# Auto-Optimization of Business Rules Through Business Process Mining for Improved System Performance

**By**

**Faqeer ur Rehman**

**NUST201464090MSEECS61314F**

**Supervisor**

**Dr. Mian M. Hamayun**

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science (MSCS)

in

**School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad, Pakistan**

**(August 2018)**

# APPROVAL

It is certified that the contents and form of thesis entitled ***"Auto-Optimization of Business Rules Through Business Process Mining for Improved System Performance"*** submitted by ***Mr.Faqeer ur Rehman*** has been found satisfactory for the requirement of the degree.

Advisor: **Dr. Mian M. Hamayun**

Signature: ____________________

Date: ________________________

Committee Member 1: **Dr. Sharifullah Khan**

Signature: __________________________

Date: ______________________________

Committee Member 2: **Dr. Imran Malik**

Signature: __________________________

Date: ______________________________

Committee Member 3: **Dr. Omar Arif**

Signature: __________________________

Date: ______________________________

## Dedication

*This thesis is dedicated to my parents, my wife and to my supervisor Dr. Mian M. Hamayun, whose untiring devotion and motivation lead me to the completion of my work.*

## CERTIFICATE OF ORIGINALITY

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by any other person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: **Faqeer ur Rehman**

Signature: __________________

## ACKNOWLEDGEMENTS

First of all, I am too much thankful to Allah Almighty for providing me with the insight, courage and knowledge to fulfill this task and pursue my dreams; nothing would have been possible without HIS blessings and benevolence. I would like to express my deep and sincere gratitude to my supervisor, Dr. Mian M. Hamayun, Ph.D. (Computer Sciences), Department of Computing, SEECS, NUST. His wide knowledge and his logical way of thinking have been of great value to me. His understanding of the topic, persistent support and constant encouragement has inspired and motivated me as a researcher and a student. I am indebted to him more than he knows.

Then I would like to thank all my GEC members Dr. Muhammad Imran Malik, Dr. Sharifullah Khan and Dr. Omar Arif who continuously guided and helped me wherever I needed. In the end I would like to thank everybody who was important to the successful realization of my thesis.

**Faqeer ur Rehman**

# ***TABLE OF CONTENTS***

# ***LIST OF TABLES***

## ***LIST of FIGURES***

# ABSTRACT

Data mining techniques exist that have the ability to extract useful knowledge from the available event logs. Over the years, there has been a tremendous enhancement in the IT sector. Organizations are moving towards automation of their business processes to improve and fasten their day to day business operations that in turn can lead to enhance their businesses. It is highly important for them to see their software(s)/ERP solution(s) reliable, having high performance and with full hardware resources utilization. To successfully execute business process, there can be hundreds or thousands of business rules/constraints that must be satisfied. Hard coded written sequence inside a system may reduce system performance and can lead to high resource wastages. In the present work, we first classify the business rules execution logs and then apply pattern mining techniques to predict a possible optimized business rules execution order. This new predicted optimized sequence is used for further execution of the software to adjust business rules execution order dynamically according to the running environment to make system self-adaptive. We have experimented the approach on 89304 no of failed business instances logs and found that using the proposed model, system resources consumption can be prevented from wastage by 69.7% (worst case) and 77.62% (best case) respectively.

# CHAPTER 1

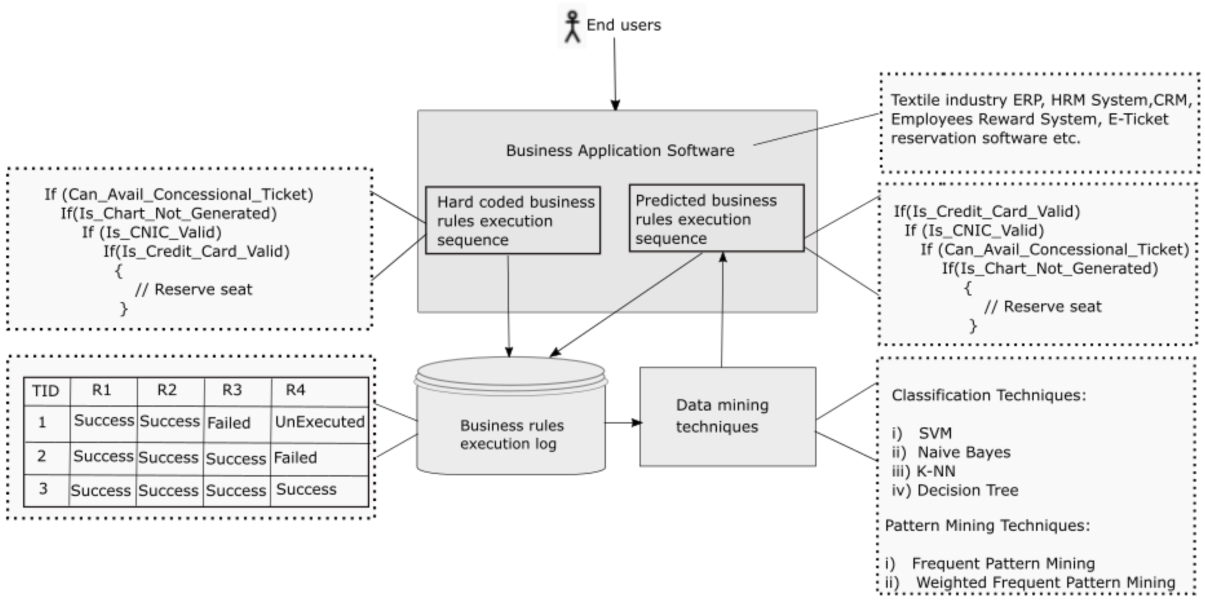
# INTRODUCTION

A business process is a set of activities and tasks that are carried out to achieve an organizational goal. Business process management system (BPMS) is a system that helps to accomplish business process management (BPM) [1].Today, large areas of applications belonging to different domains e.g. textile industry ERP systems, Pharmaceutical industry ERP systems, Car rental and lease ERP systems, large HRM systems, Stock management systems, Employee reward management systems, CRM systems, Financial and Accounting systems and other long chain of solutions automates business processes of an organizations. Those business processes contain large number of business rules to be satisfied for their successful completion. Million lines of code are written to ensure that desired business constraints must be satisfied for successful completion of a business process.

The goal of business process mining is to discover, analyze, optimize and apprehend business strategies primarily based on the run-time behavior of executed business process instances recorded in an event logs [26]. Business process management (BPM) technology has been evolved and implemented almost in every enterprise. Many businesses and companies use different type of business process oriented systems. An automated system or software implements large number of business processes for smooth execution of business operations.

Software engineer is the one who develops software using some programming language e.g. C#, F#, Visual Basic, VB.Net, Java, Python, R etc. In a normal scenario, when a requirement comes from a client to adjust new business constraint or rule in a large system, a software engineer tries to code and put it in its best supposed location inside a class along with keeping in view to maintain the desired dependency (if any) among them. Those set of written business rules execute in hard coded fashion continuously till either the system is running or the new rule is inserted to adjust new requirements. The addition of new business rule may although change execution sequence of previously existing business rules but will again keep system to continue running with the new updated hard coded sequential order. Each user has different mind filter and different mindset people interact with the system differently, especially with the application that is exposed to public. If the usage of system by the people is logged and analyzed carefully, it can be mined to predict number of different parameters e.g. nature of users that are frequently interacting with the system, which business rules are causing the business process to fail, which business rules need more focus to be matured, where the security breach could be, where business process improvements are needed and what type of preventive measures are needed to be taken. It will also help decision makers to either device new strategies or to improve and optimize the existing ones.

Using data mining techniques, an optimized business rules execution order can be predicted that will allow system to prioritize business rules as per their execution importance / cost. System can run for specific time duration by executing the business rules as per the new predicted order. Logs will be generated for that time duration and will be mined again to predict new rules order for the next time duration. Instead of using hard coded execution sequence for business rules, they should be adjusted dynamically as per the running environment, if executed can prevent system resources from wastage and can enhance system performance as well.


End users
Business Application Software
Hard coded business rules execution sequence
Predicted business rules execution sequence
Textile industry ERP, HRM System,CRM, Employees Reward System, E-Ticket reservation software etc.
Business rules execution log
Data mining techniques
Classification Techniques:
i) SVM
ii) Naive Bayes
iii) K-NN
iv) Decision Tree
Pattern Mining Techniques:
i) Frequent Pattern Mining
ii) Weighted Frequent Pattern Mining

```
If (Can_Avail_Concessional_Ticket)
   If(Is_Chart_Not_Generated)
      If (Is_CNIC_Valid)
         If(Is_Credit_Card_Valid)
         {
            // Reserve seat
         }
```

```
If(Is_Credit_Card_Valid)
  If (Is_CNIC_Valid)
     If (Can_Avail_Concessional_Ticket)
        If(Is_Chart_Not_Generated)
        {
           // Reserve seat
        }
```

| TID | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| 1 | Success | Success | Failed | UnExecuted |
| 2 | Success | Success | Success | Failed |
| 3 | Success | Success | Success | Success |

**Fig 1. Business Rules Optimization Concept**

There may exist some dependency among business rules to achieve the desired output. For that purpose, an effective way for complicated and big structures analysis is Dependency Structure Matrix (DSM) [14] that facilitates user to define the dependencies among entities and also to visualize them. DSM presents entities in the form of matrix. Consequently, complex systems may be analyzed in a simpler way. DSM includes rows and columns as system components and use to analyze the entities involved. DSM breaks down the complex system into finer subsystems / modules / elements to provide its understandable look. As a handy tool for system evaluation, DSM works well to capture the interactions and interdependencies between components of a system. To decompose the system into simpler subsystems and drawing correct dependency relationships is the real achievement of DSM. Adjacency matrix M can be treated as ‘n x n’ matrix, that depicts that either one node to other node is interconnected or not.

$$M(i,j) = \{ \text{'1' if there is edge connecting } i \,\&\, j\text{; else '0'} \}$$

In a large and complex system, when there may be hundreds or thousands of implemented business rules, it may become a hard task to prioritize them manually even by experts. Apart from that, it is also time consuming task and prone to errors as well. Also, using a manual approach, it may not be fully analyzed by those experts that when will be the best time to shuffle their priority that can boost up system performance and can prevent system resources from wastage.

Today companies are cost sensitive and moving towards cloud to lower their cost that offers ‘pay as you go’ model. Their purchased resources are if consumed fully along with the low wastage, it can profit them a lot. Our proposed approach can help organizations to make their systems cost effective by preventing system resources from wastage. Our main objective is to apply data mining techniques on event logs of executed business process instances in such a way that previously written hard coded sequentially executing business rules can be replaced dynamically with a possible optimized execution sequence. Hard coded execution sequence can slow down process's performance and may keep system busy in unhealthy tasks. Our approach makes the software engineers life easy by letting them to focus fully on writing the business rules instead of worrying about their best optimized execution order to enhance system performance. In the present work, event log is the process instances that left a trace in the log. Each trace defines business rules that were executed either successfully or unsuccessfully. An important point to be noted here is that, in our case, failure of business rule execution doesn't represent its absence; instead it shows that rule when evaluated gave a false result and was not executed successfully. Both 'Success' and 'Failed' shows the presence of an item/rule whereas 'UnExecuted' shows the absence of an item/rule execution as shown in Table 5. Each row refers to a single case and is represented as a sequence of events. A business rule can either be a simple 'IF' condition to confirm some variable value or it can be a time intensive and high resource consumption task e.g. database or a web service call.

Our proposed approach that uses data mining techniques can also help in optimization of business rules engine technology. Many corporations tend to use business rule engines to offer higher flexibility. An enterprise rule engine system segregates enterprise operations from automated system in which the enterprise operations are shown via policies and completed with the aid of a rule engine. It can sufficiently lower the designing, growing and delivering cost of software program. A business rule engine will process enterprise business instances using defined business rules and will execute the desired action if 'condition' is fulfilled. Normally, a group of rules' evaluation and execution will be done that is referred as a 'rule set'. Let suppose, given a consumer transaction for reserving a seat, there can be different business rules for a discount.

i) If the passenger is above 60 years, he can avail 'old citizen concession' facility.
ii) If the passenger is a student, having a valid student card and allowed by institute to travel can avail 'student concessional ticket'.
iii) If a passenger is an employee of same service providing transportation company, he/she can get 70% discount.

## 1.1 Motivational Scenarios

This section describes some examples of real life scenarios that motivate to have an efficient execution of business processes in which desired business rules need to be adjusted dynamically as per the running environment and the nature of users frequently communicating with the system.

### 1.1.1 Business Process Optimization

Inside a large application, high number of business rules is implemented by software engineers. These rules may belong to a vast domain of business applications, e.g. a public facing website that offers a facility to reserve tickets online for travelling. Process may contain hard coded implemented business rules that are executed sequentially for each business instance. First implemented business rule confirms that either a customer is eligible or not to avail concessional ticket and it is verified by an external web service call. Second implemented business rule verifies customer's CNIC via an exposed NADRA web service call. Third business rule verifies the train availability for booking via a Database (DB) call. Fourth implemented business rule makes a DB call to verify that either provided CNIC is of Journalist or not. Fifth implemented business rule makes a DB call to confirm that the attached coach with train is still available. Sixth implemented business rule makes a DB call to verify that a customer has enough funds in his/her bank account via the provided credit card information.

The implemented business rule for credit card authentication is at No.6. It was observed that for more than 70% customers visiting the site, their credit card authentication is failed. It can be seen that due to failure of 6th business rule, not only those passengers were unable to reserve seats but also system resources were wasted in evaluation of 5 business rules placed before that. It has also been seen that most of the time passenger's credit card is not fully enabled by the corresponding issuing bank to make online transactions. It justifies that after going through this whole process but failing to reserve seats at the end will frustrate customers and they may not like to use system for online bookings. Other reasons of credit card transaction's failure may be due to suspicious possible fraudulent transactions. Although such users were unable to make transactions by trying to penetrate into system but the fact is that system resources were consumed and wasted in evaluation of 5 business rules coming before the 'credit card authentication' business rule. What if by keeping in view the cost and frequency parameters, credit card authentication is done first?, so that if it failed, next 5 rules will not be evaluated and thus will save the consumption of resources being spent on next 5 rules. It is also possible that in future, some other frequent pattern is followed during the communication of users with the system. So the more smart approach to make the system flexible would be to adjust the business rules execution order dynamically as per the running environment. Resultantly, a business process could be optimized. Following advantages could be achieved.

- It will provide customers better user experience.
- It will make business processes efficient to execute.
- It will help analysts in identification of step(s) that needs strong focus, careful observation and proper coordination among teams.
- It will help organizations to involve relevant teams for implementation of needed identified fraudulent checks in their applications.
- It will prevent system resources from wastage and their consumption in an unhealthy task(s).

### 1.1.2 Market Basket Analysis

We can see in our daily life that different shopping sites offer various product purchasing deals. There are multiple deals offered on an online shopping application and each deal contains different number of products. Let suppose that Deal#1 contains 15 products. A customer visits the site and analyzes each product inside a deal sequentially to view its specifications. Once he/she is satisfied with all the products inside a deal, he/she purchases that deal. It was observed that among other deals, Deal#1 was most visited by customers but was not availed. Also, it was noted that gradually the number of users visiting the site got reduced. May be customers preferred to visit some other sites and did shopping from there. After careful and detailed analysis, it was identified that during review of products inside Deal#1, the customers were satisfied with first 14 products and showed their interest in buying them but due to the fact that the last one was not of their interest, they did not prefer to buy that deal. Also, it could be annoying to customers that after going through a bit time consuming process and due to the fact that first 14 products were of their interest but the presence of last disinterested product forced them to leave the deal. Facing the same issue frequently for multiple deals might be the reason to make them unhappy with the site user experience and they preferred to do shopping from some other site. What if by keeping in view the interest of frequent customers, either that product no 15 should be excluded from the deal or need to be placed at the top in a deal, so that it can be viewed initially by the customers and if it didn't appeal them, they can jump instantly to review some other deals. It will not only help them in saving their time but will also keep their interest to remain on the site for reviewing some other deals, thus there will be higher probability of some purchases on website. This approach may help to achieve following advantages.

- It will enhance user experience.
- It can help marketing department in identification of less likely product &grouping of more likely products among the deals, thus enhancing their buying probability.
- It can help organization in identification of a product that needs more concentration related to improving its quality.
- It will also prevent system resources from being consumed in an unhealthy task(s).

### 1.1.3 Medical Science

Most of us commonly see patients that have different diseases and are admitted in a hospital. Some of serious among them are continuously under treatment and needs high care. When a new patient comes to a doctor, a long list of tests is suggested to him/her in order to identify his/her possible disease. A patient visits each lab one after another. Let a patient was asked for “10 tests” and for each test he may need to wait in a long queue. He was guided to report back to doctor as soon as he is notified about his first test positivity, so that his treatment could be started instantly. On an average, for each test he is standing for almost 20 minutes in a queue. Don't forget that he is already a patient and for him/her it may not be easy to wait 20 minutes for each test in a long queue. After analysis of large number of patients' data, it has been observed that majority of patients have negative results for the first 9 tests but the blood test conducted at the last is positive. If we do some calculations, it can be seen that in previous case, that patient waited for 200 minutes when he came to know about $1^{st}$ test positivity but his waiting time could be reduced if blood test is being conducted at first. This careful analysis can help policy makers to optimize the process by changing the sequence of tests' conduction. It will not only save the rush inside hospital but will also help in the fast treatment of patients. What if, by keeping in view the attributes of patients e.g., their age, gender, weight, disease etc., if there is higher probability that a patient will have positive blood test, the blood test is conducted first among others? Following advantages could be achieved.

- It will save patient's time along with providing him/her relief.
- It will also help in saving hospital's resources.
- It can help in identification of diseases that are frequent among patients. It will motivate doctors, researchers & Government organizations to device preventive strategies.
- It will help hospitals to make sure that high demand medicines are available in stock.

## 1.2 Problem Definition

Let consider, 'R' be a set of rules: $R = \{R_i\}$, $i \in \{1,2, ...,N\}$, where N represents number of rules and $R_i$ denotes a single rule. Each single rule $R_i$ is comprised of *condition* part and the *action* part. In a similar way, business instances can be represented as $\{B_j\}$, $j \in \{1,2,..., M\}$, where $B_j$ is a single instance and 'M' represents the number of instances. For a single instance $B_j$, one by one the rules will be evaluated and their action part will be executed in a given sequential order. If *condition* part of $R_i$ is satisfied for the instance $B_j$, we say that $B_j$ activates $R_i$. If it is supposed that there is no dependency among rules then they can be executed in any order and it will not affect the final execution result. Similarly, $E = \{E_{ik}\}$, $k \in \{1,2, ...,T\}$, where $E_{ik}$ denotes the execution sequence number 'k' for rule 'i' among rules 'N'. For a given business instance, rules evaluation cost will vary depending upon which rule in a sequence was not satisfied and caused the business process to fail. Cost of each business instance will be the resources consumed on evaluation of 1st to the nth failed/success business rule. Successfully completed business process is an ideal scenario for us and justifies the correct use of consumed resources, so we will not focus on such cases. Instead our focus will be on the failed business instances logs to extract useful knowledge from them. For simplification purpose, we are treating the cost of each business rule's evaluation as single unit of time because if there are 'n' number of web service or database calls that are almost doing the same nature of work, they will have approximately same cost provided that if other parameters like network state, internet speed etc. remains the same. Let there be 'n' number of business rules involved that must be satisfied for a successful business process execution. We are interested in a case that contains rules '1,2,...,x,y,...n'. '1….x' business rules are executed successfully but the rule next to 'x', that is 'y' is not satisfied and it resultantly caused the whole business process to fail. The more the business rules are evaluated, the higher will be the cost. The cost spent on processing the business process instance $B_j$, can be denoted as the *condition evaluation cost* $C(B_j,R,E)$ which is decided on the basis of *Rules 'R'* and their *execution sequence 'E'*. An efficient system will have low *mean condition evaluation* cost. In case of failed business process instance, an early failure of business rule is preferred. By keeping the more important rules to be executed first, we can save the cost of system and can make it more productive. For 'M' business instances, mean condition evaluation cost can be calculated as

$$C_M = 1/M \sum_{k=1}^{M} B_k \text{………………………………………….. (1)}$$

$$\text{Cost of } B_k = \sum_{i=1}^{N} (\text{Cost of } R_i) \text{ …………………….. (2)}$$

A smaller $C_M$ will represent lesser consumption of system resources and vice versa. Via using this statistical measurement '$C_M$', we can make comparison of execution performance of different applied approaches. Following objective function needs to be optimized.

$$C^* = \text{Min}(C_M) \text{ ………………………………………….…. (3)}$$

Among business rules execution sequences, the one that will have lower C* will be considered as an optimized sequence.

## 1.3 Challenges

At enterprise level, application software can contain a large number of business rules that need to be fulfilled in order to successfully execute a business process. In a normal scenario, when a requirement comes to adjust new business rule inside an application, a software engineer tries to prioritize and put it in its best supposed place inside the code as per his best knowledge. Those set of rules execute in that hard coded sequence continuously till either the system is running or the new rule comes for adjustment. Although it may change the previous order but will again make a system continue to execute business rules in the same new sequential order. Each person has different mind filter and different mindset people will interact differently with an application that is exposed to public. Thus a hard coded business rules execution sequence that is executing frequently but not allowing process to complete can degrade system performance and make system resources to be wasted in an unhealthy activity.

Thus, in summary there are two main issues with the hard coded written business rules execution sequence.

1. Lowering system performance.
2. Wastage of resources.

It is because of these issues that researchers are trying to find alternative techniques for business rules optimization / prioritization. One approach can be to replace hard coded written sequence with the optimized business rules execution sequence that will dynamically change as per the running environment. If the business rules execution logs are mined and analyzed that what nature of users are frequently interacting with the system, it may help in enhancing system performance. Also an optimized business rules execution order can be predicted that will allow the system to first execute those business rules that are considered more important in-terms of execution importance / cost. As per set time frame, the sequence will dynamically be replaced with the new optimized predicted one. It may not only prevent system resources from wastage but can also boost up system performance.

## 1.4 Research Objectives

Our main objective is to study business rules in the form of *"if condition then action"* to propose a model that will mine business process instances event log and will suggest a possible optimized business rules execution order to prevent system resources from wastage and to enhance system performance. It will also prevent to execute business rules always in same hard coded fashion as written by programmers. Instead our model will take care of it that which order is the most suitable execution order and which rules need to be executed first that can help in preventing system resources from wastage. Logs will be mined for specific time duration and then the predicted business rules order will be executed for a next decided time frame. Same process will be again executed after specific time to mine the most recent logs and to execute business rules as per new predicted execution sequence for next time frame.

## 1.5 Outline

The chapters of the thesis are organized as follows.

***Chapter 2:*** Chapter 2 discusses the background and different ways to prioritize rules execution. This chapter also explains the different classification and pattern mining techniques in detail to mine the data logs.

***Chapter 3:*** This chapter presents the related work which has been carried out in recent years on rules prioritization/optimization and also related to applying pattern mining and classification techniques on different problems.

C***hapter 4***: This chapter describes the work that we performed to design a solution for the problem. A small data sample has been taken as an example to convey the idea in a simpler way.

***Chapter 5:*** In chapter 5, different experimental results after applying pattern mining and classification techniques on actual data logs are shown and discussed.

***Chapter 6***: In this chapter summary of research work is presented, along with the future work. This chapter also presents the contributions of this research work and its limitations.

# CHAPTER 2

# BACKGROUND

In this chapter, we discussed background and the concepts involved related to different pattern mining and data classification techniques.

## 2.1 Business Process

A business process is a collection of tasks that are executed in a specific sequence in order to accomplish a particular business goal. In 1776, an economist Adam Smith provided a description of business process. He was the first one to give an idea that how output could be increased by the use of labor division. Examples of business processes include registration, online ticket reservation, opening an account in a bank, checking account credit, billing process, approving an order etc. An automated system or software implements large number of business processes to make them fast, accurate and efficient.

## 2.2 Business Rules

Business rules define the constraints that must be satisfied in order to achieve business goal. Those can be applied to business processes, people working in an organization, automated systems etc. A business process automated inside software can have number of business rules that must be satisfied for its successful completion. Every organization wants that all business constraints must be satisfied in order to execute business process successfully. If any of the rules is dissatisfied, the business process should not be successfully executed. If it is executed then the system may have some bug or security breach. An example of business rules is following.

```
IF Is_Any_Pending_Order_Exists THEN    …………………………………… Business Rule 1
    IF Is_CreditCard_IsValid THEN          ………..………………………    Business Rule 2
        IF Is_Valid_Journalist_Pass_Code THEN ………………. ……………. .  Business Rule 3
            IF Is_User_Railway_Employee THEN              …………………….  Business Rule 4
                IF Is_User_Has_Valid_Railway_Concessional_Pass THEN ……..  Business Rule 5

                    // Reserve seat for a passenger (Business process).

 END
     END
         END
             END
                 END
```

## 2.3 Pattern Mining Techniques

Following are the famous types of pattern mining techniques i) Frequent pattern mining ii) Weighted Frequent pattern mining.

### 2.3.1 Frequent Pattern Mining

Frequent pattern mining is considered as an important research area in the field of Data Mining. Frequent patterns are considered as those patterns that fulfills the *minimum support* threshold in a data set. Let suppose, a set of items, which include pen and ink are frequently appearing together in a data set, so it will be considered as frequent. By generating frequent patterns from data, previously unknown knowledge can be identified that can lead organizations towards better planning and best decisions.

Association rule is identified on the basis of two measures called, *support(s)* and *confidence(c).* Mathematically, support and confidence can be represented as $s(A)= P(A)$, $s(AB) = P(AB)$ and $c(A\rightarrow B)=P(B/A) = P(AB)/P(A)$. Let $I = i_1, i_2, i_3, ..., i_m$ be considered as set of items and $D = t_1, t_2, ... , t_n$ is an input dataset in which $t_i$ is treated as a transaction. An association rule is not anything however an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$

To mine frequent patterns, Agarwal et al. [5] proposed *Apriori algorithm* that uses a breadth-first approach but multiple times scan of database creates an amazing overhead. Despite the fact that Apriori algorithm itself has made a few optimizations, the time and area intake is large and efficiency is not high. Ordinary speaking, it is miles hard to avoid three main defects [16]. 1) It generates a large number of candidate items. When producing all of the candidate *k-itemsets* $C_k$ by using frequent *(k-1) itemset I*, it needs multiple connections and large number of comparisons are made. The time complexity of connection constraints evaluation is $O(k*x^2)$. 2) It scans the database multiple times. While 'k' is 1, 2, 3....n, the database is at least scanned 'n' times. 3) It spends a lot of time in candidate item sets matching. On contrary to *Apriori*, *FP-growth* uses divide-and-conquer approach that makes it much faster than the *Apriori algorithm*. It makes use of a compact data structure called FP-tree to extract frequent item-sets without the generation of candidate item-sets.

## 2.3.2 Weighted Frequent Pattern Mining

In real applications, different items have different importance but frequent pattern mining treats all items uniformly. In some areas, real data has distinct importance, as some of the objects may give extra benefits in terms of income or profit, some may be putting extra cost etc. As an instance, sale of shirt may generate 20 cents profit, where as a cotton of shirts might generate a 40 cents profit. Although latter have a low support count but in-terms of profit, it will have greater advantage than the other. This is the reason that the idea of weighted frequent pattern mining came into existence.

Let$i_1$, $i_2$,..,$i_n$ denoted by '*I'* be a set of items and *'D'* can be considered as transactional database which represents a set of transactions in which every transaction *'T'* is a set of items such that 'T*'* subset *'I'*. An identifier, called 'TID' is attached with each transaction. An item-set that contains *'k'* items in it is called k-itemset. The set {shirt,comb} is a 2-itemset. Let *'W'* is a set of non-negative real numbers. A pair(x,w) is known as a weighted item and 'w' is the weight associated to 'x'. Weighted support [17] computation includes the multiplication of measures; *support* and *weight*. If *support* and *weight* are taken into consideration individually, then mining might bring about item-sets that need to have both the enough *weight* and also the *minimum support* count. However, this will bring about loss of interesting patterns. For instance, a product that is going under promotion and having its sale close to zero will be pruned as it does not have enough support. *Weighted support* for an itemset X can be calculated using Eq. 4.

$WSup(X) = W(X) * Sup(X)$ -------------------------- *(4)*

*W(X)* is the weight of itemset *X*, whereas *Sup(X)* presents support count of an itemset *X*.

$W(X) = Maxw(x);$ *where* $x \in X$ ---------------------- *(5)*

whereas W(X) is maximum weight of an itemset 'X'. E.g. among {pen,ink,book} pen has weight 5, ink has 9 and book has 8 then W(pen,ink,book) = 9;

## 2.4 Classification Algorithms

Classification problem has its own roots inside the area of data mining, database and information retrieval. Classification problem is described as, a training records represented as $D = X_1, X_2,...,X_n$, such that a class label is assigned to every record that can have 'k' unique discrete values listed by $1,2,...,k$. Training data is used to build a classification model and a trained model makes a prediction to predict a class label for the new input test instance for which class label was previously unknown. Some widely used classification algorithms that are applied in the proposed work are following.

### 2.4.1 Decision Tree

Decision Tree uses the splitting criteria to classify the instances. A flow chart like tree structure is built to classify the instances by breaking down the dataset into smaller subsets. Decision tree is capable to handle both categorical and numeric data. There are many decision tree algorithms e.g. C4.5, ID3, CART etc. In a Decision Tree, each node represents an attribute in an instance that is going to be classified by sorting them based on feature values [19]. An output tree is a combination of 'decision nodes' and 'leaf nodes'. *Entropy* and *Information Gain* are used to build a Decision Tree. *Entropy* measure is used in order to calculate the homogeneity of a sample. It will be '1' if the sample is equally divided and will be '0' when the sample is completely homogeneous. Entropy can be calculated using the following formula.

$$E(S) = -\sum_{k=1}^{n} p_k log_2 p_k$$

*Information Gain* can be calculated using following eq.6.

*Gain(T,S) = Entropy(T) -Entropy(T,S)* ---------------------------------------- *(6)*

*Information Gain* is used as an important measure to find the best split attribute. Attribute having high *Information Gain* will be selected as an attribute to split the data. After the calculation of *Entropy* for each branch, it will be then added proportionally to obtain *Total Entropy* for the split *Entropy(T,S).* Resultant Entropy is subtracted from the Entropy before the split *Entropy(T)* of data to get the *Information Gain(T,X).*

## 2.4.2 Naive Bayes

The *Naive Bayes* classifier is considered as probabilistic classifier that is based on to apply Bayes' theorem with strong (naive) independence assumptions. It works by calculating probabilities for hypothesis and is robust to noise as well. A *Naive Bayes* classifier assumes that given the class variable, the presence or absence of a selected attribute is unrelated to the presence or absence of any other attribute. It is a fast algorithm to build a model and can be applied on large data-sets. It finds the conditional possibility of every document to belong to each class. The conditional probability P(C|D), where 'C' is a classes and 'D' is the description of objects that needs to be labeled. Given a description ‘d’ of an object, we assign the class $argmax_c P(C=c|D=d)$.

$$argmax_c P(C=c|D=d) = argmax_c P(D=d|C=c)P(C=c)/P(D=d)$$

In order to determine maximum a-posteriori class, the denominator P(D=d) is considered as normalizing factor and can be ignored, as it is not depending on the class. An important term is P(D=d|C=c), that is probability of the description given the class.

## 2.4.3 K-Nearest Neighbour Algorithm (K-NN)

It is considered as the simplest classification algorithm that is easy to apprehend. It identifies the class of an unknown data point based on its nearest neighbour whose class is already recognized [20]. In *k-NN* the ‘*k’* nearest of neighbours are to be taken into consideration to define a class of a data point. If 'k' is '1' then the class of its nearest neighbouris assigned to the object. If k>1 then 'k' nearest neighbours are selected and the majority class of neigbours will be assigned the class to the new object. Euclidean distance is used to define the closeness, where Euclidean distance between two points, $X=\{x_1, x_2,\ldots,x_n\}$ and $Y=(y_1,y_2,\ldots,y_n\}$ is

$$d(X,Y) = \sqrt{\ } \sum_{i=1}^{n} (Xi - Yi)^{2}$$

'X' and 'Y' are the two objects that are compared to find the distance between them and 'n' is their number of attributes.

### 2.4.4 Support Vector Machine (SVM)

*SVM* is supposed as one of the effective technique for regression, pattern recognition and classification problems. *Support vector machine* is a supervised learning technique which classifies points to one of the two disjoint half-spaces. It is taken into consideration as a good classifier due to its high generalization performance without needing any prior knowledge. *SVM* is based on the idea of decision planes that define the decision boundary between classes called support vectors, treated as parameter [19]. The main objective of *SVM* is to find the best classification function that can accurately classify the instances. It can be used for both linear and non-linear data [20] to separate it between two hyperplanes. Linear data can be easily separated whereas it becomes quite challenging task to distinguish non-linear data between classes. *SVM* struggles to maximize the following function w.r.t '$\overline{w}$' and 'b', so that it can be ensured that maximum margins hyperplanes are found.

$$L_p = \frac{1}{2} \|\overline{w}\| - \sum_{i=1}^{n} \alpha_i y_i (\overline{w}.\overline{X}_1 + b) + \sum_{i=1}^{n} \alpha_i$$

where 't' is considered as the count of training examples and , i=1,...,t are non-negative numbers and '$L_p$' is called the Lagrangian, whereas the vectors '$\overline{w}$' and constant 'b' defines the hyperplane.

## 2.5 Dependency Structure Matrix

There may exist some dependency among business rules to achieve the desired output. For that purpose, an effective way for complicated and big structures analysis is DSM [14] that facilitates user to define the dependencies among entities and also to visualize them. DSM presents entities in the form of matrix. Consequently, complex systems may be analyzed in a simpler way. DSM includes rows and columns as system components and use to analyze the entities involved. DSM breaks down the complex system into finer subsystems/modules/elements to provide its understandable look. As a handy tool for system evaluation, DSM works well to capture the interactions and interdependencies between components of a system. To decompose the system into simpler subsystems and drawing correct dependency relationships is the real achievement of DSM. Adjacency matrix M can be treated as 'n x n' matrix, that depicts that either one node to other node is interconnected or not. '0' which shows that they are same one and are not dependent. Relationship is represented as '1', if objects are dependent. If a rule implementation got changed and started to depend on some other rule, dependency matrix will be updated respectively. Business rules order could be changed only if the desired dependency among them using the DSM is met.

*M (i,j) = {'1' if there is edge connecting I & j; else '0'}*

| | **A** | **B** | **C** |
|---|---|---|---|
| **Object A** | 0 | 1 | 0 |
| **Object B** | 0 | 0 | 1 |
| **Object C** | 0 | 0 | 0 |

**Table 1: DSM as a Diagonal Matrix**

## 2.6 Summary

In this chapter, we have discussed background and the concepts involved related to business process, business rules, DSM, different pattern mining and data classification techniques.

# CHAPTER 3

# RELATED WORK

In chapter 3, we have discussed important contributions by researchers that have been carried out in past few years. The related work enables us to gain vision into this research work and identifies the research gap in the existing systems.

Rule prioritization/optimization is not a new area but as per our best knowledge a limited work has been carried out in this field.

## 3.1 Rule Prioritization

Broadly speaking, prioritizing business rules for a business process can either be done explicitly by the user or can be automated by system. For each business instance in a business process, different business rules sequence can exist, as shown in Fig 2.

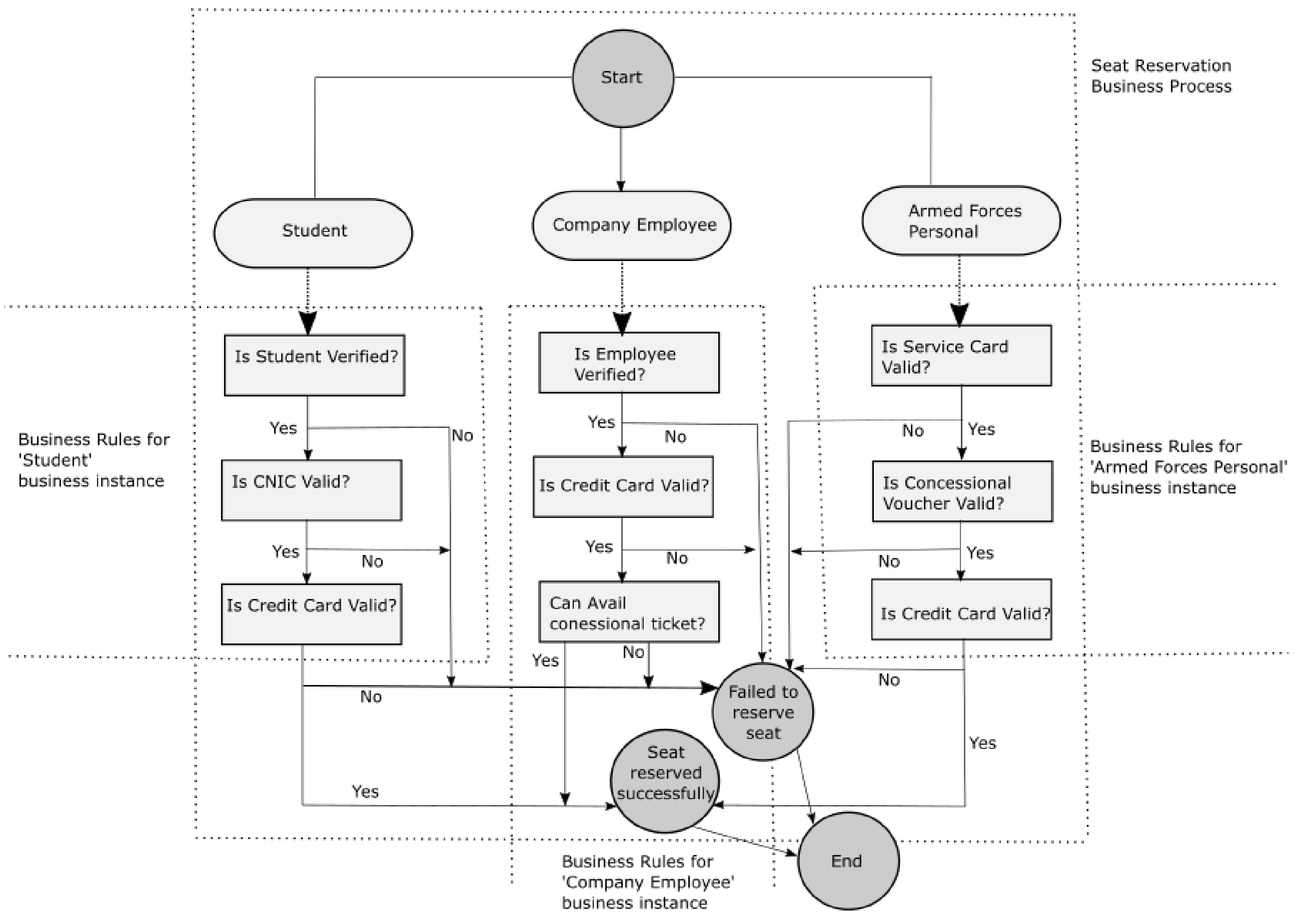

Seat Reservation Business Process
Start
Student
Company Employee
Armed Forces Personal
Is Student Verified?
Is Employee Verified?
Is Service Card Valid?
Business Rules for 'Student' business instance
Is CNIC Valid?
Is Credit Card Valid?
Is Concessional Voucher Valid?
Business Rules for 'Armed Forces Personal' business instance
Is Credit Card Valid?
Can Avail conessional ticket?
Is Credit Card Valid?
Yes
No
Failed to reserve seat
Seat reserved successfully
End
Business Rules for 'Company Employee' business instance

**Fig 2. Seat Reservation Process**

In an environment where an activation of one rule causes all other rules to be reevaluated makes a system less efficient. [21] work includes the objective to minimize the condition evaluation count to improve the execution efficiency of rule engine in such environment. Two of famous used approaches discussed in literature are as following.

### 3.1.1 Default Priorities

Embedded rules have default relative priorities which are the feature of static properties of the business rules. This function *p*, defines a default total order over the business rules. A function yielding the creation time-stamp of the rules (assuming creation timestamps are specific) is an instance of this sort of features which offers higher precedence to older business rules. A default total order is presented by 'd' such that, given two rules R & S, if P(R) >P(S)then R → d → S [28]. This approach has a serious drawback, it has a high computing cost that makes it inefficient for a large system.

## 3.1.2 User Defined Priorities

Software engineer or user may explicitly specify relative priorities among particular rules via defining a relationship among them. If the user has mentioned that rule ‘R’ precedes ‘S’, then ‘R’ is taken into consideration first for execution, irrespective of the default general ordering. User-defined priorities are transitive; this is, if ‘R’ precedes ‘S’ and ‘S’ precedes ‘T’, then ‘R’ precedes ‘T’ even though ‘S’ is not executed [28]. The work in [27] concluded that while activation of rule among rules during conflicting state, it is necessary to specify a numeric priority for each rule. User defined priorities override default priorities. Cycles are not allowed in user-described priorities. User defined priorities are dynamic and they may be dropped and added at any point during existence of rule set. Naive approach is used to implement business rules sequence as per the best knowledge of system analysts and software developers, so that if business process fails, low system resources could be wasted.

```
Set Rl = R1,R2,...,RN as all hard coded rules sequence
    For j=1 to T, following process for $B_j$ needs to be executed
        Execute R1 then R2 and so on, that are activated by $B_j$
    End.
End.
```

If all needed rules are successfully evaluated and executed, we call it ‘a business process completed successfully’. If any of hard coded rules ‘Rl’ was not satisfied, we call it ‘a business process failed to complete’. We are more interested in such cases, ’when business process failed and high system resources were consumed’.

User defined priorities has also a serious drawback. In a large and complex system, when there may be hundreds or thousands of implemented business rules, it may become a hard task to prioritize them manually even by experts. Apart from that, it is also time consuming task and prone to errors as well. Other than that, using a manual approach, it may not be fully analyzed by those experts that when will be the best time to shuffle their priority that can boost up system performance and can prevent system resources from wastage. Today companies are cost sensitive and moving towards cloud to lower their cost that offers ‘pay as you go’ model. Their purchased resources are if consumed fully along with the low wastage, it can profit them a lot. Hard coded execution sequence can slow down process's performance and may keep system busy in an unhealthy tasks. There must be some approach that can help organizations to make their systems cost effective by preventing system resources from wastage.

The study in [29] proposes that how the execution of production rules can be made efficient which try to reduce the cost of computation. A naive approach is to reevaluate all the rules that are not executed and then select one among them for activation that meet the criteria. This approach is considered safe but has equally high cost of computation. An important method proposed in [27] is a RETE Algorithm for enhancing system performance. It has network of nodes where each node (other than root) presents the pattern that is coming in a *condition* of a rule. It has two important features that make it faster, one is 'state saving' and the other one is 'sharing among similar conditions' but when it comes to situation where there is limited sharing among conditions, RETE algorithm is not applicable for that. This limitation make it useless where there is limited sharing among the given conditions.

## 3.2 Pattern Mining Techniques

There are number of frequent pattern mining techniques that can uncover hidden knowledge from the event logs. Data Mining is considered as one of the most important areas in which raw data is converted into meaningful and purposeful knowledge [4].When it comes to efficiency, strong need of an efficient algorithm arises that can cover the drawback of existing available algorithms. [6] present an important comparative study on various data mining algorithms in order to generate frequent patterns. Some was having problems of large size candidates generation, some having better efficiency and some having multiple DB scans etc. Some algorithms like DHP, Apriori, FP-Tree, Pincer and DIC search algorithm works horizontally on dataset while Eclat and Partition algorithm works vertically. Algorithms except Pincer uses bottom-up search while Pincer search algorithm uses Hybrid method. DHP, Apriori and Pincer search algorithm used Hash Tree data structure while FP-Tree uses FP Tree data structure. Prefix Tree data structure was being used by DIC while Partition and Eclat uses no particular data structure. Besides FP-Tree, all algorithms generate candidate item sets. [5] deals with enhancing the durability and efficiency of frequent pattern mining on NVM-based memory systems by keeping in view their properties. [18] suggested an efficient algorithm to mine weighted frequent itemsets in a dataset. It uses hash table data structure and uses a single database scan. To find frequent patterns, [7] proposed a balanced approach that avoids methods in which either very high or very low pruning is made. They suggested a "weighted support" measure that not only considers the association and dissociation among items but also the ‘null’ transactions impact as well in order to generate frequent patterns. [8] proposed togetherness instead of support for generation of frequent patterns. As per their definition, for all 1-itemsets, cost of togetherness is 1 and as a result all 1-itemsets are considered as frequent regardless of the *transactions* and *min_sup*. It shows that all 2-itemsets are considered as applicants in 2-items candidate generation and their support will be calculated. While using togetherness, a higher order itemsets in a good quantity are probably to be qualified as frequent itemsets.

Many researchers [9] [10] worked on to modify Apriori and Apriori-like algorithms to handle binary attributes as presented in Table.2. Each row presents a transaction and column denotes an item. ’1’ denotes the presence and '0' denotes the absence of an item in a transaction. To mine weighted frequent-regular item sets, [11] proposed a tree-and-pattern growth algorithm called WFRIM (Weighted-Frequent-Regular Item sets Miner) that uses FP-tree like structure named WFRI-tree to maintain candidate item sets during the process of mining. The work of Haiyun [12] proposed an approach of recommendation based on identified purchase patterns by analyzing the purchase history of users. In order to predict next category of purchase in a particular location, those identified patterns are used. [13] presents a comparative analysis of widely used algorithms and their variations like Apriori, Tree-projection, FP growth and Eclat for finding both frequent and weighted frequent patterns.

| **Item1** | **Item2** | **Item3** | **Item4** | **Item5** |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |

**Table 2. Binary Attributes Presentation**

## 3.3 Dependency Structure Matrix (DSM)

. Dependency matrices can be made by analyzing documents collected during services, historical data etc. [2]. As an important tool used for system analysis, DSM captures interdependencies/interactions between system elements [14]. Dependency relationships analysis & correctness is the one of the top most feature of DSM. There may exist some dependency among business rules to achieve the desired output. For that purpose, an effective way for complicated and big structures analysis is DSM that facilitates user to define the dependencies among entities and also to visualize them. DSM presents entities in the form of matrix. Consequently, complex systems may be analyzed in a simpler way. DSM includes rows and columns as system components and use to analyze the entities involved. DSM breaks down the complex system into finer subsystems/modules/elements to provide its understandable look. As a handy tool for system evaluation, DSM works well to capture the interactions and interdependencies between components of a system. To decompose the system into simpler subsystems and drawing correct dependency relationships is the real achievement of DSM. Dependency matrix can be represented in the form of diagonal matrix as shown in Table 1. ‘0’ denotes that objects are independent, whereas ‘1’ denotes that they are dependent on each other.

## 3.4 Critical Analysis

It is concluded from the related work that a good number of datamining techniques are available in literature that can be applied to discover the hidden knowledge from a large amount of data. As per our best knowledge there is still a gap to apply those techniques for optimization of automated business processes. Softwares can be made more optimized thus allowing the hardware resources to be consumed in a healthy work.

The proposed research is to perform optimization of business rules using pattern mining techniques. It will also be an innovative step especially in IT industry / software houses. Programmers will be able to focus on the core logic instead of focusing on to suggest the best business rules execution order for system optimization and better resources utilization. Software will be no more completely hard coded, instead an automation wrapper will be existing above it to make it adjustable as per running environment.

## 3.4.1 Auto Priorities Assignment

Our proposed approach includes auto prioritization of business rules to cover such gap. While using naive approach, selection and execution of hard coded written rules in changing environment is not suitable. Instead, a system must be adaptive to adjust the business rules execution sequence as per the environment in which it is running. Their execution order should be prioritized and changed dynamically with the passage of time that is best suited for the nature of users frequently interacting with it. The core idea is to mine the business process instances logs and predict an optimized business rules execution sequence, if executed in that order can enhance system performance and prevent resources from wastage.

```
Set $R_l = R_i$ satisfying $P_i > P_i+1$ having optimized predicted sequence for a given process instance $B_j$
    For j=1 to T, following process for $B_j$ needs to be executed
        Execute R1 then R2 and so on, that are activated by $B_j$
    End
End
```

If all needed rules are successfully evaluated and executed, we call ‘business process successfully completed’. If any of Rules ’R’ was not satisfied, we call it business process failed to complete.

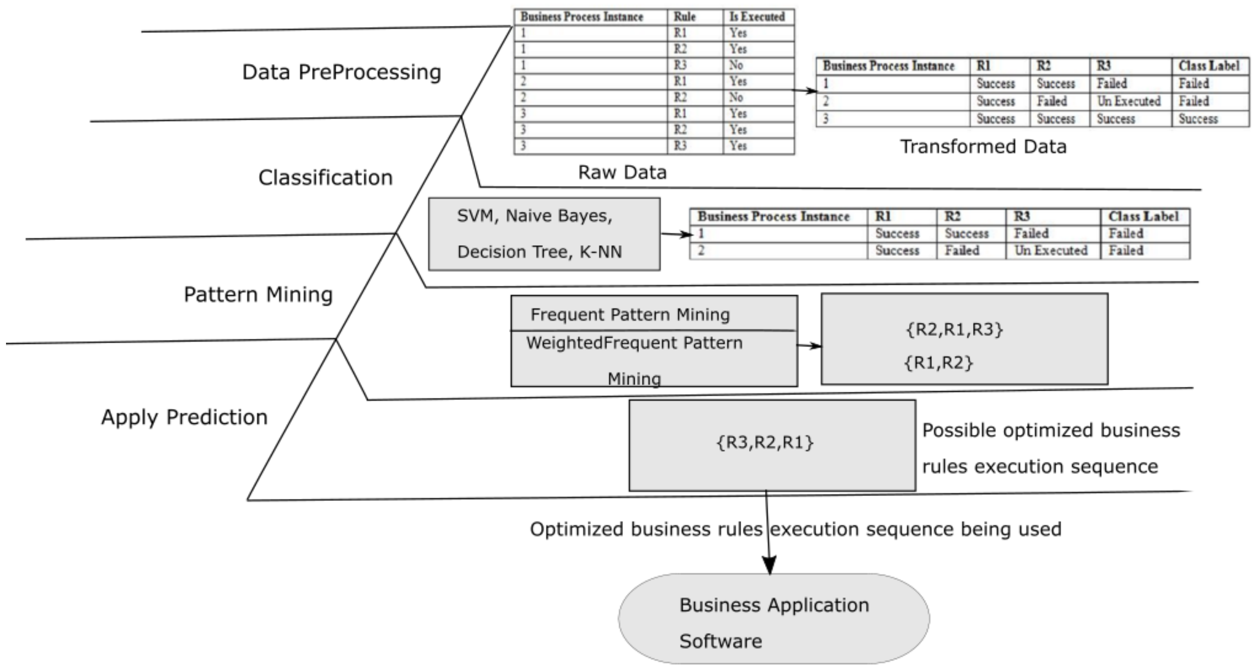
## 3.5 Summary

This chapter highlighted different studies related to rules prioritization and data mining techniques. The related work of important concepts (rules prioritization, frequent pattern mining, weighted frequent pattern mining and data classification) has also been discussed.

# CHAPTER 4

# PROPOSED METHODOLOGY

In this chapter, proposed approach to mine business rules execution data logs is discussed in detail. We have taken a sample dataset with few instances as an example to convey the idea fully. To reduce the search space, first data is classified and then pattern mining techniques are applied in order to find an optimized business rules execution order. If executed in that order, system resources can be prevented from wastage. Figure 3 illustrates different stages of our proposed solution.


Data PreProcessing
Classification
Pattern Mining
Apply Prediction
Raw Data
Transformed Data
SVM, Naive Bayes, Decision Tree, K-NN
Frequent Pattern Mining
WeightedFrequent Pattern Mining
{R2,R1,R3}
{R1,R2}
{R3,R2,R1}
Possible optimized business rules execution sequence
Optimized business rules execution sequence being used
Business Application Software

| Business Process Instance | Rule | Is Executed |
|---|---|---|
| 1 | R1 | Yes |
| 1 | R2 | Yes |
| 1 | R3 | No |
| 2 | R1 | Yes |
| 2 | R2 | No |
| 3 | R1 | Yes |
| 3 | R2 | Yes |
| 3 | R3 | Yes |

| Business Process Instance | R1 | R2 | R3 | Class Label |
|---|---|---|---|---|
| 1 | Success | Success | Failed | Failed |
| 2 | Success | Failed | Un Executed | Failed |
| 3 | Success | Success | Success | Success |

| Business Process Instance | R1 | R2 | R3 | Class Label |
|---|---|---|---|---|
| 1 | Success | Success | Failed | Failed |
| 2 | Success | Failed | Un Executed | Failed |

**Fig 3. Proposed Approach to Mine Data Logs**

Our approach consists of following steps.

## 4.1 Preprocessing

Preprocessing is considered as one of the most important phase in data mining. Achieving meaningful information from the raw data needs it to be preprocessed first. Data must be in a proper format so that data mining techniques could be applied efficiently on it. For understanding and simplicity purpose, we have selected 3 business instances' logs as a sample data as shown in Table.5. Actual data logs along with results are presented in Chapter 5. We first prepared and transformed the obtained data shown in Table 3 into a more suitable format as represented in Table 4. This data is in a format on which data mining techniques e.g. Classification &Pattern mining techniques could be applied efficiently.

| Business Process Instance | Rule | Is Executed |
|---|---|---|
| 1 | R1 | Yes |
| 1 | R2 | Yes |
| 1 | R3 | No |
| 2 | R1 | Yes |
| 2 | R2 | No |
| 3 | R1 | Yes |
| 3 | R2 | Yes |
| 3 | R3 | Yes |

**Table 3. Raw Data Log Format**

| Business Process Instance | R1 | R2 | R3 | Class Label |
|---|---|---|---|---|
| 1 | Success | Success | Failed | Failed |
| 2 | Success | Failed | UnExecuted | Failed |
| 3 | Success | Success | Success | Success |

**Table 4. Transformed Data Logs Format after Preprocessing**

## 4.2 Data Classification

Available logs contain both 'Success' and 'Failed' business instances data but we are interested in a 'Failed' instances log. As the ones those are succeeded are an ideal scenario for us and justify that resources are consumed in a fruitful activity (business process completion). So in order to work with failed business instances log, we need to classify the data. Once the data is classified and the failed business process instances are separated, those will be used for mining frequent and weighted frequent patterns. Classifying business process instances will also reduce the search space before applying pattern mining techniques and it can be considered as an optimization step before applying frequent pattern mining algorithms.

There are number of classification algorithms available in literature, among which we have applied Decision Tree, Naive Bayes, KNN and Support Vector Machine. Detailed results obtained on actual data logs are presented in Chapter 5. We have used Confusion matrix and other measures e.g. TP rate, FP rate, Precision, Recall, F-Measure etc. to summarize the results and describing the classifier's performance on a set of test data for which the True values are known. TP (True positive) and TN (True negative) are instances that are classified correctly by classifier as a given class and FP (False positive) and FN (False Negative) are the instances that are falsely classified by the classifier as a given class.

## 4.3 Frequent Pattern Mining

To mine frequent patterns in a dataset, we have proposed a compressed tree data structure to overcome the multiple DB scans problem and it can be used efficiently to mine the frequent patterns.

### 4.3.1 Construction of Compressed Tree

For understanding purpose we are considering a small set of transactions as a sample data to convey the idea fully. Actual data logs along with results are presented in Chapter 5. Let the Database 'D' containing the following set of transactions 'T'.

| TID | R1 | R2 | R3 | R4 | R5 | R6 | R7 |
|---|---|---|---|---|---|---|---|
| **1** | S | S | S | F | U | U | U |
| **2** | S | S | S | S | S | F | U |
| **3** | S | S | S | F | U | U | U |
| **4** | S | S | S | S | S | F | U |
| **5** | U | U | S | U | S | S | F |
| **6** | S | S | S | F | U | U | U |
| **7** | S | S | S | F | U | U | U |
| **8** | S | S | S | S | S | F | U |

**Table 5. Sample Dataset 'D' (Success=S, Failed=F, UnExecuted=U)**

Via single DB Scan, use each transaction $T_i$ to build a compressed tree. First transaction will be scanned and a tree will be constructed with *support* as 1. Failed business rule will be placed on right side and succeeded one will be on the left side of a tree path(s). Each tree path will indicate the occurrence of an itemset in a transaction. Each node will also maintain the *support* count. Likely, we will scan the remaining transactions to build the tree. Each node 'count' property will represent the number of times the rule either *succeeded* or *failed*. Each tree node will also contain 2 more properties. 'LNA' (left node address) and 'RNA' (right node address). Leaf nodes will have both pointing to 'NULL'.

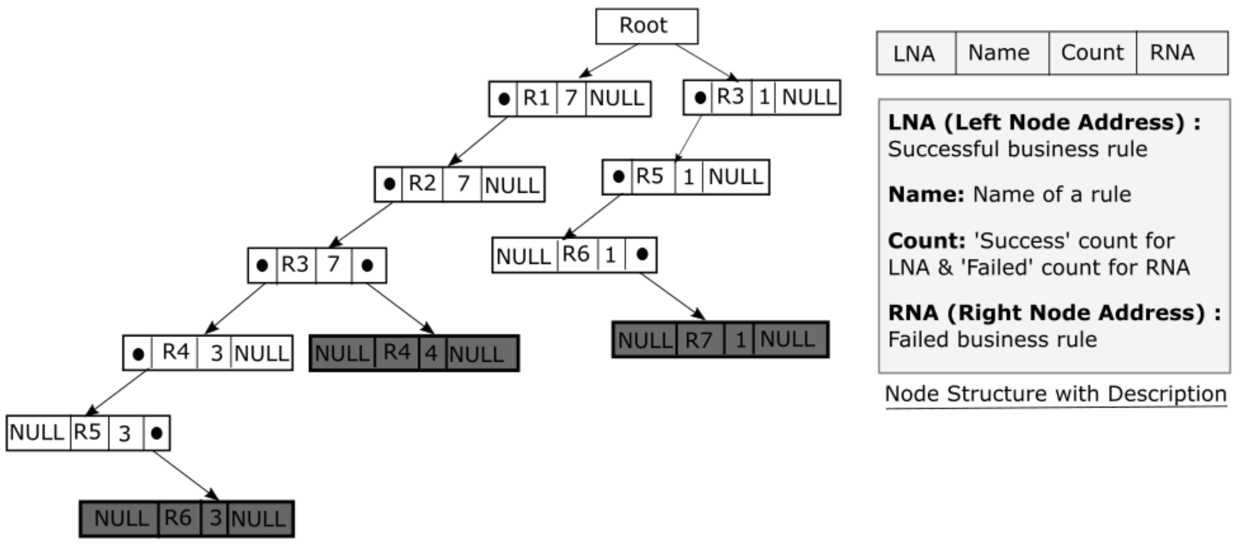

Root
● R1 7 NULL
● R3 1 NULL
● R2 7 NULL
● R5 1 NULL
● R3 7 ●
NULL R6 1 ●
● R4 3 NULL
NULL R4 4 NULL
NULL R7 1 NULL
NULL R5 3 ●
NULL R6 3 NULL
LNA Name Count RNA
LNA (Left Node Address) : Successful business rule
Name: Name of a rule
Count: 'Success' count for LNA & 'Failed' count for RNA
RNA (Right Node Address) : Failed business rule
Node Structure with Description

**Fig 4. Compressed Tree ‘T’**

## 4.3.2 Mining Compressed Tree

In the proposed work, to overcome the problem of scanning database multiple times, database mapping method has been changed. A compressed *support* count maintaining tree will be constructed via a single database scan that will be used for mining both *Frequent* and *Weighted Frequent Patterns*. Our approach uses two steps in predicting an optimized possible business rules execution order.

- Find frequent business rules patterns that are causing business process to fail along with maintaining the downward closure property.
- Pushing weight constraint to identify weighted frequent business rules patterns in order to suggest an optimized possible execution order, along with maintaining the needed dependency among them using DSM.

Two pruning conditions are i) *sup<min(sup),*the *support* of an itemset is less than the minimum set support threshold. ii) *Wsup(x)<min(sup), an* itemset weight multiplied with *support*

is less then minimum set support threshold. An itemset X is considered to be weighted infrequent itemset if either of the condition 1 or 2 is satisfied. If any itemset is not satisfying both of the conditions, then the itemset will be called weighted frequent itemset.

### 4.3.3 Identify Frequent Business Rules Patterns

[30] proposed an approach to find maximal length weighted frequent itemsets (MLWFI) by dividing the WFP (weighted frequent pattern) tree into smaller sub trees. In order to mine all frequent business rules patterns in a database for which business process has been failed, we shall mine the constructed Tree in Fig.4. In a tree, there may be multiple branches initiated from root node. We divide mining the whole tree into smaller trees which include both frequent and infrequent patterns. First of all, all branches starting from root will be extracted. After that, those will be mined by extracting all paths containing itemset starting from top node to right most leaf node, as 'failed' business rule will be existing on a right side of path.

Let suppose that the set *minSup* is 30% and minConf is also 30%. *Support* = P/|D|, whereas |D| is the 'length of DB' or 'total number of transactions' and 'P' is the 'no of transactions' containing the itemset and Confidence = c(A→B)=P(B/A) = P(AB)/P(A). Fig.6 presents the extracted *frequent* and *in-frequent* business rules paths from the tree.

As *R1, R2, R3, R4 (Support: 50%)* and *R1, R2, R3, R4, R5, R6 (Support:38%)* itemset fulfill the minimum support and confidence criteria, so those are considered as frequent patterns.

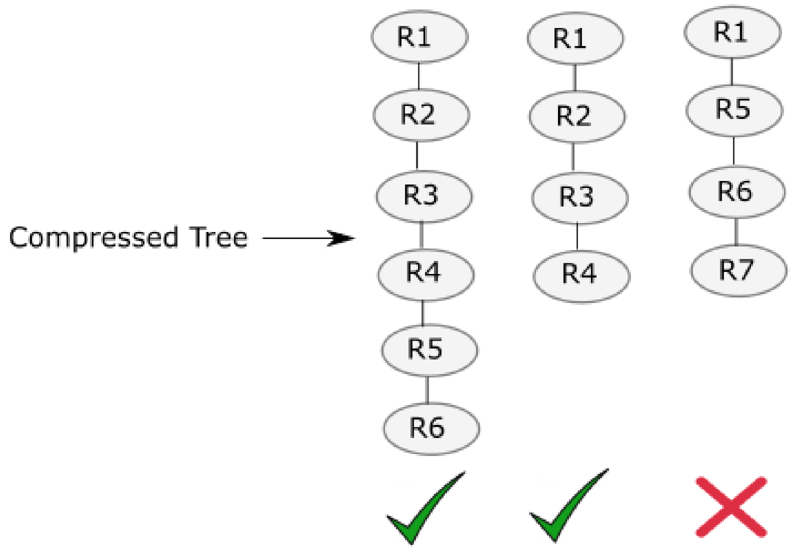


**Fig.5. Extracted Frequent & In-Frequent Paths**

## 4.3.4 Identify Weighted Frequent Business Rules Patterns

In terms of resource consumption, *k-itemset* that is *weighted frequent pattern* and also having higher *weighted support* will be the one that has higher evaluation cost among the identified *weighted frequent patterns*. Weighted infrequent itemsets will be pruned based on *weighted minimum support* set threshold. $w(x_i) = \{(w_1,x_1), (w_2,x_2),\ldots, (w_n,x_n)\}$ where $(w_i,x_i)$ is a weighted pair, showing the weight corresponding to each item $x_i$ and it will be the cost consumed starting from 1st index rule to $x_i$(failed business rule) and then taking its average to normalize it as shown in following eq. 7.

$$w(x_i) = \mathrm{Avg}(\textstyle\sum_{k=1}^{i} (w_i,x_i)) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. (7)$$

*As R1, R2, R3, R4* and *R1, R2, R3, R4, R5, R6* are identified as *frequent patterns/itemsets* and are further used to confirm that they are *weighted frequent as well*. For simplicity and understanding purpose, we are considering the cost/weight of each business rule as single unit of time e.g. '1' and minimum set *weighted support* threshold is 30%. Let first find the weight of each business rule using eq.5.

$w(x_i) = \{1/7,2/7,3/7,4/7,5/7,6/7,7/7\}$

$w(x_i) = \{0.1,0.3,0.4,0.6,0.7,0.9,1\}$

Therefore *R1, R2, R3, R4* (weighted support: 30%) & *R1, R2, R3, R4, R5, R6* (weighted support: 34%) are also identified as *weighted frequent patterns/itemsets*. Between identified weighted frequent patterns, *R1,R2,R3,R4,R5,R6* presents weighted frequent pattern that has higher probability of occurrence to cause business process fail along with consuming high system resources. Arrange the identified weighted frequent patterns in descending order w.r.t. their support as following.

$\mathrm{R1,R2,R3,R4,R5,}\overleftrightarrow{R6}$

$\mathrm{R1,R2,R3,}\overleftrightarrow{R4}$

Each identified weighted frequent path itemset has length N. Break each itemset path into two parts '1...n-1' & 'n'.

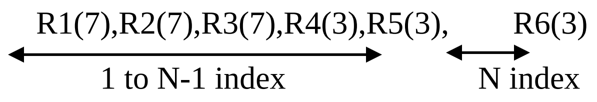


**Fig 6. Weighted Frequent Pattern Decomposition**

Extract nth index business rule (failed node in an extracted branch of a tree) from each identified *weighted frequent pattern* to generate new possible optimized execution sequence 'E'. A DSM in Table 6 will be used to confirm that desired dependency relationship is not violated among rules while prioritizing them. E.g. If R4 is dependent on R3, it cannot be prioritized higher than R3 to execute before it. Eq.8 presents predicted possible optimized business rules execution order by prioritizing the most important one at the top, if executed in that order can consume fewer resources in performing the same work as compared to previously used hard coded business rules sequence.

Optimized business rules execution sequence = E = R6,R4,R1,R2,R3,R5,R7 …………………(8)

| | **R1** | **R2** | **R3** | **R4** | **R5** | **R6** | **R7** |
|---|---|---|---|---|---|---|---|
| **R1** | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **R2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **R3** | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **R4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **R5** | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **R6** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **R7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 6. Dependency Structure Matrix (DSM) For 7 Business Rules**

Calculated cost for each instance while using hard coded business rules sequence is shown in Table 7.

| TID | R1 | R2 | R3 | R4 | R5 | R6 | R7 | Cost |
|---|---|---|---|---|---|---|---|---|
| **1** | S | S | S | F | U | U | U | 4 |
| **2** | S | S | S | S | S | F | U | 6 |
| **3** | S | S | S | F | U | U | U | 4 |
| **4** | S | S | S | S | S | F | U | 6 |
| **5** | U | U | S | U | S | S | F | 4 |
| **6** | S | S | S | F | U | U | U | 4 |
| **7** | S | S | S | F | U | U | U | 4 |
| **8** | S | S | S | S | S | F | U | 6 |

**F:Failed**
**U:UnExecuted**
**S:Success**

**Table 7. Cost of Hard coded Used Sequence**

$C_M$ = Cost of (T1 + T2 + T3 + T4 + T5 + T6 + T7 +T8)/N

= 4+6+4+6+4+4+4+6/8

= 38/8

= 4.8

We will perform simulation on same sample dataset by using the identified optimized business rules execution sequence in eq.8 for making cost comparison. We will calculate 'mean condition evaluation cost' $C_M$ for 'hard coded business rules execution sequence' and for the identified 'optimized business rules execution order' to compare the results later. Table.8 is the new representation of same sample dataset after changing the business rules order using the obtained 'optimized predicted business rules' sequence [R6,R4,R1,R2,R3,R5,R7].

| **TID** | **R6** | **R4** | **R1** | **R2** | **R3** | **R5** | **R7** |
|---|---|---|---|---|---|---|---|
| **1** | U | F | S | S | S | U | U |
| **2** | F | S | S | S | S | S | U |
| **3** | U | F | S | S | S | U | U |
| **4** | F | S | S | S | S | S | U |
| **5** | S | U | U | U | S | S | F |
| **6** | U | F | S | S | S | U | U |
| **7** | U | F | S | S | S | U | U |
| **8** | F | S | S | S | S | S | U |

**F:Failed**
**U:UnExecuted**
**S:Success**

**Table 8. Dataset Representation Using Predicted Business Rules Sequence**

Here, a challenging task is how to treat 'UnExecuted' business rules while calculating cost? Two possibilities can exist.

**i) Worst Case:** Treat 'UnExecuted' as 'Successful'

'UnExecuted' business rule coming before the 'Failed' ones will be supposed that they were successfully executed and are putting a cost on system, as the rule(s) next to this one will also be evaluated.

| **TID** | **R6** | **R4** | **R1** | **R2** | **R3** | **R5** | **R7** | **Cost** |
|---|---|---|---|---|---|---|---|---|
| **1** | S | F | S | S | S | U | U | 2 |
| **2** | F | S | S | S | S | S | U | 1 |
| **3** | S | F | S | S | S | U | U | 2 |
| **4** | F | S | S | S | S | S | U | 1 |
| **5** | S | S | S | S | S | S | F | 7 |
| **6** | S | F | S | S | S | U | U | 2 |
| **7** | S | F | S | S | S | U | U | 2 |
| **8** | F | S | S | S | S | S | U | 1 |

**F: Failed**
**U: UnExecuted**
**S: Success**

$C_M$ = Cost of (T1 + T2 + T3 + T4 + T5 + T6 + T7 +T8)/N

= 2+1+2+1+7+2+2+1/8

= 18/8

= 2.25

**ii) Best Case:** Treat 'UnExecuted' as 'Failed'

'UnExecuted' business rule coming before the 'Failed' ones will be supposed that they were 'failed' to execute thus reducing the cost on system as the rule(s) next to this one will not be evaluated more.

| **TID** | **R6** | **R4** | **R1** | **R2** | **R3** | **R5** | **R7** | **Cost** |
|---|---|---|---|---|---|---|---|---|
| **1** | F | F | S | S | S | U | U | 1 |
| **2** | F | S | S | S | S | S | U | 1 |
| **3** | F | F | S | S | S | U | U | 1 |
| **4** | F | S | S | S | S | S | U | 1 |
| **5** | S | F | F | F | F | F | F | 2 |
| **6** | F | F | S | S | S | U | U | 1 |
| **7** | F | F | S | S | S | U | U | 1 |
| **8** | F | S | S | S | S | S | U | 1 |

**F: Failed**
**U: UnExecuted**
**S: Success**

$C_M$ = Cost of (T1 + T2 + T3 + T4 + T5 + T6 + T7 +T8)/N

= 1+1+1+1+2+1+1+1/8

= 9/8

= 1.12

Results justifies that the *mean condition evaluation cost*($C_M$) using predicted sequence is lower than the hard coded used sequence and it can be considered better in terms of system resources consumption. After changing the rules execution order as per obtained optimized predicted sequence in eq.8, even in worst case it gives us the better performance results as compared to hard coded used sequence. In both cases, while treating 'UnExecuted' as 'Success' & 'Failed' gives us the lower $C_M$ .

## 4.5 Summary

In this chapter a set of sample data as an example has been presented to present the concept fully. The description of proposed methodology, followed by different applied techniques in order to find a possible optimized execution order of business rules have also been discussed. It has been found that the optimized business rules execution order has a lower cost as compared to the used ‘hard coded sequence’.

# CHAPTER 5

# RESULTS & EVALUATION

## 5.1 Results and Discussion

In this section, experimental results are presented.i7 Computer with 2.50 GHz 64bit Intel (R) Core (TM) processor and 8GB RAM was used for execution of classification and pattern mining techniques in order to find an optimized business rules execution sequence, if executed in that sequence can sufficiently able to reduce system resources from wastage. At the moment, present work focuses on to propose an idea to make softwares flexible instead of hard coding them.

Our Dataset contains logs that are generated by ‘E-Ticket reservation software’. This web portal is used in reservation offices and also accessible online by passengers to reserve tickets for travelling. It contains execution information of 105467 business instances of business process ‘Reserve seat’ having 25 business rules. Initial obtained logs are in relational form containing information of both 'Success' and 'Failed' business rules. We first prepared a feature file suitable for applying data mining techniques (preprocessing step). We mined the actual logs for the business process 'Reserve seat' denoted by 'B' containing 25 business rules 'R'. To successfully execute the business process ‘B’, all business rules/constraints 'R' must be satisfied. Logs contain information that which rules were executed successfully and which one failed that resultantly caused whole business process to fail. For simplicity, we represented each rule with shorter notation e.g. R1, R2, R3,..., R25 and are called features of business process 'B' as shown in Table 9. Initial hard coded business rules execution sequence ‘E1’ is following.
E1 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R19,
R20,R21,R22,R23,R24,R25

| S.No | Rule Name | S.No | Rule Name |
|---|---|---|---|
| **R1** | IsPassengerCNICValid | **R14** | IsSpouseNotInPermanentDisabledList |
| **R2** | IsPassengerCNICNotInFraudelentList | **R15** | IsSpouceCNICValid |
| **R3** | IsPassengerMobileNoValid | **R16** | IsJournalistCNICValid |
| **R4** | IsPassengerMobileNoNotFraudulentList | **R17** | IsPassBelongsToConcernedJournalist |
| **R5** | IsPassengerMobNoNotInPermanentBlockedList | **R18** | IsJournaistNotInPermanentDisabledList |
| **R6** | IsRailwayCardNoValid | **R19** | IsJournalistNotInFraudulentList |
| **R7** | IsEmployeeNotInFraudulentList | **R20** | IsJournalistActive |
| **R8** | IsEmployeeNotInBlockedList | **R21** | IsJournalistPassNoValid |
| **R9** | IsUserNotInTemporaryBlockedList | **R22** | IsPassBelongToConcernedEmployee |
| **R10** | IsUserValidToReserveSeatForPassenger | **R23** | IsPassAlreadyAvailed |
| **R11** | IsReservationNotStopped | **R24** | IsEmployeePassNoValid |
| **R12** | IsCoachNotDetached | **R25** | IsEmployeeOnService |
| **R13** | IsTrainAvailableForReservation | | |

**Table 9. Business Rules List**

As an optimization step, we first classified the data into ‘Success’ and ‘Failed’ instances and then applied pattern mining techniques only on relevant data (failed instances).*Class label* of each instance is a binary attribute (‘Success’ & ‘Failed’). We divide the data into training and testing data using 80 by 20 ratio. 80% of the data is used for training and 20% for modal testing. Confusion matrix for the classifiers K-NN, SVM Decision Tree and Naive Bayes is presented in Table 10, Table 11, Table 12 and Table 13 that shows the performance of applied classifiers.

| **N=21093** | **Success** | **Failed** |
|---|---|---|
| **Success** | TP(6213) | FN(2850) |
| **Failed** | FP(3611) | TN(8419) |

**Table 10. Confusion Matrix for K-NN with 3 Nearest Neighbours**

| **N=21093** | **Success** | **Failed** |
|---|---|---|
| **Success** | TP(6158) | FN(2905) |
| **Failed** | FP(3532) | TN(8498) |

**Table 11. Confusion Matrix for SVM**

| N=21093 | Success | Failed |
|---|---|---|
| **Success** | TP(6145) | FN(2918) |
| **Failed** | FP(3521) | TN(8509) |

**Table 12. Confusion Matrix for Decision Tree**

| N=21093 | Success | Failed |
|---|---|---|
| **Success** | TP(7307) | FN(1756) |
| **Failed** | FP(5033) | TN(6997) |

**Table 13. Confusion Matrix for Naive Bayes**

In order to find the classifier that performs well, Table 14 presents different performance measures e.g. Precision, Recall, ROC area etc. of each classifier that shows how much the classifier performed well on the given data.

| Classifier | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area |
|---|---|---|---|---|---|---|
| **SVM** | 0.695 | 0.309 | 0.698 | 0.695 | 0.696 | 0.693 |
| **Naive Bayes** | 0.678 | 0.290 | 0.710 | 0.678 | 0.677 | 0.714 |
| **K-NN** | 0.694 | 0.308 | 0.698 | 0.694 | 0.695 | 0.729 |
| **Decision Tree** | 0.695 | 0.309 | 0.698 | 0.695 | 0.696 | 0.711 |

**Table 14. Calculated Performance Measures on the Basis of Confusion Matrix**

We set a *minimum support* threshold as 15% in order to find frequent patterns B1,B2,B3,B4.

**Frequent Patterns:**

B1 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15, R16,R17,R18 (Support : 23%)

B2 =R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15, R16,R17,R18,R19,R20,R21 (Support:18%)

B3 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20,R21, R22 (Support : 22%)

B4 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20,R21, R22,R23 (Support:17%)

Having set *minimum confidence* threshold as 20%, following are identified association rules. 'n-1th' part of pattern are the rules that were successfully executed and 'nth' denotes failed rule that caused business process to fail. E.g. Association rule 'A4' denotes that there is 54% probability that when 'R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15, R16,R17,R18,R19,R20,R21,R22' will successfully execute, 'R23' will fail.

**Association Rules:**

A1 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15, R16,R17→R18 (Confidence: 24%)

A2 =R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15, R16,R17,R18,R19,R20 →R21 (Confidence: 25%)

A3 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20,R21 →R22 (Confidence: 41%)

A4 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20, R21,R22 →R23 (Confidence: 54%)

After arranging the frequent patterns in descending order w.r.t. to their support, 'nth' (Failed) rule of each frequent pattern will be moved at the top to prepare the new business rules execution sequence 'E2'. If it failed earlier, less will be the consumption of resources. Following optimized business rules execution order 'E2' is obtained based on identified frequent patterns.

E2 = R18,R22,R21,R23,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16, R17,R19, R20,R24,R25

We are more interested to identify the business rule(s) that are putting higher cost on system when they fail. Moving them first to execute will prevent system resources from wastage, so we have included the *weight/cost* factor in order to identify *weighted frequent patterns*. Optimized business rules execution order predicted via *weighted frequent pattern mining* will be used to make comparison with the one found using frequent pattern mining and the hard coded business rules execution sequence. Following identified patterns F1,F2,F2,F4 denotes the frequent patterns that are weighted frequent as well. Table 15 shows the weighted support of each pattern. In our case, identified pattern should be 'Frequent' and 'Weighted Frequent' as well. 'Frequent' measure denotes that a pattern has enough probability of occurrence and 'Weighted Frequent' measure fulfills the criteria that it must have enough cost, at-least the minimum set threshold. If a pattern has a very low cost that is negligible then it is better to exclude it.

**Weighted Frequent Patterns:**

F1 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20,R21 ,R22

F2 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18

F3 = R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17,R18,R19,R20, R21,R22, R23

F4 =R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15, R16,R17,R18,R19,R20,R21

| **Total Instances:89304** | **Weighted Support** |
|---|---|
| **F1 Itemset** | 0.17 = 17% |
| **F2 Itemset** | 0.15 = 15% |
| **F3 Itemset** | 0.19 = 19% |
| **F4 Itemset** | 0.16 = 16% |

**Table 15. Weighted Support of Each Itemset**

‘n-1th’ part of pattern are the rules that were successfully executed and ‘nth’ denotes the failed rule that caused business process to fail. We then arranged the identified weighted frequent patterns in descending order w.r.t. their *weighted support/(cost)* along with maintaining the needed dependency among them using DSM. ‘nth’ (Failed) rule in each identified weighted frequent pattern will be moved at the top to execute it first. If it failed earlier, less will be the consumption of resources. Following optimized business rules execution order is obtained that is sufficiently able to prevent system resources from wastage and having low consumption cost as well.

E3 = R23, R22,R21 ,R18,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,R12,R13,R14,R15,R16,R17, R19,R20,R24,R25

In Table.16, results clearly show that using the predicted sequence of business rules obtained through ‘weighted frequent pattern mining’ approach performs much better than the ‘hard coded’ used sequence and the one identified using ‘Frequent pattern mining’, as it has the lowest $C_M$, thus consuming less system resources in-case the business process is failed to complete. For such cases when larger the business processes will fail, lesser will be the resources consumption.

| | **Total Failed Instances of Business Process (‘Reserve Seat’)** | **Treat 'UnExecuted' as 'Success'** | **Treat 'UnExecuted' as 'Failed'** | **Condition Evaluation Cost($C_M$)** | **Resource Wastages Cost Prevention w.r.t ‘E1’** |
|---|---|---|---|---|---|
| **E1** | 89304 | NA | NA | 20.69 | |
| **E2** | 89304 | Yes | No | 6.29 | 69.6 % |
| **E3** | 89304 | No | Yes | 5.44 | 73.7 % |
| **E4** | 89304 | Yes | No | 6.27 | 69.7 % |
| **E5** | 89304 | No | Yes | 4.63 | 77.6 % |

**E1: Hard coded used sequence**
**E2: Predicted execution sequence using *Frequent pattern mining***
**E3: Predicted execution sequence using *Frequent pattern mining***
**E4: Predicted execution sequence using *Weighted frequent pattern mining***
**E5: Predicted execution sequence using *Weighted frequent pattern mining***

**Table 16. Cost Comparison for 89304 Instances**

## 5.2 Summary of Chapter

This chapter presents the results of the applied approaches used to classify the data and then to apply pattern mining techniques in order to find an optimized business rules execution order, if executed can enhance system performance and can also prevent system resources from wastage. Results prove that hard coded business rules execution can degrade system performance while wasting system resources.

# CHAPTER 6

# CONCLUSION & FUTURE RECOMMENDATIONS

## 6.1 Conclusion

This thesis addresses an important concept of predicting the optimized business rules execution order by applying data classification and pattern mining techniques on the generated logs. Weighted frequent pattern mining approach has been used in order to identify the high cost consumption frequent patterns that are causing business process to fail. Later on, it has been compared with the 'hard coded business rules execution sequence' and the one identified using 'frequent pattern mining'. Assigning default priorities to execute business rules may make system less efficient. Also, algorithms like RETE algorithm may not work well if there is limited sharing among rules' conditions. In the proposed work, a compressed data structure has been used that needs only one database scan for its construction unlike Apriori algorithm that requires multiple DB scans. As verified by the results, we mined 89304 no of failed business instances of a business process 'Reserve seat' and found that the cost of system resources consumption can be prevented from wastage by 69.6% (worst case) and 73.7% (best case) if only probability factor is taken. By including the cost/weight parameter along with support measure, the resources wastage can be lowered more to 69.7% (worst case) and 77.6% (best case). Telcos and IT industry / Software houses can take big advantage of our proposed model that will not only optimize their applications but will also enhance their businesses. Software engineers can focus fully in developing a quality software instead of worrying about to code an optimized business rules execution order. Research conducted and presented in this paper exhibits that optimization of business rules is appreciably beneficial in identification of bottlenecks, to improve productiveness of useful resource control and optimization of system resources to make it flexible and reliable.

## 6.2 Limitations

One of the limitations of our applied approach is that if business rules are highly depending on each other than applying the proposed mining process on logs will have no advantage. Instead it will be an extra overhead step conducted without obtaining any fruitful objective. Secondly, if the execution logs are containing a mix data of business rules belonging to multiple different business processes, our proposed model may not work well for such case. Third limitation in our proposed work is that we are not dealing with transitive dependency among business rules. If ‘A’ is dependent on ‘B’ and ‘B’ is dependent on ‘C’ then ‘C’ can’t be executed neither before ‘B’ nor ‘A’.

## 6.3 Future Work

The contribution can be extended to propose an efficient solution for predicting an optimized execution order for highly dependent business rules. Also, research can be extended to mine business rules data that are part of multiple business processes. Apart from that, a solution can be proposed to predict an optimized business rules execution order while dealing with transitive dependency in parallel.

# References

[1] Harmon, P., & Trends, B. P. (2010).Business process change: *A guide for business managers and BPM and Six Sigma professionals.* Elsevier.

[2] Singh, S., Holland, S. W., & Bandyopadhyay, P. (2010, March). Trends in the development of system-level fault dependency matrices. *in Aerospace Conference, 2010 IEEE*(pp. 1-9). IEEE.

[3] Van der Aalst, W. M., & Weijters, A. J. M. M. (2004). Process mining: a research agenda.

[4] John Lu, Z. Q. (2010). The elements of statistical learning: data mining, inference, and prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society), 173*(3), 693-694.

[5] Liu, D., Lin, Y., Huang, P. C., Zhu, X., & Liang, L. (2017). Durable and Energy Efficient In-Memory Frequent-Pattern Mining. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 36*(12), 2003-2016.

[6] Soni, H. K., Sharma, S., \& Jain, M. (2016, March). Frequent pattern generation algorithms for Association Rule Mining: Strength and challenges. In *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on*(pp. 3744-3747). IEEE.

[7] Bose, S., & Datta, S. (2015, February). Frequent pattern generation in association rule mining using weighted support. In *Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on*(pp. 1-5). IEEE.

[8] Pal, S., & Bagchi, A. (2005). Association against Dissociation: some pragmatic considerations for Frequent Itemset generation under Fixed and Variable Thresholds. *ACM SIGKDD Explorations Newsletter, 7*(2), 151-159.

[9] Liu, H.,& Wang, B. (2007). An association rule mining algorithm based on a Boolean matrix. *Data Science Journal, 6*, S559-S565.

[10] Rao, C. S., Babu, D. R., Shankar, R. S., Kumar, V. P., Rajanikanth, J., \& Sekhar, C. C. (2013). Mining association rules based on boolean algorithm-a study in large databases. *International Journal of Machine Learning and Computing, 3*(4), 347.

[11] Klangwisan, K., & Amphawan, K. (2017, February). Mining weighted-frequent-regular itemsets from transactional database. *In Knowledge and Smart Technology (KST), 2017 9th International Conference on* (pp. 66-71). IEEE.

[12] Lu, H. (2014). Recommendations based on purchase patterns. *International Journal of Machine Learning and Computing, 4*(6), 501.

[13] Ambily, M., Visakh, R. (2014). Survey on Weighted Frequent Pattern Mining. *International Journal of Computer Trends and Technology (IJCTT).*

[14] Lindemann. (2009). The Design Structure Matrix (DSM). Retrieved from http://www.dsmweb.org.

[15] 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).

[16] Bednar, P., Babič, F., Albert, F., Paralič, J., & Bartok, J. (2011, January). Design and implementation of local data mining model for short-term fog prediction at the airport. In *Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium on* (pp. 349-353). IEEE.

[17] Cai, C. H., Fu, A. W. C., Cheng, C. H., \& Kwong, W. W. (1998, July). Mining association rules with weighted items. In *Database Engineering and Applications Symposium, Proceeding IDEAS'98. International* (pp. 68-77). IEEE.

[18] Huai, Z. G., & Huang, M. H. (2011, May). A weighted frequent itemsets incremental updating algorithm base on hash table. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on* (pp. 201-204). IEEE.

[19] Sharma, S., Agrawal, J., Agarwal, S., & Sharma, S. (2013, December). Machine learning techniques for data mining: A survey. In *Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on* (pp. 1-6). IEEE.

[20] Sujatha, M., Prabhakar, S., & Devi, D. G. L. (2013). A Survey of Classification Techniques in Data Mining. *International Journal of Innovations in Engineering and Technology (IJIET),* 2(4).

[21] Liu, T., Tian, C., Zhang, H., \& Ding, W. (2009, July). Learning the priority for rule execution. In *Service Operations, Logistics and Informatics, 2009. SOLI'09. IEEE/INFORMS International Conference on* (pp. 565-572). IEEE.

[22] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

[23] Ragab, A. H. M., Noaman, A. Y., Al-Ghamdi, A. S., & Madbouly, A. I. (2014, June). A comparative analysis of classification algorithms for students college enrollment approval using data mining. In *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments* (p. 106). ACM.

[24] Forgy, C. L. (1988). Rete: A fast algorithm for the many pattern/many object pattern match problem. In *Readings in Artificial Intelligence and Databases* (pp. 547-559).

[25] Van der Aalst, W., Adriansyah, A., & van Dongen, B. (2012). Replaying history on process models for conformance checking and performance analysis.*Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2*(2), 182-192.

[26] Tiwari, A., Turner, C. J., & Majeed, B. (2008). A review of business process mining: state-of-the-art and future trends. *Business Process Management Journal, 14* (1), 5-22.

[27] Forgy, C. L. (1979). *On the efficient implementation of production systems* (Doctoral dissertation, Carnegie-Mellon University).

[28] Agrawal, R., Cochrane, R. J., & Lindsay, B. G. (1991). *On maintaining priorities in a production rule system*.

[29] Eric N. Hanson. Rule condition testing and action execution in ariel.In *SIGMOD Record*(Vol. 21, No. 2, pp. 49-58). ACM.

[30] Tong-yan, L., & Chao, C. (2011, September). Efficient mining of weighted frequent itemsets using MLWFI. In *Electronics, Communications and Control (ICECC), 2011 International Conference on* (pp. 849-852). IEEE.