(A+B)	landArea.	popDensity.	0.2649455	0.176885699	0.518132198	121	122	5
3	0.7155343	A*B/(A+B)	pctNotSpeakEng.	landArea.	0.1050782	0.264945528	0.450588776	67	121	5
4	0.73905832	A*B/(A+B)	landArea.	pctUsePubTrans.	0.2649455	0.289666552	0.449391772	121	123	5
5	0.70969801	A*B/(A+B)	pctImmig.3.	landArea.	0.1146935	0.264945528	0.444752487	62	121	5
6	0.70748939	A*B/(A+B)	pctImmig.5.	landArea.	0.118062	0.264945528	0.442504386	63	121	5

7	0.7050692 4	A*B/(A+B)	pctImmig.1 0.	landArea.	0.1274 903	0.26494 5528	0.4401 23717	65	121	5
8	0.7034769 3	A*B/(A+B)	pctImmig.8.	landArea.	0.1232 242	0.26494 5528	0.4385 31406	64	121	5
9	0.7009154 9	A*B	landArea.	pctUsePubTrans.	0.2649 455	0.28966 6552	0.4112 4894	121	123	3
10	0.6717086 2	A*B/(A+B)	pctBlack.	landArea.	0.2095 872	0.26494 5528	0.4067 63087	8	121	5
11	Time	taken	to	run:	381.78	seconds				

crimebc = combinations(dat4,132,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9769854 9	A*B	pop.	rapesPerPop.	0.8744 752	0.28601 2337	0.1025 10274	6	133	3
2	0.9765266	A*B	persUrban.	rapesPerPop.	0.8742 584	0.28601 2337	0.1022 68233	16	133	3
3	0.9572625 4	A*B	persPoverty.	rapesPerPop.	0.8526 566	0.28601 2337	0.1046 05893	33	133	3
4	0.9527582	A*B	rapesPerPop .	autoTheft.	0.2860 123	0.86582 6862	0.0869 31338	133	142	3
5	0.9426076	A*B	rapesPerPop .	burglaries.	0.2860 123	0.89367 5218	0.0489 32382	133	138	3
6	Time	taken	to	run:	393.54	seconds				

crimebd = combinations(dat4,132,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.8742130 9	A*B	landArea.	popDensity.	0.2649 455	0.17688 5699	0.6092 67559	121	122	3
2	0.7830777 3	A*B/(A+B)	landArea.	popDensity.	0.2649 455	0.17688 5699	0.5181 32198	121	122	5
3	0.7155343	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1050 782	0.26494 5528	0.4505 88776	67	121	5
4	0.7390583 2	A*B/(A+B)	landArea.	pctUsePubTrans.	0.2649 455	0.28966 6552	0.4493 91772	121	123	5
5	0.7096980 1	A*B/(A+B)	pctImmig.3.	landArea.	0.1146 935	0.26494 5528	0.4447 52487	62	121	5
6	Time	taken	to	run:	387.3	seconds				

crimeca = combinations(dat4,133,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.6342601 2	A*B/(A+B)	larcPerPop.	violentPerPop.	0.4946 205	0.58314 5773	0.0511 14349	141	146	5
2	0.6341878 8	A+B	pctMaleDiv orc.	violentPerPop.	0.5211 336	0.58314 5773	0.0510 42107	44	146	1
3	0.6303489 2	A+B	pctAllDivor c.	violentPerPop.	0.5229 39	0.58314 5773	0.0472 03151	47	146	1
4	0.6303272 1	A+B	larcPerPop.	violentPerPop.	0.4946 205	0.58314 5773	0.0471 81441	141	146	1
5	0.6301829 5	A*B/(A+B)	pctMaleDiv orc.	violentPerPop.	0.5211 336	0.58314 5773	0.0470 37182	44	146	5
6	0.6278202 6	A*B/(A+B)	pctHousWO phone.	violentPerPop.	0.4441 443	0.58314 5773	0.0446 74491	83	146	5
7	0.6272161 7	A*B/(A+B)	violentPerP op.	nonViolPerPop..	0.5831 458	0.53888 7625	0.0440 70399	146	147	5

8	0.6245660 3	A*B/(A+B)	pctAllDivor c.	violentPerPop.	0.5229 39	0.58314 5773	0.0414 20259	47	146	5
9	0.6236812	A+B	violentPerP op.	nonViolPerPop..	0.5831 458	0.53888 7625	0.0405 35429	146	147	1
10	0.6225788	A/B	rapes.	larcenies.	0.2860 123	0.16994 253	0.3365 66461	132	140	4
11	Time	taken	to	run:	382.77	seconds				

crimecb = combinations(dat4,133,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.4061392 5	A/B	pctBornStat eResid.	pctPolicAsian.	-0.0246 83	0.06130 809	0.3448 31161	100	116	4
2	0.4046877 1	A/B	pctSpeakOn lyEng.	pctPolicAsian.	0.0356 37	0.06130 809	0.3433 79618	66	116	4
3	0.3996094 1	(A+B)/(A*B)	pctHisp.	pctPolicAsian.	0.0417 05	0.06130 809	0.3383 01316	11	116	6
4	0.6225788	A/B	rapes.	larcenies.	0.2860 123	0.16994 253	0.3365 66461	132	140	4
5	0.6080841 9	A*B/(A+B)	NAperCap.	rapes.	-0.0956 42	0.28601 2337	0.3220 71858	29	132	5
6	-0.4187634	(A+B)/(A*B)	pctUrban.	murders.	-0.0495 76	0.10593 4536	0.3128 28877	17	130	6
7	0.3714710 8	(A+B)/(A*B)	pctPolicHis p.	pctPolicAsian.	0.0303 643	0.06130 809	0.3101 62991	115	116	6
8	0.3484252	A/B	pctEmployP rofServ.	pctPolicAsian.	-0.0383 78	0.06130 809	0.2871 17112	41	116	4

9	0.4037729 4	A*B/(A+B)	NAperCap.	assaults.	-0.095642	0.11911 1676	0.2846 61265	29	136	5
10	0.5706451 2	A*B/(A+B)	blackPerCap.	rapes.	-0.242204	0.28601 2337	0.2846 32781	28	132	5
11	Time	taken	to	run:	395.54	seconds				

crimecc = combinations(dat4,133,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.6342601 2	A*B/(A+B)	larcPerPop.	violentPerPop.	0.4946 205	0.58314 5773	0.0511 14349	141	146	5
2	0.6341878 8	A+B	pctMaleDivorc.	violentPerPop.	0.5211 336	0.58314 5773	0.0510 42107	44	146	1
3	0.6303489 2	A+B	pctAllDivorc.	violentPerPop.	0.5229 39	0.58314 5773	0.0472 03151	47	146	1
4	0.6303272 1	A+B	larcPerPop.	violentPerPop.	0.4946 205	0.58314 5773	0.0471 81441	141	146	1
5	0.6301829 5	A*B/(A+B)	pctMaleDivorc.	violentPerPop.	0.5211 336	0.58314 5773	0.0470 37182	44	146	5
6	Time	taken	to	run:	388.45	seconds				

crimecd = combinations(dat4,133,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.4061392 5	A/B	pctBornStateResid.	pctPolicAsian.	-0.024683	0.06130 809	0.3448 31161	100	116	4

2	0.4046877 1	A/B	pctSpeakOn lyEng.	pctPolicAsian.	0.0356 37	0.06130 809	0.3433 79618	66	116	4
3	0.3996094 1	(A+B)/(A*B)	pctHisp.	pctPolicAsian.	0.0417 05	0.06130 809	0.3383 01316	11	116	6
4	0.6225788	A/B	rapes.	larcenies.	0.2860 123	0.16994 253	0.3365 66461	132	140	4
5	0.6080841 9	A*B/(A+B)	NAperCap.	rapes.	-0.0956 42	0.28601 2337	0.3220 71858	29	132	5
6	Time	taken	to	run:	426.98	seconds				

crimeda= combinations(dat4,134,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9993572 3	A*B	persUrban.	robbbPerPop.	0.9525 015	0.31990 2076	0.0468 55723	16	135	3
2	0.9993262 2	A*B	pop.	robbbPerPop.	0.9546 475	0.31990 2076	0.0446 78739	6	135	3
3	0.9912761 8	A*B	persPoverty.	robbbPerPop.	0.9598 893	0.31990 2076	0.0313 86867	33	135	3
4	0.9893435	A+B	officDrugU nits.	assaults.	0.9240 217	0.96153 9197	0.0278 04302	118	136	1
5	0.9881178 6	A*B	kidsBornNe vrMarr.	pctSmallHousUnits.	0.9751 072	0.11934 496	0.0130 1071	55	75	3
6	0.9877736 3	A*B/(A+B)	kidsBornNe vrMarr.	assaults.	0.9751 072	0.96153 9197	0.0126 66478	55	136	5
7	0.9865600 4	A*B	pctHousWO plumb.	assaults.	0.0965 376	0.96153 9197	0.0250 20845	84	136	3

8	0.9858549	A*B	kidsBornNevrMarr.	rentQrange.	0.9751072	0.033361012	0.010747744	55	92	3
9	0.98572724	A*B	kidsBornNevrMarr.	rentMed.	0.9751072	-0.010228232	0.01062009	55	90	3
10	0.98536622	A*B	robbbPerPop.	autoTheft.	0.3199021	0.956012401	0.02935382	135	142	3
11	Time	taken	to	run:	389.38	seconds				

crimedb = combinations(dat4,134,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
1	0.95474244	A*B	landArea.	popDensity.	0.1418421	0.226842217	0.727900225	121	122	3
2	0.78995042	A*B	landArea.	pctUsePubTrans.	0.1418421	0.334173788	0.455776633	121	123	3
3	0.53273753	A*B	pctNotSpeakEng.	landArea.	0.1112602	0.141842117	0.390895414	67	121	3
4	0.5306067	A*B	pctImmig.5.	landArea.	0.1253971	0.141842117	0.388764585	63	121	3
5	0.52613159	A*B	pctImmig.3.	landArea.	0.1218312	0.141842117	0.384289474	62	121	3
6	0.52459827	A*B	pctImmig.8.	landArea.	0.1299003	0.141842117	0.382756157	64	121	3
7	0.60441871	A*B/(A+B)	landArea.	popDensity.	0.1418421	0.226842217	0.377576489	121	122	5
8	0.51410141	A*B	pctImmig.10.	landArea.	0.1316621	0.141842117	0.372259294	65	121	3
9	0.49938746	A*B/(A+B)	pctNotSpeakEng.	landArea.	0.1112602	0.141842117	0.357545344	67	121	5

10	0.4869290 3	A*B/(A+B)	pctImmig.3.	landArea.	0.1218 312	0.14184 2117	0.3450 86916	62	121	5
11	Time	taken	to	run:	410.45	seconds				

crimedc = combinations(dat4,134,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9993572 3	A*B	persUrban.	robbbPerPop.	0.9525 015	0.31990 2076	0.0468 55723	16	135	3
2	0.9993262 2	A*B	pop.	robbbPerPop.	0.9546 475	0.31990 2076	0.0446 78739	6	135	3
3	0.9912761 8	A*B	persPoverty.	robbbPerPop.	0.9598 893	0.31990 2076	0.0313 86867	33	135	3
4	0.9893435	A+B	officDrugU nits.	assaults.	0.9240 217	0.96153 9197	0.0278 04302	118	136	1
5	0.9881178 6	A*B	kidsBornNe vrMarr.	pctSmallHousUnits.	0.9751 072	0.11934 496	0.0130 1071	55	75	3
6	Time	taken	to	run:	391.21	seconds				

crimedd = combinations(dat4,134,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9547424 4	A*B	landArea.	popDensity.	0.1418 421	0.22684 2217	0.7279 00225	121	122	3
2	0.7899504 2	A*B	landArea.	pctUsePubTrans.	0.1418 421	0.33417 3788	0.4557 76633	121	123	3
3	0.5327375 3	A*B	pctNotSpea kEng.	landArea.	0.1112 602	0.14184 2117	0.3908 95414	67	121	3

4	0.5306067	A*B	pctImmig.5.	landArea.	0.1253971	0.14184 2117	0.3887 64585	63	121	3
5	0.5261315 9	A*B	pctImmig.3.	landArea.	0.1218312	0.14184 2117	0.3842 89474	62	121	3
6	Time	taken	to	run:	385.26	seconds				

crimeea = combinations(dat4,135,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.8854260 4	A+B	pctKidsBor nNevrMarr.	autoTheftPerPop.	0.7657187	0.78206 1157	0.1033 64882	56	143	1
2	0.8827102 4	A*B/(A+B)	pctKidsBor nNevrMarr.	autoTheftPerPop.	0.7657187	0.78206 1157	0.1006 49083	56	143	5
3	0.8781021 6	A+B	autoTheftPe rPop.	violentPerPop.	0.7820612	0.78958 9836	0.0885 12321	143	146	1
4	0.8756773	A*B/(A+B)	autoTheftPe rPop.	violentPerPop.	0.7820612	0.78958 9836	0.0860 87465	143	146	5
5	0.8471607 7	A*B	pctKidsBor nNevrMarr.	autoTheftPerPop.	0.7657187	0.78206 1157	0.0650 99614	56	143	3
6	0.8470920 7	A+B	pctKidsBor nNevrMarr.	violentPerPop.	0.7657187	0.78958 9836	0.0575 02232	56	146	1
7	0.8426274 9	A+B	murdPerPop .	autoTheftPerPop.	0.6899341	0.78206 1157	0.0605 66335	131	143	1
8	0.8401316 5	A*B	pctMaleNev Mar.	violentPerPop.	0.358904	0.78958 9836	0.0505 41813	45	146	3
9	0.8324531 3	A*B	autoTheftPe rPop.	violentPerPop.	0.7820612	0.78958 9836	0.0428 63294	143	146	3
10	0.8314431 9	A*B/(A+B)	pctKidsBor nNevrMarr.	violentPerPop.	0.7657187	0.78958 9836	0.0418 53354	56	146	5

11	Time	taken	to	run:	382.53	seconds				

crimeeb = combinations(dat4,135,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7391665	A*B/(A+B)	NAperCap.	robberies.	-0.024616	0.31990 2076	0.4192 64426	29	134	5
2	0.7957756 3	A/B	robberies.	larcenies.	0.3199 021	0.38193 1956	0.4138 43674	134	140	4
3	-0.5934488	A-B	perCapInc.	whitePerCap.	-0.194749	-0.06975 5855	0.3986 99405	26	27	2
4	0.6935444 8	A*B/(A+B)	blackPerCa p.	robberies.	-0.122655	0.31990 2076	0.3736 42406	28	134	5
5	0.6866664 3	A*B/(A+B)	otherPerCap .	robberies.	-0.065046	0.31990 2076	0.3667 64349	31	134	5
6	0.7101005 2	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.024616	0.35474 3488	0.3553 57029	29	55	5
7	0.6850359 7	A*B/(A+B)	blackPerCa p.	kidsBornNevrMarr.	-0.122655	0.35474 3488	0.3302 92477	28	55	5
8	0.7424043 1	A/B	robberies.	burglaries.	0.3199 021	0.41303 4269	0.3293 70037	134	138	4
9	0.6764501 8	A*B/(A+B)	otherPerCap .	kidsBornNevrMarr.	-0.065046	0.35474 3488	0.3217 06693	31	55	5

10	0.6692225 2	A*B/(A+B)	NAperCap.	autoTheft.	-0.024616	0.35547 3959	0.3137 48558	29	142	5
11	Time	taken	to	run:	393.31	seconds				

crimeec = combinations(dat4,135,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.8854260 4	A+B	pctKidsBor nNevrMarr.	autoTheftPerPop.	0.7657 187	0.78206 1157	0.1033 64882	56	143	1
2	0.8827102 4	A*B/(A+B)	pctKidsBor nNevrMarr.	autoTheftPerPop.	0.7657 187	0.78206 1157	0.1006 49083	56	143	5
3	0.8781021 6	A+B	autoTheftPe rPop.	violentPerPop.	0.7820 612	0.78958 9836	0.0885 12321	143	146	1
4	0.8756773	A*B/(A+B)	autoTheftPe rPop.	violentPerPop.	0.7820 612	0.78958 9836	0.0860 87465	143	146	5
5	0.8471607 7	A*B	pctKidsBor nNevrMarr.	autoTheftPerPop.	0.7657 187	0.78206 1157	0.0650 99614	56	143	3
6	Time	taken	to	run:	382.67	seconds				

crimeed = combinations(dat4,135,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7391665	A*B/(A+B)	NAperCap.	robberies.	-0.024616	0.31990 2076	0.4192 64426	29	134	5
2	0.7957756 3	A/B	robberies.	larcenies.	0.3199 021	0.38193 1956	0.4138 43674	134	140	4

3	-0.5934488	A-B	perCapInc.	whitePerCap.	-0.194749	-0.069755855	0.398699405	26	27	2
4	0.69354448	A*B/(A+B)	blackPerCap.	robberies.	-0.122655	0.31990 2076	0.373642406	28	134	5
5	0.68666643	A*B/(A+B)	otherPerCap.	robberies.	-0.065046	0.31990 2076	0.366764349	31	134	5
6	Time	taken	to	run:	396.96	seconds				

crimefa = combinations(dat4,136,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.99741977	A*B	pop.	assaultPerPop.	0.9395694	0.23330 2137	0.05785035	6	137	3
2	0.99738028	A*B	persUrban.	assaultPerPop.	0.9380324	0.23330 2137	0.059347894	16	137	3
3	0.985156	A*B	persPoverty.	assaultPerPop.	0.9415224	0.23330 2137	0.043633579	33	137	3
4	0.97859639	A*B	assaultPerPop.	autoTheft.	0.2333021	0.94130531	0.037291082	137	142	3
5	0.97859107	A*B	robberies.	assaultPerPop.	0.9615392	0.23330 2137	0.017051876	134	137	3
6	0.97811697	A*B/(A+B)	policCarsAvail.	robberies.	0.850788	0.96153 9197	0.016577772	124	134	5
7	0.97723975	A*B	pctWorkMom.6.	robberies.	-0.05411	0.96153 9197	0.015700552	53	134	3

8	0.9749613	A*B	pctWorkMom.18.	robberies.	-0.085999	0.96153 9197	0.0134 22129	54	134	3
9	0.9744504	A*B	pct16.24.	robberies.	0.0162294	0.96153 9197	0.0129 11229	14	134	3
10	0.9738728	A*B/(A+B)	robberies.	larcenies.	0.9615392	0.93004 476	0.0123 33652	134	140	5
11	Time	taken	to	run:	388.23	seconds				

crimefb = combinations(dat4,136,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9396213	A*B	landArea.	popDensity.	0.1833158	0.20619 5253	0.7334 26063	121	122	3
2	0.6865650	A*B/(A+B)	landArea.	popDensity.	0.1833158	0.20619 5253	0.4803 69831	121	122	5
3	0.7620215	A*B	landArea.	pctUsePubTrans.	0.1833158	0.31092 5814	0.4510 95761	121	123	3
4	0.6159797	A*B	pctNotSpeakEng.	landArea.	0.1232657	0.18331 5767	0.4326 63966	67	121	3
5	0.6083525	A*B	pctImmig.5.	landArea.	0.1344816	0.18331 5767	0.4250 36817	63	121	3
6	0.6044550	A*B	pctImmig.3.	landArea.	0.1305628	0.18331 5767	0.4211 39271	62	121	3
7	0.5986185	A*B	pctImmig.8.	landArea.	0.1385872	0.18331 5767	0.4153 02733	64	121	3
8	0.5952230	A*B/(A+B)	pctNotSpeakEng.	landArea.	0.1232657	0.18331 5767	0.4119 07286	67	121	5
9	0.5891595	A*B	pctImmig.10.	landArea.	0.1413311	0.18331 5767	0.4058 43821	65	121	3

10	0.5849677 6	A*B/(A+B)	pctImmig.3.	landArea.	0.1305 628	0.18331 5767	0.4016 51992	62	121	5
11	Time	taken	to	run:	388.1	seconds				

crimefc = combinations(dat4,136,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9974197 7	A*B	pop.	assaultPerPop.	0.9395 694	0.23330 2137	0.0578 5035	6	137	3
2	0.9973802 8	A*B	persUrban.	assaultPerPop.	0.9380 324	0.23330 2137	0.0593 47894	16	137	3
3	0.985156	A*B	persPoverty.	assaultPerPop.	0.9415 224	0.23330 2137	0.0436 33579	33	137	3
4	0.9785963 9	A*B	assaultPerP op.	autoTheft.	0.2333 021	0.94130 531	0.0372 91082	137	142	3
5	0.9785910 7	A*B	robberies.	assaultPerPop.	0.9615 392	0.23330 2137	0.0170 51876	134	137	3
6	Time	taken	to	run:	408.02	seconds				

crimefd = combinations(dat4,136,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9396213 2	A*B	landArea.	popDensity.	0.1833 158	0.20619 5253	0.7334 26063	121	122	3
2	0.6865650 8	A*B/(A+B)	landArea.	popDensity.	0.1833 158	0.20619 5253	0.4803 69831	121	122	5
3	0.7620215 8	A*B	landArea.	pctUsePubTrans.	0.1833 158	0.31092 5814	0.4510 95761	121	123	3

4	0.6159797 3	A*B	pctNotSpeakEng.	landArea.	0.1232 657	0.18331 5767	0.4326 63966	67	121	3
5	0.6083525 8	A*B	pctImmig.5.	landArea.	0.1344 816	0.18331 5767	0.4250 36817	63	121	3
6	Time	taken	to	run:	398.66	seconds				

crimega = combinations(dat4,137,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.8493075 8	A*B	pctSameState.5.	violentPerPop.	0.0186 727	0.85207 9677	-0.0027 72097	103	146	3
2	0.8481966 1	A*B	pctPolicWhite.	violentPerPop.	-0.2134 1	0.85207 9677	-0.0038 8307	113	146	3
3	0.8453486 8	A*B	pctPolicPatrol.	violentPerPop.	-0.0988 2	0.85207 9677	-0.0067 30993	126	146	3
4	0.8449149 7	A*B/(A+B)	pctSameState.5.	violentPerPop.	0.0186 727	0.85207 9677	-0.0071 64708	103	146	5
5	-0.8430336	A-B	officDrugUnits.	violentPerPop.	0.0774 935	0.85207 9677	-0.0090 46094	118	146	2
6	-0.8427894	A-B	persHomeless.	violentPerPop.	0.0796 193	0.85207 9677	-0.0092 90306	98	146	2

7	-0.8426863	A-B	persEmergShelt.	violentPerPop.	0.1229591	0.852079677	-0.009393411	97	146	2
8	0.84188506	A*B/(A+B)	pctPolicWhite.	violentPerPop.	-0.21341	0.852079677	-0.010194613	113	146	5
9	0.84145864	A*B/(A+B)	pctHousOccup.	violentPerPop.	-0.266571	0.852079677	-0.010621042	78	146	5
10	0.841408	A*B	pctHousOccup.	violentPerPop.	-0.266571	0.852079677	-0.010677168	78	146	3
11	Time	taken	to	run:	385.9	seconds				

crimegb = combinations(dat4,137,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
1	0.6947414	A/B	assaults.	larcenies.	0.2333021	0.194796337	0.461439259	136	140	4
2	0.5151063	A-B	persPerFam.	persPerOwnOccup.	-0.1263712	-0.11143958	0.388735094	48	71	2
3	0.62136679	A/B	assaults.	burglaries.	0.2333021	0.209770813	0.388064653	136	138	4
4	0.60560867	A*B/(A+B)	NAperCap.	assaults.	-0.071654	0.233302137	0.372306536	29	136	5
5	0.49301313	A/B	persPerFam.	persPerOwnOccup.	0.1263712	-0.11143958	0.366641925	48	71	4

6	0.5708426	A*B/(A+B)	blackPerCap.	assaults.	-0.207451	0.233302137	0.337540465	28	136	5
7	0.5565503 4	A*B/(A+B)	otherPerCap.	assaults.	-0.106193	0.233302137	0.323248198	31	136	5
8	0.5214799	A*B/(A+B)	policCallPerPop.	assaults.	0.1001694	0.233302137	0.288177776	109	136	5
9	0.5169134 8	A*B/(A+B)	policeFieldPerPop.	assaults.	0.0710086	0.233302137	0.283611344	107	136	5
10	0.5157627 1	A*B/(A+B)	pctWfarm.	assaults.	-0.098367	0.233302137	0.28246057	20	136	5
11	Time	taken	to	run:	381.07	seconds				

crimegc = combinations(dat4,137,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	function Option
1	0.8493075 8	A*B	pctSameState.5.	violentPerPop.	0.0186727	0.852079677	-0.002772097	103	146	3
2	0.8481966 1	A*B	pctPolicWhite.	violentPerPop.	-0.21341	0.852079677	-0.003888307	113	146	3
3	0.8453486 8	A*B	pctPolicPatrol.	violentPerPop.	-0.09882	0.852079677	-0.006730993	126	146	3

4	0.8449149 7	A*B/(A+B)	pctSameStat e.5.	violentPerPop.	0.0186 727	0.85207 9677	-0.0071 64708	103	146	5
5	-0.8430336	A-B	officDrugU nits.	violentPerPop.	0.0774 935	0.85207 9677	-0.0090 46094	118	146	2

crimegd = combinations(dat4,137,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.6947414	A/B	assaults.	larcenies.	0.2333 021	0.19479 6337	0.4614 39259	136	140	4
2	0.5151063	A-B	persPerFam.	persPerOwnOccup.	-0.1263 712	-0.11143 958	0.3887 35094	48	71	2
3	0.6213667 9	A/B	assaults.	burglaries.	0.2333 021	0.20977 0813	0.3880 64653	136	138	4
4	0.6056086 7	A*B/(A+B)	NAperCap.	assaults.	-0.0716 54	0.23330 2137	0.3723 06536	29	136	5
5	0.4930131 3	A/B	persPerFam.	persPerOwnOccup.	-0.1263 712	-0.11143 958	0.3666 41925	48	71	4
6	Time	taken	to	run:	417.1	seconds				

crimeha = combinations(dat4,138,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.9965085 7	A*B	pop.	burglPerPop.	0.9705 316	0.23780 8856	0.0259 76946	6	139	3
2	0.9960947 2	A*B	persUrban.	burglPerPop.	0.9699 414	0.23780 8856	0.0261 5336	16	139	3
3	0.9902152 5	A+B	larcenies.	autoTheft.	0.9841 198	0.97114 0929	0.0060 95482	140	142	1
4	0.9898880 2	A*B	pop.	nonViolPerPop..	0.9705 316	0.21780 5728	0.0193 5639	6	147	3
5	0.9896849 2	A*B	persUrban.	nonViolPerPop..	0.9699 414	0.21780 5728	0.0197 43554	16	147	3
6	0.9879446 3	A+B	persPoverty.	larcenies.	0.9706 786	0.98411 9765	0.0038 24865	33	140	1
7	0.9871245 8	A+B	murders.	larcenies.	0.9561 254	0.98411 9765	0.0030 04812	130	140	1
8	0.9861911 1	A+B	pop.	larcenies.	0.9705 316	0.98411 9765	0.0020 71341	6	140	1
9	0.9860994 3	A*B/(A+B)	larcenies.	autoTheft.	0.9841 198	0.97114 0929	0.0019 79668	140	142	5
10	0.9860202	A+B	persUrban.	larcenies.	0.9699 414	0.98411 9765	0.0019 0044	16	140	1
11	Time	taken	to	run:	393.32	seconds				

crimehb = combinations(dat4,138,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9703030 3	A*B	landArea.	popDensity.	0.2310 12	0.21815 6546	0.7392 91049	121	122	3
2	0.7886932 5	A*B/(A+B)	landArea.	popDensity.	0.2310 12	0.21815 6546	0.5576 81266	121	122	5
3	0.7020215 9	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1369 845	0.23101 198	0.4710 09614	67	121	5

4	0.7820015	A*B	landArea.	pctUsePubTrans.	0.231012	0.32552 7438	0.4564 74111	121	123	3
5	0.6853751	A*B/(A+B)	pctImmig.3.	landArea.	0.1429 382	0.23101 198	0.4543 63125	62	121	5
6	0.6828995	A*B/(A+B)	pctImmig.5.	landArea.	0.1477 212	0.23101 198	0.4518 87567	63	121	5
7	0.6773786 7	A*B/(A+B)	pctImmig.10.	landArea.	0.1578 415	0.23101 198	0.4463 66689	65	121	5
8	0.6767597 2	A*B/(A+B)	pctImmig.8.	landArea.	0.1531 678	0.23101 198	0.4457 4774	64	121	5
9	0.6526555 3	A*B	pctNotSpeakEng.	landArea.	0.1369 845	0.23101 198	0.4216 43553	67	121	3
10	0.6428067 4	A*B/(A+B)	pctForeignBorn.	landArea.	0.1321 005	0.23101 198	0.4117 94757	99	121	5
11	Time	taken	to	run:	387.44	seconds				

crimehc = combinations(dat4,138,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9965085 7	A*B	pop.	burglPerPop.	0.9705 316	0.23780 8856	0.0259 76946	6	139	3
2	0.9960947 2	A*B	persUrban.	burglPerPop.	0.9699 414	0.23780 8856	0.0261 5336	16	139	3
3	0.9902152 5	A+B	larcenies.	autoTheft.	0.9841 198	0.97114 0929	0.0060 95482	140	142	1
4	0.9898880 2	A*B	pop.	nonViolPerPop..	0.9705 316	0.21780 5728	0.0193 5639	6	147	3
5	0.9896849 2	A*B	persUrban.	nonViolPerPop..	0.9699 414	0.21780 5728	0.0197 43554	16	147	3
6	Time	taken	to	run:	407.44	seconds				

crimehd = combinations(dat4,138,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9703033	A*B	landArea.	popDensity.	0.231012	0.218156546	0.739291049	121	122	3
2	0.7886932 5	A*B/(A+B)	landArea.	popDensity.	0.231012	0.218156546	0.557681266	121	122	5
3	0.7020215 9	A*B/(A+B)	pctNotSpeakEng.	landArea.	0.1369845	0.23101198	0.471009614	67	121	5
4	0.7820015 5	A*B	landArea.	pctUsePubTrans.	0.231012	0.325527438	0.456474111	121	123	3
5	0.6853751 1	A*B/(A+B)	pctImmig.3.	landArea.	0.1429382	0.23101198	0.454363125	62	121	5
6	Time	taken	to	run:	424.99	seconds				

crimeia = combinations(dat4,139,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	-0.8245211	A-B	larcPerPop.	nonViolPerPop..	0.6039711	0.793695755	0.030825364	141	147	2
2	0.8134011 8	A*B/(A+B)	medRentpctHousInc.	nonViolPerPop..	0.3490639	0.793695755	0.01970543	94	147	5
3	0.8123426 4	A*B	medRentpctHousInc.	nonViolPerPop..	0.3490639	0.793695755	0.018646889	94	147	3
4	0.8078886 5	A+B	violentPerPop.	nonViolPerPop..	0.6764665	0.793695755	0.014192898	146	147	1
5	0.8078728 9	A+B	assaultPerPop.	nonViolPerPop..	0.621188	0.793695755	0.014177136	137	147	1
6	0.8007449 6	A*B/(A+B)	pctFemDivorc.	nonViolPerPop..	0.5624172	0.793695755	0.007049202	46	147	5

7	0.7986833 5	A+B	robbbPerPo p.	nonViolPerPop..	0.6446 541	0.79369 5755	0.0049 87593	135	147	1
8	0.7968958 3	A+B	pctKidsBor nNevrMarr.	nonViolPerPop..	0.6199 263	0.79369 5755	0.0032 00077	56	147	1
9	0.7963363 7	A*B/(A+B)	pctAllDivor c.	nonViolPerPop..	0.5626 962	0.79369 5755	0.0026 40613	47	147	5
10	-0.7957627	A-B	pctKids2Par .	nonViolPerPop..	-0.6874 99	0.79369 5755	0.0020 66907	50	147	2
11	Time	taken	to	run:	389.98	seconds				

crimeib = combinations(dat4,139,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.5782628 2	A*B/(A+B)	NAperCap.	burglaries.	-0.0670 81	0.23780 8856	0.3404 53963	29	138	5
2	0.4847720 5	A*B/(A+B)	NAperCap.	assaults.	-0.0670 81	0.15346 0299	0.3313 11747	29	136	5
3	0.5421189 2	A-B	persPerFam.	persPerOwnOccup.	0.0381 939	-0.21100 1705	0.3311 17217	48	71	2
4	-0.4632519	(A+B)/(A*B)	pctUrban.	murders.	0.0148 449	0.13507 6901	0.3281 7502	17	130	6
5	0.4587961 8	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.0670 81	0.13422 9516	0.3245 6666	29	55	5
6	-0.4531071	(A+B)/(A*B)	fold.	murders.	-0.0240 27	0.13507 6901	0.3180 30177	5	130	6

7	0.4093411	A/B	NAperCap.	pctPolicAsian.	-0.067081	0.094875315	0.314465786	29	116	4
8	0.41665277	A*B/(A+B)	NAperCap.	robberies.	-0.067081	0.103745792	0.312906975	29	134	5
9	0.45843183	A*B/(A+B)	NAperCap.	persPoverty.	-0.067081	0.147124157	0.311307669	29	33	5
10	0.51619339	A/B	persPerFam.	persPerOwnOccup.	-0.0381939	-0.211001705	0.305191681	48	71	4
11	Time	taken	to	run:	385.42	seconds				

crimeic = combinations(dat4,139,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	-0.8245211	A-B	larcPerPop.	nonViolPerPop..	0.6039711	0.793695755	0.030825364	141	147	2
2	0.81340118	A*B/(A+B)	medRentpctHousInc.	nonViolPerPop..	0.3490639	0.793695755	0.019700543	94	147	5
3	0.81234264	A*B	medRentpctHousInc.	nonViolPerPop..	0.3490639	0.793695755	0.018646889	94	147	3
4	0.80788865	A+B	violentPerPop.	nonViolPerPop..	0.6764665	0.793695755	0.014192898	146	147	1
5	0.80787289	A+B	assaultPerPop.	nonViolPerPop..	0.621188	0.793695755	0.014177136	137	147	1
6	Time	taken	to	run:	385.13	seconds				

crimeid = combinations(dat4,139,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.5782628 2	A*B/(A+B)	NAperCap.	burglaries.	-0.067081	0.23780 8856	0.3404 53963	29	138	5
2	0.4847720 5	A*B/(A+B)	NAperCap.	assaults.	-0.067081	0.15346 0299	0.3313 11747	29	136	5
3	0.5421189 2	A-B	persPerFam.	persPerOwnOccup.	0.0381939	-0.21100 1705	0.3311 17217	48	71	2
4	-0.4632519	(A+B)/(A*B)	pctUrban.	murders.	0.0148449	0.13507 6901	0.3281 7502	17	130	6
5	0.4587961 8	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.067081	0.13422 9516	0.3245 6666	29	55	5
6	Time	taken	to	run:	384.38	seconds				

crimeja = combinations(dat4,140,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9968374 1	A*B	pop.	larcPerPop.	0.9659872	0.17422 4127	0.0308 50162	6	141	3
2	0.9959872 4	A*B	persUrban.	larcPerPop.	0.965524	0.17422 4127	0.0304 6321	16	141	3
3	0.9879076 3	A*B	persUrban.	nonViolPerPop..	0.965524	0.23013 3204	0.0223 83607	16	147	3
4	0.9878299 3	A*B	pop.	nonViolPerPop..	0.9659872	0.23013 3204	0.0218 42679	6	147	3

5	0.9839346 3	A*B	pctWwage.	burglaries.	-0.033084	0.98411 9765	-0.0001 85136	19	138	3
6	0.9835675 7	A*B	pctUrban.	burglaries.	0.1255 524	0.98411 9765	-0.0005 52193	17	138	3
7	0.9828114 3	A+B	pop.	burglaries.	0.9659 872	0.98411 9765	-0.0013 08334	6	138	1
8	0.9827234 1	A+B	persUrban.	burglaries.	0.9655 24	0.98411 9765	-0.0013 96354	16	138	1
9	0.9822046 2	A*B	pctHousOcc up.	burglaries.	-0.070939	0.98411 9765	-0.0019 1515	78	138	3
10	0.9815907 6	A*B/(A+B)	houseVacan t.	burglaries.	0.9341 76	0.98411 9765	-0.0025 29003	77	138	5
11	Time	taken	to	run:	385.12	seconds				

crimejb = combinations(dat4,140,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9658708 1	A*B	landArea.	popDensity.	0.2398 782	0.20377 2512	0.7259 92621	121	122	3
2	0.8077467	A*B/(A+B)	landArea.	popDensity.	0.2398 782	0.20377 2512	0.5678 68508	121	122	5
3	0.7351305 5	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1287 912	0.23987 8188	0.4952 52359	67	121	5
4	0.7148910 6	A*B/(A+B)	pctImmig.3.	landArea.	0.1386 702	0.23987 8188	0.4750 12875	62	121	5

5	0.7129796 4	A*B/(A+B)	pctImmig.5.	landArea.	0.1417 332	0.23987 8188	0.4731 01449	63	121	5
6	0.7077684 3	A*B/(A+B)	pctImmig.1 0.	landArea.	0.1509 731	0.23987 8188	0.4678 90244	65	121	5
7	0.7070249	A*B/(A+B)	pctImmig.8.	landArea.	0.1467 583	0.23987 8188	0.4671 46717	64	121	5
8	0.7791256	A*B	landArea.	pctUsePubTrans.	0.2398 782	0.31986 8841	0.4592 56763	121	123	3
9	0.6779247 5	A*B/(A+B)	pctHisp.	landArea.	0.1079 265	0.23987 8188	0.4380 46559	11	121	5
10	0.6726595 8	A*B/(A+B)	pctForeignB orn.	landArea.	0.1254 297	0.23987 8188	0.4327 81397	99	121	5
11	Time	taken	to	run:	401.88	seconds				

crimejc = combinations(dat4,140,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9968374 1	A*B	pop.	larcPerPop.	0.9659 872	0.17422 4127	0.0308 50162	6	141	3
2	0.9959872 4	A*B	persUrban.	larcPerPop.	0.9655 24	0.17422 4127	0.0304 6321	16	141	3
3	0.9879076 3	A*B	persUrban.	nonViolPerPop..	0.9655 24	0.23013 3204	0.0223 83607	16	147	3
4	0.9878299 3	A*B	pop.	nonViolPerPop..	0.9659 872	0.23013 3204	0.0218 42679	6	147	3
5	0.9839346 3	A*B	pctWwage.	burglaries.	-0.0330 84	0.98411 9765	-0.0001 85136	19	138	3
6	Time	taken	to	run:	390.98	seconds				

crimejd = combinations(dat4,140,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9658708 1	A*B	landArea.	popDensity.	0.2398 782	0.20377 2512	0.7259 92621	121	122	3
2	0.8077467	A*B/(A+B)	landArea.	popDensity.	0.2398 782	0.20377 2512	0.5678 68508	121	122	5
3	0.7351305 5	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1287 912	0.23987 8188	0.4952 52359	67	121	5
4	0.7148910 6	A*B/(A+B)	pctImmig.3.	landArea.	0.1386 702	0.23987 8188	0.4750 12875	62	121	5
5	0.7129796 4	A*B/(A+B)	pctImmig.5.	landArea.	0.1417 332	0.23987 8188	0.4731 01449	63	121	5
6	Time	taken	to	run:	385.34	seconds				

crimeka = combinations(dat4,141,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9220750 6	A*B	pctPolicPatr ol.	nonViolPerPop..	-0.0474 97	0.92631 2925	-0.0042 37861	126	147	3
2	0.9202575 3	A*B	pctPolicWhi te.	nonViolPerPop..	-0.0915 15	0.92631 2925	-0.0060 55393	113	147	3
3	0.9089223 4	A*B	racialMatch.	nonViolPerPop..	-0.0669 18	0.92631 2925	-0.0173 9058	112	147	3

4	-0.9088486	A-B	officDrugUnits.	nonViolPerPop..	0.0288872	0.926312925	-0.017464288	118	147	2
5	-0.9080939	A-B	persHomeless.	nonViolPerPop..	0.0256668	0.926312925	-0.018219055	98	147	2
6	0.90772608	A*B/(A+B)	pctPolicPatrol.	nonViolPerPop..	-0.047497	0.926312925	-0.018586842	126	147	5
7	-0.9071461	A-B	persEmergShelt.	nonViolPerPop..	0.0645948	0.926312925	-0.019166808	97	147	2
8	0.90658018	A*B	pctHousOccup.	nonViolPerPop..	-0.262977	0.926312925	-0.019732744	78	147	3
9	0.90606965	A*B/(A+B)	pctPolicWhite.	nonViolPerPop..	-0.091515	0.926312925	-0.020243271	113	147	5
10	-0.9059352	A-B	robberies.	nonViolPerPop..	0.0522093	0.926312925	-0.020377689	134	147	2
11	Time	taken	to	run:	385.78	seconds				

crimekb = combinations(dat4,141,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
1	-0.4954484	A/B	medOwnCostpct.	larcenies.	-0.176043	0.174224127	0.319405631	95	140	4

2	0.35484322	A-B	pctLargHousFam.	pctLargHous.	0.0578885	-0.00577 4659	0.2969 54752	68	69	2
3	-0.3644151	(A+B)/(A*B)	pctUrban.	robberies.	-0.068776	0.05220 9252	0.2956 38928	17	134	6
4	-0.3448989	(A+B)/(A*B)	pctImmig.3.	robberies.	-0.023815	0.05220 9252	0.2926 89609	62	134	6
5	-0.359544	(A+B)/(A*B)	kidsBornNevrMarr.	robberies.	0.0669035	0.05220 9252	0.2926 40496	55	134	6
6	-0.3438313	(A+B)/(A*B)	medYrHousBuilt.	robberies.	-0.026893	0.05220 9252	0.2916 22053	82	134	6
7	-0.3437048	(A+B)/(A*B)	fold.	robberies.	-0.026567	0.05220 9252	0.2914 9553	5	134	6
8	-0.3435757	(A+B)/(A*B)	landArea.	robberies.	0.0316547	0.05220 9252	0.2913 664	121	134	6
9	-0.3421583	(A+B)/(A*B)	countyCode.	robberies.	0.0351613	0.05220 9252	0.2899 49063	3	134	6
10	-0.3385347	(A+B)/(A*B)	medOwnCostPctWO.	robberies.	-0.016077	0.05220 9252	0.2863 25471	96	134	6
11	Time	taken	to	run:	403.65	seconds				

crimekc = combinations(dat4,141,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.9220750 6	A*B	pctPolicPatrol.	nonViolPerPop..	-0.047497	0.92631 2925	-0.0042 37861	126	147	3
2	0.9202575 3	A*B	pctPolicWhite.	nonViolPerPop..	-0.091515	0.92631 2925	-0.0060 55393	113	147	3
3	0.9089223 4	A*B	racialMatch.	nonViolPerPop..	-0.066918	0.92631 2925	-0.0173 9058	112	147	3
4	-0.9088486	A-B	officDrugUnits.	nonViolPerPop..	0.0288872	0.92631 2925	-0.0174 64288	118	147	2
5	-0.9080939	A-B	persHomeless.	nonViolPerPop..	0.0256668	0.92631 2925	-0.0182 19055	98	147	2
6	Time	taken	to	run:	387.24	seconds				

crimekd = combinations(dat4,141,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	-0.4954484	A/B	medOwnCostpct.	larcenies.	-0.176043	0.17422 4127	0.3194 05631	95	140	4
2	0.3548432 2	A-B	pctLargHousFam.	pctLargHous.	0.0578885	-0.00577 4659	0.2969 54752	68	69	2
3	-0.3644151	(A+B)/(A*B)	pctUrban.	robberies.	-0.068776	0.05220 9252	0.2956 38928	17	134	6

4	-0.3448989	(A+B)/(A*B)	pctImmig.3.	robberies.	-0.023815	0.052209252	0.292689609	62	134	6
5	-0.359544	(A+B)/(A*B)	kidsBornNevrMarr.	robberies.	0.0669035	0.052209252	0.292640496	55	134	6
6	Time	taken	to	run:	383.53	seconds				

crimela = combinations(dat4,142,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.99849432	A*B	persUrban.	autoTheftPerPop.	0.9794559	0.307187029	0.01903843	16	143	3
2	0.99841585	A*B	pop.	autoTheftPerPop.	0.9807139	0.307187029	0.017701936	6	143	3
3	0.9866033	A*B/(A+B)	pop.	murders.	0.9807139	0.970705928	0.005889388	6	130	5
4	0.98658049	A*B/(A+B)	persUrban.	murders.	0.9794559	0.970705928	0.0071246	16	130	5
5	0.98597726	A/B	persPoverty.	pctSameState.5.	0.98385	-0.011421121	0.00212727	33	103	4
6	0.98577805	A*B/(A+B)	persPoverty.	burglaries.	0.98385	0.971140929	0.001928066	33	138	5
7	0.98573672	A+B	pop.	persPoverty.	0.9807139	0.983849986	0.001886734	6	33	1
8	0.98561549	A+B	pop.	murders.	0.9807139	0.970705928	0.004901572	6	130	1
9	0.9856032	A*B/(A+B)	pop.	persPoverty.	0.9807139	0.983849986	0.001753218	6	33	5
10	0.98558685	A+B	persUrban.	persPoverty.	0.9794559	0.983849986	0.001736862	16	33	1

11	Time	taken	to	run:	395.99	seconds				

crimelb = combinations(dat4,142,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9805328 1	A*B	landArea.	popDensity.	0.1952 401	0.23375 1838	0.7467 80973	121	122	3
2	0.7138423 2	A*B/(A+B)	landArea.	popDensity.	0.1952 401	0.23375 1838	0.4800 9048	121	122	5
3	0.6560390 1	A*B	pctNotSpea kEng.	landArea.	0.1436 234	0.19524 0086	0.4607 98924	67	121	3
4	0.7920421 8	A*B	landArea.	pctUsePubTrans.	0.1952 401	0.33563 3908	0.4564 08269	121	123	3
5	0.6448840 8	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1436 234	0.19524 0086	0.4496 43996	67	121	5
6	0.6354174 6	A*B	pctImmig.5.	landArea.	0.1592 682	0.19524 0086	0.4401 77376	63	121	3
7	0.6301275 1	A*B	pctImmig.3.	landArea.	0.1547 403	0.19524 0086	0.4348 87426	62	121	3
8	0.6293108 9	A*B	pctImmig.8.	landArea.	0.1648 284	0.19524 0086	0.4340 70809	64	121	3
9	0.6272137 4	A*B	pctImmig.1 0.	landArea.	0.1692 017	0.19524 0086	0.4319 73653	65	121	3
10	0.6234492 6	A*B/(A+B)	pctImmig.3.	landArea.	0.1547 403	0.19524 0086	0.4282 09178	62	121	5
11	Time	taken	to	run:	388.85	seconds				

crimelc = combinations(dat4,142,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9984943 2	A*B	persUrban.	autoTheftPerPop.	0.9794 559	0.30718 7029	0.0190 3843	16	143	3
2	0.9984158 5	A*B	pop.	autoTheftPerPop.	0.9807 139	0.30718 7029	0.0177 01936	6	143	3
3	0.9866033	A*B/(A+B)	pop.	murders.	0.9807 139	0.97070 5928	0.0058 89388	6	130	5
4	0.9865804 9	A*B/(A+B)	persUrban.	murders.	0.9794 559	0.97070 5928	0.0071 246	16	130	5
5	0.9859772 6	A/B	persPoverty.	pctSameState.5.	0.9838 5	-0.01142 1121	0.0021 2727	33	103	4
6	Time	taken	to	run:	384.6	seconds				

crimeld = combinations(dat4,142,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
	correlation	function.	attribute.A	attribute.B	cor.A.ta rget.	cor.B.tar get.	gain	index A	index B	function Option
1	0.9805328 1	A*B	landArea.	popDensity.	0.1952 401	0.23375 1838	0.7467 80973	121	122	3
2	0.7138423 2	A*B/(A+B)	landArea.	popDensity.	0.1952 401	0.23375 1838	0.4800 9048	121	122	5
3	0.6560390 1	A*B	pctNotSpea kEng.	landArea.	0.1436 234	0.19524 0086	0.4607 98924	67	121	3
4	0.7920421 8	A*B	landArea.	pctUsePubTrans.	0.1952 401	0.33563 3908	0.4564 08269	121	123	3
5	0.6448840 8	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1436 234	0.19524 0086	0.4496 43996	67	121	5

crimema = combinations(dat4,143,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7991589 4	A*B	pctKids.4w2 Par.	robbbPerPop.	-0.3872 15	0.78206 1157	0.0170 97779	51	135	3
2	0.7912412 9	A*B/(A+B)	pctSmallHo usUnits.	robbbPerPop.	0.4252 852	0.78206 1157	0.0091 80131	75	135	5
3	0.7902986 9	A*B/(A+B)	pctHousOcc up.	robbbPerPop.	-0.1147	0.78206 1157	0.0082 37533	78	135	5
4	0.7900386 2	A*B/(A+B)	pctSameCo unty.5.	robbbPerPop.	0.1140 408	0.78206 1157	0.0079 77464	102	135	5
5	0.7894387 1	A*B/(A+B)	medRentpct HousInc.	robbbPerPop.	0.2725 907	0.78206 1157	0.0073 77554	94	135	5
6	0.7894065 7	A*B/(A+B)	pctSameStat e.5.	robbbPerPop.	-0.0069 28	0.78206 1157	0.0073 45415	103	135	5
7	0.7887821 4	A*B/(A+B)	persPerFam.	robbbPerPop.	0.2127 247	0.78206 1157	0.0067 20985	48	135	5
8	0.7887561 5	A*B/(A+B)	pctWwage.	robbbPerPop.	-0.1271 56	0.78206 1157	0.0066 94991	19	135	5
9	0.7881734	A*B/(A+B)	medOwnCo stpct.	robbbPerPop.	0.2071 401	0.78206 1157	0.0061 12241	95	135	5
10	0.7880076 5	A*B/(A+B)	persPerOwn Occup.	robbbPerPop.	0.0574 38	0.78206 1157	0.0059 46498	71	135	5
11	Time	taken	to	run:	399.44	seconds				

crimemb = combinations(dat4,143,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.7074354 8	A*B/(A+B)	NAperCap.	autoTheft.	-0.017821	0.30718 7029	0.4002 48451	29	142	5
2	0.6039732 4	A*B/(A+B)	NAperCap.	robberies.	-0.017821	0.21297 2009	0.3910 01231	29	134	5
3	0.6656640 5	A*B/(A+B)	blackPerCa p.	autoTheft.	-0.061098	0.30718 7029	0.3584 77026	28	142	5
4	0.6529970 2	A*B/(A+B)	otherPerCap .	autoTheft.	-0.036364	0.30718 7029	0.3458 09992	31	142	5
5	0.5559403 6	A*B/(A+B)	blackPerCa p.	robberies.	-0.061098	0.21297 2009	0.3429 6835	28	134	5
6	0.6457127 1	A/B	robberies.	larcenies.	0.212972	0.30309 9763	0.3426 12946	134	140	4
7	0.5107476	A*B/(A+B)	NAperCap.	numForeignBorn.	-0.017821	0.17060 7705	0.3401 39893	29	57	5
8	0.5445464 5	A*B/(A+B)	otherPerCap .	robberies.	-0.036364	0.21297 2009	0.3315 74437	31	134	5
9	0.5692422 7	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.017821	0.24789 9088	0.3213 43177	29	55	5
10	0.4856526 9	A*B/(A+B)	countyCode.	numForeignBorn.	-0.040121	0.17060 7705	0.3150 44986	3	57	5
11	Time	taken	to	run:	384.92	seconds				

crimemc = combinations(dat4,143,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7991589 4	A*B	pctKids.4w2 Par.	robbbPerPop.	-0.3872 15	0.78206 1157	0.0170 97779	51	135	3
2	0.7912412 9	A*B/(A+B)	pctSmallHo usUnits.	robbbPerPop.	0.4252 852	0.78206 1157	0.0091 80131	75	135	5
3	0.7902986 9	A*B/(A+B)	pctHousOcc up.	robbbPerPop.	-0.1147	0.78206 1157	0.0082 37533	78	135	5
4	0.7900386 2	A*B/(A+B)	pctSameCo unty.5.	robbbPerPop.	0.1140 408	0.78206 1157	0.0079 77464	102	135	5
5	0.7894387 1	A*B/(A+B)	medRentpct HousInc.	robbbPerPop.	0.2725 907	0.78206 1157	0.0073 77554	94	135	5
6	Time	taken	to	run:	380.94	seconds				

crimemd = combinations(dat4,143,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.7074354 8	A*B/(A+B)	NAperCap.	autoTheft.	-0.0178 21	0.30718 7029	0.4002 48451	29	142	5
2	0.6039732 4	A*B/(A+B)	NAperCap.	robberies.	-0.0178 21	0.21297 2009	0.3910 01231	29	134	5
3	0.6656640 5	A*B/(A+B)	blackPerCa p.	autoTheft.	-0.0610 98	0.30718 7029	0.3584 77026	28	142	5
4	0.6529970 2	A*B/(A+B)	otherPerCap .	autoTheft.	-0.0363 64	0.30718 7029	0.3458 09992	31	142	5

5	0.5559403 6	A*B/(A+B)	blackPerCa p.	robberies.	- 0.0610 98	0.21297 2009	0.3429 6835	28	134	5
6	Time	taken	to	run:	391.87	seconds				

crimena = combinations(dat4,144,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9978609 2	A*B	pop.	arsonsPerPop.	0.8994 393	0.28394 4536	0.0984 21636	6	145	3
2	0.9976195 5	A*B	persUrban.	arsonsPerPop.	0.8982 302	0.28394 4536	0.0993 89365	16	145	3
3	0.9854053	A*B	persPoverty.	arsonsPerPop.	0.9013 77	0.28394 4536	0.0840 28308	33	145	3
4	0.9791152 6	A*B	autoTheft.	arsonsPerPop.	0.9094 761	0.28394 4536	0.0696 39158	142	145	3
5	0.9712477 5	A*B	larcenies.	arsonsPerPop.	0.8765 17	0.28394 4536	0.0947 30778	140	145	3
6	0.9700685 7	A*B	burglaries.	arsonsPerPop.	0.8882 271	0.28394 4536	0.0818 41472	138	145	3
7	0.9617796 4	A*B	murders.	arsonsPerPop.	0.8966 685	0.28394 4536	0.0651 11096	130	145	3
8	0.9533589 1	A*B	kidsBornNe vrMarr.	arsonsPerPop.	0.8659 61	0.28394 4536	0.0873 97914	55	145	3
9	0.9532157 1	A*B	robberies.	arsonsPerPop.	0.8500 005	0.28394 4536	0.1032 15199	134	145	3
10	0.9524472 1	A*B	numPolice.	arsonsPerPop.	0.8298 4	0.28394 4536	0.1226 07195	104	145	3
11	Time	taken	to	run:	405.8	seconds				

crimenb = combinations(dat4,144,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.8994552 1	A*B	landArea.	popDensity.	0.2061 309	0.18620 5973	0.6933 24352	121	122	3
2	0.7441623 8	A*B	pctNotSpea kEng.	landArea.	0.1305 464	0.20613 0856	0.5380 31521	67	121	3
3	0.7229023 4	A*B/(A+B)	landArea.	popDensity.	0.2061 309	0.18620 5973	0.5167 71481	121	122	5
4	0.7111790 6	A*B	pctImmig.5.	landArea.	0.1398 002	0.20613 0856	0.5050 48204	63	121	3
5	0.7087582 4	A*B	pctImmig.3.	landArea.	0.1365 175	0.20613 0856	0.5026 27387	62	121	3
6	0.7003695 2	A*B	pctImmig.1 0.	landArea.	0.1479 556	0.20613 0856	0.4942 38665	65	121	3
7	0.6998923	A*B	pctImmig.8.	landArea.	0.1441 103	0.20613 0856	0.4937 61449	64	121	3
8	0.6814649 7	A*B/(A+B)	pctNotSpea kEng.	landArea.	0.1305 464	0.20613 0856	0.4753 34118	67	121	5
9	0.6550942	A*B/(A+B)	pctImmig.3.	landArea.	0.1365 175	0.20613 0856	0.4489 63341	62	121	5
10	0.6509030 3	A*B/(A+B)	pctImmig.5.	landArea.	0.1398 002	0.20613 0856	0.4447 72171	63	121	5
11	Time	taken	to	run:	446.57	seconds				

crimenc = combinations(dat4,144,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9978609 2	A*B	pop.	arsonsPerPop.	0.8994 393	0.28394 4536	0.0984 21636	6	145	3

2	0.9976195	A*B	persUrban.	arsonsPerPop.	0.8982302	0.28394 4536	0.0993 89365	16	145	3
3	0.9854053	A*B	persPoverty.	arsonsPerPop.	0.9013 77	0.28394 4536	0.0840 28308	33	145	3
4	0.9791152 6	A*B	autoTheft.	arsonsPerPop.	0.9094 761	0.28394 4536	0.0696 39158	142	145	3
5	0.9712477 5	A*B	larcenies.	arsonsPerPop.	0.8765 17	0.28394 4536	0.0947 30778	140	145	3
6	Time	taken	to	run:	395.03	seconds				

crimend = combinations(dat4,144,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.8994552 1	A*B	landArea.	popDensity.	0.2061 309	0.18620 5973	0.6933 24352	121	122	3
2	0.7441623 8	A*B	pctNotSpea kEng.	landArea.	0.1305 464	0.20613 0856	0.5380 31521	67	121	3
3	0.7229023 4	A*B/(A+B)	landArea.	popDensity.	0.2061 309	0.18620 5973	0.5167 71481	121	122	5
4	0.7111790 6	A*B	pctImmig.5.	landArea.	0.1398 002	0.20613 0856	0.5050 48204	63	121	3
5	0.7087582 4	A*B	pctImmig.3.	landArea.	0.1365 175	0.20613 0856	0.5026 27387	62	121	3
6	Time	taken	to	run:	388.32	seconds				

crimeoa = combinations(dat4,145,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.6604025 6	A*B/(A+B)	NAperCap.	arsons.	-0.0240 1	0.28394 4536	0.3764 58026	29	144	5
2	0.6154809 2	A*B/(A+B)	blackPerCa p.	arsons.	-0.1228 83	0.28394 4536	0.3315 36381	28	144	5
3	0.599509	A*B/(A+B)	otherPerCap .	arsons.	-0.0532 47	0.28394 4536	0.3155 64467	31	144	5
4	0.5654568 4	A*B/(A+B)	policeFieldP erPop.	arsons.	0.0131 372	0.28394 4536	0.2815 12308	107	144	5
5	0.5604302 4	A*B/(A+B)	policePerPo p.	arsons.	0.0140 328	0.28394 4536	0.2764 85703	105	144	5
6	0.5604216 8	A*B/(A+B)	policePerPo p2.	arsons.	0.0140 406	0.28394 4536	0.2764 77143	111	144	5
7	0.5547229 9	A*B/(A+B)	policCallPer Pop.	arsons.	0.0655 986	0.28394 4536	0.2707 78454	109	144	5
8	0.5490164 7	A*B/(A+B)	policBudget PerPop.	arsons.	0.0198 085	0.28394 4536	0.2650 71932	129	144	5
9	0.5468371 3	A*B/(A+B)	burglPerPop .	arsons.	0.4096 327	0.28394 4536	0.1372 04434	139	144	5
10	0.5379125 5	A*B/(A+B)	pctWfarm.	arsons.	-0.1009 69	0.28394 4536	0.2539 68015	20	144	5
11	Time	taken	to	run:	384.19	seconds				

crimeob = combinations(dat4,145,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.6604025 6	A*B/(A+B)	NAperCap.	arsons.	-0.0240 1	0.28394 4536	0.3764 58026	29	144	5
2	0.6154809 2	A*B/(A+B)	blackPerCa p.	arsons.	-0.1228 83	0.28394 4536	0.3315 36381	28	144	5
3	0.599509	A*B/(A+B)	otherPerCap .	arsons.	-0.0532 47	0.28394 4536	0.3155 64467	31	144	5
4	0.5654568 4	A*B/(A+B)	policeFieldP erPop.	arsons.	0.0131 372	0.28394 4536	0.2815 12308	107	144	5
5	0.5604302 4	A*B/(A+B)	policePerPo p.	arsons.	0.0140 328	0.28394 4536	0.2764 85703	105	144	5
6	0.5604216 8	A*B/(A+B)	policePerPo p2.	arsons.	0.0140 406	0.28394 4536	0.2764 77143	111	144	5
7	0.5547229 9	A*B/(A+B)	policCallPer Pop.	arsons.	0.0655 986	0.28394 4536	0.2707 78454	109	144	5
8	0.5490164 7	A*B/(A+B)	policBudget PerPop.	arsons.	0.0198 085	0.28394 4536	0.2650 71932	129	144	5
9	0.5379125 5	A*B/(A+B)	pctWfarm.	arsons.	-0.1009 69	0.28394 4536	0.2539 68015	20	144	5
10	0.3761918 2	A*B/(A+B)	NAperCap.	robberies.	-0.0240 1	0.12437 5798	0.2518 16023	29	134	5
11	Time	taken	to	run:	407.23	seconds				

crimeoc = combinations(dat4,145,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option

1	0.66040256	A*B/(A+B)	NAperCap.	arsons.	-0.024001	0.28394536	0.376458026	29	144	5
2	0.61548092	A*B/(A+B)	blackPerCap.	arsons.	-0.122883	0.28394536	0.331536381	28	144	5
3	0.599509	A*B/(A+B)	otherPerCap.	arsons.	-0.053247	0.28394536	0.315564467	31	144	5
4	0.56545684	A*B/(A+B)	policeFieldPerPop.	arsons.	0.0131372	0.28394536	0.281512308	107	144	5
5	0.56043024	A*B/(A+B)	policePerPop.	arsons.	0.0140328	0.28394536	0.276485703	105	144	5
6	Time	taken	to	run:	389.84	seconds				

crimeod = combinations(dat4,145,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
1	0.66040256	A*B/(A+B)	NAperCap.	arsons.	-0.024001	0.28394536	0.376458026	29	144	5
2	0.61548092	A*B/(A+B)	blackPerCap.	arsons.	-0.122883	0.28394536	0.331536381	28	144	5
3	0.599509	A*B/(A+B)	otherPerCap.	arsons.	-0.053247	0.28394536	0.315564467	31	144	5
4	0.56545684	A*B/(A+B)	policeFieldPerPop.	arsons.	0.0131372	0.28394536	0.281512308	107	144	5
5	0.56043024	A*B/(A+B)	policePerPop.	arsons.	0.0140328	0.28394536	0.276485703	105	144	5

6	Time	taken	to	run:	386.42	seconds				

crimepa = combinations(dat4,146,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.91166424	A+B	robbbPerPop.	assaultPerPop.	0.7895898	0.852079677	0.059584561	135	137	1
2	0.88887104	A*B/(A+B)	rapesPerPop.	assaultPerPop.	0.5831458	0.852079677	0.036791363	133	137	5
3	0.88794828	A*B/(A+B)	pctSmallHousUnits.	assaultPerPop.	0.4445678	0.852079677	0.0358686	75	137	5
4	0.88730803	A*B/(A+B)	pctEmploy.	assaultPerPop.	-0.301905	0.852079677	0.035228357	39	137	5
5	0.88629656	A*B/(A+B)	pctWwage.	assaultPerPop.	-0.279318	0.852079677	0.034216881	19	137	5
6	0.88548571	A*B/(A+B)	pctWorkMom.6.	assaultPerPop.	-0.024673	0.852079677	0.033406037	53	137	5
7	0.8843189	A*B/(A+B)	pctFgnImmig.10.	assaultPerPop.	0.2780465	0.852079677	0.032239219	61	137	5
8	0.88411821	A*B/(A+B)	assaultPerPop.	nonViolPerPop..	0.8520797	0.627123516	0.032038531	137	147	5
9	0.88203665	A*B/(A+B)	robbbPerPop.	assaultPerPop.	0.7895898	0.852079677	0.029956968	135	137	5
10	0.88138961	A*B/(A+B)	pctWorkMom.18.	assaultPerPop.	-0.138837	0.852079677	0.029300993	54	137	5

11	Time	taken	to	run:	381.64	seconds				

crimepb = combinations(dat4,146,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.6840014 9	A*B/(A+B)	NAperCap.	assaults.	-0.0581 3	0.26640 0421	0.4176 01071	29	136	5
2	0.5365860 4	A-B	persPerFam.	persPerOwnOccup.	0.1447 522	-0.10312 5449	0.3918 33859	48	71	2
3	0.5196082 7	A/B	persPerFam.	persPerOwnOccup.	0.1447 522	-0.10312 5449	0.3748 56089	48	71	4
4	0.6359716 8	A*B/(A+B)	blackPerCa p.	assaults.	-0.1996 67	0.26640 0421	0.3695 71256	28	136	5
5	0.5620428 9	A*B/(A+B)	NAperCap.	robberies.	-0.0581 3	0.19303 1468	0.3690 11424	29	134	5
6	0.5710826 8	A*B/(A+B)	NAperCap.	kidsBornNevrMarr.	-0.0581 3	0.21404 7608	0.3570 35076	29	55	5
7	0.6228107	A*B/(A+B)	otherPerCap .	assaults.	-0.1007 44	0.26640 0421	0.3564 10284	31	136	5
8	0.5894555 8	A/B	robberies.	larcenies.	0.1930 315	0.26767 3924	0.3217 81655	134	140	4
9	-0.6203491	A-B	perCapInc.	whitePerCap.	-0.2990 19	-0.17602 1214	0.3213 30095	26	27	2

10	0.5857801 2	A*B/(A+B)	pctWfarm.	assaults.	-0.1368 37	0.26640 0421	0.3193 79695	20	136	5
11	Time	taken	to	run:	400.46	seconds				

crimepc = combinations(dat4,146,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.9116642 4	A+B	robbbPerPo p.	assaultPerPop.	0.7895 898	0.85207 9677	0.0595 84561	135	137	1
2	0.8888710 4	A*B/(A+B)	rapesPerPop .	assaultPerPop.	0.5831 458	0.85207 9677	0.0367 91363	133	137	5
3	0.8879482 8	A*B/(A+B)	pctSmallHo usUnits.	assaultPerPop.	0.4445 678	0.85207 9677	0.0358 686	75	137	5
4	0.8873080 3	A*B/(A+B)	pctEmploy.	assaultPerPop.	-0.3019 05	0.85207 9677	0.0352 28357	39	137	5
5	0.8862965 6	A*B/(A+B)	pctWwage.	assaultPerPop.	-0.2793 18	0.85207 9677	0.0342 16881	19	137	5
6	Time	taken	to	run:	386.88	seconds				

crimepd = combinations(dat4,146,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.6840014 9	A*B/(A+B)	NAperCap.	assaults.	-0.0581 3	0.26640 0421	0.4176 01071	29	136	5

2	0.53658604	A-B	persPerFam.	persPerOwnOccup.	0.1447522	-0.103125449	0.391833859	48	71	2
3	0.51960827	A/B	persPerFam.	persPerOwnOccup.	0.1447522	-0.103125449	0.374856089	48	71	4
4	0.63597168	A*B/(A+B)	blackPerCap.	assaults.	-0.199667	0.266400421	0.369571256	28	136	5
5	0.56204289	A*B/(A+B)	NAperCap.	robberies.	-0.05813	0.193031468	0.369011424	29	134	5
6	Time	taken	to	run:	382.14	seconds				

crimeqa = combinations(dat4,147,10,1)

	correlation	function.	attribute.A	attribute.B	cor.A.target.	cor.B.target.	gain	index A	index B	Function Option
1	0.96475584	A+B	burglPerPop.	larcPerPop.	0.7936958	0.926312925	0.038442911	139	141	1
2	0.91883	A*B/(A+B)	pctPolicPatrol.	larcPerPop.	-0.073406	0.926312925	-0.007482921	126	141	5
3	0.91729908	A*B/(A+B)	medRentpctHousInc.	larcPerPop.	0.2441072	0.926312925	-0.009013843	94	141	5
4	0.91590116	A*B	pctPolicPatrol.	larcPerPop.	-0.073406	0.926312925	-0.010411761	126	141	3

5	0.9157442 8	A*B/(A+B)	pctSameStat e.5.	larcPerPop.	-0.0757 27	0.92631 2925	-0.0105 68643	103	141	5
6	0.9140918 8	A*B/(A+B)	burglPerPop .	larcPerPop.	0.7936 958	0.92631 2925	-0.0122 21045	139	141	5
7	0.9134174 3	A*B/(A+B)	pctSameCo unty.5.	larcPerPop.	-0.0024 01	0.92631 2925	-0.0128 95492	102	141	5
8	0.9122451 3	A*B/(A+B)	pctPolicWhi te.	larcPerPop.	-0.1703 46	0.92631 2925	-0.0140 67796	113	141	5
9	0.9109166 5	A*B/(A+B)	pctFemDivo rc.	larcPerPop.	0.5895 53	0.92631 2925	-0.0153 96271	46	141	5
10	0.9104678 8	A*B/(A+B)	pctSmallHo usUnits.	larcPerPop.	0.4700 027	0.92631 2925	-0.0158 45046	75	141	5
11	Time	taken	to	run:	393.4	seconds				

crimeqb = combinations(dat4,147,10,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	-0.4249057	(A+B)/(A*B)	pctUrban.	robberies.	0.0010 697	0.10395 4795	0.3209 50904	17	134	6
2	0.5208485 8	A-B	pctLargHou sFam.	persPerOccupHous.	0.1807 01	-0.20061 1904	0.3202 36674	68	70	2
3	-0.4116884	(A+B)/(A*B)	fold.	robberies.	-0.0264 12	0.10395 4795	0.3077 3363	5	134	6

4	-0.4375981	(A+B)/(A*B)	pctUrban.	murders.	0.0010697	0.13016 6056	0.3074 32058	17	130	6
5	-0.4091799	(A+B)/(A*B)	landArea.	robberies.	0.0554141	0.10395 4795	0.3052 25139	121	134	6
6	-0.4085277	(A+B)/(A*B)	countyCode.	robberies.	0.0007907	0.10395 4795	0.3045 72884	3	134	6
7	-0.4080182	(A+B)/(A*B)	pctImmig.3.	robberies.	0.0957744	0.10395 4795	0.3040 63434	62	134	6
8	-0.4079615	(A+B)/(A*B)	medYrHous Built.	robberies.	-0.053618	0.10395 4795	0.3040 06741	82	134	6
9	-0.4305416	(A+B)/(A*B)	kidsBornNe vrMarr.	robberies.	0.1273166	0.10395 4795	0.3032 24924	55	134	6
10	0.53230388	A*B/(A+B)	NAperCap.	larcenies.	-0.079003	0.23013 3204	0.3021 70676	29	140	5
11	Time	taken	to	run:	387.77	seconds				

crimeqc = combinations(dat4,147,5,1)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	0.96475584	A+B	burglPerPop.	larcPerPop.	0.7936958	0.92631 2925	0.0384 42911	139	141	1
2	0.91883	A*B/(A+B)	pctPolicPatr ol.	larcPerPop.	-0.073406	0.92631 2925	-0.0074 82921	126	141	5
3	0.91729908	A*B/(A+B)	medRentpct HousInc.	larcPerPop.	0.2441072	0.92631 2925	-0.0090 13843	94	141	5

4	0.9159011 6	A*B	pctPolicPatrol.	larcPerPop.	-0.073406	0.92631 2925	-0.0104 11761	126	141	3
5	0.9157442 8	A*B/(A+B)	pctSameState.5.	larcPerPop.	-0.075727	0.92631 2925	-0.0105 68643	103	141	5
6	Time	taken	to	run:	383.73	seconds				

crimeqd = combinations(dat4,147,5,2)

	correlation	function.	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	index A	index B	Function Option
1	-0.4249057	(A+B)/(A*B)	pctUrban.	robberies.	0.0010697	0.10395 4795	0.3209 50904	17	134	6
2	0.5208485 8	A-B	pctLargHousFam.	persPerOccupHous.	0.180701	-0.20061 1904	0.3202 36674	68	70	2
3	-0.4116884	(A+B)/(A*B)	fold.	robberies.	-0.026412	0.10395 4795	0.3077 3363	5	134	6
4	-0.4375981	(A+B)/(A*B)	pctUrban.	murders.	0.0010697	0.13016 6056	0.3074 32058	17	130	6
5	-0.4091799	(A+B)/(A*B)	landArea.	robberies.	0.0554141	0.10395 4795	0.3052 25139	121	134	6
6	Time	taken	to	run:	396.97	seconds				

Appendix H: Combination Evaluations Additional Tests

Combination Creation Additional Tests - Evaluating top 10 and top 5 correlation and correlation gain combinations for 18 different target attributes

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

To help easy the readability of these results, note that the naming of the variables matches the following format.

eval_crimeLetter1Letter2

Letter 1 is the set of test. This letter ranges from a to r for the 18 different target attributes.

Letter2 is for what combinations were used and ranges from a to d.

a = top 10 sorted by correlation

b = top 10 sorted by correlation gain

c = top 5 sorted by correlation

d = top 5 sorted by correlation gain

The command executive will be displayed above each table:

eval_crimeaa = eval_combinations(dat4,131,crimeaa,10)

Model	MeanSquareError
Original Set of Attributes	0.00670764019420916
Adding Combination(A,B) to set of attributes	0.0103646440491207
Time taken:	6.2599999999984 seconds

eval_crimeab = eval_combinations(dat4,131,crimeab,10)

Model	MeanSquareError
Original Set of Attributes	0.00670764019420916
Adding Combination(A,B) to set of attributes	0.0147789955580165
Time taken:	5.70999999999913 seconds

eval_crimeac = eval_combinations(dat4,131,crimeac,10)

Model	MeanSquareError
Original Set of Attributes	0.00670764019420916

Adding Combination(A,B) to set of attributes	0.00950998454070923
Time taken:	6.20000000000005 seconds

eval_crimead = eval_combinations(dat4,131,crimead,10)

Model	MeanSquareError
Original Set of Attributes	0.00670764019420916
Adding Combination(A,B) to set of attributes	0.014352412993517
Time taken:	5.31000000000131 seconds

eval_crimeba = eval_combinations(dat4,132,crimeba,10)

Model	MeanSquareError
Original Set of Attributes	0.000293876037195803
Adding Combination(A,B) to set of attributes	6.91016114013801e-05
Time taken:	5.37999999999738 seconds

eval_crimebb = eval_combinations(dat4,132,crimebb,10)

Model	MeanSquareError
Original Set of Attributes	0.000293876037195803
Adding Combination(A,B) to set of attributes	0.000320180440272905
Time taken:	5.46000000000276 seconds

eval_crimebc = eval_combinations(dat4,132,crimebc,10)

Model	MeanSquareError
Original Set of Attributes	6.23680929914436e-05
Adding Combination(A,B) to set of attributes	0.000293876037195803
Time taken:	5.18000000000029 seconds

eval_crimebd = eval_combinations(dat4,132,crimebd,10)

Model	MeanSquareError
Original Set of Attributes	0.000293876037195803
Adding Combination(A,B) to set of attributes	0.000308490961702825
Time taken:	5.22000000000116 seconds

eval_crimeca = eval_combinations(dat4,133,crimeca,10)

Model	MeanSquareError

Original Set of Attributes	0.00424724554149057
Adding Combination(A,B) to set of attributes	0.00157808393912393
Time taken:	5.54000000000087 seconds

eval_crimecb = eval_combinations(dat4,133,crimecb,10)

Model	MeanSquareError
Original Set of Attributes	0.00424724554149057
Adding Combination(A,B) to set of attributes	0.00181590964963314
Time taken:	5.48999999999796 seconds

eval_crimecc = eval_combinations(dat4,133,crimecc,10)

Model	MeanSquareError
Original Set of Attributes	0.00424724554149057
Adding Combination(A,B) to set of attributes	0.00414302402616909
Time taken:	5.43000000000029 seconds

eval_crimecd = eval_combinations(dat4,133,crimecd,10)

Model	MeanSquareError
Original Set of Attributes	0.00424724554149057
Adding Combination(A,B) to set of attributes	0.001993093698115
Time taken:	5.15999999999985 seconds

eval_crimeda = eval_combinations(dat4,134,crimeda,10)

Model	MeanSquareError
Original Set of Attributes	3.49611557003469e-05
Adding Combination(A,B) to set of attributes	4.28662796628873e-06
Time taken:	5.39999999999782 seconds

eval_crimedb = eval_combinations(dat4,134,crimedb,10)

Model	MeanSquareError
Original Set of Attributes	3.49611557003469e-05
Adding Combination(A,B) to set of attributes	1.7081835407684e-05
Time taken:	5.45000000000073 seconds

eval_crimedc = eval_combinations(dat4,134,crimedc,10)

Model	MeanSquareError
Original Set of Attributes	3.49611557003469e-05
Adding Combination(A,B) to set of attributes	4.89199554151747e-06
Time taken:	5.29999999999927 seconds

eval_crimedd = eval_combinations(dat4,134,crimedd,10)

Model	MeanSquareError
Original Set of Attributes	3.49611557003469e-05
Adding Combination(A,B) to set of attributes	1.77289811755116e-05
Time taken:	5.2400000000016 seconds

eval_crimeea = eval_combinations(dat4,135,crimeea,10)

Model	MeanSquareError
Original Set of Attributes	0.00298902567916604
Adding Combination(A,B) to set of attributes	0.0020321947622311
Time taken:	5.39999999999782 seconds

eval_crimeeb = eval_combinations(dat4,135,crimeeb,10)

Model	MeanSquareError
Original Set of Attributes	0.00298902567916604
Adding Combination(A,B) to set of attributes	0.00180067895587209
Time taken:	5.52000000000044 seconds

eval_crimeec = eval_combinations(dat4,135,crimeec,10)

Model	MeanSquareError
Original Set of Attributes	0.00298902567916604
Adding Combination(A,B) to set of attributes	0.00259557692572883
Time taken:	5.38000000000102 seconds

eval_crimeed = eval_combinations(dat4,135,crimeed,10)

Model	MeanSquareError
Original Set of Attributes	0.00298902567916604
Adding Combination(A,B) to set of attributes	0.0026195894712114
Time taken:	5.2599999999984 seconds

eval_crimefa = eval_combinations(dat4,136,crimefa,10)

Model	MeanSquareError
Original Set of Attributes	0.000316138264570095
Adding Combination(A,B) to set of attributes	2.04207538416522e-05
Time taken:	5.4900000000016 seconds

eval_crimefb = eval_combinations(dat4,136,crimefb,10)

Model	MeanSquareError
Original Set of Attributes	0.000316138264570095
Adding Combination(A,B) to set of attributes	0.000276778090818452
Time taken:	5.54000000000087 seconds

eval_crimefc = eval_combinations(dat4,136,crimefc,10)

Model	MeanSquareError
Original Set of Attributes	0.000316138264570095
Adding Combination(A,B) to set of attributes	7.79589349077809e-06
Time taken:	5.31999999999971 seconds

eval_crimefd = eval_combinations(dat4,136,crimefd,10)

Model	MeanSquareError
Original Set of Attributes	0.000316138264570095
Adding Combination(A,B) to set of attributes	0.000311906325035002
Time taken:	5.43999999999869 seconds

eval_crimega = eval_combinations(dat4,137,crimega,10)

Model	MeanSquareError
Original Set of Attributes	0.00611747265399287
Adding Combination(A,B) to set of attributes	0.00326713165027669
Time taken:	5.45999999999913 seconds

eval_crimegb = eval_combinations(dat4,137,crimegb,10)

Model	MeanSquareError
Original Set of Attributes	0.00611747265399287
Adding Combination(A,B) to set of attributes	0.00516137521998795
Time taken:	5.71000000000276 seconds

eval_crimegc = eval_combinations(dat4,137,crimegc,10)

Model	MeanSquareError
Original Set of Attributes	0.00611747265399287
Adding Combination(A,B) to set of attributes	0.005396528919911
Time taken:	5.23999999999796 seconds

eval_crimegd = eval_combinations(dat4,137,crimegd,10)

Model	MeanSquareError
Original Set of Attributes	0.00611747265399287
Adding Combination(A,B) to set of attributes	0.00572793164742783
Time taken:	5.35000000000218 seconds

eval_crimeha = eval_combinations(dat4,138,crimeha,10)

Model	MeanSquareError
Original Set of Attributes	4.10007811076368e-05
Adding Combination(A,B) to set of attributes	3.91118885199968e-05
Time taken:	5.56999999999971 seconds

eval_crimehb = eval_combinations(dat4,138,crimehb,10)

Model	MeanSquareError
Original Set of Attributes	4.10007811076368e-05
Adding Combination(A,B) to set of attributes	4.58925149151818e-05
Time taken:	5.55999999999767 seconds

eval_crimehc = eval_combinations(dat4,138,crimehc,10)

Model	MeanSquareError
Original Set of Attributes	4.10007811076368e-05
Adding Combination(A,B) to set of attributes	3.04182112733371e-05
Time taken:	5.31999999999971 seconds

eval_crimehd = eval_combinations(dat4,138,crimehd,10)

Model	MeanSquareError
Original Set of Attributes	4.10007811076368e-05
Adding Combination(A,B) to set of attributes	4.64037499393801e-05
Time taken:	5.33000000000175 seconds

eval_crimeia = eval_combinations(dat4,139,crimeia,10)

Model	MeanSquareError
Original Set of Attributes	0.00209805350268549
Adding Combination(A,B) to set of attributes	0.000870646539032202
Time taken:	5.43000000000029 seconds

eval_crimeib = eval_combinations(dat4,139,crimeib,10)

Model	MeanSquareError
Original Set of Attributes	0.00209805350268549
Adding Combination(A,B) to set of attributes	0.00119044176261671
Time taken:	5.48999999999796 seconds

eval_crimeic = eval_combinations(dat4,139,crimeic,10)

Model	MeanSquareError
Original Set of Attributes	0.00209805350268549
Adding Combination(A,B) to set of attributes	0.000638559997052863
Time taken:	5.37000000000262 seconds

eval_crimeid = eval_combinations(dat4,139,crimeid,10)

Model	MeanSquareError
Original Set of Attributes	0.00209805350268549
Adding Combination(A,B) to set of attributes	0.00106780015903063
Time taken:	5.34999999999854 seconds

eval_crimeja = eval_combinations(dat4,140,crimeja,10)

Model	MeanSquareError
Original Set of Attributes	4.72498620524603e-05
Adding Combination(A,B) to set of attributes	1.48603100221276e-05
Time taken:	5.5 seconds

eval_crimejb = eval_combinations(dat4,140,crimejb,10)

Model	MeanSquareError
Original Set of Attributes	4.72498620524603e-05
Adding Combination(A,B) to set of attributes	4.41332679065751e-05
Time taken:	5.70999999999913 seconds

eval_crimejc = eval_combinations(dat4,140,crimejc,10)

Model	MeanSquareError
Original Set of Attributes	4.72498620524603e-05
Adding Combination(A,B) to set of attributes	1.42931923593887e-05
Time taken:	5.42000000000189 seconds

eval_crimejd = eval_combinations(dat4,140,crimejd,10)

Model	MeanSquareError
Original Set of Attributes	4.72498620524603e-05
Adding Combination(A,B) to set of attributes	4.77011673721615e-05
Time taken:	5.31999999999971 seconds

eval_crimeka = eval_combinations(dat4,141,crimeka,10)

Model	MeanSquareError
Original Set of Attributes	0.000382433732819618
Adding Combination(A,B) to set of attributes	0.000370093518770618
Time taken:	5.54999999999927 seconds

eval_crimekb = eval_combinations(dat4,141,crimekb,10)

Model	MeanSquareError
Original Set of Attributes	0.000382433732819618
Adding Combination(A,B) to set of attributes	0.000399026441191469
Time taken:	5.36999999999898 seconds

eval_crimekc = eval_combinations(dat4,141,crimekc,10)

Model	MeanSquareError
Original Set of Attributes	0.000382433732819618
Adding Combination(A,B) to set of attributes	0.000408538040131085
Time taken:	5.38999999999942 seconds

eval_crimekd = eval_combinations(dat4,141,crimekd,10)

Model	MeanSquareError
Original Set of Attributes	0.000382433732819618
Adding Combination(A,B) to set of attributes	0.000389925756413684
Time taken:	5.39000000000306 seconds

eval_crimela = eval_combinations(dat4,142,crimela,10)

Model	MeanSquareError
Original Set of Attributes	4.93651867085512e-05
Adding Combination(A,B) to set of attributes	1.88786189072047e-05
Time taken:	5.45999999999913 seconds

eval_crimelb = eval_combinations(dat4,142,crimelb,10)

Model	MeanSquareError
Original Set of Attributes	4.93651867085512e-05
Adding Combination(A,B) to set of attributes	4.05560769988033e-05
Time taken:	5.4900000000016 seconds

eval_crimelc = eval_combinations(dat4,142,crimelc,10)

Model	MeanSquareError
Original Set of Attributes	4.93651867085512e-05
Adding Combination(A,B) to set of attributes	1.8196351140289e-05
Time taken:	5.34000000000015 seconds

eval_crimeld = eval_combinations(dat4,142,crimeld,10)

Model	MeanSquareError
Original Set of Attributes	4.93651867085512e-05
Adding Combination(A,B) to set of attributes	4.62673753711535e-05
Time taken:	5.22999999999956 seconds

eval_crimema = eval_combinations(dat4,143,crimema,10)

Model	MeanSquareError
Original Set of Attributes	0.00296020776613853
Adding Combination(A,B) to set of attributes	0.00285132706660797
Time taken:	5.40999999999985 seconds

eval_crimemb = eval_combinations(dat4,143,crimemb,10)

Model	MeanSquareError
Original Set of Attributes	0.00296020776613853
Adding Combination(A,B) to set of attributes	0.00178116073477082
Time taken:	5.54000000000087 seconds

eval_crimemc = eval_combinations(dat4,143,crimemc,10)

Model	MeanSquareError
Original Set of Attributes	0.00296020776613853
Adding Combination(A,B) to set of attributes	0.00286192470169739
Time taken:	5.43000000000029 seconds

eval_crimemd = eval_combinations(dat4,143,crimemd,10)

Model	MeanSquareError
Original Set of Attributes	0.00296020776613853
Adding Combination(A,B) to set of attributes	0.00209262861917607
Time taken:	5.38999999999942 seconds

eval_crimena = eval_combinations(dat4,144,crimena,10)

Model	MeanSquareError
Original Set of Attributes	0.000559387376384578
Adding Combination(A,B) to set of attributes	1.03279939573203e-05
Time taken:	5.36999999999898 seconds

eval_crimenb = eval_combinations(dat4,144,crimenb,10)

Model	MeanSquareError
Original Set of Attributes	0.000559387376384578
Adding Combination(A,B) to set of attributes	0.000675316583927839
Time taken:	5.36999999999898 seconds

eval_crimenc = eval_combinations(dat4,144,crimenc,10)

Model	MeanSquareError
Original Set of Attributes	0.000559387376384578
Adding Combination(A,B) to set of attributes	1.13403387446071e-05
Time taken:	5.17000000000189 seconds

eval_crimend = eval_combinations(dat4,144,crimend,10)

Model	MeanSquareError
Original Set of Attributes	0.000559387376384578
Adding Combination(A,B) to set of attributes	0.000611401665748847
Time taken:	5.31000000000131 seconds

eval_crimeoa = eval_combinations(dat4,145,crimeoa,10)

Model	MeanSquareError
Original Set of Attributes	0.0145519143096122
Adding Combination(A,B) to set of attributes	0.0168094551096332
Time taken:	5.39999999999782 seconds

eval_crimeob = eval_combinations(dat4,145,crimeob,10)

Model	MeanSquareError
Original Set of Attributes	0.0145519143096122
Adding Combination(A,B) to set of attributes	0.00947436008303984
Time taken:	5.31999999999971 seconds

eval_crimeoc = eval_combinations(dat4,145,crimeoc,10)

Model	MeanSquareError
Original Set of Attributes	0.0145519143096122
Adding Combination(A,B) to set of attributes	0.0107400944123745
Time taken:	5.27000000000044 seconds

eval_crimeod = eval_combinations(dat4,145,crimeod,10)

Model	MeanSquareError
Original Set of Attributes	0.0145519143096122
Adding Combination(A,B) to set of attributes	0.0107400944123745
Time taken:	5.08000000000175 seconds

eval_crimepa = eval_combinations(dat4,146,crimepa,10)

Model	MeanSquareError
Original Set of Attributes	0.00184756684194111
Adding Combination(A,B) to set of attributes	0.00168067149592503
Time taken:	5.44999999999709 seconds

eval_crimepb = eval_combinations(dat4,146,crimepb,10)

Model	MeanSquareError
Original Set of Attributes	0.00184756684194111
Adding Combination(A,B) to set of attributes	0.00215595092304493
Time taken:	5.40999999999985 seconds

eval_crimepc = eval_combinations(dat4,146,crimepc,10)

Model	MeanSquareError
Original Set of Attributes	0.00184756684194111
Adding Combination(A,B) to set of attributes	0.00167154053832106
Time taken:	5.2400000000016 seconds

eval_crimepd = eval_combinations(dat4,146,crimepd,10)

Model	MeanSquareError
Original Set of Attributes	0.00184756684194111
Adding Combination(A,B) to set of attributes	0.00200571227505564
Time taken:	5.40999999999985 seconds

eval_crimeqa = eval_combinations(dat4,147,crimeqa,10)

Model	MeanSquareError
Original Set of Attributes	0.000455583455884328
Adding Combination(A,B) to set of attributes	0.00046014635758398
Time taken:	5.40000000000146 seconds

eval_crimeqb = eval_combinations(dat4,147,crimeqb,10)

Model	MeanSquareError
Original Set of Attributes	0.000455583455884328
Adding Combination(A,B) to set of attributes	0.000479732394929723
Time taken:	5.42999999999665 seconds

eval_crimeqc = eval_combinations(dat4,147,crimeqc,10)

Model	MeanSquareError
Original Set of Attributes	0.000455583455884328
Adding Combination(A,B) to set of attributes	0.000453685593008021
Time taken:	5.27000000000044 seconds

eval_crimeqd = eval_combinations(dat4,147,crimeqd,10)

Model	MeanSquareError
Original Set of Attributes	0.000455583455884328
Adding Combination(A,B) to set of attributes	0.000459076185357597
Time taken:	5.2400000000016 seconds

eval_crimera = eval_combinations(dat4,130,crimera,10)

Model	MeanSquareError
Original Set of Attributes	0.000186180404230569
Adding Combination(A,B) to set of attributes	9.21913139943653e-06
Time taken:	5.89999999999782 seconds

eval_crimerb = eval_combinations(dat4,130,crimerb,10)

Model	MeanSquareError
Original Set of Attributes	0.000186180404230569
Adding Combination(A,B) to set of attributes	0.000209954015798485
Time taken:	5.60000000000218 seconds

eval_crimerc = eval_combinations(dat4,130,crimerc,10)

Model	MeanSquareError
Original Set of Attributes	0.000186180404230569
Adding Combination(A,B) to set of attributes	1.07036537169381e-05
Time taken:	5.34999999999854 seconds

eval_crimerd = eval_combinations(dat4,130,crimerd,10)

Model	MeanSquareError
Original Set of Attributes	0.000186180404230569
Adding Combination(A,B) to set of attributes	0.000168098930768574
Time taken:	5.45999999999913 seconds

Appendix I: Combination Evaluation Additional Tests

Combination Evaluation Additional Tests - Evaluating the top combination of correlation and correlation gain for 18 different target attributes

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

To help easy the readability of these results, note that the naming of the variables matches the following format.

eval_crimeLetterNumber

Letter is the set of test. This letter ranges from a to r for the 18 different target attributes.

Number is for what type of combination s were used and ranges from nothing to 2 .

Empty = top correlation combination

2 = top correlation gain combionation

The command executive will be displayed above each table:

eval_crimea = eval_combination(dat4,130,6,131,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000186180404230569
Adding Combination(A,B) to set of attributes	7.78496090542863e-06
Replacing A and B with combination(A,B)	8.39533052574119e-06
Time taken:	9.84999999999999 seconds

eval_crimea2 = eval_combination(dat4,130,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000186180404230569
Adding Combination(A,B) to set of attributes	0.000186478785819829
Replacing A and B with combination(A,B)	0.000186115146709913
Time taken:	9.56 seconds

eval_crimeb = eval_combination(dat4,131,34,135,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00670764019420916

Adding Combination(A,B) to set of attributes	0.00678507545674807
Replacing A and B with combination(A,B)	0.00617546681669859
Time taken:	9.47 seconds

eval_crimeb2 = eval_combination(dat4,131,29,130,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00670764019420916
Adding Combination(A,B) to set of attributes	0.00914332431787895
Replacing A and B with combination(A,B)	0.00814516283200927
Time taken:	9.44999999999999 seconds

eval_crimec = eval_combination(dat4,132,6,133,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000293876037195803
Adding Combination(A,B) to set of attributes	7.79215563463529e-05
Replacing A and B with combination(A,B)	7.76778700775162e-05
Time taken:	9.54000000000002 seconds

eval_crimec2 = eval_combination(dat4,132,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000293876037195803
Adding Combination(A,B) to set of attributes	0.000293367222751455
Replacing A and B with combination(A,B)	0.0002918329693278
Time taken:	9.45999999999998 seconds

eval_crimed = eval_combination(dat4,133,141,146,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00424724554149057
Adding Combination(A,B) to set of attributes	0.00420846464694755
Replacing A and B with combination(A,B)	0.0042569587563881
Time taken:	9.44000000000003 seconds

eval_crimed2 = eval_combination(dat4,133,100,116,4,10)

Model	MeanSquareError
Original Set of Attributes	0.00424724554149057
Adding Combination(A,B) to set of attributes	0.00424724554149057
Replacing A and B with combination(A,B)	0.00421629299824036

Time taken:	9.54999999999998 seconds

eval_crimee = eval_combination(dat4,134,16,135,3,10)

Model	MeanSquareError
Original Set of Attributes	3.49611557003469e-05
Adding Combination(A,B) to set of attributes	5.12659322017538e-06
Replacing A and B with combination(A,B)	4.73029200272907e-06
Time taken:	

eval_crimee2 = eval_combination(dat4,134,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	3.49611557003469e-05
Adding Combination(A,B) to set of attributes	3.49852274597792e-05
Replacing A and B with combination(A,B)	3.49835518180261e-05
Time taken:	9.58000000000001 seconds

eval_crimef = eval_combination(dat4,135,56,143,1,10)

Model	MeanSquareError
Original Set of Attributes	0.00298902567916604
Adding Combination(A,B) to set of attributes	0.00294258395742132
Replacing A and B with combination(A,B)	0.00318485640084046
Time taken:	9.69999999999999 seconds

eval_crimef2 = eval_combination(dat4,135,29,134,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00298902567916604
Adding Combination(A,B) to set of attributes	0.00253761711676022
Replacing A and B with combination(A,B)	0.0041720636118731
Time taken:	9.80000000000001 seconds

eval_crimeg = eval_combination(dat4,136,6,137,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000316138264570095
Adding Combination(A,B) to set of attributes	7.59181265558452e-06
Replacing A and B with combination(A,B)	8.47356656153845e-06
Time taken:	9.78999999999996 seconds

eval_crimeg2 = eval_combination(dat4,136,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000316138264570095
Adding Combination(A,B) to set of attributes	0.00031616475980390 69
Replacing A and B with combination(A,B)	0.000316046073554983
Time taken:	9.93000000000001 seconds

eval_crimeh = eval_combination(dat4,137,103,146,3,10)

Model	MeanSquareError
Original Set of Attributes	0.00611747265399287
Adding Combination(A,B) to set of attributes	0.00614040966091622
Replacing A and B with combination(A,B)	0.00611205111346941
Time taken:	10.01 seconds

eval_crimeh2 = eval_combination(dat4,137,136,140,4,10)

Model	MeanSquareError
Original Set of Attributes	0.00611747265399287
Adding Combination(A,B) to set of attributes	0.00467869741111295
Replacing A and B with combination(A,B)	0.00132718349218429
Time taken:	9.83000000000004 seconds

eval_crimei = eval_combination(dat4,138,6,139,3,10)

Model	MeanSquareError
Original Set of Attributes	4.10007811076368e-05
Adding Combination(A,B) to set of attributes	2.74685569902853e-05
Replacing A and B with combination(A,B)	2.56949817699817e-05
Time taken:	9.75999999999999 seconds

eval_crimei2 = eval_combination(dat4,138,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	4.10007811076368e-05
Adding Combination(A,B) to set of attributes	4.11197504204162e-05
Replacing A and B with combination(A,B)	4.13714626455718e-05
Time taken:	9.81999999999999 seconds

eval_crimej = eval_combination(dat4,139,141,147,2,10)

Model	MeanSquareError
Original Set of Attributes	0.00209805350268549
Adding Combination(A,B) to set of attributes	0.000745133186555358
Replacing A and B with combination(A,B)	0.000900579988166779
Time taken:	9.81 seconds

eval_crimej2 = eval_combination(dat4,139,29,138,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00209805350268549
Adding Combination(A,B) to set of attributes	0.00106651239623822
Replacing A and B with combination(A,B)	0.00162451121038833
Time taken:	9.88999999999999 seconds

eval_crimek = eval_combination(dat4,140,6,141,3,10)

Model	MeanSquareError
Original Set of Attributes	4.72498620524603e-05
Adding Combination(A,B) to set of attributes	1.48769324981105e-05
Replacing A and B with combination(A,B)	1.46887398696818e-05
Time taken:	9.79000000000002 seconds

eval_crimek2 = eval_combination(dat4,140,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	4.72498620524603e-05
Adding Combination(A,B) to set of attributes	4.74727345762114e-05
Replacing A and B with combination(A,B)	4.75519857867333e-05
Time taken:	9.92000000000002 seconds

eval_crimel = eval_combination(dat4,141,126,147,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000382433732819618
Adding Combination(A,B) to set of attributes	0.000387719788745144
Replacing A and B with combination(A,B)	0.00355677015317519
Time taken:	9.65999999999997 seconds

eval_crimel2 = eval_combination(dat4,141,95,140,4,10)

Model	MeanSquareError
Original Set of Attributes	0.000382433732819618

Adding Combination(A,B) to set of attributes	0.000389041154281679
Replacing A and B with combination(A,B)	0.000434730032481409
Time taken:	9.70000000000005 seconds

eval_crimem = eval_combination(dat4,142,16,143,3,10)

Model	MeanSquareError
Original Set of Attributes	4.93651867085512e-05
Adding Combination(A,B) to set of attributes	7.29064665596129e-06
Replacing A and B with combination(A,B)	7.09210491275113e-06
Time taken:	9.65999999999997 seconds

eval_crimem2 = eval_combination(dat4,142,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	4.93651867085512e-05
Adding Combination(A,B) to set of attributes	4.94993904999454e-05
Replacing A and B with combination(A,B)	4.94364210137872e-05
Time taken:	9.89000000000004 seconds

eval_crimen = eval_combination(dat4,143,51,135,3,10)

Model	MeanSquareError
Original Set of Attributes	0.00296020776613853
Adding Combination(A,B) to set of attributes	0.00286761478765386
Replacing A and B with combination(A,B)	0.00281645961528142
Time taken:	9.85999999999996 seconds

eval_crimen2 = eval_combination(dat4,143,29,142,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00296020776613853
Adding Combination(A,B) to set of attributes	0.00816015362932799
Replacing A and B with combination(A,B)	0.00979282952073236
Time taken:	9.79999999999995 seconds

eval_crimeo = eval_combination(dat4,144,6,145,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000559387376384578
Adding Combination(A,B) to set of attributes	9.81082553828853e-06
Replacing A and B with combination(A,B)	1.07546990881702e-05

Time taken:	9.94 seconds

eval_crimeo2 = eval_combination(dat4,144,121,122,3,10)

Model	MeanSquareError
Original Set of Attributes	0.000559387376384578
Adding Combination(A,B) to set of attributes	0.000554126874009802
Replacing A and B with combination(A,B)	0.000553706770188589
Time taken:	9.97999999999996 seconds

eval_crimep = eval_combination(dat4,145,29,144,5,10)

Model	MeanSquareError
Original Set of Attributes	0.0145519143096122
Adding Combination(A,B) to set of attributes	0.00768825273293829
Replacing A and B with combination(A,B)	0.0125396990703746
Time taken:	9.71000000000004 seconds

eval_crimep2 = eval_combination(dat4,145,29,144,5,10)

Model	MeanSquareError
Original Set of Attributes	0.0145519143096122
Adding Combination(A,B) to set of attributes	0.00768825273293829
Replacing A and B with combination(A,B)	0.0125396990703746
Time taken:	9.85999999999996 seconds

eval_crimeq = eval_combination(dat4,146,135,137,1,10)

Model	MeanSquareError
Original Set of Attributes	0.00184756684194111
Adding Combination(A,B) to set of attributes	0.00186327182361384
Replacing A and B with combination(A,B)	0.00204001520446897
Time taken:	9.97000000000003 seconds

eval_crimeq2 = eval_combination(dat4,146,29,136,5,10)

Model	MeanSquareError
Original Set of Attributes	0.00184756684194111
Adding Combination(A,B) to set of attributes	0.00191802919647298
Replacing A and B with combination(A,B)	0.00188671277985573
Time taken:	10.04 seconds

eval_crimer = eval_combination(dat4,147,139,141,1,10)

Model	MeanSquareError
Original Set of Attributes	0.000455583455884328
Adding Combination(A,B) to set of attributes	0.000443290790666546
Replacing A and B with combination(A,B)	0.000440590843759007
Time taken:	9.90999999999997 seconds

eval_crimer2 = eval_combination(dat4,147,17,134,6,10)

Model	MeanSquareError
Original Set of Attributes	0.000455583455884328
Adding Combination(A,B) to set of attributes	0.000455583455884328
Replacing A and B with combination(A,B)	0.000387615547719589
Time taken:	9.84000000000003 seconds

Appendix J: Combination over previous combination results

Using dataset : Communities and Crime http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime

This table shows the top 10 combinations sorted by correlation when using a dataset with attributes made from a previous run of the combinations function.

#	corr	func	attribute.A	attribute.B	cor.A .target.	cor.B .target.	gain	Index A	Index B	Func Opt
1	0.8564	A*B	violentPerPop.	X.A.B.func...larcPerPop..nonViolPerPop...A.B	0.85208	-0.61112	0.004345	143	145	3
2	0.8493	A*B	pctSameState.5.	violentPerPop.	0.018673	0.85208	-0.00277	101	143	3
3	0.8481	A*B	pctPolicWhite.	violentPerPop.	-0.21341	0.85208	-0.00388	111	143	3
4	0.8453	A*B	pctPolicPatrol.	violentPerPop.	-0.09882	0.85208	-0.00673	124	143	3
5	0.8449	A*B/(A+B)	pctSameState.5.	violentPerPop.	0.018673	0.85208	-0.00716	101	143	5
6	0.8430	A-B	officDrugUnits.	violentPerPop.	0.077493	0.85208	-0.00905	116	143	2
7	0.8427	A-B	persHomeless.	violentPerPop.	0.079619	0.85208	-0.00929	96	143	2
8	0.8426	A-B	persEmergShelt.	violentPerPop.	0.122959	0.85208	-0.00939	95	143	2
9	0.8418	A*B/(A+B)	pctPolicWhite.	violentPerPop.	-0.21341	0.85208	-0.01019	111	143	5
10	0.8414	A*B/(A+B)	pctHousOccup.	violentPerPop.	-0.26657	0.85208	-0.01062	76	143	5
	Time taken to create and sort combinations: 481.68 seconds									
IndexA & IndexB are the index numbers for attribute A and attribute B										
FunctionOption is the number that represents the arithmetic combination being used.										

This is the result of running the eval_combinations method using the results above.

Note that the Original set of Attributes in this case is a set of attributes that has combinations in it.

Model	MeanSquareError	RSquared
Original Set of Attributes	0.00104090011765959	0.950834119158483
Adding Combination(A,B) to set of attributes	0.00195777863280288	0.95105866961798
Time taken:	5.40000000000146	

Appendix K: Clustering Additional Results

The additional results outlined herein are represented as the raw output from the *AttributeClustering* function described in Section 3.2.2 Attribute Clustering Function.

The descriptions for these datasets can be found at the links provided in Table 20: Attribute Clustering Resluts: Datasets and Runtimes if one wishes to manually associate the attribute numbers indicated here with the names of the attributes.

This format is presented because some datasets were not stored in a format that included their attribute names with them.

This first cluster corresponds to the Mushroom dataset clustering identified in Section 4.2.3 Attribute Clustering Report Output.

The format in which the clusters are stored is as follows:

```
[[I]]
[[I]][[J]]
[1] A
```

Where *I* indicates the cluster number for the attribute A, and *J* indicates the position attribute A occupies in the list of attributes for cluster *I*

In the case of the example below, the entry

```
[[1]]
[[1]][[1]]
[1] 1
```

Indicates that the first attribute in the first cluster is the first attribute in the original dataset. The entry

```
[[2]][[4]]
[1] 21
```

Indicates that the fourth attribute in the second cluster is the twenty-first attribute in the original dataset.

These raw outputs are not formatted for human readability but rather are included in the actual *R* format as examples of the return value of the *AttributeClustering* function

Mushroom dataset clusters:

```
[[1]]
[[1]][[1]]
[1] 1


[[2]]
[[2]][[1]]
[1] 2

[[2]][[2]]
[1] 3

[[2]][[3]]
[1] 14

[[2]][[4]]
[1] 21

[[2]][[5]]
[1] 6

[[2]][[6]]
[1] 17


[[3]]
[[3]][[1]]
[1] 4

[[3]][[2]]
[1] 5

[[3]][[3]]
[1] 20

[[3]][[4]]
[1] 11

[[3]][[5]]
[1] 9

[[3]][[6]]
[1] 19

[[3]][[7]]
[1] 22

[[3]][[8]]
[1] 10

[[3]][[9]]
[1] 15

[[3]][[10]]
[1] 8

[[3]][[11]]
[1] 13

[[3]][[12]]
[1] 7

[[3]][[13]]
[1] 12

[[3]][[14]]
[1] 18


[[4]]
[[4]][[1]]
[1] 16
```

Connect dataset clusters:

```
[[1]]
[[1]][[1]]
[1] 1

[[1]][[2]]
[1] 20

[[1]][[3]]
[1] 21

[[1]][[4]]
[1] 22

[[1]][[5]]
[1] 23

[[1]][[6]]
[1] 24

[[1]][[7]]
[1] 25


[[2]]
[[2]][[1]]
[1] 2

[[2]][[2]]
[1] 3

[[2]][[3]]
[1] 4

[[2]][[4]]
[1] 5

[[2]][[5]]
[1] 6

[[2]][[6]]
[1] 7


[[3]]
[[3]][[1]]
[1] 8

[[3]][[2]]
[1] 9

[[3]][[3]]
[1] 10

[[3]][[4]]
[1] 11

[[3]][[5]]
[1] 12

[[3]][[6]]
[1] 13
```

```
[[4]]
[[4]][[1]]
[1] 14

[[4]][[2]]
[1] 15

[[4]][[3]]
[1] 16

[[4]][[4]]
[1] 17

[[4]][[5]]
[1] 18

[[4]][[6]]
[1] 19


[[5]]
[[5]][[1]]
[1] 26

[[5]][[2]]
[1] 27

[[5]][[3]]
[1] 28

[[5]][[4]]
[1] 29

[[5]][[5]]
[1] 30

[[5]][[6]]
[1] 31


[[6]]
[[6]][[1]]
[1] 32

[[6]][[2]]
[1] 33

[[6]][[3]]
[1] 34

[[6]][[4]]
[1] 35

[[6]][[5]]
[1] 36

[[6]][[6]]
[1] 37


[[7]]
[[7]][[1]]
```

```
[1] 38

[[7]][[2]]
[1] 39

[[7]][[3]]
[1] 40

[[7]][[4]]
[1] 41

[[7]][[5]]
[1] 42
```

Cup-98 dataset clusters:

```
[[1]]

[[1]][[1]]
[1] 1


[[2]]
[[2]][[1]]
[1] 2

[[2]][[2]]
[1] 11


[[3]]
[[3]][[1]]
[1] 3

[[3]][[2]]
[1] 5


[[4]]
[[4]][[1]]
[1] 4

[[4]][[2]]
[1] 9


[[5]]
[[5]][[1]]
[1] 6

[[5]][[2]]
[1] 10

[[5]][[3]]
[1] 64

[[5]][[4]]
[1] 65

[[5]][[5]]
[1] 66

[[5]][[6]]
[1] 19


[[6]]
[[6]][[1]]
[1] 7


[[7]]
[[7]][[1]]
[1] 8


[[8]]
[[8]][[1]]
[1] 12

[[8]][[2]]
[1] 13

[[8]][[3]]
[1] 18

[[8]][[4]]
[1] 21

[[8]][[5]]
[1] 29

[[8]][[6]]
[1] 20

[[8]][[7]]
[1] 37


[[9]]
[[9]][[1]]
[1] 14

[[9]][[2]]
[1] 35

[[9]][[3]]
[1] 36


[[10]]
[[10]][[1]]
[1] 15

[[10]][[2]]
[1] 16

[[10]][[3]]
[1] 17


[[11]]
[[11]][[1]]
[1] 22

[[11]][[2]]
[1] 23

[[11]][[3]]
[1] 25

[[11]][[4]]
[1] 26

[[11]][[5]]
[1] 27

[[11]][[6]]
[1] 32

[[11]][[7]]
[1] 28

[[11]][[8]]
[1] 34

[[11]][[9]]
[1] 30

[[11]][[10]]
[1] 31


[[12]]
[[12]][[1]]
[1] 24

[[12]][[2]]
[1] 33


[[13]]
[[13]][[1]]
[1] 38

[[13]][[2]]
[1] 39

[[13]][[3]]
[1] 40

[[13]][[4]]
[1] 41

[[13]][[5]]
[1] 42

[[13]][[6]]
[1] 63


[[14]]
[[14]][[1]]
[1] 43

[[14]][[2]]
[1] 44

[[14]][[3]]
[1] 45


[[15]]
[[15]][[1]]
[1] 46

[[15]][[2]]
[1] 48

[[15]][[3]]
[1] 49

[[15]][[4]]
[1] 47
```

```
[[16]]
[[16]][[1]]
[1] 50

[[16]][[2]]
[1] 51

[[16]][[3]]
[1] 52


[[17]]
[[17]][[1]]
[1] 53

[[17]][[2]]
[1] 54

[[17]][[3]]
[1] 55


[[18]]
[[18]][[1]]
[1] 56

[[18]][[2]]
[1] 58

[[18]][[3]]
[1] 59

[[18]][[4]]
[1] 57


[[19]]
[[19]][[1]]
[1] 60

[[19]][[2]]
[1] 61


[[20]]
[[20]][[1]]
[1] 62
```

Census Income dataset
clusters:

```
[[1]]
[[1]][[1]]
[1] 1

[[1]][[2]]
[1] 4

[[1]][[3]]
[1] 2


[[2]]
[[2]][[1]]
[1] 3

[[2]][[2]]
[1] 5

[[2]][[3]]
[1] 7


[[3]]
[[3]][[1]]
[1] 6

[[3]][[2]]
[1] 8
```

Nursery dataset clusters:

```
[[1]]
[[1]][[1]]
[1] 1


[[2]]
[[2]][[1]]
[1] 2


[[3]]
[[3]][[1]]
[1] 3


[[4]]
[[4]][[1]]
[1] 4


[[5]]
[[5]][[1]]
[1] 5


[[6]]
[[6]][[1]]
[1] 6


[[7]]
[[7]][[1]]
[1] 7


[[8]]
[[8]][[1]]
[1] 8
```

Hayes-Roth dataset
clusters:

```
[[1]]
[[1]][[1]]
[1] 1


[[2]]
[[2]][[1]]
[1] 2


[[3]]
[[3]][[1]]
[1] 3


[[4]]
[[4]][[1]]
[1] 4


[[5]]
[[5]][[1]]
[1] 5
```

Car dataset clusters:

```
[[1]]
[[1]][[1]]
[1] 1


[[2]]
[[2]][[1]]
[1] 2


[[3]]
[[3]][[1]]
[1] 3


[[4]]
[[4]][[1]]
[1] 4


[[5]]
[[5]][[1]]
[1] 5


[[6]]
[[6]][[1]]
[1] 6
```

Chess dataset clusters:

```
[[1]]
[[1]][[1]]
[1] 1

[[1]][[2]]
[1] 11

[[1]][[3]]
[1] 26

[[1]][[4]]
[1] 36

[[1]][[5]]
[1] 35

[[1]][[6]]
[1] 32

[[1]][[7]]
[1] 15

[[1]][[8]]
[1] 31

[[1]][[9]]
[1] 20

[[1]][[10]]
[1] 29


[[2]]
[[2]][[1]]
[1] 2

[[2]][[2]]
[1] 5

[[2]][[3]]
[1] 7

[[2]][[4]]
[1] 8

[[2]][[5]]
[1] 9

[[2]][[6]]
[1] 22

[[2]][[7]]
[1] 10

[[2]][[8]]
[1] 21


[[3]]
[[3]][[1]]
[1] 3

[[3]][[2]]
[1] 24

[[3]][[3]]
[1] 13

[[3]][[4]]
[1] 18

[[3]][[5]]
[1] 34

[[3]][[6]]
[1] 4

[[3]][[7]]
[1] 19


[[4]]
[[4]][[1]]
[1] 6

[[4]][[2]]
[1] 27

[[4]][[3]]
[1] 33

[[4]][[4]]
[1] 30


[[5]]
[[5]][[1]]
[1] 12

[[5]][[2]]
[1] 17

[[5]][[3]]
[1] 23


[[6]]
[[6]][[1]]
[1] 14

[[6]][[2]]
[1] 16


[[7]]
[[7]][[1]]
[1] 25


[[8]]
[[8]][[1]]
[1] 28
```

TicTacToe dataset clusters:

```
[[1]]
[[1]][[1]]
[1] 1

[[1]][[2]]
[1] 6


[[2]]
[[2]][[1]]
[1] 2

[[2]][[2]]
[1] 7


[[3]]
[[3]][[1]]
[1] 3

[[3]][[2]]
[1] 4


[[4]]
[[4]][[1]]
[1] 5


[[5]]
[[5]][[1]]
[1] 8


[[6]]
[[6]][[1]]
[1] 9
```