A MACHINE LEARNING BASED

CAPACITY MANAGEMENT SYSTEM FOR MAINFRAME RESOURCES

EKREM KÜRTÜL

BOĞAZİÇİ UNIVERSITY

2022

A MACHINE LEARNING BASED
CAPACITY MANAGEMENT SYSTEM FOR MAINFRAME RESOURCES

Thesis submitted to the

Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information Systems

by

Ekrem Kürtül

Boğaziçi University

2022

DECLARATION OF ORIGINALITY

I, Ekrem Kürtül, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them

Signature……

Date……

ABSTRACT

A Machine Learning Based

Capacity Management System for Mainframe Resources

The goal of this study is to design a capacity planning tool for resource consumption of application servers which are running on mainframes, also known as Z systems, by using machine learning algorithms. This tool is aimed to ensure adequate resources are available in order to meet current and future workload demands. The desired system is intended to have capability to determine and then forecast how much additional capacity will be needed based on increasing demands. In this study, IBM Cloud Pak for Data as a Service is used to create capacity planning model by using data analysis, data engineering, data governance and Artificial Intelligence modeling services which are provided by the platform. The data is prepared outside of the platform and imported to the platform in order to perform analysis and refinement. After the data refinement step is completed, machine learning models are trained by using several algorithms. Then, functional tests are performed in order to check accuracy and performance of the models by using the test interface of the platform. Results of these tests, comments and further research opportunities are also provided. It is observed that the designed capacity planning tool is capable of making consistent predictions with acceptable error rates.

ÖZET

Ana Bilgisayar Kaynakları için
Makine Öğrenimi Tabanlı Kapasite Yönetim Sistemi

Bu çalışmanın amacı, Z sistemleri olarak da bilinen ana bilgisayarlar üzerinde çalışan uygulama sunucularının makine öğrenmesi algoritmaları kullanarak kaynak tüketimi için bir kapasite planlama aracı tasarlamaktır. Bu araç, mevcut ve gelecekteki iş yükü taleplerini karşılamak için yeterli kaynakların mevcut olmasını sağlamayı amaçlamaktadır. Arzu edilen sistemin, artan taleplere göre ne kadar ek kapasiteye ihtiyaç duyulacağını belirleme ve daha sonra tahmin etme yeteneğine sahip olması amaçlanmıştır. Bu çalışmada, platform tarafından sağlanan veri analizi, veri mühendisliği, veri yönetişimi ve Yapay Zeka modelleme hizmetlerini kullanarak kapasite planlama modeli oluşturmak için IBM Cloud Pak for Data as a Service kullanılmıştır. Veriler, platform dışında hazırlanıp analiz ve iyileştirme yapmak için platforma aktarılmıştır. Veri iyileştirme adımı tamamlandıktan sonra, makine öğrenme modelleri çeşitli algoritmalar kullanılarak eğitilmiştir. Ardından, platformun test arayüzü kullanılarak modellerin doğruluğunu ve performansını kontrol etmek için fonksiyonel testler yapılmıştır. Bu testlerin sonuçları, yorumlar ve daha fazla araştırma fırsatları da paylaşılmıştır. Tasarlanan kapasite planlama aracının kabul edilebilir hata oranları ile tutarlı tahminler yapabildiği gözlemlenmiştir.

ACKNOWLEDGEMENTS

First of all, I'd like to thank my thesis advisor, Prof. Meltem Özturan, for allocating her time for this research and sharing her knowledge in order to guide me through this research. Then, I would like to thank my jury members, Prof. Aslı Sencer and Prof. Nuri A Başoğlu for accepting to attend my thesis jury, allocating their time to evaluate my thesis and their valuable feedbacks.

I am grateful to my beloved wife Elif İlke Cebesoy. I got the biggest support from her throughout this study. Without her support and patience, I would not have been able to complete this study.

I would also like to thank my friends, Özcan Gündeş and Alper Solmaz for being with me throughout this thesis process and sharing their valuable feedbacks. Özcan is the one led to start my master journey and guided me through all steps. Alper is the one helped me to develop the idea beyond this research.

My last, biggest, and deepest gratitude is to my family who raised me and brought me to where I am today. It is wonderful to have them. It is wonderful to have everyone in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The business transaction counts increase constantly with respect to the technological developments and public recognition level improvements. As the number of transactions increases, the hardware and software requirements for the servers and the computers are changing as well. One of the most important changing requirements is the capacity needs of these systems.

Capacity means “the ability to hold or contain people or things”, “the largest amount or number that can be held or contained” or “the ability to do something: a mental, emotional, or physical ability” as it is defined in An Encyclopædia Britannica Company Merriam-Webster (*Merriam-Webster*, 2022). Capacity can also be defined as the maximum throughput that a service is able to deliver while meeting service-level objectives over time if capacity management perspective is considered. Capacity planning, on the other hand, can simply be defined as “an estimation of the computer resources required to meet service-level objectives of an application over time.” The primary goal of capacity planning is to maintain a well-balanced computer system that will meet performance goals set by the company. A system is balanced when all its resources are working together to allow the system to handle the maximum amount of workload while meeting certain goals. Capacity planning is used to estimate the number of computing resources required to fulfill the future processing needs of a workload that is currently in production (Hahn et al., 2000).

Capacity planning is an important concept because it may have an effect on client satisfaction, external image of the company, productivity, investments and income and cost levels. These effects can be explained as follows:

- It is possible that there can be some situations like performance problems or high response times in services in systems without any capacity planning so the desired service level agreements (SLAs) may be violated.
- It may take significant time to solve these situations and to re-stabilize the entire system. In the meantime, clients may have to deal with an unstable and unresponsive system. This situation may lead to client dissatisfaction, damage on the external image of the company, decrease in the income levels and increase in the maintenance costs. Moreover, this situation may also affect project schedules and it may even lead to project failures so it may directly affect the productivity (Menascé et al., 1994).

Capacity planning, also known as capacity modeling, techniques range from estimating capacity based on current resource consumption and experience to developing prototypes, full-scale benchmarks, and pilot studies. All of these have advantages and disadvantages, and they are suitable for various applications. All methods of modeling can achieve similar degrees of accuracy, but it is dependent on the information that is used to create it (Great Britain Cabinet Office, 2011). Trending, simulation modeling, and analytical modeling are the three most used capacity modeling techniques (Grummitt, 2009).

Trending, often called trend analysis, is a modeling technique that uses historical data on resource utilization and service performance to predict future behavior. Historical data is usually analyzed in a spreadsheet with graphical, trending, and forecasting tools to indicate resource utilization over time and how it is

likely to change in the future, as well as define expected growth (Great Britain Cabinet Office, 2011; Grummitt, 2009). When there are a few variables and a linear relationship between them, trending is the most effective technique. It is a relatively inexpensive modeling technique, however it just provides estimates of future resource usage, therefore it is not very accurate (Great Britain Cabinet Office, 2011).

Simulation modeling is a technique in which simulation models are created using computer programs that simulate static structure of a system as well as its various dynamic features (Menascé et al., 2004). Simulation modeling employs a traffic model that is compared to a simulation of the configuration until a solution is found (Grummitt, 2009). Simulation modeling provides a comprehensive perspective of present and future operations and can be quite precise for estimating the effects of changes on existing applications or sizing new ones. The disadvantage of this technique is that it takes a long time to create and execute the model, so it is a relatively expensive modeling technique (Great Britain Cabinet Office, 2011).

Analytical modeling uses mathematical tools to represent the behavior of a computer system (Great Britain Cabinet Office, 2011). Analytical models are made up of formulas that specify the traffic and configuration, and algorithms that solve the formulas to produce the answers. Analytical models can be used to analyze present performance and predict future performance (Grummitt, 2009). In order to be mathematically adaptable, analytical models often include a little amount of detail, making them more efficient to operate but less accurate than other modeling techniques (Menascé et al., 2004).

The concept of capacity planning can either be approached in an informal manner or in a very structured and disciplined fashion. The more accurate results can be obtained when the methodology is taken in a more disciplined way (Hahn et al.,

2000). Because of this situation, organizations rely on capacity management tools to manage various types of workloads in the business environment and to help ensuring computer capacity that is utilized to meet business goals in an efficient way. Moreover, these tools are also helpful to meet SLAs consistently. However, these tools usually require a license which is high-priced. Therefore, organizations either have other organizations do their capacity plans or create their own capacity planning tool themselves.

The objective of this study is to design a capacity planning tool for Central Processing Unit (CPU) consumption of application servers which are running on mainframes, also known as Z systems. Z systems have a CPU resource which is measured by Central Processors (CPs). Therefore, there is a limit on the CPs of these systems like storage and memory limitations. The system belongs to one of the greatest banks of Turkey. This tool is aimed to ensure adequate resources are available in order to meet current and future workload demands. The desired system is intended to have capability to determine and then forecast how much additional capacity will be needed based on increasing demands.

In this study, IBM Cloud Pak for Data as a Service is used in order to create capacity planning model by using data analysis, data engineering, data governance and Artificial Intelligence (AI) modeling services which are provided by the platform. The data, which includes all the variables that can be used to perform capacity planning, is prepared outside of the platform. After the preparation, the data is imported to the platform for analysis and refinement. Then, machine learning models are built and deployed by using several algorithms and enhanced training features.

In this chapter, the main purpose of this study has been introduced. Moreover, problem definition, possible solution approaches and contributions made by this study have been explained. To mention other parts of this thesis, Chapter 2 mentions the previous works related to capacity planning. Chapter 3 gives brief explanation about the platform, IBM Cloud Pak for Data as a Service, which is used in this study. Chapter 4 explains data preparation and refinement steps for the data which is used in this study. Chapter 5 shows machine learning model generation details. Chapter 6 presents the details of the conducted tests and the performance results obtained in these tests. Finally, Chapter 7 provides conclusions and further research opportunities to point out the possible improvement areas in this study.

CHAPTER 2

LITERATURE REVIEW

The goal of this study is to design a capacity planning tool for CPU consumption of application servers which are running on Z systems by using machine learning algorithms. In this chapter, previous works in the literature for the capacity planning by using machine learning techniques topic are mentioned.

Capacity planning is a crucial topic and there are several studies on how to approach this topic by using machine learning algorithms.

Le Duc et al., (2019) performed a survey on this subject. They reviewed how the issue of reliable sourcing in joint edge cloud environments has been investigated in the scientific literature. Their main interest was on what methods have been used to increase the reliability of distributed applications in diverse and heterogeneous network environments. They have realized that there has been a significant increase in the number of studies applying machine learning techniques to the characterization and prediction of workload and application behavior, as well as the control of complex distributed applications in recent years. Moreover, they have observed that machine learning techniques yield better results than traditional methods on average. They also discovered that it was even better when dealing with large and complex environment.

In a more technical study, Baldán et al., (2016) built an elastic cloud system which is able to manage CPU usage in an elastic manner. The system has taken the real time monitoring values and forecast values for the following states as input. These forecast values were produced by workload forecasting module which uses

historical data in order to perform forecasting. They employed different forecasting algorithms while building the workload forecasting module such as Auto-Regressive Moving Average (ARIMA), Exponential Smoothing etc. They also used some general regression methods for comparison purposes such as Linear auto-regression (AR), Lasso and Multi-adaptive regression splines (MARS). They evaluated the models by using different error measures such as Symmetric Mean Absolute Percentage Error (SMAPE) and RelMAE, which is the Mean Absolute Error (MAE), normalized by the MAE of a benchmark method. When the forecasts were produced by the workload forecasting module, they were sent to the resource provisioning system in order to make decisions. Then, the decisions were forwarded to the resource management component to take actions. The architecture of the elastic cloud system is provided in Figure 1.

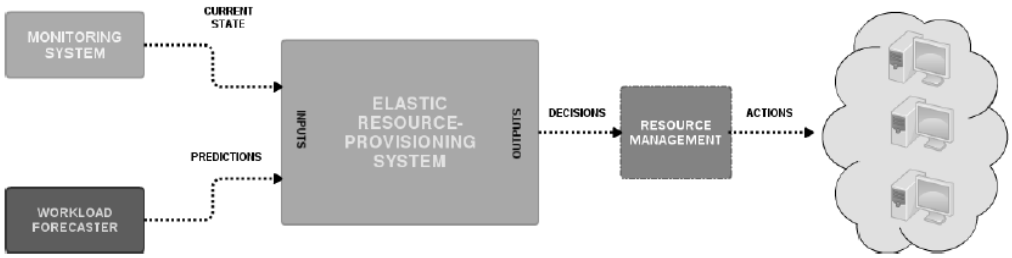

Figure 1. Elastic cloud system (Baldan et al., 2016)

In another study, Le Anh (2016) performed a study to find a solution to increase resource utilization in data centers by using workload prediction for resource management. He focused on predicting CPU core and memory consumption for both short- and long-term resource allocation in the Google cluster trace. He predicted CPU cores and memory consumptions with both short time units of 1, 2 and 5 minutes and longtime unit of 1 hour by using different prediction methods such

as Linear Regression, Adaptive Neuro-Fuzzy Inference Systems (ANFIS) and Nonlinear autoregressive network with exogenous inputs (NARX). He also compared these methods and their results in terms of accuracy and execution time since the usage areas of the predictions might vary depending on the application. If predictions were to be used for elasticity, they must be produced within a few seconds. On the other hand, if they were to be used for scheduling decisions, it might take longer to produce. He used Mean Absolute Percentage Error (MAPE) to compute prediction error of the models since it is easy to use, and it gives reliable results. He performed different experiments in order to compare the performance of the models by using different workloads.

In a different work, Kumar et al., (2021) proposed a novel self-directed workload forecasting method in order to predict the future workload on cloud servers. The proposed framework was able to learn from the past forecasts to improve future predictions. The primary contributions of this study were twofold. Firstly, they introduced forecast error feedback, which allows the model to learn from the last prediction model. The forecast module has received feedback to take advantage of it in the next forecast. The model has caught the error in the last l estimates and calculated the average deviation. Secondly, they developed a population-based meta-heuristic optimization algorithm, i.e., black hole algorithm, for better learning of network weights to get more accurate predictions. They were able to organize the population into multiple clusters or subpopulations with the help of the new learning algorithm. Moreover, unlike the standard algorithm, which only considers global best information to generate new solutions, local and global best information were included in the process of generating new solutions. They have found the incorporation of the local best information beneficial in maintaining

population diversity, which prevents early convergence. They also analyzed the forecast accuracy of the proposed model on different time interval forecasts over multiple real world data traces. They have found that the model is able to reduce the mean squared forecasting errors up to 99.99% over existing models. The prediction model workflow is provided in Figure 2.

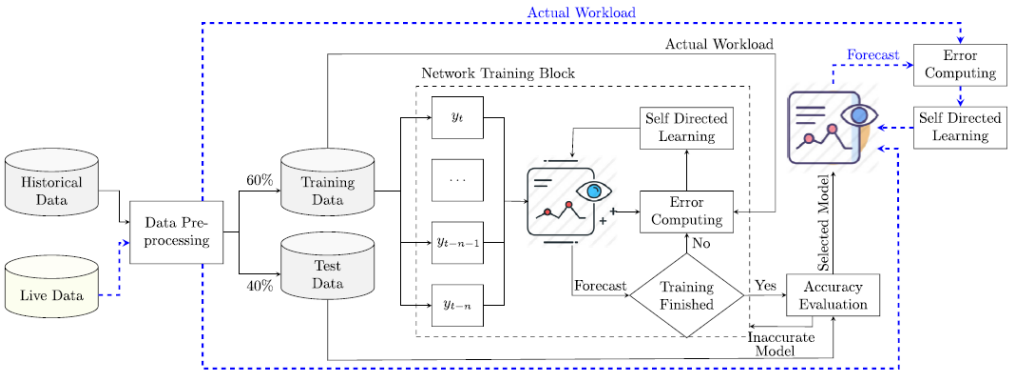

Figure 2. Prediction model workflow (Kumar et al., 2021)

There are also some studies about database workload capacity planning by using different techniques. In one of which, Higginson et al., (2020) applied different forecasting techniques to Online Transaction Processing (OLTP) and Online Analytical Processing (OLAP) workloads which make up the database workload. They aimed to capture key metrics, such as CPU, Input/ Output Operations per Second (IOPS) and Memory, which are applicable to monitoring and capacity planning through an agent. There were some specific commands that were executed on the hosts by the agent in order to retrieve the metric values. Then, these values were stored in a central repository where they were aggregated into hourly values. They executed this process for a period of 30 days in the experiments. Their data collection techniques were quite similar to the techniques used in this study which are explained in Chapter 4. They performed different experiments by using different

machine learning algorithms. The experiments were done hourly, daily, and weekly by using Seasonal Auto-Regressive Moving Average with exogenous factors (SARIMAX) and Holt-Winters Exponential Smoothing (HES) algorithms. In the flow of the experiments, data features such as stationarity, seasonality, multiple seasonality, and shocks were understood, where each model was calculated to obtain a Root Mean Squared Error (RMSE). They have chosen the best model by comparing RMSE values just like done in this thesis in Total CPU Consumption AutoAI Experiment which is explained in Chapter 5.

In another study, Müller et al., (2019) examined the applicability of machine learning techniques to the service capacity management process for commercial off-the-shelf enterprise applications. In order to train performance models for standard business functions, they used real monitoring data from more than 18,000 SAP applications and database samples running on more than 16,000 different servers. In their study, three scenarios were addressed in order to show the utility of machine learning-based techniques using performance counters. The first scenario was server sizing which was about estimation of the capacity demands of the enterprise applications before the deployment. For this reason, appropriate hardware components should be determined for the workload characteristics given in the business layer. Second scenario was about load testing, and it refers to workload changes. The following questions were tried to get an answer in this scenario: “What if the throughput [of a service] doubles?” or “What will be the effect on the response times?”. The last scenario was server consolidation, and which was about the management of existing enterprise applications includes optimizing the allocation of running services to servers on a periodic (offline) or continuous (online) basis. They used the following machine learning algorithms in their models: Support vector

machines (SVM) with Radial basis function (RBF) and with Polynomial kernel (PK), Random forests as representative of Bagging strategy, and AdaBoost as representative of Boosting strategy. They applied the trained models with these algorithms in three capacity management scenarios in order to investigate the utility and applicability.

In a different work, Cortez et al. (2017) introduced a system that collects virtual machine (VM) telemetry, learns from these behaviors, and produces models that can make predictions to various resource managers, so this system has been named as Resource Central (RC). They have provided several use cases for RC. In one of these use cases, RC can be used for predictions of the expected resource utilization of VMs before the selection of servers to run a set of new VMs. This information helps the scheduler to be able to reduce the chance of physical resource exhaustion on oversubscribed servers. In another use case, the service can suggest deployments where VMs that are predicted to be delay-insensitive would be sized tighter than interactive VMs by using RC predictions of workload class and resource utilization. In addition to these, the system can select a cluster which would likely have enough resources with the help of the prediction of maximum deployment size produced by RC, before the selection of a cluster in which to create a VM deployment. In the design of RC, Cortez et al. used different machine learning algorithms. They used Random Forests and Extreme Gradient Boosting Trees as classifiers and Fast Fourier Transform (FFT) to detect periodicity in the utilization time series. They trained the models with two months of data and tested them in the third month of their dataset. They performed experiments by using feature engineering, feature selection, normalization, and regularization techniques in order to improve quality. These types of techniques are also used in this thesis in AutoAI

experiments which are explained in Chapter 5. They have calculated the prediction accuracy of RC in the range of 79% and 90% depending on the metric. The accuracy values were also similar to the results which are provided in Chapter 6.

CHAPTER 3

ABOUT IBM CLOUD PAK FOR DATA

Cloud Pak for Data as a Service is used to perform data analysis, data engineering, data governance and AI modeling operations. It is a platform which offers a cloud native modular service. Users are able to collect, organize and perform some specific operations securely by means of an integrated data fabric to the Cloud Pak for Data as a Service. Data fabric is also provide Cloud Pak for Data as a Service to have a suite of data science and AI tools for performing data analysis and application development with AI in order to improve business outcomes (IBM, 2021c).

The benefits of Cloud Pak for Data as a Service can be listed as follows (IBM, 2021c):

- It is a fully managed cloud service platform which requires no installation, management, or updates
- Scalability
- Security
- Compliance
- Combinable services architecture

3.1 Data fabric

A data fabric is an architectural model to manage highly distributed and utterly different data. It supports the decoupling of data storage, data processing and data use since it is designed for hybrid and multi-cloud data environments. Users can transform data to business assets which are governed globally regardless of the

storage or usage location of the data with the help of intelligent knowledge catalog capabilities. Users are also able to provide business-ready data for their applications and services by means of the automatically assigned catalog assets to the metadata which describes logical connections between data sources and enriches them with semantics. Data fabric architecture provides users to perform data analysis easily and quickly and this is one the reasons why Cloud Pak for Data as a Service is chosen to be used (IBM, 2021c).

Cloud Pak for Data as a Service data fabric architecture enables users to be able to (IBM, 2021c):

- Access to the data in a simple and automated way across multi-cloud and on-premises data sources without moving it.
- Protect the use of all data.
- Have a self-service experience for finding and using data.
- Organize and automate the data lifecycle by using AI-powered capabilities.

There are five main capabilities of the data fabric and Figure 3 shows them and their connectivity between the platform and data source.

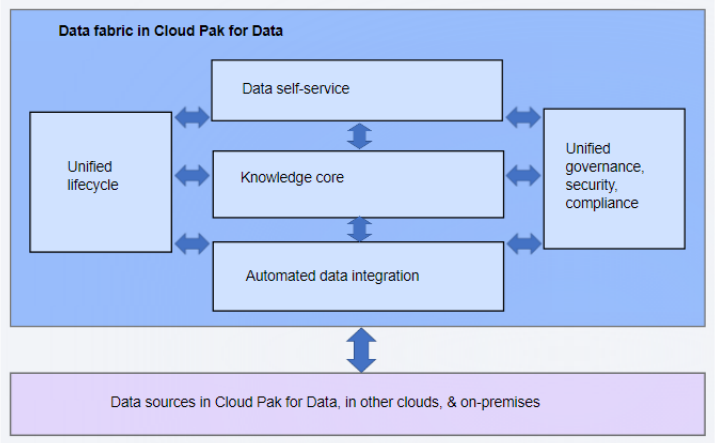

Figure 3. Data fabric in Cloud Pak for Data (IBM, 2021c)

These capabilities and their facilities can be summarized as follows (IBM, 2021c):

- Metadata-based knowledge core enables the exploration of the data sources and catalogs, enriches data assets, and performs analysis to obtain insight for more automation by using AI. It is utilized to control the marketplace with semantic search.
- Self-service data marketplace is the next-generation data catalog which helps data consumers to recover data from across the data landscape of the enterprise.
- Automated data integration is integrated with the knowledge core in order to automate data integration. It has the intelligence to choose which integration approach is best suited based on workloads and data policies. It enables data consumption by extracting, virtualizing, transforming and streaming data.
- Unified governance, security and compliance layer is able to comprehend the data format and data significance and apply the best policies to each bit of data and each prospective user. This capability enables to apply standards and rules to the data at the organizational level and propagate throughout the various data resources as needed.
- Unified lifecycle provides a unified development and operations to arrange and run all the aspects of the data platform in live environment.

3.2 Data science and AI tools of Cloud Pak for Data as a Service

Users are able to participate in finding and sharing insights with the help of the data science and AI tools on Cloud Pak for Data as a Service. By using these tools, users

are able to prepare and train models, deploy models in their applications and evaluate models for performance, quality and bias. These tools include a comprehensive tool set in which users are able to code in Python or R, visually code by creating a flow of steps on a graphical canvas or automatically build a ranked list of model candidates. Users can choose one of these methods depending on their skill levels or individual preferences (IBM, 2021c). In this thesis, the third option has been followed in which a ranked list of model candidates was built in order to focus on the output and the insights.

Data science and AI tools of Cloud Pak for Data as a Service also provides users to promote trained models to deployment spaces, deploy and score the models, review prediction scores and insights, and monitor deployment jobs in a dashboard. After the deployment, users are able to evaluate deployments for bias or drift, update data and retrain deployed models to maintain quality goals. Models are easy to understand by business units and also auditable in business transactions (IBM, 2021c).

3.3 Services architecture of the Cloud Pak for Data as a Service

The architecture of Cloud Pak for Data as a Service includes core services, related services, and a gallery of samples. Figure 4 shows the services architecture of The Cloud Pak for Data as a Service (IBM, 2021c).

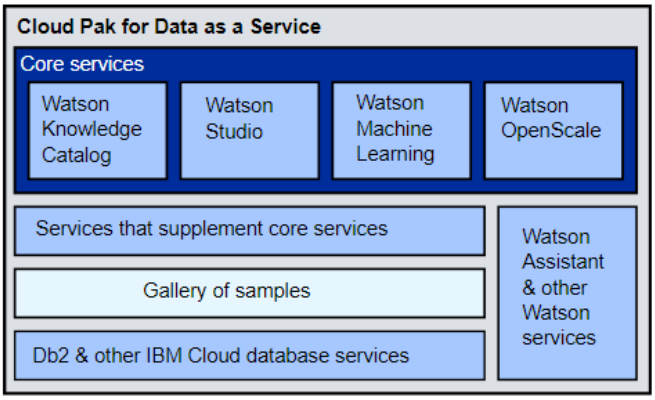

Figure 4. Services architecture of the Cloud Pak for Data as a Service (IBM, 2021c)

Core services are mainly using for analyzing and governing data and running, deploying and evaluating models. The supplementary services of the core services are responsible for adding tools, workspaces, or computation power. IBM Cloud database service is used to store data that users are able to use it in the entire platform. Watson Assistant and other Watson services provides User Interfaces (UIs) or Application Programming Interfaces (APIs) to users for analyzing data. The gallery of samples includes sample data assets, notebooks, and projects. Sample data assets and notebooks provide some samples for data science and machine learning code. Sample projects include a set of assets and detailed instructions on how to solve a particular business problem (IBM, 2021c).

3.4 Functionality in the core services and the common platform

Figure 5 shows the common platform and the core services functionality.

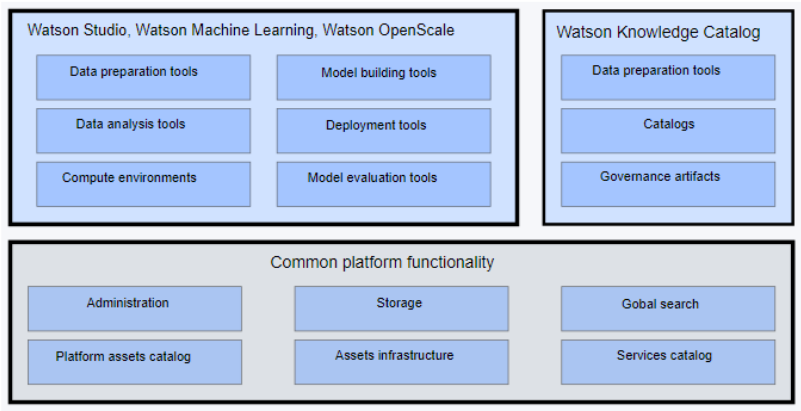

Figure 5. Functionality in the core services and the common platform (IBM, 2021c)

The common platform functionality includes account level administration including user management, storage for projects, catalogs, and deployment spaces in IBM Cloud Object Storage. It also provides common infrastructure for assets, projects, catalogs and deployment spaces (IBM, 2021c).

Watson Studio provides users to choose the tools to analyze and visualize data, to cleanse and shape data and to build machine learning models. Figure 6 shows how the architecture of Watson Studio is centered on the project (IBM, 2021g).

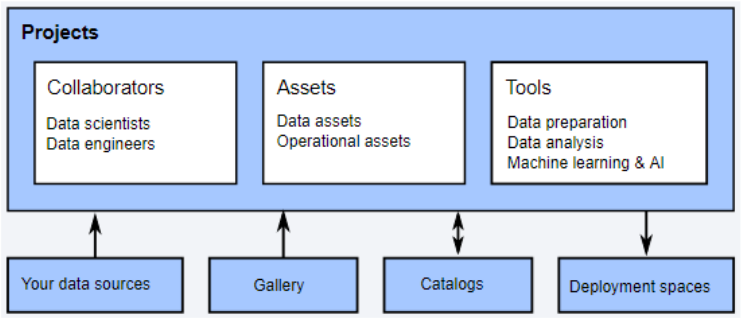

Figure 6. Architecture of Watson Studio (IBM, 2021g)

A project is a workspace where users organize their resources and work with data. Collaborators, assets, and tools are the types of resources in a project.

Collaborators are the members of a team who work with the data in different ways such as data scientist and data engineer. Assets are divided into two groups as data assets and operational assets. Data assets point to data which can be in uploaded files or accessed through connections to data sources. Operational assets, on the other hand, are the objects that users create which run code to work with the data such as models and scripts. Tools are the software that enables users to work with data. Data Refinery, Jupyter notebook editor, RStudio, SPSS modeler and Decision Optimization model builder are included with the Watson Studio service (IBM, 2021g).

Watson Machine Learning provides a vast variety of tools and services so that users are able to build, train, and deploy machine learning models. Users can choose the tool or service with the level of autonomy that matches their needs from a fully automated process to writing their own code. In this thesis, a fully automated process was proceeded in order to focus on the output and the insights. AutoAI experiment builder, Deep Learning experiments, Federated Learning and notebooks are some tools which are available on Watson Machine Learning service. AutoAI experiment builder provides automatically processing structured data to generate model-candidate pipelines. The best-performing pipelines can be saved as a machine learning model and deployed for scoring. Deep Learning experiments automates running hundreds of trainings runs while tracking and storing results. Federated Learning is used to train models using remote or disconnected data sources. Notebooks provide an interactive programming environment in order to work with data, test models and prototype rapidly. There are also other tools which can be used to view and manage model deployments (IBM, 2021f).

Watson Knowledge Catalog is an essential part of the data fabric. It provides a metadata-based knowledge core, data self-service, and unified governance. A secure enterprise catalog management platform is provided by the Watson Knowledge Catalog, and it is supported by a data governance framework. The compliance of the data access with the business rules and standards is ensured by the data governance framework (IBM, 2021e).

In this thesis, these functionalities of the Cloud Pak for Data as a Service have been used where compute usage is measured in capacity unit hours (CUH) in the platform. A capacity unit hour is a specific amount of compute capability with a set cost. There is a lite plan which includes 50 CUH per month and a lite plan has been used for this thesis.

CHAPTER 4

DATA PREPARATION

4.1 Data collection

In this thesis, one of the most important steps is the data preparation. Since the aim is to estimate capacity need of a system, a dataset covering the past few years is needed. There are some historical data in the data warehouse of the bank. Before using this raw data, it is examined and analyzed in order to see whether it is appropriate for the needs or not. After the examination, the variables used in the model are decided.

First of all, the variable which is related with the CPU consumption per day is chosen in order to reflect capacity need of the system and it is named as 'TOTCPUTIME'. Secondly, another variable which shows transaction count per day is used since it is known that CPU consumption is directly related with the transaction count, so the name of the second variable is 'TRANCOUNT'. Then, the variable which is related with the average response time per day is chosen, namely 'AVGRESPTIME', because it is one of the most important metrics of a system. The effect of response time is wanted to see in the model since it may have some drastic changes in busy days although it generally follows a stable trend. Lastly, another variable which stores the date information is used in order to have an opportunity to see historical effect in the model and it is named as 'DATE'. In addition to the raw date information, another variable, namely 'DATE_TYPE', which shows the type of day in terms of weekday (W), holiday (H) and special weekdays (WS), which are the busy days in which workload and the transaction counts generally higher, is also

added. There are some other variables in the raw data which indicates memory usage, IOPS, resource manager statistics etc. However, these variables do not contribute to the prediction of CPU usage, so they are omitted.

The data is prepared outside of the IBM Cloud Pak for Data as a Service. After the preparation, it is imported to the project which is created in the platform such that the data can be analyzed and refined afterwards.

4.2 Data refinement

The Data Refinery tool of IBM Cloud Pak for Data as a Service enables users to cleanse and shape tabular data with a graphical flow editor. Users are also able to use interactive templates to code operations, functions, and logical operators (IBM, 2021d).

A data refinery flow is created in order to cleanse and shape the data. When the data is cleansed, incorrect, incomplete, improperly formatted, or duplicated data is fixed or removed. When the data is shaped, it is customized by changing data types.

When the dataset is imported to IBM Cloud Pak for Data as a Service, the platform recognized data types of all variables as ‘String’. In data refinement step, the data type of ‘TRANCOUNT’ variable is changed to ‘Integer’, ‘AVGRESPTIME’ variable is changed to ‘Decimal’, ‘TOTCPUTIME’ variable is changed to ‘Decimal’ and ‘DATE’ variable is changed to ‘Date’ in an appropriate date format. The data types of the variables are configured as it can be seen in Table 1 as a data sample at the beginning of the data refinement step.

Table 1. Data Types of the Variables in the Data Sample

DATE Date	DATE_TYPE String	TRANCOUNT Integer	AVGRESPTIME Decimal	TOTCPUTIME Decimal
01/01/2020	H	595,244,883	0.0268	957,183.0785
02/01/2020	WS	1,039,277,770	0.0294	2,174,019.2310
03/01/2020	W	1,021,869,535	0.0279	2,169,866.9563
04/01/2020	H	628,199,857	0.0265	994,790.4582
05/01/2020	H	537,185,176	0.0211	764,919.8450
06/01/2020	WS	1,134,440,427	0.0322	2,225,460.0350
07/01/2020	W	1,022,492,232	0.0278	2,079,691.9653
08/01/2020	W	929,120,286	0.0284	1,927,726.2314
09/01/2020	W	909,692,406	0.0280	1,929,968.8540
10/01/2020	W	986,448,225	0.0268	1,986,145.1328

At the data refinement step, some statistics about the data are produced. The dataset includes data from the beginning of 2009 to the end of 2020 i.e., 4382 days of data. There are 578 special weekdays which are labeled as ‘WS’, 1365 holidays which are labeled as ‘H’ and 2440 weekdays which are labeled as ‘W’.

In Table 2, minimum, median, maximum, interquartile range, and standard deviation values of the ‘TRANCOUNT’ variable are provided. This variable represents the transaction counts per day as mentioned previously. The values of the ‘TRANCOUNT’ variable show a dispersed appearance.

Table 2. Statistics of the ‘TRANCOUNT’ Variable

Interquartile Range	375,616,848
Minimum	18,593,644
Maximum	1,383,027,406
Median	342,330,018
Standard Deviation	286,068,559.012

Histogram for ‘TRANCOUNT’ with respect to ‘DATE_TYPE’ is provided in Figure 7. The distribution of transaction counts in different date types can be followed from this table. Transaction counts of holidays are generally smaller than weekdays or special weekdays as expected. Therefore, the intensity of ‘H’ values is higher at the left of the graph. Similarly, the intensity of ‘WS’ values are higher at the right of the graph since transaction counts of special weekdays are generally greater than holidays or weekdays as expected.

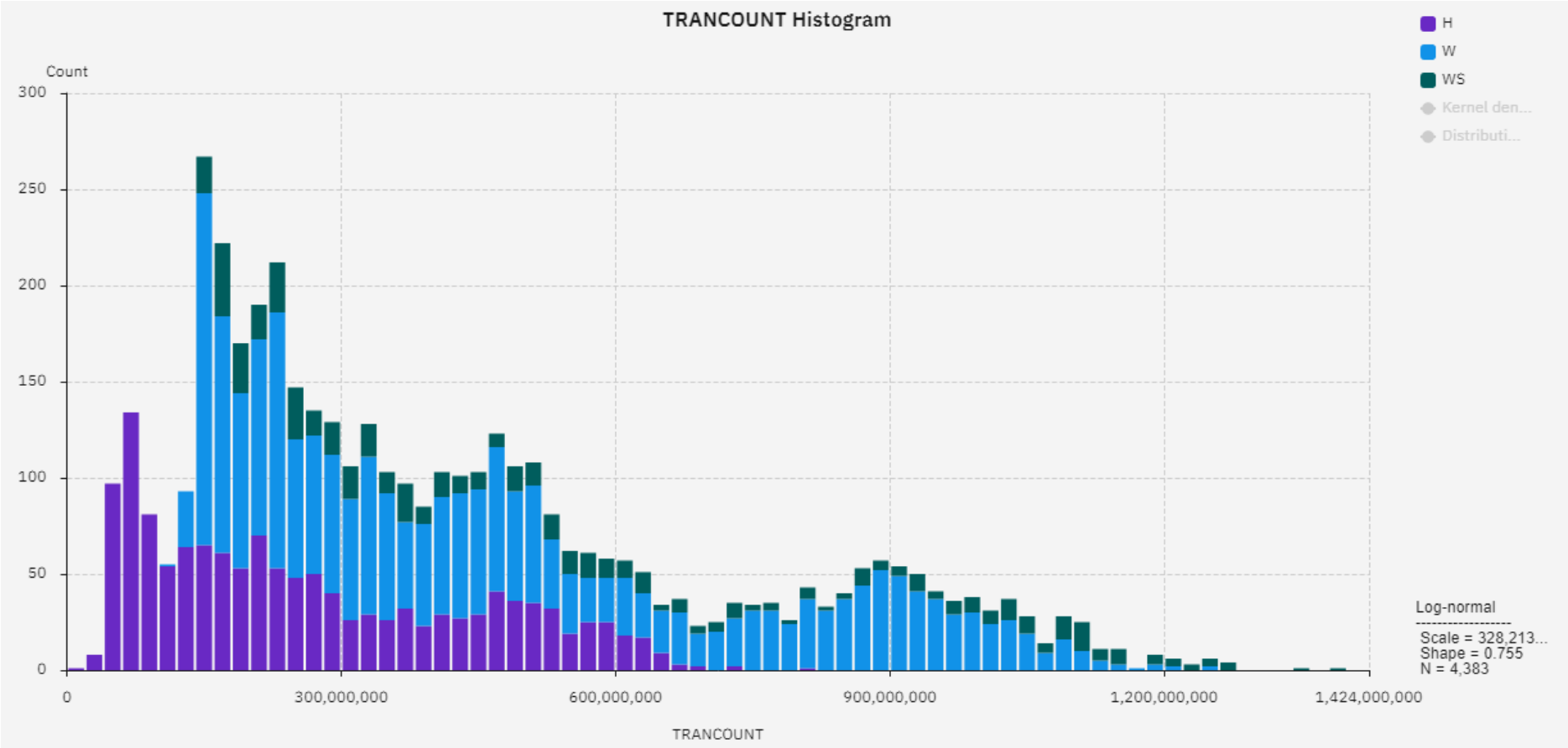

Figure 7. Histogram for 'TRANCOUNT' with respect to 'DATE_TYPE'

In Table 3, minimum, median, maximum, interquartile range, and standard deviation values of the ‘AVGRESPTIME’ variable are provided. This variable represents the average response time values, in seconds, of all transactions which run in a day. It is expected that average response time values generally follow a stable trend and the findings in Table 3 satisfies this expectation. ‘AVGRESPTIME’ variable values have shown intensity between certain values, which are 0.0162s and 0.139s.

Table 3. Statistics of the ‘AVGRESPTIME’ Variable

Interquartile Range	0.0188
Minimum	0.0162
Maximum	1.2413
Median	0.0288
Standard Deviation	0.0298

In Table 4, minimum, median, maximum, interquartile range, and standard deviation values of the ‘TOTCPUTIME’ variable are provided. This variable represents the total CPU consumption per day, in seconds, as mentioned previously. The values of this variable show a dispersed appearance as it does for ‘TRANCOUNT’ variable as expected.

Table 4. Statistics of the ‘TOTCPUTIME’ Variable

Interquartile Range	750,937.092
Minimum	4209.824
Maximum	2,606,006.326
Median	812,492.163
Standard Deviation	560,566.863

Histogram for ‘TOTCPUTIME’ with respect to ‘DATE_TYPE’ is provided in Figure 8. The distribution of total CPU time in different date types can be followed from this figure. This figure follows similar trend with the histogram for ‘TRANCOUNT’ as expected since CPU consumption is directly related with the transaction count. Total CPU consumption in holidays is generally smaller than weekdays or special weekdays as expected. Therefore, the intensity of ‘H’ values is higher at the left of the graph. Similarly, the intensity of ‘WS’ values are higher at the right of the graph since total CPU consumption special weekdays are generally greater than holidays or weekdays as expected.

When the data refinement step is completed, the data is ready to be used in order to build, train, and deploy Machine Learning models.

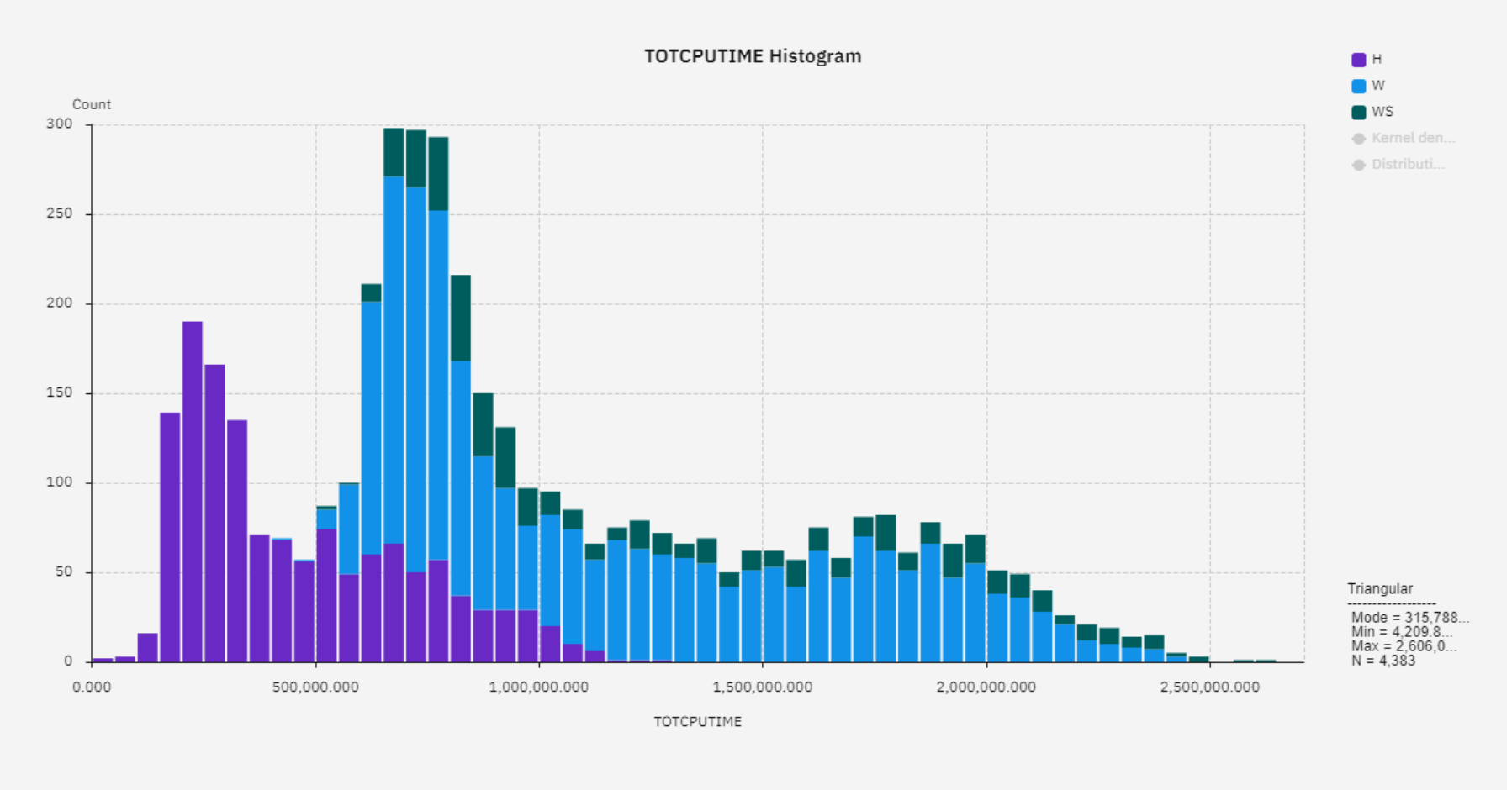

Figure 8. Histogram for 'TOTCPUTIME' with respect to 'DATE_TYPE

CHAPTER 5

ANALYZING DATA AND BUILDING MODELS

In this thesis, the AutoAI graphical tool of Watson Studio is used as mentioned previously. AutoAI graphical tool enables analysis of the data automatically and generation of candidate model pipelines which are customized for the predictive modeling problems. Machine learning models are built and deployed with enhanced training features without coding. By this means, it is aimed to focus more on the output and the insights. The tool performs most of the work for users. The workflow diagram of AutoAI tool is provided Figure 9.

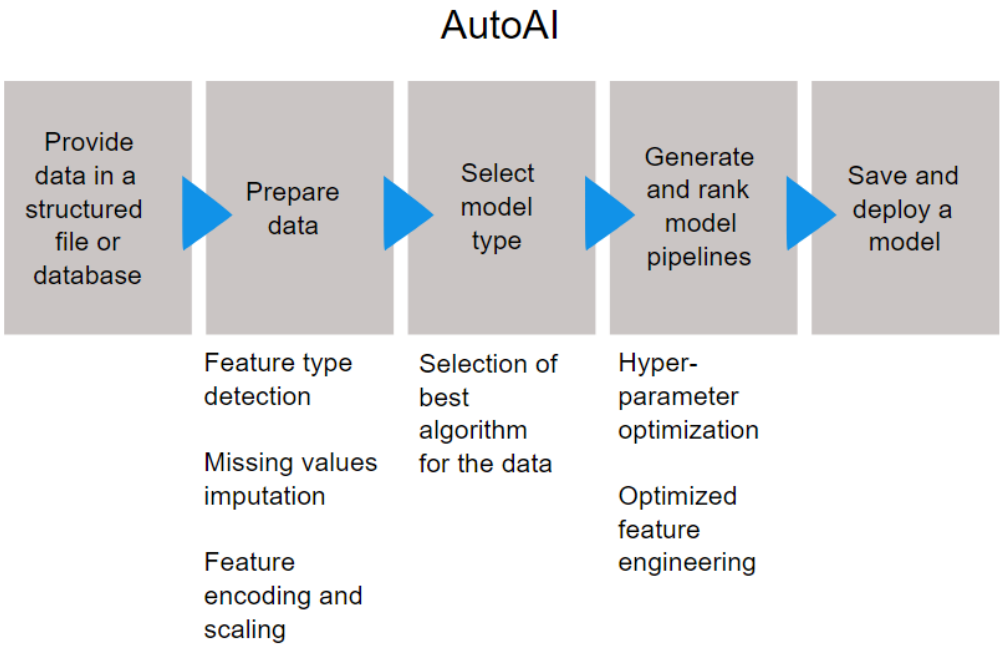

Figure 9. The workflow diagram of AutoAI tool (IBM, 2021b)

In this workflow, AutoAI automatically runs data pre-processing, automated model selection, automated feature engineering and hyper-parameter optimization tasks to build and evaluate candidate model pipelines. These model pipelines are

created iteratively when AutoAI analyzes the dataset and discovers the data transformations, algorithms, and parameter settings best suited to the problem identification (IBM, 2021b).

In data pre-processing step, AutoAI applies various algorithms in order to analyze, clean and prepare the raw data for machine learning. The data which is used in this thesis has no missing or incorrect values since a data refinement is performed and these kinds of values are fixed or removed. However, AutoAI performs another analysis on data in order to guarantee the best results from the machine learning algorithms since they work with numbers and no missing values. AutoAI automatically detects and categorizes features based on data type, such as categorical or numerical. Then, it employs hyper-parameter optimization to determine the best combination of strategies for missing value imputation, feature encoding, and feature scaling for the data, depending on the categorization (IBM, 2021b).

In automated model selection step, AutoAI utilizes a clever methodology that empowers testing and ranking candidate algorithms against small subsets of the data, gradually increasing the size of the subset for the most encouraging algorithms to show up at the best match. This methodology saves time without any performance issues. It provides ranking several candidate algorithms and choosing the best match for the data (IBM, 2021b).

Feature engineering endeavors to transform the raw data into the combination of features which best represents the problem to achieve the most accurate prediction. In automated feature engineering step, AutoAI utilizes an extraordinary methodology that investigates different feature development decisions in a structured, non-exhaustive way, while continuously maximizing model accuracy using reinforcement learning. This results in an optimized sequence of

transformations for the data that best match the algorithms of the model selection step (IBM, 2021b).

As a final step, a hyper-parameter optimization step refines the best performing model pipelines. AutoAI utilizes a novel hyper-parameter optimization algorithm enhanced for expensive function evaluations such as model training and scoring that are common in machine learning. This methodology empowers quick combination to a decent arrangement notwithstanding long evaluation times of each iteration (IBM, 2021b).

In this thesis, two different machine learning models are trained in order to predict capacity needs of the system. In the first model, it is aimed to predict transaction count by using a time series forecast algorithm. In this first model, a dataset which contains only date and transaction count information i.e., 'DATE' and 'TRANCOUNT' variables, is used. In this sub dataset, 'DATE' variable is also divided into 2 groups in such a way that first group contains only weekdays and special weekdays, and the second group contains only holidays. The reasons beyond this choice are to prevent deviations in forecast values because there is a significant difference in transaction counts between weekdays and holidays, to get more accurate forecast results and to focus more on the capacity needs in weekdays and special weekdays.

In the second and the actual model, it is aimed to predict total CPU consumption by using a regression algorithm. In this model, the actual dataset that contains all the variables, which are 'DATE', 'DATE_TYPE', 'TRANCOUNT', 'AVGRESPTIME' and 'TOTCPUTIME', is used. When both of these models are trained, it is aimed to give the output of the first model as an input for the

‘TRANCOUNT’ variable of the second model. By this means, it is wanted to predict two unknowns at the same time and produce more realistic results.

5.1 Configurations of Transaction Count Forecasting AutoAI Experiment

A time series forecast algorithm is used in order to predict future transaction counts. For this reason, the AutoAI experiment is configured in such a way that the prediction type is appropriate for this model after the sub dataset which contains only date and transaction count information, i.e., ‘DATE’ and ‘TRANCOUNT’ variables, is imported. ‘DATE’ variable is the independent and ‘TRANCOUNT’ is the dependent variable for this experiment. Therefore, in this AutoAI experiment, the prediction variable is ‘TRANCOUNT’. There are four available metrics for time series forecast prediction type in order to optimize the experiment, namely Symmetric Mean Absolute Percentage Error (SMAPE), R^2, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The recommended metric, which is SMAPE, is chosen for this model because SMAPE is one of the most widely used metrics for evaluating the performance of the forecast model. Mean Absolute Percentage Error (MAPE) is asymmetric version of SMAPE, and it is more sensitive to negative errors, when forecast values are higher than the actual values, than the positive ones because of the fact that percentage error is not able to go beyond 100% for forecasts which are too low while there is no upper limit for the forecasts which are too high. Therefore, MAPE is most useful for models which perform under-forecast rather than over-forecast. SMAPE, on the other hand, overcomes this asymmetry and it has both the lower (0%) and the upper (200%) bounds.

There are seven different algorithms for time series forecast prediction type. These algorithms are ARIMA, Box-Cox Transformation, Autoregressive moving

average residuals (ARMA), Trend and Seasonality (BATS), Ensembler, Holt-Winters, Linear Regression, Random Forest and SVM. Brief explanations of these algorithms are provided as follows:

- ARIMA: Autoregressive integrated moving average is a statistical analysis model which uses time series data in order to comprehend a data set or to forecast future trends. ARIMA is a type of regression analysis that gauges the strength of a dependent variable relative to other varying variables. The goal of ARIMA is to forecast future trends by analyzing the differences between the values in the series rather than the actual values. There are three components that make up ARIMA. Auto regression (AR) component refers to a model that represents a changing variable that regresses relative to its lagging or previous values. This component is represented by parameter *p* in the ARIMA function as the number of lag observations in the model. Integrated (I) component represents the difference of raw observations to allow the time series to become stationary. This component is represented by parameter *d* in the ARIMA function as the number of times that the raw observations are differenced. Lastly, moving average (MA) component includes the dependence between an observation and a residual error from a moving average model applied to lagged observations. This component is represented by parameter *q* in the ARIMA function as the size of the moving average window (Hayes, 2021).
- BATS: It is an algorithm consisting of a combination of Exponential Smoothing Method, Box-Cox Transformation and ARMA model for residuals. In BATS, Box-Cox Transformation deals with the non-linear data and ARMA model for residuals can be correlated with the time series data (De Livera, 2010; De Livera et al., 2011).

- Ensembler: An ensemble is made up of two or more forecasts that attempt to realize possible uncertainties in a numerical forecast (Cheung, 2001). A well designed, reliable ensemble prediction system should aim to represent random errors in trends. This can be accomplished by utilizing alternative numerical and physical formulations in each integration, by including a stochastic part intended to address the difference between alternative, physically reasonable representations of a given process, or by doing both. There are four main approaches in ensemble prediction to represent model uncertainties. In multimodel approach, different models are used in each ensemble members. In perturbed parameter approach, all ensemble integrations are prepared with the same model but with different parameters that define the settings of the model components. In perturbed tendency approach, stochastic schemes designed to simulate the random model error component are used to simulate the fact that tendencies are known only approximately. In stochastic backscatter approach, for the processes that the model is not able to resolve, a Stochastic Kinetic Energy Backscatter (SKEB) scheme is used (Robertson & Frédéric, 2019).
- Holt-Winters: It is an algorithm which is used to forecast future trends. This algorithm is used to smooth a time series data and make predictions by using that data. Exponential smoothing assigns exponentially decreasing weights and values against historical data in order to reduce the value of the weight for the older data. There are three types of exponential smoothing methods used in Holt-Winters. Single Exponential Smoothing is used to forecast data without trend or seasonal patterns. Double Exponential Smoothing is used to forecast data with a trend. Lastly, Triple Exponential Smoothing is used to forecast data with trend and/or seasonality (Smarten, 2018).

- Linear Regression: In this algorithm, the dependent variable is continuous, but the independent variable can be continuous or discrete. The regression line is in the form of linear. In linear regression algorithm, a relationship is created between the dependent variable (*y*) and independent variable or variables (*x*) by using the best fit straight line. This relationship is defined by the following equations:

$$y = a + bx + e$$

$$y = a + b_1x_1 + b_2x_2 + \cdots + b_nx_n + e$$

  In these equations, *a* represents the intercept, *b* represents the slope of the line and *e* represents the error term (Sarker, 2021).

- Random Forest: A random forest is a classifier which includes a set of decision trees. In this set, each tree is built by applying an algorithm on a training set and an additional random vector. This random vector is sampled independently and identically distributed from some distribution. The prediction of the random forest is acquired by majority vote over the predictions of the individual trees (Shalev-Shwartz & Ben-David, 2014).

- SVM: Support Vector Machine builds a hyper-plane or set of hyper-planes in high or infinite dimensional space. Intuitively, the hyperplane with the largest distance from the nearest training data points in any given class achieves strong separation because in general the larger the margin, the lower the generalization error of the classifier. This algorithm is effective in high dimensional spaces and its behavior may change depending on different mathematical functions known as kernels (Sarker, 2021).

All the available algorithms are selected in order to be considered when the experiment is run. AutoAI experiment is configured to select top three performing

pipelines to complete because more pipelines increase the runtime and use more resources.

Since the dataset is sufficiently large and there are enough records to cover all possible cases, the experiment is adjusted to use 80% of the data as training data and 20% of the data as holdout data in order to optimize and validate pipelines. The reason beyond this splitting operation is to estimate the performance of the machine learning model on the new data.

Furthermore, the number of hyper-parameter optimization iterations to apply to pipelines after model selection per algorithm is set to 10, number of feature engineering iterations per algorithm is set to 30 and number of hyper-parameter optimization iterations after feature engineering per algorithm is set to 25 as runtime settings. In this experiment, Watson Machine Learning service of IBM Cloud Pak for Data as a Service is used, and the runtime is configured to have 8 CPU and 32GB RAM and to consume 20 capacity units per hour.

Each flow in the Transaction Count Forecasting AutoAI Experiment is started with reading the dataset. Then dataset splitting operation takes place followed by reading the training data. Afterwards, lookback window is generated before the pipeline selection. Then the model starts to run with all the available algorithms that are chosen in the experiment configuration. AutoAI selects top three performing pipelines to complete at the pipeline evaluation step since it is configured to do so. At the final step, back testing is performed with the holdout data after these pipelines are generated.

The results of AutoAI experiment are provided in Table 5. It shows the automatically generated model pipelines ranked according to the problem optimization objective which is SMAPE. SMAPE is an accuracy measure based on

percentage errors. Percentage error is nothing but the absolute error as a percentage of the average of the forecast and the actual values. The lower the SMAPE value of a prediction, the higher its accuracy. The pipeline which uses the Ensembler algorithm performs the best result based on SMAPE value, which is 5.487% in back test, and it is a quite good error for this AutoAI experiment. This pipeline is selected by AutoAI, and it is now ready to be deployed as a machine learning model.

Table 5. Pipeline Leaderboard of Transaction Count Forecasting AutoAI Experiment

Rank	Name	Algorithm	SMAPE	Enhancements	Build Time
1	Pipeline 5	Ensembler	5.487%	HPO, FE	00:00:01
2	Pipeline 3	Linear Regression	6.009%	HPO, FE	00:00:02
3	Pipeline 2	SVM	6.219%	HPO, FE	00:00:03

Pipelines other than the top three were discarded in favor of better performing pipelines even if they were considered during the experiment. The list of the discarded pipelines is provided in Table 6.

Table 6. Table of Discarded Pipelines

Name	Algorithm	Projected SMAPE	Training Data Used
Pipeline 1	Random Forest	6.385%	79%
Pipeline 10	BATS	7.710%	79%
Pipeline 4	Ensembler	7.205%	79%
Pipeline 6	Ensembler	7.127%	79%
Pipeline 7	Holt-Winters	7.423%	79%
Pipeline 8	Holt-Winters	7.443%	79%
Pipeline 9	ARIMA	7.291%	79%

In Figure 10, actual and predicted values for the ‘TRANCOUNT’ variable is visualized for a period of 4.5 months.

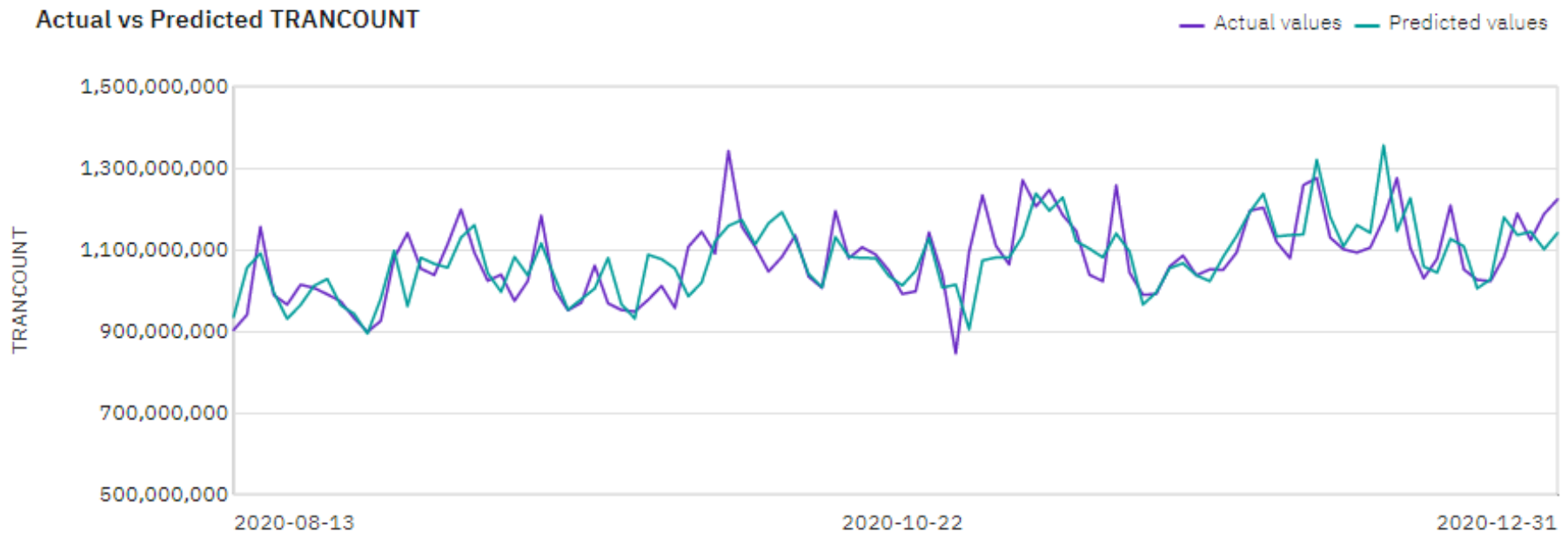

Figure 10. Predictions over time for 'TRANCOUNT' variable

5.2 Configurations of Total CPU Consumption AutoAI Experiment

A regression algorithm is used in order to predict future CPU consumptions. For this reason, the AutoAI experiment is configured in such a way that the prediction type is appropriate for this model after the actual dataset which contains all the variables namely ‘DATE’, ‘DATE_TYPE’, ‘TRANCOUNT’, ‘AVGRESPTIME’ and ‘TOTCPUTIME’ is imported. ‘DATE’, ‘DATE_TYPE’, ‘TRANCOUNT’ and ‘AVGRESPTIME’ variables are the independent variables and ‘TOTCPUTIME’ is the dependent variable for this regression experiment. Therefore, in this AutoAI experiment, the prediction variable is ‘TOTCPUTIME’. There are eight available metrics for regression prediction type in order to optimize the experiment, namely Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), Root Mean Squared Log Error (RMSLE), Mean Squared Log Error (MSLE), Explained Variance and R^2. The recommended metric, which is RMSE, is chosen for this model because RMSE is one of the most widely used metrics for evaluating the performance of the model. RMSE values can be calculated by using the following formula:

$RMSE_{fo} = \left[\sum_{i=1}^{N} \frac{(z_{fi} - z_{oi})^2}{N}\right]^{1/2}$, where f represents expected values or unknown results and o represents observed values or known results (Barnston, 1992).

The formula is put into words, RMSE values are being calculated by taking square of the difference between estimated and corresponding observed values and then taking the average over the sample. Then, the square root of the average gives the RMSE value. The RMSE is more sensitive to the large errors since the errors are squared before they are averaged. Therefore, RMSE is most useful when these types of errors are particularly undesirable, and it is not wanted the model which is created in this thesis to have significant errors.

The algorithm selection is also optimized so that AutoAI selects the algorithms with the highest score in the shortest run time. The reason beyond this selection is that it is aimed the model to have high performance and low response time. There are 11 different algorithms for regression prediction type. These algorithms are Decision Tree Regressor, Extra Trees Regressor, Gradient Boosting Regressor, Light Gradient Boosting Machine (LGBM) Regressor, Linear Regression, Random Forest Regressor, Ridge, Snap Boosting Machine Regressor, Snap Decision Tree Regressor, Snap Random Forest Regressor and Extreme Gradient Boosting (XGB) Regressor. Brief explanations of these algorithms are provided as follows:

- Decision Tree Regressor: Decision tree algorithms construct regression or classification models in the form of a tree structure. It separates a dataset into smaller subsets while simultaneously an associated decision tree is incrementally developed. At the end, a tree which has decision nodes and leaf nodes is built. Decision nodes have two or more branches, and these branches represent values for the tested attribute. Leaf nodes represent a decision on the numerical target. Furthermore, root node is the topmost decision node in a tree which represents the best predictor. There is a top down, greedy search through the space of possible branches with no backtracking at the core algorithm for building decision trees. Decision tree algorithms supports both categorical and numerical data (Rathore & Kumar, 2016).
- Extra Trees Regressor: It is an algorithm which creates several unpruned decision trees and makes predictions by averaging the predictions of the decision trees (Geurts et al., 2006).

- Gradient Boosting Regressor: It is an ensemble learning algorithm which builds a final model based on a set of individual models, typically decision trees. The gradient is used to minimize the loss function (Sarker, 2021).
- LGBM Regressor: Light Gradient Boosting Machine is a gradient boosting framework that uses tree based learning algorithm (IBM, 2021a). In LGBM, the tree grows vertically while it grows horizontally in other tree-based algorithms. In other words, the tree grows leaf-wise in LGBM, and it grows level-wise in the others. LGBM will choose the leaf in such a way that it has maximum delta loss to grow. Leaf-wise algorithm can reduce more loss than level-wise algorithm while growing the same leaf. LGBM is a high-speed algorithm, so it is prefixed as Light GBM. LGBM can deal with huge size of data, and it takes lower memory to run. Moreover, LGBM focuses on the accuracy of results and it also supports GPU learning (Ke et al., 2017).
- Linear Regression: In this algorithm, the dependent variable is continuous, but the independent variable can be continuous or discrete. The regression line is in the form of linear. In linear regression algorithm, a relationship is created between the dependent variable (*y*) and independent variable or variables (*x*) by using the best fit straight line. This relationship is defined by the following equations:

$$y = a + bx + e$$

$$y = a + b_1x_1 + b_2x_2 + \cdots + b_nx_n + e$$

In these equations, *a* represents the intercept, *b* represents the slope of the line and *e* represents the error term (Sarker, 2021).

- Random Forest Regressor: It is an algorithm which builds multiple decision trees in order to produce the mean prediction of each decision tree. This algorithm supports both categorical and continuous variables (IBM, 2021a).
- Ridge: It is an algorithm which is used to analyze datasets which have multicollinearity, i.e., the predictors that are correlated with other predictors. This algorithm performs L2 regularization which is the squared magnitude of coefficients. Therefore, Ridge regression forces the weights to be small, but never sets the coefficient value to zero, making it a non-sparse solution (Sarker, 2021).
- Snap Boosting Machine Regressor: It is an algorithm which provides a boosting machine that uses the IBM Snap ML library which can be used to construct an ensemble of decision trees (IBM, 2021a).
- Snap Decision Tree Regressor: It is an algorithm which provides a decision tree that uses the IBM Snap ML library (IBM, 2021a).
- Snap Random Forest Regressor: It is an algorithm which provides a random forest that uses the IBM Snap ML library (IBM, 2021a).
- XGB Regressor: Extreme Gradient Boosting is a form of gradient boosting algorithm. It considers more detailed approximations while deciding the best model. By calculating second-order gradients of the loss function, it minimizes loss and advanced regularization, which reduces overfitting and improves performance and model generalization. XGB is a high speed algorithm and it can deal with huge size of data (Sarker, 2021).

All the available algorithms are selected in order to be considered when the experiment is run. AutoAI experiment is configured to select top two performing

algorithms to complete. It is limited to two because each algorithm generates four pipelines, and more pipelines increase the runtime and use more resources.

Since the dataset is sufficiently large and there are enough records to cover all possible cases, the experiment is adjusted to use 90% of the data as training data and 10% of the data as holdout data in order to optimize and validate pipelines. The reason beyond this splitting operation is to estimate the performance of the machine learning model on the new data.

Furthermore, the number of hyper-parameter optimization iterations to apply to pipelines after model selection per algorithm is set to 10, the number of feature engineering iterations per algorithm is set to 30 and the number of hyper-parameter optimization iterations after feature engineering per algorithm is set to 25 as runtime settings. The text feature engineering is also enabled in order to transform columns detected as text into vectors to better analyze semantic similarity between strings. In this experiment, Watson Machine Learning service of IBM Cloud Pak for Data as a Service is used, and the runtime is configured to have 8 CPU and 32GB RAM and to consume 20 capacity units per hour.

Each flow in the Total CPU Consumption AutoAI Experiment is started with reading the dataset. Then dataset splitting operation takes place followed by reading the training data. Afterwards, preprocessing operation is performed before the model selection. Then the model starts to run with all the available algorithms that is chosen in the experiment configuration. AutoAI selects two best algorithms to create four pipelines for each since it is configured to do so. Finally, optimization and feature engineering operations are performed for different pipelines in order to find the best pipeline to proceed.

The results of AutoAI experiment are displayed on Table 7. It shows the automatically generated model pipelines ranked according to the problem optimization objective which is RMSE. AutoAI selects two best algorithms as LGBM Regressor and XGB Regressor. This situation is reasonable since the dataset is quite large, i.e., it includes data from the beginning of 2009 to the end of 2020, and both of these two algorithms can deal with huge size of data effectively.

The RMSE value of the best pipeline is 77,374.265 and it is a quite good value even if it seems a big number. RMSE can get all the values in the interval [0, ∞). The lower the RMSE is better, but it changes with respect to the dataset. RMSE is nothing but the standard deviation of the prediction errors and the prediction errors are a measure of how far from the regression line data points are. Therefore, RMSE is a measure of how spread out these prediction errors are. In this AutoAI experiment, the prediction variable is 'TOTCPUTIME', and the value of this variable varies between 500,000 and 2,606,000, so RMSE value of the best pipeline is reasonable and acceptable. The pipeline which uses the LGBM Regressor algorithm and enhanced by hyper-parameter optimization (HPO-1) and feature engineering (FE) performs the best result based on RMSE value. Pipelines labeled by 'Pipeline 3' and 'Pipeline 4' have the same RMSE value. AutoAI selects the 'Pipeline 3' as the best performing pipeline since its build time is smaller. 'Pipeline 3' is now ready to be deployed as a machine learning model.

Table 7. Pipeline Leaderboard of Total CPU Consumption AutoAI Experiment

Rank	Name	Algorithm	RMSE	Enhancements	Build Time
1	Pipeline 3	LGBM	77,374.265	HPO-1, FE	00:00:27
2	Pipeline 4	LGBM	77,374.265	HPO-1, FE, HPO-2	00:01:16
3	Pipeline 1	LGBM	78,002.703	None	00:00:01
4	Pipeline 2	LGBM	78,002.703	HPO-1	00:00:32
5	Pipeline 7	XGB	79,950.126	HPO-1, FE	00:00:30
6	Pipeline 8	XGB	79,950.126	HPO-1, FE, HPO-2	00:00:48
7	Pipeline 5	XGB	80,191.844	None	00:00:01
8	Pipeline 6	XGB	80,191.844	HPO-1	00:00:15

CHAPTER 6

TESTS AND RESULTS

In this thesis, two different machine learning models are trained and deployed in order to predict capacity needs of the system by using AutoAI graphical tool of Watson Studio. In the first model, transaction counts are predicted by using a time series forecast algorithm. Therefore, in order to perform tests, there is only 1 input variable for this model, and it is ‘DATE’ variable. In the second and the actual model, total CPU consumptions are predicted by using a regression algorithm. Therefore, in order to perform tests, there are 4 input variables for this model which are ‘DATE’, ‘DATE_TYPE’, ‘AVGRESPTIME’ and ‘TRANCOUNT’. For ‘TRANCOUNT’ variable, the output of the first model is given as the input for the second model during the test phase. By this means, it is aimed to predict two unknowns at the same time and to produce more realistic results.

The models are trained by using a dataset which includes data from the beginning of 2009 to the end of 2020. Then, some tests are performed by using 2021 data in order to see the performance and accuracy of the models. IBM Cloud Pak for Data as a Service provides a test interface in order to perform performance and accuracy tests. Users can provide input data either as a list or as in JSON (JavaScript Object Notation) format.

Different tests are performed by using the test interface of the platform. Firstly, it is aimed to see the performance and accuracy of the transaction count forecasting model for the first 102 weekdays of 2021. This test is performed by using data for only the weekdays and the special weekdays because there are drastic changes between the weekdays and holidays in terms of both transaction counts and CPU consumption values. Another reason is that the transaction count forecasting model is trained by using data for only the weekdays and the special weekdays in order to focus more on the capacity needs in those days. The test data is prepared in JSON format to make predictions. The results of this test are provided in Figure 11. The prediction error of future transaction counts is in the range of (-11.63%, +16.75%). Actually, most of the errors are in the range of (-10%, +10%). The reason beyond this situation is that there are some extraordinary days in which the workload and transaction counts take unusual values, so the predictions become less accurate. The reason why error percentages take negative and positive values is that the predictions are sometimes more than the actual values and sometimes less. In other words, there are over predictions and under predictions, so the error values are reasonable and acceptable.

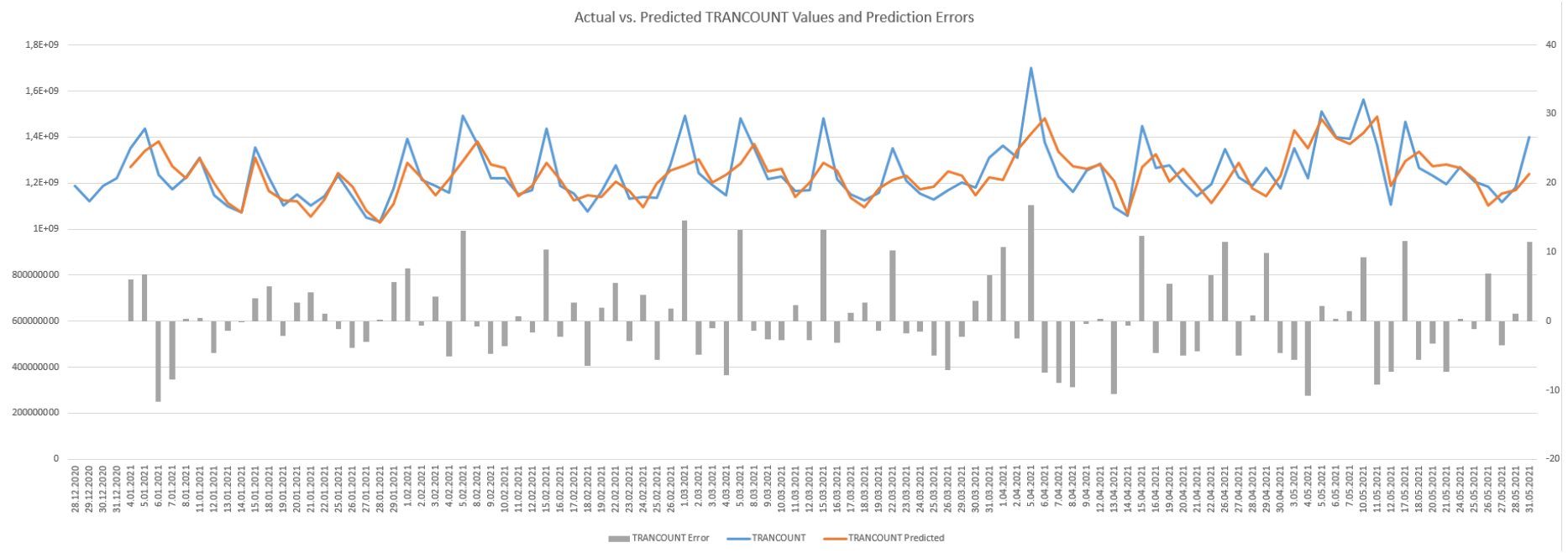

Figure 11. Actual vs. predicted transaction count values and prediction errors

Secondly, it is aimed to see the performance and accuracy of the total CPU consumption model for the first 102 weekdays of 2021. This test is performed by using data for only the weekdays and the special weekdays because there are drastic changes between the weekdays and holidays in terms of both transaction counts and CPU consumption values. Another reason is that the transaction count forecasting model is trained by using data for only the weekdays and the special weekdays in order to focus more on the capacity needs in those days. The test data is prepared in JSON format in such a way that the output of the first model is given as the input for the transaction count variable of this model. The results of this test are provided in Figure 12. The prediction error of future total CPU consumption values is in the range of (-18.32%, +18.65%). Actually, most of the errors are in the range of (-10%, +10%). The reason beyond this situation is that there are some extraordinary days in which transaction counts and total CPU consumptions take unusual values, so the predictions become less accurate. The reason why error percentages take negative and positive values is that the predictions are sometimes more than the actual values and sometimes less. In other words, there are over predictions and under predictions, so the error values are reasonable and acceptable.

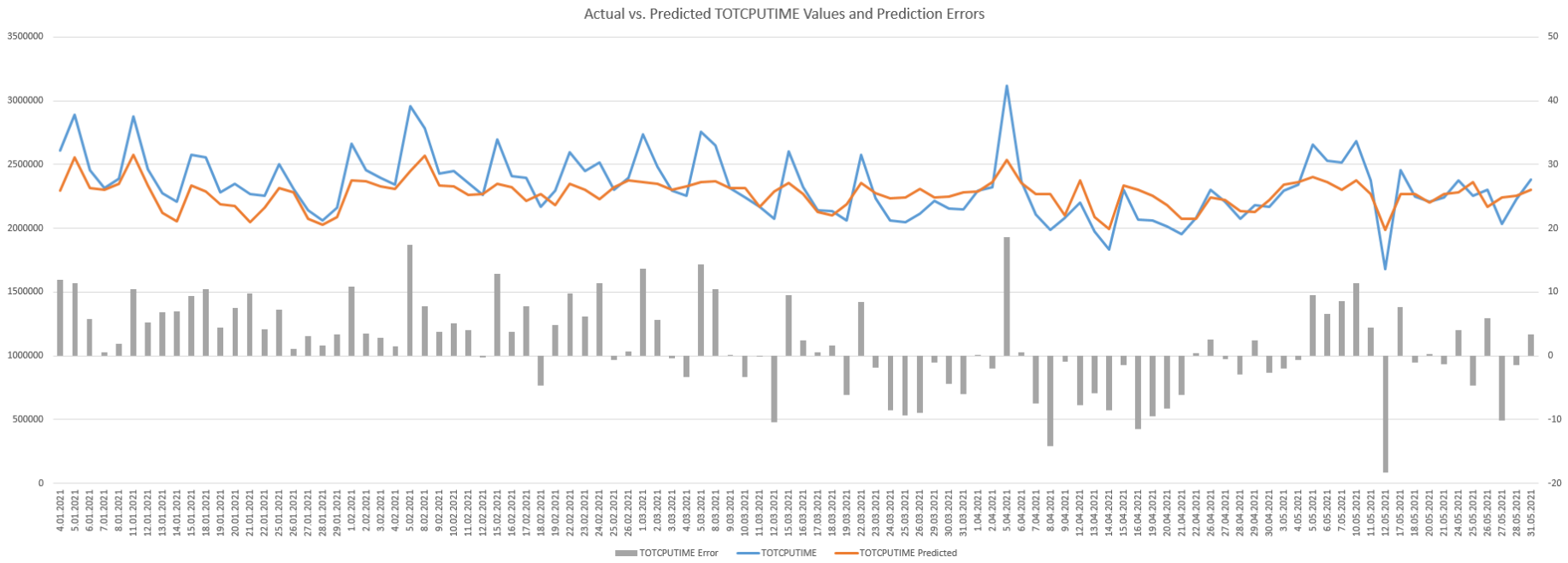

Figure 12. Actual vs. predicted total CPU consumption values and prediction errors

Lastly, another test is performed in order to check the effect of the "AVGRESPTIME" and "DATE_TYPE" variables. It is aimed to see the effect of response time in the model since it is examined that it may have some drastic changes in busy days although it generally follows a stable trend. It is also aimed to see the effect of type of day in the model because the workload and the transaction counts generally higher on the special weekdays than the ordinary weekdays. In order to observe these effects, a few days are selected randomly and changed their 'DATE_TYPE' and 'AVGRESPTIME' values respectively. It is expected that the model to perform more CPU estimation on special weekdays or on days in which the average response time is higher. The test results have met this expectation. In Table 8, test results for three different days are provided.

Table 8. Test Results for the Effects of "AVGRESPTIME" and "DATE_TYPE" Variables

DATE	DATE_TYPE	TRANCOUNT	AVGRESPTIME (seconds)	TOTCPUTIME (seconds)	TOTCPUTIME Predicted (seconds)	TOTCPUTIME Error
06/01/2021	WS	1,215,987,230	0.0319s	2,214,869.035s	2,139,865.698s	-3.386%
06/01/2021	W	1,215,987,230	0.0319s	2,214,869.035s	2,096,532.785s	-5.343%
06/01/2021	WS	1,215,987,230	0.0255s	2,214,869.035s	2,106,308.965s	-4.901%
07/01/2021	W	1,031,672,230	0.0262s	2,084,582.963s	2,195,265.256s	5.309%
07/01/2021	WS	1,031,672,230	0.0262s	2,084,582.963s	2,268,745.784s	8.834%
07/01/2021	W	1,031,672,230	0.0325s	2,084,582.963s	2,221,693.596s	6.577%
08/01/2021	W	938,080,135	0.0274s	1,937,815.232s	1,862,365.458s	-3.894%
08/01/2021	H	938,080,135	0.0274s	1,937,815.232s	1,625,478.632s	-16.118%

For the day dated '06/01/2021' which is a special weekday, so it is labeled by 'WS'. There are 1,215,987,230 transaction counts on this day with an average response time of 0.0319s and total CPU consumption is 2,214,869.035s. The model is predicted total CPU consumption as 2,139,865.698s with a -3.386% error. Then, the 'DATE_TYPE' variable is changed to 'W' to label the day as an ordinary weekday in order to see the effect of type of day. In this case, the model is predicted

total CPU consumption as 2,096,532.785s with a -5.343% error. The new prediction is lower than the actual one and it is the expected result. Afterwards, the average response time is decreased to 0.0255s in order to see the effect of response time. In this case, the model is predicted total CPU consumption as 2,106,308.965s with a -4.901% error. The new prediction is lower than the actual one and it is the expected result.

On the other hand, for the day dated '07/01/2021' which is an ordinary weekday, so it is labeled by 'W'. There are 1,031,672,230 transaction counts on this day with an average response time of 0.0262s and total CPU consumption is 2,084,582.963s. The model is predicted total CPU consumption as 2,195,265.256s with a 5.309% error. Then, the 'DATE_TYPE' variable is changed to 'WS' to label the day as a special weekday in order to see the effect of type of day. In this case, the model is predicted total CPU consumption as 2,268,745.784s with an 8.834% error. The new prediction is higher than the actual one and it is the expected result. Afterwards, the average response time is increased to 0.0325s in order to see the effect of response time. In this case, the model is predicted total CPU consumption as 2,221,693.596s with a 6.577% error. The new prediction is higher than the actual one and it is the expected result.

Furthermore, for the day dated '08/01/2021' which is an ordinary weekday, so it is labeled by 'W'. There are 938,080,135 transaction counts on this day with an average response time of 0.0274s and total CPU consumption is 1,937,815.232s. The model is predicted total CPU consumption as 1,862,365.458s with a -3.894% error. Then, the 'DATE_TYPE' variable is changed to 'H' to label the day as a holiday in order to see the effect of type of day. In this case, the model is predicted total CPU

consumption as 1,625,478.632s with a -16.118% error. The new prediction is significantly lower than the actual one and it is the expected result.

The test results give an opportunity to observe performance and accuracy of the models and they also give a chance to make decisions whether the models are reliable or not. When the test results are examined, it can be said that the models are quite reliable since the error percentages are not high and the results are expected when changes take place.

CHAPTER 7

CONCLUSIONS AND FURTHER RESEARCH

This thesis aims to design a capacity planning tool for CPU consumption of application servers which are running on Z systems by using machine learning algorithms. In this thesis, IBM Cloud Pak for Data as a Service is used in order to create capacity planning model by using data analysis, data engineering, data governance and AI modeling services which are provided by the platform. The dataset includes date, type of date, transaction count per day, average response time of the application servers per day and total CPU consumptions of the application servers per day variables. The dataset covers data from the beginning of 2009 to the end of 2020. The data is prepared outside of the platform and imported to the platform in order to perform analysis and refinement. After the data refinement step is completed, several machine learning model pipelines are built and trained and AutoAI selected best pipelines according to the selection criteria. Then the selected pipelines are deployed as a machine learning models.

In this thesis, two different machine learning models are trained in order to predict capacity needs of the system. In the first model, it is aimed to predict transaction count by using a time series forecast algorithm. In this model, the pipeline which uses the Ensembler algorithm performed the best result based on SMAPE value, which was 4.661%. In the second and the actual model, it is aimed to predict total CPU consumption by using a regression algorithm. The output of the first model is given as an input for the second model in order to predict two unknowns at the same time and to produce more realistic results. In this model, the

pipeline which uses the LGBM Regressor algorithm performed the best result based on RMSE value, which was 77,374.265 and it was a quite good value even if it seemed a big number. In this thesis, one optimization metric is used for each model. In the future study, different optimization metrics can be used together to make comparisons between the algorithms and the models. This will give an opportunity to have more precise results.

Some accuracy and performance tests are also performed by using the test interface of the platform. In the first test, it is aimed to see the performance and accuracy of the transaction count forecasting model for the first 102 weekdays of 2021. The prediction error of future transaction counts was in the range of (-11.63%, +16.75%). Secondly, another test is performed to see the performance and accuracy of the total CPU consumption model for the first 102 weekdays of 2021. The prediction error of future total CPU consumption values was in the range of (-18.32%, +18.65%). Actually, most of these two errors were in the range of (-10%, +10%). The reason beyond this situation was that there were some extraordinary days in which transaction counts and total CPU consumptions took unusual values, so the predictions became less accurate. In the final test, it is aimed to see the effect of the average response time and type of date variables. It is aimed to see the effect of response time in the model since it is examined that it may have some drastic changes in busy days although it generally follows a stable trend. It is also aimed to see the effect of type of day in the model because the workload and the transaction counts generally higher on the special weekdays than the ordinary weekdays. These two effects are observed at the end of the test and the results are met the expectations. The test results give an opportunity to observe performance and

accuracy of the models and they also give a chance to say that these models are reliable. Therefore, the results obtained in this thesis are promising.

In this thesis, a capacity planning tool for CPU consumption of application servers which are running on Z systems is designed. The tool is designed to make predictions for the total workload of the bank. There are several channels of the banks which create the workload such as mobile banking, branches, Point of Sale (POS) transactions etc. Every channel has its own characteristics and workload. For example, the usage of mobile banking channel increases significantly with the widespread use of mobile devices. Therefore, workload of the mobile banking channel increases as well, and it brings a necessity to monitor this channel closely. In the future study, workload separation based on channels can be considered in order to make predictions for a specific channel. Resource and capacity management can be performed by using these predictions and so by following the customer trends. When predictions are performed on a channel basis, proactive capacity management techniques can be applied easily.

REFERENCES

Baldan, F. J., Ramirez-Gallego, S., Bergmeir, C., Herrera, F., & Benitez, J. M. (2016). A forecasting methodology for workload forecasting in cloud systems. *IEEE Transactions on Cloud Computing*, *6*(4), 929–941. https://doi.org/10.1109/TCC.2016.2586064

Barnston, A. G. (1992). Correspondence among the correlation, RMSE, and Heidke forecast verification measures; refinement of the Heidke score. *Climate Analysis Center*, 699–709.

Cheung, K. K. W. (2001). A review of ensemble forecasting techniques with a focus on tropical cyclone forecasting. *Meteorological Applications*, *8*(3), 315–332. https://doi.org/10.1017/S1350482701003073

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., & Bianchini, R. (2017). *Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms*. https://doi.org/10.1145/3132747.3132772

De Livera, A. M. (2010). *Automatic forecasting with a modified exponential smoothing state space framework* (Vol. 10, Issue April). http://www.buseco.monash.edu.au/depts/ebs/pubs/wpapers

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, *106*(496), 1513–1527. https://doi.org/10.1198/jasa.2011.tm09771

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Great Britain Cabinet Office. (2011). ITIL Service Design. In *The Stationery Office*.

Grummitt, A. (2009). *Capacity management - A practitioner guide* (1st ed.). Van Haran Publishing.

Hahn, S., Jackson, M. H. A., Kabath, B., Kamel, A., Meyers, C., Matias, A. R., Osterhoudt, M., & Robinson, G. (2000). *Capacity planning for business intelligence applications: Approaches and methodologies*.

Hayes, A. (2021). *Autoregressive Integrated Moving Average (ARIMA)*. https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

Higginson, A. S., Dediu, M., Arsene, O., Paton, N. W., & Embury, S. M. (2020). Database workload capacity planning using time series analysis and machine learning. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 769–783. https://doi.org/10.1145/3318464.3386140

IBM. (2021a). *AutoAI implementation details*. https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-details.html?context=cpdaas&audience=wdp

IBM. (2021b). *AutoAI Overview*. https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-overview.html?context=cpdaas&audience=wdp

IBM. (2021c). *Overview of Cloud Pak for Data as a Service*. https://dataplatform.cloud.ibm.com/docs/content/wsj/getting-started/overview-cpdaas.html?context=cpdaas&audience=wdp

IBM. (2021d). *Refining data*. https://dataplatform.cloud.ibm.com/docs/content/wsj/refinery/refining_data.html?context=cpdaas&audience=wdp

IBM. (2021e). *Watson Knowledge Catalog on Cloud Pak for Data as a Service*. https://dataplatform.cloud.ibm.com/docs/content/svc-welcome/wkc.html?context=cpdaas&audience=wdp

IBM. (2021f). *Watson Machine Learning on Cloud Pak for Data as a Service*. https://dataplatform.cloud.ibm.com/docs/content/svc-welcome/wml.html?context=cpdaas&audience=wdp

IBM. (2021g). *Watson Studio on Cloud Pak for Data as a Service*. https://dataplatform.cloud.ibm.com/docs/content/svc-welcome/wsl.html?context=cpdaas&audience=wdp

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 3147–3155.

Kumar, J., Singh, A. K., & Buyya, R. (2021). Self directed learning based workload forecasting model for cloud resource management. *Information Sciences*, *543*, 345–366. https://doi.org/10.1016/j.ins.2020.07.012

Le Anh, T. (2016). *Workload prediction for resource management in data centers*.

Le Duc, T., Leiva, R. G., Casari, P., & Östberg, P. O. (2019). Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. *ACM Computing Surveys*, *52*(5), 39. https://doi.org/10.1145/3341145

Menascé, D. A., Almeida, V. A. F., & Dowdy, L. W. (1994). *Capacity planning and performance monitoring from mainframes to client-server systems*. Prentice Hall PTR.

Menascé, D. A., Almeida, V. A. F., & Dowdy, L. W. (2004). *Performance by design: Computer capacity planning by example*. Prentice Hall PTR.

Merriam-Webster (2022). In *merriam-webster dictionary*. Retrieved January 2022, from https://www.merriam-webster.com/dictionary/capacity

Müller, H., Bosse, S., & Turowski, K. (2019). On the utility of machine learning for service capacity management of enterprise applications. *Proceedings - 15th International Conference on Signal Image Technology and Internet Based Systems, SISITS 2019*, 274–281. https://doi.org/10.1109/SITIS.2019.00053

Rathore, S. S., & Kumar, S. (2016). A decision tree regression based approach for the number of software faults prediction. *ACM SIGSOFT Software Engineering Notes*, *41*(1), 1–6. https://doi.org/10.1145/2853073.2853083

Robertson, A. W., & Frédéric, V. (2019). *Sub-seasonal to seasonal prediction the gap between weather and climate forecasting*. Candice Janco. https://doi.org/https://doi.org/10.1016/C2016-0-01594-2

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, *2*(3), 1–23. https://doi.org/10.1007/s42979-021-00592-x

Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. In *Cambridge University Press* (Vol. 9781107057). https://doi.org/10.1017/CBO9781107298019

Smarten. (2018). *What is the Holt-Winters Forecasting Algorithm and How Can it be Used for Enterprise Analysis?* https://www.elegantjbi.com/blog/what-is-the-holt-winters-forecasting-algorithm-and-how-can-it-be-used-for-enterprise-analysis.htm