AN EFFECTIVE APPROACH FOR COMPARISON OF ASSOCIATION RULE MINING ALGORITHMS BASED ON CONTROLLED DATA, STATISTICAL INFERENCE AND MULTIPLE CRITERIA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SANAM AZADIAMIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

FEBRUARY 2016

Approval of the thesis:

AN EFFECTIVE APPROACH FOR COMPARISON OF ASSOCIATION RULE MINING ALGORITHMS BASED ON CONTROLLED DATA, STATISTICAL INFERENCE AND MULTIPLE CRITERIA

submitted by **SANAM AZADIAMIN** in partial fulfillment of the requirement for the degree of **Master of Science in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Murat Köksalan _____
Head of Department, **Industrial Engineering**

Prof. Dr. Gülser Köksal _____
Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Nur Evin Özdemirel _____
Industrial Engineering Dept., METU

Prof. Dr. Gülser Köksal _____
Industrial Engineering Dept., METU

Prof. Dr. Murat Caner Testik _____
Industrial Engineering Dept., HÜ

Assoc. Prof. Dr. Esra Karasakal _____
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Cem İyigün _____
Industrial Engineering Dept., METU

Date: 02.2.2016

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: SANAM AZADIAMIN

Signature :

ABSTRACT

AN EFFECTIVE APPROACH FOR COMPARISON OF ASSOCIATION RULE MINING ALGORITHMS BASED ON CONTROLLED DATA, STATISTICAL INFERENCE AND MULTIPLE CRITERIA

Azadiamin, Sanam

M.S., Department of Industrial Engineering

Supervisor: Prof. Dr. Gülser Köksal

February 2016, 89 pages

Association rules are an important set of data mining results, which are helpful in handling large amount of data and extracting useful association information from them. There are many algorithms developed for finding interesting association rules and also some other algorithms for rule reduction purposes. All of the proposed methods have some strong and weak points, which can be useful according to their application areas. In the literature, there exist several comparison studies trying to find the best algorithm according to the user's interests. But every comparison approach considers these algorithms using different measures, and it is hard to assess performance of an algorithm with respect to a measure since interesting association rules are unknown. A novel comparison method has been proposed by Jabarnejad (2010) based on interesting rules generated by logistic regression to compare rule reduction algorithms. In this study, this approach is extended to cover all association rule mining algorithms, on a broader set of test data developing and using relevant

comparison measures. This approach utilizes design and analysis of experiments to generate test data. Furthermore, it defines several comparison measures, and the dependency and importance of these measures are analyzed using statistical methods such as factor analysis, ANOVA and nonparametric hypothesis tests. Finally, if statistical analyses show significant differences between applied association rule mining methods, it handles multiple comparisons using PROMETHEE. The approach is demonstrated by comparing three association rule mining algorithms. The results are discussed and future research directions are presented.

Key Words: Association rule mining, comparison of association rule mining methods, interesting rules, comparison measures, factor analysis, ANOVA, nonparametric hypothesis test, PROMETHEE.

ÖZ

BİRLİKTELİK KURAL MADENCİLİĞİ ALGORİTMALARININ KARŞILAŞTIRILMASI İÇİN KONTROLLÜ VERİ, İSTATİSTİKSEL ÇIKARIM VE ÇOK KRİTER TABANLI ETKİLİ BİR YAKLAŞIM

Azadiamin, Sanam

Yüksek Lisans, Endüstri Mühendisligi Bölümü

Tez Yöneticisi: Prof. Dr. Gülser Köksal

Şubat 2016, 89 Sayfa

Birliktelik kuralları, veri madenciliğinin önemli sonuçlarından biri olarak hacimli verilerin analizine ve onlardan faydalı bilgiler çıkarılmasına yardımcı olur. İlginç birliktelik kuralların bulunması ve bunların azaltılması için bir çok algoritma geliştirilmiştir. Tüm önerilen metotların güçlü ve zayıf noktaları vardır ve bu metotlar uygulanılan veriye göre faydalı olabilir.

Literatürde birliktelik kural madenciliği algoritmalarını karşılaştıran bazı çalışmalar mevcuttur. Ancak bunlar en iyi algoritmayı belirlemede yeterince başarılı değildir. Her karşılaştırma yöntemi bu algoritmaları farklı ölçülere göre değerlendirmekte ve doğru kurallar bilinmediği için bu değerlendirme yeterince güvenilir sonuç veremeyebilmektedir. Jabarnejad (2010) lojistik regresyona dayalı bir mekanizmadan ilginç kurallar elde eden ve bunları bulmada en başarılı olan kural

azaltma algoritmasını belirleyen bir yöntem geliştirmiştir. Bu çalışmada, bu yöntem genel olarak birliktelik kural madenciliği algoritmalarını karşılaştırmak üzere genişletilmiştir. Bu amaçla doğru kuralların nasıl türetileceği, algoritmaların hangi veriler üzerinde test edileceği, karşılaştırmada hangi ölçülerin nasıl kullanılacağı ile ilgili bir yaklaşım önerilmiştir. Test verilerinin oluşturulması için istatistiksel deney tasarımı ve analizi; karşılaştırma ölçülerinin ilişkilerinin ve önemlerinin değerlendirilmesi için faktör analizi, ANOVA ve parametrik olmayan hipotez testi gibi istatistiksel metotlar kullanılmıştır. Sonuçta, eğer karşılaştırılan birliktelik kural madenciliği algoritmaları arasında önemli istatistiksel farklar varsa, bunların karşılaştırması PROMETHEE ile yapılmıştır. Yöntem, örnek olarak seçilen üç algoritmanın karşılaştırılması için uygulanmıştır. Sonuçlar tartışılmış, ileri araştırma konuları sunulmuştur.

Anahtar kelimeler: Birliktelik kural madenciliği, birliktelik kural madenciliği metotlarının karşılaştırılması, ilginç kurallar, karşılaştırma ölçüleri, faktör analizi, ANOVA, parametrik olmayan hipotez testi, PROMETHEE.

To my family

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Prof. Dr. Gülser Köksal for her continuous support, guidance, counselling, encouragement, and especially patience throughout this study.

I would like to express my appreciation to the rest of my thesis committee members.

I am grateful to my family for their endless support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Data mining is a well advanced field of study that helps data analysts in interpreting very large amount of data, and extracting interesting and useful information from them. Many data mining approaches have been presented in the literature. One of the most useful approaches is association rule mining. Association rule mining is searching the data to find the relationships and associations between different attributes of data (Narvekar et al., 2015). To find such associations and the rules between them, many algorithms and methods have been proposed and developed, each method has several advantages and shortcomings, and each one is useful for a specific application or a specific data type. Each algorithm mines some rules according to some defined interestingness measures without knowing the exact desired rules. Every run of these algorithms cause a large number of mined association rules which may contain many redundant rules. To overcome these problems some approaches are introduced. One of them is the concept of closed set of items which drastically reduces the rule set and helps in giving more abstract information (Zaki, 2000). Other approaches consider some other interestingness measures in addition to support and confidence (Brin et al., 1997; Fukuda et al., 1996; Nakaya et al., 1999; Padmanabhan et al., 1998). Some methods are also developed to group and prune redundant rules and get desired rules (Bayardo et al., 2000; Berrardo et al. 2007; Strehl et al., 1999; Ng et al., 1998; Srikant et al., 1997; Toivonen et al., 1995). But there also exist approaches that do the both job at the same time; they find the association rules while pruning the redundant ones (Vu et al., 2014).

Some comparisons of these algorithms have been performed in the literature by considering different measures and by testing the algorithms on various real and artificial data sets. Most of these comparison methods consider the algorithmic aspects of association rules (Hipp et al., 2000), or considering the running time of the algorithms. Some of them are also considering some common interestingness measures like support or confidence. In fact, to the best of our knowledge, no method in the literature considers all these criteria simultaneously and can select the best algorithm accordingly. Actually an important problem is that the user's interesting rules or in other words "true" rules are not known in any of these cases, so the comparisons are subject to inaccuracies or even errors.

In this thesis study, this comparison problem is addressed. For this purpose, we use a novel comparison approach proposed by Jabarnejad (2010), and develop and test it further to compare association rule mining algorithms in general. Use of a statistical experiment is proposed to compare the algorithms for different sizes and types of data as well as other factors such as number of attributes and support value. New performance measures are developed to compare the association rule mining algorithms, since those developed by Jabarnejad (2010) are appropriate only to compare the rule reduction algorithms. As there may be dependencies between some of these newly defined performance measures, and as we prefer to use measures that show different properties of the association rule mining algorithms, we propose to find and select independent measures, to the best we can, based on a factor analysis of the experimental results. Our comparison approach, then, proposes to perform hypothesis tests to find out if all of the algorithms have the same average performance or not. Such statistical comparisons of the algorithms may reveal that algorithms perform equally well for some or all selected comparison measures. Otherwise, if according to the statistical test results the algorithms seem to be different for at least two comparison measures, we propose the use of an appropriate Multi Criteria Decision Making approach such as PROMETHEE to compare the algorithms.

In order to demonstrate our comparison method, some association rule mining algorithms are selected among the ones commonly used in the literature for comparison (Hipp et al., 2000; Zheng et al., 2001; Margahny et al., 2006; Vu et al., 2014; Fournier-Viger et al., 2014) and also by considering availability of their software (Fournier-Viger et al., 2014; Borgelt, 2015).

The thesis is organized as the following: In Chapter 2, a literature review is given and a background about association rule mining algorithms and their comparison methods is provided for the thesis work. A review about some multi criteria decision making approaches including PROMETHEE is also provided in this Chapter. In Chapter 3, Jabarnejad's comparison method (Jabarnejad, 2010) is reviewed in detail, and true rules generation using sample regression model is covered. Then experimental data generation is described, comparison measures are developed, and the comparison approach is presented. Chapter 4 contains information about an application of the comparison method on the selected algorithms. Statistical analyses of these selected algorithms are also explained in this chapter as part of the proposed comparison approach. Conclusions and future work directions are provided in Chapter 5.

CHAPTER 2

LITERATURE REVIEW AND BACKGROUND

Association rules include important information for data interpreters, and many association rule mining algorithms are developed to extract these information for different applications. In the literature a brief description about association rules and the related algorithms is given, and comparisons done in various studies about these methods considering several comparison measures are provided. Furthermore, a background on PROMETHEE, one of the most effective multi criteria decision making approaches, is given.

2.1 Association Rule Mining and Rule Reduction Algorithms

Association rules are first introduced by Agrawal et al. (1993). An association rule shows a transaction in the form of $x \Rightarrow y$, in which x and y are two sets of items that do not share common items. These two sets are called an item set. In this kind of expression, x stands as an antecedent, and y stands as a consequent of the association rule. The goal of association rule discovery is to find these kind of associations among items from a set of transactions in data set. The most evident example of one association rule can be found in a market basket data set which shows the relation of two items; when someone buys a bread he will probably buy an egg. So a bread and an egg are an item set of this rule.

There are many algorithms developed for finding association rules. Most of them and especially the most basic ones are trying to find all association rules. This leads to finding many association rules that most of them may be redundant and not interesting for users. These algorithms work with some predefined measures or interestingness measures which help users in finding association rules. The most common used interestingness measures, are *support* and *confidence* that are defined according to user's interests. The support of an item set $x \Rightarrow y$ in the database D is defined as the percentage of transactions that contain $x \Rightarrow y$. It measures the generality of the rule. The confidence of ($x \Rightarrow y$) is the percentage of transactions in D containing x that also contain y. It measures the strength of the rule. The user defines the minimum thresholds for support and confidence and if the rule's support and confidence are above specified thresholds, it will be discovered by that association rule mining algorithm.

There are many algorithms trying to find the association rules by searching the data set and counting the support values of frequent item sets. These algorithms can be categorized into two approaches according to the search strategy they apply on data sets (Hipp et al., 2000). The first approach employs the breadth-first search (BFS) strategy, and the other one employs the depth-first search (DFS) strategy. The strategies work like this: if there are k-item sets in the data, BFS strategy counts support values of all (k - 1)-itemsets before counting the support values of the k-itemsets. Unlike this, DFS starts counting the support values of k-itemsets, and then proceeds to counting other support values recursively. The most well-known association rule mining algorithm is Apriori algorithm developed by Agrawal et al. (1993), which is the basic of many other algorithms developed later. It discovers all significant association rules in data sets by using the BFS strategy. AprioriTID and AprioriHybrid are extensions of the basic Apriori algorithm that were developed in order to improve some properties of it (Agrawal et al., 1994). Later Han et al. (2000) proposed FP-growth to mine the frequent itemsets. It works according to the DFS

strategy and frequent-pattern tree structure based on prefix-tree. FP-growth improves the efficiency of the mining process by avoiding the costly and repeated data scans, and mining a set of smaller tasks using partitioning-based method in order to reduce the search space. This causes the faster performance in comparison to Apriori. There are many developed algorithms introduced later for finding association rules like Eclat (Zaki, 2000), Charm (Zaki et al., 2002), Closet (Pei et al., 2000) which discover frequent itemsets. Recursive elimination, known as Relim (Borgelt, 2005), is also one of these algorithms for finding frequent itemsets. It works without applying prefix trees, processes the transactions directly, and performs the task of mining by using the simple recursive structure.

There are several software packages that apply these methods such as Weka (Weka, 2016) and SPMF (Fournier-Viger et al., 2014). SPMF, which is used for the application of association rule mining algorithms in this thesis, is a java open source data mining library which provides java codes of more than 100 data mining algorithms with a simple user interface for application purpose. The user defines some thresholds like the minimum support value according to the selected method, and gets the output in the form of text file. Different performance tests are provided in this source in order to evaluate the performance of SPMF.

Association rule finding algorithms are not enough by themselves to find interesting and desired rules, since they may find many redundant rules. In order to prune these redundant rules, some methods have been developed which help in delivering the interested results. This sometimes also happens by defining some more interestingness measures in addition to support and confidence. These methods and approaches are also known as rule reduction algorithms (Zaki, 2000; Brin et al., 1997; Fukuda et al., 1996; Nakaya et al., 1999; Padmanabhan et al., 1998; Bayardo et al., 2000; Berrardo et al. 2007; Strehl et al., 1999; Ng et al., 1998; Srikant et al., 1997; Toivonen et al., 1995).

2.2 Comparison Methods of Association Rule Mining Algorithms

By studying the comparisons of association rule mining algorithms published in the literature, it can be seen that they almost use the same measures such as execution time of algorithms on different data, or they look at usability of these algorithms in sparse or dense data sets, and by considering different support values defined by user.

Hipp et al. (2000) deals with the algorithmic aspects of association rule mining algorithms. In their work, the performance analyses are done using both runtime experiments and theoretic considerations. In this work, three important algorithms, namely Apriori, Eclat, and Partition have been compared, and although they have employed different strategies, runtime behavior is found similar for them in the performed experiments.

Zheng et al. (2001) compares five well-known algorithms (Apriori, FP growth, Closet, Charm, and MagnumOpus) for their running time and by considering different support values, on several real and artificial data sets. The results showed that FP-growth has the best performance in running time. It also showed that new algorithms like FP-growth and Charm are much faster than Apriori. However, Apriori is faster than others for high minimum support in these experiments. Also on the real dataset, for minimum support, FP-growth is better than Apriori. It also shows that the algorithm selection is mostly dependent on the support value.

Margahny et al. (2005) compares Apriori, Eclat, and FP-Growth according to the number of data scans and also data structures, and at last develops a method to address the deficiency of these algorithms. It can be seen that FP-growth is better than the others, since it has showed less number of data scans in these experiments.

Vu et al. (2014) also compares the running time of three well-known algorithms (Apriori, Eclat, and FP-Growth) on sparse and dense data sets. According to the results, each algorithm shows different performance on different data types. Eclat performance is the best on dense data, while FP-growth has the fastest run on the sparse data. Apriori shows the weakest performance from the point of support with regard to other mentioned methods.

Fournier-Viger et al. (2014) also compares the running time of some well-known algorithms. Execution time of several important algorithms including Apriori, FP-growth, and Relim on dense and sparse dataset samples have been compared. FP-growth shows the best performance for both execution time and memory usage measures.

There is also a novel comparison method proposed by Jabarnejad (2010), which enables data analysts to precisely evaluate the performance of different rule reduction methods on controlled data sets for which true rules are known. This method is used and extended in this thesis study.

2.3 Comparison Measures

In the literature certain measures are used to compare the algorithms. The execution time is one of them, which is used in almost every comparison study. Interestingness measures which play an essential role in association rule mining in order to find the desired rules according to user's interest, can also be used as comparison measures. There are many studies in the literature, which deal with interestingness measures to find the best rules especially in the post processing step of association rule mining. The most well-known and classical measures to characterize association rules are

support and confidence. Jimenez et al. (2013) defines interestingness measures for standard association rules. These include support, confidence, lift and conviction. Omiecinski (2003) introduces three metrics according to the rules confidence to find the interesting rules. McGarry (2005) divides the measures of interest into subjective and objective measures, and the characteristics of them have been discussed in his work. In his work, objective criteria such as rule coverage, rule complexity, rule confidence and rule completeness are often used as a measure of the interestingness of the discovered rules. Geng et al. (2006) also provides the list of probability based objective interestingness measures for rules in which some privilege measures like support, confidence, and lift can be seen. Bramer (2007) also uses confidence, support and completeness as three main and common measures. There are also other works like Vo et al. (2011) that make use of these common interestingness measures. In Choi et al. (2005) business values of rules are discussed according to three categorizations. Tan et al. (2002) provides a comparative study according to certain attributes and an original approach to the selection of measures by an expert. Later, Lenca et al. (2008) has completed this work by providing the list of some important and different measures which help users to find the best rules. This measures list is developed by defining some attributes which help users to select interesting and important measures by the means of PROMETHEE approach. The list of the most preferable measures determined by PROMETHEE includes BF, CONV, CENCONF, LOE, and CONF.

2.4 PROMETHEE and Other MCDM Approaches

In many problems, several objectives or criteria should be considered at the same time to find the desired results and solve the problems. These problems are handled in the domain of multi criteria decision making (MCDM), and they can be solved using methods available in this area. The study of the association rule mining methods considering several performance measures defined for comparing purpose,

can be considered as one of these multi criteria decision making problems since different algorithms may behave differently for different criteria, and the best method selection is not the easy challenge for the users in many cases.

Many MCDM approaches make decision making process easier. Analytic Hierarchy Process (AHP) (Saaty, 1988) is one of the most well-known methods in this area, which is helpful in finding the most desirable alternative solution considering several independent criteria. Founder of AHP, Saaty, says "Many decision problems cannot be structured hierarchically because they involve the interaction and dependence of higher-level elements on lower-level elements. Not only does the importance of the criteria determine the importance of the alternatives as in a hierarchy, but also the importance of the alternatives themselves determines the importance of the criteria. Feedback enables us to factor the future into the present to determine what we have to do to attain a desired future" (Saaty, 2000). In such cases, Saaty (2000) proposes to use Analytic Network Process (ANP). The Analytical Network Process (ANP) is a generalization of the Analytic Hierarchy Process (AHP) and developed by Saaty (2000) which deals with problems with interdependent elements, and it works with constructed network structure. ANP does pairwise comparisons by asking several questions to the decision maker.

PROMETHEE (Preference Ranking Organization Method for Enrichment Evaluation) is another MCDM approach introduced by Brans (1985). Priorities of alternatives under multiple criteria can be evaluated by this method. PROMETHEE assumes criteria are independent of each other. Therefore in the case of dependent criteria, some researchers prefer finding weights of criteria by ANP and then using PROMETHEE for ranking the alternatives (Anaklı, 2009; Tseng, 2009; Barve et al., 2015; Sakthivel et al., 2015). PROMETHEE has several steps which is briefly covered here (Anaklı, 2009).

Step 1: Data matrix is constructed and notation is:

A: Alternatives

K: Set of criteria (the criteria indices)

F: Real valued criteria

W: weight of criteria (relative importance of criterion f_k, which can be obtained using ANP Method for dependent criteria, and AHP Method for independent criteria.)

Step 2: preference functions, which represent the intensity of the preference of one alternative over another, are determined for each criterion according to the properties of each criterion. Preference function has to be a non-decreasing function and its value equals to zero for negative values of d.

Then we define $d = f_k(A1) - f_k(A2)$ for all criteria and for all pairwise alternative comparisons.

We can also define $P_k(A_1,A_2) = p(f_1(A_1) - f_2(A_2))$.

After choosing the preference function and calculating d values, next step is determination of corresponding parameters for preference function by asking several questions to decision maker. Then according to these values and defined preference function, P_k is calculated for all pairwise comparisons.

Step 3: multi – criteria preference index, $\prod$, is calculated as:

$$\prod (A_1, A_2) = \frac{\sum_{j=1}^{k} w_j P_j(A_1,\ A_2)}{\sum_{j=1}^{k} w_j}$$

$\prod (A_1, A_2)$ represents the decision maker's preference intensity of alternative A_1 over A_2 by considering all sub criteria at the same time.

$\prod = 0$ means weak preference of alternatives, and $\prod = 1$ means strong preference of alternatives.

Using above formula, $\prod$ values are calculated for pairwise comparisons of all alternatives and then gained the overall comparison of all alternatives by considering all criteria.

Step 4:

Then leaving and entering flows for each alternative are defined, like:

$\Phi^{-}(A_1) = \sum_{i \in I} \prod(A_{i,}, A_1)$ Entering flow of alternative 1

$\Phi^{+}(A_1) = \sum_{i \in I} \prod(A_{1,}, A_i)$ Leaving flow of alternative 1

$\Phi(A_1) = \Phi^{+}(A_1) - \Phi^{-}(A_1)$ Net flow of alternative 1

Step 5:

Complete preorders are determined in this step by comparing the net flow values. So, Priorities can be determined by this way, and the best association rule mining algorithm will be selected.

CHAPTER 3

ASSOCIATION RULE MINING ALGORITHM COMPARISON METHOD

A new comparison method based on regression model, statistical analyses, and MCDM approaches is developed. This method applies the comparison process between different association rule mining algorithms considering the real interesting rules defined by the user using regression model. Several steps are defined for this comparison approach including defining the real interesting rules or true rules, generating experimental data, defining comparison measures, and finally comparing algorithms using the values of comparison measures and different analyses.

3.1 Overview of the Method

The idea of the comparison method proposed by Jabarnejad (2010) is to intentionally generate sample data considering several factors with evident interesting association rules (true rules), then apply different association rule mining algorithms on the generated data, and finally evaluate the performance of the applied association rule mining algorithms based on some measures. Data consist of independent and random variables (in his thesis work some process variables and one failure status). In this method, a logistic regression model is used to simulate the failure incidence of, say, a manufacturing system. The power of this method is that, unlike other comparison methods, we know the true rules expressed in the form of a logistic regression model.

Therefore, we can compare the rules derived by the algorithms with the true rules. Jabarnejad (2010) compares rule reduction algorithms. We extend this method to the case where we can compare association rule mining algorithms, in general. For this purpose, we revise the comparison measures, develop new ways of generating test data including sparse and dense data. We use some statistical analyses in the proposed comparison approach. We also improve the way we compare the algorithms under multiple criteria or measures.

3.2 Generation of True Rules

After selecting the algorithms and defining the comparison measures, we need several data sets to perform our comparison. For this purpose, we generate some artificial data with known rules which we call true rules, apply selected algorithms on them, and finally measure performance of each algorithm on each data set using a set of comparison measures. For generating these data, we basically use the approach of Jabarnejad (2010), which is explained here briefly. According to this approach, some independent binary variables $x_1, x_2, \ldots, x_n$ representing, say, manufacturing process variables and one binary variable z representing the failure status of the manufacturing system are defined. Then a logistic regression model is used to predict the probability of not observing the failure event. In order to explain the true rule's generation process clearly, one sample regression model can be defined as in Equation (3.1).

$$y = 2.999 - x_1 - x_2 - x_3 - x_4 - x_5 + \varepsilon \qquad (3.1)$$

For our proposed model, we consider five process variables which have significant effects on the failure event. These are effective process variables which are independent of each other, and have uniform distributions. Value of a process

variable is taken as 1, if it is available and active in the system, otherwise zero. For this model, we also consider that all these five process variables have negative effect on logit *y,* so their coefficients are considered to be −1 in the given logistic regression model. According to this model, logit *y* is calculated using Equation (3.1) in which ε , representing the error, is supposed to have a normal distribution with mean 0 and variance 0.1. The constant term of the model which can be considered as the initial effect of the setup on the system, is taken as 2.999 to obtain a clear failure status as explained below.

The independent process variables are assigned random values according to a designed experiment, ε is assigned a random value, and from Equation (3.1) the corresponding logit (y) value is obtained. Probability of not observing the failure, *f*(*y*), is calculated after placing the *y* value in Equation (3.2)

$$f(y) = \frac{e^y}{e^y+1} \tag{3.2}$$

Then the failure status is determined by Equation (3.3):

$$z = \begin{cases} 1 \text{ if } f(y) < 0.5 \\ 0 \text{ if } f(y) > 0.5 \end{cases} \tag{3.3}$$

If $z = 1$ there is a failure, and if $z = 0$ no failure occurs. When $f(y) = 0.5$, failure status remains at a borderline. To avoid this situation, the constant term in Equation (3.1) is selected as slightly different than 3.

In this thesis, we use the logistic regression model of Equation (3.1). The sample generated data with four runs is given in Table 3.1.

Table 3.1 Sample Generated Data with Four Runs

x_1	x_2	x_3	x_4	x_5	z
0	0	0	0	1	0
0	0	1	0	0	0
0	0	0	1	0	0
1	1	0	0	1	1

Since there are two levels for each of the five process variables in the defined model, there are 32 possible combinations of the process variable values. The resulting expected logistic function values for all possible combinations of the process variables are listed in Appendix A. In the table provided in appendix A, $f'(y)$ shows the probability of observing the failure, which is $f'(y)=1-f(y)$, and $f(y)$ shows the probability of observing the success.

Every one of these 32 combinations can be considered as an association rule. When association rule mining algorithms are applied on these generated data, we have two consequents which are failure probability $f'(y)$, or success probability $f(y)$. So we have 64 association rules with defined confidences. The confidences of rules, with the success event as consequent, are consistent with the probabilities of not observing a failure event or $f(y)$. Similarly, the confidences of rules, with the failure event as consequent, are consistent with the probabilities of observing a failure event or $f'(y)$. We assume the data analyst is interested in failure event with minimum confidence of 50%, as we have 32 rules consistent with failure event and just 16 of them have confidence more than 50%, these 16 association rules become important failure association rules. In this thesis, we call them as true rules which are listed in Table 3.2. These true rules will be considered as a benchmark to analyze the association rule mining algorithms.

Table 3.2 True Rules

	x_1	x_2	x_3	x_4	x_5	y	$f(y)$	**Confidence**
1	0	1	0	1	1	-0.0010	0.4997	0.5002
2	1	0	1	0	1	-0.0010	0.4997	0.5002
3	1	0	1	1	0	-0.0010	0.4997	0.5002
4	1	1	0	1	1	-1.0010	0.2687	0.7312
5	1	1	0	0	1	-0.0010	0.4997	0.5002
6	1	1	1	1	0	-1.0010	0.2687	0.7312
7	0	0	1	1	1	-0.0010	0.4997	0.5002
8	0	1	1	0	1	-0.0010	0.4997	0.5002
9	1	1	1	0	1	-1.0010	0.2687	0.7312
10	1	1	0	1	0	-0.0010	0.4997	0.5002
11	1	1	1	0	0	-0.0010	0.4997	0.5002
12	1	0	0	1	1	-0.0010	0.4997	0.5002
13	0	1	1	1	0	-0.0010	0.4997	0.5002
14	0	1	1	1	1	-1.0010	0.2687	0.7312
15	1	0	1	1	1	-1.0010	0.2687	0.7312
16	1	1	1	1	1	-2.0010	0.1190	0.8809

3.3 Design of Experiments and Generation of Experimental Data

By applying the regression analysis method explained in the previous section, we can generate artificial data with known rules, and by defining desired confidence level and thresholds, we can determine true rules. By doing this, we will know interesting rules and use them as our benchmark in evaluating the performance of applied association rule mining algorithms. As mentioned above, we obtained 16 true rules shown in Table 3.2. Jabarnejad (2010) in his thesis work defines four true rules and uses them in comparing two grouping and pruning methods for association rules, and the method is good, if it does not prune these rules and also if it does not keep other rules except them as much as possible. As a result, important failure rules, namely, true rules can be used to measure the performance of rule reduction methods from the view of information loss. In this thesis, we extend this method and evaluate

the association rule mining algorithms to see if they give us these true rules, and how they are efficient in giving the less number of redundant rules and more of the true rules. We also evaluate their performance on several generated data sets based on some performance measures.

In this study, in order to have better and more realistic comparison results, we propose to apply the comparison method on data sets generated according to a statistically designed experiment taking into account properties of real data sets. In the proposed experimental design, the following factors are taken into account: Data type (DT) (sparse, dense), number of attributes (NA), size of data (number of runs of the desired regression model) (NR), support value (SV). These different data sets can be generated using the logistic regression based method explained in Section 3.2.

Let us define these factors in detail, and the way they are defined in generated data. Data type (DT) is one of the most important factors which should be considered when applying different association rule mining algorithms. As it is mentioned in the literature, many association rule mining algorithms have different results on different data sets and in order to understand and compare them better, they should be applied on different data types. Sparsity and density show data type which may have different definitions according to their application areas. In numerical analysis, sparsity describes the percentage of cells in a database table that most of the elements are zero. By contrast, density shows the percentage of cells in a database table that most of the elements are nonzero. The fraction of zero elements over the total number of elements in a matrix is called the sparsity (Tewarson, 1973). By using these definitions, we have calculated sparsity considering the 0 and 1 elements in our generated data sets; if sparsity is below 45, we have considered that data set as dense, otherwise sparse. So we have considered two levels for Data type (DT): sparsity < 45 (D) or sparsity > 45 (S).

Number of attributes (NA) is also one important factor which have been considered in generating data sets. As mentioned in part 3.2, some process variables should be defined for suggested regression model in order to generate association rules. For our model we have considered five process variables or attributes which are effective in failure / success events. However, in real data, we commonly encounter with many process variables that make data more complex, and due to this, many redundant association rules are mined that makes the role of association rule mining algorithms more important in finding true rules and also finding the least number of redundant rules. So in order to make our artificial data show the characteristics of real data, we add some other ineffective process variables in the model, which do not have any effect on failure / success event, but make artificial data more real and complex. Some of these variables are assigned randomly, and some of them can take values dependent on other effective process variables' values. Two kinds of these attributes have been considered in our model. The first type is dependent on other effective process variables, and can be defined as:

If $x_1 + x_2 + x_3 + x_4 + x_5 \geq 3$, then $x_6 = 1$, otherwise $x_6 = 0$.

This variable shows that if at least 3 effective process variables have the value of 1, then it will be 1. Otherwise, it will be 0.

Second type ineffective attribute also does not have any effect on failure/success event, and also it is not dependent on other effective process variables. The value of this variable is assigned randomly as 0 or 1.

Two levels have been considered for number of attributes (NA) factor, 7 or 14, in which 7 attribute level shows data set having 5 effective and 2 ineffective attributes, and this data set may not reflect the real data sets completely. In contrast, 14 attributes may reflect real data sets better, by considering much more ineffective attributes as defined above.

One other factor that is defined for generating data sets is data size (NR), which shows the number of runs of the suggested regression model. 100 or 10000 tuples is considered for data generation, the first one shows a small data set, the second level shows a big data set.

As mentioned in literature, support value is one of the most effective factors the user determines. As mentioned, different association rule mining algorithms show different performances with different support values. Support value (SV) is not effective in generating the data sets, but it is the threshold we define when applying association rule mining algorithms on generated data. Since it is important in performance of association rule mining algorithms, we consider it as one factor in experimental designs. Two levels are defined for it: 0.05 or 0.20. The first one is considered as a low support value, and the second one as a high support value. The reason of setting low support value as 0.05 is that, it causes mining approximately all association rules by selected algorithm. And, the reason for selecting 0.20 as high support value is that, setting higher values cause approximately no rule mining. A Matlab code is developed for generation of the suggested data sets.

In creating different data sets, we consider DT, NA and NR. SV is not important for generating different data sets, because support values are just considered in applying different algorithms on data sets. Hence, we have $2\times2\times2=8$ different data sets. We can compare any number of algorithms on these data sets, and with two different support values. For example, if we compare three algorithms, this leads to a full factorial design with $8\times2\times3=48$ different experiments to conduct. But we can also use a fractional factorial design instead of the full one (Hedayat et al., 2012). Since we need a comprehensive comparison considering all defined factors and levels, we use a full factorial design. The full factorial design for collecting data to compare three algorithms using selected factors and levels is available in Appendix B.

We need to run each association rule mining algorithm on these designs, and totally for three association rule algorithms, we have 48 experimental runs.

For having better results, we suggest replicating generating datasets at least twenty times, therefore 960 experiments to be performed.

3.4 Comparison Measures

We have used Tan et al. (2002) and Lenca et al. (2008) in the selection of interestingness measures, since they have provided a complete evaluation of these measures used in previous works, and ranked them according to user's interest by using multi criteria decision making approaches. The interestingness measures selected are shown in Table 3.3. In this selection, we have tried to choose commonly used and different ones that constitute a complete set.

Their absolute and relative definitions are available in Table 3.4, based on the following explanations and notation.

Given a rule A→ B, define

$r1 = n_{ab}$= the number of records satisfying both A and B (the examples of the rule),

$r2 = n_b$= the number of records satisfying B**,**

$r3 = n_{\bar{b}}$= the number of records not satisfying B**,**

$r4 = n_{a\bar{b}}$= the number of records satisfying A but not B (the counterexamples of the rule),

$r5 = n_a$= the number of records satisfying A,

$r6$ = n = the total number of records,

r7= the number of records satisfying both A and B, and also including other redundant attributes in antecedent and consequent

P_a = probability of observing *a*

P_b = probability of observing *b*

$P_{\bar{b}}$ = probability of not observing *b*

$P_{a\bar{b}}$ = probability of observing a but not *b*

$P_{a/b}$ = probability of observing *a* if *b* is available in the consequent of a rule

$P_{a/\overline{b}}$ = probability of observing *a* if *b* is not available in the consequent of a rule

$P_{b/a}$ = probability of observing *b* if a is available in the antecedent of a rule

For the case of true rules, A is the antecedent of the true rules, and B is the failure event.

We have defined 7 rules as defined by notations (*r*1, *r*2, …, *r*7) for true rules, using a Matlab code, for calculating the above mentioned interestingness measures.

Table 3.3 Selected Interestingness Measures

Measure name	**Abbreviation**
Bayes Factor	BF
Confidence	CONF
Conviction	CONV
Lift	LIFT
Loevinger	LOE

Table 3.4 Absolute and Relative Definitions of the Selected Measures (Lenca et al., 2008)

	Absolute definition	**Relative definition**
BF	$\frac{\boldsymbol{n_{ab}n_{\bar{b}}}}{\boldsymbol{n_b n_{a\bar{b}}}}$	$\frac{P_{a/b}}{P_{a/\bar{b}}}$
CONF	$\frac{\boldsymbol{n_{ab}}}{\boldsymbol{n_a}}$	$P_{b/a}$
CONV	$\frac{\boldsymbol{n_a n_{\bar{b}}}}{\boldsymbol{n n_{a\bar{b}}}}$	$\frac{P_a P_{\bar{b}}}{P_{a\bar{b}}}$
LIFT	$\frac{\boldsymbol{n n_{ab}}}{\boldsymbol{n_a n_b}}$	$\frac{P_{b/a}}{P_b}$
LOE	$\frac{\boldsymbol{n n_{ab} - n_a n_b}}{\boldsymbol{n_a n_{\bar{b}}}}$	$\frac{P_{b/a} - P_b}{1 - P_b}$

We have customized these interestingness measures according to the true rules defined in our proposed method. Hence, we define the following five interestingness measures to be used in the comparison:

- $\mathbf{M_1} =$

$$\mathrm{BF}' = \frac{\text{probability of the availability of antecedents of true rules if failure exist}}{\text{probability of the availability ofantecedents of true rules if success exist}}$$

- $\mathbf{M_2} =$

$$\mathrm{CONF}' = \text{probability of failure if antecedents of true rules exist}$$

- $\mathbf{M_3} =$

$$\mathrm{CONV}' = \frac{\text{probability of the availability of antecedents of true rules} * \text{probability of success}}{\text{probability of success in availability of antecedents of true rules}}$$

- $\mathbf{M_4} =$

$$\text{LIFT}' = \frac{\text{probability offailure if antecedents of true rules exist}}{\text{probability of failure}}$$

- $\mathbf{M_5} =$

$$\text{LOE}' = \frac{\text{probability of failure if antecedents of true rules exist} - \text{probability of failure}}{\text{probability of success}}$$

These measures are calculated for true rules presented with our logistic regression model for each association rule mining algorithm.

We interpret these measures from statistical viewpoint (as suggested by Omiecinski, 2003; Tan, 2004; Hahsler, 2016) as follows.

Bayes factor (BF) is the degree to which we favor one hypothesis over another. For example, for the case of true rules, we want to know to what degree observing the antecedents of true rules in failure events is favored to observing the antecedents of true rules in success events. If BF is greater than 1, since our defined true rules have failure in their consequents, it means that the data favor true rules, so the measure is good and desirable for this experiment. Similarly, if BF is smaller than 1, this means that the algorithm does not perform well regarding the BF measure and the case this happened. BF ranges between $[0, \infty]$.

Confidence (CONF) also shows the proportion of the transactions that contains antecedent of the rule which also contains consequent of the rule. For the case of defined true rules, confidence means the probability of observing the true rules by the algorithm. So, if the CONF is close to 1, the algorithm is more efficient in finding desired true rules. CONF ranges between [0, 1].

Conviction (CONV) is supposed as an alternative to confidence which is not sufficient to capture the direction of associations. Conviction can be interpreted as the ratio of the expected frequency that antecedent of the rule occurs without its consequent. In other words, it is the frequency that the rule makes an incorrect prediction. For the case of true rules, CONV can be interpreted as the probability of not observing the true rule. It also ranges between [0, 1].

Lift is another interestingness measure which considers the dependency degree of the antecedent and consequent of the rule. It measures how many times more often antecedent and consequent of the desired rule occur together than expected if they were statistically independent. For the case of true rules, it shows the occurrence of true rules by considering the statistically dependency of the antecedent and consequent of the true rules. Lift ranges between $[0, \infty]$. When it becomes 1, it means antecedent and consequent are independent.

Loevinger (LOE), which is also known as certainty factor, is a measure of variation of the probability that consequent of the desired rule is in a transaction when only considering transactions with antecedent of that rule. An increasing LOE or a negative LOE shows the decrease of the probability that consequent is not in a transaction that antecedent is in. In the case of true rules, an increasing or negative LOE shows a high probability of observing true rules when applying the desired association rule mining algorithm. LOE ranges between [-1, 1].

Besides these five interestingness measures, we propose to use two other measures which can be more helpful in evaluating the strength of association rule mining algorithms in finding interesting and also non redundant rules.

If we define $r1$ and $r7$as

$r1$= the number of all mined true rules

$r7$= the number of all mined true rules and all other true rules including redundant attributes

Then, $\frac{r1}{r7}$ shows the ratio of true rules without redundant elements to true rules including redundant attributes. It ranges between [0, 1]. When it becomes 1, it means that all the found true rules have non-redundant elements. If it is 0, it means that the algorithm could not find any non-redundant true rule, and this is the worst case.

$\frac{n_{ab}}{n_b}$, is another measure that shows the ratio of true rules to all rules including failure event. It also ranges between [0, 1], and the value 1 means that all the rules including the failure event in their consequent are true rules, and this is the best case.

Besides the above measures, we include two other measures that most of the comparison studies in the literature have used them in their works. These two measures are, Time and memory use. We get the values of them from the software we use for applying the selected algorithms on generated data. As it is evident, Time shows the execution time it takes for the algorithm to find the association rules on the generated data. Memory use shows the memory used by the applied algorithm in finding the rules.

A complete list of all the measures proposed for use in comparing the association rule mining algorithms is available in Table 3.5.

Table 3.5 Complete List of All Comparison Measures

Measure name	**Measure name**
BF	$\frac{r1}{r7}$
CONF	$\frac{n_{ab}}{n_b}$
CONV	Time
LIFT	Memory use
LOE	

3.5 Preference Functions

As mentioned, in the second step of PROMETHEE, preference functions are to be determined for each criterion according to the properties of the criterion. There are six main types of preference functions including Usual, U-shape, V-shape, Level, Linear, and Gaussian (Brans et al., 1985). In order to select one of these preference functions for each criterion, we need to define the criteria and relevant objective functions properly. Then related parameters for each preference function are determined by asking several questions to decision maker and also using some statistical analyses.

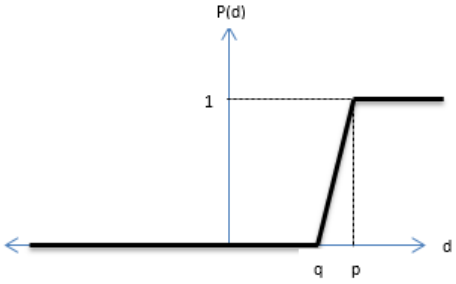
The V-shape and Linear preference functions are appropriate for quantitative criteria (Brans et al., 1985; Behzadian et al., 2010).

Since our defined comparison measures are all quantitative criteria, the Linear or V-shape preference functions are suitable. So, according to the pairwise comparisons of algorithms as mentioned in the next section, the preference and indifference parameters are determined for each criterion. For each criterion defined in this thesis, if the decision maker decides not to define an indifference parameter for a criterion, V-shape function can also be chosen. Otherwise the Linear preference function is

appropriate. These preference functions need to be calibrated according to the statistical analyses.

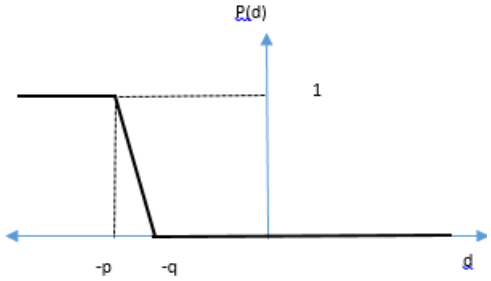
Since indifference parameter is important in defined comparison measures, two different shapes of linear preference function are used for selected comparison measures according to the definitions and objective functions. The proposed preference function and its formula for BF, CONF, LIFT, and LOE measures are as defined in Figure 3.1.

The proposed preference function and its formula for CONV, Time, and Memory use measures are as defined in Figure 3.2.



$$P(d) = \begin{cases} 0, & q \geq d \\ \dfrac{d-q}{p-q}, & q < d \leq p \\ 1, & p < d \end{cases}$$

Figure 3.1 Preference Function for BF, CONF, LIFT, and LOE Measures



$$P(d) = \begin{cases} 0, & d > -q \\ \dfrac{-d-q}{p-q}, & -p < d \leq -q \\ 1, & d < -p \end{cases}$$

Figure 3.2 Preference Function for CONV, Time, and Memory Use Measures

3.6 Comparison Method

The first step in comparison of a set of association rule mining algorithms is to collect comparison measure data by applying the algorithms on the data sets generated according to the experimental design and the logistic regression model explained in Sections 3.3 and 3.2, respectively.

In comparing association rule mining algorithms and identifying the most favorable ones based on multiple criteria, it is preferred to consider an independent and complete set of evaluation criteria. Some of the criteria defined in the previous section may be correlated with each other for a given set of association rule mining algorithms (alternatives) and data sets they are applied on. Therefore, before proceeding to a comparison, a factor analysis of the collected data is suggested to be performed to identify highly correlated measures. It is advisable to choose a single measure in each group of highly correlated measures loaded under a factor, for use in the overall comparison. In choosing these measures, we can utilize matrix plots to observe types of dependencies (linear or nonlinear) between pairs of these measures. The measures distributed independently from the others should be favored.

The next step is testing the set of hypotheses that all algorithms perform the same on the average or not with respect to the selected comparison measures, separately. For this purpose, ANOVA of the collected measure data can be performed. The ANOVA model should consider the main effects of data size (NR), sparsity (DT), number of attributes (NA), and support value (SV) as blocking variables, and the algorithm as the main variable we are interested in. If ANOVA assumptions that errors are distributed normally with a constant variance are not satisfied, an appropriate data transformation can be tired. If this does not help satisfying the error assumptions, then a nonparametric hypothesis test alternative such as Friedman test can be used. The Friedman test may consider a main factor (the algorithm), and also a blocking

variable. The blocking variable can be a combined one of some or all of the variables NR, DT, NA and SV. For this purpose, we can investigate plots of measure values of the algorithms versus the variables NR, DT, NA and SV. If these plots indicate that performances of the algorithms change to a considerable extend with some or all of these blocking variables, then a new blocking variable can be identified by combining the influential variables in such a way that the levels of the combined variable correspond to the tested combinations of the levels of the individual variables.

Unless the mean performance of at least one algorithm is different than those of the others in statistical sense for at least one comparison measure, we conclude that the algorithms perform equally well. Otherwise, we identify for how many measures the algorithms seem to be different. If algorithms seem to be different for only one measure, we conclude based on the statistical test results (parametric or non-parametric). If the algorithms seem to be different for at least two measures we can use a combination of AHP/ANP and PROMETHEE to compare them.

PROMETHEE requires identification and use of preference function values in comparing pairs of algorithms for each of the selected comparison measures. If the difference between a given pair of algorithms is not statistically significant according to a certain measure under consideration, then the preference function value corresponding to that difference is advised to be taken as zero. In order to help the assessment of preference function values of such differences, we can perform hypothesis tests of equality of means of each and every possible pair of algorithms (using again the blocking variables) for each and every comparison measure selected. Since this requires, for each comparison measure and n algorithms, n (n-1)/2 tests, type I errors of them might add up to an undesirable amount. In order to overcome this problem, we can use a low significance level, α value, for each of these pairwise tests. For all practical purposes, $\alpha = 2\times0.10/ (n (n\text{-}1))$ can be used.

As it is explained in Section 2.4, PROMETHEE consists of several steps. The first step is data matrix construction, and for this step weights of the comparison criteria need to be identified. These weights can be found by using AHP, if the criteria are independent of each other. In spite of the use of factor analysis results in selecting the criteria, if the selected criteria are believed to have considerable dependencies, then one can identify their weights by using ANP.

At the second step of PROMETHEE, preference functions are determined separately for the criteria. In Section 3.5, certain preference function types are suggested for the comparison measures. Here, these functions need to be calibrated for the selected measures based on the pairwise comparison of the algorithms. For example, if the difference of means of any two algorithms is found statistically insignificant, then the preference function value corresponding to the absolute difference between their means (and any lower difference value) can be taken as zero or close to zero. The other shape parameters of the preference functions such as the difference corresponding to a maximum preference value can be identified again by considering the maximum difference value observed in the data and also by using the expert knowledge about the algorithms and measures.

At the third step, a multi-criteria preference index is calculated for each pairwise comparison. According to these results, entering and leaving flows are defined for each alternative. Finally at the last step, priorities are determined by comparing the net flows of all alternatives.

We can use PROMETHEE in two separate ways for comparing the algorithms. First, we can consider algorithms as alternatives, and compare them comprehensively considering all the criteria and all data types, as explained above. Second, we can compare the algorithms under specific levels of the blocking variables (data size, sparsity or data type, number of attributes, and support value) separately. In the latter case, alternatives can be considered as algorithms under specific levels of the

blocking variables. For example, it is possible to compare algorithms to each other only for the cases of small and sparse data with small number of attributes and small support value. Similarly, it is possible to compare algorithms under certain levels of the blocking variables to those under certain other levels of the blocking variables. Such comparisons can be done in a similar manner as explained above for the overall comparison of the algorithms.

CHAPTER 4

APPLICATION OF THE METHOD

In this chapter, use of the comparison method is demonstrated on some selected association rule mining algorithms. Use of the method in comparison of other algorithms are discussed.

4.1 Selected Algorithms for Application

In order to demonstrate the comparison method, the following association rule mining algorithms are selected among the ones used in the literature for comparison: Apriori (Agrawal et al., 1994), FP-growth (Han et al., 2000) and Relim (Borgelt, 2005). In this selection, availability of their software (Fournier-Viger et al., 2014) is considered besides their being subjects to a comparison in the literature (Zheng et al., 2001; Margahny et al., 2006; Vu et al., 2014; Fournier-Viger et al., 2014), showing different performance to some extent.

4.2 Design of Experiments, Experimental Data, True Rules

We have used the full factorial design described in Chapter 3 and available in Appendix B to collect the data needed for comparison of the algorithms. Other than the algorithm and support value, we have three factors (DT, NA, NR) that can be

used in generating the data sets. As explained in Chapter 3, there are 8 possible combinations of these factors each at two levels.

For each combination, say data generator, we have generated 20 data sets as replicates. Each data set generated has a different sparsity value, which we categorize into low and high levels by considering the 0.45 threshold for sparsity. The complete list of these sparsity values obtained for the replicates is given in Table 4.1.

Table 4.1. Sparsity Values for 20 Replications

Data Index	**rep1**	**rep2**	**rep3**	**rep4**	**rep5**	**rep6**	**rep7**	**rep8**	**rep9**	**rep10**
1	0.56	0.51	0.52	0.52	0.52	0.52	0.52	0.55	0.54	0.51
2	0.56	0.52	0.50	0.49	0.51	0.51	0.52	0.50	0.52	0.51
3	0.40	0.36	0.39	0.37	0.39	0.38	0.38	0.38	0.40	0.40
4	0.31	0.27	0.26	0.30	0.25	0.27	0.26	0.28	0.29	0.27
5	0.54	0.51	0.51	0.51	0.52	0.51	0.51	0.51	0.51	0.52
6	0.52	0.51	0.51	0.51	0.50	0.51	0.51	0.51	0.50	0.51
7	0.42	0.39	0.40	0.38	0.39	0.39	0.39	0.39	0.39	0.39
8	0.29	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27
Data Index	**rep11**	**rep12**	**rep13**	**rep14**	**rep15**	**rep16**	**rep17**	**rep18**	**rep19**	**rep20**
1	0.51	0.52	0.49	0.49	0.48	0.50	0.50	0.52	0.54	0.50
2	0.50	0.52	0.51	0.51	0.49	0.51	0.52	0.51	0.52	0.51
3	0.38	0.39	0.37	0.37	0.37	0.38	0.36	0.37	0.41	0.40
4	0.28	0.29	0.25	0.30	0.29	0.29	0.29	0.26	0.28	0.28
5	0.51	0.51	0.51	0.51	0.52	0.51	0.52	0.51	0.52	0.51
6	0.51	0.51	0.50	0.50	0.51	0.50	0.51	0.51	0.50	0.51
7	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39
8	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27	0.27

Selected algorithms have been run on each and every replicate data set, using the java code provided in SPMF website (Fournier-Viger et al., 2014), according to the experimental design provided in Appendix B.

4.3 Results for Comparison Measures

We have calculated the comparison measures provided in Chapter 3, based on the collected data, for Apriori (Algorithm 1), FP-growth (Algorithm 2), and Relim (Algorithm 3). A Matlab code has been developed and used for this purpose. The sample results of the calculated measures for one replication are shown in Table 4.2.

Table 4.2 Results for Comparison Measures for Some Experiments

	Alg.	NR	DT	NA	SV	BF	CONF	CONV	LIFT	LOE	n_{ab}/n_b	r1/r7	Time	Memory
1	1	1	1	1	1	1.16	0.46	0.54	2.13	0.83	0.16	0.40	20	12.44
2	1	1	1	1	2	*	*	*	*	*	0	*	2	20.32
3	1	1	1	2	1	1.61	0.47	0.91	1.76	0.43	0.01	0.05	175	25.16
4	1	1	1	2	2	*	*	*	*	*	0	*	6	27.38
5	1	1	2	1	1	1.07	0.47	0.55	1.94	0.80	0.15	0.33	45	8.38
6	1	1	2	1	2	*	*	*	*	*	0	*	2	10.6
7	1	1	2	2	1	1.42	0.46	0.88	1.60	0.37	0.01	0.05	86	14.94
8	1	1	2	2	2	*	*	*	*	*	0	*	5	23.51
9	1	2	1	1	1	1.10	0.47	0.56	1.94	0.77	0.17	0.36	159	9.87
10	1	2	1	1	2	*	*	*	*	*	*	*	41	17.24
11	1	2	1	2	1	1.78	0.47	1.03	1.72	0.36	0.01	0.07	2120	30.25
12	1	2	1	2	2	*	*	*	*	*	0	*	314	17.7
13	1	2	2	1	1	1.09	0.47	0.56	1.94	0.77	0.17	0.35	91	25.84
14	1	2	2	1	2	*	*	*	*	*	*	*	68	10.46
15	1	2	2	2	1	1.79	0.47	1.03	1.73	0.36	0.01	0.07	2109	24.59
16	1	2	2	2	2	*	*	*	*	*	0	*	328	8.65
17	2	1	1	1	1	1.22	0.46	0.53	2.27	0.88	0.18	0.45	15	19.74
18	2	1	1	1	2	*	*	*	*	*	0	*	12	22.18
19	2	1	1	2	1	1.61	0.47	0.91	1.76	0.43	0.01	0.05	152	28.04
20	2	1	1	2	2	*	*	*	*	*	0	*	17	17.07

*: indefinite values

4.4 Comparison of the Algorithms

After calculating the defined measures on different data sets according to the experimental design, we need to compare the results of these measures to find the most effective algorithm(s). The results of the comparison measures may not dominantly favor one specific algorithm. Every algorithm will probably have its own pros and cons. Therefore, we need to use a multi criteria decision making approach to have a proper comparison of the results by considering all criteria simultaneously. We use PROMETHEE for this purpose, but before proceeding to it, the statistical analyses suggested in Chapter 3 are needed to be done.

First, comparison measures should be studied in detail in order to find the correlations among them for the purpose of reducing them to independent and complete set of evaluation criteria for PROMETHEE, to the best we can.

In order to identify highly correlated measures, a factor analysis is performed of the collected data in our experiments. As it can be seen in Table 3.5, we use 7 interestingness measures, and two other measures (time and memory use) used in the previous comparisons in the literature. Time and memory usage are two uncorrelated measures according to these results as shown in Table 4.2. But high correlations exist among the interestingness measures BF, CONF, LIFT, and LOE, and also among CONV, $\frac{n_{ab}}{n_b}$, and $\frac{r1}{r7}$.

Rotated Factor Loadings and Communalities
Varimax Rotation

Variable	Factor1	Factor2	Factor3	Factor4	Communality
BF	0.809	-0.523	0.227	0.047	0.982
CONF	0.976	-0.049	0.083	0.091	0.971
CONV	0.435	-0.841	0.291	-0.012	0.981
LIFT	0.965	0.167	0.024	0.108	0.972
LOE	0.903	0.378	-0.048	0.110	0.972
nab/nb	0.248	0.938	-0.142	0.061	0.966
r1/r7	0.192	0.960	-0.139	0.051	0.981
time	0.088	-0.287	0.953	-0.016	0.999
memory	-0.154	-0.060	0.014	-0.986	1.000
Variance	3.6749	3.0415	1.0942	1.0135	8.8241
% Var	0.408	0.338	0.122	0.113	0.980

Figure 4.1 Factor Analysis Results

We select a representative measure from each of these groups of correlated measures. In order to have a proper selection, we utilize matrix plots shown in Figure 4.2 to observe types of dependencies (linear or nonlinear) between pairs of these measures. We try to eliminate measures with not only strong linear dependencies, but also having distinct nonlinear dependency patterns, if any. We have chosen BF and CONV that are suggested to give the useful information about association rule mining algorithms according to some works in the literature and definitions used for them (Tan et al., 2002; Lenca et al., 2008). They also show the less dependency with the other measures. We could also use some other measures instead of them like LIFT, but we prefer to use BF and CONV in this work. As a result, we reduce the measures to four important ones: BF, CONV, time, and memory use.

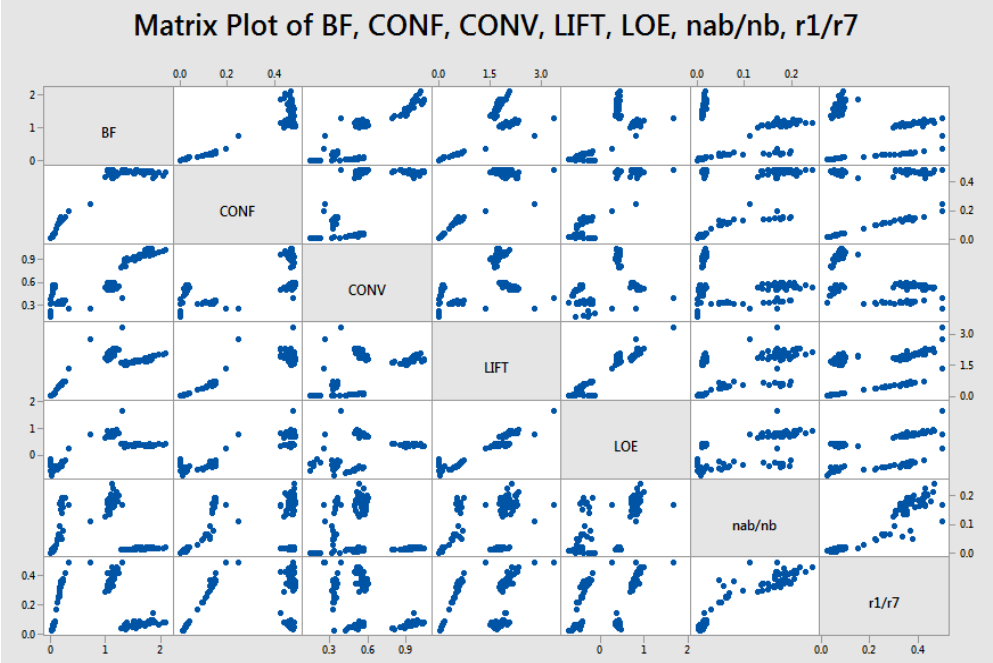


Figure 4.2 Matrix Plots of the Interestingness Measures

The next step is testing the set of hypotheses that all algorithms perform the same on the average or not with respect to the selected comparison measures, separately. For this purpose, ANOVA of the collected measure data is performed. The ANOVA has been done considering the main effects of data size, sparsity, number of attributes, and support value as blocking variables, and the algorithm as the main variable we are interested in. But as stated before in Chapter 3, we should check if errors are distributed normally with a constant variance or not as one of the important assumptions of ANOVA, and if it is not satisfied for a measure, an appropriate data transformation should be done for that measure. For this purpose, we have used Box-Cox transformation for all comparison measures except CONV. Since CONV ranges between [0, 1], we have tried Arcsin√CONV transformation for it. But for BF, even the transformations have not helped satisfying the error assumptions. Therefore, instead of ANOVA, we have used a nonparametric hypothesis test option, the Freidman test, and evaluated all the variables data size, number of attributes, support value, and sparsity as blocking variables, and the algorithms as the main factor. We

have noticed from plots of measure values of the algorithms versus the variables NR, DT, NA and SV, that performances of the algorithms change to a considerable extend with almost all of these blocking variables. Therefore, in order to reflect the effect of all these factors simultaneously, a new blocking variable is identified by combining all these variables in such a way that the levels of the combined variable correspond to the tested combinations of the levels of the individual variables. The residual plots and results of all these parametric and nonparametric hypotheses tests are available in Appendix C. It should also be noted that we have considered all the attributes categorical in ANOVA tests, because all of them have 2 or 3 levels. We have also taken $\alpha = 0.1$ for these hypotheses tests.

In all of these tests, we have observed that at least one algorithm shows a significantly different performance than the others on the average, for all comparison measures. Since the algorithms seem to be different for all measures, we can use PROMETHEE to compare them. The preference functions to be used in PROMETHEE need to be calibrated based on pairwise comparisons of the algorithms using hypothesis tests as explained in Chapter 3. We have performed all these hypothesis tests for pairwise comparisons of algorithms with $\alpha = 0.03$. According to the results of these tests summarized in Appendix D, algorithms 1 and 2 cannot be considered as significantly different from each other on the average in their CONV and BF performance, but all the other pairwise comparisons show statistically significant differences between the algorithms.

Comparison of the algorithms using PROMETHEE is performed according to the steps described in Chapter 2 as follows:

Step 1: A data matrix is constructed from alternatives (algorithms), criteria (comparison measures), and weights of criteria, W. We have assumed the same weights for all the criteria. (It is possible to adjust these weights using AHP for these

independent criteria, if found more appropriate.) The averages of the algorithms for the comparison measures BF (f_1), CONV (f_2), Time (f_3) and Memory use (f_4) are provided in Table 4.3.

Table 4.3 Data Matrix Structure

	A_1	A_2	A_3	W
f_1	0.5973	0.5967	0.4914	0.2500
f_2	0.9777	0.9783	0.5768	0.2500
f_3	354.8941	137.1500	152.7844	0.2500
f_4	40.4407	48.2535	94.7381	0.2500

Step 2: Preference functions are determined for each criterion according to the data type. These functions need to be calibrated for the selected measures based on the pairwise comparison of the algorithms. Ranges of the Criteria and Objective Functions are given in Table 4.4. Since the data are real valued, continuous functions are preferable. And, since small differences between two alternative methods are negligible up to a point, and preference intensity starts to increase from that point, we choose the "Criterion with Linear Preference and Indifference area" preference function for all four criteria (see Section 3.5). Figures 3.1 and 3.2 display the mentioned preference function for defined criteria.

Table 4.4 Ranges of Criteria and Objective Functions

Criteria	Objective Function
$0 \leq \text{CONV} \leq 1$	Min
$0 \leq \text{BF} \leq +\infty$	Max
$0 \leq \text{Time} \leq +\infty$	Min
$0 \leq \text{Memory} \leq +\infty$	Min

We define $d = f_k(A_i) - f_k(A_j)$ for criterion k and for all (A_i, A_j) pairwise alternative comparisons. These values are shown in Table 4.5.

Table 4.5 Calculated *d* Values for Alternative Pairs

Criterion	$d(A_1,A_2)$	$d(A_1,A_3)$	$d(A_2,A_3)$	$d(A_2,A_1)$	$d(A_3,A_1)$	$d(A_3,A_2)$
BF	0.0006	0.1059	0.1053	-0.0006	-0.1059	-0.1053
CONV	-0.0006	0.4008	0.4014	0.0006	-0.4008	-0.4014
Time	217.7441	202.1098	-15.6344	-217.7440	-202.1100	15.6343
Memory	-7.8127	-54.2973	-46.4846	7.8127	54.2973	46.4845

Then, *q* and *p*, which are the indifference and preference thresholds, respectively, are determined according to the literature and results of the pairwise comparisons of the algorithms by statistical hypothesis tests. Two important questions are asked for this purpose:

1) What is the smallest *d* value at which the preference function, P (d), equals to 1? This gives the *p* value.

2) What is the highest d value at which preference function, P (d), equals to 0? This gives the *q* value.

Statistical tests show that A1=A2 can be assumed for CONV and BF. Therefore, these two algorithms show the same performance for these two measures statistically, and *q*, which is the indifference threshold, can be assigned for these two measures according to the absolute differences between A1 and A2. So the absolute difference between CONV (A1) and CONV (A2) is calculated, similarly the absolute difference between BF (A1) and BF (A2) is calculated. The results of these calculations can be seen in the Table 4.5. Then, *q* values for CONV and BF are assigned close to these absolute differences. Since other pairwise comparisons show that algorithms have different performances, *p* values are assigned considering the absolute differences between all algorithms.

Similarly, *p* and *q* values for Time and Memory are assigned according to the most and the least absolute differences between all algorithms. The assigned *p* and *q* values are shown in Table 4.6.

Table 4.6 *p* and *q* Values of Criteria

Criterion	*p*	*q*
BF	0.1000	0.0010
CONV	0.4000	0.0010
Time	200.0000	10.0000
Memory Use	50.0000	5.0000

Then according to these *p* and *q* values, and defined preference function, we have calculated P_k for all pairwise comparisons as it can be seen in Table 4.7:

Table 4.7 Calculation of Preferences for All Pairwise Comparisons

Criterion	$P(A_1, A_2)$	$P(A_1, A_3)$	$P(A_2, A_3)$	$P(A_2, A_1)$	$P(A_3, A_1)$	$P(A_3, A_2)$
BF	0	1	1	0	0	0
CONV	0	0	0	0	1	1
Time	0	0	0.0296	1	1	0
Memory	0.0625	1	0.9218	0	0	0

Step 3: The multi criteria preference index is calculated for all pairwise comparisons of the alternatives. The results are given in Table 4.8.

Table 4.8 Multi Criteria Preference Index

$\prod (A_1, A_2)$	$\prod (A_1, A_3)$	$\prod (A_2, A_3)$
0.0156	0.5000	0.4878
$\prod (A_2, A_1)$	$\prod (A_3, A_1)$	$\prod (A_3, A_2)$
0.2500	0.5000	0.2500

Step 4: Leaving and entering flows for each alternative is defined and the results are provided in Table 4.9.

Table 4.9 Calculated Flows for Each Alternative

$\varphi^-(A_1)$	$\varphi^-(A_2)$	$\varphi^-(A_3)$
0.7500	0.2656	0.9878
$\varphi^+(A_1)$	$\varphi^+(A_2)$	$\varphi^+(A_3)$
0.5156	0.7378	0.7500
$\varphi(A_1)$	$\varphi(A_2)$	$\varphi(A_3)$
-0.2343	0.4722	-0.2378

Step 5: Priorities can be determined by this way:

$$\mathbf{A_2 > A_1 > A_3}$$

FP-growth > Apriori > Relim

As a result, taking into consideration all of the selected comparison criteria and all studied data characteristics, we can conclude that FP-growth is the best algorithm among all three algorithms.

It is also possible to use PROMETHEE for comparing the algorithms under specific levels of the blocking variables (data size, sparsity, number of attributes, and support value) separately. For example, we may want to compare algorithms for sparse data, and also for dense data, separately. As we have 3 algorithms, this leads to six alternatives for PROMETHEE analyses. In other words, the alternatives can be considered as algorithms under specific levels of a blocking variable. The detailed results of these comparisons are available in Appendix E. As it can be seen from the provided results, for most of the cases and factors, FP-growth is better than the others. The results obtained for time and memory use measures are also consistent with the literature results (Zheng et al., 2001; Vu et al., 2014; Fournier-Viger et al., 2014). But there exist some differences between the results we obtain by using the

proposed method and the literature results. As it is mentioned in Zheng et al. (2001), for time measure, FP-growth performs better than Apriori in low support values. Meanwhile Apriori is faster in high support values. But, our results show that FP-growth is the best algorithm for both low and high support values. We also find that for small data and low attribute numbers, Apriori is the best algorithm. This is not provided in the literature for Apriori, and our proposed method reveals this fact.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK SUGGESTIONS

Selecting the most appropriate association rule mining algorithm for the desired application has always been considered as a challenging problem, since there exist many different association rule mining algorithms, several criteria are considered in comparing them, and true interesting rules are unknown for the test data available in the literature. This study provides an objective way of comparing the association rule mining algorithms, based on known true rules, considering all relevant comparison criteria and data characteristics as well as statistically significant differences. Jabarnejad (2010) addresses this issue by proposing the novel method for comparing the rule reduction methods. In this thesis, we have used the main idea presented in Jabarnejad (2010), and extended it to the case of comparing association rule mining algorithms. We have contributed to this approach by systematically generating a representative and wide variety of data sets using the logistic regression models and considering many effective factors such as sparsity. This method enables data analysts to precisely evaluate association rule mining algorithms by considering these various data sets, several interestingness and other comparison measures, and the most important of all by knowing the exact true rules defined by the user using logistic regression models.

Another contribution of the thesis is to suggest and demonstrate use of statistical and multi criteria decision making approaches in an integrated manner in this particular case of comparing the association rule mining algorithms.

Although we propose a method with defined comparison measures for comparing the association rule mining algorithms, there also exist algorithms performing rule reductions. As a future work, similar analyses can be done on these rule reduction algorithms by defining appropriate comparison measures for them. Larger series of experiments can be conducted for these kinds of algorithms to select the most desirable one(s) which can mine the association rules more efficiently and also group and prune the redundant rules at the same time.

We provide several comparison measures for comparing the association rule mining algorithms. These measures can be studied further to include other relevant and important measures to express more efficient and comprehensive results.

REFERENCES

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. ACM SIGMOD Record, 22(2), 207-216.

Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).

Anakli, Z. (2009). A Comparison of data mining methods for prediction and classification types of quality problems. Master's Thesis, Middle East Technical University.

Barve, A., Deepa, N., & Ravi, C. (2015). A Generalized Multi Criteria Decision Making Method Based on Extension of ANP by Enhancing PAIR WISE Comparison Techniques. Cybernetics and Information Technologies, 15(4), 3-12.

Bayardo Jr, R. J., Agrawal, R., & Gunopulos, D. (2000). Constraint-based rule mining in large, dense databases. Data mining and knowledge discovery, 4(2-3), 217-240.

Behzadian, M., Kazemzadeh, R. B., Albadvi, A., & Aghdasi, M. (2010). PROMETHEE: A comprehensive literature review on methodologies and applications. European journal of Operational research, 200(1), 198-215

Berrado, A., & Runger, G. C. (2007). Using metarules to organize and group discovered association rules. Data Mining and Knowledge Discovery, 14(3), 409-431.

Borgelt, C. (2005, August). Keeping things simple: finding frequent item sets by recursive elimination. In Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations (pp. 66-70). ACM.

Borgelt, C. (2015, May 18). Christian Borgelt's Web Pages. Retrieved from http://www.borgelt.net/software.html, (last accessed on February 22, 2016)

Bramer, M. (2007). Principles of data mining (Vol. 131). London: Springer.

Brans, J. P., & Vincke, P. (1985). Note—A Preference Ranking Organization Method: (The PROMETHEE Method for Multiple Criteria Decision-Making).Management science, 31(6), 647-656

Brin, S., Motwani, R., & Silverstein, C. (1997, June). Beyond market baskets: Generalizing association rules to correlations. In ACM SIGMOD Record (Vol. 26, No. 2, pp. 265-276). ACM.

Choi, D. H., Ahn, B. S., & Kim, S. H. (2005). Prioritization of association rules in data mining: Multiple criteria decision approach. Expert Systems with Applications, 29(4), 867-878.

Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C. W., & Tseng, V. S. (2014). SPMF: a java open-source pattern mining library. The Journal of Machine Learning Research, 15(1), 3389-3393.

Fukuda, T., Morimoto, Y., Morishita, S., & Tokuyama, T. (1996). Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. ACM SIGMOD Record, 25(2), 13-23.

Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. ACM Computing Surveys (CSUR), 38(3), 9.

Hahsler, M. (2016, January 05). A Probabilistic Comparison of Commonly Used Interest Measures for Association Rules. Retrieved from http://bayesfactor.blogspot.com.tr/2014/02/the-bayesfactor-package-this-blog-is.html, (last accessed on February 22, 2016)

Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. In ACM Sigmod Record (Vol. 29, No. 2, pp. 1-12). ACM.

Hedayat, A. S., Sloane, N. J. A., & Stufken, J. (2012). Orthogonal arrays: theory and applications. Springer Science & Business Media.

Hipp, J., Güntzer, U., & Nakhaeizadeh, G. (2000). Algorithms for association rule mining—a general survey and comparison. ACM sigkdd explorations newsletter, 2(1), 58-64.

Jabarnejad, M. (2010). An improved organization method for association rules and a basis for comparison of methods. Master's Thesis, Middle East Technical University.

Jimenez, A., Berzal, F., Cubero, J.-C. (2013). Interestingness Measures for Association Rules within Groups. Intelligent Data Analysis, 17 (2): 195-215.

Lenca, P., Meyer, P., Vaillant, B., & Lallich, S. (2008). On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. European journal of operational research, 184(2), 610-626.

Margahny, M. H., & Mitwaly, A. A. (2005, December). Fast algorithm for mining association rules. In the conference proceedings of AIML, CICC, pp (36-40) Cairo, Egypt (pp. 19-21).

McGarry, K. (2005). A survey of interestingness measures for knowledge discovery. The knowledge engineering review, 20(01), 39-61.

Nakaya, A., & Morishita, S. (1999). Fast parallel search for correlated association rules. Unpublished manuscript.

Narvekar, M., & Syed, S. F. (2015). An Optimized Algorithm for Association Rule Mining Using FP Tree. Procedia Computer Science, 45, 101-110.

Ng, R. T., Lakshmanan, L. V., Han, J., & Pang, A. (1998, June). Exploratory mining and pruning optimizations of constrained association rules. In ACM Sigmod Record (Vol. 27, No. 2, pp. 13-24). ACM.

Omiecinski, E. R. (2003). Alternative interest measures for mining associations in databases. Knowledge and Data Engineering, IEEE Transactions on, 15(1), 57-69.

Padmanabhan, B., & Tuzhilin, A. (1998, August). A Belief-Driven Method for Discovering Unexpected Patterns. In KDD (Vol. 98, pp. 94-100).

Pei, J., Han, J., & Mao, R. (2000, May). CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. In ACM SIGMOD workshop on research issues in data mining and knowledge discovery (Vol. 4, No. 2, pp. 21-30).

Saaty, T. L. (1988). What is the analytic hierarchy process? (pp. 109-121). Springer Berlin Heidelberg.

Saaty, T. L. (2000). Fundamentals of decision making and priority theory with the analytic hierarchy process (Vol. 6). Rws Publications.

Sakthivel, G., & Ilangkumaran, M. (2015). A hybrid multi-criteria decision making approach of ANP and TOPSIS to evaluate the optimum fuel blend in IC engine. International Journal of Decision Support Systems, 1(3), 268-293

Srikant, R., Vu, Q., & Agrawal, R. (1997, August). Mining Association Rules with Item Constraints. In KDD (Vol. 97, p. 67).

Strehl, A., Gupta, G. K., & Ghosh, J. (1999). Distance based clustering of association rules. In Proceedings ANNIE (Vol. 9, No. 1999, pp. 759-764).

Tan, P. N., Kumar, V., & Srivastava, J. (2002, July). Selecting the right interestingness measure for association patterns. In Proceedings of the eighth

ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 32-41). ACM.

Tan, P. N., Kumar, V., & Srivastava, J. (2004). Selecting the right objective measure for association analysis. Information Systems, 29(4), 293-313.

Tewarson, R. P., & Reginald, P. (1973). Sparse Matrices (Part of the Mathematics in Science & Engineering series)

Toivonen, H., Klemettinen, M., Ronkainen, P., Hätönen, K., & Mannila, H. (1995). Pruning and grouping discovered association rules. Proceedings of the Mlnet workshop on statistics, machine learning, and discovery in databases, 47-52

Tseng, M. L. (2009). Application of ANP and DEMATEL to evaluate the decision-making of municipal solid waste management in Metro Manila. Environmental monitoring and assessment, 156(1-4), 181-197.

Vo, B., & Le, B. (2011). Interestingness measures for association rules: Combination between lattice and hash tables. Expert Systems with Applications, 38(9), 11630-11640

Vu, L., & Alaghband, G. (2014). Novel parallel method for association rule mining on multi-core shared memory systems. Parallel Computing, 40(10), 768-785.

Weka. (2016, January). Retrieved from: http://www.cs.waikato.ac.nz/ml/weka/, (last accessed on February 22, 2016)

Zaki, M. J. (2000, August). Generating non-redundant association rules. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 34-43). ACM.

Zaki, M. J. (2000). Scalable algorithms for association mining. Knowledge and Data Engineering, IEEE Transactions on, 12(3), 372-390.

Zaki, M. J., & Hsiao, C. J. (2002, April). CHARM: An Efficient Algorithm for Closed Itemset Mining. In SDM (Vol. 2, pp. 457-473).

Zheng, Z., Kohavi, R., & Mason, L. (2001, August). Real world performance of association rule algorithms. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 401-406). ACM.

APPENDIX A

EXPECTED LOGISTIC FUNCTION FOR ALL COMBINATIONS OF PROCESS VARIABLE VALUES

Table A.1 Expected Logistic Function for All Combinations of Process Variable Values

x_1	x_2	x_3	x_4	x_5	$E(y\|x)$	$f(y)$	$f'(y)$
0	1	0	1	1	-0.0010	0.4997	0.5002
1	0	1	0	1	-0.0010	0.4997	0.5002
0	0	1	1	0	0.9990	0.7308	0.2691
1	0	1	1	0	-0.0010	0.4997	0.5002
1	1	0	1	1	-1.0010	0.2687	0.7312
1	0	0	0	0	1.9990	0.8806	0.1193
1	0	1	0	0	0.9990	0.7308	0.2691
0	0	0	0	1	1.9990	0.8806	0.1193
0	0	1	0	0	1.9990	0.8806	0.1193
0	0	0	1	0	1.9990	0.8806	0.1193
1	1	0	0	1	-0.0010	0.4997	0.5002
1	1	1	1	0	-1.0010	0.2687	0.7312
0	0	1	1	1	-0.0010	0.4997	0.5002
0	1	0	0	0	1.9990	0.8806	0.1193
0	1	0	1	0	0.9990	0.7308	0.2691
0	1	0	0	1	0.9990	0.7308	0.2691
1	0	0	1	0	0.9990	0.7308	0.2691
0	1	1	0	1	-0.0010	0.4997	0.5002
0	1	1	0	0	0.9990	0.7308	0.2691
1	1	1	0	1	-1.0010	0.2687	0.7312
0	0	0	1	1	0.9990	0.7308	0.2691
1	1	0	1	0	-0.0010	0.4997	0.5002
1	1	1	0	0	-0.0010	0.4997	0.5002

Table A.1 (Continued)

x_1	x_2	x_3	x_4	x_5	$E(y\|x)$	$f(y)$	$f'(y)$
1	0	0	1	1	-0.0010	0.4997	0.5002
0	1	1	1	0	-0.0010	0.4997	0.5002
1	1	0	0	0	0.9990	0.7308	0.2691
0	1	1	1	1	-1.0010	0.2687	0.7312
1	0	1	1	1	-1.0010	0.2687	0.7312
1	0	0	0	1	0.9990	0.7308	0.2691
0	0	0	0	0	2.9990	0.9525	0.0474
1	1	1	1	1	-2.0010	0.1190	0.8809
0	0	1	0	1	0.9990	0.7308	0.2691

APPENDIX B

FULL FACTORIAL DESIGN ON SELECTED FACTORS AND LEVELS

Table B.1 Full Factorial Design on Selected Factors and Levels

Experiment	DS	DT	NA	SV
1	100	S	7	0.05
2	100	S	7	0.2
3	100	S	14	0.05
4	100	S	14	0.2
5	100	D	7	0.05
6	100	D	7	0.2
7	100	D	14	0.05
8	100	D	14	0.2
9	10000	S	7	0.05
10	10000	S	7	0.2
11	10000	S	14	0.05
12	10000	S	14	0.2
13	10000	D	7	0.05
14	10000	D	7	0.2
15	10000	D	14	0.05
16	10000	D	14	0.2

APPENDIX C

THE RESIDUAL PLOTS AND RESULTS OF ALL PARAMETRIC AND NONPARAMETRIC TESTS

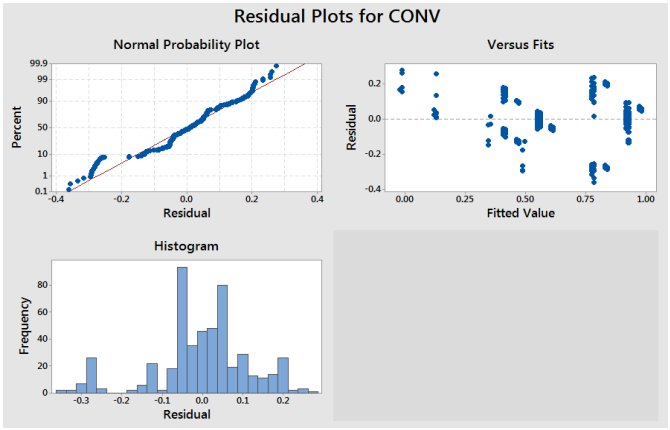


Figure C.1 CONV before Transformation

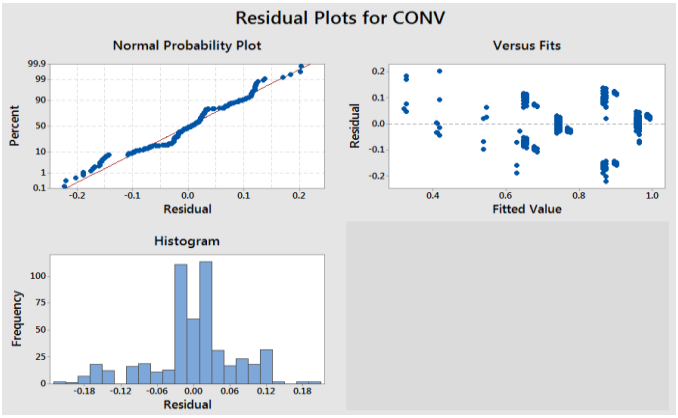


Figure C.2 CONV after Transformation with Box-Cox

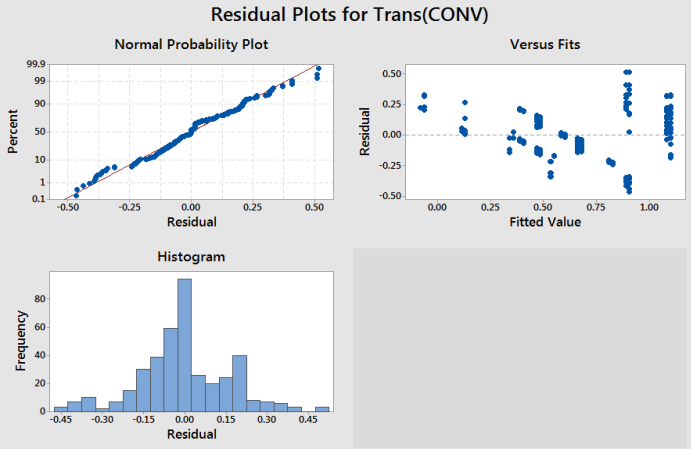


Figure C.3 CONV after Transformation with arcsin

General Linear Model: Trans(CONV) versus Algorithm, Data Size, Sparsity, ...

```
Method

Factor coding  (-1, 0, +1)
Rows unused    557


Factor Information

Factor         Type   Levels  Values
Algorithm      Fixed       3  1, 2, 3
Data Size      Fixed       2  1, 2
Sparsity       Fixed       2  1, 2
Attributes     Fixed       2  1, 2
Support Value  Fixed       2  1, 2


Analysis of Variance

Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        2   3.4319   1.7160    61.49    0.000
  Data Size        1   0.4639   0.4639    16.63    0.000
  Sparsity         1   0.0285   0.0285     1.02    0.313
  Attributes       1  14.2025  14.2025   508.97    0.000
  Support Value    1   8.0027   8.0027   286.79    0.000
Error            396  11.0501   0.0279
  Lack-of-Fit     25   3.4710   0.1388     6.80    0.000
  Pure Error     371   7.5791   0.0204
Total            402  38.8920


Model Summary

       S    R-sq  R-sq(adj)  R-sq(pred)
0.167046  71.59%     71.16%      70.52%
```

Figure C. 4 ANOVA Results for CONV

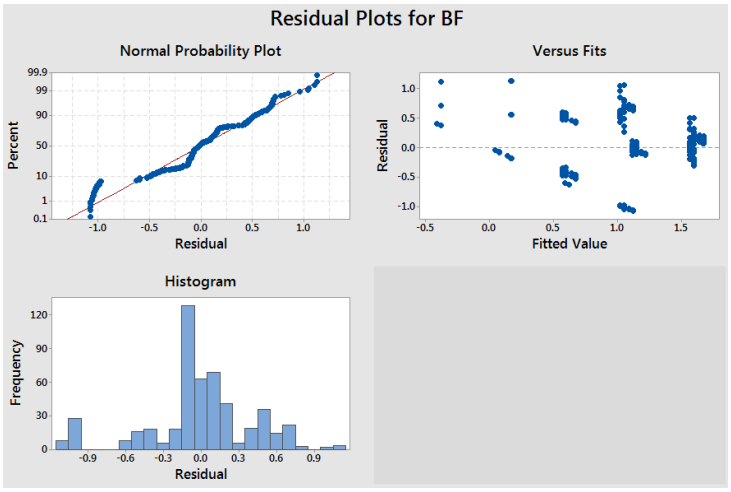


Figure C.5 BF before Transformation

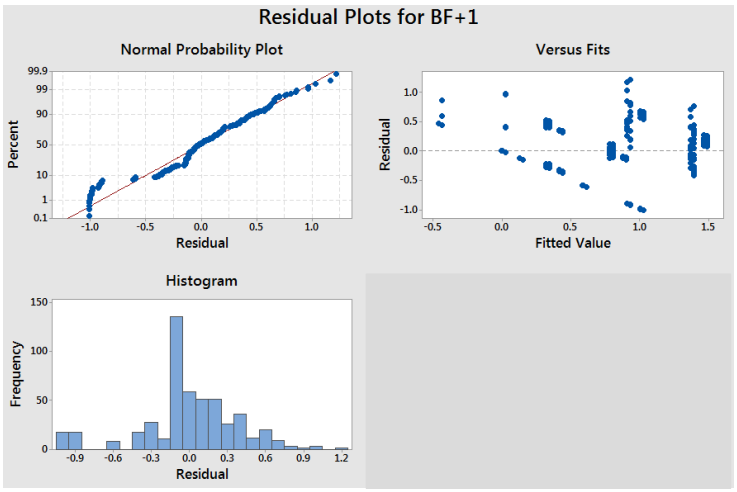


Figure C.6 BF after Transformation with Box-Cox

General Linear Model: BF+1 versus Algorithm, Data Size, Sparsity, ...

```
Method

Factor coding           (-1, 0, +1)
Rows unused             450

Box-Cox transformation
Rounded λ               2.32794
Estimated λ             2.32794
95% CI for λ            (2.02744, 2.63244)


Factor Information

Factor         Type   Levels  Values
Algorithm      Fixed       3  1, 2, 3
Data Size      Fixed       2  1, 2
Sparsity       Fixed       2  1, 2
Attributes     Fixed       2  1, 2
Support Value  Fixed       2  1, 2


Analysis of Variance for Transformed Response

Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        2   24.238  12.1189    79.30    0.000
  Data Size        1    1.150   1.1498     7.52    0.006
  Sparsity         1    0.087   0.0865     0.57    0.452
  Attributes       1   44.247  44.2470   289.54    0.000
  Support Value    1   16.315  16.3148   106.76    0.000
Error            503   76.869   0.1528
  Lack-of-Fit     29   10.451   0.3604     2.57    0.000
  Pure Error     474   66.417   0.1401
```

Figure C.7 ANOVA Results for BF

Friedman Test: Average(BF) versus Treatment blocked by Block variable

```
S = 8.00  DF = 2  P = 0.018
S = 10.11  DF = 2  P = 0.006 (adjusted for ties)

                          Sum of
Treatment   N  Est Median   Ranks
1          12      1.1036    28.0
2          12      1.1034    28.0
3          12      0.6865    16.0

Grand median = 0.9645
```

Figure C.8 Friedman Test Results for BF

We used algorithms as the main variable, and a combination of other factors as the blocking variable in Friedman test.

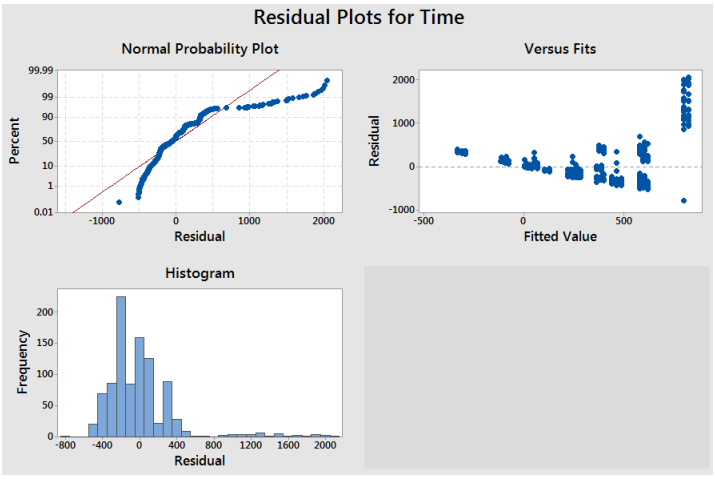


Figure C.9 Time before Transformation

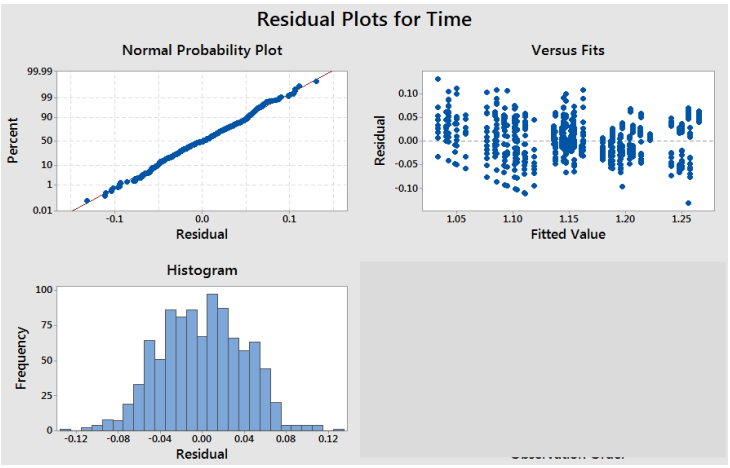


Figure C.10 Time after Transformation with Box-Cox

General Linear Model: Time versus Algorithm, Data Size, Sparsity, ...

Method

Factor coding (-1, 0, +1)

Factor Information

Factor	Type	Levels	Values
Algorithm	Fixed	3	1, 2, 3
Data Size	Fixed	2	1, 2
Sparsity	Fixed	2	1, 2
Attributes	Fixed	2	1, 2
Support Value	Fixed	2	1, 2

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Algorithm	2	9440564	4720282	33.19	0.000
Data Size	1	31109443	31109443	218.74	0.000
Sparsity	1	136873	136873	0.96	0.327
Attributes	1	26924996	26924996	189.32	0.000
Support Value	1	11486808	11486808	80.77	0.000
Error	953	135536139	142221		
Lack-of-Fit	41	106987118	2609442	83.36	0.000
Pure Error	912	28549021	31304		
Total	959	214634823			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
377.121	36.85%	36.46%	35.92%

Figure C.11 ANOVA Results for Time

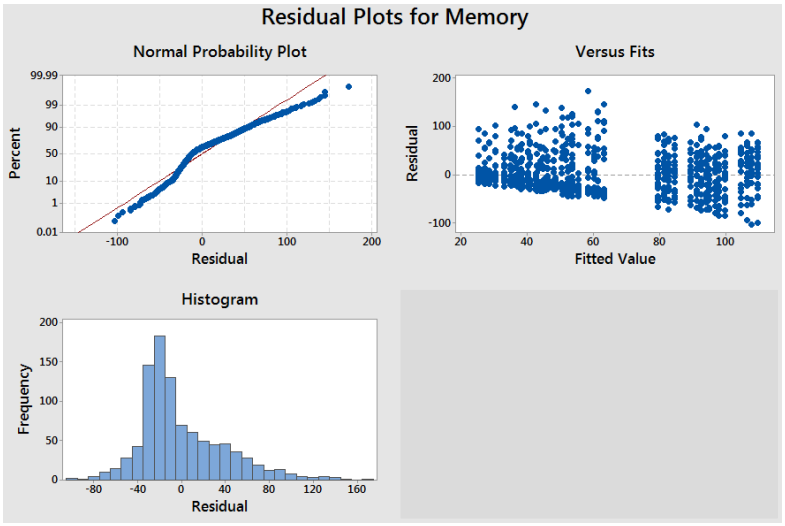


Figure C.12 Memory Use before Transformation

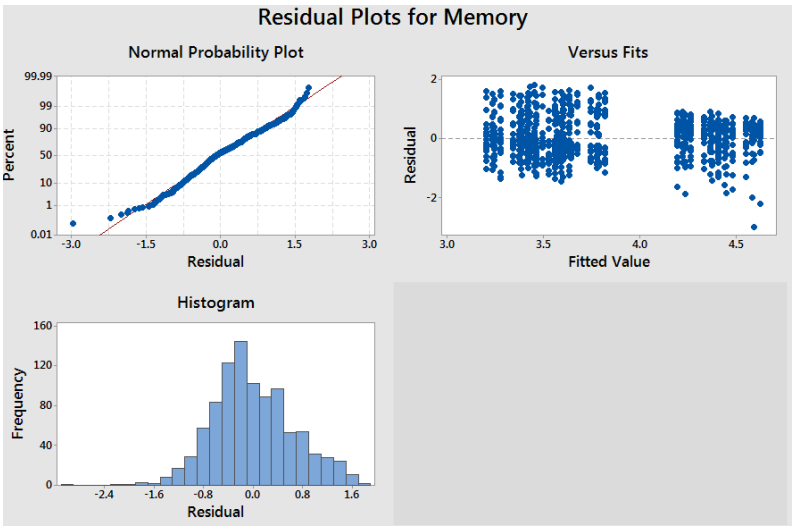


Figure C.13 Memory Use after Transformation with Box-Cox

General Linear Model: Memory versus Algorithm, Data Size, Sparsity, ...

```
Method

Factor coding           (-1, 0, +1)

Box-Cox transformation
Rounded λ               0
Estimated λ             -0.00551444
95% CI for λ            (-0.0860144, 0.0749856)


Factor Information

Factor         Type   Levels  Values
Algorithm      Fixed       3  1, 2, 3
Data Size      Fixed       2  1, 2
Sparsity       Fixed       2  1, 2
Attributes     Fixed       2  1, 2
Support Value  Fixed       2  1, 2


Analysis of Variance for Transformed Response

Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        2  177.412  88.7058   205.23    0.000
  Data Size        1   11.301  11.3006    26.15    0.000
  Sparsity         1    0.247   0.2470     0.57    0.450
  Attributes       1    4.756   4.7560    11.00    0.001
  Support Value    1    0.412   0.4123     0.95    0.329
Error            953  411.904   0.4322
  Lack-of-Fit     41   14.948   0.3646     0.84    0.756
  Pure Error     912  396.955   0.4353
Total            959  606.031
```

Figure C.14 ANOVA Results for Memory Use

APPENDIX D

THE RESIDUAL PLOTS AND RESULTS OF ALL PARAMETRIC AND NONPARAMETRIC TESTS FOR PAIRWISE COMPARISONS

ANOVA for comparing algorithms 1 and 2 with regard to CONV:

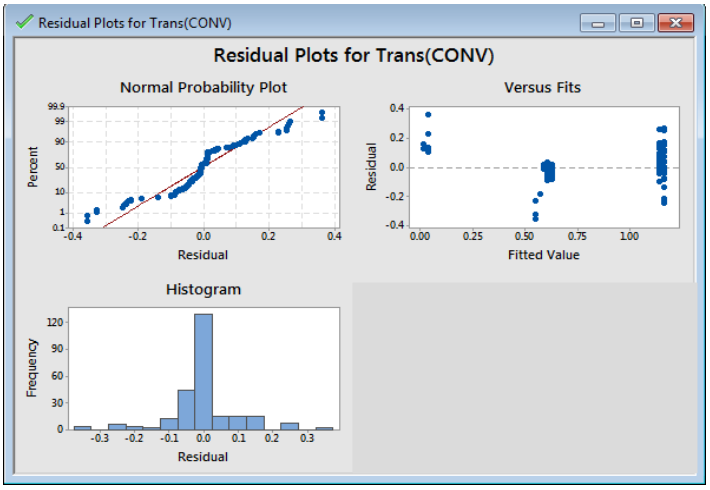


Figure D.1 Residual Plots for Trans (CONV) for Comparing Algorithms 1 and 2

General Linear Model: Trans(CONV) versus algorithm, Data Size, Sparsity, ...

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
algorithm	1	0.0000	0.0000	0.00	0.969
Data Size	1	0.0138	0.0138	1.38	0.242
Sparsity	1	0.0245	0.0245	2.44	0.119
Attributes	1	12.6739	12.6739	1263.69	0.000
support value	1	6.2005	6.2005	618.24	0.000
Error	251	2.5173	0.0100		
Lack-of-Fit	14	1.2214	0.0872	15.95	0.000
Pure Error	237	1.2960	0.0055		
Total	256	24.8902			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.100146	89.89%	89.68%	89.00%

Figure D.2 ANOVA Results for Trans (CONV) for Comparing Algorithms 1 and 2

ANOVA for comparing algorithms 1 and 2 with regard to BF:

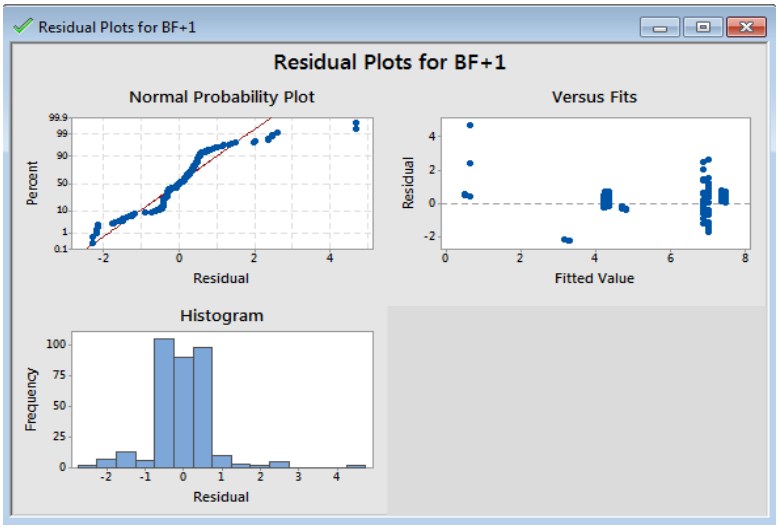


Figure D.3 Residual Plots for BF for Comparing Algorithms 1 and 2

Friedman Test: Average(BF) versus Treatment blocked by Block variable

```
S = 0.00  DF = 1  P = 1.000
S = 0.00  DF = 1  P = 1.000 (adjusted for ties)

                          Sum of
Treatment   N  Est Median   Ranks
1          12      1.1035    18.0
2          12      1.1035    18.0

Grand median = 1.1035
```

Figure D.4 Friedman Test Results for BF for Comparing Algorithms 1 and 2

We used algorithms as the main variable, and a combination of other factors as the blocking variable. P values show that Algorithms 1 and 2 algorithms 1 and 2 cannot be considered as significantly different from each other on the average with α=0.03.

ANOVA for comparing algorithms 1 and 2 with regard to Time:

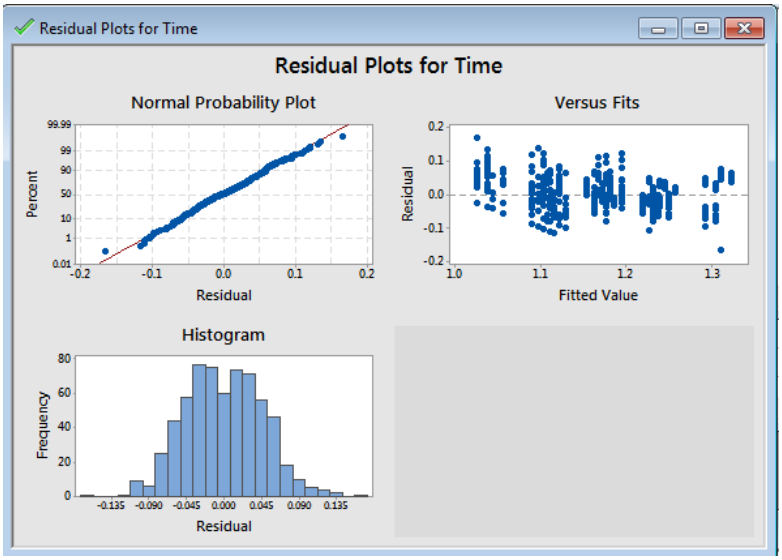


Figure D.5 Residual Plots for Transformed Time for Comparing Algorithms 1 and 2

General Linear Model: Time versus Algorithm, Data Size, Sparsity, ...

```
Box-Cox transformation
Rounded λ               0.0410131
Estimated λ             0.0410131
95% CI for λ            (0.0165131, 0.0655131)

Analysis of Variance for Transformed Response
Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        1  0.05368  0.05368    24.36    0.000
  Data Size        1  2.67194  2.67194  1212.74    0.000
  Sparsity         1  0.02662  0.02662    12.08    0.001
  Attributes       1  0.84949  0.84949   385.57    0.000
  Support Value    1  0.68159  0.68159   309.36    0.000
Error            634  1.39685  0.00220
  Lack-of-Fit     26  0.46489  0.01788    11.67    0.000
  Pure Error     608  0.93195  0.00153
Total            639  5.68017

Model Summary for Transformed Response
        S    R-sq  R-sq(adj)  R-sq(pred)
0.0469386  75.41%     75.21%      74.94%
```

Figure D.6 ANOVA Results for Time for Comparing Algorithms 1 and 2

ANOVA for comparing algorithms 1 and 2 with regard to Memory Use:

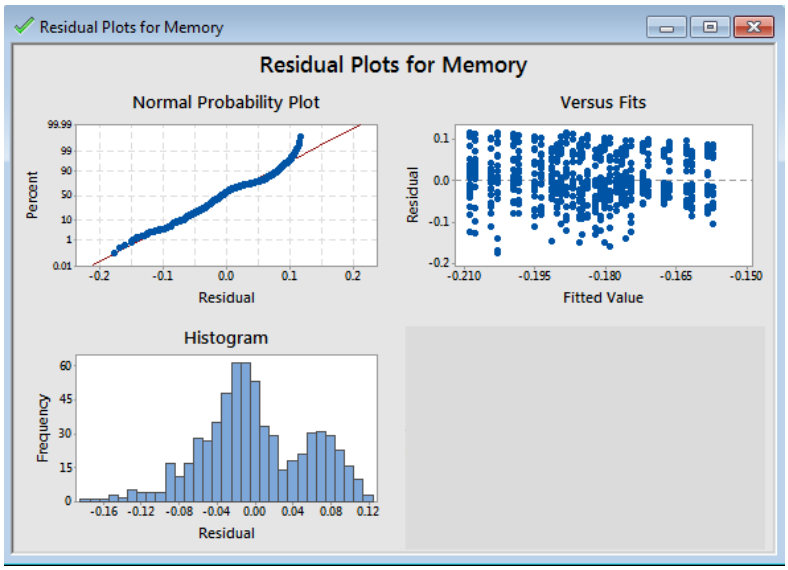


Figure D.7 Residual Plots for Transformed Memory Use for Comparing Algorithms1 and2

General Linear Model: Memory versus Algorithm, Data Size, Sparsity, ...

```
Box-Cox transformation
Rounded λ               -0.5
Estimated λ             -0.402076
95% CI for λ            (-0.503576, -0.300576)

Analysis of Variance for Transformed Response
Source            DF   Adj SS    Adj MS  F-Value  P-Value
  Algorithm        1  0.04806  0.048060    14.76    0.000
  Data Size        1  0.05936  0.059356    18.23    0.000
  Sparsity         1  0.00022  0.000222     0.07    0.794
  Attributes       1  0.01358  0.013575     4.17    0.042
  Support Value    1  0.00336  0.003357     1.03    0.310
Error            634  2.06471  0.003257
  Lack-of-Fit     26  0.08805  0.003387     1.04    0.408
  Pure Error     608  1.97665  0.003251
Total            639  2.18927

Model Summary for Transformed Response
        S   R-sq  R-sq(adj)  R-sq(pred)
0.0570669  5.69%      4.95%       3.90%
```

Figure D.8 ANOVA Results for Memory Use for Comparing Algorithms 1 and 2

ANOVA for comparing algorithms 1 and 3 with regard to CONV:

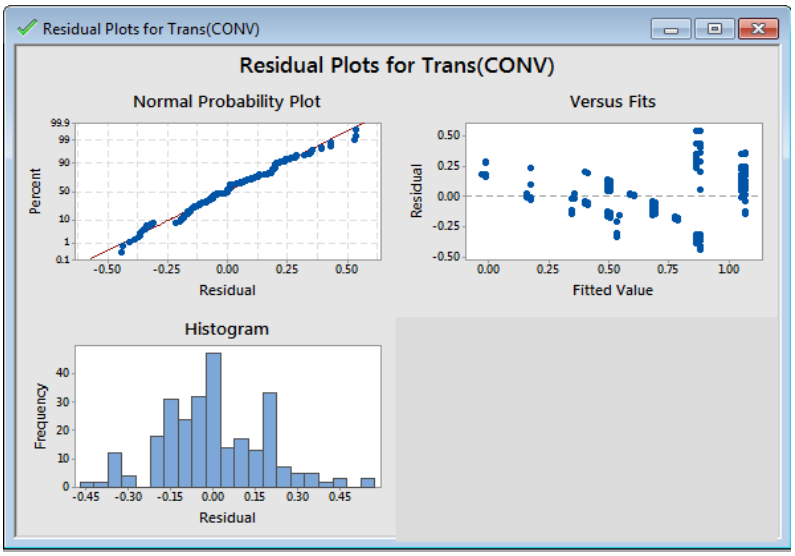


Figure D.9 Residual Plots for Trans (CONV) for Comparing Algorithms 1 and 3

General Linear Model: Trans(CONV) versus Algorithm, Data Size, Sparsity, ...

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Algorithm	1	2.3593	2.35926	68.09	0.000
Data Size	1	0.4646	0.46461	13.41	0.000
Sparsity	1	0.0144	0.01440	0.42	0.520
Attributes	1	8.0826	8.08263	233.27	0.000
Support Value	1	4.8376	4.83761	139.61	0.000
Error	268	9.2861	0.03465		
Lack-of-Fit	16	2.3739	0.14837	5.41	0.000
Pure Error	252	6.9122	0.02743		
Total	273	25.5409			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.186144	63.64%	62.96%	62.03%

Figure D.10 ANOVA Results for Trans (CONV) for Comparing Algorithms 1 and 3

ANOVA for comparing algorithms 1 and 3 with regard to BF:

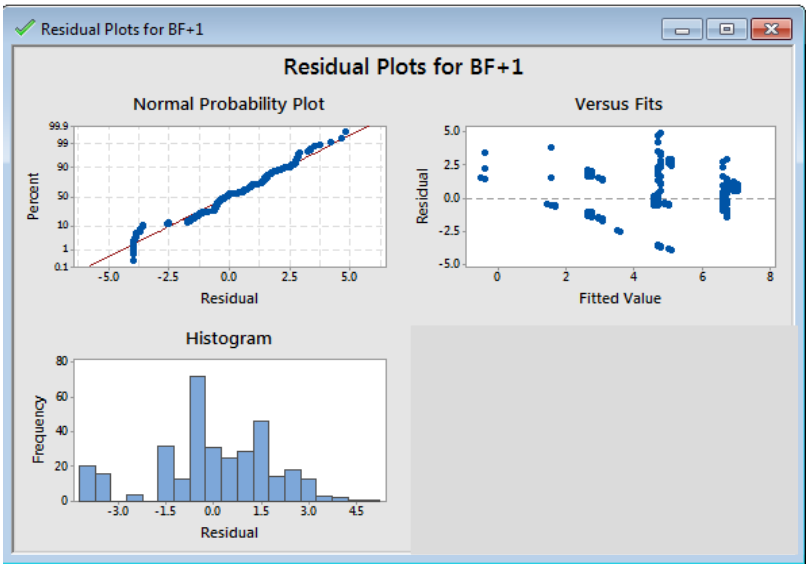


Figure D.11 Residual Plots for BF for Comparing Algorithms 1 and 3

Friedman Test: Average(BF) versus Treatment blocked by Block variable

```
S = 5.33  DF = 1  P = 0.021
S = 6.40  DF = 1  P = 0.011 (adjusted for ties)

                          Sum of
Treatment   N  Est Median   Ranks
1          12      1.1039    22.0
3          12      0.6864    14.0

Grand median = 0.8952
```

Figure D.12 Friedman Test Results for BF for Comparing Algorithms 1 and 3

P values show that Algorithms 1 and 3 can be considered as significantly different from each other for this measure with α=0.03.

ANOVA for comparing algorithms 1 and 3 with regard to Time:

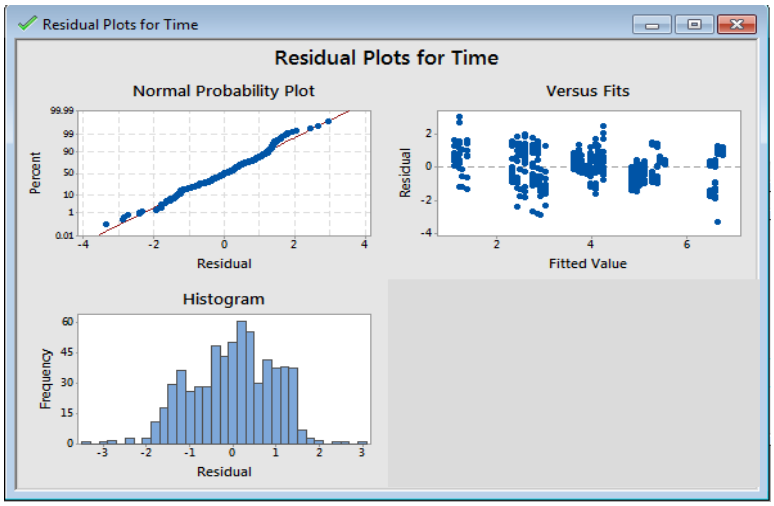


Figure D.13 Residual Plots for Transformed Time for Comparing Algorithms 1 and 3

General Linear Model: Time versus Algorithm, Data Size, Sparsity, ...

```
Box-Cox transformation
Rounded λ               0
Estimated λ             0.0245588
95% CI for λ            (-0.000941159, 0.0500588)

Analysis of Variance for Transformed Response
Source             DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm         1     4.34     4.34     4.71    0.030
  Data Size         1  1030.40  1030.40  1120.04    0.000
  Sparsity          1     2.17     2.17     2.36    0.125
  Attributes        1   438.19   438.19   476.31    0.000
  Support Value     1   237.89   237.89   258.59    0.000
Error             634   583.26     0.92
  Lack-of-Fit      26   194.93     7.50    11.74    0.000
  Pure Error      608   388.33     0.64
Total             639  2296.25

Model Summary for Transformed Response
       S    R-sq  R-sq(adj)  R-sq(pred)
0.959150  74.60%     74.40%      74.12%
```

Figure D.14 ANOVA Results for Transformed Time for Comparing Algorithms 1 and 3

ANOVA for comparing algorithms 1 and 3 with regard to Memory Use:

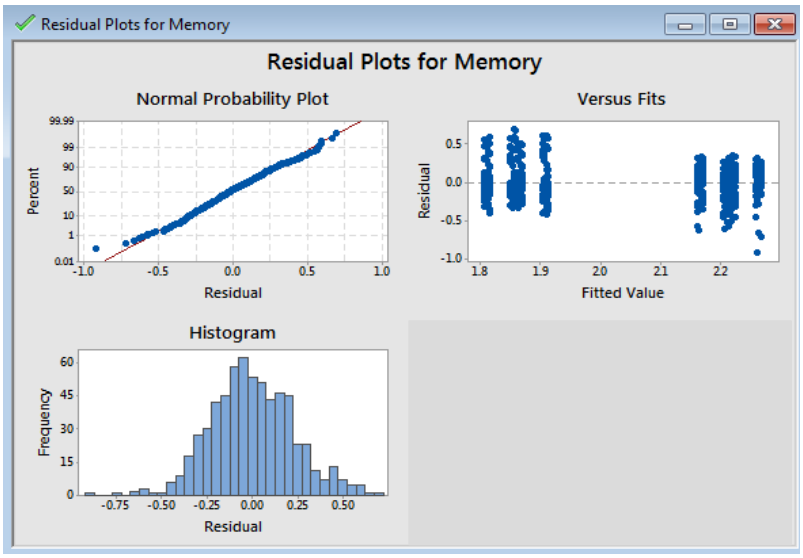


Figure D.15 Residual Plots for Transformed Memory Use for Comparing Algorithms 1 and 3

General Linear Model: Memory versus Algorithm, Data Size, Sparsity, ...

```
Box-Cox transformation
Rounded λ              0.179075
Estimated λ            0.179075
95% CI for λ           (0.0825749, 0.277575)

Analysis of Variance for Transformed Response
Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        1  20.1131  20.1131   371.02    0.000
  Data Size        1   0.2955   0.2955     5.45    0.020
  Sparsity         1   0.0071   0.0071     0.13    0.717
  Attributes       1   0.4459   0.4459     8.23    0.004
  Support Value    1   0.0014   0.0014     0.03    0.873
Error            634  34.3693   0.0542
  Lack-of-Fit     26   0.6906   0.0266     0.48    0.987
  Pure Error     608  33.6787   0.0554
Total            639  55.2324

Model Summary for Transformed Response
       S    R-sq  R-sq(adj)  R-sq(pred)
0.232831  37.77%     37.28%      36.59%
```

Figure D.16 ANOVA Results for Transformed Memory Use for Comparing Algorithms 1 and 3

ANOVA for comparing algorithms 2 and 3 with regard to CONV:

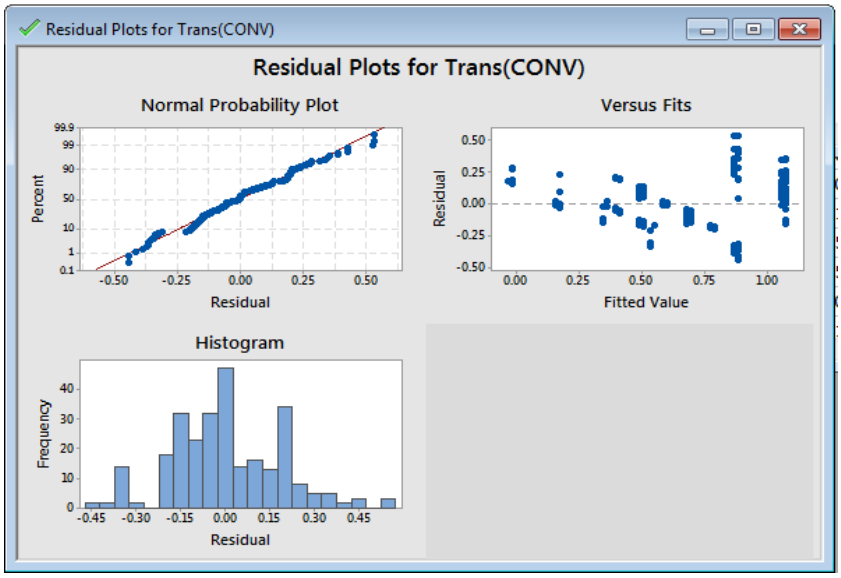


Figure D.17 Residual Plots for Trans (CONV) for Comparing Algorithms 2 and 3

General Linear Model: Trans(CONV) versus Algorithm, Data Size, Sparsity, ...

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Algorithm	1	2.4069	2.40695	68.94	0.000
Data Size	1	0.4807	0.48070	13.77	0.000
Sparsity	1	0.0184	0.01841	0.53	0.468
Attributes	1	8.2990	8.29903	237.69	0.000
Support Value	1	4.8722	4.87215	139.54	0.000
Error	269	9.3923	0.03492		
Lack-of-Fit	16	2.4423	0.15264	5.56	0.000
Pure Error	253	6.9500	0.02747		
Total	274	26.0968			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
0.186857	64.01%	63.34%	62.41%

Figure D.18 ANOVA Results for Trans (CONV) for Comparing Algorithms 2 and 3

ANOVA for comparing algorithms 2 and 3 with regard to BF:

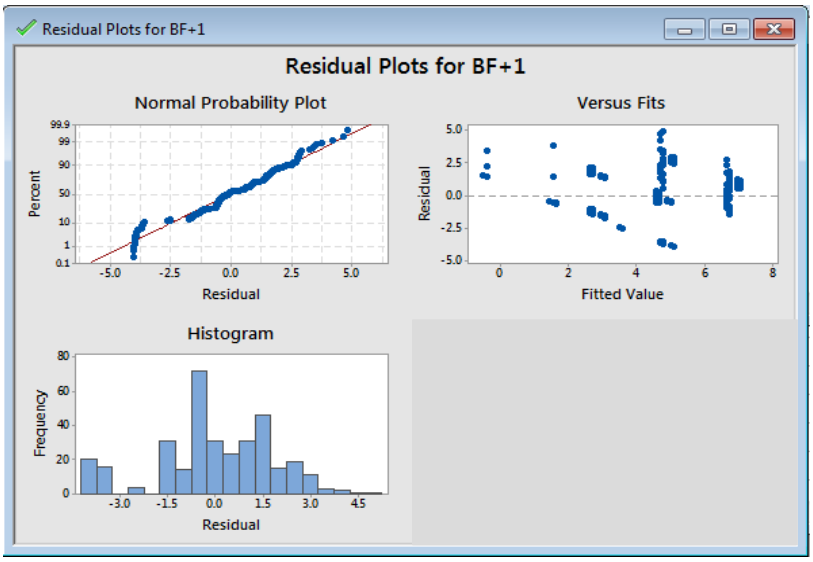


Figure D.19 Residual Plots for BF for Comparing Algorithms 2 and 3

Friedman Test: Average(BF) versus Treatment blocked by Block variable

```
S = 5.33  DF = 1  P = 0.021
S = 6.40  DF = 1  P = 0.011 (adjusted for ties)

                          Sum of
Treatment   N  Est Median   Ranks
2          12      1.1031    22.0
3          12      0.6864    14.0

Grand median = 0.8948
```

Figure D.20 Friedman Test Results for BF for Comparing Algorithms 2 and 3

ANOVA for comparing algorithms 2 and 3 with regard to Time:

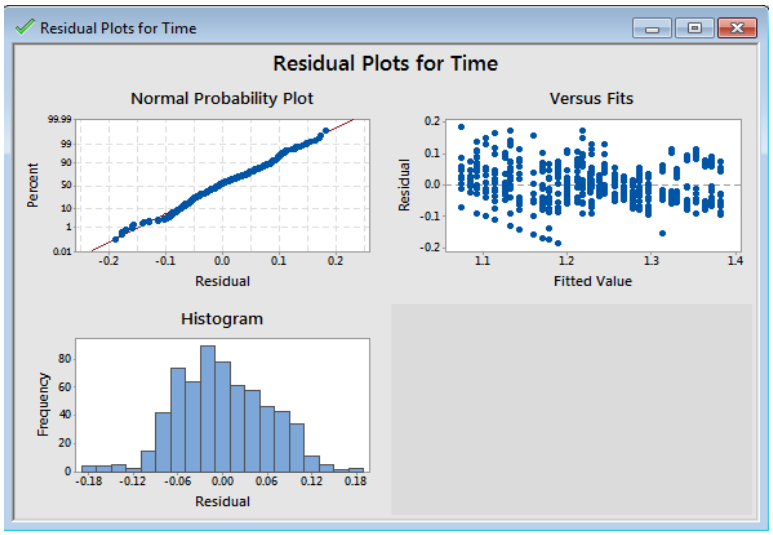


Figure D.21 Residual Plots for Transformed Time for Comparing Algorithms 2 and 3

General Linear Model: Time versus Algorithm, Data Size, Sparsity, ...

```
Box-Cox transformation              Rounded λ              0.0534844
Estimated λ             0.0534844
95% CI for λ            (0.0229844, 0.0839844)

Analysis of Variance for Transformed Response
Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        1  0.02074  0.02074     5.33    0.021
  Data Size        1  3.77056  3.77056   969.19    0.000
  Sparsity         1  0.05313  0.05313    13.66    0.000
  Attributes       1  1.19427  1.19427   306.98    0.000
  Support Value    1  0.26101  0.26101    67.09    0.000
Error            634  2.46653  0.00389
  Lack-of-Fit     26  0.24489  0.00942     2.58    0.000
  Pure Error     608  2.22164  0.00365
Total            639  7.76624

Model Summary for Transformed Response
S    R-sq  R-sq(adj)  R-sq(pred)
0.0623733  68.24%     67.99%       67.64%
```

Figure D.22 ANOVA Results for Transformed Time for Comparing Algorithms 2 and 3

ANOVA for comparing algorithms 2 and 3 with regard to Memory Use:

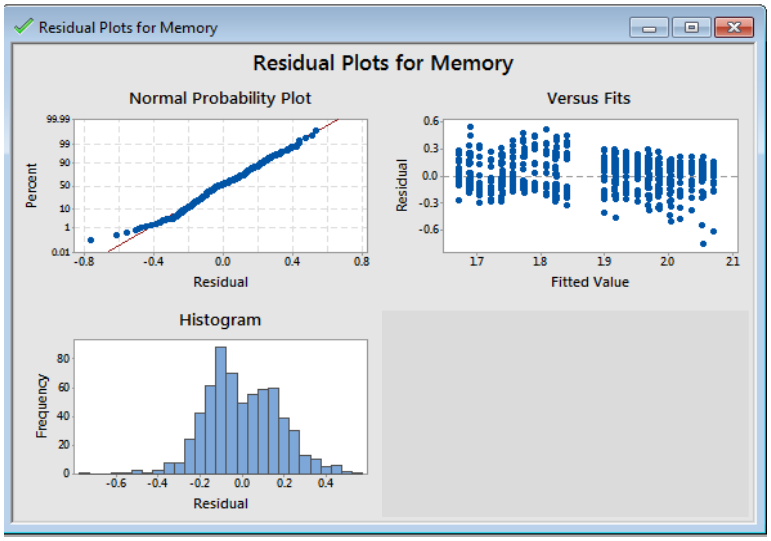


Figure D.23 Residual Plots for Transformed Memory Use for Comparing Algorithms 2 and 3

General Linear Model: Memory versus Algorithm, Data Size, Sparsity, ...

```
Box-Cox transformation
Rounded λ              0.154683
Estimated λ            0.154683
95% CI for λ           (0.0511826, 0.260183)

Analysis of Variance for Transformed Response
Source            DF   Adj SS   Adj MS  F-Value  P-Value
  Algorithm        1   8.4357  8.43572   263.88    0.000
  Data Size        1   1.1748  1.17476    36.75    0.000
  Sparsity         1   0.0403  0.04029     1.26    0.262
  Attributes       1   0.4195  0.41950    13.12    0.000
  Support Value    1   0.0497  0.04971     1.55    0.213
Error            634  20.2680  0.03197
  Lack-of-Fit     26   0.6826  0.02625     0.81    0.730
  Pure Error     608  19.5855  0.03221
Total            639  30.3880

Model Summary for Transformed Response
       S    R-sq  R-sq(adj)  R-sq(pred)
0.178797  33.30%     32.78%      32.03%
```

Figure D.24 ANOVA Results for Transformed Memory Use for Comparing Algorithms 2 and 3

APPENDIX E

COMPARISON OF THE ALGORITHMS WITH REGARD TO DATA SIZE, SPARSITY, NUMBER OF ATTRIBUTES, AND SUPPORT VALUE FACTORS

PROMETHEE with regard to Sparsity:

Table E.1 Notations for Alternatives (Regarding DT)

Algorithm1= Apriori	Algorithm2=FP-growth	Algorithm3= Relim
A_1= Algorithm1 on sparse data	A_3= Algorithm2 on sparse data	A_5= Algorithm3 on sparse data
A_2= Algorithm 1 on dense data	A_4= Algorithm 2 on dense data	A_6= Algorithm 3 on dense data

Table E.2 Data Matrix Structure (Regarding DT)

	A_1	A_2	A_3	A_4	A_5	A_6	W
f_1	0.5801	0.6145	0.5796	0.6137	0.4776	0.5051	0.25
f_2	0.9360	1.0193	0.9374	1.0191	0.5528	0.6008	0.25
f_3	368.2563	341.5320	151.1500	123.1500	161.2438	144.3250	0.25
f_4	40.8390	40.0425	46.5290	49.9781	93.3005	96.1757	0.25

Table E.3 Calculated Net Flows for Each Alternative (Regarding DT)

A_1	A_2	A_3	A_4	A_5	A_6
-0.5707	-0.3323	0.8488	1.0304	-0.5747	-0.4014

Priorities are:

$A_4 > A_3 > A_2 > A_6 > A_1 > A_5$

Algorithm 2 performs better on dense data than on sparse data. Overall, it performs better for both sparse and dense data than the others.

PROMETHEE with regard to Number of Attributes:

Table E.4 Notations for Alternatives (Regarding NA)

Algorithm1= Apriori	Algorithm2=FP-growth	Algorithm3= Relim
A_1= Algorithm1 with 7 NA	A_3= Algorithm2 with 7 NA	A_5= Algorithm3 with 7 NA
A_2= Algorithm 1 with 14 NA	A_4= Algorithm 2 with 14 NA	A_6= Algorithm 3 with 14 NA

Table E.5 Data Matrix Structure (Regarding NA)

	A_1	A_2	A_3	A_4	A_5	A_6	W
f_1	0.4397	0.7549	0.4385	0.7548	0.3652	0.6175	0.25
f_2	0.8214	1.1340	0.8216	1.1349	0.5028	0.6509	0.25
f_3	48.8625	660.9258	50.5687	223.7313	42.9812	262.5875	0.25
f_4	39.2484	41.6331	43.7816	52.7254	85.6579	103.8183	0.25

Table E.6 Calculated Net Flows for Each Alternative (Regarding NA)

A_1	A_2	A_3	A_4	A_5	A_6
1.2663	-0.6396	0.7088	0.0424	-0.1367	-0.7406

Priorities are:

$A_1 > A_3 > A_4 > A_5 > A_2 > A_6$

Algorithm 1 is good for low attribute numbers, but Algorithm 2 is good for high attribute numbers.

PROMETHEE with regard to Data Size:

Table E.7 Notations for Alternatives (Regarding NR)

Algorithm1= Apriori	Algorithm2=FP-growth	Algorithm3= Relim
A_1= Algorithm1 with low NR	A_3= Algorithm2 with low NR	A_5= Algorithm3 with low NR
A_2= Algorithm 1 with high NR	A_4= Algorithm 2 with high NR	A_6= Algorithm 3 with high NR

Table E.8 Data Matrix Structure (Regarding NR)

	A_1	A_2	A_3	A_4	A_5	A_6	W
f_1	0.4961	0.7997	0.4956	0.7988	0.4157	0.6426	0.25
f_2	0.7428	1.4474	0.7428	1.4493	0.4405	0.8494	0.25
f_3	43.5062	666.2820	28.9687	245.3313	32.3062	273.2625	0.25
f_4	37.7192	43.1623	35.5018	61.0052	87.0812	102.3950	0.25

Table E.9 Calculated Net Flows for Each Alternative (Regarding NR)

A_1	A_2	A_3	A_4	A_5	A_6
1.6223	-0.7291	0.9612	-0.4445	-0.2444	-1.1654

Priorities are:

$A_1 > A_3 > A_5 > A_4 > A_2 > A_6$

Algorithm 1 is good for small data. Algorithm 2 is good for large data.

PROMETHEE with regard to Support Value:

Table E.10 Notations for Alternatives (Regarding SV)

Algorithm1= Apriori	Algorithm2=FP-growth	Algorithm3= Relim
A_1= Algorithm1 with low SV	A_3= Algorithm2 with low SV	A_5= Algorithm3 with low SV
A_2= Algorithm 1 with high SV	A_4= Algorithm 2 with high SV	A_6= Algorithm 3 with high SV

Table E.11 Data Matrix Structure (Regarding SV)

	A_1	A_2	A_3	A_4	A_5	A_6	W
f_1	0.7697	0.2525	0.7688	0.2525	0.6198	0.2344	0.25
f_2	1.4028	0.1273	1.4037	0.1273	0.8310	0.0685	0.25
f_3	612.0008	97.7875	194.1563	80.1437	166.8313	138.7375	0.25
f_4	40.3347	40.5468	52.1138	44.3933	95.6903	93.7858	0.25

Table E.12 Calculated Net Flows for Each Alternative (Regarding SV)

A_1	A_2	A_3	A_4	A_5	A6
-0.6746	1.0154	0.2457	1.0802	-0.8939	-0.7351

Priorities are:

$A_4 > A_2 > A_3 > A_1 > A_6 > A_5$

Algorithm 2 is good for both low and high support value. It performs better for high support values.