UTS UNIVERSITY OF TECHNOLOGY SYDNEY

# Supporting consumer decisions in cloud computing using trust model

**by Muhammad Raza**

Thesis submitted in fulfilment of the requirements for the degree of

**Doctor of Philosophy**

under the supervision of Associate Professor Farookh Khadeer Hussain

University of Technology Sydney
Faculty of Engineering and Information Technology

May 2021

# CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Muhammad Raza declare that this thesis, is submitted in fulfilment of the requirements for the award of PhD, in the School of Computer Science, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 26-04-2021

# Acknowledgements

I would like to express my sincere gratitude to my principal supervisor A/Prof. Farookh Khadeer Hussain for his continuous support, guidance and encouragement during my Ph.D studies. His generosity and patience always provided me with extra energy throughout difficult times. I would also like to thank my co-supervisor A/ Prof. Zia ur Rehman for his help and support. His presence and cooperation allowed me to continue and complete my Ph.D journey.

I specially thank A/Prof. Omar Khadeer Hussain for his guidance, support and untiring willingness to help that kept me continue my research journey. I would also like to express my appreciation to Mr. Ming Zhao for his excellent support and helpful discussions.

I would like to extend my gratitude to my friends at UTS, Dr. Quynh Do, Dr. Supannada Chotipant, Dr. Mohammad Ikram and Dr. Omar Alshaweesh whose company and encouragement during my Ph.D journey have been invaluable.

Finally, I would like to dedicate this thesis to my family. My father, who is not with us to see this accomplishment and my mother, whose passion for reading and learning is the core motivation for my academic choices. I would like to thank my brother and sister for being with me throughout the difficult times and my uncle for his kind support.

# List of Publications

**Journal Articles**

1. Raza, M., Hussain, F. K., Hussain, O. K., Zhao, M., & Rehman, Z. u. (2019). A comparative analysis of machine learning models for quality pillar assessment of SaaS services by multi-class text classification of users' reviews. Future Generation Computer Systems, 101, 341-371. (***JCR(Q1), Impact Factor (6.125)***)
2. Raza, M., Hussain, F.K., Hussain, O.K. et al. Imputing sentiment intensity for SaaS service quality aspects using T-nearest neighbors with correlation-weighted Euclidean distance. Knowl Inf Syst 63, 2541–2584 (2021). ***JCR(Q2), Impact Factor (2.936)***)

# Table of Contents

# List of Figures

# List of Tables

# Glossary of terms

In this section, I present the key definition of the terms used in this chapter and rest of the thesis.

**Cloud computing**

Cloud computing is a computing paradigm in which the computer resources are virtualized and are accessed over the internet as web services.

**Cloud service**

A virtualized computing resource that is accessed over the internet.

**Cloud service provider**

An entity that owns and provides cloud services.

**Cloud service customer**

An entity that uses cloud services. The terms cloud service customers, cloud service users and cloud service consumers are used interchangeably in this thesis.

**Cloud services delivery model**

The way a cloud service is delivered to the cloud customer i.e., as a virtualized infrastructure (Infrastructure as a Service), as a virtualized application development and deployment platform (Platform as a Service) and as a virtualized software (software as a Service).

**Software as a Service**

A shared virtualized software that is used by the cloud service customers over the internet.

**Cloud service selection**

The process of choosing cloud services from the available services.

**Well-architected framework**

Frameworks proposed by Amazon web services (AWS, 2018) and Microsoft Azure (Azure, 2018) which are based on architectural best practices for cloud application design and operations.

**Service factors**

The key factors for the optimal design and operations of cloud applications and services. The terms service aspect, service factor, trust factor, software quality pillars and pillars of well-architected framework are used interchangeably in this thesis.

**Customer review**

Textual feedback by a cloud service customer based on previous experience.

**Label**

Representation of a textual data into a given category.

**Data pre-processing**

The process of manipulating a textual data into an acceptable format as input for the next stage.

**Ensemble model**

A technical combination of two or more machine learning models.

**Sentiment intensity**

The strength of cloud service customer's opinion.

**Trust**

The belief in an entity.

**Trust model**

A formal representation of the belief in an entity. In this thesis, it is the belief in a Software as a Service.

**Trustworthiness level**

The linguistic representation of the level of belief in a Software as a Service.

**Trust value**

A numerical representation of the level of belief in a Software as a Service.

**Missing data**

An unobserved data during the data collection stage.

**Imputation**

The process of inserting a new value for an unobserved value of a given variable.

**Inference**

To deduce a missing observation by learning from the existing evidence following a reasoning technique.

**Threshold**

A numerical acceptance level for a given measurement.

**Time spot**

The instance of time for which an observation or forecast is referred to.

**Time slot**

An interval of time for which a set of observations or forecasts are divided.

**Time stamp**

A date or time label associated to a time spot.

**Forecasting**

The process of predicting a value for a given time spot in the future.

# Abstract

In recent years, cloud computing has gained much attention in research. Compared to its predecessor computing paradigms, cloud computing is more complex in its infrastructure and the nature of services it provides to its customers. Customers are reluctant to adopt cloud computing because of the ongoing issues such as data security, privacy and highly abstract nature of the cloud services. There is a need to facilitate consumer decisions when it comes to cloud services selection. Trust can play an important role in bringing the consumers closer to cloud computing as it has been successfully implemented in several domains. The complex nature of cloud services specially, the Software as a Service (SaaS), brings new challenges for trust modelling compared to the trust models in other domains. In the existing literature, there are a number of approaches proposed to model the trust in cloud services. However, the majority of existing research is focused on establishing the concept of trust and to assess the willingness of cloud service providers. Trust is multi-faceted, dynamic and is based on multiple factors. Besides cloud service providers willingness, the existing literature does not consider the SaaS architectures and their dependencies in their trust models. Besides that, computing trust in SaaS requires the information regarding all the trust factors. There is no approach in the existing literature to address the issue of missing information in the trust factors. Furthermore, the existing literature has not focused on forecasting the trust factors for the future time spots which is important in boosting customers confidence in the adoption of SaaS. The computation of trust in SaaS must adopt to time and should be reliable. The existing trust models for cloud services do not provide comprehensive solution that captures both dynamicity and reliability of trust factors.

Based on the shortcoming mentioned above, in this thesis I propose and develop a comprehensive trust management framework for SaaS. The proposed framework identifies the key trust factors from SaaS architectural best practices. Each component of the framework is designed to address the issues faced by trust models for SaaS. In the first component of the framework, an ensemble machine learning approach automatically categorizes and extracts the trust factors from SaaS customer reviews. An intelligent Threshold-based Nearest Neighbour (T-NN) method is proposed to impute the missing customer sentiments in SaaS service factors. The third component of the framework is designed to forecast the values of the SaaS service

factors. The forecast model is designed through a comprehensive study of the traditional and latest forecasting approaches. Finally, the last component of the framework, models the trust using an intelligent and dynamically weighted fuzzy trust model. Each component of the proposed trust management framework is validated against several well-known approaches during experiments.

This thesis will help and facilitate cloud services consumers in their decisions before utilizing cloud services and also the SaaS providers to improve different aspects of their offered services.

# Chapter 1: Introduction

## 1.1 INTRODUCTION

Cloud computing is a modern computing paradigm and is the future of server-based computing. Virtualization is the key concept behind cloud computing where computing resources are accessed virtually over the internet. The virtual computing resources on the cloud are made accessible through web services. These cloud services are delivered in the form of Infrastructure as a Service, where the infrastructural resources are delivered virtually, Platform as a Service for application development and deployment and Software as a Service, where software are designed under multitenant architecture and made accessible virtually. The dynamic and elastic nature of cloud computing brings a lot of flexibility and benefits to the cloud users. Some of the benefits include cost effectiveness, less overhead in resource management, ease in upgrade and deployment. Beside all these benefits, cloud computing also brings several challenges and risks to the cloud market.

Software as a Service is one of the main cloud service delivery models and is receiving more attention from customers with time. It is crucial for the Software as a Service customers to make informed decisions when selecting the most suitable service for their businesses and a certain level of trust in the SaaS products help in their decisions. In this chapter, I introduce the importance and the key issues related to trust in Software as a Service.

In the next section, I provide an overview of cloud computing and main service delivery and deployment models. In Section 1.3, the key challenges in cloud computing are discussed. In Section 1.4, I present the trust related issues in cloud services. The objectives of the thesis are described in Section 1.5, followed by the scope of the thesis in Section 1.6. In Section 1.7, I explain the significance of the thesis and present the plan of the thesis in Section 1.8. Section 1.9 concludes this chapter.

## 1.2 OVERVIEW OF CLOUD COMPUTING

In this section, I provide an overview of cloud computing that includes the evolution of computing and the key concepts. I also discuss the cloud service deployment and delivery models.

### 1.2.1 What is cloud computing?

Cloud computing is the future of modern computing paradigm due to the innovative system architecture. The core concept of cloud computing is based on its predecessor technologies and service models namely Grid computing, Utility computing and Autonomic computing. Cloud computing can be seen as a merge of different deployment models and service models (Zissis & Lekkas, 2012). Moreover, cloud computing is also the product of the increase trend towards external deployment of Information and Telecommunication Services (ITS) resources (Stanoevska-Slabeva, Wozniak, & Ristol, 2009).

The emergence of cloud computing as a computing model enables ubiquitous, convenient and on-demand network access to shared pool of configurable computing resources such as, networks, computing services, storage, and applications services. These resources are provisioned rapidly with less service provider interaction and released with minimum management efforts (Mell & Grance 2011). Figure 1.1 depicts the conceptual reference model of cloud computing with the major actors along with their associated activities and functions in cloud computing.

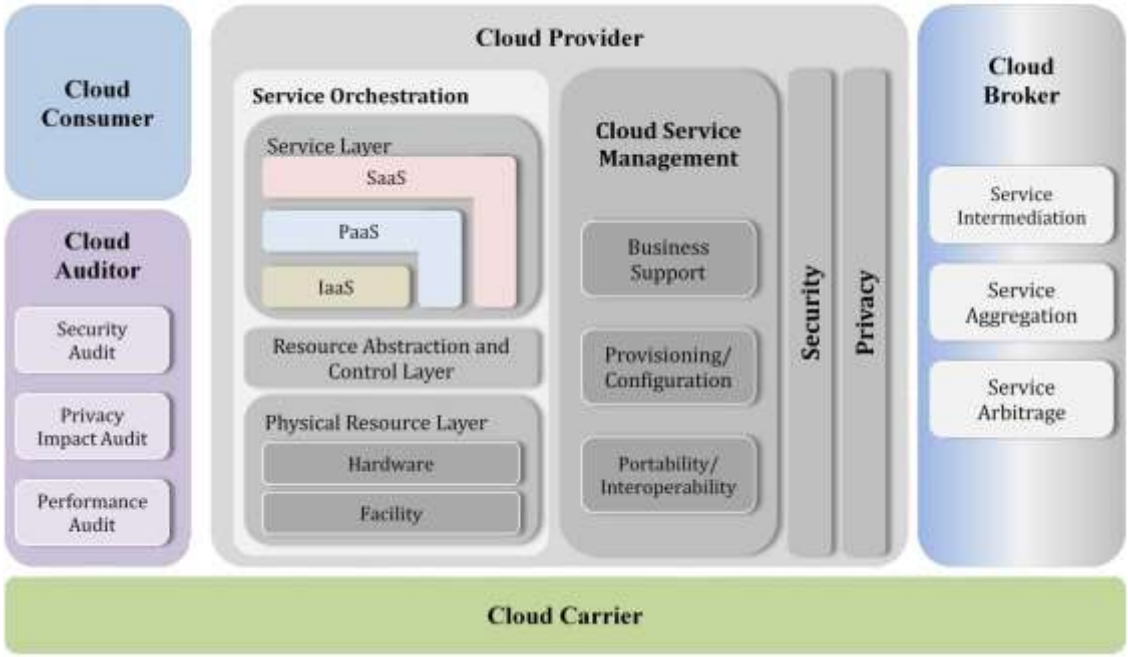



Figure 1.1 Cloud computing conceptual reference model (Liu et al., 2011)

Cloud services are provided and managed by businesses called the Cloud Providers. On the other hand, these cloud services are paid for and used by individuals or businesses called the Cloud Consumers (or Cloud Customers).

Cloud computing brings several benefits for the customers that includes reduced cost of the computing resources. Dynamic resource sharing is one key factor in bringing the cost of cloud services down. The pay-per-use and pay-as-you go cost model of cloud service makes it affordable for large as well as small businesses and start-ups. On the provider side, the management of the services become easier and efficient in rolling out of new services and service features.

### 1.2.2 Cloud services deployment models

Cloud computing infrastructures are deployed and operated under one of the following deployment models (Liu et al., 2011):

- Public cloud
- Private cloud
- Community cloud
- Hybrid cloud

The basic cloud service deployment models differ from each other on the basis of several aspects of its physical architecture. From geographical isolation to their interoperability, they bring a lot of crucial issues regarding security, privacy and trust among all the others. In a private cloud the infrastructure is provisioned for exclusive use by single organization whereas in a public cloud the infrastructure is for open use by general public (Liu et al., 2011). In a community cloud the infrastructure is owned, managed and operated by one or more organizations. These organizations as a group are called as communities which are based on their shared concerns. A hybrid cloud is a composition of two or more distinct cloud infrastructures.

### 1.2.3 Cloud services delivery models

Cloud services offerings are categorized under three service models (Mell & Grance 2011):

- Software as a Service (SaaS)
- Platform as a Service (PaaS)
- Infrastructure as a Service (IaaS)

In Software as a Service model, a SaaS provider offers an application to the consumers in a multi-tenant architecture. SaaS can be accessed over the internet using a web browser or application specific clients. In a shared responsibility model, most of the responsibilities of managing the hardware infrastructure and applications is assumed by the SaaS provider. The SaaS provider is responsible for

the deployment, configurations, maintenance and updates of the software application. On the other hand, the SaaS consumers have very limited control over the administration of the applications. Some of the well-known examples include Salesforce, Microsoft Office365 and SAP Concur.

In Platform as a Service model, the consumers deploy their applications on the platform provided by the PaaS providers. The platform may support applications developed in different programming languages by providing with related libraries, runtime environments and tools. In a shared responsibility model, the PaaS provider have control in managing the platform infrastructure and the environment settings. The PaaS provider is responsible for managing different components of the platform such as the middleware components, runtime software execution stack and databases. Whereas, the PaaS consumer have control over the applications and limited environment settings apart from the PaaS hardware and virtualization infrastructures such as hardware devices, servers, operating systems, networks and hypervisors. The well-known examples are AWS Elastic Beanstalk, Google AppEngine, Heroku and Microsoft Azure.

In Infrastructure as a Service model, the IaaS provider provides virtual computation capabilities to provision resources such as computing, storage, network and operating systems. In the shared responsibility model of IaaS, the IaaS provider maintains control over the hardware and virtualization infrastructures but grants more control and access to fundamental resources such as operating systems, storage and network components to the IaaS consumer. IaaS providers maintain control to the hardware resources such as servers, storage and network infrastructures. Some of the well-known IaaS are Amazon Web Services, Microsoft Azure and Google Cloud Platform.

## 1.3 KEY CHALLENGES IN CLOUD COMPUTING

Beside all the benefits of cloud computing, a number of challenges is also associated with cloud services due to its complex and remote infrastructure. From customer point of view, the main obstacles for the acceptance of cloud services are their abstract and distributed nature. Some of the key challenges related to cloud service are briefly explained below.

### 1.3.1 Privacy

One of the basic concerns on the customer side is regarding the privacy of their data and applications on cloud services. Informational privacy (Martucci et al., 2012) (Fischer-Hübner, 2001), is referred to as the rights of a person to determine how and when their personal information is communicated to the others and to what extent. The issue of privacy of customer assets being outsourced on the

cloud is based mainly on the deployment methods of the cloud services. Customer data being outside the organization boundaries raises the confidentiality questions of how and to what extent their information is exposed outside their organization. Keeping this in mind, some of the organizations still allow less sensitive organizational data to be transferred to the cloud (Martucci et al., 2012). Privacy of customer information on the cloud brings reluctance in the adoption of cloud services by cloud customers.

### 1.3.2 Security

The nature of cloud service and their deployment architecture is different and relatively more complex compared to the traditional computing paradigm, peer-to-peer computing, grid computing and distributed computing. Due to the complex nature of the cloud service architecture, the effectiveness and of traditional security mechanisms is not suitable for cloud services (Zissis & Lekkas, 2012). One of the main concerns on the customer side that leads to hesitation in the adoption of cloud services is related to the loss of control over data and computation.

In the context of cloud computing, each service model discussed in the previous section, exposes different aspects of the service and creates vulnerable attacking surfaces for the adversaries (Liu et al., 2011). In IaaS model, the virtual machines share the same host and are managed by hypervisors. Similar is the case of customer data on the shared storage devices on the remote host servers. This brings security challenges regarding the virtual machines, hypervisors, storage devices and storage management software applications. In case of SaaS, the security and integrity of multitenant software applications accessed over the internet connections through client web browsers creates relatively different vulnerable entry points. Where, customer data at rest and on transit, both creates a risk in security breach. Similar security issues arise with the deployment method of cloud service such as in public clouds, where multiple tenants share the same resources. The solutions for security related issues can be broadly categorized under the concepts such as authentication, authorization, monitoring and encryption. Therefore, the security and integrity of customer data and application creates another hurdle in the adoption of cloud services.

### 1.3.3 Trust

Trust and security are two different concepts and extensively studied in different research areas. In traditional computing paradigm, security is related to the physical protection of resources such as servers, network infrastructures, storage devices and software. Whereas in cloud computing, the

concept of security is associated with the protection of software components in virtual information technology infrastructures and services.

On the other hand, trust is subjective in nature. It is not purely a technical issue rather in reality it is a social problem (X. Wu, 2012). Also known as Soft trust (Varadharajan, 2009), it is concerned with intrinsic human emotions and perception, interaction and exchange of information. Trust normally deals with the dependability of one party on another. It is a measurable belief in a behaviour, in a given context, for a specified period of time (Olmedilla, Rana, Matthews, & Nejdl, 2005). In cloud computing, trust plays a crucial part in the decision-making process of cloud service adoption and selection. Some of the key issues related to trust in cloud services specifically software as a service, are briefly discussed in the following section.

## 1.4 TRUST RELATED ISSUES IN SAAS

As discussed in the previous section, Software as a Service (SaaS) is one of the well-known types of cloud service delivery model (Liu et al., 2011). SaaS is software developed by the developers and unlike tradition software, it is delivered to the consumers in the form of web services. The development of the software (as a Service) may or may not follow the traditional software development models. The SaaS is generally deployed in one of the following two approaches: a) The whole service stack owned by the SaaS provider or; b) the application and service stack developed by the by the developers but is deployed on the infrastructure of other cloud service providers. In the second case, the performance and security of SaaS products are partially dependent on the offerings by the IaaS providers. It can be stated over here, that these IaaS providers have most in-depth and up-to-date information regarding the issues faced by the hosted SaaS on their infrastructure. From the above discussion, it becomes clear that the structure of SaaS is different from the other service models and brings extra complexity with their design. In order to establish trust in SaaS, the following issues arise:

### 1.4.1 Factors affecting trust in SaaS

Trust is multi-faceted and it is dependent on more than one factor. The concept of trust is being studied thoroughly in different domain and is modelled using different approaches. To establish trust in an entity, a number of factors are involved which in general, can be divided into two categories. 1) domain specific factors and, 2) the approach being used to model trust. Among the others, trust has been studied in several domains such as eCommerce (Audun Jøsang & Ismail, 2002), multi agent systems (Sabater & Sierra, 2002), social networks (Golbeck & Hendler, 2005), wireless sensor

networks (Ganeriwal, Balzano, & Srivastava, 2008) and cloud computing (Abawajy, 2009). A comprehensive review of trust in different domains is presented in Chapter 2. There are several approaches used to model trust in different domain such as reputation-based (Audun Jøsang & Ismail, 2002) (Xiong & Liu, 2003) (Pramod S. Pawar, Rajarajan, Dimitrakos, & Zisman, 2013), role-based (Huynh, Jennings, & Shadbolt, 2006), policy-based (Sheikh Mahbub Habib, Ries, Mühlhäuser, & Varikkattu, 2014), attribute-based (Manuel, 2015) and metric-based (E. D. Canedo, Junior, Carvalho, & Albuquerque, 2012).

In cloud computing, establishing trust using one service delivery model is different from the others, but there are dependencies as well. For example, the performance and efficiency of a SaaS product using storage components of another cloud infrastructure can be compromised because of its dependency on the disk Input-Output (I/O) capacity of the infrastructure provider. On the other hand, the performance and efficiency of the SaaS may be affected by the design of the SaaS and not due to the underneath infrastructure. To assess SaaS products and compute trust, it is crucial to identify the most reliable and relevant factors which is one of the main issues addressed in this thesis. A detailed review of the suggested factors to assess cloud services is presented in Chapter 2.

Furthermore, besides the identification of the relevant factors, it is also important to obtain effective information regarding the identified factors. In this thesis, I utilize the architectural best practices for the identification of most relevant and reliable factor and use customer sentiment towards each of these factors to model trust in SaaS. In the existing literature, there are number of factors used to establish trust but there is no such approach that uses the architectural best practices and customer sentiment to compute trust in SaaS. The identification and tagging of SaaS customer reviews under different SaaS service factors should be automated for future customer reviews. In literature, this problem is addressed as the text classification also known as text categorization. The detailed design and implementation of the solution for this research issue is presented in Chapter 5 of this thesis.

### 1.4.2 Missing information regarding trust factors

As discussed in the previous section, trust depends on multiple factors in SaaS and a complete trust level can only be computed with the information regarding all the factors. Missing information is very common issue faced in many research areas. Missing information can occur under several circumstances such as malfunctioning of sensors in industrial setups, Internet of things, unwillingness of respondents in surveys, human negligence or simply unavailability of certain factors in the data collection process.

The same is true in the modelling and computation of trust from multiple factors. There are circumstances in which the information regarding some of the trust factors are not available. In this thesis, as the main source of information for the trust model is based on customer text reviews, the customers simply do not highlight different factors of the SaaS products in their reviews. This results in the unavailability of information regarding useful service factors of SaaS which is helpful in the modelling and computation of trust. To facilitate customer decisions in SaaS products selection, the related information regarding all the service factors must be available which is used during the modelling and computation of trust levels. Modelling trust with partially available information could lead to unrealistic assumptions towards the SaaS trust levels.

In the existing literature there is no such approach that focuses on the missing sentiment intensity of service factors of SaaS products for trust computation. The details of the design and implementation of this solution is presented in Chapter 6 of this thesis.

### 1.4.3 Forecasting information regarding trust factors

Trust is time dependent and its value changes over time. In cloud computing, one of the aspects which continuous effect the cloud service is the economies of scale, which provide significant financial benefits for the providers and effects the overall quality of their service. Due to this benefit, the overall quality of service changes over time and the effects can be observed through the available information for each service factor. Another aspect that effects the cloud services is the ever-increasing number of SaaS products. This creates a competitive cloud marketplace where the services continuously bring innovations, added features, cost reductions and improved customer support. In this scenario, the performance of each service factor keeps on changing. The rapid changes in cloud related technologies are another aspect that effects the overall quality of SaaS products. Such improvement can be seen in new virtual storage technologies, improved virtual machine features and improved network performances. A trust model for SaaS products, should have the capability to capture and utilize the dynamicity in the cloud marketplace in trust modelling.

The SaaS customer on the other hand, prefer to use those SaaS products that maintains and constantly improve the quality of their product. A trust model should be able to provide forecasted trust values for future time spots. This trust forecasts help the customers in their decisions for long term contracts with SaaS providers. In the existing literature, no such approach exist that forecast the values for service factors for trust modelling. The details of the proposed forecast model are presented in Chapter 7.

### 1.4.4 Modelling trust for Software as a Service

As discussed in previously, the trust in SaaS depends on multiple factors and each factor has different contribution in the computation of trust score. Generally, the trust factors are prioritised based on customer preferences but the priorities can be captured dynamically and automatically by utilizing the relevant available information for the SaaS products. This is important in the modelling and computation of trust as the nature of trust is dynamic and it varies with time. The dynamic changes in the values of the trust factors should be captured by the trust model to compute the weights for each factor and rank them accordingly. This ranking helps in prioritizing the trust factors that are used to compute the final trust level of SaaS products. Such metrics is suitable for the dynamic nature of trust and the weights for each trust factor can vary with time that in effect changes the priorities among them. The detail design and implementation of the trust model is presented in Chapter 8 of this thesis.

## 1.5 OBJECTIVES OF THE THESIS

In the previous sections, I outlined the importance of trust in SaaS products and the issues related to the modelling and computation of trust in SaaS. The main aim of this thesis is to propose a trust model along with a solution framework that assist the SaaS customers in their decisions for the selection of SaaS products. For the solutions of the issues outlined in the previous sections, the objectives of the thesis are summarized as follows:

1. To develop an intelligent method that automatically identifies the key service factors from the SaaS products.
2. To develop a method for imputing the missing information regarding the identified service factors of SaaS products.
3. To develop a reliable method for forecasting the future values for the SaaS service factors.
4. To develop an intelligent and dynamic method that models the trust level of SaaS products.
5. To validate the developed techniques for the solution of the identified research issues in this thesis.

## 1.6 SCOPE OF THE THESIS

Cloud service are delivered using modes such as Infrastructure as a Service, Platform as a Service and Software as a Service. The solutions proposed in this thesis that includes the framework and its

constituent components are aimed only at solving the issues related to the Software as a Service cloud delivery model.

There are several key research issues that hinders the adoption of SaaS such as privacy, security and trust. This thesis is only focused on the trust related issues in SaaS and does not aim to directly address the other issues. There is a considerable work in the literature on trust modelling and trust computation in different application domain. However, the solutions designed in this thesis for different steps in the computation of trust is focused on SaaS only.

The solutions developed in this thesis are tested and implemented on the SaaS dataset collected and prepared only for this study.

## 1.7 SIGNIFICANCE OF THE THESIS

There is a significant growth in the adoption of cloud services specially the software as a service delivery model. Due to the increase in the quantity of the similar SaaS products, the need to make low-risk decisions in the adoption of these services has become more critical. To assist the SaaS customers in their decisions, the existing literature, to the best of our knowledge, does not take account of the architectural best practices in accessing the trustworthiness of the SaaS products. The core significance of this thesis lies in the above statement that, to assess and model the trust in SaaS products based on factors derived from the architectural best practices. The key significances of this thesis are summarized as follows:

1. To assist SaaS customer in their decisions, this thesis develops a comprehensive framework for assessing and modelling trust in SaaS products
2. This thesis develops an efficient model that classifies customer text reviews into key SaaS service factors
3. In the existing literature, the trust in cloud services is computed using many different factors, however, to the best of my knowledge, this thesis is the first to consider customer sentiment intensity towards the service factors in computing the trust.
4. It is less realistic to model trust in a SaaS product with missing information. In this thesis, I develop a high performing imputation model that imputes the missing sentiment scores for the SaaS service factors.
5. For better informed decision, it is helpful for the customers to be able to obtain the trust values of SaaS product in the future time spots. This thesis develops a forecast model as

part of the proposed framework, that forecasts the sentiment scores for SaaS service factors in the future.

6. In this thesis, as part of the proposed framework, I develop a trust model for the SaaS products that is based on the factors with dynamic weights.

## 1.8 PLAN OF THE THESIS

In this chapter I explained the importance of the trust model for SaaS. I presented my intention to develop a trust management framework that models the trust in SaaS products that assist the SaaS customers in their decisions for acquiring these services. I have highlighted the main objectives of this thesis in this chapter which are achieved and their details are arranged in the following chapters. The rest of the thesis is organized as follows:

**Chapter 2**: In Chapter 2, I provide an extensive review of the existing approaches for modelling trust in different domain including cloud computing. I present a critical analysis of the current literature on trust modelling and assessment of SaaS products. Based on the thorough review of the literature, I highlight the shortcoming in the existing literature related to trust in SaaS.

**Chapter 3**: In Chapter 3, I outline and define the research problem of this thesis that emerge from the shortcomings in the trust assessment and modelling in existing literature. I explain the key terminologies used in the domain of trust and cloud computing. I explain the research methodology that I have selected to following in this thesis. This chapter present the key research issues and a set of objectives that I have set to achieve in this thesis.

**Chapter 4**: In Chapter 4, I present an overview of the proposed trust management framework with a brief explanation of each of its components. The components of the framework and their associated tasks are designed to address and solve the issues highlighted in Chapter 3.

**Chapter 5**: This chapter presents an ensemble multi-class text classification model that is developed to categorize customer reviews into SaaS service factors. All the pre-processing of textual data is presented with detailed implementation steps.

**Chapter 6**: In this chapter, a detailed design and development of a proposed imputation model is presented that is developed to impute the sentiment intensity scores for the missing service factors.

**Chapter 7**: Chapter 7 presents a forecast model for forecasting the sentiment intensity values of service factors. The proposed model utilizes the historic sentiment time series data for SaaS products to forecast the values in the future.

**Chapter 8**: In this chapter, the design and development of the trust model for SaaS products is presented. The service factors are considered as trust factors for the trust model which are ranked using the proposed ranking weight metrics.

**Chapter 9**: Chapter 9 concludes the thesis and highlights the future research directions based on the learning and achievements during this thesis.

## 1.9 CONCLUSION

Software as a service is one of well-known cloud service delivery model. There has been an increasing trend in the adoption of SaaS and its preference has increased over the on-premises standalone software solutions. However, the customers are still reluctant in acquiring SaaS for their business due to ongoing issues such as trust. The customers have a choice to select SaaS products offered by variety of providers. However, customers need to make informed decisions when selecting SaaS products. In comparison with the SaaS providers, the customers do not have the visibility into the SaaS deployed architecture. In this thesis, I aim to develop a framework that models the trust in SaaS products that is based on the key service factors. In this chapter, I explained the key challenges in the adoption of cloud service specially SaaS. I discussed the research issues that relates to the trust in SaaS. I highlighted the main objectives of this thesis and discussed the scope and limitation of this thesis. In this chapter I outlined the significance of this thesis and provided the plan of the thesis. In the next chapter, I will discuss in details the current literature relevant to cloud services and provide a critical evaluation of existing literature on trust in SaaS.

# Chapter 2:
# Literature Review

## 2.1 INTRODUCTION

In this chapter, I present an in-depth background study that is focused on trust, the trust assessment and modelling theories and applications. Section 2.2 presents an overall understanding to the concept of trust that includes formal trust definitions, trust modelling approaches and the well-known trust models in different application domains. In Section 2.3, a comprehensive survey of the existing literature on trust modelling in cloud computing is presented followed by a discussion on the importance of trust factors in Section 2.4. In Section 2.5, I present a critical analysis of the existing fuzzy-based trust models in cloud computing. A comprehensive critical analysis of the trust model for cloud computing is presented in Section 2.6 along with a summary of the emerging gaps in existing literature. Section 2.7 concludes the chapter.

## 2.2 TRUST ASSESSMENT AND TRUST MODELS

Trust is a mature concept and there are several formal definitions of trust in existing literature. It is modelled and measured using a number of theories and approaches for different application domains. In the following subsection, I present some of the well-known definitions of trust, followed by the trust modelling theories and approaches and the application of trust in different domain.

### 2.2.1 Definition of Trust

One of the well-known definitions of trust by McKnight and Chervany (1996) defines trust as "the extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible". The definition provided by McKnight and Chervany (1996) can be considered as generic. The key highlights of this definition suggest that trust has an 'extent' which can be measured, it is on the 'will' of one party which indicates its subjectiveness and it is situational, indicating that it is context specific.

Another definition of trust in the context of service-oriented environment is presented by Olmedilla et al., (2005) which states that "Trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to

service X)". The definition of trust by Olmedilla et al., (2005) also suggests that trust is a measurable belief, it is subjective, 'time dependent' indicates its dynamicity and also context dependent.

Table 2.1 Trust modelling approaches

| Modelling Approaches | References |
|---|---|
| Probability Theory | (Schryen et al., 2011), (Hang & Singh, 2011), (Xiao et al., 2010) |
| Bayesian | (Buchegger & Le Boudec, 2003), (Wang & Vassileva, 2003), (Ries, 2009a), (Whitby et al., 2004), (Jøsang & Ismail, 2002), (Hang & Singh, 2011), (Habib et al., 2011) |
| Role-based | (Huynh et al., 2006), (Billhardt et al., 2007) |
| Propagation Model | (Levien, 2009) |
| Subjective Probability | (Gambetta, 2000), (Habib et al., 2013), (Habib et al., 2014), (Qiang et al., 2011), (Habib et al., 2011), (Pawar et al., 2013), (Josang, 2009), (Zhou et al., 2011), (Habib et al., 2012) |
| Computational Model | (Marsh, 1999) |
| Negotiation Model | (Winslett et al., 2002), (Lee et al., 2009) |
| Behavioural Model | (Castelfranchi & Falcone, 2010) |
| Policy Based | (Habib et al., 2013) |
| Attribute-based | (Li & Du, 2013), (Manuel, 2015) |
| Cryptography | (Zissis & Lekkas, 2012) |
| Metric based | (Canedo et al., 2012), (Manchala, 2000) |
| Fuzzy logic | (Wu, 2012) |
| Domain-based | (Yang et al., 2010) |
| Reputation-based | (Xiong & Liu, 2003), (Huynh et al., 2006), (Limam & Boutaba, 2010), (Levien, 2009), (Abawajy, 2009), (Kamvar et al., 2003), (Whitby et al., 2004), (Sabater & Sierra, 2002), (Ganeriwal et al., 2008), (Jøsang & Ismail, 2002), (Kinateder et al., 2005), (Resnick & Zeckhauser, 2002), (Macías & Guitart, 2012), (Wu, 2012), (Pawar et al., 2013), (Xiao et al., 2010) |

### 2.2.2 Trust modelling approaches

Trust has been modelled using different approaches in different domain. The Trust Models are classified as, PKI Based Trust Model; Feedback Creditability based Global Trust Model, Behavioural Based Trust Model, Subjective Trust Model and Domain-based Trust Model (Saravanakumar & Arun, 2014). Similarly, Kanwal et al., (2013) have classified the trust models into five categories namely, Agreement based trust models, Certificate/secret key based trust models, Feedback based trust models, Domain based trust models and Subjective Trust models. Trust has also been assessed in literature based on entities behaviour. Entities behaviour is captured in the form of their reputation, intentions, capabilities and competencies which can be measured in terms of their trust values (Habib et al., 2012).

The entities willingness, trust relationship, trust propagation, prediction and the uncertainty in trust using the well-known trust modelling approaches illustrated in Table 2.1.

### 2.2.3 Trust in different domain

Regardless of methods and approaches through which trust is modelled and implemented, the concept gained a huge attraction in solving the issue of dependability among entities in traditional computing paradigms. Table 2.2 shows the well-known trust models in different application domains where Trust is applied successfully to bring more understanding among entities.

Table 2.2 Trust in different application domain

| Applied Domain | References |
|---|---|
| SOE | (Schryen et al., 2011), (Billhardt et al., 2007), (Hang & Singh, 2011) |
| eCommerce | (Jøsang & Ismail, 2002), (Xiong & Liu, 2003) |
| P2P | (Wu, 2011), (Wang & Vassileva, 2003), (Kamvar et al., 2003), (Kinateder et al., 2005) |
| Mobile Ad-hoc Networks | (Buchegger & Le Boudec, 2003) |
| MAS | (Sabater & Sierra, 2002), (Xiong & Liu, 2003), (Huynh et al., 2006), (Wang & Singh, 2007) |
| Distributed Systems | (Schryen et al., 2011), (Beth et al., 1994) |
| Online Services | (Jøsang et al., 2007) |
| File Sharing | (Wang & Vassileva, 2003) |
| Web based Social Networks | (Golbeck & Hendler, 2005), (Hang et al., 2009) |
| Cloud Computing | (Abawajy, 2009), (Naseer et al., 2014), (Habib et al., 2011), (Habib et al., 2013), (Habib et al., 2014), (Li & Du, 2013), (Manuel, 2015), (Macías & Guitart, 2012), (Qiang et al., 2011), (Canedo et al., 2012), (Wu, 2012), (Pawar et al., 2013), (Yang et al., 2010) |
| Grid computing | (Lin et al., 2004) |
| Wireless Sensor Networks | (Ganeriwal et al., 2008) |
| Ubiquitous Computing | (Ries, 2009b) |

Table 2.2 shows that the concept of trust is very important in bringing a level of certainty in transactions among entities regardless of the domain it is used. The concept of trust has also been studied in the context of cloud computing. As discussed previously, cloud computing has a complex architecture and there is a great deal of hesitation on the cloud consumers side in adopting this modern and cost-effective computing paradigm. A comprehensive discussion and analysis on the trust assessment and trust models in cloud computing is presented in Section 2.3.

Besides its importance in research domain, it has been successfully adopted in several e-commerce companies like eBay, Amazon and other app markets for mobile applications (Habib et al., 2011).

## 2.3 TRUST ASSESSMENT AND TRUST MODELS IN CLOUD COMPUTING

Modelling trust includes modelling real-world social trust characteristics (Abdul-Rahman & Hailes, 2000). As discussed in the previous section, the concept of trust has been studied in several domains such as, Service Oriented Environments, Multi-agent systems and Information systems. Recently, trust modelling has also been used in cloud environment to tackle the ongoing issues of trustworthiness in cloud services. Trust modelling faces several challenges due to the complex nature of cloud computing. The non-transparent nature of cloud computing raises some important issues that needs to be addressed while modelling trust in cloud computing.

To identify and analyse issues related to security, privacy, trust and interoperability in cloud computing, a survey is conducted by (Saravanakumar & Arun, 2014). Their survey considers security as the key parameter to secure customers data on the cloud. The security issues in cloud computing have been classified as, Security Issues, Risk Management issues, access level issues and service composition and evaluation plan.

Privacy issues, Trust issues, inter-cloud and intra-cloud standards of cloud interoperability have been highlighted by Saravanakumar and Arun (2014) to identify the challenges during the cloud interaction. Their survey focuses on cloud interoperability, security, privacy and trust based on standards and guidelines and analysed.

They have also classified the issues related to SLAs as, Resource Access Level, Billing Level, SLA Level, Log Level, CSP Level, Separation Level, Border Level, Data and Resource Migration Level, Bandwidth Level and Customer Level.

One of main focus of their survey is on the interoperability among different cloud service providers for effective interaction by maximizing the QoS of cloud computing. The survey discusses the Cloud Interoperability Standards into two different contexts i.e., in internal cloud management, design communication standards and inter cloud interoperability and federation framework. The internal cloud management standards are, Distributed Management Task Force, Storage Networking Industry Association and OGF Standards. The Intercloud Interoperability and Federation Framework is described as, ITu-T Focus Group on Cloud Computing, NIST Cloud Computing related Standards and ITU-T Cloud Network Infrastructure Models.

To study the importance of trust in cloud computing, Uusitalo et al. (2010) have also conducted a semi-structured qualitative expert interview to find out what trust related aspects the experts deem noteworthy when surrendering organization's computing services into foreign hands. The total

number of participants was 33 and majority of the participants belong to technology domain. The research questions which were considered included, aspects that contribute and affect trust in cloud computing, experience of trust or lack of trust in cloud and how to measure trust in cloud.

According to the participants the Non-functional factors that affect the trust in cloud services include, Brand (reputation, brand, image, history and name), auditing and agreement, own experience, critical mass of user, price, customer care, search engine results others including outsourcing, size of customer and multi-tenancy. The functional aspects that the participants thought to be effectors of trust include, privacy and security, transparency and reliability, user experience, good functionality and interoperability, availability and resources and language.

The second question of the interview was regarding the experience of trust or lack of trust in cloud. The non-functional aspects mentioned by the participants are, Brand, reputation background of company, auditing, SLAs and time. The functional aspects were usefulness of services, tailor ability, user experience, Security and privacy, transparency and reliability and complexity of the environment. The results of the interview conducted by Uusitalo et al. (2010) suggested that, brand, reputation, image, history and name were the most important effectors while privacy and security were the most important functional aspects that enforce trust in cloud.

Besides the benefits of cloud computing such as, reduced cost, dynamic resource sharing, pay-per-use, faster time to roll out new services and dynamic resource availability, it also brings possible threats and risks involved in using cloud computing. These threats and risks include threats to security, threats to privacy, lack of trust, lack of identity management solutions for federated clouds (Federated Identity Management FIM), lack of latency a bandwidth guarantees, weak service level agreements, lack of standards and interoperability, lack of customer support, perceived lack of reliability and absence of independent quality assurance body (Habib et al., 2010). In their work, Habib et al. (2010) propose a new research direction regarding trust and reputation system in selecting trustworthy cloud providers. They suggest some recommendations regarding trustworthy cloud service providers such as an independent mediation layer and to stop the manipulation of evaluation process the evaluation framework should be trusted. Moreover, they suggest that the evaluation of the cloud service providers should be based on fine-grained QoS parameters together with feedbacks from consumers, recommendation and other specific parameters related to the cloud computing environment.

The complex nature of the cloud infrastructure makes the users reluctant in choosing the suitable and trustworthy cloud service provider. They are not sure who has access to their data and to what extent (Zissis & Lekkas, 2012). Consumer data is stored in data centres that are geographically distributed and services could be hosted at multiple sites managed and hosted by multiple cloud service providers. Mostly, cloud service providers in the Service Level Agreements with the consumers, only present the service description and technical specification of their services (Zissis & Lekkas, 2012). One such example is the telecommunication industry. Beside privacy and security, trust is considered as key requirement in the adoption of cloud computing for the telecommunication industry (Martucci et al., 2012). Trust requirements are categorised into hard trust such as certificates, audits and secure hypervisor, and soft trust such as interactions and exchange of experience, loyalty and perceptions. These trust requirements are very helpful in addressing the challenges associated in integration of cloud services with the telecommunication infrastructure.

To identify the important aspects that shape user trust in cloud service providers specially on data storage, Rashidi and Movahhedinia (2012) have proposed a model for user trust. They have highlighted the issue of low trust on cloud service providers especially when it comes to critical data storage. The authors have recognized 8 risk elements and suggest that cloud service providers should devise a mechanism to eliminate these risks. Rashidi and Movahhedinia (2012) conducted a survey based on these important elements that shape user trust on cloud service providers. Their survey is based on a self-administered questionnaire which considers eight elements. The selected elements include, Data location, Investigation, Data segregation, Availability, Long-term viability, Regulatory compliance, Backup and recovery and Privileged user access. The questionnaire was distributed to cloud users online and 72 of the credible questionnaires were collected. Statistical analysis of all the 8 elements of the questionnaire were performed in SPSS and using AMOS. The results of their survey show that Backup and recovery has the strongest impact on user's trust while the Data segregation and Investigation have the weakest impact. Their study is mainly focused on the theoretical perspective on trust without any real-life applicability and evaluation.

Similarly, due to the geographical distribution of cloud resources, a centralized trust and reputation framework for the developers in social cloud applications is proposed by Moyano et al. (2013)**.** Their proposed framework is focused on security concerns due the infrastructure of the cloud computing that includes heterogeneity, geographical distribution of resources, multiple providers and unknown identity of the providers. In social cloud applications the users act as service providers, which raises a lot of security and trust concerns. Cloud resources are distributed across heterogeneous and

geographically separated locations and are being managed by several providers who often are anonymous to each other. In their study (Moyano et al., 2013), trust and reputation are considered to be beneficial in these scenarios and can help to address security concerns. They suggest that a holistic approach in which both trust and reputation requirements are considered to assist developers.

To find efficient and trustworthy cloud service providers, Naseer et al. (2014) proposed a trust model that considers the data from a regulatory authority, performance of cloud service provider for the last one year and customer feedback. Their proposed model considers attributes such as downtime, uptime, fault tolerance capability, application update frequency and customer support experience. The authors suggest high priority for the quality of service over the security measurements will be more suitable to the users. The final score is computed using the weighted attributes. However, Naseer et al. (2014) have not provided information regarding the sources of the data being and the name(s) of the regulatory authority and how have they collected and analysed customer feedbacks. The approach is simplistic in nature and considered the physical attributes related to hardware.

To address the issue of trust in cloud computing marketplaces, Habib et al. (2014) proposed a multi-faceted architecture that considers multiple attributes and CAIQ as the source of information regarding the cloud service providers. Their proposed model is able to identify trustworthy cloud service providers in terms of attributes such as compliance, data governance and information security. Their proposed trust management system is able to combine multi-attribute-based trust from multiple sources under uncertainty. The operators used to combine those attributes are considered as equivalent to subjective logic and compatible with the standard probabilistic approach.

There are limitations to their proposed approach such as the CAIQ information from the STAR repository is not classified according to the types of the services. Moreover, synthetic dataset is used for the validity and applicability of the approach. The second limitation is the inconsistencies found in the completed CAIQs such as, blank assertions by the cloud service providers. Trust factors extracted from sources such as CAIQ, raises the key issues of reliability and dynamicity in the reported documents by the providers. Trust is a time dependent concept and the trustworthiness in cloud services changes over time with the changes in each factor on which trust is based. Previously, Habib et al. (2011) have argued that the trustworthiness of cloud service providers should be assessed based on different cloud attributes other than standard Service Level Agreements.

To enforce Service Level Agreements, continuous monitoring of the trust attributes is very important because of the dynamic nature of cloud computing. Li and Du (2013) proposed an Adaptive Trust

management model called the Cloud-Trust which is based on the user opinion. The proposed model is a combination of Adaptive and multi-attribute-based trust model for SLA in cloud computing.

Their proposed attribute-based trust management scheme considers an objective, attribute-based scheme based on user's direct experiences. The adaptive model for measuring multi-dimensional trust attributes uses rough set theory for Trust analysis. Li and Du (2013) have used rough set theory to adaptively conduct knowledge discovery of multiple trust attribute values called evidences which are collected by their Cloud-Trust model. In their study, they have integrated two kinds of adaptive modelling tools namely, rough sets and induce ordered weighted averaging IOWA operators. Their proposed model considers the performance requirements of cloud service providers such as, Security, availability and reliability. Their proposed SLA scheme is designed to eliminate the collaborative cheating or fraudulent practices issues which are faced by the traditional Reputation based Trust models. Their proposed model does not consider the reputation aspect of the cloud service providers, Li and Du (2013) acknowledges that it is a challenge to motivate users to submit their feedback to the trust measurement engine.

Generally, the cloud service users are assured by the cloud service providers regarding the quality of their services through the Service Level Agreements (SLAs). From the customers perspective, these SLAs are not surety for the trustworthiness of the cloud service providers and are not transparent enough. A multi-faceted Trust Management System which considers multiple attributes is important in gain customers confidence. A multi attribute Trust Management System for cloud service is suggested by (Habib et al., 2011). Their proposed Trust Management system architecture considers multiple attributes to compute trust. Their proposed system provides trust scores of the cloud provider's trustworthy behaviour of the underlying systems and their answers to Consensus Assessment Initiative Questionnaire (CAIQ) designed by Cloud Security Alliance. Habib et al. (2011) have considered multiple parameters such as Quality of Service (QoS), SLA, compliance, portability, interoperability, geographical location of date centres, customer support, performance test, deployment model, federated identity management, security measures and user feedbacks. They have proposed a Bayesian trust model using subjective probability. Using trust metric such as CertainTrust and CertainLogic, their proposed method provides a customized trust score of cloud service Provider which is based on the attributes selected by cloud customers.

To select cloud service providers based on their level of trustworthiness in the context of sensitive data or critical business applications, Alhamad et al., (2010) have proposed a Trust model based on

SLA and Trust management. Their proposed Trust model select the desired service providers based on the functional (matching the desired requirements regarding a service) and non-functional requirements. The SLA model deals with two attributes namely, functional requirements which finds average of millions of datasets to match the desired service and Non-functional requirements that entails the time to process the tasks or Data security. Their proposed architecture also uses a trust management model to work along with the SLA model to finalize the trustworthiness of the service provider. The trust model manages the relationship between the cloud service providers and users. Three sources of information are used by their trust model such as, local experience with the cloud providers and users, opinions of external cloud services and report provided by the SLA agent. However, the authors did not provide any evidence regarding the evaluation and implementation of their proposed model.

The Trust model proposed by the Manuel (2015) is based on past credentials and present capabilities of the cloud service providers. The proposed trust model calculates the trust value using four QoS parameters namely, availability, reliability, turnaround efficiency and data integrity. Their proposed model is implemented with a trust management system where the QoS parameters were assigned different predetermined weights which are based on their priority. Manuel (2015) presented the SLA preparation using QoS requirements namely, Turnaround time, Cost, Security, Computing Power and Networking speed. They have used CloudSim to simulate their model in the experimental setup. The QoS model has been compared and analysed against the FIFO model and the combines trust model which is the combination of three popular models such as Identity-based Trust model, Capability-based Trust model and Behaviour-based Trust model. Manuel (2015) have performed four experiments for the analysis of their model against the others in the context of Turnaround Time, Reliability, Availability and Data integrity simultaneously. The results of their experiment show that the QoS model outperforms the other two models in all the described contexts.

However, Manuel (2015) conclude that they have only used four attributes to calculate trust value of a cloud service provider but there are some other attributes which could be helpful in calculating the trust values such as Honesty, Return on Investment and Utilization of Resources, accountability and audit ability to measure trust.

To validate SLA choreographies, Haq et al. (2010) have proposed a distributed trust management model. Their proposed trust management model is designed to aggregate SLA from multiple cross-Virtual Organizations which are based on service-enriched environment such as cloud or grid,

requires validation. Their proposed solution is not specific to cloud neither it is targeting the cloud domain directly. Their proposed trust model integrates the traditional PKI based authentication which is a policy-based authentication system and reputation-based trust system. In their proposed conceptual framework, the hybrid trust model is in the form of third-party trust manager which selects the services before the SLA stage to prevent SLA violations. A use case scenario is used to validate their proposed hierarchical SLA aggregation with the hybrid trust model. However, their proposed model is in the form of a simple framework and a real-life implementation is not presented for the validation of their proposed trust model.

Online reputation systems are helpful for users in cloud computing markets. In their study, Macías and Guitart (2012) have identified two major drawbacks in online reputation systems, such as, the disagreement with the cost and ownerships of centralized reputation systems and the vulnerability of reputation system to reputation attacks from dishonest parties that want to increase their reputation. To overcome this issue, Macías and Guitart (2012) proposed a reputation system to help clients choose a suitable cloud service provider and allow avoiding the providers with low QoS.

The trustworthiness of cloud services not just rely on reputation but rather more factors are involved. Wang and Wu (2014) have proposed a trustworthiness measurement model to address the various types of possible factors. The first step in their proposed approach is to recognize the trust factors. Once the trust factors are recognized, these factors are categorized into shared factors (the collection of factors available for most cloud services) and unique factors (enforced by individual cloud services only). The shared factors recognized are feedback rating, third party recommendation, time, profile, transaction history or credit history, risk and friendship. The unique factors are transmission speed, stable level, capability, price scope, and availability and security settings. Wang and Wu (2014) have handled the unique factors through an ontology alignment approach. All the factors are then placed on the trust dimension using different weight and positioning approach. The three dimensions having six directions are defined as direct-indirect, subjective-objective and inflow-outflow trust dimensions. Once all the trust factors are located into the trust dimensions then they are converted to trust vectors.

Their proposed multi-criteria trust assessment mechanism is a structured approach to determine the overall preference among alternative options. It consists of three analysis functions such as, a trust vector aggregation which is used to analyse the overall cloud service preference. The second function is called support factor clustering which analyses the cloud service trustworthiness based on preference of factor characteristic. The third function is a minimum polyhedron, which is used to

analyse the overall cloud service trustworthiness. In their proposed approach, an adjustment is provided which is based on knowledge learning to adjust priority of criteria in case the measurement results are inconsistent with the user preference. Their proposed model does not capture the dynamicity of trust.

To create a level of trust regarding the security of cloud service provider's platform and the integrity of customer data, Wu (2012) proposed a trust model using fuzzy logic inferences. The Fuzzy Reputation-based trust model which is designed to handle uncertainty, fuzziness and incomplete information in cloud trust reports that help to select the most trustworthy cloud service provider. Their proposed model, calculates local trust scores of the cloud service providers and also their aggregate global reputation.

In their model the local trust score of a cloud service provider is evaluated by using cloud customer satisfaction. They have modelled customer satisfaction by using membership functions and by considering two fuzzy variables namely, Service quality and service price. After collecting the local trust scores, the model then produces a global reputation score for each provider by aggregating those local trust values. Fuzzy inference is used to obtain the aggregation weights for the three variables i.e., cloud provider's reputation, interaction time and interaction context. To prototype their model, they have used five fuzzy inference rules.

Alhamad et al. (2011) have proposed a trust-evaluation metric for cloud applications for IaaS to minimize security risks and malicious attacks in cloud computing. Their proposed evaluation approach considers four dimensions of trust (factors impacting the trust values) in cloud computing which are, Scalability, Availability, Security and Usability. Each dimension is modelled using Fuzzy set theory. The overall trust value is measured using Sugeno fuzzy-inference approach. For the collection of data and evaluation of their proposed solution, Alhamad et al. (2011) have used an E-Learning application and an online survey has been developed to collect dataset for their experiments. Due to the large number of rules in which all of them are not useful for their specific approach, they have used neural network to reduce the number of fuzzy rules. Their proposed trust evaluation metric lacks the feature to capture the dynamic nature of trust and the factors are not comprehensive enough in the context of cloud computing.

The proposed Peer-to-Peer architecture by Wu (2012) is decentralized in nature that would not need to be managed by a central authority and can help to solve the issue of the cost of implementation of such model. In their proposed trust model, the cost of implementation of their model is shared by all

the peers rather than by any centralized component. They suggested that their proposed model is designed for Cloud computing business models and also it can be implemented in a decentralized Peer-to-Peer network. Their proposed a Reputation system calculates the trust relationship from a client to a provider. Their model also defines trust relationship between peers and updates them in function to statistically analyse for the trustworthiness of their reports and weights them accordingly in the overall trust calculation. The validation of their proposed model is performed using the Market Reputation Simulator. During their experiments they have considered Service Level Objectives such as CPU, Disk and Network Bandwidth.

Their study focuses on the validity of the reputation model from an experimental point of view and is focused on the definition of the model which need real market implementation.

A reputation-based trust model is proposed by Pawar et al. (2012) that consider the uncertainty when computing opinions for cloud service providers. Their proposed model support service providers to evaluate trustworthiness of infrastructure providers. Their proposed model calculates individual trust values based on an opinion model in terms of belief, disbelief and uncertainty. The trust values are based on three parameters namely, SLA monitoring compliance, Service Provider ratings and Service Provider behaviour. The trustworthiness of the IP is then modelled as the expectation by combining all the three computations using conjunction and discounting operators. Their proposed model uses subjective logic operators with the opinion values. The uncertainty is defined as a function of an ellipse area and shape.

To improve the security in cloud computing, Qiang et al. (2011) proposed a trust evaluation model called the Extensible Trust Evaluation model for Cloud computing environments (ETEC). In their proposed algorithm, trust is modelled as a level of subjective probability to assist and make correct decisions, resist the malicious information, enhance robustness, fault tolerance and security of the cloud computing. The proposed ETEC algorithm is divided into three parts i.e., Recommendation Trust algorithm, Direct Trust algorithm and Dynamic Trust algorithm. For expressing the Direct Trust Qiang et al. (2011) have used a time-variant comprehensive evaluation method and for calculating Recommendation Trust they have used the space-variant evaluation method. Their proposed algorithm is implemented using CloudSim toolkit (Buyya et al., 2009).

Resource provisioning in cloud computing is another major challenge especially when it comes to Quality of Services. To achieve this, Xiao et al. (2010) proposed a reputation mechanism to select the most promising service form the fittest service set. Their proposed reputation-based QoS service

provisioning scheme assists in minimizing service cost of computing resources and for efficient service provisioning. In their proposed reputation management framework, the reputation of a data centre is evaluated and updated according to the user feedbacks. The user feedbacks are categorized under unsatisfactory, basic satisfactory and satisfactory. The appropriate resources are selected using three QoS metrics that includes Service cost, response time and reputation. The response time is considered as a practical metric and their scheme considers it as a statistical probability rather than a typical mean response time. The reputation of cloud services is constructed using the Dirichlet multinomial model, which is a multinomial Bayesian probability distribution mathematical model for reputation. To examine their proposed system a common tandem workflow-based queue model is considered and the efficiency and effectiveness of their proposed scheme is evaluated using numerical simulation. Their proposed QoS provisioning algorithm does not consider security and privacy metrics. Moreover, their study does not provide aggregation method for the metrics values.

A centralized trust model to address information security and data privacy concerns is proposed by Rizvi et al. (2014). Their proposed model involves Cloud service providers, Cloud service users and third-party auditing bodies to determine a fair score for each service provider by combining third party auditing score and feedbacks from each CSU. Their proposed centralized model is objective in nature. In their proposed model, the base line trust values are obtained from the initial scores by the third-party auditors. These initial scores are then combined using mathematical formula with the feedback scores from the cloud service users to obtain the final trust value of cloud service providers. Their proposed model considers scores for specific context rather than an overall generic score of the cloud service provider. The trust value is updated using a time decay function introduced by (Yang et al., 2010). However, the trustworthiness of the feedbacks by Cloud service users is not considered neither the effectiveness of the third-party auditors is proved.

Trust models can also help consumers in selecting suitable peers for reliable file exchanges (Canedo et al., 2012). This is closely related to the data integrity issue in cloud computing. Reputation based trust management schemes have been introduced in cloud computing which can address data integrity and security issues related to cloud provider's platform (Wu, 2012).

Secure access control of the resources in cloud computing is of great importance and the cloud users are unsure about the integrity of their data and are reluctant to reveal their data to the cloud providers. The cloud service providers do not provide a solution for the reliability of file exchange among peers. To address and ensure reliable file exchange in private clouds, Canedo et al. (2012) have proposed a

trust model which calculates the trust values among users or nodes. Their proposed model ranks between the trustworthy nodes using a predefined metrics. The attributes considered in the metrics are Processing Capacity, Operating System, Storage and Link Capacity. Each attribute is assigned with different weights such as, Storage and Processing is assigned 35% each while 15% to Link and Operating System. The value of trust could be between [0, 1]. Each file sharing node manages two Trust tables in a distributed fashion i.e., List of Direct trust values and a list of recommended trust values. These values are stored based on the previous interactions or history by the node in the direct trust table and values from recommendation by other nodes regarding the trustworthiness of the particular nodes. In order to share or transfer files to other nodes, the sending node first check the trust value of the receiving node in the direct trust table if not found, then it queries other peers regarding the trustworthiness of that node. To simulate their proposed model, Canedo et al. (2012) have used CloudSim framework as a simulation tool. However, their proposed trust model considers only Hardware features to calculate trust values.

Malicious feedbacks from users (Pawar et al., 2013) and malicious access towards cloud domains (Yang et al., 2010) are among the well-known challenge being faced by trust models.

To evaluate the trustworthiness of Infrastructure Provider, Pawar et al. (2013) proposed a trust model which considers different cloud characteristics called as the dimensions and their associated features. The dimensions considered in their work include, on-demand self-service, resource pooling, rapid elasticity and measured service. The identified features specify the context within the dimension. The on-demand self-service dimension has features like *availability_d* and timely_d whereas the dimension rapid elasticity has the features *availability_e* and *timely_e*. The affinity and legal are the features of the resource pooling dimension while viewable, controllable and reportable are the features of measured service dimension. The proposed trust model works with an opinion model that considers uncertainty and uses early filtering to reduce the impact of the malicious feedback providers.

In the model proposed by Pawar et al. (2013), the trust of the infrastructure provider is evaluated by the cloud broker who takes trust information in the form of feedbacks from the cloud service providers. The trust value is computed and combining reputation and reliability using subjective probability. The reputation of the infrastructure provider is computed using the feedbacks gathered from individual service providers. The reputation value is the result of the computation of the values assigned to each of the features in cloud dimensions. The reliability of the infrastructure provider is

based on direct interactions between the service providers and infrastructure providers. The reliability value is also computed regarding all the features in the cloud dimensions and the service providers updates its reliability and rating for each feature of the dimensions. Finally, the credibility of the feedback provider is computed which has impact on the reputation and reliability values and also helps to filter out the malicious feedback providers. The outlier detection technique is used to filter out the malicious feedback providers. Their model is evaluated using simulation for the robustness against malicious feedback providers using OPTIMIS project using simulation data.

The security requirements of the cloud computing environment are higher than the traditional environment because of the large-scale, distributed, heterogeneous and dynamic nature. To meet the security requirements of the cloud computing environment, Yang et al. (2010) proposed a trust model that can work in collaboration with firewalls to estimate dynamic context and define risk signals. Their proposed model is similar to the domain-based trust models in which, cloud environment is divided into a number of autonomous domains. Each domain maintains its local trust values at node level and also at domain level. The trust values are extracted from direct interactions by the nodes and also recommendations kept in separate tables. Yang et al. (2010) proposed an algorithm for acquiring trust values among nodes of in cloud. The domains are defined using quantitative theory and the trust values are updated using the Time Decay Function. Their proposed model is very focused and not comprehensive to cover the most common trust related issues related to cloud service.

A trust management model proposed by Sun et al. (2011) addresses the two main issues related to security in cloud computing environments such as, relation of trust levels and evaluation of trust objects using different evaluation attributes. Their proposed model perceives trust as a subjective probability and is based on the fuzzy set theory. Their proposed model builds a trust relationship between cloud users and cloud service providers which helps the cloud users in their decision making towards searching for cloud services from suitable cloud service providers.

Their model measures trust, direct trust relation, intro-domain direct trust relation, direct trust expression, recommended trust and connection and incorporation of trust using fuzzy set theory. An algorithm is presented by Sun et al. (2011) for the inter-domain and intro-domain direct trust evaluation. The trust evaluation attributes used in their model includes, flow of network produced by interaction, bandwidth of current network, successful communication time of nth interaction, tentative communication time of the nth interaction and the time consumed. All the trust evaluation attributes are considered as time decaying which means that their value changes constantly. They

have used a time decaying function to model this dynamicity. Beside the high accuracy rate of their model, the practicability and rationality of their proposed model is yet to be justified.

Cloud service can be deployed under different deployment models which brings more challenges in the context of trust. To determine the trustworthiness of federated cloud computing entities in intercloud computing, Abawajy (2009) proposed trust management system to provide help and satisfy client QoS requirements by selecting high quality cloud services. One of the major features of cloud computing is that application deployment could require resources from multiple distributed platforms located on several autonomous clouds. One of the main issues here is that the interacting users and services either have no previous information about the other parties and the information is not enough for service assessment. The type of information required and the distribution of this information is another issue in intercloud computing. This intercloud resource sharing leads to the issue of the trustworthiness of these entities which are practically located on separate cloud infrastructures. The proposed reputation-based trust management system determines the trustworthiness of intercloud computing entities using a distributed framework.

In a similar study, to improve security issues in cross-clouds environment, Li and Ping (2009) have proposed a trust model which is a domain-based model. Their proposed model ensures security of both cloud providers and consumers. The domain-based cloud model divides the cloud environment into several domains and computes trust locally. Their proposed model divides the cloud environment into two separate domains. One of the domains contains the cloud provider's resource nodes and an agent to compute the trust of the domain entities. The other domain contains the cloud customers. Each domain has different domain-specific trust tables and trust in each domain is computing through different strategies. Their proposed trust model itself is implanted as a service in this domain-based model. Before every transaction, entities check the threshold for trust decisions. If the other party has a trust valued bigger than the threshold then they proceed with the transaction. This threshold is set independently by each domain. Li and Ping (2009) have presented algorithms for Direct trust, Recommendation Trust and updating trust values. Updating of trust values is based on two assumptions. First, trust valued could be updated based on time and second, trust values could be updated after every transaction. A set of experiments have been performed to evaluate their proposed model. Two evaluation criteria are set for their proposed model namely, trust accuracy and transaction success rate. However, their proposed model is merely a simulation that does not reflect real life scenario as cloud entities behaviour are more complex in real life.

The study by Abawajy (2009) assumes that each cloud computes and maintains their reputation locally and the trust is in the form of relationship between entities. This makes the nature of the trust value storage distributed when seen in intercloud context. The reputation manager inside their proposed infrastructure computes the reputation of an entity by using three types of ratings namely, Personal experience, Reputation ratings and Honesty ratings. The reputation manger also updates the reputation of the entities using the first-hand or second-hand information through a trust update algorithm.

Most of the proposed trust models in cloud computing are focused on the trustworthiness of the cloud service providers. Prajapati et al. (2013) have proposed a Trust management model which is focused on SaaS and Trust is considered as a level of subjective probability. Their proposed trust model is responsible for managing Trust, its properties and different components of Trust model. Their proposed model is focused on SaaS and Trust is considered as a level of subjective probability. Their proposed model computes Trust based on Direct experience, entity's reputation and recommendation from third party. The direct trust is defined by time-variant evaluation method and the recommended trust with the space-variant evaluation method. A decay function is used to estimate the trust value and uncertainty of each peer. The evaluate the feasibility of their proposed model, a case study is presented in a distributed file sharing service scenario using SaaS. However, their proposed trust model is not comprehensive and does not consider multi-factor trust computation.

For multicloud environments, a User-centric trust management framework is proposed by Soleymani et al. (2021) to facilitate trust worthy cloud service provider selection. Their proposed framework considers both subjective and objective attributes to calculate trust values for cloud service providers. The subjective trust values are calculated from feedbacks and objective trust values from parameters such as availability, reliability, response time, security, privacy, transparency, and consumer protection. The final trust values from the subjective and objective parameters are calculated using fuzzy rules. Their proposed framework has a feedback evaluation component for the rectification of fake user feedbacks. However, their proposed framework does not consider forecasting trust values and user sentiment towards the service parameters.

As discussed earlier, security and privacy are among the key challenges that is faced in cloud computing. To enhance data security and privacy in cloud based distributed storage framework Alshammari et al. (2021) proposed a trust model that is integrated with cryptographic task role-based access control. They have considered inheritance and hierarchy in their trustworthiness evaluation of

roles and tasks. Their study in focused on security and privacy of data stored on the cloud and does not consider other aspects cloud services other capabilities.

Recently, an integrated multi-stakeholder framework for cloud-based trust building is proposed by Lynn et al. (2021). The framework is based on the assurance and accountability in cloud service providers. Their proposed approach focuses on trust building and trust repair mechanisms. However, their proposed approach is a high-level framework and a detailed discussion and implementation of each component is not provided. Trabay et al. (2021) proposed a trust framework using multi-criteria decision-making to evaluate cloud services. Their proposed framework applies fuzzy logic on criteria such as performance, agility, finance, security and usability. MCDM approaches such as TOPSIS and VIKOR is used for the ranking and selection of trust in cloud service providers. Their proposed framework does not consider dynamicity in the selection and ranking of cloud service providers.

Trust management frameworks are also gaining attention in federated cloud environment. To facilitate cloud federation, Latif et al. (2021) proposed a federated cloud trust management framework. In their proposed framework, the trust evaluation is performed on service level agreements and feedbacks from customers and cloud service providers. Questionnaires are used to collect customer feedback regarding the security and privacy related features. However, their study does not consider the missing information in the SLAs and customer feedbacks.

## 2.4 FACTORS EFFECTING TRUST

One of the major issues highlighted in (Habib et al., 2012) is the unwillingness of the cloud consumers to depend on the providers in relinquishing the control over their potentially business relevant information, data and internal processes. This is a key challenge in developing a trust model for SaaS products. The existing trust models for cloud services, consider either the subjective or objective factors without the actual understanding of the cloud services internal architectures and processes. The trustworthiness of a cloud entity or service can be evaluated using several mechanisms (Habib et al., 2012) such as, Black box approach, Inside-out approach and Outside-in approach. The Black-box approach only considers user feedback without considering the internal components and processes of the service. In the Inside-out approach, the trustworthiness of an entity or service is derived based on the knowledge about the architecture and trustworthiness of its components. The Outside-in approach requires knowledge of the internal architecture and components of a service along with the observed behaviour.

In a similar study (Habib et al., 2014), CAIQ is proposed as the root for the trust factors. As discussed in the previous section, one of the key issues with trust sources such as CAIQ includes reliability of the information provided by the cloud service providers related to their platform, processes and service architectures. The trust modelling process for the cloud services must not be influenced by the cloud service providers. Moreover, trust models tend to become dependent and can struggle to acquire on-demand updates for the cloud services, hence compromising the dynamicity of the models.

The key to the identification of appropriate trust factors is to understand the architectures of deployed services and the practices in place to maintain and monitor these services without the influence of cloud service providers.

Insufficient and missing information is an obstacle in predicting the provider's quality of service and trustworthiness which effects the adoption of cloud computing by the customers. In the existing cloud computing literature, to the best of my knowledge, the missing factors for trust modelling has not been explored.

## 2.5 FUZZY TRUST MODELS IN CLOUD COMPUTING

As discussed in Section 2.2, the concept of trust is subjective and different theories and approaches have been used to model trust. Fuzzy sets and fuzzy inference systems are based on subjectivity and has been used to model trust in several application domains. The studies using fuzzy system to model trust in cloud computing is shown in Table 2.3.

Most of the existing literature on trust models in cloud computing which utilize fuzzy systems do not assign dynamic weights to the input factors and leave it open to customer preferences. The weights are assigned by the users manually by the users based on their preferences. Moreover, the focus of these studies is on the objective factors in cloud computing.

The proposed trust models for cloud services have strengths and weaknesses which can compromise the effectiveness of these models. Kanwal et al. (2013) have proposed assessment criteria for the evaluation of trust models in cloud computing. They have pointed out seven assessment criteria each having one of the three levels High, Medium and Low. Their proposed assessment criteria include, Data Integrity, Data control and Ownership, Process execution control, Quality of Service attributes, Detection of untrusted entities, Dynamic trust update and logging model complexity. However, in their study, Kanwal et al. (2013) have not provided any experimental results to validate their proposed assessment criteria for trust models.

Table 2.3 Fuzzy trust models in cloud computing

| Reference | Domain | Fuzzy Approach | Input type | Input weights | Dynamic |
|---|---|---|---|---|---|
| Nagarajan, et al. (2017) | Cloud services | - | QoS factors | Fuzzy | - |
| Lee et al. (2006) | Generic | Sugeno | Generic | Open | - |
| Alhamad et al. (2011) | Cloud apps | Sugeno | Objective | Open | - |
| Kesarwani & Khilar, (2019) | Cloud computing | Mamdani | Objective | Open | - |
| Selvaraj & Sundararajan, (2017) | Cloud services | Sugeno | IaaS Objective | Max. Info | Yes |
| Rizvi et al. (2020) | Cloud service providers | Mamdani | Security factors | Open | - |
| Wu (2012) | Cloud computing | - | Service quality and service price | - | - |

## 2.6 CRITICAL ANALYSIS

From the above discussion, it can be concluded that the concern is not entirely about the cloud provider's intentions, rather on the capabilities of the cloud computing itself (Khan & Malluhi, 2010). Trust in cloud computing is not only limited to the technology but also associated to the customer's confidence in the lack of transparency, loss of control over data and security assurances provided by the cloud service providers.

Table 2.4 Critical analysis of Trust models in cloud computing

| Reference | Cloud Entity | Model | Architectural Best Practices | Missing Trust factors | Trust factors Forecasting | Dynamic trust Factors weights | Fuzzy logic-based Trust model |
|---|---|---|---|---|---|---|---|
| Wang et al. (2018) | Generic | Trust | - | - | - | - | - |
| Manuel (2015) | CSP | Trust | - | - | - | - | - |
| Hwang et al. (2009) | Generic | Trust | - | - | - | - | - |
| Hassan et al. (2020) | CSP | QoS | - | - | - | - | - |
| Abdelrazek et al. (2015) | SaaS | SM | - | - | - | - | - |
| Hussain et al. (2014) | Generic | SLA | - | - | - | - | - |
| Tan et al. (2008) | SaaS | SLA | - | - | - | - | - |
| Tang and Liu (2015) | SaaS | SC | - | - | - | - | - |
| Hedabou et al. (2020) | SaaS | TPM | - | - | - | - | - |
| Prajapati et al. (2013) | SaaS | Trust | - | - | - | - | - |
| Sule et al. (2016) | Generic | SC | - | - | - | - | ✓ |
| Chou and Chiang (2013) | SaaS | QoS | - | - | - | - | - |
| Habib et al. (2011) | CSP | Trust | - | - | - | - | - |
| Naseer et al. (2014) | CSP | Trust | - | - | - | - | - |
| Qiang et al. (2011) | Generic | Trust | - | - | - | - | - |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Macías and Guitart (2012) | CSP | Trust | - | - | - | - | - |
| Li and Du (2013) | Generic | Trust | - | - | ✓ | - | - |
| Habib et al. (2014) | Generic | Trust | - | - | - | - | - |
| Moyano et al. (2013) | SCA | Trust | - | - | - | - | - |
| Canedo et al. (2012) | PC-Nodes | Trust | - | - | - | - | - |
| Wu (2012) | CSP | Trust | - | - | - | - | ✓ |
| Pawar et al. (2013) | Generic | Trust | - | - | - | - | - |
| Abawajy (2009) | FC | Trust | - | - | - | - | - |
| Yang et al. (2010) | Generic | Trust | - | - | - | - | - |
| Rizvi et al. (2014) | CSP | Trust | - | - | - | - | - |
| Pawar et al. (2012) | CSP | Trust | - | - | - | - | - |
| Alhamad et al. (2010) | CSP | Trust | - | - | - | - | - |
| Sun et al. (2011) | Generic | Trust | - | - | - | - | ✓ |
| Li and Ping (2009) | Generic | Trust | - | - | - | - | - |
| Canedo et al. (2011) | PC-Nodes | Trust | - | - | - | - | - |
| Rashidi and Movahhedinia (2012) | CSP | Trust | - | - | - | - | - |
| Alhamad et al. (2011) | IaaS | Trust | - | - | - | - | ✓ |
| Wang and Wu (2014) | Generic | Trust | - | - | - | - | - |
| Nagarajan et al. (2017) | Generic | Trust | - | - | - | - | ✓ |
| Kesarwani and Khilar (2019) | Generic | Trust | - | - | - | - | ✓ |
| Selvaraj and Sundararajan (2017) | Generic | Trust | - | - | - | ✓ | ✓ |
| Rizvi et al. (2020) | CSP | Trust | - | - | - | - | ✓ |

*SM = Security Metric, SCA = Social Cloud Applications, PC = Private Cloud, CSP = Cloud Service Provider, FC = Federated Cloud, SC = Security Controls, TPM = Trusted Platform Modules*

The Trust in cloud computing does not mean to compensate when a violation of agreement occurs rather it is a mean to prevent such violations. Thus, Trust models in cloud computing should focus on preventing failures and violations than post-failure compensations (Khan & Malluhi, 2010).

Table 2.4, shows a critical evaluation of the existing literature on trust models in cloud computing where, a '✓' sign indicates that the article has addressed the specified aspect of trust model and '-' indicates vice versa. None of the trust models in the existing literature have considered the cloud services architectural best practices as the key trust factors. Similar is the case of missing trust factors, where the existing trust models have not considered this key issue during the modelling of trust for cloud services. Among the existing literature, Li and Du (2013) have proposed a forecasts component for their trust model. However, their proposed model forecast the overall trust model rather than individual trust factors. Li and Du (2013) have also proposed a Simple Trust Model (STM) that uses a simple average approach to assign weights to trust attributes which is not dynamic in nature. In the context of dynamic weighted trust factors, Selvaraj and Sundararajan (2017) have used the dynamic weights for the input trust factors using maximum information for IaaS.

Based on the critical analysis from Table 2.4, the emerging current issues related to trust models for cloud services is summarized as follows:

1. Most of the existing literature on trust models are focused on the cloud service providers trustworthiness. However, as discussed in Section 2.2, trust is context specific and the trust models should consider modelling trust for the services separately. This means that cloud service provider's capabilities may differ in providing a specific type of cloud service from another. This is important because a cloud service provider can offer multiple services where the quality of each service may differ from the others. This position also stands when comparing different type of cloud services such as SaaS, IaaS and PaaS.
2. As discussed previously, cloud delivery models have dependencies and independencies, which means, that the factors considered in modelling the trust of IaaS are relatively different when modelling trust in SaaS. SaaS is a special case of cloud delivery models in which the entire cloud infrastructural stack underneath the SaaS, may or may not be owned by the SaaS provider. To the best of my knowledge, there is not approach in the existing literature that considers factors that are specific to SaaS in modelling trust.
3. None of the existing approaches have considered the cloud applications and services architectural best practices as trust factors to compute the trustworthiness of SaaS.
4. There is no approach for modelling trust that provides solution for the missing information regarding the trust factors.
5. As trust is computed using the most relevant trust factors, it is important for the trust models to have the capability to forecast these trust factors for a point in time in the future and eventually, compute the trust value from the forecasted trust factors. In the existing literature, there is no approach to forecast the values of the relevant trust factors.
6. None of the existing fuzzy trust models uses dynamic weights to rank each trust factor.

## 2.7 CONCLUSION

In this chapter, an extensive survey of the existing literature on trust models for cloud service was presented. At first, the chapter provided an overview of the trust assessment and modelling concepts that included formal definitions of trust, the well-known trust modelling theories and methods and few of the well-known trust models in different application domains. This was followed a comprehensive survey of the existing literature on trust assessment and trust modelling in cloud

computing. The existing fuzzy-based trust models is critically analysed based on their inference methods, input data types and input weighing methods. Finally, a critical analysis of the existing trust models in cloud computing is presented using the criteria that is based on the emerging gaps in the current literature related to cloud trust models.

In the next chapter, the main research problem being addressed in this thesis is described along with the research issues which are based on the gaps in the existing literature.

# Chapter 3:
# Problem Definition

## 3.1 INTRODUCTION

As discussed in the previous chapters, it is very important for the Software as a Service customers to make informed decisions when selecting the most suitable service for their businesses. A certain level of trust needs to be established in the SaaS products before a decision is made by the customers. Moreover, in the frequently expanding and enhancing cloud marketplace, the level of trustworthiness in the services vary accordingly. On one hand, the SaaS providers must keep up their pace to adopt the latest innovation in cloud related technologies and on the other hand, the customers need to be provided with the latest information regarding the trustworthiness of the SaaS providers. The trust in SaaS products comprise of many factors and the core understanding of these factors is very important and need to be researched thoroughly. Furthermore, the selection of SaaS products by the customers are generally based on different priorities among the underlying trust factors. Therefore, it is also very important that the SaaS customers should be able to have access to the performance level of each of the core trust factors.

In order to facilitate the SaaS customers in their decisions, it is very important to understand the up-to-data industry standards and best practices for the identification of core service factors (Raza et al., 2019). In addition to that, to make a realistic assessment of trust in the SaaS products, it is important to obtain the most relevant and reliable information regarding the all the core service factors of SaaS products. However, as discussed in Chapter 2, the existing approaches consider the objective aspects and questionnaires designed for the service providers to fill-in. This brings into question the reliability of the information regarding different service factors of the services. Furthermore, the up-do-date industry standards and best practices are not considered as the source of identification for the relevant service factors.

Based on these limitations of the current studies, in this chapter, I present and formally define the problem that this aim to address. The aim of this thesis is to develop a methodology that support the SaaS customer in making informed decision by assessing the trust in SaaS products. I explain, the sub-problems in this thesis as fined grained research issues. For the solution of each research issue identified in this chapter, I set corresponding objectives that this thesis aims to achieve.

In the next section, the key terms and concepts are defined that are used in this chapter and throughout the thesis. In Section 3.2, I formally define the thesis problem. I will present the breakdown of the thesis problem into a number of research issues in Section 3.3. In Section 3.4, I will explain the objectives set to solve the research issues mentioned in Section 3.3 that I aim to achieve in this thesis. In Section 3.5, I present the research approach which is followed to develop the solution for the problem identified in this chapter. Section 3.6 will conclude this chapter.

## 3.2 PROBLEM DEFINITION

In the previous chapters, I have discussed the details regarding the importance of customer trust in cloud services specially in the Software as a Service delivery model. There is a variety of Software as Service available in different software categories on the cloud marketplace. Choosing the most suitable SaaS is a crucial task for businesses. There are a number of challenges associated with using the SaaS products including their shared infrastructure, the host location and the delivery of these service over the internet. The SaaS customers do not have enough visibility into the security and deployment structure of these services which raises many questions regarding the security of their data, privacy and trust.

As technologies related to cloud computing are advancing constantly, it directly effects different aspects of the SaaS products. For example, most of the SaaS products are hosted on third party infrastructures and these infrastructure providers offer the benefits for their economy of scale to the customers (in this case the SaaS providers). Hence, the cost of these SaaS products must be constantly adjusted accordingly. Moreover, other factors such as the availability also known as the uptime, is improving as the infrastructure providers are constantly expanding and enhancing their infrastructure. The SaaS products on the other hand, must keep up with these changes to facilitate their own customers. Besides that, for the available SaaS products, their existing customer base is growing as well, which consistently effects the performance, management and customer support. The introduction of new SaaS products in the cloud marketplace, generally, support and utilize latest cloud related technologies. This entices the customers of the existing SaaS products and persuade them to adapt to the latest offerings. From the above examples, it is very clear that there is an urge to constantly monitor the SaaS marketplace for these frequent changes. This puts a healthy competition among the SaaS providers to constantly improve their offerings. At the same time, it is very important that the SaaS customers also should be able to assess these services based on these changes in different service factors.

Furthermore, business using the SaaS products have different preferences for their business needs. For example, a business might prioritize data security over the other service factors and another business might rely heavily on cost as a key in choosing a suitable SaaS product. Therefore, it is very important that the SaaS customers must also be able to assess the services based on their preferred service factors. The key issue that arises from this discussion is:

"*What are the key service factors of SaaS that can potentially be used as trust factors to build customer trust, and what are the basis for the identification of these service factors*?"

The SaaS customers must have access to such frameworks that enables them to assess the SaaS products based on different service factors. The key challenges over here are to (1) identify the core service factors and acquire reliable information regarding them and (2) a framework that can model the trust in the SaaS products based on the core service factors.

Identification of core service factor that can help in the assessment of cloud services is a key issue. As discussed in Chapter 1 and Chapter 2, in the current literature, there are a number of suggested factors to assess cloud services. Most of the suggested factors are objective and are focused on the infrastructure level services. Furthermore, service providers and their services are also assessed based on structured questionnaires. In either of the assessment approaches, the main challenge is the relevance and reliability of the information regarding the information acquired for the service factors. For example, the efficiency of a SaaS product can be assessed based on the response time which can partially be dependent on the application and service design and partially on the infrastructure underneath. Also, the information collected from the service providers through the questionnaires does not provide a reliable or measurable level of trust. Therefore, in order to compute trust in the SaaS products, it is very necessary to have relevant and reliable information regarding the core service factors.

Once the core service factors for the SaaS products are identified and relevant information is collected, the next challenge is to model the trustworthiness in the SaaS products with the help of acquired information. As discussed in chapter 2, different factors are considered when modelling trust in different domain. Beside the reliability and the method used for collecting information for these core service factors, the consistency in the observed information regarding each service factor is very important. For example, to devise a level of trust for SaaS products that facilitate customer decision in service selection, information regarding all the core factors must be available. The unavailability of information regarding one or more service factor can lead to unrealistic assumptions towards the

trust level. Besides that, as discussed earlier in this section, businesses have different priorities when selecting the most suitable SaaS products. Missing information related to their preferred service factor pose a major challenge for any trust assessment framework.

Furthermore, as discussed earlier, due to rapid changes in cloud related technologies, the SaaS products must keep up with ongoing changes. This requires any trust assessment framework to have the capability to predict the performance of each service factor at a given future time spot. This is very important for individual customers and businesses that plan ahead in acquired SaaS products. A trust assessment framework should have this capability to forecast the trust levels as well as the performance of individual service factor, of the potential SaaS products, for a future time spot.

Based on the discussion above, the problems that I aim to address in this thesis is formally defined as: ***How to develop a trust assessment methodology to support Software as a Service customers in selecting most suitable services based on factors rooted in architectural best practices for cloud application design and operations***?

## 3.3 RESEARCH ISSUES

In the wake of a comprehensive solution for the main thesis problem mentioned in the previous section, a number of research issues are raised. These research issues are described as follows:

1. *How to develop an intelligent method for the identification of the key service* factor*s for Software as a Service?* The proposed framework and method should be able to identify each service factor from the obtained information with higher accuracy.
2. *How to develop a method for inferring and imputing the missing information regarding the identified service* factor*s of Software as a Service?* During the collection of information regarding the service factors of SaaS, there are missing information related to one or more service factor. A methodology is needed to infer and impute these missing values with high precision.
3. *How to develop an efficient method to forecast the future values for the service* factor*s of Software as a Service?* For the support of the customers, a methodology is needed that should be able to forecast the quantitatively, the performance values of each service factor at a required time spot in the future.

4. *How to develop an intelligent method that can model the trust level of Software as a Service based on the relevant factors of SaaS?* A comprehensive framework is needed that should model the trust level in Software as a Service products for any given time spot.
5. *How to validate the developed techniques for the research issues in this thesis?* The developed techniques and methodologies need to be evaluated to ascertain their validity.

## 3.4 RESEARCH OBJECTIVES

To solve the thesis problem and the issues identified in the previous section, I have set the following objective to achieve in this thesis.

**Objective 1: To develop an intelligent method that automatically identifies the key service factors of the SaaS**

The first part of this objective is focused on to investigate the most relevant information. In the existing literature, the trust assessment approaches for cloud services broadly consider either the objective factors of the services or rely on the questionnaires that are filled in by the service providers regarding the key factors. In this thesis, I intend to consider the service factors that are rooted in architectural best practices for cloud services and applications design and operations. The second part of this objective is to obtain the reliable information for the SaaS products. As discussed above, the information collected for the objective factors of SaaS or the from the questionnaires filled in by the service providers has been considered in many studies and are considered to be less reliable as they can be impacted by the providers. I only consider the information regarding the SaaS products that are free from the providers impacts.

In the second part of objective 1, an intelligent method is developed that is able to automatically identify each service factor from the obtained information with higher accuracy. This is only possible when the relevant and reliable information is collected and pre-processed. This means that, a component of the framework will use intelligent techniques to filter and convert the information in an acceptable format for the other components of the framework to perform the identification and categorization tasks. There are a number of studies that have used different classification and categorization techniques in different domains. However, to the best of my knowledge, there is no such approach that are used for the categorization of the SaaS service factors.

**Objective 2: To develop a method for imputing the missing information regarding the identified service factors of SaaS**

As discussed in the previous section, in order to assess the trust levels in a SaaS product, the information regarding each service factor is required. The unavailability of information regarding any of the service factor results in the unrealistic computation of trust values. A methodology is developed in this thesis, that is capable to impute the missing values for the service factors with high precision. This methodology is developed primarily to achieve Objective 2, but also, paves a way for the achievement of Objective 4 which depends on complete information for SaaS products. To the best of my knowledge, there is no such approach developed to infer missing information of service factors of SaaS products.

**Objective 3: To develop a reliable method for forecasting the future values for the SaaS service factors**

The service factors information is crucial to the SaaS customers in their decisions to select suitable SaaS products, for the current product standings as well as for any given time in the future. In order to get the performance status of the service factors, the proposed methodology should be able to forecast these performance values in the future based on the past information. In the existing literature, there are many timeseries forecasting methods that are successfully implemented in several domains. However, to the best of my knowledge, there is currently no approach that are used to forecast the performance of service factors of SaaS products. This is a key objective of this research as the proposed methodology use the most optimal technique for forecasting the data specifically collected for this thesis.

**Objective 4: To develop an intelligent and dynamic method that models the trust level of SaaS based on the relevant factors of SaaS?**

A comprehensive framework is developed that models the trust level in Software as a Service products for any given time spot based on the services factors. The trust assessment framework is capable not only to compute the trust levels of the SaaS products as of its current standings, but also, for any given time spot in the future. For such capabilities, the framework developed to achieve this objective is, dependent on the success of the methodologies and techniques developed in the previous objectives. In the existing literature, there are number of trust models and trust assessment techniques in different domains. However, for the trust assessment of SaaS products, no existing approach use the service factor information of the SaaS products as trust factors.

**Objective 5: To validate the developed techniques for the solution of the identified research issues in this thesis.**

Each of the developed techniques in this thesis are assessed for their effectiveness and applicability by validating them against the relevant approaches. The components of the main solution framework are integrated as a prototype. The validation is conducted on the data collected specifically for the purpose of this thesis.

## 3.5 RESEARCH APPROACH

To address the research issues outlined in the Section 3.4, this thesis aims to develop a trust framework for Software as a Service products with subsequent testing and validation. To achieve the aim and objectives of this thesis, I follow an appropriate scientific research method during the complete development cycle of the trust framework. In this section, I provide an overview of the most commonly used categories of the scientific research methods and a detailed explanation of the selected scientific research method for this thesis.

The science and engineering approach and the social sciences approach are the two main categories of research approaches in information system research. The social sciences research approach broadly involves social and cultural issues and objectives are set to prove or disprove hypothesis and are not concerned with the development of new methods and artefacts. On the other hand, the science and engineering research approaches are focused in designing new methods and artefacts to achieve the research objectives. The research following the science and engineering approach are carried out at three broad levels namely, the conceptual level, the perceptual level and the practical level. Following are a brief explanation of each level.

- At the conceptual level, which is the first level of the research approach, the problems are identified, analysed and conceptual frameworks are constructed.
- At the perceptual level, new methods and system architectures are analysed, designed and developed.
- At the practical level, based on the methods designed in the previous step, experiments, testing and evaluation of the new methods and system is performed.

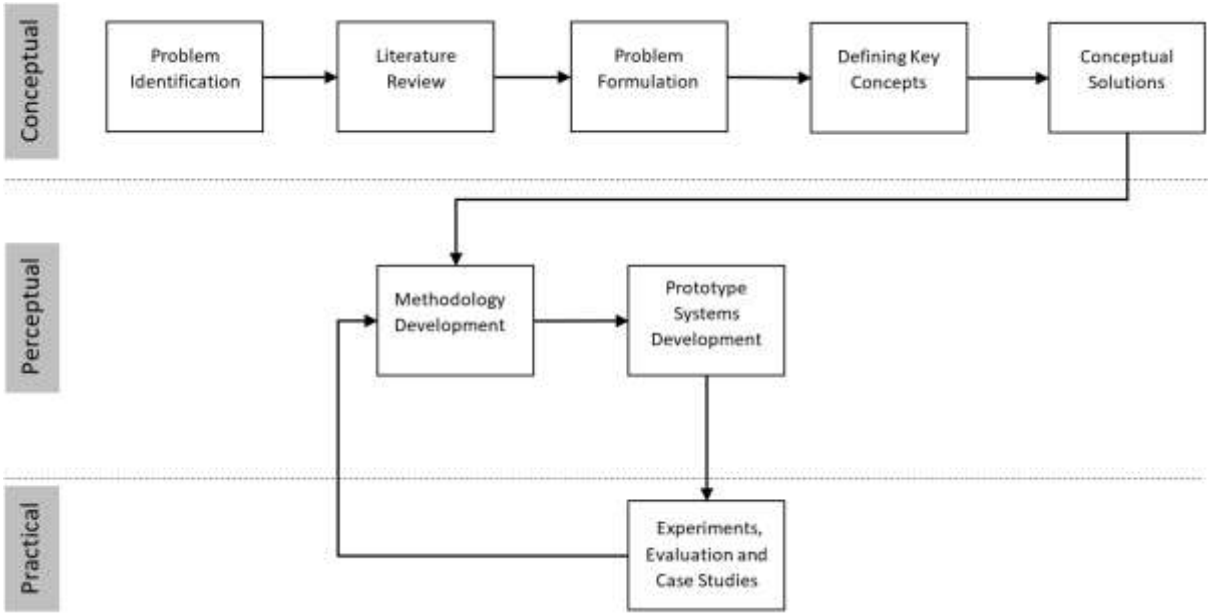



Figure 3.1 Research approach

Based on the brief explanation above, this thesis clearly falls under the science and engineering research approach with the development of the trust framework as the aim of this thesis. Figure 3.1 depicts the three levels of this research approach with specific set of tasks at each level. I follow the research method proposed by Nunamaker et al. (1990) in the detailed design, development and validation of the research output. In the first level of my research, I extensively research the literature in the domain of cloud computing with particularly focusing on the customer support and trust models for Software as a Service. This led to the identification of the key and open issues in the relevant literature in cloud computing. I designed the conceptual framework with appropriate formulation of the identified problems. At the perceptual level, for each of the identified problem in the conceptual framework, I developed approaches for the components that includes the identification of SaaS service factors from the customer reviews, customer sentiment intensity computation, imputing missing service factors, forecasting sentiment intensities and a trust model. Each developed component is prototyped accordingly at the end of this level. Finally, using the data collected in the previous steps, experiments and case studies are conducted and the developed systems are evaluated in the practical level.

## 3.6 CONCLUSION

In this chapter, I summarized the research issues concerning to the trust assessment and modelling in Software as a Service. I formulated the definition of the problem that this thesis aims to address. The key terms used throughout the thesis is also formally defined in this chapter. The main research

problem is divided into sub-problems and for each of the sub-problems, research objectives are set to be achieved in this thesis. I explained the details of the science and engineering research methodology that this thesis employs to achieve the research objectives. In the next chapter, I provide an overview of the solutions for the research issues identified in this chapter.

# Chapter 4:
# Solution Overview

## 4.1 INTRODUCTION

In the previous chapter, I explained the issues arising from the modelling of trust in Software as a Service that supports customers in making informed decisions. Based on the identified research issues in the previous chapter, each issue is studied as sub-problems in this thesis. To address the research issues, in this chapter, I present a conceptual solution framework. As discussed in the previous chapters, trust modelling and assessment has been studied extensively in different application domain and several approaches have been used to define the concept of trust. However, the assessment and modelling of trust in Software as a service based on the factors from architectural best practices has not be considered in the current literature. In this chapter, I propose a definition of trust in the context of Software as a Service and a solution framework for modelling trust.

In Section 4.2, I present the definition of trust model in the context of Software as a Service, followed by the overview of the solution framework for the proposed trust assessment model in Section 4.3. In Section 4.4, I provide an overview of the component 1 of the framework for the identification of SaaS service factors. In Section4. 5, I give an overview of the component 2 of the solution framework that is developed for imputing the missing sentiment intensity scores of the SaaS service factors. In Section 4.6, I discuss component 3 of the solution framework by providing an overview of the component 3 developed for forecasting the sentiment intensity scores of the SaaS service factors. In Section 4.7, I explain the modelling of SaaS service factors as trust factors by providing an overview of component 4 of the solution framework. I conclude the chapter in Section 4.8.

## 4.2 TRUST MODEL FOR SOFTWARE AS A SERVICE

Trust is subjective in nature and the concept of trust has been extensively researched in existing literature in several research and application domain. From generic to domain specific, there are several formal definitions of trust in the existing literature. In this thesis, I define the trust in SaaS that is derived from the definition by Olmedilla et al. (2005) with extended specifics. I define trust as:

"a measurable belief from one entity onto another in delivering a service or perform a well-defined task. An entity is a person, business or service. Trust is time-dependent and the level of trust vary with time. Trust is multi-faceted and is rooted in more than one factor. A complete trust level can only be established in an entity, when the information regarding all the factors is available. The importance of the trust factors differs from one another in the computation of the final trust value and trust level"

Based on the above definition, a trust model for SaaS, computes the trust level in SaaS products for the current or future point in time by utilizing the information from all the trust factors.

The detailed explanation of trust model for SaaS along with the implementation is presented in Chapter 8.

## 4.3 OVERVIEW OF PROPOSED SOLUTION

In Chapter 3, I have explained the importance of trust in the decision-making process by the customers in acquiring SaaS products. I have explained the challenges and issues related to trust assessment of the SaaS products that need to be addressed as part of the trust assessment methodology that has the capability to assist the customers in their decisions. In Chapter 3, I have also highlighted the importance of the architectural best practices in designing the cloud applications especially Software as a Service.

In general, cloud service consumers especially SaaS consumers, have a very basic knowledge of the physical implementation and functionalities of the cloud applications and services. The cloud service providers on the other hand have great visibility and ability to access these cloud applications and services from a standard perspective. These standards are set independently by each cloud service provider according to architectural best practices. The general guidelines and general design principles in this regard that is put forth by Amazon Web Services (AWS) is called as pillars of the well-architected framework (AWS, 2018). The AWS well-architected framework provides guidance in five conceptual areas or pillars namely, security, reliability, performance efficiency, cost optimization and operational excellence.

The second top cloud service provider is Microsoft. Microsoft Azure is the second largest cloud infrastructure provider (Inc, 2017). Similar to AWS, Microsoft Azure also suggests five pillars of software quality for successful cloud applications. The five software quality pillars suggested by Microsoft Azure are the same as AWS. However, previously Microsoft Azure well-architected frame

work included pillar such as scalability, availability, resiliency, management and security (Azure, 2018).

In this section, I give an overview of the proposed trust management framework that is capable to model the trust in SaaS products. The framework addresses all the research issues identified in Section 2.6. The proposed framework shown in Figure 4.1, has four main components and is designed specifically to address and solve the research issues mentioned in Chapter 3. In Section 4.2, I provided a formal definition of trust and trust model for SaaS. According to the definition, trust is multi-faceted and is rooted in multiple factors. The proposed trust model must be provided with information regarding all the trust factors. The trust factors are identified based on the architectural best practices used to design and develop cloud applications and SaaS. The factors in architectural best practices (as explained in the previous chapters) is referred to as the service factors of the SaaS products, in this thesis.

The first component of proposed framework addresses the first three research issues highlighted in Section 2.6. It includes tasks such as collecting the SaaS customer reviews from multiple available sources, identification and categorization of reviews under the service factors.

The second component of the framework addresses the issue of missing information highlighted in Section 2.6. It utilizes the classification model from component 1 to identify the service factors from the text reviews. The second component has two main tasks. The first task is focused on computing the sentiment intensity scores for the text reviews and review segments. The second task of this component deals with developing imputation model that imputes the sentiment intensity scores of the missing service factors.

The third component provides the capability to forecast the sentiment intensity scores of each service factor of a given SaaS product. It provides the solution for the forecasting issue mentioned in Section 2.6. It utilizes the historic sentiment intensity scores of the service factors for a given SaaS product and forecasts these values in future time spots. Finally, the last component of the framework, models the trust levels of the SaaS products by utilizing the classification model and sentiment intensities from component 1 and component 2. During the trust modelling and computation in component 4, for the SaaS products with missing information the imputation model from component 2 is utilized to impute the missing sentiment intensity scores. In order to compute the trust levels and trust score for a given SaaS product in the future time spots, the forecast model from component 3 is utilized initially to forecast the sentiment intensity scores for each service factor which are then used by the

trust model in component 4. Component 4 of the proposed framework addresses the research issue of dynamic ranking highlighted in Section 2.6.

Each of the components depicted in Figure 4.1, are explained in detail in the following subsections of this chapter.

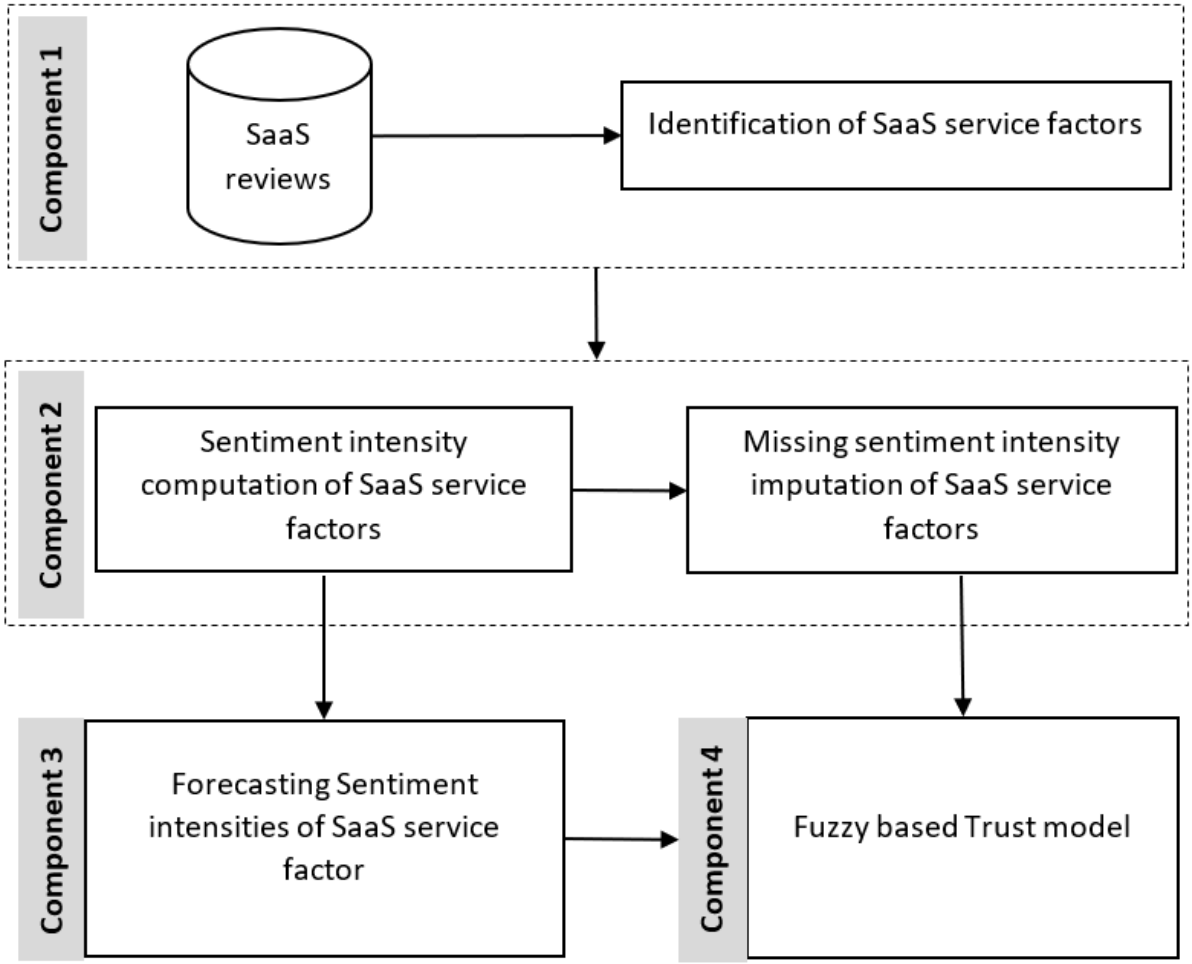



Figure 4.1. The proposed trust management framework for SaaS

## 4.4 OVERVIEW OF SOLUTION FOR COMPONENT 1: IDENTIFICATION OF SERVICE FACTORS

The main objective of this component of the proposed framework is to develop an ensemble classification model for text reviews from SaaS products. The classification model associates the text reviews with one of the SaaS service factors. The first component consists of the following two main tasks:

1. Collecting textual reviews for available SaaS products
2. Classification of the textual reviews into one of the service factors

The customer reviews for SaaS products are collected from multiple online sources. The advantage of reviews from multiple sources help to reduce bias at platform level. It means that collecting reviews both from service providers platform and independent platforms reduces the probability of bias in the collected reviews. A typical review by a reviewer on a given SaaS products includes the product details, textual reviews, review ratings, review dates and reviewer information. The initial raw text reviews are manually labelled as one of the SaaS service factors. The initial manual labelling of the raw textual data is important as the identification of SaaS service factors is based on the subjective interpretation of the architectural best practices. This results in a labelled dataset solely focused on the SaaS service factors. The classifiers trained on this labelled dataset (described in Chapter 5) will have the capability to automatically label customer reviews. The relationship of each task along with the flow and shape of the SaaS information is depicted in Figure 4.2.

To achieve the first task the following steps are involved:

- Scrap review data from multiple sources
- Pre-process the text data by text cleaning techniques
- Manually label the text reviews for each SaaS product

The next task in this component is to design an automated system that can classify any text review into one of the service factors. The solution for the identification and classification of SaaS service factors from customer text reviews is designed and implemented under the multi-class text classification setup also known as text categorization. Text classification is a mature field and the methods applied for text categorization has proven to be effective on a number of text corpora. However, the classification of text reviews under the SaaS service factors from SaaS textual reviews has not been explored in the literature.

The following steps are involved in the second task:

- Selecting the relevant text classification methods
- Tunning the hyper-parameters and train on the labelled text data
- Designing an ensemble model by selecting the high performing trained classification models

As shown in Figure 4.2, after the completion of first task in this component of the framework, the textual reviews along with the associated service factor labels are fed to the second task of this

component. The outcome of second task results in output data that has the identified service factors for a given SaaS product. Component 1 of the proposed framework is the foundation of this thesis as all of the other components of the framework utilized the trained classification model to identify and categorize textual reviews for SaaS products. The proposed ensemble model achieves the textual categorization task with the accuracy of 86%, outperforming the selected well-known classification methods. K-NN's performance is the lowest with the accuracy of 38%.

Throughout this thesis, the review data collected, processed and the data generated as output of the classification model available to the remaining components. The data passed from component 1 acts as the core dataset for the remaining components where they are further processed based on the specific requirements. In Chapter 5, I explain the detail design and implementation of component 1 of the framework.

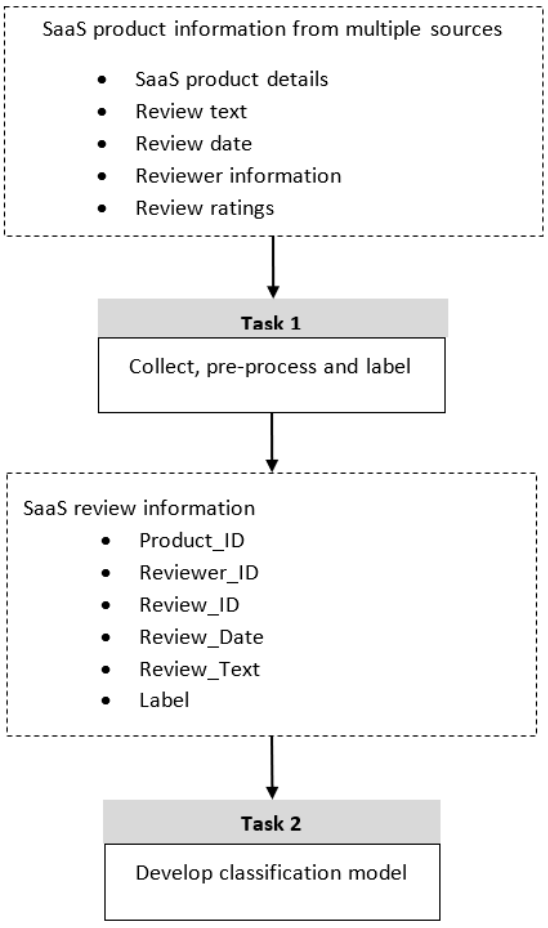



Figure 4.2. Overview of classification model

## 4.5 OVERVIEW OF SOLUTION FOR COMPONENT 2: IMPUTING MISSING SENTIMENT INTENSITY SCORES FOR SERVICE FACTORS

Based on the definition in Section 4.2, a trust in a SaaS can only be established when the information regarding all the trust factors is available. The main objective of the second component of the proposed framework is to develop an imputation model for the SaaS service factors with missing sentiment scores. The second component of the proposed framework consists of three main tasks:

1. Segmentation of text reviews for the collected SaaS products
2. Computation of sentiment intensity scores for each review segment
3. Imputation of sentiment intensity scores for the missing service factors

The first task of the second component involves two steps. In the first step, the text reviews are separated into small text segment following linguistic structure that defines a sentence. This step is important as in general, a single customers reviews consist of feedback regarding more than one service factor. The segmentation step helps to identify these service factors from each sentence separately. The second step of this task utilizes the classification model developed in the first component to categorize each of the review text segments into one of the service factors.

The second tasks of this component compute the sentiment intensity scores for each of the text segments. The text segments at this stage are already associated with a label that categorizes them under one of the service factors. In few studies, the customer sentiment is calculated on three level scale namely, negative, neutral and positive. However, I use the sentiment intensity of the text reviews which is the requirement for modelling the trust level and trust scores implemented in component 6 of the proposed framework (explained in Section 4.7). After the completion of the second task, the newly created dataset depicts text reviews in small segments where each text segment is associated with a label indicating the service factors and a numeric sentiment intensity score.

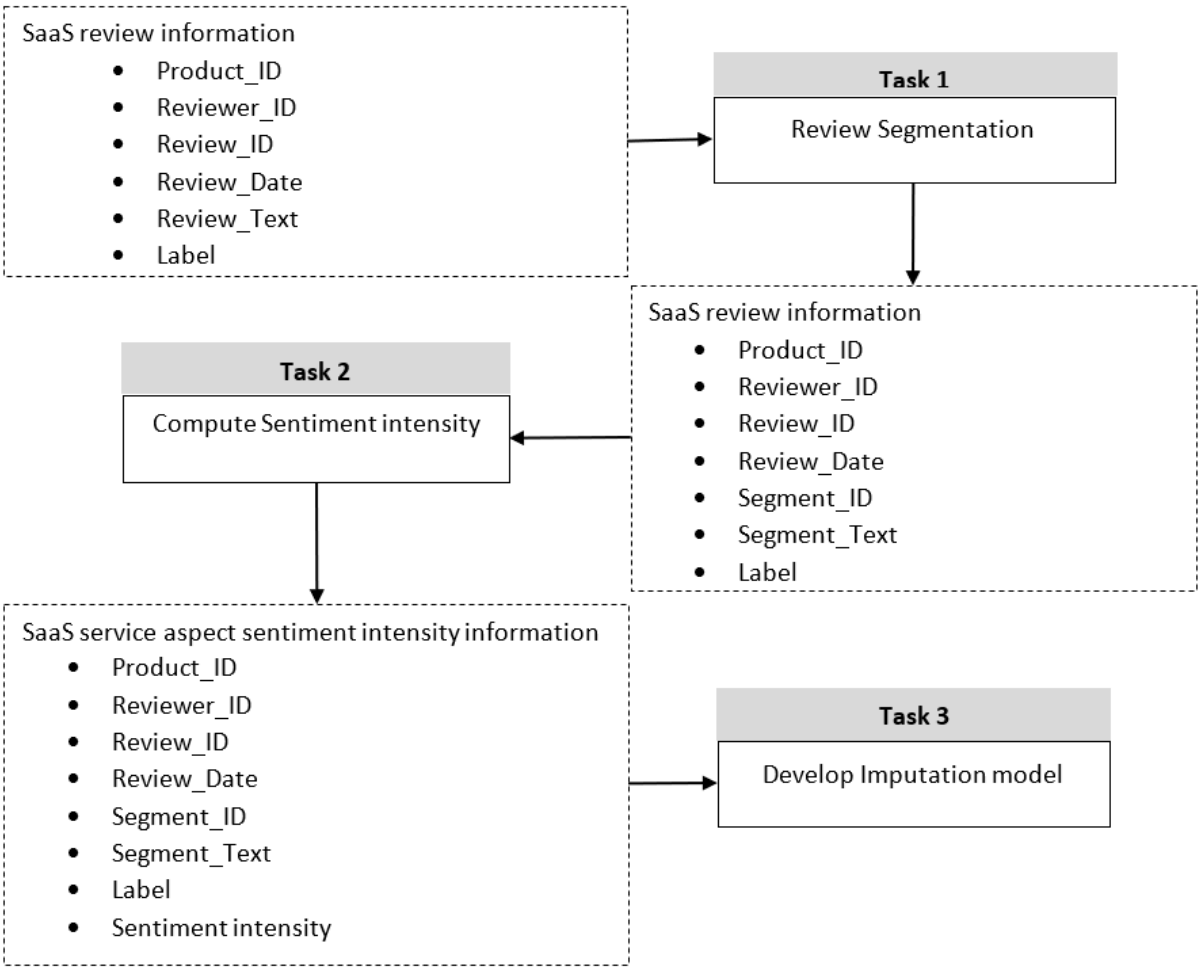



Figure 4.3. Overview of the imputation model

The third task of this component is to develop a model that identifies the service factors with missing sentiment intensity scores and imputes them accordingly. The solution for missing numerical sentiment intensity scores for SaaS service factors is designed and implemented under the missing values imputation setup. Missing values has been studied and several methods have been proposed in literature in different domains. However, the imputation of missing sentiment intensity scores of SaaS service factors has not been studied in current literature.

Figure 4.3, depicts the main tasks of this component, their relationship and the flow of structured data among them. As shown in Figure 4.3, the data used by task 1 is the output from component 1. After task 1 is successfully performed, the output data is converted into small text segments for each SaaS product and each text segment has an associated label. This data is fed to task 2, where, for each text segment a numeric sentiment intensity score is calculated and associated with the class label (service factor). Finally, the last task of this component identifies the service factors for the SaaS products that has missing sentiment intensity scores and develops an imputation model.

The output data of this component is available and utilized by the trust model briefly explained in Section 4.7 of this chapter. The detailed design and implementation of component 2 of the framework is presented in Chapter 6 of this thesis.

## 4.6 OVERVIEW OF SOLUTION FOR COMPONENT 3: FORECASTING SENTIMENT INTENSITY SCORES FOR SERVICE FACTORS

As highlighted in Section 4.2, a trust model for SaaS is capable to compute the trust level of SaaS products at the current as well as in the future point in time. In order to compute the trust level for the future time spots, the information regarding all the trust factors must be available for the given future time spot. The main objective of the third component of the proposed framework is to develop a forecast model for the sentiment intensity of the SaaS products. The third component consists of the following main tasks:

1. Preparing and modelling sentiment data as time series
2. Forecasting the customer sentiment for the identifies service factors

The first task of this component is to prepare the sentiment data in the form of time series and model different features of the time series data. Data preparation for forecasting is important and helpful in the selection and development of the appropriate forecast model. This task involves the application of the standard data modelling techniques to identify the features and behaviour of the time series data and the transformation into acceptable form for input to the forecast model.

The second task of this component of the proposed framework is to develop a forecast model that is capable to forecast the sentiment scores for each service factor in the future time spots. The forecast component is crucial to the proposed framework as for any future business interaction, it is important for the customers to be able to acquire trust levels of SaaS products in the future time slots. Such requirement is achieved using time series forecasting methods that utilizes the historic information to forecast future values. Time series forecasting has been studies widely in literature and high performing methods has been proposed in several domain such as statistics and machine learning. Forecasting sentiment intensity scores of SaaS service factors has not been studied in the current literature.

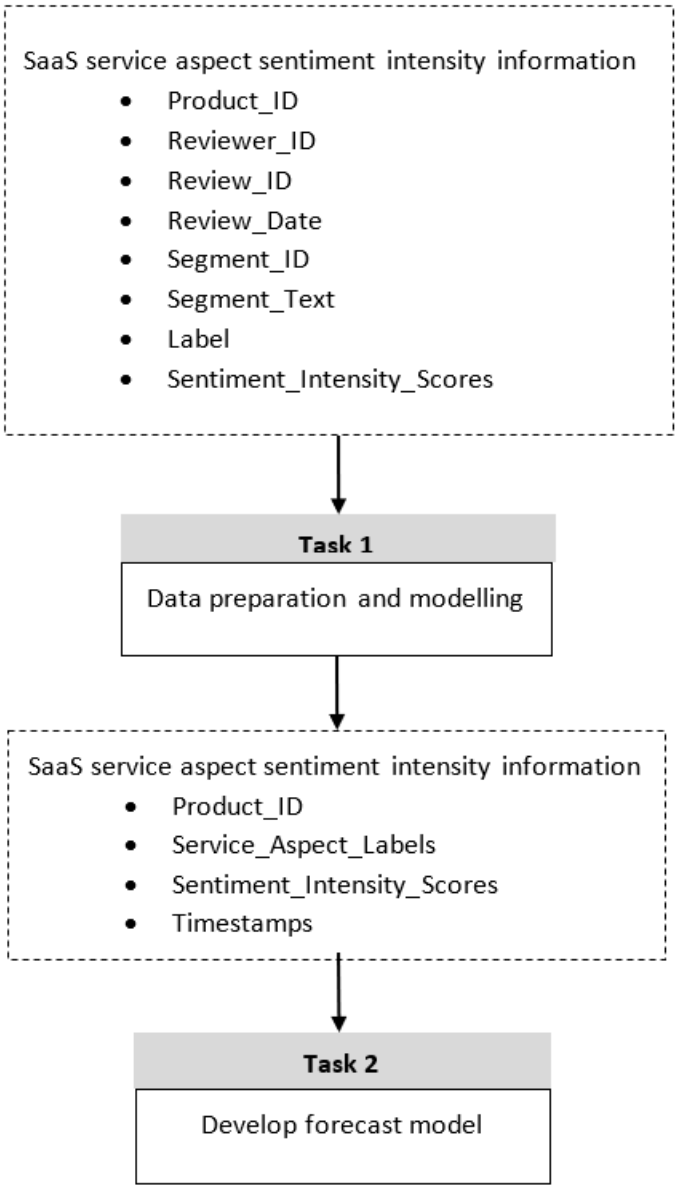



Figure 4.4. Overview of the forecast model

Figure 4.4, depicts the relationship among the tasks of component 3 and shows the type of information acquired and produced by each task. Task 1 utilized the data from the component 2 of the framework and prepares a time series data for each SaaS product. After the modelling the data during task 1, the time series data is used by task 2 for the development of the forecast model. The forecast model is trained on the time series data and is capable to predict the sentiment scores for the service factors in the future time spots.

The forecasted values by this component of the framework is available to be used by the trust model introduced in Section 4.7 of this chapter. The detailed discussion and development of the time service forecast model for the SaaS sentiment forecasting is presented in Chapter 7 of this thesis.

## 4.7 OVERVIEW OF SOLUTION FOR COMPONENT 4: MODELLING TRUST LEVELS IN SOFTWARE AS A SERVICE

Based on the definition in Section 4.2, trust is a measurable belief and is multi-faceted. It is measured from the available information for all the trust factors with different level of importance. The main objective of the fourth component of the proposed framework is to develop a trust model for the SaaS products that utilizes the sentiment intensity scores for all the service factors as trust factors, to compute the final trust level and trust value. To support customers decision for any current and future SaaS signups and purchases, it is important to provide a realistic trust value for each SaaS product. This proves to be valuable when the trust values are computed directly from the previous users' sentiments towards different service factor compared to a standalone rating for a SaaS product. Some of the benefit of the sentiment-based trust model over the other rating systems include:

- The ability to choose SaaS products based on custom preferences
- An automated system that extracts the sentiment intensity towards different service factor from lengthy reviews
- Providing efficiency and ease in the decision-making process on customer side by avoiding the reading of hundreds of user reviews

Providing an automated feedback system that identifies the strength and weaknesses categorically which can be utilized by the SaaS providers to improve specific factors of their products.

The two main tasks involved in achieving the objective of component 4 are:

1. Computing the importance of each service factor of the SaaS products
2. Computing the trust level and trust scores for the SaaS products

The first task of this component utilizes the data from Component 2 of the proposed framework. It develops a metrics through which it computes the importance of each service factor and rank them accordingly.

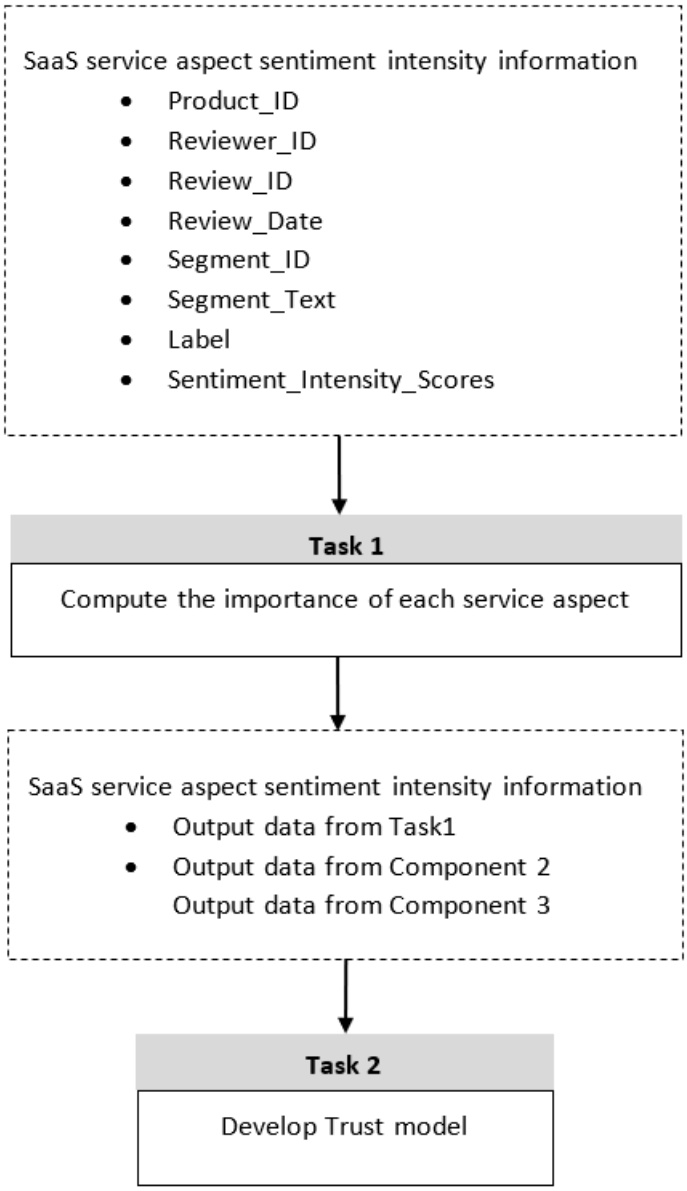



Figure 4.5. Overview of the trust model

The second task of this component is to develop a trust model that computes the trust level and trust score of the SaaS products. The trust model considers the SaaS service factors as the input trust factors and uses the ranking scores from the previous task as weights for the trust factors in computing the trust scores. The proposed model utilizes the output data from Component 2 of the framework to compute the current trust score of SaaS products. In order to compute the trust score for a SaaS product in the future time spot, the trust model utilizes the output data from the forecast model in Component 3 of the proposed framework. Based on the concept of trust and trustworthiness, the trust levels and trust scores for the SaaS products is modelled though fuzzy inference system. Fuzzy inference models are well known in literature and proved to be effective in many application domains.

However, fuzzy trust inference model for SaaS products that utilizes the sentiment intensity scores of service factors has not been explored in existing literature.

Figure 4.5 depicts the tasks and the associated data as input and output to these tasks. From Figure 4.5, it can be seen that the trust model developed in tasks 2, utilizes data from three different processes. The output data from the first tasks of this component is used to rank the trust factors and the sentiment data for the SaaS products from the output of Component 2 and Component 3 is used to compute the trust level and trust score for the SaaS products.

In Chapter 8, I present the details for the development and implementation of the proposed weight metrics and the fuzzy based trust model.

## 4.8 CONCLUSION

In this chapter, I presented a general overview of the proposed framework. Each component of the framework is briefly explained along with the associated tasks that are designed to solve the issues discussed in Chapter 3. The proposed framework is composed of four main components. The first component includes tasks for collecting SaaS reviews, processing the text reviews and developing a text classification model. The sentiment intensity of the text reviews is computed and the imputation model for the missing sentiment scores for the service factors is developed as part of the component 2 of the framework. A forecast model is developed for forecasting the sentiment intensity values of the SaaS products utilizing the historic time series values in component 3. Finally, in component 4 of the proposed framework a trust model is developed for the SaaS products.

In the next chapter, I present the details of the first component of the proposed framework that leads to the development of a text classification model for the SaaS service factors from customer reviews.

# Chapter 5:
# An Ensemble approach for identifying SaaS service factors

## 5.1 INTRODUCTION

In the previous chapters, I highlighted the core component of the proposed trust framework that identifies the key factors of the SaaS products which effects the trustworthiness of these services. The primary issue addressed by the framework is to identify these key factors in SaaS marketplace. As explained in Chapter 4, the software quality pillars and the pillars of the well-architected frameworks suggested by the major cloud service providers is used as key factors to identify the trust of cloud applications and services. These service quality pillars which I refer to as service factors in this study, is learned from the customer reviews of the SaaS products in this chapter. This brings this primary issue under the umbrella of multi-class text classification problem which is also known as text categorization in literature. In this chapter, I explain each stage of the proposed approach for text-classification from SaaS product reviews in details.

The SaaS products reviews are crawled and collected from multiple online sources and are prepared by going through several pre-processing steps to convert them in an acceptable format for the methods. From textual to numerical matrices, methods such as bag-of-words and TF-IDF are applied to achieve the final workable format. I explain and implement 11 machine learning methods that are commonly used for text classification with their strengths and weaknesses. All the selected text classification algorithms are thoroughly cross-validated using the SaaS reviews dataset and their core parameters are estimated. The cross-validation is performed on both the original and resampled data using stratification and SMOTE to further examine the effects of class-imbalance. The fitted models are then evaluated on test data and their performance is compared using several error measures. Also, there are performance are tested using Freidman significance test and Nemenyi's post-hoc test to examine the significance difference among their performances. In the final stage of the proposed approach, and ensemble of the best performing classifiers are created. The problem addressed in the chapter with the proposed solution has been published in (Raza et al., 2019).

In the next section, I present an overview of the machine learning approaches for text classification. In Section 5.3, the detailed stages of the proposed approach for text-classification of SaaS customer

reviews are presented. In Section 5.4, I present the details of the experiments and evaluation results of all the 11 selected classification methods and the proposed ensemble. Section 5.5 concludes this chapter.

## 5.2 MACHINE LEARNING BASED TEXT CLASSIFICATION

The core concept of machine learning is to learn a function that can map input variables to output variable(s) using the provided training data. For example, in this study, the machine learning algorithms estimate the mapping function $Y = f(X)$ that maps the input data samples in $X$ (customer reviews) to the output labels in $Y$ (service factors). There are a number of machine learning algorithms that works with different assumptions about this learning function. They can be broadly categorized into parametric and non-parametric ML algorithms. The parametric machine learning algorithms summarizes the data with a finite set of parameters. While the non-parametric algorithms are parameter-less, they perform the classification tasks using one of many techniques such as, instance-based learning and searching, using similarity and distance measures among samples.

While using machine learning algorithms with textual data also known as text categorization, takes few extra steps in preparing the dataset. ML algorithms works comfortably with numerical input data. Therefore, the textual data has to go through a number of steps using different methods at each step to prepare it in a workable format. In the next, I briefly discuss some of the famous machine learning classification methods used for text classification.

## 5.3 AN ENSEMBLE MACHINE LEARNING APPROACH FOR MULTI-CLASS TEXT CLASSIFICATION OF SAAS REVIEWS

As discussed in Section 5.1, the main objective of this component is to identify the SaaS service factors from the customer reviews using machine learning methods and develop an ensemble of the best performing ML methods. As discussed in Chapter 4, the concept of service factors is derived from the pillar of software quality and well-architected frameworks. To achieve this objective, the sequence of steps in our proposed approach is depicted in Figure 5.1. The proposed approach is designed based on the nature of the problem that this study is addressing.

### 5.3.1 Overview of SaaS service factor identification from customer reviews

The four main stages of the proposed approach are *data pre-processing stage, parameter estimation stage and model evaluation stage.* After the evaluation of the models implemented in this study, an ensemble of the best models is developed in the final stage.

In the first stage of the proposed approach, data is prepared which involves multiple well-defined steps. In the pre-processing step the customer reviews are crawled and collected from different websites. The reviews are labelled according to the service factors and cleansed for special characters. This is followed by transforming the textual data into metrics with numerical values by creating a feature vector set. After the feature vector is created, the feature selection also known as dimensionality reduction is performed which generally improves the efficiency of the models. Most of the classification models work well when the dataset and all the values in each feature are in a suitable format i.e., they fall within the same scale, hence feature scaling is performed on the reduced feature vector in the last step of this stage. These steps make the data ready and understandable for the machine learning methods to work on (Brownlee, 2016b). The effective pre-processing of textual data is crucial to obtain an acceptable performance and better quality of text classification (Richert & Coelho, 2013) (Trstenjak et al., 2014).

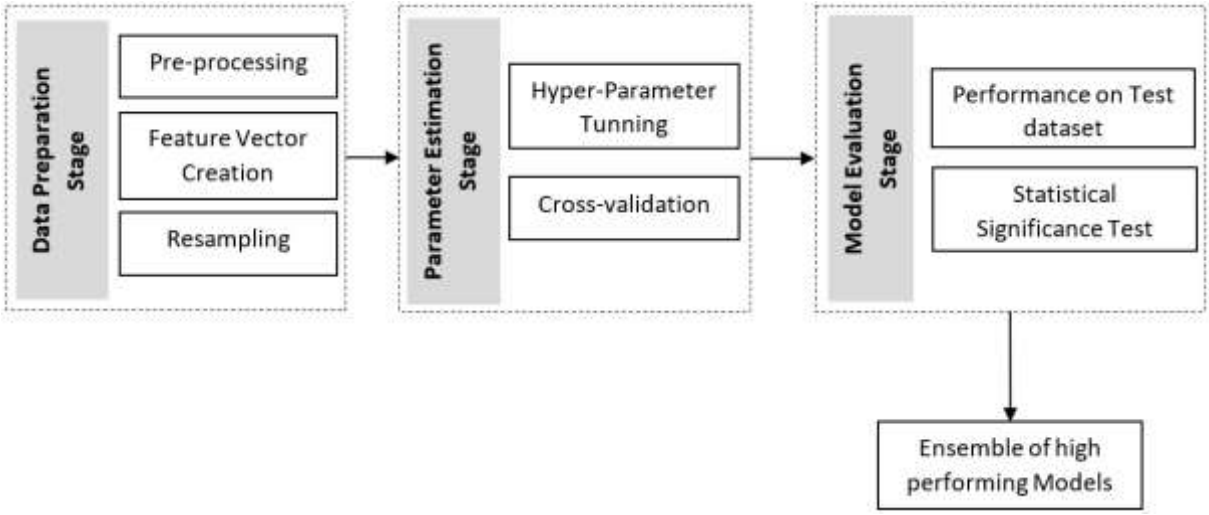



Figure 5.1 Sequence of steps for the SaaS service factors identification approach

The next stage of the proposed approach involves the selection of the appropriate ML algorithms for the classification. This preliminary investigation involves, the identification of the famous approaches applied in large text classification problems in both the parametric and non-parametric categories. Also, a set of error measures is investigated and selected that can help to examine the performance of the models from multiple perspectives.

After the potential methods have been selected, the next stage will thoroughly investigate each method by training them on the dataset prepared in the first stage, resampled version of the datasets, cross validating the estimator's multiple times on the dataset and model specific hyper-parameter tunning. As we are dealing with multi-labels, the resampling ensures that the data considered to train

each label are consistent across all of them and during the cross-validation, the training dataset is sub-divided into different folds to assist in choosing the best parameter from the available ones that will give the most accurate output. During the training process, each machine learning algorithm needs to be tuned to the parameter values on which it gives an output that is within the defined agreeable threshold of error.

In the following stage, the fitted models on the dataset are introduced to test dataset which is new to the models. The performances of each model are evaluated and compared among them using the error measures discussed in the second stage of the proposed approach. Furthermore, a statistical significance test is conducted to confirms the performance difference among the selected models. On the basis of this, the models are ranked and the best models are selected for the final stage in which an ensemble model is created that fits best with the textual dataset collected for this study. Table 5.1 shows some of the symbols and notations that are used in this chapter.

Table 5.1. Notations of the variables used in the experiments

| Notation | Description |
|---|---|
| $R^d$ | Vector space with $d$ dimensions, where $d = samples \times features$ |
| $X$ | Training samples (matrix), where $X \subset R^d$ |
| $x_i$ | $i^{th}$ training sample (vector), where $x_i = \{x_1, x_2, \dots, x_n\} \in X$ |
| $Y$ | Set of class labels |
| $y_i$ | $i^{th}$ class label in label set $Y$, where $y_i = \{y_1, y_2, \dots, y_{10}\} \in Y$ |
| $C$ | An individual class ($C_i$ is the $i^{th}$ class) |
| $TP$ | True positive ($TP_i$ is the true positive value for the $i^{th}$ class) |
| $TN$ | True negative ($TN_i$ is the true negative value for the $i^{th}$ class) |
| $FP$ | False positive ($FP_i$ is the false positive value for the $i^{th}$ class) |
| $FN$ | False negative ($FN_i$ is the false negative value for the $i^{th}$ class) |

As depicted in Figure 5.1, there are four main stages of the proposed solution. Each stage along with their steps and set of tasks are discussed in the following subsections.

### 5.3.2 Data preparation

The first stage of the proposed approach deals with the data preparation which is used by the selected machine learning algorithms in the following stage. The data preparation is achieved through multiple steps such as pre-processing, creating feature vector from text data, assigning weights to features, reducing the dimensions and scaling the feature space onto a single scale. The first step in data preparation is called the pre-processing which is described in the following subsection.

***Pre-processing***

The goal of the data pre-processing phase is to collect and process the customers' reviews so that they are in a format in which machine learning algorithms are able to work. The following steps are carried out in this phase:

*Step 1: Data collection*

In this step, the reviews from cloud consumers are collected using a web crawler. The crawler used in our framework is developed in the Python programming language using Scrappy (Scrapy). To address the drawback of having a limited number of reviews from a single source, we crawl three different sources, namely the AWS marketplace (AWS, 2017), GetApp.com (GetApp, 2017) and Serchen.com (Serchen, 2017) to obtain a comprehensive representation of user opinions. As a result, we harvested and collected 2,297 unique reviews from the AWS marketplace, 15,545 reviews from GetApp.com and 11,519 reviews from Serchen.com. In total, 29,361 reviews were collected by combining reviews from all the three sources.

*Step 2: Labelling*

In this step, each sample (review) in the training dataset is manually labelled using the labels described in Table 5.2. Each of these labels represents a unique service factor and pillar of SaaS, including an *unknown (N)* label, which is assigned to those reviews that do not mention any service pillars.

In the function, $Y = f(X)$, $X$ is the input data samples (customer reviews) and the output $Y = \{A, C, E, L, M, N, O, P, R, S\}$ represents the set of output labels.

However, there are two important points which must be noted at this stage.

- We consider this to be a multi-class single label problem. In other words, in our approach, only a single label for a review is assigned even in cases when the review has comments related to more than one factor of the SaaS products.
- The process of assigning a label to a review is a subjective decision by the experts.

For instance, the customer review, "… offers an excellent cost-effective service, but the service responded very slowly during the holiday season", indicates more than one factor of the SaaS product. Based on the description of the factors explained earlier, this review relates to *performance*, *scalability* (indicating its incapability to scale on increased load) and *cost optimization* (indicates the

service is delivered at an acceptable cost). However, only one label from the possible cases is assigned to the review and the decision as to which label is assigned is subjective.

Table 5.2. Service factors with associated tags

| Service factor | Label/Ta | Numeric |
|---|---|---|
| Unknown | N | 0 |
| Security | S | 1 |
| Reliability | R | 2 |
| Performance | P | 3 |
| Cost optimization | C | 4 |
| Operational | O | 5 |
| Availability | A | 6 |
| Scalability | L | 7 |
| Resiliency | E | 8 |
| Management | M | 9 |

*Step 3: Data cleansing*

In this step, the reviews are cleansed of special characters and symbols. The reviews normally come with a lot more than just informative text, such as HTML tags, punctuations errors, characters without any expressive meanings and so on. These words hinder the process of learning about the text and text classification. So, in this step, the following techniques are applied to clean the reviews and convert them into standard text strings:

- Remove punctuation, emoticons and other special characters;
- Convert text to lowercase;
- Remove stop words such as ‘a’, ‘an’, ‘of’ etc.;
- Convert words to their root forms using lemmatization.

Table 5.3. Pre-processed review samples with associated labels

| | Reviews | Label |
|---|---|---|
| **0** | postrouting prerouting possible seem nice appliance vpn nat ipsec tunnel possibility use …… | **O** |
| **1** | fantastic always last minute need move part aws solution multiple external dedicate server …. | **M** |
| **2** | good product phenomenal support migrate away nat instance six virtual private gateway ……. | **C** |
| **3** | 2 hour make progress quick start guide anything date easy read think make assumption clear … | **M** |
| **4** | work like charm use product many year amaze project product 200 000 user globally build …. | **S** |

Compared with stemming, lemmatization produces grammatically correct forms of individual words. Stemming creates singular instances of words, but in the process, can create non-real words as well. Table 5.3 depicts few samples from the dataset after the pre-processing steps.

***Feature Vectors creation***

The main objective in this subsection is to create numerical feature vectors from the labelled and cleansed text reviews. This process involves three steps explained as follows:

*Step 1: Numerical feature vector creation using Bag-of-Words*

The bag-of-words approach on the cleansed dataset is used to create a feature vector. The features in this case represent the words which are mentioned in the customer reviews. Converting the textual reviews to a vocabulary of bag-of-index terms is important as they do not themselves have any internal structure or ordering (Lewis, 1998) (Raschka, 2014). The core idea of the bag-of-words approach is to represent each word in the text documents (in our case all the review text) in the form attribute-value and create a vocabulary from the entire corpus. The attribute-value representation shows how often each word from the vocabulary appears in each review text. The vocabulary of the bag-of-index terms which is in textual form is then converted to numerical feature vectors to perform machine learning on them for classification. This step is necessary since the machine learning algorithms require inputs of fixed length vectors with numerical features.

The objective of this step is achieved in the following three tasks,

- tokenization
- vocabulary building
- encoding

During the tokenization task, each review is split into tokens based on the whitespaces. Each word in the review is converted into a single token during this step. In the vocabulary building task, all the unique words (tokens) from each review are collected, numbered in alphabetical order and added to the vocabulary. The created vocabulary is referred to as the feature vector for this corpus and each word in the vocabulary represents a feature. Once the vocabulary is created from the entire corpus of reviews, then for each review, the number of occurrences of each word is counted. In the encoding task, which is also called vectorization, each review is then encoded with numerical codes which represent the number of times each feature from the vocabulary has appeared in that review. The

resultant output is known as the feature vector. The feature vector shows the built vocabulary for each review with the number of times each word from the vector occurs in that review.

*Step 2: Feature weights and Dimensionality reduction using Tf-Idf*

At the end of the previous step, there may be many feature columns that contain an encoded value of zero for a given review. This will result in a long list of features with zero values that will impact negatively during the classifier learning step. To improve the performance and time required to train the classifiers, in this step, the extracted features from the previous step are pruned to have a better representation of the review dataset. This is done using the following sub-steps:

a) *Only considering those features for the training and testing of classifiers that occur more than a defined threshold level:*

The objective of this sub-step is to omit those features from the built vocabulary (and hence the training process) that do not occur often in the review dataset. We achieve this by setting a threshold variable of *five*. In other words, we only consider those features from the built vocabulary that appear in at least *five* reviews. This threshold level of five is not fixed and can be changed, but this will help to reduce the number of features from the feature vector in the next steps.

b) *Ascertain the weight of those features from the corpus that occur frequently in a review:*

Some features will appear many times in particular reviews but not in all of them. These particular features are highly descriptive of the content of the specific reviews in which they occur. Hence, assigning higher weights to these features and linking them to the service factor that has been assigned to that review will have a positive impact during the process of training and testing of the classifiers. The weights to be assigned to those features are determined using the term frequency-inverse document frequency (tf-idf) method (Salton, 1991) (Raschka & Mirjalili, 2019) (Salton & Buckley, 1988) (Muller & Guido, 2016). In our case, term refers to the features and a single customer review represent a document. tf-idf determines the number of times a term occurs in a review (known as term frequency) over the number of times it occurs in all the documents (inverse document frequency). In other words, the tf-idf of each feature is the product of the term frequency (*tf*) and inverse document frequency (*idf*), calculated as:

$$tf - idf(t,d) = tf(t,d) \times idf\ (t,d) \tag{5.1}$$

where

*tf (t, d)* is the number of times the term (feature) *t* occurs in a document (review) *d*.

$idf\ (t,d)$ for the same term is the number of times it occurs in all the documents.

Equations 5.2 and 5.3 calculate the tf-idf score for term *t* appearing in document *d* (review).

$$idf(t,d) = log\frac{1+n_d}{1+df(d,t)} \tag{5.2}$$

$$tf - idf(t,d) = tf\ log(\frac{1+n_d}{1+df(d,t)}) + 1 \tag{5.3}$$

where

$n_d$ represents the total number of documents in the training set;

*df(d,t)* represents the number of times the term occurs in documents *d*. (Trstenjak et al., 2014) (Joachims, 1997).

Table 5.4. Sparse matrix representation of review samples with TF-IDF scores for features

| **Review 0** | | **Review 1** | | **Review 2** | | **Review 3** | | **Review 4** | |
|---|---|---|---|---|---|---|---|---|---|
| **(Row, Feature)** | **TF-IDF** | **(Row, Feature)** | **TF-IDF** | **(Row, Feature)** | **TF-IDF** | **(Row, Feature)** | **TF-IDF** | **(Row, Feature)** | **TF-IDF** |
| (0, 6501) | 0.20367 | (1, 1350) | 0.33553 | (2, 19783) | 0.10394 | (3, 2769) | 0.24726 | (4, 17607) | 0.11988 |
| (0, 958) | 0.21543 | (1, 4932) | 0.13761 | (2, 3940) | 0.12237 | (3, 19671) | 0.25105 | (4, 6632) | 0.14978 |
| (0, 20730) | 0.25179 | (1, 1349) | 0.31869 | (2, 3663) | 0.10777 | (3, 11024) | 0.21063 | (4, 17536) | 0.16844 |
| (0, 6291) | 0.24549 | (1, 14162) | 0.29815 | (2, 7938) | 0.09931 | (3, 17328) | 0.35876 | (4, 20696) | 0.17938 |
| (0, 19636) | 0.22592 | (1, 25488) | 0.26795 | (2, 19818) | 0.12032 | (3, 7739) | 0.43976 | (4, 19937) | 0.17397 |
| (0, 24262) | 0.25179 | (1, 5885) | 0.33553 | (2, 4637) | 0.08698 | (3, 2766) | 0.16636 | (4, 13729) | 0.10972 |
| (0, 13101) | 0.25556 | (1, 15671) | 0.33152 | (2, 23122) | 0.07101 | (3, 23630) | 0.25781 | (4, 17577) | 0.08982 |
| (0, 4348) | 0.14230 | (1, 27224) | 0.33152 | (2, 3117) | 0.11209 | (3, 12649) | 0.23216 | (4, 25458) | 0.13810 |
| (0, 26180) | 0.16636 | (1, 17371) | 0.16777 | (2, 9694) | 0.09971 | (3, 11017) | 0.15038 | (4, 6545) | 0.05924 |
| (0, 6295) | 0.05992 | (1, 6800) | 0.17863 | (2, 4622) | 0.11019 | (3, 6122) | 0.21988 | (4, 17535) | 0.14175 |
| (0, 932) | 0.10309 | (1, 20247) | 0.09333 | (2, 26584) | 0.10288 | (3, 14255) | 0.11341 | (4, 704) | 0.14078 |
| (0, 1486) | 0.18135 | (1, 4841) | 0.10205 | (2, 21152) | 0.11558 | (3, 3510) | 0.27880 | (4, 25759) | 0.15131 |
| (0, 18771) | 0.17080 | (1, 1342) | 0.15675 | (2, 27176) | 0.09619 | (3, 17126) | 0.25456 | (4, 11178) | 0.18398 |
| (0, 11077) | 0.15256 | (1, 14153) | 0.17012 | (2, 20748) | 0.10572 | (3, 7723) | 0.27645 | (4, 3379) | 0.16312 |
| (0, 26290) | 0.14319 | (1, 25021) | 0.06469 | (2, 23833) | 0.09446 | (3, 17360) | 0.15577 | (4, 16040) | 0.09126 |
| (0, 3572) | 0.19056 | (1, 5884) | 0.24651 | (2, 6114) | 0.08013 | (3, 25021) | 0.10277 | (4, 18228) | 0.12632 |
| (0, 15805) | 0.16962 | (1, 16546) | 0.20913 | (2, 24090) | 0.10189 | (3, 19618) | 0.12736 | (4, 16459) | 0.13895 |
| (0, 23275) | 0.13102 | (1, 15632) | 0.15519 | (2, 19782) | 0.10012 | (3, 24091) | 0.08668 | (4, 3144) | 0.16509 |
| ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... |
| ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... |
| ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... | ..... |

Using the tf-idf method for scaling and assessing the term frequencies in feature vectors helps us to determine the most important features for a review which eventually improves the performance of the classifiers.

Table 5.4 shows the results of the TF-IDF calculation on the sample reviews from Table 5.3. In each review, the features that is present has a non-zero value and the absent features are set to zero. Table 5.4 shows a sparse matrix representation of each review features by showing only the features with a non-zero value and the feature location in the feature vector space. A TF-IDF score is calculated for each of the feature as depicted in Table 5.4.

*Step 3: Feature scaling*

The calculated TF-IDF score for each feature can vary over a wide range, hence having a rough and undefined numerical distribution. Most of the machine learning algorithms such as k-nearest neighbours (k-NN) (excluding decision trees and random forests) perform much better if the values for each feature in the vector set follow the same scale (Brownlee, 2016b). Feature scaling is generally performed to avoid one attribute dominating the distance measure. Normalization is one of the well-known approaches used for feature scaling. The normalization process is applied on the TF-IDF scores of each feature which converts all the term values on a common scale between 0 – 1. Scaling inputs to unit norms is a common operation for text classification or clustering, for instance. Traditionally, inputs are normalized (rescaled) to values between 0 and 1 (Brownlee, 2016b). In our work, we use the L2-norm, also known as the Euclidean norm (Muller & Guido, 2016) (Toman et al., 2006) to normalize the values over a scale of 0 to 1 using:

$$x_{norm} = \frac{x}{\sqrt{\sum_{k=1}^{n} |x_k|^2}} \tag{5.4}$$

where

$x$ is the feature;

k represents all the features in a review where feature x appears;

$x_{norm}$ is the normalized value of x;

x is the tf-idf value of the feature $\boldsymbol{x}$

***Resampling***

Class imbalance impacts the overall performance of the trained classifiers and in this study, we are dealing with class imbalance in the number of the training data available for each label. If such an

imbalance before the training is not addressed, it will lead to scenarios where the algorithm for a label which has large number of samples performs at an agreeable level while testing as compared to other labels which do not have a sufficiently large number of samples. As we are using 10-fold cross-validation for each of the classifiers in our experiments, it is very important that in each fold, there must be same number of samples from each class.

To achieve a good performance for the classifiers and better understanding of the dataset, resampling techniques are frequently used in data classification. In this study, along with the actual (imbalanced) dataset, I used two approaches to deal with the class imbalance and to improve the performance of classifiers.

1. ***Stratification***: Stratified sampling is well known approach (Muller & Guido, 2016) (Kohavi, 1995) (Mullin & Sukthankar, 2000) (Forman, 2003) used for the imbalanced sample distribution among classes that we are dealing with in this study. Stratified sampling also helps in dealing with the class imbalance problem. During cross-validated training, stratification uses a balanced proportion of samples from each class for each fold. As we are using k-fold cross-validation with stratification (or stratified k-fold cross-validation), each fold of the cross-validation will have the same percentage of the samples per class. Using stratified cross-validation also addresses the issue of having a consistent amount of data for each label thereby avoiding overfitting and scenarios where the trained model does not perform well on testing or new data as it has been fitted too much on the trained data.
2. ***Synthetic Minority Over-sampling Technique (SMOTE):*** Beside stratification during cross validation, I also used a well-known oversampling technique for imbalance datasets known as the Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002). SMOTE is an oversampling technique which creates synthetic samples from the minority classes to match the number of samples in the majority class. It uses k-nearest neighbours to select similar instances for creating synthetic samples. In our experiments we have created synthetic samples for all the classes to match the number of samples in the majority class by considering 5 nearest neighbouring instances. Figure 5.2 depicts the review samples per class (left) and the resampled samples using SMOTE (right).

In Section 5.4, the fitted classifiers performance using both stratification and SMOTE is compared for further evaluation.

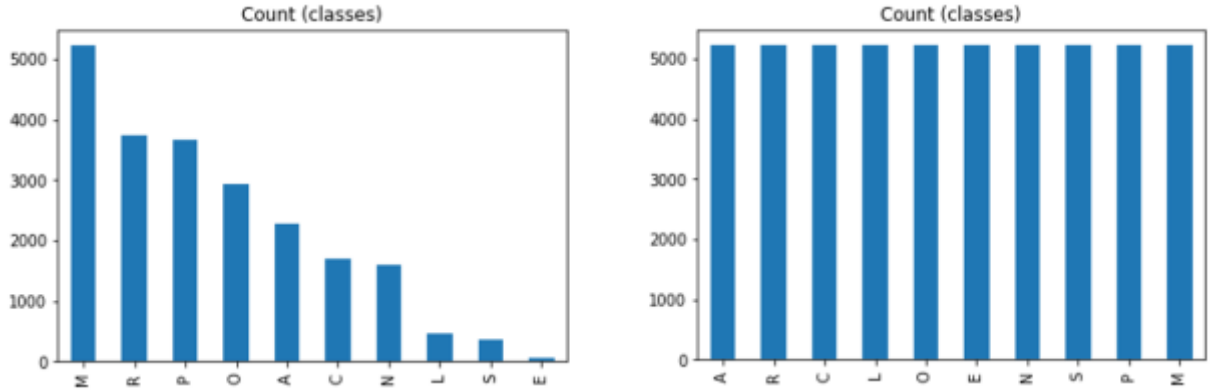



Figure 5.2. Number of samples per class before and after using SMOTE

### 5.3.3 Parameter estimation

The preliminary investigation of the machine learning algorithms that are suitable for the purpose of text classification is performed in this stage. The most relevant machine learning approaches for text classification shown in Table 5.5 are selected to model the problem addressed by this study.

Table 5.5 Selected text classification approaches

| **Text classification approach** | **Related references** |
|---|---|
| K Nearest Neighbours | (Muller & Guido, 2016) (Cover & Hart, 1967) |
| Naïve Bayes (Multinomial and Bernoulli) | (Lewis, 1998) (Raschka, 2014) |
| Rocchio | (Ittner et al., 1995) (Schapire & Singer, 2000) |
| Perceptron | (Crammer et al., 2006) (Crammer & Singer, 2003) |
| Ridge | (Hoerl & Kennard, 2000) (Yang et al., 2003) |
| Stochastic Gradient Descent | (Brownlee, 2016b) (Rumelhart et al., 1986) |
| Classification and Regression Trees | (Breiman et al., 1984) (Sakakibara et al., 1993) |
| Linear Support Vector Classification | (Joachims, 1998) (Zhang & Yang, 2003) |
| Passive Aggressive | (Crammer et al., 2006) (Martínez-Rego et al., 2015) |
| Logistic Regression | (Brownlee, 2016b) (Raschka & Mirjalili, 2019) |

To create an ensemble, the 11 machine learning algorithms from both the parametric and non-parametric branches are selected based on the key aspects of the problem such as:

- The learning is Supervised
- The output is multi-class
- The input is text
- The data has high dimensions

It must be noted over here that most of the text classification algorithms are sensitive to the type and nature of the dataset (Muller & Guido, 2016) (Schapire & Singer, 2000), they depend on factors such as the number of classes, class imbalance (number of samples per class), feature scaling, number of

training samples, number of features and so on. Furthermore, different algorithms follow different approaches to solve multi-class classification problems, which also affects their performance.

The main objective of this stage is to find the best fit for each selected method on the SaaS review dataset with using strategies such as cross-validation and hyper-parameter tunning.

***Hyper-parameter tunning***

The objective of hyper-parameter tunning step is to address the issue of overfitting and bias-variance trade-off. As discussed in Section 5.2, the goal of machine learning algorithms is to best estimate the mapping function or target function which maps input data to output variables. The ability of each algorithm to achieve this goal is measured by the values it obtains for the performance metrics discussed in the previous subsection. Generally, these predictions come with measurable errors that can be broken down into bias error, variance error and irreducible error (Brownlee, 2016b). Bias error occurs when the model makes simplifying assumptions to learn the target function. More suggestions about the form of the target function by the model results in high bias and vice versa. Variance error suggests changes to the estimate of the target function if changes are made to the training dataset or if new dataset is used. A high variance suggests that large changes will be expected to the estimates made by the target function with the introduction of new training data. High variance is directly related to overfitting which means that the model becomes too complex by learning very complex details of the training data and does not generalize well to unseen data.

Table 5.6. Selected parameters for tunning

| **Classification methods** | **Selected Parameters for tunning** |
|---|---|
| kNN | *K, Distance metric* |
| NB (Multinomial) | *Smoothing parameter 'alpha'* |
| NB (Bernoulli) | *Smoothing parameter 'alpha', binarizing threshold* |
| Rocchio | *Distance metric, Shrinking threshold* |
| Ridge | *Solver* |
| Perceptron | *Regularization term (Penalty), Learning rate (alpha)* |
| LSVC | *Multi-class scheme, Penalties, Error term penalty (C), Loss* |
| PA | *Loss function, Regularization term (Penalty), epochs* |
| DT | *Splitting criteria* |
| SGD | *Loss function, Regularization term (Penalty), Learning rate (alpha), epochs* |
| LR | *Multi-class scheme, Penalties, Inverse of regularization (C), Solver* |

Every supervised machine learning algorithm aims to achieve low bias and low variance but all of them face these errors. Aside from irreducible error, bias and variance errors can be controlled by tuning the parameters of each machine learning algorithm. Algorithms with high variance are

generally low in bias and vice versa. The tuning of the parameters for each classifier is done in the *hyper-parameter tuning* step. During our experiments with each classification algorithm the hyper-parameter tuning is done on the training set and in this phase, the classifiers are run on each possible value for their parameters. The parameter values which give the least error in prediction is selected as the value for that parameter. Table 5.6 shows the list of text classification methods along with their selected parameters for tunning.

***Cross-validation***

*k*-fold cross-validation is one of the well-known sampling techniques (Breiman et al., 1984) (Kohavi, 1995). Compared to normal estimation from a classifier, a cross-validated estimate is more reliable estimate by the models when generalizing on new data. Applying k-fold cross-validation on the training dataset divides the training set into *k* disjoint subsets. A single training is changed into k training cycles on the training set. During each cycle one subset is left out as a test set and the remaining k-1 subsets are used for training the model. Which means that each classifier is trained and tested *k* times on the training data so that the error is within acceptable levels before they are applied on unseen (testing) data. A 10-fold cross-validation is performed on each of the selected method during the training phase. There is no formal rule to select the value for k. However, a smaller value for the k can introduce bias in the trained model's skills. Higher values of k are preferred to reduce the bias. The leave-one-out cross-validation (LOOCV) technique which considers large number of subsets has shown similar results to 10-fold cross-validation. The 10-fold cross-validation may be preferred to LOOCV for large datasets with computationally intensive analysis (Molinaro et al., 2005).

### 5.3.4 Model evaluation

The performance of the trained classifiers is evaluated and compared using a number of error measures and also a statistical significance test. Once the classification models are trained and fitted with their optimal parameters on the training dataset, their performance is examined by introducing them with a test dataset which is new to them. During this study, I have used 75% of the core dataset for the training of the methods and the remaining 25% for the testing during the model evaluation step. Using the error measures, the class prediction errors by the trained classification models are compared with the actual class labels. The results from the error measures on the selected classifiers are compared which will help to rank then based on the level of their performance.

***Error measures***

In this subsection, we discuss the details of the evaluation metrics used to assess the performance of the classifiers during the process of being trained and tested. In the context of text classification, accuracy and error rates are not necessarily good performance measures (Yang & Liu, 1999) (Joachims, 1998) (Wang & Chiang, 2007) (Yang, 1999). To understand the performance of the classifiers in detail, we use macro-averaging and micro-averaging for some of the metrics in our multi-class problem. Macro-averaging treats all classes equally while micro-averaging favours bigger classes (Sokolova & Lapalme, 2009). Macro-average computes the given metric independently for each class and then takes the average. For example, when calculating the precision of the classifier, the precision for each class is calculated first then the average is calculated, whereas micro-average aggregates the contributions of all classes globally to compute the average metric (Özgür et al., 2005) (Yang & Liu, 1999). The error measures used to evaluate the performance of the classification models are discussed in the following subsections.

*Accuracy*

Accuracy is the most common metric used to evaluate the performance of classifiers. The accuracy of a prediction by a classifier is defined as the proportion of the correctly classified instances (Raschka & Mirjalili, 2019) (Muller & Guido, 2016) or in simple terms, it is the effectiveness of a classifier (Sokolova & Lapalme, 2009). In multiclass problems, it is important to look at the accuracy of each class in addition to the average accuracy for all the classes. For example, the accuracy for an individual class $C_i$ is calculated in the one-vs-the-rest way, where the sum of the true positive (correctly predicted instances belonging to class $C_i$) and true negative predictions (correctly predicted instances not belonging to class $C_i$) divided by the total predictions shows the effectiveness of the classifier.

$$Accuracy\ (C_i)\ = \frac{TP_i + TN_i}{TP_i + TN_i + \ FP_i + FN_i} \tag{5.5}$$

The average accuracy of the classifier over multiple classes is calculated as:

$$Average\ Accuracy\ = \frac{\sum_{i=1}^{n} \frac{TP_i + TN_i}{TP_i + TN_i + \ FP_i + FN_i}}{n} \tag{5.6}$$

where $n$ represents the total number of classes

*Precision*

Precision (also known as positive predictive value) is another commonly used performance evaluation metric in classification problems (Richert & Coelho, 2013) (Raschka & Mirjalili, 2019). It is the ability of the classifier not to label a negative sample as positive or in other words, when the goal is to limit the number of false positives (Muller & Guido, 2016). In multi-class problems, the precision of a class $C_i$ is calculated as:

$$Precision\ (C_i)\ =\ \frac{TP_i}{TP_i + FP_i} \tag{5.7}$$

For multiple classes, the macro-averaged and micro-averaged precision (Sokolova & Lapalme, 2009) is calculated as:

$$Precision_{\ \mu}\ =\ \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i\ + FP_i)} \tag{5.8}$$

$$Precision_{\ M}\ =\ \frac{\sum_{i=1}^{n} \frac{TP_i}{TP_i + FP_i}}{n} \tag{5.9}$$

The precision scores are calculated in the interval [0, 1]. The precision score of 1 is considered as the best and a 0 as the worst.

*Recall (Sensitivity or True Positive rate)*

Also known as sensitivity or the true positive rate is the ability of the classifier to find all the positive samples (Richert & Coelho, 2013) (Raschka & Mirjalili, 2019) (Muller & Guido, 2016). In multi-class problems, the recall for a class $C_i$ is calculated as:

$$Recall\ (C_i)\ =\ \frac{TP_i}{TP_i + FN_i} \tag{5.10}$$

For multiple classes, the macro-averaged and micro-averaged recall (Shalev-Shwartz & Singer, 2005) is calculated as:

$$Recall_{\ \mu}\ =\ \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i\ + FN_i)} \tag{5.11}$$

$$Recall_{M} = \frac{\sum_{i=1}^{n} \frac{TP_i}{TP_i + FN_i}}{n} \quad (5.12)$$

The recall scores are calculated in the interval [0, 1] and a value of 1 is considered as the best and a 0 as the worst.

*F-Measure*

Also known as F-Score or F1-measure is the harmonic mean of the precision and recall (Raschka & Mirjalili, 2019) (Muller & Guido, 2016) (Sokolova & Lapalme, 2009) (Özgür et al., 2005). The F-measure values are in the interval [0, 1]. A larger F-measure value corresponds to higher classification quality (Özgür et al., 2005). The macro-averaged F1 is calculate as:

$$F1\,(C_i) = 2 \times \frac{Precission_i \times Recall_i}{Precission_i + Recall_i} \quad (5.13)$$

*Specificity*

Specificity, also known as the true negative rate, is a measure of how a classifier identifies negative labels (Sokolova & Lapalme, 2009). In a multi-class problem, the specificity of a class $C_i$ is calculated in a one-vs-the-rest way and is written as:

$$Specificity\,(C_i) = \frac{TN_i}{TN_i + FP_i} \quad (5.14)$$

The specificity values are in the interval [0, 1], where it reaches its best value at 1 and worst at 0.

*Cohen's Kappa*

Cohen's kappa is a statistic that measures inter-annotator agreement. Cohen's kappa was initially introduced (Cohen, 1960) to measure the degree of agreement or disagreement of two or more people observing the same phenomenon. Later (Ben-David, 2008), it was used successfully to demonstrate the accuracy of classifiers. Cohen's kappa measures the degree of agreement between the classification model's predictions and the actual values (Ben-David, 2008). Cohen's kappa score expressing the level of agreement between two annotators on a classification problem is written as:

$$k = \frac{(p_o - p_e)}{(1 - p_e)} \quad (5.15)$$

where $p_o$ is the empirical probability (total agreement probability) of agreement on the label assigned to any sample (review) called the observed agreement ratio, and

$p_e$ is the expected agreement (agreement probability which is due to chance) when labels are assigned randomly.

The kappa statistic is a number between -1 and 1 where lower values to -1 indicate total disagreement, 0 being a random classification and a maximum value to 1 shows total agreement (Cohen, 1960) (Ben-David, 2008).

*Mathews' Correlation Coefficient (MCC)*

Introduced by Matthews (1975), the MCC was initially used to assess the performance of protein secondary structure prediction. The coefficient ranges between the values of -1 to 1. A coefficient of +1 represents a perfect prediction, 0 represents a random prediction and −1 indicates total disagreement between prediction and observation (Boughorbel et al., 2017). In multi-class problems, it can easily be calculated from a confusion matrix in terms of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) (Baldi et al., 2000) (Gorodkin, 2004) (Jurman et al., 2012) (Scikit-learn, 2017b).

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (5.16)$$

*Hamming Loss*

Hamming loss is a useful measure to evaluate the number of times the label of a review is misclassified. In simple terms, it is the fraction of incorrectly classified single labels over the total number of labels (Schapire & Singer, 2000) (Tsoumakas & Katakis, 2009) (Schapire & Singer, 1999). For a multiclass problem, it is defined as:

$$Hamming\ Loss = \frac{1}{k}\sum_{i=1}^{k}\frac{Y_i \,\Delta\, x_i}{L} \quad (5.17)$$

Where $k$ is the number of samples, $L$ is the finite set of class labels, $Y_i$ is the actual, $x_i$ is the predicted value, and $\triangle$ stands for the symmetric difference between two sets of labels (i.e., $Y_i$ and $x_i$).

In multiclass classification problems, hamming loss corresponds to the Hamming distance between the actual labels ($Y_i$) and predicted labels ($x_i$) which is equivalent to the subset zero-one loss function (Scikit-learn, 2017a).

***Statistical significance test***

For the purpose of model evaluation, besides the error measures discussed in the previous subsection, I also conduct a statistical significance to verify the significance difference among the performances of the classifiers. I used the well-known Friedman significance test (Friedman, 1940) which is a non-parametric statistical significance test widely used in comparing multiple algorithms using their performance samples to identify any significant difference among them. The Friedman statistics is computed as:

$$X_F^2 = \frac{12n}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{5.18}$$

where, a larger value of $X_F^2$ indicates that there is a significant difference among the classifiers and the null hypothesis can be rejected. Further, the Nemenyi's post hoc test (Nemenyi, 1963) is applied to report the significant differences between individual classifiers. According to Nemenyi's post hoc test, the performance of two classifiers is significantly different if their average rank differs by at least the critical difference calculate as:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{5.19}$$

The statistical significance tests are also very helpful in the selection of the best classification models. Once the models are selected based on the preliminary study, the parameters of each method are estimated.

### 5.3.5 Creating Ensemble

Creating ensemble of the trained models is one method to improve classification performance. There are a number of techniques to created model ensembles such as, Bagging, Boosting and Voting. Voting ensembles are the simplest and effective method of combining individual classifiers output to improve classification performance. Besides that, voting ensemble can work with different types of base classifiers compared to the others. There are several voting schemes that include weighted and unweighted voting methods. In this study, I have applied a simple weighted majority voting ensemble

on our base classifiers. For the voting ensemble, the core idea is to only include the classifiers with high performances and lowest miss-classification rates.

There are three steps involved in the implementation of the weighted majority voting ensemble:

1. The first step of the weighted voting ensemble calculates a threshold level for selecting the base classifiers. This threshold level is set on the classification performance of the base classifiers. It helps to avoid adding the classifiers to the ensemble, that have lower classification performance which will negatively affect the overall performance of the ensemble.

$$T = \alpha_B - (\frac{1 - e_B}{N}) \tag{5.20}$$

where, $\alpha_B$ represents the classification accuracy of the best classifier, $e_B$ represents the misclassification error rate of the best classifier and $N$ represents the total number of selected classifiers. Thus, the classifiers for the ensemble are shortlisted as:

$$E = \{c \mid c \in C, c_a \geq T\} \tag{5.21}$$

where $C$ represents the set of trained classifier and $c_a$ represents the classification accuracy of a given classifier $c$.

2. In the second step, weights are assigned to the selected classifiers. This step is also necessary in combination with the first step, as it assigns higher weights to the base classifiers with the lowest mis-classification rates. Assigning weights are very helpful specially when there is a tie when selecting among more than one class label. Let $C_A$ be the set of classification accuracies for the selected classifiers over the threshold for the ensemble, $C_A = \{c_{a1}, c_{a2}, \ldots, c_{an}\}$

$$w_c = \frac{1}{2} + \left(\frac{c_a - \min(C_A)}{\max(C_A) - \min(C_A)}\right) \tag{5.22}$$

where $w_c$ represents the weight of a given classifier '$c$' and a value of ½ is added to set a base weight.

3. Finally, for each test sample '*x'*, the class '*y'* is assigned according to the weighted votes from the selected base classifiers.

Algorithm 5.1 shows the detailed steps of the proposed approach. The complexity of algorithm 5.1 considering the three loop structures can be written as O (R + C * CV * T).

Algorithm 5.1: Different stages of the Ensemble approach for SaaS service factor identification

```
    // Pre-processing dataset
1   D: Dataset
2   Rc: Clean reviews
3   R ← ExtractReviews(D)
4   L ← ExtractLabels(D)
5   for each r ϵ R: do
6           Remove URLs from r
7           Remove non-alphabets from r
8           Remove multiple periods from r
9           Remove multiple spaces from r
10          Lemmatize r
11          Append Rc ← r
12   end for
    // Vectorization and Bag of Words
13  Tokenize Rc
14  Build Vocabulary from Tokenized Rc
15  Build Feature Vector from Vocabulary
16  Reduce Dimensions of feature vector (mindf: 5)
17  Assign Feature weights using tf-idf
18  Normalize tf-idf scores
19  Split Reviews Rc as Xtrain, Xtest
20  Split Labels L as Ytrain, Ytest
    // Resampling:
21  Apply SMOTE on Xtrain
22  Split and Stratify Xtrain into 10-folds as CV
    // Training the classifiers and hyper-parameter tunning
23  C: Classifiers selected for training
24  P: Relevant parameter set for each classification algorithm
25  T: Total permutations of parameters
26  for each c ϵ C: do
27          for each f ϵ CV: do
28                  for each t ϵ T: do
29                          Train classifier c on f with t
30                          Select the parameters setting with best performance score
31                          Test classifier c on f's test subset with t
32                          Save Classifier name, performance score, f, t
33                  end for
34          end for
35  Save classification model c
36  end for
37  Identify any significant difference among classifiers performance using Friedman test and
    Nemenyi's post hoc test
38  CM: Saved classification models
39  M: Set of performance metrics
40  for each m ϵ CM: do
41          Test m's performance on Xtest using M
42          Save m's performance scores
43   end for
44  Compare classification models over performance metrics M
45  Select the best performing classifiers
```

## 5.4 EXPERIMENTS AND EVALUATION RESULTS

In this section, I present the details of the dataset used during the experiments, the optimal parameter from the training phase and evaluate the performance of each classifier. The performance of the proposed ensemble is discussed at the end of this section.

### 5.4.1 Dataset description

I use Python (version 3.6.2) for all the experiments in this study. As a result of the data preparation and pre-processing stage, in the pre-processed dataset, the total samples in the vector space $R^d$ have the final form of 29,362 review samples and 27,657 features ($d = 29362 \times 27657$). For all the experiments, the dataset is divided into training and test sets with the split of 75% and 25% respectively. The training set consists of 22020 samples and the test set consists of 7341 samples. As shown in Table 5.7, the number of samples for each service factor is different. This indicates a class imbalance which is common in real world datasets. The main implication of this problem is that the classifiers have fewer samples to learn from for some of the classes compared to the others. However, in the experiments using SMOTE resampling, the number of training samples has increased to 52240 due to the addition of synthetic instances. The results from both the actual and SMOTE resampled data is compared for performance during the experiments. Each of the samples in the training and test set is labelled with exactly one of the ten class labels listed in Table 5.3.

Table 5.7. Dataset description with the number of reviews per service factor

| Service factor | Complete Dataset | Training | Test |
|---|---|---|---|
| Availability | 3020 | 2272 | 748 |
| Cost optimization | 2288 | 1693 | 595 |
| Resiliency | 64 | 53 | 11 |
| Scalability | 613 | 455 | 158 |
| Management | 6984 | 5224 | 1760 |
| Unknown | 2189 | 1602 | 587 |
| Operational Excellence | 3882 | 2943 | 939 |
| Performance | 4858 | 3663 | 1195 |
| Reliability | 4971 | 3747 | 1224 |
| Security | 492 | 368 | 124 |
| **Total reviews** | **29361** | **22020** | **7341** |

The selected machine learning algorithms in this study are implemented using Python (version 3.6.2) and scikit-learn's (version 0.19.0) libraries. Also, the dataset is vectorized with unigrams, bigrams and 3-grams throughout the experiments.

Table 5.7 shows the number of review samples for each service factor in the considered data corpus. Since we are also using oversampling. As the dataset faces the challenge of class imbalance, stratified k-fold cross validation along with SMOTE is used for resampling of the dataset. Figure 5.2 shows the number of samples per class after applying SMOTE to the training dataset. In stratified cross-validation, the subsets from the training data are created having roughly the same proportion of each service factor (class) samples. The class returns a stratified randomized fold from the training data. We have applied 10-fold cross-validation for all the machine learning algorithms with the test split of 10 percent from each fold to accurately determine the value of the best performing parameters. The same set of experiments are conducted again using SMOTE.

### 5.4.2 Model evaluation

In this subsection, I compare and discuss the performance of all the fitted classification models selected in this study against the proposed ensemble text classification model using the error measures discussed in Section 5.4. Table 5.8 shows the optimal values for the parameters estimated for each of the classifier with lower classification errors. The details of the parameter tunning and CV for all the selected text classification approaches is presented in Appendix A.

Table 5.8. Optimal parameters selected after training

| **Classification methods** | **Optimal values for selected Parameter** |
|---|---|
| kNN | *n_neighbors=20, metric='euclidean', leaf_size=30, algorithm='auto', weights='uniform'* |
| NB (Multinomial) | *alpha = 0.1, class_prior=None, fit_prior=True* |
| NB (Bernoulli) | *alpha = 0.5, Binarize threshold = 0, class_prior=None, fit_prior=True* |
| Rocchio | *metric='euclidean', shrink_threshold=None* |
| Ridge | *alpha= 1, solver= svd* |
| Perceptron | *alpha=1e-05, n_iter=6, penalty='l1'* |
| LSVC | *C= 0.5, loss= hinge, multi_class= crammer_singer, penalty= l2* |
| PA | *C=0.1, loss='hinge', n_iter=10* |
| DT | *criterion= gini, max_depth= 96/ none, min_samples_split= 2* |
| SGD | *alpha= 5e-05, loss= hinge, n_iter= 9, penalty= 'l1'* |
| LR | *C= 5, multi_class= ovr, penalty= l1, solver= liblinear* |

In this section, the performance of the classification methods is evaluated using both macro- and micro-averaged scores. In addition, we use two widely used measures to assess the degree of agreement between the actual sample classes and the predicted class labels. The main reason for looking at the performance of the classification models through different evaluation metrics is that our dataset is imbalanced in terms of number of samples per class and also there are class overlaps

Table 5.9. Comparative analysis of the multiclass text classification approaches using Stratified k-folds and SMOTE

| Ts. time | Tr. time | MCC | C. kappa | Specificity | Ham. loss | F1 (μ) | F1 (m) | Rec (μ) | Rec (m) | Prec (μ) | Prec (m) | Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **7.814** | **1370.8** | **0.250** | **0.240** | **0.924** | **0.62** | **0.38** | **0.26** | **0.38** | **0.25** | **0.38** | **0.39** | **0.38** | St | k-NN |
| 19.074 | 10810.8 | 0.040 | 0.005 | 0.901 | 0.92 | 0.08 | 0.05 | 0.08 | 0.12 | 0.08 | 0.42 | 0.08 | SMOTE | |
| **0.009** | **4.4** | **0.398** | **0.396** | **0.940** | **0.51** | **0.49** | **0.40** | **0.49** | **0.40** | **0.49** | **0.42** | **0.487** | St | NB - M |
| 0.009 | 11.8 | 0.376 | 0.373 | 0.938 | 0.53 | 0.47 | 0.39 | 0.47 | 0.39 | 0.47 | 0.40 | 0.47 | SMOTE | |
| **0.018** | **6.5** | **0.426** | **0.423** | **0.942** | **0.49** | **0.51** | **0.38** | **0.51** | **0.39** | **0.51** | **0.42** | **0.512** | St | NB -B |
| 0.018 | 17.1 | 0.385 | 0.376 | 0.937 | 0.52 | 0.48 | 0.36 | 0.48 | 0.35 | 0.48 | 0.48 | 0.48 | SMOTE | |
| **0.013** | **6.4** | **0.450** | **0.447** | **0.945** | **0.47** | **0.53** | **0.47** | **0.53** | **0.51** | **0.53** | **0.47** | **0.53** | St | Rocchio |
| 0.013 | 16.5 | 0.440 | 0.437 | 0.944 | 0.48 | 0.52 | 0.45 | 0.52 | 0.48 | 0.52 | 0.46 | 0.52 | SMOTE | |
| **0.009** | **30.1** | **0.641** | **0.636** | **0.963** | **0.30** | **0.70** | **0.62** | **0.70** | **0.57** | **0.70** | **0.76** | **0.70** | St | Ridge |
| 0.009 | 148.3 | 0.613 | 0.612 | 0.961 | 0.33 | 0.67 | 0.65 | 0.67 | 0.64 | 0.67 | 0.66 | 0.67 | SMOTE | |
| **0.009** | **9.2** | **0.707** | **0.706** | **0.971** | **0.25** | **0.75** | **0.66** | **0.75** | **0.66** | **0.75** | **0.67** | **0.75** | St | Percep |
| 0.009 | 31.5 | 0.728 | 0.727 | 0.973 | 0.23 | 0.77 | 0.70 | 0.77 | 0.73 | 0.77 | 0.69 | 0.77 | SMOTE | |
| **0.010** | **55.2** | **0.721** | **0.720** | **0.972** | **0.23** | **0.77** | **0.67** | **0.77** | **0.63** | **0.77** | **0.79** | **0.77** | St | LSVC |
| 0.009 | 3727.4 | 0.717 | 0.717 | 0.972 | 0.24 | 0.76 | 0.70 | 0.76 | 0.69 | 0.76 | 0.71 | 0.76 | SMOTE | |
| **0.009** | **8.7** | **0.727** | **0.725** | **0.972** | **0.23** | **0.77** | **0.69** | **0.77** | **0.65** | **0.77** | **0.77** | **0.77** | St | PA |
| 0.010 | 29.3 | 0.722 | 0.722 | 0.972 | 0.24 | 0.76 | 0.70 | 0.76 | 0.67 | 0.76 | 0.74 | 0.76 | SMOTE | |
| **0.015** | **126.9** | **0.729** | **0.729** | **0.973** | **0.23** | **0.77** | **0.67** | **0.77** | **0.65** | **0.77** | **0.69** | **0.77** | St | DT |
| 0.016 | 455.8 | 0.739 | 0.739 | 0.974 | 0.22 | 0.78 | 0.74 | 0.78 | 0.75 | 0.78 | 0.73 | 0.78 | SMOTE | |
| **0.009** | **12.0** | **0.805** | **0.804** | **0.980** | **0.16** | **0.84** | **0.70** | **0.84** | **0.68** | **0.84** | **0.75** | **0.835** | St | SGD |
| 0.010 | 41.7 | 0.794 | 0.792 | 0.980 | 0.18 | 0.82 | 0.78 | 0.82 | 0.84 | 0.82 | 0.76 | 0.82 | SMOTE | |
| **0.011** | **37.0** | **0.817** | **0.817** | **0.981** | **0.15** | **0.845** | **0.77** | **0.845** | **0.74** | **0.845** | **0.83** | **0.845** | St | LR |
| 0.010 | 162.8 | 0.814 | 0.814 | 0.981 | 0.16 | 0.84 | 0.80 | 0.84 | 0.79 | 0.84 | 0.82 | 0.84 | SMOTE | |
| - | - | **0.829** | **0.828** | **0.983** | **0.14** | **0.856** | 0.750 | **0.856** | 0.725 | **0.856** | 0.800 | **0.856** | SMOTE | Ensembl |

which means that the labelling of samples is subjective in most of the cases. The selected error metrics used in this study are famous in evaluating the classification performance of the models trained and tested on imbalanced datasets. Following is the detailed discussion on the performance results for each of the models on test data.

*Naïve Bayes (Multinomial)*

With the optimal parameter i.e., alpha = 0.1, the model's prediction on new data (test set) is evaluated with different metrics. The results show no overfitting in terms of classification accuracy of 0.49 and loss of 0.51. As we are dealing with a multiclass problem and facing a class imbalance, scikit-learn gives the flexibility to look further into the classification performance of the model using the micro-averaged metrics. The micro-averaged precision shows the performance by weighting each sample equally hence showing the capability of naïve Bayes to classify each sample with precision and recall of 0.49. The specificity ($TN/TN + FP$) or the true negative rate is 0.94. As scikit-learn implements multiclass scoring via the one-vs-all classification approach, this means that most of the instances will be predicted as members of the remaining classes (true negatives), hence a higher specificity value. Another interesting perspective to investigate the performance of the classifier in a multiclass problem is the capability to find all the positive samples known as the true positive rate or recall. Here, the naïve Bayes multinomial shows a micro-recall score of 0.49 and an overall macro-averaged recall of 0.40. The degree of agreement is measured using two famous measures used frequently in multiclass classification problems i.e., the Cohen kappa measure with 0.396 and the MCC with 0.398, which is at an acceptable level when it is not in the negative value range.

In addition to the low performance shown here by the naïve Bayes multinomial classifier, it shows minimal difference when comparing performance scores for the training and test data.

*Naïve Bayes Bernoulli*

With the best estimated parameters i.e., alpha = 0.5 and binarizing threshold = 0, the performance of the model on the different metrics on the test dataset the model achieves an accuracy of 51% (0.51), precision of 42%, recall of 39% and F1 score of 38%. Since we are dealing with a multiclass classification problem, looking at the micro-average scores is very helpful. The specificity or the true negative rate of the classification is at 0.94; while the degree of the agreement between the actual and predicted classes measured, using Cohen Kappa is 0.422 and with MCC is at an acceptable sore of 0.426.

The NB Bernoulli classifier shows a minimal difference between the performance scores on the training and test dataset using different evaluation metrics. Similar to the multinomial version, the naïve Bayes Bernoulli is also very quick to train but the performance is not very encouraging.

*Nearest Centroid (Rocchio)*

The results show that the difference in the performance on the training and testing dataset is minimal when the algorithm is used with the optimal distance measure i.e., Euclidean. The macro-accuracy, precision, recall and F1 measure remains at 53% (0.53), 0.47, 0.51 and 0.47 respectively on test data. The true negative rate (specificity) is 0.945 and the degree of agreement measured with Cohen's kappa is 0.446 and 0.449 with MCC.

*Stochastic Gradient Descent (SGD)*

The classification accuracy of 84% (0.84) on test data is very impressive. The macro-precision (75%), recall (68%) and F1 (70%) scores are also very significant. The most interesting outcome is the micro-averaged scores (as it is very important in multiclass problems). The micro-averaged scores are very impressive while facing the issue of class imbalance. The true negative rate (specificity) is high as usual, as we are using the one-vs-all approach for the multiclass problem. Compared with the results from the other classifiers, the degree of agreement measured with Cohen's kappa (0.803) and MCC (0.805) is significantly higher.

*K-NN*

Using the best estimated parameters i.e., n_neighbors=20, metric='euclidean', leaf_size=30, algorithm='auto', weights='uniform', identified during the cross-validation of the training data, the macro-averaged accuracy, precision, recall and F1 scores are recorded as 0.38 (38%), 0.39, 0.25 and 0.26 respectively. The micro-averages scores are low as well. The true negative rate (specificity) is 0.923 and the degree of agreement measured by Cohen's kappa (0.24) and MCC (0.25) is also lower compared to the other classifiers used. The main reason for the low performance of the k-NN classification algorithm is that it is not suitable for large datasets with high dimensionality (a large number of features) (Soucy & Mineau, 2001) (Raschka & Mirjalili, 2019). The k-NN algorithm is well known to perform better with small datasets with fewer features.

*Passive Aggressive*

The passive aggressive algorithm with the best parameters achieved an accuracy score of 77% (0.77), the passive aggressive algorithm achieved a macro-averaged precision, recall and F1 score of 77%

(0.77), 65% (0.65) and 69% (0.69) respectively. As we are dealing with an issue of class imbalance in a multiclass problem, the micro-averaged scores give very interesting results. The micro-averaged precision, recall and F1 scores from the test dataset are 0.77 for each of these metrics. The true negative rate (specificity) is 0.972, the degree of agreement between the actual and predicted classes measured using Cohen's kappa and MCC are 0.725 and 0.726, respectively. A comparison between the performance of the passive aggressive classifier on the training and test datasets shows that there is no significant difference between the performance of the passive aggressive classifier.

In addition to the number of parameters to tune, the passive aggressive algorithm for classification tasks is very cheap to train and test on larger datasets. The test performance of the passive aggressive classifier makes it a strong candidate for our multiclass problem.

*Perceptron*

The accuracy of the classification on text dataset is at an acceptable level of 75% (0.75). The macro-precision, recall and F1 scores are 0.75, 0.66 and 0.66 respectively. The micro-averaged precision, recall and F1 scores are 0.75 each. The micro-averaged classification performance scores of the perceptron algorithm show very promising results as it focuses on individual samples. The specificity or the true negative rate is a high score of 0.970. The degree of agreement measured by Cohen's kappa and MCC are 0.706 and 0.707, respectively. It is very important to look at the performance using several metrics as it provides a detailed breakdown of the nature of the model for specific classification problems.

A minimal difference between the performance of the perceptron is observed on the training and test data. In addition to the number of parameters to tune, the perceptron is very efficient to train and the entire 10-fold stratified cross-validation on our relatively large dataset is completed in just under 3780 seconds.

*Logistic Regression*

Using the best parameters for logistic regression, a classification accuracy of 85% (0.845) is achieved on test dataset. The macro-averaged precision, recall and F1 scores are 0.845, 0.74 and 0.77, respectively. It is very interesting to see the micro-averaged scores for logistic regression on the test data. The micro-averaged precision, recall and F1 scores are recorded as 0.845 for each of these measures, which is very promising as we are dealing with class imbalance. The true negative rate (specificity) is 0.981 which is another promising outcome of the logistic regression for multiclass problems. The degree of agreement measured using Cohen's kappa is 0.817 and MCC is 0.817 which

again shows a high level of agreement between the actual and the predicted classes. A very small difference between the classification performance on the training and test dataset is observed during the experiments. The cost of training logistic regression for a large dataset such as the one we used in this study is not very high, even with a 10-fold cross-validation.

*Decision Tree*

The overall classification accuracy of the decision tree is 77% (0.77) on test dataset. The macro-averaged precision, recall and F1 scores are 0.69, 0.65 and 0.67 respectively. However, the micro-averaged scores give us a good insight in the model's performance with a high precision of 77% (0.77). The degree of agreement between the actual and predicted classes by the decision tree is also very promising with a Cohen's kappa score of 0.728 and an MCC score of 0.728. It is observed that the difference between the performance of the decision tree classifier on the training and test data is also very low.

*Ridge*

The Ridge classifier achieved a classification accuracy of 70% (0.70), while the macro-averaged precision, recall and F1 scores are 0.76, 0.57 and 0.62 respectively using the best estimated parameters on test data. The micro-averaged scores for precision, recall and F1 are all 0.70. The specificity (true negative rate) is 0.962 while the degree of agreement measured using the Cohen's kappa and MCC are 0.635 and 0.641 respectively. The results are encouraging, and the Ridge classifier is very simple and efficient to use on relatively large datasets. Furthermore, our results show that the difference in the classification performance on the training and test datasets using the optimal parameters is also minimal.

*Linear SVM (LSVC)*

On test dataset the LSVC achieved an accuracy of 77% (0.77), the macro-averaged precision, recall and F1 scores are at the promising levels of 0.79, 0.63 and 0.67 respectively. The micro-averaged precision, recall and F1 scores are also very interesting as we are dealing with class imbalance. The degree of agreement measures with Cohen's kappa and MCC are 0.720 and 0.721, indicating a good classification performance by linear SVC. In addition to the number of parameters to adjust to achieve optimal performance, linear SVM is highly efficient with large datasets and the entire training is completed within 1361 seconds.

The results from Table 5.9 show that the logistic regression classification model has the highest accuracy of 85% (0.845) among all the individual classification models. The accuracy score for the SGD classification model is the second highest at 84% (0.84). As discussed in the previous subsection, the SGD algorithm is used with the hinge loss function among others, which is the same as the soft-margin SVM. The results show that the lowest accuracy score is achieved by the k-NN classification method at 38% (0.38) followed by the two versions of the naïve Bayes classifiers i.e., naïve Bayes multinomial and naïve Bayes Bernoulli. Although the macro-averaged scores for the logistic regression models is very promising, the macro-averaged scores are relatively less expressive of the classification details as they are dominated by the rare classes (classes with fewer samples), giving equal weight to all classes. As we are dealing with the issue of class imbalance as shown in Table 5.7, the micro-averaged scores provide a good understanding of the model's performance. Interestingly, the micro-averaged scores are 85% (0.845) which is very significant on text data.

Both the precision and recall scores for logistic regression are very good in terms of multiclass text classification. Furthermore, looking the performance through the harmonic means (F1) of the precision and recall shows high scores above 75%. The true negative rate, also known as the specificity of the logistic regression model, is the highest of all the classifiers i.e. 0.981. When working with multiclass problems, the true negative rate of the models is generally high and it increases with an increase in the model's classification accuracy. Another approach used to assess the performance of the classification model is to measure the degree of agreement between the prediction and observation. We used two of the most well-known approaches to measure the degree of agreement, namely Cohen's kappa score and Matthews' correlation coefficient. The scores of both measures are good for the logistic regression model with a Cohen's kappa score of 0.817 and MCC of 0.818.

Considering the global performance of the classification models from Table 5.9, it can be concluded that the classes are linearly separable. One of the main reasons for the low performance of some of the classifiers, such as k-NN and naïve Bayes, is that they do not work well with a large number of features. It has been reported previously that the performance of k-NN is very low when the dataset is facing the issue of class imbalance (Trstenjak et al., 2014) (Schapire & Singer, 2000). The same is the case with naïve Bayes, in which the classifiers perform below expectations with imbalanced classes (Joachims, 1997) (Brownlee, 2016b). Similarly, studies also indicate that the performance of the Rocchio algorithm is very low when it comes to multi-class text classification with noisy data and class imbalance (Brownlee, 2016b).

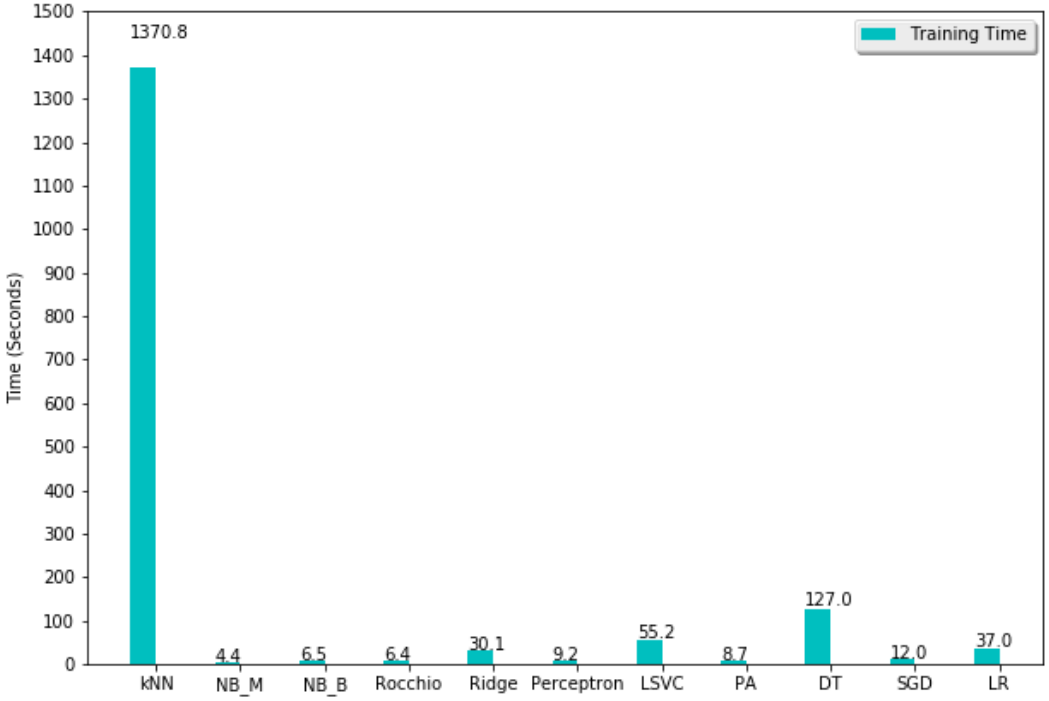



Figure 5.3. Comparison of the training time for the classification models with their best parameters

Table 5.9 shows that using SMOTE oversampling technique on the training data did not improve the performance of the classifiers when they are introduced to new data (test data). Even in some cases the classifiers trained just on 10-fold stratified cross-validation performs slightly better than an additional oversampling step. However, when the trained Logistic regression classifier is applied on the new data (test data), the performance drops and the test performance is same as the Logistic regression classifier trained without oversampling of training data.

Table 5.9 depicts the training times of all the classification methods used in this study. In addition to the efficient training time of the logistic regression approach, the other methods such as passive aggressive, perceptron, ridge and naïve Bayes classification approaches are also very efficient. The training time of the k-NN classification algorithm is the highest of all the methods used in this study. On the same training dataset, the k-NN algorithm took 1370.8 seconds with its best parameters identified in the first phase of parameter tuning all the methods. The training time of the decision tree (using CART) is the second highest of all the classification methods used in this study, with a training time of 127 seconds. Figure 5.3, shows the total training time consumed by all the selected algorithm.

### 5.4.3 Statistical significance test

Table 5.10 shows the significant differences among classifier pairs with bold and italic p-values. The Friedman's test is conducted initially on the combined results of the classifiers for which, the null hypothesis is rejected at p-value of less than 0.001 with a test statistic of 98.055. Using Nemenyi's post hoc test, the results at a significance level of 5% ($\alpha = 0.05$) is shown in Table 5.10.

Table 5.10. Results from Nemenyi's post hoc test for classifier pairs

| | DT | KNN | LR | LSVC | NB_B | NB_M | PA | Percp. | Ridge | Rocch | SGD |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **DT** | -1.0000 | **0.0010** | 0.9000 | 0.9000 | **0.0074** | **0.0010** | 0.9000 | 0.8240 | 0.3499 | 0.0712 | 0.9000 |
| **KNN** | **0.0010** | -1.0000 | **0.0010** | **0.0010** | 0.9000 | 0.9000 | **0.0010** | **0.0247** | 0.2010 | 0.6143 | **0.0010** |
| **LR** | 0.9000 | **0.0010** | -1.0000 | 0.5305 | **0.0010** | **0.0010** | 0.7401 | 0.0862 | **0.0074** | **0.0010** | 0.9000 |
| **LSVC** | 0.9000 | **0.0010** | 0.5305 | -1.0000 | 0.1031 | **0.0122** | 0.9000 | 0.9000 | 0.8660 | 0.4428 | 0.6143 |
| **NB_B** | **0.0074** | 0.9000 | **0.0010** | 0.1031 | -1.0000 | 0.9000 | **0.0385** | 0.5724 | 0.9000 | 0.9000 | **0.0010** |
| **NB_M** | **0.0010** | 0.9000 | **0.0010** | **0.0122** | 0.9000 | -1.0000 | **0.0034** | 0.1723 | 0.6143 | 0.9000 | **0.0010** |
| **PA** | 0.9000 | **0.0010** | 0.7401 | 0.9000 | **0.0385** | **0.0034** | -1.0000 | 0.9000 | 0.6563 | 0.2336 | 0.8240 |
| **Percp.** | 0.8240 | **0.0247** | 0.0862 | 0.9000 | 0.5724 | 0.1723 | 0.9000 | -1.0000 | 0.9000 | 0.9000 | 0.1237 |
| **Ridge** | 0.3499 | 0.2010 | **0.0074** | 0.8660 | 0.9000 | 0.6143 | 0.6563 | 0.9000 | -1.0000 | 0.9000 | **0.0122** |
| **Rocch** | 0.0712 | 0.6143 | **0.0010** | 0.4428 | 0.9000 | 0.9000 | 0.2336 | 0.9000 | 0.9000 | -1.0000 | **0.0010** |
| **SGD** | 0.9000 | **0.0010** | 0.9000 | 0.6143 | **0.0010** | **0.0010** | 0.8240 | 0.1237 | **0.0122** | **0.0010** | -1.0000 |

### 5.4.4 Performance of the Ensemble

The performance results for the ensemble model using different error measures are shown in Table 5.9. The performance of the weighted voting ensemble is better than the other base classifiers with a classification of accuracy of 0.856 compared to the best performance of the Logistic regression among the base classifiers at 0.846. The improvement is also reflected in the number of accurately classified samples i.e., 6281 compared to that of the Logistic regression at 6250.

Table 5.11. Extended classification report of individual classes using ensemble model

| | Accuracy | Precision | Recall | F1 | Specificity | Support |
|---|---|---|---|---|---|---|
| **Availability** | 0.972 | 0.925 | 0.791 | 0.853 | 0.993 | 748 |
| **Cost optimization** | 0.974 | 0.838 | 0.837 | 0.838 | 0.986 | 595 |
| **Resiliency** | 0.998 | 0.333 | 0.091 | 0.143 | 1.000 | 11 |
| **Scalability** | 0.993 | 0.901 | 0.747 | 0.817 | 0.998 | 158 |
| **Management** | 0.925 | 0.809 | 0.899 | 0.852 | 0.933 | 1760 |
| **Unknown** | 0.979 | 0.851 | 0.888 | 0.869 | 0.987 | 587 |
| **Operational Excellence** | 0.959 | 0.852 | 0.825 | 0.838 | 0.979 | 939 |
| **Performance** | 0.966 | 0.888 | 0.905 | 0.896 | 0.978 | 1195 |
| **Reliability** | 0.958 | 0.879 | 0.868 | 0.873 | 0.976 | 1224 |
| **Security** | 0.987 | 0.725 | 0.403 | 0.518 | 0.997 | 124 |

The performance score of the ensemble model for each of the 10 classes is depicted in Table 5.11. The support column of Table 5.11 shows the actual number of samples per class in the test dataset.

Table 5.12. Confusion matrix for the ensemble classification model

| | **A** | **C** | **E** | **L** | **M** | **N** | **O** | **P** | **R** | **S** |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 592 | 15 | 1 | 1 | 26 | 6 | 39 | 36 | 31 | 1 |
| **C** | 3 | 498 | 0 | 1 | 52 | 11 | 8 | 10 | 8 | 4 |
| **E** | 3 | 2 | 1 | 0 | 1 | 0 | 2 | 2 | 0 | 0 |
| **L** | 3 | 2 | 0 | 118 | 7 | 0 | 14 | 11 | 3 | 0 |
| **M** | 19 | 14 | 1 | 4 | 1582 | 33 | 39 | 32 | 29 | 7 |
| **N** | 0 | 2 | 0 | 1 | 59 | 521 | 1 | 0 | 3 | 0 |
| **O** | 4 | 16 | 0 | 0 | 62 | 6 | 775 | 33 | 40 | 3 |
| **P** | 3 | 16 | 0 | 2 | 42 | 9 | 15 | 1082 | 25 | 1 |
| **R** | 10 | 29 | 0 | 4 | 74 | 19 | 12 | 11 | 1062 | 3 |
| **S** | 3 | 0 | 0 | 0 | 50 | 7 | 5 | 2 | 7 | 50 |

The confusion matrix for the ensemble model is shown in Table 5.12. Each row and column in the confusion matrix of Table 5.12 is labelled according to the tags defined in Table 5.2. The confusion matrix is based on the performance of the classification models' computing using the actual (observed) class labels for each test sample and the predicted class labels by the model. A confusion matrix is very useful in analysing the accuracy of predictions from the trained models. It helps to calculate misclassifications in terms of the true positive, false positive, true negative and false negative values. It also helps to look at the number of correctly and incorrectly predicted classes which can be used to improve the labelling of the dataset. From Table 5.12, the samples labelled under *Resiliency, Management and Security* need to be revisited and updated which will eventually help to improve the classification performance.

The performance of the ensemble model on the training and test dataset is depicted in Figure 5.4. The classification performance in terms of accuracy shows an acceptable amount of overfitting by the trained ensemble classification model.

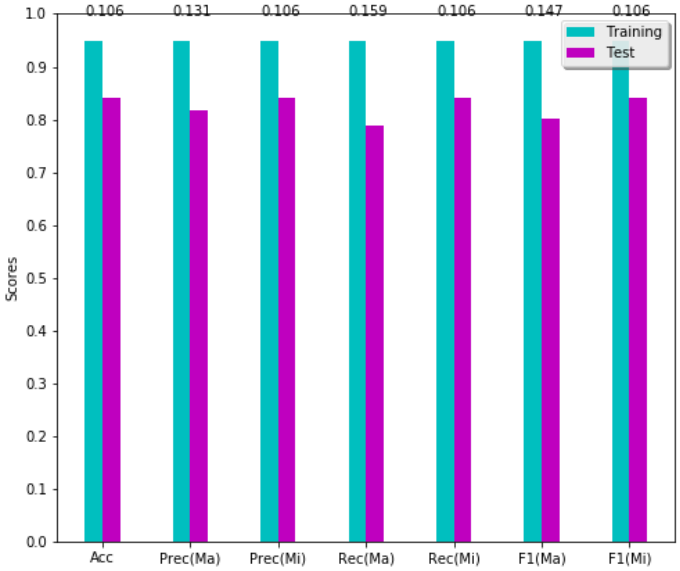



Figure 5.4. Performance of the Ensemble model on training and test data with optimal parameters using SMOTE

## 5.5 CONCLUSION

In this chapter, I discussed the importance of the core factors that is necessary to select SaaS products and are key to the development of trust models for SaaS. I have also explained the importance of the software quality pillars and the pillars of well-architected frameworks which are used by major cloud services vendors as key factors to identify the quality of cloud applications and services. I termed these key factors as service factors in this study and propose an approach to identify them from customer reviews of SaaS products. I discussed the categorization of the SaaS customer reviews into different service factors as a multi-class text classification problem and explained how the machine learning based text classification methods are can be used to identify them. The proposed approach builds an ensemble of the top performing machine learning models that fits best on the SaaS reviews dataset by going through well-defined stages. I prepare the customer reviews using different methods such as the bag-of-words, in order to make them workable with the machine learning algorithms. The analysis of the dataset shows a class-imbalance among the output labels. I have used methods such as stratification and SMOTE to minimize the effect of class-imbalance. Based on the nature of the problem, I investigated and selected 11 of the most appropriate machine learning algorithms that is used for text classification. Each of the selected machine learning algorithms are thoroughly cross-validated and their parameters are tunned systematically in the model's parameter estimation stage.

After the parameters for each method that fits best on the dataset, is identified, the trained and fitted models are introduced to completely new test data. Their performance is evaluated using appropriated error measure and also through Friedman's significance test with Nemenyi's post-hoc tests. The performance of the classifiers is compared, ranked and the best performing classifiers on the SaaS dataset is selected to build an ensemble model. The experimental results on the SaaS dataset shows that the performance of the logistic regression method is the best among the others.

The proposed ensemble for SaaS reviews text classification approach in this chapter is the first component of the framework discussed in Chapter 4 and is considered as the foundation for the remaining components of the proposed framework. In order to build an effective trust model, for each of the identified service factor, the customer's sentiment towards them also need to be computed. Furthermore, the trust in a SaaS product cannot be computed if the customer's sentiment regarding any of the core service factors are missing. There issues are discussed in details in the next chapter.

# Chapter 6:
# T-NN: A Threshold-based Nearest Neighbour approach for Imputing missing values of SaaS service factors

## 6.1 INTRODUCTION

In the previous chapter, I discussed the identification of SaaS service factors from customer reviews and presented an ensemble approach using machine learning methods. As discussed in Chapter 4, the next step in the trust management framework is dealing with the missing SaaS service factor and sentiment intensity scores related to each service factor of the SaaS products. The missing values in the service factors effects and degrades the decision-making capabilities in selecting the SaaS products and computing their trust levels. Imputing precisely the missing sentiment intensity scores of the service factors enables the other components of the framework such as forecasting and modelling the trust levels using these service factors, to perform optimally. In this chapter, I have proposed a novel sentiment intensity imputation approach which is highlighted as a component of the trust management framework in Chapter 4.

In this chapter, the two main area of investigation include, computing the sentiment intensity scores for each identified service factors for the SaaS products and imputing the missing sentiment intensity scores in any of the identified service factors. It is important to accurately quantify the user's assessment in a service factor. For example, while the following two sentences of an assessment *'the operational quality was good'* and *'the operational quality was the best I have ever experienced'* represent a positive sentiment of the user, the sentiment intensity in the later sentence is higher than in the former one. Thus, when the sentiment intensity of the service factor is being determined, the second sentence should get a higher positive score than the first one. Furthermore, inferring the sentiment intensity of service factors which are not mentioned in the users' assessment is another common problem which occur when customers do not provide their opinion in all the service factors.

This chapter is organized as follows. In the next section, I provide a brief overview of the different pattern of missingness in data, their effects on any analysis on that data and a broad category of methods involved in dealing with these patterns. In Section 6.3, I present the detailed development stages of the proposed imputation approach called the Threshold-based Nearest Neighbours (T-NN).

In Section 6.4, I provide a detailed discussion on the experiments and evaluation results of T-NN against the well-known imputation approaches. In Section 6.5, I present an example case study to demonstrate the steps involved in the imputation of a missing value for a given SaaS service factor. Section 6.6 concludes the chapter.

## 6.2 MISSING DATA MECHANISMS

The missingness of data can be categorised under three mechanisms (Rubin, 1976) (Little & Rubin, 1986):

1. Missing Completely at Random (MCAR)
2. Missing at Random (MAR)
3. Missing Not at Random (MNAR)

The missingness of data is assumed as MCAR when the missing value of a given variable does not depend on the values (observed or missing) of other variables. In the MAR assumption, the missingness of the values in one variable may be dependent on the observed values of other variables. MCAR and MAR are also known as ignorable missingness. When the missingness of values is dependent on the unobserved data, then the missingness is assumed to be MNAR. The MNAR mechanism is also called on-ignorable missingness. The data can be missing in one of many patterns that includes, univariate, multivariate, monotone, general (Little & Rubin, 1986). The pattern of missingness in the dataset used in this study follows a general pattern i.e., the missing values are spread randomly across all the variables.

For most of the predictive models missing values in the datasets can be problematic. The methods that deal with datasets with missing values generally take two different approaches.

1. In the first approach the instances with missing values are discarded from the dataset which is also known as *complete case analysis*.
2. In the second approach the missing values are imputed using different approaches.

Discarding the instances with missing values can lead to several challenges such as shrinking the dataset size as the possibility of different variables having missing values on different units, leaving only those complete units that are not representative of the whole population (thus adds significant bias) (Gelman & Hill, 2006). On the other hand, imputing the missing values in instances helps to retain those instances which can help to reduce the bias regarding population representation.

Missing data imputation can be achieved through univariate imputation methods i.e., imputing a single variable by utilizing information from the variable itself or it can be achieved via multivariate imputation methods that impute missing data in more than one variable using a multivariate model.

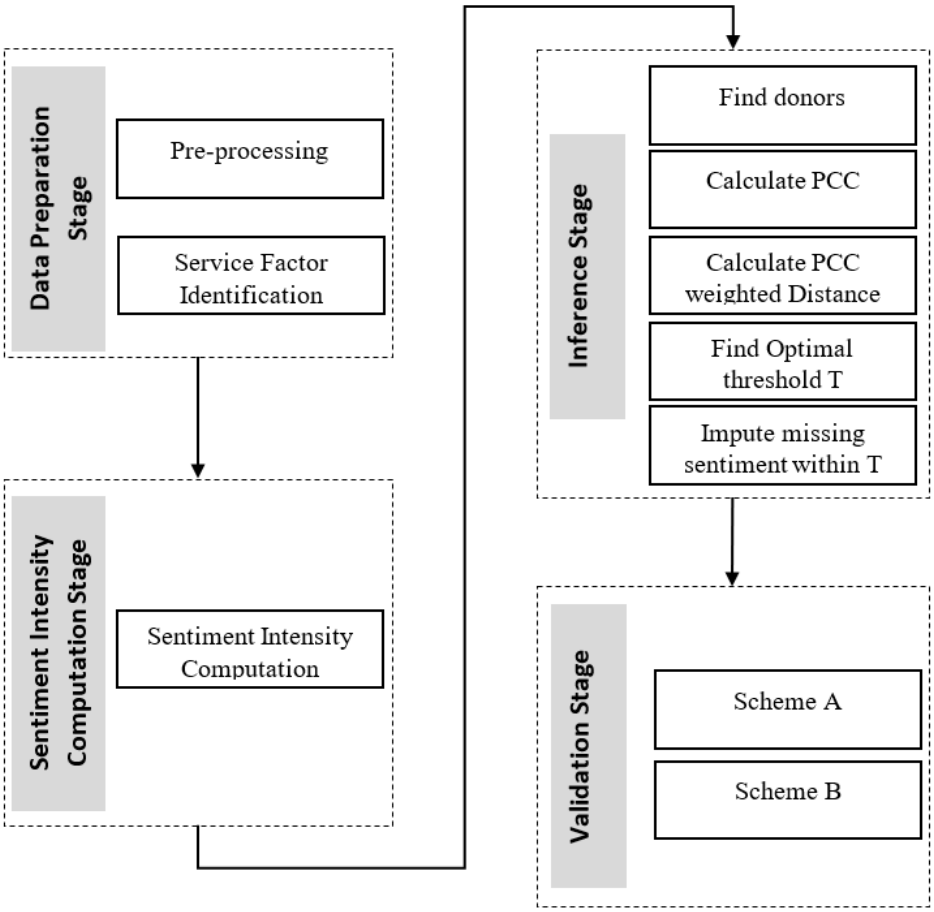



Figure 6.1 Methodology for imputation of the missing service factors of SaaS

## 6.3 THRESHOLD-BASED NEAREST NEIGHBOUR APPROACH

In this section, an overview of the proposed threshold-based Nearest Neighbour (T-NN) imputation approach is presented along with the details for each of the development stages.

### 6.3.1 Overview of Missing sentiment imputation methodology

The flow of the solution methodology for imputing the missing sentiment intensity scores in SaaS service factors is depicted in Figure 6.1. It is composed of four different stages and each stage is organized with well-defined steps. In the Pre-processing stage SaaS customer reviews are collected, segmented and the service factors are identified. In the second stage, the sentiment intensity of each text segment is computed. The body of the proposed T-NN approach is developed in the Inference

stage with a well-defined set of steps. Finally, in stage 4, the proposed T-NN imputation approach is validated in two different schemes with other imputation approaches implemented in this study.

As shown in Figure 6.1, our proposed solution methodology consists of four stages namely:

1. Pre-processing stage
2. Sentiment Intensity computation stage
3. Inference stage
4. Validation stage

*Pre-processing* is the first stage in which we utilize the method developed in Chapter 5 to determine and extract the service quality factors from textual reviews. *Sentiment intensity computation* is the second stage in which the sentiment intensity in the users' reviews linked to a service factor in the first component is determined. We utilize the rule-based model known as the "*Valence Aware Dictionary for sEntiment Reasoning*" (VADER) (Hutto & Gilbert, 2015) to achieve this aim. *Inference stage* is the third stage of the methodology that aims to infer the sentiment intensity values of service factors for which no user reviews are present. We propose a modified and enhanced version of the traditional kNN approach namely T-NN: A Threshold based nearest neighbour with weighted Euclidean distance to infer the values of the missing factors. *Testing stage* is the last stage of the methodology in which the performance of T -NN in inferring the missing sentiment intensity values are compared with other similar techniques.

In the next subsections, we explain in detail the workings of each stage of the methodology.

### 6.3.2 Data preparation stage

The objective of this stage which is to collect the reviews and pre-process them is done in three sub-stages as follows:

#### *Pre-processing*

The pre-processing stage is responsible for collecting customer reviews, processing and segmenting them and determining which SaaS quality factor they relate to. The details of the scrapping and collecting SaaS customer reviews are presented in Chapter 5, where, from three different sources such as, AWS marketplace, GetApp.com and Serchen.com we collected 29,361 reviews from 3215 SaaS products.

Mathematically, the collected reviews $(R_T)$ is the combination of all the reviews for all the products from *n* number of review sources considered ($R_{Sn}$), (which in this case is 3) and is written as:

$$R_T = \bigcup \{R_{S1}, .., R_{Sn}\} \tag{6.1}$$

The reviews for the product $P_i$ can be represented as:

$$R_{Pi} = \{r \mid r \in R_T\} \tag{6.2}$$

All the collected reviews are pre-processed and cleansed for further analysis. Each review is then split into a different number of text segments. As outlined in Chapter 5, this step is very important because a single review may tend to address multiple service factors. The review segments for the product $P_i$ can be represented as:

$$S_{Pi} = \{s \mid s \in R_{Pi}\} \tag{6.3}$$

The total number of review segments in the whole dataset can be represented as:

$$S_T = \{s \mid s \in S_{Pi} \mid i \in P_T\} \tag{6.4}$$

where $P_T$ represents total number of products.

***SaaS service factor identification***

Each review segment is then labelled using the trained ensemble classifier developed in Chapter 5. As a result, each of the text segments have a label associated with it as shown in Table 6.1. The label for each segment either identifies the SaaS quality pillar (service factor) they refer to or an unidentified segment is labelled *"N"* if they do not refer to any of the service pillars.

Table 6.1. SaaS service quality pillars (service factors)

| **Service** factor | **Label/Ta** | **Numeric** |
|---|---|---|
| Availability | A | 0 |
| Cost optimization | C | 1 |
| Resiliency | E | 2 |
| Scalability | L | 3 |
| Management | M | 4 |
| Unknown | N | 5 |
| Operational | O | 6 |
| Performance | P | 7 |
| Reliability | R | 8 |
| Security | S | 9 |

Mathematically, the label set is expressed in Equation 6.5, and the labelled dataset $X_L$ with SaaS service factors can be represented as expressed in Equation 6.6.

$$Y = \{A, C, E, L, M, N, O, P, R, S\} \tag{6.5}$$

$$X_L = \{(s_1, y_1), (s_2, y_2), \dots, (s_n, y_n) \mid s \in S_T \mid y \in Y\} \tag{6.6}$$

### 6.3.3 Sentiment intensity computation stage

The objective of this stage is to compute the sentiment intensity score for each of the text segments from previous subsection, which we do by using Valence Aware Dictionary for sEntiment Reasoning (VADER) (Hutto & Gilbert, 2015). VADER uses a reliable valence-based sentiment lexicon used widely in sentiment analysis of online product reviews, social media and other online systems (Kim et al., 2016) (Davidson et al., 2017) (Cheng et al., 2017) (Butticè et al., 2017). As its output, it provides a positive and negative sentiment valence (polarity + intensity) score for text segments. Each text segment is associated with a compound score (which is a normalized valence score) ranging between -1 to +1. A value of -1 indicates the most negative or worst sentiment intensity score, and a value of +1 indicates a positive or best sentiment intensity score. VADER computes the compound sentiment score for a text segment by summing up the valence scores for all the words in the text segment and then normalizing the sum between -1 and +1. The compound sentiment score for a given segment $s_i$ is calculated as below:

$$V_S = \sum_{j=1}^{n} V_j \tag{6.7}$$

$$s_i = \frac{V_S}{\sqrt{V_S \,.\, V_S + \alpha}} \tag{6.8}$$

where $V_s$ represents the sum of valance scores and $\alpha$ is used to approximate the max expected value.

As a result of this computation, the sentiment intensity values for each feature *j* of the SaaS product *i* is a value between -1 and 1 and is written as:

$$S_{ij} \in [-1\ 1]\ (i = 1,2, \dots, n; j = 1,2, \dots, m) \tag{6.9}$$

At the end of this stage, for a given product and for each of the identified service factors which the customer mentions in their reviews, the corresponding sentiment intensity scores are aggregated. For instance, if product *A* has 10 text segments associated with it and each of the 10 segments have

identified the service factor *R (Reliability)*, then the sentiment intensity score for service factor *R* for product *A*, is the aggregated sentiment scores over all the 10 segments is calculated as:

$$\acute{S}_{AR} = \frac{\sum_{j=1}^{n} s_{AR_j}}{n} \tag{6.10}$$

where *n=10* in this case.

As a result of this analysis a dataset is created that include the aggregated sentiment intensity scores for the services factors associated with each SaaS product. If the review segments do not mention any of the service factors, then they are assigned with the label '*N*' and the sentiment scores for those segments are aggregated under label '*N*'. This value is very helpful and needed in the inference stage where the overall sentiment intensity score is needed (regardless of the sentiment intensity scores for each factor).

### 6.3.4 Inference stage

The objective of this stage is to infer or impute the sentiment intensity values in the service factors that do not have any associated user reviews and hence missing sentiment intensity score. In our methodology, we achieve this by using the T-NN approach which is an enhanced version of the kNN approach. In the traditional kNN approach the *k* most similar variables, based on their distance to the candidate variable being studied, are used to decide which class it should belong to. When used in classification problems, the classes of the selected *k* instances are used to find the class label with higher votes (most frequent) as the selected class of the candidate variable. When used in regression, the aggregated value (mean) of the selected *k* instances is used to determine/infer the value of the candidate variable. However, the traditional implementation of the kNN approach has two main drawbacks:

1. It assumes equal weights for all the variables (features) in the dataset when calculating the distances between the candidate variable and other variables. This assumption can negatively affect the prediction results, as variables in real datasets tend to correlate at different levels. Thus, assigning all the variables the same weight undermines correctly estimating the values of the candidate variable.
2. Selecting *k* instances based on their distances from the candidate variable can also lead to unrealistic estimations. For example, if the distance among the *k* selected variables is too large, it can increase the error when inferring the values of the candidate variable. This is especially relevant in cases of regression when the aggregated values from the *k* variables are used to

determine the value of the candidate variable, which is the nature of the problem dealt with in this chapter.

To address the first drawback, in T-NN we used a weighted Euclidean measure to calculate the distances among the different variables (features or service factors). The weight to be applied to any two features is determined by the Pearson's correlation coefficient that measures the strength and direction of relationship between them. The rationale here is that if a feature is strongly (either positively or negatively) related to the candidate feature (which has the missing value), then a higher weight should be placed on the value of the feature while inferring the value of the candidate feature. To address the second drawback, we introduce a threshold parameter ($T$) to select the donor instances that I consider inferring the value of the candidate feature. The parameter $T$ replaces the traditional selection approach based on the $k$ closest instances and select 'the most relevant' donor features that improve the accuracy of the inferred values rather than increasing the prediction error. Figure 6.1 shows the five different steps in our proposed T-NN approach to impute the sentiment intensity value in a feature (service factor) of a SaaS product (instance).

***Step 1: Finding donor instance(s) for the candidate instance***

A *candidate instance* refers to that SaaS product in which the sentiment intensity value of a feature is to be imputed. A *donor instance* refers to that instance which can be used to impute the missing sentiment value in the feature of the candidate instance. In order to be a donor, an instance must fulfil the following two conditions:

1. it should have the sentiment intensity value in the feature for which it needs to be imputed in the candidate instance.
2. it should have values in those features where the donor has observed values

In other words, the donor instance should have the sentiment intensity value in the feature to be imputed in the candidate instance.

Representing this mathematically, if $X$ is the dataset with $n$ instances $\{x_1, x_2, x_3, \ldots, x_n\}$ and $m$ features $\{k_1, k_2, k_3, \ldots, k_m\}$, where the features contain the sentiment intensity scores, then the sentiment intensity value of the $i^{th}$ instance in $j^{th}$ feature can be represented as $S_{ij}$. If the value in a candidate $x_{ij}$ needs to be imputed, then those instances that have a value in the $j^{th}$ feature are donor instances $D = \{d_1, d_2, d_3, \ldots, d_m\}$.

$$D = \{x_i \in D | \, D \subseteq X, x_{ij} \neq NA \} \tag{6.11}$$

where, $x_{ij}$ represents the $j^{th}$ feature of the $i^{th}$ instance in the dataset and '*NA*' represents a missing sentiment intensity value.

***Step 2: Determining the degree of correlation between features***

Pearson's correlation coefficient (PCC) is one of the most used measures of the strength of the linear relationship between two variables. Correlation also known as the bivariate correlation is used to measure the strength and direction of relationship between two variables (Lee Rodgers & Nicewander, 1988). The values of the correlation vary between +1 and -1. A positive correlation between two features indicates that the value of a feature tends to increase (positive) with an increase in the value of the other feature and vice versa. A correlation value of +1 means that there is a perfect positive relationship between the two variables. On the other hand, a negative correlation between two variables indicates that the value of the feature tends to decrease (negative) with the increase in the other feature and vice versa. A correlation value of -1 means that there is a perfect negative relationship between the two variables. A correlation value of 0 means that there is no relationship between the two variables. The correlation coefficient for two variables is calculated as:

$$\rho_{xy} = \frac{Cov(x,y)}{\sigma_x \, \sigma_y} \tag{6.12}$$

where *Cov(x, y)* represents the covariance of the two variables *x* and *y* and is calculated as:

$$Cov(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1} \tag{6.13}$$

where $x_i$ and $y_i$ are the $i^{th}$ values (data points) of the variables *x* and *y*, and $\bar{x}$ and $\bar{y}$ are the mean values of the variables *x* and *y*.

$\sigma_x$ represents the standard deviation of *x* and $\sigma_y$ represents the standard deviation of *y* and is calculated as follows:

$$\sigma_x = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{6.14}$$

Similarly, the standard deviation of *y* can be calculated by using Equation 6.14 with values related to *y*. These coefficients are used when determining the distance between the features of the candidate and the donor instances in the next step. It should be noted that the correlation values calculated from the dataset that has no missing values are different from the ones with missing values in their features. In such a case, the correlation values between the features needs to be computed according to the specifics of the dataset. The PCC values among the features of the dataset are depicted in Table 6.3.

Table 6.2. Correlation values among the features

| | **A** | **C** | **E** | **L** | **M** | **N** | **O** | **P** | **R** | **S** |
|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 1.0000 | 0.1769 | 0.2258 | 0.0512 | 0.1970 | 0.1881 | 0.1183 | 0.2172 | 0.2095 | 0.0939 |
| **C** | 0.1769 | 1.0000 | 0.2084 | 0.0002 | 0.2326 | 0.2437 | 0.2590 | 0.1538 | 0.0681 | 0.2929 |
| **E** | 0.2258 | 0.2084 | 1.0000 | 0.4756 | 0.2375 | 0.3846 | 0.2009 | 0.2866 | 0.2556 | 0.7025 |
| **L** | 0.0512 | 0.0002 | 0.4756 | 1.0000 | 0.1129 | 0.2068 | -0.0397 | 0.1194 | 0.0818 | 0.4026 |
| **M** | 0.1970 | 0.2326 | 0.2375 | 0.1129 | 1.0000 | 0.4302 | 0.3606 | 0.2307 | 0.3652 | 0.3623 |
| **N** | 0.1881 | 0.2437 | 0.3846 | 0.2068 | 0.4302 | 1.0000 | 0.2827 | 0.2422 | 0.3635 | 0.3681 |
| **O** | 0.1183 | 0.2590 | 0.2009 | -0.0397 | 0.3606 | 0.2827 | 1.0000 | 0.1961 | 0.2749 | 0.3573 |
| **P** | 0.2172 | 0.1538 | 0.2866 | 0.1194 | 0.2307 | 0.2422 | 0.1961 | 1.0000 | 0.2547 | 0.0546 |
| **R** | 0.2095 | 0.0681 | 0.2556 | 0.0818 | 0.3652 | 0.3635 | 0.2749 | 0.2547 | 1.0000 | 0.2829 |
| **S** | 0.0939 | 0.2929 | 0.7025 | 0.4026 | 0.3623 | 0.3681 | 0.3573 | 0.0546 | 0.2829 | 1.0000 |

***Step 3: Determine the correlation weighted distance between the donor instance(s) and the candidate instance***

In this step, the distance between the donor instance and the candidate instance is computed. The distance between any two variables is computed by using one of many techniques such as Euclidean, Manhattan, cosine. Among these techniques, Euclidean distance measure is the most commonly used one as it is a good choice when all the input variables contain similar types of data (Brownlee, 2016b). This fits the input data of our methodology that consists of sentiment intensity scores scaled between -1 and +1. The distance between a candidate instance $x_i$ and donor instance $x_j$ is computed across all the features $m$ that have sentiment intensity values in both the instances. Mathematically, the distance between the two instances $x_i$ and $x_j$ are calculated as follows:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{m} (x_{ik} - x_{jk})^2} \tag{6.15}$$

where $x_i$ and $x_j$ represent the candidate and the donor instance respectively,

$m$ represents the number of features that have sentiment values in features of both the instances.

As discussed earlier, our proposed T-NN approach uses a weighted version of the Euclidean distance to accurately determine the distance between the donor and the candidate instances. This is done by using weight $w_k$ from the perspective of the feature whose value needs to be imputed. This weight is applied while computing the distance between any two features of the candidate and donor instances. $w_k$ is calculated by determining the correlation between the candidate feature and other features. Once

the $w_k$ values between the candidate and each other feature is determined, the distance between $x_i$ and $x_j$ is calculated as shown in Equation 6.15:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{m} w_k (x_{ik} - x_{jk})^2} \tag{6.16}$$

where $w_k$ represents the weight for the feature $k$ and is calculated as:

$$w_k = 1 - |\rho_{kl}| \tag{6.17}$$

where, $\rho_{kl}$ represents the pearson's correlation coefficient between the features $k$ and $l$.

***Step 4: Finding the optimal value for the threshold parameter (T) to select donor instances for imputation***

The threshold parameter *(T)* is one of the core components of the T-NN approach. As explained in the previous section, using the threshold range to select the donor instances is more suitable as compared to selecting instances based on counts. Assuming a given dataset $X$ and a set of donor instances $D$, where $D \subseteq X$, the threshold value $T$ helps to select a subset $\overline{D}$ from the donor instances from $D$. If $S_{dc}$ is the distance of the closest donor instance $d_c$ to the candidate instance, then the members of $\overline{D}$ are selected as:

$$\{\mathrm{d} \mid \mathrm{d} \in \mathrm{D}, \mathrm{S}_d - \mathrm{S}_{d_c} \leq \mathrm{T}\} \tag{6.18}$$

The threshold value defines the closeness among the selected donor instances. $T$ for a given dataset needs to be determined according to the understanding of the features in it and the count with missing values. This helps us to reduce errors in the estimation process as compared to only just randomly considering the value of $T$. A range of threshold values is tested to find the best one in the tunning process. The threshold value is considered the best, when it is used in the T-NN inference process and results in the lowest error value i.e., the predicted value is the closest to the actual value. When the value of $T$ is set to 0, only the donor instance that has the least distance from the candidate instance is selected.

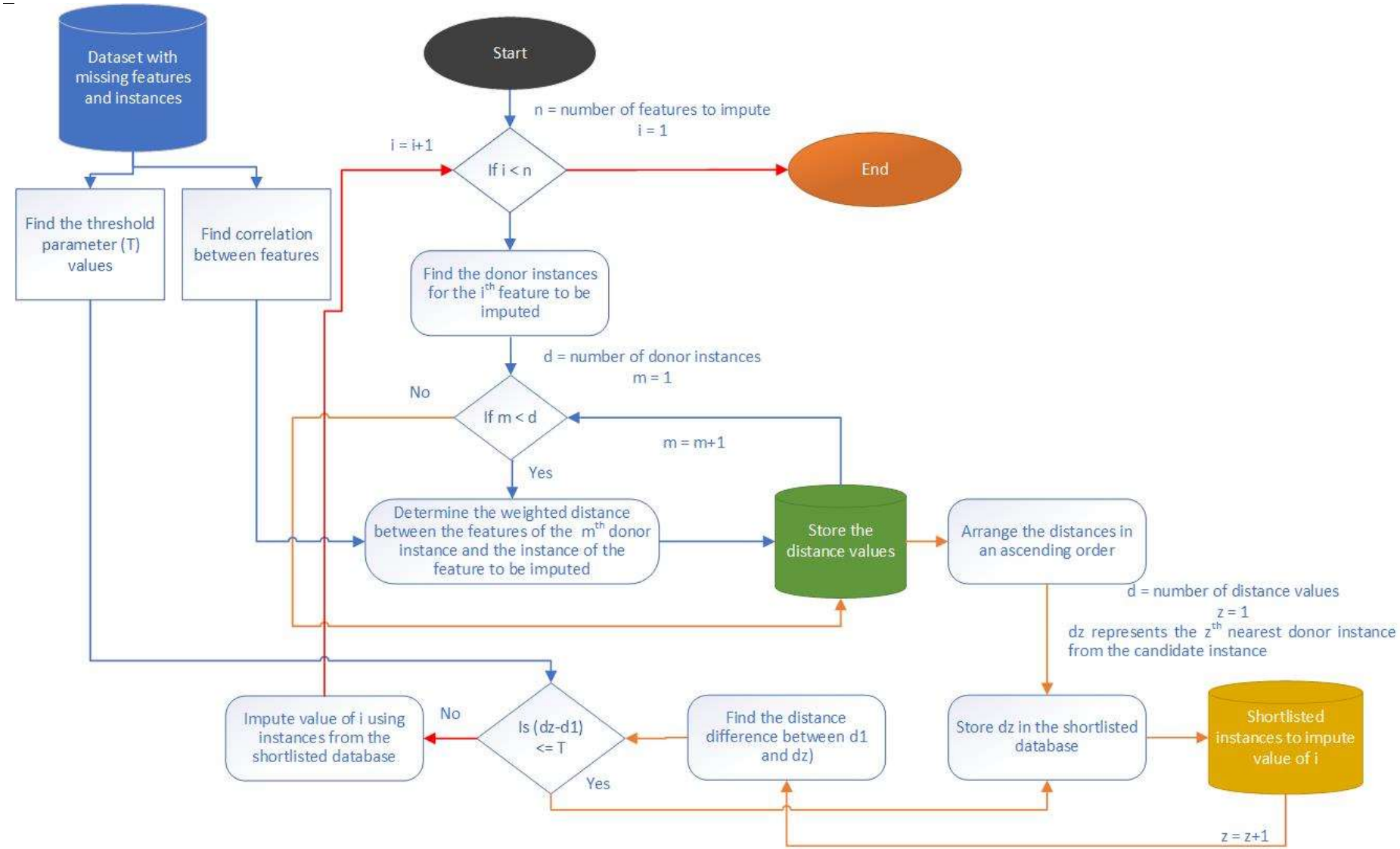



Figure 6.2 Sequence of steps in imputing the missing sentiment intensity values in a dataset

***Step 5: Imputing the missing values of the feature in the candidate instance from the donor instances within the threshold***

The observed values in each variable $j$ from $\overline{D}$ are then averaged to determine the value of the candidate feature of instance $x_i$ where $x_i \in X$.

$$x_{ij} = \frac{\sum_{k=1}^{n} d_{kj}}{n}, d \in \overline{D} \tag{6.19}$$

where d represents the donor instances in $\overline{D}$ and n represents the total number of instances in $\overline{D}$.

Figure 6.2 shows the process of imputing the sentiment intensity value of the missing features in a dataset. To summarise, the step-by-step process of T-NN in imputing the value of a feature in a candidate instance from donor instances are as follows:

1. Identify all missing values locations in *X.*
2. For each missing value in the candidate instance $x_i$ on a specific variable $k_j$, select the donor instances *D* based on the following conditions, (a) donor instances must have the value for that specific variable $k_j$, (b) donor instances must at least have values for the features that have observed values in the candidate instance $x_i$. The donor instances $D = \{d_1, d_2, d_3, ...., d_m\}$ *is a subset of X* (i.e., $D \subseteq X$).
3. Calculate the Pearson's correlation coefficients among the variables in X by using Equation 6.12.
4. Calculate the weighted Euclidean distance using Equation 6.16 between donor instances in *D* and candidate instance $x_i$. The weights for each feature are calculated using the PCC between all the features in step 3. Assuming that the data point $x_{il}$ has the value missing, where $l$ represents a given variable with missing value, then the weight for the variable $k$ is calculated as $w_k = 1 - |\rho_{kl}|$. $\rho_{kl}$ is the PCC between $l$ and $k$. For example, in relation to the missing value in $x_{il}$, when calculating the distance between $x_i$ and $d_j$, the weights for the variable $k$ are assigned based on the PCC value between variable $l$ and $k$.
5. Sort the donor instances according to their distances
6. Select the donors $\overline{D}$ (where, $\overline{D} \subseteq D$) falling within the threshold range. In this step, the donor instance with the least distance $d_c$ is selected first. Other donor instance(s) are then selected if their distance(s) from instance $d_c$ falls within the given threshold range.
7. The values from the variable $j$ of the selected donor instance $\overline{D}$ is aggregated and assigned to the variable $l$ of the candidate instance $x_i$ (the datapoint with the missing value)

8. Steps 2 to 7 is repeated to impute all the missing values in *X*.

### 6.3.5 Validation stage

The objective of this stage is to test the accuracy of the sentiment intensity values imputed in the previous step. The validation is done in two schemes namely scheme *A* and *scheme B*. Each scheme uses a different dataset for experimentation, evaluation procedure and comparison techniques to ascertain the accuracy of T-NN imputation. In both the schemes, the performance of T-NN is compared against the well-known missing values imputation approaches shown in Table 6.3.

Table 6.3 Imputation approaches

| **Imputation approaches** | **References** |
|---|---|
| K Nearest Neighbours | (Batista & Monard, 2003) (Troyanskaya et al., 2001) |
| Singular-Value Decomposition | (Troyanskaya et al., 2001) (Ghazanfar & Prügel-Bennett, 2013) |
| Decision Tree | (Rockel et al., 2017) (Vateekul & Sarinnapakorn, 2009) |
| Stochastic Gradient Descent | (Ma & Needell, 2018) (Sportisse et al., 2020) |
| Expectation Maximization | (Rahman & Islam, 2011) (García-Laencina et al., 2010) |

The details of the validation process of each scheme are explained in the following subsections.

***Error Measures***

The two error measures used in all the experiments in this chapter are Root Mean Squared Error and Adjusted R-Squared. Both of these error measures are used during evaluation and testing during:

Tunning the threshold value T using the SVR model in both Scheme A and Scheme B

Performance evaluation of all the approaches including T-NN in Scheme A

Performance evaluation of all the approaches including T-NN in Scheme B using SVR

*Root Mean Squared Error*

The root mean squared error (RMSE) is a very common metric used in regression analysis where the values to be compared are continuous in nature. The RMSE is calculated as the standard deviation of the difference between the true and imputed values (residuals).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_i^{true} - X_i^{predicted})^2}{n}} \tag{6.20}$$

where $X_i^{true}$ represents the $i^{\text{th}}$ observed value of the variable $X$ and $X_i^{predicted}$ represents the $i^{\text{th}}$ imputed value of the variable $X$.

The smaller values of RMSE (closer to 0) indicates a better performance by an estimator on the imputed data. In other words, the RMSE values on an imputed dataset translates to the performance of the imputation (or prediction) approaches.

*R-Squared (Adjusted)*

R-squared computes the coefficient of determination. It measures the proportion of variance in dependent variable accounted by the independent variables. R-squared is calculated as the ratio of the sum of squares of residuals to the total sum of squares.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{6.21}$$

where $y_i$ is the true value for the $i^{th}$ instance in $y$, $\hat{y}_i$ is the corresponding predicted value and $\bar{y}$ is the mean of $y$ over $n$ instances.

An R-squared value of 1 is the best outcome. One weakness of R-squared is that its value tends to increase with the increase in the number of the features. Adjusted R-squared (adjusted coefficient of determination) can be used instead, which penalizes addition of features that does not bring significant improvement to the performance of the model.

$$Adjusted\ R^2 = 1 - (1 - R^2)\frac{n - 1}{n - (p + 1)} \tag{6.22}$$

where $n$ represents number of instances and $p$ represents the total number of features (independent variables).

***Scheme A***

The core idea in *Scheme A* validation is to replace the actual sentiment intensity values of random features and randomly selected instances by missing values. By using T-NN, the missing values are imputed and then compared with the actual values to ascertain their accuracy. This process is repeated 10 times with varying percentages of missing instances in each iteration. The values of missing percentages are 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50. The dataset for experiment is created under the MCAR assumption by randomly inserting missing values into the features. The missing instances in the dataset vary according to the different values of percentage considered. Furthermore, in the missing instances, a random number of features were marked as having missing values within each instance of the subsets. The features to have missing values in an instance were selected randomly but which should satisfy the condition of there being at least two features with observed values

(sentiment intensity scores) in an instance. The other values can be missing. This condition is chosen to allow the imputation approaches used in this study to utilize the value from the existing feature to impute the missing values. The detailed steps in Scheme A are depicted in Figure 6.3.

To determine the accuracy of the imputed values in scheme A by T-NN, they are compared with the observed values. Other approaches from the literature such as SGD, K-Neighbours, Mean, Median, SVD, Decision Tree and EM too are used to impute the values in the missing instances. The performances of the approaches are quantified and compared by determining their root mean squared error (RMSE) and the adjusted R-squared (Adj-$R^2$).

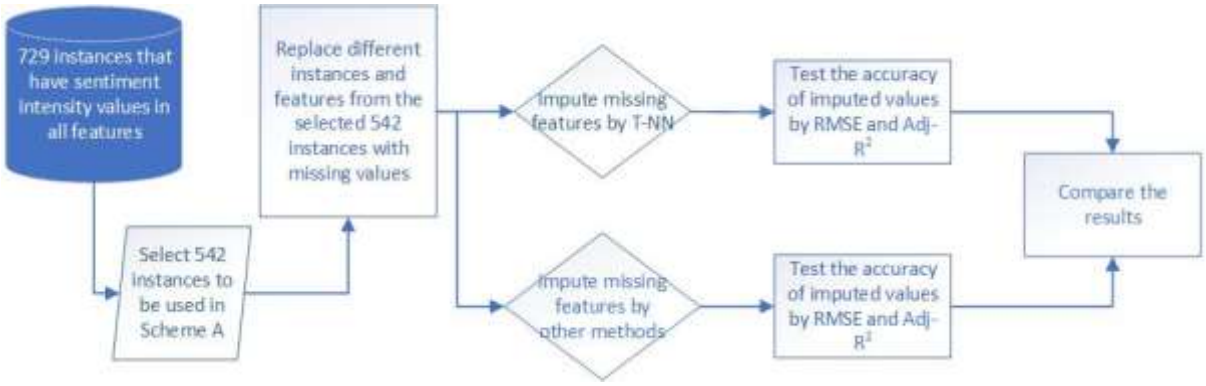



Figure 6.3 Steps in the validation process of Scheme A

***Scheme B***

The core idea behind *Scheme B* validation is to impute the random missing values in different features of the generated datasets using T-NN, then train an SVR estimator on these imputed datasets, and evaluate the performance of the SVR estimator. This scheme helps to evaluate the effect on the performance of SVR when used on imputed datasets. Similarly, the other imputation approaches generated their imputed datasets and SVR is trained on those datasets. Finally, the performance of the SVR on all the imputed datasets by different approaches including T-NN is compared.

For scheme B, we started with a dataset of 542 product instances from the 729 available ones. The remaining 187 instances were kept for testing purposes as explained later. In *scheme B* the missing values are introduced to the dataset by adding new data instances that contain missing values in random features and test the accuracy of T-NN in imputing them. Scheme B differs from Scheme A in two ways, as follows:

- Dataset: Scheme B adds additional data instances with missing values to the initial dataset. As seen in Table 6.8, in scheme A, all the datasets have the same number of total instances

(542) with varying levels of missing instances and features. Scheme B, as shown in Table 6.8, starts with a dataset of 542 instances. It then creates 10 more datasets by adding new instances that have missing values to the initial dataset. The missing instances have varying levels of missing values in each feature as compared to the different datasets of Scheme B and Scheme A. For example, Table 6.8 shows that the variable *E* which represent the '*Resiliency*' service factor, has the most missing values compared to the other datasets of the scheme. The variable *M* representing the '*Management or DevOps*' service factor, has the least number of missing values in each dataset. The variable *N* which represents the segments that are not associated with any service factor, also have fewer missing values. Adding new instances and features with missing values represents an online scenario where the incoming data may be incomplete to different degrees. The objective now is to test how effective and accurate the proposed T-NN approach is to impute the missing values in such scenarios.

- Validation approach: Scheme B uses a separate machine learning-based approach to test the accuracy of T-NN to impute missing values. It consists of the following steps:
    - After the missing values in each dataset are imputed, the dataset is considered as the training dataset. The 187 instances which had service intensity values in all the features from the initial dataset are termed as the testing dataset.
    - In both the training and testing dataset, the feature variable *M* that had the least number of missing values (from Table 6.8) is considered as the output variable.
    - On the training dataset, SVM is used to train and determine the sentiment intensity value of the feature with label *M* based on the values of the other features. The training dataset is divided into 10 folds x 3 repetitions and the accuracy in the prediction of each is evaluated by using RMSE and R-squared.
    - Once the SVR model is trained, it is applied on the unseen testing dataset that has 187 instances with sentiment intensity value in each feature. While testing, the objective is to predict the values of the feature variable *M* and compare it with the observed values of M in the testing dataset. The predicted output is then compared with the actual output and the accuracy of the approach is determined by using RMSE and Adj-$R^2$.
    - The SVR model is also compared against the baseline performance model by using the RMSE and Adj-$R^2$ techniques.

- Apart from the dataset imputed from T-NN, the above process is also applied on the datasets imputed from SGD, K-Neighbours, Mean, Median, SVD, Decision Tree and EM approaches. The value in feature variable *M* is predicted in each approach and compared with its actual value.

*Performance evaluation with SVR:*

The Support Vector Machine (SVM) was initially introduced by Vapnik (Vapnik, 1998) to solve binary classification problems (V. Vapnik, 1999). Beside classification, the idea of support vector machines has been extended and successfully used in regression problems as support vector for regression (SVR) (Wu et al., 2005) (Yu et al., 2006). SVM belongs to the family of the margin-based classification (and regression) approaches where the hyperplane that separates the instances is constructed using the maximum margin in the feature space. The generic estimation function for SVR can be written as:

$$f(x) = w \cdot x + b \tag{6.23}$$

The SVR is implemented using different kernels. These kernels can be linear and non-linear in nature. Non-linear kernels such as radial basis function (RBF) are more useful when the training data cannot be separated by a straight line (linearly). A linear SVR kernel function takes the form:

$$k(x, x_i) = x \cdot x_i \tag{6.24}$$

whereas, the RBF function kernel is:

$$K(x, x_i) = exp\left(-\gamma\left|\left|x - x_i\right|\right|^2\right), \gamma > 0 \tag{6.25}$$

In scheme B, SVR is used during the cross-validation and on the test dataset. We have also used SVR in tuning the T-NN for the optimal threshold value in both the schemes. The evaluation is performed with both the Linear and RBF kernels. All the performance results by SVR in scheme B are compared to a baseline estimator implemented using a zero-rule algorithm with median as a central tendency (Brownlee, 2016a). Figure 6.4 shows the steps in the validation process of Scheme B.

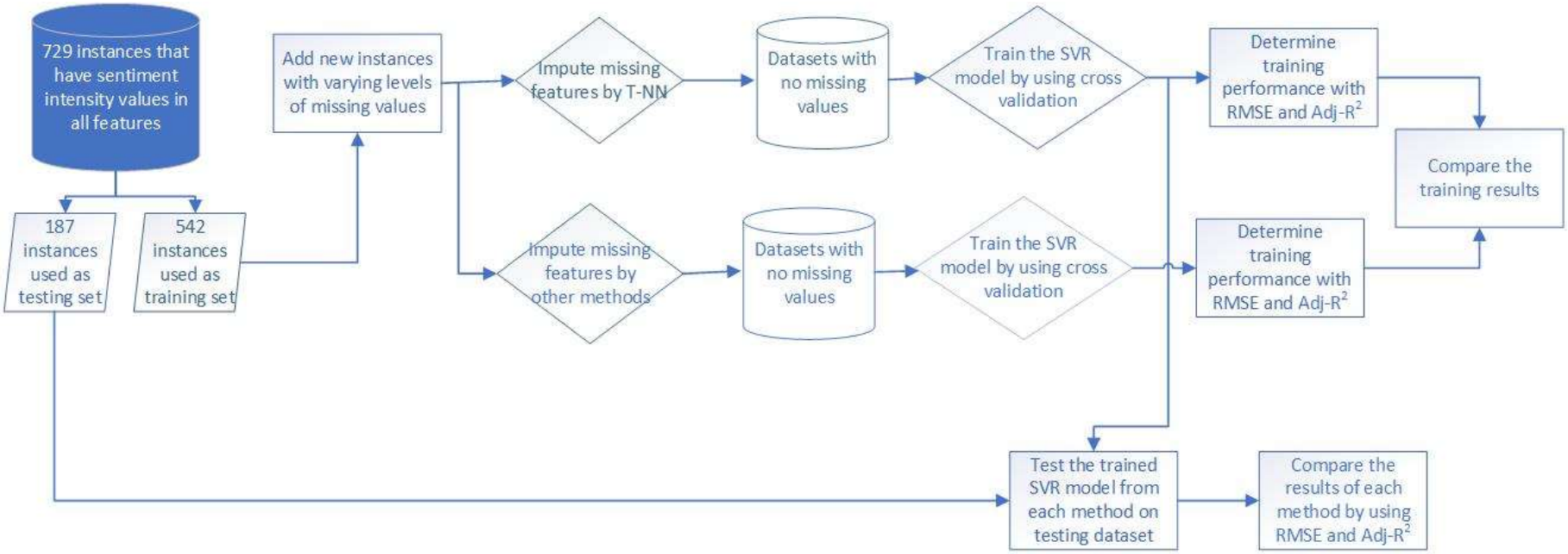



Figure 6.4 Steps in the validation process of Scheme B

## 6.4 EXPERIMENTS AND EVALUATION RESULTS

This section explains the steps in the implementation of Scheme A and Scheme B as depicted in Figure 6.3 and Figure 6.4.

### 6.4.1 Scheme A

***Dataset description***

The initial dataset is created following the steps discussed in the pre-processing stage of our proposed solution methodology which include the aggregated sentiment intensity scores for the services factors associated with each SaaS product. In this study we considered a dataset which had reviews for 3215 SaaS products. From those reviews, only 729 products were found as those that have an aggregated sentiment intensity score for all the (9) service factors. For scheme A, we created a dataset of 542 product instances from the 729 available which had the aggregated sentiment intensity values in all the features. The dataset for experiment is created under the MCAR assumption by randomly inserting missing values into the features.

Table 6.4. Datasets Description (scheme A)

| **Missing percentage** | **Total instances** | **Missing instances** | **Number of missing values in each feature** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **A** | **C** | **E** | **L** | **M** | **N** | **O** | **P** | **R** | **S** |
| **0** | 542 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 542 | 27 | 13 | 10 | 19 | 15 | 15 | 13 | 10 | 14 | 12 | 14 |
| **10** | 542 | 54 | 25 | 26 | 25 | 20 | 30 | 23 | 16 | 27 | 21 | 19 |
| **15** | 542 | 81 | 39 | 36 | 42 | 33 | 38 | 35 | 32 | 43 | 29 | 35 |
| **20** | 542 | 108 | 48 | 42 | 47 | 49 | 54 | 52 | 46 | 56 | 42 | 47 |
| **25** | 542 | 136 | 68 | 55 | 67 | 62 | 58 | 70 | 68 | 73 | 59 | 64 |
| **30** | 542 | 163 | 79 | 71 | 85 | 80 | 78 | 70 | 77 | 77 | 68 | 82 |
| **35** | 542 | 190 | 79 | 86 | 88 | 78 | 74 | 93 | 80 | 93 | 76 | 87 |
| **40** | 542 | 217 | 93 | 92 | 94 | 92 | 99 | 96 | 105 | 102 | 97 | 98 |
| **45** | 542 | 243 | 111 | 111 | 126 | 101 | 119 | 108 | 110 | 121 | 116 | 114 |
| **50** | 542 | 271 | 116 | 127 | 122 | 116 | 116 | 136 | 121 | 126 | 124 | 118 |

As discussed earlier, the core idea in *scheme A* is to replace the actual sentiment intensity values of random features and randomly selected instances by missing values. By using T-NN, the missing values are imputed and then compared with the actual values to ascertain their accuracy. This process is repeated 10 times with varying percentages of missing instances in each iteration. The values of missing percentages are 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50. As shown in Table 6.4, the missing instances in the dataset vary according to the different values of percentage considered. Furthermore, as shown in Table 6.4, in the missing instances, a random number of features were marked as having missing values within each instance of the subsets. The features to have missing values in an instance

were selected randomly but which should satisfy the condition of there being at least two features with observed values (sentiment intensity scores) in an instance. The other values can be missing. This condition is chosen to allow the imputation approaches used in this study to utilize the value from the existing feature to impute the missing values.

As shown in Table 6.4, for *scheme A*, the initial complete dataset consists of 542 instances (each instance representing a SaaS product and its related sentiment intensity scores for the service factors). Table 6.4 shows that for each of the generated dataset, the number of instances with missing values is increased with a 5 percent increment. In each of the generated datasets, the number of missing values in each of the variables (service factors) are different as well. Figure 6.3 shows in detail, the steps in the validation process of Scheme A. After the datasets are prepared; the Pearson's correlation coefficient is calculated for all the features in the dataset. Table 6.2 shows the PCC calculated for the feature with no missing values.

***Tunning for optimal threshold***

Tables 6.4 shows the different level/s of missing instances and features used in schemes A. For each dataset, we need to determine the threshold parameter *T.* We do that by tuning and determining the optimal threshold value for parameter *T* according to the following steps:

*Step 1:* We consider each dataset that has different percentages of missing values i.e., 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50.

*Step 2:* We define a possible set of *T* as *{*0, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20, 0.22, 0.24, 0.26, 0.28, 0.30, 0.32, 0.34, 0.36, 0.38 and 0.40}. In other words, the possible set represents the range from which the value of *T* is chosen.

*Step 3:* For each value of the possible set of *T,* apply the proposed T-NN approach to impute the missing values in each of the datasets from step 1. This result in 11 x 21 imputed datasets.

*Step 4:* Using SVR, run the 10 x 3 folds cross-validation on each of the imputed datasets from step 3 (using RMSE as the evaluation criteria).

*Step 5:* Select that T value that results in the lowest RMSE score aggregated over all the datasets with different missing percentages.

Table 6.5. RMSE values by increasing the threshold values (scheme A)

| | **0** | **0.02** | **0.04** | **0.06** | **0.08** | **0.1** | **0.12** | **0.14** | **0.16** | **0.18** | **0.2** | **0.22** | **0.24** | **0.26** | **0.28** | **0.3** | **0.32** | **0.34** | **0.36** | **0.38** | **0.4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 |
| **5** | 0.1371 | 0.1367 | 0.1367 | 0.1367 | 0.1334 | 0.1335 | 0.1335 | 0.1331 | 0.1333 | 0.1332 | 0.1331 | 0.1332 | 0.1333 | 0.1333 | 0.1332 | 0.1333 | 0.1333 | 0.1333 | 0.1333 | 0.1335 | 0.1336 |
| **10** | 0.1425 | 0.1350 | 0.1325 | 0.1310 | 0.1300 | 0.1296 | 0.1298 | 0.1301 | 0.1299 | 0.1297 | 0.1297 | 0.1298 | 0.1298 | 0.1297 | 0.1297 | 0.1298 | 0.1298 | 0.1298 | 0.1299 | 0.1299 | 0.1301 |
| **15** | 0.1323 | 0.1316 | 0.1306 | 0.1307 | 0.1303 | 0.1290 | 0.1295 | 0.1288 | 0.1291 | 0.1290 | 0.1288 | 0.1291 | 0.1292 | 0.1294 | 0.1294 | 0.1295 | 0.1292 | 0.1294 | 0.1296 | 0.1296 | 0.1297 |
| **20** | 0.1355 | 0.1344 | 0.1340 | 0.1337 | 0.1330 | 0.1313 | 0.1311 | 0.1315 | 0.1311 | 0.1309 | 0.1306 | 0.1303 | 0.1306 | 0.1306 | 0.1306 | 0.1307 | 0.1308 | 0.1309 | 0.1313 | 0.1314 | 0.1315 |
| **25** | 0.1354 | 0.1314 | 0.1314 | 0.1311 | 0.1312 | 0.1309 | 0.1307 | 0.1307 | 0.1308 | 0.1308 | 0.1307 | 0.1307 | 0.1307 | 0.1309 | 0.1309 | 0.1308 | 0.1309 | 0.1310 | 0.1311 | 0.1312 | 0.1313 |
| **30** | 0.1349 | 0.1332 | 0.1297 | 0.1295 | 0.1294 | 0.1278 | 0.1280 | 0.1277 | 0.1280 | 0.1276 | 0.1281 | 0.1278 | 0.1283 | 0.1283 | 0.1285 | 0.1286 | 0.1287 | 0.1287 | 0.1287 | 0.1289 | 0.1292 |
| **35** | 0.1317 | 0.1297 | 0.1298 | 0.1299 | 0.1290 | 0.1291 | 0.1286 | 0.1285 | 0.1277 | 0.1277 | 0.1282 | 0.1285 | 0.1286 | 0.1286 | 0.1288 | 0.1293 | 0.1292 | 0.1294 | 0.1302 | 0.1302 | 0.1304 |
| **40** | 0.1311 | 0.1285 | 0.1283 | 0.1273 | 0.1258 | 0.1251 | 0.1255 | 0.1252 | 0.1252 | 0.1253 | 0.1251 | 0.1249 | 0.1248 | 0.1249 | 0.1250 | 0.1252 | 0.1251 | 0.1253 | 0.1253 | 0.1253 | 0.1254 |
| **45** | 0.1347 | 0.1288 | 0.1255 | 0.1260 | 0.1260 | 0.1253 | 0.1250 | 0.1250 | 0.1251 | 0.1251 | 0.1251 | 0.1251 | 0.1250 | 0.1248 | 0.1250 | 0.1253 | 0.1256 | 0.1259 | 0.1259 | 0.1260 | 0.1261 |
| **50** | 0.1364 | 0.1311 | 0.1286 | 0.1297 | 0.1288 | 0.1276 | 0.1273 | 0.1271 | 0.1267 | 0.1259 | 0.1260 | 0.1257 | 0.1259 | 0.1261 | 0.1261 | 0.1263 | 0.1266 | 0.1268 | 0.1273 | 0.1273 | 0.1276 |
| **Avg** | 0.1351 | 0.1322 | 0.1310 | 0.1309 | 0.1301 | 0.1294 | 0.1294 | 0.1293 | 0.1292 | 0.1290 | 0.1291 | **0.1290** | 0.1291 | 0.1292 | 0.1292 | 0.1293 | 0.1294 | 0.1295 | 0.1297 | 0.1298 | 0.1299 |

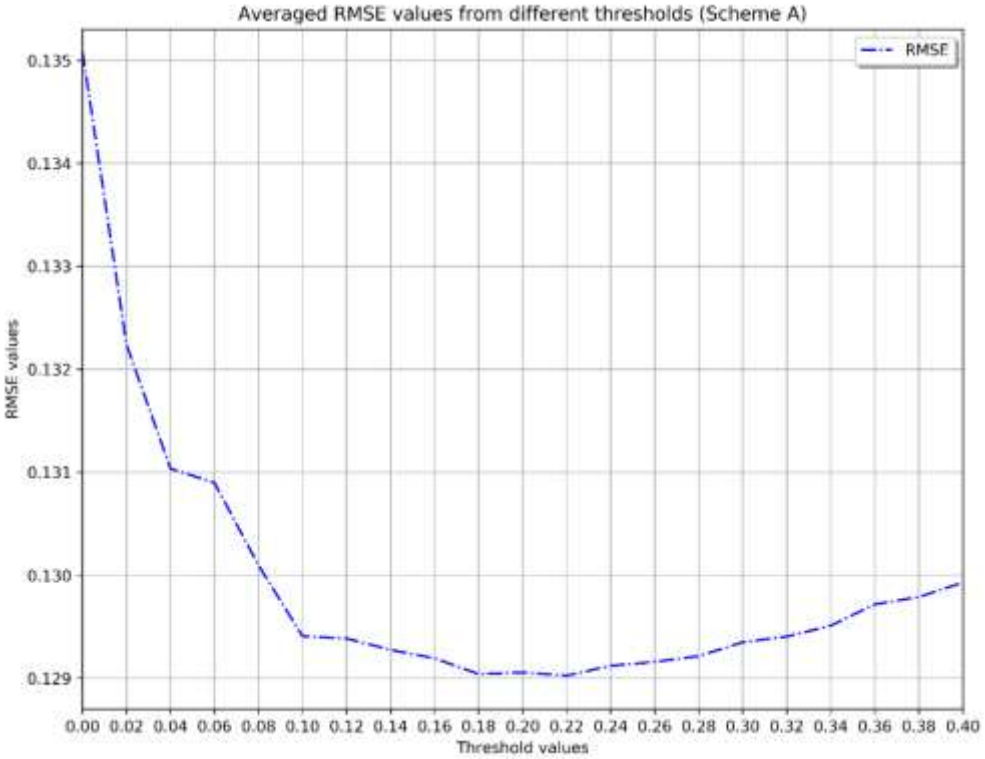



Figure 6.5 RMSE values averaged on all the datasets with different missing percentages of Scheme A

Table 6.5 depict the RMSE scores on each set of the imputed datasets used in Schemes A. It can be seen that the aggregated RMSE score tends to drop with an increase in the threshold value before rising again as shown in Figure 6.5. The lowest aggregated RMSE score of 0.12902 is observed at the threshold value *(T)* of 0.22. Using the lowest values of *T* in each scheme, the performance results of the proposed T-NN approach in imputing the missing features is compared with the results of other approaches in the next section.

***Performance evaluation***

Figure 6.6 plots the performance of the proposed T-NN approach against SGD, K-Neighbours, Mean, Median, SVD, Decision Tree and EM approaches in terms of their RMSE scores on different datasets. Table 6.6 shows the RMSE values of each approach on different datasets that have varying levels of missing values. From Table 6.6, it can be seen that T-NN outperforms the other imputation approaches with lesser RMSE scores on each dataset. The performance of EM is the worst among all the other approaches including the univariate Mean and Median imputation approaches. The performance of SGD, although it is close to the proposed T-NN approach, it is not better.

Table 6.6. RMSE scores with different missing percentages (scheme A)

| Imputers | Missing values percentage | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **T-NN** | 0.2324 | 0.2225 | 0.2128 | 0.2213 | 0.2280 | 0.2165 | 0.2246 | 0.2179 | 0.2141 | 0.2058 |
| **SGD** | 0.2465 | 0.2281 | 0.2187 | 0.2221 | 0.2291 | 0.2183 | 0.2252 | 0.2197 | 0.2144 | 0.2102 |
| **K-NN** | 0.2397 | 0.2260 | 0.2237 | 0.2293 | 0.2381 | 0.2211 | 0.2310 | 0.2258 | 0.2192 | 0.2178 |
| **Mean** | 0.2757 | 0.2412 | 0.2401 | 0.2453 | 0.2448 | 0.2396 | 0.2407 | 0.2375 | 0.2322 | 0.2243 |
| **Median** | 0.2800 | 0.2432 | 0.2422 | 0.2475 | 0.2460 | 0.2398 | 0.2421 | 0.2367 | 0.2324 | 0.2245 |
| **SVD** | 0.2653 | 0.2718 | 0.2524 | 0.2519 | 0.2592 | 0.2568 | 0.2482 | 0.2584 | 0.2570 | 0.2425 |
| **D Tree** | 0.3332 | 0.3086 | 0.2835 | 0.3109 | 0.3146 | 0.3191 | 0.3129 | 0.2952 | 0.3040 | 0.2829 |
| **EM** | 0.3943 | 0.3094 | 0.3165 | 0.3221 | 0.3280 | 0.3258 | 0.3290 | 0.3400 | 0.3287 | 0.3221 |

Table 6.7. Adjusted R-squared values with different missing percentages (scheme A)

| Imputers | Missing values percentage | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **T-NN** | 0.2826 | 0.1715 | 0.2403 | 0.2177 | 0.1736 | 0.2013 | 0.1435 | 0.1809 | 0.1638 | 0.1891 |
| **SGD** | 0.1925 | 0.1293 | 0.1975 | 0.2124 | 0.1658 | 0.1877 | 0.1390 | 0.1673 | 0.1618 | 0.1545 |
| **K-NN** | 0.2365 | 0.1457 | 0.1607 | 0.1603 | 0.0992 | 0.1672 | 0.0940 | 0.1201 | 0.1240 | 0.0914 |
| **Mean** | -0.0103 | 0.0269 | 0.0328 | 0.0388 | 0.0479 | 0.0222 | 0.0170 | 0.0272 | 0.0164 | 0.0371 |
| **Median** | -0.0414 | 0.0106 | 0.0163 | 0.0221 | 0.0380 | 0.0204 | 0.0048 | 0.0336 | 0.0149 | 0.0355 |
| **SVD** | 0.0646 | -0.2356 | -0.0683 | -0.0136 | -0.0679 | -0.1238 | -0.0452 | -0.1518 | -0.2042 | -0.1258 |
| **D Tree** | -0.4749 | -0.5938 | -0.3477 | -0.5437 | -0.5723 | -0.7351 | -0.6618 | -0.5034 | -0.6851 | -0.5317 |
| **EM** | -1.0655 | -0.6016 | -0.6802 | -0.6567 | -0.7098 | -0.8089 | -0.8377 | -0.9948 | -0.9702 | -0.9862 |

Figure 6.7 and Table 6.7 shows the performances of the approaches in terms of the adjusted R-squared metric. From the results, it can be seen that though there are approaches depicting lower values (i.e., lower proportion of variance is explained by the fits), T-NN fits relatively better than the others. The negative values for the adjusted R-squared in the cases of EM, DT and SVD indicate that these models have the worst fits on the datasets.

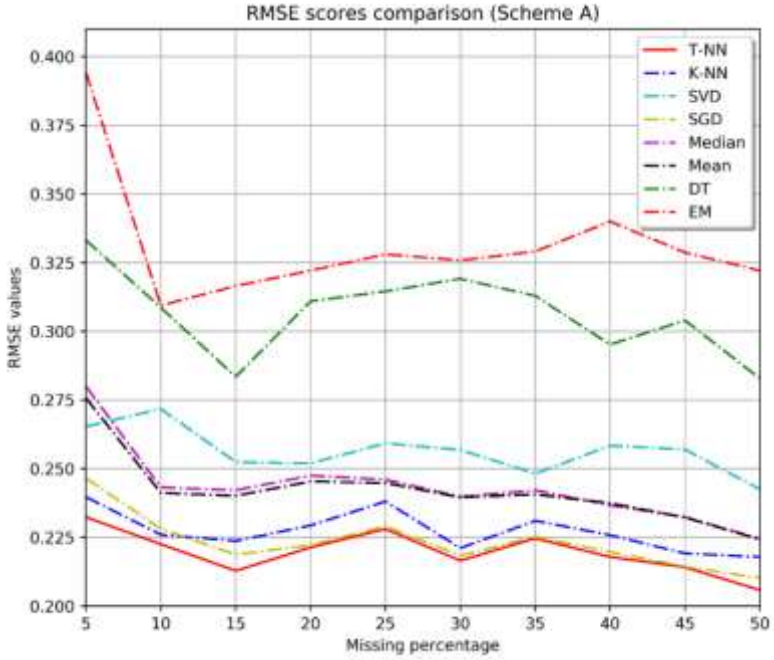



Figure 6.6 RMSE scores comparison on all imputed datasets with different missing percentages

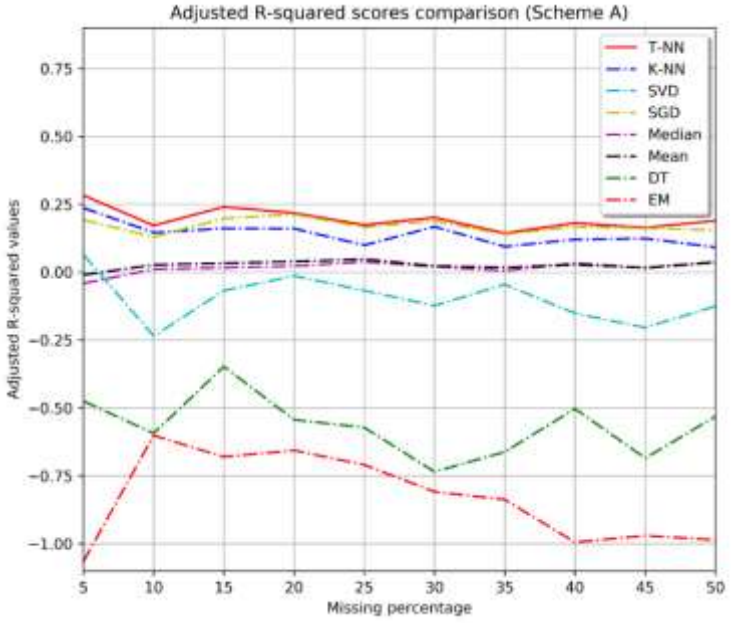



Figure 6.7 Adjusted R-squared scores comparison on all imputed datasets with different missing percentages

### 6.4.2 Scheme B

***Dataset description***

Initially the dataset is created following the steps in the pre-processing stage of our proposed solution methodology that include the aggregated sentiment intensity scores for the services factors associated

with each SaaS product. The analysis indicates that, out of 3215 products, only 729 products have the aggregated sentiment intensity scores for all the (9) service factors while the remaining products are missing the aggregated sentiment intensity scores for one or more service factors.

As mentioned in earlier, in this study we had only 729 products which had an aggregated sentiment intensity score for all the (9) service factors. For scheme B, we started with a dataset of 542 product instances from the 729 available ones. The remaining 187 instances were kept for testing purposes as explained later. As discussed in subsection, the main idea for *scheme B* is to introduce missing values to the dataset by adding new data instances that contain missing values in random features and test the accuracy of T-NN in imputing them.

Table 6.8. Datasets Description (scheme B)

| **Missing percentage** | **Total instances** | **Missing instances** | **Number of missing values in each feature** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **A** | **C** | **E** | **L** | **M** | **N** | **O** | **P** | **R** | **S** |
| **0** | 542 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 571 | 29 | 22 | 7 | 29 | 29 | 0 | 0 | 15 | 6 | 1 | 7 |
| **10** | 603 | 61 | 22 | 12 | 61 | 61 | 0 | 1 | 33 | 14 | 2 | 39 |
| **15** | 638 | 96 | 22 | 22 | 96 | 96 | 0 | 1 | 47 | 23 | 4 | 74 |
| **20** | 678 | 136 | 25 | 29 | 136 | 136 | 1 | 3 | 67 | 30 | 6 | 114 |
| **25** | 723 | 181 | 70 | 29 | 181 | 181 | 1 | 4 | 67 | 30 | 6 | 159 |
| **30** | 775 | 233 | 112 | 59 | 231 | 209 | 1 | 6 | 96 | 51 | 30 | 189 |
| **35** | 834 | 292 | 135 | 90 | 290 | 268 | 7 | 10 | 140 | 74 | 44 | 225 |
| **40** | 904 | 362 | 140 | 119 | 360 | 338 | 13 | 17 | 195 | 103 | 60 | 295 |
| **45** | 986 | 444 | 222 | 140 | 442 | 420 | 13 | 22 | 230 | 123 | 66 | 377 |
| **50** | 1084 | 542 | 317 | 170 | 539 | 504 | 17 | 30 | 291 | 147 | 89 | 469 |

***Tunning for optimal threshold***

Tables 6.8 shows the different level/s of missing instances and features used in schemes B. For each dataset, we need to determine the threshold parameter *T.* We do that by tuning and determining the optimal threshold value for parameter *T* according to the following steps:

<u>*Step 1:*</u> We consider each dataset that has different percentages of missing values i.e., 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 and 50.

<u>*Step 2:*</u> We define a possible set of *T* as *{*0, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18, 0.20, 0.22, 0.24, 0.26, 0.28, 0.30, 0.32, 0.34, 0.36, 0.38 and 0.40}. In other words, the possible set represents the range from which the value of *T* is chosen.

<u>*Step 3:*</u> For each value of the possible set of *T,* apply the proposed T-NN approach to impute the missing values in each of the datasets from step 1. This result in 11 x 21 imputed datasets.

*Step 4:* Using SVR, run the 10 x 3 folds cross-validation on each of the imputed datasets from step 3 (using RMSE as the evaluation criteria).

*Step 5:* Select that T value that results in the lowest RMSE score aggregated over all the datasets with different missing percentages.

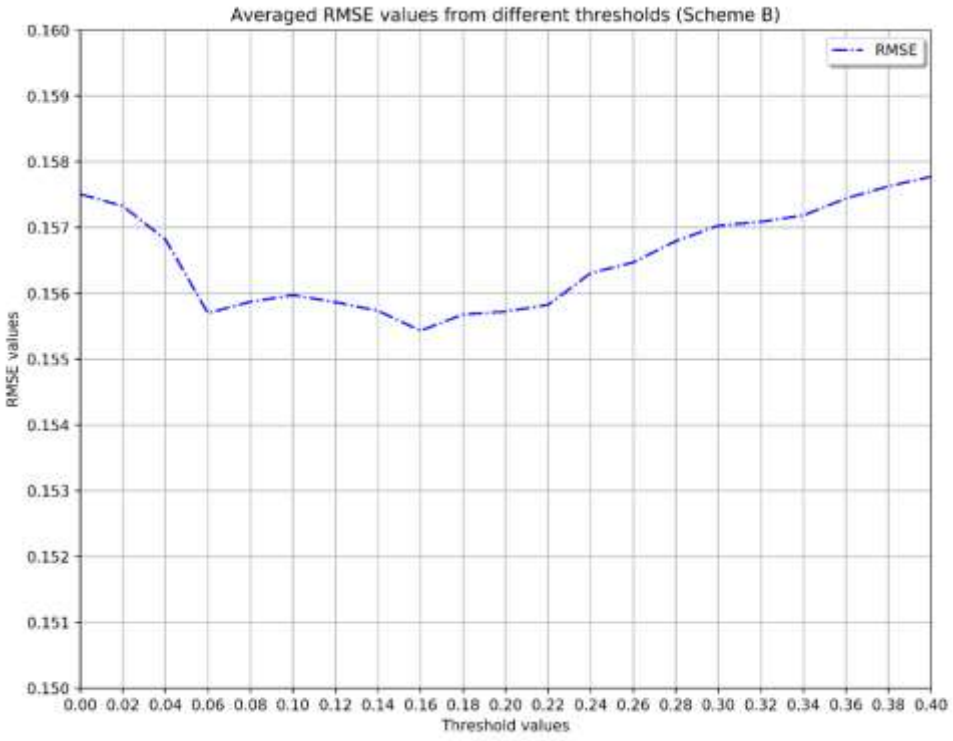



Figure 6.8 RMSE values averaged on all the datasets with different missing percentages of Scheme B

Tables 6.9 depict the RMSE scores on each set of the imputed datasets used in Schemes B. The RMSE value decreases with an increase in the *T* value before increasing again, as shown in Figure 6.8. From Table 6.9, it can be seen that the lowest aggregated RMSE score of 0.15543 is observed at the threshold of 0.16 before rising again. In the next section, using the lowest values of *T* in each scheme, the performance results of the proposed T-NN approach in imputing the missing features is compared with the results of other approaches.

Table 6.9. RMSE values by increasing the threshold values (scheme B)

| | **0** | **0.02** | **0.04** | **0.06** | **0.08** | **0.1** | **0.12** | **0.14** | **0.16** | **0.18** | **0.2** | **0.22** | **0.24** | **0.26** | **0.28** | **0.3** | **0.32** | **0.34** | **0.36** | **0.38** | **0.4** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 | 0.1342 |
| **5** | 0.1371 | 0.1377 | 0.1371 | 0.1370 | 0.1370 | 0.1371 | 0.1371 | 0.1373 | 0.1372 | 0.1373 | 0.1373 | 0.1373 | 0.1371 | 0.1372 | 0.1372 | 0.1372 | 0.1373 | 0.1373 | 0.1373 | 0.1373 | 0.1374 |
| **10** | 0.1410 | 0.1412 | 0.1407 | 0.1404 | 0.1403 | 0.1404 | 0.1402 | 0.1399 | 0.1397 | 0.1398 | 0.1397 | 0.1397 | 0.1397 | 0.1398 | 0.1399 | 0.1399 | 0.1400 | 0.1401 | 0.1401 | 0.1401 | 0.1403 |
| **15** | 0.1487 | 0.1486 | 0.1483 | 0.1467 | 0.1469 | 0.1470 | 0.1470 | 0.1470 | 0.1469 | 0.1472 | 0.1472 | 0.1472 | 0.1474 | 0.1475 | 0.1476 | 0.1478 | 0.1479 | 0.1479 | 0.1480 | 0.1481 | 0.1482 |
| **20** | 0.1505 | 0.1497 | 0.1501 | 0.1487 | 0.1488 | 0.1489 | 0.1492 | 0.1491 | 0.1488 | 0.1493 | 0.1493 | 0.1492 | 0.1497 | 0.1500 | 0.1505 | 0.1507 | 0.1507 | 0.1508 | 0.1512 | 0.1514 | 0.1516 |
| **25** | 0.1554 | 0.1538 | 0.1545 | 0.1526 | 0.1531 | 0.1534 | 0.1540 | 0.1536 | 0.1533 | 0.1537 | 0.1536 | 0.1534 | 0.1541 | 0.1544 | 0.1549 | 0.1552 | 0.1553 | 0.1553 | 0.1557 | 0.1559 | 0.1561 |
| **30** | 0.1629 | 0.1625 | 0.1619 | 0.1610 | 0.1608 | 0.1603 | 0.1606 | 0.1607 | 0.1600 | 0.1605 | 0.1606 | 0.1606 | 0.1609 | 0.1613 | 0.1616 | 0.1619 | 0.1619 | 0.1620 | 0.1625 | 0.1626 | 0.1628 |
| **35** | 0.1670 | 0.1668 | 0.1659 | 0.1646 | 0.1646 | 0.1645 | 0.1644 | 0.1639 | 0.1636 | 0.1639 | 0.1642 | 0.1640 | 0.1648 | 0.1650 | 0.1655 | 0.1658 | 0.1658 | 0.1659 | 0.1662 | 0.1665 | 0.1667 |
| **40** | 0.1747 | 0.1742 | 0.1729 | 0.1710 | 0.1712 | 0.1716 | 0.1713 | 0.1707 | 0.1704 | 0.1704 | 0.1705 | 0.1708 | 0.1717 | 0.1720 | 0.1726 | 0.1729 | 0.1731 | 0.1733 | 0.1736 | 0.1740 | 0.1742 |
| **45** | 0.1785 | 0.1786 | 0.1773 | 0.1756 | 0.1765 | 0.1769 | 0.1762 | 0.1760 | 0.1755 | 0.1755 | 0.1755 | 0.1760 | 0.1769 | 0.1768 | 0.1773 | 0.1777 | 0.1779 | 0.1781 | 0.1786 | 0.1789 | 0.1790 |
| **50** | 0.1827 | 0.1834 | 0.1821 | 0.1807 | 0.1811 | 0.1814 | 0.1804 | 0.1805 | 0.1801 | 0.1805 | 0.1808 | 0.1817 | 0.1829 | 0.1828 | 0.1833 | 0.1839 | 0.1839 | 0.1841 | 0.1846 | 0.1849 | 0.1850 |
| **Av** | 0.1575 | 0.1573 | 0.1568 | 0.1557 | 0.1559 | 0.1560 | 0.1559 | 0.1557 | **0.1554** | 0.1557 | 0.1557 | 0.1558 | 0.1563 | 0.1565 | 0.1568 | 0.1570 | 0.1571 | 0.1572 | 0.1574 | 0.1576 | 0.1578 |

***Performance evaluation***

As discussed earlier, *in scheme B,* the datasets are generated by adding instances with missing values to the complete dataset. Each generated dataset has different percentages of missing values with different total number of instances. The datasets are then imputed using the selected imputation approaches in this study. In *scheme B*, as the actual values for the missing datapoints are unknown, the imputed datasets by the selected approaches are evaluated using cross-validations and test dataset. The cross validation and test performances on the imputed datasets highlights the effectiveness of the imputation approaches on unseen data.

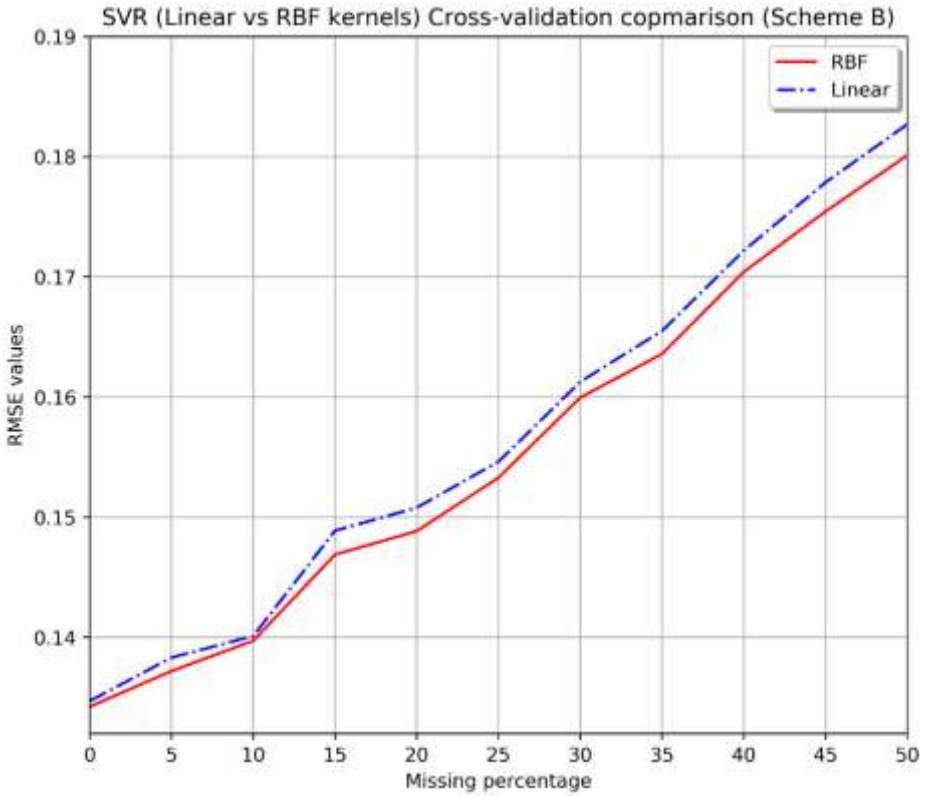



Figure 6.9 Cross-validation results from Linear and RBF kernels (RMSE) on T-NN imputed datasets

*SVR kernel selection*

Similar to the procedure applied in threshold tuning earlier, the cross-validation in *scheme B* is also performed using SVR. All the experiments are performed using SVR with a linear kernel and a non-linear (RBF) kernel and compared to a baseline implementation using a zero rule. The zero rule is implemented using the central tendency (mean values) of each variable. However, the results from

our experiments show that the RBF kernel performs better than the linear kernel with higher performance scores in Figures 6.9 and Figure 6.10. Therefore, we are reporting the results from the SVR with the RBF kernel only. In the next subsections, we show the results by performing cross-validation and test results.

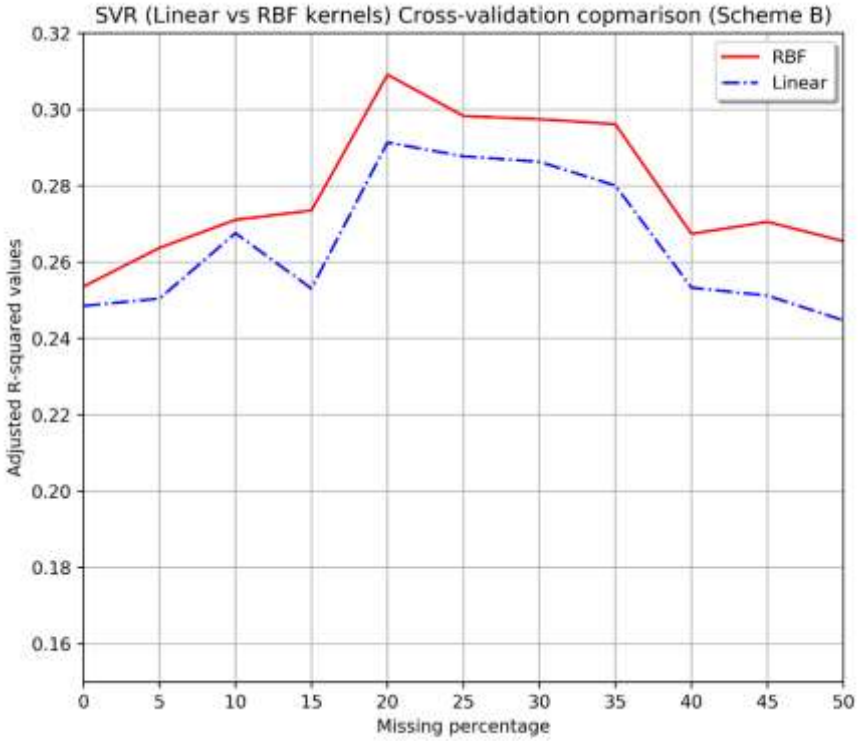



Figure 6.10 Cross-validation results from Linear and RBF kernels (Adjusted R-squared) on T-NN imputed datasets

*Cross-validation results performed on the training dataset*

As mentioned in earlier, the imputed datasets by all the selected approaches in this study are evaluated with 10 x 3 folds cross-validation. Table 6.10 shows the RMSE scores for all the approaches during the cross-validation. The results show that the proposed T-NN outperforms all the other imputation approaches with lowest RMSE scores on the datasets with different missing value percentages. Just as with the results in *scheme A*, the performance of the EM approach is the worst among the other imputation approaches. Overall, an increase in the RMSE error can be observed with the gradual addition of missing values as seen in Figure 6.11. The adjusted R-squared from Table 6.11 also shows that the proposed T-NN fits better compared to the other approaches consistently throughout the datasets used (although a lower proportion of variance is explained by the fits). Figure 6.12 shows a decrease in the adjusted R-squared scores with the increase in the number of missing values in the datasets.

Table 6.10. Cross-validation results with RMSE values using SVR on imputed datasets (scheme B)

| Imputers | Missing values percentage | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **T-NN** | 0.1342 | 0.1371 | 0.1397 | 0.1469 | 0.1488 | 0.1533 | 0.1600 | 0.1636 | 0.1704 | 0.1754 | 0.1801 |
| **K-Neighbors** | 0.1342 | 0.1373 | 0.1400 | 0.1474 | 0.1499 | 0.1543 | 0.1617 | 0.1655 | 0.1713 | 0.1758 | 0.1811 |
| **SVD** | 0.1342 | 0.1374 | 0.1395 | 0.1472 | 0.1502 | 0.1538 | 0.1594 | 0.1638 | 0.1705 | 0.1759 | 0.1815 |
| **SGD** | 0.1342 | 0.1373 | 0.1400 | 0.1479 | 0.1509 | 0.1555 | 0.1617 | 0.1658 | 0.1727 | 0.1776 | 0.1832 |
| **Median** | 0.1342 | 0.1376 | 0.1407 | 0.1494 | 0.1536 | 0.1578 | 0.1648 | 0.1691 | 0.1765 | 0.1822 | 0.1884 |
| **Mean** | 0.1342 | 0.1376 | 0.1407 | 0.1494 | 0.1535 | 0.1577 | 0.1648 | 0.1691 | 0.1767 | 0.1825 | 0.1887 |
| **Decision Tree** | 0.1342 | 0.1367 | 0.1397 | 0.1467 | 0.1540 | 0.1542 | 0.1612 | 0.1641 | 0.1737 | 0.1781 | 0.1888 |
| **EM** | 0.1342 | 0.1371 | 0.1403 | 0.1499 | 0.1544 | 0.1581 | 0.1670 | 0.1732 | 0.1814 | 0.1898 | 0.1927 |

Table 6.11. Cross-validation results with Adjusted R-squared values using SVR on imputed datasets (scheme B)

| Imputers | Missing values percentage | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **T-NN** | 0.2536 | 0.2638 | 0.2711 | 0.2735 | 0.3092 | 0.2983 | 0.2975 | 0.2962 | 0.2674 | 0.2706 | 0.2655 |
| **K-Neighbors** | 0.2536 | 0.2628 | 0.2687 | 0.2683 | 0.3004 | 0.2900 | 0.2819 | 0.2823 | 0.2638 | 0.2693 | 0.2601 |
| **SVD** | 0.2536 | 0.2612 | 0.2731 | 0.2702 | 0.2970 | 0.2932 | 0.3015 | 0.2979 | 0.2698 | 0.2688 | 0.2565 |
| **SGD** | 0.2536 | 0.2617 | 0.2686 | 0.2641 | 0.2909 | 0.2792 | 0.2818 | 0.2778 | 0.2480 | 0.2519 | 0.2412 |
| **Decision Tree** | 0.2536 | 0.2681 | 0.2710 | 0.2749 | 0.2631 | 0.2906 | 0.2881 | 0.2985 | 0.2571 | 0.2677 | 0.2167 |
| **Median** | 0.2536 | 0.2592 | 0.2614 | 0.2487 | 0.2658 | 0.2597 | 0.2552 | 0.2493 | 0.2147 | 0.2140 | 0.1969 |
| **Mean** | 0.2536 | 0.2583 | 0.2611 | 0.2485 | 0.2667 | 0.2603 | 0.2553 | 0.2488 | 0.2129 | 0.2116 | 0.1942 |
| **EM** | 0.2536 | 0.2639 | 0.2653 | 0.2461 | 0.2600 | 0.2596 | 0.2366 | 0.2197 | 0.1832 | 0.1543 | 0.1656 |

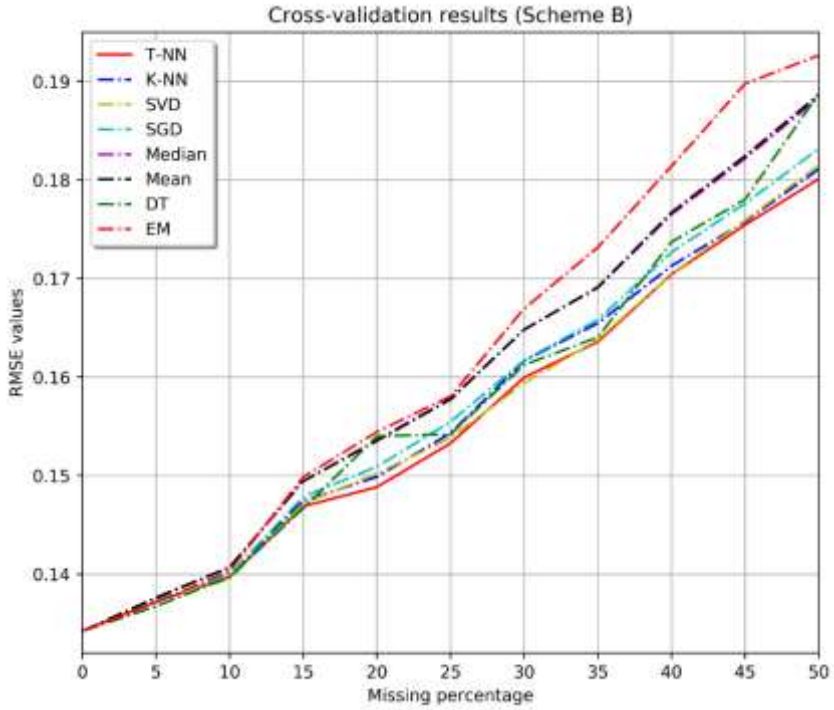



Figure 6.11 RMSE scores comparison on all imputed datasets with different missing percentages during cross-validation

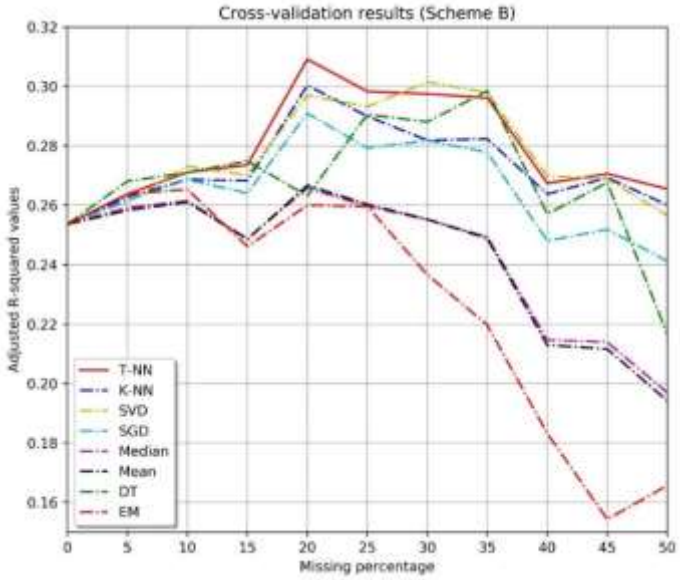



Figure 6.12 Adjusted R-squared scores comparison on all imputed datasets with different missing percentages during cross-validation

In *scheme B*, it must be noted here that, in each dataset the imputed values are estimates which means that these values have corresponding errors associated to them. Any predictions made using the

imputed datasets propagates these errors with them. The interpretation and adjustment of this additional error is not within the scope of this study.

Table 6.12 Test results with RMSE values using SVR on imputed datasets (scheme B)

| Imputers | Missing values percentage | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **T-NN** | 0.2783 | 0.2776 | 0.2759 | 0.2757 | 0.2753 | 0.2750 | 0.2737 | 0.2744 | 0.2730 | 0.2720 | 0.2717 |
| **SGD** | 0.2783 | 0.2783 | 0.2771 | 0.2761 | 0.2763 | 0.2755 | 0.2745 | 0.2757 | 0.2745 | 0.2731 | 0.2733 |
| **SVD** | 0.2783 | 0.2783 | 0.2767 | 0.2761 | 0.2762 | 0.2757 | 0.2746 | 0.2758 | 0.2757 | 0.2753 | 0.2768 |
| **K-Neighbors** | 0.2783 | 0.2783 | 0.2773 | 0.2759 | 0.2757 | 0.2755 | 0.2754 | 0.2755 | 0.2746 | 0.2728 | 0.2725 |
| **Decision Tree** | 0.2783 | 0.2772 | 0.2765 | 0.2762 | 0.2789 | 0.2770 | 0.2756 | 0.2776 | 0.2779 | 0.2765 | 0.2772 |
| **Mean** | 0.2783 | 0.2785 | 0.2773 | 0.2768 | 0.2774 | 0.2778 | 0.2768 | 0.2774 | 0.2769 | 0.2761 | 0.2763 |
| **Median** | 0.2783 | 0.2784 | 0.2771 | 0.2768 | 0.2775 | 0.2780 | 0.2770 | 0.2775 | 0.2766 | 0.2762 | 0.2765 |
| **EM** | 0.2783 | 0.2784 | 0.2771 | 0.2764 | 0.2773 | 0.2763 | 0.2773 | 0.2807 | 0.2804 | 0.2817 | 0.2798 |

Table 6.13 Test results with Adjusted R-squared using SVR on imputed datasets (scheme B)

| Imputers | Missing values percentage | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **T-NN** | 0.2913 | 0.2954 | 0.3043 | 0.3062 | 0.3086 | 0.3109 | 0.3177 | 0.3147 | 0.3226 | 0.3278 | 0.3301 |
| **K-Neighbors** | 0.2913 | 0.2920 | 0.2973 | 0.3049 | 0.3069 | 0.3085 | 0.3092 | 0.3095 | 0.3143 | 0.3239 | 0.3259 |
| **SGD** | 0.2913 | 0.2920 | 0.2983 | 0.3039 | 0.3036 | 0.3081 | 0.3137 | 0.3085 | 0.3150 | 0.3225 | 0.3221 |
| **Mean** | 0.2913 | 0.2910 | 0.2975 | 0.3007 | 0.2982 | 0.2965 | 0.3022 | 0.2998 | 0.3031 | 0.3077 | 0.3072 |
| **Median** | 0.2913 | 0.2915 | 0.2987 | 0.3006 | 0.2975 | 0.2958 | 0.3011 | 0.2993 | 0.3047 | 0.3073 | 0.3060 |
| **SVD** | 0.2913 | 0.2917 | 0.3007 | 0.3041 | 0.3043 | 0.3072 | 0.3136 | 0.3080 | 0.3091 | 0.3118 | 0.3048 |
| **Decision Tree** | 0.2913 | 0.2972 | 0.3016 | 0.3036 | 0.2902 | 0.3009 | 0.3082 | 0.2986 | 0.2978 | 0.3056 | 0.3027 |
| **EM** | 0.2913 | 0.2914 | 0.2985 | 0.3024 | 0.2986 | 0.3045 | 0.2996 | 0.2833 | 0.2854 | 0.2792 | 0.2895 |

*Validation on test dataset*

In *scheme B*, the approaches, once trained on the imputed datasets, are tested on a validation dataset. This validation step is like *scheme A*, in that the actual target values for which the predictions are made are known. The difference between these schemes as discussed earlier is that the models are trained on complete data in *scheme A*, whereas in *scheme B*, the models are trained on partially imputed datasets. The RMSE scores on the test dataset imputed by all the selected approaches is depicted in Table 6.12 and Figure 6.13. The SVR model trained on the datasets imputed by the proposed T-NN approach has lower RMSE score compared to the models trained on the imputed datasets by the other approaches. The RMSE scores from the datasets imputed by the EM approach is the highest. Similarly, the adjusted R-squared scores from Table 6.13 and Figure 6.14 show that SVR model fits better on the datasets imputed with T-NN compared to the datasets imputed by the other approaches (although a lower proportion of variance is explained by the fits).

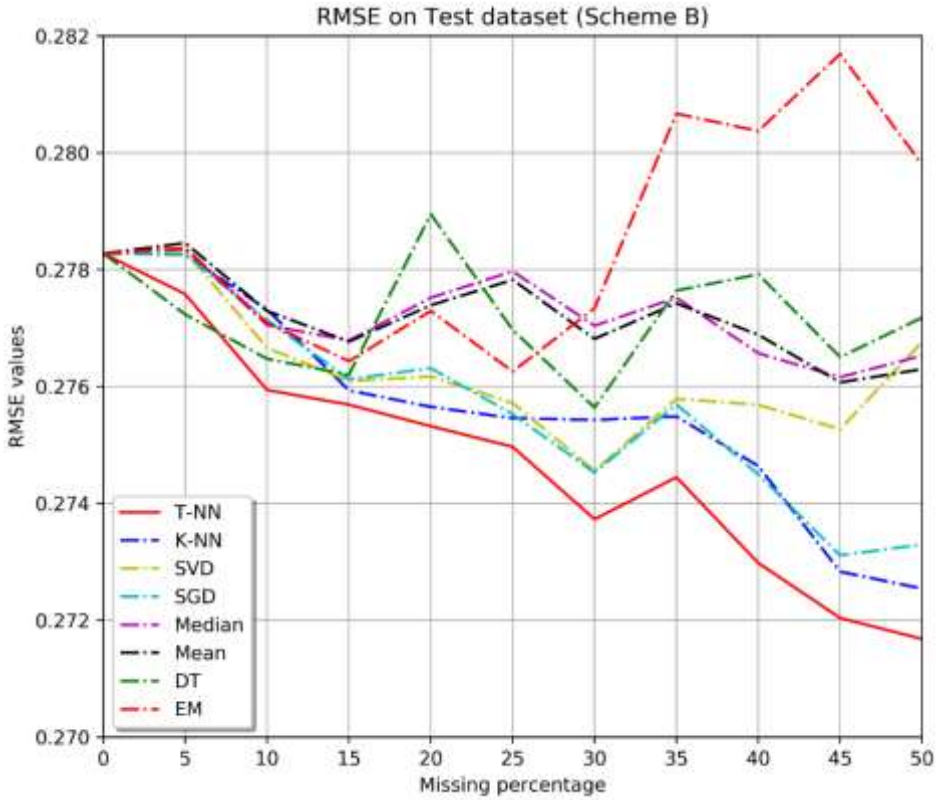



Figure 6.13 RMSE scores comparison on all imputed datasets with different missing percentages on test dataset

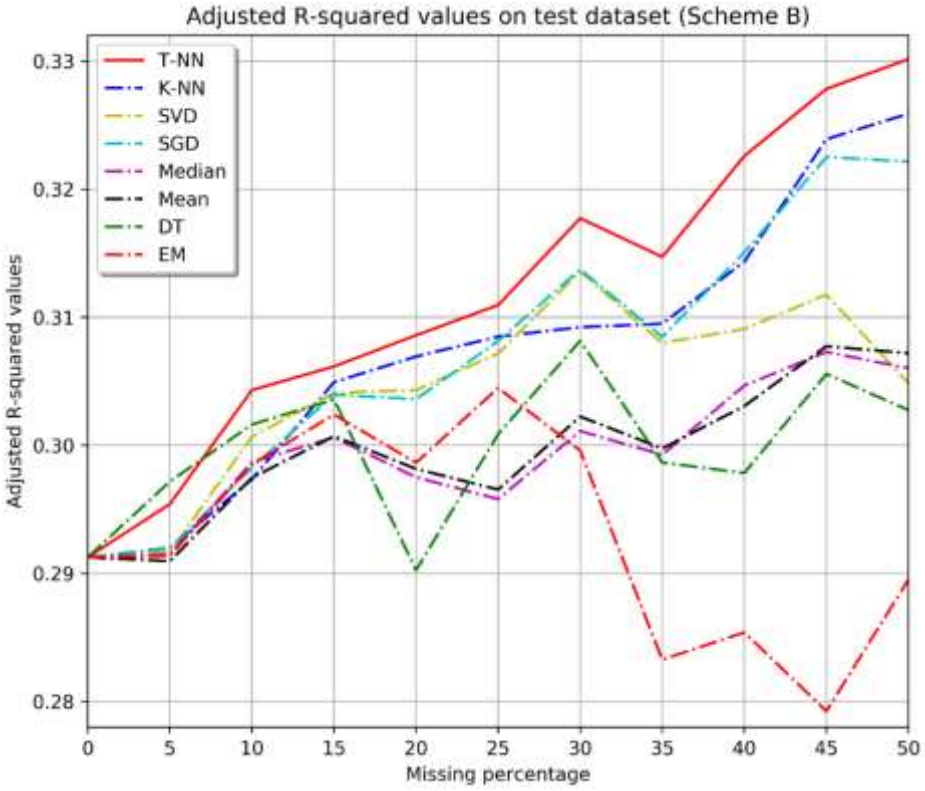



Figure 6.14 Adjusted R-squared scores comparison on all imputed datasets with different missing percentages on test dataset

The performance comparison of the SVR model with the baseline is depicted in Table 6.14. Table 6.14 shows the results from the experiments on datasets imputed with the T-NN approach. SVR performs better than the baseline with lower RMSE scores as shown in Figure 6.17 and higher adjusted R-squared scores in Figure 6.18.

Table 6.14. SVR vs Baseline performance with RMSE and Adjusted R-Squared

| | | **Estimators** | **Missing percentage (Scheme B)** | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **0** | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **40** | **45** | **50** |
| **RMSE** | CV | SVR | 0.1342 | 0.1372 | 0.1397 | 0.1469 | 0.1488 | 0.1533 | 0.1600 | 0.1636 | 0.1704 | 0.1755 | 0.1801 |
| | | Baseline | 0.1612 | 0.1645 | 0.1682 | 0.1780 | 0.1836 | 0.1883 | 0.1956 | 0.1986 | 0.2030 | 0.2088 | 0.2137 |
| | Test | SVR | 0.2783 | 0.2776 | 0.2759 | 0.2757 | 0.2753 | 0.2750 | 0.2737 | 0.2744 | 0.2730 | 0.2720 | 0.2717 |
| | | Baseline | 0.3375 | 0.3378 | 0.3377 | 0.3381 | 0.3382 | 0.3379 | 0.3374 | 0.3376 | 0.3382 | 0.3381 | 0.3376 |
| **Adjusted R-squared** | CV | SVR | 0.2536 | 0.2638 | 0.2711 | 0.2735 | 0.3092 | 0.2983 | 0.2975 | 0.2962 | 0.2674 | 0.2706 | 0.2655 |
| | | Baseline | -0.0531 | -0.0485 | -0.0522 | -0.0427 | -0.0429 | -0.0382 | -0.0375 | -0.0303 | -0.0351 | -0.0251 | -0.0294 |
| | Test | SVR | 0.2913 | 0.2954 | 0.3043 | 0.3062 | 0.3086 | 0.3109 | 0.3177 | 0.3147 | 0.3226 | 0.3278 | 0.3301 |
| | | Baseline | -0.0425 | -0.0436 | -0.0420 | -0.0433 | -0.0436 | -0.0408 | -0.0365 | -0.0373 | -0.0396 | -0.0380 | -0.0347 |

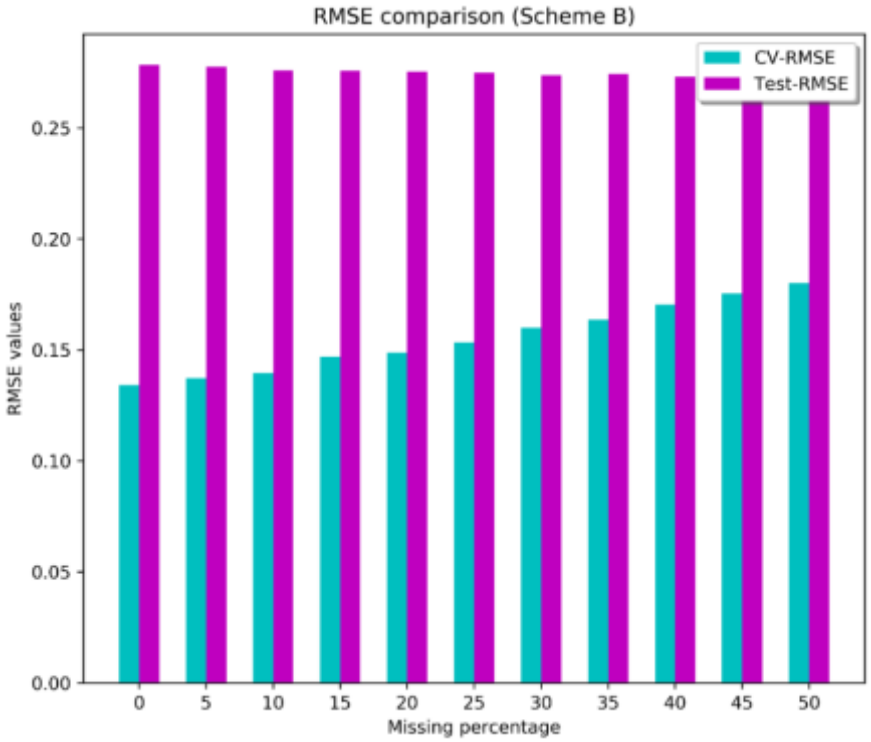



Figure 6.15 RMSE scores comparison on imputed datasets using T-NN

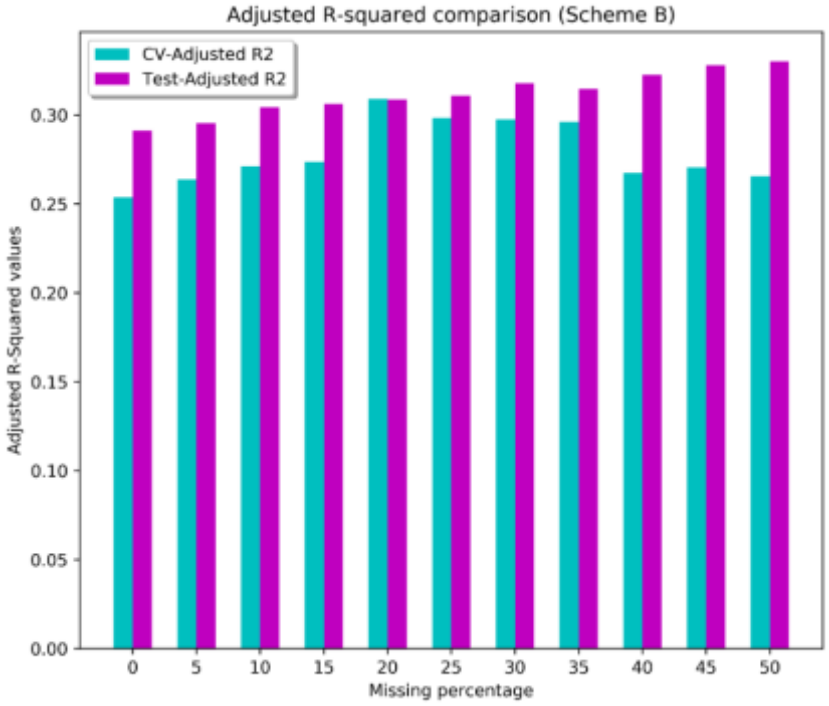



Figure 6.16 Adjusted R-squared scores comparison on imputed datasets using T-NN

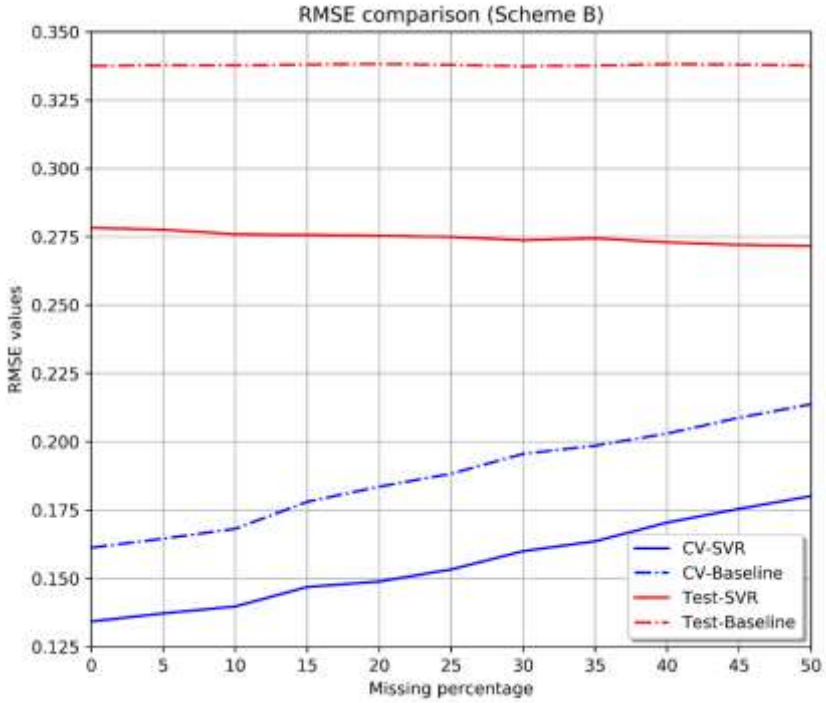



Figure 6.17 SVR and Baseline evaluation comparison on T-NN imputed and test datasets

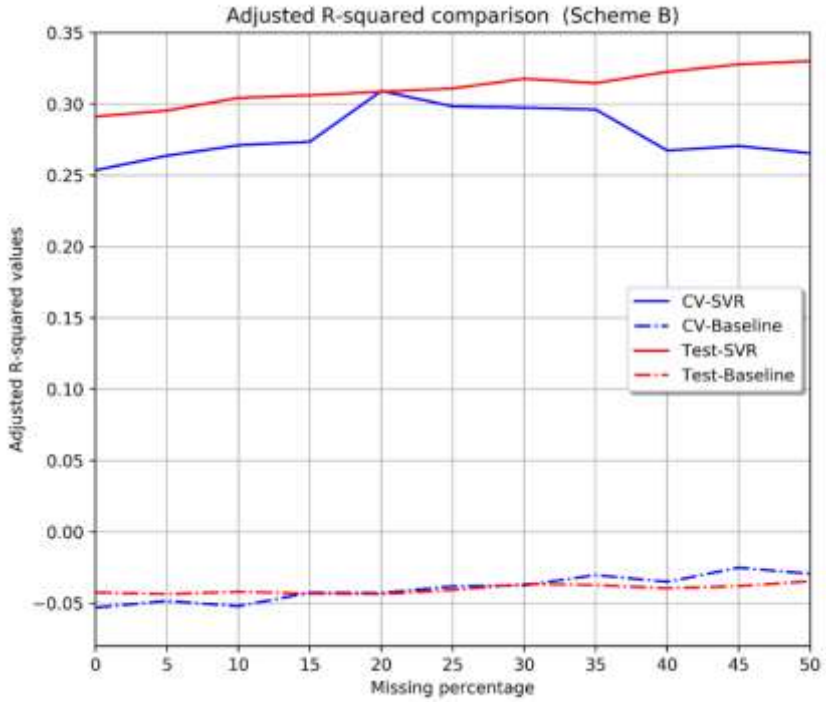



Figure 6.18 SVR and Baseline evaluation comparison on T-NN imputed and test datasets

A comparison of the performance of the SVR model from training and test results is depicted in Figure 6.15 and Figure 6.16. From the experiments it can be concluded that the proposed T-NN approach outperforms the other selected imputation approaches such as EM, SGD, SVD, DT, kNN

and univariate mean and median imputations. The evaluation using both the schemes shows lower RMSE scores by the T-NN and higher adjusted R-squared scores compared to the others. However, in all cases, a lower proportion of variance is explained by fitting the SVR on the datasets imputed by all the selected imputation approaches. This can be attributed to the nature of the dataset used in this study.

The RMSE scores in *scheme A* are relatively lower than the RMSE scores observed in *scheme B*. This difference between the RMSE scores is due to the fact that the datasets in *scheme B* contain more instances with missing values compared to the datasets in *scheme A*. Hence, based on the number of missing feature values and instances in the dataset, it is important to ascertain the value of *T* accordingly as it impacts on the performance accuracy.

## 6.5 IMPUTING SENTIMENT INTENSITY SCORE OF A SERVICE FACTOR: AN EXAMPLE CASE STUDY

A subset of the services from the dataset having 15% of the instances with missing values from *scheme A* is selected for this case study. The sample dataset is shown in Table 6.15 that contains SaaS products with random missing values in different service factors. The aim of this case study is to demonstrate how the imputation of a missing value in SaaS product *P586* for the service factor *R* is performed using the proposed T-NN imputation approach.

### 6.5.1 Finding donor instances

In the first step, the task is to find the potential donors that have sentiment intensity scores available in the service factor *R*. Table 6.15 shows a sample of the dataset where, rows have the service instance, columns show the service factors and the intersecting cell of the service factor with the instance show the sentiment intensity value for that instance in that service factor. Let us assume that service *P586* is the candidate instance whose sentiment intensity value in feature *R* needs to be imputed. The donor instances in this case are all service instances except *P247, and P949* which does not satisfy both the conditions for selecting a donor instance i.e., as *P247* does not have the value in the feature *R and P949* does not have values for the features that is required by the candidate instance.

$$X = \{P2, P586, P2843, P942, P949, P247, P114\} \tag{6.26}$$

For the candidate SaaS product *P586,* the set of donors using Equation 6.18 is given by:

$$D = \{P2, P2843, P942, P114\} \tag{6.27}$$

Table 6.15. Sample dataset with the sentiment intensity value SaaS service quality pillars (service factors)

| PID | A | C | E | L | M | N | O | P | R | S |
|---|---|---|---|---|---|---|---|---|---|---|
| **P2** | 0.62 | 0.36 | 0.01 | -0.08 | 0.25 | 0.06 | 0.17 | 0.00 | 0.24 | 0.06 |
| **P586** | 0.62 | 0.60 | 0.54 | 0.63 | 0.54 | 0.47 | 0.41 | 0.72 | N/A | N/A |
| **P2843** | 0.49 | 0.63 | 0.16 | 0.24 | 0.54 | 0.45 | 0.62 | 0.71 | 0.79 | 0.02 |
| **P942** | 0.18 | 0.00 | 0.10 | 0.46 | 0.17 | 0.19 | -0.54 | 0.02 | 0.00 | 0.02 |
| **P949** | N/A | 0.51 | N/A | N/A | N/A | N/A | N/A | N/A | 0.42 | 0.07 |
| **P247** | N/A | 0.39 | N/A | N/A | 0.45 | 0.31 | 0.53 | 0.40 | N/A | N/A |
| **P114** | 0.16 | 0.34 | 0.40 | 0.57 | 0.41 | 0.24 | 0.67 | 0.66 | 0.57 | 0.80 |

### 6.5.2 Calculating Pearson's correlation coefficient

As we are only demonstrating few samples for the dataset in this case study, the Pearson's correlation coefficient must be calculated for the entire dataset. Table 6.16 shows the correlation coefficients among the features calculated using Equation 6.12 from the base dataset that has no missing values. We use the PCC values from Table 6.16 in this case study. As we intend to impute a missing value that belong to service factor *R,* therefore, we need the PCC values between the feature *R* and other features. These coefficients are used to calculate the weights for each feature when determining the distance between the features of the candidate and the donor instances in the next step. Table 6.15 shows the PCC values between *R* and others. It also shows the PCC value of *R* with itself is 1 which indicates a full positive correlation.

Table 6.16. Correlation values among the features

| | A | C | E | L | M | N | O | P | R | S |
|---|---|---|---|---|---|---|---|---|---|---|
| **R** | 0.2095 | 0.0681 | 0.2556 | 0.0818 | 0.3652 | 0.3635 | 0.2749 | 0.2547 | 1.0000 | 0.2829 |

### 6.5.3 Calculating weighted distance

After the correlation between feature *R* and other features are calculated in the previous step, these values are used to calculate weights for each feature using Equation 6.17. Table 6.17 shows the weights assigned to each feature.

Table 6.17. Service factors with correlation values and weights

| | A | C | E | L | M | N | O | P | R | S |
|---|---|---|---|---|---|---|---|---|---|---|
| **R** | 0.2095 | 0.0681 | 0.2556 | 0.0818 | 0.3652 | 0.3635 | 0.2749 | 0.2547 | 1.0000 | 0.2829 |
| **Weights** | 0.7905 | 0.9319 | 0.7444 | 0.9182 | 0.6348 | 0.6365 | 0.7251 | 0.7453 | 0 | 0.7171 |

The distances between the candidate instance *P586* and the donor instance *P2* is calculated using Equation 6.16 as:

$$d(P586, P2) = \sqrt{0.79 * (0.62 - 0.62)^2 + 0.93 * (0.60 - 0.36)^2 + 0.74 * (0.54 - 0.01)^2 + 0.92 * (0.63 - 0.08)^2 + 0.63 * (0.54 - 0.25)^2 + 0.64 * (0.47 - 0.06)^2 + 0.73 * (0.41 - 0.17)^2 + 0.75 * (0.72 - 0.0)^2}$$

$$d(P586, P2) = \sqrt{0 + 0.05 + 0.21 + 0.28 + 0.05 + 0.11 + 0.04 + 0.39}$$

$$d(P586, P2) = \sqrt{1.13}$$
$$d(P586, P2) = 1.06$$

Similarly, the distances between the candidate instance *P586* and the other donor instances is calculated and are shown in Table 6.18.

Table 6.18. Forming a subset D¯ from the donor instances

| PID | A | C | E | L | M | N | O | P | R | S | $d(x_i, x_j)$ | Distance from the closest donor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **P2** | 0.62 | 0.36 | 0.01 | -0.08 | 0.25 | 0.06 | 0.17 | 0.00 | 0.24 | 0.06 | 1.06 | 0.52 |
| **P586** | 0.62 | 0.60 | 0.54 | 0.63 | 0.54 | 0.47 | 0.41 | 0.72 | N/A | N/A | - | - |
| **P2843** | 0.49 | 0.63 | 0.16 | 0.24 | 0.54 | 0.45 | 0.62 | 0.71 | 0.79 | 0.02 | ***0.54*** | ***0*** |
| **P942** | 0.18 | 0.00 | 0.10 | 0.46 | 0.17 | 0.19 | -0.54 | 0.02 | 0.00 | 0.02 | 1.35 | 0.81 |
| ~~**P949**~~ | ~~N/A~~ | ~~0.51~~ | ~~N/A~~ | ~~N/A~~ | ~~N/A~~ | ~~N/A~~ | ~~N/A~~ | ~~N/A~~ | ~~0.42~~ | ~~0.07~~ | - | - |
| ~~**P247**~~ | ~~N/A~~ | ~~0.39~~ | ~~N/A~~ | ~~N/A~~ | ~~0.45~~ | ~~0.31~~ | ~~0.53~~ | ~~0.40~~ | ~~N/A~~ | ~~N/A~~ | - | - |
| **P114** | 0.16 | 0.34 | 0.40 | 0.57 | 0.41 | 0.24 | 0.67 | 0.66 | 0.57 | 0.80 | 0.58 | 0.04 |

As shown in Table 6.18, *P2843* is closest donor to the candidate with a weighted distance of *0.54*.

***Selecting final donors and imputing the missing value in candidate***

The threshold value is used to select the final donors to impute the missing value in service factor (feature) *R* of the candidate *P586*. For this case study I am using the threshold value calculated for *scheme A* i.e., *T=0.22*.

The threshold value defines the closeness among the selected donor instances. Table 6.18 in the column $d(x_i, x_j)$, shows the distance between candidate instance *P586* and the other donor instances. Donor instance *P2843* is the most relevant donor with the closest distance to the candidate instance. The last column in Table 6.18 computes the distance between *P2843* and the other donor instances. Using the last column, considering the value of *T=0.22*, then only *P2843* and *P114* from the donor instances is used to impute the value of feature *R* as both fall within the threshold value.

Finally, the missing value in service factor *R* of *P586* is imputed using Equation 6.19 as:

$$P2843_R = \frac{0.79 + 0.57}{2} = 0.68$$

here, *n=2* as only two instances are within the threshold range.

The complete steps of the imputation process using T-NN is shown in Algorithm 6.1.

**Algorithm** 6.**1: TNN-Impute (*X*)**

**Input:** Dataset *X* with random missing values
**Output:** Imputed dataset
Let *V* be a set of feature variables, $V = \{v_1, v_2, \ldots, v_n\}$
Let *T* be the selected threshold value
Calculate Pearson's correlation coefficient *PCC[X]* among all variables in
Find all missing values indexes, *MI[X]*
for each $i \in MI[]:$ do
    If donor instances from X contain the value for missing for *candidateInstance[v]*
        Select donor instance, *donorIndexes[] = i*
end for
for each $d \in donorIndexes[]$: do
    Use weights relative to *MI[v]* and other variables from *PCC[]*
    Calculate weighted Euclidean distance instances *d* using only the variables
without *NaN*
end for
Sort the distances of the selected donor instances
for each $d \in donors[]$: do
    if *d[distance] – d[with least Distance to candidate] <= T*
        Select donor instance
    end if
Calculate the aggregated value from the selected variable *i* of donor instances *d[]*
Assign the aggregated value to the variable *i* of the candidate instance

## 6.6 CONCLUSION

In this chapter, I discussed the importance of the imputation techniques for missing values in datasets. I discussed the most commonly used imputation methods, their strengths and weaknesses. I presented the details of a novel approach for the solution of the missing sentiment intensity scores in the service factor of SaaS. The proposed approach called the T-NN which stands for Threshold-based nearest neighbours, is an extension of the famous non-parametric method k-nearest neighbours. One of the main contributions of T-NN is that, it uses the weighted distance among the candidate instance and donors. The weights for each service factor are calculated using the Pearson's correlation coefficient among features. The relative service factors to the factors with missing value gets higher weights when they are highly correlated and vice versa. The second contribution of the T-NN approach is using a threshold value to select the final donors for imputation. Using the threshold value helps to reduce the error between the actual and imputed sentient intensity scores.

I have used two different implementation schemes to validate the performance of T-NN approach. In the first scheme called scheme A, 10 datasets with different number of missing values is generated from a base dataset with complete observations. The imputed values by the T-NN is validated against the actual observations and error is calculated. The validation results by T-NN is scheme A is compared with the imputation results by the other implemented methods such as Traditional k-NN, SVD, SGD, Decision trees, EM, Mean and median. In the second scheme called scheme B, 10 datasets are generated with different among of missing values. The main difference between the datasets used in scheme A and scheme B is the total number of instances in each dataset. In scheme A, the total number of instances in each dataset remains the same while the number of factors with missing values are increased incrementally by 5 in each generated dataset. Whereas, in scheme B, the base dataset with complete observations is part of all the generated datasets while in each dataset, extra instances with missing values are added incrementally by 5. These extra instances are also true observations having missing values in random features. The goal of scheme B is validating the performance of the predictive methods on the datasets which are imputed by T-NN. I have implemented the Support Vector for Regression (SVR) method to for this purpose using the datasets imputed by T-NN and other methods implemented in this part of my studies.

The results from the experiments shows that the proposed T-NN approach outperforms the other famous approaches implemented and discussed in this chapter. Furthermore, the implementation of T-NN and the resulting imputed datasets are crucial when forecasting the future values for these service factors as the missing values in any factor create a bottleneck for the forecasting model which is discussed in the next chapter.

# Chapter 7:
# A methodology for time series Forecasting of SaaS service factors

## 7.1 INTRODUCTION

In the previous chapter, I discussed the importance of sentiment intensity scores for each of the identified service factors of SaaS products and the potential hinderance faced by the trust model in computing the trust scores for the SaaS products due to the missing sentiment intensity scores for any of the service factors. I presented a method for imputing such missing sentiment intensity scores along with the implementation details. The next component of the trust management framework, discussed briefly in chapter 4, is the ability of the trust model to accurately forecast the trust scores of SaaS products for a point in time in the future. This is particularly important when a SaaS service is intended to be acquired for a particular duration of time in the future and an expected trust level is required by the signed service level agreement (SLA). As a precursor to the trust assessment model discussed and implemented in next chapter, the trust model can only be capable to compute and infer the trust score of SaaS products at a future point in time by having the capability to forecast each trust factor for the future time spots. The proposed methodology presented in this chapter brings the forecasting capability into the trust model for all the considered trust factors.

The trust factors, considered as the service factors of the SaaS products in this thesis, is forecasted for future time slots based on the historical observations of each service factor at different timeslots in the past. The observations for each service factor are in the form of timeseries and the forecasting of the values for these services factors is framed as a time series problem in this chapter. Time series analysis and forecasting techniques is well known in literature and are extensively applied in different problem domains with variety of data in nature. However, as discussed in the previous chapters, time series analysis and forecasting techniques has not been investigated for sentiment intensity scores of the SaaS service factors. As part of the methodology proposed in this chapter, I investigate and implement different time series forecasting techniques ranging from classical approaches to some of the latest and most popular ones. The techniques are investigated using the sentiment intensity dataset for SaaS service factors and the results are compared to identify the most suitable method that is used for the forecasting component of the proposed trust model.

The rest of the chapter is organized as follows. In Section 7.2, I present a background on time series and the related concepts. In Section 7.3, I discuss the timeseries forecasting techniques investigated in this study. I discuss the in details, each stage of the proposed forecasting methodology in Section 7.4. I present the details of the experiments and evaluation results of the selected timeseries forecasting techniques in Section 7.5. Section 7.6 concludes the investigation in this chapter.

## 7.2 BACKGROUND

In this section I will briefly explain the key concepts related to time series data modelling and forecasting.

### 7.2.1 Time series

The data that is collected sequentially over fixed intervals is called time series (Box, Jenkins, & Reinsel, 2008). Generally, time series is the numerical result of observations of an object or multiple objects over certain period of time. Throughout this chapter, I write $Y_n$ as a variable representing a series of length $n$ for one of the SaaS service factors observed at times $t = 1, 2, \ldots n$ and an observation at time $t$ is written as $y_t$. Mathematically the series can be written as:

$$Y_n = \{y_t | t = 1, \ldots n\} \quad (7.1)$$

### 7.2.2 Regular and irregular series

A typical time series data is regular in nature. Regular time series data are evenly spaced sequence of observations having constant intervals whereas in many cases, a series of data can be observed at irregular time stamps and the observations are unevenly spaced without constant intervals. Examples of regular time series are daily temperature observations and hourly monitoring signals in industrial setups. On the other hand, irregular series are observations such as earthquakes and aftershocks, process triggers in internet of things (IoT) setups and user feedbacks and ratings for online systems. Regular time series values are observed at predefined settings which can be spaced naturally or setup artificially at fixed intervals. Whereas, irregular series are likely to be event-driven observations with irregular timestamps. Most of the time series modelling and forecasting techniques work better with the time series data observed at constant intervals. Series of observations at irregular time stamps may affect the performance of the data modelling and forecasting. An irregular series can be transformed into a regular time series using techniques such as interpolation (Bayen & Siauw, 2015) and Last Observation Carried Forward (LOCF) (Kenward & Molenberghs, 2009).

### 7.2.3 Time series components

A time series data can portray different behaviours such increasing or decreasing trends, seasonality or simply a random without any pattern. A time series can be decomposed into three components namely: trend, seasonal and irregular. The seasonality in a time series data is the periodic repetition of a particular pattern that is systematic and follow a calendar related movement such as seasons of the year. The trend component in a time series is the long-term movement in the form of increase or decrease of observed values such as the growth in world population. The random or residual component results from short term fluctuations that do not follow any pattern and they are neither predictable nor systematic. Figure 7.1 shows the decomposition of a time series into the components.

A time series can be decomposed into the constituent components using two decomposition models namely: the additive and the multiplicative decomposition model. The additive decomposition model is suitable for time series in which, the seasonal and random variation do not change along the trend. In simple words, it assumes that the series is the sum of its constituent components.

For an observed time series *Y*, the additive model assumes *Y* to be the sum of the seasonal component *S*, the trend component *T* and random component *R*, which can be written as:

$$Y_t = S_t + T_t + R_t \tag{7.2}$$

The multiplicative decomposition model is suitable for time series in which, the seasonal and random variations increase along the trend. The multiplicative decomposition assumes that the observed series is the product of its constituent components and can be written as:

$$Y_t = S_t \times T_t \times R_t \tag{7.3}$$

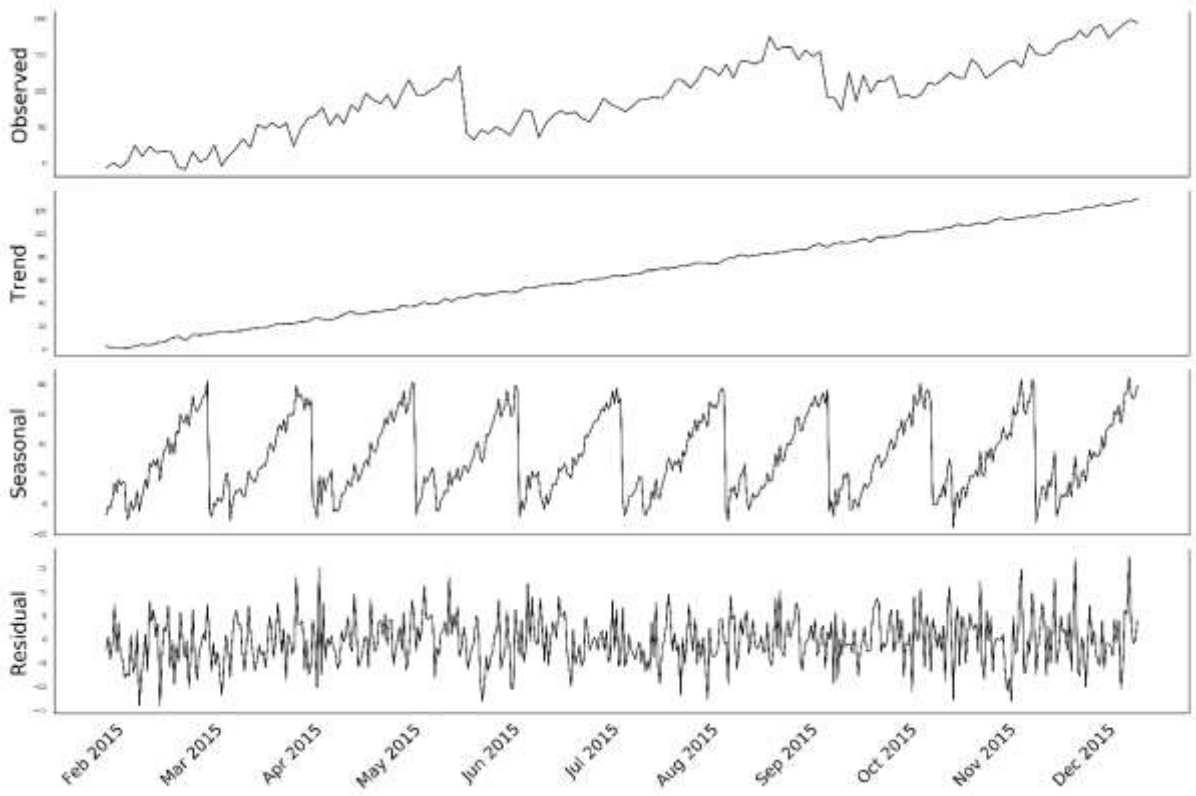



Figure 7.1 Components of time series

### 7.2.4 Stationary and non-stationary series

Stationarity is a characteristic of time series data. Stationarity of a time series is observed through its statistical properties such as mean, variance and covariance. The mean and variance of stationary time series remains constant over time and its covariance remains independent of time. Ideally, time series modelling and forecasting techniques works better when the time series data is stationary. Which means that the series must be adjusted for its constituent component discussed in the previous sections.

As discussed in the previous sections, time series can be decomposed into its constituent components using additive or multiplicative decomposition. After the decomposition of a series, a line plot for each of the constituent can help to visually observe their presence. In many cases the components of a time series are not visually observable and using unit root tests can help to identify the non-stationarity of the series and the components of time series can be adjusted accordingly. One of the well-known statistical tests for testing time series stationarity is called the augmented Dicky-Fuller test (Dickey & Fuller, 1979) also known as unit root testing.

A typical seasonally adjusted series can be achieved by removing the estimated seasonal component from the series. For an additive model it can be expressed as:

$$YS_t = X_t - \hat{S}_t = T_t + R_t \tag{7.4}$$

Where, $YS_t$ is the seasonally adjusted series and $\hat{S}_t$ represents the seasonally adjusted estimates.

Similarly, for a multiplicative model, the seasonal adjustment can be expressed as:

$$YS_t = Y_t \div \hat{S}_t = T_t \times R_t \tag{7.5}$$

### 7.2.5 Univariate and multivariate time series

Time series data can be univariate that is collected for a single numerical variable over a period of time or it can be multivariate or vector time series. A multivariate time series has numerical observations for multiple variables where instances in each variable share the same time stamps. In some cases, the vector time series can provide more insight into events at specific time stamps. Working with multivariate time series provides better forecasting results as it is capable to capture and describe any dynamic behaviour of the time series data.

### 7.2.6 Time series forecasting

Using time series data, a model can be created that, based on the past behaviours in the time series, can forecast the future values in continuation to the existing series. The forecasted value at time *t* for the next timestep *t+1* is represented as $\hat{y}_{t+1}$ and is called single step forecasting. Time series forecasting can be performed for multiple time steps ahead in the future $\hat{y}_{t+n}$ where n is the number of timesteps in the future. Some of the well-known timeseries forecasting methods are explained briefly in the next section.

### 7.2.7 Correlations

Correlation defines the dependency between entities. In the context of time series, correlation can be of two types namely: serial correlation and cross correlation.

Serial correlation also known as autocorrelation, measures the relationship between observations at different point in time within the same series. Whereas the cross correlation is the relationship between the observed values in two separate series. Capturing autocorrelations is useful when modelling a series with AR to determine the number of lags.

Autocorrelation provides useful insight into the time series data and is helpful in time series modelling and forecasting. Cross correlation is used when time series forecast models are build using multivariate time series data. Some of the timeseries forecasting models discussed in the next section are capable to work with both univariate and multivariate time series data.

### 7.2.8 Interpolation

Interpolation is an estimation process for creating new data points between two known ones. Interpolation is useful in many problems where time series is involved such as, imputing for missing values in a series and transforming an irregularly observed series with time stamps into a regular time series.

## 7.3 TIMESERIES FORECASTING APPROACHES

Time series modelling and forecasting is an extensively research area. There are a number of techniques to model time series for forecasting that range from traditional approaches such as moving average, auto regression and exponential smoothing, to advanced methods such as artificial neural networks and deep neural networks. I briefly describe the most well-known and commonly used time series modelling and forecasting techniques in the following subsection.

### 7.3.1 Autoregressive model (AR)

The Autoregressive model use the observations in a series at prior time steps to regress for a value at current time spot or to forecast the value in the next step using a linear function.

The autoregressive model takes a single parameter *p* for the linear function. The parameter *p* defines the order of regression also known as order-p which represents the number of observations from the previous time steps to be considered for regression. An AR(p) model can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \quad (7.6)$$

Where $y_t$is the value for the time spot *t*, *c* is the intercept or constant term, $\phi$ is coefficient of lag of y, *p* is the number of preceding observations used and $\epsilon_t$ is the error term.

The AR models are suitable for stationary series without any seasonal and trend components. The AR model is also used to forecast univariate time series.

$$y_{t+1} = c + \phi_1 y_t + \phi_2 y_{t-1} + \cdots + \phi_p y_{t-p} + \epsilon_t \quad (7.7)$$

where $y_{t+1}$ is the forecasted value for the next time step *t+1*.

A vector version of AR model called the Vector Autoregression (VAR) is used to model multivariate time series (Ltkepohl, 2007). Similar to AR model, the VAR method takes the order of AR(p) as parameter to the linear function and is suitable for timeseries series without trend and season components.

In VAR model, each variable is a linear function of past values of itself and the past values of the other variables.

A VAR model of two variables with lag order 1 VAR(1) can be written as:

$$y_{t,1} = \mathrm{c}_1 + \phi_{11} y_{t-1,1} + \phi_{12} y_{t-1,2} + \epsilon_{t,1} \quad (7.8)$$

$$y_{t,2} = \mathrm{c}_2 + \phi_{21} y_{t-1,1} + \phi_{22} y_{t-1,2} + \epsilon_{t,2} \quad (7.9)$$

An *n* variable, *p-lag* VAR model can be written as:

$$Y_t = \mathrm{c} + \Pi_1 Y_{t-1} + \Pi_2 Y_{t-2} + \cdots + \Pi_p Y_{t-p} + \epsilon_t \quad (7.10)$$

where $Y_t$is a vector of time series variables $Y_t = (y_{t,1}, y_{t,2}, \ldots y_{t,n})$, $\mathrm{c} = (\mathrm{c}_1, \mathrm{c}_2, \ldots \mathrm{c}_n)$, $\Pi_j$represents AR coefficient matrices with *(n x n)* dimensions where, j= (1, …, p) and $\epsilon_t$ is the error matrix of (n x 1) dimensions. VAR is suitable for multivariate stationary time series in which variables influence each other.

### 7.3.2 Moving Average model (MA)

Moving average is the simplest technique to model time series. As the name suggest, it calculates the next observation in a given series using the average of the past observations. In a given series, the MA models the next step as a linear function of the residual errors from mean at the previous steps. The number of past observations also known as the order $q$, can be adjusted to get the best forecast results using the window method. Generally, a larger window size produces a smoother forecast. The first order moving average can be written as:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} \quad (7.11)$$

where $\varepsilon_t$is the white noise and $\theta$is the moving average term. Similarly, the $q^{th}$ order moving average model *MA(q)* is written as:

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \quad (7.12)$$

The moving average model is also suitable for stationary series without seasonal and trend component and is used to model and forecast univariate time series.

### 7.3.3 Autoregressive Moving Average model (ARMA)

The Autoregressive Moving Average (ARMA) method combines both the Autoregressive (AR) and Moving Average (MA) models. It models the next step in a series as a linear function of both the observations and residual errors at the previous time steps. The ARMA model takes two parameters, the order of AR(p) and MA(q) and is written as ARMA(p, q).

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \quad (7.13)$$

The ARMA model is suitable for series without trend and seasonal components. It is used for modelling and forecasting univariate time series. However, a generalized vector version of ARMA called the Vector Autoregression Moving-Average (VARMA) can model and forecast for the next time step of each series in a multivariate series using the ARMA model. The VARMA models take as parameter the order of AR(p) and MA(q) for the function VARMA(p, q). Similar to ARMA model, the VARMA model is also suitable for stationary time series without trend and seasonal components. A vector autoregressive model of order 1 for three variable series is written as:

$$Y_t = \mathrm{c} + \Pi_1 Y_{t-1} + \Pi_1 Y_{t-2} + \cdots + \Pi_p Y_{t-p} + \varepsilon_t + \Theta_1 \varepsilon_{t-1} + \Theta_2 \varepsilon_{t-2} + \cdots + \Theta_q \varepsilon_{t-q} \quad (7.14)$$

where $Y_t$is a vector of time series variables $Y_t = (y_{t,1}, y_{t,2}, \ldots y_{t,n})$, $\mathrm{c} = (\mathrm{c}_1, \mathrm{c}_2, \ldots \mathrm{c}_n)$, $\Pi_j$represents AR coefficient matrices with *(n x n)* dimensions where, j=(1, …, p), $\Theta_k$represents MA coefficient matrices with *(n x n)* dimensions where, k=(1, …, q) and $\epsilon_t$ is the error matrix of (n x 1) dimensions. VAR is suitable for multivariate stationary time series in which variables influence each other.

A one step ahead forecast with the VARMA model is given by:

$$\hat{Y}_{t+1} = \mathrm{c} + \Pi_1 Y_t + \Pi_1 Y_{t-1} + \cdots + \Pi_p Y_{t-p} + \varepsilon_t + \Theta_1 \varepsilon_t + \Theta_2 \varepsilon_{t-1} + \cdots + \Theta_q \varepsilon_{t-q} \quad (7.15)$$

### 7.3.4 Auto Regressive Integrated Moving Average (ARIMA)

The ARIMA approach explains a time series by capturing the autocorrelation in its past observations. Similar to the ARMA model explained in the previous subsection, ARIMA model combines the AR and MA models in modelling and forecasting a time series. The main difference between the ARMA model an ARIMA model is that ARMA models work with stationary time series. The seasonal and trend components are adjusted using differencing then used with ARMA models. Whereas in ARIMA model, the differencing process is integrated in the model. An ARIMA model is denoted as ARIMA(p, d, q) where the term p represents the order of AR, d represents number of differencing and q represents the order of MA. An ARIMA model with particular order in its terms are similar to

the other models such as, an ARIMA(p,0,0) model is the same as the Autoregressive AR(p) model, an ARIMA(0,0,q) model equals a Moving Average MA(q) model and an ARIMA(p,0,q) is equivalent to an ARMA model. An ARIMA model with first order differenced series can be written as:

$$y'_t = c + \phi_1 y'_{t-1} + \phi_2 y'_{t-2} + \cdots + \phi_p y'_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \quad (7.16)$$

where $y'_t$ is the first order differenced series. For a one step ahead forecast, the ARIMA model can be written as:

$$\hat{y}_{t+1} = c + \phi_1 y'_t + \phi_2 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \varepsilon_t + \theta_1 \varepsilon_t + \theta_2 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \quad (7.17)$$

### 7.3.5 Exponential Smoothing

Exponential Smoothing (ES) is similar to the moving average modelling technique with an added weight factor (Gardner Jr., 1985). In exponential smoothing, a decreasing weight is assigned to the past observations as it moves further from the current observations.

A Simple Exponential Smoothing (SES) method can be written mathematically as:

$$y_t = \alpha y_t + (1 - \alpha) y_{t-1}, t > 0 \quad (7.18)$$

Where, $\alpha$ is a smoothing factor that determines the rate of decrease in weights for the previous observations. The values for the smoothing factor $\alpha$ are taken between 0 and 1. Generally, lower values of $\alpha$ produces a smoother forecast.

The Simple exponential smoothing method suitable for series without the trend and seasonal component and can only work with univariate time series. A one step ahead forecast using the SES is given by:

$$\hat{y}_{t+1} = \alpha y_t + (1 - \alpha) \hat{y}_t \quad (7.19)$$

The $y_t$ is selected as the initial level for the SES.

To support the trend component in a time series, the Double Exponential Smoothing method is used instead of SES. The Double Exponential Smoothing is also called the Holt's linear trend model (Holt, 2004) named after the original developer of the model, deals with the additive or linear trend component of the series.

Besides the $\alpha$ smoothing parameter, the Holt's linear trend method takes another parameter β which controls the influence of the trend component in a series. A one step ahead forecast equation for Holt's linear model is given by:

$$\hat{y}_{t+1} = l_t + b_t \tag{7.20}$$

where

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{7.21}$$

and

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{7.22}$$

The level and trend components at time t is represented as $l_t$ and $b_t$ respectively.

The extension of Exponential Smoothing with added support for seasonality in a time series is called the triple exponential smoothing or Holt-Winters Exponential Smoothing (HWES) (Holt, 2004) (Winters, 1976). The new parameter $\gamma$ that controls the seasonal component $s_t$ in a time series. It models the next time step of a series as an exponentially weighted linear function of the observed values at previous time steps. Similar to Simple Exponential Smoothing, the HWES can work with univariate time series only. Like the other SES models, the HWES model also have additive and multiplicative models. The equation for additive Holt-Winters Exponential Smoothing model is given by:

$$\hat{y}_t = l_t + b_{t-1} + s_{t-p} \tag{7.23}$$

where

$$l_t = \alpha\big(y_t + s_{t-p}\big) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{7.24}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{7.25}$$

and

$$s_t = \gamma(y_t - l_t) + (1-\gamma)s_{t-p} \tag{7.26}$$

Both the additive and multiplicative ES models, a damping parameter is used to control the constant growth of the trend over time. The damping coefficient is represented by $\phi$.

### 7.3.6 Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP) is an Artificial Neural Network (ANN) with multiple layers of neurons (Murtagh, 1991). The core component of artificial neural networks is the artificial neuron. A neuron also called a node, is a simple computational unit that takes some inputs, process the information and outputs the results. All the inputs to the neuron are weighted. Similar to the weights, each neuron also has bias values. The neuron sums up these weighted inputs and maps them to the output using an activation function given by:

$$Y = f(X) \tag{7.27}$$

where $X$ and $Y$ are the input and output vectors respectively. The function $f$ is optimized by adjusting the weights during a training process till an optimal mapping from $X$ to $Y$ is reached.

The most common activation functions are:

- sigmoid function: a non-linear function that outputs values between 0 and 1
- tanh function: a hyperbolic tangent function which is also non-linear in nature and outputs values between -1 and 1.

As the name suggest, neurons are organized into networks at different layers. The three types of network layers are:

- Input layer: where the input values are received by the network and passes these values onto the other layers of the network
- Hidden layer: the hidden neurons of the network which receives the values from the input layer
- Output layer: responsible for the outputting the computed values by the network

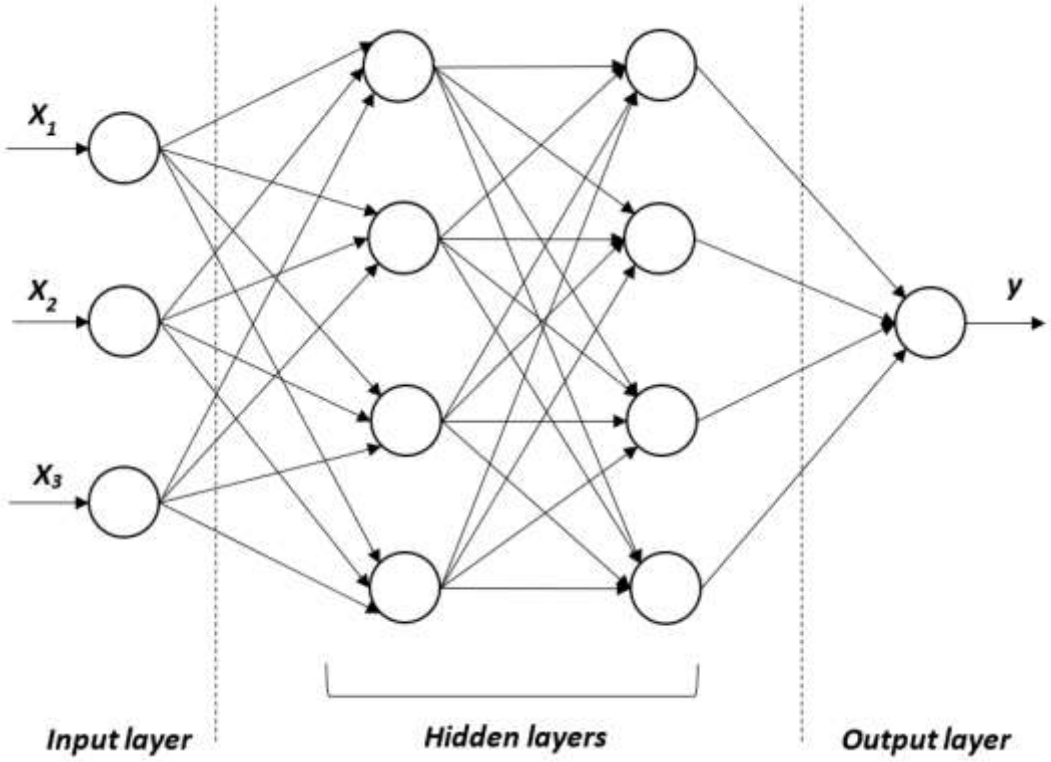



Figure 7.2 A multilayer perceptron with two hidden layers

Figure 7.2 depicts an MLP with two hidden layers. The training process of a neural network includes passing forward all the data samples through the network, from input to the output layer. Because of this forward directed process the MLP is also called feed-forward network. A complete pass of the entire dataset through the network is called an epoch. A neural network is normally trained over many epochs and the network weights are updated during this process either for prediction errors after each sample or at the end of an epoch. The training continues till a desired mapping from network inputs to the outputs is achieved. As part of the learning by the network, the network errors also known as the prediction error is propagated back through the network layers and the weights are updated accordingly. This approach for training a neural network is called backpropagation.

### 7.3.7 Long Short-term Memory (LSTM)

LSTM is a type of Recurrent Neural Network introduced by (Hochreiter & Schmidhuber, 1997). In RNN a given output not only depends on the current input but also considers the previous state. The main problem faced by the RNN is its inability to capture long-term dependencies due to the vanishing gradients. LSTM is designed to learn long-term dependencies by intelligently solving the problem of vanishing gradients. This is achieved by the introduction of a memory cell that can maintain its state

over time. The LSTM memory cell consists of a memory unit called the cell state vector for memorizing previous states and a set of gates responsible for the flow of information through the cell.

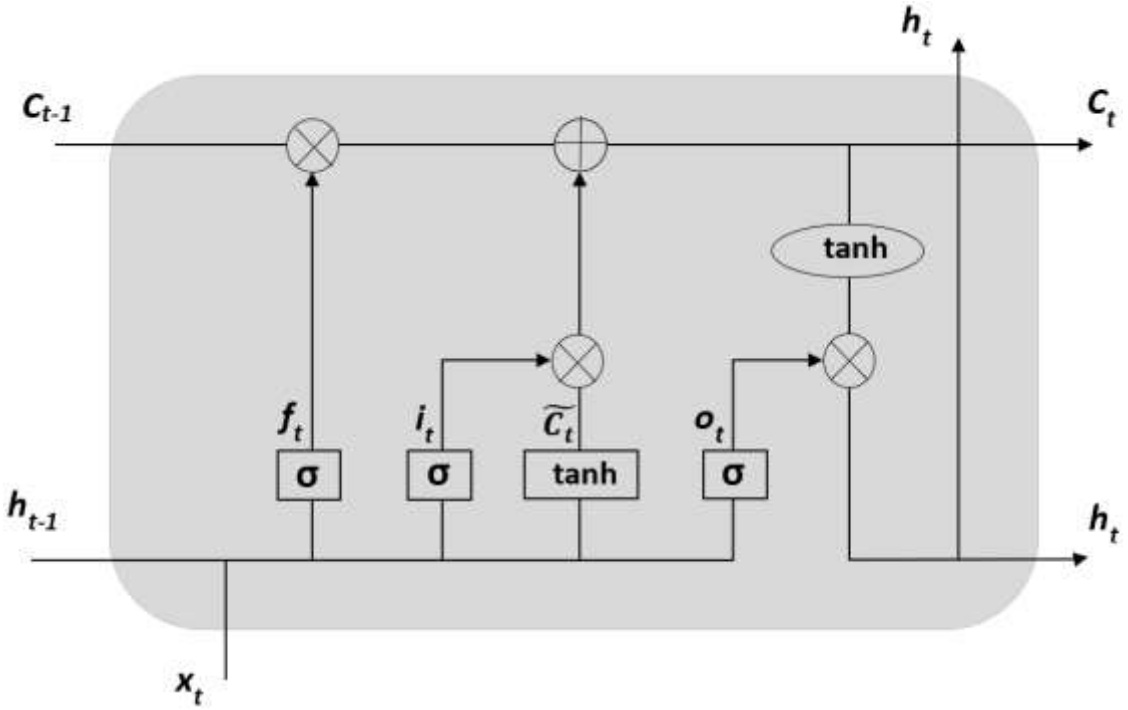



Figure 7.3 Overview of a LSTM cell

Figure 7.3 depicts the relationship between the units of LSTM memory cell and the flow of information through the cell. At the top of the diagram, the cell state, represented by $C$ is passed from one cell to another and $t$ represents a given time stamp. $X_t$ represents the input and $h_t$ is the output for the time stamp $t$. Inside the LSTM memory cell, the information in the cell state is updated using the gates. Gates are composed of sigmoid neural network layer with a pointwise multiplication operator. A single LSTM memory cell has three gates named as forget gate ($f_t$), input gate ($i_t$) and output gate ($o_t$). The output from the sigmoid layer controls the amount of information flowing through the cell. To completely block the flow of information, the sigmoid layer outputs a value of 0 while a value of 1 let all the information to pass though. The sigmoid layer can also regulate partial flow of information through the cell by its output values between 0 and 1.

For a given input $x_t$, the memory cell uses the forget gate to decide, the amount of information to let through the cell state $C_{t-1}$ by looking at the values in $x_t$ and output from the previous cell $h_{t-1}$. Mathematically it is written as:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f \quad (7.28)$$

where $W_f$ is the weight matrix for the forget gate and $b_f$ represent the bias.

The memory cell then decides about the new information to be stored in the cell state. It first uses the input gate layer to identify the value to be updated, then a vector of the new candidate values $\widetilde{C_t}$ are created by the tanh layer. These two steps are given by:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i \quad (7.29)$$

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c \quad (7.30)$$

where $W_i$ and $W_C$ are the weight matrices for the input and cell state, $b_i$ and $b_c$ are their respective biases.

Once the candidate vector is ready and the values to be updated are identified, the memory cell then updates the cell state $C_{t-1}$ into the new cell state $C_t$. The update is written as:

$$C_t = f_i * C_{t-1} + i_t * \tilde{C}_t \quad (7.31)$$

After the cell state is updated, the LSTM memory cell decides about the output. The cell uses a sigmoid layer to decides about the parts of the cell state for the output and using a *tanh* layer, the cell state is transformed and multiplied by the output of the tanh layer.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o \quad (7.32)$$

$$h_t = o_t * \tanh(C_t) \quad (7.33)$$

where $o_t$ is the output for time stamp t, $W_o$ is the weight matrix for the output gate and $b_o$ represent the bias.

LSTM has many advantages compared to the other forecasting approaches such as the ability to work with non-linear and non-stationary time series and without the need to provide information on lagged observations. Apart from the strengths and benefits of LSTM, overfitting is one major challenge for LSTM models. However, this issue can be prevented by ignoring the randomly selected neurons during training with the use of dropout layers (Srivastava et al., 2014).

All of the time series methods discussed in this section is capable to work with multivariate time series except exponential smoothing. In the next section, I explain the details of all the stages of the proposed forecasting methodology for SaaS service factors.

## 7.4 A METHODOLOGY FOR TIME SERIES FORECASTING OF SAAS SERVICE FACTORS

### 7.4.1 Overview of sentiment intensity forecasting methodology

The sequence of steps of the proposed forecast methodology for the SaaS service factors is depicted in Figure 7.4. As discussed in Chapter 4, the sentiment intensity forecasting component is responsible to provide forecast values for each of the SaaS service factor for future timeslots. These forecasted values are then utilized by the trust model to compute the overall trust values.

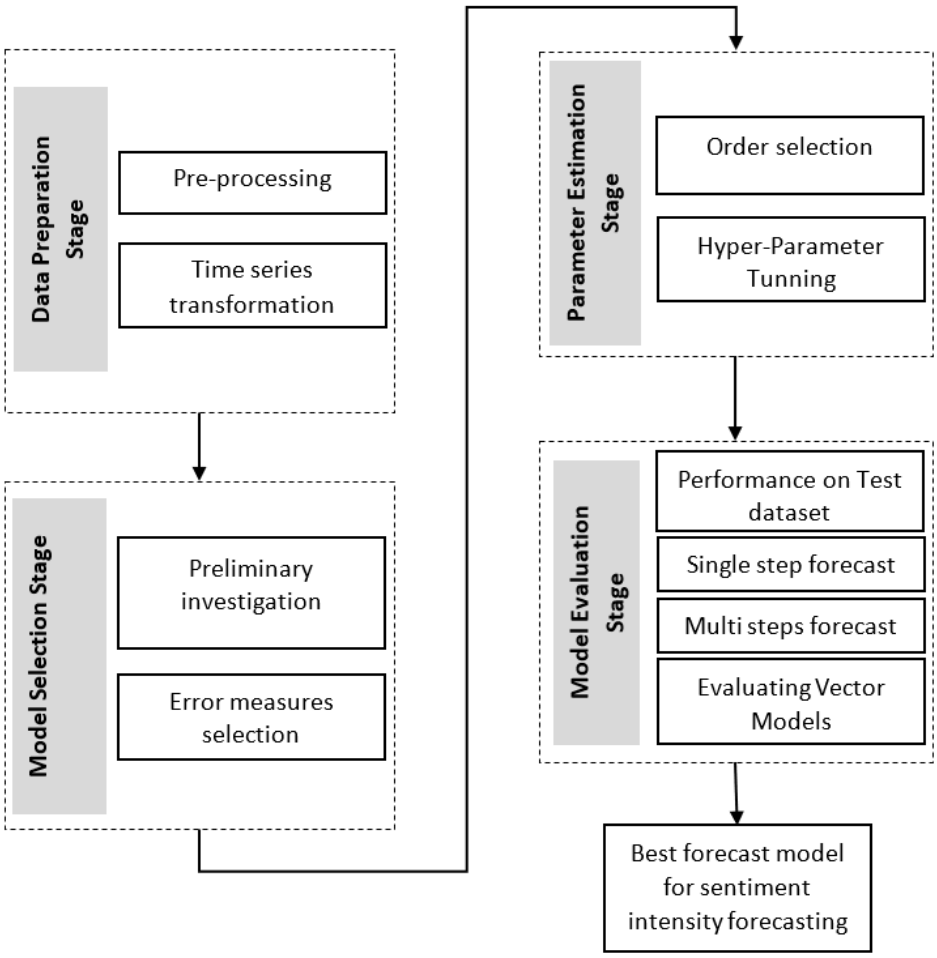



Figure 7.4 Sequence of steps for the SaaS service factors forecasting methodology

The proposed methodology for the development of the forecast component comprises of the following four stages:

1. Data preparation stage
2. Model selection stage
3. Parameter estimation stage

4. Model evaluation stage

In the data preparation stage, a set of tasks are performed to prepare the sentiment intensity timeseries data for the forecasting techniques. Initially, the timeseries for the past observation of each of the service factor of the SaaS products is aligned on a common historic timestamp. Once the dataset is prepared, resampling of the same timeseries is performed to transform the data into supervised input format as some of the forecasting methods are trained and fitted on such input format.

In the model selection stage, a preliminary investigation is conducted by studying various characteristics of the sentiment intensity time series data. This will help to identify the suitable timeseries forecasting technique for the timeseries problem framed in this study. For example, identification of correlation among the SaaS service factors, identification of non-stationary features is performed as part of the preliminary investigation. This step is important as some of the forecasting techniques performs optimally on stationary data. The time series data is then transformed into stationary if required. Based on the preliminary investigation, a number of suitable timeseries forecasting methods are selected along with the relevant error measures used for the evaluation of these methods.

After the timeseries forecasting methods are selected, in the parameter estimation stage, the parameters of each selected model that fits the best on the timeseries data is identified through rigorous tests. In the final stage of the methodology, the selected timeseries forecasting methods with their optimal parameters are evaluated on the test dataset. Finally, the technique with the best performance on the sentiment intensity timeseries dataset is selected for the forecasting component of the trust model which will be used to generate forecasts for the each of service factors (trust factors) in the future time slots.

### 7.4.2 Data preparation

The dataset used for the forecast component of the proposed trust management framework contains the sentiment intensity scores for each service factor of the SaaS products obtained over the lifetime of SaaS products. For the experiments with the selected models, a random service is selected that has sentiment intensity scores for all the service factors. For all the SaaS, the initial dataset contains sentiment intensity scores calculated at irregular time stamps. The initial dataset needs to be in the form of regular time series for modelling and forecasting by the time series forecast models. In the initial dataset, the series of observations for each of the SaaS service factors are associated with time stamps, however, the distance among the values are not consistent, hence, lacking a unit time length.

It is very important for a series to have a constant interval between each consecutive observation as it defines a specific unit also known as 'step' for which time series is modelled and forecasted. To overcome this challenge, following tasks are performed:

- a new time stamp is created by using the first observed date and the last observed date from the initial dataset. This creates a regular time stamp with a single solar day as a unit.
- the observations from the initial dataset is inserted on the relevant time stamp of the new time series. This results in sparse series for all the service factors.
- linear interpolation approach is applied using to impute the missing values between the observed values. The missing values at the start of each series is imputed using the first observed value in the series by filling backwards and the missing values at the end of the series is imputed by carrying forward the last observed value in the series.

Interpolation is the process of finding a value between two known points on a line or curve and is written as:

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1} \tag{7.34}$$

where, $x$ represents the numerical indexes on the time stamp and $y$ represents the observed or missing values on $x$. The start and end indexes are represented by $x_1$ and $x_2$, and the observed values are represented as $y_1$ and $y_2$.

The interpolation can be univariate or multivariate. A univariate interpolation is performed on the function of a single variable whereas, multivariate interpolation is performed on multiple variables. Figure 7.5 depicts the time series dataset created during the data preparation stage.

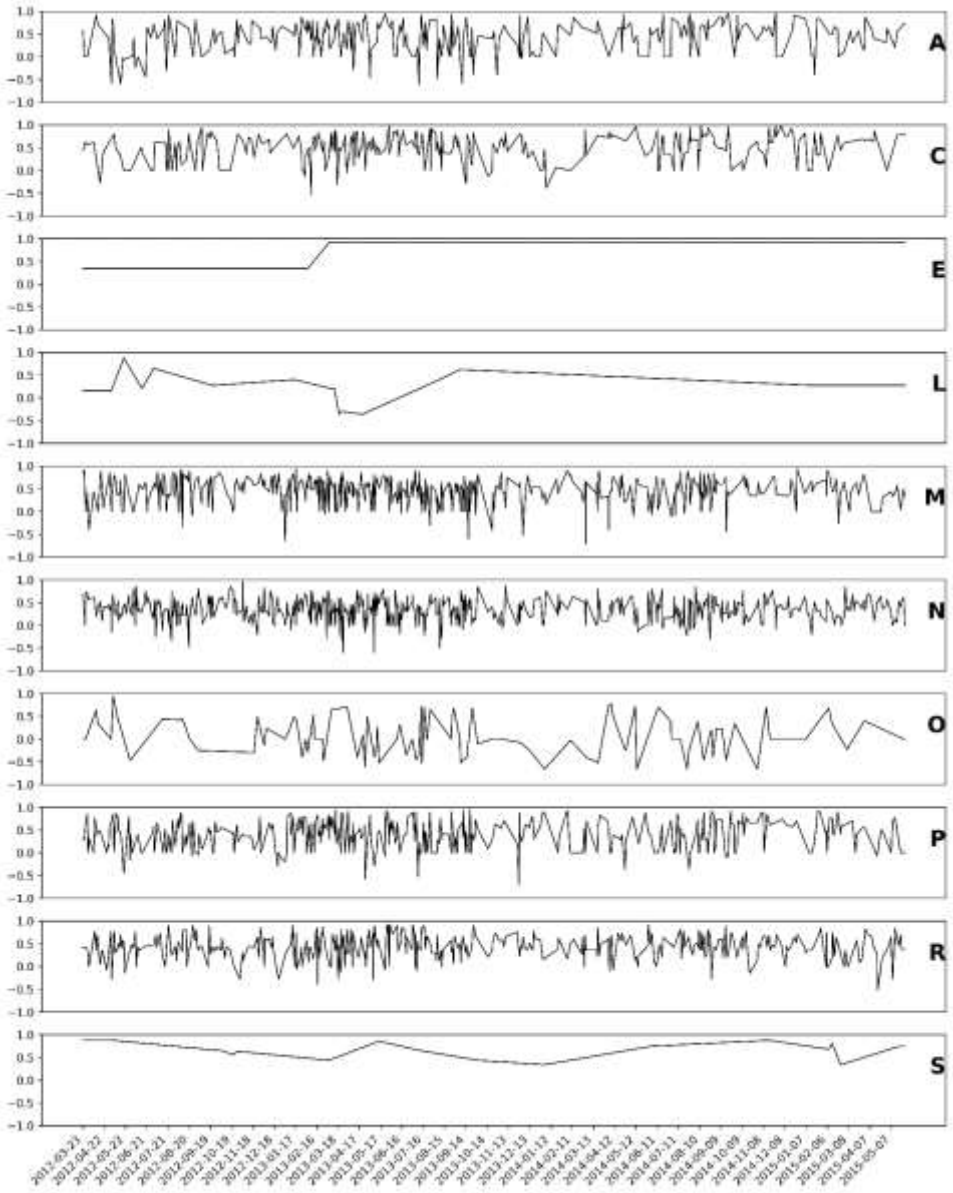



Figure 7.5 Time series of each service factor

Further to the transformation process to time series format, it should be noted that, some of the approaches used for time series forecasting use a supervised method of learning on the training data such as in sequence learning using MLP and LSTM. In such cases, each series must be transformed into features (independent) and target (dependent) format. A series can be transformed for supervised learning models by deciding on the number of input values as features also known as time lag ($l$) and the number of output values ($o$) or targets. The number of output values can be adjusted for multistep forecast settings. For a given series $Y_{n,}$ from time $t$, a supervised observation can be created with independent features as $[y_t, y_{t+l}]$ and the dependent target(s) as $(y_{t+l}, y_{t+l+o}]$.

Figure 7.6 shows the result of the transformation of a series for service factor *M* into supervised data with input features *X1* and *X2* and a target *Y1*.

| M |
|---|
| 0.8622 |
| 0.8804 |
| 0.8986 |
| 0.9168 |
| 0.6585 |
| 0.4002 |
| 0.0000 |
| 0.1005 |
| 0.2010 |
| 0.3014 |
| … |
| … |

| X1 | X2 | Y1 |
|---|---|---|
| 0.8622 | 0.8804 | 0.8986 |
| 0.8804 | 0.8986 | 0.9168 |
| 0.8986 | 0.9168 | 0.6585 |
| 0.9168 | 0.6585 | 0.4002 |
| 0.6585 | 0.4002 | 0.0000 |
| 0.4002 | 0.0000 | 0.1005 |
| 0.0000 | 0.1005 | 0.2010 |
| 0.1005 | 0.2010 | 0.3014 |
| … | … | … |
| … | … | … |
| … | … | … |
| … | … | … |

Figure 7.6 Time series transformation into supervised data

### 7.4.3 Model selection

In this stage, the most suitable approaches are selected for forecasting the sentiment intensity time series for the service factors. The initial selection of the approaches is mainly based on the nature of the sentiment intensity time series. A number of statistical tests are performed on the time series data to identify its statistical properties. The results from these tests help to select the suitable approaches to model the time series data for forecasting. The statistical tests for data analysis used in this chapter is explained in the preliminary investigation subsection.

***Preliminary investigation***

In the preliminary investigation the sentiment intensity time series data explained in the previous subsection is examined for its properties by using different tests. The results of these test are plotted and their summary statistics are used to identify for stationarity of the series and the amount of differencing required to convert the data into stationary series. The plots from the statistical test also help to identify any seasonality, trend and the correlation between the service factors and between the observations in each series.

*Stationarity test*

In the preliminary investigation, the time series for the sentiment intensity of SaaS service factors is tested using the Augmented Dickey-Fuller test (Dickey & Fuller, 1979). Like most of the statistical

tests, the Dickey-Fuller tests the null hypothesis that a unit root is present and the series is not stationary with *p-value* greater than the significance threshold ($p > 0.05$). On the other hand, the series is considered to be stationary if $p < 0.05$ and null hypothesis is rejected.

If the ADF test results on any of the series fails to reject then, the time series data can be easily transformed from non-stationary to stationary using different techniques.

One of the well-known approaches for removing the trend and seasonal components from a time series and to transform it into a stationary series for modelling and forecasting is called differencing (Hyndman, 2018). Differencing is a simple process where a new series is constructed in which each value is the result of subtracting the observed value at the previous time stamp from each observation in the time series. Mathematically it be expressed as:

$$y'_t = y_t - y_{t-1} \tag{7.35}$$

After the differencing process is performed on a series, the unit test is applied on the newly created differenced series. If the results of the unit test show that the trend component still exist, then the differencing is performed again on the differenced series from previous operation until the series become stationary. The number of repetitions of the difference is called the order of differencing. A second-order differencing process in which the differencing is performed twice can be expressed:

$$y''_t = y'_t - y'_{t-1} \tag{7.36}$$

Table 7.1. Augmented Dickey-Fuller Test (Significance level 0.05)

| | A | C | E | L | M | N | O | P | R | S |
|---|---|---|---|---|---|---|---|---|---|---|
| Test Stats | -10.950 | -6.524 | -1.928 | -2.599 | -14.917 | -9.989 | -6.797 | -12.421 | -12.883 | -2.399 |
| p-value | 0.000 | 0.000 | 0.319 | 0.093 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.142 |
| No. of lags | 1.000 | 16.000 | 1.000 | 23.000 | 0.000 | 12.000 | 7.000 | 1.000 | 1.000 | 21.000 |
| No. of Obs. | 1931.000 | 1916.000 | 1931.000 | 1909.000 | 1932.000 | 1920.000 | 1925.000 | 1931.000 | 1931.000 | 1911.000 |
| Critical Value (1%) | -3.434 | -3.434 | -3.434 | -3.434 | -3.434 | -3.434 | -3.434 | -3.434 | -3.434 | -3.434 |
| Critical Value (5%) | -2.863 | -2.863 | -2.863 | -2.863 | -2.863 | -2.863 | -2.863 | -2.863 | -2.863 | -2.863 |
| Critical Value (10%) | -2.568 | -2.568 | -2.568 | -2.568 | -2.568 | -2.568 | -2.568 | -2.568 | -2.568 | -2.568 |
| Conclusion | Stationary | Stationary | Non-Stationary | Non-Stationary | Stationary | Stationary | Stationary | Stationary | Stationary | Non-Stationary |

The results of Augmented Dickey-Fuller test on all the service factors are shown in Table 7.1. At the significance level of 0.05, the ADF statistics shows the p-values for most of the service factors are below the significance level indicating the associated series are stationary. However, the p-values of

0.319, 0.093 and 0.142 for the service factors E, L and S respectively, are greater than the threshold (0.05) and indicates that the series are non-stationary.

As explained in earlier, differencing is used to transform a non-stationary time series into stationary. Based on the ADF test results, the first order differenced stationary series for all the service factors is depicted in Figure 7.7. Applying a first order differencing transforms the series for E, l and S service factors into stationary series. A repeated ADF test confirms the successful transformation. The initial non-stationary series for the service factors E, L and S is further investigated by breaking them down into their constituent components.

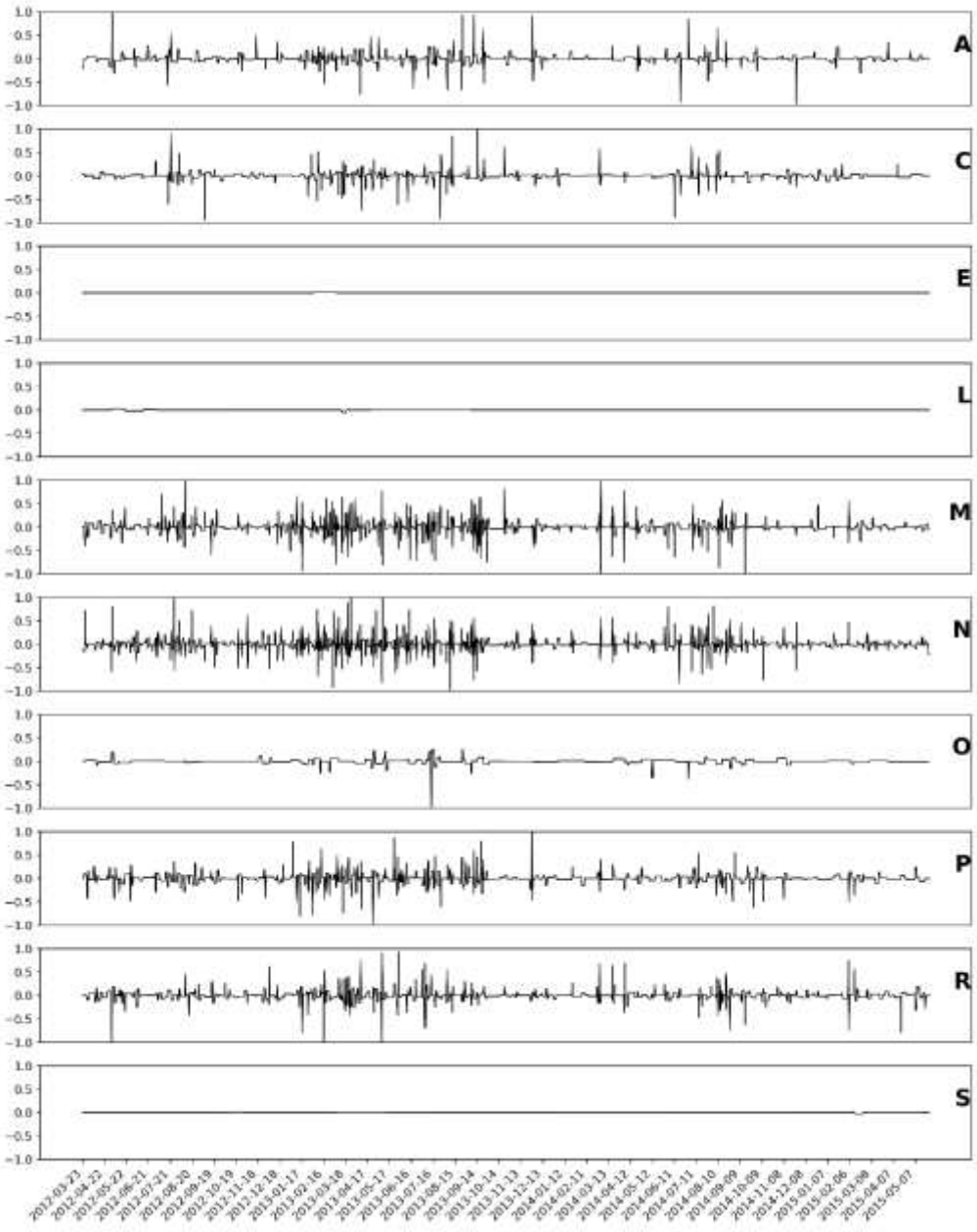



Figure 7.7 Stationary series after 1st-order differencing

The constituent components of the series for the service factors E, L and S are depicted in Figure 7.8, Figure7. 9 and Figure 7.10 respectively. In all the three series we can see that the entire series was taken as the trend component. There are no seasonal component and residual plot shows a straight horizontal line. This breakdown will help in selecting the suitable approach that can handle additive trend in a time series.


Figure 7.8 Components of service factor E

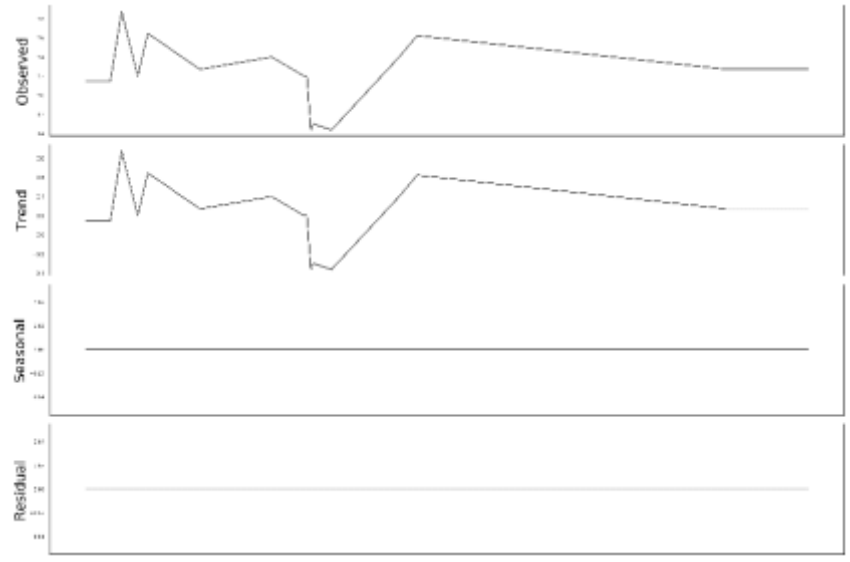



Figure 7.9 Components of service factor L

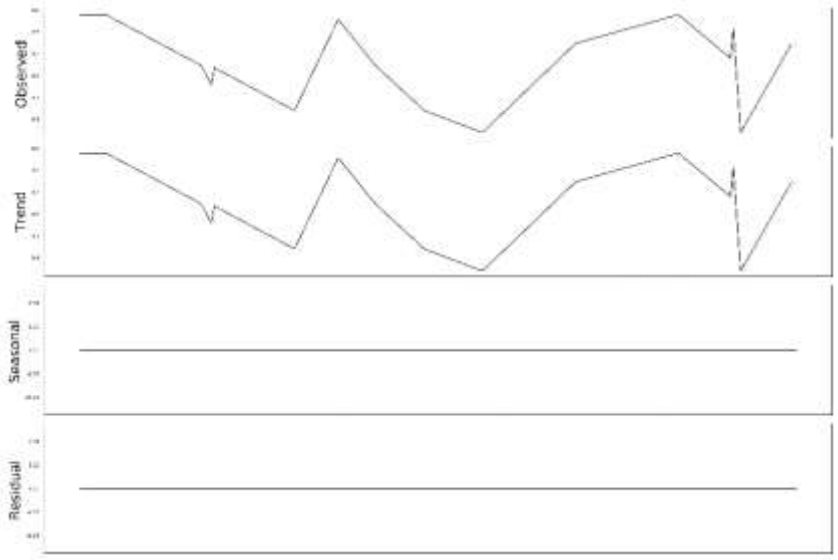



Figure 7.10 Components of service factor S

*Causation test*

When modelling multivariate time series for forecast purposes, it can occur that the observations in one series may be causing the values in another series. This insight into multiple parallel time series can help to select an appropriate modelling approach. The causality between a pair of series can be captured by using the Granger's causality test (Granger, 1969). Given two series *X* and *Y,* the Granger's causality tests the null hypothesis that the series in *X* does not cause the series in *Y*. The output p-values from the test are tested against the significance level of 0.05. The null hypothesis can be rejected if the p-values from the test falls below the significance level. Hence, suggesting that the observations in *X* does have statistically significant effect on values in *Y*. In this case a vector model can prove to be useful for modelling the series for forecasting. As part of the preliminary investigation, the Granger's causality is performed on all the service factors.

The Granger causality test is applied on the dataset and the test results are shown in Table 7.2. The results of the Granger causality test are the output p-values on all the service factors where, the columns represent the predictors *X* and the rows *Y* are the response variables.

The results in Table 7.2 shows the presence of granger causality between some of the service factors. For example, the predictor *A* from Table 7.2 is causing the responses *C, N, O* and *P* while for the remaining service there is no significant causation by *A*. Similar is the case with the other service factors where there is causation between some of the variables. Based on the outcome of the Granger's causality the selection of a vector time series modelling may or may not be suitable for forecast. In

this case, the suitable model can be identified with extensive experiments with both univariate and multivariate time series modelling approaches.

Table 7.2. Granger Causality test (Significance level = 0.05)

| | | X | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | C | E | L | M | N | O | P | R | S |
| Y | A | 1 | 0.3274 | 0.6029 | 0.0515 | 0.0977 | 0.0081 | 0.0489 | 0 | 0.0088 | 0.2818 |
| | C | 0.0135 | 1 | 0 | 0.0081 | 0.0007 | 0.3939 | 0.0624 | 0.033 | 0.1603 | 0.6469 |
| | E | 0.2418 | 0.1629 | 1 | 0.776 | 0 | 0.0001 | 0.1105 | 0 | 0.7783 | 0.0361 |
| | L | 0.1947 | 0.0017 | 0.5855 | 1 | 0.0125 | 0.0575 | 0.6562 | 0.0032 | 0.024 | 0.5943 |
| | M | 0.2523 | 0.0138 | 0.7087 | 0.0006 | 1 | 0.0577 | 0.0337 | 0.1688 | 0.0028 | 0.8314 |
| | N | 0.0194 | 0.0541 | 0.0002 | 0 | 0.0744 | 1 | 0 | 0.2756 | 0.001 | 0.8826 |
| | O | 0.0002 | 0.3362 | 0.3235 | 0.7652 | 0.0531 | 0.0016 | 1 | 0.0019 | 0.1928 | 0.9311 |
| | P | 0.034 | 0.0036 | 0.0189 | 0.0369 | 0.347 | 0.0787 | 0.0225 | 1 | 0.6686 | 0.5655 |
| | R | 0.0567 | 0.0396 | 0.1727 | 0.0002 | 0.5392 | 0.3783 | 0.2254 | 0.0012 | 1 | 0.1092 |
| | S | 0.9089 | 0.6238 | 0.8652 | 0.3692 | 0.1892 | 0.7995 | 0.3358 | 0.657 | 0.1296 | 1 |

*Cross-Correlations test:*

Cross-correlation test is performed next to find the correlation among the different service factors. This step is important in the selection of appropriate category of the models. As discussed in Section 7.3, the time series forecasting approaches can works with univariate time series as well as with multivariate series. In the absence of a significant correlation among the service factors, the vector version of the time series forecasting approaches are not suitable and forecast models for each service factor is designed separately. However, a significant presence of relationship between service factors can prove to be useful in modelling and forecasting the sentiment intensity values for the SaaS service factors. In this step, I have applied Pearson's correlation coefficient (Lee Rodgers & Nicewander, 1988) which is a well-known measure for computing the linear relationship between two variables. The correlation between the values of two service factors $x$ and $y$ is calculated as the covariance of the two series divided by the product of the standard deviations of $x$ and $y$, written as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{7.37}$$

where $r_{xy}$ represents the correlation coefficient between the variables $x$ and $y$ with n number of observations, $x_i$ and $y_i$ are the $i^{th}$ values of the variables $x$ and $y$, and $\bar{x}$ and $\bar{y}$ are the mean values of $x$ and $y$. The result of the correlation test is a value between -1 and 1. A correlation value of 1 means a perfect positive correlation between $x$ and $y$ and indicates an increase with the same magnitude in the value of $y$ when $x$ increases. A correlation value of -1 is a perfect negative relationship and indicates

a decrease in the value of $y$ when $x$ decreases with the same magnitude. A correlation value of 0 means the complete absence of any relationship between $x$ and $y$.

Besides the importance of cross-correlation between the service factors, the series correlation or the correlation among the observations of the same series can be utilized to find the appropriate forecast model. Further to cross-correlation test, serial correlation tests for autocorrelation and partial autocorrelation for the series in each service factor is performed.

The correlation between the series in different service factors plays an important part in the model selection and parameter estimation. A correlation between two series also known as cross-correlation, can help in the selection of an individual or vector version of the forecast models. The results of the cross-correlation test using Pearson's Correlation Coefficient method is shown in Table 7.3. The correlations between the service factors are calculated using Equation 7.37 and the results are shown in Table 7.3. The correlation results from Table 7.3 shows no significance correlation between any of the service factors as all the values are closer to 0. The absence of a strong correlation between the service factors suggests that the modelling the series for each service factor is more suitable compared to a vector version of modelling using all the service factor under a single model.

Table 7.3. Correlation results between service factors

| | A | C | E | L | M | N | O | P | R | S |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 1.0000 | 0.0699 | 0.1209 | -0.1092 | 0.1044 | 0.0413 | 0.0525 | -0.0217 | 0.0132 | 0.0515 |
| C | 0.0699 | 1.0000 | 0.0921 | -0.1812 | 0.0139 | -0.0924 | 0.1433 | 0.0398 | 0.0617 | 0.1091 |
| E | 0.1209 | 0.0921 | 1.0000 | -0.1054 | -0.0731 | -0.0467 | -0.0210 | 0.1301 | 0.1615 | -0.1819 |
| L | -0.1092 | -0.1812 | -0.1054 | 1.0000 | -0.0191 | -0.0001 | -0.0939 | -0.0082 | 0.0107 | -0.2892 |
| M | 0.1044 | 0.0139 | -0.0731 | -0.0191 | 1.0000 | -0.0429 | -0.0525 | 0.0912 | 0.0622 | 0.0084 |
| N | 0.0413 | -0.0924 | -0.0467 | -0.0001 | -0.0429 | 1.0000 | -0.0797 | -0.0202 | -0.0658 | -0.0483 |
| O | 0.0525 | 0.1433 | -0.0210 | -0.0939 | -0.0525 | -0.0797 | 1.0000 | -0.0032 | 0.0818 | 0.1368 |
| P | -0.0217 | 0.0398 | 0.1301 | -0.0082 | 0.0912 | -0.0202 | -0.0032 | 1.0000 | 0.0844 | -0.0481 |
| R | 0.0132 | 0.0617 | 0.1615 | 0.0107 | 0.0622 | -0.0658 | 0.0818 | 0.0844 | 1.0000 | 0.0421 |
| S | 0.0515 | 0.1091 | -0.1819 | -0.2892 | 0.0084 | -0.0483 | 0.1368 | -0.0481 | 0.0421 | 1.0000 |

*Serial-Correlations test:*

The correlation can also exist between the observations of the same time series also known as the serial correlation. Multiple linear regression models can fail to provide an optimal fit on the series if these correlations are not captured. The significant level of correlation between a given value in a time series with its preceding values can help to find the optimal fit for a linear regression model.

The coefficient of correlation between two observations in a time series is called the autocorrelation function (ACF). The ACF measures the relationship between a given observation $y_t$ with an observation from prior time step $y_{t-n}$ (direct correlation) and also with all the observations between the time steps *t* and *t-n* (indirect correlation). The partial autocorrelation function (PACF) can be seen as the subset of ACF in which, the correlation between a given observation $y_t$ and an observation at the prior time step $y_{t-n}$ is calculated without including the relationships of $y_t$ with the intervening observations.

The ACF and PACF helps in understanding the time series for better selection of the autoregression order AR(p) and the order for moving average MA(q) models. A series generated by AR process considers the *p* previous observations also called lags in its regression process. This entails that for a given series y, finding the best estimate for the value of *p* can help to capture the relationship among the observations and fit the optimal AR model on *y.* In case of MA(q) models, a series can be generated by using the time series of residual errors from prior values where, *q* is the number of prior time steps considered in the MA process. This means that, estimating the best value for *q* can help to build an optimal MA(q) model.

For the time series forecast model performance, it is important to find the model's optimal parameter and ACF and PACF in that regard.

The autocorrelation and partial-autocorrelation between the observations are depicted in Figure 7.11 and Figure 7.12. The significance threshold also known as the confidence intervals are set to 95% and the correlation values for the lags outside these thresholds is considered significant. Figure 7.11 shows the Autocorrelation plot of all the service factors up to the first 40 lags.

The ACF graphs in Figure 7.11 shows significance autocorrelations at multiple lags for each service factor. In this case, selecting the most suitable lag order for the models such AR is challenging and the models has to be fit with different lag orders to identify the best fit.

From the PACF graphs in Figure 7.12, for each of the series, there are partial autocorrelations significantly different from 0 and are above the significant threshold. The results from the PACF graph are used in selecting the best parameter for the model estimation in the next section.

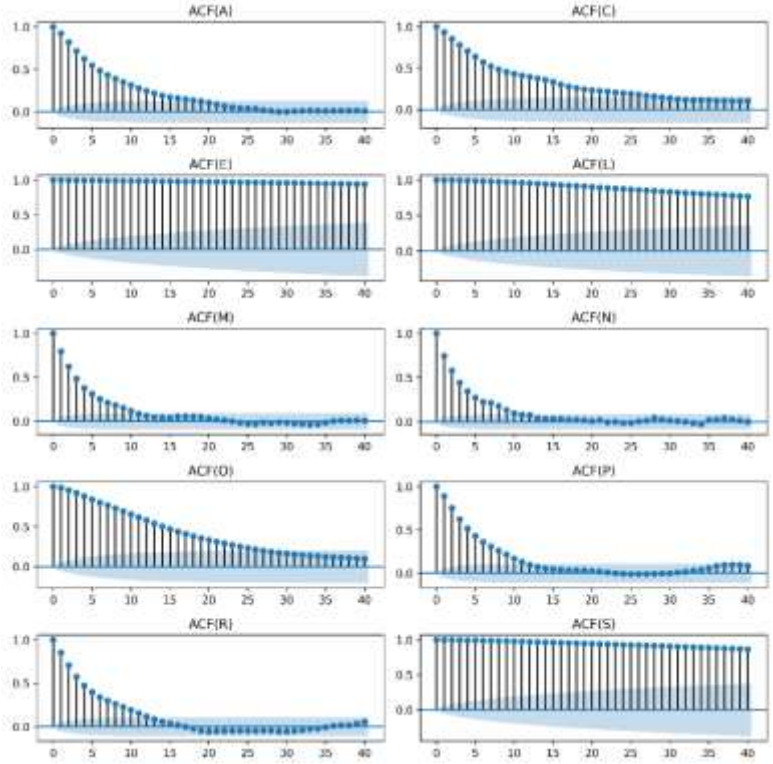



Figure 7.11 Autocorrelations within the series for each service factor

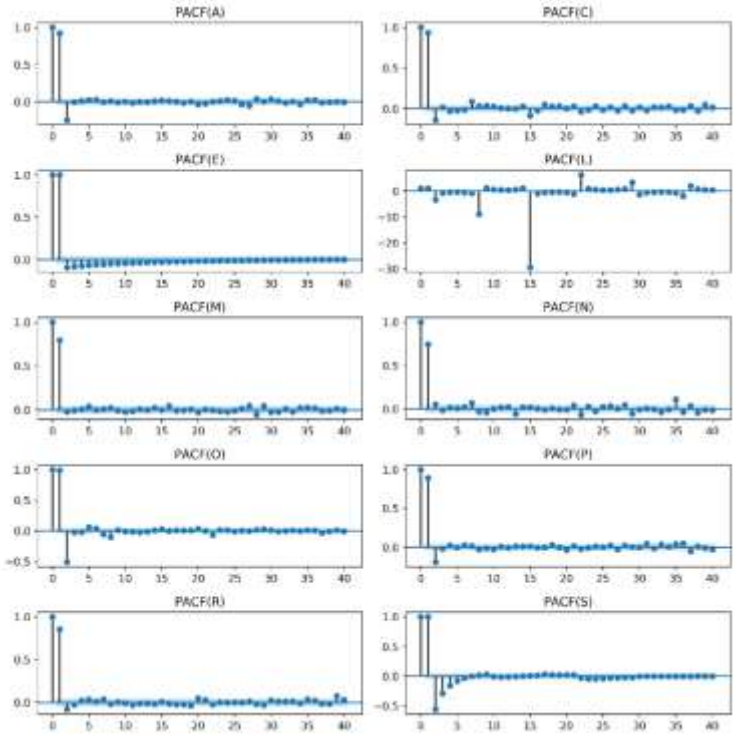



Figure 7.12 Partial-autocorrelations within the series for each service factor

The preliminary investigation of the time series data for SaaS sentiment intensity scores for model selection suggest that:

- The results of the Granger's causality test suggest that there is significant causation between some of the service factors. However, most of the service factors do not show any significant causation. Therefore, experimental results from both univariate and multivariate forecasting models will be decisive.
- There is no significant cross-correlation between the service factors. Therefore, modelling the series for each service factor separately is more suitable compared to modelling of all the service factors a vector.
- Both the ACF and PACF graphs show multiple significance series correlation between the series of most of the service factors. Therefore, an extensive parameter estimation using different lag values is needed to find the optimal fit for the models.
- The ADF test suggest that the series for the service factors E, L and S are non-stationary in nature and for forecasting models such as ARIMA, these series should be converted to stationary with first order differencing. Figure 7.8, Figure 7.9 and Figure 7.10 shows the breakdown of the series for service factors E, L and S into its components. There is no observable long-term trend, however, these series contain smaller short-term trends. For exponential smoothing models Holt's linear is more suitable for modelling the service factors E, L and S.

***Error measures***

In this subsection I explain the error measures used for evaluating the time series forecasting models. The error measure, as the name suggest, are used to measure the difference between the observed sentiment intensity values and the forecasted values by the fitted time series forecasting models. These error measures also used in tunning the parameters of the time series forecasting approaches such as MLP and LSTM. For every permutation of the parameters, the model is fitted to the time series data and forecasts are recorded by the fit model. The forecast errors are computed using the error measures for each permutation and are compared to select the best permutation. The selected error measures for the proposed methodology are briefly explained below.

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)

The MAE measures the error between two observations. Generally, one of the values belong to the actual observed variable the other is a predicted value for the same variable. It is measured as the arithmetic average of absolute errors. Mathematically it is written as:

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{7.38}$$

where $\hat{y}_i$ is the predicted and $y_i$ is the $i^{th}$ actual observed value.

The RMSE is the square root of the mean squared error which is calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \tag{7.39}$$

The lower error values from the MAE and RMSE indicates high performance by the forecast models.

***Measures for model selection***

As discussed in the previous subsection, for some of the forecasting model such as MLP and LSTM, the optimal parameters for the best fit is identified by using the error measure RMSE. Beside these error measures, the best fit for the time series forecasting models such as ARIMA and Exponential smoothing, can be identified using the following measures.

- Akaike's Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

AIC is a maximum likelihood-based measure for the best fit of a model. It is calculated as:

$$AIC = 2k - 2\ln(\hat{L}) \tag{7.40}$$

Where k is the number of model parameters and $\hat{L}$ is the maximum value of the likelihood function. Lower values of AIC indicate a better model-fit for forecasting.

BIC is another similar measure to find best model fit. BIC is calculated as:

$$BIC = k\ln(n) - 2\ln(\hat{L}) \tag{7.41}$$

Where k represents the number of model parameters and n is the number of data points.

### 7.4.4 Parameter estimation

In this stage of the proposed SaaS sentiment intensity forecast methodology, the optimal fit for the selected models is estimated using the measures for model selection and error measures described in the previous subsection. The parameter estimation for all the selected time series forecasting methods

is performed by using a training dataset which consists of 70% of observations from the time series dataset described in Section 7.4.1. The remaining 30% of the observations is used as test data in the model evaluation stage discussed in the next section.

Based on the outcome of the preliminary investigation performed in the previous subsection, four time series forecasting approaches are selected and their parameters are estimated as follows:

*ARIMA model:*

In the parameter estimation of ARIMA models I perform a fitting procedure to find the coefficients of the regression model. For the *ARIMA(p, d, q)* model, since the series for service factor E, L and S is not stationary therefore, an *ARIMA(p, 1, q)* is fitted on these service factors. For the rest of the service factors *ARIMA(p, 0, q)* is fitted. The models are fitted with different permutations of the p and q. The p-orders for ARIMA models are fitted with *p = (0, 1, 2, 3, 4, 5, 6)* and the range of values for the q-order is from *q = (0, 1, 2, 3, 4, 5, 6)*. This extensive parameter estimation is important as the ACF and PACF may suggest multiple lags with significant correlations. Each fitted model is cross-compared using the AIC and BIC measure and the model parameters with the lowest AIC is selected for further evaluation. For the multivariate implementation (VARMA), the models are fitted with similar permutations on stationary time series data.

Table 7.4. ARIMA optimal parameter for each service factor

| Factor | p | d | q | AIC | BIC | HQIC | LLF | Sigma |
|---|---|---|---|---|---|---|---|---|
| **A** | 2 | 0 | 1 | -2073.77 | -2047.72 | -2064.02 | 1041.886 | 0.012532 |
| **C** | 6 | 0 | 4 | -2275.62 | -2213.1 | -2252.21 | 1149.812 | 0.010667 |
| **E** | 6 | 1 | 5 | -17113.4 | -17045.7 | -17088 | 8569.693 | 1.81E-07 |
| **L** | 6 | 1 | 5 | -12224.7 | -12157 | -12199.3 | 6125.352 | 6.78E-06 |
| **M** | 5 | 0 | 2 | -1109.19 | -1062.29 | -1091.63 | 563.5928 | 0.025371 |
| **N** | 5 | 0 | 4 | -1276 | -1218.69 | -1254.54 | 648.9983 | 0.022339 |
| **O** | 4 | 0 | 2 | -4070.13 | -4028.45 | -4054.52 | 2043.063 | 0.002838 |
| **P** | 6 | 0 | 5 | -1635.65 | -1567.92 | -1610.28 | 830.8237 | 0.017119 |
| **R** | 6 | 0 | 4 | -1844.5 | -1781.98 | -1821.09 | 934.2485 | 0.014613 |
| **S** | 5 | 1 | 6 | -16793.2 | -16725.5 | -16767.8 | 8409.588 | 2.31E-07 |

As discussed in the previous section, both the ACF and PACF graph show autocorrelations and partial autocorrelation that are above the significant threshold line. In this scenario it is challenging to find the suitable lag term *p* for the autoregressive *q* for the moving average part of ARIMA model. Therefore, I have fitted different ARIMA models up to the *p* and *q* orders of 6, and selected the model with the lowest AIC. The optimal parameter for the selected ARIMA models is shown in Table 7.4.

For each service factor, a separate ARIMA model is fitted by thoroughly investigating with different orders for *p* and *q* terms. As discussed in the previous subsection, the series for service factors E, L and S are non-stationary. Therefore, ARIMA models with first order differencing ARIMA(*p, 1, q*) is fitted on these series. Based on ADF test results reported in the previous subsection, the series for the rest of the service factors are stationary and need to differencing. ARIMA model ARIMA(*p, 0, q*) is fitted on the rest of series. The ARIMA models for each service factor with the lowest AIC values are presented in Table 7.4.

Figure 7.13 shows the residuals plots from the fitted ARIMA models. Beside some irregular fluctuations, the residual plots show no significant trend or seasonality in the model residuals. The model residuals for the service factors E, L and S are flat and the ARIMA models captured all the information in these series. Keeping in mind that the actual observations for these service factors were initially identified as non-stationary. The variance in the model residuals of all service factors can be treated as constant as there is no significant variance in the residual data.

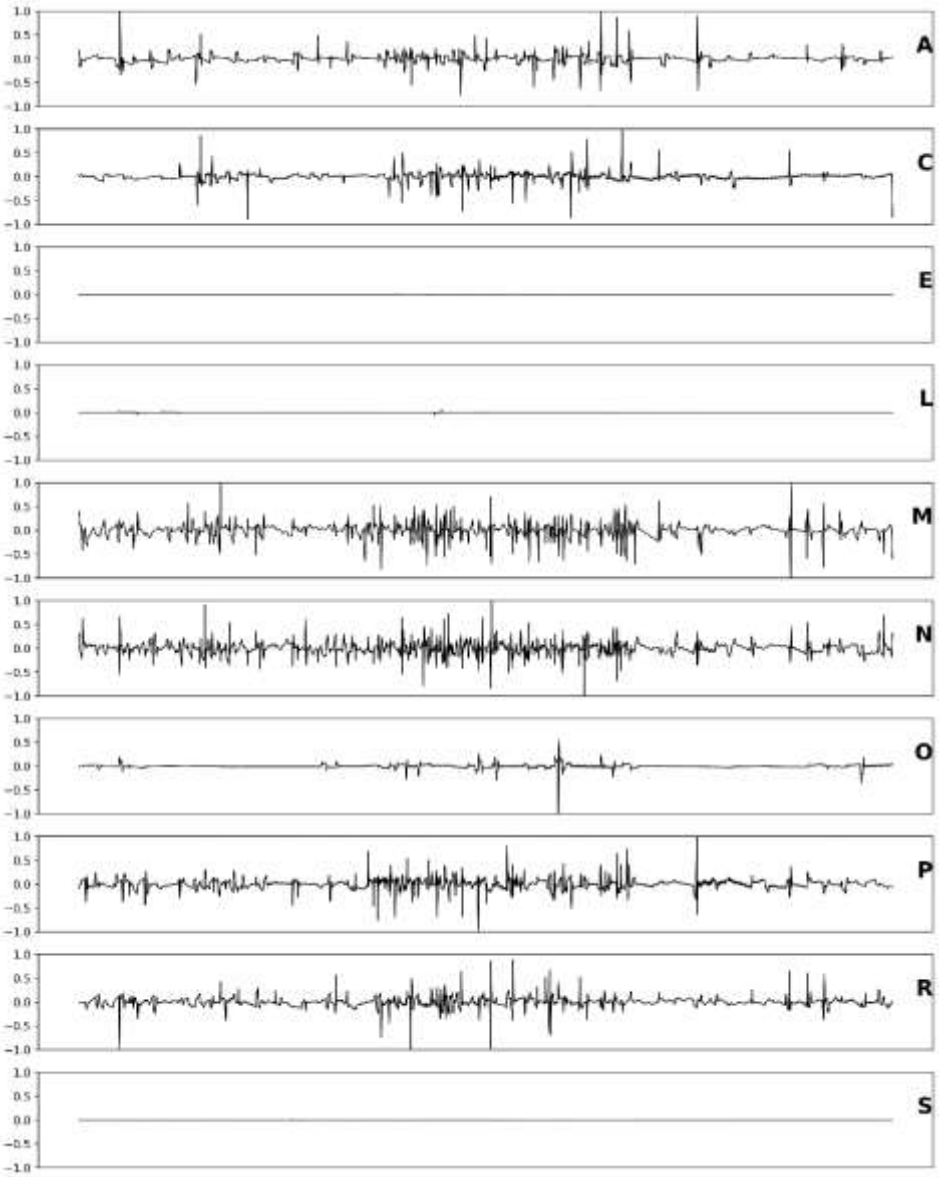



Figure 7.13 ARIMA model residuals

The ACF and PACF of the model residuals for all the service factors, depicted in Figure 7.14 and Figure 7.15, shows few significant autocorrelations and partial autocorrelations at random lags above the significance threshold. This indicates that a small amount of data is not captured by the fitted models. However, the ACF and PACF shows that the fitted ARIMA models have captured significant amount of data correctly. Therefore, it can be stated that the parameters selected for the ARIMA models in this stage of methodology is optimal.

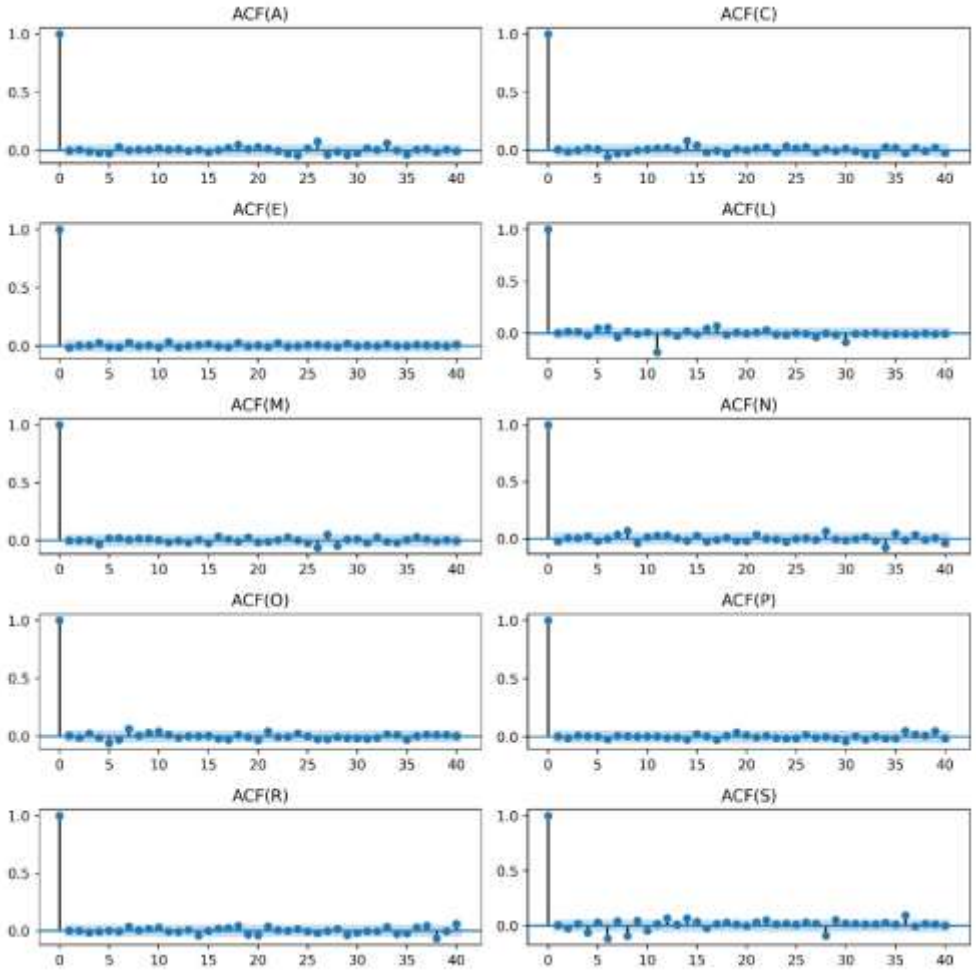



Figure 7.14 Autocorrelations in ARIMA model residuals

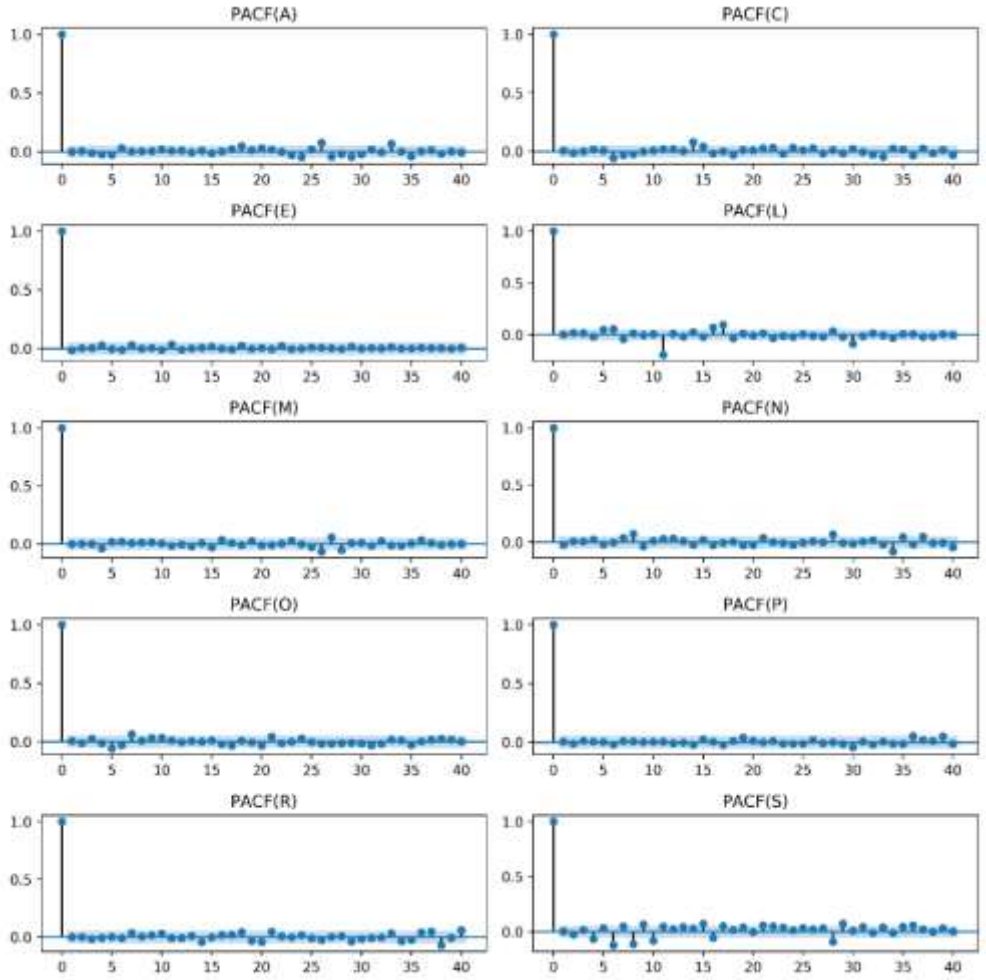



Figure 7.15 Partial autocorrelations in ARIMA model residuals

The Kernel Density Estimate (KDE) plots of the residuals is shown in Figure 7.16. The probability density of the residuals for all the service factors are normally distributed with less spread that shows good predictive quality of the models with less bias.

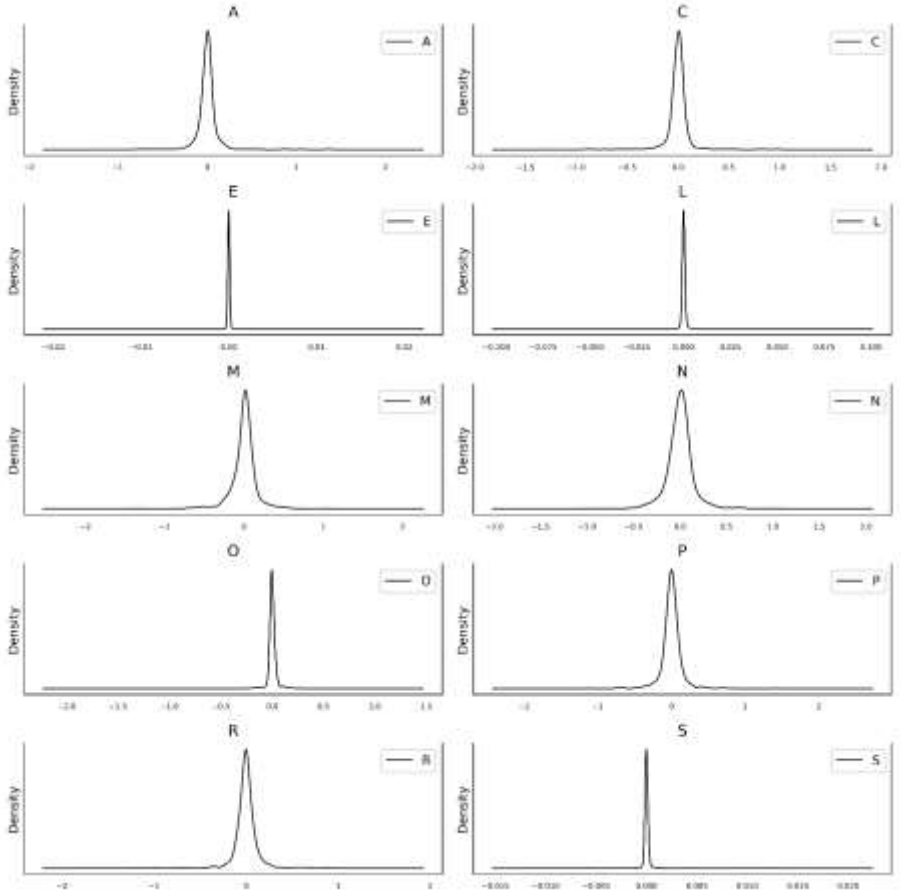



Figure 7.16 KDE of the ARIMA model residuals

*Exponential smoothing model:*

Simple exponential smoothing models (SES) are fitted for the service factors with stationary series by estimating the optimal value for model parameter $\alpha = [0, 1]$. Alpha ($\alpha$) is the smoothing coefficient for the level. These SES models are also known as (N, N) model which means that the model does not consider any tend and seasonal component in the series during modelling and forecasting.

For the service factors E, L and S with non-stationary series, the Holt's linear trend method (A, N) is used and the optimal values for the parameters $\alpha$ and $\beta$ are estimated for each model. Beta ($\beta$) is the smoothing coefficient to control trend.

The exponential smoothing model parameters with optimal fit are depicted in Table 7.5. The fitted models for the service factors A, C, O, P and R have the maximum value (1) for $\alpha$ parameter which indicates that these models only consider the most recent observations in the exponential smoothing process for forecasting. However, these models will be computationally very efficient. For the models fitted on the series of service factors E, L and S, Table 7.5 shows the optimal smoothing coefficient

β used for the Holt's linear trend models. The fitted exponential smoothing models with the optimal parameters from Table 7.5 is selected for further evaluation in the next stage.

Table 7.5. Exponential Smoothing model fit parameters

| | AIC | BIC | α | β | $l_0$ | $b_0$ |
|---|---|---|---|---|---|---|
| **A** | -5772.96 | -5762.54 | 1 | | 0.571874 | |
| **C** | -6047.64 | -6037.22 | 1 | | 0.440405 | |
| **E** | -20560.7 | -20539.8 | 0.9999 | 0.999904 | 0.339261 | 0.00963 |
| **L** | -15938.7 | -15917.9 | 0.999901 | 0.999904 | 0.152361 | 0.00963 |
| **M** | -4803.8 | -4793.38 | 0.866299 | | 0.864914 | |
| **N** | -4962.11 | -4951.69 | 0.733976 | | 0.631641 | |
| **O** | -7560.48 | -7550.06 | 1 | | 0 | |
| **P** | -5347.88 | -5337.46 | 1 | | 0.28595 | |
| **R** | -5559.6 | -5549.18 | 1 | | 0.4201 | |
| **S** | -20201.7 | -20180.8 | 0.9999 | 0.999904 | 0.879961 | 0.00963 |

The exponential smoothing model residuals for each service factor depicted in Figure 7.17, shows no significant trend or seasonality. The ACF and PACF of the exponential smoothing model residuals are shown in Figure 7.18 and Figure 7.19. Similar to the ARIMA models, the exponential smoothing model residuals also show few significant correlations at different lags which indicates that the models failed to capture the series completely. However. Most of the correlations are below the significance threshold.

The KDE plots for the exponential smoothing model residuals in Figure 7.20, shows the residuals have less bias with normal distribution.

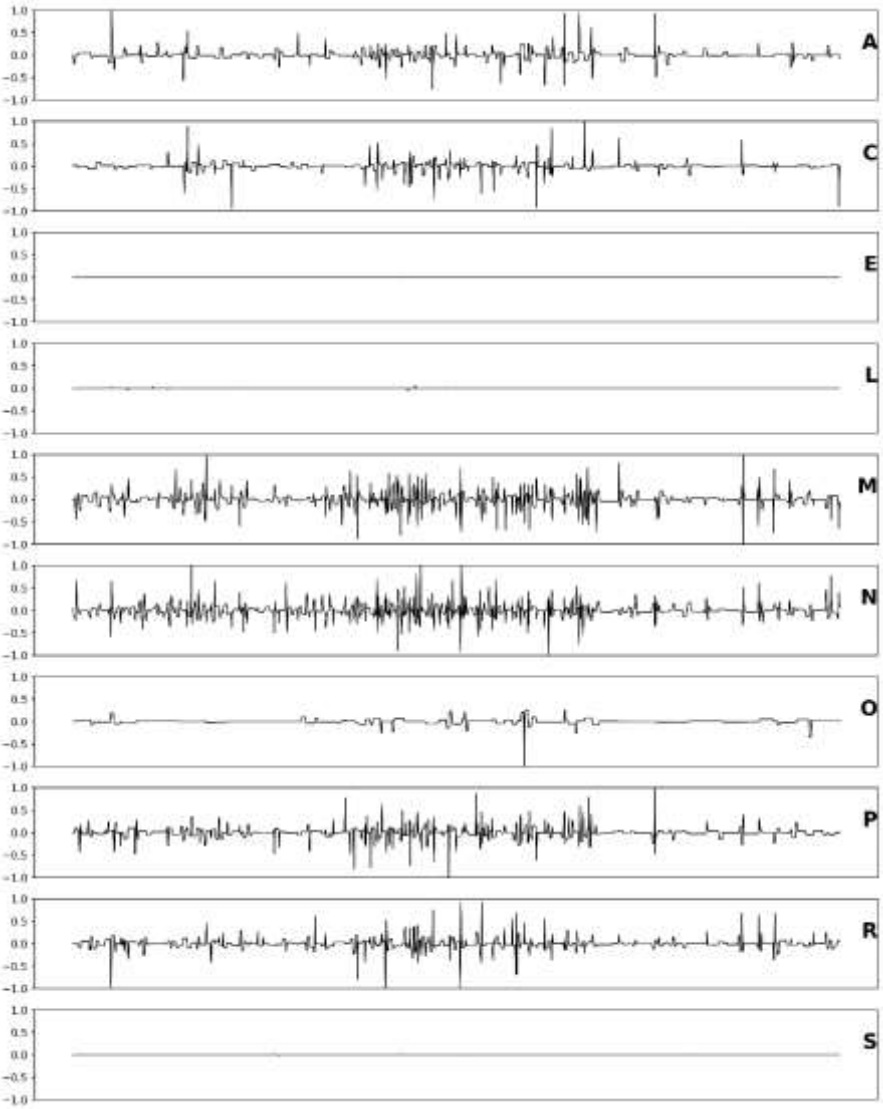



Figure 7.17 Exponential smoothing model residuals

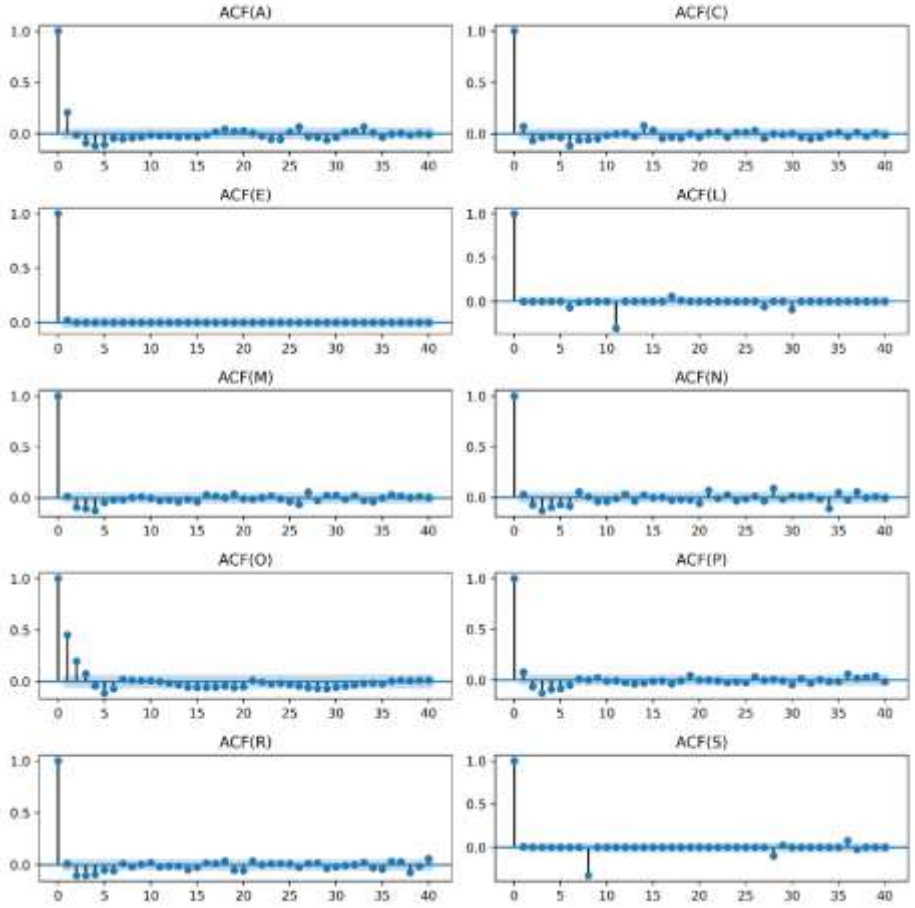



Figure 7.18 Autocorrelations in exponential smoothing model residuals

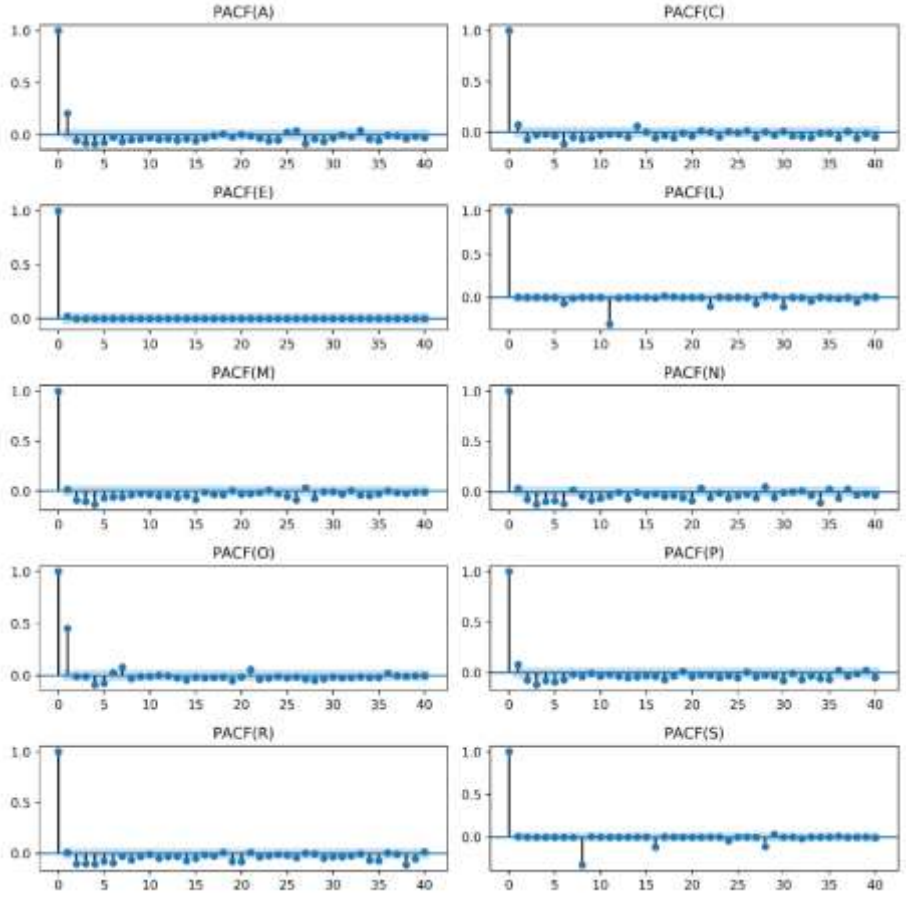



Figure 7.19 Partial autocorrelations in exponential smoothing model residuals

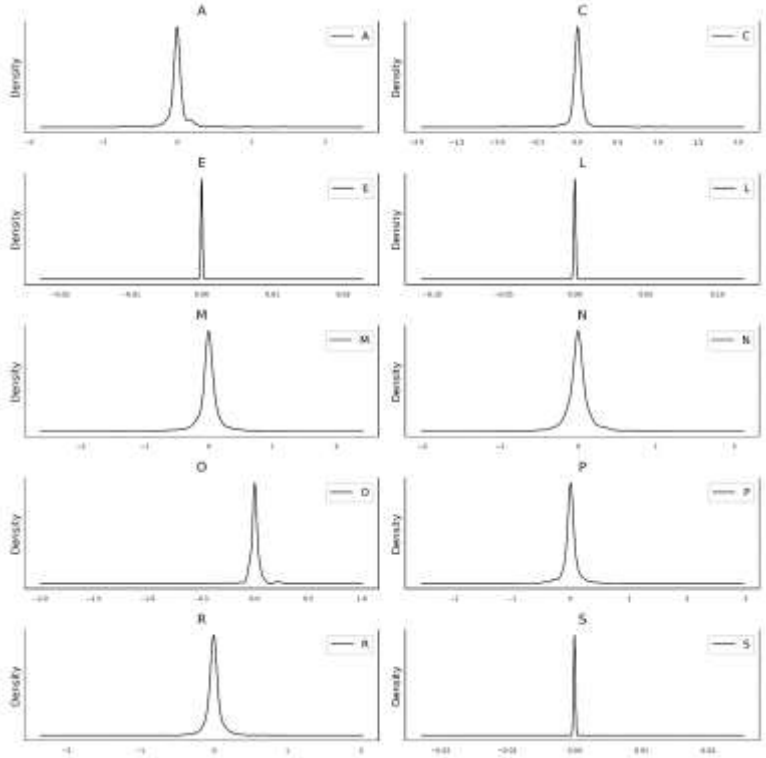



Figure 7.20 KDE of the exponential smoothing model residuals

*MLP:*

The MLP is fitted on the sentiment intensity time series data in a supervised setting for sequence learning as discussed in Section 7.4.1. The MLP models are fitted with different permutations of the parameters depicted in Table 7.6.

Table 7.6. MLP and LSTM parameters

| Parameters | Value |
|---|---|
| Lags as Features | 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 |
| Nodes | 20, 40, 60, 80, 100, 120, 140, 160, 180, 200 |
| Epochs | 50, 100, 150, 200, 250 |
| Batch size | 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 |
| Activation | 'tanh', 'relu' |

The parameters of the fitted MLP model with the lowest RMSE scores are selected and the selected model is used for further evaluation in the next step.

The parameters of the fitted MLP model with the least RMSE score on the training time series data is shown in Table 7.7. The summary of the MLP forecast model is shown in Figure 7.21. In the next section, these optimal model parameters are used for further evaluation on the model's forecast performance on the test data.

Table 7.7. Parameters of the fitted MLP model

| Lags | nodes | epochs | batch size | activation |
|---|---|---|---|---|
| 1 | 20 | 50 | 80 | tanh |

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 60)                120
_________________________________________________________________
dropout_2 (Dropout)          (None, 60)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 61
=================================================================
Total params: 181
Trainable params: 181
Non-trainable params: 0
```

Figure 7.21. MLP model summary

*LSTM:*

For LSTM models, the same parameter set is tuned to find the best fitted model on the SaaS sentiment intensity data. The LSTM model with the lowest RMSE score is selected for further evaluation.

The parameters for each of the selected methods, involve a walk forward validation with single step forecasting.

Table 7.8 shows the optimal parameters of the LSTM model fitted on the sentiment intensity time series data. Similar to the MLP parameter estimation process, the LSTM models are fitted on the time series data of each SaaS service factor. Each of the fitted model use different permutations of the parameter values. Figure 7.22 shows the summary of the LSTM model.

Table 7.8. Parameters of the fitted LSTM model

| Lags | nodes | epochs | batch size | activation |
|---|---|---|---|---|
| 1 | 60 | 200 | 10 | tanh |

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 160)               103680
_________________________________________________________________
dropout_3 (Dropout)          (None, 160)               0
_________________________________________________________________
dense_5 (Dense)              (None, 160)               25760
_________________________________________________________________
dropout_4 (Dropout)          (None, 160)               0
_________________________________________________________________
dense_6 (Dense)              (None, 580)               93380
=================================================================
Total params: 222,820
Trainable params: 222,820
Non-trainable params: 0
```

Figure 7.22 LSTM model summary

For both MLP and LSTM, I have selected the ‘tanh’ activation function due to the reason that, by default it works with input and output ranges of [-1, 1] which fits best for our problem.

### 7.4.5 Model evaluation

During the model evaluation stage of the proposed methodology, the performance of the fitted models from the previous stage with optimal parameters are evaluated on test data. As discussed in the previous section, during the parameter estimation stage, the models are fitted on the training dataset

which consists of 70% of the time series dataset prepared for the models in this chapter. Each selected forecast models are evaluated in four settings as follows:

1. single-step forecasts with univariate series
2. multi-steps forecast with univariate series
3. single-step forecasts with multivariate series
4. multi-step forecasts with multivariate series

Apart from the Exponential smoothing models, the rest of the fitted models are evaluated in both univariate and multivariate settings.

A comparative analysis of the selected models based on their forecast performances is conducted at the end of this stage. The comparative analysis helps to identify the best forecast model for the trust management framework for SaaS products.

In the next section, I present the implementation details of each stage of the proposed methodology along with relevant results.

## 7.5 EXPERIMENTS AND EVALUATION RESULTS

In this section I discuss the details of the in-depth results during the experiments. I discuss the initial form of the dataset used to develop a forecast model for the sentiment intensities of each SaaS service factor and the details of the approach to transform the dataset into regular time series. This is followed by the implementation and results of the data analysis methods used to study the sentiment intensity time series dataset. The details of the parameters estimated for each approach is discussed with the summary of the fitted model in a model estimation subsection. At the end of this section, the forecast results from each of the selected model is presented and a comparative analysis is performed to identify the best performing forecast model.

### 7.5.1 Data preparation and Pre-processing

As discussed in Section 7.4.1, the time series forecasting approaches work with regular time series data where the values are observed at constant intervals throughout the series. The dataset used int this chapter consist of sentiment intensity scores observed at irregular time stamps. Converting the irregular sentiment intensity series brought sparsity in the series with missing values. This sparsity is imputed by linear interpolation using Equation 7.34. The time series for the SaaS service factors obtained after the linear interpolation process is depicted in Figure 7.5. The detailed descriptive

statistics of the time series data used in this chapter is shown in Table 7.9. The mean statistics for each service factor time series is not similar to each other. The mean and median of each series show less skewed for most of the service factors. As discussed previously, each series contains the sentiment intensity scores between the values of -1 and 1. The statistics from Table 7.9 for mean, min and max shows that that most sentiment scores are positive and towards the maximum statistics. The count indicates daily observations of 1933 days from 23-3-2012 to 7-7-2017 (approximately above 5 years and 3 months) for the selected SaaS product.

Table 7.9. SaaS sentiment intensity time series data description

| Factor | Count | Mean | Std | Min | 25% | 50% | 75% | Max |
|---|---|---|---|---|---|---|---|---|
| **A** | 1933 | 0.4294 | 0.2892 | -0.6249 | 0.2300 | 0.4787 | 0.6449 | 0.9607 |
| **C** | 1933 | 0.4715 | 0.2727 | -0.5312 | 0.2997 | 0.5217 | 0.6808 | 0.9805 |
| **E** | 1933 | 0.7536 | 0.2594 | 0.3400 | 0.3400 | 0.9198 | 0.9198 | 0.9198 |
| **L** | 1933 | 0.3404 | 0.2134 | -0.3595 | 0.2732 | 0.3550 | 0.4812 | 0.8779 |
| **M** | 1933 | 0.4646 | 0.2420 | -0.7003 | 0.3445 | 0.4988 | 0.6354 | 0.9432 |
| **N** | 1933 | 0.3549 | 0.2140 | -0.5994 | 0.2048 | 0.3685 | 0.5140 | 0.9633 |
| **O** | 1933 | 0.0337 | 0.3176 | -0.6597 | -0.2253 | 0.0000 | 0.2576 | 0.9459 |
| **P** | 1933 | 0.3965 | 0.2715 | -0.7146 | 0.2010 | 0.4044 | 0.6154 | 0.9632 |
| **R** | 1933 | 0.4180 | 0.2259 | -0.5093 | 0.2940 | 0.4252 | 0.5756 | 0.9270 |
| **S** | 1933 | 0.6487 | 0.1586 | 0.3400 | 0.5155 | 0.6689 | 0.7881 | 0.8807 |

### 7.5.2 Model evaluation and comparative analysis

In this section, the fitted models using the selected approaches are evaluated on their forecasting capabilities and the results are cross-compared.

Table 7.10. Single-step forecast errors

| | ARIMA | | ES | | MLP | | LSTM | |
|---|---|---|---|---|---|---|---|---|
| | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** |
| **A** | 0.1025 | 0.0526 | 0.0978 | 0.0397 | 0.1015 | 0.0601 | 0.0999 | 0.0558 |
| **C** | 0.0865 | 0.0588 | 0.0771 | 0.0384 | 0.0835 | 0.0531 | 0.0910 | 0.0638 |
| **E** | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0030 | 0.0030 | 0.0404 | 0.0404 |
| **L** | 0.0002 | 0.0002 | 0.0000 | 0.0000 | 0.0046 | 0.0045 | 0.0178 | 0.0178 |
| **M** | 0.1410 | 0.1052 | 0.1128 | 0.0550 | 0.1136 | 0.0724 | 0.1135 | 0.0720 |
| **N** | 0.1223 | 0.0629 | 0.1237 | 0.0689 | 0.1176 | 0.0753 | 0.1203 | 0.0802 |
| **O** | 0.0357 | 0.0218 | 0.0355 | 0.0212 | 0.0430 | 0.0327 | 0.0379 | 0.0246 |
| **P** | 0.1167 | 0.0869 | 0.0949 | 0.0507 | 0.0976 | 0.0632 | 0.0959 | 0.0622 |
| **R** | 0.1099 | 0.0602 | 0.1070 | 0.0567 | 0.1051 | 0.0638 | 0.1043 | 0.0606 |
| **S** | 0.0051 | 0.0022 | 0.0021 | 0.0002 | 0.0456 | 0.0390 | 0.0289 | 0.0238 |

The single-step forecast errors for the selected models for all the service factors are shown in Table 7.10. The RMSE scores for the ARIMA model are at acceptable level when compared with RMSE scores of the other models. The RMSE scores for the service factors E, L and S are significantly better compared to the other service factors. The main reason behind this low RMSE scores on the time series for these service factors is that, the number of actual observations for these factors were significantly lesser than the observations in the other factors. The interpolation process on the initial series specifically for service factors E, L and S, resulted in a well-defined pattern of distribution. The ARIMA and the Exponential smoothing models captured these patterns in high precision with lowest errors compared to MLP and LSTM. As explained in earlier, due to the interpolation on the initial dataset, the series for the service factors E, L and S showed additive trend. Hence, the Holt linear trend model is fitted on these three series which captured the trend component with the lowest error rates. However, for the rest of the service factors, the forecast performance of MLP and LSTM are relatively better than then ARIMA and ES models.

Table 7.11. Multi-steps forecast errors

| | ARIMA | | ES | | MLP | | LSTM | |
|---|---|---|---|---|---|---|---|---|
| | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** | **RMSE** | **MAE** |
| **A** | 0.2851 | 0.2421 | 0.2643 | 0.2106 | 0.2699 | 0.2257 | 0.2709 | 0.2265 |
| **C** | 0.2625 | 0.2244 | 0.5776 | 0.5190 | 0.2521 | 0.2130 | 0.2532 | 0.2142 |
| **E** | 0.0755 | 0.0729 | 0.0000 | 0.0000 | 0.0043 | 0.0033 | 0.0058 | 0.0042 |
| **L** | 0.1761 | 0.1557 | 0.0329 | 0.0175 | 0.0605 | 0.0542 | 0.0720 | 0.0707 |
| **M** | 0.2200 | 0.1716 | 0.4393 | 0.3969 | 0.2210 | 0.1720 | 0.2212 | 0.1711 |
| **N** | 0.2006 | 0.1618 | 0.2198 | 0.1723 | 0.2007 | 0.1636 | 0.1942 | 0.1579 |
| **O** | 0.2563 | 0.1957 | 0.6555 | 0.5971 | 0.2372 | 0.1935 | 0.2395 | 0.1925 |
| **P** | 0.2845 | 0.2494 | 0.5056 | 0.4379 | 0.2746 | 0.2381 | 0.2756 | 0.2388 |
| **R** | 0.2255 | 0.1755 | 0.2698 | 0.2298 | 0.2250 | 0.1719 | 0.2240 | 0.1707 |
| **S** | 0.1277 | 0.0925 | 0.2523 | 0.1608 | 0.1840 | 0.1616 | 0.1802 | 0.1591 |

Table 7.11 show the multi-step forecast error results of each service factor by the selected univariate models. The RMSE scores for the multi-steps forecast by the ES models shows relatively poor forecast performance compare to the others. The MLP and LSTM models show similar errors on each series.

The average forecast errors on all the service factors for the selected univariate models is shown in Table 7.12. As discussed in the previous sections, the error scores are on the same scale as the sentiment intensity scores that ranges from -1 to 1. Therefore, a fractional difference in the forecast errors by the models is considered for the selection of best performing model.

Table 7.12. Average forecast errors

| | Single-step | | Multi-steps | |
|---|---|---|---|---|
| | RMSE | MAE | RMSE | MAE |
| ARIMA | 0.09 | 0.05 | 0.22 | 0.17 |
| ES | 0.08 | 0.03 | 0.38 | 0.27 |
| MLP | 0.08 | 0.05 | 0.21 | 0.16 |
| LSTM | 0.08 | 0.05 | 0.21 | 0.16 |

Table 7.12 shows that in single-step forecast settings, the ARIMA models show the highest errors compared to the other models. The average forecast errors for ES, MLP and LSTM model are similar in the single-step forecast setting. In multi-steps forecasting, the performance of the MLP and LSTM models are similar and better than the ARIMA and ES models with lower error rates. Figure 7.23 depicts the LSTM model single-step forecasts along with the actual observed sentiment intensity values. The forecasted values are promising when considering the amount of noise in the data.

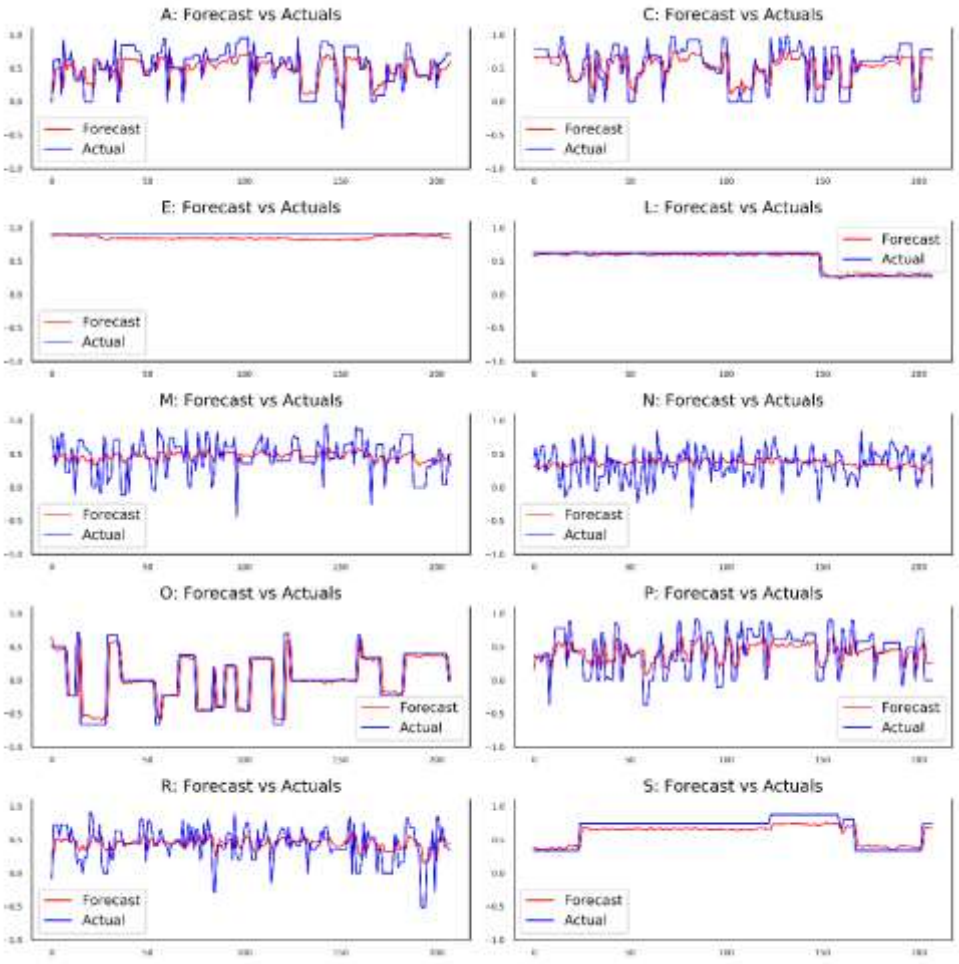



Figure 7.23 LSTM Single-step (Actual vs. Forecast)

The multi-steps forecasts by the LSTM model depicted in Figure 7.24 are significantly better than the forecast by ARIMA and ES models. From the plots for each service factor, the multi-step forecast can be does follow the patterns to a considerable extent. This makes the LSTM model ideal for multi-steps forecasting on SaaS sentiment intensity data.

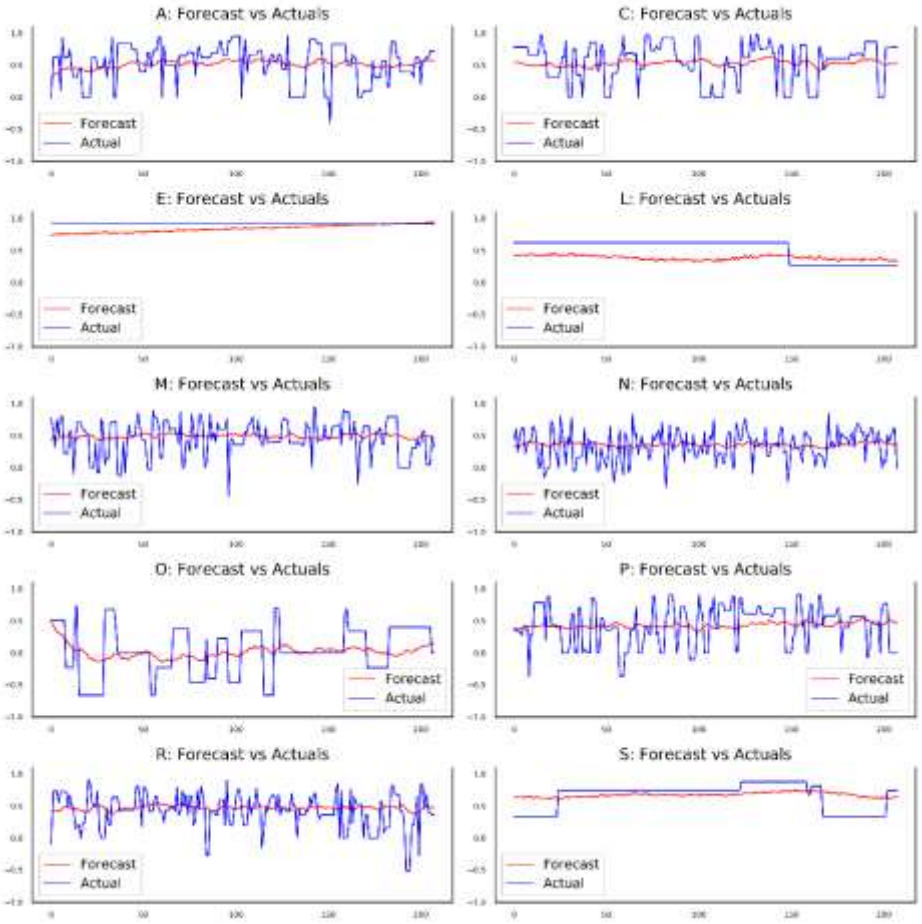



Figure 7.24 LSTM Multi-steps (Actual vs. Forecast)

The forecast results in this section so far are from the models fitted on the univariate series of each service factor. As discussed in section 7.4.3, the statistical tests on the series for each service factors suggest causality and cross-correlations between some of the service factors. Therefore, I have also fitted the selected models apart from the ES models, on the multivariate series as well. The multivariate models are fitted on the combined series in vector form. Figure 7.25 depicts the forecast error comparison among the selected multivariate forecast models. The plot on the left-hand side in Figure 7.25 shows the RMSE scores on the single-step forecasts. The plot shows that the VARMA model performs better on the service factors E, L and S with lower RMSE scores compared to MLP and LSTM multivariate models. This low error rates by VARMA is attributed to the nature of these

high interpolated series. However, for the rest of the service factors the VARMA model performed adversely. The plot on the right-hand side of Figure 7.25 shows the comparison among the RMSE scores for multi-steps forecasts by the selected multivariate forecast models. Apart from the RMSE score for service factor E, the VARMA model performed poorly with higher RMSE scores. The performances of multivariate MLP and LSTM models are better comparing to VARMA model with lower RMSE scores. The performance of MLP and LSTM model in both univariate and multivariate modelling are identical.

Single-step Multi-steps

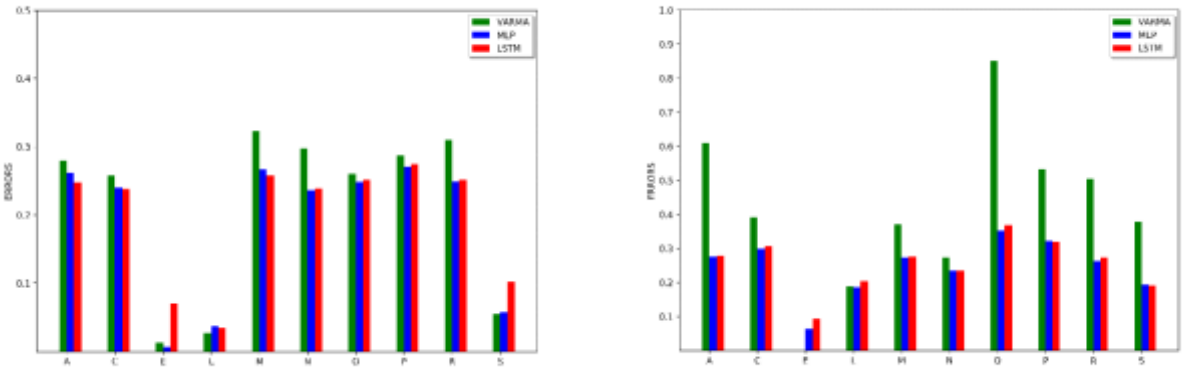



Figure 7.25 Multivariate models forecast errors

Single-step Multi-steps

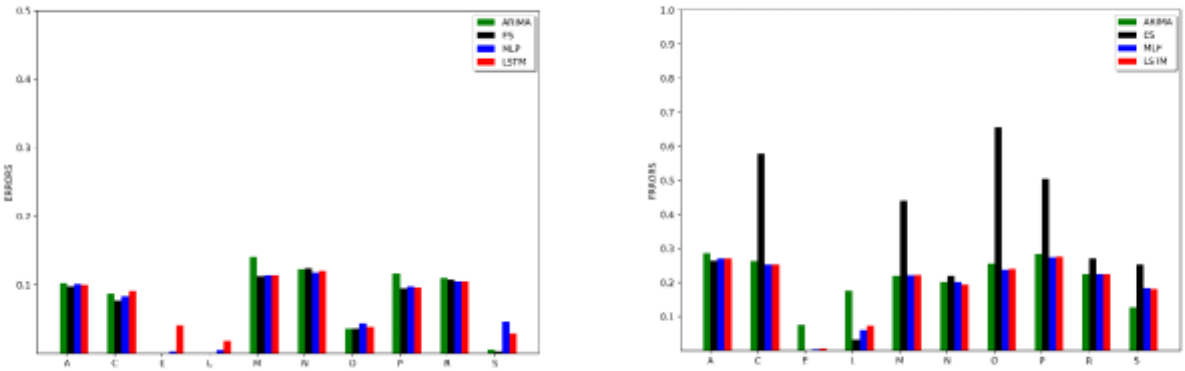



Figure 7.26 Univariate models forecast errors

The comparative analysis in terms of RMSE scores for each service factor by the selected univariate forecast models is depicted in Figure 7.26. From Figure 7.26, for the service factor E, L and S all the

forecast models have high performance with lower RMSE scores. The main reason behind this relatively lower RMSE scores for some service factors is the lesser number of observations. As discussed in the Section 7.4.3, the initial dataset used for this part of the thesis contains irregular series of sentiment intensity observations over a SaaS lifetime. To model and forecast the sentiment intensity scores of the SaaS service factors, I have converted them into a regular time series using linear interpolation. This resulted the most of the values to be imputed using linear interpolation method creating a certain pattern that the models captured with least errors. The more interpolated values a service factor contains, the easier it is to be captured by the models, hence the lower errors.

The average RMSE scores of the selected forecast models are in univariate and multivariate settings are compared in Figure 7.27. The solid-coloured boxes represent the univariate model errors while the dashed boxes represent the errors for the multivariate models. In both the single-step and multi-step forecasting, the univariate versions of the models have lower RMSE scores compared to their vector versions. The causations and cross-relations between the service factors proved less useful in modelling and forecasting the sentiment intensity data for SaaS products.

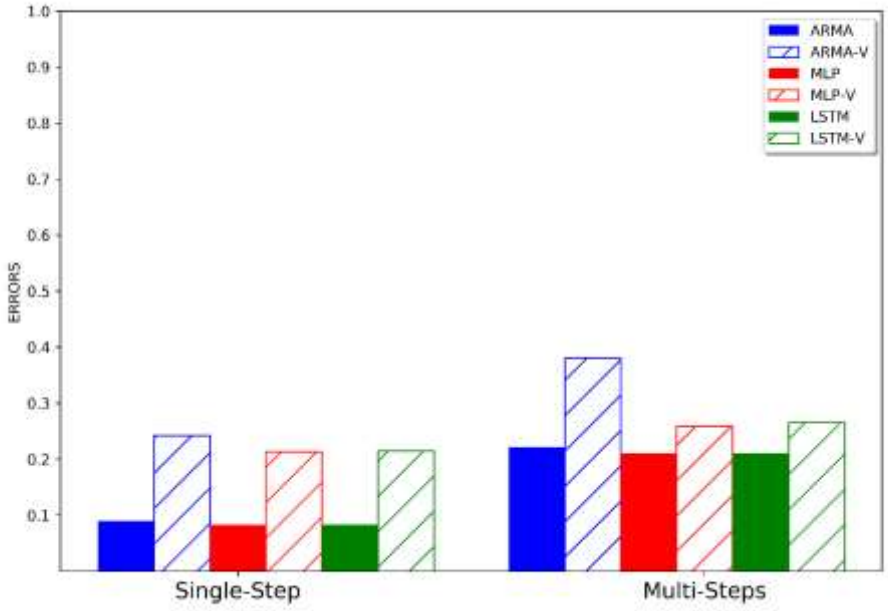
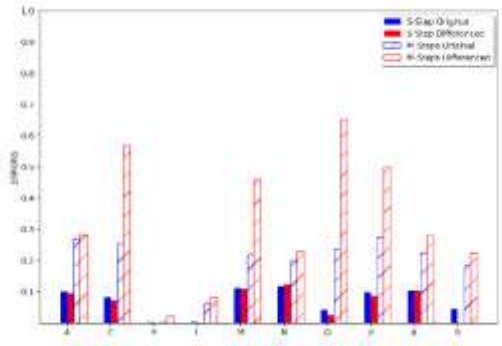



Figure 7.27 Univariate vs. Multivariate models average forecast errors

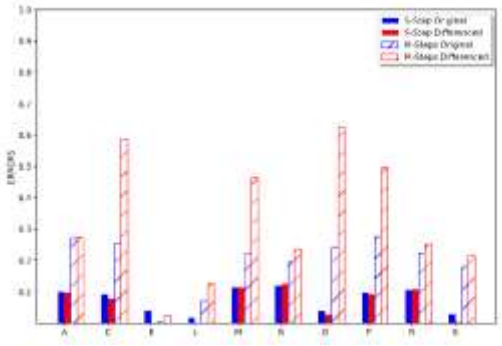
The nature of the time series forecasting models is discussed in details in the previous sections. Time series modelling approaches such as ARIMA works better with stationary series in both univariate and multivariate implementations. The same is the requirement for SES models. However, the other

forecasting methods such as MLP and LSTM are well known to work with non-stationary data. As part of model evaluation, I have fitted the MLP and LSTM models on both the stationary and non-stationary form of the same series. Figure 7.28 depicts the plots of the forecast errors by the MLP and LSTM models on stationary and non-stationary series. The solid-coloured boxes represent the RMSE scores for single-step forecast and the dashed boxes represent the RMSE scores for the multi-steps forecasts. The blue colour is used for the non-stationary (original) series and red colour is used for stationary series. For multivariate models, both the MLP and LSTM models have lower RMSE scores on original non-stationary series. This difference is highly significant in multivariate LSTM models.

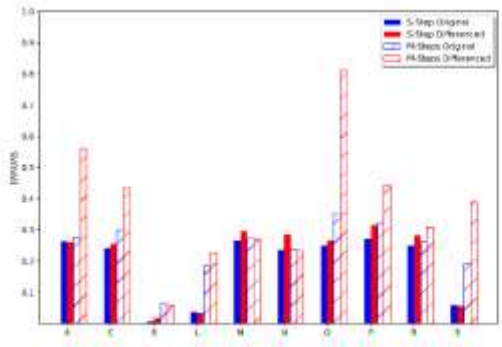
MLP Univariate



LSTM Univariate



MLP Multivariate



LSTM Multivariate

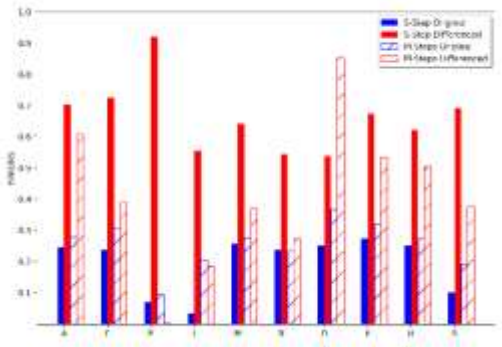


Figure 7.28 Forecast errors comparison (Actual vs. Differenced)

As mentioned earlier, the RMSE scores for the forecasting approaches are in fractions since they are on the same scale as the sentiment intensity scores that ranges between -1 and 1. In the next chapter, the trust levels and trust scores in SaaS products are computed by using the sentiment intensity scores and small fractional differences result in a different trust score.

Based on the in-depth analysis of the SaaS sentiment intensity dataset and the performance of the forecasting approaches, the performance results can be summarized as:

- Modelling each service factor separately compared to vector approach of modelling achieves lower RMSE scores by all the selected approaches in this study. Each service factor should be separately modelled as there is lack of correlation among all of them (as depicted in Table 7.3)
- Time series forecasting approaches such as MLP and LSTM achieve relatively lower error scores when used with series in original form as observed compared to stationary series.
- Time series forecasting approaches such as ARIMA and Exponential Smoothing performs better with lesser RMSE scores when used with highly interpolated series compared to MLP and LSTM.
- The performance of ARIMA model on series with higher amount of noise is worse compared to the other selected approaches in this study.
- In a univariate modelling for a single step forecast, there is no significance difference among the performances of MLP, LSTM and ES. ARIMA model's performance is relatively lesser compared to the other approaches.
- In univariate multi step forecasting, MLP and LSTM achieves lower error scores compared to ARIMA and ES approaches with ES being the worst in performance. However, there is no significance different between the performances of MLP and LSTM in the univariate multi step forecasting setup.

The key selection criteria from the forecasting models are based in the ability to forecast with least error for multi-steps in the future. Though the results in this study shows the approaches perform well in a single-step forecast settings, I select the forecasting approach with the least error scores in multi-step settings. Based on the above-mentioned criteria, the LSTM forecasting model is most suitable for the SaaS trust model. Besides the relatively higher performance of LSTM over the other approaches, it also requires less effort in data preparation such as conversion of data into stationary series. The results in this study suggest that LSTM forecasting model works better on the dataset in its actual observed format. Although, both MLP and LSTM showed similar performance based on their forecast error scores, the performance of LSTM models can be improved by including more parameters in the parameter estimation stage.

## 7.6 CONCLUSION

In this chapter, I presented a novel methodology for forecasting sentiment intensities of SaaS service factors. I used well-known time series modelling and forecasting methods such as ARIMA, Exponential smoothing, MLP and LSTM for understand the behaviour of the sentiment intensity data to develop a forecast model.

I used techniques such as linear interpolation to transform the irregular series of observations for the SaaS service factors into regular time series data. I performed statistical tests on the time series data to learn and identify its characteristics. Based on the results of these tests, I have selected the appropriate methods to model and forecast the sentient intensity series. The fitted models are evaluated for their forecast performance both in single-step and multi-steps settings. I have also implemented the multivariate versions of the selected models on the sentiment intensity data.

The forecast results by the models in different settings suggest that the classic time series forecasting models such as ARIMA and Exponential smoothing captures the highly interpolated series with lower error rates. However, MLP and LSTM have lower error rates for multi-step forecasting.

In the next chapter, I develop a trust model that dynamically computes the trust levels and scores for SaaS products by utilizing the sentiment intensity scores of each service factor.

# Chapter 8:
# A Fuzzy-based Trust model for SaaS

## 8.1 INTRODUCTION:

In this chapter, an approach is proposed for modelling the trustworthiness of the cloud services i.e., SaaS products. The service factors that are based on the pillars of well-architected frameworks (AWS, 2018) (Azure, 2018) are considered as the key trust factors that impact the trustworthiness of cloud services specifically SaaS. The proposed trust model is based on Mamdani fuzzy inference system which takes as input, the sentiment intensity scores of the SaaS products service factors discussed in Chapter 3 and 4. The output trust values obtained from the fuzzy-based trust model is both in the form of numeric values and also in linguistic terms. As discussed in section 2.2, trust is subjective in nature and its interpretation varies with the methods used in modelling. Fuzzy sets and fuzzy inference systems are also well-known in modelling the solutions rooted in subjectivity (discussed in section 2.5). Fuzzy inference systems are easy to implement and practical in nature. Therefore, in this study, a fuzzy inference system is used to model the trust in SaaS. However, fuzzy inference systems are not dynamic in nature and it is important that a fuzzy based trust model must adapt with time. A dynamic component for the fuzzy-based trust model is presented in section 8.3 that is capable to capture the changes in the service aspect preferences over time. As explained in Chapter 4, the software quality pillars in the well-architected framework (AWS, 2018) (Azure, 2018) (also termed as the service factors in this study) is considered as the key factors in the computation of the trustworthiness of SaaS products. As explained in Chapter 4, the fuzzy trust model is the final component in the framework of our proposed approach and is developed to achieve the fourth objective of this study. The proposed fuzzy trust model is capable to compute the trustworthiness levels of SaaS products on both the aggregated historic sentiment intensity scores of SaaS products (proposed in Chapter 6) and the trust levels at a given timestep based on sentiment intensity scores from all the service factors for a given SaaS product (proposed in Chapter 7). As explained in Chapter 3, to compute the trust values of a SaaS product, it is very important to identify the key factors that contribute to the trustworthiness of the SaaS and also, the relative importance of these factors.

In this chapter, I describe the components of the proposed fuzzy trust model, the detailed explanation of steps involved and the implementation of the trust model using the SaaS products sentiment intensity dataset. In the next section, I explain the concept of trust and its application in cloud services.

The overview of the propose approach for modelling trust in SaaS products is presented in Section 8.3. The details of the novel MDR weight metrics are presented in Section 8.4. I describe the details of the fuzzy-based trust model for SaaS products in Section 8.5 followed by its implementation and an example case study in Section 8.6. Section 8.7 concludes this chapter.

Table 8.1. SaaS service factors used as Trust factors for the fuzzy-based trust model

| **Label** | **Service** factor/ **Trust Factor** | **description** | **Values range** |
|---|---|---|---|
| **AWS well-architected framework** | | | |
| **C** | Cost optimization | Cost-aware system development | [-1, 1] |
| **O** | Operational excellence | Running and monitoring systems and continually improving processes and procedures to deliver business value | [-1, 1] |
| **P** | Performance efficiency | Focuses on efficient resource utilization and maintaining that efficiency as business needs evolve | [-1, 1] |
| **R** | Reliability | Quick recovery from failures to meet business and customer demands | [-1, 1] |
| **S** | Security | Protecting system and information | [-1, 1] |
| **Azure architectural framework** | | | |
| **C** | Cost | Cost management in relation to product value | [-1, 1] |
| **M** | DevOps (Management) | Process in operations that keep a system running in production | [-1, 1] |
| **E** | Resiliency | The ability to recover from failure and returning an application to fully functional state | [-1, 1] |
| **L** | Scalability | The ability to handle increased load and adapt to changes | [-1, 1] |
| **S** | Security | Application and data protection throughout the Lifecyle of the application | [-1, 1] |
| **A** | Availability (Previous version) | Measured as a percentage of uptime. The availability of software system on the cloud is described as the proportion of time that the software system is functional and working. | [-1, 1] |

## 8.2 OVERVIEW OF THE FUZZY-BASED TRUST MODEL FOR SAAS:

The overview of our proposed fuzzy-based trust model in depicted in Figure 8.1. The input variables for the model is shown in Table 8.1 which comprises of the service factors of SaaS products having the sentiment intensity scores as their values.

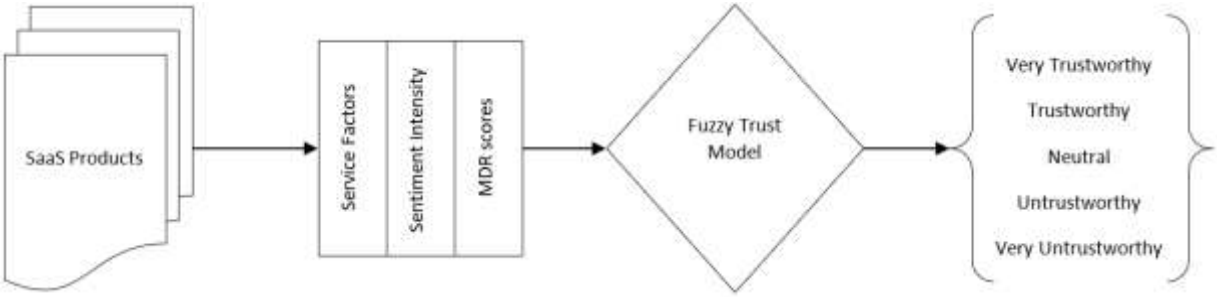



Figure 8.1 Overview of the framework for fuzzy-based trust model

Initially, the detailed information is collected regarding the SaaS products that include customer textual reviews, review dates, reviewer information. This information is kept in the core dataset explained and used in Chapter 5, Chapter 6 and Chapter 7. For the proposed solution addressed in this chapter, the core dataset is processed again to extract information particular to the problem here. The service factors are identified and extracted from the reviews using the method proposed in Chapter 5 and for each identified service factor, the sentiment intensity is computed using the solution proposed in Chapter 6. The MDR weight metrics is developed and the service factors are weighted and ranked based on their importance by using the metrics. The MDR weight metrics is explained in details in the next section. Using the MDR metrics, the fuzzy rules are generated for the fuzzy trust model. The output of the fuzzy trust model is the is a crisp trust value along with five levels of trustworthiness of the SaaS products. The detailed steps of the fuzzy-based trust model are shown in Figure 8.7.

## 8.3 MDR WEIGHT METRIC

In order to compute trust value for SaaS products, it is crucial to consider the most important service factors in this computation. Each service factor can play different role in the overall trustworthiness of a SaaS product. For example, a service factor for which a very small amount of information is available is more likely to be less helpful compared to another service factor with relatively more information. Moreover, as we are using these service factors as input variables for the proposed fuzzy trust model, the importance of each service factors is very helpful in determining the strength of fuzzy rules which will compute the five different levels of trustworthiness for the SaaS products. Generally, the ranking and selection of the input variables are left at customers choice. However, dynamically selecting weights for these input variables will reduces the overhead on customer side of decision making.

MDR stands for Maturity, Density and Reviewers of the SaaS products. The basic purpose of the MDR metrics is to identify the importance of each input variables (service factor) for the generation of fuzzy rules for our trust model using different combination of these input variables. As discussed earlier, each input variable has different level of importance in decision making and can be weighted and ranked based on these weights. This will help us to dynamically select and generate rules with proper combination of these input variables. Moreover, MDR will help in optimal ranking of the input variables by assigning them different weights. This is due to the fact that the input variables represent the SaaS service factors and contains the aggregated sentiment intensity scores over a long period which is based on reviews of multiple customers.

The main advantages of using MDR weight metrics to rank the input variables are:

- Identifies the relative importance of input variables among them
- Assign different preference level to each input variable
- Helps to reduce the biases among the input variable values
- Helps to reduce the number of the fuzzy rules in the rule base.
- An alternative to the static domain-based rulesets creation.

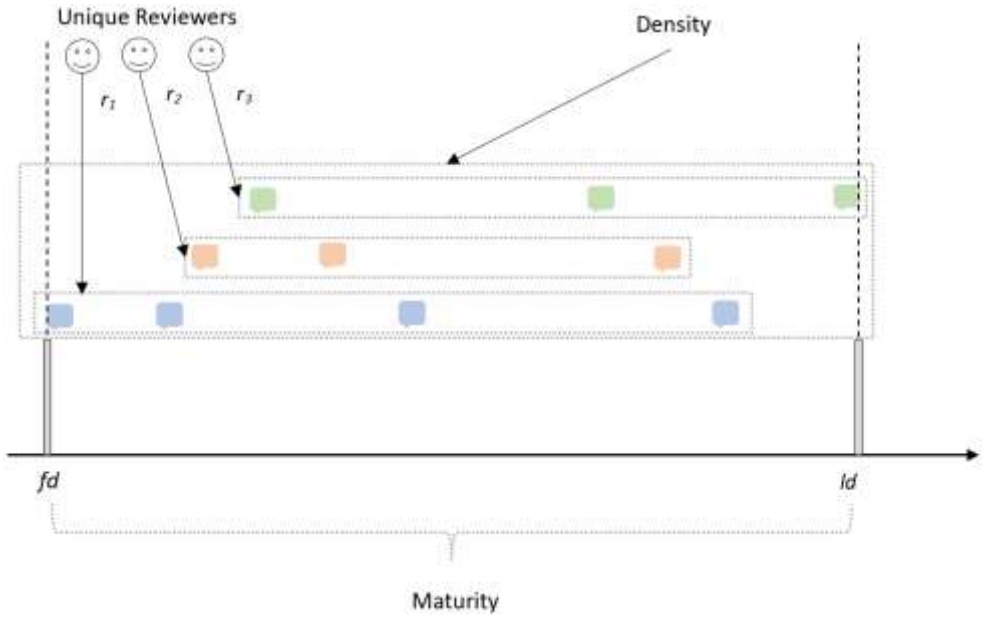



Figure 8.2 Graphical representation of the individual metric in MDR

Following are the detailed explanation and collection of MDR weight metrics.

### 8.3.1 Maturity:

Maturity is defined as the total amount of time since the first review is made for a given service factor of a SaaS product, till the latest review date. Figure 8.2 depicts the maturity of a given service factor. The unit of maturity is a solar day. For a given SaaS product, the maturity for each identified service factor can be different. For example, reviewers might be providing their written feedbacks for the 'Cost' service factor for the last 400 days and compared to that, for the 'Performance' factor of the same SaaS product, the reviews are made just in the past 30 days. In this scenario, the information regarding the 'Cost' service factor is less likely to give a biased view of the SaaS product compared to the 'Performance' service factor. This makes even more sense when the information regarding each service factor is based on the sentiment intensity scores. For a given SaaS product, when an aggregated sentiment intensity scores for each service factor is calculated, the service factor with high maturity will produce relatively less bias and can be preferred over the others during the decision-making process. During the calculation of the MDR scores for each service factor, we are using an average maturity score over 542 SaaS products. The average maturity scores for each service factor are depicted in Table 8.6. If $P$ represents $n$ SaaS products $P = \{p_1, p_2, p_3, ...., p_n\}$ and $A$ represents $m$ service factors $A = \{a_1, a_2, a_3, ...., a_m\}$, then the maturity value for the $j^{th}$ service factor of the $i^{th}$ SaaS product is calculated as:

$$m_{ij} = fd_{ij} - ld_{ij} \tag{8.1}$$

where $fd_{ij}$ represent the first review date for the $j^{th}$ service factor of the $i^{th}$ SaaS product and $ld_{ij}$ represent the last recorded review date for the $j^{th}$ service factor of the $i^{th}$ SaaS product. The average maturity of service factor $j$ for all the SaaS products in the dataset $X$ is calculated as:

$$M_j = \frac{\sum_{i=1}^{n} m_{ij}}{n} \tag{8.2}$$

where n represents the total number of SaaS products in the dataset.

### 8.3.2 Density:

The Density metric is defined as the total number of reviews since the first review for a given service factor, till the latest review date as depicted in Figure 8.2. For a given service factor, the value for the density metric is increment by 1 each time a review is made for that service factor. The density for each service factor of a given SaaS product can be different. For example, for a given SaaS product P1000, the reviewers have provided 100 written feedbacks regarding the service factor 'Cost' since the first feedback date for this service factor. While for the same SaaS product, there are 20 written

feedbacks for the 'Performance' service factor. In the above example, the information regarding the 'Cost' service factor is relatively less likely to lead to a biased decision compared to the information from the 'Performance' service factor. Also, service factor with higher density will be preferred in the decision-making especially when modelling the trustworthiness of the SaaS product. In this study, during the calculation of the MDR scores, we are using an average density score for each service factor over 542 SaaS products. Figure 8.2 shows the average density of each service factor. The density value for the $j^{th}$ service factor of the $i^{th}$ SaaS product is calculated as:

$$d_{ij} = tr_{ij} \tag{8.3}$$

where $tr_{ij}$ is the total number of reviews for the $j^{th}$ service factor of the $i^{th}$ SaaS product between the first review date $fd_i$ and the last recorded review date $ld_i$. The average density of service factor $j$ for all the SaaS products in the dataset $X$ is calculated as:

$$D_j = \frac{\sum_{i=1}^{n} d_{ij}}{n} \tag{8.4}$$

where $n$ represents the total number of SaaS products in the dataset.

### 8.3.3 Reviewers:

The Reviewers or unique reviewers is defined as the total number of unique reviewers for a given service factor as depicted in Figure 8.2. The value of Reviewers metric is incremented by 1 each time a unique reviewer provides written feedback for a given service factor. The Reviewers metric helps to reduces the bias in calculating the weights using the MDR which is not possible to capture by the other two metrics. For example, there are 30 written feedbacks regarding the 'Cost' service factor of a given SaaS product, by 10 unique reviewers. For the same SaaS product, 30 feedbacks are written for the 'Performance' service factor but in this case by 3 unique reviewers. In the above example, the feedback information regarding the 'Cost' service factor is relatively less likely to be biased compared to the information provided by the reviewers regarding the 'Performance' service factor. Furthermore, during the decision-making process the service factor with higher number of unique reviewers will be preferred as compared to the others. The average number of reviewers for each service factor is shown in Table 8.6. For the calculation of the final MDR scores for each service factor, we have used the average number of unique reviewers over 542 SaaS products. Let $U$ be the set of unique reviewers $\{rw_1, rw_2, \ldots, rw_n\}$ for the $j^{th}$ service factor of the $i^{th}$ SaaS product, then the number of unique reviewers is given by:

$$r_{ij} = |U| \tag{8.5}$$

The average number of unique reviewers of service factor *j* for all the SaaS products in the dataset *X* is calculated as:

$$R_j = \frac{\sum_{i=1}^{n} r_{ij}}{n} \tag{8.6}$$

where *n* represents the total number of SaaS products in the dataset.

The MDR score is calculated as the weighted sum of the normalized values of maturity, density and unique reviewers. If *M* represents the maturity values for *n* service factors *{M1, M2, …., Mn},* then the normalized maturity value $\acute{M}$ for the service factor *j* is calculated as:

$$\acute{M}_j = \frac{M_j - \min(M)}{\max(M) - \min(M)} \tag{8.7}$$

Similarly, if *D* represents the density values for *n* service factors *{D1, D2, …., Dn} and R* represents the unique reviewers for *n* service factors *{R1, R2, …., Rn},* then for the service factor *j*, the normalized density value $\acute{D}$ and normalized unique reviewer's value $\acute{R}$ is calculated as:

$$\acute{D}_j = \frac{D_j - \min(D)}{\max(D) - \min(D)} \tag{8.8}$$

$$\acute{R}_j = \frac{R_j - \min(R)}{\max(R) - \min(R)} \tag{8.9}$$

Finally, the MDR score for service factor *j* is calculated as:

$$MDR_j = \acute{M}_j \times w_m + \acute{D}_j \times w_d + \acute{R}_j \times w_r \tag{8.10}$$

where, $w_m$, $w_d$ and $w_r$ represent the weights for the metrics and is assigned uniformly an equal value of 0.33. This gives an equal importance to each metric in the final MDR score for a given SaaS service factor. However, these weights can be assigned dynamically according to user preferences. The MDR scores calculated for each service factor is depicted in Figure 8.2 and Table 8.6. Algorithm 8.1 shows the steps by step procedure of the MDR weights calculation.

**Algorithm 8.1: MDR_scores(*X*)**

**Input:** Dataset *X* having the sentiment intensity scores for service factors *A* of SaaS products *P*
**Output:** Ranked MDR scores for each service factor
Let *A* be a set of m service factors, $A = \{a_1, a_2, \ldots, a_m\}$
Let *P* be a set of *n* SaaS products, $P = \{p_1, p_2, \ldots, p_n\}$
for each $a \in A$ do:
  for each $p \in P$ do:
    *Total_maturity += calculate_Maturity($p_a$[last_review_date], $p_a$[first_review_date])*
    *Total_density += review_Count($p_a$[last_review_date], $p_a$[first_review_date])*
    *Total_Unique_reviewers += unique_Reviewers($p_a$[last_review_date], $p_a$[first_review_date])*
  end for
  *Average_Maturity_of[a] = Total_maturity/n*
  *Average_Density_of[a] = Total_density /n*
  *Average_Reviewers_of[a] = Total_Unique_reviewers/n*
  *Scaled_Maturity = Min-Max_Scale(Average_Maturity_of[a])*
  *Scaled Density = Min-Max_Scale(Average_Density_of[a])*
  *Scaled_Reviewers = Min-Max_Scale(Average_Reviewers_of[a])*
  $W_M = W_D = W_R$ = *assign_Uniform_Weights(1/3)*
  *MDR[a] = Scaled_Maturity* $W_M$ *+ Scaled Density* $W_D$ *+ Scaled_Reviewers* $W_R$
end for
*Ranked_Service_Factors[] = sort(A, MDR(A))*

## 8.4 FUZZY-BASED TRUST MODEL:

In order to compute the trust values and trustworthiness levels for the SaaS products, the fuzzy inference system takes as input the sentiment intensity scores for all the SaaS service factors. The core concept of a fuzzy inference system (FIS) is to infer the values of fuzzy output variables from the given input variables with the help of predefined inference rules. FIS is based on the concept of the Fuzzy sets theory which were introduced by (Zadeh, 1965).

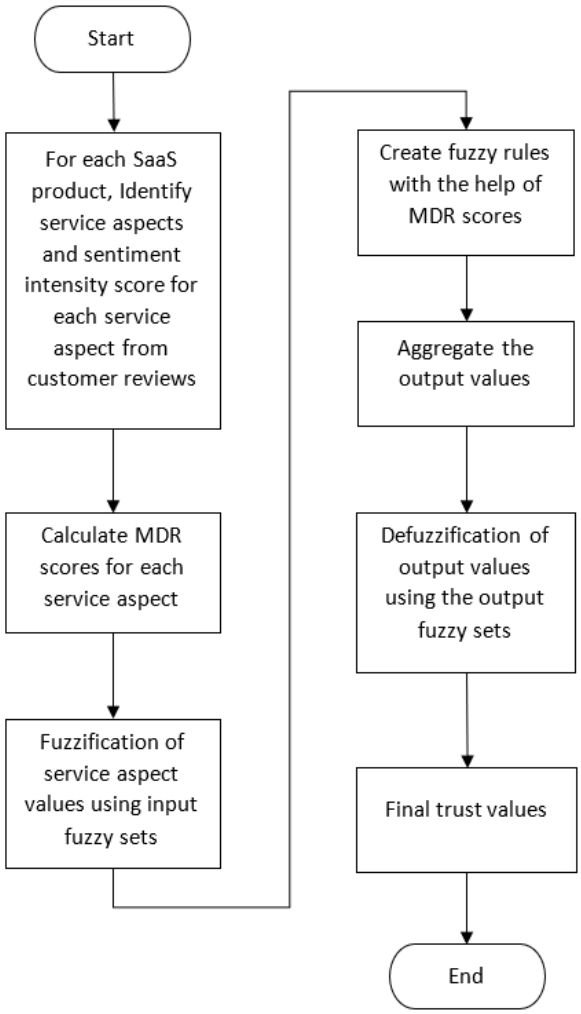



Figure 8.3 Flowchart for the sequence of tasks in fuzzy-based trust model

Compared to the classical set theory in which an element can either be a member (entirely contained having a value of 1) of a given set or not be a member (not contained having a value of 0), the elements in a fuzzy set can have same or different degree of membership between the values of 0 and 1. The degree of membership is followed by the fuzzy sets for both inputs and output. The fuzzy sets are defined using membership functions. For example, a fuzzy set $A$ defined through a membership function $\mu_A$ that assigns the degree of membership to a given input element $x$ can be written as:

$$A: \mu_A(x) \in [0, 1] \tag{8.11}$$

The operations on the fuzzy sets are facilitated by using the fuzzy operators which are similar to the standard Boolean operations such as AND, OR and NOT. Figure 8.4 shows the main components of the fuzzy inference system.

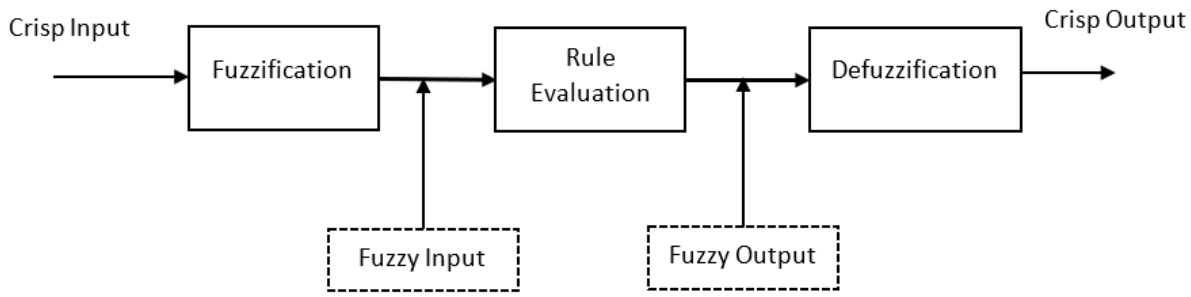



Figure 8.4 Main component of a Fuzzy Inference System

One of the most famous and extensively used fuzzy inference system is proposed by (Mamdani & Assilian, 1975). In this study, the proposed fuzzy trust model is developed using the Mamdani fuzzy inference system (Mamdani & Assilian, 1975). Mamdani inference system is widely accepted due to its intuitiveness and is exceptionally suited to human input. Figure 8.3 shows the flow of steps for the proposed fuzzy inference system. Initially the system collects the sentiment intensity scores for each SaaS factors (outcomes from chapter 5 and chapter 6). These service factors act as the trust factors in the trust model and is used as inputs for the fuzzy inference system. For each input variable, using the triangular membership function, three fuzzy sets are introduced which fuzzifies the crisp values into fuzzy values (between the values of 0 and 1). The fuzzy rules are generated using the MDR weight metrics. The fuzzy rules are then applied to the fuzzified inputs and the rule strengths are calculated. The rule strengths are aggregated to get the fuzzy output and defuzzified using the centroid defuzzification method. The final defuzzified value represents the overall trust value of a given SaaS product. Figure 8.5 depicts the inputs and output of the Mamdani fuzzy inference system. The inputs for the fuzzy inference system are the trust factors explained in Section 8.3.

### 8.4.1 SaaS service factors as Trust factors

The SaaS service factors which is based on the cloud services and application quality pillars (AWS, 2018) (Azure, 2018) is considered as the Trust factors for the proposed fuzzy trust model. Table 8.1 shows the complete list of the service factors. Chapter 5 and chapter 6 which addresses the first and second objective of this study, identifies these service factors for the SaaS products from the customer reviews. Furthermore, for each of the identified service factor, the sentiment intensity scores are also computed.

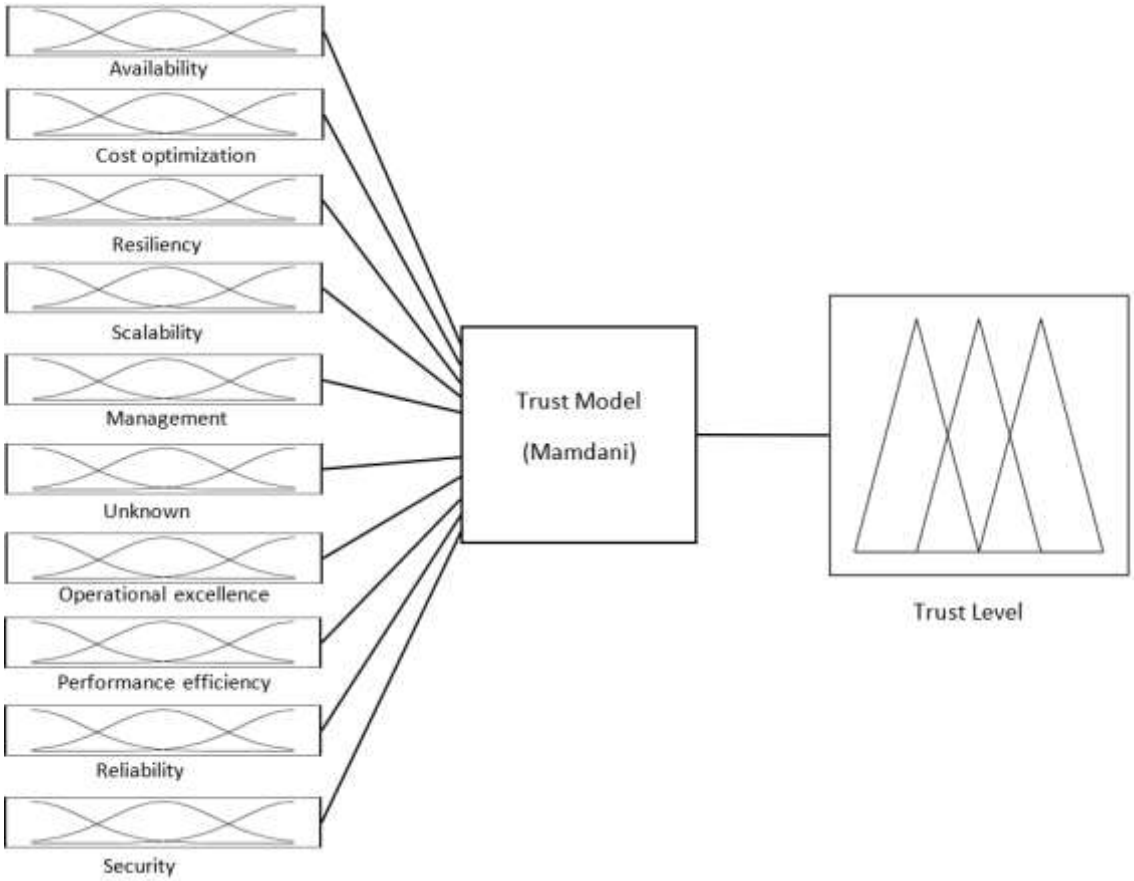



Figure 8.5 The input and output of Mamdani FIS

### 8.4.2 Fuzzification of input variables and input membership functions

The Second step in the implementation of the fuzzy trust model is to convert these crisp input values into fuzzy inputs known as the fuzzification. The fuzzification is achieved using membership functions. The membership functions define the membership degree of an input to the fuzzy sets. In this study we have used the most commonly used triangular membership function. Among the others, the triangular fuzzifier is simpler to compute and is famous for its tolerance against disruptions.

The proposed trust model in this study uses the sentiment intensity scores for each service factor as antecedents or inputs. The sentiment intensity scores range between -1 to 1 and are crisp values, where -1 represents a strongly negative customer sentiment and 1 represents a strongly positive customer sentiment. Hence the universe of discourse for each of the input variable also known as the universal set is written as:

$$U = \{r | -1 \leq r \leq 1; r \in \mathbb{R}\} \quad (8.12)$$

For each of the antecedents, three fuzzy sets named as Negative, Neutral and Positive is defined using triangular membership functions as shown in Figure 8.6, that covers the entire domain of each our

input variables. A triangular membership function takes three parameters (a, b, c) to specify its three vertices. The details of each input fuzzy set generated by the triangular membership function and its parameters is shown in Table 8.2. For example, using a triangular membership function, the degree of membership in the fuzzy set *Neutral* is calculated as:

$$\mu_{Neutral}(x) = \begin{cases} 0, & x \leq a \\ \dfrac{x-a}{b-a}, & a < x \leq b \\ 1, & x = b \\ \dfrac{c-x}{c-b}, & b < x < c \\ 0, & x \geq c \end{cases} \tag{8.13}$$

where $\mu_{Neutral}$ is the membership in the fuzzy set *Neutral* of crisp input *x*.

Table 8.2. Specifications of the triangular input fuzzy sets with their linguistic terms

| **Linguistic terms** | **Triangular membership function parameters (a, b, c) (Sentiment intensity range)** |
|---|---|
| Negative | (-1, -1, 0) |
| Neutral | (-0.25, 0, 0.25) |
| Positive | (0, 1, 1) |

Table 8.2 shows the fuzzy sets identified by their linguistic terms each with the associated range of the sentiment intensity scores. As a result of the fuzzification of the values in a given input variable using the membership functions, the fuzzy sets for that input variable will have a degree of membership between 0 and 1.

The outputs or consequents of our fuzzy inference system is defined by using five fuzzy sets namely, Very Trustworthy, Trustworthy, Neutral, Untrustworthy and Very Untrustworthy generated by using the triangular membership function. The consequents in here represent the trust levels in linguistic terms as shown in Table 8.3. Figure 8.7 shows the triangular membership functions with the associated range of the trust scores.

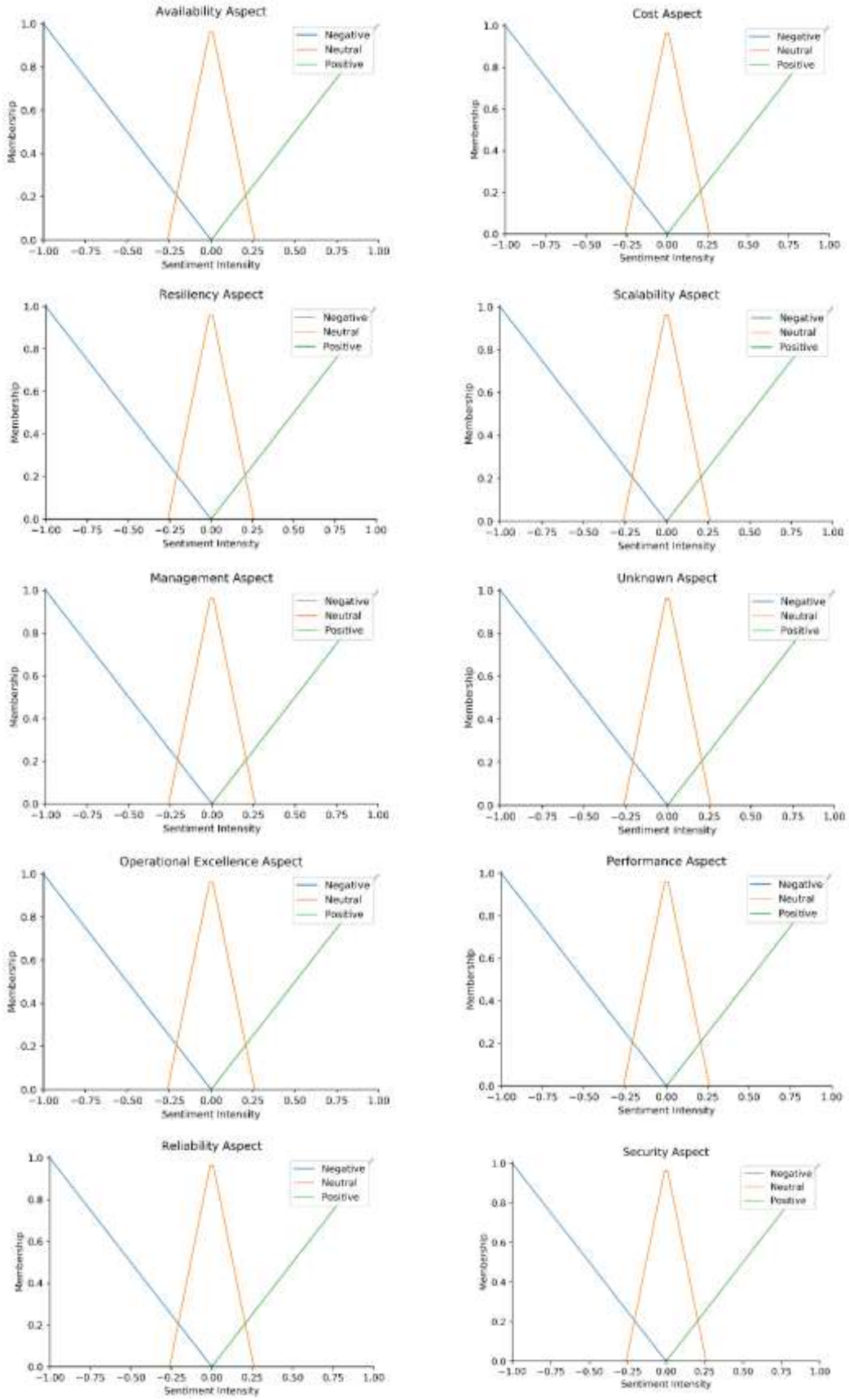



Figure 8.6 Input membership functions for all the input variables

Table 8.3. Specifications of the triangular output fuzzy sets with their linguistic terms

| **Linguistic terms** | **Triangular membership function parameters (a, b, c) (Trust values range)** |
|---|---|
| Very Untrustworthy | (1, 1, 2) |
| Untrustworthy | (1, 2, 3) |
| Neutral | (2, 3, 4) |
| Trustworthy | (3, 4, 5) |
| Very Trustworthy | (4, 5, 5) |

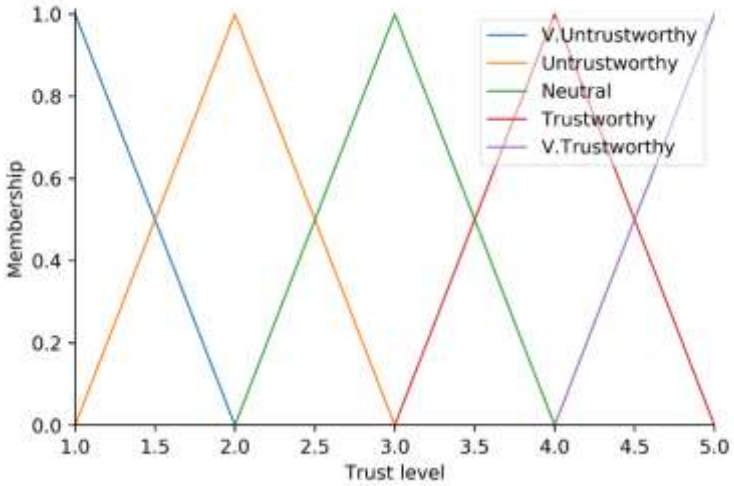



Figure 8.7 Output membership functions for the fuzzy-based trust model

### 8.4.3 Fuzzy inference rules for the trust model

The fuzzy inference rules take the *IF-THEN* form where, the *IF* part checks the fuzzy values(s) in the given input membership function(s) (antecedent(s)) and *THEN* part directs it to a given output membership function (consequent). A fuzzy rule uses fuzzy operators such as *AND, OR* and *NOT* when more than one condition is considered. The operations by these fuzzy operators are similar to the ones used in Boolean operation. A simple example of a fuzzy rule can be written as:

$$IF\ x_1\ IS\ A_1\ AND\ x_2\ IS\ A_2\ THEN\ y\ IS\ B$$

where, $x_1$ and $x_2$ are input values, $A_1$ and $A_2$ are the linguistic values (fuzzy sets) for the input values, $y$ is the output variable and $B$ represent the linguistic value of $y$.

In order to devise the most effective fuzzy rules for the trust model we have ranked the input variable based on their importance. The importance of these input variables is computed using our proposed *MDR* weight metrics. A set of fuzzy rules is generated with the help of *MDR* metrics as shown in Table 8.4. For clarity, the remaining fuzzy rules are placed in Section 8.4.1. The fuzzy rules are

organized into five different categories based on specific combinations of the input variables weights. The following subsections explain the generation of rules in each of the five rule categories.

Table 8.4. Fuzzy inference rules with MDR scores for each Trust factor

| | | SaaS service aspects (input variables) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MDR | 1.00 | 0.95 | 0.37 | 0.36 | 0.32 | 0.29 | 0.22 | 0.04 | 0.03 | 0.00 | |
| | Rule | Unknown is | Management is | Performance is | Reliability is | Cost is | Operational Excellence is | Availability is | Scalability is | Security is | Resiliency is | Then Trust level is |
| Category A | 1 | Negative | Negative | Negative | Negative | - | - | - | - | - | - | Very Untrustworthy |
| | 2 | Neutral | Neutral | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 3 | Positive | Positive | Positive | Positive | - | - | - | - | - | - | Very Trustworthy |
| Category B | 4 | Negative | Negative | Negative | - | - | - | - | - | - | - | Untrustworthy |
| | 5 | Neutral | Neutral | Neutral | - | - | - | - | - | - | - | Neutral |
| | 6 | Positive | Positive | Positive | - | - | - | - | - | - | - | Trustworthy |
| | 7 | Negative | Negative | - | Negative | - | - | - | - | - | - | Untrustworthy |
| | 8 | Neutral | Neutral | - | Neutral | - | - | - | - | - | - | Neutral |
| | 9 | Positive | Positive | - | Positive | - | - | - | - | - | - | Trustworthy |
| | 10 | Negative | - | Negative | Negative | - | - | - | - | - | - | Untrustworthy |
| | 11 | Neutral | - | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 12 | Positive | - | Positive | Positive | - | - | - | - | - | - | Trustworthy |
| | 13 | - | Negative | Negative | Negative | - | - | - | - | - | - | Untrustworthy |
| | 14 | - | Neutral | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 15 | - | Positive | Positive | Positive | - | - | - | - | - | - | Trustworthy |
| Category C | 16 | Positive | Positive | Neutral | Neutral | - | - | - | - | - | - | Trustworthy |
| | 17 | Positive | Neutral | Positive | Neutral | - | - | - | - | - | - | Trustworthy |
| | 18 | Positive | Neutral | Neutral | Positive | - | - | - | - | - | - | Trustworthy |
| | 19 | Neutral | Positive | Positive | Neutral | - | - | - | - | - | - | Trustworthy |
| | 20 | Neutral | Positive | Neutral | Positive | - | - | - | - | - | - | Trustworthy |
| | 21 | Neutral | Neutral | Positive | Positive | - | - | - | - | - | - | Trustworthy |
| | 22 | Negative | Negative | Neutral | Neutral | - | - | - | - | - | - | Untrustworthy |
| | 23 | Negative | Neutral | Negative | Neutral | - | - | - | - | - | - | Untrustworthy |
| | 24 | Negative | Neutral | Neutral | Negative | - | - | - | - | - | - | Untrustworthy |
| | 25 | Neutral | Negative | Negative | Neutral | - | - | - | - | - | - | Untrustworthy |
| | 26 | Neutral | Negative | Neutral | Negative | - | - | - | - | - | - | Untrustworthy |
| | 27 | Neutral | Neutral | Negative | Negative | - | - | - | - | - | - | Untrustworthy |

***Rule categories***

A total of 159 rules are generated with the help of MDR scores which are divided into five different categories. The high number of rules is due to the number of inputs and also based on wider range of considerations by the fuzzy inference system. The rule categories are discussed in details in the following subsections.

*Rules category A:*

In this category of rules, the selected input variables having their *MDR* scores above the average MDR score. The average MDR score is calculated as:

$$MDR_{avg} = \frac{\sum_{j=1}^{m} MDR_j}{m} \tag{8.14}$$

The above equation calculates the average MDR score, where *m* represents the total number of input variables. If *A* is the set of all the input variables (service factors) and *T* is the set of MDR values for the input variables, then the input variables for the rules in *category I* is selected as:

$$CAT_I = \{j | j \in A, MDR_j \geq MDR_{avg}\} \tag{8.15}$$

where *j* represents the input variables (service factors).

The rules in this category have no dependencies on the inputs having their MDR scores below the average threshold. The selected the most effective variables will help to target precisely the *Very Trustworthy*, *Very Untrustworthy* and *Neutral* output trust levels. For example, if in each of the selected variable, the term '*Negative*' has the highest membership value, then the rule will result into the term '*Very Untrustworthy'* (output fuzzy set). All the rules generated in this category is depicted in Table 8.4.

*Rules category B:*

The rules in *category B* leaves one variable out from the set of input variables having MDR scores above the average MDR and have no dependencies on the remaining inputs. This makes the strength of rules in this category relatively weaker than the rules in *category I*. The rules in *category II* targets the output trust levels of *Trustworthy*, *Neutral* and *Untrustworthy* represented by respective output fuzzy sets.

*Rules category C:*

Similar to *category B*, the strength of rules in category is also relatively weaker than the rules in *category I*. The rules in *category C* follows the baseline strategy of *categories A and B* by considering the input variables that have the MDR scores above the average MDR and have no dependencies on the remaining inputs. The rules in this category targets the output trust levels of *Trustworthy, Neutral* and *Untrustworthy* by restricting that, in half of the selected input variables, the term '*Neutral*' must have the highest membership value. For example, in half of the selected variables, the term '*Negative*' is checked for its membership value and in the remaining half of the input variables, the term '*Neutral*' is selected for its membership value, then the rule results in the term '*Untrustworthy*' as the output fuzzy set.

*Rules category D:*

The aim of the rules in this category is to target the '*Neutral*' trust level in the output fuzzy set. This category of rules also considers those variables that have the *MDR* scores above the average *MDR*. The rules in this category dictates that, in half of the selected input variables, the term '*Neutral*' must be checked for its membership value and half of the remaining variables above the average MDR must consider '*Positive*' or '*Negative*' with their membership value. The rules in this category have no dependencies on the remaining inputs.

*Rules category E:*

Most of the rules in our ruleset belong to this category. This rules in this category considers the combination of all the input variables in our dataset. Each rule in this category has different dependencies on the inputs. The strength of rules in category is also relatively weaker than the rules in *category I* and they target the output trust levels of *Trustworthy*, *Neutral* and *Untrustworthy*.

### 8.4.4 Rule aggregation and evaluation

In order to calculate the consequent or output of the fuzzy rules, first, for each fuzzy rule the fuzzified inputs are combined (also known as T-norms) using fuzzy operators (also known as implication operators) to identify the rule strength. The most commonly used operator is the fuzzy *AND* which is written as:

$$\mu_{A\cap B} = T\big(\mu_A(x), \mu_{\boldsymbol{B}}(x)\big) \tag{8.16}$$

where $\mu_A$ is the membership of input $x$ in class A and $\mu_B$ is the membership of input $x$ in class B.

Similarly, the fuzzy *OR* operator is written as:

$$\mu_{A\cup B} = T\big(\mu_A(x), \mu_{\boldsymbol{B}}(x)\big) \tag{8.17}$$

To calculate the rule strength by combining the fuzzy inputs, I have used the Zadeh's fuzzy *AND* operator (*min*) which computes the minimum of the given membership values given below:

$$w_i = \min\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\} \tag{8.18}$$

where $w_i$ represents the rule strength for fuzzy rule $R_i$, $x$ represents the input vector and $A_n$ represents given input fuzzy sets. Once the rule strength for each rule is calculated using the fuzzy combinations, then the rule strength is combined with the corresponding output membership function (consequent) which returns a clipped output membership function given as:

$$\mu_{\acute{B}_i}(y) = \min(w_i, \mu_{B_i}(y)) \tag{8.19}$$

where $\mu_B$ represents the output fuzzy set $B$ (consequent of rule $R_i$).

Finally, to obtain a single fuzzy output distribution, the output of all the fuzzy rules are aggregated using an aggregation operator. I have used the Zadeh's fuzzy *OR* operator (*max*) which performs the maximum operation on the output fuzzy sets.

$$\mu_{\acute{B}}(y) = \max\{\mu_{B_1}(y), \ldots, \mu_{B_n}(y)\} \tag{8.20}$$

### 8.4.5 Defuzzification to crisp trust values

During the defuzzification step, the resultant membership function obtained from the aggregated fuzzy outputs of the fuzzy rules, are converted into crisp trust values. In our proposed approach we have used the well-known centroid defuzzification method. Among the other defuzzification methods, centroid is simpler in terms of computation. Thus, a crisp output $y$ representing a trust value is computed through:

$$y = \frac{\sum_{i=1}^{n} \mu_B(x_i) \cdot S_i}{\sum_{i=1}^{n} \mu_B(x_i)} \tag{8.21}$$

where $n$ represents the number of fuzzy rules, $\mu_B(x_i)$ represents the degree of membership of value $x_i$ in class $B$ (output fuzzy set) and $S_i$ represents the centroid point of the corresponding output membership function.

## 8.5 IMPLEMENTATION AND CASE STUDY

As explained in Chapter 5, the primary data is collected for SaaS products reviews from 3 different sources and for each SaaS product, the reviews are segmented, then, from each segment the service

factors are identified and its sentiment intensity is computed (explained in chapters 5 and 6). From the core dataset, the dataset for this section of our work is created which contains the aggregated sentiment intensity values of each service factor for 542 SaaS products.

The rows in the dataset represent the SaaS products and the columns represent the aggregated sentiment intensity scores for each service factor of these products. Each column representing the service factor is used as input variables for the fuzzy trust model. Each data item in the dataset rows represent the aggregated sentiment intensity values of *n* review segments for each service factor of a given SaaS product. The aggregated sentiment intensity score for the $j^{th}$ service factor of the $k^{th}$ SaaS product *P* is calculated as:

$$P_{kj} = \frac{\sum_{i=1}^{n} S_{ij}}{n} \tag{8.22}$$

where $S_{ij}$ represent the sentiment intensity value of $i^{th}$ review segment for the $j^{th}$ service factor. Table 8.5 shows samples of SaaS products from the dataset with the aggregated sentiment intensity scores calculating using Equation 8.22 for each of the service factor (trust factors).

Table 8.5. Samples of SaaS products with the sentiment intensity scores for each input Trust factor

| PID | A | C | E | L | M | N | O | P | R | S |
|---|---|---|---|---|---|---|---|---|---|---|
| P2760 | 0.48 | 0.32 | 0.36 | 0.24 | 0.59 | 0.54 | 0.78 | 0.64 | 0.61 | 0.47 |
| P0 | -0.34 | 0.67 | -0.16 | -0.19 | 0.11 | -0.25 | 0.09 | 0.00 | 0.05 | -0.05 |
| P100 | 0.38 | 0.36 | 0.21 | 0.52 | 0.38 | 0.43 | -0.57 | -0.35 | 0.23 | 0.32 |
| P1003 | 0.14 | -0.07 | 0.34 | 0.18 | 0.26 | 0.04 | 0.00 | 0.34 | 0.29 | 0.25 |
| P1004 | 0.25 | 0.25 | 0.20 | 0.94 | 0.47 | 0.23 | 0.32 | 0.20 | 0.39 | 0.00 |
| P1015 | 0.34 | 0.28 | 0.19 | 0.69 | 0.44 | 0.25 | 0.32 | 0.35 | 0.44 | 0.26 |
| P1024 | 0.64 | 0.43 | 0.44 | 0.95 | 0.43 | 0.28 | -0.57 | 0.46 | 0.56 | 0.42 |
| P103 | 0.78 | 0.01 | 0.47 | 0.54 | 0.47 | 0.51 | 0.26 | 0.17 | 0.49 | 0.58 |
| P1035 | 0.23 | -0.04 | 0.17 | 0.48 | 0.09 | 0.01 | 0.11 | 0.21 | 0.13 | 0.05 |
| P1037 | -0.08 | 0.31 | 0.11 | 0.37 | 0.15 | 0.23 | 0.03 | -0.17 | -0.04 | 0.21 |
| P104 | 0.59 | 0.41 | 0.44 | 0.44 | 0.45 | 0.53 | 0.38 | 0.44 | 0.32 | 0.55 |

### 8.5.1 MDR scores calculation

After the dataset is prepared by following the previous step, the MDR weights are calculated for each of the service factor. The MDR weight is crucial in ranking the trust factors (service factors) by assigning them different weights which will help to generate the fuzzy rules. The step-by-step implementation of MDR weight calculation is shown in Algorithm 8.1.

Table 8.6 shows the average maturity of each SaaS service factors, average density of reviews for each service factor and the number of unique reviewers for each service factor of the selected 542

SaaS services along with the average number of reviewers per service factor. For example, for the service factor *A (Availability factor)*, the average maturity of is calculated using Equation 8.2 as:

$$M_A = \frac{183384}{542} = 338.35$$

the averaged density of reviews is calculated using Equation 8.4 as:

$$D_A = \frac{3089}{542} = 5.7$$

the averaged unique reviewers for service factor A are calculated using Equation 8.6 as:

$$R_A = \frac{2601}{542} = 4.8$$

Before the final *MDR* score is calculated for service factor *A*, the values for metrics *M, D* and *R* is normalized between the values of 0 and 1 using Equation 8.7, Equation 8.8 and Equation 8.9 respectively as:

$$\acute{M}_A = \frac{338.35 - 13.51}{824.46 - 13.51} = 0.4$$

$$\acute{D}_A = \frac{5.7 - 0.17}{70.51 - 0.17} = 0.08$$

$$\acute{R}_A = \frac{4.8 - 0.14}{27.49 - 0.14} = 0.17$$

Finally, the *MDR* score for service factor *A* is calculated using Equation 8.10 as:

$$MDR_A = 0.4 \times 0.33 + 0.08 \times 0.33 + 0.17 \times 0.33$$
$$MDR_A = 0.22$$

Here, each metric is weighted uniformly. However, the weights for each of these metrics can be adjusted based on the application domain and user preference. Similarly, the MDR scores for the other service factors are calculated following the above example. The MDR scores of all the service factors is shown in Table 8.6 along with the average MDR score and graphically depicted in Figure 8.8.

Table 8.6. Details of individual metric used in MDR score calculation

| Metric | A | C | E | L | M | N | O | P | R | S | Sum/Avg | Values | Products |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Maturity | 338.35 | 466.29 | 13.51 | 73.14 | 803.19 | 824.46 | 414.92 | 498.73 | 378.99 | 30.02 | 3841.61 | Averaged | 542 |
| Maturity | 183384 | 252729 | 7325 | 39640 | 435328 | 446859 | 224887 | 270311 | 205415 | 16272 | 2082150 | Count | 542 |
| Density | 5.70 | 10.18 | 0.17 | 1.02 | 63.16 | 70.51 | 9.30 | 12.57 | 15.46 | 1.98 | 190.05 | Averaged | 542 |
| Density | 3089 | 5516 | 92 | 553 | 34233 | 38219 | 5038 | 6813 | 8381 | 1071 | 103005 | Count | 542 |
| Reviewers | 4.80 | 7.61 | 0.14 | 0.92 | 26.97 | 27.49 | 6.91 | 9.33 | 11.42 | 1.59 | 97.19 | Averaged | 542 |
| Reviewers | 2601 | 4126 | 77 | 500 | 14619 | 14900 | 3745 | 5058 | 6189 | 863 | 19864 | Count | 542 |

| **MDR Scores** | **0.22** | **0.32** | **0.00** | **0.04** | **0.95** | **1.00** | **0.29** | **0.37** | **0.36** | **0.03** | **0.358** | | **542** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

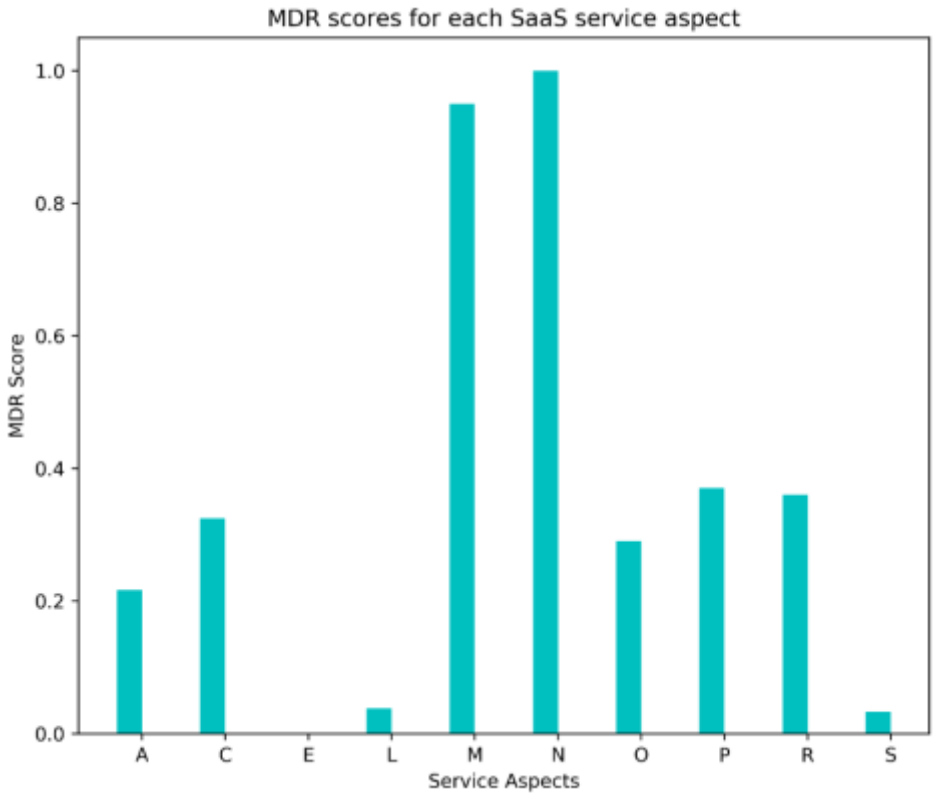



Figure 8.8 MDR scores for all the service factor from the SaaS dataset

### 8.5.2 Fuzzy inference

After the detailed explanation of the proposed approach in the example in the previous subsection, calculation of MDR scores for each service factor, a case study is presented in this subsection to explain the working of the fuzzy trust inference model.

From the dataset explained in the previous subsection, a SaaS product with product *ID: P2760* is selected to demonstrate the computation of its trust score and trust level. Table 8.6 shows the averaged sentiment intensity values for each of the service factor. Here, to remind ourselves, the service factors are the actual trust factors for the proposed trust model with crisp values.

Table 8.7. Fuzzy membership in each of the input membership function for the input variables

| **PID** | **N** | **M** | **P** | **R** | **C** | **O** | **A** | **L** | **S** | **E** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P2760 | 0.54 | 0.59 | 0.64 | 0.61 | 0.32 | 0.78 | 0.48 | 0.24 | 0.47 | 0.36 | Crisp | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Negative | Fuzzy Membe |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.04 | 0 | 0 | Neutral | |
| | 0.54 | 0.59 | 0.64 | 0.61 | 0.32 | 0.78 | 0.48 | 0.24 | 0.47 | 0.36 | Positive | |

***Fuzzification of sentiment intensity of trust factors (service factors):***

Given the values of the vertices of the triangular membership function as:

$$Negative(a,b,c) = Negative(-1,-1,0)$$
$$Neutral(a,b,c) = Neutral(-0.25,0,0.25)$$
$$Positive(a,b,c) = Positive(0,1,1)$$

Table 8.8. Rule strengths and output fuzzy set association

| **Rule** | **Rule Strength (min)** | **Output Fuzzy set** |
|---|---|---|
| R3 | 0.54 | *Very Trustworthy* |
| R6 | 0.54 | *Trustworthy* |
| R9 | 0.54 | *Trustworthy* |
| R12 | 0.54 | *Trustworthy* |
| **R15** | **0.59** | ***Trustworthy*** |
| R54 | 0.24 | *Trustworthy* |
| R57 | 0.24 | *Trustworthy* |
| R60 | 0.24 | *Trustworthy* |
| R63 | 0.32 | *Trustworthy* |
| R66 | 0.32 | *Trustworthy* |
| R69 | 0.48 | *Trustworthy* |
| R72 | 0.24 | *Trustworthy* |
| R75 | 0.24 | *Trustworthy* |
| R78 | 0.24 | *Trustworthy* |
| R81 | 0.32 | *Trustworthy* |
| R84 | 0.32 | *Trustworthy* |
| R87 | 0.48 | *Trustworthy* |
| R90 | 0.24 | *Trustworthy* |
| R93 | 0.24 | *Trustworthy* |
| R96 | 0.24 | *Trustworthy* |
| R99 | 0.32 | *Trustworthy* |
| R102 | 0.32 | *Trustworthy* |
| R105 | 0.48 | *Trustworthy* |
| R108 | 0.24 | *Trustworthy* |
| R111 | 0.24 | *Trustworthy* |
| R114 | 0.24 | *Trustworthy* |
| R117 | 0.32 | *Trustworthy* |
| R120 | 0.32 | *Trustworthy* |
| R123 | 0.48 | *Trustworthy* |
| R126 | 0.24 | *Trustworthy* |
| R129 | 0.24 | *Trustworthy* |
| R132 | 0.24 | *Trustworthy* |
| R135 | 0.32 | *Trustworthy* |
| R138 | 0.32 | *Trustworthy* |
| R141 | 0.48 | *Trustworthy* |
| R144 | 0.24 | *Trustworthy* |
| R147 | 0.24 | *Trustworthy* |
| R150 | 0.24 | *Trustworthy* |
| R153 | 0.32 | *Trustworthy* |
| R156 | 0.32 | *Trustworthy* |
| R159 | 0.48 | *Trustworthy* |
| **Max** | **0.59** | ***Trustworthy*** |

the membership values (fuzzy values) for the trust factor *A (Availability)* in the fuzzy sets Negative, Neutral and Positive are calculated using Equation 8.13 as:

$$x = 0.48$$
$$\mu_{Negative}(0.48) = 0, \quad (x \geq c)$$
$$\mu_{Neutral}(0.48) = 0, \quad (x \geq c)$$
$$\mu_{Positive}(0.48) = \frac{0.48 - 0}{1 - 0} = 0.48, \quad (a \leq x \leq b)$$

Table 8.7 shows the SaaS product (P2760) membership values in each of the input fuzzy sets for all the trust factors with columns arranged from highest MDR weights to lowest moving from left to right.

***Fuzzy rules evaluation:***

The fuzzy rules are evaluated in this step using the values from the Table 8.7 in the antecedents of the rules. The following example shows some of the effective rules (resulting in non-zero output memberships when the rule strength calculation using min operator):

**Rule 3:** If **N** is *Positive* and **M** is *Positive* and **P** is *Positive* and **R** is *Positive* then **Trust Level** is *Very Trustworthy*

The rule strength $l_{R3}$ is calculated using Equation 8.18 as:

$$l_{R3} = \min(0.54, 0.59, 0.64, 0.61) = 0.54$$

This will result in the output fuzzy set *Very Trustworthy* clipped at 0.54 membership degree (height). Table 8.8 shows the rule strengths of all the effective fuzzy rules.

For clarity, the detailed calculation of the rule strengths of all the rules with a non-zero membership in the output membership functions is placed in Section 8.4.2.

***Aggregation of the fuzzy rules:***

From the results in Table 8.8, the rules strengths are aggregated using Equation 8.20 to get a single consequent (output fuzzy set) using the fuzzy OR (*Maximum*) operator. Figure 8.9 depicts the result of the aggregation operation where it indicates that the degree of membership is larger in the Trustworthy output fuzzy set. Hence, the trust level of the SaaS product *P2760* is considered as *Trustworthy* based on the sentiment intensity values in the input trust factors.

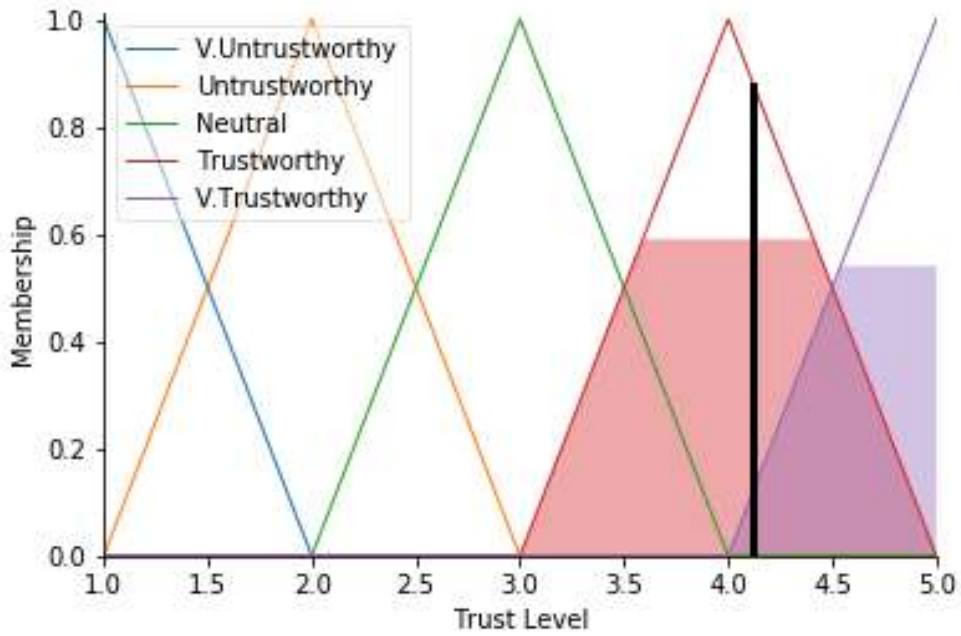



Figure 8.9 Output membership functions for a given SaaS product's trust score and trust level

***Defuzzification of the output distribution:***

Finally, the output distribution is defuzzified to calculate a crisp output trust score for the example SaaS product using Equation 8.21. For simplicity of calculation, here, I am using the centroid point of each output fuzzy sets also known the First Infer, Then Aggregate (FITA) for the defuzzification. This centroid calculation method calculates the product of centre of gravity of the consequent of each rule with its membership degree first, then aggregate the to get the final centroid of the consequent or defuzzified output value. The defuzzification returns a crisp trust value of 4.1 for the SaaS product.

### 8.5.3 Experiments and Simulation results:

The proposed fuzzy trust model is simulated using the dataset with 542 SaaS products. As depicted in Figure 8.10, the simulation results show that 83.3 percent of the SaaS products in our dataset fall under the *Trustworthy* level on the trust scale. This is followed by the 16.1 percent that are *Neutral* on the trust scale. A very small number of SaaS products with the percentage of 0.2% are inferred as Untrustworthy. The simulation found no such SaaS product which could either be scaled as Very Trustworthy or Very Untrustworthy.

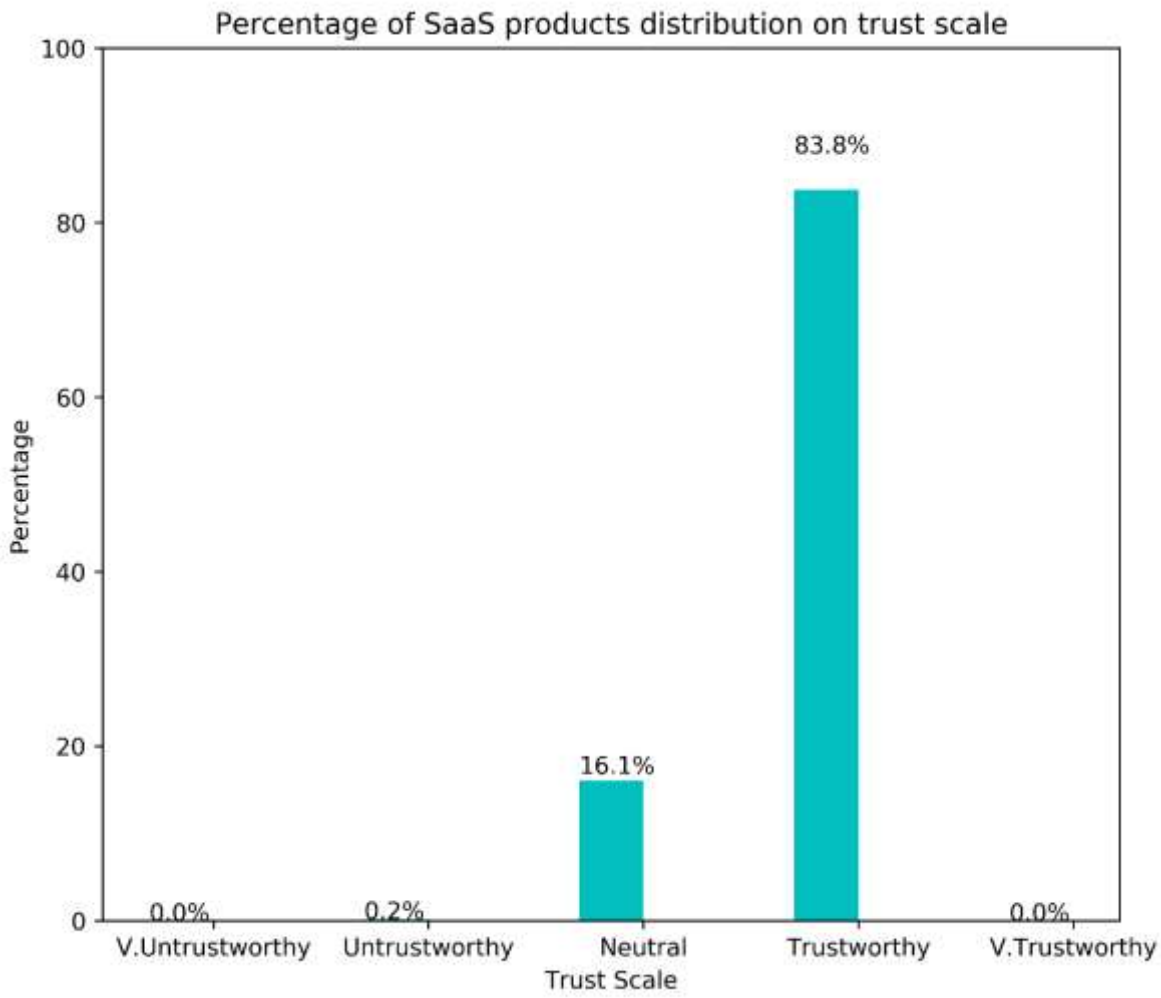



Figure 8.10 simulation results on the SaaS dataset and the percentage distribution of SaaS products based on their computed trust levels

Few of the computed results from the simulation is depicted in Figure 8.11 and Figure 8.12 with the final trust scores of 2.5 and 3.5 respectively.

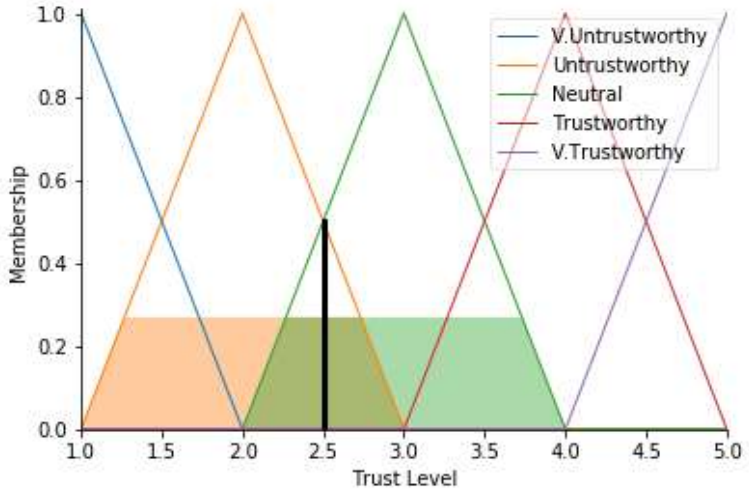



Figure 8.11 Example output membership functions with trust levels and trust scores

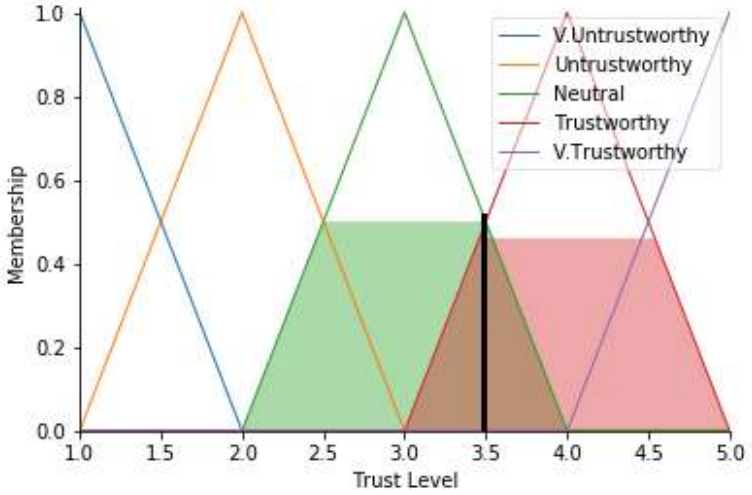



Figure 8.12 Example output membership functions with trust levels and trust scores

***Trust scores over given timesteps:***

In the previous explanation and examples, the dataset used consist of aggregated sentiment intensity scores for each trust factors for the available 542 SaaS products. The trust scores calculated indicates the trust scores of the SaaS products based their entire lifespan. However, the proposed fuzzy trust model is capable to compute the trust values of a given SaaS product at different historic timesteps. To demonstrate this capability, we have utilized the dataset used in the discussion in Chapter 7. For a given SaaS product *P409*, the sentiment intensity scores for all the input trust factors between 30-01-2014 and 03-06-2017 is depicted in Figure 8.13. It can be observed from Figure 8.13 that the sentiment intensity of service factor is different at a single timestamp. Thus, proving the importance of a weighted fuzzy inference system to compute an acceptable trust value. Based on these input values, the final trust scores computed by the proposed fuzzy trust model, for each timestep is depicted

in Figure 8.14. It can be observed from the simulation results that the trust scores vary with the changes in user sentiment towards different service factor.

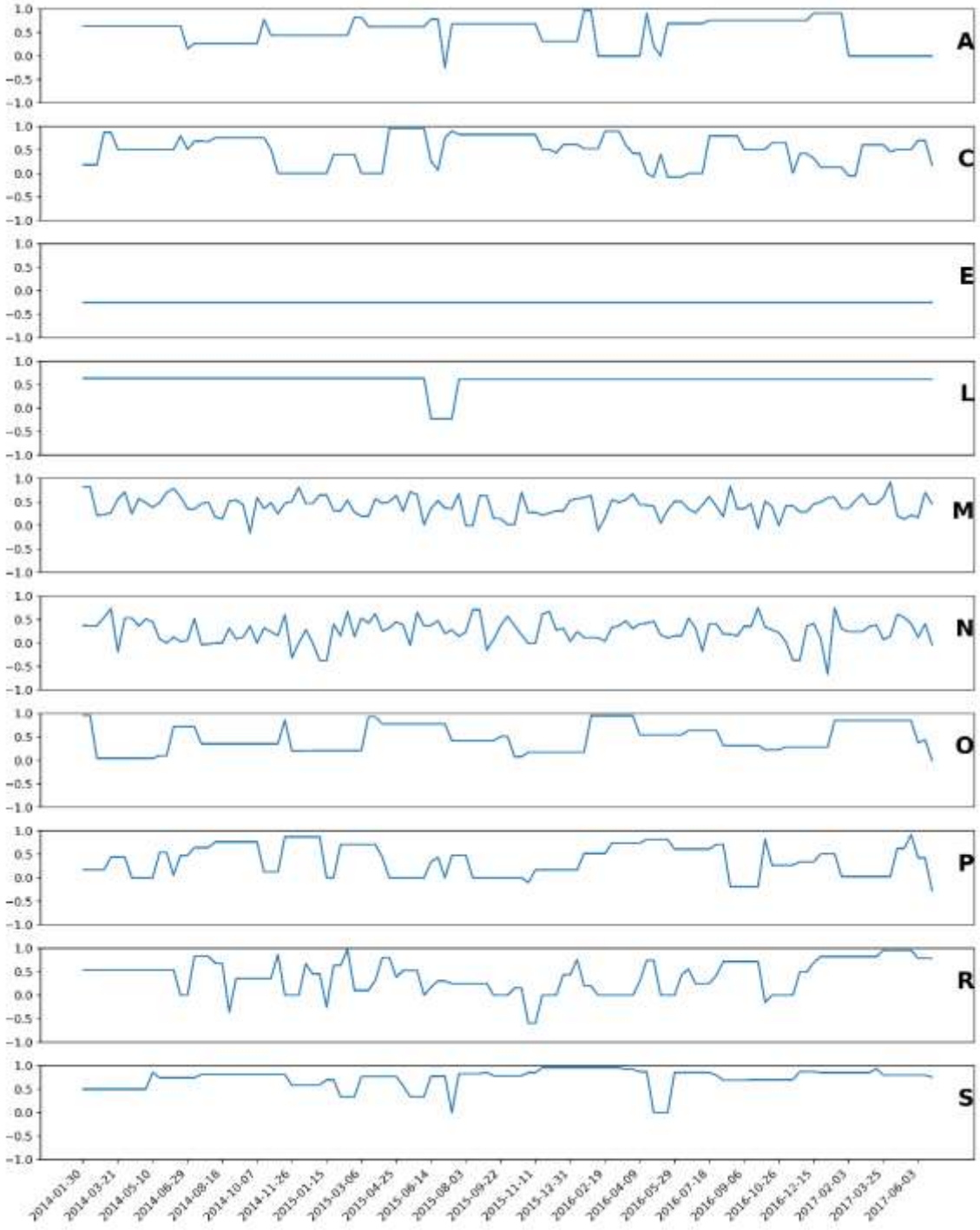



Figure 8.13 Sentiment intensity scores for each service factor of a given SaaS product, used as trust factors for inputs in fuzzy trust model

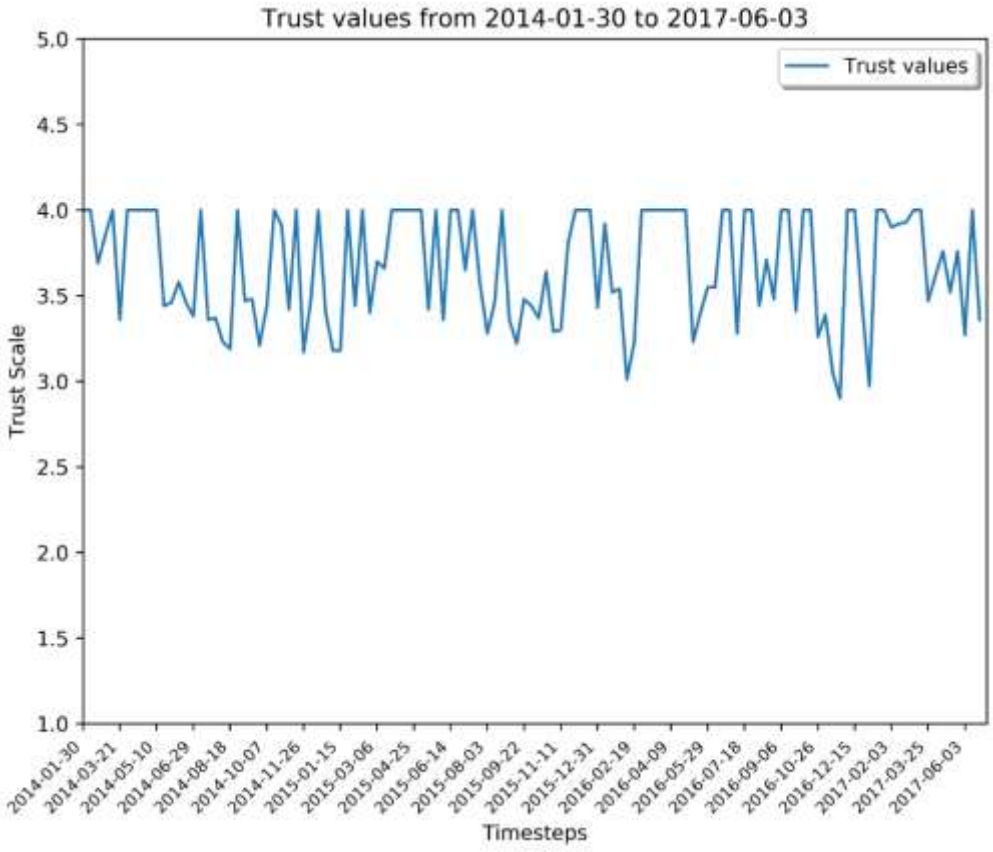



Figure 8.14 The output trust scores at each time spot for which the input trust factors are observed

## 8.6 CONCLUSION

During the trust modelling stage, the solution to the core issues which are solved in the initial stages that includes trust factor identification, sentiment intensity computation for each trust factor, imputation of missing sentiment intensity scores for trust factors and efficient method to forecast the trust factors, helps to make sure the fuzzy trust model is designed with high precision and strong capability. In this chapter, a model is developed to compute the trust scores and trustworthy levels of SaaS products using Mamdani fuzzy inference system. The model takes as an input the sentiment intensity scores for the service factors of SaaS products which is utilized as trust factors to compute the trust scores. The proposed model outputs the trust scores as well as the trustworthy levels of any given SaaS product. The proposed fuzzy trust model has demonstrated the capability to precisely compute trust scores for SaaS products both at current standings and at any point in time. To improve the performance of the fuzzy inference system, a new method called the MDR is developed that ranks the service factors by assigning them different weights. This ranking helps to choose the most effective service factors in the computation of the final trust scores.

The developed approach is demonstrated by using real dataset collected from different sources. A case study is conducted that demonstrates the step-by-step implementation and working of the developed technique. The simulation results on two different type of datasets shows promising results by successfully capturing the fluctuating trust scores of different SaaS products.

# Chapter 9:

# Recapitulation and future work

## 9.1 INTRODUCTION

Cloud computing has become the core technological backbone for the businesses today. Among the other cloud service delivery models, Software as a Service is the top priority solution for businesses and individuals for their business needs. The adoption of SaaS is increasing at a rapid pace, however, this change in the software marketplace brings a number of challenges such as the trustworthiness of these SaaS products. The information technology infrastructures used by these SaaS products are also advancing and bringing more strength and flexibility to the SaaS architectures. Moreover, the economies of scale bring financial benefits for the SaaS providers. In such dynamic software marketplace, the customers need to make timely decision in selecting the SaaS products meets their business requirements and are trustworthy based on critical service factors. The SaaS customers do not have significant visibility to structure and architecture of the offered SaaS products. An acceptable level of trust towards a SaaS product is helpful in their decision to sign-up for a service.

As discussed in Chapter 2, the cloud service providers have more visibility to the deployed SaaS architectures compared to the customers. Most of the providers consider the cloud applications and SaaS architectural best practices in designing and maintaining their SaaS products. The SaaS customers on the other hand can benefit in their decisions by utilizing the reliable information regarding the SaaS architectures.

In this thesis, I presented a trust management framework that identifies the SaaS service factors that is based on cloud applications architectural best practices and models the trust in SaaS products using the customer past sentiment towards these service factors.

In the next section, I recapitulate the issues addressed in this thesis. In Section 9.3, I discuss the contribution of this this thesis in solving the identified the research issues for different aspects of trust modelling. In Section 9.4, I provide the future research directions of this study and Section 9.5 concludes the chapter.

## 9.2 RECAPITULATION

The aim of thesis is to facilitate the SaaS customer decisions during the selection of suitable SaaS products. For this purpose, the thesis proposes a trust management framework that models the trustworthiness in SaaS products. The concept of trust has been studied in several domain including cloud computing. However, the nature of SaaS products differs from traditional software architectures with more dependencies and dynamics. To provide a level of trust for the SaaS products that benefits the SaaS customers, the SaaS products must be assessed through architectural best practice. In the existing literature, the trust in SaaS products has not been studied in this context.

In order to model the trust in SaaS products, a number of challenges and issues arise. In this thesis, I set separate objectives to solve the identified issues which are highlighted as follows:

1. To develop a method for the identification of the SaaS service factors automatically from customer reviews
2. To develop a method for the imputation of unobserved information on the SaaS service factors
3. To develop a methodology for forecasting the information regarding each SaaS service factor
4. To develop a method for dynamically modelling the trust in SaaS products by utilizing the information from the SaaS service factors
5. To validate all the developed techniques in this thesis

The novelty of each of the proposed solution in this thesis is summarized as:

1. This thesis considers the cloud applications and services architectural best practices as trust factors which is not focused on in the existing literature.
2. The T-NN approach proposed in this thesis, imputes the missing or unobserved information in the SaaS service factors which is not addressed in the existing literature.
3. Forecasting the SaaS service factors to compute trust in the future time is not addressed in the existing literature. This thesis proposes a forecast model for SaaS service factors that exhibits high performance.
4. The MDR approach proposed in this thesis, dynamically weights and ranks the trust factors which is not addressed in the existing literature.

In the next section, a summary of the contributions by this thesis is provided.

## 9.3 CONTRIBUTION OF THE THESIS

This thesis will help to understand the relationship between the SaaS architectural best practices customer experiences. This study will bring forward the most accurate approach that utilizes user reviews to compute the trust values for SaaS products in cloud marketplaces. This thesis will help SaaS consumers in their decisions to select SaaS products based on the strength of each service factor according to the dynamic ranking of each service factors. This study will also help the SaaS providers to focus on the key factors of their SaaS products which could result in better user experience and increased revenue.

The key contribution of this thesis to the existing body of the literature on the trust models in SaaS are discussed in the following subsections.

### 9.3.1 Contribution 1: A novel framework that establishes the root of trust in SaaS using Architectural best practices

The first contribution of the thesis is a novel framework that utilizes the cloud applications and services architectural best practices to identify the key SaaS service factors and obtains relevant information for these service factors. The service factors are decided based on the pillars of well-architected frameworks introduced by major cloud service providers such as Amazon Web Services and Microsoft Azure.

As part of the first contribution of this thesis, the proposed framework identifies these service factors from customer reviews based on their previous experience. The customer reviews are collected and the reviews are labelled under the relevant categories of SaaS service factors. The details of proposed framework are presented in Chapter 5. Initially, the labels are assigned manually by finding the relevant words and phrases that hints a particular service factor. This contribution is significant and different modelling techniques can be applied on the SaaS reviews dataset that provide solutions in several problem domain.

### 9.3.2 Contribution 2: An ensemble model that automatically identifies SaaS service factors

The second contribution of this thesis is an ensemble model for text classification that is trained on SaaS customer reviews. The development of the ensemble text classification model involves four stages that starts with a data pre-preparation stage. The textual data prepared for the classification models using techniques such as data cleansing, *tf-idf* vectorization and scaling and feature normalization.

In the second stage of development, suitable machine learning methods are selected based on the problem specification. The ML approaches both from the parametric and non-parametric categories are selected in this stage. The selected ML approaches have the capability to learn in a supervised setting, useful for multi-class text classification and can work with input data with high dimensions.

In the third stage the optimal parameters of the selected machine learning approaches are estimated by cross-validating on the training data from the SaaS review dataset. Data resampling techniques such as stratification and SMOTE are applied on the training data during the parameter estimation to achieve the best fit of the ML models.

In the last stage of the ensemble model development, the performances of the selected ML models are evaluated on test data from the SaaS review dataset. To select the ML models for ensemble creation, their performances are cross-compared and Friedman's statistical significance test with Nemenyi's post-hoc test is performed. Finally, the ensemble model is created from the best performing ML models. The detailed steps for the development and validation of the ensemble model are presented in Chapter 4. The proposed ensemble model for SaaS reviews is novel and is a significant contribution to the existing literature.

### 9.3.3 Contribution 3: An imputation method for missing sentiment scores of SaaS service factors

The third contribution of this is a novel method for imputing the missing information regarding the service factors. In Chapter 6, the methodology is developed in four stages, Initially, the collected customer reviews are segmented and for each text segment the SaaS service factor is identified using the proposed method from the previous objective. In the second stage of methodology, the sentiment intensity of each text segment is computed using VADER which is a rule-based valence computation approach.

The novel Threshold-based Nearest Neighbour (T-NN) method for imputing the sentiment intensity scores of the missing SaaS service factors is developed in the third stage of the methodology. T-NN uses the correlation weights to calculate the distances and the donors are selected using a threshold value.

In the final stage, the imputation performance of the T-NN is evaluated against the well-known imputation approaches such as K-NN, SVD, Decision Tree, EM, SGD and the Mean and Median imputations. The performance of T-NN is investigate and thoroughly validated under two different schemes. In Chapter 6, I presented the details of each stage of the methodology along with the model

validation in both the schemes. To the best of knowledge, there is no such study that investigates and imputes the missing sentiment intensities of SaaS service factors. The T-NN is a novel contribution to the existing literature for imputing the sentiment intensity scores of missing SaaS service factors.

### 9.3.4 Contribution 4: A forecast model for SaaS service factors

The fourth contribution of this thesis is a methodology to forecast the sentiment intensity scores of SaaS service factors. The detailed development steps of the proposed methodology are presented in Chapter 7. The forecast model is capable to forecast SaaS sentiment intensity scores for future time stamps with optimal performance. The forecasted values are used by the trust model to compute the trust level and trust scores of SaaS products for future points in time. The methodology is developed in four stages. The dataset used to develop the forecast model contained the irregular observations for sentiment intensity scores of the SaaS service factor. The irregular series is transformed into regular time series data using the Linear Interpolation and LOCF techniques in the first stage of the methodology.

In the second stage, a preliminary investigation is performed through statistical tests such as Augmented Dickey-Fuller test, Granger Causality test and Pearson's Correlation Coefficient test to understand the sentiment intensity time series for their statistical properties. Based on the preliminary investigation, time series forecasting models are selected such as ARIMA, Exponential smoothing, MLP and LSTM for evaluation on the time series data. All the selected models are evaluated for their single-step and multi-steps forecasting capabilities on the SaaS sentiment intensity time series data. I have also developed and evaluated the multivariate versions of the selected models for more in-depth analysis of their performance. Based on the validation results of the selected models, I used the LSTM model for the forecast component of the trust management framework. The details stages of development and model validation is presented in Chapter 7. To the best of my knowledge, there is no study in the existing literature that focuses on the development of forecast model for SaaS sentiment intensities.

### 9.3.5 Contribution 5: A dynamic trust model for SaaS

In Chapter 8, a dynamic trust model is developed using Mamdani Fuzzy Inference approach. The proposed trust model is capable to compute the trust levels and trust score for SaaS products at current point in time and also for future time slots by utilizing the sentiment intensity scores of the SaaS service factors. A novel dynamic weighing and ranking approach called MDR, is proposed to identify

the priority of trust factors in trust computation. The dynamic trust model is used to model the trust levels of the SaaS products for further insight into current SaaS marketplace.

To the best of knowledge, there is such approach in existing literature that dynamically utilized the sentiment intensity scores of SaaS service factors to model the trust in SaaS products.

## 9.4 FUTURE WORK

In this thesis, a trust management framework for SaaS products is introduced for supporting customer decisions in adopting SaaS products. This thesis addressed the key issues during the development of the proposed framework and presented solutions for the identified issues. However, during the research process in this study, several directions for future research are identified in the relevant research area. The key areas for the identified future research are:

1. The initial reviews were labelled directly by manual identification of service factors. I aim to improve the labelling method of the SaaS customer reviews by adding more relevant terms associated to each service factor. The current automated labelling method considers a limited set of terms for the identification of the service factors. The results obtained from the ensemble approach in Chapter 5, such as the confusion matrix and extended classification reports for individual predicted classes to improve the labelling of the dataset which will result in better prediction results
2. The proposed text classification methodology for SaaS reviews will further be compared with the other approaches for text classification. The significant features will be incorporated in the existing methodology to improve the classifiers performance.
3. The proposed imputation model for the sentiment intensity scores for SaaS service factors will be applied on datasets from other application domains to further validate its performance. Currently, the imputation approach uses valence-based sentiment intensity computation method (VADER), to compute the sentiment intensity scores of SaaS review text. The imputation model will be studied with other imputation methods such as LSTM to investigate further performance improvement of the proposed imputation model.
4. For the forecast model, the results for more SaaS products time series data will be collected to provide an aggregated result of the forecast model performance. Furthermore, the LSTM approach is implemented with many parameters. In this thesis I have used few of the parameters for the model estimation. As part of the future work of this thesis, I intend to

perform experiments using other parameters to further improve the LSTM model's forecasting performance.

5. The process of collecting the customer reviews for SaaS products will be automated to update the review database with latest reviews for new and existing SaaS products.

## 9.5 CONCLUSION

In this chapter, I highlighted the objectives of this thesis and recapitulated the research conducted during this study and discussed the key contributions. I described briefly, the future research directions that is set by this study and highlighted the potential extensions of the proposed methods. To data, a significant portion of this study has been published in peer-reviewed and well-reputed international scientific journals. The remaining unpublished portions of this study is intended to be sent for publication in peer-reviewed journals.

# References


Abawajy, J. (2009, 14-16 Dec. 2009). *Determining Service Trustworthiness in Intercloud Computing Environments.* Paper presented at the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks.

Abdelrazek, M. A., Grundy, J., & Ibrahim, A. S. (2015, 9-12 Dec. 2015). *Improving Tenants' Trust in SaaS Applications Using Dynamic Security Monitors.* Paper presented at the 2015 20th International Conference on Engineering of Complex Computer Systems (ICECCS).

Abdul-Rahman, A., & Hailes, S. (2000). *Supporting Trust in Virtual Communities*. Paper presented at the Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6 - Volume 6.

Alhamad, M., Dillon, T., & Chang, E. (2010, 14-16 Sept. 2010). *SLA-Based Trust Model for Cloud Computing.* Paper presented at the 2010 13th International Conference on Network-Based Information Systems.

Alhamad, M., Dillon, T., & Chang, E. (2011). A Trust-Evaluation Metric for Cloud applications. *International Journal of Machine Learning and Computing, 1*, 416-421. doi:10.7763/IJMLC.2011.V1.62

Alshammari, S. T., Albeshri, A., & Alsubhi, K. (2021). Integrating a High-Reliability Multicriteria Trust Evaluation Model with Task Role-Based Access Control for Cloud Services. *13*(3), 492.

AWS. (2017). AWS Marketplace Retrieved from https://aws.amazon.com/marketplace

AWS. (2018). AWS Well-Architected Framework. Retrieved from https://d1awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

Azure, M. (2018). Pillars of software quality. Retrieved from https://docs.microsoft.com/enus/azure/architecture/guide/pillars

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C., & Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: An overview. *Bioinformatics (Oxford, England), 16*, 412-424.

Batista, G. E. A. P. A., & Monard, M. C. (2003). *Experimental comparison pf K-NEAREST NEIGHBOUR and MEAN OR MODE imputation methods with the internal strategies used by C4.5 and CN2 to treat missing data*.

Bayen, A. M., & Siauw, T. (2015). Chapter 14 - Interpolation. In A. M. Bayen & T. Siauw (Eds.), *An Introduction to MATLAB® Programming and Numerical Methods for Engineers* (pp. 211-223). Boston: Academic Press.

Ben-David, A. (2008). Comparison of classification accuracy using Cohen's Weighted Kappa. *Expert Systems with Applications, 34*(2), 825-832. doi:https://doi.org/10.1016/j.eswa.2006.10.022

Beth, T., Borcherding, M., & Klein, B. (1994, 1994//). *Valuation of trust in open networks.* Paper presented at the Computer Security — ESORICS 94, Berlin, Heidelberg.

Billhardt, H., Hermoso, R., Ossowski, S., & Centeno, R. (2007). *Trust-based service provider selection in open environments*. Paper presented at the Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea. https://doi.org/10.1145/1244002.1244298

Boughorbel, S., Jarray, F., & El-Anbari, M. (2017). Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PloS one, 12*, e0177678. doi:10.1371/journal.pone.0177678

Box, G., Jenkins, G., & Reinsel, G. (2008). *Time series analysis: forecasting and control* (Vol. 4th): Willey.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and Regression Trees*: Taylor & Francis.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees* (Second ed.): Wadsworth International Group.

Brownlee, J. (2016a). *Machine Learning Algorithms From Scratch with Python*: Machine Learning Mastery.

Brownlee, J. (2016b). *Master Machine Learning Algorithms: Discover how They Work and Implement Them from Scratch*.

Buchegger, S., & Le Boudec, J.-Y. (2003). A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks. *P2PEcon 2004*.

Butticè, V., Colombo, M., & Wright, M. (2017). Serial Crowdfunding, Social Capital, and Project Success. *Entrepreneurship Theory and Practice*. doi:10.1111/etap.12271

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599-616. doi:https://doi.org/10.1016/j.future.2008.12.001

Canedo, E. D., Albuquerque, R. d. O., & de Sousa Junior, R. T. (2011). *Trust model for file sharing in cloud computing*. Paper presented at the Second International Conference on Cloud Computing, GRIDs, and Virtualization.

Canedo, E. D., Junior, R. T. d. S., Carvalho, R. R. d., & Albuquerque, R. d. O. (2012, 26-28 June 2012). *Trust model for private cloud.* Paper presented at the Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec).

Castelfranchi, C., & Falcone, R. (2010). *Trust Theory: A Socio-Cognitive and Computational Model*: Wiley Publishing.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *J. Artif. Int. Res., 16*(1), 321–357.

Cheng, J., Bernstein, M., Danescu-Niculescu-Mizil, C., & Leskovec, J. (2017). Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions. *CSCW : proceedings of the Conference on Computer-Supported Cooperative Work. Conference on Computer-Supported Cooperative Work, 2017*. doi:10.1145/2998181.2998213

Chou, S.-W., & Chiang, C.-H. (2013). Understanding the formation of software-as-a-service (SaaS) satisfaction from the perspective of service quality. *Decision Support Systems, 56*, 148-155. doi:https://doi.org/10.1016/j.dss.2013.05.013

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement, 20*(1), 37-46. doi:10.1177/001316446002000104

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21-27. doi:10.1109/TIT.1967.1053964

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online Passive-Aggressive Algorithms. *7*, 551–585.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *3*(null %J J. Mach. Learn. Res.), 951–991. doi:10.1162/jmlr.2003.3.4-5.951

Davidson, T., Warmsley, D., Macy, M. W., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *CoRR, abs/1703.04009*.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association, 74*(366), 427-431. doi:10.2307/2286348

Fischer-Hübner, S. (2001). *IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms*: Springer.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res., 3*(null), 1289–1305.

Friedman, M. (1940). A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics, 11*(1), 86-92.

Gambetta, D. (2000). Can We Trust Trust? Diego Gambetta.

Ganeriwal, S., Balzano, L. K., & Srivastava, M. B. (2008). Reputation-based framework for high integrity sensor networks. *ACM Trans. Sen. Netw., 4*(3), Article 15. doi:10.1145/1362542.1362546

García-Laencina, P. J., Sancho-Gómez, J.-L., & Figueiras-Vidal, A. R. (2010). Pattern classification with missing data: a review. *Neural Computing and Applications, 19*(2), 263-282. doi:10.1007/s00521-009-0295-6

Gardner Jr., E. S. (1985). Exponential smoothing: The state of the art. *Journal of Forecasting, 4*(1), 1-28. doi:https://doi.org/10.1002/for.3980040103

Gelman, A., & Hill, J. (2006). Missing-data imputation. In A. Gelman & J. Hill (Eds.), *Data Analysis Using Regression and Multilevel/Hierarchical Models* (pp. 529-544). Cambridge: Cambridge University Press.

GetApp. (2017). GetApp. Retrieved from https://www.getapp.com/
Ghazanfar, M. A., & Prügel-Bennett, A. (2013). The Advantage of Careful Imputation Sources in Sparse Data-Environment of Recommender Systems: Generating Improved SVD-based Recommendations. *Informatica (Slovenia), 37*(1), 61-92.
Golbeck, J. A., & Hendler, J. (2005). *Computing and applying trust in web-based social networks.* University of Maryland at College Park,
Gorodkin, J. (2004). Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry, 28*(5), 367-374. doi:https://doi.org/10.1016/j.compbiolchem.2004.09.006
Granger, C. W. J. (1969). Investigating Causal Relations by Econometric Models and Cross-spectral Methods. *Econometrica, 37*(3), 424-438. doi:10.2307/1912791
Habib, S. M., Hauke, S., Ries, S., & Mühlhäuser, M. (2012). Trust as a facilitator in cloud computing: a survey. *Journal of Cloud Computing: Advances, Systems and Applications, 1*(1), 19. doi:10.1186/2192-113X-1-19
Habib, S. M., Ries, S., & Muhlhauser, M. (2011, 16-18 Nov. 2011). *Towards a Trust Management System for Cloud Computing.* Paper presented at the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications.
Habib, S. M., Ries, S., & Mühlhäuser, M. (2010). Cloud Computing Landscape and Research Challenges Regarding Trust and Reputation. *2010 7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing*, 410-415.
Habib, S. M., Ries, S., Mühlhäuser, M., & Varikkattu, P. (2014). Towards a trust management system for cloud computing marketplaces: using CAIQ as a trust information source. *Sec. and Commun. Netw., 7*(11), 2185–2200. doi:10.1002/sec.748
Habib, S. M., Varadharajan, V., & Mühlhäuser, M. (2013). *A framework for evaluating trust of service providers in cloud marketplaces*. Paper presented at the Proceedings of the 28th Annual ACM Symposium on Applied Computing, Coimbra, Portugal. https://doi.org/10.1145/2480362.2480727
Hang, C.-W., & Singh, M. P. (2011). Trustworthy Service Selection and Composition. *ACM Trans. Auton. Adapt. Syst., 6*(1), Article 5. doi:10.1145/1921641.1921646
Hang, C.-W., Wang, Y., & Singh, M. P. (2009). *Operators for propagating trust and their evaluation in social networks*. Paper presented at the Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, Budapest, Hungary.
Haq, I. U., Alnemr, R., Paschke, A., Schikuta, E., Boley, H., & Meinel, C. (2010). *Distributed Trust Management for Validating SLA Choreographies*, Boston, MA.
Hassan, H., El-Desouky, A. I., Ibrahim, A., El-Kenawy, E. M., & Arnous, R. (2020). Enhanced QoS-Based Model for Trust Assessment in Cloud Computing Environment. *IEEE Access, 8*, 43752-43763. doi:10.1109/ACCESS.2020.2978452
Hedabou, M., Azougaghe, A., & Bentajer, A. (2020). *TPM Based Design for Enhanced Trust in SaaS Services*.
Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9*(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735
Hoerl, A. E., & Kennard, R. W. (2000). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics, 42*(1), 80-86. doi:10.2307/1271436
Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting, 20*(1), 5-10. doi:https://doi.org/10.1016/j.ijforecast.2003.09.015
Hussain, W., Hussain, F. K., & Hussain, O. K. (2014). *Maintaining Trust in Cloud Computing through SLA Monitoring*, Cham.
Hutto, C. J., & Gilbert, E. (2015). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*.
Huynh, T. D., Jennings, N. R., & Shadbolt, N. R. (2006). An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems, 13*(2), 119-154. doi:10.1007/s10458-005-6825-4

Hwang, K., Kulkareni, S., & Hu, Y. (2009, 12-14 Dec. 2009). *Cloud Security with Virtualized Defense and Reputation-Based Trust Mangement.* Paper presented at the 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing.

Hyndman, R. J. A., George. (2018). *Forecasting: Principles and Practice* (2nd Ed.). Australia: OTexts.

Inc, G. (2017). Magic Quadrant for Cloud Infrastructure as a Service, Worldwide. Retrieved from https://www.gartner.com/doc/reprints?id=1-2G2O5FC&ct=150519

Ittner, D. J., Lewis, D., & Ahn, D. D. (1995). *Text categorization of low quality images*.

Joachims, T. (1997). *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization*. Paper presented at the Proceedings of the Fourteenth International Conference on Machine Learning.

Joachims, T. (1998, 1998//). *Text categorization with Support Vector Machines: Learning with many relevant features.* Paper presented at the Machine Learning: ECML-98, Berlin, Heidelberg.

Josang, A. (2009, 6-9 July 2009). *Fission of opinions in subjective logic.* Paper presented at the 2009 12th International Conference on Information Fusion.

Jøsang, A., & Ismail, R. (2002). *The Beta Reputation System*. Paper presented at the 15th Bled Electronic Commerce Conference.

Jøsang, A., Ismail, R., & Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision Support Systems, 43*(2), 618-644. doi:https://doi.org/10.1016/j.dss.2005.05.019

Jurman, G., Riccadonna, S., & Furlanello, C. (2012). A Comparison of MCC and CEN error measures in multi-class prediction. *PloS one, 7*, e41882. doi:10.1371/journal.pone.0041882

Kamvar, S. D., Schlosser, M. T., & Garcia-Molina, H. (2003). *The Eigentrust algorithm for reputation management in P2P networks*. Paper presented at the Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary. https://doi.org/10.1145/775152.775242

Kanwal, A., Masood, R., Ghazia, U. E., Shibli, M. A., & Abbasi, A. G. (2013, 20-23 Aug. 2013). *Assessment Criteria for Trust Models in Cloud Computing.* Paper presented at the 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing.

Kenward, M. G., & Molenberghs, G. (2009). Last Observation Carried Forward: A Crystal Ball? *Journal of Biopharmaceutical Statistics, 19*(5), 872-888. doi:10.1080/10543400903105406

Kesarwani, A., & Khilar, P. M. (2019). Development of trust based access control models using fuzzy logic in cloud computing. *Journal of King Saud University - Computer and Information Sciences*. doi:https://doi.org/10.1016/j.jksuci.2019.11.001

Khan, K. M., & Malluhi, Q. (2010). Establishing Trust in Cloud Computing. *IT Professional, 12*(5), 20-27. doi:10.1109/MITP.2010.128

Kim, Y. B., Kim, J. G., Kim, W., Im, J. H., Kim, T. H., Kang, S. J., & Kim, C. H. (2016). Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies. *PloS one, 11*(8), e0161197-e0161197. doi:10.1371/journal.pone.0161197

Kinateder, M., Baschny, E., & Rothermel, K. (2005). *Towards a Generic Trust Model – Comparison of Various Trust Update Algorithms*, Berlin, Heidelberg.

Kohavi, R. (1995). *A study of cross-validation and bootstrap for accuracy estimation and model selection*. Paper presented at the Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, Montreal, Quebec, Canada.

Latif, R., Afzaal, S., & Latif, S. (2021). A novel cloud management framework for trust establishment and evaluation in a federated cloud environment. *The Journal of Supercomputing, 77*. doi:10.1007/s11227-021-03775-8

Lee, A. J., Winslett, M., & Perano, K. J. (2009, 2009//). *TrustBuilder2: A Reconfigurable Framework for Trust Negotiation.* Paper presented at the Trust Management III, Berlin, Heidelberg.

Lee, K. M., Hwang, K., Lee, J.-H., & Kim, H.-J. (2006). *A Fuzzy Trust Model Using Multiple Evaluation Criteria*, Berlin, Heidelberg.

Lee Rodgers, J., & Nicewander, W. A. (1988). Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician, 42*(1), 59-66. doi:10.1080/00031305.1988.10475524

Levien, R. (2009). Attack-Resistant Trust Metrics. In J. Golbeck (Ed.), *Computing with Social Trust* (pp. 121-132). London: Springer London.

Lewis, D. D. (1998, 1998//). *Naive (Bayes) at forty: The independence assumption in information retrieval.* Paper presented at the Machine Learning: ECML-98, Berlin, Heidelberg.

Li, W., & Ping, L. (2009). *Trust Model to Enhance Security and Interoperability of Cloud Environment*, Berlin, Heidelberg.

Li, X., & Du, J. (2013). Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing. *IET Information Security, 7*(1), 39-50. doi:https://doi.org/10.1049/iet-ifs.2012.0232

Limam, N., & Boutaba, R. (2010). Assessing Software Service Quality and Trustworthiness at Selection Time. *IEEE Trans. Softw. Eng., 36*(4), 559–574. doi:10.1109/tse.2010.2

Lin, C., Varadharajan, V., Wang, Y., & Pruthi, V. (2004). *Enhancing Grid Security with Trust Management*. Paper presented at the Proceedings of the 2004 IEEE International Conference on Services Computing.

Little, R. J. A., & Rubin, D. B. (1986). *Statistical analysis with missing data*: John Wiley & Sons, Inc.

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). NIST Cloud Computing Reference Architecture. *Special Publication (NIST SP)*. doi:https://doi.org/10.6028/NIST.SP.500-292

Ltkepohl, H. (2007). *New Introduction to Multiple Time Series Analysis*: Springer Publishing Company, Incorporated.

Lynn, T., van der Werff, L., & Fox, G. (2021). Understanding Trust and Cloud Computing: An Integrated Framework for Assurance and Accountability in the Cloud. In T. Lynn, J. G. Mooney, L. van der Werff, & G. Fox (Eds.), *Data Privacy and Trust in Cloud Computing: Building trust in the cloud through assurance and accountability* (pp. 1-20). Cham: Springer International Publishing.

Ma, A., & Needell, D. (2018). Stochastic Gradient Descent for Linear Systems with Missing Data. *Numerical Mathematics: Theory, Methods and Applications, 12*(1), 1--20. doi:https://doi.org/10.4208/nmtma.OA-2018-0066

Macías, M., & Guitart, J. (2012). *Cheat-Proof Trust Model for Cloud Computing Markets*, Berlin, Heidelberg.

Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies, 7*(1), 1-13. doi:https://doi.org/10.1016/S0020-7373(75)80002-2

Manchala, D. W. (2000). E-Commerce Trust Metrics and Models. *IEEE Internet Computing, 4*(2), 36–44. doi:10.1109/4236.832944

Manuel, P. (2015). A trust model of cloud computing based on Quality of Service. *Annals of Operations Research, 233*(1), 281-292. doi:10.1007/s10479-013-1380-x

Marsh, S. (1999). *Formalising Trust as a Computational Concept.*

Martínez-Rego, D., Fernández-Francos, D., Fontenla-Romero, O., & Alonso-Betanzos, A. (2015). Stream change detection via passive-aggressive classification and Bernoulli CUSUM. *305*(C %J Inf. Sci.), 130–145. doi:10.1016/j.ins.2015.01.022

Martucci, L. A., Zuccato, A., Smeets, B., Habib, S. M., Johansson, T., & Shahmehri, N. (2012, 4-7 Sept. 2012). *Privacy, Security and Trust in Cloud Computing: The Perspective of the Telecommunication Industry.* Paper presented at the 2012 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing.

Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure, 405*(2), 442-451. doi:https://doi.org/10.1016/0005-2795(75)90109-9

McKnight, D., & Chervany, N. (1996). *The Meanings of Trust*.

Mell, P., & Grance , T. (2011). The NIST Definition of Cloud Computing. *Special Publication (NIST SP)*. doi:https://doi.org/10.6028/NIST.SP.800-145

Molinaro, A. M., Simon, R. M., & Pfeiffer, R. M. J. B. (2005). Prediction error estimation: a comparison of resampling methods. *21 15*, 3301-3307.

Moyano, F., Fernandez-Gago, C., & Lopez, J. (2013). A framework for enabling trust requirements in social cloud applications. *Requirements Engineering, 18*(4), 321-341. doi:10.1007/s00766-013-0171-x

Muller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python*: O'Reilly Media, Inc.

Mullin, M., & Sukthankar, R. (2000). *Complete cross-validation for nearest neighbour classifiers*. Paper presented at the Seventeenth International Conference on Machine Learning.

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing, 2*(5), 183-197. doi:https://doi.org/10.1016/0925-2312(91)90023-5

Nagarajan, R., Selvamuthukumaran, S., & Thirunavukarasu, R. (2017, 5-7 Jan. 2017). *A fuzzy logic based trust evaluation model for the selection of cloud services.* Paper presented at the 2017 International Conference on Computer Communication and Informatics (ICCCI).

Naseer, M. K., Jabbar, S., & Zafar, I. (2014, 18-20 Jan. 2014). *A novel trust model for selection of Cloud Service Provider.* Paper presented at the 2014 World Symposium on Computer Applications & Research (WSCAR).

Nemenyi, P. (1963). *Distribution-Free Multiple Comparisons.* (Ph. D), Princeton University,

Nunamaker, J. F., Chen, M., & Purdin, T. D. M. (1990). Systems Development in Information Systems Research. *Journal of Management Information Systems, 7*(3), 89-106. doi:10.1080/07421222.1990.11517898

Olmedilla, D., Rana, O., Matthews, B., & Nejdl, W. (2005). *Security and Trust Issues in Semantic Grids.* Paper presented at the Semantic Grid.

Özgür, A., Özgür, L., & Güngör, T. (2005, 2005//). *Text Categorization with Class-Based and Corpus-Based Keyword Selection.* Paper presented at the Computer and Information Sciences - ISCIS 2005, Berlin, Heidelberg.

Pawar, P. S., Rajarajan, M., Dimitrakos, T., & Zisman, A. (2013). *Trust Model for Cloud Based on Cloud Characteristics*, Berlin, Heidelberg.

Pawar, P. S., Rajarajan, M., Nair, S. K., & Zisman, A. (2012). *Trust Model for Optimized Cloud Services*, Berlin, Heidelberg.

Qiang, G., Dawei, S., Guiran, C., Sun, L., & Wang, X. (2011, 18-20 Jan. 2011). *Modeling and evaluation of trust in cloud computing environments.* Paper presented at the 2011 3rd International Conference on Advanced Computer Control.

Rahman, M. G., & Islam, M. (2011). *A Decision Tree-based Missing Value Imputation Technique for Data Pre-processing*.

Raschka, S. (2014). Naive Bayes and Text Classification I - Introduction and Theory. doi:10.13140/2.1.2018.3049

Raschka, S., & Mirjalili, V. (2019). *Python Machine Learning*: Packt.

Rashidi, A., & Movahhedinia, N. (2012). *A Model for User Trust in Cloud Computing.* Paper presented at the CloudCom 2012.

Raza, M., Hussain, F. K., Hussain, O. K., Zhao, M., & Rehman, Z. u. (2019). A comparative analysis of machine learning models for quality pillar assessment of SaaS services by multi-class text classification of users' reviews. *Future Generation Computer Systems, 101*, 341-371. doi:https://doi.org/10.1016/j.future.2019.06.022

Resnick, P., & Zeckhauser, R. (2002). Trust among strangers in internet transactions: Empirical analysis of eBay' s reputation system. In M. R. Baye (Ed.), *The Economics of the Internet and E-commerce* (Vol. 11, pp. 127-157): Emerald Group Publishing Limited.

Richert, W., & Coelho, L. P. (2013). *Building Machine Learning Systems with Python*: Packt Publishing.

Ries, S. (2009a). *Extending Bayesian trust models regarding context-dependence and user friendly representation*. Paper presented at the Proceedings of the 2009 ACM symposium on Applied Computing, Honolulu, Hawaii. https://doi.org/10.1145/1529282.1529573

Ries, S. (2009b). *Trust in ubiquitous computing.* Retrieved from http://tuprints.ulb.tu-darmstadt.de/1948/

https://nbn-resolving.org/urn:nbn:de:tuda-tuprints-19486

http://d-nb.info/997932635

http://d-nb.info/998730459

Rizvi, S., Mitchell, J., Razaque, A., Rizvi, M. R., & Williams, I. (2020). A fuzzy inference system (FIS) to evaluate the security readiness of cloud service providers. *Journal of Cloud Computing, 9*(1), 42. doi:10.1186/s13677-020-00192-9

Rizvi, S., Ryoo, J., Liu, Y., Zazworsky, D., & Cappeta, A. (2014, 9-10 May 2014). *A centralized trust model approach for cloud computing.* Paper presented at the 2014 23rd Wireless and Optical Communication Conference (WOCC).

Rockel, T., Joenssen, D. W., & Bankhofer, U. (2017). *Decision Trees for the Imputation of Categorical Data*.

Rubin, D. B. (1976). Inference and Missing Data. *Biometrika, 63*(3), 581-592. doi:10.2307/2335739

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations* (pp. 318–362): MIT Press.

Sabater, J., & Sierra, C. (2002). *Reputation and social network analysis in multi-agent systems*. Paper presented at the Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, Bologna, Italy. https://doi.org/10.1145/544741.544854

Sakakibara, Y., Misue, K., & Koshiba, T. (1993, 1-5 March 1993). *Text classification and keyword extraction by learning decision trees.* Paper presented at the Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications.

Salton, G. (1991). Developments in Automatic Text Retrieval. *Science, 253*(5023), 974-980.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management, 24*(5), 513-523. doi:https://doi.org/10.1016/0306-4573(88)90021-0

Saravanakumar, C., & Arun, C. (2014, 27-29 Nov. 2014). *Survey on interoperability, security, trust, privacy standardization of cloud computing.* Paper presented at the 2014 International Conference on Contemporary Computing and Informatics (IC3I).

Schapire, R. E., & Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning, 37*(3), 297-336. doi:10.1023/A:1007614523901

Schapire, R. E., & Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning, 39*(2), 135-168. doi:10.1023/A:1007649029923

Schryen, G., Volkamer, M., Ries, S., & Habib, S. M. (2011). *A formal approach towards measuring trust in distributed systems*. Paper presented at the Proceedings of the 2011 ACM Symposium on Applied Computing, TaiChung, Taiwan. https://doi.org/10.1145/1982185.1982548

Scikit-learn. (2017a). sklearn.metrics.hamming_loss. Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.hamming_loss.html

Scikit-learn. (2017b). sklearn.metrics.matthews_corrcoef. Retrieved from http://scikit-learn.org/stable/modules/generated/sklearn.metrics

Scrapy. Scrapy. Retrieved from https://scrapy.org/

Selvaraj, A., & Sundararajan, S. (2017). Evidence-Based Trust Evaluation System for Cloud Services Using Fuzzy Logic. *International Journal of Fuzzy Systems, 19*(2), 329-337. doi:10.1007/s40815-016-0146-4

Serchen. (2017). Serchen. Retrieved from https://www.serchen.com

Shalev-Shwartz, S., & Singer, Y. (2005, 2005//). *A New Perspective on an Old Perceptron Algorithm.* Paper presented at the Learning Theory, Berlin, Heidelberg.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427-437. doi:https://doi.org/10.1016/j.ipm.2009.03.002

Soleymani, M., Abapour, N., Taghizadeh, E., Siadat, S., & Karkehabadi, R. (2021). Fuzzy Rule-Based Trust Management Model for the Security of Cloud Computing. *Mathematical Problems in Engineering, 2021*, 6629449. doi:10.1155/2021/6629449

Somesh Kumar Prajapati, Suvamoy Changder, & Sarkar, A. (2013). *Trust Management Model for Cloud Computing Environment*. Paper presented at the International Conference on Computing Communication and Advanced Network, India.

Soucy, P., & Mineau, G. W. (2001). *A Simple KNN Algorithm for Text Categorization*. Paper presented at the Proceedings of the 2001 IEEE International Conference on Data Mining.

Sportisse, A., Boyer, C., Dieuleveut, A., & Josse, J. (2020). *Debiasing Stochastic Gradient Descent to handle missing values*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res., 15*(1), 1929–1958.

Stanoevska-Slabeva, K., Wozniak, T., & Ristol, S. (2009). *Grid and Cloud Computing: A Business Perspective on Technology and Applications*: Springer Publishing Company, Incorporated.

Sule, M., Li, M., & Taylor, G. (2016, 29 March-2 April 2016). *Trust Modeling in Cloud Computing.* Paper presented at the 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE).

Sun, X., Chang, G., & Li, F. (2011, 21-24 Sept. 2011). *A Trust Management Model to Enhance Security of Cloud Computing Environments.* Paper presented at the 2011 Second International Conference on Networking and Distributed Computing.

Tan, L., Chi, C., & Deng, J. (2008, 28 July-1 Aug. 2008). *Quantifying Trust Based on Service Level Agreement for Software as a Service.* Paper presented at the 2008 32nd Annual IEEE International Computer Software and Applications Conference.

Tang, C., & Liu, J. (2015). Selecting a trusted cloud service provider for your SaaS program. *Computers & Security, 50*, 60-73. doi:https://doi.org/10.1016/j.cose.2015.02.001

Toman, M., Tesar, R., & Jezek, K. (2006). Influence of Word Normalization on Text Classification.

Trabay, D. W., El-Henawy, I., & Gharibi, W. (2021). A Trust Framework Utilization in Cloud Computing Environment Based on Multi-criteria Decision-Making Methods. *The Computer Journal*. doi:10.1093/comjnl/bxaa138

Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., . . . Altman, R. B. (2001). Missing value estimation methods for DNA microarrays. *Bioinformatics, 17*(6), 520–525. doi:https://doi.org/10.1093/bioinformatics/17.6.520

Trstenjak, B., Mikac, S., & Donko, D. (2014). KNN with TF-IDF based Framework for Text Categorization. *Procedia Engineering, 69*, 1356-1364. doi:https://doi.org/10.1016/j.proeng.2014.03.129

Tsoumakas, G., & Katakis, I. (2009). Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining, 3*, 1-13. doi:10.4018/jdwm.2007070101

Uusitalo, I., Karppinen, K., Juhola, A., & Savola, R. (2010, 30 Nov.-3 Dec. 2010). *Trust and Cloud Services - An Interview Study.* Paper presented at the 2010 IEEE Second International Conference on Cloud Computing Technology and Science.

Vapnik, V. (1999). *The Nature of Statistical Learning Theory*: Springer New York.

Vapnik, V. N. (1998). *Statistical Learning Theory*: Wiley-Interscience.

Varadharajan, V. (2009). A Note on Trust-Enhanced Security. *IEEE Security & Privacy, 7*(3), 57-59. doi:10.1109/MSP.2009.59

Vateekul, P., & Sarinnapakorn, K. (2009). *Tree-Based Approach to Missing Data Imputation*.

Wang, L., & Wu, Z. (2014, 8-11 Dec. 2014). *A Novel Trustworthiness Measurement Model for Cloud Service.* Paper presented at the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing.

Wang, T.-Y., & Chiang, H.-M. (2007). Fuzzy support vector machine for multi-class text categorization. *Information Processing & Management, 43*(4), 914-929. doi:https://doi.org/10.1016/j.ipm.2006.09.011

Wang, Y., & Singh, M. P. (2007). *Formal trust model for multiagent systems*. Paper presented at the Proceedings of the 20th international joint conference on Artifical intelligence, Hyderabad, India.

Wang, Y., & Vassileva, J. (2003, 13-17 Oct. 2003). *Bayesian network-based trust model.* Paper presented at the Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003).

Wang, Y., Wen, J., Zhou, W., & Luo, F. (2018, 1-3 Aug. 2018). *A Novel Dynamic Cloud Service Trust Evaluation Model in Cloud Computing.* Paper presented at the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE).

Whitby, A., Jøsang, A., & Indulska, J. (2004). *Filtering Out Unfair Ratings in Bayesian Reputation Systems*.

Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., . . . Yu, L. (2002). Negotiating trust in the Web. *IEEE Internet Computing, 6*(6), 30-37. doi:10.1109/MIC.2002.1067734

Winters, P. R. (1976). Forecasting Sales by Exponentially Weighted Moving Averages. In *Mathematical Models in Marketing: A Collection of Abstracts* (pp. 384-386). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wu, C.-H., Ho, J.-M., & Lee, D. (2005). Travel-Time Prediction With Support Vector Regression. *Intelligent Transportation Systems, IEEE Transactions on, 5*, 276-281. doi:10.1109/TITS.2004.837813

Wu, X. (2011). A Stable Group-based Trust Management Scheme for Mobile P2P Networks. *International Journal of Digital Content Technology and Its Applications, 5*, 116-125. doi:10.4156/jdcta.vol5.issue2.13

Wu, X. (2012). A Fuzzy Reputation-based Trust Management Scheme for Cloud Computing. *International Journal of Digital Content Technology and Its Applications, 6*, 437-445.

Xiao, Y., Lin, C., Jiang, Y., Chu, X., & Shen, X. (2010, 23-27 May 2010). *Reputation-Based QoS Provisioning in Cloud Computing via Dirichlet Multinomial Model.* Paper presented at the 2010 IEEE International Conference on Communications.

Xiong, L., & Liu, L. (2003). *A reputation-based trust model for peer-to-peer ecommerce communities [Extended Abstract]*. Paper presented at the Proceedings of the 4th ACM conference on Electronic commerce, San Diego, CA, USA. https://doi.org/10.1145/779928.779972

Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval, 1*(1), 69-90. doi:10.1023/A:1009982220290

Yang, Y., & Liu, X. (1999). *A re-examination of text categorization methods*. Paper presented at the Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, Berkeley, California, USA. https://doi.org/10.1145/312624.312647

Yang, Y., Zhang, J., & Kisiel, B. (2003). *A scalability analysis of classifiers in text categorization*. Paper presented at the Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada. https://doi.org/10.1145/860435.860455

Yang, Z., Qiao, L., Liu, C., Yang, C., & Wan, G. (2010, 14-16 April 2010). *A collaborative trust model of firewall-through based on Cloud Computing.* Paper presented at the The 2010 14th International Conference on Computer Supported Cooperative Work in Design.

Yu, P.-S., Chen, S.-T., & Chang, I. F. (2006). Support vector regression for real-time flood stage forecasting. *Journal of Hydrology, 328*(3), 704-716. doi:https://doi.org/10.1016/j.jhydrol.2006.01.021

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338-353. doi:https://doi.org/10.1016/S0019-9958(65)90241-X

Zhang, J., & Yang, Y. (2003). *Robustness of regularized linear classification methods in text categorization*. Paper presented at the Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada. https://doi.org/10.1145/860435.860471

Zhou, H., Shi, W., Liang, Z., & Liang, B. (2011). Using new fusion operations to improve trust expressiveness of subjective logic. *Wuhan University Journal of Natural Sciences, 16*(5), 376. doi:10.1007/s11859-011-0766-3

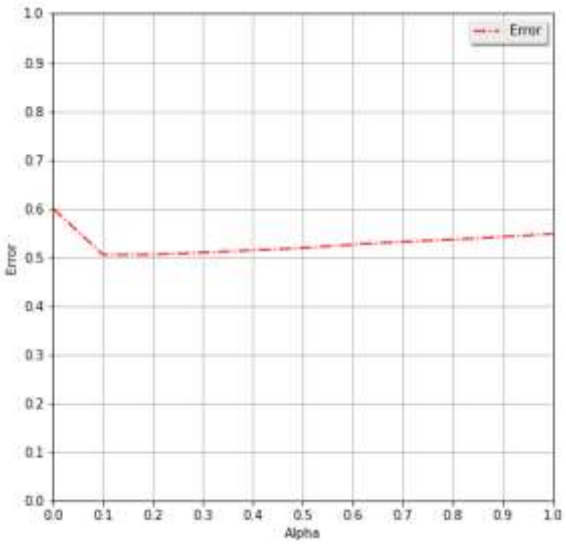
Zissis, D., & Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation Computer Systems, 28*(3), 583-592. doi:https://doi.org/10.1016/j.future.2010.12.006

# Appendices

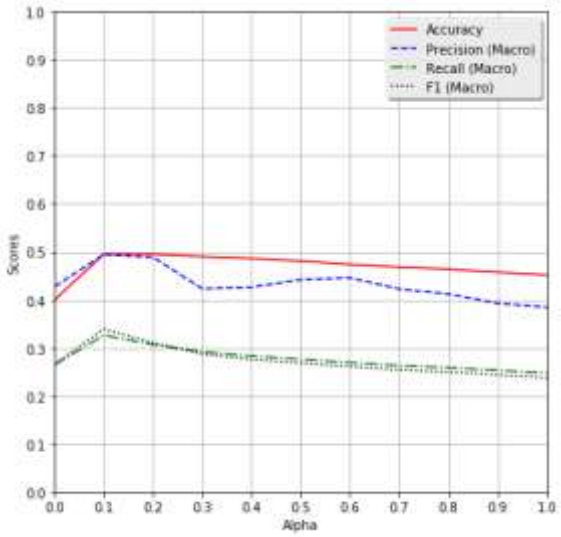
## Appendix A

### Parameter tunning and CV results from Chapter 5

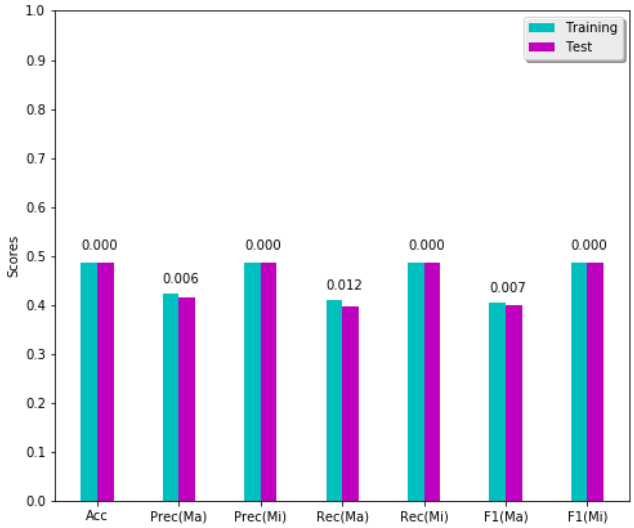
*Naïve Bayes (Multinomial)*




NB(M) - Cross Validation errors with different values of alpha

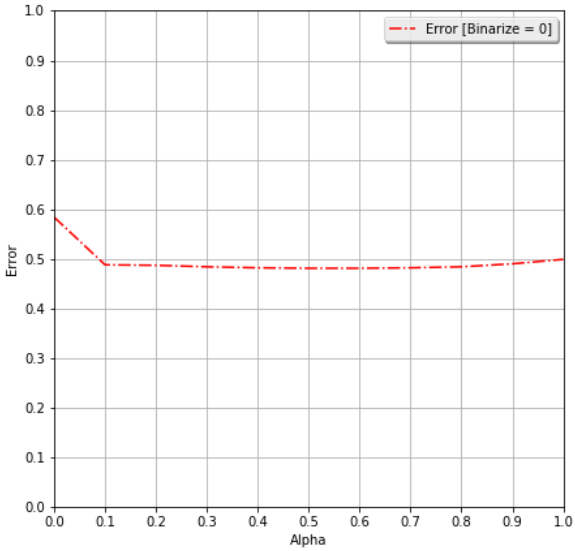



NB(M) - Cross validation with different value of alpha

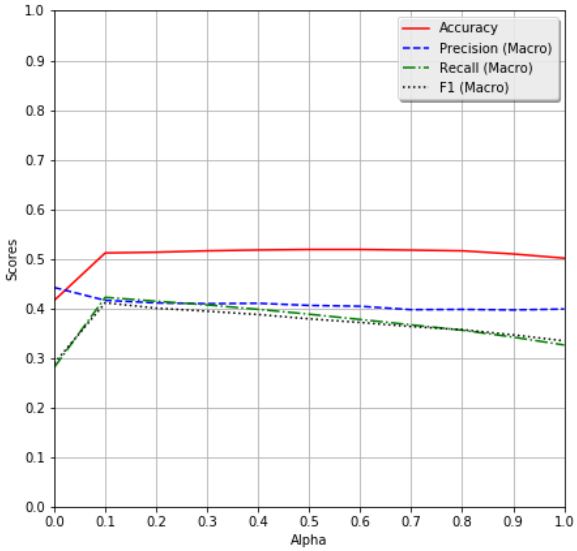



NB(M) - Training vs. Test performance comparison

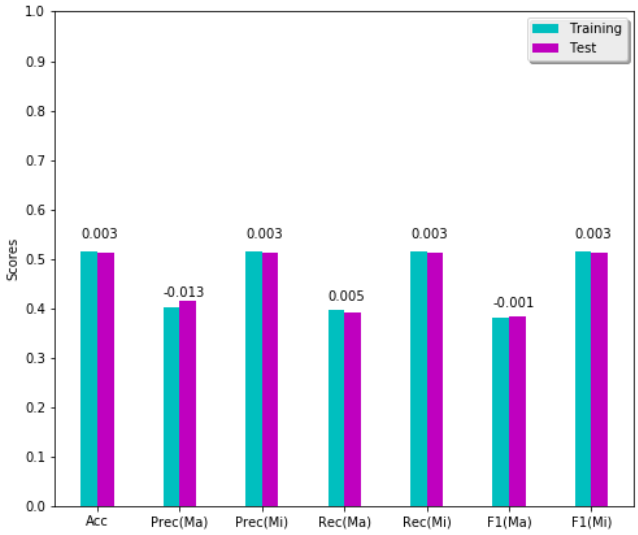
*Naïve Bayes (Bernoulli)*




Errors with different values of alpha during cross validation on training data. Binarize threshold is set to 0 (optimal)

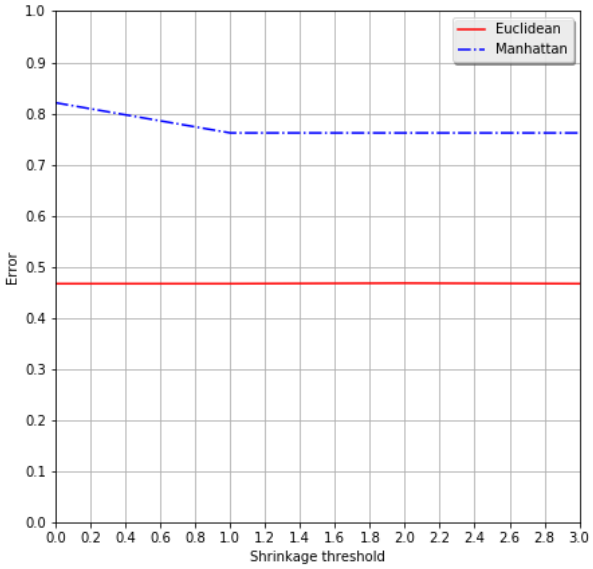



Cross validation of Naive Bayes Bernoulli with different metrics over Binarize threshold of 0

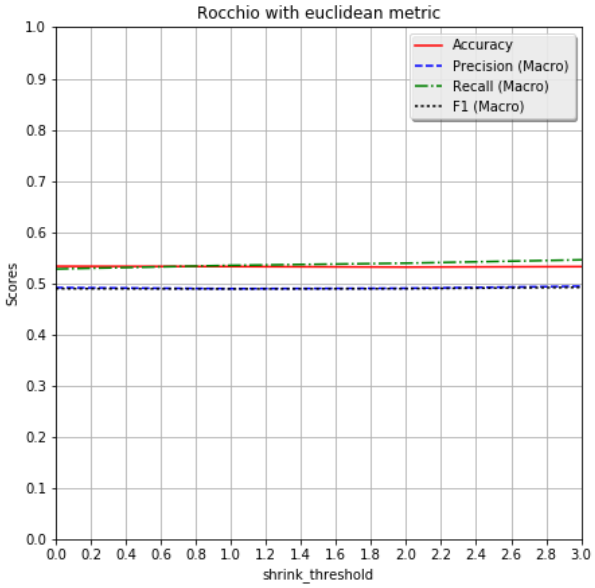



NB (B) - Training vs. Test performance comparison

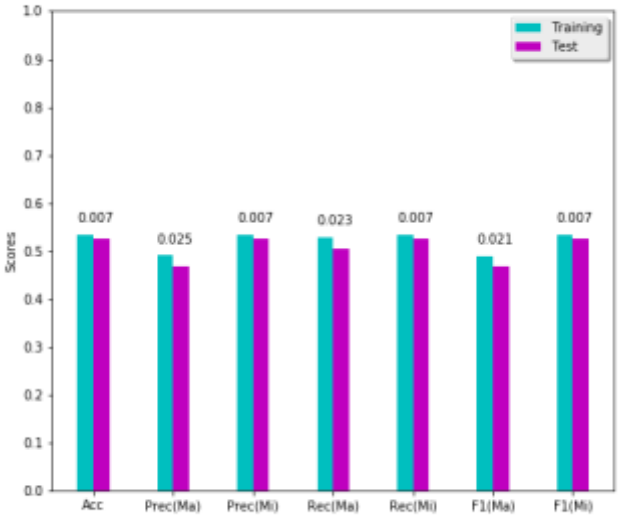
*Nearest Centroid (Rocchio)*




Error values during cross validation of Nearest Centroid with Euclidean and Manhattan measures

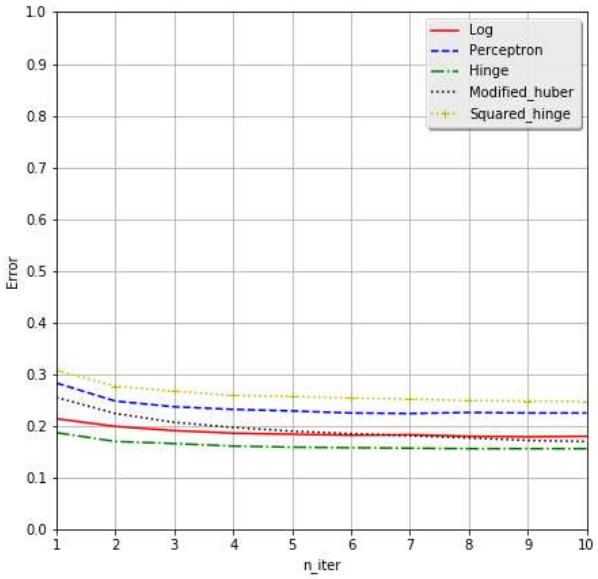



Cross Validation performance scores of Nearest Centroid with different performance metrics

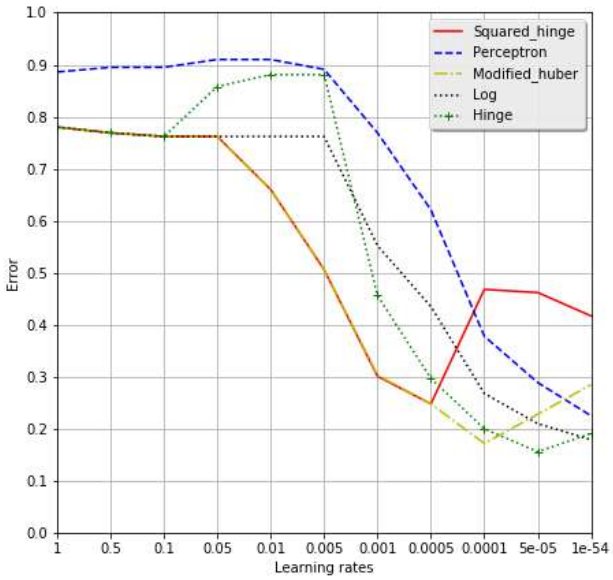



Nearest Centroid (Rocchio) - Training vs. Test performance comparison

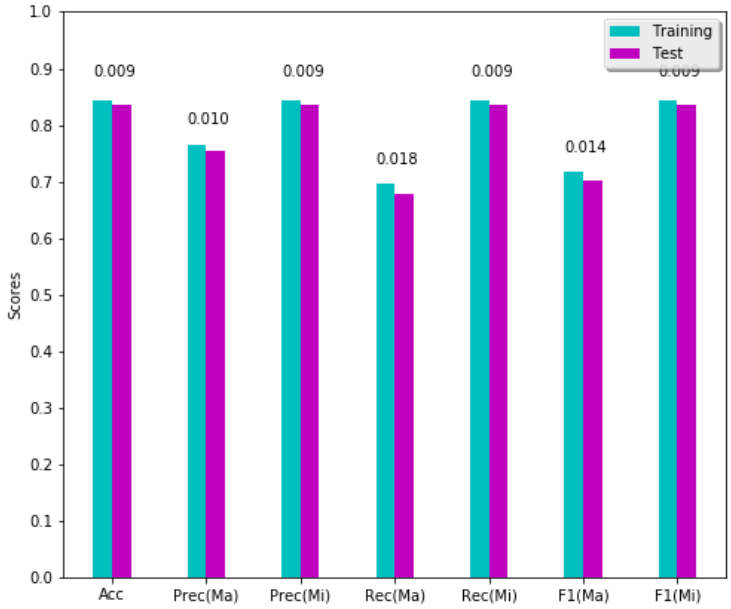
*Stochastic Gradient Descent (SGD)*




Classification Error by increasing the number of iterations on training dataset

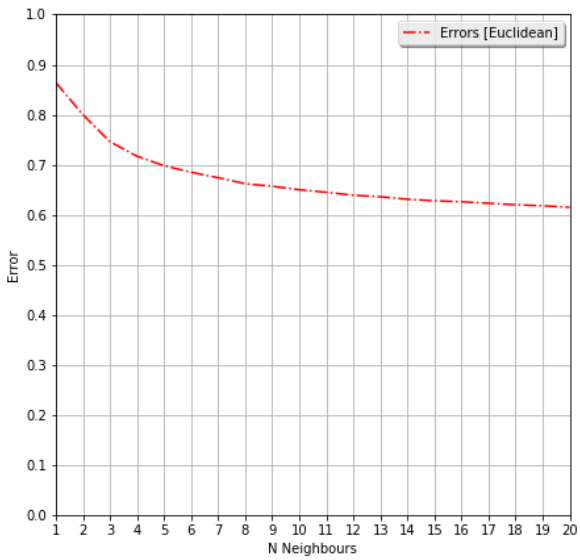



Classification Error with decreasing learning rates during training (n_iter=9)

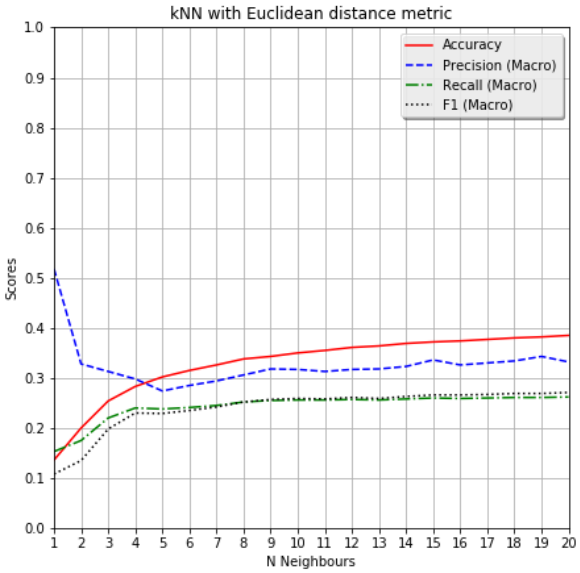



SGD - Training vs. Test performance comparison

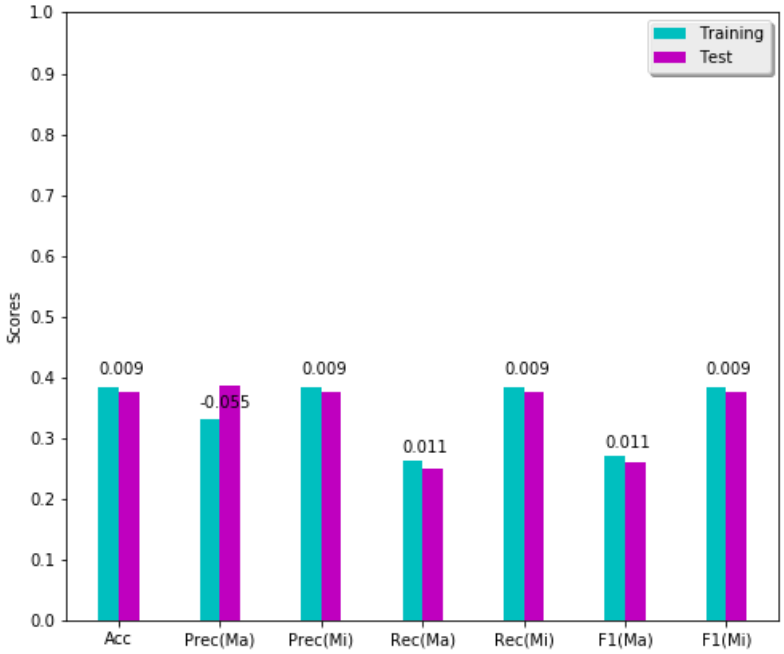
*K-NN*




Classification error by increasing the number (k) of N Neighbours on training data

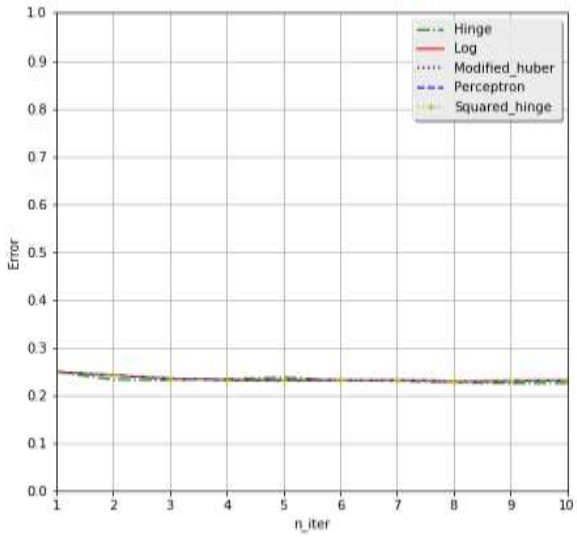



Performance scores with gradual increase in N Neighbours on training data

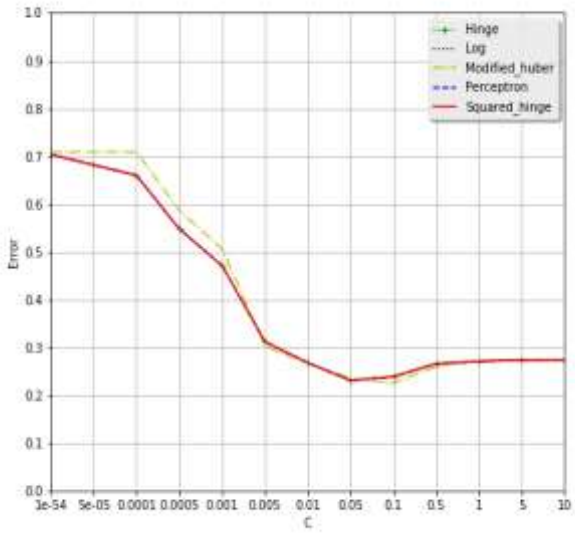



k-NN - Training vs. Test performance comparison

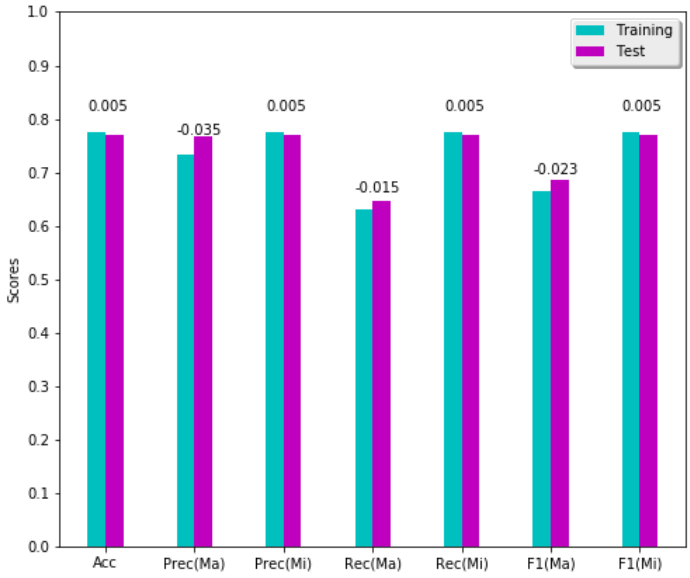
*Passive Aggressive*




Classification error by increasing the number of iterations on training data

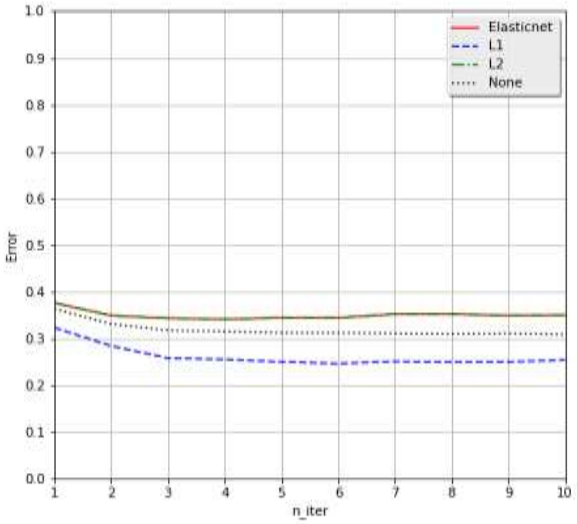



Classification error by increasing the value of regularization parameter 'C' on training data

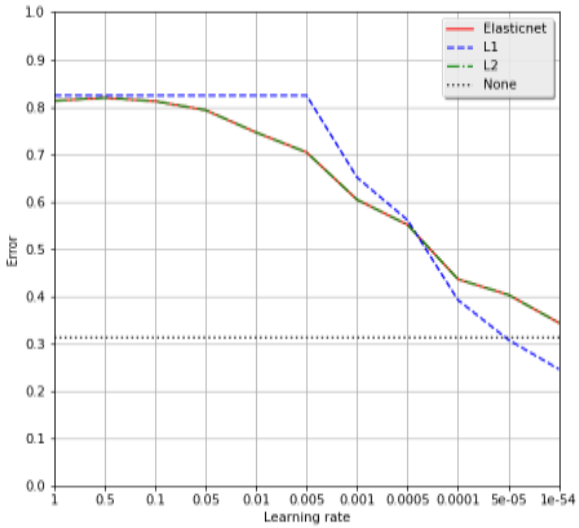



Passive Aggressive - Training vs. Test performance comparison

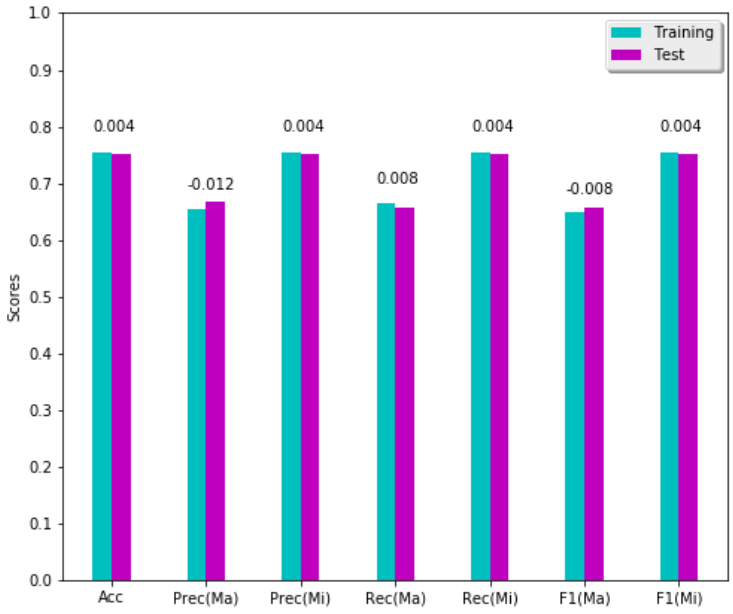
*Perceptron*




Perceptron Classification errors by increasing the number of iterations on training data

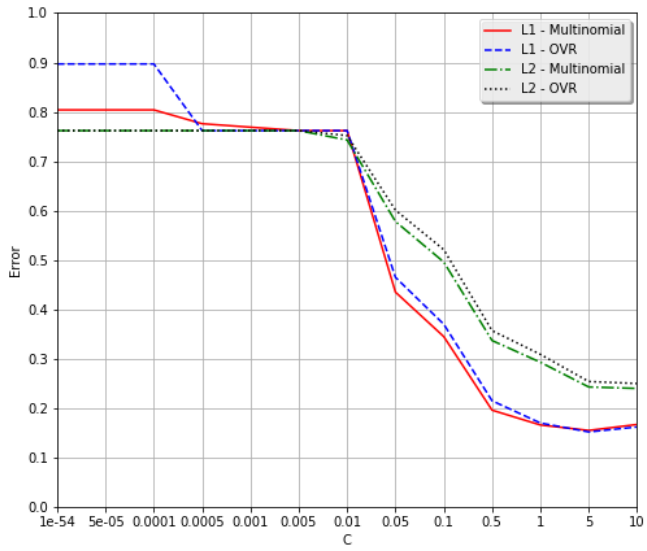



Perceptron Classification errors by decreasing the learning rate during training

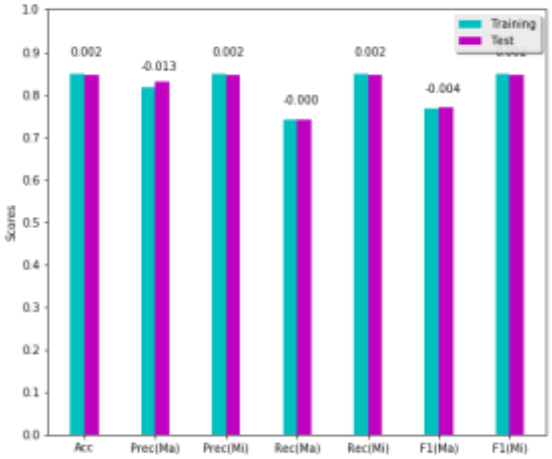



Perceptron - Training vs. Test performance comparison

## *Logistic Regression*

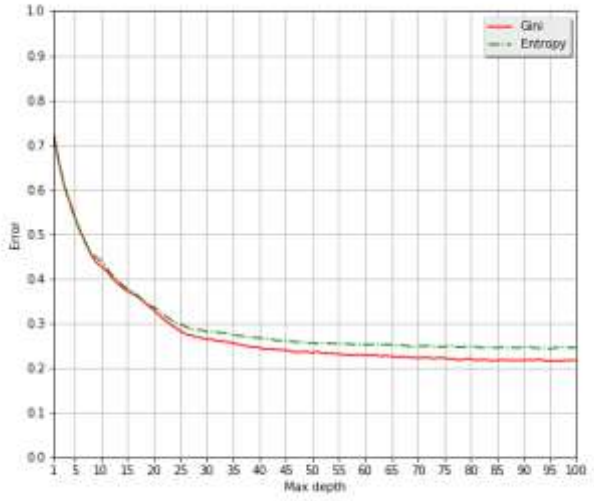



Logistic Regression Classification errors by increasing the inverse regularization parameter during training

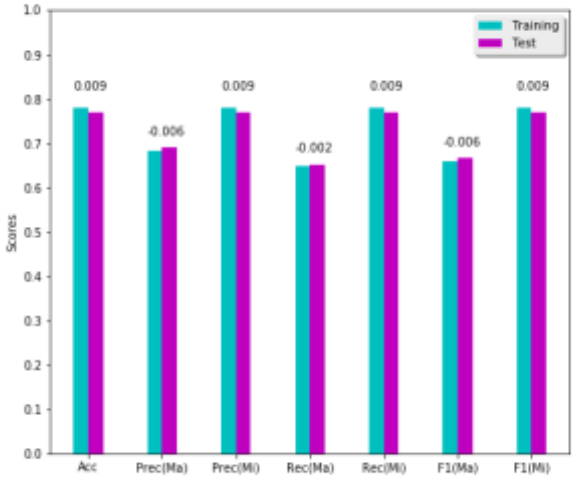



Logistic Regression - Training vs. Test performance comparison

*Decision Tree*

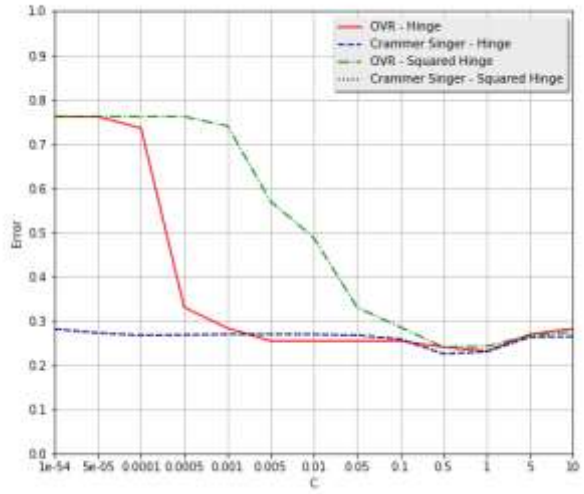



Decision tree classification error by increasing the depth of the tree on training data

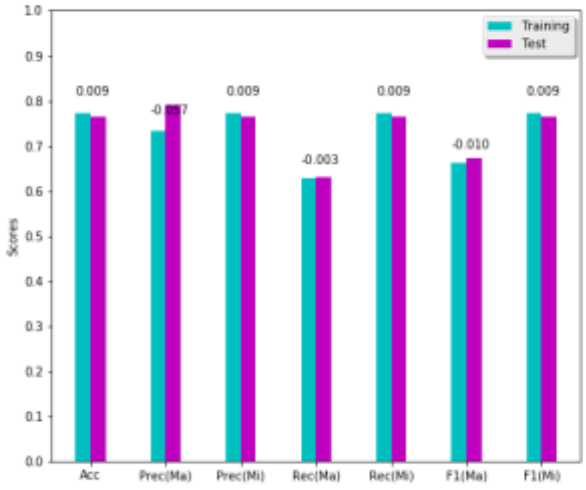



Decision tree - Training vs. Test performance comparison

*Linear SVM (LSVC)*

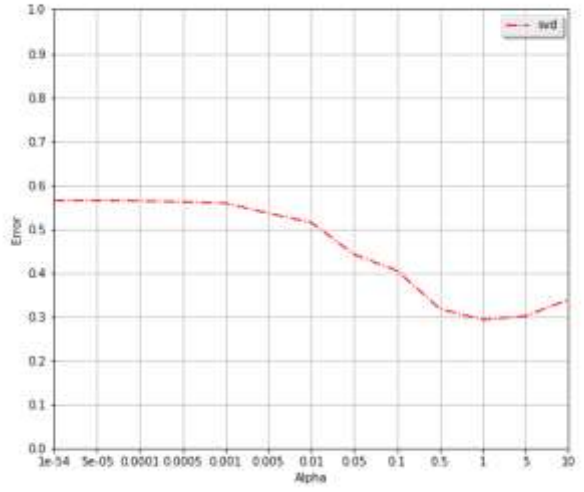



Linear SVC Classification errors during cross validation on training data by increasing C

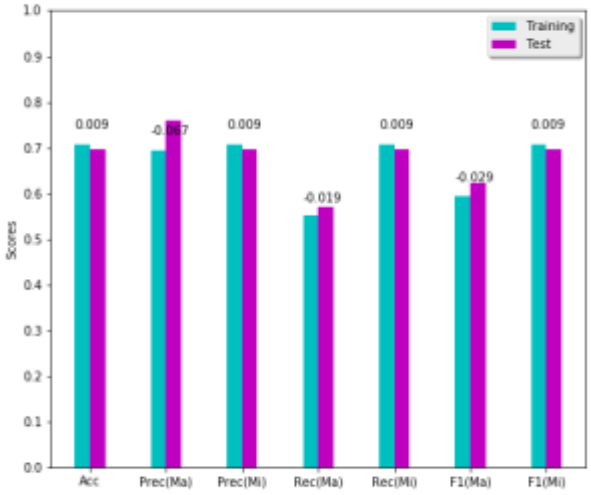



Linear SVC- Training vs. Test performance comparison

*Ridge*




Classification errors using Ridge - with increasing alpha




Ridge - Training vs. Test performance comparison

# Appendix B

## Fuzzy Rules

Complete set of fuzzy rules generated using the MDR metrics in Chapter 8.

| | | SaaS service aspects (input variables) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MDR | 1.00 | 0.95 | 0.37 | 0.36 | 0.32 | 0.29 | 0.22 | 0.04 | 0.03 | 0.00 | |
| | Rule | Unknown is | Management is | Performance is | Reliability is | Cost is | Operational Excellence is | Availability is | Scalability is | Security is | Resiliency is | Then Trust level is |
| Category A | 1 | Negative | Negative | Negative | Negative | - | - | - | - | - | - | Very Untrustworthy |
| | 2 | Neutral | Neutral | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 3 | Positive | Positive | Positive | Positive | - | - | - | - | - | - | Very Trustworthy |
| Category B | 4 | Negative | Negative | Negative | - | - | - | - | - | - | - | Untrustworthy |
| | 5 | Neutral | Neutral | Neutral | - | - | - | - | - | - | - | Neutral |
| | 6 | Positive | Positive | Positive | - | - | - | - | - | - | - | Trustworthy |
| | 7 | Negative | Negative | - | Negative | - | - | - | - | - | - | Untrustworthy |
| | 8 | Neutral | Neutral | - | Neutral | - | - | - | - | - | - | Neutral |
| | 9 | Positive | Positive | - | Positive | - | - | - | - | - | - | Trustworthy |
| | 10 | Negative | - | Negative | Negative | - | - | - | - | - | - | Untrustworthy |
| | 11 | Neutral | - | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 12 | Positive | - | Positive | Positive | - | - | - | - | - | - | Trustworthy |
| | 13 | - | Negative | Negative | Negative | - | - | - | - | - | - | Untrustworthy |
| | 14 | - | Neutral | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 15 | - | Positive | Positive | Positive | - | - | - | - | - | - | Trustworthy |
| Category C | 16 | Positive | Positive | Neutral | Neutral | - | - | - | - | - | - | Trustworthy |
| | 17 | Positive | Neutral | Positive | Neutral | - | - | - | - | - | - | Trustworthy |
| | 18 | Positive | Neutral | Neutral | Positive | - | - | - | - | - | - | Trustworthy |
| | 19 | Neutral | Positive | Positive | Neutral | - | - | - | - | - | - | Trustworthy |
| | 20 | Neutral | Positive | Neutral | Positive | - | - | - | - | - | - | Trustworthy |
| | 21 | Neutral | Neutral | Positive | Positive | - | - | - | - | - | - | Trustworthy |
| | 22 | Negative | Negative | Neutral | Neutral | - | - | - | - | - | - | Untrustworthy |
| | 23 | Negative | Neutral | Negative | Neutral | - | - | - | - | - | - | Untrustworthy |
| | 24 | Negative | Neutral | Neutral | Negative | - | - | - | - | - | - | Untrustworthy |
| | 25 | Neutral | Negative | Negative | Neutral | - | - | - | - | - | - | Untrustworthy |
| | 26 | Neutral | Negative | Neutral | Negative | - | - | - | - | - | - | Untrustworthy |
| | 27 | Neutral | Neutral | Negative | Negative | - | - | - | - | - | - | Untrustworthy |

| | | SaaS service aspects (input variables) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MDR | 1.00 | 0.95 | 0.37 | 0.36 | 0.32 | 0.29 | 0.22 | 0.04 | 0.03 | 0.00 | |
| | Rule | Unknown is | Management is | Performance is | Reliability is | Cost is | Operational Excellence is | Availability is | Scalability is | Security is | Resiliency is | Then Trust level is |
| Category D | 28 | Neutral | Neutral | Positive | - | - | - | - | - | - | - | Neutral |
| | 29 | Neutral | Neutral | - | Positive | - | - | - | - | - | - | Neutral |
| | 30 | Neutral | Positive | Neutral | - | - | - | - | - | - | - | Neutral |
| | 31 | Neutral | - | Neutral | Positive | - | - | - | - | - | - | Neutral |
| | 32 | Neutral | Positive | - | Neutral | - | - | - | - | - | - | Neutral |
| | 33 | Neutral | - | Positive | Neutral | - | - | - | - | - | - | Neutral |
| | 34 | Positive | Neutral | Neutral | - | - | - | - | - | - | - | Neutral |
| | 35 | - | Neutral | Neutral | Positive | - | - | - | - | - | - | Neutral |
| | 36 | Positive | Neutral | - | Neutral | - | - | - | - | - | - | Neutral |
| | 37 | - | Neutral | Positive | Neutral | - | - | - | - | - | - | Neutral |
| | 38 | Positive | - | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 39 | - | Positive | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 40 | Neutral | Neutral | Negative | - | - | - | - | - | - | - | Neutral |
| | 41 | Neutral | Neutral | - | Negative | - | - | - | - | - | - | Neutral |
| | 42 | Neutral | Negative | Neutral | - | - | - | - | - | - | - | Neutral |
| | 43 | Neutral | - | Neutral | Negative | - | - | - | - | - | - | Neutral |
| | 44 | Neutral | Negative | - | Neutral | - | - | - | - | - | - | Neutral |
| | 45 | Neutral | - | Negative | Neutral | - | - | - | - | - | - | Neutral |
| | 46 | Negative | Neutral | Neutral | - | - | - | - | - | - | - | Neutral |
| | 47 | - | Neutral | Neutral | Negative | - | - | - | - | - | - | Neutral |
| | 48 | Negative | Neutral | - | Neutral | - | - | - | - | - | - | Neutral |
| | 49 | - | Neutral | Negative | Neutral | - | - | - | - | - | - | Neutral |
| | 50 | Negative | - | Neutral | Neutral | - | - | - | - | - | - | Neutral |
| | 51 | - | Negative | Neutral | Neutral | - | - | - | - | - | - | Neutral |

| | | SaaS service aspects (input variables) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MDR | 1.00 | 0.95 | 0.37 | 0.36 | 0.32 | 0.29 | 0.22 | 0.04 | 0.03 | 0.00 | |
| | Rule | Unknown is | Management is | Performance is | Reliability is | Cost is | Operational Excellence is | Availability is | Scalability is | Security is | Resiliency is | Then Trust level is |
| Category E | 52 | Negative | Negative | - | - | - | - | Negative | Negative | Negative | Negative | Untrustworthy |
| | 53 | Neutral | Neutral | - | - | - | - | Neutral | Neutral | Neutral | Neutral | Neutral |
| | 54 | Positive | Positive | - | - | - | - | Positive | Positive | Positive | Positive | Trustworthy |
| | 55 | Negative | Negative | - | - | - | Negative | - | Negative | Negative | Negative | Untrustworthy |
| | 56 | Neutral | Neutral | - | - | - | Neutral | - | Neutral | Neutral | Neutral | Neutral |
| | 57 | Positive | Positive | - | - | - | Positive | - | Positive | Positive | Positive | Trustworthy |
| | 58 | Negative | Negative | - | - | Negative | - | - | Negative | Negative | Negative | Untrustworthy |
| | 59 | Neutral | Neutral | - | - | Neutral | - | - | Neutral | Neutral | Neutral | Neutral |
| | 60 | Positive | Positive | - | - | Positive | - | - | Positive | Positive | Positive | Trustworthy |
| | 61 | Negative | Negative | - | - | Negative | Negative | - | - | - | - | Untrustworthy |
| | 62 | Neutral | Neutral | - | - | Neutral | Neutral | - | - | - | - | Neutral |
| | 63 | Positive | Positive | - | - | Positive | Positive | - | - | - | - | Trustworthy |
| | 64 | Negative | Negative | - | - | Negative | - | Negative | - | - | - | Untrustworthy |
| | 65 | Neutral | Neutral | - | - | Neutral | - | Neutral | - | - | - | Neutral |
| | 66 | Positive | Positive | - | - | Positive | - | Positive | - | - | - | Trustworthy |
| | 67 | Negative | Negative | - | - | - | Negative | Negative | - | - | - | Untrustworthy |
| | 68 | Neutral | Neutral | - | - | - | Neutral | Neutral | - | - | - | Neutral |
| | 69 | Positive | Positive | - | - | - | Positive | Positive | - | - | - | Trustworthy |
| | 70 | Negative | - | Negative | - | - | - | Negative | Negative | Negative | Negative | Untrustworthy |
| | 71 | Neutral | - | Neutral | - | - | - | Neutral | Neutral | Neutral | Neutral | Neutral |
| | 72 | Positive | - | Positive | - | - | - | Positive | Positive | Positive | Positive | Trustworthy |
| | 73 | Negative | - | Negative | - | - | Negative | - | Negative | Negative | Negative | Untrustworthy |
| | 74 | Neutral | - | Neutral | - | - | Neutral | - | Neutral | Neutral | Neutral | Neutral |
| | 75 | Positive | - | Positive | - | - | Positive | - | Positive | Positive | Positive | Trustworthy |
| | 76 | Negative | - | Negative | - | Negative | - | - | Negative | Negative | Negative | Untrustworthy |
| | 77 | Neutral | - | Neutral | - | Neutral | - | - | Neutral | Neutral | Neutral | Neutral |
| | 78 | Positive | - | Positive | - | Positive | - | - | Positive | Positive | Positive | Trustworthy |
| | 79 | Negative | - | Negative | - | Negative | Negative | - | - | - | - | Untrustworthy |
| | 80 | Neutral | - | Neutral | - | Neutral | Neutral | - | - | - | - | Neutral |
| | 81 | Positive | - | Positive | - | Positive | Positive | - | - | - | - | Trustworthy |
| | 82 | Negative | - | Negative | - | Negative | - | Negative | - | - | - | Untrustworthy |
| | 83 | Neutral | - | Neutral | - | Neutral | - | Neutral | - | - | - | Neutral |
| | 84 | Positive | - | Positive | - | Positive | - | Positive | - | - | - | Trustworthy |
| | 85 | Negative | - | Negative | - | - | Negative | Negative | - | - | - | Untrustworthy |
| | 86 | Neutral | - | Neutral | - | - | Neutral | Neutral | - | - | - | Neutral |
| | 87 | Positive | - | Positive | - | - | Positive | Positive | - | - | - | Trustworthy |
| | 88 | Negative | - | - | Negative | - | - | Negative | Negative | Negative | Negative | Untrustworthy |
| | 89 | Neutral | - | - | Neutral | - | - | Neutral | Neutral | Neutral | Neutral | Neutral |
| | 90 | Positive | - | - | Positive | - | - | Positive | Positive | Positive | Positive | Trustworthy |
| | 91 | Negative | - | - | Negative | - | Negative | - | Negative | Negative | Negative | Untrustworthy |
| | 92 | Neutral | - | - | Neutral | - | Neutral | - | Neutral | Neutral | Neutral | Neutral |
| | 93 | Positive | - | - | Positive | - | Positive | - | Positive | Positive | Positive | Trustworthy |
| | 94 | Negative | - | - | Negative | Negative | - | - | Negative | Negative | Negative | Untrustworthy |
| | 95 | Neutral | - | - | Neutral | Neutral | - | - | Neutral | Neutral | Neutral | Neutral |
| | 96 | Positive | - | - | Positive | Positive | - | - | Positive | Positive | Positive | Trustworthy |
| | 97 | Negative | - | - | Negative | Negative | Negative | - | - | - | - | Untrustworthy |
| | 98 | Neutral | - | - | Neutral | Neutral | Neutral | - | - | - | - | Neutral |
| | 99 | Positive | - | - | Positive | Positive | Positive | - | - | - | - | Trustworthy |
| | 100 | Negative | - | - | Negative | Negative | - | Negative | - | - | - | Untrustworthy |
| | 101 | Neutral | - | - | Neutral | Neutral | - | Neutral | - | - | - | Neutral |
| | 102 | Positive | - | - | Positive | Positive | - | Positive | - | - | - | Trustworthy |
| | 103 | Negative | - | - | Negative | - | Negative | Negative | - | - | - | Untrustworthy |
| | 104 | Neutral | - | - | Neutral | - | Neutral | Neutral | - | - | - | Neutral |
| | 105 | Positive | - | - | Positive | - | Positive | Positive | - | - | - | Trustworthy |
| | 106 | - | Negative | Negative | - | - | - | Negative | Negative | Negative | Negative | Untrustworthy |
| | 107 | - | Neutral | Neutral | - | - | - | Neutral | Neutral | Neutral | Neutral | Neutral |
| | 108 | - | Positive | Positive | - | - | - | Positive | Positive | Positive | Positive | Trustworthy |
| | 109 | - | Negative | Negative | - | - | Negative | - | Negative | Negative | Negative | Untrustworthy |
| | 110 | - | Neutral | Neutral | - | - | Neutral | - | Neutral | Neutral | Neutral | Neutral |
| | 111 | - | Positive | Positive | - | - | Positive | - | Positive | Positive | Positive | Trustworthy |
| | 112 | - | Negative | Negative | - | Negative | - | - | Negative | Negative | Negative | Untrustworthy |
| | 113 | - | Neutral | Neutral | - | Neutral | - | - | Neutral | Neutral | Neutral | Neutral |
| | 114 | - | Positive | Positive | - | Positive | - | - | Positive | Positive | Positive | Trustworthy |
| | 115 | - | Negative | Negative | - | Negative | Negative | - | - | - | - | Untrustworthy |
| | 116 | - | Neutral | Neutral | - | Neutral | Neutral | - | - | - | - | Neutral |
| | 117 | - | Positive | Positive | - | Positive | Positive | - | - | - | - | Trustworthy |
| | 118 | - | Negative | Negative | - | Negative | - | Negative | - | - | - | Untrustworthy |
| | 119 | - | Neutral | Neutral | - | Neutral | - | Neutral | - | - | - | Neutral |
| | 120 | - | Positive | Positive | - | Positive | - | Positive | - | - | - | Trustworthy |
| | 121 | - | Negative | Negative | - | - | Negative | Negative | - | - | - | Untrustworthy |
| | 122 | - | Neutral | Neutral | - | - | Neutral | Neutral | - | - | - | Neutral |
| | 123 | - | Positive | Positive | - | - | Positive | Positive | - | - | - | Trustworthy |
| | 124 | - | Negative | - | Negative | - | - | Negative | Negative | Negative | Negative | Untrustworthy |
| | 125 | - | Neutral | - | Neutral | - | - | Neutral | Neutral | Neutral | Neutral | Neutral |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 126 | - | Positive | - | Positive | - | - | Positive | Positive | Positive | Positive | Trustworthy |
| 127 | - | Negative | - | Negative | - | Negative | - | Negative | Negative | Negative | Untrustworthy |
| 128 | - | Neutral | - | Neutral | - | Neutral | - | Neutral | Neutral | Neutral | Neutral |
| 129 | - | Positive | - | Positive | - | Positive | - | Positive | Positive | Positive | Trustworthy |
| 130 | - | Negative | - | Negative | Negative | - | - | Negative | Negative | Negative | Untrustworthy |
| 131 | - | Neutral | - | Neutral | Neutral | - | - | Neutral | Neutral | Neutral | Neutral |
| 132 | - | Positive | - | Positive | Positive | - | - | Positive | Positive | Positive | Trustworthy |
| 133 | - | Negative | - | Negative | Negative | Negative | - | - | - | - | Untrustworthy |
| 134 | - | Neutral | - | Neutral | Neutral | Neutral | - | - | - | - | Neutral |
| 135 | - | Positive | - | Positive | Positive | Positive | - | - | - | - | Trustworthy |
| 136 | - | Negative | - | Negative | Negative | - | Negative | - | - | - | Untrustworthy |
| 137 | - | Neutral | - | Neutral | Neutral | - | Neutral | - | - | - | Neutral |
| 138 | - | Positive | - | Positive | Positive | - | Positive | - | - | - | Trustworthy |
| 139 | - | Negative | - | Negative | - | Negative | Negative | - | - | - | Untrustworthy |
| 140 | - | Neutral | - | Neutral | - | Neutral | Neutral | - | - | - | Neutral |
| 141 | - | Positive | - | Positive | - | Positive | Positive | - | - | - | Trustworthy |
| 142 | - | - | Negative | Negative | - | - | Negative | Negative | Negative | Negative | Untrustworthy |
| 143 | - | - | Neutral | Neutral | - | - | Neutral | Neutral | Neutral | Neutral | Neutral |
| 144 | - | - | Positive | Positive | - | - | Positive | Positive | Positive | Positive | Trustworthy |
| 145 | - | - | Negative | Negative | - | Negative | - | Negative | Negative | Negative | Untrustworthy |
| 146 | - | - | Neutral | Neutral | - | Neutral | - | Neutral | Neutral | Neutral | Neutral |
| 147 | - | - | Positive | Positive | - | Positive | - | Positive | Positive | Positive | Trustworthy |
| 148 | - | - | Negative | Negative | Negative | - | - | Negative | Negative | Negative | Untrustworthy |
| 149 | - | - | Neutral | Neutral | Neutral | - | - | Neutral | Neutral | Neutral | Neutral |
| 150 | - | - | Positive | Positive | Positive | - | - | Positive | Positive | Positive | Trustworthy |
| 151 | - | - | Negative | Negative | Negative | Negative | - | - | - | - | Untrustworthy |
| 152 | - | - | Neutral | Neutral | Neutral | Neutral | - | - | - | - | Neutral |
| 153 | - | - | Positive | Positive | Positive | Positive | - | - | - | - | Trustworthy |
| 154 | - | - | Negative | Negative | Negative | - | Negative | - | - | - | Untrustworthy |
| 155 | - | - | Neutral | Neutral | Neutral | - | Neutral | - | - | - | Neutral |
| 156 | - | - | Positive | Positive | Positive | - | Positive | - | - | - | Trustworthy |
| 157 | - | - | Negative | Negative | - | Negative | Negative | - | - | - | Untrustworthy |
| 158 | - | - | Neutral | Neutral | - | Neutral | Neutral | - | - | - | Neutral |
| 159 | - | - | Positive | Positive | - | Positive | Positive | - | - | - | Trustworthy |

## Appendix C

### Fuzzy rules evaluation

The detailed calculation of the rule strengths for all the fuzzy rules that result in a non-zero membership in the output fuzzy sets demonstrated in Section 8.5. The rule strengths are calculated using equation 8.18.

**Rule 3:** If **N** is *Positive* and **M** is *Positive* and **P** is *Positive* and **R** is *Positive* then **Trust Level** is *Very Trustworthy*

$$l_{R3} = \min(0.54, 0.59, 0.64, 0.61) = 0.54$$

**Rule 6:** If **N** is *Positive* and **M** is *Positive* and **P** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R6} = \min(0.54, 0.59, 0.64) = 0.54$$

**Rule 9:** If **N** is *Positive* and **M** is *Positive* and **R** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R9} = \min(0.54, 0.59, 0.61) = 0.54$$

**Rule 12:** If **N** is *Positive* and **P** is *Positive* and **R** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R12} = \min(0.54, 0.64, 0.61) = 0.54$$

**Rule 15:** If **M** is *Positive* and **P** is *Positive* and **R** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R15} = \min(0.59, 0.64, 0.61) = 0.59$$

**Rule 54:** If **N** is *Positive* and **M** is *Positive* and **A** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R54} = \min(0.54, 0.59, 0.48, 0.24, 0.47, 0.36) = 0.24$$

**Rule 57:** If **N** is *Positive* and **M** is *Positive* and **O** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R57} = \min(0.54, 0.59, 0.78, 0.24, 0.47, 0.36) = 0.24$$

**Rule 60:** If **N** is *Positive* and **M** is *Positive* and **C** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R60} = \min(0.54, 0.59, 0.32, 0.24, 0.47, 0.36) = 0.24$$

**Rule 63:** If **N** is *Positive* and **M** is *Positive* and **C** is *Positive* and **O** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R63} = \min(0.54, 0.59, 0.32, 0.38) = 0.32$$

**Rule 66:** If **N** is *Positive* and **M** is *Positive* and **C** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R66} = \min(0.54, 0.59, 0.32, 0.48) = 0.32$$

**Rule 69:** If **N** is *Positive* and **M** is *Positive* and **O** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R69} = \min(0.54, 0.59, 0.78, 0.48) = 0.48$$

**Rule 72:** If **N** is *Positive* and **P** is *Positive* and **A** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R72} = \min(0.54, 0.64, 0.48, 0.24, 0.47, 0.36) = 0.24$$

**Rule 75:** If **N** is *Positive* and **P** is *Positive* and **O** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R75} = \min(0.54, 0.64, 0.78, 0.24, 0.47, 0.36) = 0.24$$

**Rule 78:** If **N** is *Positive* and **P** is *Positive* and **C** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R78} = \min(0.54, 0.64, 0.32, 0.24, 0.47, 0.36) = 0.24$$

**Rule 81:** If **N** is *Positive* and **P** is *Positive* and **C** is *Positive* and **O** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R81} = \min(0.54, 0.64, 0.32, 0.78) = 0.32$$

**Rule 84:** If **N** is *Positive* and **P** is *Positive* and **C** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R84} = \min(0.54, 0.64, 0.32, 0.48) = 0.32$$

**Rule 87:** If **N** is *Positive* and **P** is *Positive* and **O** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R87} = \min(0.54, 0.64, 0.78, 0.48) = 0.48$$

**Rule 90:** If **N** is *Positive* and **R** is *Positive* and **A** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R90} = \min(0.54, 0.61, 0.48, 0.24, 0.47, 0.36) = 0.24$$

**Rule 93:** If **N** is *Positive* and **R** is *Positive* and **O** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R93} = \min(0.54, 0.61, 0.78, 0.24, 0.47, 0.36) = 0.24$$

**Rule 96:** If **N** is *Positive* and **R** is *Positive* and **C** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R96} = \min(0.54, 0.61, 0.32, 0.24, 0.47, 0.36) = 0.24$$

**Rule 99:** If **N** is *Positive* and **R** is *Positive* and **C** is *Positive* and **O** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R99} = \min(0.54, 0.61, 0.32, 0.78) = 0.32$$

**Rule 102:** If **N** is *Positive* and **R** is *Positive* and **C** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R102} = \min(0.54, 0.61, 0.32, 0.48) = 0.32$$

**Rule 105:** If **N** is *Positive* and **R** is *Positive* and **O** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R105} = \min(0.54, 0.61, 0.78, 0.48) = 0.48$$

**Rule 108:** If **M** is *Positive* and **P** is *Positive* and **A** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R108} = \min(0.59, 0.64, 0.48, 0.24, 0.47, 0.36) = 0.24$$

**Rule 111:** If **M** is *Positive* and **P** is *Positive* and **O** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R111} = \min(0.59, 0.64, 0.78, 0.24, 0.47, 0.36) = 0.24$$

**Rule 114:** If **M** is *Positive* and **P** is *Positive* and **C** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R114} = \min(0.59, 0.64, 0.32, 0.24, 0.47, 0.36) = 0.24$$

**Rule 117:** If **M** is *Positive* and **P** is *Positive* and **C** is *Positive* and **O** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R117} = \min(0.59, 0.64, 0.32, 0.78) = 0.32$$

**Rule 120:** If **M** is *Positive* and **P** is *Positive* and **C** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R120} = \min(0.59, 0.64, 0.32, 0.48) = 0.32$$

**Rule 123:** If **M** is *Positive* and **P** is *Positive* and **O** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R23} = \min(0.59, 0.64, 0.78, 0.48) = 0.48$$

**Rule 126:** If **M** is *Positive* and **R** is *Positive* and **A** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R126} = \min(0.59, 0.61, 0.48, 0.24, 0.47, 0.36) = 0.24$$

**Rule 129:** If **M** is *Positive* and **R** is *Positive* and **O** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R129} = \min(0.59, 0.61, 0.78, 0.24, 0.47, 0.36) = 0.24$$

**Rule 132:** If **M** is *Positive* and **R** is *Positive* and **C** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R32} = \min(0.59, 0.61, 0.32, 0.24, 0.47, 0.36) = 0.24$$

**Rule 135:** If **M** is *Positive* and **R** is *Positive* and **C** is *Positive* and **O** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R35} = \min(0.59, 0.61, 0.32, 0.78) = 0.32$$

**Rule 138:** If **M** is *Positive* and **R** is *Positive* and **C** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R38} = \min(0.59, 0.61, 0.32, 0.48) = 0.32$$

**Rule 141:** If **M** is *Positive* and **R** is *Positive* and **O** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R141} = \min(0.59, 0.61, 0.78, 0.48) = 0.48$$

**Rule 144:** If **P** is *Positive* and **R** is *Positive* and **A** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R144} = \min(0.64, 0.61, 0.48, 0.24, 0.47, 0.36) = 0.24$$

**Rule 147:** If **P** is *Positive* and **R** is *Positive* and **O** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R147} = \min(0.64, 0.61, 0.78, 0.24, 0.47, 0.36) = 0.24$$

**Rule 150:** If **P** is *Positive* and **R** is *Positive* and **C** is *Positive* and **L** is *Positive* and **S** is *Positive* and **E** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R150} = \min(0.64, 0.61, 0.32, 0.24, 0.47, 0.36) = 0.24$$

**Rule 153:** If **P** is *Positive* and **R** is *Positive* and **C** is *Positive* and **O** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R153} = \min(0.64, 0.61, 0.32, 0.78) = 0.32$$

**Rule 156:** If **P** is *Positive* and **R** is *Positive* and **C** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R156} = \min(0.64, 0.61, 0.32, 0.48) = 0.32$$

**Rule 159:** If **P** is *Positive* and **R** is *Positive* and **O** is *Positive* and **A** is *Positive* then **Trust Level** is *Trustworthy*

$$l_{R159} = \min(0.64, 0.61, 0.78, 0.48) = 0.48$$