Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

A Learning Classifier System for Detection of Service-Level Agreement Violations in Business Process

A Thesis

Submitted to the Council of the College of Information Technology for Postgraduate Studies of the University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology – Software

By

Hawraa Abdulameer Subeh Mohammed

Supervised by

Asst. Prof. Dr. Ahmed Khelfa Obeid Hamza

2021-2022 A.C. **1442-1443 A.H.**

"وَإِذْ قَالَ رَبُّكَ لِلْمَلَائِكَةِ إِنِّي جَاعِلٌ فِي الْأَرْضِ خَلِيفَةً ۖ قَالُوا أَتَجْعَلُ فِيهَا مَن يُفْسِدُ فِيهَا وَيَسْفِكُ الدِّمَاءَ وَنَحْنُ نُسَبِّحُ بِحَمْدِكَ وَنُقَدِّسُ لَكَ ۖ قَالَ إِنِّي أَعْلَمُ مَا لَا تَعْلَمُونَ* وَعَلَّمَ آدَمَ الْأَسْمَاءَ كُلَّهَا ثُمَّ عَرَضَهُمْ عَلَى الْمَلَائِكَةِ فَقَالَ أَنبِئُونِي بِأَسْمَاءِ هَٰؤُلَاءِ إِن كُنتُمْ صَادِقِينَ* قَالُوا سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا ۖ إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ"

صدق الله العَلِيُّ العَظِيمُ

سورة البقرة

Declaration

I hereby declare that this dissertation entitled “A Learning Classifier System for Detection of Service-Level Agreement Violations in Business Process” submitted to the University of Babylon in partial fulfillment of requirements for the degree of Master in Information Technology \ Software, has not been submitted as an exercise for a similar degree at any other University. I also certify that this work described here is entirely my own except for experts and summaries whose source is appropriately cited in the references.

Signature:

Name: Hawraa Abdulameer Subeh

Date: / / 2022

Supervisor Certification

I certify that the thesis entitled “A Learning Classifier System for Detection of Service-Level Agreement Violations in Business Process” was prepared under my supervision at the department of Software / College of Information Technology / the University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology - Software.

Signature:
Supervisor Name: Asst. Prof. Dr. Ahmed Khelfa Al-Ajeli
Date: / / 2022

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled “A Learning Classifier System for Detection of Service-Level Agreement Violations in Business Process” for debate by the examination committee.

Signature:
Asst. Prof. Dr. Ahmed Saleem Abbass
Head of Software Department
Date: / / 2022

Certification of the Examination Committee

We, the undersigned, certify that (Hawraa Abdulameer Subeh) candidate for the degree of Master in Information Technology - Software, has presented his thesis of the following title ("A Learning Classifier System for Detection of Service-Level Agreement Violations in Business Process") as it appears on the title page and front cover of the thesis that the said thesis is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on:

Signature:
Name: Dr. Ahmed Talib Abdulameer
Title: Assistant Professor
Date: 6 / 6 / 2022
(**Chairman**)

Signature:
Name: Dr. Rafid Sagban Abbood
Title: Assistant Professor
Date: 6 / 6 / 2022
(**Member**)

Signature:
Name: Dr. Wadhah R.Baiee
Title: Lecturer
Date: 6 / 6 / 2022
(**Member**)

Signature:
Name: Dr. Ahmed Khelfa Al-Ajeli
Title: Assistant Professor
Date: 6 / 6 / 2022
(**Supervisor**)

Signature:
Name: Hussein Atiya Lafta
Title: Professor
Date: 6 / 6 / 2022
(Dean)

Dedications

I dedicate this dissertation...

- To Almighty Allah, my strong pillar, my source of inspiration, wisdom, knowledge, and understanding. He has been the source of my strength throughout this time…

-Everything I am or ever will be I owe it, my warmest home, my paradise, my mother

-To the one who never stops giving of himself in countless ways, my first love and my only hero, my father

-To my beloved brother and his family who has been a constant source of support and encouragement during the all time

-To my life partner who leads me through the valley of darkness with the light of hope

-To all my loyal friends who stand by me when things look bleak and keep supporting me

- My sweet baby, Saden.

Acknowledgment

I would like to take this opportunity to acknowledge my academic supervisor who guided me in this process and the committee who kept me on track. All thanks, appreciation and respect to **Dr. Ahmed Khelfa Al-Ajeli**. Also, I would like to thank the teachers who helped and guided me to successfully complete my study career and they all have the credit to make me stand here today.

Last but not least, I would like to thank those who helped me with a word of encouragement, a smile on a dark day, or scientific information that benefited me in this research, thank you all.

Abstract

A Service-Level Agreement (SLA) is a contract made between service providers and customers. This contract includes some constraints whose aim is to maintain a certain level of quality of service. A violation of such SLA constraints can be seen as a fault which means the failure of service guarantee. This could lead to undesirable results such as paying fines, reducing the profit percentage, reputation defalcation, and service outage. This thesis proposes a machine learning technique based on Learning Classifier System (LCS) to detect such violations. Since events in the system being analyzed are partially observable, this raises the most interesting case to address this problem. The Anticipatory Classifier System (ACS) model – a form of the LCS to deal with the case of partial observation is adopted here. The ACS is used to produce a rule-based model which is then used to detect a violation of SLA when observing a sequence of events. A typical example in a telecommunication company is used as a case study to generate the dataset utilized to train the model. The dataset generated is imbalanced, and an oversampling technique is applied to overcome the problem of bias to the dominated class. As a post-processing stage, the rule compaction technique is applied to obtain a compact ruleset which is then used in the online phase as a detector. The experimental results show a promising performance of the proposed approach in achieving a high accuracy outcome of 99%. In addition, the resulting model can give a timely decision whether a violation has occurred or not, and this is beneficial for a service provider to avoid paying fines when a violation occurs.

Declaration Associated with this Thesis

Some of the works presented in this thesis have been accepted as below.

Title: A Learning Classifier System for Detection of Service-Level Agreement Violations in Business Process

Author: Ahmed Al-Ajeli and Hawraa Abdulameer Subeh,

Journal: The 2nd International Conference on Information Technology to Enhance E-Learning and Other Applications - [2nd IT-ELA 2021]

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Abbreviation	Description
ACS	Anticipatory Classifier System
ALP	Anticipatory Learning Process
DES	Discrete Event System
FP	False Positive
FN	False Negative
GA	Genetic Algorithm
IFME	Integer Fourier Motzkin Elimination
LCS	Learning Classifier System
ML	Machine Learning
PDRC	Parameter Driven Rule Compaction
QoS	Quality of Service
RL	Reinforcement Learning
SLA	Service Level Agreement
SLS	Service Level Specification
SMOTH	Synthetic Minority Oversampling TEchnique
FP	False Positive
FN	False Negative

Chapter One

Introduction

Chapter One Introduction

1.1 Introduction

A business process is a collection of tasks and activities by which a specific sequence of services is provided [1]. The business process plays an important role in any business nowadays and occurs in all levels of organizations. Using the business process comes with the benefit of improving customer satisfaction and increasing the profit of organizations. Besides, there is no doubt that technology is the driver of the success of modern businesses. Technology affects the efficiency and relationships of a business. It also affects the security of clandestine information and business advantages. One of the most important aspects of any business process is the Service-Level Agreement (SLA), which is a contract between service providers and customers [2]. The SLA guarantees the quality of the service to be provided within an acceptable level. Also, the SLA can be considered as a good criterion to distinguish various levels of service guarantee and establish a comparison to other service providers. Furthermore, it helps to remain focused on customer requirements and ensure that the internal processes take the right direction [3].

Failure to award the agreed service can be viewed as an SLA violation for which a detection method should be created. The SLA violation in this sense does not represent any activity (event) in the system being analyzed, instead, it is a violation of a constraint defined in the SLA contract. Since not all activities are visible for the external observer, two different types of events: observable and unobservable (unseen events) arise.

Detecting a violation under the partial observation raises a more interesting case that might leads to uncertain states as explained later. Within this state, two sequences of activities can be seen the same for two different detection states, i.e., one sequence causes a violation while the other does not [4].

The previous work has addressed the problem of detecting an SLA violation in two different ways. The first one uses model-based systems while the second one uses data-based systems. Bordbar et al. [5] proposed the integer Fourier Motzkin Elimination (IFME) method to diagnose the violation of constraint in Discrete-Event Systems (DES) modeled by Petri nets (similar to automata models). Al-Ajeli and Parker [6] have employed an algebraic approach to diagnosing the violation of the constraint to the more general cases of Petri nets, in addition to dealing with a variety of fault types. Due to the time and space complexity required by the previous methods, researchers resorted to using machine learning techniques. As the work that applies artificial intelligence to solve the SLA violations increases, many papers have proposed methods to use machine learning techniques to detect such violations, especially those related to the business process. Agarwal [7] have predicted SLA violations in web services using regression machine learning techniques. An investigation for the detection of SLA violations in the Cloud Computing field has been presented by Zeng et al. [8]. While Hemmat and Hafid [9] applied two different machine learning techniques, namely naive Bayes and random forest to detect SLA violations. A gradient backpropagation method based on supervised learning to predict SLA violations has been used by Upadhyay et al. [10]. An ontology-based system is a method that was adopted to detect SLA violations by Jules et al. [11], Karamanlioglu and Alpaslan [12].

In this work, an approach that attempts to address the problem of detecting violations of SLA constraints under partial observation is presented. In this approach, a Learning Classifier System (LCS) is adapted, in particular Anticipatory

Learning Classifier System (ACS2), as a training model. This type of LCS can estimate the next state in the system being monitored. Starting from a dataset that includes simulation data generated from a model which captures the behaviour of a business process in a telecommunication company, an optimized rule-based model is created. The created model is used online to detect whether a violation has occurred or not after observing a sequence of events.

1.2 Related Work

In recent years, many models have been proposed to tackle the problem of detecting SLA violations to help service providers in reducing or minimizing SLA violations. Some of them work on the detection of SLA violations, while the others attempt to predict violations. In this section, the most relevant one will be review. The authors in [5] and [6] mainly base their work on model-based systems, whereas the remaining authors rely on the machine learning techniques to build their models by either using regression methods or classification method.

Al-Ajeli and Bordbar [5], employ the integer Fourier Motzkin Elimination Method (IFME) method to diagnose the violation of constraint in Discrete Event System (DES), by firstly creating the state equations of Petri nets. The state equations are indeed a series of inequalities with integer variables that denote the number of transitions that have been fired. The sets of inequalities derived from the state equations are then extended to two additional sets. The first is formed by expanding the failure inequality. The second is made by expanding the inequality for failure by its negation. The relevant factors related to unobserved events will be removed using the IFME approach on the two sets of inequalities that arise. Then, they used the resulting set of inequalities as a diagnoser.

Al-Ajeli and Parker [6] have employed an algebraic approach to overcome the limitations of the prior methods resulted in [5], and extended the work to the case where cycling may exist in Petri nets, but only in the form of observed transitions. Many real-world applications need adding cycles to Petri nets to mimic recurring behaviors of the system being analyzed. In addition, they also take into account the case of various fault forms. They are based mainly on the track of diagnosing histories. The set of inequalities that are resulted is utilized to make a diagnoser. They have shown that their method is an accurate for both finite and infinite systems.

Machine Learning techniques have been adopted by Agarwal [7] to predict the SLA violations and prevent them by optimizing the service into a unified framework. It depends on the incoming response time for SLA violation detection and prevention efficiency. To reach the needed reaction time and to prevent a violation, eight forms of a regression model are utilized, and a series of experiments are conducted to show that the suggested machine learning model can accomplish a robust performance for detecting and preventing the violation. The performance was good, but the model was difficult to update when receiving more training data.

Zeng et al. [8] have proposed work that detects SLA violations for cloud-hosted analytics applications of big data (BDAA). They applied four different machine learning techniques for detection (Random forests, Artificial Neural Network, Extreme Gradient Boosting, and Logistics Regression), and apply twelve diverse resampling techniques on the ML techniques to deal with data skewness problem. They tested the proposed techniques on a real dataset that is newly released by Alibaba. The proposed work not only assists providers in selecting the best effective technique for the detection, but also allows them to discover the hidden pattern of numerous BDAA settings across layers.

Hemmat and Hafid [9] have proposed two different machine learning techniques namely naive Bayes and random forest to predict SLA violation in cloud computing. Trace of Google cloud computing is used as a dataset for training the proposed model. They used several methods to handle the skewness of data. The extracted features to work with are only the task that is dislodged and re-schedule again. The proposed model will detect this task as a case of violation. As a result, the performance of the model with the random forest is better rather than naïve Bayes.

Upadhyay et al. [10] have used a scaled conjugate gradient backpropagation method based on supervised learning to predict SLA violations for cloud services. They prefer SCGB on normal backpropagation because it takes less time by doing a linear search so it can find steep descent direction in the first iteration. The data set used to train and test the model include two important feature: response time and throughput. If the response time was more and throughput was less than a specified threshold, the SLA violation has occurred.

Detecting SLA violations have also been proposed in the cloud computing domain. Jules et al. [11] have proposed a model that allows customers to select a trustworthy service provider according to their reputation by using a Bayesian network and suggest a smart SLA scheme that employs a probabilistic ontology to detect violations and notifying the service provider as soon as it occurs. The authors implemented their model using data taken from Amazon (EC2) and SLAs for Google Compute Engine. The results of the trials reveal that the probabilistic ontology is effective in predicting the SLA violation across a range of SLA parameters in a cloud context.

An expert system was proposed by Karamanlioglu and Alpaslan [12] to detect SLA violations in different fields with the use of the ontology-based approach. They

store SLA values in a triple such as RDF (Resource Description Framework) triples to represent the constraints. They recorded the SLA values by including all data about metrics, SLS (Service Level Specification), and SLA-related data. The proposed model converts the requested service into queries to infer if there was a violation. This model has been tested passing 1036 triple data from the SLA that belongs to the telecommunication. They measured the accuracy of the system by considering different situations with different inputs. Furthermore, they produce Synthetic data, which consists of 10000 triple data to test the model in a larger dataset.

Schubert et al. [13] have proposed a hybrid strategy that combines Aspect-Oriented Programming (AOP) with dedicated measurement software (agents) to detect and arbitrate SLA violations in the cloud. The AOP to monitor application-level metrics by detecting service requests and responses within the application. While an agent is used by the service provider for monitoring system-level metrics. This work has been tested by requesting the image manipulation service that is hosted on the amazon web service (AWS) and use different sizes of images at different times to detect the violation and the success ability converges to around 97%, which corresponds to a 3% chance of failure.

As an outline of the limitations in the previous work, it has been noticed that all of them did not deal with the partially observable environment that is intended to be explored in the present work. Table (1.1) shows the outline of the related work:

Table (1.1) Outline of the Related Work

Reference	Method	Dataset	Application Domain	Results
[5] and [6]	Model-based Systems in particular Integer	A Petri net of the simplified business	Business Process	the time and space complexity

	Fourier Motzkin Elimination(IFME)	process used within a typical telecommunication company, where events in the system are partially observed		required by this method is very high.
[7]	different regression models like Random Forest, Kernel Ridge, Gradient Boosting, and Time Series are applied	real-world data, which are fully observed	Web Services	The performance was good, but the model was difficult to update when receiving more training data
[8]	Machine learning techniques (Random forests, Artificial Neural Network, Extreme Gradient Boosting, and Logistics Regression)	Fully observed real dataset that is released by Alibaba	cloud-hosted analytics applications of big data	Unusual patterns dose not identified.
[9]	machine learning techniques naive Bayes and random forest	Fully observed real data by Google Cloud Cluster trace	Cloud Computing	As a result, the performance of the model with the random forest is better rather than naïve Bayes, but it is not trivial to update the knowledge representation of the model based on the new Coming examples.

[10]	scaled conjugate gradient backpropagation	The dataset having real-world measurement of QoS on different time slots from 339 users and 5,825 web services known as WSDREAM	Cloud Computing	Take less time compared to normal backpropagation
[11]	Bayesian network with the use of probabilistic ontology	A network with specific parameters is built to format the dataset using random process in Matlab	Cloud Computing	Dose not implemented in a real environment
[12]	ontology-based approach	the SLA data of the telecommunication domain is used	Several domains were investigated	Synthetic data used to measure the success of the system
[13]	Aspect-Oriented Programming (AOP) with dedicated measurement software (agents)	amazon web service (AWS) are used to test the system	Cloud Computing	Only the centralized third part can work with

1.3 Motivation

The motivation of this work is a running example coming from a telecommunication company [4] [5]. In this example, a business process is modeled using a modeling language called Petri nets which are similar to automata, and all

events in the system are partially observed. Detecting SLA violations in such systems using model-based system methods requires a long time according to the large complex system, this was the motivation to solve the problem using machine learning techniques.

1.4 Problem Statement

In general, detecting SLA violations is a very important issue in any business process whether the SLA is concluded between two or more organizations or different teams within the same organization. As business volume grows in industries where outsourcing is common, SLAs have become commonplace agreements to promote accountability and quality of the service provided, therefore detection of SLA violations has become an urgent issue that the organization must take into account. In addition, detection of the SLA violations in a partial observed environment is considered a big challenge for most researchers that what the previous research did not deal with. It will be impossible for the agent to have a full and comprehensive sense of the status of the environment in many real-world situations. Nevertheless, complete transparency is required for learning. A robot, for example, might observe whether it is in a corridor, an open room, or another location, and those observations could be inaccurate. This issue is also known as the "incomplete perception" problem.

1.5 Aim and Objectives of Thesis

This thesis aims to address the problem of detecting SLA violations in business processes under partial observations. This needs to achieve three main objectives:

1. Generate the dataset by simulating the Petri net model which captures the behavior of the system being analyzed.
2. Proposing a data-driven method that applies the anticipatory classifier system to deal with partially observed systems. This method generates an optimal rule-based system to make the detection decisions.
3. Evaluating the performance of the proposed method using a case study that represents a typical example in a telecommunication company.
4. Implementing the proposed method online to detect whether a violation has occurred or not for a sequence.

1.6 Research Contribution

The main contributions of this thesis are:

1. Proposing an Anticipatory Classifier System (ACS) – a form of the LCS to deal with the case of partial observation. This system generates an optimized set of rules which provide high accuracy to detect SLA violations.
2. Obtaining a dataset which is a partially observable by a simulation process.
3. Obtaining an efficient model with a compact set of rules results in a low time required to compute the detection of SLA violations online.
4. Proving the ability of the proposed method to solve a real problem within a telecommunication company.

1.7 Thesis Layout

The remainder of the thesis can be organized as follows :

1. Chapter Two gives an extensive description of the main concepts of business process, SLA, data preprocessing, machine-learning algorithms, post-processing, and methods for evaluating the data-driven model.
2. Chapter Three shows the proposed methodology. It explains dataset generation, feature projection, and sampling. Follows applying Anticipatory Learning Classifier System (ACS2), and how the model is utilized online.
3. Chapter Four presents and discusses the result obtained from implementing the proposed model.
4. Chapter Five draws conclusions that have been reached through the thesis and gives suggestions for future work.

Chapter Two

Theoretical Background

2.1 Overview

In this chapter, the theoretical background of business processes, Service-Level Agreements (SLA), and machine learning are discussed. The first section includes a general introduction to the business process. The Knowledge of the SLA is covered in section (2.3). The SLA Management Life Cycle, the importance of SLA in the modern market, and using the SLA for QoS are discussed. The remaining section discusses data preprocessing, machine learning, which includes machine learning fundamentals, Learning Classifier System (LCS), Post-processing, rule compaction, prediction, and evaluation.

2.2 Business Process

A business process is a sequence of actions or logically related tasks that must be completed in order to provide a value to customers or achieve another strategic objective [14]. This process aims to create a desired result by combining a set of actions within an enterprise. Successful systems begin with an understanding of business processes within an organization [15].

The business world is complicated, almost everywhere, firms undergo large and quick changes as a result of factors such as changing customer expectations, new technology, and increasing global rivalry. Consequently, many organizational business processes are dynamic and ever-changing. Practitioners are compelled to constantly alter their business procedures to stay survive in such situations due to swift changes. Some approaches and technologies are available to assist companies in improving their processes [16].

The efficient fulfillment of customer requirements is a critical component of total business performance, regardless of the clearly stated objectives. Therefore, it is critical to have a well-designed business process. The overall goal of a profit-maximizing corporation is usually to maximize long-term earnings. In simple terms, this is accomplished by continually maximizing revenues while lowering expenditures over time, which entails efficiently meeting client needs [17].

2.3 Service Level Agreement (SLA)

Service-Level Agreement (SLA) is a formal contract between the service provider and customer, where all the details of its use are described, as well as the measure of responsibility of both parties. The main thing to note is that this agreement is not about the service, but about the quality of its provision, and it represents a part of the IT service management mechanism [18].

The SLA was created to keep the IT process running. Not every company is capable of doing this with its internal resources. Certainly, there is another option - to attract a third-party company that specializes in IT outsourcing, but this also requires money. The SLA in this case is the lifeline, for both the client and the service provider, and to bring the process of maintaining a high quality of service provided to another level [19]. According to different users, there are three types of service agreements:

1. Service-Level Agreement (SLA) - is a contract between IT service units and IT users, such as business units. The service level agreement should be written in business terms. Communication between the IT and business units is essential. It is also a good idea to value the quality of service and change the contract [19].

2. Operation Level Agreement (OLA) - is a contract between IT services terms in an IT service unit. Following the signing of the service level agreement, the CIO (the chief information officer) should dismantle the service work and assign it to IT projects based on IT technology. For example, if a user expects the communication service to not go down three times a year, the CIO should break down the task into worker terms that support the network server and communication units such as routers and switches. In the context of IT services, the operation level agreement can serve as a basis for valuing employees [19].
3. Underpinning Contract (UC) - Agreement between the operator and a third party. It holds when the service provider needs additional services. Most often, such an agreement is conducted for the transfer of some functions that do not require constant maintenance, for example, maintenance of air conditioning systems, UPS, fire extinguishing or checking fire alarms, etc. [19].

The SLA is a product of long negotiations between the parties, and, as it happens with any contract or agreement, there may be shortcomings and mistakes, this, in turn, has the potential to ruin everyone life. A clear definition of the area of responsibility of the parties will help to avoid mistakes in inaccurate wording, as a result of which the responsibility can be shifted from the service provider to the customer [20].

2.3.1 The SLA Management Life Cycle

Each SLA passes through several phases starting from terms identification and conditions, monitoring, activation of the stated terms, and when hosting ends,

the contract expires, because it is one of the basic conditions in the contract. Such a steps sequence is called the SLA life cycle as shown in Figure (2.1) [21].

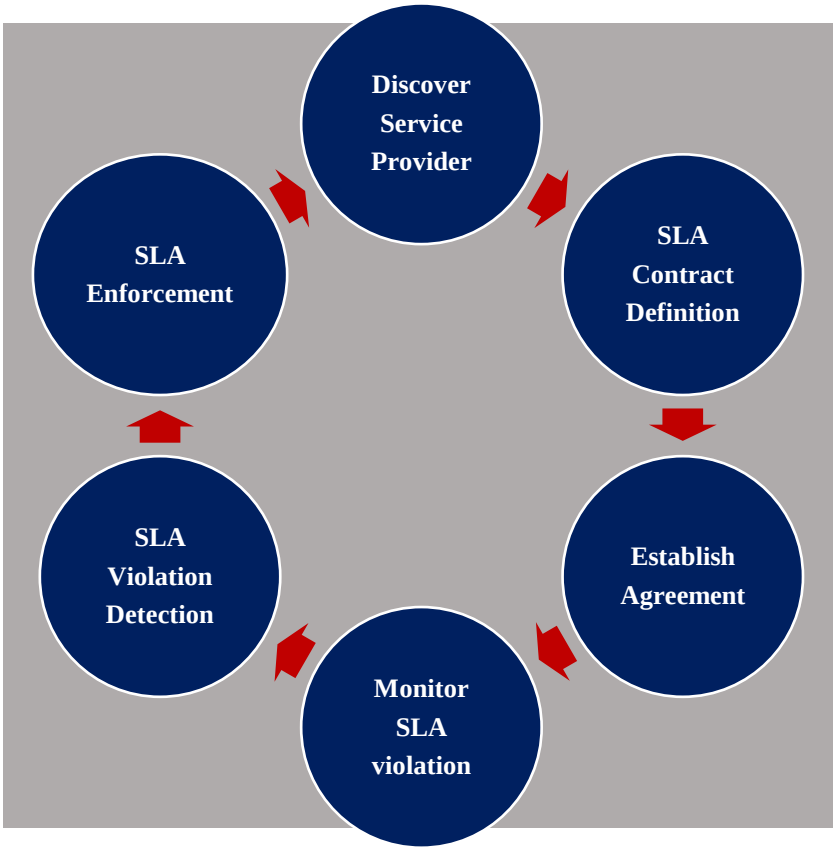

Figure (2.1): SLA Management Life Cycle

- **Discover Service Provider**

The first stage represents active service contributions that are announced by the service provider through standard publishing media. Forcing customers to adopt the search in a catalog to locate a service provider. The freedom of choice is left to clients to define competitive offers that meet their previously identified requirements for further negotiation [21].

- **The SLA Contract Definition**

During this stage, the QoS parameters with the penalty rule and the basic scheme are determined. Service level agreements are commonly defined by basic/standard formats, or by using standard layouts [21].

- **Establish Agreement**

In this part, the service provider works to meet the needs of the customer. The terms, conditions, and settlement of the service level agreement are negotiated. The service provider must assess the SLA to see service performance, availability and scalability before agreeing to the terms of the SLA to avoid fines. Once this stage ends, the contracting parties to the agreement are obligated to implement it [21].

- **Monitor The SLA Violation**

In this stage, the provided service has been estimated according to the agreement conducted between the two parties. This stage is an essential part of portraying the violations by monitoring the SLA. This stage helps the service provider identify resources before violations occur [21].

- **Detection of the SLA Violations**

Deviations are determined based on the SLA key factors [21]. The related punishment is directly executed when the violation is detected.

- **The SLA Enforcement**

In the last stage, the penalties imposed according to the violation of the SLA will be implemented. After identifying the violation as discussed in the previous

stage, appropriate measures are taken for that violation. Penalty charges are brought to the parties concerned after being informed, and the agreement will be terminated due to the violation [21].

2.3.2 The Importance of the SLA in the Modern Market

In order to choose a service provider, the customer will be interested in three issues: availability, performance, and quality of operation of the application providing the service. At the same time, the user expects that the operator will ensure not only a smooth operation of services but also the rapid introduction of new services. It also assumes that the service will function with adequate speed and reliability. To ensure that these requirements are met, the SLA can be used by the provider, which presents a tool for identifying, defining, monitoring, and managing the services that are delivered through the providers infrastructure [22].

The SLA can vary from one provider to another and is usually concerned together with the availability of the services and the reliability of data transmission. Usually, the SLA violations caused by the service provider are compensated to users when billing in the subsequent period of using the service [22].

Since the provision of high-quality services can be a decisive factor for a provider in attracting and retaining beneficial users, in today highly competitive market, the SLA is an important tool in providing users with the desired QoS. To meet the ever-growing demands of users for high-level services and expand the scope of their business, providers are forced to provide the SLA for their customers. In doing so, they should be able to provide users with reports that would prove that the required level of service is being maintained. The SLA allows you to convince the user of the providers ability to maintain high quality while providing expensive

services (and the need to pay for them). In this regard, the provision of the SLA is not a luxury, but a necessary means of survival for the provider in the service market [23].

2.3.3 The SLA and QoS

When defining the SLA, the QoS indicators are established and the methods that will be used to measure these indicators are determined. Typically, these indicators are availability, latency, throughput, end-to-end latency, mean recovery time, and load time. It should be noted that since QoS means something different for each user, the provider must provide different SLAs for different users based on different levels of performance, availability, and cost [23].

As the service is running, selected indicators are measured by proactive network testing or by collecting information about the performance of various network elements. Exceeding these indicators thresholds means that the conditions specified in the SLA are no longer met and the QoS has decreased to an unacceptable level. Continuous monitoring of the QoS indicators allows the provider to identify and correct degradation of services and traffic before the SLA violations lead to financial losses [23].

It is important to remember that the QoS parameters values used in the SLA must be real. That means, what the provider can guarantee. Otherwise, both the provider and the customer will constantly face a situation of SLA violation. If the provider cannot guarantee very strict QoS parameter values, then this should be taken into account when drawing up the SLA [23].

2.4 Petri Nets

Petri nets are a form of a systematic mathematical modeling framework that is used to model a variety of systems. Carl Adam Petri developed them as part of his PhD thesis. Similar to automata in formal languages, Petri nets capture the behavior of the system being modeled through the description of states and their transition. This description is used to study an information process that are infinite, asynchronous, concurrent, parallel, dynamic, and/or nondeterministic. Many examples to use Petri nets in modeling include computer science, communication, astronomy, nuclear physics, biology, in addition to their use in modeling business processes as discussed in this work. Tokens are also used to model system processes that are both dynamic and simultaneous in these nets. This mathematical method can be used to build state equations and other mathematical models that guide the behavior of systems. In addition, it is a tool for simulating business processes [24].

A Petri net [4] [24] is represented as a tuple N= (P, T, pre, post), in which $P=p_1 \ldots\ p_m$ and $T=t_1 \ldots\ t_n$ are finite sets that are not empty of positions and transitions. An input and output position for a transition t ∈ T, is a place p in which pre (p, t) (post (p, t)) is positive. The incidence matrix A= $[a_{ij}]$ is an integer array of size m×n, in which a_{ij} = post (p, t) − pre (p, t), presuming that the set of positions and transitions are ordered to coincide with the coordinates of the matrix. A marking (M) is a state of a Petri net that holds the number of tokens at each position (place). Usually, a marking is referred to as m×1 non-negative integer array. At a marking M, a transition t is enabled if M (p) ≥ pre (p,t). A transition that is enabled can fire, resulting in a new marking M `. The vector that fires **v** is referred to as an n×1 column vector of the form **v**=(0,...,0,0,...,1), The only one in the $j^{th}$ place denotes that the $j^{th}$ transition is now occurring. It is feasible to find the marking M ` that is reached by

M`=M +A**v** by using a firing transition for v on marking M. Initial marking M_0 refers to the status of the system when it is first turned on.

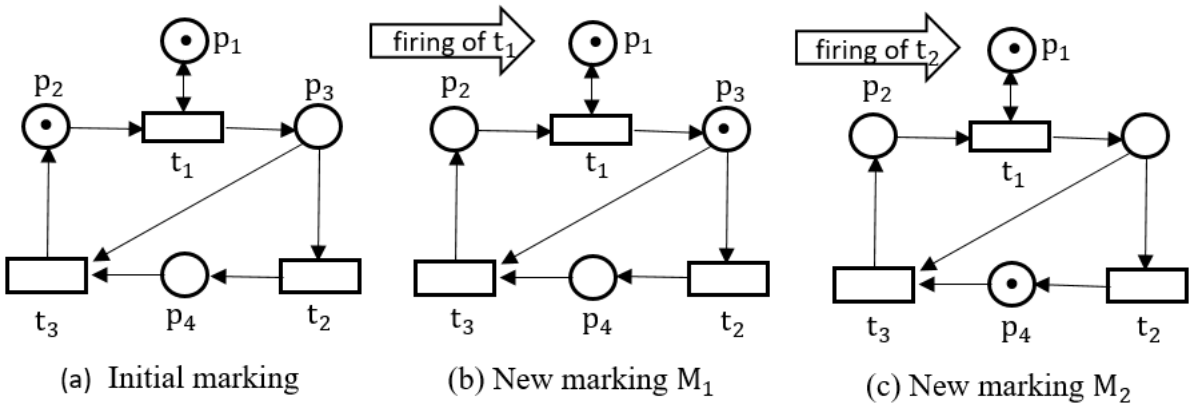

Figure (2.2): An example of Petri nets

Figure (2.2) shows an example of Petri nets, where the initial marking M_0= [1, 1, 0, 0]. The set of places P= $\{p_1, p_2, p_3, p_4\}$ and the set of transitions T= { $t_1, t_2, t_3\}$. Assume that sequence $t_1 t_2$ fires, then the marking resulting from this case can be obtained as follows

Pre=

	t_1	t_2	t_3
p_1	1	0	0
p_2	1	0	0
p_3	0	1	1
p_4	0	0	1

Post=

	t_1	t_2	t_3
p_1	1	0	0
p_2	0	0	1
p_3	1	0	0
p_4	0	1	0

A=Post-Pre

	t_1	t_2	t_3
p_1	0	0	0
p_2	-1	0	1
p_3	1	-1	-1
p_4	0	1	-1

M`=M +A**v**

$M_{2=} M_0 + A\mathbf{v}$

v= [1, 1, 0], where v represents the number of occurrences of events in the sequence.

$$M_{2=}[1, 1, 0, 0] + \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 1 & -1 & -1 \\ 0 & 1 & -1 \end{bmatrix} * [1, 1, 0]$$

$M_{2=}$ [1, 1, 0, 0] + [0, -1, 0, 1]

$M_{2=}$ [1, 0, 0, 1]

2.4.1 Running Example

In this section, a running example that has motivated this work is described. This example emerges from a telecommunication company where a simplified scenario for resolving broadband connection problems that are reported by the customers is described [4] [25]. A task is refer to an issue as it occurs, such as a sluggish broadband link. When the assignments arrive (t_1), each task is assigned to three teams, based on the type of issue being mentioned (see Figure 2.3). There are a few vast and complicated workflows within each department. Inside their organization, then the task is solved (transitions labeled "R_1, R_2, R_3") or the assigned task cannot be solved (transitions labeled "N_1, N_2, N_3"). If the work is settled within each department the job is done until t_{12}, t_{13}, or t_{14} occur, resulting in the occurrence of t_{15}, which marks the completion of the (overall) task. If a department is unable to finish the works (t_6, t_8, or t_{10}), go on to the next phase. As a consequence, a token is inserted in p_1 to allow the customer support department to reallocate the work. Transitions t_1 and t_{15}, which represent the arrival and completion of the task, are presumed to be visible. Transitions indicating the arrival of a task in each department (t_3, t_4, and t_5) can also be seen. Strong rectangles reflect observable transformations in Figure (2.3), the empty rectangles reflect unobservable transitions.

In the case of repetition of occurring t_6, t_8, t_8, we say that a right first time (RFT) fault has occurred. The occurrence of RFT faults may result in dissatisfaction with customers. Therefore, working on detecting RFT faults is essential. To ensure that no RFT fault has occurred, the following constraint must be held:

$$x_2 - x_{15} \leq b \tag{2.1}$$

Where x_2 refers to the number of allocations of a task in the model, x_{15} refers to the number of completions of a task and b is a threshold. If the above inequality is true, then there is no violation, otherwise, an SLA violation has occurred.

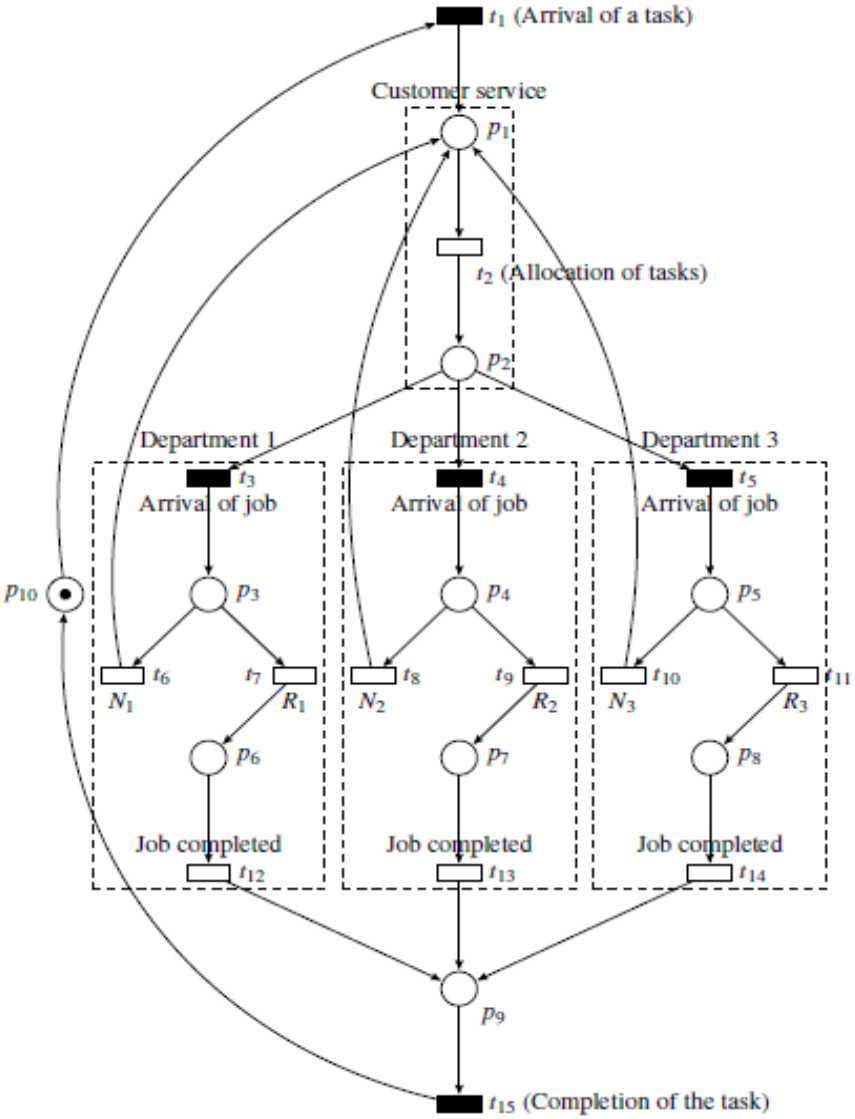

Figure (2.3): A problem of a resolving system

2.5 Data Preprocessing

Data preprocessing methods are essential to prepare the dataset. All of these methods, in general, fall into one of two categories: selecting data items and attributes for analysis or creating/changing attributes. Several ways for dealing with dataset difficulties such as noise, missing values, and inconsistent data are included in this method. In general, the goals of the preprocessing are [26]:

- Reducing the dataset size to improve analysis efficiency in terms of time, cost, and quality.
- Adapting the dataset to fit the chosen analytic approach.

2.5.1 Data Sampling

When faced with unbalanced sequence data sets, traditional classification methods typically struggle. They may assume that the distributions of the classes are balanced. In imbalance situations, however, the majority class profits more than the minority. The minority class is overwhelmed, and the classifier boundary is pushed to the minority region. As a result, the false-negative rate is high, and the recall rate is low [27].

Data sampling is the practice of picking subsets of data instances from a larger data set for analysis and reporting. It is a crucial stage in machine learning to ensure that learning systems can effectively generalize new unseen observations [28]. Data sampling alters the distribution of training data by either over-sampling or under-sampling techniques. Over-sampling techniques add instances of the minority class to the data set. Whereas, under-sampling eliminates instances from the majority classes. Random under-sampling and random over-sampling are the most basic sampling techniques. Random under-sampling (RUS) picks (at random) examples

from the majority class to be removed. Random oversampling (ROS) duplicates minority class examples at random. Under-sampling can result in information loss (due to example deletion), whereas random oversampling can result in overfitting [29].

2.6 Machine Learning Techniques

Machine learning is a subfield of artificial intelligence in which a computer analyzes data to predict the next task it will perform. The computer can access data in the form of digitized training sets or by contact with the environment. Machine learning algorithms, unlike static programming methods that require explicit human instruction, are built in such a way that they can learn and generate predictions from data [30]. Almost every scientific field has been covered by ML due to its use in a wide range of applications, resulting in a significant impact on research and society [31].

In general, there are three types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. In a nutshell, supervised learning requires the use of labeled data with inputs and outputs. Unsupervised learning, in a contrast to supervised learning, does not require labeled training data, and the environment only gives inputs without the intended targets. Reinforcement learning allows to learn from the feedback obtained from interactions with the outside world. Data analysis is the emphasis of both supervised and unsupervised learning, while reinforcement learning is chosen for decision-making tasks. Supervised learning deals with classification, regression, and estimation data processing tasks. While unsupervised learning deals with clustering, and prediction data processing tasks [31].

2.7 Learning Classifier System

Learning Classifier System (LCS) is a rule-based system that refers to a group of machine learning methods that are based on a concept developed to represent complicated adaptive systems such as the human brain, weather, or economy. The information is stored in an evolving population P of classifiers. Each classifier is made up of an "IF-THEN" rule that describes the environmental state in which it can be used, as well as some additional metrics that describe its usefulness. All rules are constantly re-evaluated during the interaction with the environment, and their metrics are adjusted accordingly. The rules that are more accurate are preserved in the population, while others are rejected. Then, the required result of working the LCS algorithm makes a decision based on these classifiers.

To obtain that result, the LCS algorithms employ two approaches; learning and evolution where learning leads the evolutional component to obtain an optimal set of rules. These notions are respectively expressed by mechanisms of two: a learning mechanism, and the genetic algorithm compatible with the given problem. Both mechanisms depend on the entire environment. Within the LCS context, the environment is simply the source of input data to the LCS algorithm. This data is used for training where one instance is passed to the LCS at a time. Then, feedback in the shape of a numeral reward is received by the LCS from the environment to guide the learning process [32].

2.7.1 Partially Observable Learning in LCS

The LCS is similarly well adapted to multi-step where feedback is delayed for many (potentially varied) stages. Since few other methods are suitable for both domains, there has been no study contrasting single-step and multi-step learning.

This may be due to the longer training periods expected and data mining relative lack of popularity in multi-step tasks [32].

Non-Markovian and Markovian multi-step domains may be further distinguished. The same state in Markov problems often maps to the same action (i.e. the present situation is adequate to decide the best action).

Non-Markov is multi-step domains with partial observation, where the state that presents is inadequate to decide the best action. Knowledge regarding one or more primary states is required in certain partly visible or seemingly identical states, known called aliased states, to select the correct action. As a result, prior states must be considered to decide the best action provided the latest sensor readings [32].

2.7.2 Anticipatory Classifier System

Anticipatory Classifier System (ACS) is a learning classifier system framework with an anticipation part. Later ACS2 has appeared, which combines the current ACS and some previously released revisions, as well as some additional changes and enhancements. A genetic generalization method has been added as a result of recent advances. The fact that ACS2 contains conditions, actions, and effects that anticipate little improvement in the world is a significant distinction between it and ACS. These scenarios were only indirectly defined in the ACS by the default expectation of no change. It has been assumed that if there is no classifier for a given situation-action tuple, the action would have no impact. The name ACS2 is intended to provide a new name for the existing state-of-the-art of this specific anticipatory learning classifier scheme, rather than to distinguish it from the previous ACS [33].

The ACS2 is represented by a population P, which is a set of classifiers. Each classifier is a C-A-E (condition-action-effect) structure that predicts the next model state when the action is carried out given the provided condition. In ACS2, a classifier always defines a complete outcome state. It is made up of the following elements [34]:

- The condition part (C) defines the range of scenarios in which a classifier can be used.
- The action part (A) provides a possible action.
- The effect part (E) anticipates the consequences of the provided action under the given conditions.
- The mark (M) saves the values of each attribute in all scenarios where the classifier failed to anticipate properly.
- The quality (q) determines how accurate the anticipations are.
- The reward prediction (r) evaluates the reward received after performing action A in condition.
- The immediate reward prediction (ir) measures the quantity of direct reinforcement received after performing action A in condition.
- The GA time stamp (t_{ga}) time once GA has just been applied.
- The ALP time stamp (t_{alp}) time once ALP has just been applied.
- The application average (aav) estimates a classifier updating frequency during ALP.
- The experience counter (exp) the how often the classifier was subjected to the ALP.
- The numerosity (num) specifies the number of micro-classifiers represented by this macro-classifier.

Table (2.1) Summery of the ACS2 Parameters

Parameters	**Default Value**	**Use**
inadequacy threshold (θ_i)	0.1	Used to check whether the ALP classifier quality is insufficient.
reliability threshold (θ_r)	0.9	Used to check if the new classifier coming from the ALP and GA process is a subsumer.
experience threshold (θ_{exp})	20	
learning rate (β)	0.05	Used to update ir, avv, and q of a classifier.
discount factor (γ)	0.95	Reduces the maximum reward expected in the next state.
Specificity threshold (u_{max})	Perceived string length	Defines the total number of condition attributes that are expected to remain the same in the effect part.
Exploration probability (ε)	0.95	Used to guide the GA process.
GA application threshold (θ_{ga})	100	
mutation rate (μ)	0.3	
crossover probability (χ)	0.8	
action set size threshold (θ_{as})	20	

The values perceived from the dataset (environment) and '#'-symbols make up the effect and condition parts. The '#'-symbol (also called 'don't-care'-symbol) in the condition refers that any value of the corresponding attribute in the data instance being matched by the classifier. While the effect part gives a different meaning; the '#'-symbol (or 'pass-through' symbol) indicates that the classifier expects the value of the corresponding attribute to remain unchanged once the specified action A is executed. The parameters that guide the learning process are shown in Table (2.1) [34].

Initially, the ACS2 starts with an empty set of a classifier. The first classifier is generated by the covering process which defines the observed changes in the condition and effect parts, as well as the action that was taken. During each behavioral act, the following steps are performed, see Figure (2.4) [33].

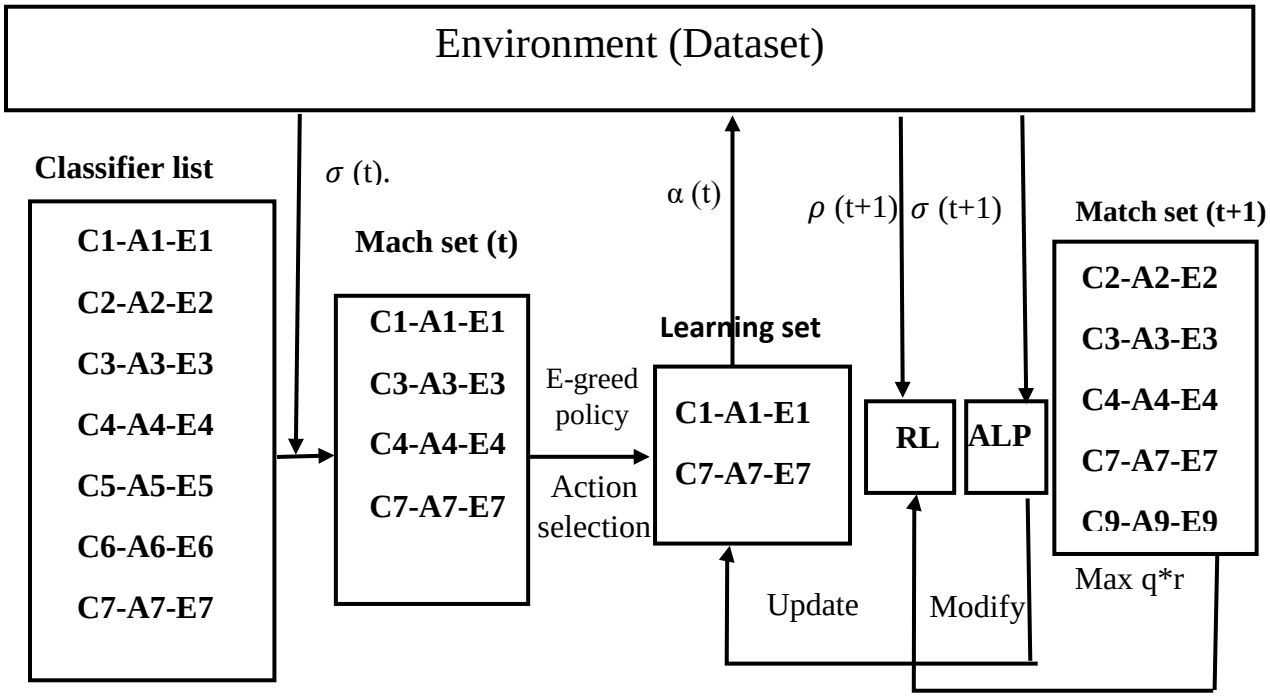

Figure (2.4): A behavioral act of ACS2

- **Step 1:** The system obtains the information perception and rewards at time t from the environment σ (t).

- **Step 2:** From the present population of classifiers**,** a matching set M is created. This set includes all classifiers whose condition parts are satisfied by the current perception σ (t).

- **Step 3:** The set M is used to select a classifier. An ε-greedy mechanism with a strength (quality * reward) was used to choose this classifier. The "active" classifier is the one that has been chosen. Finally, the action that will be performed is the one represented by the active classifier.

- **Step 4:** The M is used to create the learning set L. This set includes all classifiers that have the same action part as the current classifier. The learning set is defined as an action set.

- **Step 5:** Environment interaction, the output interface receives the selected action and executes it in the environment.

- **Step 6:** The system picks up information from the environment σ (t+1), ρ (t+1), and M_{t+1} is created.

- **Step 7:** the learning set L is presented to the Anticipatory Learning Process (ALP), which takes σ (t+1) into consideration. In the present population of classifiers, new classifiers that may have been produced during the ALP are added. Nonetheless, in step 2, the pair σ (t+1), ρ (t+1) begins a new behavioral act as a new perspective. The ALP updates the condition and the effect parts of the classifier based on the anticipation and current state. Certain situations may be presented as a result of this comparison [35]:

  -Useless-case: The environment perceives no change in perception after the execution of an action. The quality of the classifier (q) declines.

-Unexpected-case. When the new perception σ (t+1) does not match the effect part of the classifier (cl.E). The incorrect classifier is punished by decreasing its quality, and a new one with a matching effect part is created, the quality of classifiers decreased as in (2.2):

$$q \leftarrow q - \text{learning rate} * (1 - q) \quad (2.2)$$

- Expected-case. When the new observed perception matches with the classifier prediction. The quality of classifiers (cl. q) has improved, as in (2.3):

$$q \leftarrow q + \text{learning rate} * (1 - q) \quad (2.3)$$

- **Step 8:** Reinforcement Learning Process (RLP) is applied in the learning set L with the reward ρ (t+1). For each classifier in L, the reward prediction r, as well as the immediate reward prediction ir, are constantly updated to learn an ideal behavioral policy by using the highest values of the payoff predicted in M_{t+1} (MaxOf-P) as in (2.4) and (2.5):

$$r \leftarrow r + \text{learning rate} * (\rho + \text{discount factor} * \text{MaxOf-P} - r) \quad (2.4)$$

$$ir \leftarrow ir + \text{learning rate} * (\rho - ir) \quad (2.5)$$

- **Step 9:** the set L is subjected to the major discovery mechanism such as GA and two classifiers (parents) are chosen using a roulette wheel based on fitness (quality * reward), and offspring are formed to reintroduce into the population.

2.8 Post-processing

After the LCS has been applied, it is popular to use some type of rule compaction before using the resulting set of rules (Population P), as a model for predicting [32].

2.8.1 Rule Compaction

To use the population P to post-process the resulting rules, rule compaction and a similar technique known as condensation are often used after the LCS is applied. The aim is to eliminate poor or duplicate rules of P and create a complete sample of classifiers P_c. One of the very basic rules compaction solutions eliminates such a classifier in P that has never seen any training data yet (new rule) and the rule accuracy is less than 0.5. While we have a balanced data collection, which means that each class has an equal number of examples, this 0.5 accuracy cutoff becomes essential [32]. There are some rule compaction strategies [36] [37]:

Quick Rule Compaction: sort rules decreasingly by their accuracy, then calculate Match count and consider any rule with this parameter larger than zero for all instances in a dataset.

Quick Rule Filter: a simple filter looks at the rule collection and removes those rules with an accuracy of less than 0.5. In addition, if a rule covers less than two instances in the dataset, it is eliminated.

Parameter Driven Rule Compaction: takes into account different rule criteria (numerosity, generality, and accuracy). Throughout each iteration of the LCS, these parameters are modified. These parameters are taken into account in the PDRC technique as follows: Find the most effective rules that have the largest numerosity, accuracy, and generality product value.

2.8.2 Prediction

The output of an LCS method in supervised learning, whether or not rule compaction is used, is a collection of rules that may be used to draw conclusions

based on unknown examples. The prediction component uses the match set M, to render the class label for a specific instance by using the rule that fit that instance. This is usually accomplished by a voting system (i.e. the prediction goes to the class that earns the most points from the matching set). The prediction for a specific instance would be determined by the rule with the most number of votes (e.g., fitness and numerosity). More complex prediction methods have been suggested, and the prediction procedure has a crucial impact on reinforcement learning [32].

2.9 Evaluation

To evaluate the proposed model, a variety of resampling approaches are utilized, including the Cross-Validation, Holdout Method, and others. The Cross-Validation approach is presented in this section as it is used in this thesis. In addition, Time complexity is one of the most important criteria that used to evaluate the performance of the model.

2.9.1 The Cross-validation Method

To establish mutually independent repeating training and testing datasets, the cross-validation method is used. Cross-validation can be done in three different ways [38]:

- Two-fold cross-validation: The dataset is separated into two equal parts in this method. The first partition will be utilized for training, while the second will be used for testing. The partitions roles are then switched, with the prior training partition becoming the testing partition and conversely. Each

instance is utilized exactly once for training and once for testing in this method.

- K-fold cross-validation: This approach separates the dataset into K equal-sized subsets. This method is divided into K phases. In each phase, one subset is utilized for testing and the rest is utilized for training.
- Leave-one-out cross-validation: This is a type of k-fold cross-validation in which the dataset is equally divided into K partitions with k=N, where N is the number of instances. That means that N separate times, the function approximation is trained on all the data except for one point and a prediction is made for that point [39].

2.9.2 Evaluation Metrics

The evaluation measures are used to assess the trained model generalization power and quality when it is tested with new data. Different metrics can be used to measure the performance of classification methods. This includes accuracy, F1-measure, precision, and recall measures.

The accuracy of models is one of the most commonly used criteria to assess their generalization power [40]. The accuracy of the trained model is determined by the total number of instances correctly predicted by the trained model when tested with unseen data. To cope with imbalanced class difficulties and optimize accuracy performance, the recall, precision, and F1-measure metrics are used [40]. Calculating the confusion matrix provides the basis for these calculations. The number of cases predicted incorrectly or correctly by a classification model is summarized in this matrix, see Table (2.2) [41]:

Table (2.2): Confusion matrix

Confusion Matrix		Actual	
		Positive (1)	Negative (0)
Predicted	Positive (1)	TP	FP
	Negative (0)	FN	TN

- **TP:** True positives occur when the data instance real class is (1 = True) and the expected class is also (1 = True).
- **TN:** True negatives occur where the data instance real class is (0 = False) and the expected value is also (0 = False).
- **FP:** False positives occur where the data instance real class is (0 = False) but the expected class is (1 = True).
- **FN:** False negatives occur when the data instance real class is (1 = True) but the expected value is (0 = False).

Based on the just mentioned definitions presented through the confusion matrix, the evaluation measures are formally presented as follows [41]:

1. **Accuracy**: The ratio of correctly expected classifications (True Positives and True Negatives) to the overall test dataset is called accuracy [41].

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (2.6)$$

2. **Precision:** is the percentage of accurately expected positive instances (True-Positives) to other predicted positive instances, including accurate (True-Positives) and wrong (False-Positives) [41].

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (2.7)$$

3. **Recall**: The percentage of accurately expected positive instances (True-Positives) to any observations throughout the real class is known as recall (Actual Positives) [41].

$$Recall = \frac{TP}{TP+FN} \tag{2.8}$$

4. **F1-score**: is a weighted average of recall and precision [41].

$$\mathrm{F}_1 - \mathrm{score} = \frac{2\times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}} \tag{2.9}$$

Chapter Three

The Proposed Approach

3.1 Overview

In this chapter, the steps followed to achieve the key aim of this thesis are described. This includes proposing a model for the detection of violations in service-level agreement with partially observable environments. The architecture of this model is depicted first in section (3.2). Then, the steps of dataset generation are covered in section (3.3). The pre-processing stage explained in section (3.4). The training stage which include the validation of the model and applying the classification model are explained in detail in section (3.5). While rule compaction as a post-processing is discussed in section (3.6). The last section shows how the classification model can be implemented online.

3.2 Outline of the Proposed Approach

In this section, the proposed approach is covered. The steps involved in this approach are depicted in Figure (3.1). Mainly, there are offline and online steps in which all the other steps are included. During the offline step, the dataset used to train the model is generated by firstly defining the SLA constraint and then applying the generation algorithm which simulates a Petri net model. Following that the resulting dataset is subjected to pre-processing to eliminate the imbalance problem of data. Then a trained model is produced using the ACS2 classification model. Finally, this model is used online to detect SLA violations.

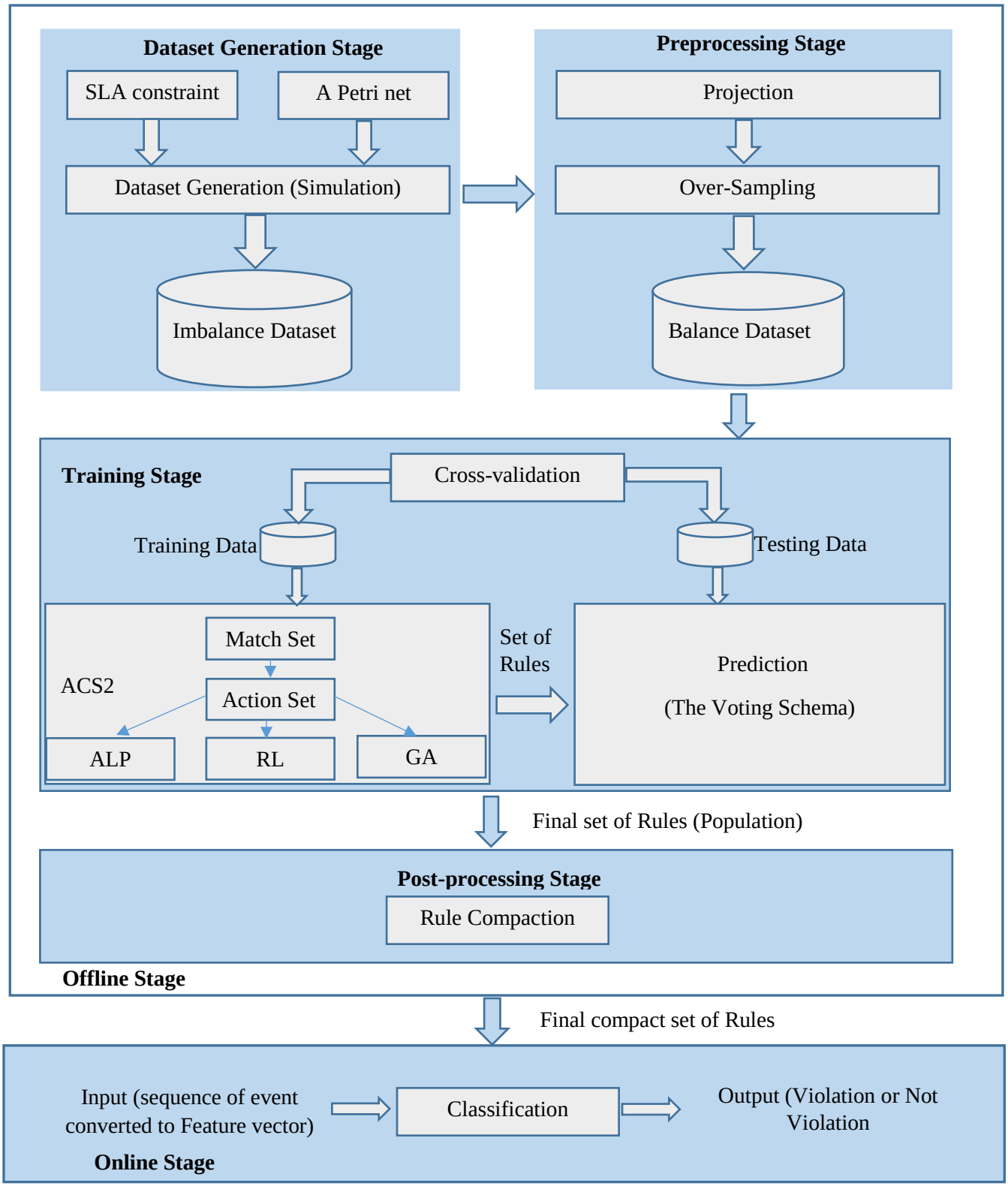

Figure (3.1): The Proposed Approach

3.3 Dataset Generation

The dataset is generated by simulating the Petri net model of the running example in Figure (2.3) of section 2.4.2. The dataset generation stage consists of two steps as shown below:

1. **The SLA Constraint Definition**:

A SLA includes several constraints which can be formulated as inequalities. These inequalities could take the following general form: ($a_1x_1 + a_2x_2 + \cdots a_nx_n \leq b$), where x refers to the SLA parameters (observable events) under the specified threshold b, i.e. x_i is the number of occurrences of a given event e_i and a_i is a constant. For instance, in the telecommunication company example (see Figure (2.3)), a SLA parameter is represented by a relationship (an inequality) between the number of allocations (t_2) and the number of completions (t_{15}) of the task that happens. If the constraint was ($x_2 - x_{15} \leq 5$), then x_2 is the number of occurrences of the event t_2 and x_{15} refers to the number of occurrences of the event t_{15}; also a_2=1 and $a_{15} = -1$. Therefore, depending on the constraint specified in the SLA, a decision is made on whether a violation occurs when observing a sequence of events.

2. **Applying Generation Algorithm**:

Each instance of the dataset is obtained as follows. Firstly, a sequence of events (observable and unobservable) is generated. For each sequence, a label (0 means the sequence satisfies the constraint and 1 means there is a violation) is computed and added to the sequence. Assume that the SLA constraint is defined as in (3.1).

$$x_2 - x_{15} \leq 2 \quad (3.1)$$

Where x_2 refers to the number of allocations of a task (number of firing t_2) and x_{15} refers to the number of completion of the same task (number of firing t_{15}). Note that the transition t_2 is an unobservable one which is the most interesting case. If the inequality in (3.1) is true, there is no violation (the label is 0), else an SLA violation has occurred (the label is 1).

For example, referring to Figure (2.3), suppose that the following two sequences $t_1\ t_2$ and $t_1\ t_2 t_3\ t_6 t_2\ t_3 t_6\ t_2$ are generated. Based on the constraint specified in (3.1), the first sequence has no violation, while the second sequence has caused a violation. As a result, the sequences $t_1\ t_2$: 0 and $t_1\ t_2 t_3\ t_6 t_2\ t_3 t_6\ t_2$: 1 are added to the dataset, see Algorithm (3.1).

Given an SLA constraint and a Petri net model N, Algorithm (3.1) outputs a dataset D. The dataset D is initialized in step 3; and incidence matrix A of Petri nets is computed in 4, Based on the set of the enabled transitions E, one transition is chosen randomly from this set to be fired in step 8. Then, in step 9, the features vector is updated according to the fired transition by increasing its corresponding feature entry by one, to determine the number of times that this event appears in the sequence. In step 10, the state equation is used to determine the next (marking) state of a task resulting from firing an event. Steps from 11 to 15 explain how the class label of the features vector (data instance) is determined based on the given SLA constraint. The determined vector is added to the dataset, and this loop continues until the task is completed to start a new one, where each task has a number of instances.

Algorithm (3.1). Dataset generation

Input: A SLA constraint, and a Petri net Model N=(P, T, Pre, Post, M_0), where n =|P| and m=|T|; and K is the number of tasks

Output: A dataset, D

```
Let E be the set of enable transitions
Let X = (x1, x2, … , xm) be the features vector
Initialize D to an empty set
A ← Post – Pre
k ←0
while k < K do
    while the taskk is not completed do
        choose randomly a transition er from E // r is a
        random number between [0-m]
        X[xr] ← X[xr] + 1
        M ← M0 + A * X
        if the SLA constraint w.r.t. X is held then
            label ← 0
        Else
            label ← 1
        End if
        Append label to X
        Add X to D
        M0 ← M
    End while
  k ← k+1
End while
```

3.4 Pre-processing Stage

Pre-processing is intended to prepare a dataset in an appropriate form for machine learning techniques. This stage has been used in this thesis to balance the

dataset and remove all unobservable features. Data pre-processing consists of two steps:

1. **Feature Projection**

Feature projection transforms the data from the high-dimensional space to a space of fewer dimensions. The dataset generated previously has two sets of features based on whether these features correspond to observable or unobservable events in the systems being monitored. The detector of SLA violations can see only the observable events which are used to make the detection (classification) decision. To simulate the idea of not being able to see unobservable (unseen) events, the features capturing these events will be simply eliminated (ignored) leaving only the set of features corresponding to observable events. In other words, the feature space is projected on another space having only the set of features of the observable events. This gives fewer features to represent each data instance and limits the number of features to the number of observable events in the system.

2. **Data Sampling**

Note that when generating the dataset the number of instances that have a violation is less than the number of non-violated instances. This is an expected case as the violation realistically happens rarely compared with the non-violated instances. The dataset obtained is highly imbalanced, a phenomenon known as data skewness, which makes classification extremely difficult. This is due to the fact that the classifier will almost always predict the dominant class. When minority class instances are missing, however, the true misclassification costs may be significantly higher. For example, the real "violated" tasks which are incorrectly predicted can harm providers reputation and result in a loss of profit.

Generating an instance from the minority class is preferable to replicating it. This form of data augmentation can be really beneficial. Synthetic Minority Oversampling Technique (SMOTE) is the most widely used method for producing new instances to address the problem of imbalanced classes that leads to incorrect classifications. To be somewhat more explicit, a random instance of participants from the minority class is chosen first. The closest k neighbors for that instance are then found. One of the k neighbors is picked randomly, and a new instance is constructed at a random position in feature space between two instances.

3.5 Training Stage

The balance dataset obtained previously is then used as training data to validate the performance of the proposed model. K-fold stratified cross-validation method is applied to train the model using an ACS2.

3.5.1 Validation of the Proposed Model

For the sequence data, K-fold stratified cross-validation method is presented to validate the performance of the classifier system. This method is used on a rolling basis beginning with the whole dataset and dividing it into k partitions as follows. Assume that D is the whole dataset and S represents its size. Then, D is divided such that D={ $d_1, d_2, d_3, \ldots, d_k$), where the size of d_i is S/k. In iteration i, $\bigcup_{j=1}^{i} d_j$ is assigned to the training set and d_{i+1} to the testing set. In each iteration, the value of the evaluation metric being considered is computed and taken the average over the k iterations and report as a final score, see Figure (3.2).

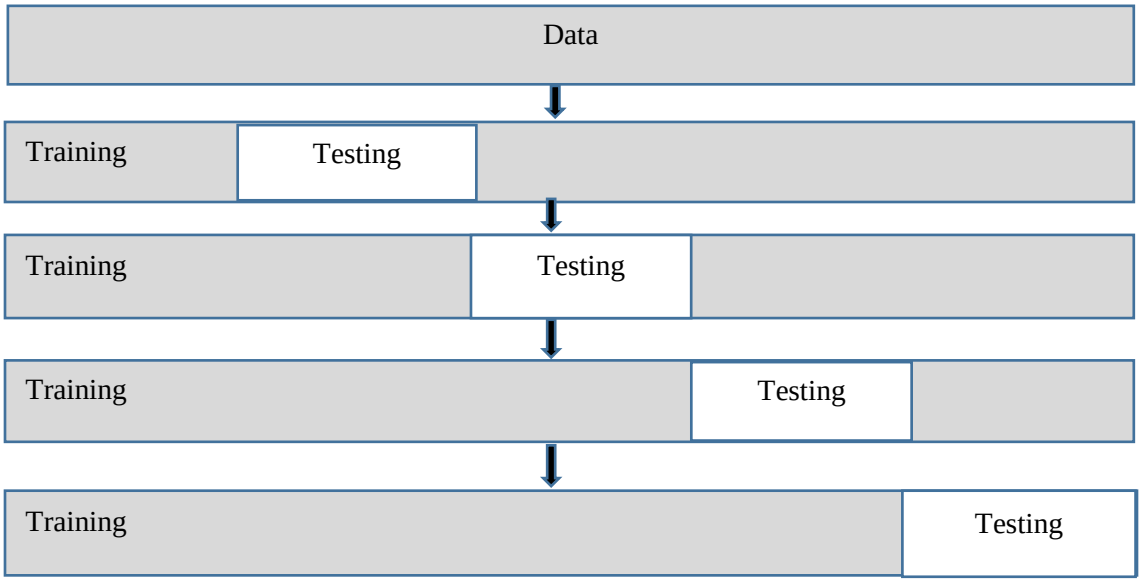

Figure (3.2): K-fold Cross-validation

3.5.2 Anticipatory Classifier System

Before an ACS2 run can begin, the parameters used to guide the learning process need to be determined, and the dataset used for training need to be constructed.

In the ACS2 Algorithm (3.2), the present state is first sensed (X). Second, all classifiers that match the state (X) are combined to generate the match set. The ALP, RL, and GA are applied in the last action set (last learning set L_{-1}) if this is not the beginning of a task (trail). Following that, an action is selected for execution, the action is carried out, and an action set (learning set L) is formed from all classifiers in the match set that specify the action. The received payoff is determined based on the selected action as in steps from 13 to 16. If the execution of the action resulted in the completion of a task, the ALP, RL, and GA may be used in the action set (learning set L) after some parameter updates. To avoid improper learning, the

previous learning set must be cleared at the end of the task. Many sub-procedures are specified in the main loop which is described in more detail below.

Algorithm (3.2). ACS2 algorithm

Input: Dataset
Output: population P

Initialize match set M, learning set L, and population P with an empty set
Initialize payoff $\rho \leftarrow 0$
while The requirements for termination are not met do
    $X \leftarrow$ get an instance from the dataset
    constructing the match set M
    if L_{-1} is not null then
        Apply ALP () in L_{-1} // algorithm (3.3)
        Apply RL () in L_{-1} // algorithm (3.7)
        Apply GA () in L_{-1}
    End if
    selected action ← Choose Action by ε-greedy policy
    constructing the learning set L
    if the selected action is equal to X label then
        $\rho \leftarrow 1000$
    Else
        $\rho \leftarrow 0$
    End if
    $X_{-1} \leftarrow X$
    $X \leftarrow$ get an instance from the dataset
    if $X[-1] = 1$ // task completed
        Apply ALP () in L
        Apply RL () in L
        Apply GA () in L
    End if
    $L_{-1} \leftarrow L$
End while

A. Generating Match Set

The present state (X) and the given population are fed into the process of generating the match set. All classifiers in the given population are basically compared to X, and the match set is expanded to include all classifiers that its condition part matches the current state X.

In the matching process, a (#) symbol in Condition matches any value in the relevant point of the current state (X). A (non-#) can only be matched with the same value in the same position. Each specified attribute in the classifier condition is compared to the current situation corresponding attribute. The classifier matches and the matching process returns true only if all comparisons hold.

B. Choosing an Action

The ε-greedy approach is commonly used in ACS2 for action selection. Because multiple different classifiers can represent one state, it is difficult to say which action is the best to take. In this scenario, a straightforward strategy is used for selecting the action of the most promising classifier (the best classifier in M which has maximum quality * reward).

C. Generating Learning Set

When the matching set (M) is constructed and the action is determined to be executed, the learning set is generated based on M. It contains all classifiers in M that recommend the selected action.

D. Anticipatory Learning Process

The use of the ALP is somewhat sensitive. The newly created classifiers need to be inserted into learning set L but not revisited in the present ALP application due to its synchronous creation and deletion of classifiers. Deleted classifiers must be removed from L to avoid interfering with the update process. The main loop of the ALP process is described briefly in the Algorithm (3.3).

Algorithm (3.3). Anticipatory Learning Process ()

Input: Learning set L, X_{-1}, selected action, X, time t, and population P
Output: New classifier add to the population P

```
Initialize Cl-new to an empty classifier
For each classifier in L
    classifier experience ++
    Update application average(avv) for that classifier
    if the classifier correctly anticipate X then
        Cl-new ⟵ Expected Case () // algorithm (3.4)
    Else
        Cl-new ⟵ UnExpected Case () // algorithm (3.5)
    End if
    if classifier quality < inadequacy threshold then
        delete classifier from L and P
    End if
    Add Cl-new to P and L
 if no classifier in L anticipate X correctly then
    Cl-new ⟵ Covering Process () // algorithm (3.6)
    Add Cl-new to P and L
 End if
 End For
```

Moyenne adaptive modif´ee technique has been used to adjust the application average for each classifier in L to get at a precise application average (avv) value

quickly and smoothly. The ALP process considers each classifier anticipation (Effect part) in sequence. The expected case (step 6) or unexpected case (step 8) is called if the classifier is correctly or incorrectly anticipated the next state. It is required to evaluate if the quality of a classifier has dropped below the (θ_i), in this case, the classifier needs to be removed (step 11).

Checking for equivalent classifiers and maybe subsuming classifiers is required when introducing a new classifier (steps 13 and 16). In both the GA and the ALP applications, ACS2 searches for subsuming classifiers. A subsumer must be reliable, experienced, and unmarked in order for a classifier to subsume another one. Furthermore, the condition component of the subsumer must be more general, and the effect part must be the same. Then, a covering classifier is constructed and introduced if no classifier in L appropriately anticipates the state X (step 15).

Two directions can be explored in the expected case, whether a new classifier is introduced or not, see Algorithm (3.4). No classifier is created when there are no differences identified between the presented classifier mark and the current state (X) (step2). The classifier quality is improved in this situation (step 3), and no additional classifier is produced. In the case of there being differences identified between the classifier mark and the state (X), offspring is produced. Removing specified attributes process (steps 10, 14, and 17) applied when the generated offspring is specified to too many attributes value or when the number of specified attributes in the offspring exceeds the specificity threshold.

Algorithm (3.4). Expected Case ()

Input: current active classifier, and current state X
Output: New classifier to be added to the population P
1 dif ← differences between the mark and the current state
2 **if** (dif = $\{\#\}^L$) then

```
3          increase classifier quality // as in equation (2.3)
4          return empty
5       Else
6          Spec 1 ← number of specified attributes in classifier
           Condition
7          Spec 2 ← number of specified attributes in dif
8          offspring ← copy of classifier
9          if ( Spec 1 = specificity threshold) then // default value
           of specificity threshold equal to length of X
10            remove specific attributes in offspring condition
11            Spec 1--
12            while ( Spec 1+ Spec 2 > specificity threshold)
13                  if  Spec 1 > 0 and random number (0-1) < 0.5
14                     remove specific attributes in offspring
                       condition randomly
15                     Spec 1--
16                  Else
17                     remove specific attributes in dif randomly
18                     Spec 2--
10                  End if
20            End while
21         Else
22            while ( Spec 1+ Spec 2 > specificity threshold)
23                  remove specific attributes in diff randomly
24                  Spec 2--
25            End while
26         End If
27         specify offspring Condition with dif
28         if  offspring quality < 0.5 then
29            offspring quality = 0.5
30         End if
31         offspring  experience ← 1
32      End if
```

When the active classifier does not anticipate the next state correctly, an unexpected case procedure is invoked, see Algorithm (3.5). In steps 3 and 4, the classifier quality is decreased, and the classifier mark is updated with respect to the previous state X_{-1}. The requirement for creating an offspring classifier is essential. The offspring is created when it is possible to adjust the effect part to anticipate correctly the next state by only specifying attributes when required. In the steps from 4 to 8 no classifier is created because there is no change is countered.

Algorithm (3.5). Unexpected Case ()

Input: current active classifier, X_{-1}, and X
Output: New classifier to be added to the population P

```
1    Decrease classifier quality // as in equation (2.2)
2    Classifier mark ← X_-1
3    For all attributes c in X
4       if classifier effect [c] ≠ #
5          if classifier effect [c] ≠ X[c] or X_-1 [c] = X[c]
6             Return null
7          End if
8       End if
9    End For
10   offspring ← copy active classifier
11   For all attributes c in X
12      if classifier effect [c] = # or X_-1 [c] ≠ X[c]
13         offspring condition[c] ← X_-1 [c]
14         offspring Effect[c] ← X [c]
15      End if
16   End For
17   if classifier quality < 0.5
18      classifier quality = 0.5
19   End if
20   offspring experience ← 1
```

In ACS2, the covering process is used to encompass all conceivable situations that occur, which are represented by a triple (C-A-E). During the ALP, if the learning set L does not contain a classifier that anticipates the current state X correctly, covering is applied. Covering creates a classifier that defines any changes in the condition and effect part from the previous state (X_{-1}) to the current state (X). The new classifier action part (A) is assigned to the selected action. See Algorithm (3.6).

Algorithm (3.6). Covering Process ()

Input: X_{-1}, selected action, X, and current time t
Output: classifier anticipate the next state correctly

```
1    Initialize an empty classifier Cl
2    For all positions c in X
3        if  X_-1 [c] ≠ X[c]
4            Cl condition[c] ← X_-1 [c]
5            Cl effect[c] ← X [c]
6        End if
7    End For
8    Specify classifier Cl action part with the selected action
```

When the ALP creates a new classifier (offspring), it is essential to check if there is a subsumed classifier or equal classifier to the new offspring generated before the insertion process is done. If none were discovered, the classifier generated is added to the population and the current learning set L as a new one. If a subsumer or identical classifier is identified, the old classifier quality is improved, and the new one is rejected.

E. Reinforcement Learning

In reinforcement learning (RL), the highest possible payoff estimated in the next match set M_{t+1} (MaxOf-P) are used to update the reward and the immediate reward for each classifier in the current Learning set L. See Algorithm (3.7).

Algorithm (3.7). Reinforcement Learning ()

Input: learning set L, ρ, Maximum Payoff in the next time-step MaxOf-P
Output: Updated reward r, and immediate reward ir

```
1    For each classifier  L
2        Update classifier reward w.r.t the ρ  and Maximum
         Payoff ()  // as in equation (2.4)
3        Update classifier immediate reward w.r.t  to ρ    // as in
         equation (2.5)
4    End For
                          Maximum Payoff ()
     Cl← first classifier  in M_t+1
6    For each classifier c ∈ M_t+1
7       if  c.quality * c.reward >Cl.quality *Cl.reward
8           Cl←c
9       End if
10   End For
11   MaxOf-P ←  Cl.quality * Cl.reward
```

F. Genetic Algorithm

The genetic algorithm (GA) procedure begins by determining whether or not a GA should take place, which is determined by the summation of numerosity for all classifiers in learning set L. The most accurate, over-specialized classifiers are selected using a Roulette-Wheel Selection mechanism when genetic generalization

is applied. After the offspring is selected, the mutation operator is applied by changing specified attributes in the condition part into the #-symbol. Following that a crossover operator is applied between two offspring chosen from the set L if only the two selected classifiers (offspring) have the same effect part. Unnecessary classifiers are removed from L before the newly generated classifier is added. The classifier has a worse quality than the rest is removed. Marked classifiers are deleted first if all classifiers have equal quality, or the least applied classifier is chosen. When a classifier is deleted from the population, or when the classifier numerosity drops to 0, it must be deleted from both the learning set and the entire population.

The GA insertion approach is somewhat in the same way as the ALP insertion approach, it differs in two key ways. First, rather than increasing the quality of an old, subsuming or identical classifier, the numerosity is increased. Second, an identical classifier numerosity is only raised if the classifier is not marked.

3.5.3 Prediction

The final rule set resulting from training the model using ACS2 is used for predicting the testing dataset (unlabeled instance). Firstly, for each testing instance, the match set M is generated. Then, a voting scheme is applied to the M to predict the class label for that instance. A voting scheme is applied by dividing the matching set into two subsets, the first one includes all rules that carry a violated action, while the second subset includes all rules that have no violated action. Then, in each subset, the rule with the highest fitness (quality * reward) is selected. After that, the model prediction of that instance would be determined by the rule with the largest vote contribution, see Algorithm (3.8).

Algorithm (3.8). Prediction

Input: Testing dataset, Final Rule Set
Output: The class label for that testing instance

```
Initialize Predictions to an empty set
Testing-instance ← get an instance from the testing dataset
Predictions ←generate Match set M for Testing-instance
Predicted-label ← Apply Voting Scheme on Predictions
                    Voting Scheme
Initialize two sets: Set 1, Set 2
Initialize two classifiers: CL 1 , CL 2
For each classifier cl in Predictions
   if cl. action equal to 1 then
      Adding cl to Set 1
   Else
      Adding cl to Set 2
   End If
End For
CL 1 ← classifier with maximum quality * reward in Set 1
CL 2 ← classifier with maximum quality * reward in Set 2
if CL 1.quality > CL 2.quality then
   Predicted-label ← 1
Else
   Predicted-label ← 0
End if
```

3.6 Post-Training Stage

When the training phase has been ended, rule compaction need to be applied as a post-training process. Some common algorithm of rule compaction is applied to obtain a set of optimal rules, which is then used to make predictions on unseen samples.

3.6.1 Rule Compaction

The process of adding new classifiers to the population and deleting ones (the rule that has less quality) makes the population change continuously. An undetermined set of rules are expected to present in the rule population at every single learning iteration that makes little or no impact on the system overall performance. These include newly created rules which are identified as inadequate rules (classifiers quality is less than inadequacy threshold). Recently, some techniques for cleaning post-training rule populations developed by Learning Classifier System (LCS) have been offered. Parameter Driven Rule Compaction (PDRC) is one of the most suitable techniques used for rule compaction in LCS systems. The PDRC identifies the classifier in learning set L that has the highest product of accuracy, generality, and numerosity for each training instance and keeps that rule in the final rule set (population P), see Algorithm (3.9).

Algorithm (3.9). Parameter Driven Rule Compaction

Input: Population P, Training Dataset
Output: Compact optimal rule set (Final Rule Set)

```
1    Initialize Final Rule Set to an empty set
2    For each instance c in the training data
3        constructing Match Set M
4        Action ← the class label for instance c
5        constructing learning set L for Action
6        ThebestCl← classifier with the maximum product of
         accuracy, generality, and numerosity in L.
7        if ThebestCl ∈ Final Rule Set then
8            Pass
9        Else
10           Adding ThebestCl to the Final Rule Set
11       End if
12   End For
```

3.7 Online Stage (Performance Measurement)

During the online step, the training model can be utilized as a detector to determine whether a SLA violation has occurred or not. The system is then being monitored to obtain sequences of observed events. These sequences are transformed to feature vectors, each sequence is converted to a feature vector by counting the occurrences of each event in the sequence. The next step is to use the model (a set of rules) generated in the offline step to decide the SLA violations. To make a decision whether the violation occurs or not, the prediction algorithm (3.8) is applied.

So the time needed for the prediction can be estimated as follows:

1. Generating the match set for a record needs to be tested, this step needs linear search O (nm), where n refers to the number of rules, and m refers to the length of rule condition.
2. Applying a voting scheme, see Algorithm (3.8), will take O (n). Hence, given a sequence of observed events, the total time required to make a detection decision is:

$$f(n) = O(nm)$$

Chapter Four

Experimental Results and Discussion

Chapter Four Experimental Results and Discussion

4.1 Overview

The proposed methodology outlined in Chapter three has been implemented to accomplish the research objectives outlined in Chapter one. The results of the experiments are detailed and discussed in this chapter.

4.2 Software and Hardware Requirements

The proposed model has been implemented using the following hardware and software requirements.

Hardware: Processor Intel ci7, RAM 12GB, Storage 1000 GB.

Operating System: Windows10 (64) bit.

Programming Language: Python 3.8.

4.3 The Dataset

The dataset used to train the model is generated by simulating the running example of a telecommunication company as mentioned in Section (2.4.2). Different types of datasets are generated to train and test the proposed model, each of them represents a model with a different problem space, see Table (4.1). Each data instance in this dataset consists of a number of features each of which corresponds to the number of occurrences of a certain event within the system.

The number of features generated for each data instance is determined by the problem space (number of departments within the system). Whenever the number of departments increases, the number of features generated increases too. The generated features are divided into two subsets observable (seen), and unobservable (unseen). Where the rate of observed features is only 30-33% of the total number of features. The rate of the observed event can be computed as follows: (number of observed events / total number of events) *100%. For example, if the system being analyzes has five departments and the number of events occurs within the system is 23 events, only seven of these events are observable, so the rate of the observed events is 30%.

Table (4.1): Datasets Generated

	Number of departments	Number of features	Number of instances
Dataset 1	3	15	8900
Dataset 2	5	23	8916
Dataset 3	10	43	9064
Dataset 4	15	63	9870
Dataset 5	20	83	9947

Determining the class label of the dataset instances is mainly based on SLA constraints that are predefined by the service provider in agreement with the customer. Considering dataset 1 which has three departments and fifteen transitions (events) that will represent the features of that dataset, see Table (4.2) that shows what each feature represents and its type that obtained from figure (2.3). Assume that the SLA constraint is defined as in (4.1).

$$x_2 - x_{15} \leq 2 \quad (4.1)$$

Where x_2 refers to the number of allocations of a task (number of firing t_2) and x_{15} refers to the number of completion of the same task (number of firing t_{15}). Note that the transition t_2 is an unobservable one which is the most interesting case. If the inequality in (4.1) is true, there is no violation (the label is 0), else an SLA violation has occurred (the label is 1), see Table (4.3) which shows a sample of the dataset generated based on algorithm (3.1).

Table (4.2): Features Description for Dataset 1

Features	Description	Type
t_1	the arrival of the task to the system	observable
t_2	the allocation of the task to one of the departments	unobservable
t_3	the arrival of the task to department one	observable
t_4	the arrival of the task to department two	observable
t_5	the arrival of the task to department three	observable
t_6	the task in department one was not resolved	unobservable
t_7	the task in department one was resolved	unobservable
t_8	the task in department two was not resolved	unobservable
t_9	the task in department two was resolved	unobservable
t_{10}	the task in department three was not resolved	unobservable
t_{11}	the task in department three was resolved	unobservable
t_{12}	the task within department one is completed	unobservable
t_{13}	the task within department two is completed	unobservable
t_{14}	the task within department three is completed	unobservable
t_{15}	the task within the system is completed	observable

Table (4.3): Sample of Dataset 1 Generated

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	label
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0
1	2	0	0	1	0	0	0	0	1	0	0	0	0	0	0
1	2	1	0	1	0	0	0	0	1	0	0	0	0	0	0
1	2	1	0	1	1	0	0	0	1	0	0	0	0	0	0
1	3	1	0	1	1	0	0	0	1	0	0	0	0	0	1
1	3	2	0	1	1	0	0	0	1	0	0	0	0	0	1
1	3	2	0	1	2	0	0	0	1	0	0	0	0	0	1
1	4	2	0	1	2	0	0	0	1	0	0	0	0	0	1
1	4	2	0	2	2	0	0	0	1	0	0	0	0	0	1
1	4	2	0	2	2	0	0	0	2	0	0	0	0	0	1
1	5	2	0	2	2	0	0	0	2	0	0	0	0	0	1
1	5	3	0	2	2	0	0	0	2	0	0	0	0	0	1
1	5	3	0	2	2	1	0	0	2	0	0	0	0	0	1
1	5	3	0	2	2	1	0	0	2	0	0	0	1	0	1
1	5	3	0	2	2	1	0	0	2	0	0	0	1	1	1

Each instance of the dataset represents the state of the task in each movement. The process will continue within the system until the end of the task (completion) to then start a new task. To extract more information from the sequences of events in order to increase the recognition characteristics, a sequence order (ID) is added as a feature to the dataset instances.

The number of instances generated is based mainly on the number of tasks entered into the system to be solved and the SLA constraint threshold (fault tolerance). Increasing the number of tasks results in an increase in the number of instances obtained, where each task requires many instances to be completed, see Figure (4.1). The number of instances that belong to one task depends on the SLA constraint threshold. The more the system allows a high fault tolerance rate, the longer the time to complete the task, and the increase in the number of instances that belong to a task as shown in Figure (4.2). These are not optimal cases, it makes the customer dissatisfied, the perfect system must be very restricted and allow in a few numbers of faults (violations).

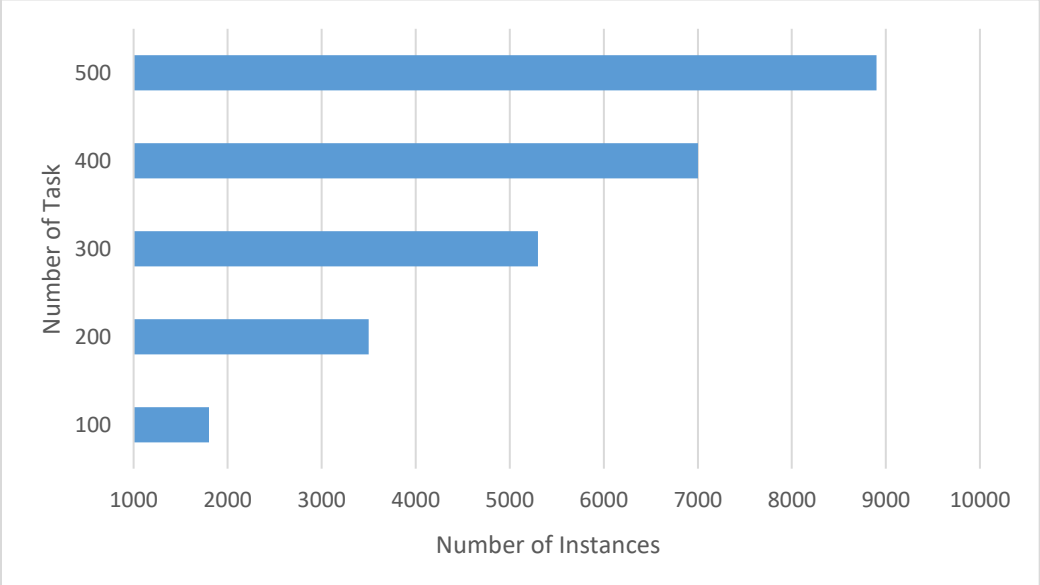

Figure (4.1): Number of Instances with Respect to Tasks Number

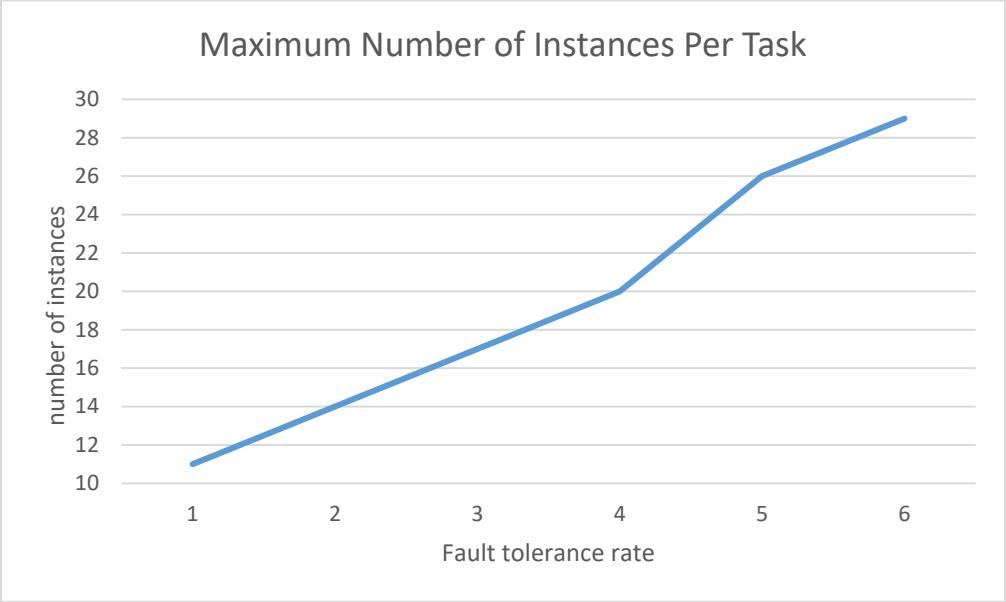

Figure (4.2): Number of Instances with Respect to Fault Tolerance

4.4 Result of Pre-processing

Two important steps have been performed on the dataset obtained from the previous step, namely feature projection, and the over-sampling technique.

In many cases, the agent will not be able to have a precise and comprehensive understanding of the status of the environment. Some of the feature values will be unobserved. To simulate such a case in the present work, a feature projection on the observable features is applied where all unobservable features are ignored and the training model mainly relies on the observable features in the learning process. Although the decision of the model may be based on some of these unobservable features. The form of the dataset after feature projection will be in the following form, see Table (4.4).

Table (4.4): The Dataset 1 After the Feature Projection Step

t_1	t_3	t_4	t_5	t_{15}	label
1	0	0	0	0	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
1	0	0	1	0	0
1	1	0	1	0	0
1	1	0	1	0	0
1	1	0	1	0	1
1	2	0	1	0	1
1	2	0	1	0	1
1	2	0	1	0	1
1	2	0	2	0	1
1	2	0	2	0	1
1	2	0	2	0	1
1	3	0	2	0	1
1	3	0	2	0	1
1	3	0	2	0	1
1	3	0	2	1	1

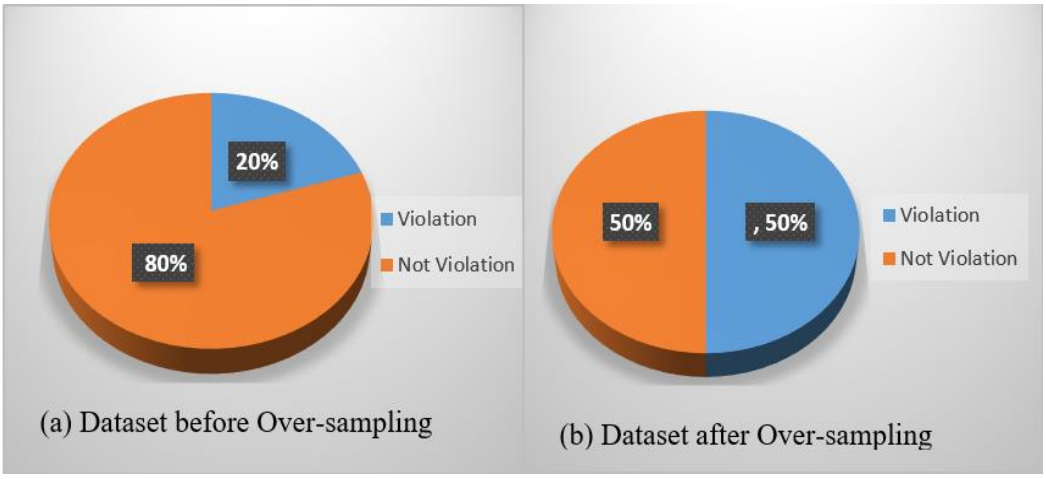

Figure (4.3): Dataset Before and After Over-sampling

Violation is a very rare event. Therefore the dataset generated is an imbalanced dataset, where the number of instances that have violations is only 20% of the overall dataset. The accuracy obtained from training the model with an imbalance dataset is only 88%. To increase the accuracy of the model, an oversampling technique is applied. As mentioned, in this work, The Synthetic Minority Oversampling Technique (SMOTE) is adopted. As a result, a balanced dataset is generated, see Figure (4.3), and the accuracy obtained after oversampling is applied is 99%.

4.5 Anticipatory Classifier System

Before an ACS2 is run, the parameter values used to control the learning process are set as follows: θ_i = 0.1 (inadequacy threshold), θr = 0.8 (reliability threshold), β = 0.05 (learning rate), γ = 0.95 (discount factor), u_{max} = L (specificity threshold, L is the length of string percept from the dataset), ε = 0.95 (exploration probability), θ_{ga} = 100 (GA application threshold), μ = 0.001 (mutation rate), χ =

0.8 (crossover probability), θ_{as} = 20 (action set size threshold) and θ_{exp} = 20 (experience threshold), which are standard values.

Firstly, we will train the model by dataset 2 generated by simulating a model with five departments, 8916 instances (half of them have a violation), and the number of features obtained is 23 features. Only 7 of these features will be selected for training (observed features). These instances were used to construct our model and evaluate it by using a 10-fold stratified cross-validation technique. The number of instances subjected for training and testing in each fold is explained in detail in Table (4.5). As shown, all data instances will be trained and tested except the data in the first and last fold. The data instances in the first fold will be subjected to training only, and the data instances in the last fold will be subjected to testing only.

Table (4.5) show the result obtained from training the dataset in ACS2. At each learning iteration, several rules (classifiers) are added to the population P, many of them are generated from the Anticipatory Learning Process (ALP), and the rest comes from applying the genetic generalization pressure (GA) either by mutation or cross-over operators. The number of rules added to the population increased through time from 181 rule to 528, as well as the number of reliable classifiers (classifier quality more than reliability threshold), which gives a good impression about the rules included in P which in turn increases the accuracy of the model.

In the ALP process, an increasing number of the classifiers generated due to the expected case means that the model is learning in the right direction and the accuracy of prediction increases, as shown from Table (4.5), it is increased from 1 to 11 in each learning iteration.

Table (4.5): Result of ACS2 Training

k-fold	1	2	3	4	5	6	7	8	9	10
Training instances	816	1626	2436	3246	4056	4866	5676	6486	7296	8106
Testing instances	810	810	810	810	810	810	810	810	810	810
Number of rules	181	206	292	337	352	417	433	486	495	528
Reliable rules	12	16	16	20	20	21	21	22	23	24
ALP Rules	1	2	3	3	5	6	8	8	9	11
GA rules	0	0	0	0	1	1	3	3	5	6
Deleted rules	0	0	3	8	12	34	43	56	79	106

In ACS2, an optimal model is defined as a full model represented by the fewest number of classifiers, with all classifiers specifying the fewest number of features in the condition part and the model being accurate. Applying ALP leads to generating over-specialized rules (many numbers of specified features). Therefore genetic generalization pressure is intended to obtain more general rules. Before the two generated classifier from GA is inserted into the population P, a deletion process is intended if the number of rules in the learning set is over the specified threshold (θ_{as} = 20). The number of deleted classifiers increased through time as shown in Table (4.5), this step is to obtain an optimal model with less number of rules and excludes poor redundant rules from the population, where the number of deleted rules arrives at 106 rule.

4.6 Result of Rule Compaction

The implementation of the PDRC rule compaction algorithm results in a compact rule set in which the number of rules is reduced while preserving the accuracy rate high. Where all classifiers with high fitness (quality * reward), large numerosity, and more generally are included in the final compact ruleset which is considered as the detection model. Table (4.6) show how the number of rules decreased after PDRC was applied for all datasets generated, and the accuracy before and after rule compaction.

Table (4.6): Reducing the number of rules using PDRC

Problem Space	Number of Rules	Accuracy before rule compaction	Final Rule Set after Rule compaction	Accuracy after Rule compaction
Dataset 1	510	0.989	40	0.986
Dataset 2	521	0.988	41	0.982
Dataset 3	698	0.967	44	0.966
Dataset 4	711	0.966	46	0.963
Dataset 5	730	0.954	47	0.95

4.8 Evaluating the Model

In the first experiment, dataset 2 of size 8916 instances divided as 8106 instances for training and 810 instances for testing is used. The number of rules obtained from training the model is 521 where 24 of them are reliable classifiers whose quality is more than 0.9. The rate of accuracy obtained from the overall splits in 10-fold cross-validation is 98%, and the number of instances that are correctly classified is depicted via a confusion matrix in Table (4.7). It can be observed that only 13 instances out of 488 instances that have a violation are misclassified.

Table (4.7): Confusion Matrix

	Actual (0)	Actual (1)
Predicted (0)	322	0
Predicted (1)	13	475

The metrics used to evaluate the model are accuracy, recall, precision, and F-measure. As expected, Figure (4.4) shows how the evaluation metrics values increase with increasing the training dataset size. As the training dataset increases, the model accuracy increases. The dataset size has a high impact on the performance of the model. Increasing the problem space (increasing number of departments) requires increasing the dataset size to avoid the overfitting problem and obtain high-quality performance of the training model. Figure (4.5) depicts the effect of the training data size on the performance of the model when the problem space is very large.

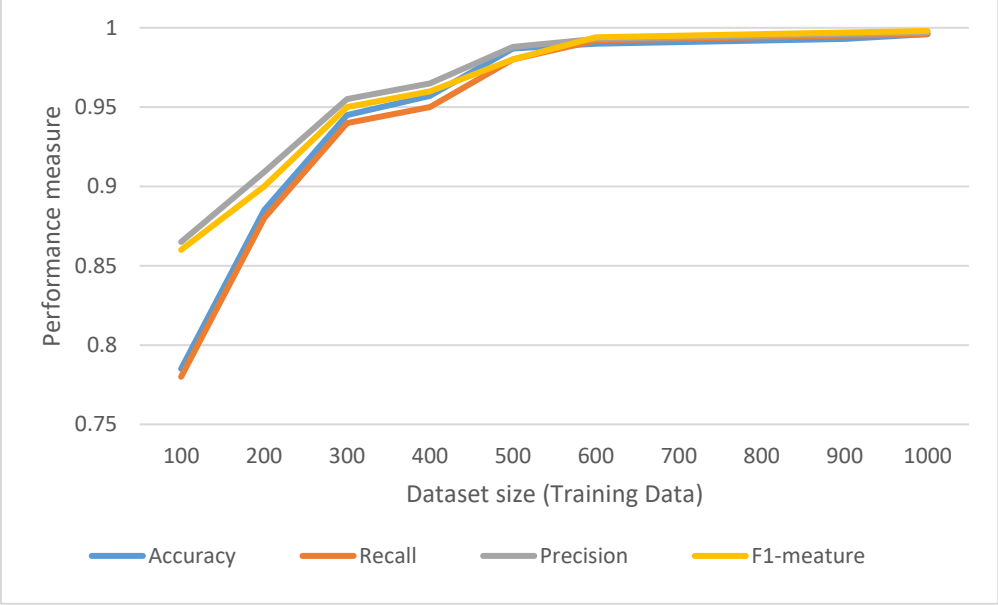

Figure (4.4): Evaluation Metrics

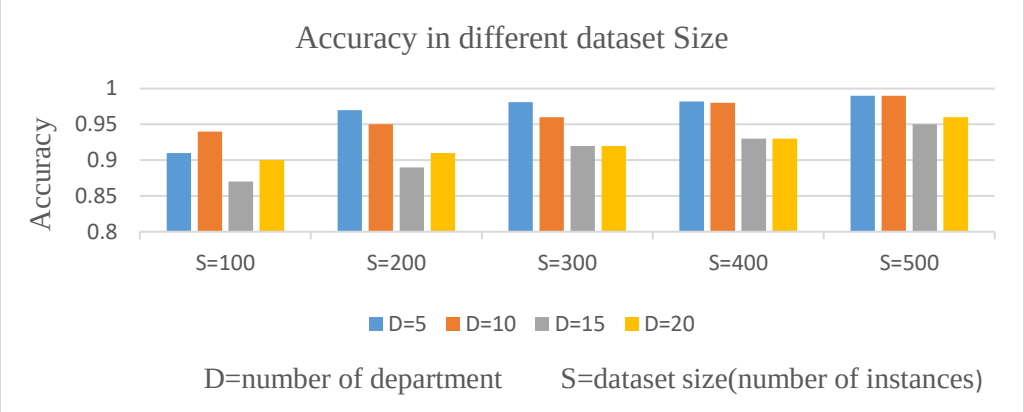

Figure (4.5): the Effect of the Dataset Size

One of the most challenging issues to deal with is how to build an accurate model trained on a partially observed environment. Table (4.8) provides a result that shows how the built model is accurate even if the observed features used were less than 50% of the overall features.

Table (4.8): the performance of the model under partially observed features

System Size	Percent of observed Features (events)	Accuracy
Dataset 1	31%	0.99
Dataset 2	30%	0.98
Dataset 3	27%	0.96
Dataset 4	26%	0.96
Dataset 5	26%	0.95

To evaluate the effectiveness of the rules included in the final rule set, Figure (4.6) shows how the quality of the classifiers increased through increasing the training dataset size.

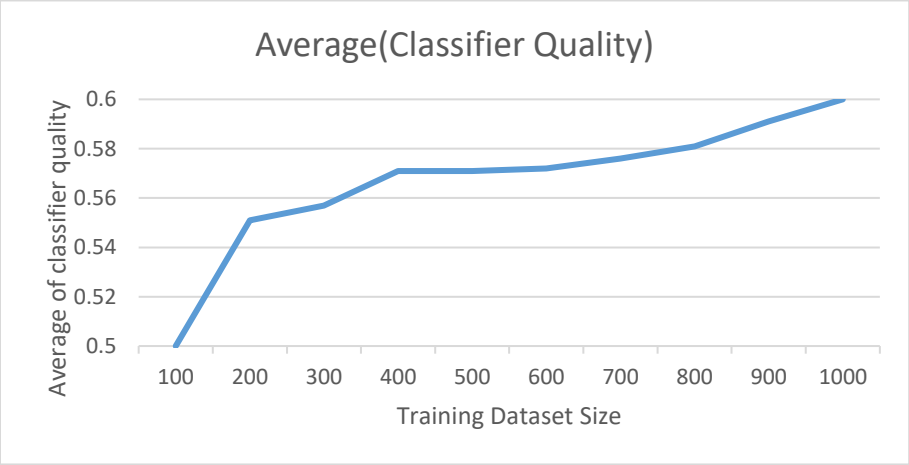

Figure (4.6): The Quality of Classifiers

Table (4.9) shows a sample of the rules included in the final rule set which considers the resulting model obtained from training dataset 2 using ACS2. The quality of the rules is a good metric to evaluate the performance of the rule. In addition to the reward prediction obtained through the training stage when a rule is presented. The symbols C, A, E, q, r, and num refers to condition part, action, effect part, quality, reward, and numerosity of a rule respectively.

Table (4.9): Sample of The Final compact Rule Set Details

C							A	E							q	r	num
#	#	#	#	#	#	#	1	#	#	#	#	#	#	#	0.99	2867.6	17
#	#	#	1	#	#	#	1	#	#	#	2	#	#	#	0.98	3589.3	15
#	#	#	#	1	#	#	1	#	#	#	#	2	#	#	0.98	3694.9	16
#	1	#	#	0	#	#	0	#	1	#	#	#	#	#	0.98	735.06	9
#	0	#	#	#	#	#	0	#	1	#	#	#	#	#	0.96	696.70	8

4.8 Performance Measurement (Online Stage)

The main aim of this thesis is not only to propose a model that can detect SLA violations in a partial observation environment but also have the ability to give a timely decision whether a violation occurs or not. Therefore to evaluate the performance of implementing the model online, compute time complexity is required. Table (4.10) shows the actual time required to predict a class label for the unseen sequence in detail with respect to the model size. As shown from Table (4.10) the time required for detection and the space memory required for saving the proposed model is very little compared to time and space required by a method of model-based system which is used by Al-Ajeli and Parker [6].

Table (4.10): The actual Time Required for Detection of a Violation

Problem Space	Model Size (number of resulting rules)	Time required in seconds
Dataset 1, (D=3)	40	43
Dataset 2, (D=5)	41	46
Dataset 3, (D=10)	44	57
Dataset 4, (D=15)	46	64
Dataset 5, (D=20)	47	116

Chapter Five

Conclusion and Future Work

Chapter Five Conclusion and Future Work-

5.1 Conclusion

Detection of SLA violations is desirable and beneficial for both customers and service providers. In this work, an LCS-based approach is proposed to address the problem of detecting SLA violations. In particular, the ACS2 is adopted to learn how to anticipate the next state of the system under partial observation where not all events can be seen. Four main steps have been used to implement the proposed approach. Starting with generating a dataset by simulating a Petri net model of a telecommunication company as a case study. Then, features projection and over-sampling techniques are applied to handle the simulated dataset. Next, this dataset is presented to train the model by applying the ACS2 model. Finally, the rule compaction step takes the training model and produces a reduced compact set of rules applied then online for the detection of SLA violations. The five types of datasets were generated with respect to problem space have been used to demonstrate the performance of the proposed approach. The experimental results show a high level of accuracy and the produced model is computationally efficient.

The most important characteristics that have been discovered through the implementation of the proposed methodology and the discussion of its results are as follows:

1. The problem of detecting SLA violations under a partially observable environment is very complex. It is very hard to give the ultimate decision when some of the features are unobserved. For this reason, an anticipatory learning system in particular ACS2 has been presented to deal with this problem. The ACS2 has shown a high ability to deal with the partially observable environment.

2. Violation is a very rare event, where the number of violated instances is only 20% of the overall dataset. Therefore over-sampling techniques, in particular, SMOTH is applied. SMOTH has the ability to balance the dataset by increasing the number of violated (minority) instances while saving the sequence order of the dataset.
3. Whatever the rate of the observed features was very low, the model accuracy is very high arriving at 98%.
4. Different types of the dataset generated by simulating a model with different sizes are used to train the model, which gives a good indicator of the flexibility of the proposed model.
5. The resulting model is an optimal rule-based system represented by a small number of rules and includes all rules that are more general and accurate.
6. The resulting model can give a decision in a short time due to its small size, where the time required for the detection of SLA violation of unseen instances is only a few seconds or one minute at most.

5.2 Future Works

Some future directions can be highlighted here:

1. Consider modeling time at which an event occurs within the system being analyzed. Taking into account the time factor in making a decision makes the problem more complex and a big challenge for most researchers.
2. Exploring different application domains. For instance, healthcare systems and computer networks can be counted as the following targets to be investigated for more examples on using the SLA constraints.

3. Working on the prevention of the violation instead of only detecting it to increase customer satisfaction as well as the reputation of the service providers by controlling the service that is presented.
4. Extend the proposed approach to handle a dataset that is mainly generated based on multiple constraints instead of one.
5. Proposed a methodology that can deal with imbalanced datasets without the need to treat it using sampling methods.

References

[1] M. Hammer, “What is business process management?,” in *Handbook on business process management 1*, Springer, 2015, pp. 3–16.

[2] P. Cheng, “An Approach of QoS Evaluation for Web Services Design with Optimized Avoidance of SLA Violations,” 2020.

[3] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendy, and Y.-C. Tian, “Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing,” *IEEE Access*, vol. 6, pp. 55923–55936, 2018.

[4] A. Al-Ajeli and D. Parker, “Online monitoring for diagnosis of violations of constraints in Petri net models,” *IFAC-PapersOnLine*, vol. 52, no. 11, pp. 25–30, 2019.

[5] A. Al-Ajeli and B. Bordbar, “Fourier-motzkin method for failure diagnosis in petri net models of discrete event systems,” in *2016 13th International Workshop on Discrete Event Systems (WODES)*, 2016, pp. 165–170.

[6] A. Al-Ajeli and D. Parker, “Online fault diagnosis in Petri net models of discrete-event systems using Fourier-Motzkin,” in *2018 UKACC 12th International Conference on Control (CONTROL)*, 2018, pp. 397–402.

[7] S. Agarwal, “Scholarship at UWindsor An Approach of SLA Violation Prediction and QoS Optimization using Regression Machine Learning Techniques An Approach of SLA Violation Prediction and QoS Optimization using Regression Machine Learning Techniques By,” 2020.

[8] X. Zeng *et al.*, “Detection of SLA violation for big data analytics applications

in cloud," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 746–758, 2020.

[9] R. A. Hemmat and A. Hafid, "SLA violation prediction in cloud computing: A machine learning perspective," *arXiv Prepr. arXiv1611.10338*, 2016.

[10] P. K. Upadhyay, A. Pandita, and N. Joshi, "Scaled conjugate gradient backpropagation based sla violation prediction in cloud computing," in *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2019, pp. 203–208.

[11] O. Jules, A. Hafid, and M. A. Serhani, "Bayesian network, and probabilistic ontology driven trust model for sla management of cloud services," in *2014 IEEE 3rd international conference on cloud networking (CloudNet)*, 2014, pp. 77–83.

[12] A. Karamanlioglu and F. N. Alpaslan, "An ontology-based expert system to detect service level agreement violations," in *International Symposium on Business Modeling and Software Design*, 2018, pp. 362–371.

[13] C. Schubert, M. Borkowski, and S. Schulte, "Trustworthy Detection and Arbitration of SLA Violations in the Cloud," in *European Conference on Service-Oriented and Cloud Computing*, 2018, pp. 90–104.

[14] P. Trkman, "The critical success factors of business process management," *Int. J. Inf. Manage.*, vol. 30, no. 2, pp. 125–134, 2010.

[15] C. Houy, P. Fettke, and P. Loos, "Empirical research in business process management–analysis of an emerging field of research," *Bus. Process Manag. J.*, 2010.

[16] S. Adesola and T. Baines, "Developing and evaluating a methodology for business process improvement," *Bus. Process Manag. J.*, 2005.

[17] M. Laguna and J. Marklund, *Business process modeling, simulation and design*. Chapman and Hall/CRC, 2018.

[18] Q. He, J. Yan, R. Kowalczyk, H. Jin, and Y. Yang, "Lifetime service level agreement management with autonomous agents for services provision," *Inf. Sci. (Ny).*, vol. 179, no. 15, pp. 2591–2605, 2009.

[19] G. Chen and K. Hu, "Notice of Retraction: Design Service Level Agreement Based on Dynamic IT," in *2009 Pacific-Asia Conference on Knowledge Engineering and Software Engineering*, 2009, pp. 21–24.

[20] A. Keller and H. Ludwig, "The WSLA framework: Specifying and monitoring service level agreements for web services," *J. Netw. Syst. Manag.*, vol. 11, no. 1, pp. 57–81, 2003.

[21] S. Zhang and M. Song, "An architecture design of life cycle based SLA management," in *2010 The 12th International Conference on Advanced Communication Technology (ICACT)*, 2010, vol. 2, pp. 1351–1355.

[22] K. Terplan, "3.6 Telecommunications Support Processes," *CRC Handb. Mod. Telecommun.*, vol. 3, p. 120, 2010.

[23] M. R. Raza and A. Varol, "QoS parameters for viable SLA in cloud," in *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, 2020, pp. 1–5.

[24] M. Tadao, "Petri nets: properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, 1990.

[25] M. Alodib and B. Bordbar, "A model-based approach to fault diagnosis in service oriented architectures," in *2009 seventh IEEE European conference on web services*, 2009, pp. 129–138.

[26] A. Jović, K. Brkić, and N. Bogunović, “A review of feature selection methods with applications,” in *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, 2015, pp. 1200–1205.

[27] Z. Gong and H. Chen, “Model-based oversampling for imbalanced sequence classification,” in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 1009–1018.

[28] H. D. Nguyen, X.-S. Vu, Q.-T. Truong, and D.-T. Le, “Reinforced data sampling for model diversification,” *arXiv Prepr. arXiv2006.07100*, 2020.

[29] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, “A comparative study of data sampling and cost sensitive learning,” in *2008 IEEE International Conference on Data Mining Workshops*, 2008, pp. 46–52.

[30] R. Choudhary and H. K. Gianey, “Comprehensive review on supervised machine learning algorithms,” in *2017 International Conference on Machine Learning and Data Science (MLDS)*, 2017, pp. 37–43.

[31] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, “A survey of machine learning for big data processing,” *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, pp. 1–16, 2016.

[32] R. J. Urbanowicz and W. N. Browne, *Introduction to learning classifier systems*. Springer, 2017.

[33] M. V. Butz, *Anticipatory Learning Classifier Systems*, vol. 4. Boston, MA: Springer US, 2002.

[34] M. V Butz and W. Stolzmann, “An Algorithmic Description of ACS2.”

[35] “Lecture Notes in Artificial Intelligence 2661 Subseries of Lecture Notes in

Computer Science."

[36] J. Tan, J. Moore, and R. Urbanowicz, "Rapid Rule Compaction Strategies for Global Knowledge Discovery in a Supervised Learning Classifier System," Aug. 2013, pp. 110–117, doi: 10.7551/978-0-262-31709-2-ch017.

[37] K. Borna, S. Hoseini, and M. A. M. Aghaei, "Customer satisfaction prediction with Michigan-style learning classifier system," *SN Appl. Sci.*, vol. 1, no. 11, Nov. 2019, doi: 10.1007/s42452-019-1493-1.

[38] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, Springer New York, 2016, pp. 1–7.

[39] "P A N G. N I N G T A N."

[40] H. M and S. M.N, "A Review on Evaluation Metrics for Data Classification Evaluations," *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, Mar. 2015, doi: 10.5121/ijdkp.2015.5201.

[41] P. Agarwal and M. Alam, "A Lightweight Deep Learning Model for Human Activity Recognition on Edge Devices," in *Procedia Computer Science*, 2020, vol. 167, pp. 2364–2373, doi: 10.1016/j.procs.2020.03.289.

Appendix A

Formal Acceptance

2nd International Conference of Information Technology to enhance E-learning and other Application-2021

[IT-ELA 2021]

To/ Hawraa Abdulameer Subeh, Ahmed Al-Ajeli

Dear respected author(s):

With heartiest congratulations. We are pleased to inform you that based on the recommendations of the reviewers and the Technical Committees, your paper entitled:

"A Learning Classifier system for detection of Service-Level Agreement violations in business process"

It has been accepted for oral presentation at the 2nd International Conference of Information Technology to enhance E-learning and other application-2021[IT-ELA 2021], technically sponsored by IEEE, to be held on Dec 28-29, 2021, Baghdad, Iraq.

Your paper will be submitted (**within the conference proceeding**) to the IEEE Xplore digital library for (**final acceptance of uploading to the Digital library**).

IT-ELA 2021 Baghdad – Iraq

Email:it-ela2021@baghdadcollege.edu.iq

Website: https://baghdadcollege.edu.iq/it-ela2021

IT-ELA2021

Baghdad College of Economic Sciences University

Computer Sciences Department

IEEE

ID: 1570769732

الخلاصة

أتفاقية مستوى الخدمة (SLA) هي عقد مبرم بين مقدمي الخدمة والعملاء. يتضمن هذا العقد بعض القيود التي تهدف الى الحفاظ على مستوى معين من جودة الخدمة. يمكن أعتبار أنتهاك قيود أتفاقية مستوى الخدمة (SLA) بمثابة خطأ وهذا يعني فشل ضمان الخدمة. قد يؤدي ذلك الى نتائج غير مرغوب بها مثل دفع الغرامات الماليه وتقليل نسبه الربح وتشويه السمعه وأنقطاع الخدمة. تقترح هذه الرسالة أسلوب التعلم الألي القائم على نظام التعليم المصنف (LCS) لأكتشاف مثل هذا الانتهاك. نظراً لأن الاحداث في النظام يمكن ملاحضتها جزئياً ، هذا يجعل المشكله الأكثر أهتمام لمعالجتها. تم أعتماد نظام التصنيف التوقعي (ACS) الذي يعتبر نموذج من نظام (LCS) للتعامل مع هذهِ البيانات الجزئية. نظام (ACS) نموذج قائم على القواعد التي تستخدم بعد ذلك لكشف الانتهاك عند مراقبة سلسله من الأحداث. البيانات المستخمة لتدريب النموذج تم توليدها من خلال محاكاة نظام تابع لشركة أتصالات كنموذج للأختبار. مجموعة البيانات التي تم انشاؤها غير متوازنة، لذلك يتم تطبيقة أخذ العينات الزائدة للتغلب على مشكلة التحيز للفئة المسيطرة. بالاظافة الى ذلك، تم تطبيق تقنية ضغط القواعد للحصول على مجموعه قواعد مضغوطة يتم أستخدامها بعد ذلك في مرحلة الكشف . أضهرت النتائج التجربية أداءاً جيداً للنموذج المقترح في تحقيق دقه عالية تصل الى 99% ، بالأضافه الى ذلك الحصول على نموذج يعطي قراراً في الوقت المناسب بشأن حدوث الأنتهاك او لا.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جـامعـــة بـابـــل
كلية تكنولوجيا المعلومات ـ قسم البرمجيات

كشف أنتهاك أتفاقيه مستوى الخدمة في مجال عمليات العمل من خلال تعليم نظام التصنيف

رسالة مقدمة إلى
مجلس كلية تكنولوجيا المعلومات ـ جامعة بابل كجزء من متطلبات
نيل درجة الماجستير في تكنولوجيا المعلومات / البرمجيات

من قبل

حوراء عبد الأمير صبح محمد

بإشـراف

د. أحمد خلفه عبيد حمزة

1443-1442 هـ
2022-2201 م