VRIJE UNIVERSITEIT AMSTERDAM

MASTER THESIS
MSC BUSINESS ANALYTICS

Validation of Call Center Assumptions and Prediction of a Single-Skill Call Center Performance Measures via Machine Learning Approach

---

Ezgi Izel Yuce

Graduation Supervisor:
Eliseo Ferrante

Second Reader:
Dennis Dobler

External Supervisor:
Siqiao Li

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1081a
1081 HV Amsterdam

August,2020

Preface

This thesis is written to obtain the degree of MSc.Business Analytics from Vrije Universiteit Amsterdam. Master Business Analytics is a two-year program consisting 6 months internship in a company. This research is conducted during a six-months internship in CCMath. During my internship in CCMath, I had the opportunity to work on the challenging problems of call center modelings and workforce management.

The first part of the research focuses on validation of the most common assumptions in multi-skill call center modeling. The second part focuses on prediction of performance measures of single-skill call center by machine learning models. More specifically, since multi-skill models do not have analytical solutions, simulation method is preferred in modeling. To depict the reality in simulation models, some assumptions have to be made. This research focuses first on how these assumptions effect the accuracy of performance measures' predictions. Subsequently, the second part of the research focuses on predicting performance with a promising alternative, machine learning.

I would like to thank Eliso Ferrante for his guidance. Also for supporting me in my decision about my new internship. Furthermore, I would like to thank Dennis Dobler for taking the time to correcting my thesis.

Furthermore, I would like to thank Siqiao Li for all the guidance she has given me. I have learnt a lot from her during my internship. I count myself lucky to have such a supervisor like Siqiao. Besides, I would like to thank Ger Koole for his help and feedback.

In addition, I would like to thank Hans Frenk, who was my professor during my bachelor's, for all his guidance and help in my decision to pursue a master's degree in the Netherlands. Besides, I would like to thank my parents for their loving support even though we were far from each other during my internship period due to corona outbreak. And lastly, Alexander, many thanks for your endless support.

Executive Summary

Call centers all over the world have the same goal of meeting the service level targets. Every day, call center managers calculate the staffing levels in order to meet these targets. They can use different OR methods to model their call centers. Multi-skill call centers do not have a closed-form of analytical calculations, unlike single-skill models such as Erlang-C and Erlang-X. Therefore, simulation methods are preferred in modeling of multi-skill call centers. At the end, some simplifications and assumptions have to be made to depict the reality in the modeling phase. However, these assumptions can lead to inaccuracy in the estimations of performance. Therefore, this research answers the following question:

How do the assumptions of multi-skill call center affect the accuracy of performance measures?

Although simulation gives reliable results, it can be time-consuming. With developing technology, traditional methods have been replaced by machine learning methods for various types of businesses. In the future, workforce management can be one of these businesses. Instead of simulation methods, machine learning methods can be used to predict the performance measures. Workforce management of call centers is a broad research area, and multi-skill call centers have a very complex structure to model; therefore the problem should be narrowed down to single-skill call centers. The possible solution of single-skill may be a pioneer for the future work to predict multi-skill call center performance measures. Therefore, this research also answers the following question:

Can a machine learning algorithm provide a better fit than a queueing model?

While the solution related to the first question shows the possible inaccuracies of predicting performance measures via modeling with certain assumptions, the second question's solution shows a promising alternative way to predict performance measures without making assumptions such as handling time being exponentially distributed or arrival process being in-homogeneous Poisson process.

Firstly, data is analysed for the validation part of the project. To validate the assumptions of features such as arrival process, handling time, break time and wrap up time, several different models are used. The base model is an empirical model of the reality. The other models are representatives of each assumption. For instance, the exponential assumption of handling time is modeled as the following: handling time is exponentially distributed, and other features are the same as the empirical model. Finally, a comparison is made between the performance measures from the real data and those from the simulation results.

After this research on the validation of call center assumptions, the conclusion is made that the in-homogeneous Poisson arrival process and the handling time being exponentially distributed do not significantly affect the accuracy of performance measure predictions. On the other hand, the exponential assumption of patience, ignoring break time during staffing level calculations, and not including wrap-up time to handling time cause a significant inaccuracy in predictions. These results hold for a specific data set. Another data set is analysed however, the call center was very unique in terms of features such as routing policy, handling time, break time, and wrap-up time. Therefore,

it is decided not to simulate this data set. Future work is recommended to validate a similar size call center to have more generalised results.

Random data is generated for the feature sets of Erlang models to predict the service level of queueing models. Subsequently, the service level of the randomly generated features are calculated with Erlang Calculators of CCMath. Linear Regression, Beta Regression, SVR, Random Forest and Gradient Boosting with XGBoost framework are applied to predict service level of Erlang C and X models.

The results show that, even though the performance metrics indicate a good fit, the range of the predictions is not realistic for some of the machine learning algorithms since the service level predictions must be between 0 and 1. It is shown that Random Forest is the most suitable algorithm for the service level predictions of Erlang C and X models. It gives the predictions in a range [0,1] and it provides low error and high R squared.

Further work is recommended to focus on the predictions of service level for more complex call center models via Random Forest and Gradient Boosting with XGBoost framework. The complexity of future models should be increased gradually.

Contents

List of Tables

List of Abbreviations

OR Operation Research

SL Service Level

ASA Average Speed of Answer

Ab Abandonment Rate

HT Handling Time

AHT Average Handling Time

FC Forecast(Arrivals)

AWT Acceptable Waiting Time

WAE Weighted Average Error

Chapter 1

Introduction

A call center is a centralized office where large numbers of telephone calls are handled to provide customer services by agents. A general definition of a call center agent can be that an individual who handles the inbound or outbound calls to provide service for the business. Nowadays, the job description of agents is broader than merely handling calls. Emails and webchats are also included in the job definition of agents. This reveals that modeling of call centers is becoming harder for researchers while the job definition of agents is becoming broader.

Depending on the call type, call centers can be categorized into two types such as inbound call centers and outbound call centers [1]. In an inbound call center, calls are initiated by the customers and agents provide customers services. In an outbound call center, calls are initiated by an agent on behalf of a business or a client. Telemarketing, fundraising and sales are some examples of outbound calls.

Another categorization of call centers is based on the agent skills, divided as single-skill or multi-skill call centers [1]. A single-skill call center has one type of service; in other words, agents are homogeneous since they can handle every call. On the other hand, a multi-skill call center has different types of services. The agents can have different skills from one another; thus they can handle different types of calls.

Nowadays, most of the call centers are using workforce management to optimize the efficiency of the business. Workforce management determines the required staffing level of agents to reach a certain service level target [16]. Service level is one of the most important performance measures in call centers. Staffing levels are turned into agent schedules, which is a crucial step of call center optimization and an enormous part of the cost of a call center. Staffing, by itself, is another problem to be solved, and it is out of the scope of this research.

Call center managers make schedules and calculate the staffing level to meet the service level target. Simulation is one of the OR methods to model multi-skill call centers [3]. In order to model the call centers via simulation, some simplifications and assumptions of features are performed. For example, the most common assumption of arrival process is in-homogeneous Poisson arrivals [3]. In simulation, the arrivals are modeled as in-homogeneous Poisson by piecewise constant rate. However, these assumptions

can effect the accuracy of the predictions of performance measures. Therefore, based on the objective of the research, the first research question is defined as follows:

How do the assumptions of multi-skill call center effect the accuracy of performance measures?

In this research, the most common assumptions of call centers are verified by modeling each assumption as a different model and assessing the accuracy of prediction of performance measures for each model. The performance measures from simulation results of the models and the performance measures from real data are compared.

Simulation is one of the most common methods to model multi-skill call centers. However, the assumptions and simplifications can effect the accuracy of the performance measures. Machine learning can be a very efficient alternative modeling. Unlike simulation, machine learning does not require assumptions of features. In the future, instead of simulation, machine learning methods can be used to predict multi-skill call centers' performance measures. Multi-skill call centers are relatively complex to model; therefore, the problem should be narrowed down to a prediction of single-skill call centers. Solutions to single-skill queueing models will be helpful for future researches to model more complex models. Therefore, based on the objective of the research, the second research question is defined as follows:

Can a machine learning algorithm provide a better fit than a queueing model?

In this research, the service level of a single-skill call center is predicted. Understanding of single-skill results can be a pioneer for future work of multi-skill modeling via a machine learning approach.

1.1 Related Work

In this section, a variety of studies that have been performed in the call center modeling will be introduced. In order to model call centers, some assumptions of features should be made. In this research, the most common assumptions about the arrival process, handling time, customer patience, break time and wrap time are validated. Studies about the validation of multi-skill call centers are introduced. Subsequently, related work about machine learning will be discussed.

[3] discusses call center modeling from a broad perspective. They discuss common solutions such as simulation and queueing models to have performance analysis of a call center. Since multi-skill call centers do not have analytical solutions, the simulation method is preferred in modeling multi-skill call centers. Single-skill call centers can be modeled with queueing models [3]. The required inputs of simulation methods to model a call center are discussed by [4].

The first research question focuses on the validation of common assumptions in modeling multi-skill call centers. These assumptions from literature are introduced as follows. The most common approach for the arrival process in the call center literature is modeling as an in-homogeneous Poisson process with a piecewise constant rate [3].

Handling time or service time is assumed exponentially distributed with a constant mean [6]. In addition to the distribution of handling time, the assumption of average handling time varies per day is validated. Patience is the willingness of a customer to wait for service. Most of the papers in the call center literature have modeled patience as exponentially distributed with a constant mean [7]. Moreover, it will be validated that ignoring agent breaks in staffing level calculations cause inaccuracy in the prediction of performance measures.

Lastly, it is shown that ignoring wrap-up time in handling time calculations leads to errors in performance measure predictions. Wrap-up time is administrative time after a call ends. Agents make documentation about the call during wrap-up time and cannot handle any other call during this time. Therefore, it should be included in handling time [8].

The second research question focuses on predicting performance measures of a single-skill call center with a machine learning approach. [2] and [5] provide an overview of queueing models, which will be building blocks for the machine learning approach in this research. Moreover, [18] and [19] provide the analytic calculation of queueing models Erlang-C and Erlang-A, respectively. Knowledge of the queueing model is crucial to understand the performance of machine learning algorithms and possible solutions to improve performance.

Modeling a multi-skill call center via a machine learning approach is a complex problem. It is an ongoing search in literature. Therefore, the problem should be divided into sub-problems. A machine learning approach is applied by [32]. They answer a sub-problem of multi-skill call center modeling. For a given staffing-level, performance measures are predicted. They only use staffing-level data and they generate the data required via simulation. In other words, staffing levels are the features and the performance measures are target variables. In this research, a similar approach is applied. The data is generated as in the following. The input data of queueing models are generated randomly and the performance measures are calculated according to the analytic solution of queueing models. The inputs of a single-skill call center are features and performance measures are target variables. Via the machine learning approach, the performance measures of a single-skill call center are predicted.

1.2 Outline

This paper is organised as follows. Due to having two different research questions, the thesis divided into two parts. The first part introduces the validation related data analysis, methodology, experiment setup and model evaluation. The second part introduces the machine learning approach. Data analysis, methodology, experiment setup and model evaluation of the machine learning approach are introduced in part two with the following sections, consequently, 6,7,8 and 9. Conclusion is performed in chapter 10 and discussion is introduced in chapter 11 for both parts.

Part I

Validation

Chapter 2

Data Analysis and Data Preparation

In this section, data analyses of the data sets available for validation are introduced. In this section, data description and analysis are introduced. There were two data set available in CCMath for the validation. The first one is Vanad data set and the second one is Swiss data set. Both data sets' description and analysis are introduced. Furthermore, the discussion concerning the data set selection for validation will be made.

2.1 Vanad Data Set

This data set is from Vanad Laboratories which is a multi-skill call center. The data contains the records of the year 2014. The data consists of two different data sets such as the call logs and the agent activity logs.

Call logs data set includes the information of arrival time of the call, the time that the agent starts handling the call,departure time, the service type that a caller request, the identity of the agent who handles the call, etc.

Call Arrival Time	Service ID	Agent ID	Answered time	Call Departure Time
1/2/2014 8:03:21	30175	6935	1/2/2014 08:33:22	1/2/2014 08:05:54
1/2/2014 8:04:37	30560	6931	1/2/2014 08:04:38	1/2/2014 08:09:49
1/2/2014 8:06:36	30172	8058	1/2/2014 08:06:27	1/2/2014 08:15:17
1/2/2014 8:08:07	30172	9182	1/2/2014 08:08:08	1/2/2014 08:10:37
1/2/2014 8:08:26	30172	6048	1/2/2014 08:08:27	1/2/2014 08:14:59

Table 2.1: An example of the call log

The explanation of the columns are as the following:

- **Call Arrival Time:** The time that a caller starts waiting for the service
- **Service ID:** The identification of the service that a caller request
- **Agent ID:** The identification of the agent that handles the call
- **Answered Time:** The exact time that the agent picks up the coming call
- **Call Departure Time:** The time that the call finishes

Activity log data comprises of the activity log of agents with the following information: the index of the activity, the start time of the activity, the end time of the activity and the identification of the agent. The duration of the activities are calculated by subtracting the start time of the activity from the end time of the activity.

Activity ID	Start time	End time	Agent ID
3	1/2/2014 8:44:29	1/2/2014 9:00:02	6934
6	1/2/2014 9:00:31	1/2/2014 9:06:30	6883
9	1/2/2014 9:09:43	1/2/2014 9:10:47	9045
7	1/2/2014 9:12:47	1/2/2014 9:14:54	8495
7	1/2/2014 9:13:28	1/2/2014 9:19:23	8493

Table 2.2: An example of the agent activity log

- **Activity ID:** The identification of the activity
- **Start time:** The start time of the activity
- **End time:** The end time of the activity
- **Agent ID:** The identification of the agent that has the activity record

There are 1543164 calls from 27 different service IDs in the data set. 312 agents are working during the year 2014. The number of calls for each service ID differs. Not all the service IDs have enough calls during the year. The aim is to choose the service IDs that in total they represent 99% of the call logs. 8 service IDs are chosen out of 27 different service IDs. The rest of the services does not have more than 200 calls in a year. Therefore, the data is performed for validation has 1524731 number of calls from 8 different service IDs.

As a routing policy, first come fist served(FCFS) is used. The first arrived call is assigned to the longest idle agent who can handle the service type of the arriving call.

2.1.1 Arrivals

It is common to assume in call center modelings that arrivals follow a Poisson distribution. During the day, the volume of arrival calls may be different. It is usual to

see the volume decreases at the end working hours. In this case, we cannot expect to have one single rate of arrivals for a day when modeling Poisson arrivals. Therefore, an in-homogenous Poisson process with a piecewise constant rate could be a better option in changing the volume of arrivals during the day.

In order to see the Poisson arrivals intra-year, intra-week and intra-day(hour base and interval base) are analyzed.

As one can see from the Figure 1.a., the volume of the calls increased from week 1 to week 27. Then it starts to decrease gradually until week 32. On the other hand, there is a drastic decline in December. Figure 1.b. shows the intra-week pattern, which means the days of the week during a year. Each point represents a day of a week, such as Monday,Tuesday, etc. The call center is closed on Sundays; thus Sunday is ignored in advance. Each day of the week has a different pattern but overall every week has a similar pattern in terms of days of week. For example, each week, Monday has the highest volume of calls and Saturday has the lowest volume of calls. From figure 1.c. one can see the intra-day plot. The volume of calls has an increasing trend during the morning and a decreasing trend in the afternoon. This volume changes during a day can be clearly seen in Figure 1.d. as well. 15 minutes interval graph shows that each interval has a different volume of arrivals.

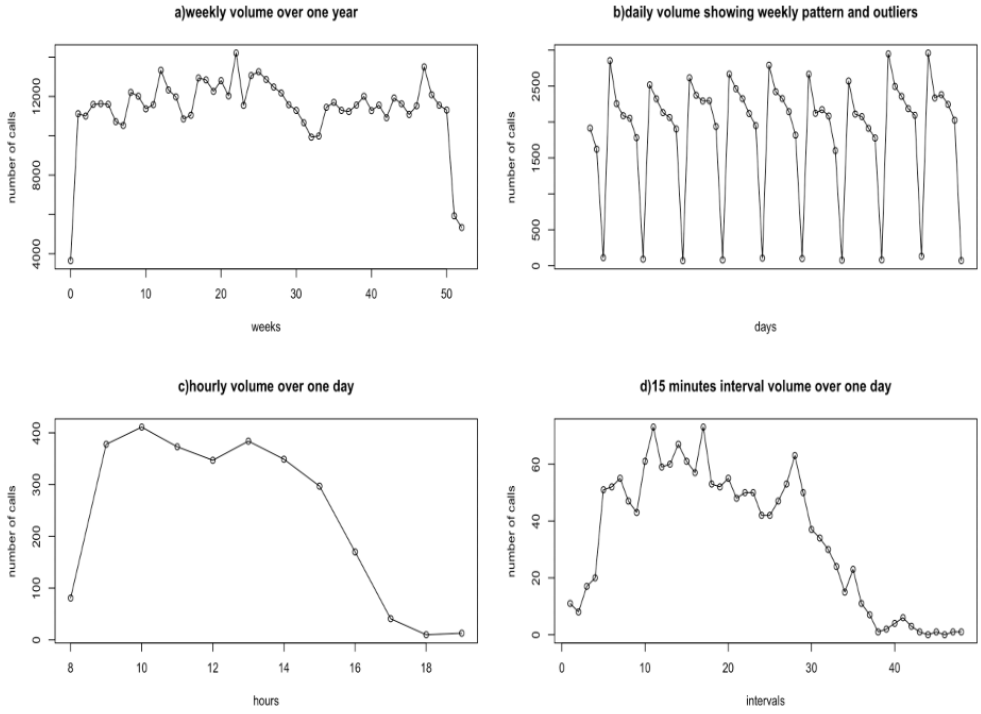


Figure 2.1: The arrival processes

2.1.2 Handling Time

Handling time is the duration of service of an agent. Handling time starts when a call is assigned to an agent and ends when the call terminates. Moreover, the

documentation about a call, in other words wrap-up time, should included in handling time. During a call, an agent is labelled as 'occupied' in the system, thus, that agent cannot handle any other call. Also during wrap-up time the agent is still 'occupied' with the same call. The details of wrap-up time will be explained in the section 2.3.

As one can see from Table 2.1, a call has the following information: arrival time, answered time and departure time. Unfortunately, the ring time which is the time period until the moment that a agent answers the call after he is assigned to a call, is not recorded. It is the same logic as wrap-up time. During the ring time an agent is 'occupied'. Generally the time between a call is enqueue and is answered is quite short. Nevertheless, it could be a better modeling if there was a ring time information.

Handling time of vanad data is calculated as subtracting the answered time from the departure time. Figure 2.7 shows the empirical histogram of the HT of one skill.

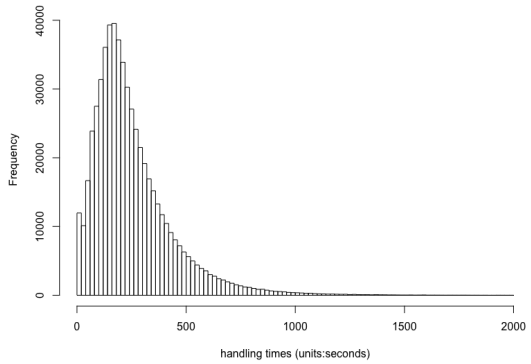


Figure 2.2: Handling times histogram, (units:seconds)

As one can see from the Figure 2.2, the data does not fit to exponential distribution. The calls with handling time being less than 15 seconds are ignored. Since these calls are too short to be able provide a service and they probably disconnected calls.

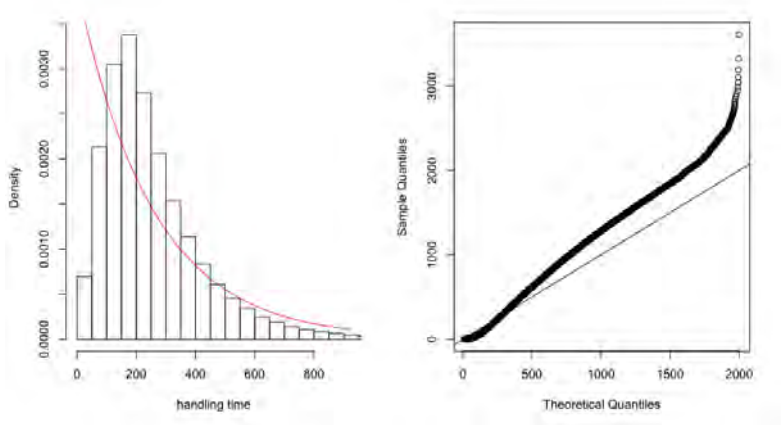


Figure 2.3: Handling times histogram, (units:seconds)

As one can see from Figure 2.3, exponential distribution does not fit to the data. From the graph it can be guessed that the data has log-normal distribution. The aim of this research is not proving a certain distribution for features such as arrivals, handling time. It is aimed to assess the effect of the general assumptions of call centers on predictions of performance measures.

The AHT per day is shown in Figure 2.4. As one can see from the graph, AHT fluctuates per day. These changes in AHT per day gives insight of the assumption of taking AHT as a whole year average. Since AHT varies day to day, making the assumption of each day has whole year average may lead inaccuracy in the predictions.

Moreover, checking AHT per agent during a year may provide us more insights about the variation of AHT per day. Since experienced agents are expected to have smaller AHT than the new agents, this heterogeneity in terms of agents can be the reason of variation of AHT per day. The graph on the left side in Figure 2.5 shows the experienced agents' AHT during a year. Each color represents an agent and each point on a line represents the monthly AHT of that agent. As expected, the experienced agents have more constant AHT than the new agents. When a new agent starts working, it is expected that the agent's AHT would decrease in time which can be confirmed by the Figure 2.5 . [10] observes the agent heterogeneity and proposes a method to assign learning rate for the agents. In this way they propose four different models to predict of AHT of each agent. Learning rate and predicting AHT of agents is an extensive problem and can be formulated as a research question. Although this research does not cover the learning rate of agents, it is important to show a possible reason of AHT fluctuations per day.

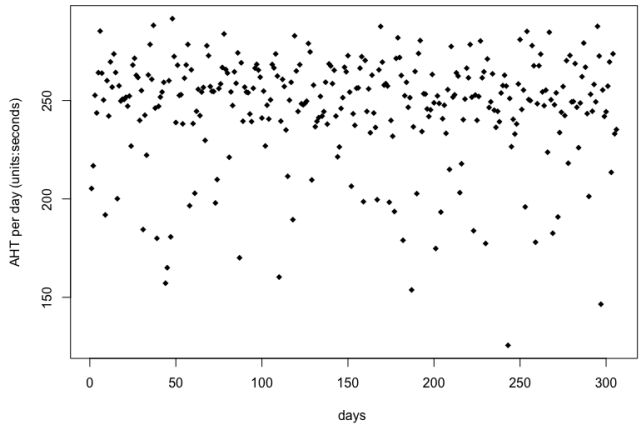


Figure 2.4: AHT per day, (units:seconds)

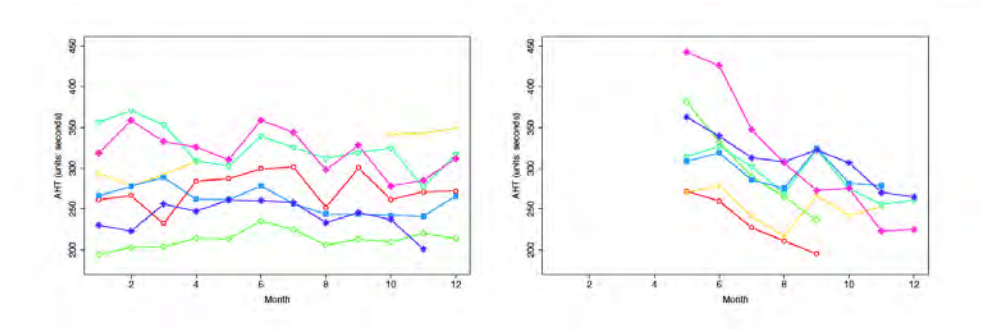


Figure 2.5: AHT per month of experienced agents and new agents (units:seconds)

2.1.3 Patience

Patience can be defined as the the time that a caller is willing to wait in queue to have service. At the end, callers would leave the queue or abandon if they are not served for some time. This abandonment behavior changes customer to customer. But we can estimate it for each service type by Kaplan-Meier estimator [12]. One service type is selected and patience of this service is estimated. Figure 2.6 shows the CDF and hazard rate of patience for the selected service type. Hazard rate can be defined as the following: the conditional probability of a caller to abandon depending on the waiting time he reaches. The hazard rate calculation can be found in section 3.2. As one can see from the hazard rate graph, there is an increasing trend of hazard rate.

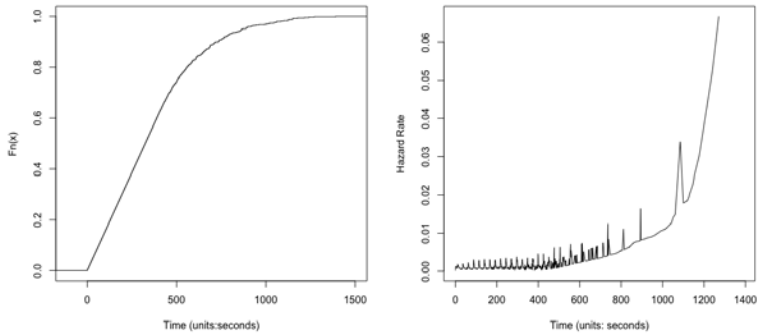


Figure 2.6: CDF and hazard rate for patience, (units:seconds)

2.1.4 Wrap-up Time

The agents are required to make documentation about the calls. They can do it during the call or after the call. Call center can have strict rules about wrap-up times such as wrap-up time must not exceed 60 seconds after the call finished.

The time required for the documentation is called as wrap-up time. Even though it is small amount of time, it can lead to inaccuracy if it is ignored. Because if an agent is

doing wrap-up after a call, it means that agent is occupied and cannot handle an arriving call.

The wrap-up times are recorded in agent activity logs. This means we do not have the information of wrap-up time call by call but we can estimate it. In agent activity log data, wrap-up times are recorded by an activity index 16. An example of an activity record can be seen from Table 2.2. The logs with activity index 16 are selected. The average of these logs are taken as an average wrap-up time for all types of calls since we do not have service type information of wrap-up time. The average wrap-up time is approximately 3.2 seconds. As one can see from the Figure 2.7 wrap-up times distribute between 0 and 40 seconds. The most of the wrap-up times have duration of 0 seconds. It means agents handle the wrap-up during the call. This is a common situation for call centers which has intense traffic. Some call centers may have strict threshold for wrap-up time for the agents.

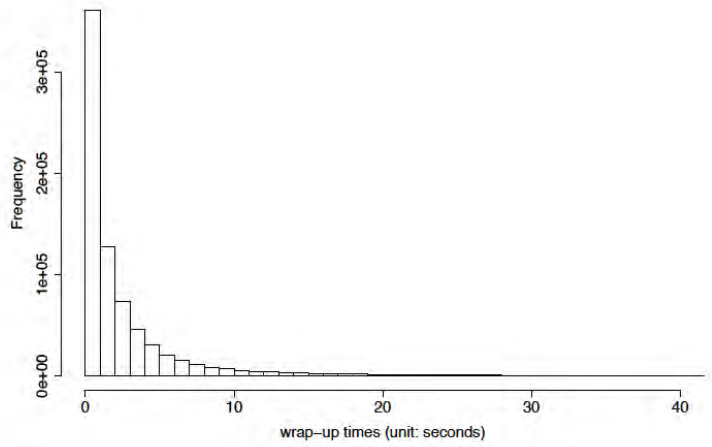


Figure 2.7: Wrap-up times histogram

2.1.5 Breaks

Shrinkage is an important concept for staffing and workforce management. Shrinkage can be defined as the following:

$$\text{shrinkage} = \frac{\text{time taking breaks}}{\text{shift duration(time handling call,waiting for the calls and taking breaks)}}$$

Shrinkage is the percentage of time that an agent does not handle calls or wait for the calls during his shift. The shift information is not provided by Vanad, but shift duration can be calculated approximately. From the activity log data, the difference between the first log and the last log of an agent, the duration can be calculated.

Break durations of agents are shown in the Figure 2.8. We can see that there are three obvious peaks approximately at 300, 600 and 1000th seconds which correspond 5, 10

and 15 minutes respectively. Since we do not have the shift information of agents, we do not know the planned breaks during a day. [13] explains the distinction between plannable and unplannable lost agent time, breaks. In vanad data set, the distinction between breaks does not exist and it cannot be easily estimated from the data. Therefore, the activity logs which are recorded as 'breaks' are used as break time without any distinction.

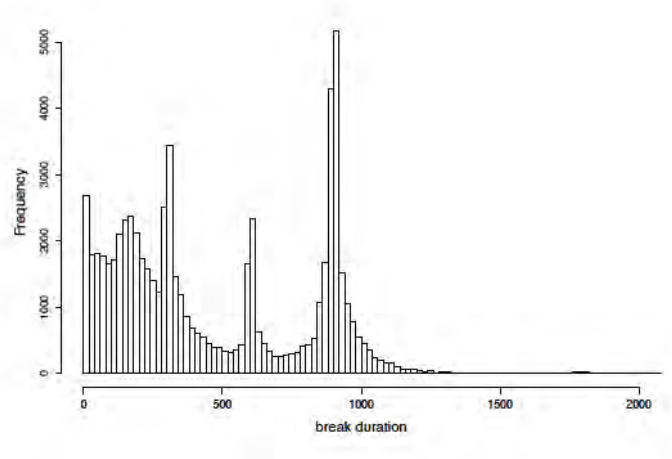


Figure 2.8: Wrap-up times histogram, (units:seconds)

2.2 Swiss Data Set

The data is from a multi-skill call center which is a travel agency from Switzerland. They have several services such as booking flight tickets and hotels. The data is provided in two different data sets. The first one is the call logs which has the records of each call with the following information: call arrival times, departure times, waiting time, service type of the call, the identity of the agent who handled the call, etc. Table 1 shows an instance from the call log data.

Call Arrival Time	LenQ0	LenRing	LenTalk	LenHold	LenWork	Wait Time	Service ID	Agent ID
2016-01-04 08:02:07	0	16	482	14	0	26	1008	1302
2016-01-04 08:02:58	28	10	438	0	60	0	1008	1300
2016-01-04 08:03:14	23	9	327	0	60	0	1008	1305
2016-01-04 08:04:32	10	15	258	0	15	0	1008	1287
2016-01-04 08:05:01	36	11	1048	0	25	0	1092	1249
2016-01-04 08:05:53	14	13	374	0	19	0	1092	1287

Table 2.3: Call Log

The definitions of column names

- **Call Arrival Time:** The date and time information of the moment the caller connects to the call center
- **LenQ0:** The time that a caller spends in IVR. In other words the time that a caller spends on the selection of the service

- **LenRing:** When a caller is the first to be handled and if there is an available agent, the phone is ringing for agent side. The time between the agent's phone starts ringing and the agent picks up the call is ringing time.
- **LenTalk:** The time that an agent handles a call
- **LenHold:** During the call, an agents may consult to another colleague. The during the consultation the call is in hold. This time represents the holding time during the call.
- **LenWork:** The wrap-up time. It is discussed in detail in section 2.1.5
- **Wait Time:** After the selection in IVR, the callers are enqueued. The time between an agent picks up a call and the call is enqueued is the wait time of the caller
- **Service ID:** The identity of a service that a caller chooses in IVR
- **Agent ID:** The identity of an agent who answers the call

The second data set is agent activity logs. Each row represent an activity which is recorded with the following information: Date Time of the activity, activity number, agent ID and shrinkage information.

Date Time	Activity Number	Agent ID	Shrinkage
2016-01-04 12:26:38	5	1020	1
2016-01-04 12:26:52	1	1020	0
2016-01-04 12:27:49	2	1017	1
2016-01-04 12:27:52	2	1016	1
2016-01-04 12:28:55	2	1053	1
2016-01-04 12:28:58	1	1053	0

Table 2.4: Agent Activity Log

- **Date Time:** The start time of an activity log
- **Activity Number:** The index of the activity that an agent is busy with
- **Agent ID:** The identification of an agent
- **Shrinkage:** If an agent wait for a call or handles a call, shrinkage is 0. Otherwise 1.

As one can see from the table 2, the activity logs only comprise of the start time of an activity. In order to extract length of the activity duration a new data set is generated. The difference between the second activity log and first activity log of an specific agent is the duration of an activity.

Start Date Time	End Date Time	Activity Number	Agent ID	Shrinkage
2016-01-04 12:26:38	2016-01-04 12:26:52	5	1020	1
2016-01-04 12:26:52	2016-01-04 12:27:49	1	1020	0
2016-01-04 12:27:49	2016-01-04 12:27:52	2	1017	1
2016-01-04 12:27:52	2016-01-04 12:28:55	2	1016	1
2016-01-04 12:28:55	2016-01-04 12:28:58	2	1053	1
2016-01-04 12:28:58	2016-01-04 12:29:07	1	1053	0

Table 2.5: Agent Activity Log

The data comprise of the calls from 5 years, starting from 2014 and ending at the first quarter of 2018. The number of calls for each year is shown in the Table 4. Our first choice was the year 2017 to validate. However, after data analysis, 2017 has a service ID 1095, 80% of these calls are without agent ID. We cannot ignore the calls with this service ID. The same problem exists for 2016, but the number of calls without agent ID consists of %0.5 of the calls from service ID 1095. Thus it is agreed to work on year 2016.

2014	2015	2016	2017	2018
84768	124481	137911	163598	5330

Table 2.6: Data length over the years

The routing policy of Swiss data is complex compared to Vanad data. The service types have the priority order. According to priority, different rules can apply. For example, priority 1 calls always assign first to the longest idle agents even though there is another waiting call which is not priority 1. Moreover, to handle priority 2 calls there is a waiting time threshold for agents. Let us assume there is an idle agent and priority 2 call is waiting to be served. The idle agent waits until the threshold to serve priority 2 because they want to be sure that until the threshold a call with priority 1 is not arriving. A call is assigned to the longest idle agent.

In the following subsections, the analyses of arrivals, handling time, patience, break times and wrap-up times are introduced.

2.2.1 Arrivals

In most of the call center modelings, it is assumed that arrival process is a NHPP with a piecewise-constant rate. The piecewise-constant rate is the number of arrivals per minute in a specified interval. Interval can be 15 minutes or 30 minutes. Many studies concerning the analysis of arrival processes have done. These studies comprise of several statistical methods. But it is out of scope of this research. The aim is to check whether the most common assumptions of call centers hold with the reality. Call centers evaluate the arrival processes in four different scales such as intra-year, intra-week, intra-day and intra-hour. In order to plot these scales of arrivals, a service ID is selected. Figure 2.14 shows the four scales of arrival processes of a given service

ID.

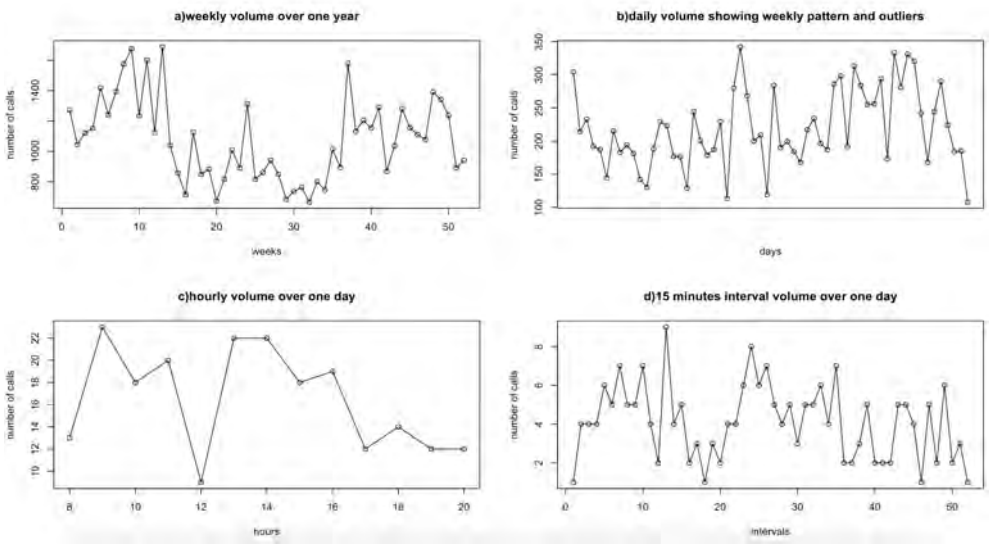


Figure 2.9: The arrival processes

As one can see from the Figure 2.9 a, intrayear shows that from week 1 to week 12, the number of calls have a increasing trend. At the end of August, the volume is very low. Then after august, the volume increases drastically. Figure 2.9b shows the intra-week volumes. Each point represents a day of week. The call center is closed on Sundays. Therefore, Sundays are ignored in advance. We can barely see the first couple of weeks, the trend is similar. Saturdays have almost the same trend during a year. Figure 2.9 c shows the intra-hour volumes. Similar to Vanad data, the first hours in the morning there is a increasing trend of number of arrival calls. In the afternoon, the volume of the arrival calls decreases. The same pattern can be seen from Figure 2.9d. Each point in the graph represents the volume of arrivals in 15 minutes long interval. We can say that the arrival rate changes during a day and 15 minutes intervals. We can conclude that there is a non-homogeneous process of arrivals.

2.2.2 Handling Time

Handling time is the service duration of an agent. Swiss call center data has higher handling times compared to Vanad data. There are also some extreme cases such as handling time being more than 1 hour. Furthermore, the wrap-up times also have some extreme cases and generally wrap-up time averages for each skill is at least 100 seconds which was 3 seconds for Vanad data. Therefore the following analysis of handling time considers the wrap-up times. Moreover, each step of a call is recorded and the ring time information is also present. Ring time should be considered as handling time as well since during ring time agent's status is changed as 'occupied' and cannot handle any other call.

In order to have more analytically tractable results, it is very common to have the assumption of HT being exponentially distributed in call center modelings. Before fitting exponential distribution to the handling time, it is checked that whether there is

abnormality in the data. Even though ring time and wrap up time are included in the HT, there are some calls having HT less than 15 seconds. These points are ignored.

One service ID is selected to analyze. Figure 2.10 shows the histogram of HT for the year 2016. The calls which take longer than 1 hour has 0.42% of the handling time of selected service. Since the percentage for these calls is quite low, the calls which take longer than 1 hour can be ignored. The shape of histogram suggests that HT is exponentially distributed. Figure 2.11 shows the fitted exponential density and its Q-Q plot of HT for the selected service.

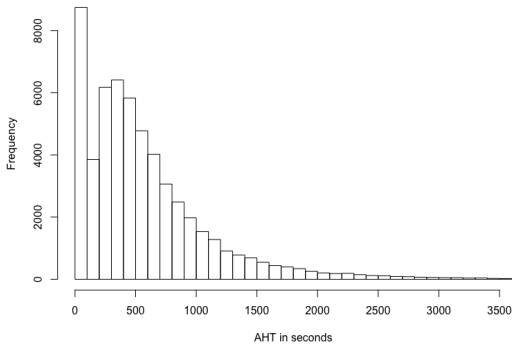


Figure 2.10: Histogram of the HT

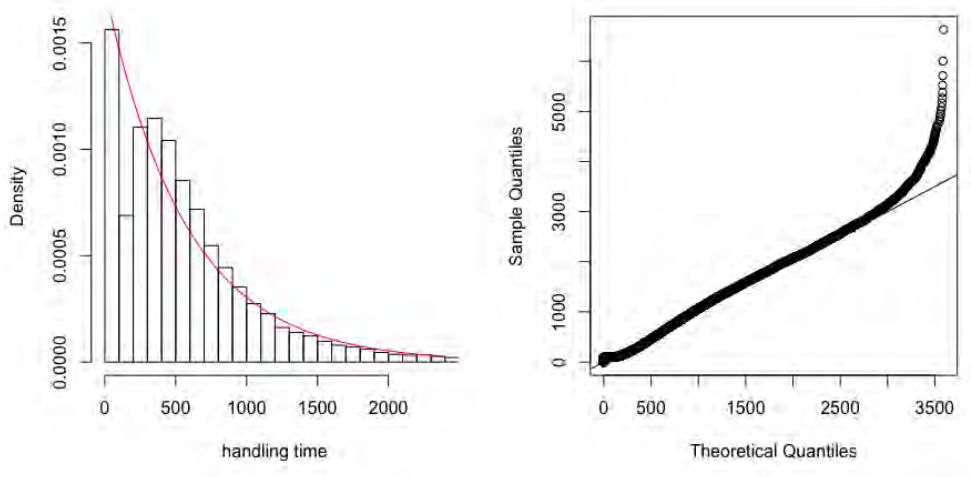


Figure 2.11: Histogram and QQ-Plot of HT

As one can see from the Figure 2.11 exponential distribution fits the data better than Vanad data. Nevertheless, it cannot be claim that exponential distribution fits swiss data well. As one can see from Q-Q Plot in the Figure 2.11, the right tail does not fit well.

One of the assumptions will be validated in this research is the effect of average

handling time. It is common in call centers to have AHT as per day instead of using whole year average. Figure 2.12 shows AHT of the selected service type per day for the year 2016. As one can see from the figure, AHT varied from day to day. One possible reason can be the agent heterogeneity since we can expect that experienced agents have lower AHT than the new agents. In order to see the agent heterogeneity effect on AHT, the monthly AHT of agents who have been working the whole year is plotted. The AHT of these agents per month is shown in Figure 2.13.

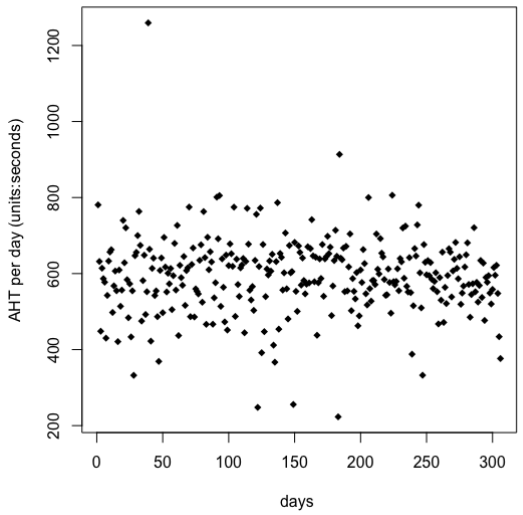


Figure 2.12: AHT per day

In Figure 2.13 each line represents a different agent's AHT. We can expect 1st, 5th and 6th agents to be experienced agents since their AHT more or less the same during a year. On the other hand, AHT of 3rd and 4th agents are fluctuating during the year. 4th agent has the peak in month 3, then there is another peak in month 8. These huge amount fluctuations are not expected for experienced agents. We contacted the company for further information about agents. We have been informed that the agents do not have unique IDs. In other words, if an agent leaves the job, his ID remains in the system and presumably is given to a new hired agent. Therefore, the company does not have unique IDs for each individual agent. For this reason, we could not work further on agent heterogeneity.

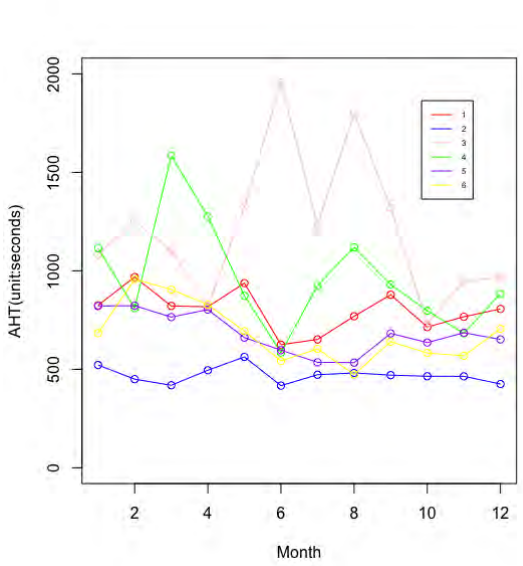


Figure 2.13: AHT per month

In order to search more the about handling time fluctuations during a day, intra-day patterns are plotted for the selected 6 different service IDs. Figure 7.1 shows intra-day AHT for the year 2015. Red lines show the volume of the calls and black lines show the AHT. Another possible reason effects handling time is behavior of agents during heavy load of calls. [11] reports that during periods of heavy load, agents initially work faster but afterwards agents may tire and work slower if the heavy load is sustained. We could not observe this pattern in intra-day plots in Figure 7.1. Moreover, it is expected to see that during a day AHT fluctuates depending on the time of a day. A common pattern in call centers is increasing of AHT at the end of the working hours. However, only service ID 1101 has the expected pattern.

In conclusion, Swiss call center is a complex call center which has relatively greater handling time than vanad data. Moreover, adding wrap-up times to handling time cause handling time being very large for some cases, wrap-up time of swiss data will be introduced in section 2.4.6. AHT fluctuates per day, however we could not find an intra-day pattern for AHT. Agents are not identified uniquely in swiss call center. This restrains the further search on handling time of swiss data.

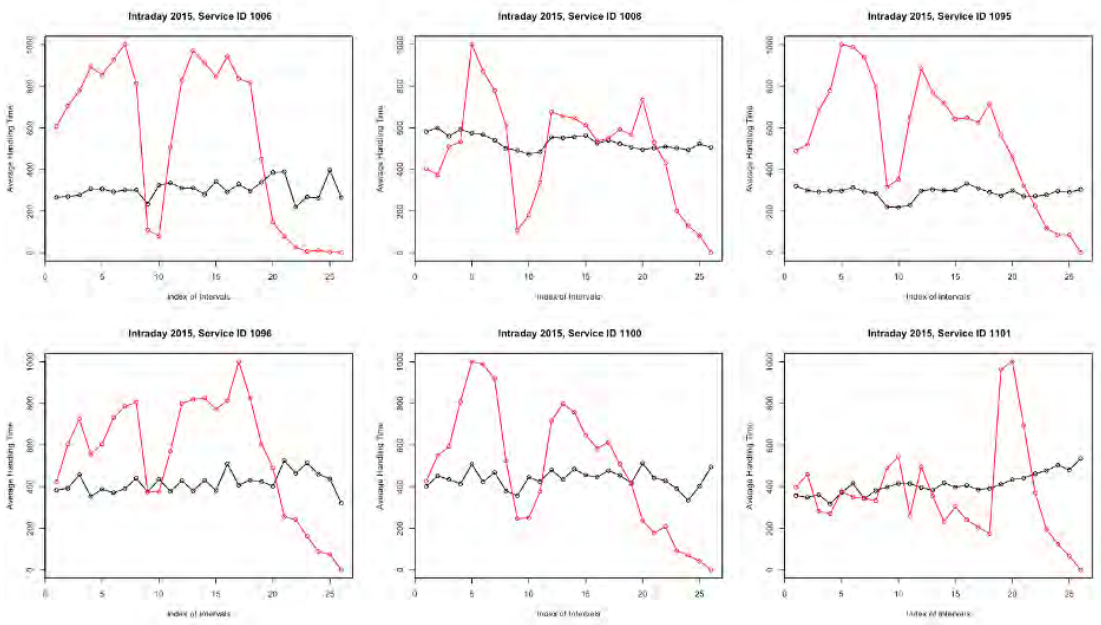


Figure 2.14: Intraday unit(seconds), AHT:Black Lines, Volume of Calls:Red Lines

2.2.3 Patience

Kaplan-Meier estimator is performed to estimate the patience of the callers [12]. Figure 2.15 shows the empirical distribution function of patience time and the hazard rate of patience respectively. The callers are very patient until 1000th second of their wait. After 1000th seconds, the number of abandonment start increasing. The callers' willingness to wait decrease drastically approximately after 25th minutes.

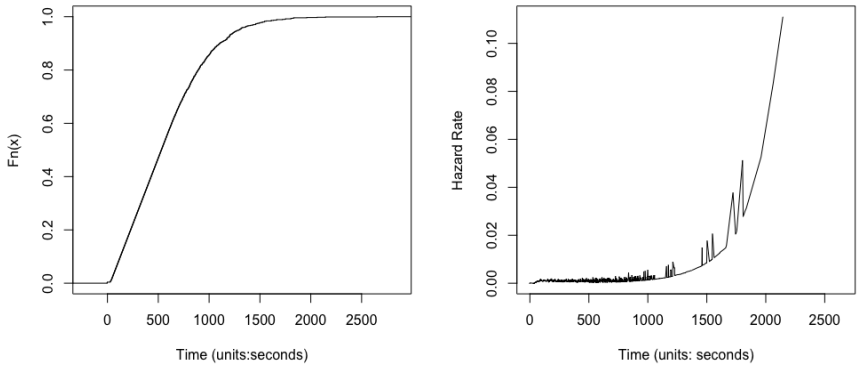


Figure 2.15: CDF and hazard rate for patience

2.2.4 Breaks

Break duration of agents are calculated from agent activity logs. Similar to Vanad data, swiss data does not have the agent shift information. Therefore we cannot distinguish 'plannable' and 'unplannable' breaks. Therefore, we include all the break times are recorded as specific activity ID which refers to 'break'. Figure 2.16 shows the break durations of agents. Agents in average has 227 seconds of break and break time has a peak around 200 seconds. Agents generally have long breaks which last 10-15 minutes during their shifts. Break duration which last more than 10 minutes comprises of 5% of the agent activity logs. This is not an usual situation to see in call centers. One can see Appendix A to check the average shrinkage percentage of the agents during the year.

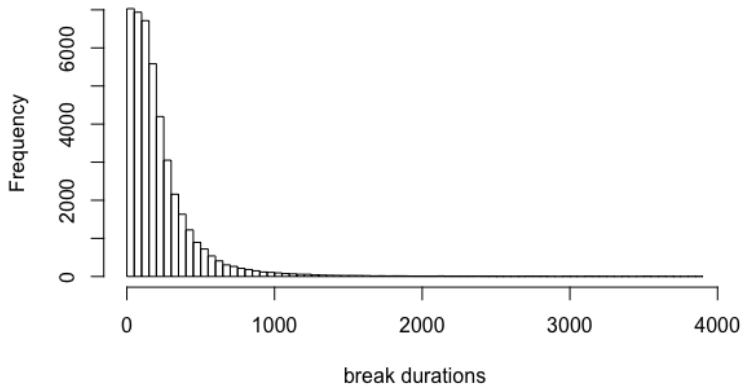


Figure 2.16: Histogram of break durations unit(seconds)

2.2.5 Wrap-Up

The way of recording of wrap-up time can change call center to call center. Some call centers only record wrap-up time as an agent activity and they do not have the information about wrap-up time in call-log data. This leads to take an average wrap-up time of all the agents. Therefore, there would be one wrap-up time for any skills and it can cause miscalculations of handling time. The most accurate way to record wrap-up time is recording it by calls.

The swiss call center data has records of wrap-up times in the call-log data. A call can have a zero wrap-up time. It means the agent who handled the call has finished the wrap-up during the call. However, the wrap-up times of Swiss data are higher than the vanad data. From Figure 7.2 we can see that there are many cases that wrap-up time is higher than 60 seconds. Moreover, there are some extreme cases of wrap-up time being 20000 seconds. For the sake of readability the graph, the wrap-up times being more than 5000 seconds are deleted from the histogram.

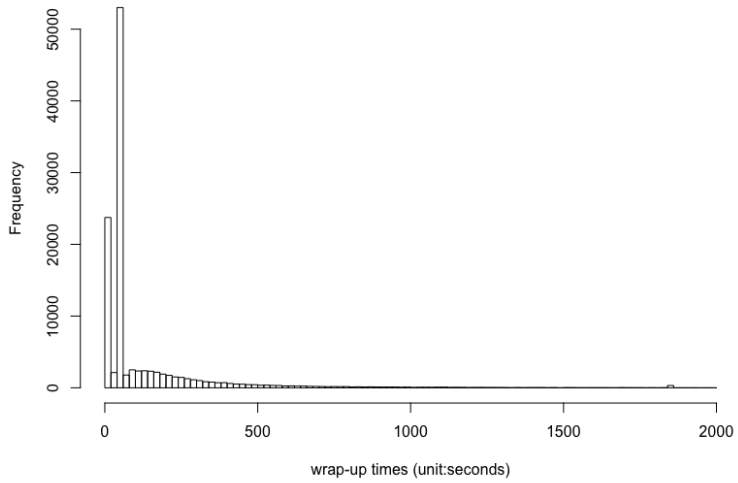


Figure 2.17: Wrap-up Times Histogram

Unlike Vanad data, the wrap-up time of swiss data is provided call by call. Thus, we can check the wrap-up time of different service types. Figure 2.18 shows wrap-up time of 7 different service types during the year 2016. The calls from 7 selected services represent 99% of the data. As one can see from the wrap-up distributions, the wrap-up times indeed different for each service type.

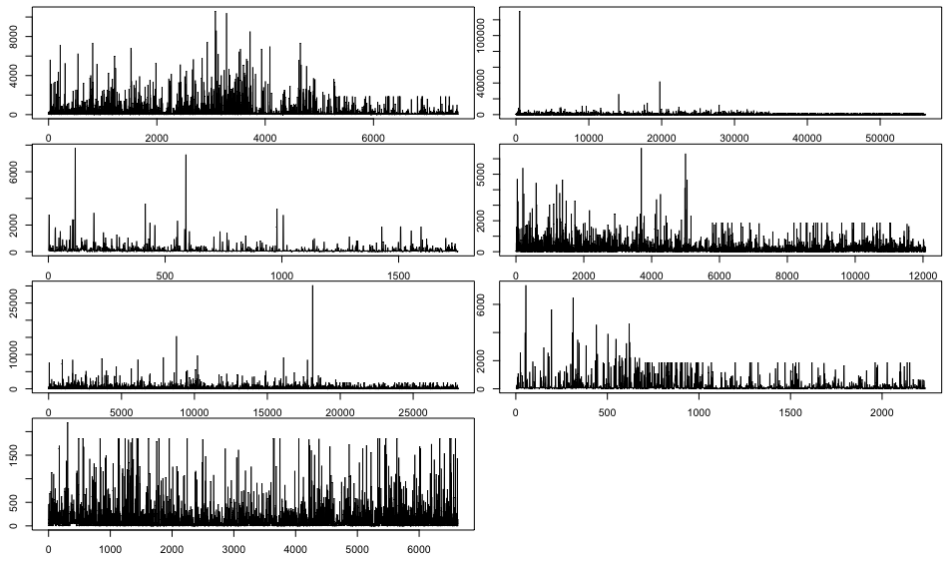


Figure 2.18: Wrap-up Times

2.3 Comparison of Data Sets

Swiss data set is unique in terms of its properties concerning handling time, wrap-up time and break time. It has a complex structure to analyze and simulate. The routing policy is not FCFS and it has more sophisticated procedure than Vanad. Simulation such a call center may lead to misrepresentation of the call center model and may give misleading results. In the view of such information, it is agreed to use Vanad data for validation. Vanad data is not complex as swiss data and it is more or less analogous with the same size multi-skill call centers in terms of its properties concerning arrival process, handling time, wrap-up time and break time. Therefore, it is more likely to generalise the results.

Even though the second data set is not validated, it is powerful to show its analysis. Because the validation results should be strengthened with validating another data set, however validating such a unique call center like Swiss data and comparing the results with the first data set would not be fair. Therefore, for future work, the secondary data should be chosen from a call center with a similar size with Vanad data and similar patterns.

Chapter 3

Methodology

In this chapter, methodology used for validation is introduced.

3.1 Validation Set Up

This section describes the methods which are used to validate call center assumptions. The model descriptions of the validation part is introduced in this section.

In order to validate call center assumptions, we create different models and each assumption is incorporated as a different model. As input we use the parameters from call log data and activity log data. First we divide the operation hours which is between 08:00:00 and 20:00:00 into 24 intervals of 30 minutes. For each interval the following parameters are calculated: Number of arrivals, HT, wrap-up time, number of agents that works in selected services and patience of customers. Acceptable waiting time is decided as 60 seconds.

To compare the model's performance measures, first the performance measures of real data is calculated. These performance measures that are calculated from call log data are labeled as the actual SL, actual ASA and actual Ab. Each model is built with a one of the assumptions which are explained in section 1.2, of the following features, Arrival, HT, AHT per day, Wrap-up, Patience and Breaks. The assumptions of features can be interpreted as follows:

Arrival: Arrivals being 'Empirical' means that the arrival times are identical in simulation and in real data. We calculate the arrival moment of a call in terms of seconds and give as an input to the simulation. For example, if there is an arrival at time t in real data, we schedule an arrival at the exact same moment in simulation. Arrivals being 'IPP' means, calls arrive with in-homogenous Poisson process with piecewise constant rate. For each intervals in simulation, we use the rate of intervals and create arrivals in simulation. For instance, if there is A arrivals within certain interval in the data, then we schedule A' arrivals within that interval in simulation, with A' is Poisson(A).

HT and AHT per day: HT being 'Empirical' means that in simulation for each customer we select a random number from the handling time of calls from that day or from the whole year. If AHT per day is 'Yes', random handling time is selected from that specific day's handling times. If AHT per day is 'No', random handling time is selected from the whole year. HT being 'Exp' means that we assume HT has an exponential distribution. The mean of the exponential distribution depends on the AHT per day assumption. If AHT per day is 'Yes', the mean of the distribution is the average of the HT over that day in simulation. If AHT per day is 'No', the mean of the distribution is the mean of HT over the whole year.

Wrap-up: Wrap-up being 'Yes' means, in simulation the mean of wrap-up times over a year is added to the HT. Wrap-up being 'No' means, the wrap-up time is ignored.

Patience: Patience being 'Empirical' means, a random patience is assigned to a customer in simulation. The random patience is generated from the empirical CDF estimated by the Kaplan-Meier Estimator. Patience being 'Exp' means that patience has exponential distribution with its mean being the empirical mean of the patience from the real data.

Breaks: Breaks being 'Yes' means, if an agent takes a break, the duration of the break is subtracted from the duration of working time (the time that agents busy with answering calls or wrap-up or waiting for a call) of that interval. In this manner the proportional staffing level that required during the break is subtracted the total staffing level of that interval. If breaks 'No', the total staffing level is not changed. For example, agents worked 300 minutes in a specific interval. 2 agents have breaks in that interval and each break lasts 10 minutes. In total 20 minutes of breaks occurs in that interval. If breaks is 'Yes' the total staffing level of that interval is (300-20)/30 = 9.3, in simulation it is rounded to 9. If breaks is 'No', the total staffing level of that interval is 300/30 = 10 in simulation.

	Arrival	HT	AHT per day	Wrap-up	Patience	Breaks
Base Model	Empirical	Empirical	Yes	Yes	Empirical	Yes
Arrival Model	IPP	Empirical	Yes	Yes	Empirical	Yes
HT Model	IPP	Exp	Yes	Yes	Empirical	Yes
Patience Model	IPP	Empirical	Yes	Yes	Exp	Yes
HT and Patience Model	IPP	Exp	Yes	Yes	Exp	Yes
Breaks Model	IPP	Empirical	Yes	Yes	Empirical	No
Wrap-up Model	IPP	Empirical	Yes	No	Empirical	Yes
Yearly Model	IPP	Empirical	No	Yes	Empirical	Yes

Table 3.1: Models.

3.2 Kaplan Meier Estimator for Patience

Kaplan Meier is used to estimate survival function from lifetime data [12]. In our case, survival is the waiting time of a customer who waited until he gets a service from an agent. Thus, survival is the waiting time of customers who wait until they get service.

At the end we are interested in the abandonment probability of the customers. It can be derived from the survival probability of customers.

The call types can be divided into several categories. In this problem we are interested only in abandoned and answered calls. 'Answered' are the calls that agents pick up and provide services. 'Abandoned' calls are the calls that customers do not wait until an agent pick up and they leave before the service is provided.

Answered call logs are censored data. Censored data could be defined as the observations for which the time until the event of interest is unknown yet for some information exist. Answered calls are served before they reached the limit of their patience. The event of interest is the patience of these answered calls. This information is unknown, however; the waiting time before they are served is available. If the calls are all abandoned, the data is not censored since the time when they reached the limit of their patience is known. Thus we do not need to use Kaplan-Meier if all the calls are abandoned.

The estimator of survival function is given by;

$$\hat{s}(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}) \tag{3.1}$$

where;
$\hat{s}(t_i)$ = The probability that life is longer than t
d_i = The number of events that happened at time t_i
n_i = The individuals known to have survived up to time t
The probability of survival is s(t) and the abandonment probability is 1-s(t).

let us define probability density function of abandonment as f(t). Hazard rate represents the conditional probability intensity that a caller that waits t unit of time will abandon in the next moment [14].

Hazard rate is defined as:

$$h(x) = \frac{f(x)}{1 - F(x)} \tag{3.2}$$

3.3 Performance Measures of Call Centers

Service level: Percentage of calls answered within acceptable waiting time.

$$\text{Service Level} = \frac{\text{Number of Calls Handled Before Threshold}}{\text{Total Calls Offered - Total Calls Abandoned}}$$

Abandonment Rate: percentage of calls abandoned by customers before speaking to an agent.

Average Speed of Answer: The average time that a call waits before an agent answers the call. Each call is assigned to a queue according to service that is selected

by customers in IVR. A call in a specific queue is handled when an agent becomes idle and is assigned to a call. The time period starts with an agent is assigned to a call and ends with the moment that the agent answers the call is the speed of answer for that specific agent.

3.4 Model Evaluation

Weighted Average Error:

WAE measures the difference between the simulation results and the actuals. Part of the difference is caused the choice of the model. The other part is the variability of performance measures SL, Ab and ASA. The error caused by choice of the model can be calculated with the equation 3.3. The variability of performance measures is caused by the fact that there are 1000 simulation for each day. Each simulation have different performance measures. Therefore, we should also calculate the variability error.

$$WAE_x = \frac{\sum_{i=1}^{n} A_i \mid \mathbb{E}\, X_i^{sim} - X_i^{act} \mid}{\sum_{i=1}^{n} A_i} \tag{3.3}$$

$$\begin{aligned} Variability_x &= \frac{\sum_{i=1}^{n} A_i B}{\sum_{i=1}^{n} A_i} \\ B &= \frac{\sum_{j=1}^{k} \mathbb{E}\, X_i^{sim} - X_{ij}^{sim}}{k} \end{aligned} \tag{3.4}$$

where A_i is the number of arrival calls at day i. j is the number of simulations.

Confidence Interval:

As [17] discusses. confidence interval is an significant performance metric in order to make objective decision for comparing simulation results. The second performance metric is the confidence interval of the simulation outcomes. The percentages of actual SL, Ab and ASA that are within the α confidence interval of the simulation outcomes. $\alpha = 0.95$ in this research.

Quantile:

For each day, there is one actual result and 1000 simulation outcomes for SL, Ab and ASA. Therefore, the percentage of actual SL, Ab and ASA that is higher than the 50% quantile of the simulation outcomes are calculated respectively.

Chapter 4

Experiment Setup

CCMath has already the simulation code in C++. The call center is modeled via discrete-event simulation. 3 lists are created to depict the discrete events, event list, agent list and queues. One can see the Appendix C for the depiction of simulation. The change of system states triggered by events [30]. For example, an arrival happens and there is an idle agent for the arrival call, then the call is handled and "departure" event is inserted into the event list. [30] explains the simulation of multi-skill call center. The code in CCMath is parallel to the simulation structure mentioned in [30]. The difference is, [30] includes e-mails and chats to the simulation. In this research, only call services are considered in multi-skill call center. The structure of simulation code can be fined in Appendix C.

In simulation 302 days are simulated 1000 times. For a single run of simulation, various parameters are recorded. Vanad data is analyzed in R and according to the requirements of inputs in C++ code, the inputs are generated from R as txt files. These txt files are read in in C++ code for simulation.

The operation hours is between 08:00:00 and 20:00:00 and there are 24 intervals of 30 minutes. For each interval the following parameters are calculated in R: Number of arrivals, HT, wrap-up time, number of agents that work in selected services and patience of customers. Acceptable waiting time is decided as 60 seconds.

The model development is described in Chapter 3. Each model is simulated 1000 times. The computational time of a model's simulation is high which is approximately 14 hours. The results of each simulation of models are saved after the completion of simulation.

Chapter 5

Model Performance

In this section, the performance of validation is reported.

The models are simulated and the result of each model are compared according to WAE, variability, CI and quantile. First of all, numeric results are introduced in the following tables. Then how the models should be compared is explained. Table 5.1 shows the WAE and CI results. ASA is in seconds. CI is constructed with $\alpha = 0.5$.

	WAE_{SL}	WAE_{Ab}	WAE_{ASA}	$\text{I}_{\alpha,\text{SL}}$	$\text{I}_{\alpha,\text{Ab}}$	$\text{I}_{\alpha,\text{ASA}}$
Base Model	3.01×10^{-2}	8.32×10^{-3}	7.57	66.0%	37.9%	51.0%
Arrival Model	3.03×10^{-2}	7.70×10^{-3}	6.71	86.4%	78.9%	76.7%
HT Model	2.93×10^{-2}	6.72×10^{-3}	6.14	92.4%	88.0%	89.0%
Patience Model	4.88×10^{-2}	6.51×10^{-3}	12.49	62.8%	83.9%	46.4%
HT and Patience Model	4.54×10^{-2}	5.65×10^{-3}	11.83	73.4%	90.2%	57.3%
Breaks Model	8.50×10^{-2}	20.01×10^{-3}	17.34	31.0%	15.6%	16.8%
Wrap-up Model	4.23×10^{-2}	11.00×10^{-3}	9.40	72.5%	48.9%	57.8%
Yearly Model	5.70×10^{-2}	15.40×10^{-3}	12.49	64.0%	55.0%	55.9%

Table 5.1: Performance measures of models.

Variability of SL, Ab and ASA is shown in the Table 5.2.

	$Variability_{SL}$	$Variability_{Ab}$	$Variability_{ASA}$
Base Model	1.67×10^{-2}	2.69×10^{-3}	2.37%
Arrival Model	2.47×10^{-2}	4.48×10^{-3}	3.76%
HT Model	2.73×10^{-2}	5.2×10^{-3}	4.32%
Patience Model	4.01×10^{-2}	7.00×10^{-3}	2.94%
HT and Patience Model	2.43×10^{-2}	4.97×10^{-3}	3.43%
Breaks Model	1.90×10^{-2}	3.01×10^{-3}	2.82%
Wrap-up Model	2.30×10^{-2}	3.90×10^{-3}	2.54%
Yearly Model	2.48×10^{-2}	4.11×10^{-3}	3.63%

Table 5.2: Variability of performance measures

For each model the percentage of actuals above the median of simulation results are shown in the Table 5.3.

	$P(SL > Q_{0.5,SL})$	$P(Ab > Q_{0.5,Ab})$	$P(ASA > Q_{0.5,Ab})$
Base Model	29.1%	83.1%	84.4%
Arrival Model	24.5%	84.1%	85.0%
HT Model	33.8%	78.9%	78.1%
Patience Model	12.7%	75.9%	94.5%
HT and Patience Model	18.1%	70.9%	90.7%
Breaks Model	21.1%	98.7%	98.7%
Wrap-up Model	13.5%	94.9%	95.4%
Yearly Model	35.0%	69.6%	71.3%

Table 5.3: Percentage of actuals above the median.

5.1 Interpretation of Performance

For each model, the model's aim and how to compare it with the appropriate model are explained.

Base Model: Arrival, HT and patience are empirical. AHT per day is 'Yes'; thus, a random HT from the simulated day is assign to a customer in simulation. Wrap-up time is added to the HT. Breaks of agents are considered when the proportional staffing levels are calculated. This model serves as a base model for the validation and it is the only model with the assumption of empirical arrival.

Arrival Model: 'Arrival Model' differs from 'Base Model' only in terms of the arrival assumption. Arrival process follows IPP and the influence of the assumption of 'IPP' arrivals can be deducted by comparing Base Model and Arrival Model. As one can see that from Table 5.1 that whether the arrival process is 'Empirical' or

'IPP' does not have a strong influence on the accuracy in predicting performance measures SL, Ab and ASA. This can be concluded by comparing the WAE values of corresponding models. Moreover, Table 5.3 shows that the percentage of actuals above median does not change significantly.

HT Model: Since the accuracy of prediction does not change from 'Arrival Model' to 'Base Model', we can update our base model as Arrival Model. The only difference in the assumptions between the revised base model(Arrival Model) and HT Model is HT. Thus it can be concluded that 'HT Model' is designed to assess the influence of the exponential assumption of HT. As one can see that from Table 5.1 the exponential assumption of HT does not have a strong influence on the accuracy by comparing 'Arrival Model' and 'HT Model'. Even though 'HT Model' has a smaller error than the base model(Arrival Model), the objective of the comparison is assessing the effect of assumptions on accuracy.

Patience Model: 'Patience Model' differs from 'Arrival Model' only in patience assumption. Patience follows exponential distribution in this model. As one can see that from Table 5.1 that the patience having exponential distribution has a strong influence on the accuracy in predicting SL, Ab and ASA.

HT and Patience Model: HT and patience are exponentially distributed. Although the previous model shows that the exponential assumption of patience distribution has a strong influence on the accuracy, this model aims to show whether the exponential assumption of the patience distribution has a stronger influence on the accuracy of the models comparing to the exponential assumption of the HT. This can be concluded by comparing WAE values of models 'Arrival', 'HT', 'Patience' and 'HT and Patience'. It can be concluded that exponential assumption of the patience has stronger effect than the exponential assumption of the handling time.

Breaks Model: Agent breaks have a drastic influence on the accuracy of the models. Breaks Model is introduced as a model that does not take into consideration of agent breaks when the proportional staffing levels are calculated. The effect of agent breaks in the accuracy can be concluded by comparing models 'Arrival' and 'Breaks'.

Wrap-up Model: 'Wrap-up Model' differs from 'Arrival Model' only in wrap-up time assumption. Although the wrap-up times are in general very short in a call center, ignoring them could lead to inaccuracy. As one can see from the comparison of 'Arrival Model' and 'Wrap-up Model' from Table 5.1, wrap-up time has a strong influence on the accuracy of the prediction. Without considering wrap-up time, the weighted average error increases; in other words, it leads to the inaccuracy of the prediction.

Yearly Model: The variability of AHT of each day has a strong influence on the accuracy of prediction. This model is designed with the assumption of Empirical HT is Yes and AHT per day No. In other words, we select a random number from the empirical HT of the whole year. The day effect in HT is disregarded. By comparing 'Arrival Model' and 'Yearly Model', one can conclude that having the assumption of daily AHT has a strong influence on accuracy.

The results can be summarised as in the following: assuming NHPP of arrival process does not effect the accuracy of predicting performance measures. Therefore, the base

model is updated as Model 2. The exponential assumption of handling time does not effect the accuracy of predicting performance measures. On the other hand, assumption of exponentially distributed patience cause significant changes in the accuracy. Even though wrap-up time is very small, ignoring wrap-up time in the handling-time calculations lead to inaccuracy of predicting performance measures. Lastly, ignoring break time in the staffing level calculations leads to the most drastic changes in the performance measures.

Part II

Machine Learning Approach

Chapter 6

Data Analysis

From a broader perspective, the point of origin of the machine learning part of the project is to understand if call center managers can use machine learning to predict their service level instead of simulation models. This is an assertive aim and requires a lot of search and time. A good starting point is to predict the service level of single-skill models such as Erlang-C and Erlang-X rather than a complex multi-skill call center. In this section, two different data sets are explained. Queueing models Erlang-C and Erlang-X have different assumptions to calculate the service level. Both models have online calculators as products in CCMath. These calculators for the given information, such as forecasted arrivals(FC), average handling time(AHT), acceptable waiting time(AWT) and the number of agents, calculates the service level.

Two different data sets are randomly generated. [32] generates its data via simulation. They use simulation since they are predicting the performance of a multi-skill call center. Unlike single-skill call center modeling, for multi-skill call center modeling, the analytic solution does not exist; therefore simulation method is used. In this research, performance measures of queuing models are predicted. The service levels of these two data sets, which includes randomly generated features, are calculated via CCMath calculators. The first subsection explains the Erlang-C data generation and in the second subsection, Erlang-X data generation is explained. For the Erlang calculator, one can check Appendix B.

6.1 Erlang-C Data

In order to test the prediction power of the machine learning algorithms, first we need to generate a data set contains enough samples of different values of SL. It is not a necessity to have specific shape of data but it is important to cover various SL values. For example a data set with service level calculation mostly distributed on two or three values is not adequate to use for machine learning. The graph below shows that the data set is highly skewed data. As one can see from the graph service level is distributed mostly on the values 0-0.1 and 0.9-1.

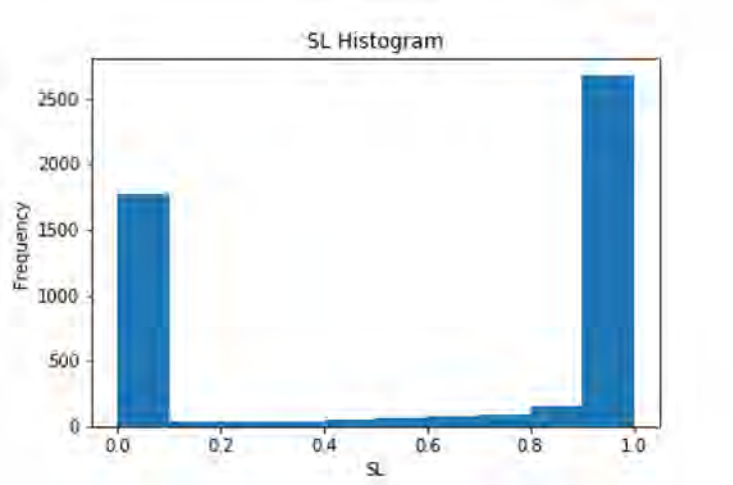


Figure 6.1: SL Histogram

The data set is generated by assigning random numbers with the following range settings of features:

Features	FC	AHT	AWT	Number of Agents
Range of Random Numbers	(1,10)	(1,10)	(0,1)	(1,150)

10000 data instances are generated. FC, AHT and AWT can be non-integer but number of agents have to be integer. The distribution of SL values can be seen from the graph above.

The important step to having a better-distributed data is choosing the number of agents according to given FC and AHT value since load and overcapacity have a significant effect on the SL calculations. If FC and AHT are chosen as random numbers between (1,8) and (1,6) correspondingly, the highest value that load can get is 48. Therefore, we have to have at least 48 agents to have SL value higher than 0 since SL is higher than 0 if and only if the number of agents is higher than load. Otherwise, if the load value is lower than number of agents, SL would be zero. This is a case that we try to avoid. There should be cases with SL being 0; however, it should not be distributed mostly on zero.

On the other hand, when number of agents is significantly higher than the load, the service level is 1. As one can see in the graph, most of the service level values are distributed on the value 1. The reason is that the range of number of agents is between 1 and 80. We can lower the range limit of number of agents, but in that case, we increase the possibility that service level being 0. The reason is the lower range leads to an increase in the possibility that number of agents being a number lower than the load value. As a result of that, we have to manipulate the process in favor of variety. Three important cases determine service level as the following:

- number of agents is higher than the load value (s >a)
- number of agents is equal to the load value (s=a)
- number of agents is lower than the load value (s <a)

Among the three cases, the second case does not cause any problem. The possibility of the second case is low. Although it happens, it does not affect the variety of service level values since the main problem of having highly skewed data is the difference between number of agents and load. Let us assume FC is 7, and AHT is 5. The load is 35 and in order to have SL higher than 0, we need at least 35 agents. The number of agents are generated randomly with the following range (1,80). Although SL is also dependent on AWT, the load being 35 and number of agents' maximum value being 80 is already causing SL mostly distributed on 0 or approximately 1. Changing AWT in case of 50 agents does not affect SL in a significant way. The Table 6.1 shows the case of 50 agents:

FC	AHT	AWT	Number of Agents	SL
7	5	0.1	50	0.991832
7	5	0.2	50	0.993949
7	5	0.3	50	0.995518
7	5	0.4	50	0.996679
7	5	0.3	50	0.997540

Table 6.1: AWT experiment

As one can see from the table, even though AWT is increasing, the changes in SL do not represent significantly different cases. For this reason, we should manipulate the way of generating random number of agents. If the load is 35, there should be at least 35 agents but not more than 40. However, this arrangement is hard to perform. Therefore, a random process is designed. First of all, the load values are calculated and the rows with the following two cases are determined: s<a and s>a. For both cases, the main idea is the same: the number of agents is approximated to the value of the load.

Case s <a
The rows that number of agents are manipulated are chosen randomly. Uniformly distributed random numbers between 0 and 1 are generated. If a generated number is higher than 0.5, then the number of agents is approximated to the load value. Otherwise, the number of agents is not changed. The approximation is performed by rounding up the load value then adding plus 1 to the rounded-up value. For example, FC is 3.6 and AHT is 2.4. FC x AHT = 8.64, Roundup (8.64)=9, Number of agents = 9+1 = 10.

Case s >a
The rows that number of agents are manipulated are chosen randomly. Uniformly distributed random numbers between 0 and 1 are generated. Instead of choosing the rows one threshold in the first case, three different ranges and added value for these ranges are determined as the following:

- If random number is between the range (0,0.25): Round up (FC*AHT + 1)
- If random number is between the range (0.25,0.5): Round up (FC*AHT + 1)

- If random number is between the range (0.5,1): Round up (FC*AHT + 1)
- If random number is between the range (0.75,1):Round up (FC*AHT)

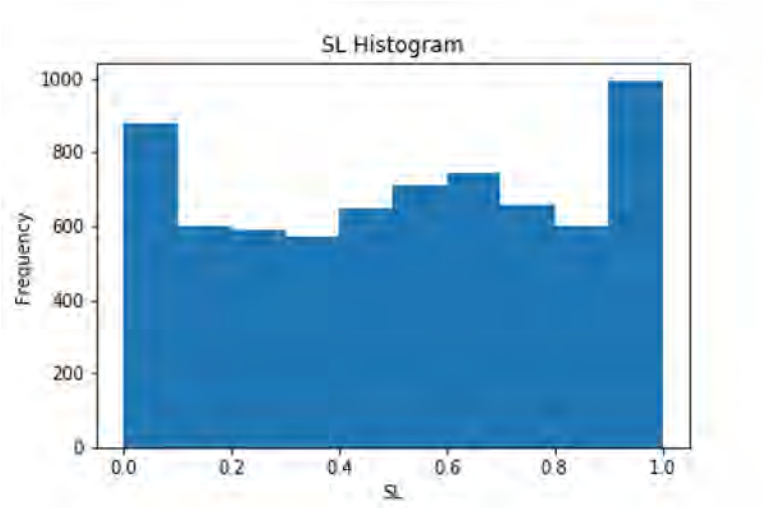
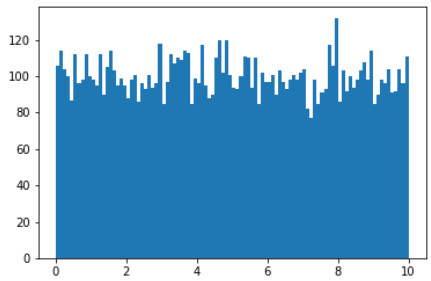
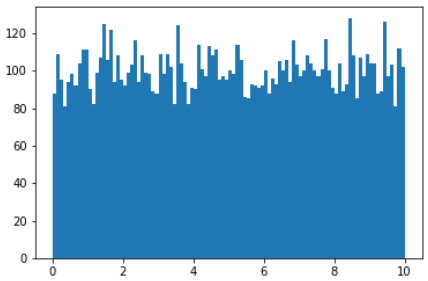


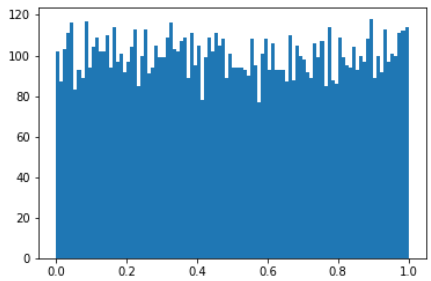
Figure 6.2: SL Histogram



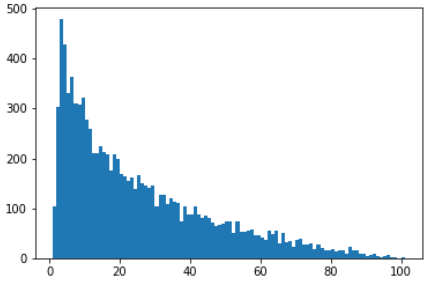
(a) Forecast Histogram



(b) AHT Histogram



(c) AWT Histogram



(d) Number of Agents Histogram

Figure 6.3: Distribution of Features

6.2 Erlang X Data

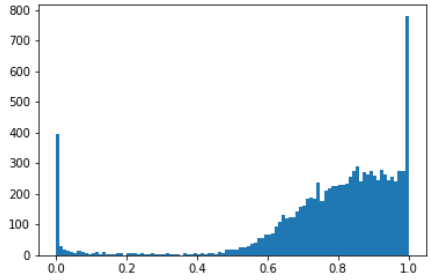
Unlike Erlang-C, Erlang-X takes the abandonments into account. A caller abandons some time later. This time can be referred to as patience time. Erlang-X is explained in detail in section 3.4. Erlang-X takes patience as input. Even though Erlang-X can model redials, it is decided to ignore redials. The number of lines is 1000, which is higher than the number of agents in any case.

Even though Erlang-X models the abandonments, the distribution of SL with the random data generation is not good enough to use in the machine learning algorithm. Because most of the points are distributed on (0.9,1), thus, the same manipulation process for the Erlang-C random input is performed to the Erlang-X input.

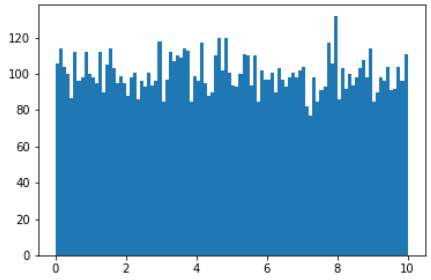
With the same setting of random data generation, Erlang-X has fewer points SL being zero. The reason is abandonments help call center to handle the other callers who still wait in the queue. Therefore, zero service level is less frequent in this abandonment setting. For the given FC, AHT, Patience, AWT and Number of Agents, SL is calculated with Erlang add-in. The random inputs are generated with the following ranges.

Features	FC	AHT	Patience	AWT	Number of Agents
Range of Random Numbers	(1,10)	(1,10)	(1,10)	(0,1)	(1,150)

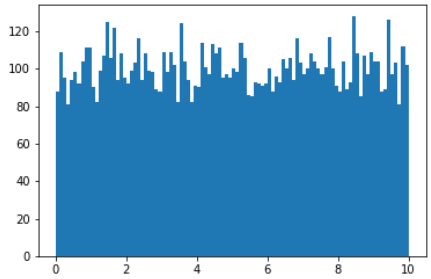
Since the same manipulation process is applied, the rows being s <a are manipulated if the random number generated is bigger than 0.5. On the other hand, the rows with random number less than 0.5, their number of agents are not approximated to the load value. In some extreme cases such as $s = 3$ and $a = 70$, the calculator gives minus value for SL which is not possible to have. SL can be in the range (0,1). This is a technical problem due to very low number of number of agents. These rows are changed as service level being 0.



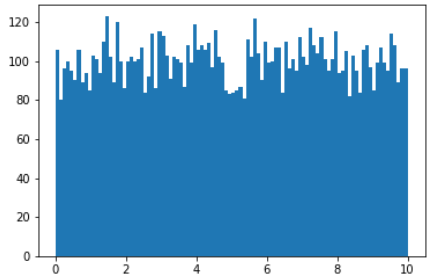
(a) SL Histogram



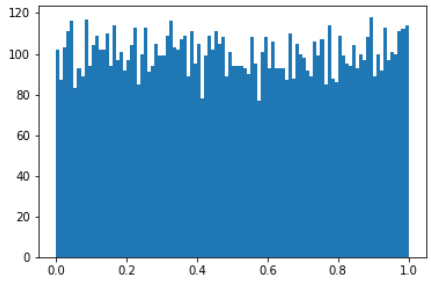
(b) Forecast Histogram



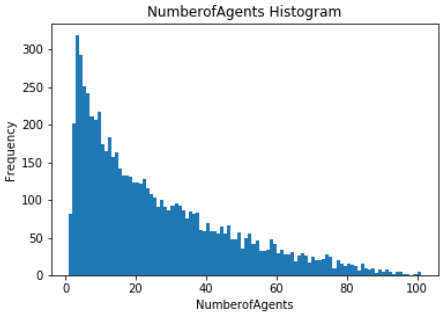
(c) AHT Histogram



(d) Patience Histogram



(e) AWT Histogram



(f) Number of Agents Histogram

Figure 6.4: Prediction Plots

6.3 Feature Engineering

Feature enginnering is an important step for developing prediction models. Extra features can be helpful for the model to explain the variability of the model and to increase the performance of the model.

Erlang-C model has four features such as FC, AHT, AWT, Number of Agents. The target variable is service level which is a continuous variable between 0 and 1. The calculations of Erlang-C model is explained in Section 3.4. Considering the equations 7.1 from Section 3.4 and the properties of Erlang-c, extra features are added. First of all, in order to have SL being larger than 0, the condition s >a must hold, where s is number of agents and a is the 'load'. 'Load' value is implemented as a new feature. 'Overcapacity' which is the difference between number of agents and load is added as

feature. Lastly, from Equation 7.1 the exponent of exponential constant is implemented as 'Feature1'. Table 8.7 shows the list of features that are used in machine learning algorithms.

Features
FC
AHT
AWT
Number of Agents
Load = FC x AHT
Overcapacity = Number of agents - Load
Feature1 = (Number of Agents - Load) x (AWT/AHT)

Table 6.2: Feature List of Erlang-C

Features
FC
AHT
Patience
AWT
Number of Agents
Load = FC x AHT
Overcapacity = Number of agents - Load
Feature1 = (Number of Agents - Load) x (AWT/AHT)

Table 6.3: Feature List of Erlang-X

6.4 Feature Scaling

In machine learning, feature scaling has significant effect on the performance of algorihm. The necessity transformation depends on the algorithm. Where random forest and XGBoost do not require feature transformation, support vector machines are sensitive to the scales of the features. Performing algorithm without any feature transformation can lead to inaccurate predictions.

Min-max scaling is applied to the features. Min-max method can scale the data in the range [0,1] or [-1,1]. The definition for the target range of features [0,1] is given in the Equation 6.1.

$$x^{'} = \frac{x - min(x)}{max(x) - min(x)} \tag{6.1}$$

It is a common approach to make logarithmic transformations in regression problems. [15] discusses that for linear regression problems where non-linearity exists between

features and target values, it may lead to better prediction results with log-transformed features. We could not find a linear relationship between the features in table 8.7 and target value. Therefore, log-transformation of features is performed for linear regression model which is explained is section 3.5.

Chapter 7

Methodology

In this section, first of all Erlang-C and Erlang-X will be introduced. Then the machine learning algorithms which are applied to predict service levels of Erlang-C and Erlang-X will be introduced.

7.1 Queueing Models

7.1.1 Erlang C

Queueing models is applied to complex systems such as call centers in order to predict the performance. When queueing models are built, some assumptions and simplifications have to be performed due to the complexity of the reality. Erlang C is one of the most important queueing model that is named after the Danish mathematician Anger Krarup Erlang. Erlang C is the traffic modelling formula that have certain assumptions to fulfill in order to apply. Although some assumptions of Erlang-C is unrealistic, some definitions of Erlang-C are valid for performance calculations of call centers. Erlang-C has the following assumptions:

- Arrivals follow Poisson distribution. In other words each interval (15 minutes or 30 minutes) has the same constant rate of arrival.
- Service duration (handling time) is random and exponentially distributed.
- The agents are homogeneous. It means they cannot be distinguished.
- There is no abandonments. The customers wait in the queue until they get served and there is no limit for the length of the queue.

In order to handle the offered traffic it is assumed that number of agents is higher than the load which is the exact number of agents that is needed to handle the traffic. Load is denoted as a and it is calculated as $a = \lambda\beta$, where is FC, β is AHT. We assume that s>a in order to have SL being larger than zero. If s <a we would have SL being zero since scause that arrivals would be more than departures. Therefore, the number of

calls in the queue would increase and they would exceed the acceptable waiting time threshold [16].

Let us assume there are s agents where s>a. The difference between s and a is overcapacity. In other words, overcapacity is the extra agents in the system. The offered load a, is shared between s agents. If the difference between s and a high, it would lead to high service level.

Service level calculation of Erlang-C is shown in the Equation 7.1. t is the AWT, in minutes. C(s,a) is the probability of delay which means the probability that a caller finds all the agents occupied. [18]

$$SL = 1 - C(s,a) \times e^{-(s/\beta - \lambda)t} \tag{7.1}$$

$$C(s,a) = \frac{a^s}{(s-1)!(s-a)} \left[\sum_{j=0}^{s-1} \frac{a^j}{j!} + \frac{a^s}{(s-1)!(s-a)} \right]^{-1} \tag{7.2}$$

7.1.2 Erlang X

Erlang-X model can be seen as an extension of Erlang-C model which takes abandonments, blocking due to finite number of lines and redials into consideration. Abandonment modeling is introducted by C.Palm and redials are introduced by Sze. These contributions now constitute the Erlang-X model. [16]

In this research we ignored the redials. Therefore, Erlang-X has the properties of the Palm/Erlang-A Queue model [19].

Erlang-X model is more robust than Erlang-C model. For example, number of agents can have drastic effects on Erlang-C model since the condition s >a must hold in order to have SL >0. Abandonment modeling helps the system to have space for other callers to get served in time even though s >a condition does not hold. Because the abandoned calls or impatient customers leave the queue and give space for the other calls waiting for the service. Average patience is required in order to calculate erlang-x performance measures.

Blocking refers that the calls are disconnected due to capacity of number of lines. Erlang-X assumes number of lines is bigger than the number of agents. Number of lines include both the number of callers wait in the queue and the number of callers having service. Since there is no space for a call to wait in the queue, call is disconnected. Therefore a long waiting time is avoided.

Redials are multiple calls coming from a specific caller. The callers that they have already abandoned the queue and they re-attempt to call. Redials are modeled by adding the average number of abandoned calls per minute to the FC [16].

In this research we ignored the redials. The number of lines is decided as 100.

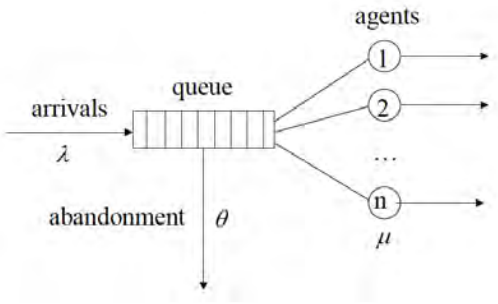


Figure 7.1: Erlang-X without redials

Erlang-X calculator can be find in http://www.gerkoole.com/CCO/erlang-x.php.

7.2 Machine Learning Algorithms

7.2.1 Linear Regression

Linear regression is one of the most common and basic regression problem. Most of the studies have been using linear regression as a base model. Let y be dependent variable and $x_1, \cdots, x_n$ explanatory variables. The linear regression model is defined as

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \cdots + \beta_n x_{in} + \epsilon_i$$

where β_0 is the intercept and $\beta_1 \cdots \beta_n$ are the slope coefficients of the explanatory variables. ϵ is the model's error term.

Linear regression is based on the following assumptions [20]:

- There is a linear relationship between the dependent variable and explanatory variables.
- There is no multicollinearity between the explanatory variables. In other words explanatory variables are not highly correlated.
- Residuals are normally distributed with a mean 0 and variance σ^2.

The distribution of explanatory variables are significant factor to have a well fitted model since linear regression cannot handle higly skewed data. Thus, transformation of explanatory variables are very common approach. In some cases, dependent variables are also transformed before training. One of the most common transformation is logarithmic transformation of the explanatory variables [15]. Logarithmic transformation lead explanatory variable to have more approximately normal distribution.

Using gradient descent algorithm is very common machine learning problems especially the data size is relatively large [21]. Especially the cases that computation time of

linear regression is problematic, gradient descent algorithm helps the computational time to reduce. Since the size of the Erlang-C data size is 10000 which is relatively small data size, the computational time does not cause any problem. Therefore gradient descent is not performed to find the optimum intercept and coefficients of linear regression.

7.2.2 Beta Regression

Service level is continuous variable in the range [0,1]. Linear regression does not provide a range for the prediction.

Beta regression is a part of generalized linear models. It is developed by Ferrari and Cribari-Neto [22]. Besides the other regression models, beta regression assumes the response variable is continuous in the range (0,1).

Beta regression has the flexibility to handle heteroskedasticity. The beta density can have several different shapes according to the combination of parameter values.

Beta distribution is a continuous probability distribution defined over the range (0,1) with the following density function with two shape parameters.

$$f(y;\mu,\phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, 0<y<1,$$

where $0<\mu<1$, $\phi>0$ and $\Gamma(.)$ is the gamma function.

Parameters μ denotes the expected value of y , i.e. $\mathrm{E}(y)=\mu$ and ϕ denotes the precision parameter and $\mathrm{Var}(y)=\mu(1-\mu)/(1+\phi)$.The response variable y is defined as $y \sim \mathcal{B}(\mu,\phi)$

Let $y_1, \cdots, y_n$ be a random sample such that $y \sim \mathcal{B}(\mu_i,\phi)$, i=1,$\cdots$,n. The beta regression model is defined as

$$g(\mu_i) = x_i^\top B = \eta_i$$

where $B=(B_1,\cdots,B_k)$ is a $k\times 1$ vector of unknown regression parameters (k<n) $x_i = x = (x_{i1},\cdots,x_{ik})$ is the vector of k independent variables and η_i is a linear predictor. To map linear predictor into the space of observed values which are in the range of (0,1) g(.) link function is performed.

$g(\cdot):(0,1)\mapsto\mathbb{R}$ is a link function. $g(\cdot)$ is strictly increasing and twice differentiable. There are different choices for $g(\cdot)$ link function. It is decided to use logit function $g(\mu)=\log(\mu/(1-\mu))$as link function $g(\cdot)$.

The parameter estimation is performed by maximum likelihood.

The log-likelihood function is $l(B,\phi)=\sum_{i=1}^{n} l_i(\mu_i,\phi)$, where

$$l_i(\mu_i,\phi) = \log\Gamma(\phi) - \log\Gamma(\mu_i\phi) - \log\Gamma((1-\mu_i)\phi) + (\mu_i\phi-1)_i + (1-\mu_i)\phi - 1log(1-y_i)$$

For the extreme cases of response variable being 0 or 1, before applying beta regression the response variables have to be transformed via the following expression proposed

by [22]

$$\frac{y(N-1)+0.5}{N} \tag{7.3}$$

where N is the sample size

After the prediction of test set, the predictions can be re-transformed with the inverse of Equation 7.3.

7.2.3 Support Vector Machine

Support vector machine is commonly used machine learning algorithm for both classification and regression problems. For classification task basically algorithm draws a hyper-plane that classifies the data into categories within the margin of error. Regression problems are also solved by the same logic of classification problems. Support vector machine which can solve regression problem is named as Support Vector Regression.

The first problem that one can encounter is the independent variables being linearly separable or not. In case of not being linearly separable, SVR uses kernel functions in order to map the variables into a higher dimensional space. In this way the separation becomes easier. The visualisation of SVR for linear and non-linear regression is shown in the figure below.

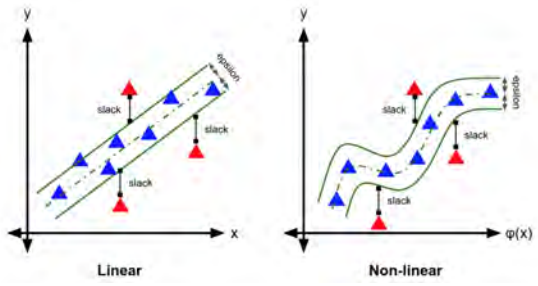


Figure 7.2: SVR for linear and non-linear regression
Source:https://medium.com/it-paragon/support-vector-machine-regression-cf65348b6345

Any function f(x) in support vector regression can be defined as [23]:

$$f(x) = w^T \phi(x) + b \tag{7.4}$$

where w are weights and $\phi(x)$ is the kernel function that maps the features into higher dimension feature space. The cost function of SVR is defined as:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\zeta_i + \zeta^*) \tag{7.5}$$

with the following constraints:

$$\begin{aligned} y_i - wx_i - b &\leq \epsilon + \zeta_i \\ wx_i + b - y_i &\leq \epsilon + \zeta_i^* \\ \zeta_i,\zeta_i^* &\geq 0 \end{aligned} \tag{7.6}$$

where ζ is slack variable, y is the target variable and x is the independent variable. The final form of SVR is defined as:

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)K(x_i, x) + b \tag{7.7}$$

Non-linear SVR is used for this research and kernel function is decided as Radial basis function (rbf) kernel.

Radial basis function is used to map the features into a higher dimensional space. The mapping of features $\boldsymbol{x}$ to $\boldsymbol{x'}$ via rbf kernel K , is defined as

$$K(\boldsymbol{x}, \boldsymbol{x'}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x'}\|^2}{2\sigma^2}\right)$$

7.2.4 Random Forest

Random forest is one of the most popular and powerful machine learning algorithm for both classification and regression tasks. It is an ensemble learning method which combines multiple models into one larger model by bagging method. Bagging, or bootstrap aggregation is a method to reduce variance and increase the accuracy. Thus it helps to reduce the chance of over-fitting.

Random forest is based on decision trees. Each tree is built by bagging. In other words, trees are built with a random subset of the data by resampling the data with replacement. At the end, the final prediction is made by combining all the tree predictions.Thus, instead of making prediction based on individual decision trees, it combines multiple decision trees [24]

General algorithm for bagging:

Algorithm 1: Bagging Trees

for *For b = 1 to B* **do**
  Draw a bootstrap sample size of $\mathbf{Z_b}$ from the training data with replacement
  Train a tree T_b;
end
Predictions ← average of all the trees

While building decision trees with random forest, number of features that is used in individual tree, number of estimators which refers to the number of trees and minimum leaf size can be decided. A representation of random forest algorithm can be seen in Figure 7.3.

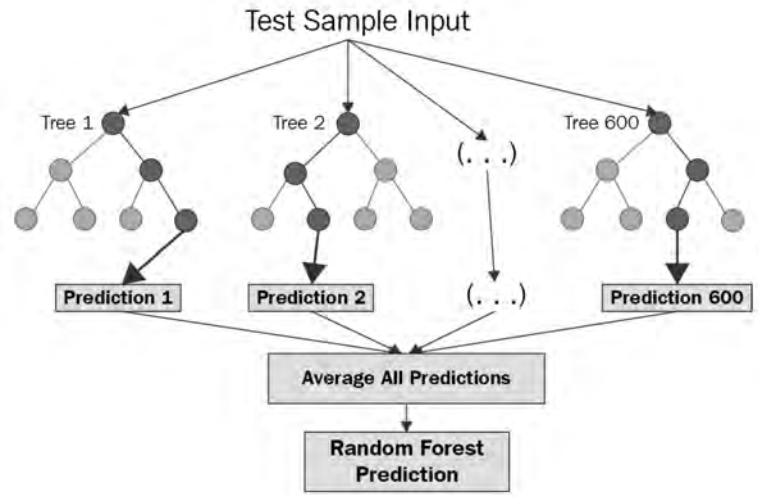


Figure 7.3: Random Forest
Source:https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

7.2.5 Gradient Boosting

Gradient Boosting is one of the ensemble learning method for both classification and regression tasks. It is similar to decision trees in terms of structure. But in gradient boosting, each tree is a modified version of the former tree. In this way the weak learners become strong learners. The difference between the bagging and boosting is, where bagging generates the trees individually, boosting generates sequentially and dependently to the former tree. Gradient boosting is implemented by Friedman in 1999 [25].

In this research, a decision-tree based algorithm which is implemented by gradient boosting framework is used. XGBoost library, which is implemented by [26], is used for gradient boosting. The model which is implemented by gradient boosting is mentioned as XGBoost in the following sections. The general algorithm for gradient

boosting tree is shown in Algorithm 2.

Algorithm 2: Gradient Boosting Tree

Initialize model with F_0 where $F_0 = \arg\sum_{i=1}^{N} L(y_i, \gamma)$
for *For b = 1 to B* **do**
For $i = 1, 2, \cdots$,N compute pseudo residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

Fit a regression tree
For $j = 1, \cdot, J_m$
end

For a given data set $D = (x_i, y_i) : i = 1, \cdots, n, x_i \epsilon R^m, y_i \epsilon R$ where x_i is the feature and y_i is the response variable. Gradient boosting is an additive model. Therefore, it uses K additive functions in the tree ensemble to predict the output [26].

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \epsilon F \tag{7.8}$$

where $f_k(.)$ is a learner, K is the independent tree structure and F is the space of regression trees. This structure of tree boosting can be used for both regression and classification problems.

Optimal parameters are found by minimizing the loss function

$$L(\phi) = \sum_{i} l(\hat{y_i}, y_i) + \sum_{k} \Omega(f_k) \tag{7.9}$$

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{7.10}$$

where l is the differentiable convex loss function and Ω is the regularization term. T represents the number of leaf nodes of the tree. w is is the weight of the leaf nodes.

Equation (7.9) and (7.10) are optimized in Euclidean space. However, for tree ensemble model optimization cannot be performed with the traditional methods. Therefore, the model is trained in an additive method which is shown in Equation (7.8)

In each iteration, let us assume it is the t-th iteration, it adds f_t that improves the model. The new loss function has the form of:

$$L^t = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)}) + f_t(x_i)) + \Omega(f_t) \tag{7.11}$$

In this way, the weaker learner f_t becomes strong learner. The new loss function is minimized via Taylor second order expansion.

$$L^t = \sum_{i=1}^{n} \left[l(y_i, \hat{y_i}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{7.12}$$

where

$$g_i = \frac{\partial l(y_i, \hat{y_i}^{(t-1)})}{\hat{y_i}^{(t-1)}} \tag{7.13}$$

$$h_i = \frac{\partial^2 l(y_i, \hat{y_i}^{(t-1)})}{\hat{y_i}^{(t-1)}} \tag{7.14}$$

In order to simplify the Equation 7.12, the constant terms can be removed. Then the equation becomes:

$$\tilde{L^{(t)}} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{7.15}$$

q is the tree structure and the instance set of leaf j can be written as $\mathrm{I}_j = \{i | q(x_i) = j\}$.

The regularization term Ω is defined in the Equation 7.10. let us rewrite it in the Equation 7.12:

$$\begin{aligned} \tilde{L^{(t)}} &= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[(\sum_{i \epsilon I_j} g_i) w_j + \frac{1}{2} \big(\sum_{i \epsilon I_j} h_i + a \big) w_j^2 \right] + \lambda T \end{aligned} \tag{7.16}$$

The optimal weight w_j of the leaf j is computed as the following:

$$w_j^* = -\frac{\sum_{i \epsilon I_j} g_i}{\sum_{i \epsilon I_j h_i} + \lambda} \tag{7.17}$$

Then the loss function at the leaf node j for the tree structure q becomes:

$$\tilde{L^{(t)}}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \epsilon I_j} g_i)^2}{\sum_{i \epsilon I_j} h_i + a} + \gamma T \tag{7.18}$$

After each branching(splitting), the information gained can be formulated as:

$$Gain = \frac{1}{2} \left[\frac{(\sum_{i \epsilon I_{jL}} g_i)^2}{\sum_{i \epsilon I_{jL}} h_i + a} + \frac{(\sum_{i \epsilon I_{jR}} g_i)^2}{\sum_{i \epsilon I_{jR}} h_i + a} - \frac{(\sum_{i \epsilon I_j} g_i)^2}{\sum_{i \epsilon I_j} h_i + a} \right] - \gamma \tag{7.19}$$

where $I = I_L \cup I_R$, L and R represent the left and right nodes after the split respectively. [26]

7.3 Hyperparameter Tuning

During the implementations of machine algorithms in Python or R, the algorithms perform with the default values of hyperparameters. However, they are not necessarily the optimal learning parameters. Hyperparameter tuning is a process of choosing the optimal set of learning parameters of a machine learning algorithm. There are several approaches to perform tuning such as manual search, grid search, random search and Bayesian optimization.

Random search is simply giving a range for a learning parameter and in each step randomly selecting a combination from the given grid set of range. Random search can be expensive in terms of computational time. The difference between Grid search and random search, where grid search every combination of hyperparameters are tried random search chooses random combinations. On the other hand, it can be efficient for tuning one or two parameters without causing any computational time problems.

7.3.1 Bayesian Optimization of Hyperparameters

Hyperparameter tuning can be expensive in terms of computational time. Algorithms such as gradient boosting has many learning parameters. Bayesian optimization of hyperparameters is efficent when the data set is large and there are several combinations of parameters to be evaluated since the search space grows exponentially when the number of parameters to be tuned increase.

Grid search and random search are common methods for hyperparameter tuning. The advantage of Bayesian optimization over grid search and random search is, it keeps track of the previous evaluations of the parameters. Grid and random search try every possible combination of parameter sets, therefore evaluation takes longer than Bayesian optimization. Bayesian optimization chooses the parameters according to the previous iteration, the past information. These information is used to form a probabilistic model mapping hyperparameters to a probability score on the objective function. It can be shown as $P(score|hyperparameter)$. Bayesian optimization uses Gaussian processes [27].

The generic algorithm for Bayesian Optimization is shown below.

Algorithm 3: General algorithm for Bayesian Optimization

```
Build a surrogate (P(score|hyperparameter)) probability model of the
 objective function f
for For b = 1 to B do
    Find the hyperparameters that performs well on the surrogate model
    Check the true objective function with the hyperparameters
    Update surrogate model with the new results
end
```

Surrogate model is the model used for approximating the objective function. The objective function is the black-box function f, in other words the machine learning algorithm. Acquisation function is the function that reflects the location we want to evaluate.

Bayesian optimization is a Sequential Model Based Optimization. Sequential refers to the learning process from the evaluation of different hyperparameters and update the surrogate model. SMBO differs according to the choice of surrogate function. In this research, Tree-Structured Parzen Estimator(TPE) is used.

Instead of using $P(y|x)$, TPE uses $P(x|y)$ which means $P(hyperparameter|score)$. TPE defines $P(x|y)$ with the following densities:

$$P(x|y) = \begin{cases} l(x) \text{ if } y < y^* \\ g(x) \text{ if } y > y^* \end{cases}$$

where x is the selected set of parameters, y is the corresponding value of the objective function using parameter x. y^* is the threshold value of the objective function.

Surrogate probability is defined. The next step is to choose acquisition model. TPE uses maximization of expected improvement (EI). EI is proportional to the ratio $l(x)/g(x)$, therefore the ratio is maximized. This refers to having parameters x with high probability/density under $l(x)$ and lower density under $g(x)$. In each iteration the next parameter is taken as the point which maximize the ratio $l(x)/g(x)$.

7.4 Model Evaluation

In order to explain the performance of models and to make fair comparison between models, model evaluation metrics are required. The proposed metrics are explained in the following sections 3.6.1 and 3.6.2.

7.4.1 R Squared

R^2 is a statistical measure to assess the proportion of variance in the dependent variable that can be explained by the independent variable. In other words, R-squared shows the power of model in terms of explaining the variability in the model.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Residual sum of squares:

$$SS_{res} = \sum_i (y_i - f_i)^2$$

where y_i is the actual and f_i is the predicted value.

Total sum of squares:

$$SS_{tot} = \sum_{i} (y_i - \overline{y})^2$$

where where y_i is the actual and $\overline{y}$ is the average of actual values.

The mean value of the actual data:

$$\overline{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

7.4.2 MSE

MSE assess the quality of predicted values. Each error caused by prediction is $y_i - \hat{y}_i$ where y_i is the actual value of ith data point and $\hat{y}_i$ is the predicted or estimated value of ith data point.

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Chapter 8

Experiment Setup

Python programming language is used to implement machine learning algorithms. Only for beta regression model, R programming language is preferred. The training data comprises of % 70 of the data, whereas the test data contains %30 of the data. The training data is used to train the models and tune the parameters. The test set is only used to assess the performance of the models. The same random seed is used for train/test splitting.

8.1 Linear Regression

The implementation of linear regression is performed by scikit-learn module [29]. Scikit-learn is one of the most popular open-source libraries in Python which provides large variety of machine learning algorithms. Scikit-learn implements ordinary least squares which is a type of linear least squares method that estimates the weights of the independent variables by minimizes sum of squares of the difference between the dependent variable and prediction.

8.2 Beta Regression

Instead of python, beta regression model is built in open-source R statistical computing environment since an extensive package, *betareg* [22] exists in R. In order to have fair comparison, the same train and test set which are created in python environment are used as train and test set since different programming language can have different settings for random seed which may lead to different train/test split.

8.3 SVR

The implementation of SVR is performed by scikit-learn module [29]. SVR has two hyperparameters such as C and gamma. Hyperparameter tuning of SVR is performed

with a 5-fold cross validation and 100 iterations via Bayesian Optimization.

Hyperparameter	Range
C	(0.01,200)
Gamma	(0.01,100)

Table 8.1: Parameters for SVR for Erlang-C and Erlang-X

The best parameter setting from the optimization process is shown in the Table 8.3.

Hyperparameter	Value
C	62.89
Gamma	0.025

Table 8.2: Parameters for SVR Erlang-C

The range of parameters for optimization process are the same with Erlang-C setting. Table 8.1 represents the range for both Erlang data. Table 8.3 shows the best performing C and gamma parameters of Erlang-X data.

Hyperparameter	Value
C	59.41
Gamma	0.011

Table 8.3: Parameters for SVR Erlang-X

8.4 Random Forest

Scikit-learn Random forest implementation is performed [29]. Random forest is sensitive to its hyperparameters. Bayesian optimization is used with a 5-fold cross validation and 100 iterations to find the optimal parameters. However, not all hyperparameters are optimized in order to reduce the computational time and to avoid increasing the complexity of the optimization process. Table 8.4 shows the hyperparameters to be tuned for Erlang-C data. The explanation of each hyperparameter with their search range is also provided in the table.

Hyperparameter	Explanation	Range
Number of estimators	Number of trees	[100,1000]
Max features	Number of features	[1,11]
Max depth	Maximum depth of a tree	[1,50]

Table 8.4: Bayesian Optimization settings for Random Forest

The best performing parameter setting that is found with Bayesian Optimization process is shown in the Table 8.5.

Hyperparameter	Value
Number of estimators	111
Max features	10
Max depth	21

Table 8.5: Optimal hyperparameters for Random Forest

Table 8.6 shows the hyperparameters to be tuned for Erlang-X data. The explanation of each hyperparameter with their search range is also provided in the table.

Hyperparameter	Explanation	Range
Number of estimators	Number of trees	[100,1000]
Max features	Number of features	[1,12]
Max depth	Maximum depth of a tree	[1,50]

Table 8.6: Bayesian Optimization settings for Random Forest

Hyperparameter	Value
Number of estimators	262
Max features	8
Max depth	45

Table 8.7: Optimal hyperparameters for Random Forest

8.5 XGBoost

The implementation of XGBoost is performed by XGBoost module of python [26]. XGBoost is sensitive to its hyperparameters as random forest. Therefore, hyperparameter tuning perfomed via Bayesian optimization, with a 5-fold cross validation and 100 iterations in order to find the optimal parameters. Table 8.8 shows the range that optimization process performed for Erlang-C data. Some of the parameters are not considered to reduce the complexity of the optimization process and avoid the high computational time.

Hyperparameter	**Explanation**	**Range**
Number of estimators	Number of trees	[100,1000]
Learning Rate		[0,1]
Max depth	Maximum depth of a tree	[3,10]
Min child weight		(0,5]
Subsample		[0.5,1]

Table 8.8: Bayesian Optimization settings for XGBoost Erlang-C

Table 8.9 shows the best performing parameters for Erlang-C data. The non-mentioned parameters are chosen as their default values [31].

Hyperparameter	**Value**
Number of estimators	274
Learning Rate	0.024
Max depth	6
Min child weight	3.069
Subsample	0.706

Table 8.9: Optimal hyperparameters for XGBoost

Table 8.10 shows the search range of Bayesian optimization for Erlang-X data. Hyperparameter tuning performed with 5-fold cross validation and 100 iterations.

Hyperparameter	Explanation	Range
Number of estimators	Number of trees	[100,1000]
Learning Rate		[0,1]
Max depth	Maximum depth of a tree	[3,10]
Min child weight		(0,5]
Subsample		[0.5,1]

Table 8.10: Bayesian Optimization settings for XGBoost for Erlang-X

Table 8.11 shows the best performing parameters for Erlang-X data. The non-mentioned parameters are chosen as their default values [31].

Hyperparameter	Value
Number of estimators	726
Learning Rate	0.2096
Max depth	5
Min child weight	4.882
Subsample	0.8286

Table 8.11: Optimal hyperparameters for XGBoost

Chapter 9

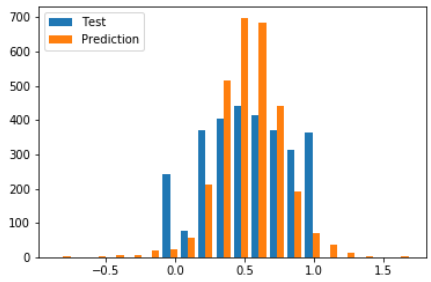
Model Performance

In this section, Erlang-C and Erlang-X prediction results is reported. Firstly, Erlang-C predictions of 5 different machine learning algorithms are presented. Afterwards, Erlang-X predictions are reported. The interpretation of results and comparison of Erlang models are discussed thereafter the reporting of results.

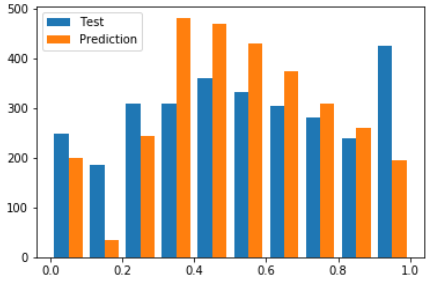
The MSE and R squared values of 5 different models with the range of the predictions made, are shown in the Table 9.1. Figure 9.1 shows the comparison of predictions and real values of service levels for 5 different machine learning algorithms.

	MSE	R Squared	Prediction Range[min,max]
Linear Regression	0.0283	0.6687	[-0.87,1.7]
Beta Regression	0.0197	0.8986	[7.144081e-05,1.000214]
SVR	0.0066	0.9212	[-0.2,1.19]
Random Forest	0.000007	0.9990	[0,1]
XGBoost	0.00010	0.9987	[-0.030,1.018]

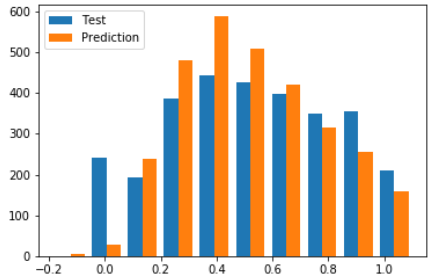
Table 9.1: Erlang-C Machine Learning Models Results



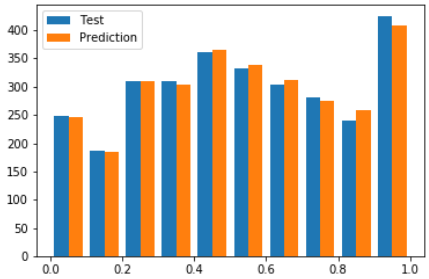
(a) Linear Regression



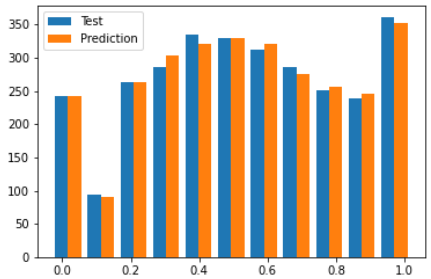
(b) Beta Regression



(c) SVR



(d) Random Forest



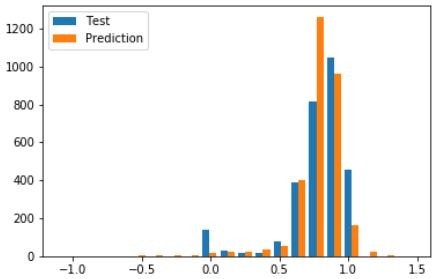
(e) XGBoost

Figure 9.1: Distribution of Features

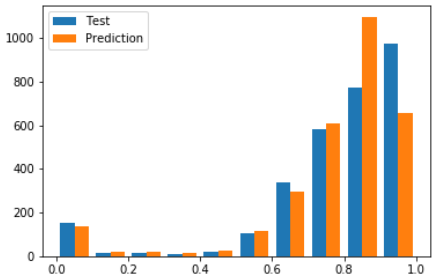
The MSE and R squared values of 5 different models of Erlang-X data with the range of the predictions made, are shown in the Table 9.2. Figure 9.2 shows the comparison of predictions and real values of service levels for 5 different machine learning algorithms.

	MSE	R Squared	Prediction Range[min,max]
Linear Regression	0.0136	0.7356	[-1.10,1.47]
Beta Regression	0.005	0.8946	[7.144832e-05,1.000214]
SVR	0.0039	0.9232	[-0.2,1.13]
Random Forest	0.00082	0.9839	[8.433050489235956e-11,1]
XGBoost	0.00043	0.9915	[-0.03,1.05]

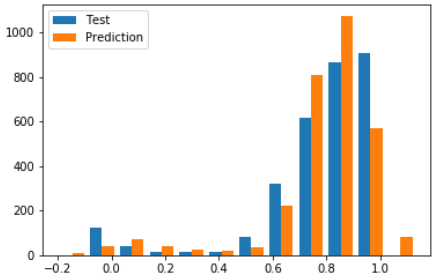
Table 9.2: Erlang-X Machine Learning Models Results



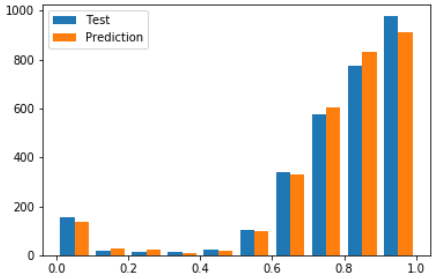
(a) Linear Regression



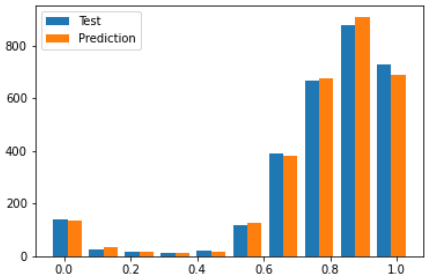
(b) Beta Regression



(c) SVR



(d) Random Forest



(e) XGBoost

Figure 9.2: Distribution of Features

9.1 Interpretation of Performance

The common result for both Erlang-C and Erlang-X service level prediction models is that random forest and XGBoost outperform among other machine learning models. The worst model is linear regression for both. Beta regression model has better results than linear regression, it cannot outperform SVR, random forest and XGBoost.

Even though MSE and R squared are significant determinants to choose the best performing model among machine learning models, the range of predictions is also important for us to decide which model has the best performance. Service level has the range of [0,1] and it is not possible to SL being minus values or values larger than 1 since service level is a percentage data. Therefore, the range of the predictions for each model is provided in the tables 9.1 and 9.2.

Linear regression is the worst model among other machine learning models for both Erlang-C and Erlang-X. The independent variables and dependent variables do not have linear relationship. This can be a possible reason for not having a well fitted model. The range of the predictions are not compatible with the reality.

Beta regression is the second best model for both Erlang-C and Erlang-X. Both training data sets has 0 and 1 values, therefore, the target values are transformed as it is explained in section 3.6. Futhermore, a tranformation is required for test set to evaluate predictions. There are two possible ways to perform comparison. One can perform transformation on test set target values and compare predictions with transformed test set or perform back transformation on prediction and compare with the original test set target values. By all manner of means, the range of predictions for beta regression is not (0,1) anymore.

SVR is better than linear regression and beta regression, however it cannot outperform random forest and XGBoost of both Erlang-C and Erlang-X. Moreover, SVR cannot predict 0 and 1 values well. There are some prediction values that are less than 0 and greater than 1.

Random forest is the best model among other machine learning models of Erlang-C. It has very low value of MSE and high value of R squared. The prediction range is [0,1]. On the other hand, even though random forest for erlang-x does not outperform XGBoost, it performs well. The range of predictions is subset of the range of target values of test set.

Looking at the performance metrics, it can be concluded that XGBoost is the second best model among 5 machine learning models of Erlang-C, where it is the best performing model of Erlang-X. XGBoost has lower MSE and higher R sqaured value than random forest in erlang-x service level prediction. However, the range of predictions for both Erlang-C and Erlang-X exceed the range of [0,1].

In conclusion, ensemble learning methods such as bagging and boosting outperforms SVR, linear and beta regression models. For the Erlang-C service level prediction problem, random forest is the best performing model by looking at the performance metrics. On the other hand, XGBoost is the best performing model of Erlang-X service level prediction problem. A notable observation is that random forest is the

only algorithm among others that can predict exactly 0 and 1 service level.

An interesting observation is the robustness of the algorithms. The data set is generated with a determined range of features. FC and AHT have the ranges of (0,10]. The models do not know the values higher than 10. The behaviour in extreme situation can be a good interpretation of robustness of the machine learning models. In test set, one row is changed as the following: FC is 18, AHT is 20, patience is 1.4, AWT is 0.33 and number of agents is 78. This setting gives the service level of 0.24. For this scenario of Erlang-X, random forest has the closest prediction with 0.35. The worst prediction of this extreme case is linear regression with -1.31. XGBoost has the prediction of 0.58. We can conclude that for among the Erlang-X service level prediction models random forest is the most robust model. In the same extreme case of Erlang-C, ignoring patience, random forest has the best prediction among other models. The worst prediction of the extreme case is linear regression. Moreover, random forest has better performance on the extreme case than XGBoost. It can be concluded that for Erlang-C and Erlang-X, random forest is more robust than any other algorithm in this research.

Chapter 10

Conclusion

This research focuses on the questions: *How do the assumptions of the multi-skill call center effect the accuracy of performance measures? Can a machine learning algorithm provide a better fit than queueing model?* In this chapter, the research questions will be answered by summarising how assumptions are assessed for a multi-skill call center and summarising machine learning algorithms' performance on predicting performance measures of queuing models.

To assess the effect of the modeling assumptions of call centers on real data, first a data analysis is required. There were two available data sets available for this research. Unfortunately, after analysing the second data set, it was decided to work on the first data set to assess the effect of modeling assumptions on a call center's performance measures. The second data set has unusual patterns for call center properties such as handling time, break time of agents, wrap-up time, and routing policy. Simulating such a unique call center can lead to misleading results.

Models are created to assess the effect of the assumptions on the accuracy of the call center's performance measures. Each model is representative of an assumption. The base model is built as an empirical model. The comparison of models are based on updating the base model. For example, the NHPP arrival assumption did not effect the accuracy of the performance measures; for this conclusion model 1 and 2 are compared. Therefore the base model is updated as model 2. Each model is simulated 1000 times for each day in the data set since call center modeling is a non-stationary system. The weighted average error is calculated for the models. Error is the difference between the expected simulation result and realization. Moreover, due to simulating a day 1000 times, the error caused by the variability of performance measures arises. The models are compared, considering such different performance metrics.

The results can be summarised as in the following: assuming the NHPP of arrival process does not effect the accuracy of predicting performance measures, the assumption of handling time being exponential does not affect the accuracy of predicting performance measures. On the other hand, the assumption of exponentially distributed patience cause significant changes in the accuracy of performance measures. Even though wrap-up time is very short, ignoring wrap-up time in the handling-time calculations leads to inaccuracy of predicting performance measures. Lastly, ignoring break time in the

staffing level calculations leads to the most drastic changes in the performance measures.

Predicting the performance measures of a multi-skill call center by running simulations, is not entirely accurate. Some assumptions are required to depict the reality in the model. Moreover, simulating non-stationary systems brings the variability of performance measures since the simulation should be performed many times. Therefore, simulation results are not entirely accurate. The question arises in this case, is it possible to replace simulation with machine learning? It may be possible, but multi-skill call centers have a very complex structure to model into machine learning algorithms. The question should be divided into the sub-questions. For instance, [32] tries to answer the question, using only staffing level information, can machine learning predict the performance measures of a multi-skill call center? In this research, instead of dividing components of multi-skill call center model such as using only staffing level, I try to predict the service level using all the components of a simple call center model such as Erlang-C and Erlang-X.

To predict the service level of Erlang-C and Erlang-X models, Linear Regression, Beta Regression, Support Vector Regression, Random Forest and XGBoost are performed. Random Forest was the best performing algorithm for predicting Erlang-C service levels. Gradient boosting model, XGBoost, is the best performing algorithm for Erlang-X service level predictions. However, predicting percentage data, such as service levels, introduces some challenges. These predictions must be in the range of [0,1]; otherwise the results are not meaningful nor realistic. Only random forest has predictions in the range of [0,1].

The possible reason that Erlang-X is slightly worse than Erlang-C is patience. Adding patience to the queueing model makes it harder to have low values of service level. Callers who are out of patience, leave the queue. Therefore, next callers in the queue can have service earlier. In this way, service level can increase.

In conclusion, assumptions such as patience being exponentially distributed, wrap-up time is an element of handling time calculation and break time of agents are considered in staffing level, have a significant effect on the accuracy of performance measures. Secondly, Random Forest has very accurate predictions for both Erlang-C and Erlang-X. Moreover, it is the most robust algorithm for predicting extreme cases, among other models used in this research. Therefore, random forest is recommended to use in case of predicting service levels of Erlang-C and Erlang-X models.

Chapter 11

Discussion

This research's goals were checking the effect of assumptions on service level predictions of a multi-skill call center and predicting service level of queuing models. It has been shown that some assumptions have significant effects on the predictions, some of them do not effect the accuracy of predictions. However, during the validation, it was discovered that one of the data sets is problematic in terms of interpretability of components of call center models such as handling time, agents, break time and wrap-up time.

It has been shown that machine learning can predict the service level of queuing models Erlang-C and Erlang-X. However, it was discovered that the range of predictions is not compatible with the reality.

In this research, the first goal is to validate the most common assumptions of call centers. Two data sets were provided by CCMath, namely Vanad and Swiss datasets. Moreover, there were pre-studies about Vanad data before this research. Swiss data is analysed in the same way with Vanad data. However, it was discovered that Swiss data demonstrates unusual patterns. Handling time and wrap-up time has relatively higher values. Agents are not identified uniquely. Moreover, even though they handled calls, some agents had a high percentage of shrinkage during a year. The percentage of shrinkage is shown in Appendix A. These agents may be managers who rarely answer calls. In conclusion, it is decided that simulating such a call center can give misleading results.

The Vanad dataset is used to validate the assumptions of call centers. Pre-studies are extended and the models are simulated. Unlike the Swiss dataset, Vanad data shows different patterns of handling time for experienced and new agents. The learning rate of the agents may help understand more about the fluctuations in handling time. Moreover, individual handling time for each agent could be implemented in the simulation. This could be formulated as another research question.

Furthermore, in Section 3.1, when break time is considered in staffing level calculations, the duration of break time is subtracted from the total working time of that interval and the remaining is divided into the length of the interval. The result is the staffing level of that interval. If the result is a fractional number, it is rounded to the closest integer since the number of agents cannot be a fractional number. This could be changed into a more sophisticated process. For example, a random process can be generated and

according to this process, it could be decided to round-up or round-down the number of agents.

A limitation of this research is having one data set for validation. If another call center data with a similar size with the Vanad call center was available, the validation results would have been strengthened. At this point, the validation results are compatible with Vanad data only; we could not generalise the validation results for the similar size call centers.

Another limitation of this research is the target value being a 'percentage data'. Beta regression offers a prediction range of (0,1); therefore, the model can never predict 0 and 1 values. A transformation method is offered for 0 and 1 values. However, the transformation adds bias to the model.

Moreover, XGBoost has the best performance of Erlang-X service level predictions. The result of the random forest model is slightly different than XGBoost. However, the range of the predictions of XGBoost is out of the range [0,1]. On the other hand, random forest has more realistic predictions. Therefore, even though XGBoost has a slightly better performance in terms of MSE and R squared, random forest is recommended.

From different algorithms, the robustness is compared. Since the feature sets are randomly generated, there are certain ranges for each feature. A feature set with the values out of the range of original feature set is used for the experiment. Among the 5 different machine learning algorithms, random forest gives the closest prediction to the reality for both Erlang-C and Erlang-X. Being more robust does not imply that random forest is the best-performing model. It can be concluded that random forest is the most robust algorithm among the others.

In future work, Random Forest and XGBoost can be implemented for more complex call center modeling. This research focuses on the prediction of a single-skill call center. The next step of this search may be the prediction of the multi-skill call center's performance measures with only two skills.

Bibliography

[1] Gans, N., Koole, G., & Mandelbaum, A. (2003). Telephone call centers: tutorial, review, and research prospects. Manufacturing Service Operations Management, 5(2), 79–141. https://doi.org/10.1287/msom.5.2.79.16071

[2] Koole, G., & Mandelbaum, A. (2002).Queueing Models of Call Centers: An Introduction. Annals of Operations Research 113, 41–59 .

[3] Aksin, Z., Armony, M. & Mehrotra, V.(2007). The Modern Call Center: A Multi-Disciplinary Perspective on Operations Management Research. Production and Operations Management 16(6),pp.665-688.

[4] Mehrotra, Vijay & Fama, J. (2004). Call center simulation modeling: Methods, challenges, and opportunities. Winter Simulation Conference Proceedings. 1. 135 - 143 Vol.1. 10.1109/WSC.2003.1261416.

[5] Zeltyn, S., Mandelbaum, A. (2005). Call Centers with Impatient Customers: Many-Server Asymptotics of the M/M/n + G Queue. Queueing Syst 51, 361–402.

[6] Harris, C.M., Hoffman, K.L. & Saunders, P.B. (1987). Modeling the IRS Telephone Taxpayer Information System. Operations Research 35(4),pp.504-523.

[7] Garnett, O., Mandelbaum, A., & Reiman, M. (2002). Designing a call center with impatient customers. Manufacturing Service Operations Management, 4(3), 208–227. https://doi.org/10.1287/msom.4.3.208.7753

[8] Hampl, P. Analytical Call Center Model with Voice Response Unit and Wrap Up. (2015). Information and Communication Technologies and Services 13(4), 273-279.

[9] Avramidis, A. N. & L'Ecuyer, P., (2005). In Proceedings of the Winter Simulation Conference Orlando.

[10] Gans, N., Liu, N., Mandelbaum, A., Shen, H. & Ye, H. (2010).Service times in call centers: Agent heterogeneity and learning with some operational consequences. Borrowing Strength: Theory Powering Applications – A Festschrift for Lawrence D. Brown, 99–123, Institute of Mathematical Statistics, Beachwood, Ohio, USA, . doi:10.1214/10-IMSCOLL608.

[11] Sze, D. (1984). A Queueing Model for Telephone Operator Staffing. Operations Research, 32(2), 229-249.

[12] Kaplan, E.L. Meier, P. (1958). Nonparametric Estimation from Incomplete Observations.Journal of American Statistical Association. 53(282),pp.457-481.

[13] Mehrotra, Vijay & Fama, J. (2004). Call center simulation modeling: Methods, challenges, and opportunities. Winter Simulation Conference Proceedings. 1. 135 - 143 Vol.1. 10.1109/WSC.2003.1261416.

[14] Brown, L., Gans, N., Mandelbaum, A., Sakov A., Shen, H., Zeltyn S. & Zhao, L. (2005). Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective. Journal of American Statistical Association, 100(469),pp.36-50.

[15] Benoit, K. (2011). Linear Regression Models with Logarithmic Transformations. Methodology Institute, London School of Economics.

[16] Koole, G. (2013). Call Center Optimization. Amsterdam, Netherlands:MG Books, pp 55-74.

[17] Sargent, R. (2011). Verification and validation of simulation models. Engineering Management Review, IEEE. 37. 166 - 183.

[18] Erik, C., Tibor, M., Matej, K. (2011). Erlang c formula and its use in the call centers. Advances in Electrical and Electronic Engineering, 9(1), 7–13. https://doi.org/10.15598/aeee.v9i1.34

[19] Mandelbaum, A. & Zeltyn, S. (2005). The Palm/Erlang-a queue, with applications to call centers.

[20] Allen, R. (1939). The Assumptions of Linear Regression. Economica, 6(22), new series, 191-201. doi:10.2307/2548931

[21] Bottou L. (2010) Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier Y., Saporta G. (eds) Proceedings of COMPSTAT'2010. Physica-Verlag HD

[22] Cribari-Neto, F. & Zeileis, A. (2010). Beta Regression in R. Journal of Statistical Software, 34(2), 1–24. URL http://www.jstatsoft.org/v34/i02/.

[23] Drucker, H., Burges, C.J.C., Kaufman L., Smola, A. & Vapnik, V. (1997). Support Vector Regression Machines. Advances in Neural Information Processing Systems 9, 155-161.

[24] Breiman, L. Random Forests.(2001).Machine Learning, 45(1),5-32.

[25] J. Friedman. (2001).Greedy function approximation: a gradient boosting machine. Annals of Statistics, 29(5):1189–1232.

[26] Chen, T. & Guestrin, C. (2016). XGBoost: a scalable tree boosting system. In Proc.ACM SIGKDD International Conference on Knowledge Discovery and Data Mining

[27] Snoek, J., Larochelle, H., & Adams, R.P. (2012). Practical Bayesian optimization of machine learning algorithms. In Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12), 2 , 2951–2959.

[28] Bergstra, J., Bardenet, R., Bengio Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11).2546–2554.

[29] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M. & Duchesnay E. (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12,2825–2830.

[30] Li, S., Koole, G., & Jouini, O. (2019). A Simple Solution for Optimizing Weekly Agent Scheduling in a Multi-Skill Multi-Channel Contact Center. In Proceedings of Winter Simulation Conference (WSC 2019), pp. 3657-3668. https://doi.org/10.1109/WSC40007.2019.9004714

[31] XGBoost Retrieved: June 10, 2020 from, https://XGBoost.readthedocs.io/en/latest/parameter.html

[32] Li, S., Wang, Q. & Koole, Ger. (2019). Predicting Call Center Performance with Machine Learning: Proceedings of the 2018 INFORMS International Conference on Service Science. 10.1007/978-3-030-04726-9_19.

Appendices

Appendix A

Agent Shrinkage

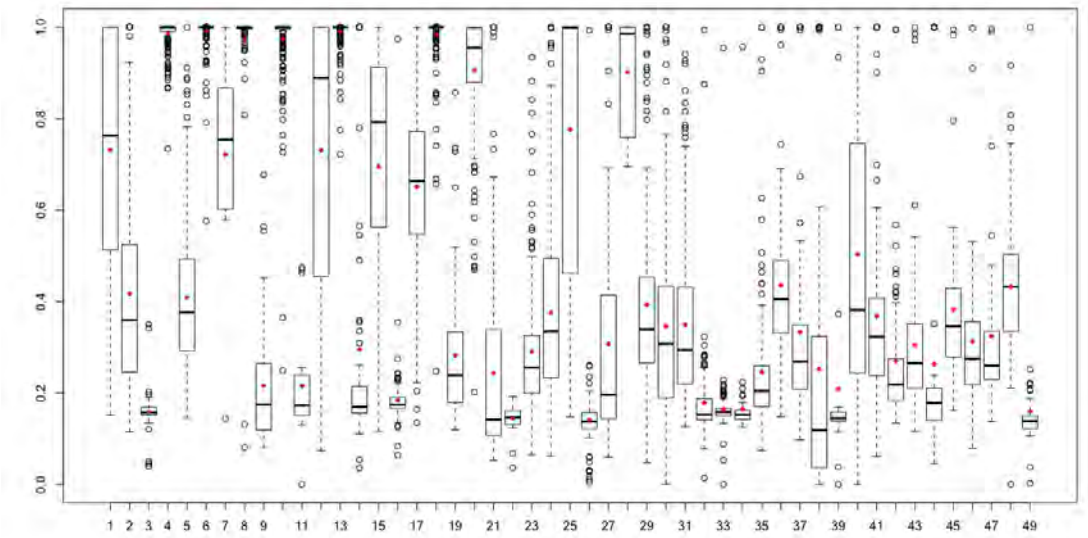


Figure A.1: Percetage of shrinkage of Agents

Appendix B

Erlang Calculators

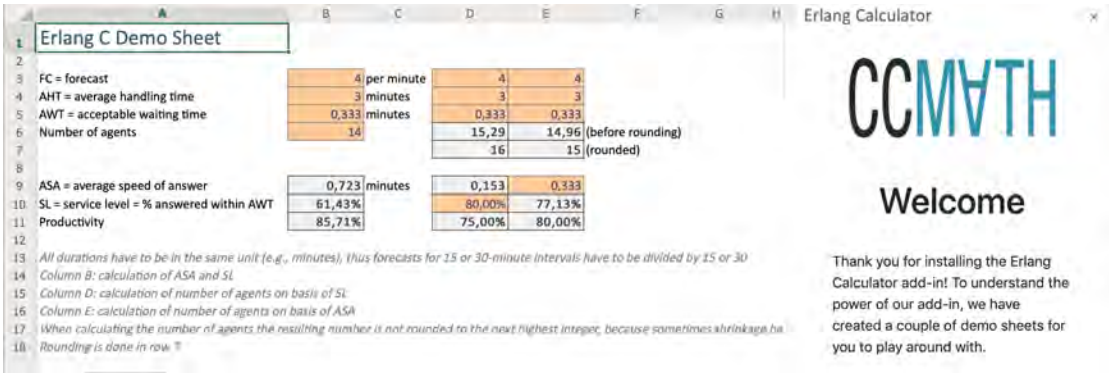


Figure B.1: Erlang C demo sheet

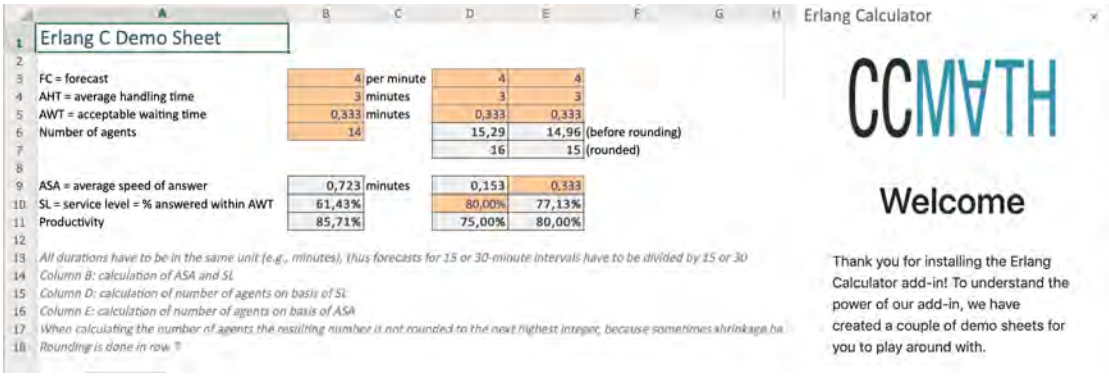


Figure B.2: Erlang X demo sheet

Appendix C

Simulation

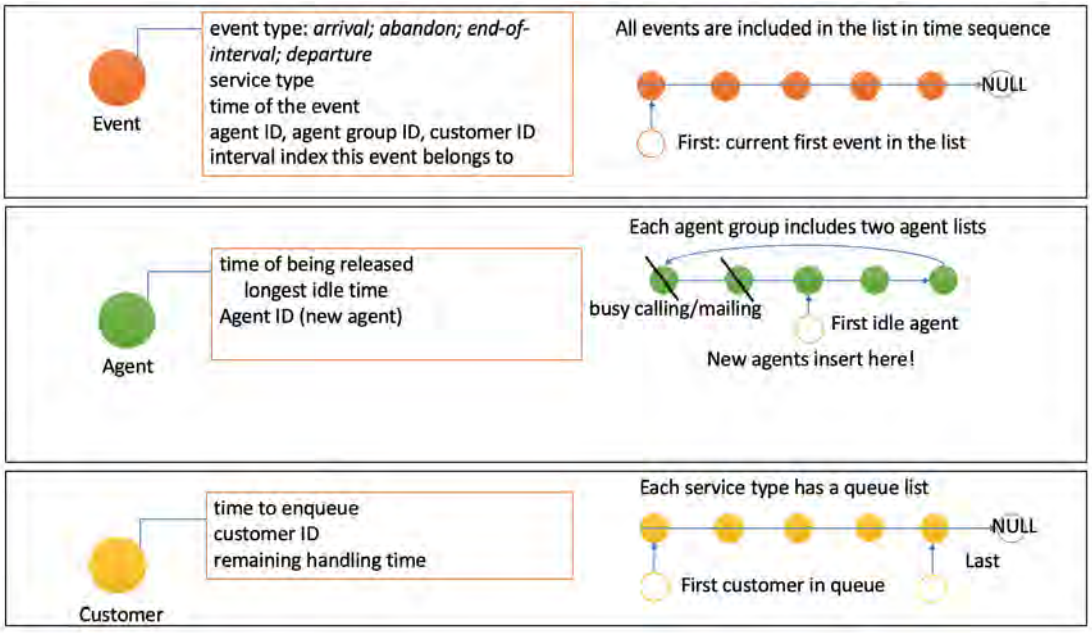


Figure C.1: Simulation