7-7-2020

An Approach of SLA Violation Prediction and QoS Optimization using Regression Machine Learning Techniques

Saurav Agarwal
University of Windsor

An Approach of SLA Violation Prediction and QoS Optimization using Regression Machine Learning Techniques

By

Saurav Agarwal

A Thesis

Submitted to the Faculty of Graduate Studies

through Computer Science

In Partial Fulfillment of the Requirements

for the Degree of Master of Science

at the University of Windsor

Windsor, Ontario, Canada

2020

An Approach of SLA Violation Prediction and QoS Optimization using Regression Machine Learning Techniques

by

Saurav Agarwal

APPROVED BY:

_____

H. Wu

Department of Electrical and Computer Engineering

_____

J. Lu

School of Computer Science

_____

X. Yuan, Advisor

School of Computer Science

May 08th, 2020

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights. Any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Along with the acceptance of Service-Oriented Architecture (SOA) as a promising style of software design, the role that Quality of Service (QoS) plays in the success of SOA-based software systems has become much more significant than ever before. When QoS is documented as a Service-Level Agreement (SLA), it specifies the commitment between a service provider and a client, as well as monetary penalties in case of any SLA violations. To avoid and reduce the situations that may cause SLA violations, service providers need tools to intuitively analyze if their service design provokes SLA violations and to automatically guide them preventing SLA violations. Due to the dynamic nature of service interaction during the operation of SOA-based software systems, the avoidance of SLA violations requires prompt detection of potential violations before prevention takes place at real-time. To overcome the low latency time in practice, this thesis research develops an approach of using Machine Learning techniques to not only predict SLA violations but also prevent them by means of optimization. This research discusses the algorithm and framework, along with the results of the experiments, which will help to examine its usefulness for service providers working on the construction and refinement of services.

DEDICATION

Dedicated to God, my grandparents, my beloved mummy and papa without whose support, I would not have made it this far, my loving brother, my supportive sister-in-law, my girlfriend, my adorable nephew, and the rest of my family and friends.

ACKNOWLEDGEMENTS

First and foremost, I would like to express profound thankfulness to my supervisor, Dr. Xiaobu Yuan, who has supported me throughout my thesis with his knowledge and expertise on this exciting field of research. His ideas and suggestions have helped me become more creative, without which I would not have been able to complete this research.

I would like to offer my sincere gratitude to the advisory group members, Dr. Jianguo Lu and Dr. Huapeng Wu for their significant remarks and recommendations for my research.

I would like to thank all my friends especially Rajasi and Anjali, who have supported and helped me throughout my studies, here in Canada. I also thank my brother and sister-in-law for their motivation and financial support which enabled me to complete my studies successfully.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS/SYMBOLS

IT	Information Technology
SOA	Service Oriented Architecture
QoS	Quality of Service
SLA	Service Level Agreement
ML	Machine Learning
CPU	Central Processing Unit
SLO	Service Level Objectives
WS	Web Server
DB	Database
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
SVM	Support Vector Machine
NN	Neural Networks
LR	Linear Regression

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Over the last four decades, software architectures have attempted to deal with increasing levels of software complexity. As the level of complexity continues to evolve, traditional architectures do not seem to be capable of dealing with the current problems. Service-Oriented Architecture (SOA) is being advocated in the industry as the next evolutionary step in software architecture to aid IT organizations meet their complex set of challenges that traditional architectures cannot meet. A service-oriented architecture is essentially a collection of services, among which communication can involve either simple data transfer or could involve two or more services coordinating some activity, thereby requiring a means of connecting other services to each other.

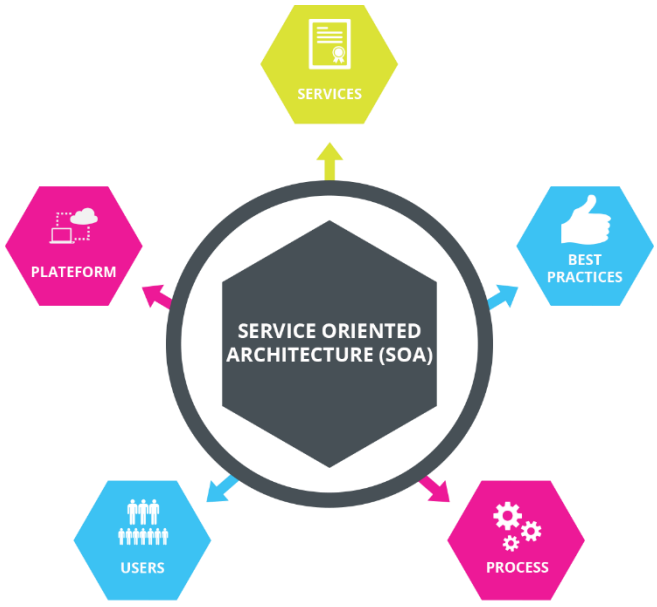


Figure 1.1: Service Oriented Architecture

In recent years, Web services have become the technology of choice for realizing Service-Oriented Architecture and its associated set of strategic goals [1]. A service is a function that is well defined, self-contained, and does not fully depend on the context or state of other services [10]. A Web Service is a software system that is designed to support interoperable machine-to-machine interaction through a network. The superiority of SOA comes from the fact that it promises the highly desired benefits of improved reusability, increased reliability, and reduced costs for development and deployment in a scalable and dynamic environment. In a service-level agreement (SLA), QoS is documented to guarantee that services fulfill their official commitments in terms of both functionality and quality [3].

"*Quality of service (QoS) represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application*" [4]. In order to guarantee a basic level of QoS, careful management of IT resources is essential. Management of assets and taking care of variable volumes of client necessities are a piece of SLA between consumer and service provider. However, obeying SLA has been proven to be a challenging task as QoS is influenced by a variety of different factors. Because of variations in workload, computing resources, and even network conditions, it is common for Web services to exhibit fluctuation in performance, leading to the possibility of violating SLA [6].

QoS management includes assisting users to find the essential characteristics of the wanted service and adaptation of IT assets in such a manner that it considers SLA and enhances the system performance and efficiency. In a situation that the effective QoS doesn't conform to the base QoS concurred in SLA, the QoS manager gives a case of SLA violation. Any service provider who does not take action to prevent SLA violations will have to face monetary penalties and lost revenue due to a damaged relationship with clients [10]. Therefore, service providers are in desperate need of such tools that can not only help them in predicting if their service design provokes SLA violation, but also guides them in optimizing service design to prevent the violation from happening.

SLA violation forecast benefits both service providers and clients. From a service provider's point of view, SLA violation results in paying fines in terms of both, reputations as well as money. By foreseeing the violations ahead of time, providers can re-allocate the needs and resources to avoid future violations [8]. All the process of resource allocation is done behind the scene in this manner; thus, from a client's perspective, better resource allocation brings about a reliable supplier. Additionally, customers would like to receive the service on-demand and with no interference. Thus, a system wherein a service supplier or a third party could give the expectation of SLA infringement for the client can be exceptionally canny.

It is worth mentioning that violations do happen in the real world. For instance, Amazon Elastic Cloud confronted a blackout in 2011 when it crashed, and numerous clients, for example, Reddit and Quora, were down for more than one day [5].

1.2 Motivation and Problem Statement

SOA's exceptional capacity in software development has driven ongoing work within the academic community and software industry explorations. To improve the design of Web services, encouraging progress has been achieved for the evaluation of QoS, prediction of SLA violations, and QoS optimization. For example, most of the approaches use the technique of collaborative filtering to evaluate SOA systems with historical datasets obtained from other systems running a similar application and to predict the possibility of SLA violations based upon the calculation of QoS values for existing services [48, 49, 50]. For QoS optimization, some approaches work for groups of services by maximizing aggregated utility values and others for individual services by allocating more resources [28, 29]. However, current methods still lack objective evaluation of software operating in different application domains, and their separation of related processes has resulted in limited success in practice.

A couple of recent publications proposed an innovative framework that tackles the main issues of SLA violation by combining three techniques into a unified process to analyze, predict, and prevent SLA violation [1] [2]. However, a manual process is used in their

proposed framework for an SLA prediction, which is time-consuming. It also uses a basic fitted response model, which cannot tackle complex real-world data. Furthermore, for optimization, only one controllable factor i.e. cache level is used, which will not lead to the optimization of complex systems. SLA violation prevention must be performed in real-time to detect violations quickly and hence, avoid them. However, there is a very low latency time to avoid. Thus, to tackle such a scenario, an automated technique is required.

In this thesis, we propose to utilize Machine Learning to predict and prevent SLA violations into unified framework to help service providers analyze, predict, and prevent SLA violations. Machine Learning techniques can be used to automate their proposed framework, which will eliminate many manual steps. Violation prediction and avoidance can be viewed as a regression problem in the terminology of Machine Learning. Past research mostly relies on heuristic methods for prediction of violations. Even though Machine Learning (ML) has been utilized in different territories of QoS management, the experiments done for the most part are in a confined setting, which isn't ascendable to real-world data. Nonetheless, this research adopts a systematic machine learning approach applied on real-world data that provides an insightful set of experiments. This automated process will greatly enhance other requirements in terms of availability, performance, robustness, response time, and cost.

1.3 Thesis Contributions

Major contributions of this research work can be summarized as follows:

- To construct and train the system using Machine Learning techniques in a way that enables them to not only predict the SLA violations but also prevent them by optimizing the service into unified framework.
- To predict the response time of an incoming for detecting SLA violation.
- To achieve the required response time and to prevent a violation by using a multi-control optimization technique.
- To conduct a series of experiments for verifying that the proposed ML model can achieve a robust SLA violation detection and prevention efficiency, get a satisfying

performance and reduce the monetary cost.

1.4 Organization of this thesis

In this thesis, Chapter 2 introduces the main concepts of Service Oriented Architecture and discusses the important role that service level agreements play in quality of services. Chapter 3 clarifies the terminologies and basic concepts in machine learning in which various machine learning models such as regression and classification are discussed. It also presents the method of measuring the performance of a model in machine learning. Chapter 4 presents an outline of existing contributions on SLA violation prediction; specifically, it will introduce the confinements of these contributions and how our proposed model aims to overcome them. Chapter 5 displays the proposed method that is utilized to predict and prevent SLA violations in unified framework. Chapter 6 exhibits the details of the assessment and the execution of our proposal. Finally, Chapter 7 concludes the thesis and presents future work.

CHAPTER 2

Service Oriented Architecture

Service Oriented Architecture (SOA) is being advocated in the industry as the next evolutionary step in software architecture to aid IT organizations meet their complex set of challenges. A service-oriented architecture is essentially a collection of services, among which communication can involve either simple data transfer or could involve two or more services coordinating some activity, thereby requiring a means of connecting other services.

Figure 2.1: Service Oriented Architecture

SOA offers the much-needed benefits like easier component reuse, increased productivity, improved reliability, shorter time-to-market, and reduced deployment costs [3]. It is one of the most successful architectural styles, in which applications make use of reusable services via internet. In the next decade, the SOA principles will be at the core of a new era of business engagements that transact at Internet scale across locations, devices, people,

processes and information [6].

The principles of service-orientation are independent of any product, vendor or technology. SOA just makes it easier for software components over various networks to work with each other.

There are two major roles within Service-oriented Architecture:

1. **Service provider:** The service provider is the maintainer of the service and the organization that makes one or more services available for others to use. To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.
2. **Service consumer:** The service consumer has the ability to consume (use) the SOA through a program or an individual, who requests a service [3].

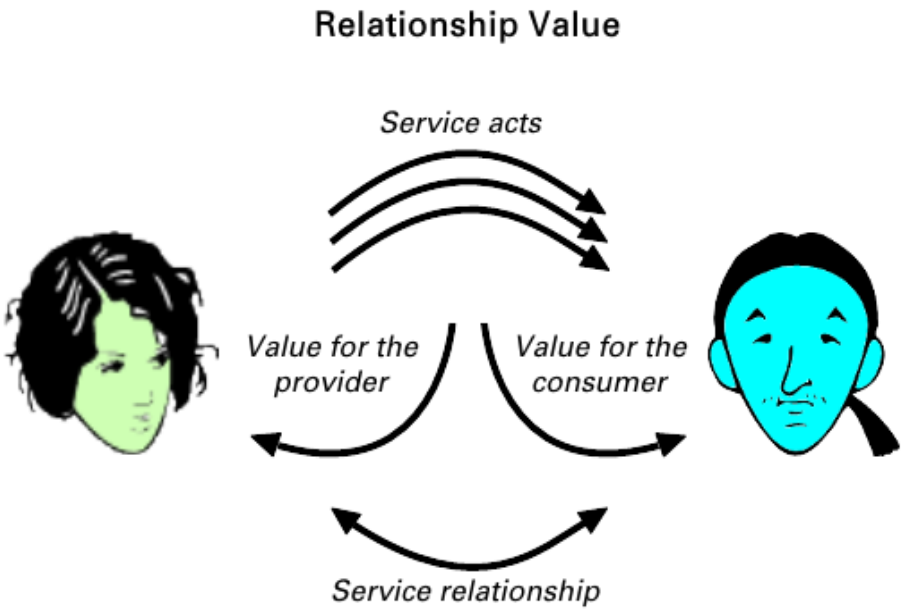


Figure 2.2: Relation between Service Provider and Service Consumer

Each service may be offered by various providers and can be used by one or more customers. A service consumer can be a service or application that reuses other services.

As per the SOA frameworks that service consumers are developing, they pick the most reasonable administrations from various applicants with comparable functionality and use them to make their application. A service provider, on the other hand, can be an individual or an association that creates and keeps reusable services. These services are accessible for service consumers to reuse. Competition is incredibly fierce between different service providers as there are others available consistently with the similar highlights. If a service is not fit for fulfilling service consumers in terms of quality and usefulness, service customers may surrender this service and pick another service provider [8].

2.1 Web Service

A service is a function that is well defined, self-contained and does not fully depend on the context or state of other services [10]. A Web Service is a software system that is designed to support interoperable machine-to-machine interaction through a network. The technology of Web services is most likely the go-to connection option of service-oriented architecture. Web services offer a potential solution for developing distributed business processes and applications that can be accessed via the Internet. The use of Web services in SOA systems have many benefits for the development of new applications [8]. It also has the benefits of lower cost, higher reliability, and lesser time to market for further development of new applications. There are four primary tasks in web services: publish, discovery, request, and response. Publish is a process by which a service provider announces its service as well as the service associated interfaces. Generally, a service provider announces its service by entering service information into a specialized registry [6]. The consumers of the services discover the services in various ways. Discovery is a process of finding an appropriate service that provides the required functionality. Upon discovery, the consumer requests the functionality by providing the required input. The service responds to the consumer with the desired output.

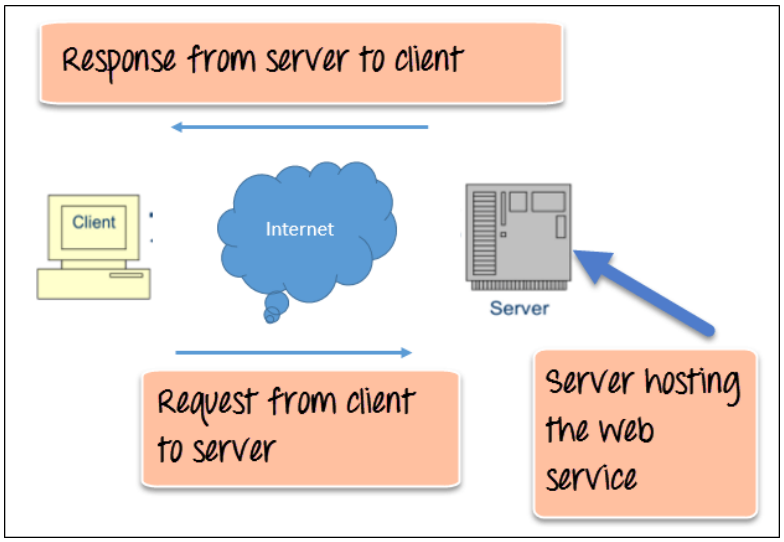


Figure 2.3: Web Service Architecture Diagram

The above diagram shows a very simplistic view of how a web service would work. The client invokes a series of web service calls via requests to a server that would host the actual web service.

Web services play three major roles in an SOA system:

- Service Directory
- Service Provider
- Service Consumer

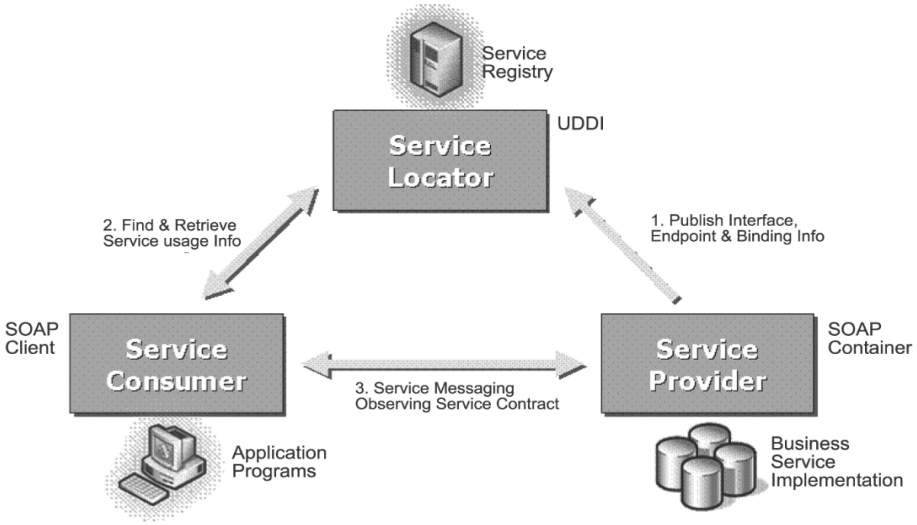


Figure 2.4: Three Web Service Roles

With service directory acting as a centralized directory, service providers publish information of Web services for service consumers to select according to their preferences of software development. For example, a Web Service can help consumers to identify the top 10 webpage links for different e-commerce websites that offer the lowest price for a certain product [1].

2.2 Quality of Service (QoS)

"*Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application*" [4]. QoS plays an important role in service selection in an SOA environment. It specifies how a component is supposed to behave. Through QoS, consumers can select an SOA service provider based on the quality of service. As more competitive web services have become available for the consumers, QoS has become a decisive factor to distinguish the reputation of various service providers [11]. By estimating the QoS of a system, the performance can be enhanced and guaranteed ahead of time. Subsequently, QoS measurement expands the dependability and accessibility of the system. In SOA systems, QoS is a fundamental viewpoint, as service consumers' needs to have a measure of the service performance and a service provider needs to find the best exchange off between the provided service and the expense.

QoS manager distributes different measures of resources (CPU, memory, or storage) and further decides the agreements in SLA based on four sources of information: (1) The accessible resources of the computing system, (2) The requested IT resources for each user task, (3) Information about the least possible QoS agreed in SLA, and (4) The historical information about the system's load. QoS manager, usually using a heuristic method, decides how to prevent SLA violation. For example, in the application of video streaming such as YouTube, the QoS manager may delay the video by a couple of moments to buffer and prevent interruption in the middle of the video. On the other hand, in some other applications such as video conference of Google Hangouts, in which significant delay is

not tolerable, QoS manager may diminish the resolution of the video or the sound quality to avoid any violation of the service [13]. Hence, it is necessary to be able to forecast when an SLA violation may happen beforehand.

At the infrastructure level of computing, several QoS parameters can be measured as stated below [14]:

- **Compute**: outage length, availability, server reboot time
- **Network:** packet loss, availability, latency, mean/max jitter, bandwidth
- **Storage:** input/output per second, availability, processing time, max restore time, latency with internal compute resource.

Service providers guarantee the QoS with Service Level Agreements (SLAs). We discuss the definition of SLA and SLA management life cycle in the following sections.

2.3 Service Level Agreements

The association between a service provider and a customer is governed with a Service Level Agreement (SLA). SLA is negotiated between parties and a level of the service, QoS and its related expenses are agreed upon. SLA is an official document of QoS which contains specific parameters and a minimum level of quality of service. It is mutually agreed between a service provider and prospective consumers. This is a part of the contract and is an assurance to the service consumers that they will get the services that they pay for, by obligating service providers to fulfill contractual promises [15]. Service Level Objectives (SLOs) are a key element of SLA, which are the qualitative parameters of an SLA that includes availability, throughput, and response time. SLA clearly defines monetary penalties in case of any violation of the written agreement. Hence, service providers have a strong interest in keeping their commitments to avoid and reduce the situations that may cause SLA violation.

Any SLA mainly describes two things:

- Different Service Level Objectives (SLOs) in terms of values for Quality of Service metrics.

- The penalties to be applied if the objectives have not been accomplished [5]

From an application facilitating perspective, SLA has two different types: Application SLA and Infrastructure SLA. Infrastructure SLA ensures a level of consistency on infrastructures such as power, data center, latency and so forth by dedicating resources exclusively to the customer. An example is shown in Table 2.2. Application SLA is suitable for hosting models on which numerous applications are co-located. In such a setting, service resources are available to applications according to the application demands. Thus, in application SLA, service providers guarantee meeting application demands. An example of application SLA is shown in Table 2.3.

For instance, SLA can demonstrate 99.99 % accessibility for requests of disk, CPU, and memory. An SLA might also contain constraints on the response time for each request.

SLA is a significant piece of each agreement because a provider would like to allocate the minimal amount of resources for each customer to reduce the expense of its server infrastructure. Simultaneously, the provider needs to avoid having penalties due to the failure of providing the contracted service. The failure of providing a service is called an SLA violation. The client would like to receive the service on request and with no interference. Regardless of these high accessibility rates, infringement does occur in a genuine world and has caused both the provider and the client's substantial expenses [18].

Table 2.1 – Components of a Web Service Level Agreement [16]

Service-Level Parameter	Describes a noticeable property of a service whose value is measurable
Metrics	Measures to assess, compare or track performances
Availability and uptime	The duration and frequency for which the services provided must be available to the customer. Uptime percentage is usually measured and reported monthly.
Performance standards	Specific benchmarks that are determined by the client. Actual vendor service-level performance is measured against these values to ensure the performance standards have been met
Response time	Defines the minimum and maximum amount of time allotted to the service provider for responding to a request or issue
Resolution time	States the minimum and maximum amount of time that a vendor is given to resolve a particular task or issue

Table 2.2 – An example of infrastructure SLA [16]

Availability of Hardware	99 % uptime in a month
Availability of Power	99.99 % of the time in a month
Availability of data center network	99.99 % of the time in a month
Availability of Backbone network	99.99 % of the time in a month
Credit for Service unavailability	Refund of service credit for downtime period
Blackout notification guarantee	Notification to customers within 1 hour of downtime
Internet latency guarantee	When latency is measured at 5-min intervals to an upstream provider, the average doesn't exceed 60 msec.
Packet loss guarantee	Shall not exceed 1 % in a calendar month.

Table 2.3 – An example of application SLA [16]

Service-level parameter metric	Website response time (e.g., max of 3.5 sec per user request).
Function	Latency of web server (WS) (e.g., max of 0.2 sec per request). Latency of DB (e.g., max of 0.5 sec per query) Average latency of WS = (latency of web server 1 + latency of web server 2) /2 Website response time = Average latency of web server + latency of database
Measurement directive	DB latency available via http://mgmtserver/em/latency. WS latency available via http://mgmtserver/ws/instanceno/latency
Penalty	Website latency < 1 sec when concurrent connection < 1000. 1000 USD for every minute while the SLO was breached.
Credit for Service unavailability	Refund of service credit for downtime period

2.3.1 SLA Management Life Cycle

Each SLA goes through a sequence of steps starting from identification of terms and conditions, activation and monitoring of the stated terms and conditions, and eventual termination of the contract once the hosting relationship ceases to exist. Such a sequence of steps is called the SLA life cycle.

According to [17], it consists of the following six phases:

- Discover Service Provider
- SLA Contract Definition
- Establish Agreement

- SLA Monitoring
- SLA Violation Detection
- SLA Enforcement

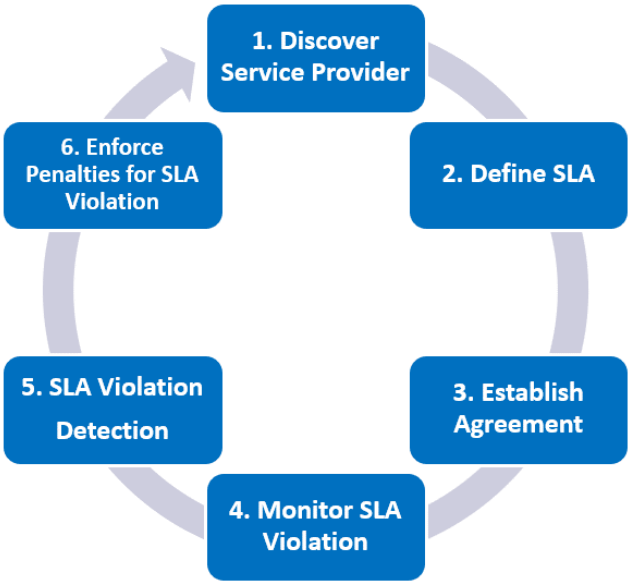


Figure 2.6: SLA Management Life Cycle

Discover Service Provider

In this period, the service provider publicizes these base service contributions through standard publication media, and the customers should be able to locate the service provider by searching the catalog. The customers can look through different competitive offerings and choose a few that fulfill their pre-requisites for further negotiation.

SLA Contract Definition

In this section, the service and its equivalent price, QoS parameters with a fundamental schema and the penalty rule is defined. SLAs are commonly defined using standard/base formats or by customization of these base layouts.

Establish Agreement

In this stage, a customer finds a service provider that meets the customer's needs. The terms and conditions of the SLA are negotiated and settled upon. A service provider needs

to evaluate the SLA in terms of scalability, availability, and performance of its services to avoid fines before approving the specification of SLA. By the completion of this phase, parties start to commit to the agreement.

Monitor SLA violation

In this part, the provider's presentation in delivery of the service is estimated against the agreement. A crucial part of SLA monitoring is to be able to envisage violations, assisting providers to reallocate the resources accordingly before the violations happen.

SLA Violation Detection

In this stage, the factors inside SLA are estimated and any deviation is determined. In the case of SLA violation, SLA enforcement is conducted.

SLA Enforcement

This segment is to implement penalties for SLA infringement. In this period, suitable actions are taken when the violation has been identified in the earlier phase. The concerning parties are notified and penalty charges are taken. After SLA implementation, SLA may end due to break or violation.

2.3.2 Cause of SLA Violations

Failure of service providers to render an agreed service as described in an SLA is called an SLA violation. Due to variation in workload, computing resources, and network conditions, it is common for Web services to exhibit fluctuation in performance, leading to the possibility of violation of an SLA [2]. SLA assurance is a critical objective for every provider, as violation will lead to heavy penalties for the provider, in terms of money and reputation [2]. In terms of availability, when Amazon Elastic Cloud crashed in 2011, it faced an outage and many big customers such as Quora and Reddit were down for more than a day. Such crashes affect service providers and service consumers. Predicting the occurrence of an SLA violation has become an important research topic. This subject can be viewed either from the perspective of the service consumer or from that of the service provider.

CHAPTER 3

PREDICTION MODELS

Machine learning is the research and formation of programs and algorithms that can study from historical data and make a prediction when exposed to new data. There are three common types of algorithms used in machine learning to solve different problems: supervised learning algorithms, unsupervised learning algorithms, and reinforcement learning [19].

- **Supervised Learning** intents to find a function, mapping the input to the output given to the labeled dataset.
- **Unsupervised Learning** aims to recognize structures and trends within an unlabeled dataset provided input.
- **Reinforcement Learning** targets at discovering a role that generates a sequence of acts that optimizes costs or rewards.

The focus of this thesis is on supervised learning. Consequently, supervised learning is applied in more profundity after an investigation of certain terminologies of machine learning. First, primary concepts such as Generalization, Bias-Variance Trade Off, and Cross Validation are addressed in machine learning. Finally, we will discuss how a model is evaluated in machine learning and specifically discuss Root Mean Square Error (RMSE), Mean Absolute Error (MAE), R^2 and Adjusted R^2.

3.1 Terminology

In this segment, we introduce the basic machine learning terminology that is utilized in the rest of this chapter. A dataset is given in a set of rows and columns in a typical supervised machine learning task. Each dataset row corresponds to one single data point, which is called an example of training or an instance of training. Input variables, functions,

or attributes are called columns. Each data point has at least one or more label(s), targets, or output variables linked with each other.

The dataset is characteristically split into two sets: training set and test set. The training set is utilized to learn the underlying variance factors in the data, while the test set is used for the final assessment. To start with, given the training set, the model is trained, and, during testing, the model is provided with an example described by its features, and the output is the expected label.

3.2 Supervised Machine Learning: Concepts and Definitions

Two pieces of information are given to the algorithm in supervised machine learning: a set of input instances $X = \{x_1, x_2, ..., x_m\}$ and a relating set of targets $Y = \{y_1, y_2, ..., y_m\}$. Classically, each of these m input instances contains a set of n features $x = \{x_1, x_2, ..., x_n\}$. Generally speaking, every x_i function can take any value, either numerical (values are real numbers) or categorical (values are unordered set members). Nevertheless, features may be expected to be converted to certain forms depending on the task at hand.

There is constantly a true function $f^*(\cdot)$, which maps each conceivable x to the most ideal y. In any case, we never have access to this unknown function. Supervised learning, therefore, amounts to approximating function $f^*(\cdot)$ based on the information provided in the sets of X and Y. The process of approximating $f^*(\cdot)$ using a function $f_\theta(\cdot)$ in which θ is a set of parameters is called learning.

Learning algorithms become familiar with the parameters θ of the function $f_\theta(\cdot)$ by limiting the errors that the model makes. Formally, a function that maps the discrepancy between the output prediction of the model and the true target into a real number is called the loss function [19].

If the true target y is a discrete variable, the prediction task is called Classification. On the other hand, if y is continuous, the task is called Regression. In the accompanying subsections, we discuss these two types of supervised learning algorithms in more detail after formally presenting learning.

3.2.1 Learning

Approximating function $f^*(\cdot)$ using function $f_\theta(\cdot)$ corresponds to extracting the underlying factors of variation from data instances and mapping them to the output. These underlying factors could be a probability table, a graph structure, or weights depending on which learning algorithm is utilized to find the data. Generally, learning adds up to finding the best parameters θ to minimize a loss function over all the examples in the dataset [19]. Therefore, the learning process can be formulated as follows,

$$\hat{\boldsymbol{\theta}} = argmin_{\theta}\{\sum_{i=1}^{m} l(y_i, o_i;\ \theta)\}, \tag{3.1}$$

in which $\hat{\boldsymbol{\theta}}$ is the learned set of parameters, y_i and o_i are the target and output of the model for the i^{th} sample.

3.2.2 Classification

In a supervised classification task, the prediction output y is from one of the total C distinct classes {1, 2, ..., C}. To get a forecast for new examples, the model can simply output a class label, or the output can be a set of probabilities. Each probability corresponds to one of C classes that indicates how probable it is that the unseen input x belongs to a specific class. In models that output probabilities, to get a discrete prediction out of the model, either the class with the highest probability is chosen or the class label is drawn by sampling from the output distribution.

3.2.3 Regression

Similar to a classification task, in regression problems, the objective is to learn a mapping function from an n-dimensional vector x into a real-valued number O as the prediction. A regression model uses the historical relationship between an independent and a dependent variable to predict the future values of the dependent variable. Mathematically, regression is about learning a model $f(y) = f(x) + \varepsilon$, where **ε** is a noise/error term that describes everything that cannot be captured by the model.

A simple regression model shows the relationship between the magnitude of one variable and that of a second - for example, as X increases, Y also increases or as X increases, Y decreases. It estimates exactly how much *Y* will change when *X* changes by a certain amount.

3.3 Generalization

The goal of machine learning is to train models that can predict the labels for new examples that are not seen previously. Consequently, generalization to new examples is an important aspect of every learning algorithm. Usually, we are looking for models that perform well on testing data as well as on training data. As a consequence, we must prevent learning algorithms from merely memorizing training data; instead, such algorithms must learn the underlying variation factors.

3.3.1 Bias-Variance Trade off

To decide how reliable a model is, we must comprehend the reasons behind errors. Bias and variance of a prediction model help us formally measure these errors. Bias and variance of a prediction model allow us to compute these errors formally. To define bias and variance over a model, we must assume that we can train the same model multiple times with different randomly selected data points. In this thesis, each trained model is called a model instance. Errors in bias and variance predictions are called errors due to bias and error due to variance respectively [20].

Bias corresponds to the distance between the expected prediction of the model and the true target [22]. Considering f(x) as the model, the bias is defined as follows:

$$\text{bias} = | E[f(x)] - y |^2, \qquad \textbf{(3.2)}$$

where E[.] is the expectation and y is the true target. On the other hand, variance corresponds to the variability in different predictions of multiple instances of a model [22]:

$$\text{variance} = | f(x) - E[f(x)] |^2 \qquad \textbf{(3.3)}$$

The total error of a model in terms of bias and variance is defined as follows:

$$\text{error} = E[(f(x) - y)^2] = \text{bias}^2 + \text{variance}. \quad \textbf{(3.4)}$$

Given the limited amount of data, there is always a trade-off between bias and variance. The trade-off happens in a way that reducing one may lead to increasing the other. As a result, minimizing the total error requires a careful balance between bias and variance. A graphical illustration of this trade-off is shown in Figure 3.1.

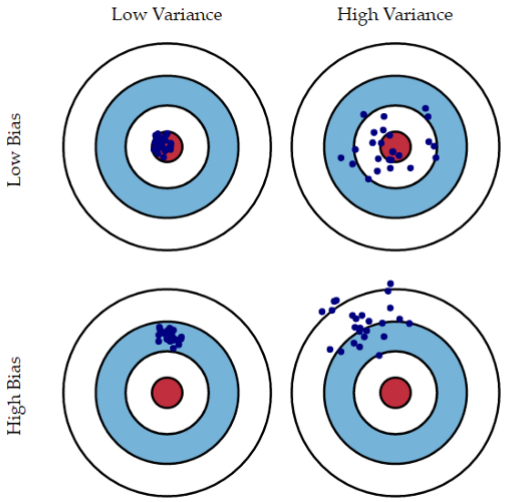


Figure 3.1 - Dart chart: A graphical illustration of bias-variance trade-off

3.3.2 Cross Validation

To find the parameters of the model that generalize the best, we need to know if the model has been overfit. Cross validation helps us to find an overfit model. Overfitting happens when the error rate in the training set decreases but the error on the test set increases. As shown in Figure 3.2, as we increase the complexity of the model, the error rate in the training set decreases but at some point, the error in the test set passes the minimum and increases. When the error in the test set increases with higher model complexity, the model is overfit. In cross validation, the dataset is divided into training and validation sets. To increase the validity of the model, *k-fold cross validation* is used where

the dataset is partitioned into k equal subsets. We define *d* as the complexity order of the model. For each order-d hypothesis class:

— Repeat k times:

— Set aside one of the subsets.

— Use the rest of the data points to find θ (model parameters).

— Compute prediction error on the held-out subset.

— Average the prediction error over the k rounds/folds. Use this as the estimated true prediction error for the order-d hypothesis class [19].

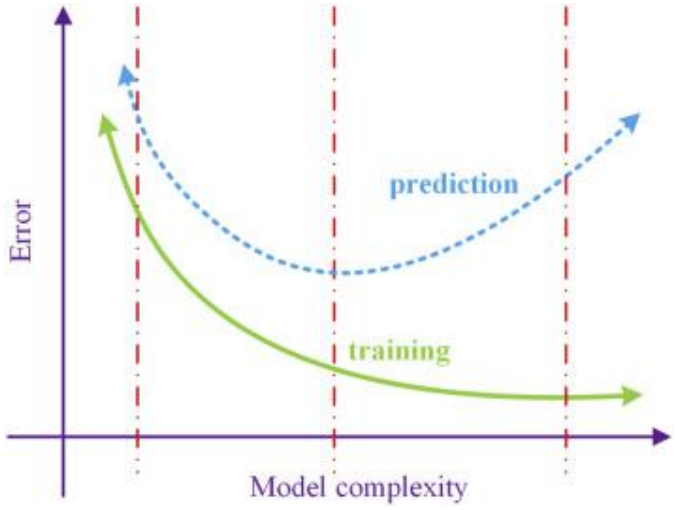


Figure 3.2 – Test & training error as the function of model complexity.

The goal is to find *d* with the lowest estimated true prediction error. It is worth mentioning that k-fold cross validation increases computation k-times. Thus, with larger datasets or complex models, a smaller value of k is preferred [19].

3.4 Performance Evaluation

In this subsection, we introduce the common error metrics used for evaluating a regression model: (1) Root Mean Square Error, (2) Mean Absolute Error, (3) R^2, and (4) Adjusted R^2 [23]. Error metrics help us indicate how good the model will perform when exposed to unseen data. Thus, after the model is trained on the training set and the best performing model is chosen, it will be tested on an intact test set. This approach helps us

select a model that will have a good performance on unseen data.

3.4.1 Root Mean Square Error (RMSE)

The root-mean-squared error (RMSE) is a measure of how well our model performed. It does this by measuring the difference between predicted values and the actual values. It is the standard deviation of the residuals (prediction errors).

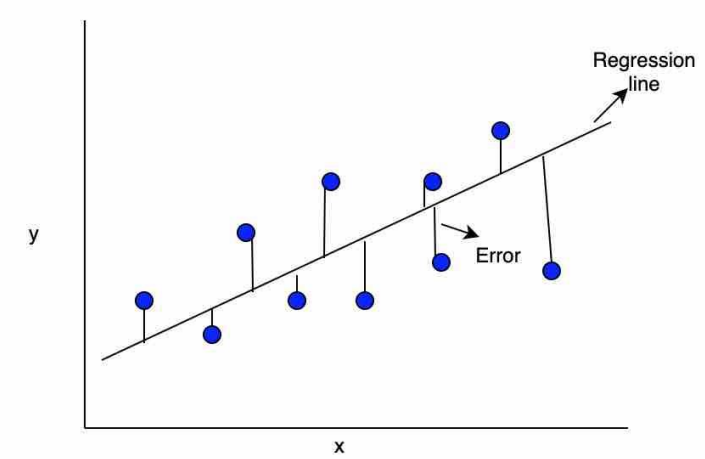


Figure 3.3: RMSE – Predicted vs Observed values

Residuals are a measure of how far from the regression line data points are. RMSE is a measure of how spread out these residuals are. It shows how concentrated the data is around the line where it fits best. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}} \quad \textbf{(3.5)}$$

3.4.2 Mean Absolute Error (MAE)

MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \qquad \textbf{(3.6)}$$

It is a measure of the difference between two continuous variables. Assume *X* and *Y* are variables of paired observations that express the same phenomenon. Examples of *Y* versus *X* include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. Consider a scatter plot of *n* points, where point *i* has coordinates (x_i, y_i). Mean Absolute Error (MAE) is the average vertical distance between each point and the identity line. MAE is also the average horizontal distance between each point and the identity line.

3.4.3 R^2

It is a statistical measure that represents the usefulness or fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better the model will be fitted. R-square is a comparison of the residual sum of squares ($SS_{residuals}$) with a total sum of squares (SS_{total}). A total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line [19].

Sum Squared Regression Error

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} \qquad \textbf{(3.7)}$$

Sum Squared Total Error

It is a statistical measure that represents the proportion of the variance for a dependent variable that is explained by an independent variable or variables in a regression model. Whereas, correlation explains the strength of the relationship between an independent and dependent variable, R-squared explains to what extent the variance of one variable explains the variance of the second variable. So, if the R^2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs.

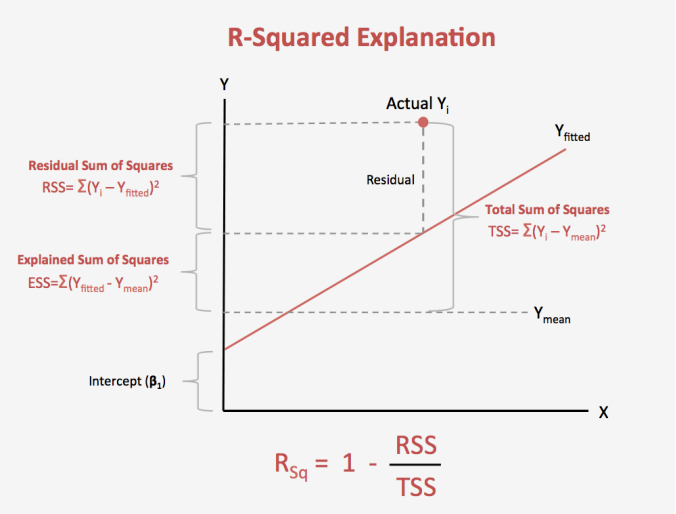


Figure 3.4: R-Squared Explanation

3.4.3 Adjusted R^2

It is a modified version of R-squared that has been adjusted for the number of predictors in the model. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance and if not understood. It only decreases if a predictor improves the model and the amount that is less expected by chance. It is always lower than the R-squared [10].

Adjusted R-squared measures the variation in the dependent variable (or target), explained by only the features which help make predictions. Unlike R-squared, the Adjusted R-squared would penalize for adding features that are not useful for predicting the target.

Let us mathematically understand how this feature is accommodated in Adjusted R-Squared. Here is the formula for adjusted r-squared:

$$R^2 = 1 - \frac{RSS}{TSS} \quad \textbf{(3.8)}$$

$$R^2_{adj} = 1 - ((1 - R^2)\frac{N-1}{N-M-1} \quad \textbf{(3.9)}$$

Here R^2 is the r-squared calculated, N is the number of rows and M is the number of columns. As the number of features increases, the value in the denominator decreases.

CHAPTER 4

Related Works

This chapter discusses the relevant background of recent works in SLA violation prediction and prevention.

4.1 SLA Violation Prediction

SLA violation prediction is an essential task in web service as an SLA violation might cause interruptions for the clients' accessibility of service and force penalties on the supplier. An assortment of contributions has been proposed for SLA violation prediction. For SLA violation prediction from the perspective of the service provider, several approaches have been proposed in recent years. Publications surveyed in this direction, with highlights of their contributions and limitations, are listed below.

Rafael et al. [40] proposed a technique which focuses on anticipating the demand of the future resources for meeting SLA requirement. The author used business-level SLAs (throughput and response time) as input parameters to the chosen prediction approaches. Machine Learning techniques like Support Vector Machine (SVM), Neural Networks (NN), and Linear Regression (LR) were used for prediction. However, no real-world data was considered or was probed on, for simulating a realistic scenario.

Authors in Jules et al. [39] use an intelligent and dynamic Service Level Agreement (SLA) based on a probabilistic ontology that detects and alerts potential violations of contract parameters for a cloud computing environment. Despite its good performance, the dataset generated using simulation does not necessarily represent a real environment. It contains 40% violations and dismisses the way that in a genuine world, infringement is extremely uncommon (∼2.0%).

In a similar work for predicting SLA violations in composite services, in Leitner et al.

[24], propose a regression machine learning model; the regression model is implemented using the WEKA framework, which cannot be scaled to real-world environments. In [25], the authors introduce an efficient system that predicts SLA violation before it occurs and recommends how to mitigate those violations to avoid any penalties. A profile-based model of SLA violation prediction from the provider's perspective was proposed. It helps service providers in making decisions about whether to form SLA and avoiding SLA violations.

To achieve service level agreements, a prediction method based on Bayes model was designed by Zhang et al. [26] to predict the mean load over a long-term time interval as well as the mean load in consecutive future time intervals by identifying novel predictive features of host load that capture the expectation, predictability, trends and patterns of host load. This prediction model of the workload can help a service provider estimate the possibility of whether SLA violation will occur.

Wong et al. [27] proposed to use the SVM model to predict possible SLA violations before any issue emerges so that remedial action can be taken. While the approaches in [26] and [27] can help a service provider know beforehand whether SLA violation will take place, it lacks the capability of helping service providers evaluate QoS quantitatively.

Recently, Cheng et al. [1] [2] have proposed a framework to utilize the sensitivity analysis for the identification of influential factors with dominating impacts on QoS. They used metamodel-based analysis to select a fitted surrogate model for domain-independent prediction of SLA violation. The residual error between predicted and validated response time are calculated to select the best-fitted model for prediction. In the suggested method, the process of evaluation can be used by service consumers for service selection, and it can be used by service providers to study SLA violations [1]. However, in the proposed framework for SLA prediction, a manual process is employed, which is time-consuming and cannot tackle complex real-world data.

4.2 SLA Violation Prevention

The subject of SLA violation prevention can be either composite services or individual services. For individual services, resource provision is employed to analyze workloads, to classify them based on common patterns, and to plan for workloads before actual scheduling. For composite services, several approaches suggest calculating aggregated QoS

values of all possible service combinations and choosing the one that maximizes the aggregated utility value while satisfying global constraints. Many different approaches have been proposed for SLA violation prevention.

For individual services, resource provision has been utilized in [28] to analyze workloads, to categorize them on the basis of common patterns, and to plan for workloads before actual scheduling. The authors of [28] later enhanced their work with automated processing in [29]. Although these methods can optimize QoS by allocating more resources, no attention has been given to resource over-provision, which is a serious issue for Web services as it wastes resources and causes an increase in operational cost [30].

Wu et al. [31] proposed *ProfminVMminAvaiSpace*, an algorithm that maps users' requirements into infrastructure resources to provide a reliable service, and at the same time, maximize resource allocation to prevent violations.

Uriarte et al. [32] used unsupervised learning to cluster the resource usage and duration of services to avoid violations of the Google Cluster trace dataset. If a violation happens inside a cluster of services, the other services inside the cluster will be assigned to other resources, to avoid the violation. This helps in violation avoidance in the cluster, but there is no explicit prediction of SLA violation for each service.

Chana et al. [28] proposed an approach where they enhanced their work with automated processing. Although these methods can optimize QoS by allocating more resources, no attention has been given to resource over-provision, which is a serious issue for Web services, since resources are wasted, and operational cost becomes exorbitant.

Cheng et al. [2] presented a new approach for QoS optimization to improve the quality of Web services. In this approach, four procedures were performed: identifying influential factors, collecting observed data, fitting collected data with the MARS model, and identifying global optimum till SLA is prevented. However, for optimization, only one controllable factor (cache level) is used, which will not lead to optimization for a complex system. So, there is a need to automate the proposed approach using Machine Learning techniques, which can easily adapt to any complex system and overcome the limitation of these approaches. The automated process will greatly enhance other requirements in terms

of availability, performance, robustness, response time, and cost.

As a result, this thesis will develop a novel approach to help a service provider intuitively analyze if their service design provokes SLA violation. If it does, then this approach will guide them for optimizing the service to prevent violation because SLA violation will lay a hefty penalty to the provider. As part of the proposed unified framework, this approach uses Machine Learning models to construct and train the system in a way that enables them to not only predict the violations but also prevent them by optimizing the service.

4.3 Related Work

SLA Violation Prediction		
Paper	**Contribution**	**Limitations**
Hemmat et al. (2016), "SLA Violation Prediction In Cloud Computing: A Machine Learning Perspective" [33]	- In this paper, two machine learning classification models: Naive Bayes and Random Forest classifiers, were used to predict SLA violations. - Several re-sampling methods such as Random Over and Under Sampling, SMOTE, NearMiss (1,2,3), One-sided Selection were used to re-balance the dataset. - Two classes- violated and unviolated were generated for the prediction problem. - Accuracy, ROC area, Precision, Recall, and F value were used for model performance evaluation.	- No continuous prediction was done, as only task violation is predicted using the classification technique. - Basic classification techniques were applied to complex data to predict violation.

Zhu et al. (2017), "Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization" [34]	- This paper proposed a collaborative QoS prediction approach, namely adaptive matrix factorization (AMF) - AMF method has been presented and evaluated on a real-world QoS dataset of Web services. - Median Relative Error (MRE) and Ninety-Percentile Relative Error (NPRE) were used to evaluate AMF approach.	- Due to incurred service invocations, this approach is costly, especially when applied to a large number of candidate services. - Evaluations are conducted offline, which is not desirable on real large-scale applications.

SLA Violation Prevention		
Paper	**Contribution**	**Limitations**
Khan et al. (2016), "An Adaptive Monitoring Framework for Ensuring Accountability and Quality of Services in Cloud Computing" [35]	- In this paper, a framework to dynamically monitor QoS metrics and performance measures to verify compliances for respective SLAs is proposed. - Three main components namely, a component for the digitization of SLA parameters; interactive components for dynamic monitoring of QoS	- No new technique is proposed for QoS optimization. - This method optimizes QoS by allocating more resources but no attention has been given to resource over-provision.

	and core component for dynamic detection of violation and adaptive remedy rectification are advised.	
Zhou et al. (2018), "Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms" [36]	- This paper proposes two novel adaptive energy-aware algorithms for maximizing energy efficiency and minimizing the SLA violation rate in Cloud datacenters. - Application types, as well as the CPU and memory resources, are also considered during the deployment of VMs. - Experimental analysis is performed using a real-world workload, which comes from more than a thousand PlanetLab VMs.	- Predicts future resource requirements, and arranges appropriate additional virtual resources in advance, before the peak occurs. This is done to maximize performance and avoid service violations, but it always wastes resources and causes an increase in operational cost.

Model Construction		
Paper	**Contribution**	**Applicability**
Joseph et al. (2015), "Construction and Use of Linear Regression Models for Processor Performance Analysis" [37]	- The paper proposes an iterative process for constructing accurate regression models of processor consisting of all significant main effects and interaction terms using a reasonable number of simulations. - It constructs and obtains accurate estimates of all significant coefficients with the minimum number of simulations. - Regression model construction steps include: Obtaining the Best Model, Determining Model Adequacy and an interactive procedure to obtain linear models at any specified level of accuracy.	- This iterative method can be used for constructing regression models to predict SLA Violations. - It can be extended to various applications like: - Sales forecasting - Weather forecasting - House Price Prediction

Adhikari et al. (2013), "An Introductory Study on Time Series Modeling and Forecasting," [38]	- This paper proposes how a time series regression model is constructed and applied in a different real-world domain. - To evaluate forecast accuracy as well as to compare among different models fitted to a time series, five performance measures are used: MSE, MAD, RMSE, MAPE and Theil's U-statistics. - Applied Time Series forecasting on the following domains: Stochastic models, Artificial Neural Networks, Support Vector Machines	A combined Time Series regression model can be used for SLA prediction and prevention. - It can be extended to various applications like: -Economic Forecasting - Budgetary Analysis - Stock Market Analysis - Yield Projections - Process and Quality Control - Inventory Studies

CHAPTER 5

PROPOSED METHODOLOGY

The idea of this thesis is to automate the unified framework for the prediction and prevention of SLA developed by Cheng et al. [1] [2]. To automate the existing process, the proposed approach uses various machine learning models and techniques. Prior projects working on SLA violation prediction or avoidance have generally neglected the challenges of using real-world data. In the proposed framework, a real-world dataset is used to construct machine learning models for efficient analysis of the unseen data. SLA violation prediction and prevention can be simply considered as a regression problem.

Data analysis, SLA violation prediction, and design optimization are three crucial, yet typically separate techniques for QoS evaluation and optimization [52]. As illustrated in Figure 5.1, this thesis proposes an innovative framework to solve the main issues of SLA violation by automating these three techniques into a unified framework, which analyzes, predicts, and prevents SLA violations.

The process of data analysis in this framework initiates the construction of a single dataset table through data selection and preprocessing, which is responsible for cleaning and preparing data to address the machine learning problem. With the number of features reduced to only those influential to the service's performance, the second process fits a machine learning model to predict SLA violation. The development of a model starts with a candidate in a specific type and form, such as a regression model, from a pool of choices. The process of SLA violation prediction then checks the adequacy of this chosen candidate with new experiments. Unless a rejection results in the choice of another candidate, the model is ready to be used for quantitative analysis of service quality and for the prediction of situations when SLA violations could take place.

The last process of QoS optimization is necessary to avoid the predicted SLA violations by adjusting the values of controllable factors by service providers, such as CPU, storage,

and cache. This process includes three steps as below:

- Identifying the most influential controllable factors
- Fitting new data with a machine learning regression model
- Conducting experiments with single/multiple factors and collecting observational data

This three-step procedure is either repeated until SLA prevention is achieved, or the list of controllable factors is exhausted. In the latter case, a re-design of the web service by the provider is a practical recommendation.

The rest of this chapter is organized as follows. Section 5.1 first discusses data analysis, which helps to understand its characteristics, features, and class distribution; this facilitates the discovery of models that can effectively make predictions with such features. Then, it discusses the approach of data selection and preprocessing. Section 5.2 presents the machine learning models that can efficiently address the regression task of SLA violation prediction on a skewed dataset. Finally, Section 5.3 presents our approach for preventing the predicted SLA violation using multi-factor optimization on a selected machine learning model.

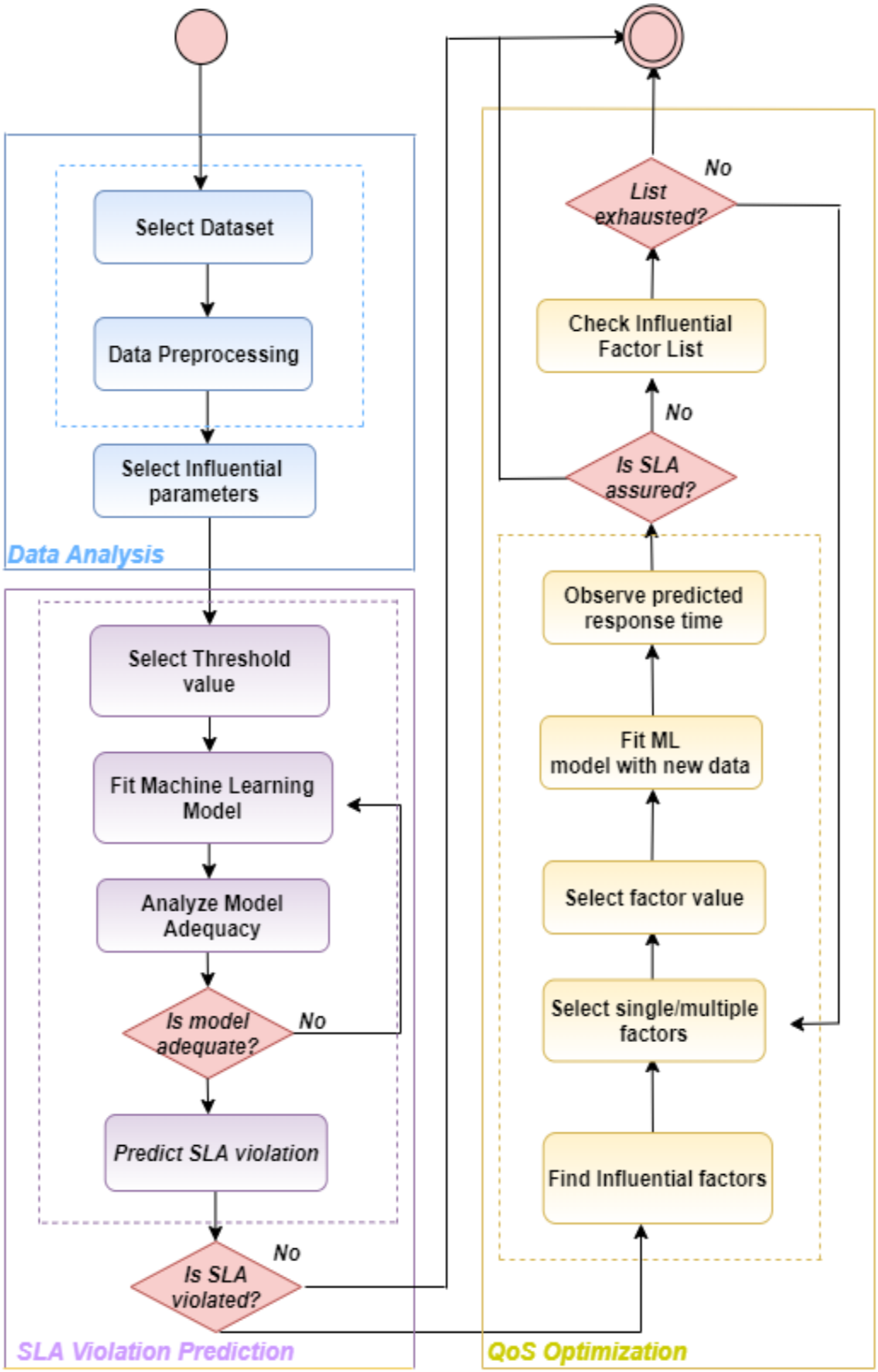


Figure 5.1: A Unified Framework of SLA Violation Prediction and Prevention

5.1 Data Analysis

Figure 5.2 illustrates the first module of the proposed framework for data analysis, with details presented in Algorithm 1. Since the availability rate is very high (97.8%) and violations are rare, machine learning models tend to predict the absence of violations. However, this is not desirable in a real-world scenario. Thus a few re-sampling techniques are applied in machine learning to handle the skewness of data.

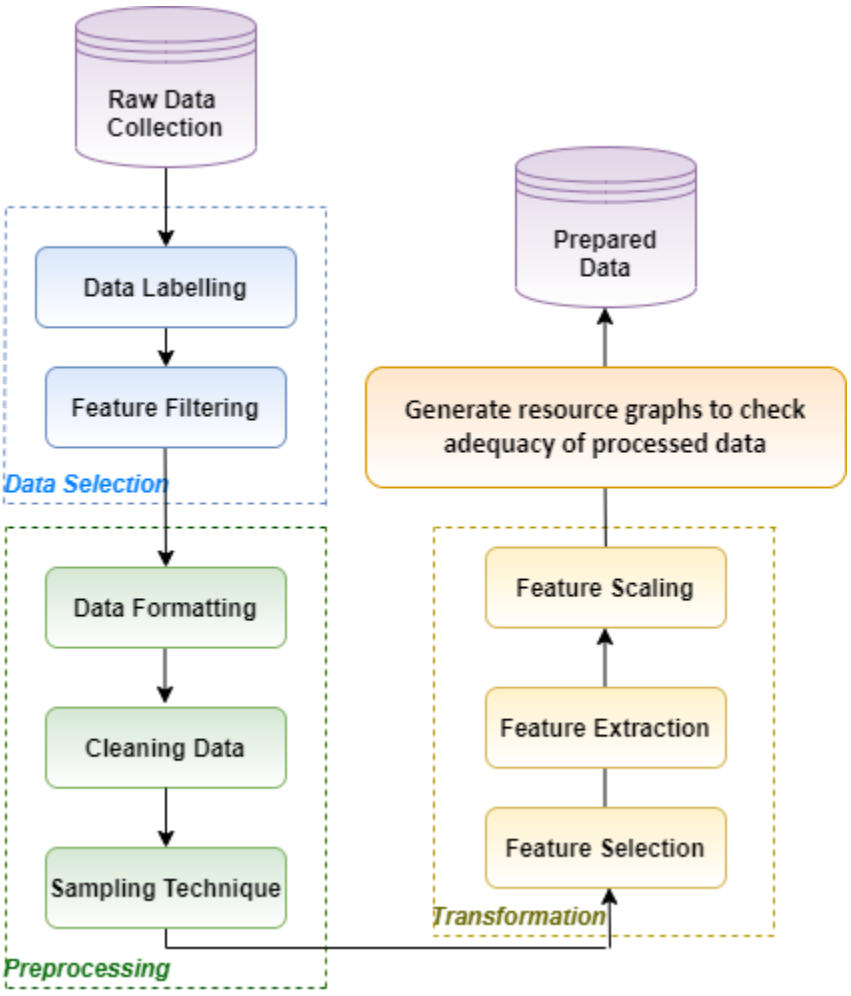


Figure 5.2 – Data Analysis

Algorithm 1: Data Selection and Preprocessing

```
Declare Raw Data tables as D
Step-1: Selected_Data():
foreach datatable in D do
    foreach column in datatable do
        if column is not labelled then
            columnlabel ← do labelling of the column;
        end
        if column is relavant then
            Selected_Data ← Add column to selected data;
        else
            remove column from data table
        end
    end
end
Step-2: Preprocessed_Data():
foreach column in Selected_Data do
    if column is NULL then
        remove column from Selected_Data
    end
    else if column is duplicate then
        remove column from Selected_Data
    end
end
foreach row in Selected_Data do
    if row is NULL then
        remove row from Selected_Data
    end
    else if row is duplicate then
        remove row from Selected_Data
    end
end
Cleaned_Data ← Selected_Data;
foreach column in Cleaned_Data do
    Formatted_Data ← add formatted consistent column;
    if Formatted_Data is not balanced then
        Preprocessed_Data ← Sampling_Technique(Formatted_Data);
    end
end
Step-3: Data_Transformation():
Selected_Features ← Select relevant and most useful features from
  Preprocessed_Data;
foreach job_id in Selected_Features do
    sum(all tasks(all rows) having job_id=job_id)
end
Extracted_Features ← Selected_Features;
Scaled_Features ← Normalize and rescale Extracted_Features between 0
  and 1;
Step-4: Generate_Resource_Graph():
Generate Different Analysis Graph(Scaled_Features)
Step-5: Check_Data_Graph():
if data is adequate then
    Prepared_Data ← Scaled_Features;
end
```

The process of data analysis is responsible for the selection and examination of raw data. Due to the fact that datasets always contain irrelevant information for SLA analysis, it is necessary to group raw data into labelled columns and keep only relevant columns by filtering out the others. If an original dataset uses spaces or commas to separate data, they need to be removed to avoid false prediction results. This results in a well-formatted table with useful information after Algorithm 1 finishes the first step of processing. However, datasets collected from real-world applications always contain null and duplicated data. Therefore, the proposed algorithm examines the table in the second step and remove the null or duplicated columns and rows. This cleanup applies to columns and rows that have unique values as they do not add any value to the prediction of SLA violations.

In order to make the dataset more useful and productive after preprocessing, data transformation is applied in the third step. Machine Learning is all about data, and the training success of any model depends on the way data is transformed and fed into the machine learning algorithm. Feature transformation is a function that transforms features from one representation to another.

There are several reasons for transforming the features:

- Data types are not suitable to be fed into a machine learning algorithm, e.g., text, categories.
- Feature values may cause problems during the learning process, e.g., data represented on different scales.
- Reduce the number of features to plot and visualize data, speed up training or improve the accuracy of a specific model.

In the proposed approach for transformation, normalization technique is applied to rescale data from the original range to a new range between 0 and 1. Normalization is only applied to the data when the data has input values with differing scales. The required features taken from the data set through the normalization process can vary. Since predicting the response time of the incoming request is needed in this case, the information of resources used by the user is gathered.

After preprocessing, a single dataset is generated to use it further in the SLA process of violation prediction and prevention. In the fourth step of the algorithm, several graphs are generated. The adequacy and compatibility of processed data are checked in the fifth step so that features such as the amount of requested CPU, disk, and memory of violated tasks and available resources at the time of request can be studied to predict future violations.

5.2 SLA Violation Prediction

In the process of data analysis, the performance model is treated as a BlackBox, as the features are identified by analyzing only the outputs of selective inputs. As the prediction of SLA violation is expected to work in all use cases of the service, the proposed framework fits various machine learning models and predicts service performance by evaluating these fitted models. As illustrated in Figure 5.3, the overall process goes through the following activities whose details are presented in Algorithm 2.

The processed data is used as input in the Fitted Machine Learning module. Initially, based on the data analysis result, a threshold value is selected for the maximum response time (line 3). If the response time is close or above the threshold, it implies that SLA is violated. First, the processed data is split into two parts: training and testing dataset. Two-third of the dataset is used as the training set and fed to the model in each training, while the remaining one-third is applied as the test set (line 7). The aggregate of the model is considered for reporting as the final result.

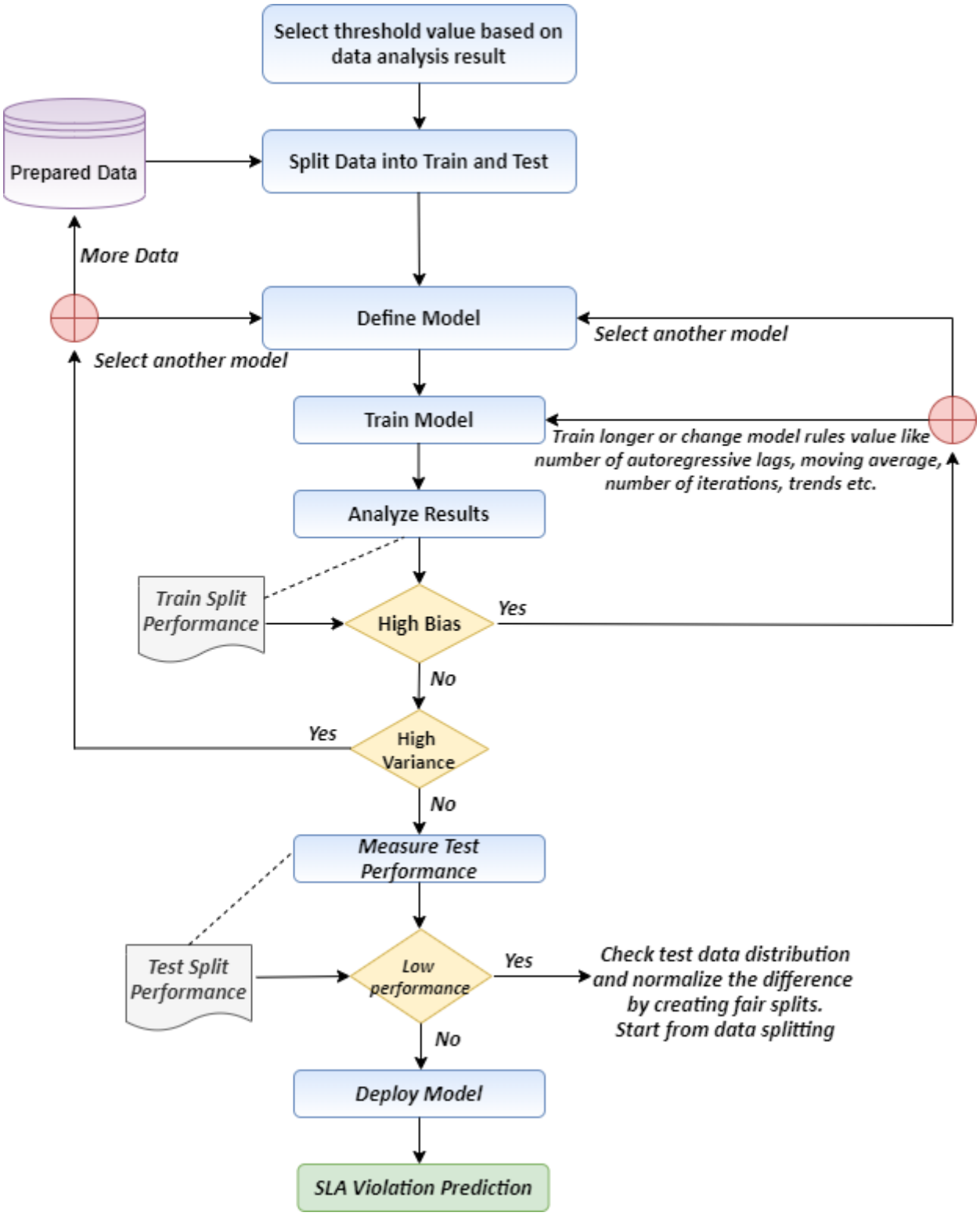


Figure 5.3 – SLA Violation Prediction

Algorithm 2: SLA Violation Prediction

```
1  Input: Thresold value
2  Output: SLA Violation predictions
3  Threshold_val ← Select Max(response time) as threshold value
4  Pdata ← Prepared data from output of data selection and
     preprocessing
5  ind_params ← Select independent features
6  dpd_params ← Select dependent features
7  Train_data, Test_data ← Split_train_test [dpd_params, ind_params,
     split_ratio ← 3:1]
8  Define/Select a regression model to train
9  Trained_model ← train the regression model with the Train_data
10 Analyzed_Result ← analyze the train model by reviewing "Train Split
     Performance"
11 if Analyzed_Result has HighBias then
12 |  Train the regression model longer or change the parameter values
   |    [goto step 9]
13 |  OR
14 |  Select another model [goto step 8]
15 else
16 |  continue
17 end
18 if Analyzed_Result has HighVariance then
19 |  Select another model [goto step 8]
20 else
21 |  Measured_Performance ← Measure test performance by reviewing
   |    "Test Split Performance"
22 end
23 if Measured_Performance is Low then
24 |  check test data distribution and try to normalize the difference by
   |    creating fair splits
25 else
26 |  Deployed_Model ← Deploy the model for prediction"
27 end
28 SLA_Violation ← let the Deployed_Model predict SLA Violations
```

In a machine learning dataset, a training set is used to build up a model, while a test set is applied to validate the built model in a dataset. Data in the training set is excluded from the test set. Hence, in the proposed method, the training set is selected at first, and then different regression models like Random Forest Regressor, Kernel Ridge, Gradient Boosting Regressor, and Time Series are applied to train the model for prediction (line 8). While training the data, various measures are taken to check if the chosen training data

results are sufficient to validate their application on testing data. The split performance of training data as a result of checking high bias and high variance leads to choosing a new model as the selected model is not capable of predicting or changing model rules value like a number of auto-regressive lags, moving average, number of iterations, trends, etc., and training the model any longer. In case the performance results into high variance, either a different model is selected, or more data is considered in order to train the model again and check the performance. The entire process is repeated until a model is applied with low bias and low variance results for training set performance (lines 11-22).

Once the model is trained, the same model is used to predict using the testing data, i.e., the unseen data. Tested data performance is evaluated to see if the model is appropriate for real-time prediction based on different evaluation factors. This step checks the fitting quality of the chosen model in the selected form with experiments. At this point, the dataset which was put aside earlier comes into play. Evaluation allows testing of the model against the data that has never been used for training. This should reflect as to how the model will actually perform in the real world. If the candidate model lacks the required quality, the process of model-based analysis repeats the work to choose another model (lines 22-27). A different range of performance metrics, such as R^2 in Eq. 3.7, adjusted R^2 in Eq. 3.9, MAE in Eq. 3.6, and RMSE in Eq. 3.5 is applied to check the performance of a model on unseen data. The complete process is repeated until a model with good performance is selected.

Finally, the residual error between predicted and validated response time are calculated to check how well the fitted model predicts the response time in the future. In situation when a rejection occurs, the whole process starts again from making a model selection to the validation of the fitted model and then the prediction of the response time. The prediction will be accurate when the new observed response time lies within the 90% prediction interval, and the residual errors are less than 5%.

At this time, the model is ready to be used for quantitative analysis of the influence of different factors on service quality and the prediction of situations when SLA could be violated. After the model is selected, it is ready to make SLA violation prediction. While doing so, if the predicted response time is close to a threshold value or above the predicted value, then SLA is violated else the process continues to work normally.

A time-series regression model is proposed for continuous checking of SLA violation. The Time series model is an effective fitted model for this problem as this model makes a continuous prediction of violation in real-time, which is the main requirement for any service provider.

5.2.1 Fitted Model with Machine Learning

This section introduces the machine learning models that are used for the task of SLA violation prediction and prevention. Those models are ARIMAX (Autoregressive Integrated Moving Average with Explanatory Variable) time series regression model and VARMAX (Vector Autoregressive Moving Average with eXogenous) regression model.

Time series models use the past movements of variables in order to predict their future values. Unlike structural models that relate the variable, we want to forecast with a set of other variables. However, in terms of forecasting, the reliability of the estimated equation should be based on out-of-sample performance [21]. The time series model can mostly produce quite accurate forecasts, especially in the case that there are multidimensional relationships among variables. Time-series refers to an ordered series of data. It usually forecasts what comes next in the series using the Box-Jenkins approach.

Time series models are divided into two part:

- **Univariate Time Series:** Only one variable is varying over time; for example, the data collected from a sensor that measures the temperature of a room every second. Therefore, each second, there is only a one-dimensional value, which is temperature.
- **Multivariate Time Series:** Multiple variables are varying over time, for example, a tri-axial accelerometer. There are three accelerations, one for each axis (x, y, z), and they vary simultaneously over time. Considering the data that you showed in the question, you are dealing with a multivariate time series, where value_1, value_2, and value_3 are three variables changing simultaneously over time.

ARIMAX (AutoRegressive Integrated Moving Average with Explanatory Variable)

The standard ARIMA (AutoRegressive Integrated Moving Average) model allows

making forecasts based only on the past values of the forecast variable. The model assumes that future values of a variable linearly depend on its past values, as well as on the values of past (stochastic) shocks. The ARIMAX model is an extended version of the ARIMA model. It also includes other independent (predictor) variables. The model is also referred to as the vector ARIMA or the dynamic regression model. The ARIMAX model is similar to a multivariate regression model. Additionally, it allows taking advantage of autocorrelation that may be present in residuals of the regression to improve the accuracy of a forecast.

For obtaining a non-seasonal ARIMAX model, it is required to combine differencing with autoregression and a moving average model. The full model can be written as:

$$y_t^{'} = c + \phi_1 y_{t-1}^{'} + ... + \phi_p y_{t-p}^{'} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} + \varepsilon_1 \qquad \textbf{(5.1)}$$

where y'_t is the differenced series (it may have been differenced more than once). The predictors on the right-hand side include both lagged values of y_t and lagged errors. This is called an ARIMAX (p, d, q) model. In the model, the variables or parameters are listed as below:

- p is the number of autoregressive terms,
- d is the number of nonseasonal differences needed for stationarity, and
- q is the number of lagged forecast errors in the prediction equation.

VARMAX (Vector Autoregressive Moving Average with eXogenous)

The Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX) is an extension of the VARMA model that also includes the modeling of exogenous variables. It is a multivariate version of the ARMAX method. Exogenous variables are also called covariates and can be thought of as parallel input sequences that have observations at the same time steps as the original series. The primary series are referred to as endogenous data to contrast it from the exogenous sequence(s). The observations for exogenous variables are included in the model directly at each time step and are not modeled in the same way as the primary endogenous sequence (e.g., as an AR, MA, etc. process). The VARMAX method can also be used to model the subsumed models with exogenous

variables, such as VARX and VMAX. The method is suitable for multivariate time series without trend and seasonal components with exogenous variables.

The below formula represents a VARMAX model:

$$Y_t = \sum_{i=1}^{p} \Phi_i Y_{t-i} + \sum_{i=0}^{b-1} B_i X_{t-i} + \sum_{i=1}^{q} \Theta_i E_{t-i} + C + E_t$$

Where,

- Y_t is a vector of n response variables (dependent)
- X_t is a vector of m exogenous variables (independent)
- p is the number of previous periods of the endogenous variables included in the model
- q is the number of previous periods included in the moving average
- b is the number of previous periods of exogenous variables included
- Φ_i is an n * n matrix of autoregressive parameters
- B_i is an n * m matrix of exogenous variable parameters
- Θ_i is an n * n matrix of moving average parameters
- E_t is the difference between the actual and the predicted value of Y_t, (Y_t - $\hat{Y}_t$)

5.3 SLA Violation Prevention

Once the selected model predicts SLA violation, the final step is to avoid SLA violation through QoS optimization. Figure 5.4 illustrates the last module of the proposed framework for SLA Prevention, with details presented in Algorithm 3. For optimization, the response time of the job is reduced and kept below a threshold to avoid violation. The response time of incoming jobs is optimized by selecting single or multiple controllable factors and increasing or decreasing their value to reduce response time and avoid violation.

The proposed approach of QoS optimization goes through the following steps to prevent SLA violation:

- Identifying the most influential controllable factors
- Fitting new data with a machine learning regression model
- Conducting experiments with single/multiple factors and collecting observational data

This procedure is repeated until SLA prevention is achieved, or the list of controllable factors are exhausted.

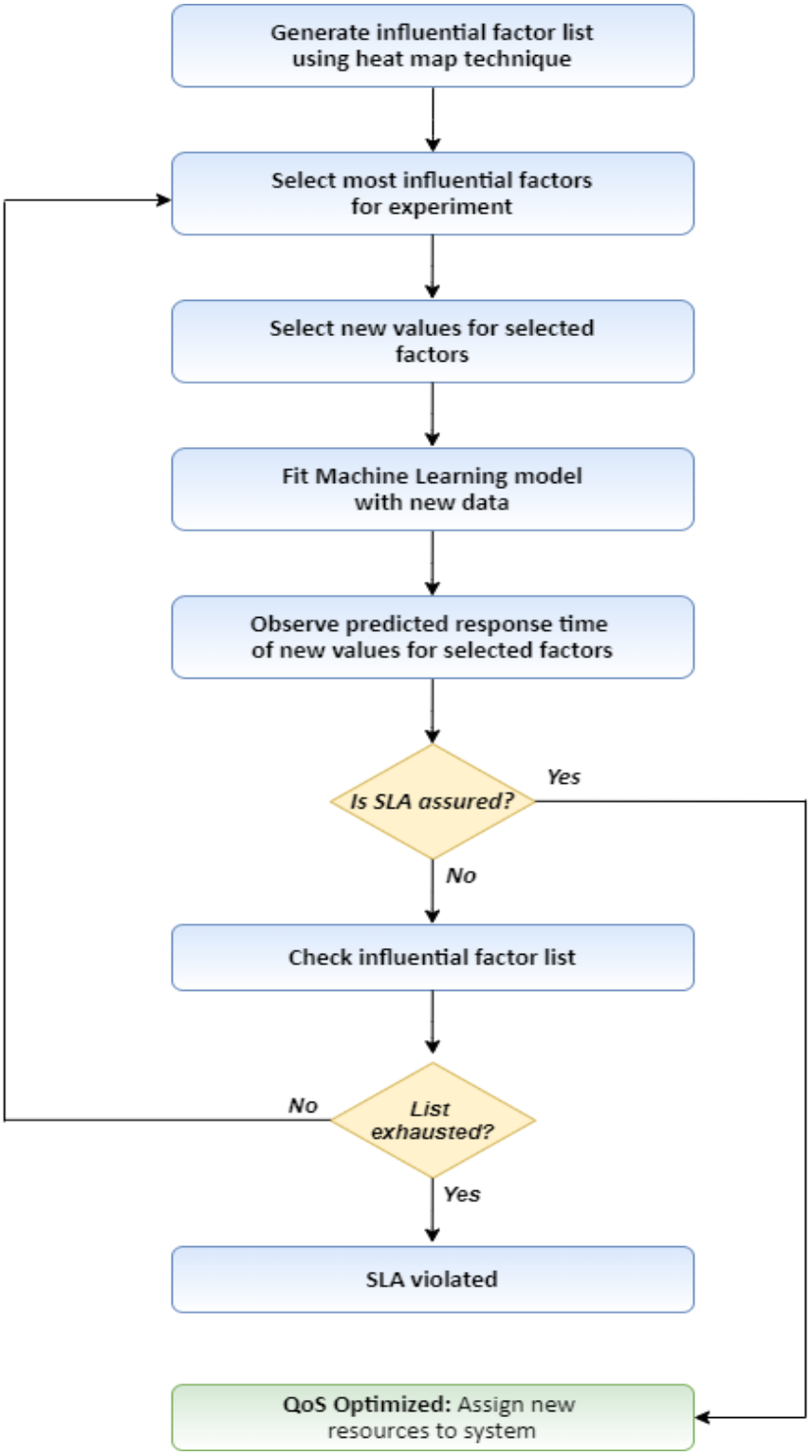


Figure 5.4 – SLA Violation Prevention

Algorithm 3: SLA Violation Prevention

```
1  Input: SLA Violated Data
2  Output: SLA Prevented or SLA Violated
3  Violated_Data ← SLA violated data
4  Influential_factor_list ← Generate influential factor list using heatmap
     technique
5  foreach influential_factor in Influential_factor_list do
6  |   Most_influential_factors ← Select most influential factors for
   |     experiment
7  end
8  New_Data ← Select or assign new values to Most_influential_factors
9  Predicted_response_time ← Fit already selected model with New_Data
     and predict response time
10 if Predicted_response_time is preventing SLA violation then
11 |   QoS_optimized_resources ← Assign QoS optimized resources to the
   |     system to prevent SLA violations
12 else if System is Exhausted then
13 |   Output ← SLA Violated
14 |   Quit
15 else
16 |   check Influential_factor_list [got step 5]
17 end
```

5.3.1 Heatmaps

Heatmap is a data matrix, visualizing values in the cells using a color gradient. This gives a good overview of the largest and smallest values in the matrix. Rows and/or columns of the matrix are often clustered so that users can interpret sets of rows or columns rather than individual ones [41].

In other words, a heatmap is a type of graphical representation of data that consists of a set of cells, in which each cell is painted with a specific color according to a specific value attributed to the cell. The term “heat” in this context is seen as a high concentration of geographical objects in a particular place. Heatmaps show the distribution of objects or phenomena across the entire surface. More generally, heatmaps can be viewed as the surfaces of densities. Such surface density fairly illustrates the location of the concentration

of points or linear objects [41]. An example of a heatmap is shown below in Figure 5.5.

At a fundamental level, heatmaps are implemented as spatial matrices with cells colored after their values. Explicitly, they encode a continuous quantitative variable as a color in space through a color transfer function to a sequential color scheme [42].

Broadly speaking, they fall into two classes:

- Image-based heat maps and
- Data-matrix heat maps.

Image-based heat maps display numerical information that is mapped over an image, an object, or a geographic location. On the other hand, data-matrix heat maps display numerical data in a pseudo-colored tabular or matrix format. The data may be subsequently clustered using various measures of similarity or dissimilarity [43].

Figure 5.5 – Heatmap generated on a sample image [42]

5.3.2 Process of SLA Prevention

The selection of the right influential factor is the most crucial step, as this factor helps most in preventing SLA violations with optimization. Influential factors can be categorized either as controllable or uncontrollable ones [44]. Controllable factors refer to the ones that service providers can control or configure during the execution of web services, such as CPU, storage, cache, etc. Uncontrollable factors are those that cannot be controlled by service providers, such as workload, network environment, or the requirements of services consumers. Controllable factors can be further subdivided into two groups. One group includes factors that cause an increase or decrease in the operational cost while changing their values. Another group contains factors that have no impact on the operational cost

while their levels are modified. The objective of the proposed approach is to achieve design optimum without causing an unnecessary increase in the operational cost while reconfiguring web services and optimizing its QoS. The feature importance or heatmap technique is applied to find the right influential factor from the list of controllable elements (lines 4-7). By using this technique, the list of influential factors affecting the response time is acquired.

SLA Violation Prevention can be modeled as a 4-tuple (Factor, QoS, Model, Result), in which:

- Factor is the input variable of the local model, such as CPU, cache level, etc.
- QoS is the output response (variable) of the fitted model, such as response time.
- Model is the selected model that reflects the relationship between one controllable factor and the response. It is done either using one or multiple influential factors.
- Result is the reduced response time.

Based on the generated heatmap results, the maximum value is the most influential factor for service optimization; hence its value is changed (line 8). Initially, experiments are performed using a single most influential factor, and if response time is not reduced as required to avoid SLA violation, then multiple influential factors are considered. For multiple influential factors, the value of more than one influential factor is changed and applied for prediction. The newly selected controllable factor value is put in the selected machine learning model to predict the new response time (line 9). In case the new predicted response time is below the threshold, the SLA is prevented, and resources are applied to the system. However, if the predicted response time is not sufficient to avoid SLA, the whole process is repeated with varied influential factors and diverse values until the SLA is avoided or the list of influential factors is exhausted (lines 10-16).

5.4 Manual Versus Automated Unified Framework of SLA prevention

Among a diverse range of approaches working on QoS optimization, the Sensitivity analysis, SLA violation prediction, and Design optimization are three crucial, yet normally

separated techniques. Presented in Figure 5.6 is a manual versus automated framework that tackles the main issue of SLA violation by combining these three techniques into a unified process to analyze, predict, and prevent SLA violations.

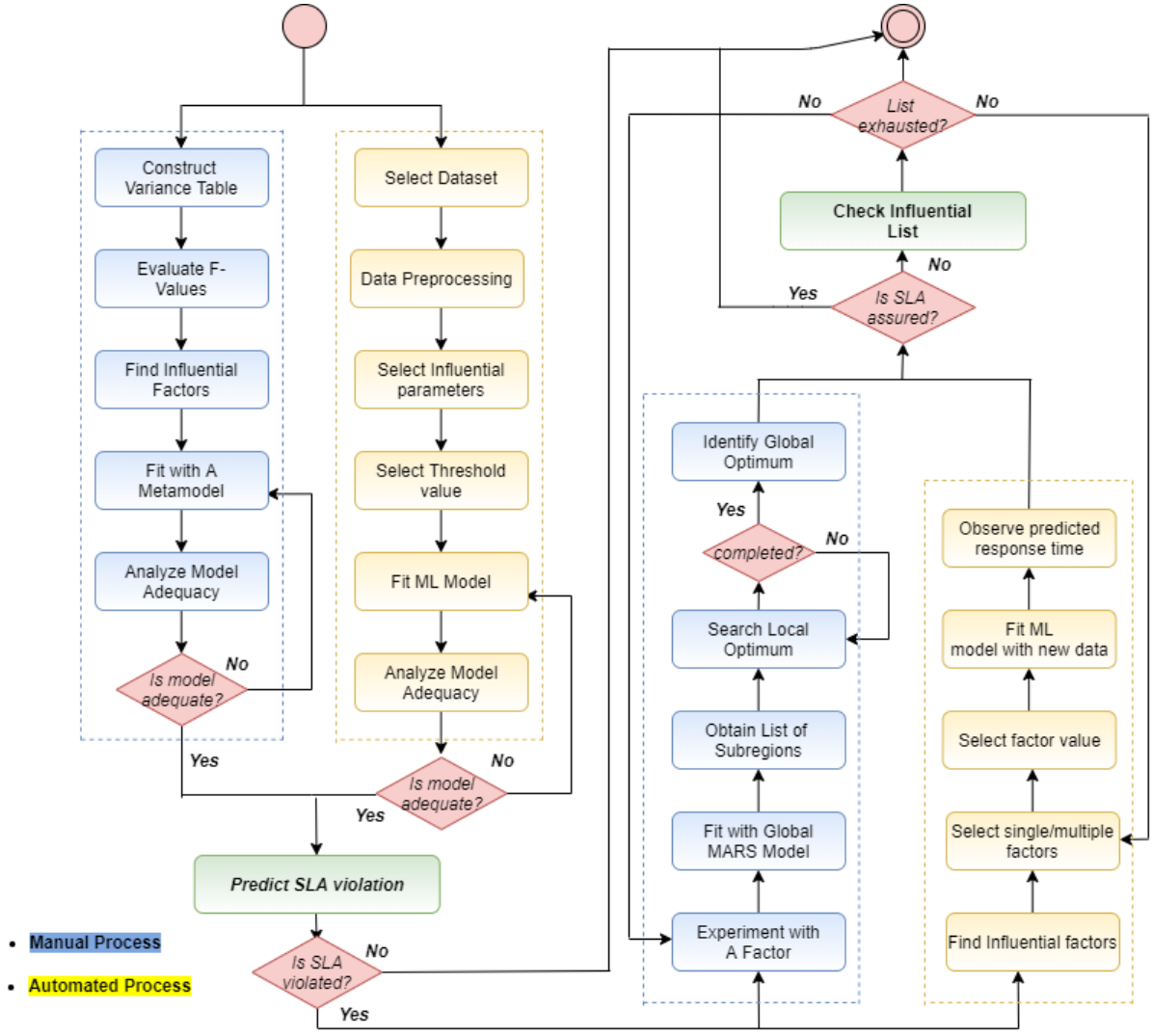


Figure 5.6 – Overall flowchart for Manual versus Automated process

The overall goal of the manual and automated process is to predict and prevent SLA violations. Recently, Cheng et al. [1] proposed a framework to utilize the Sensitivity analysis for the identification of influential factors with dominating impacts on QoS, as shown in the manual process. They used metamodel-based analysis to select a fitted surrogate model for domain-independent prediction of SLA violation. The residual error

between predicted and validated response time is calculated to select the best-fitted model for prediction. In the suggested method, the process of evaluation can be used by service consumers for service selection, and it can be used by service providers to study SLA violations [1]. However, in the proposed framework for SLA prediction, a manual process is employed, which is time-consuming and cannot tackle complex real-world data.

Cheng et al. [2] also presented a new approach for QoS optimization to improve the quality of web services. In this approach, four procedures were performed: identifying influential factors, collecting observed data, fitting collected data with the MARS model, and identifying global optimum till SLA is prevented. However, for optimization, only one controllable factor (cache level) is used, which will not lead to optimization for a complex system. So, there is a need to automate the proposed approach using Machine Learning techniques, which can easily adapt to any complex system and overcome the limitation of these approaches.

This thesis proposes to utilize Machine Learning to predict and prevent SLA violations into a unified framework to help service providers analyze, predict, and prevent SLA violations. Machine Learning techniques are used to automate their proposed framework, which eliminates many manual steps. Past research mostly relies on heuristic methods for the prediction of violations. Even though Machine Learning has been utilized in different territories of QoS management, the experiments done for the most part come under a confined setting, which isn't ascendable to real-world data. Nonetheless, this research adopts a systematic machine learning approach applied to real-world data that provides an insightful set of experiments. This automated process significantly enhances other requirements in terms of availability, performance, robustness, response time, and cost.

CHAPTER 6

IMPLEMENTATION AND EXPERIMENTS

In the previous chapters, several methods and algorithms for the regression task of SLA violation prediction and prevention were introduced. In this chapter, section 6.1 explains the toolkit and environments in which the experiments are performed. Later, section 6.2, reports and analyzes the results of several machine learning techniques. As an assessment of the proposed approach, section 6.3 examines accuracy, efficiency and robustness in comparison with existing methods.

6.1 Environments and Toolkits

In this section, the details of the environment and toolkit that were used for the implementation are described.

Software and Hardware Requirement

The implementation of proposed methodology was performed on Processor: Intel (R) Core (TM) i7-5820K CPU @ 3.30GHz, 3301 Mhz, 6 Core(s), 12 Logical Processor(s)

Hardware	1 GPU
Operating System	Windows 10 (64-bit)
Programming Language	Python 3.7
Markup Language	XML (eXtensible Markup Language)
Integrated Development Environment	Jupyter Notebook 5.5.0
Source Code Editor	Notepad++
Libraries	Keras, numpy, metplot, statsmodels.api, etc

Python

Python is a general-purpose, interpreted, dynamic programming language that is widely used for data analysis. The robust collection of scientific, statistical and mathematical tools in python allow easier implementation of machine learning models. Libraries such as NumPy (Python's Numerical Library) [44], SciPy (Python's scientific library) [45], Scikit-Learn and Imbalanced-learn are built on top of python to provide easy computation and analysis on data. In this work, we have used Python 3.7 along with many other libraries has been used.

Scikit-Learn

Scikit-Learn is a machine learning library built on top of Python, Scipy, and NumPy [46]. Scikit-Learn provides various tools for data mining and analysis and is also an open source and commercially usable. It features different classifications, regression, and clustering algorithms such as Random Forests, Gradient Boosting, k-means, and Naive Bayes.

Imbalanced-learn

Imbalanced-learn is a python library built on top of Scikit-Learn, Scipy, and Numpy [47]. It offers many re-sampling techniques for unbalanced data such as over-sampling, under-sampling, and combination of both.

6.2 A Unified Framework Model Experiments

To examine the effectiveness of the proposed approach, experiments have been conducted on different user's application, which submits its required resources as jobs to the cluster in which each job contains several tasks. For readability, results are presented in the following three subsections: Data Analysis, SLA violation prediction, and prevention.

6.2.1 Data Analysis

Dataset

The dataset used contains a 29-day trace of Google's Cloud Compute, which was

published in 2011. For security reasons, part of the trace has been omitted or obfuscated. For example, the values for CPU, disk, and memory have been rescaled by dividing each value with their corresponding largest value in the trace. Also, the names of the users' applications have been hashed. The trace has six separate tables: Job Events, Task Events, Task Usage, Machine Events, Machine Attributes, and Task Constraints. The entity-relationship diagram of the database is shown in Figure 6.1.

The user's application submits its required resources as jobs to the cluster. Each job has several tasks. The entity-relationship diagram of jobs and tasks is depicted in Figure 6.1. The Job Events table traces the event cycle of the jobs that were submitted to the cluster. The tasks inside each job are tracked in the Tasks Events table. Each task is then assigned to a specific machine. Machine Events table shows the removal or addition of a machine to the cluster or update of its resources. Machine Attributes table shows the attributes of each machine, such as kernel version, clock speed, and presence of an external IP address [51]. Tasks can have constraints (e.g., A task may have zero or more task placement constraints, which restrict the machines on which the task can run.) on machine attributes, which are recorded in the Tasks Constraints table.

Metrics such as requested CPU requested memory, requested disk space, scheduling class, and priority of the task are all recorded in the Tasks Events table. The Task Usage table contains the actual usage of resources for each task. It contains information such as assigned memory and memory usage [51].

The features requested CPU, requested memory, requested disk space, scheduling class, and priority for each task are supplied to the regression model. A different set of data points are used for training and validation of the model. In three-fold cross-validation, one-third of data is used for validation, and the rest is used for training. These features are considered as high-level features that can semantically represent each task thoroughly. Other criteria, such as state and load of each machine, can also be considered as features when a request takes place. However, to prevent the model from overfitting, the use of too many features is avoided.

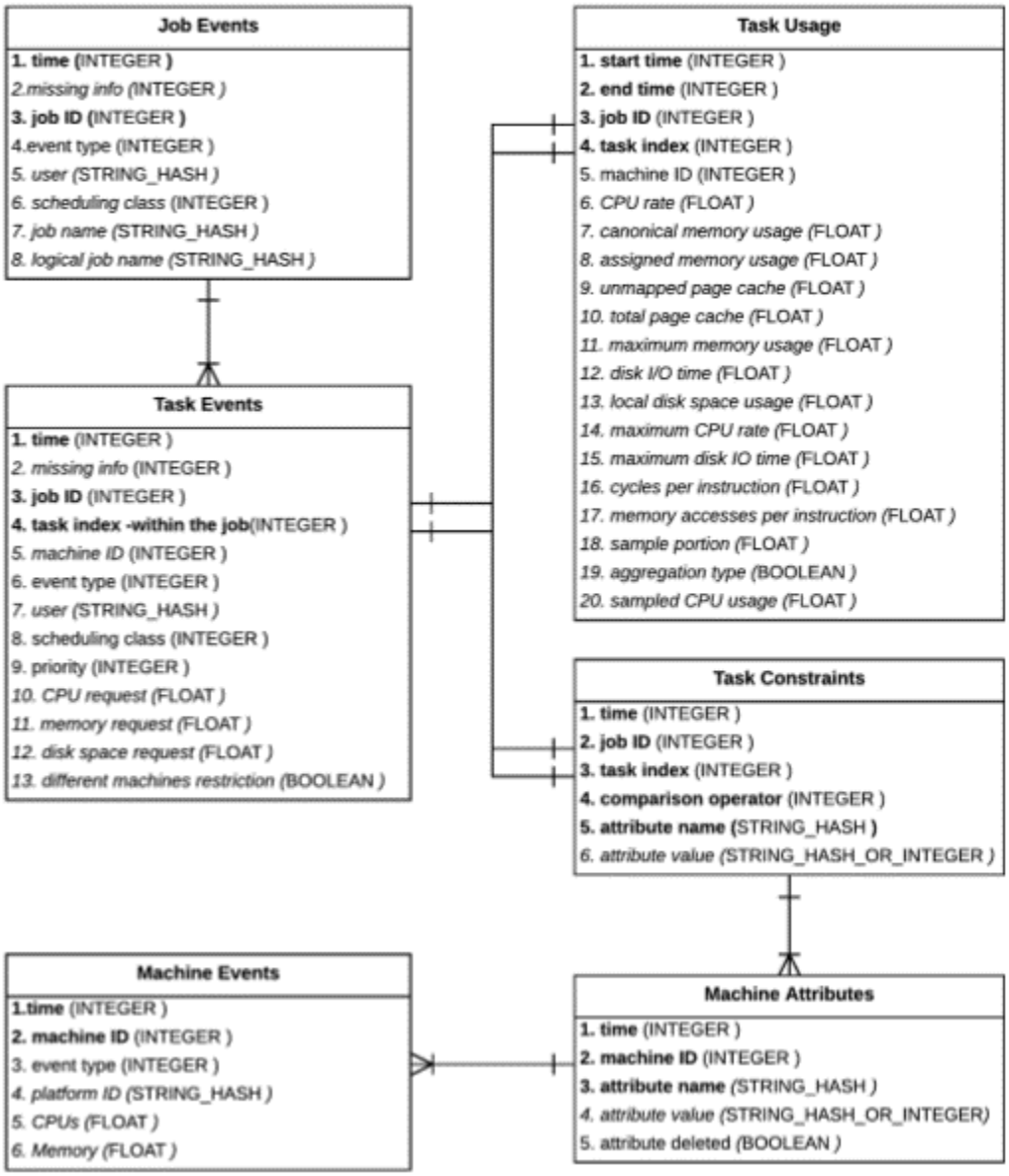


Figure 6.1 – Google's cluster trace dataset ERD (Entity Relationship Diagram)

Experiments

The first three steps of data analysis produce a single dataset. From the first three steps, the newly generated dataset is appropriate to apply in machine learning models by removing unnecessary columns, rows, and features from the dataset as it largely affects the response time of the incoming job request. From the single dataset, different graphs are generated to check the adequacy of the dataset and understand the relationship between various

resources. Through the generated graph, violated tasks and available resources at the time of the request are studied to predict future violations. The average response time is taken from the dataset to select the threshold value.

Figure 6.2 illustrates the mechanism of resource allocation in Google's Cluster Dataset. It shows the state of the cluster at 500 random snapshots. Snapshot is defined as a moment in time when the total sum of the requested resources is calculated. Similarly, available or allocated resources are calculated at each snapshot. In Figure 6.2, the total requested memory, assigned memory, memory usage, and available memory of the cluster at each snapshot is reckoned using Task Events, Task Usage, and Machine Events tables. Since all the requested resources are not used at the same time, it is the nature of the cloud to allocate fewer resources than requested resources and accept more requests than its available resources. Figure 6.2 shows that at all the 500 snapshots, the requested memory to the cluster is much higher than the actual usage of memory. Google scheduler has reserved a safe margin between the assigned memory and usage of memory at these snapshots. Thus, the availability rate is very high, and violations are rare.

Similarly, the available CPU and its usage are shown in the CPU analysis graph. As shown in Figure 6.3, there is a large gap between CPU availability and its usage, which means a lot of operational costs is wasted just to avoid the SLA violation.

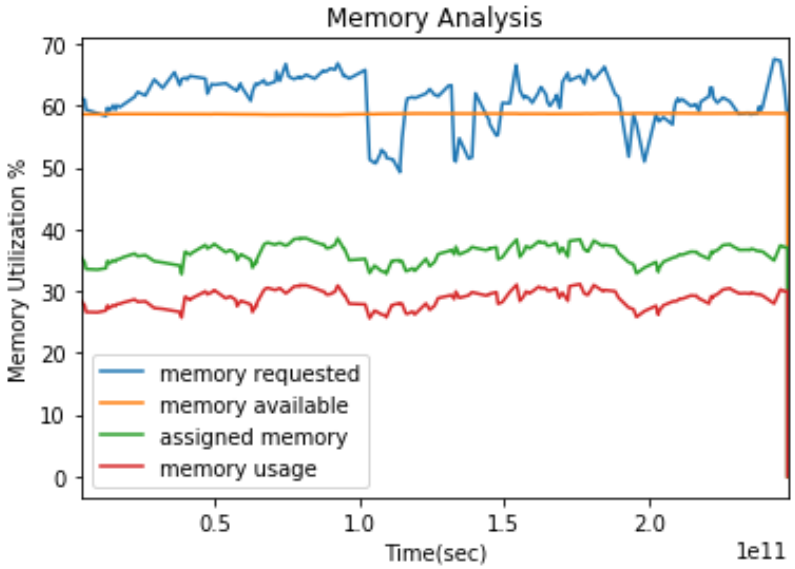


Figure 6.2 – Memory analysis graph. It shows the requested, available, assigned and used memory of the cluster

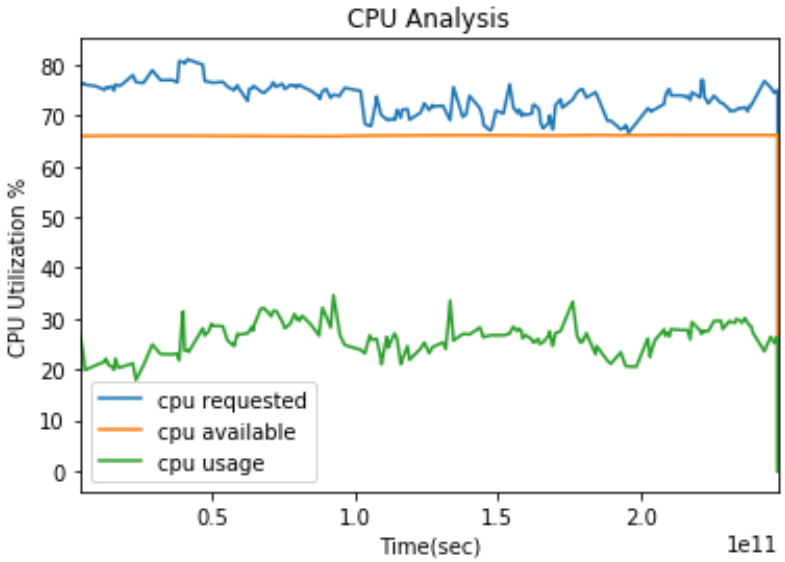


Figure 6.3 – CPU Analysis Graph

Through the generated graph, violated tasks and available resources at the time of the request are studied to predict future violations. Average response time is taken from the dataset to select the threshold value and is further used for SLA violation prediction.

6.2.2 SLA Violation Prediction

The first step is to perform data analysis to generate a single dataset that has a significant feature for predicting the response time of an incoming task. The next step is to perform experiments based on these features, to specify an appropriate type of model and to fit its surrogate model. The experiment data is a generated table from data analysis, which will be used to fit the surrogate model. It can be found that the performance behavior, in this case, is low-order nonlinearity by an initial analysis of these experimental data. As a result, the regression models are selected as the selected model type.

The eight models listed in the first column of Table 6.1, become the candidate forms of a regression model. Each of the eight candidate forms, fitting with the method of least squares, produces their R^2, adjusted R^2, MAE, and RMSE values (Table 6.1). As VARMAX yields the largest value of R^2, it is selected to predict the performance of the web service. In the cell some are NULL values as ARIMAX is univariant model and VARMAX is multivariant model, so value is not applicable for them in some cases.

Table 6.1: Comparison based on Different Models

Model	**R2**	**R2(Adj)**	**MAE (Univariant)**	**RMSE (Univariant)**	**MAE (Multivariant)**	**RMSE (Multivariant)**
Random Forest Regressor	0.801	0.781	5.7	2.39	6.06	2.46
Kernel Ridge	0.901	0.875	3.94	1.98	7.16	2.68
Gradient Boosting Regressor	0.821	0.802	5.79	2.41	6.47	2.54
Linear model Ridge(alpha=.5)	0.781	0.726	5.99	2.45	6.98	2.64
Linear model RidgeCV()	0.651	0.611	5.49	2.45	6.94	2.63
XGB Regressor	0.856	0.822	5.46	2.34	6.08	2.47
ARIMAX	0.938	0.901	2.32	1.52	NULL	NULL
VARMAX	0.962	0.931	NULL	NULL	1.75	1.32

The value of R^2 determines how well a model will predict the response time for new observations. According to the largest R^2 value of the VARMAX model shown in Figure 6.4, it is selected as the most appropriate form of the model to predict response time and is expected to have a better prediction accuracy than the other seven models. The smallest RMSE and MAE values of the VARMAX model, shown in Figure 6.6 and Figure 6.7, suggest that it outperforms the other models in terms of prediction accuracy. As a result, it is essential to use the R^2 to select the most appropriate one from a series of candidate forms for the model.

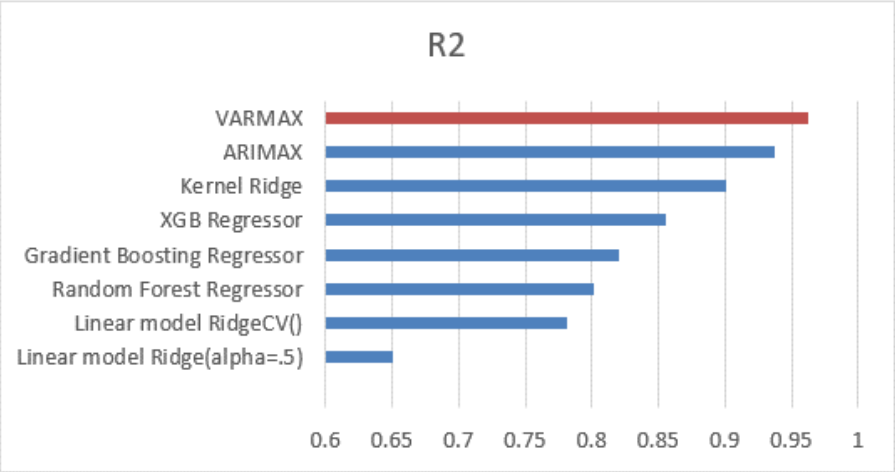


Figure 6.4 – R^2 value comparison between all fitted models

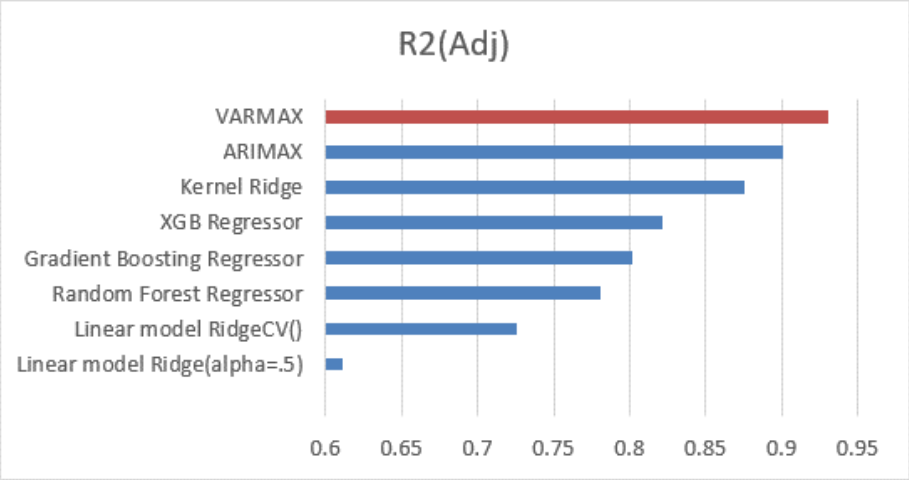


Figure 6.5 – Adjusted R^2 value comparison between all fitted models

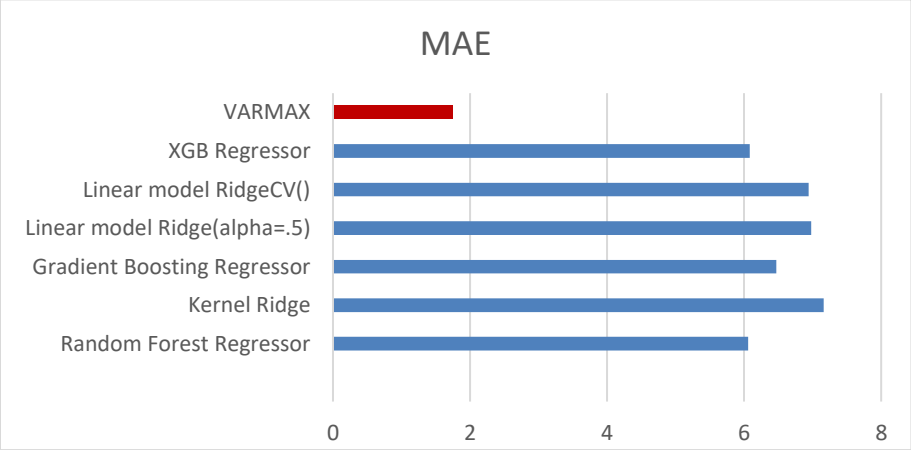


Figure 6.6 – MAE value comparison between all fitted models

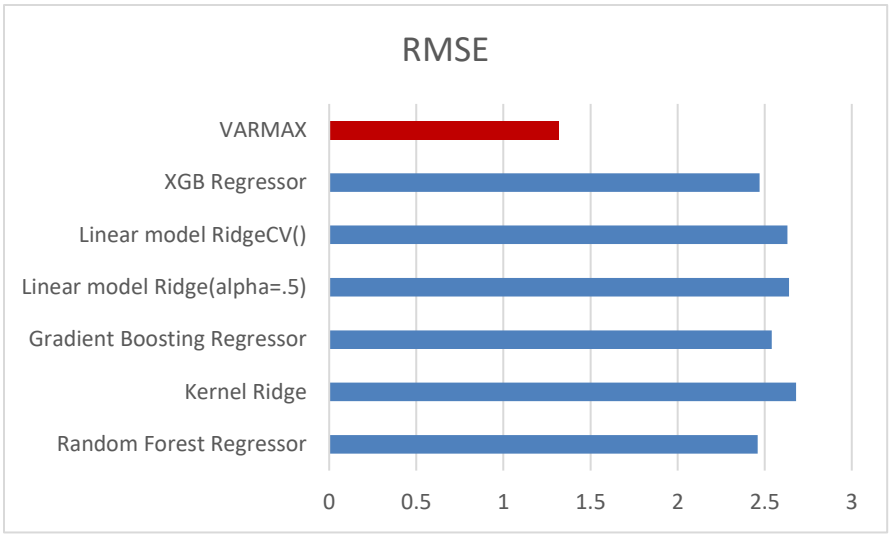
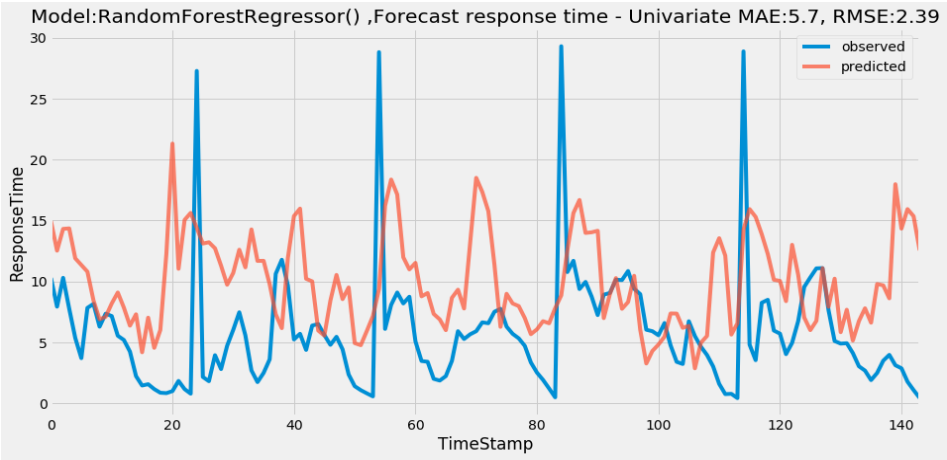
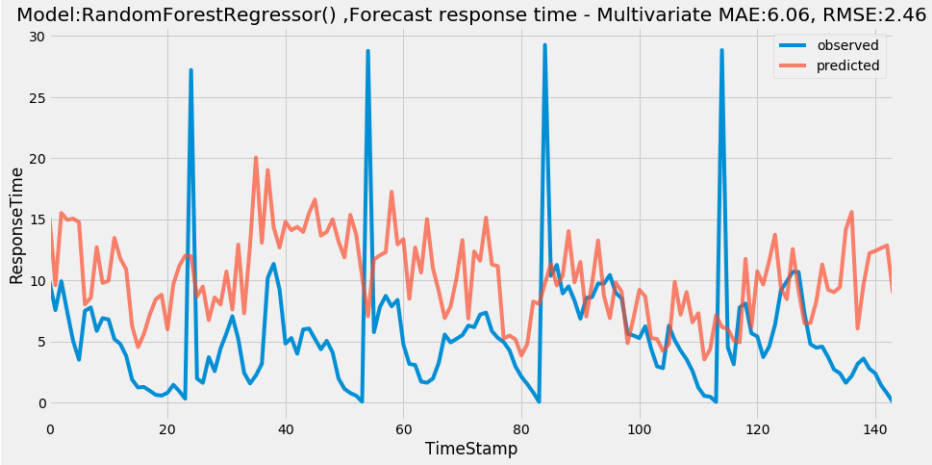
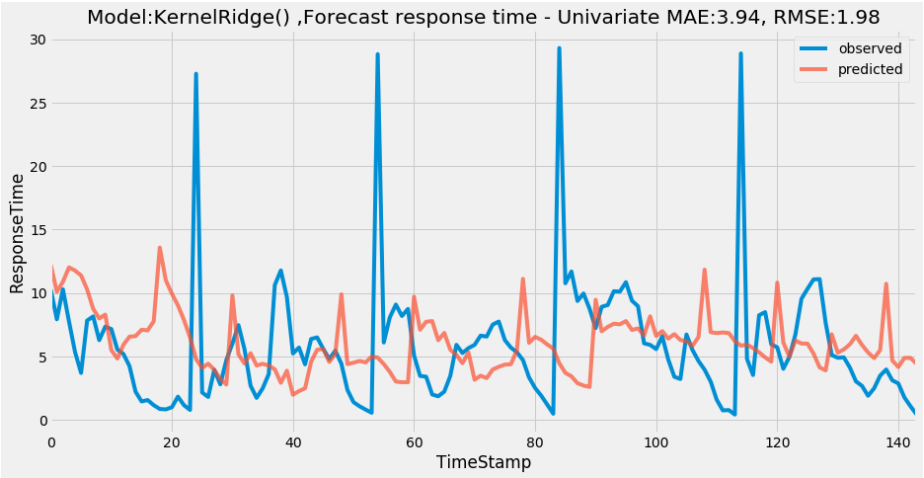


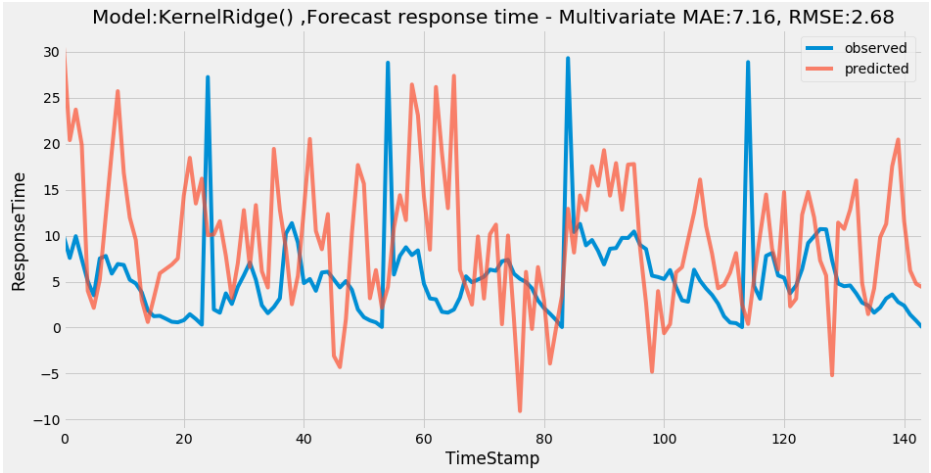
Figure 6.7 – RMSE value comparison between all fitted models

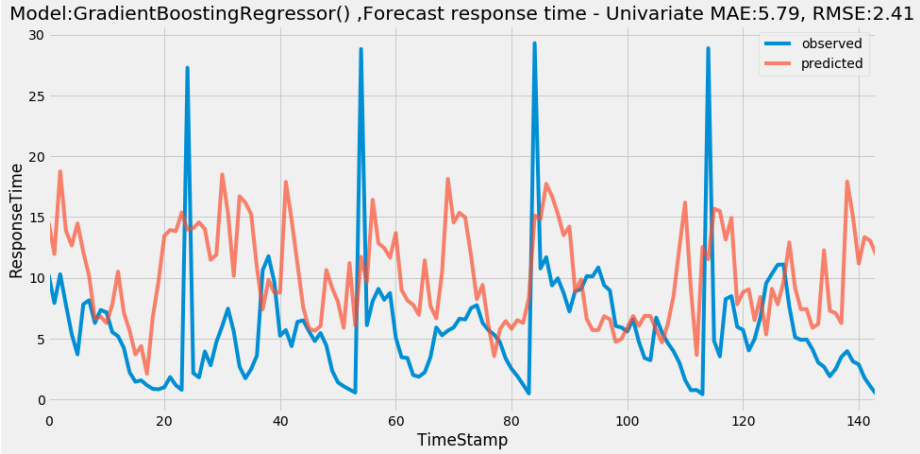
In addition to residual normality, it should also be checked as to how accurate the fitted model is. The plot of predicted vs. actual shown in Figure 6.9 for VARMAX is used to show that this fitted model is accurate. Consequently, there is a strong correlation between the predicted response time of the fitted model and the observed corresponding results. A visualized comparison between data with univariant and multivariant techniques is also shown in Figure 6.8.

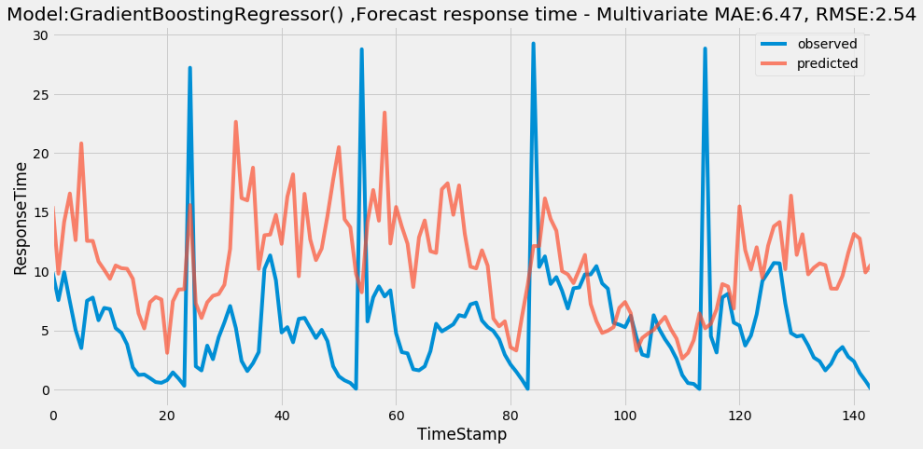


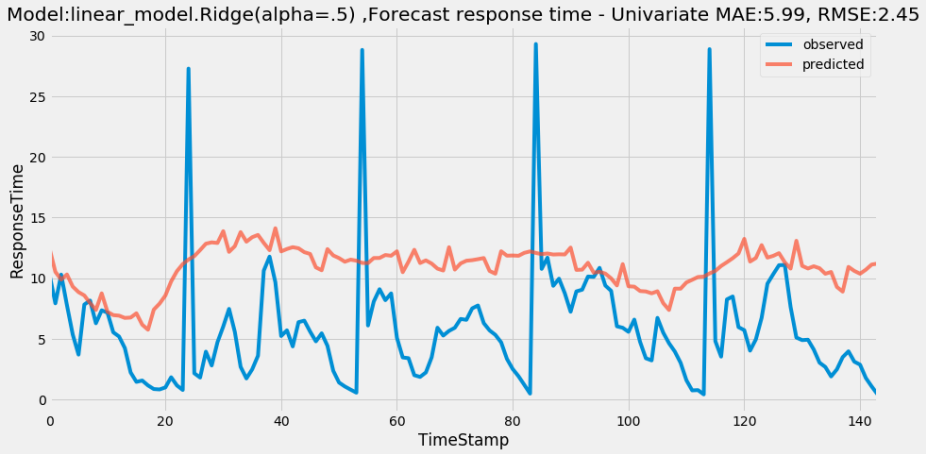

Model:RandomForestRegressor() ,Forecast response time - Multivariate MAE:6.06, RMSE:2.46
observed
predicted
ResponseTime
TimeStamp
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140

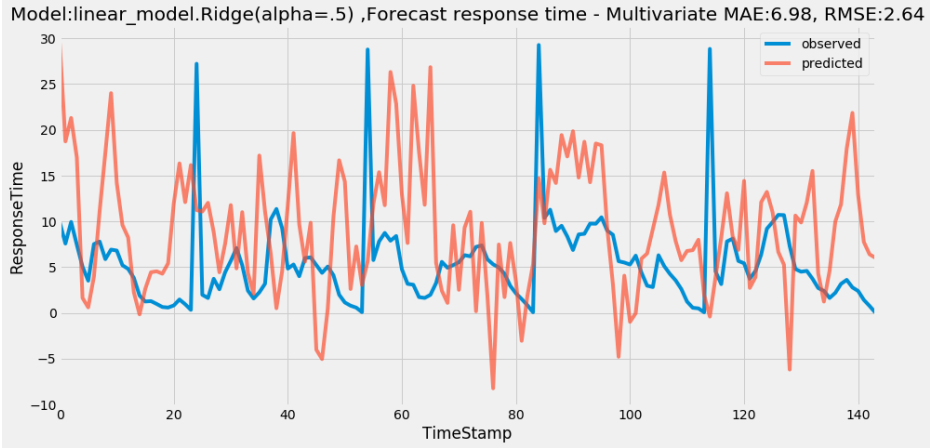

Model:KernelRidge() ,Forecast response time - Univariate MAE:3.94, RMSE:1.98
observed
predicted
ResponseTime
TimeStamp
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140

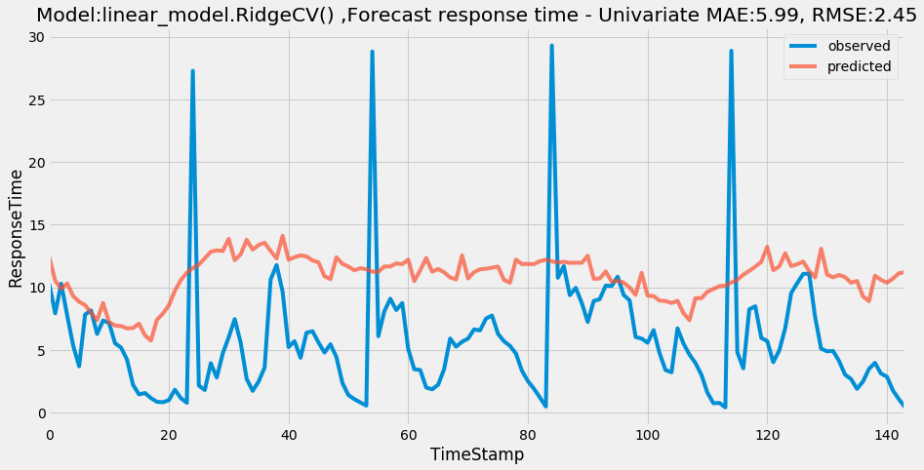

Model:KernelRidge() ,Forecast response time - Multivariate MAE:7.16, RMSE:2.68
observed
predicted
ResponseTime
TimeStamp
−10
−5
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140

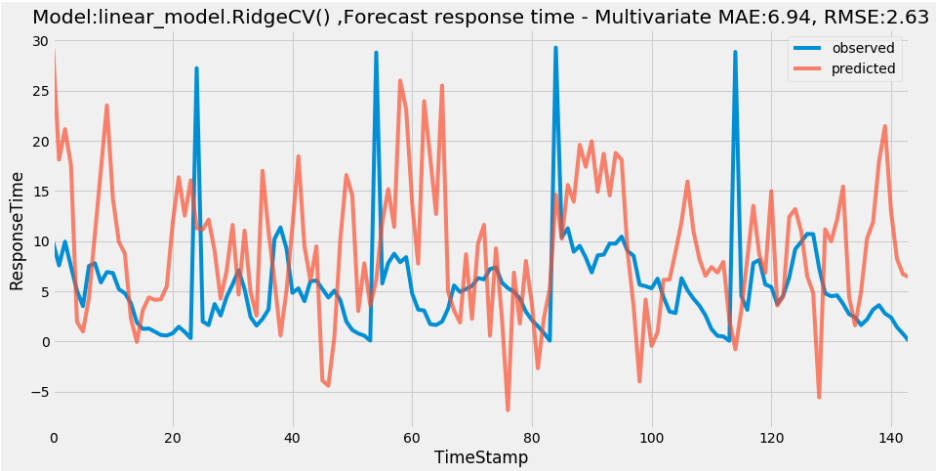

Model:GradientBoostingRegressor() ,Forecast response time - Univariate MAE:5.79, RMSE:2.41
observed
predicted
ResponseTime
TimeStamp
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140


Model:GradientBoostingRegressor() ,Forecast response time - Multivariate MAE:6.47, RMSE:2.54
observed
predicted
ResponseTime
TimeStamp
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140


Model:linear_model.Ridge(alpha=.5) ,Forecast response time - Univariate MAE:5.99, RMSE:2.45
observed
predicted
ResponseTime
TimeStamp
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140


Model:linear_model.Ridge(alpha=.5) ,Forecast response time - Multivariate MAE:6.98, RMSE:2.64
observed
predicted
ResponseTime
TimeStamp


Model:linear_model.RidgeCV() ,Forecast response time - Univariate MAE:5.99, RMSE:2.45
observed
predicted
ResponseTime
TimeStamp


Model:linear_model.RidgeCV() ,Forecast response time - Multivariate MAE:6.94, RMSE:2.63
observed
predicted
ResponseTime
TimeStamp

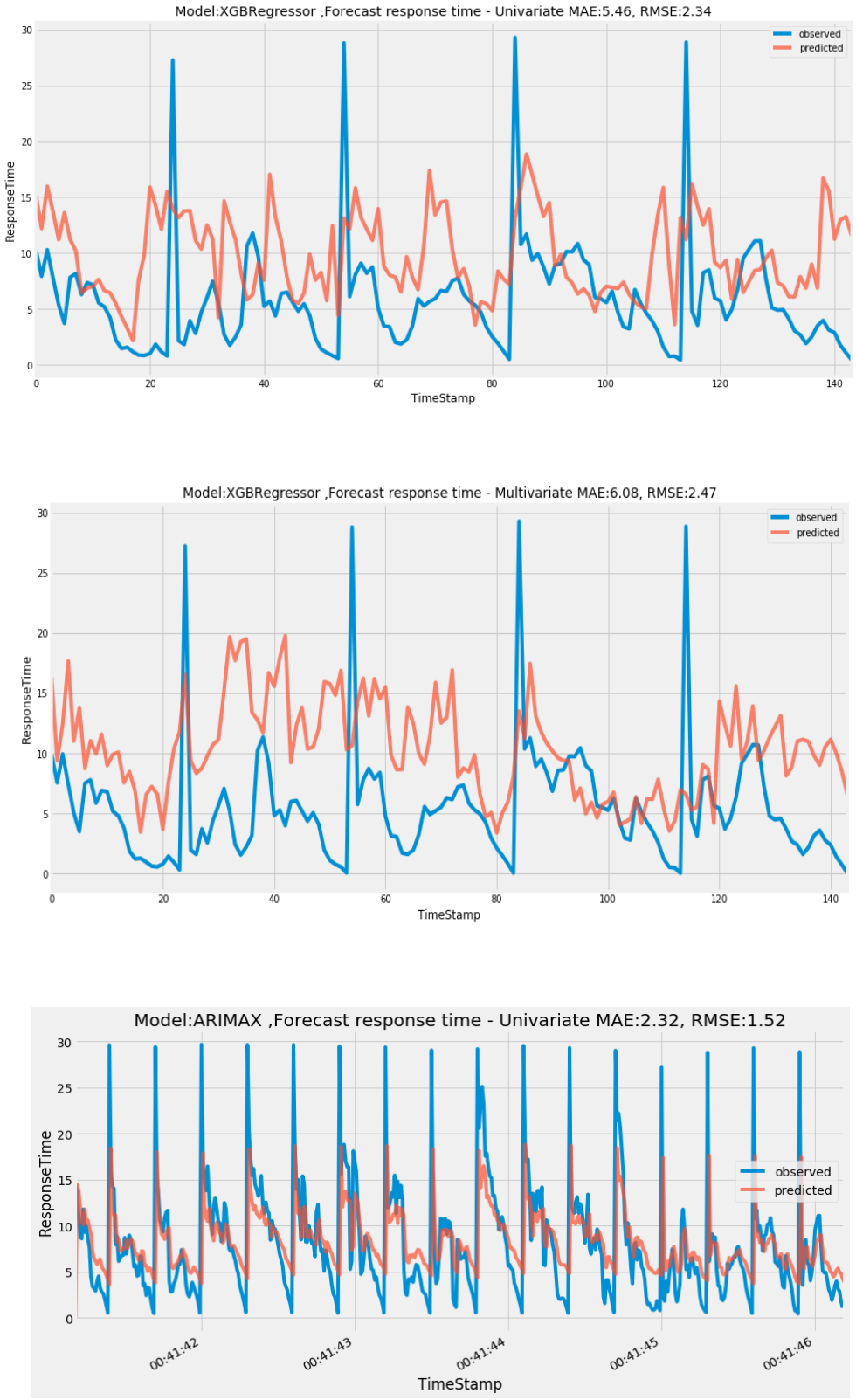


Figure 6.8 – The comparison between different selected regression models for Predicted Vs. Actual

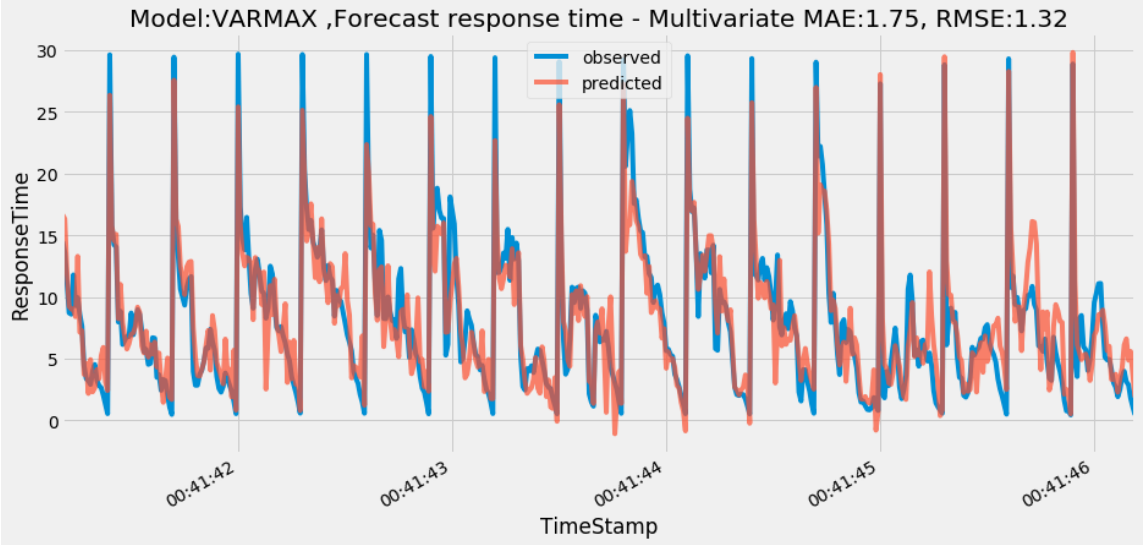


Figure 6.9 – Predicted Vs. Actual for VARMAX model

The results of the analysis bolstered with consistent evidence insinuate that the selected model can be used for predicting the response time of the web service under examination. As a result, this model can be used to analyze the effect of change in resource and their interaction on response time and to predict SLA violation.

After conducting experiments on several different regression models, the results show that time series models like ARIMAX and VARMAX predict accurate response time compared to some other interpolation techniques without overfitting of data. Even with the comparison of univariant and multivariant parameters, the multivariant model shows better results. As per the results, the machine learning model that can detect the SLA violation is selected.

The process of SLA violation prediction can be used by service consumers to select the most appropriate services while constructing an SOA system. Meanwhile, service providers can use the evaluation results to find out if and when SLA might be violated due to the fluctuation in different factors, to better understand the performance of a service with its fitted model, and to identify performance bottlenecks caused by influential factors.

In conducted experiments, response time is predicted to check the SLA violation detection. Using the same approach, different resources that are part of the SLA agreement

can check if the regulation of the SLA is met or not can be predicted, e.g., predict CPU assigned by a provider and check if violate the rules or not.

The next step is to avoid SLA violation detected through multi-factor QoS optimization. If the violation is not avoided, then a heavy penalty will be laid on the service provider, and it can harm the provider's reputation and business.

6.2.3 SLA Violation Prevention

The proposed approach of QoS optimization goes through the following steps to prevent SLA violation:

- Identifying the most influential controllable factors
- Fitting new data with the selected machine learning models
- Conducting experiments with single/multiple factors and collecting observational data

Identifying the most influential controllable factors

For selecting the right influential factor to reduce response time, feature importance, and heatmap technique is applied.

```
print(feat_imp)
```

```
CPU_usage_mean                      0.107083
memory_usage                        0.044761
assigned_memory                     0.047069
unmapped_page_cache_memory_usage    0.046828
page_cache_memory_usage             0.071576
maximum_memory_usage                0.039558
local_disk_space                    0.044319
CPU_rate                            0.408166
cycles_per_instruction              0.078510
memory_accesses_per_instruction     0.103164
aggregation_type                    0.008967
dtype: float64
```

Figure 6.10 – Different influential factor values from feature importance technique

Figure 6.10 shows feature importance results, in which the highest value from all the results is considered to be the most influential factor for predicting the response time, which is the CPU rate. The list of influential factors goes on from highest to lowest value as per result.

Heatmap:

Similarly, Figure 6.11 shows the relation between the parameters for predicting the response time, which helps to select a controllable factor. So, by getting this list of parameters through a graph, the parameter can be tuned to reduce response time.

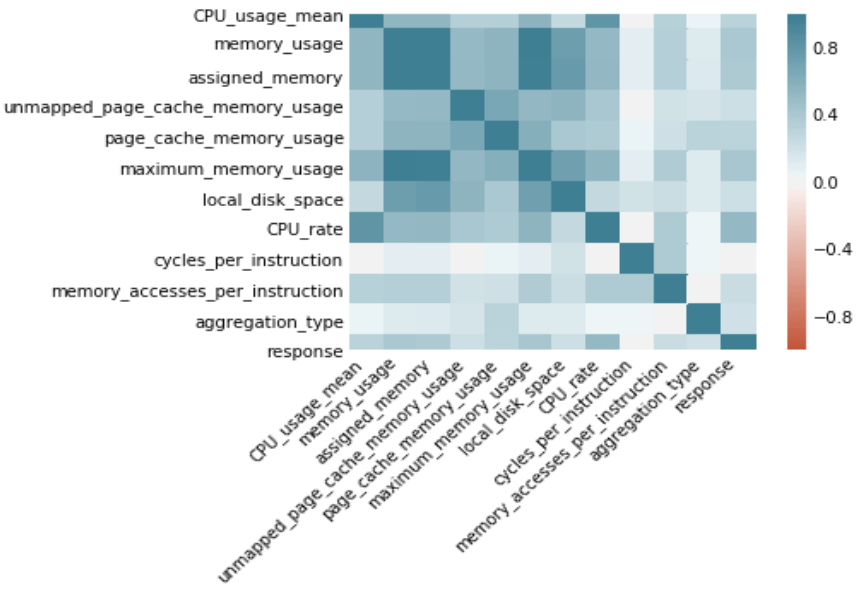


Figure 6.11 – The Comparison between different influential factors in respect to response time

Fit new data and Conduct experiments with single/multiple factors:

Once the influential factors are selected, the new different value is applied to the VARMAX model, which is used for SLA violation prediction. Figure 6.12 shows that once the influential factor value is changed and applied to ARIMAX model, the prediction is accurate for the new unseen data which denotes that the selected model is good for continuous SLA violation detection and prevention.

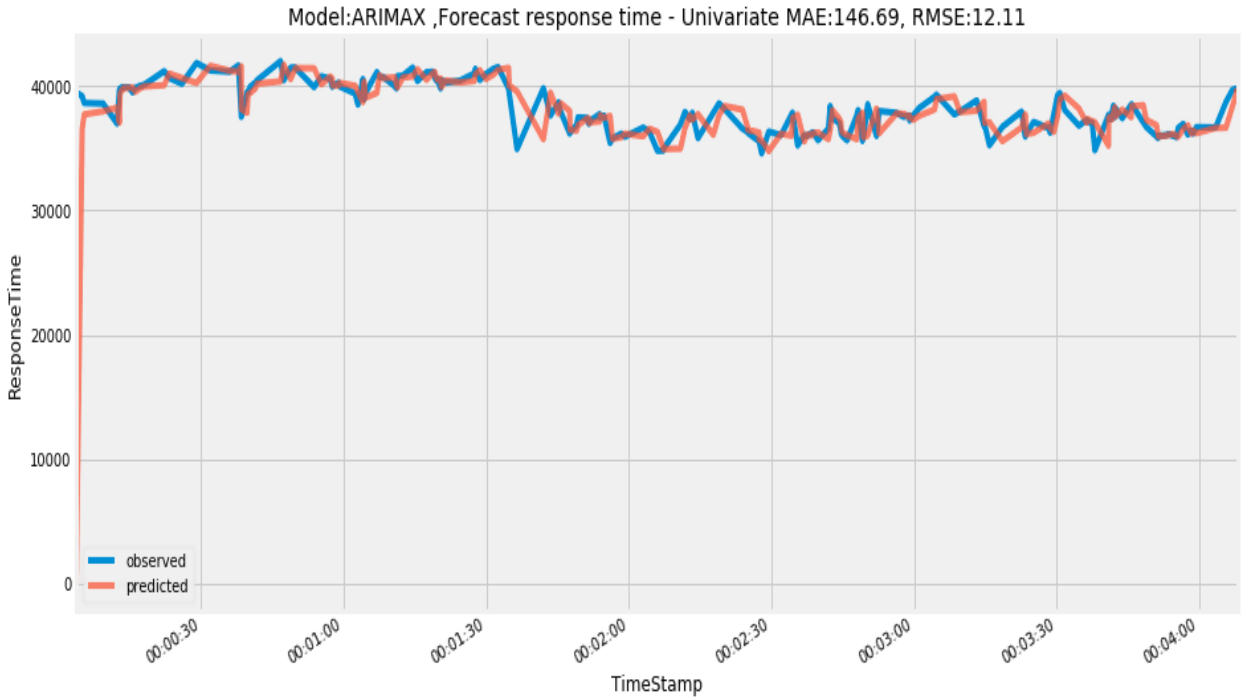


Figure 6.12 – ARIMAX model result with changed influential factor values

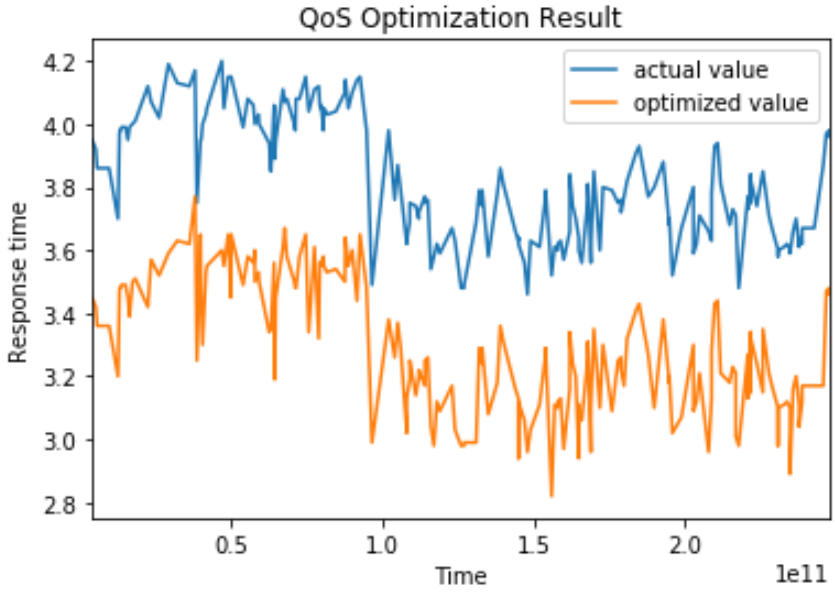


Figure 6.13 – Actual Vs. Optimized response time value

Figure 6.13 clearly shows that once multiple factors value is tuned, the response time of the incoming job reduces to a great extent.

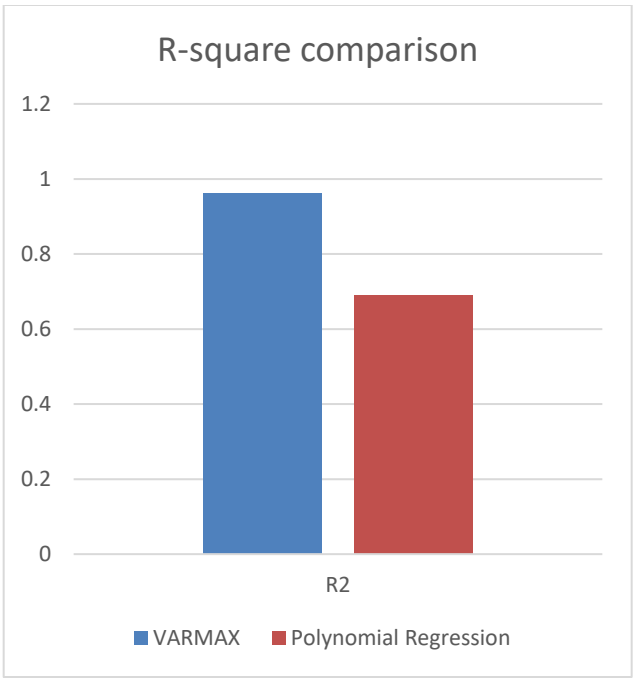
6.3 Comparison and Discussion

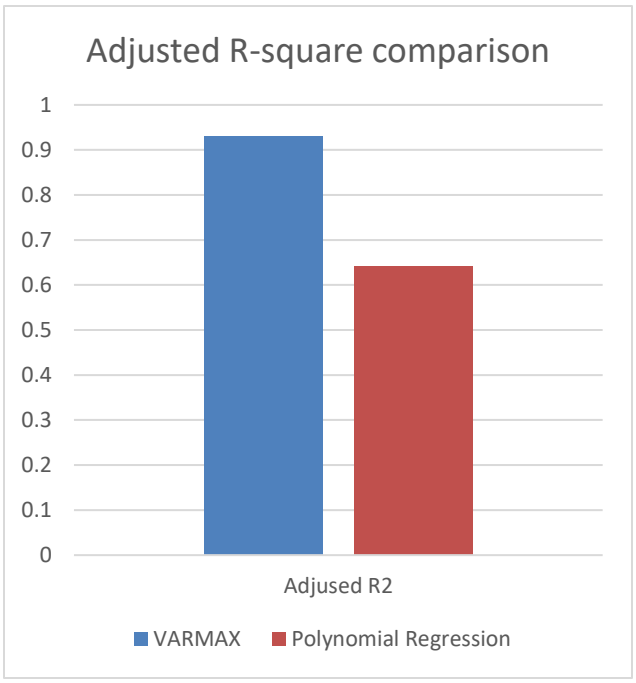
The service provider is always in need of tools that intuitively analyze if their service design provokes SLA violations and the ones that automatically guide them in preventing SLA violations. The proposed automated process will meet this requirement of a service provider.

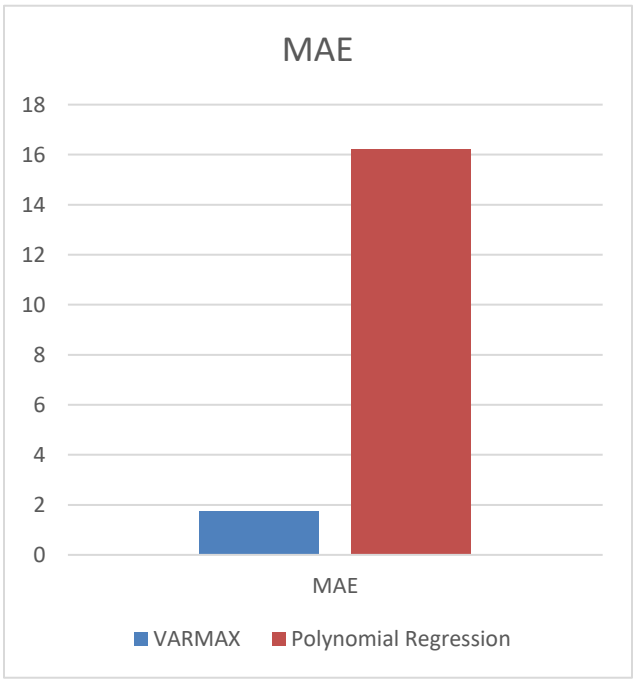
To check the adequacy of this approach, the proposed SLA violation prediction model is compared with the different proposed approaches. For SLA violation prediction, the result of the proposed approach is compared with the approach used by Cheng et al. [1]. In a selected dataset, the models used in different approaches are applied, and the performance result of those models is shown in Figure 6.14. From the result, it is evident that our approach can deal with complex data. The proposed model is capable of making a continuous prediction while other approaches are not capable of doing that for SLA violation prediction.

SLA violation prevention approach is compared with Cheng et al. [2]. In their approach, only one controllable factor is used to prevent the violation, which won't deal with big complex real-time service, while our proposed approach is capable of using multiple controllable factors to optimize the service and avoid violation easily.

A unified process proposed by Cheng et al. [1] [2] can tackle the main issues of SLA violation by combining three techniques (analyze, predict, and prevent SLA violation). However, for each technique, a different method is proposed, which is time-consuming. It also uses a basic fitted model, which, when applied in our dataset, shows that it cannot tackle complex real-world data. Furthermore, for optimization, only single controllable factor is used, which does not lead to the optimization of complex systems. Also, their proposed framework uses only a single web service to conduct experiments and collect results, which is not reliable. Our proposed automated technique can easily tackle this issue while consuming less time, as the same machine learning model is used to handle both the issues. Compared to other proposed models, our proposed approach successfully handles other requirements in terms of availability, performance, robustness, response time, and cost.







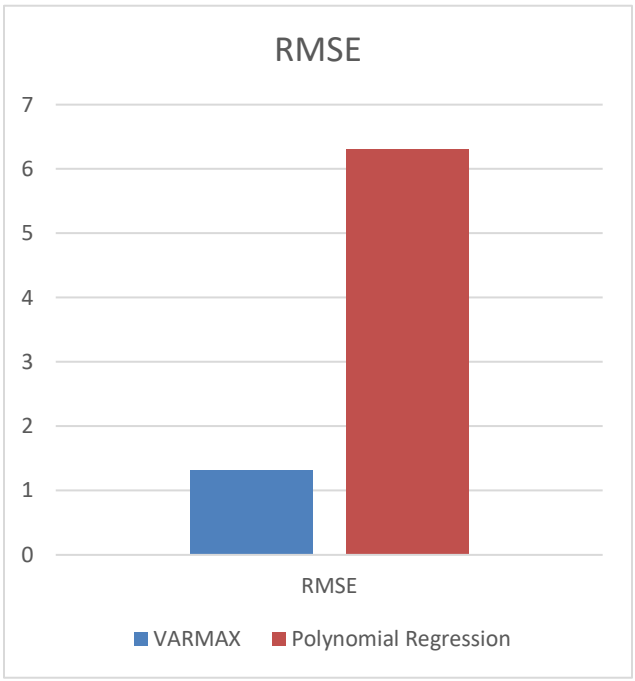


Figure 6.14: Performance evaluation between different approaches

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis research proposes an approach of using Machine Learning techniques to not only predict SLA violations but also prevent them through optimization. It helps the service provider to intuitively analyze if their service design provokes SLA violations and to automatically guide them from preventing SLA violations. Service providers can use the evaluation results to find out if and when SLA could be violated due to fluctuation in the incoming request.

The process of QoS evaluation can be used by a service provider to select the value of the most appropriate resources when preventing violation. If SLA is violated during run time check, the optimization strategy presented in this approach can be used to prevent a violation by selecting the right influential factor and apply the value to the same selected model. In addition to details of the proposed method, this thesis also includes a set of experiments, which will help to examine its usefulness for service providers working on the construction and refinement of services.

7.2 Future work

This research work provides some more possibilities for further improvement:

- One of the future works might be to explore other models that can be easily updated when receiving more training data.
- This work can also be extended, which includes experiments on more web services or SOA applications and the development of a working tool for use in practice.

- Experiments can also be extended using other machine learning models such as Support Vector Machines and Neural Networks.

REFERENCES

[1] Yuan, X., & Cheng, P. (2019). An approach of sensitivity and metamodel-based analyses for SLA violation prediction. In *2019 Canadian Conference of Electrical and Computer Engineering*.

[2] Yuan, X., & Cheng, P. (2019, May). A Strategy of QoS Optimization for Web Services. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)* (pp. 1-6). IEEE.

[3] Unnamalai, V. E., & Thresphine, J. R. (2014). Service-oriented architecture for cloud computing. *International Journal of Computer Science and Information Technologies*, *5*(1), 251-255.

[4] Vogel, A., Kerherve, B., von Bochmann, G., & Gecsei, J. (1995). Distributed multimedia and QoS: A survey. *IEEE multimedia*, *2*(2), 10-19.

[5] Rahimi, M. R., Ren, J., Liu, C. H., Vasilakos, A. V., & Venkatasubramanian, N. (2014). Mobile cloud computing: A survey, state of art and future directions. *Mobile Networks and Applications*, *19*(2), 133-143.

[6] Bani-Ismail, B., & Baghdadi, Y. (2018, August). A literature review on service identification challenges in service oriented architecture. In *International Conference on Knowledge Management in Organizations* (pp. 203-214). Springer, Cham.

[7] Huang, A. F., Lan, C. W., & Yang, S. J. (2009). An optimal QoS-based Web service selection scheme. *Information Sciences*, *179*(19), 3309-3322.

[8] Bass, L., Clements, P., & Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley Professional.

[9] Rozanski, N., & Woods, E. (2012). *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley.

[10] Woodside, C. M. (2001). Software resource architecture. *International Journal of*

Software Engineering and Knowledge Engineering, *11*(04), 407-429.

[11] Heckel, R., & Lohmann, M. (2005). Towards contract-based testing of web services. *Electronic Notes in Theoretical Computer Science*, *116*, 145-156.

[12] Yuan, X., Duan, S., & Huang, T. (2006, May). Analysis of service-oriented architectures with sensitivity analysis. In *2006 IEEE International Conference on Electro/information Technology* (pp. 372-376). IEEE.

[13] Darmann, A., Pferschy, U., & Schauer, J. (2010). Resource allocation with time intervals. *Theoretical Computer Science*, *411*(49), 4217-4234.

[14] John, M., Gurpreet, S., Steven, W., Venticinque, S., Rak, M., David, H., ... & Ryan, K. (2012). Practical Guide to Cloud Service Level Agreements.

[15] Casalicchio, E., & Silvestri, L. (2013). Mechanisms for SLA provisioning in cloud-based service providers. *Computer Networks*, *57*(3), 795-810.

[16] Buyya, R., Broberg, J., & Goscinski, A. M. (Eds.). (2010). *Cloud computing: Principles and paradigms* (Vol. 87). John Wiley & Sons.

[17] Gallizo, G., Kuebert, R., Oberle, K., Menychtas, A., & Konstanteli, K. (2009, October). Service level agreements in virtualised service platforms. In *eChallenges 2009*.

[18] Leavitt, N. (2009). Is cloud computing really ready for prime time?. *Computer*, (1), 15-20.

[19] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[20] Sammut, C., & Webb, G. I. (Eds.). (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

[21] Geurts, P. (2002). *Contributions to decision tree induction: bias/variance tradeoff and time series classification* (Doctoral dissertation, University of Liège Belgium).

[22] Wasserman, L. (2013). *All of statistics: a concise course in statistical inference*. Springer Science & Business Media.

[23] Stehman, S. V. (1997). Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, *62*(1), 77-89.

[24] Leitner, P., Michlmayr, A., Rosenberg, F., & Dustdar, S. (2010, July). Monitoring, prediction and prevention of sla violations in composite services. In *2010 IEEE International Conference on Web Services* (pp. 369-376). IEEE.

[25] Hussain, W., Hussain, F. K., Hussain, O., & Chang, E. (2015, November). Profile-based viable service level agreement (SLA) violation prediction model in the cloud. In *2015 10th international conference on P2P, parallel, grid, cloud and internet computing (3PGCIC)* (pp. 268-272). IEEE.

[26] Zhang, Y., Zheng, Z., & Lyu, M. R. (2014). An online performance prediction framework for service-oriented systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*(9), 1169-1181.

[27] Wong, T. S., Chan, G. Y., & Chua, F. F. (2018, May). A Machine Learning Model for Detection and Prediction of Cloud Quality of Service Violation. In *International Conference on Computational Science and Its Applications* (pp. 498-513). Springer, Cham.

[28] Singh, S., & Chana, I. (2015). Q-aware: Quality of service based cloud resource provisioning. *Computers & Electrical Engineering*, *47*, 138-160.

[29] Singh, S., Chana, I., & Buyya, R. (2017). STAR: SLA-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing*.

[30] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50-58.

[31] Wu, Q., Zhang, X., Zhang, M., Lou, Y., Zheng, R., & Wei, W. (2014). Reputation

revision method for selecting cloud services based on prior knowledge and a market mechanism. *The Scientific World Journal*, *2014*.

[32] Uriarte, R. B., Tsaftaris, S., & Tiezzi, F. (2015, May). Service clustering for autonomic clouds using random forest. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 515-524). IEEE.

[33] Hemmat, R. A., & Hafid, A. (2016). SLA violation prediction in cloud computing: A machine learning perspective. *arXiv preprint arXiv:1611.10338*.

[34] Zhu, J., He, P., Zheng, Z., & Lyu, M. R. (2017). Online QoS prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, *28*(10), 2911-2924.

[35] Khan, H. M., Chan, G. Y., & Chua, F. F. (2016, January). An adaptive monitoring framework for ensuring accountability and quality of services in cloud computing. In *2016 International Conference on Information Networking (ICOIN)* (pp. 249-253). IEEE.

[36] Zhou, Z., Abawajy, J., Chowdhury, M., Hu, Z., Li, K., Cheng, H., ... & Li, F. (2018). Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms. *Future Generation Computer Systems*, *86*, 836-850.

[37] Joseph, P. J., Vaswani, K., & Thazhuthaveetil, M. J. (2006, February). Construction and use of linear regression models for processor performance analysis. In *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.* (pp. 99-108). IEEE.

[38] Adhikari, R., & Agrawal, R. K. (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

[39] Jules, O., Hafid, A., & Serhani, M. A. (2014, October). Bayesian network, and probabilistic ontology driven trust model for sla management of cloud services. In *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)* (pp.

77-83). IEEE.

[40] Uriarte, R. B., Tsaftaris, S., & Tiezzi, F. (2015, May). Service clustering for autonomic clouds using random forest. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 515-524). IEEE.

[41] Kulyk, V., & Sossa, R. (2018). Determining the tourist attractive regions by GIS analysis using heatmaps. *Geodesy and Cartography*, *44*(1), 22-27.

[42] Pryke, A., Mostaghim, S., & Nazemi, A. (2007, March). Heatmap visualization of population based multi objective algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 361-375). Springer, Berlin, Heidelberg.

[43] Roa Rodríguez, R., & Lundin, R. (2016). Heatmap Visualization of Neural Frequency Data.

[44] Walt, S. V. D., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, *13*(2), 22-30.

[45] Jones, E., Oliphant, T., & Peterson, P. (2001). SciPy: Open source scientific tools for Python.

[46] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*(Oct), 2825-2830.

[47] Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, *18*(1), 559-563.

[48] Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2010). Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on services computing*, *4*(2), 140-152.

[49] Zheng, Z., Ma, H., Lyu, M. R., & King, I. (2012). Collaborative web service qos prediction via neighborhood integrated matrix factorization. *IEEE Transactions on Services Computing*, *6*(3), 289-299.

[50] Su, K., Xiao, B., Liu, B., Zhang, H., & Zhang, Z. (2017). TAP: A personalized trust-aware QoS prediction approach for web service recommendation. *Knowledge-Based Systems*, *115*, 55-65.

[51] Reiss, C., Wilkes, J., & Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, 1-14.

[52] Orta, E., Ruiz, M., Hurtado, N., & Gawn, D. (2014). Decision-making in IT service management: a simulation based approach. *Decision Support Systems*, *66*, 36-51.

VITA AUCTORIS

NAME:	Saurav Subhash Agarwal
PLACE OF BIRTH:	West Bengal, India
YEAR OF BIRTH:	1993
EDUCATION:	Bachelor of Engineering, 2011-2015 Gujarat Technological University, Ahmedabad, Gujarat, India Master of Science in Computer Science, co-op, 2018-2020 University of Windsor, Windsor, ON