Industrial Engineering Department
INDR 491 Industrial Engineering Design Project

Fall 2023

Design of an Analytical Framework for Data Center Efficient Energy Management

19.01.2023

Team Members:

Umur Berkay Karakaş
Sena Karadağ
Bora Aktaş
Serra Işık
Batuhan Gül

Academic Supervisor:

Fikri Karaesmen

Industrial Supervisor: Aylin Yorulmaz

Industrial Engineering Department
INDR 491 Industrial Engineering Design Project

Table of Contents

1. Brief system description

KoçSistem, founded in 1945 with its headquarters in Istanbul, is a prominent company within the Koç Bilgi Grubu AS group. Initially known as Koç-Unisys, KoçSistem has evolved over the years, expanding its offerings in the information technology sector. The company provides a variety of IT services, including network solutions.

Today, KoçSistem's range of services has grown to include IT consulting and software product development, serving clients not just in Turkey but in over thirty countries, including notable presences in Azerbaijan, Malaysia, England, and China.

The company's offerings encompass a broad range of IT solutions. These include Internet of Things (IoT) applications, analytics for better data understanding, business solutions for data centers, and cloud services for data management and storage.

KoçSistem operates a data center in Üsküdar, Istanbul, which features two floors with 12 different rooms housing server racks. The KS10 room in this data center is equipped with various servers, four coolers, three chillers, and two temperature sensors that record data every five minutes. The system manages heat through cool air inflow and chiller-supplied cold water. Although fan speed is typically automated in data centers, for this project it is assumed to be manually controlled. In this project, while coolers traditionally operate based on predetermined set parameters to perform the cooling process, a key focus will be on optimizing these set parameters. The aim is to establish an automated system that adjusts itself based on the optimized parameters, thereby enhancing the efficiency and effectiveness of the cooling process. The data center's layout is unique, with some rooms dedicated to individual servers and others accommodating multiple server racks. Interestingly, the building housing the data center was originally constructed as a hospital.

2. Literature review and sector analysis

There have been various approaches to making data centers more efficient using both predictive and optimization tools. Prediction-based approaches mainly focus on predicting future efficiency metric values, most importantly Power Usage Efficiency (PUE), based on historical data provided by the physical sensors in the server room. Optimization-based approaches include linear and dynamic programming on both hardware and software aspects of data center frameworks. PUE, which will be explained in detail below, is a metric used to determine the energy efficiency of a data center. It's a simple ratio that compares the total amount of energy used by the data center to the energy consumed by the computing equipment within it.

2.1. Prediction-based Approaches

The first ground-breaking predictive method for data center efficiency was proposed by Gao (2014) from Google. His main goal was to create a model that predicts PUE from Google data centers' 2-year historical data using MLP (multilayer perceptron) with 5 hidden layers and 50 nodes in each hidden layer. The features of the neural network were as follows:

1. Total server IT load [kW]
2. Total Campus Core Network Room (CCNR) IT load [kW]
3. Total number of process water pumps (PWP) running
4. Mean PWP variable frequency drive (VFD) speed [%]
5. Total number of condenser water pumps (CWP) running
6. Mean CWP variable frequency drive (VFD) speed [%]
7. Total number of cooling towers running
8. Mean cooling tower leaving water temperature (LWT) setpoint [F]
9. Total number of chillers running
10. Total number of dry coolers running
11. Total number of chilled water injection pumps running
12. Mean chilled water injection pump setpoint temperature [F]
13. Mean heat exchanger approach temperature [F]
14. Outside air wet bulb (WB) temperature [F]
15. Outside air dry bulb (DB) temperature [F]
16. Outside air enthalpy [kJ/kg]
17. Outside air relative humidity (RH) [%]
18. Outdoor wind speed [mph]
19. Outdoor wind direction [deg]

Having such a robust and reliable model allowed Google to do sensitivity analysis and anomaly detection easily. Using controllable variables from the listed features, they analyzed their effects on PUE on several scenarios to decrease PUE.

$$PUE = \frac{P_{total}}{P_{IT}} = \frac{P_{cooling} + P_{IT} + P_{electricalLosses} + P_{misc}}{P_{IT}}$$

For an efficient data center, the power consumption for non-IT components should be as small as possible, i.e., PUE should be close to 1. In anomaly detection scenarios, they used the MLP model to catch erroneous readings of the individual sensors when actual PUE values were higher than predicted PUE values.

Shoukourian et al. (2017) proposed to model the Coefficient of Performance (COP) rather than PUE, as they discussed that COP indicates the power consumption of four cooling circuits, whereas PUE is the combination of COPs from all cooling elements in one number.

$$COP = \frac{Q_{CoolingCircuits}}{P_{CoolingCircuits}}$$

where $Q_{CoolingCircuits}$ is the aggregated amount of cold generated by four cooling circuits (in watts) and $P_{CoolingCircuits}$ is the aggregated amount of power generated by four cooling circuits (in watts).

They argued that using COPs, it is easier to observe the effects of various control variables on the overall efficiency. In their model, they used LSTM with two hidden layers and a final linear layer with the following features extracted from Leibniz Supercomputing Centre (LRZ):

1. Aggregated amount of cold generated by each cooling circuit
2. Aggregated amount of power consumed by the fans of each cooling tower
3. The number of active cooling towers
4. Wet bulb temperature
5. Inlet water temperature (to the distribution bar) from each cooling circuit
6. Return water temperature (to the distribution bar) to each cooling circuit

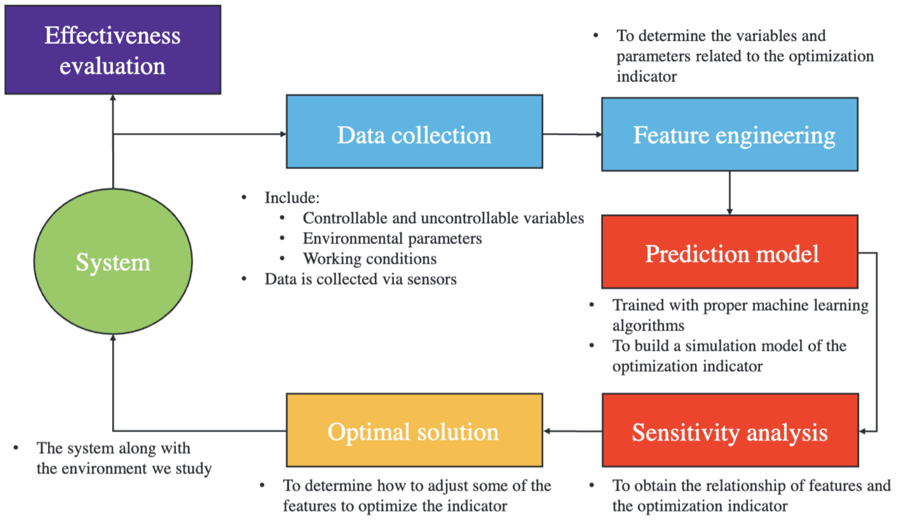

Effectiveness evaluation
Data collection
Feature engineering
To determine the variables and parameters related to the optimization indicator
System
Include:
Controllable and uncontrollable variables
Environmental parameters
Working conditions
Data is collected via sensors
Prediction model
Trained with proper machine learning algorithms
To build a simulation model of the optimization indicator
Optimal solution
Sensitivity analysis
The system along with the environment we study
To determine how to adjust some of the features to optimize the indicator
To obtain the relationship of features and the optimization indicator

Figure 1: PUE optimization framework of Yang et. al (2021)

Yang et al. (2021) proposed another approach similar to Gao (2014) for the PUE optimization framework. First, they collected data from individual sensors in Tencent Tianjin Data Center building No. 4, containing 38 features, similar to the features Gao (2014) used, ranging from January 2018 until November 2019. After applying some feature engineering, they set up different prediction models, i.e., neural networks (NNs), light gradient boosting machine (LightGBM), random forest, and RNN. Finally, they conducted a sensitivity analysis within the constraints to optimize PUE.

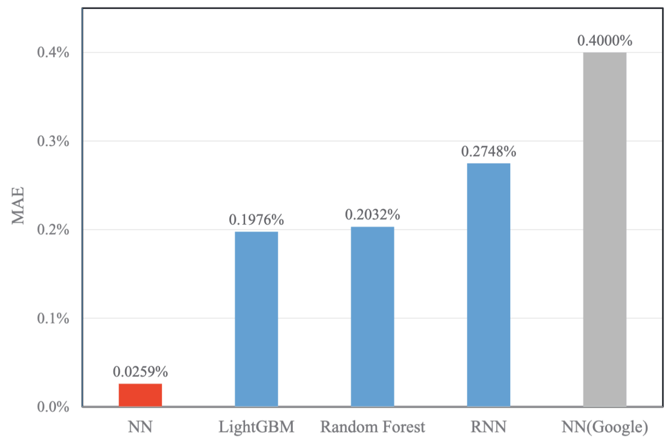


Figure 2: MAE of models Yang et al. (2021) and Gao (2014) used

They found out that the 10 most important features affecting PUE are total server IT load, condenser water flow of chillers, outside air enthalpy, chillers' current percentage, condenser water input temperature of chillers, chilled water output temperature of chillers, indoor air enthalpy, condenser water output temperature of cooling towers, chilled water of chillers and outside air wet bulb temperature. Among the controllable variables, the condenser water flow of chillers is the most important variable. They also outperformed Gao (2014) in the PUE prediction model and their best-performing model was a neural network with 3 hidden layers.

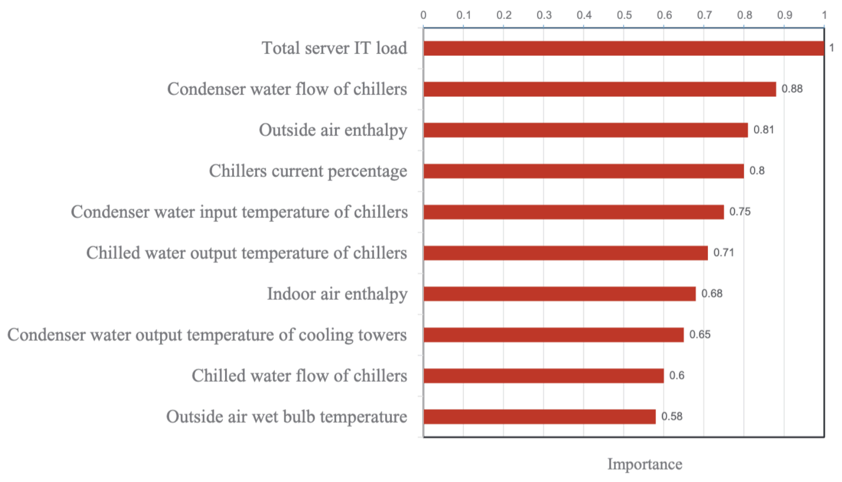


Figure 3: Top 10 most important features affecting PUE from Yang et al. (2021)

2.2. Optimization-based Approaches

Two optimization-based approaches that focus on different aspects of data center systems have previously been proposed. Stamatescu et. al (2018) came up with an approach in which they optimize the scheduling of virtual machines (VMs) in the data center servers with a mixed integer linear programming problem. The decision variables are the assignment of each virtual machine to each server at time t. The constraints mainly focus on the CPU and memory capacity of each server. The objective of the problem is to minimize the energy consumption caused by working servers, migration of VMs between different servers, and turning servers on/off in consecutive time steps.

COMMERCIAL ALGORITHM - ENERGY CONSUMPTION

Hour	1	2	3	4	5
Server 1	660	632	445	335	529
Server 2	160	166	158	146	135
Server 3	142	180	180	178	173
Server 4	10	174	269	8	0
Server 5	5	244	250	272	7
Server Consumption	977	1396	1302	939	844
Total IT Consumption of the Data Center: 5458 kWh/hour					

THE PROPOSED ALGORITHM - ENERGY CONSUMPTION

Hour	1	2	3	4	5
Server 1	628	740	645	0	0
Server 2	0	0	0	140	0
Server 3	0	0	0	0	0
Server 4	0	250	261	250	25
Server 5	258	270	261	270	265
Server Consumption	886	1260	1167	660	290
Total IT Consumption of the Data Center: 4263 kWh/hour					

Figure 4 and 5: Energy consumption comparison between commercial algorithm and Stamatescu et. al (2018)'s algorithm

In a comparison with a commercial algorithm, their approach decreased the total IT consumption by 22%, in a system with 5 servers and 5 types of application.

Lazic et al. (2018) came up with a linear optimization problem with a quadratic objective, by which they optimize the scheduling of air handling units (AHUs) to control the temperature and the airflow of the data center.

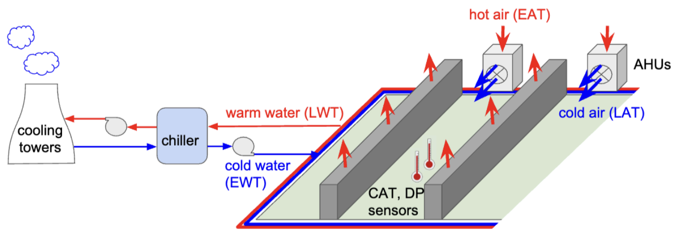


Figure 6: Data center cooling loop from Lazic et al. (2018)

They classified the variables of a data center system into three sections. "Control" variables (variables that can be manipulated) of their system were "fan speed" and "valve opening" for each AHU. "State" (variables that are predicted and regulated) variables were "differential air pressure" (DP), "cold-aisle temperature" (CAT), "entering air temperature" (EAT), and "leaving air temperature" (LAT). "Disturbance" variables were "server power usage" and "entering water temperature" of the chilled water at each AHU. Their optimization model had 1.2K variables along with a "large number" of linear and range constraints.

This review of different methods to make data centers more energy-efficient teaches us a lot about how to manage energy in these important buildings. We looked at two kinds of methods: ones that use predictions and ones that focus on optimization.

From Google's advanced prediction model to other studies that looked at different parts of data center efficiency, we see how these methods are getting better over time. These studies show us that collecting good data is very important, and that computer models like MLP and LSTM can really help predict and improve how much energy is used. They also show that being able to control certain parts of the system is key to making the whole data center work better.

Overall, this review highlights the many ways people are trying to cut down on energy use and be more efficient in data centers. It points out that ongoing research in areas like machine learning and optimization is essential for finding even better ways to manage energy in these centers. In our project with KoçSistem, we have been inspired by the work of Gao (2014) and Yang et al. (2021) mostly, in which they create a predictive model to predict PUE using other features. However, they lack an optimization part that uses an optimization algorithm rather than a sensitivity analysis, which is our main contribution to their approaches. However, due to the violation of the linearity of the mathematical model, we have avoided using nonlinear prediction frameworks used in these works such as LightGBM, LSTM and neural networks that will later be integrated to the mathematical model. The existing optimization-based approaches were not suitable for our case, as Stamatescu et al. (2018) focused on VM allocation to servers and Lazic et al. (2018) was a design-specific application.

3. System Analysis

3.1. Problem Definition

KoçSistem is facing challenges with an ineffective cooling scheduling mechanism in its data centers. The data center's inefficiency is evident via its high Power Usage Effectiveness (PUE) number, as well as the information provided by the responsible operator during the group's visit. The inefficiency originates from the operation of cooling systems that exceed the necessary capacity. Consequently, the Power Usage Effectiveness (PUE) number, which is influenced by variations in the overall server IT load, temperature, humidity, and energy use of cooling systems, exceeds the intended level.

A major challenge is the absence of predictability when it comes to server workload. As a result, this results in the need for manual cooling procedures that are often inaccurate. The elevated PUE score indicates a lack of efficiency in energy use. The issue lies in the fact that cooling operations are carried out manually due to the absence of predictability about server demand. This ambiguity leads to stress on the server and interruption of operations. As a result, this increases the amount of work on the data servers, decreases job performance, and may lead to customer discontent, which might result in financial penalties for KoçSistem in the form of fines.

3.2. The Objectives and Scope of the Project

The main goal of this project is to enhance the efficiency of the cooling system scheduling in KoçSistem's data centers. The project does not include any alterations to the current physical air conditioning systems. On the contrary, it prioritizes the strategic planning of cooling activities to decrease energy use and minimize cooling costs.

The primary objective of this project is to develop a mathematical model that may assist in addressing KoçSistem's issue of excessive energy consumption and reducing the expenses associated with cooling their server rooms at the data centers.

Upon project completion, we will enhance server efficiency by mitigating CPU overheating, hence minimizing client task delays. Our objective is to create a mathematical model that maximizes the efficiency of cooling systems in server rooms. Using the advanced methodology, our objective is to enable staff to efficiently oversee energy use. The aim of this strategy is to effectively ascertain the optimal cooling schedule and timing in accordance with the server workload, including both scheduled and unforeseen server loads. This dynamic method ensures the data servers operate at their highest efficiency while reducing energy consumption.

Sensitivity analysis is a crucial component of our approach. Through sensitivity analysis, we will examine the impact of parameters on energy consumption and the Power Usage Effectiveness (PUE) value. This research will investigate how variations in factors such as workload, outside temperature, and cooling system use might influence energy usage. Furthermore, this will enable us to maintain the server temperature within a certain range by closely monitoring the impact of these factors on the server's thermal conditions.

3.3. Data Analysis

For our data analysis, we started by looking at spreadsheets that were split into different sheets. We were provided with various data in different weeks, given what data was thought to be useful and our requests as our data analysis continued. We figured out that the common interval of the data provided by KoçSistem was from the beginning of April 2023 until the end of October 2023. Although this situation causes a lack of understanding of the system since it doesn’t include the winter months, we decided to use the data that is available to us rather than creating synthetic data.

Through the data center, there are different rooms with different layouts and cooling units. We received data from rooms KS2, KS8, and KS10, but among them, the most promising room was KS10 because its data consisted of more features. We received daily and 5-minute data from KoçSistem. We decided to use only the 5-minute data to keep our merged data consistent with its timestamps, and the information provided by the daily data was not worth upsampling to 5-minute periods.

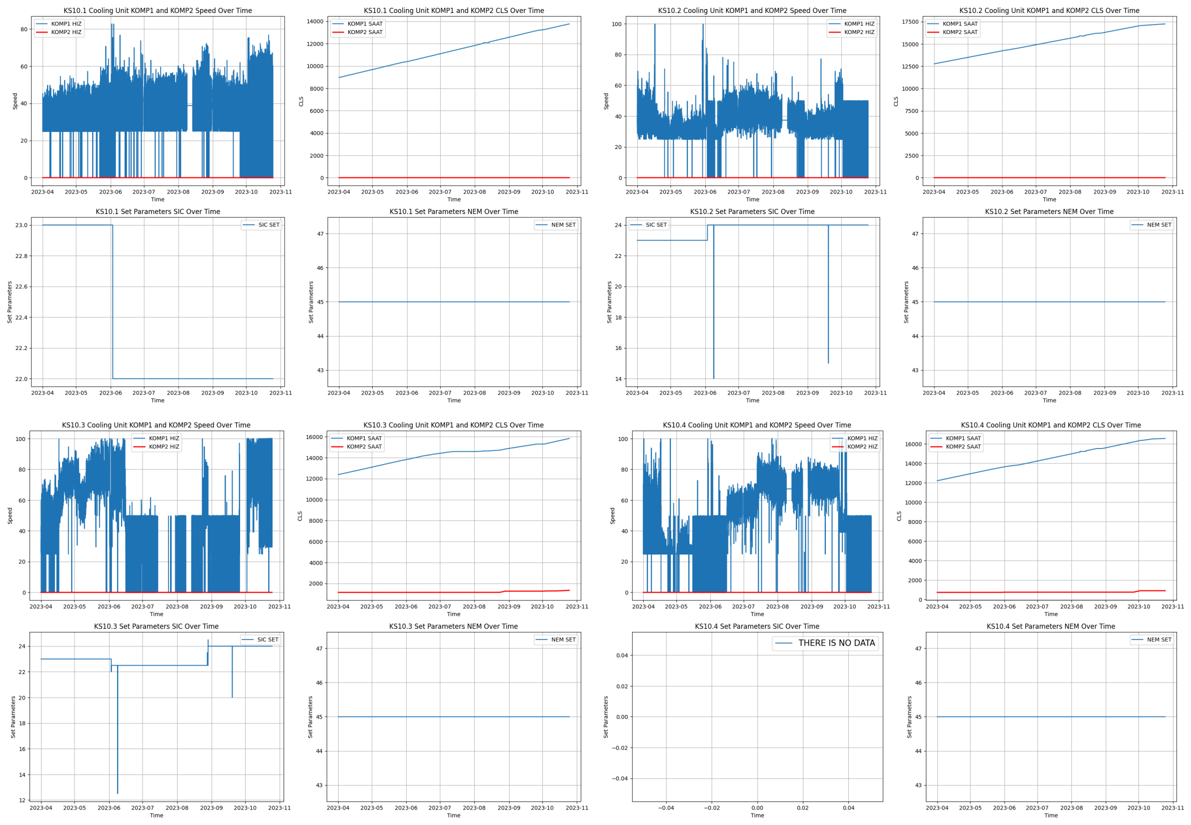


Figure 7: Cooling unit fan speeds,temperature, and humidity set parameters for the servers in KS10.

We noticed that out of two cooling units on each server, only one of them had been utilized, as can be seen in Figure 7. The humidity set parameters for each server were held constant, and the temperature set parameters were mostly kept the same. Clock values kept track of the total time each cooling unit was being utilized. Therefore, we only decided to include KOMP1 cooling unit speeds, as they are the only meaningful parameter among the server-related parameters. These graphs are indicative of a system designed to monitor and adjust the cooling parameters of a unit, likely to optimize its efficiency and performance. The Figure 7 shows a mix of steady control setpoints and variable speeds, which could reflect adaptive responses to operational needs or external factors. The absence of data in certain parameters for KS10.4 suggests discrepancies in data collection or perhaps an operational issue that needs to be addressed. Overall, the graphs are tools for ensuring the cooling units are functioning correctly and efficiently highlighting areas where maintenance or adjustments might be necessary.

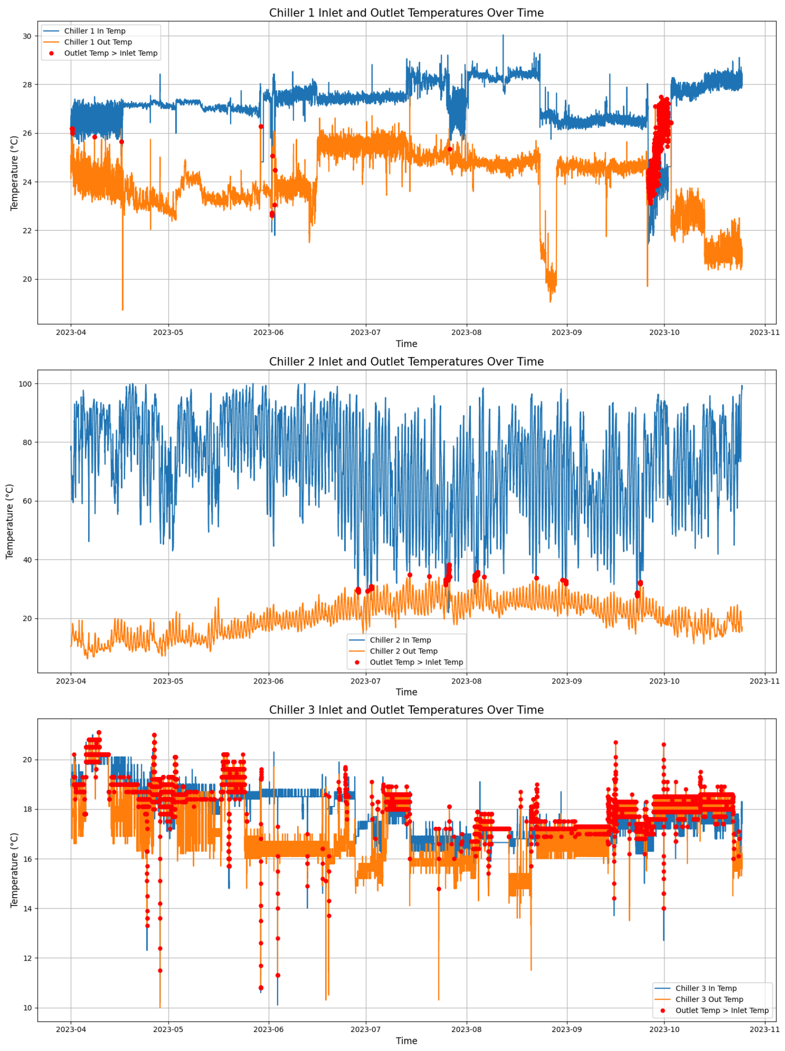


Figure 8: Chiller inlet and outlet water temperatures in the KS10 room.

In the KS10 room, there are three distinct chillers designed to regulate the system's temperature. According to the system's operational guidelines, the chillers receive water at the inlet temperature, which is the heat-carrying water emerging from the servers. After the cooling process, the chillers then leave the water at the outlet temperature back to the servers.

The expected outcome of this cooling cycle is that the outlet temperature should be lower than the inlet temperature, indicating that heat has been removed from the water circulating through the servers. In the initial two graphs of Figure 8, this relationship holds true, as the inlet temperatures are generally higher than the outlet temperatures, which aligns with the standard operation of such cooling systems.

However, an anomaly is observed in the third graph marked with red dots. There are numerous instances where the inlet temperature is unexpectedly lower than the outlet temperature. This reversal suggests that instead of the chiller extracting heat, there is an introduction of warmth into the water. Such a scenario typically indicates the engagement of a supplementary cooling mechanism. This additional cooling seemingly occurs within the server room itself, lowering the water temperature before it reaches the chiller.

When we consider the performance of the third chiller in light of this data, it's evident that, contrary to its intended function, it often acts to elevate the system's temperature. This irregularity indicates that, for most of the observed timestamps, the third chiller is not effectively cooling the system and may, in fact, be contributing to an increase in temperature.

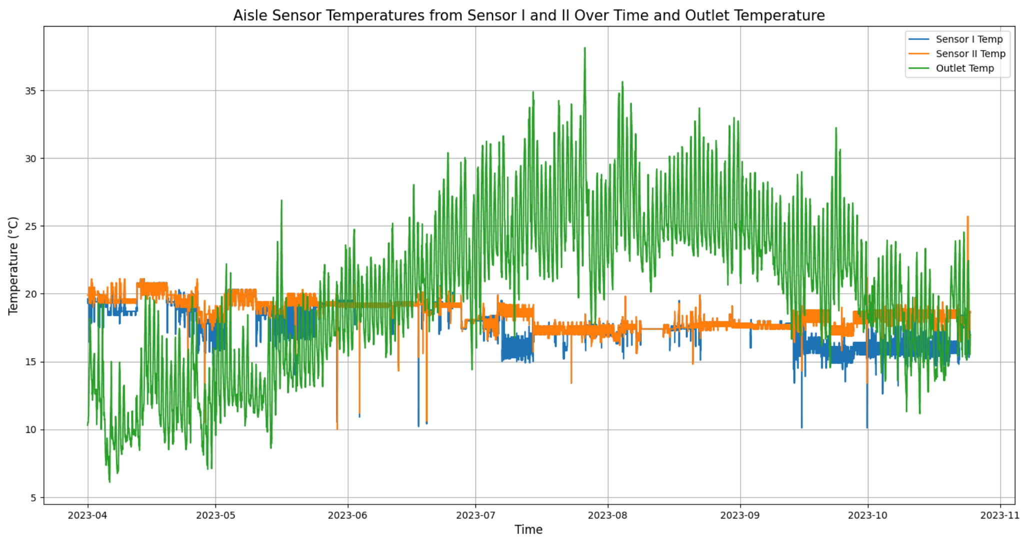


Figure 9: Sensor and outlet temperatures between April-November 2022 for the KS10 room.

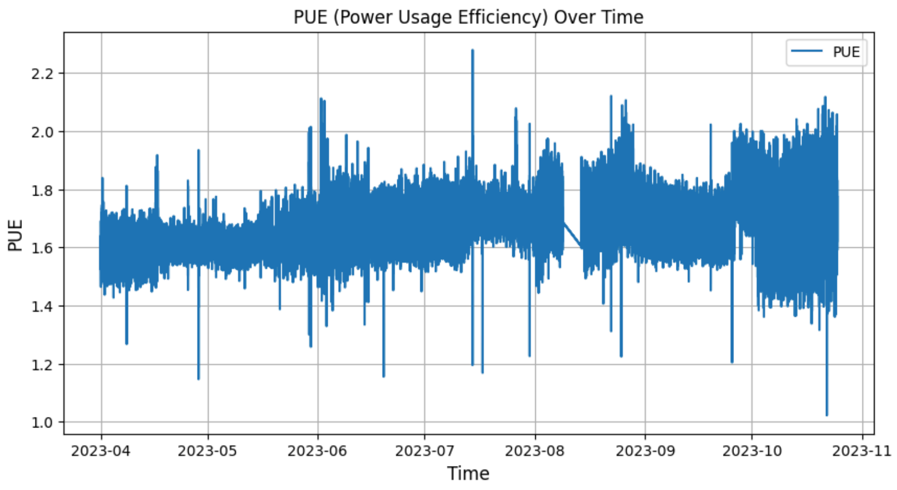


Figure 10: PUE between April-November 2023 of KS10 room.

Also, as seen in Figure 10, the PUE of the KS10 room fluctuates mostly between 1.4 and 2.0. However, Google achieved a PUE of 1.1 in 2014, as Gao (2014) expresses. Through nine years, the KS10 room has never reached a stable PUE value close to 1.1. Although Google's cooling systems are probably much more sophisticated and optimized, our observation is that having an average PUE of 1.7 is significantly inefficient and there is a room for improvement considering the fact that there are some observation at which outlet temperature of chillers are warmer than the inlet temperatures, which means that the water used in cooling servers actually cools inside the servers. Therefore, we come to the conclusion that the KS10 room's cooling scheduling can be significantly improved.

4. Proposed System

4.1. Inputs and Outputs

In the proposed system, the inputs include the speeds of four cooling units, which are monitored every five minutes; in and out water temperatures for three chillers; and room temperatures from two sensors. Additionally, the system acquires minute weather forecasts from online sources. These inputs are crucial for the system to determine the optimal operational speeds for the cooling units. The system makes decisions for the upcoming hour and provides an estimated PUE value and room temperature.

4.2. Major Components

Major components are categorized into data preprocessing, predictive modeling, model validation and selection, prediction models, and decision models.

4.2.1. Data Preprocessing

We incorporated the raw data that KoçSistem provided for data preprocessing. In our examination of the provided data, we realized that there are some NaN values at some timestamps for some features. We filled those gaps instead of removing the incomplete rows to preserve the 5-minute structure. We also replaced any extreme values that didn't fit the pattern by taking the average at the subsequent timestamps of such features. We added further dummies and lagged values of specific features to our data to improve the accuracy of the models that we are planning to train. The included dummies are days of the week, hours of the day, and months of the year. As we noticed that outlet temperature and humidity values are also important, we extracted those values using web scraping and incorporated them into our data. We also added a 1-day lagged value of PUE to be used in the PUE prediction model. After data preprocessing, our finalized data consists of 59342 data points and 53 features. Features include PUE, room aisle temperatures from 2 sensors, chiller inlet-outlet water temperatures, cooling unit fan speeds, outlet temperature and humidity values, dummies, and lagged features.

Before training and validating models, we split the data into training data, which is 80% of the whole data, and test data by random splitting.

4.2.2. Predictive Modeling and Model Validation

In predictive modeling, we implement several machine learning models, such as OLS, Random Forest, and Gradient Boosting, to predict PUE values and room temperature. All models take into account the features mentioned above (speed of the cooling units, in and out temperatures of the chillers, temperatures recorded by the sensors in the KS10 room, outlet weather conditions, dummy, and lag variables) when predicting the PUE values. For room temperature forecasting, models take the features excluding PUE and sensor temperatures.

In the model validation and selection part, we investigated each trained machine learning model for both PUE and temperature prediction. RMSE and MAPE values are compared between models for train and test predictions. We also checked that residuals are normally distributed to ensure that our models predict properly without neglecting key assumptions. We have also considered removing some of the features that are correlated with many of the other features, however as seen in Figure 12, although some features are correlated with some other features, we didn't consider removing a feature meaningful as one single feature wasn't correlated with many other features.

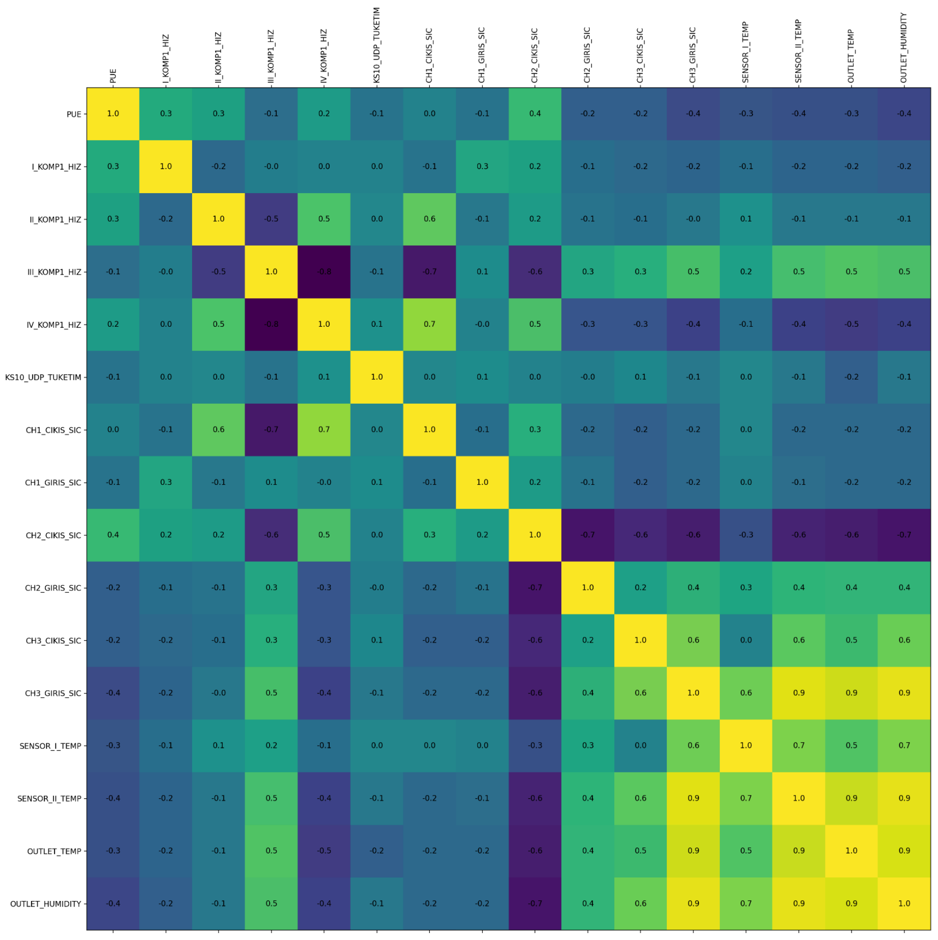


Figure 12: Correlation matrix for features (without dummies) used in prediction models

After the validation part, we evaluated the candidate models for PUE and temperature prediction. The selected models will be used while exploring the possible solutions to the scheduling problem of cooling unit speeds. You can find the implementation of our predictive models and time series plots comparing true and predicted values for each model and predicted feature in the appendix.

Model	**Parameters**	**Train MSE**	**Train RMSE**	**Train MAE**	**Train MAPE**	**Test MSE**	**Test RMSE**	**Test MAE**	**Test MAPE**
OLS	None	0,0027	0,0519	0,0379	0,0225	0,0027	0,0524	0,0225	0,0226
Random Forest	max_features = 'sqrt'	**0,0002**	**0,0142**	**0,0101**	**0,0060**	0,0015	0,0387	0,0276	0,0164
Gradient Boosting	learning rate = 0.1 depth = 5	0,0011	0,0325	0,0238	0,0142	**0,0012**	**0,0360**	**0,0253**	**0,0151**
Lasso	alpha = 2e-5	0,0032	0,0566	0,0420	0,0249	0,0033	0,0571	0,0423	0,0250

Table 1: Error values for various predictive models of PUE

In PUE prediction, we have achieved a test MAPE of 1.51% in our best-performing model, gradient boosting. However, to preserve the linearity of the mathematical model to be used, we have selected OLS for PUE prediction, which has a MAPE of 2.26%, only 0.75% worse than the best performing model.

Model	**Parameters**	**Train MSE**	**Train RMSE**	**Train MAE**	**Train MAPE**	**Test MSE**	**Test RMSE**	**Test MAE**	**Test MAPE**
OLS	None	0,5008	0,7077	0,5447	0,0310	0,5015	0,7081	0,5435	0,0309
Random Forest	max_features = 'sqrt'	**0,0121**	**0,1102**	**0,0599**	**0,0034**	**0,0883**	**0,2973**	**0,1596**	**0,0092**
Gradient Boosting	learning rate = 0.1 depth = 5	0,1147	0,3387	0,2120	0,0121	0,1346	0,3669	0,2195	0,0126
Lasso	alpha = 2e-5	0,5642	0,7511	0,5730	0,0328	0,5660	0,7523	0,5731	0,0328

Table 2: Error values for various predictive models of Sensor I temperature

Model	**Parameters**	**Train MSE**	**Train RMSE**	**Train MAE**	**Train MAPE**	**Test MSE**	**Test RMSE**	**Test MAE**	**Test MAPE**
OLS	None	0,1580	0,3975	0,2142	0,0113	0,1523	0,3902	0,2136	0,0112
Random Forest	max_features = 'sqrt'	**0,0029**	**0,0541**	**0,0320**	**0,0017**	**0,0172**	**0,1315**	**0,0837**	**0,0045**
Gradient Boosting	learning rate = 0.1 depth = 5	0,0275	0,1659	0,1074	0,0057	0,0291	0,1708	0,1095	0,0058
Lasso	alpha = 2e-5	0,1717	0,4143	0,2373	0,0125	0,1650	0,4062	0,2367	0,0125

Table 3: Error values for various predictive models of Sensor II temperature

In sensor temperature prediction, we have achieved a test MAPE of 0.88% for sensor-I and 0.41% for sensor-II in our best-performing model, random forest. Similar to the PUE case, we have also used OLS for prediction to preserve the linearity of the mathematical model.

4.2.3. Decision Model

4.2.3.1. Dynamic Optimization with LP

In our first approach, we have created an LP model that optimizes the system by scheduling the speed of the cooling units for each 5 minutes to minimize the PUE value of the KS10 room while satisfying the desired range for room temperature. For the decision model, we will implement a linear programming model to select the optimal speeds of cooling units for each 5-minute timestamp. With this decision model, we are planning to schedule cooling unit speed without sharp changes over subsequent periods by introducing additional constraints, because of the reason that speed changes can cool or heat the system after a certain time but not instantly.

In terms of the mathematical model, our model will be as follows:

$$minimize \qquad PUE$$

$$s.t.$$

$$OLS_{PUE}(\{v_{t,1}, v_{t,2}, v_{t,3}, v_{t,4}\}\ U\ V_{t,1}) = PUE \qquad (1)$$

$$OLS_{Sensor-I}(\{v_{t,1}, v_{t,2}, v_{t,3}, v_{t,4}\}\ U\ V_{t,2}) \leq 24 \qquad (2)$$

$$OLS_{Sensor-I}(\{v_{t,1}, v_{t,2}, v_{t,3}, v_{t,4}\}\ U\ V_{t,2}) \geq 14 \qquad (3)$$

$$OLS_{Sensor-II}(\{v_{t,1}, v_{t,2}, v_{t,3}, v_{t,4}\}\ U\ V_{t,2}) \leq 24 \qquad (4)$$

$$OLS_{Sensor-II}(\{v_{t,1}, v_{t,2}, v_{t,3}, v_{t,4}\}\ U\ V_{t,2}) \geq 14 \qquad (5)$$

$$v_{t,i} \leq 1.1 * v_{(t-1),i} \qquad for\ \forall i \in \{1, 2, 3, 4\} \qquad (6)$$

$$v_{t,i} \geq 0.9 * v_{(t-1),i} \qquad for\ \forall i \in \{1, 2, 3, 4\} \qquad (7)$$

$$v_{t,i} \leq 100 \qquad for\ \forall i \in \{1, 2, 3, 4\} \qquad (8)$$

$$v_{t,i} \geq 0 \qquad for\ \forall i \in \{1, 2, 3, 4\} \qquad (9)$$

$$v_{t,i} \in R \qquad for\ \forall i \in \{1, 2, 3, 4\} \qquad (10)$$

where;

- V_1 is the set of features to be used in the PUE prediction model except cooling unit speeds, which are chiller inlet and outlet water temperatures, outlet temperature and humidity, room sensor temperatures, server workload, dummies and 1-day lagged value of PUE.

- V_2 is the set of features to be used in sensor temperature prediction models except cooling unit speeds, which are chiller inlet and outlet water temperatures, outlet temperature, outlet humidity, server workload and dummies.

- $v_{t,1}$, $v_{t,2}$, $v_{t,3}$ and $v_{t,4}$ are the cooling unit speeds in servers 1, 2, 3 and 4 at time t. $v_{t,i}$ will be our decision variables, and we will have the information of the set $\{v_{s,i} \mid s < t\}$.

- Constraint 1 is the PUE prediction of our OLS model trained previously. We will try to minimize the predicted PUE with changing values of $v_{t,i}$. Since OLS is used, the constraint is linear as we use the coefficients coming from OLS.

- Constraints 2-5 are the sensor temperature predictions of our OLS models trained previously for sensor temperature prediction. We will keep the room temperature within specified limits (14-24 ℃) while changing the cooling unit speed.

- Constraints 6 and 7 ensure that we do not have a sharp decrease or increase in cooling unit speeds, which can cause redundant energy consumption and cooling unit malfunctioning. In the model above, we allow a maximum change of 10%, but we have also additionally tested 25% and 50% to observe its effects on the model.

- Constraints 8 and 9 ensure that cooling unit speeds lie between 0 and 100.

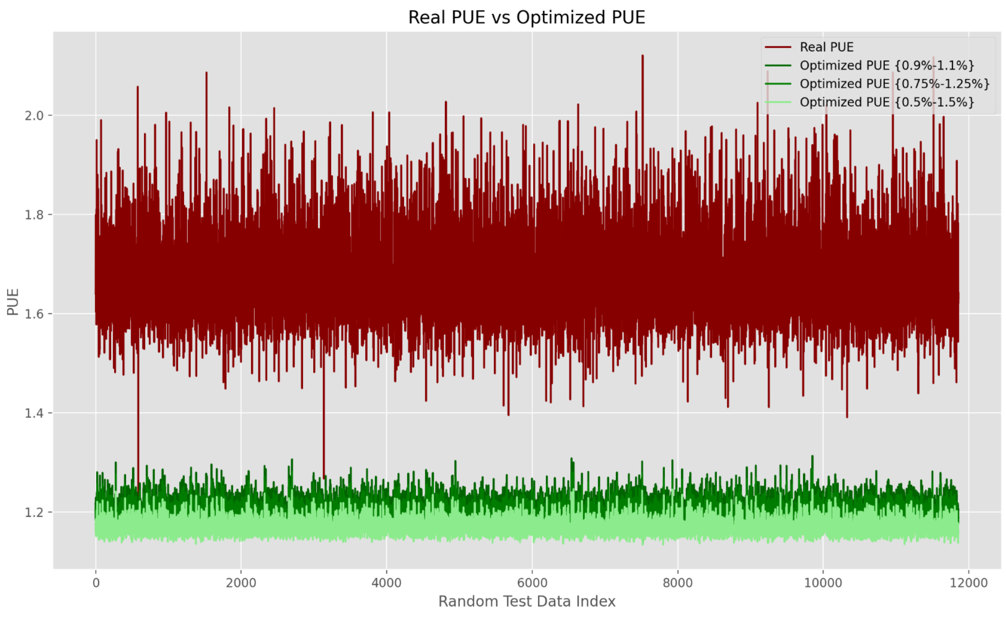


Figure 13: Results of our dynamic LP model for 3 different flexibility settings

Our dynamic cooling unit speed scheduling model was evaluated on a randomly selected test split of 11868 data points. Out of all 11868 data points, our LP model resulted in an optimal solution in 11858 data points, 10 data points resulted in infeasible solution. From an average

PUE of 1.67 of KoçSistem's data center, our model resulted in average PUEs of 1.17, 1.19 and 1.20 for 50%, 25% and 10% flexibility values, corresponding improvements of 29.9%, 28.6% and 27.7%. PUE decreases as the flexibility of fan speeds increase, due to the fact that the prediction models cannot estimate that fan speed changes cannot be reflected to the room temperatures instantaneously. Therefore, it is essential to keep this constraint in our model to have a more accurate solution.

4.2.3.2. Multi-timestamp Optimization by Using Tabu Search Algorithm

Metaheuristic algorithms solve optimization problems that exact methods find hard to solve. They are approximate algorithms that can find a good solution in a short time. They are also flexible algorithms to use for different problems. In our problem, we have to schedule the cooling speeds of the cooling components in the data center in order to minimize the PUE of the data center while keeping the sensor temperature within a range of +/- 4 degrees from their set parameters. Additionally, we have to keep the cooling speeds as close as possible to the previous cooling speeds to keep the cooling system stable. With all these constraints, because we will be trying to schedule multiple timestamps at one time while considering the future of the system, it is hard to solve this problem with an LP or MILP model. Also, because we have to predict the future of the system, we have to use different prediction models that can be non-linear. As described in this section, we build a framework that can first predict the future of the system and then schedule the cooling speeds of the cooling components in the data center according to our objective by using the Tabu Search algorithm.

Tabu Search Algorithm

Tabu Search is a metaheuristic algorithm to solve optimization problems. It is a local search algorithm. However, thanks to its memory-keeping, unlike other local search algorithms, it can escape from local optima. It uses a tabu list to keep track of the search history, and it forbids the search to go back to the previous states. This way, it can explore the search space more efficiently. Furthermore, the incorporation of an aspiration criterion enables the search to revisit previous states if the current state surpasses the best solution found thus far. In general, Tabu Search is an appropriate and advantageous algorithm to use in our cooling speed scheduling problem for several reasons. First of all, it is suitable for this multi-timestamp scheduling problem because it can handle multiple objectives without needing linearization. Secondly, its flexibility allows us to easily implement the constraints of the problem and use all our prediction models, even though they are not linear. Finally, it is a local search algorithm, so it is fast and efficient to find a good solution in a short time.

Representation of Solution

In our problem, we have 4 different cooling speeds for each timestamp, and there are multiple timestamps that we can decide their number. We keep the speeds of each timestamp in a tuple that has a length of 4, and we keep all timestamps in a list. So, our solution is a list of tuples that has a length of several timestamps that we can decide. For example, if we have 3 timestamps, and 4 different cooling speeds, our solution will be like this: [(s1, s2, s3, s4), (s1, s2, s3, s4), (s1, s2, s3, s4)] where s1, s2, s3, s4 are cooling speeds.

Besides cooling speeds, we also need to keep all the information about each timestamp. We need to calculate PUE and sensor temperature for each timestamp, and we need to keep track of the features that have an effect on PUE and sensor temperature in each timestamp. Because of that reason, we keep the information of a timestamp as an object with the attributes of cooling speed, PUE, sensor temperature, and other features. Every time we need to change the cooling speed of a timestamp and calculate PUE and sensor temperature, we update the information about that timestamp in that object and calculate the objective function value by using the updated objects of timestamps in a list.

Initialization

In order to maintain similar cooling speeds, we begin by assigning the previous cooling speed to the current cooling speed for each timestamp. Starting the search process with the cooling speed of the previous solution is an effective strategy to start exploring the search space from the part where we want to find a solution.

Furthermore, it is necessary to initialize the data for each timestamp that we plan to schedule, in addition to setting the cooling rates. We need to predict features in these timestamps to use them while predicting the PU and sensor temperature of each timestamp in the objective function.

Prediction of the Future System with ARIMA Models:

We use ARIMA models to predict the features that have an effect on PUE and sensor temperature by using the historical data of these features. Mostly we first differentiate the data to make it stationary, and then we use ARIMA models that are shown in the following for each feature.

- CH1_CIKIS_SIC > ARIMA([6, 12], 1, 0)
- CH1_GIRIS_SIC > ARIMA([6, 12], 1, 0)
- CH2_CIKIS_SIC > ARIMA([12], 1, 0)
- CH2_GIRIS_SIC > ARIMA([12], 1, 0)
- CH3_CIKIS_SIC > ARIMA(0, 1, 1)
- CH3_GIRIS_SIC > ARIMA(1, 1, 0)
- KS10_UDP_TUKETIM > ARIMA(1, 1, 0)

The autocorrelation and partial autocorrelation plots were the primary factors influencing our choice of these models. We included these plots in the appendix section to demonstrate the reason behind our model selection. To examine them thoroughly, please check the appendix section.

Objective Function

Our objective function is to minimize the PUE of each timestamp while keeping the sensor temperature in the range of +/- 4 degrees of their set parameters and keeping the cooling speeds as close as possible to the previous cooling speeds. In addition to these, because we have multiple timestamps, we have to first calculate the PUE and sensor temperature of each timestamp, and then we have to check if that timestamp obeys the constraints. To calculate the PUE and sensor temperature of each timestamp separately, we use another function that takes the information of a timestamp as input and returns the objective function value of that timestamp. Then, we take the weighted average of the objective function values of all timestamps. Pseudocode of the objective function is as follows:

Algorithm 1 Objective Function

```
1:  procedure calculate_objective (speed_list)
2:      input: a list of tuples that has a length of a number of timestamps that we can decide
3:      output: objective function value of the solution
4:      total_objective = 0
5:      for i from 0 to number of timestamps do
6:        for j from 0 to length of a timestamp do
7:          Set the speed of component j in the object of this timestamp to speed_list[i][j]
8:        end for
9:        objective_timestamp = calculate the objective of this timestamp
10:       if i is 0 then
11:         for j from 0 to length of a timestamp do
12:           if current speed of component j is not in the range of +/- percentage that we can change from the initial speed of component j then
13:             objective_timestamp += Penalty
14:           end if
15:         end for
16:       else
17:         for j from 0 to length of a timestamp do
18:           if this speed of timestamp i is not in the range of +/- percentage that we can change from the previous speed of timestamp i then
19:             objective_timestamp += Penalty
20:           end if
21:         end for
22:       end if
23:       total_objective += (objective_timestamp x (no_timestamps - i) / no_timestamps)
24:     end for
25:     objective = total_objective / sum(from 1 to number of timestamps)
26:     return objective
27: end procedure
```

Algorithm 2 Objective Timestamp Function

```
1:  procedure calculate_objective_timestamp (s_timestamp)
2:      input: an object representing a single timestamp
3:      output: objective value for the provided timestamp
4:      PUE, sensor_I_temp, sensor_II_temp = predict_func(s_timestamp)
5:      temperature_penalty = 0
6:      if sensor_I_temp - s_timestamp.SIC_I_SET > 4 then
7:          temperature_penalty += Penalty x 2
8:      end if
9:      if s_timestamp.SIC_I_SET - sensor_I_temp > 4 then
10:         temperature_penalty += Penalty x 2
11:     end if
12:     if sensor_II_temp - s_timestamp.SIC_II_SET > 4 then
13:         temperature_penalty += Penalty
14:     end if
15:     if s_timestamp.SIC_II_SET - sensor_II_temp > 4 then
16:         temperature_penalty += Penalty
17:     end if
18:     pue_penalty = (Penalty / 100) x ((PUE – 1) / 0.1)
19:     objective_timestamp = pue_penalty + sic_set_penalty
20:     return objective_timestamp
21: end procedure
```

Neighborhood Functions

We have seven different neighborhood functions that we can use to change the cooling speeds of timestamps. These functions can be used by randomly selecting one of them each time the Tabu Search algorithm generates a new neighbor. We prefer to use this randomization approach as it enables us to efficiently explore the search space.

N1 - Uniform Averaging: This function calculates the average speed across four components for each timestamp and sets all component speeds to this average, effectively homogenizing the speeds at each timestamp.

N2 - Single Timestamp Averaging: It selects a random timestamp and sets the speed of all four components within this single timestamp to their average, leaving other timestamps unchanged.

N3 - Random Speed Increase: A random speed from a random timestamp is chosen and increased by 10%, adjusting only that single speed value and maintaining the others as they are.

N4 - Random Speed Decrease: Similar to N3, but instead, it decreases the chosen random speed by 10%, affecting only that particular speed within the timestamp.

N5 - Uniform Random Decrease:
This function selects a random timestamp and applies a random uniform decrease of 5%, 10%, or 15% to all speeds within that timestamp.

N6 - Uniform Random Increase:
It mirrors the N5 function but applies a uniform random increase to all speeds within a randomly selected timestamp.

N7 - Reversion to Initial Speeds:
Resets the speeds of all components in all timestamps to their initial values, effectively undoing any changes made during the search process.

The Algorithm

The Tabu Search algorithm improves an initial solution by systematically choosing random neighborhood functions to investigate neighboring solutions while rejecting previously visited solutions kept in a tabu list. The algorithm selects the best solution unless a tabu condition forbids it while excluding solutions that are superior to the current best solution. After a predetermined number of cycles, the procedure consistently updates the best solution discovered and concludes by returning the optimum solution once all iterations are finished.

Algorithm 3 Tabu Search

```
1:  procedure tabu_search (initial_solution, neighborhood_functions,
    neighborhood_size, tabu_size, max_iterations)
2:      input: an initial solution, a list of neighborhood functions, the number of
    neighbors in   the neighborhood, the maximum size of the tabu list, and the
    maximum number of iterations
3:      output: the best solution found by the algorithm
4:      best_solution ← initial_solution
5:      best_score ← calculate_objective(initial_solution)
6:      tabu_list ← []
7:      for iteration in 1 to max_iterations do
8:          neighborhood_function ← random_choice(neighborhood_functions)
9:          neighborhood ← populate_neighborhood(neighborhood_function,
    current_solution, neighborhood_size)
10:         candidate_solution ← null
11:         candidate_score ← INFINITY
12:         for neighbor in neighborhood do
13:             if (neighbor not in tabu_list) or (calculate_objective(neighbor) <
    best_score) then
14:                 temp_score ← calculate_objective(neighbor)
15:                 if temp_score < candidate_score then
16:                     candidate_solution ← neighbor
```

```
17:                 candidate_score ← temp_score
18:               end if
19:             end if
20:           end for
21:           if candidate_solution is not null then
22:             current_solution ← candidate_solution
23:           if length(tabu_list) >= tabu_size then
24:             REMOVE_OLDEST_ENTRY(tabu_list)
25:           end if
26:           APPEND(tabu_list, candidate_solution)
27:           if candidate_score < best_score then
28:             best_solution ← candidate_solution
29:             best_score ← candidate_score
30:           end if
31:         end if
32:       end for
33:       return best_solution
34:   end procedure
```

Framework

We build a framework that can schedule the cooling speeds of the cooling components in the data center for multiple timestamps by using the Tabu Search algorithm. In this framework, we first initialize the cooling speeds of the timestamps, and then we predict the features that have an effect on PUE and sensor temperature for each timestamp. After that, we use the Tabu Search algorithm to find the best solution by using the objective function that we described above. Thanks to this framework, we can schedule the cooling speeds of the cooling components in the data center for multiple timestamps by considering the future of the system. However, because of the prediction errors, optimizing the last timestamp may not be the best solution. To solve this problem, we apply a rolling-horizon approach. After we schedule the cooling speeds of the cooling components in the data center for the first timestamp, we update the information of the first timestamp with the real values of the first timestamp and solve the problem again by starting with the second timestamp. So, in this way, every time the algorithm schedules multiple timestamps, it corrects farther timestamps in the next run.

Results

We use this framework 20 times by starting with different timestamps to schedule 10 timestamps in every run and 100 timestamps in total. We use the following parameters in our framework:

Neighborhood size: 30
Tabu list size: 10
Maximum number of iterations: 30000
Penalty: 10^6
Percentage that we can change: 25%

The following figure depicts the comparison of the PUE values of the timestamps that we scheduled with the PUE values of the timestamps that are scheduled by KoçSistem in real life. As can be seen in the figure, because our algorithm takes the future of the system into account, it schedules the cooling speeds of the cooling components in the data center more efficiently than KoçSistem. However, because we cannot test our algorithm in real life and collect the real data, at the end of the runs, we have more PUE values that are higher than the PUE values of KoçSistem in some timestamps. This is because as the algorithm progresses, the prediction errors accumulate, and the system runs out of the data that we used to train our prediction models. To solve this problem, we run this framework by starting with 20 different timestamps and scheduling 10 timestamps in every run. As a result of this run, which can be seen in Figure 14, while we are obtaining the 1.44 PUE value on average, KoçSistem obtains the 1.69 PUE value.

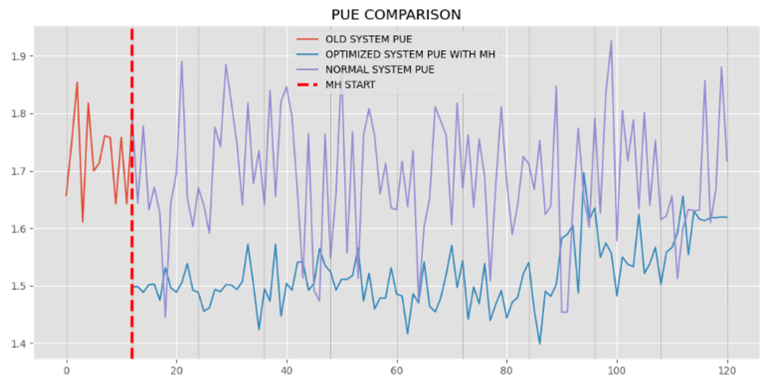


Figure 14: Comparison of optimized and real PUE with tabu search algorithm

Furthermore, as it can be seen from Figure 15, the scheduling of the cooling speeds is more stable than KoçSistem as we expected.

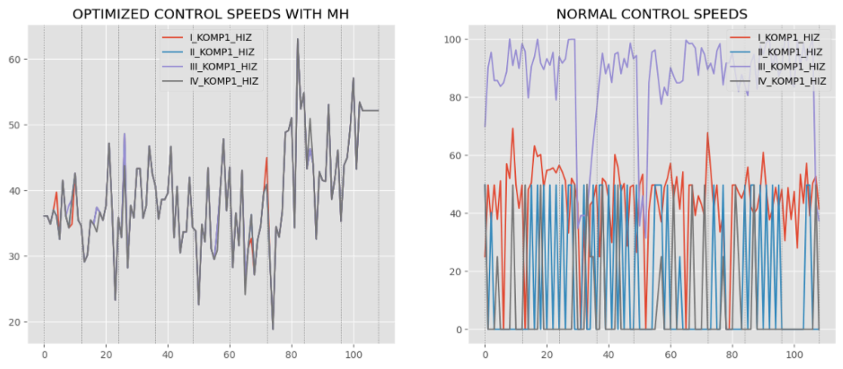


Figure 15: Comparison of compartment speeds in cooling units with tabu search algorithm

5. Conclusion

Our comprehensive study involved a detailed analysis of existing predictive and optimization-based approaches in data center energy management. The project used the power of machine learning, with models like OLS, Gradient Boosting, and Random Forest, to predict the Power Usage Effectiveness (PUE) and room temperatures to optimize cooling processes. The prediction framework allowed us to use two different approaches for the optimization of cooling scheduling: a dynamic optimization model using Linear Programming (LP) and a multi-timestamp optimization framework employing the Tabu Search algorithm.

The LP model successfully scheduled cooling unit speeds, reducing the average PUE from 1.67 to values between 1.17 and 1.20, depending on the flexibility settings. This result signifies a substantial improvement in energy efficiency. The Tabu Search algorithm, integrating ARIMA model predictions, further enhanced our ability to schedule cooling speeds over multiple timestamps, considering the future system states via predictions of different features. It reduced the average PUE of 1.69 on its test set to 1.44, while outputting a cooling speed value for the next 50 minutes. Our approach also not only optimized PUE values but also ensured stable cooling speed schedules compared with KoçSistem's current systems.

References

1. Gao, J. (2014), “Machine learning applications for data center optimization”, Google.
2. Lazic, N.; Lu, T.; Boutilier, C.; Ryu, M.; Wong, E.; Roy, B. and Imwalle, G. (2018) “Data center cooling using model-predictive control”, Proceedings of the 32nd International Conference on Neural Information Processing Systems, 3818-3827.
3. Le, V. D. (2021). "Air free-cooled tropical data center: design, evaluation, and learned lessons". IEEE Transactions on Sustainable Computing.
4. Shoukourian, H.; Wilde, T.; Labrenz, D. and Bode, A. (2017), “Using machine learning for data center cooling infrastructure efficiency prediction”, 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 954-961.
5. Stamatescu, I.; Ploix, S.; Fagarasan, I.; and Stamatescu, G. (2018). “Data center server energy consumption optimization algorithm”. 2018 26th Mediterranean Conference on Control and Automation (MED).
6. Yang, Z.; Du, J.; Lin, Y.; Du, Z.; Xia, L.; Zhao, Q.; and Guan, X. (2021). “Increasing the energy efficiency of a data center based on machine learning”. Journal of Industrial Ecology, 26(1), 323–335.

Appendix

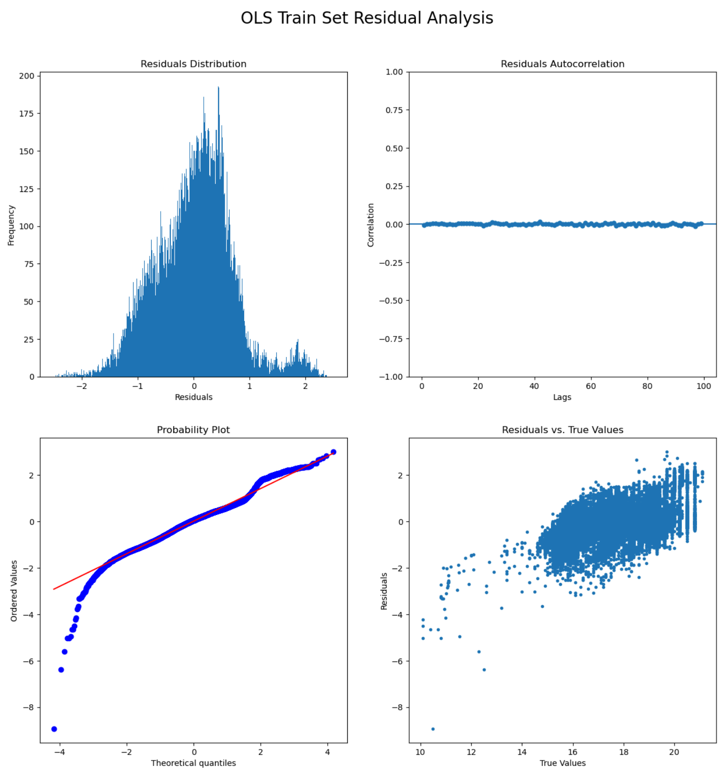


Figure 16: Residual analysis of OLS model for PUE prediction on training set

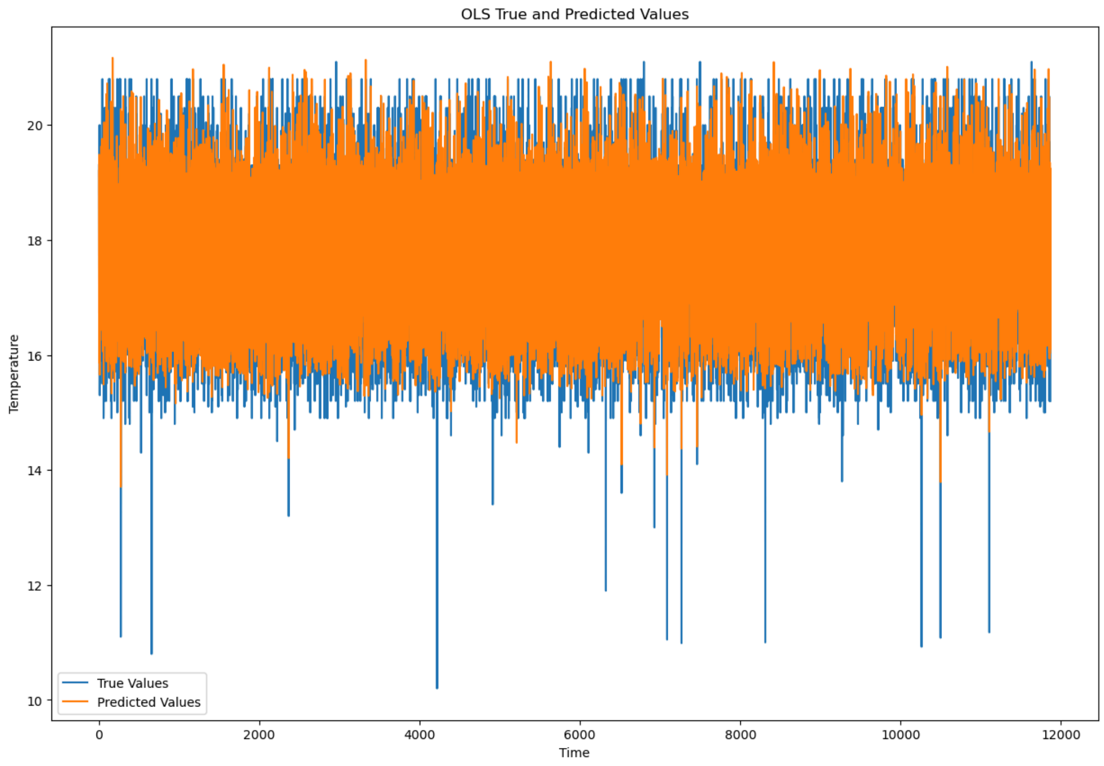


Figure 17: Predicted vs. actual PUE values in OLS model for test set

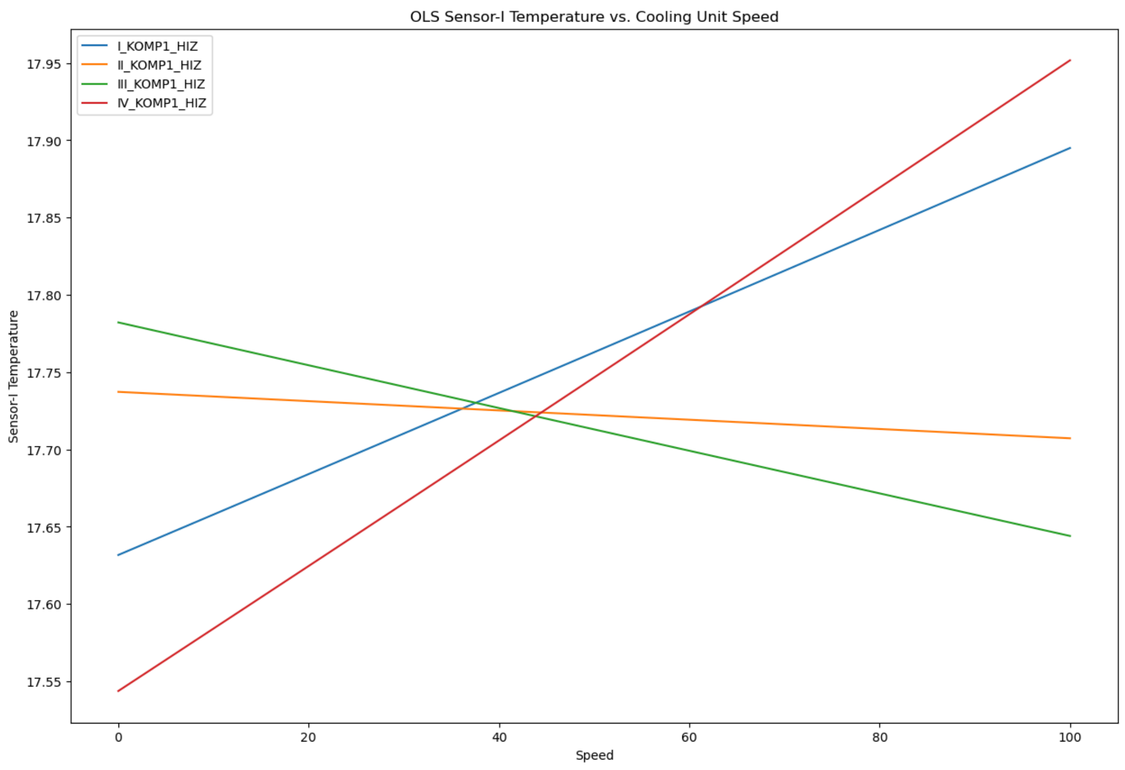


Figure 18: Sensitivity analysis for the effect of cooling unit speeds on PUE in OLS model trained for PUE prediction

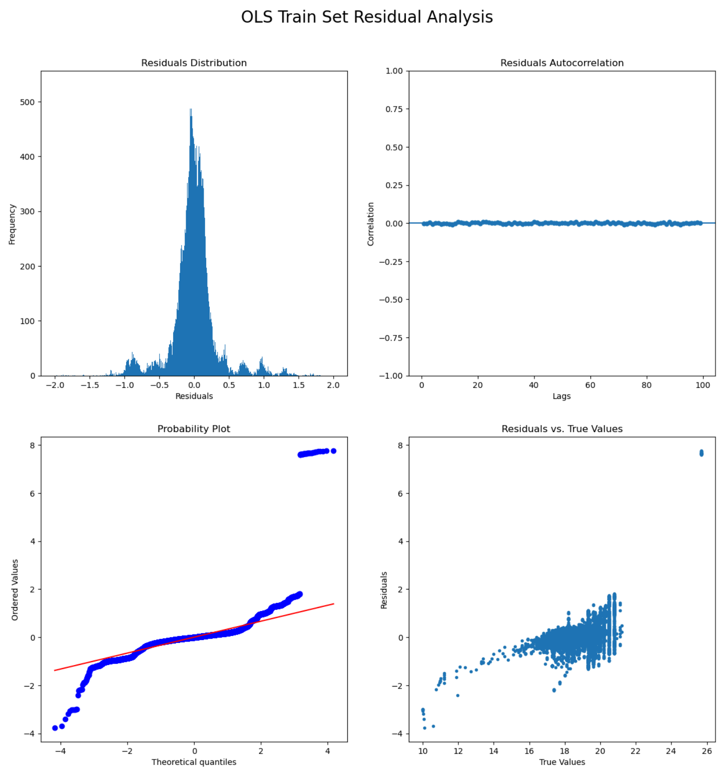


Figure 19: Residual analysis of OLS model for Sensor-I temperature prediction on training set

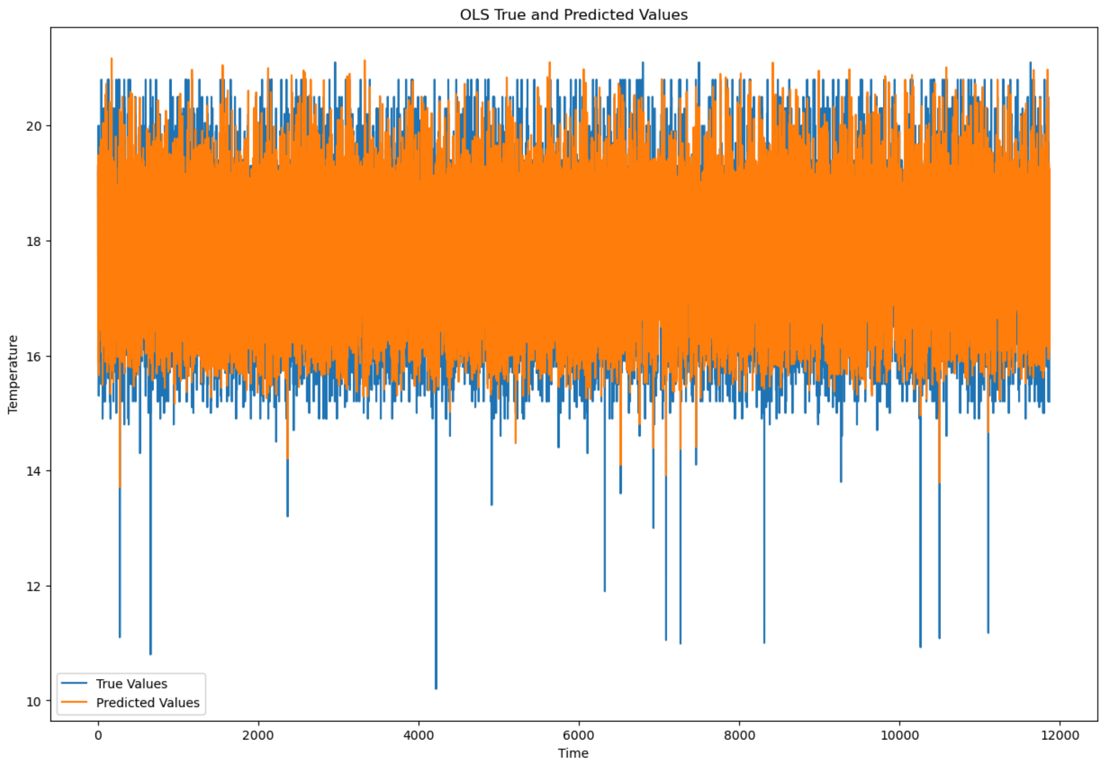


Figure 20: Predicted vs. actual Sensor-I temperature values in OLS model for test set

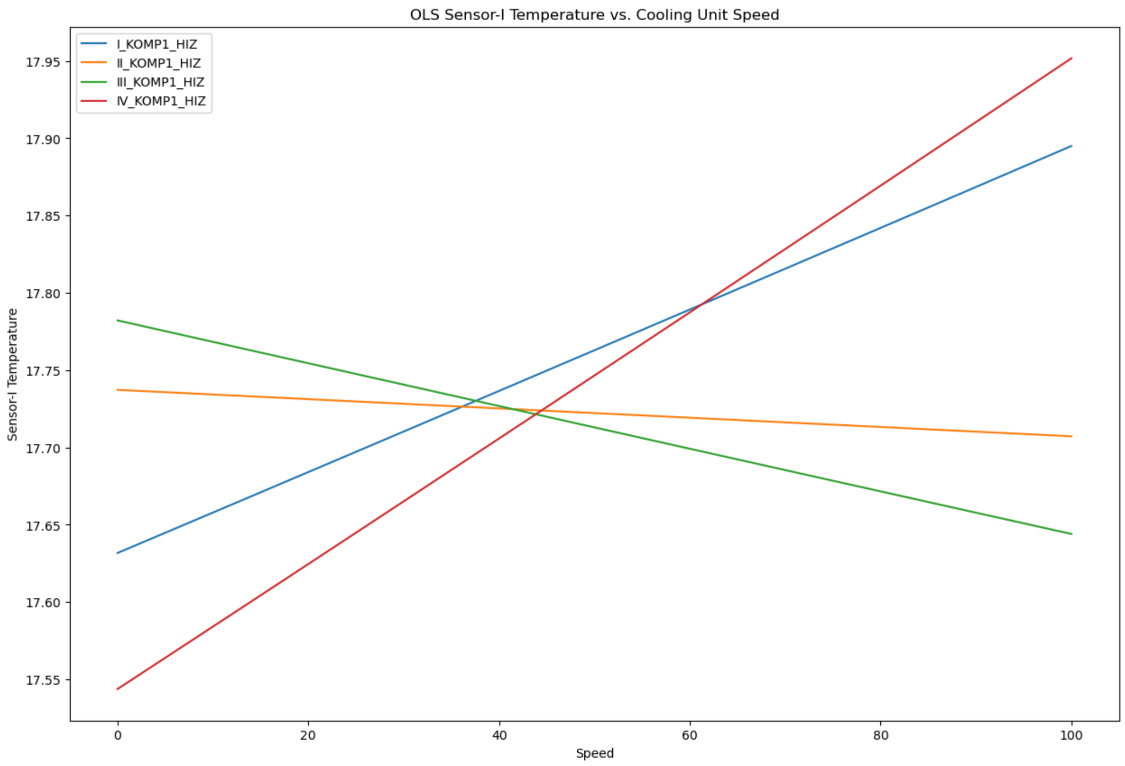


Figure 21: Sensitivity analysis for the effect of cooling unit speeds on Sensor-I temperature in OLS model trained for sensor-I temperature prediction

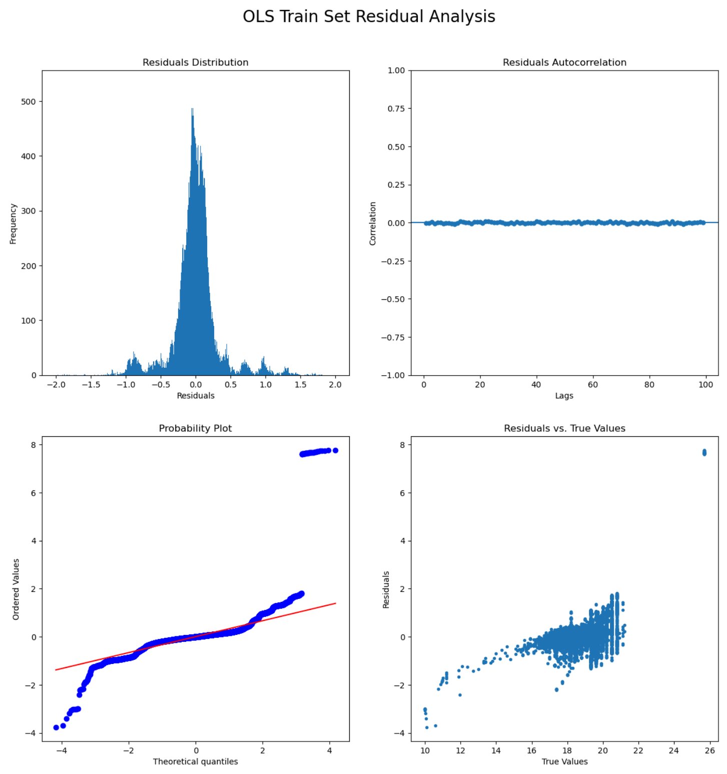


Figure 22: Residual analysis of OLS model for Sensor-II temperature prediction on training set

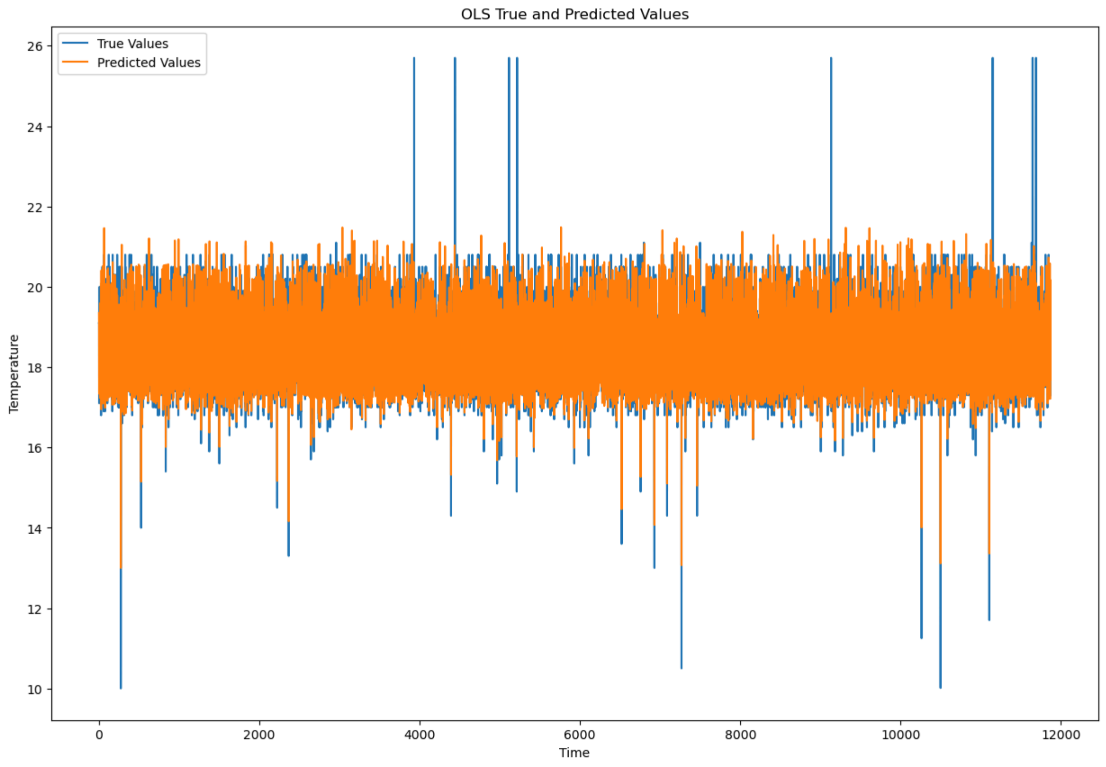


Figure 23: Predicted vs. actual Sensor-II temperature values in OLS model for test set

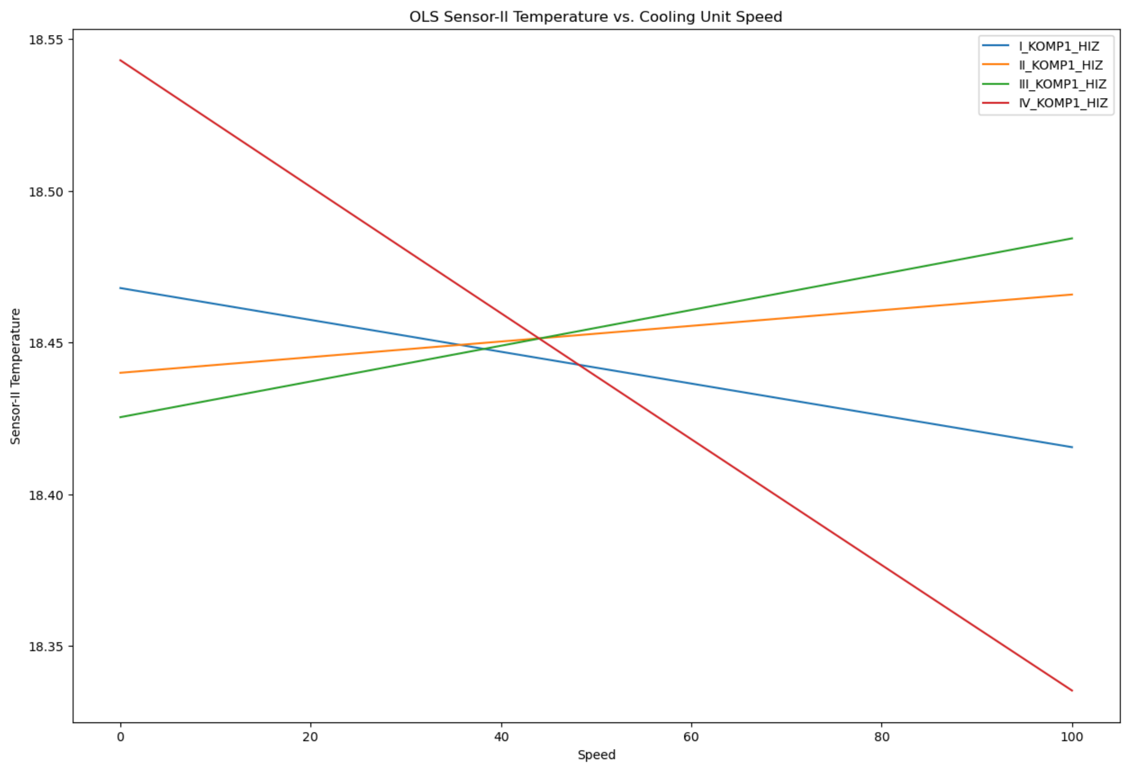


Figure 24: Sensitivity analysis for the effect of cooling unit speeds on Sensor-II temperature in OLS model trained for sensor-I temperature prediction

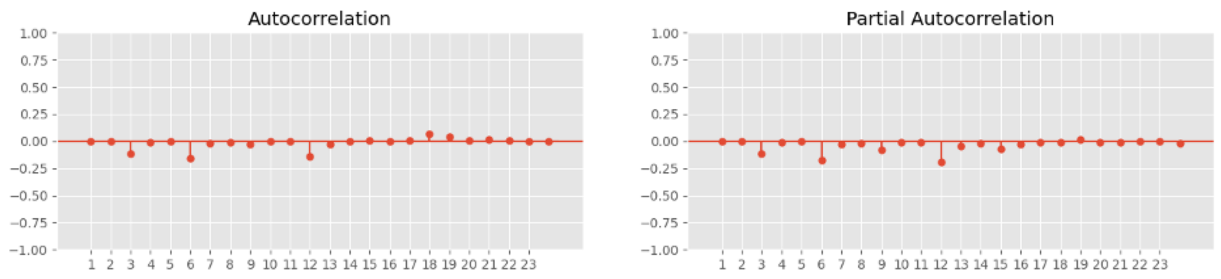


Figure 25: Autocorrelation and Partial Autocorrelation Plots of CH1_CIKIS_SIC (Differenced Data)

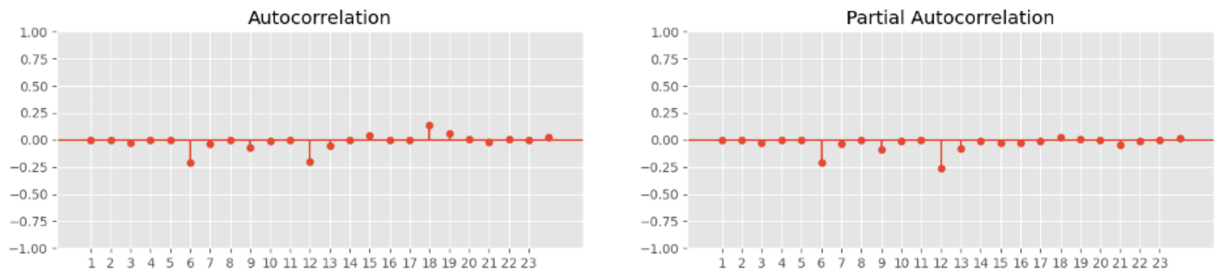


Figure 26: Autocorrelation and Partial Autocorrelation Plots of CH1_GIRIS_SIC (Differenced Data)

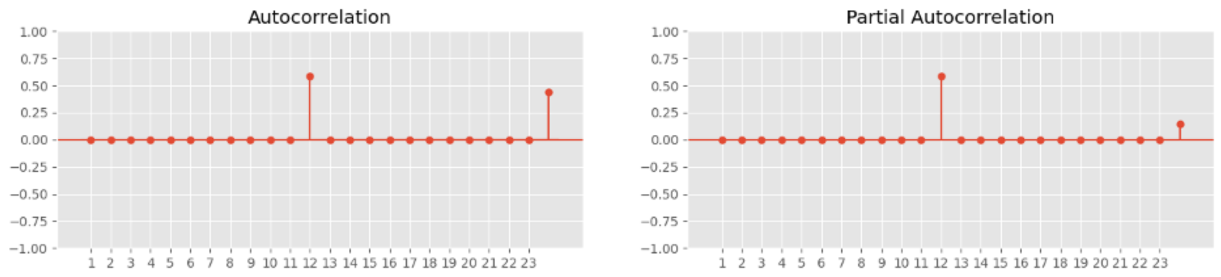


Figure 27: Autocorrelation and Partial Autocorrelation Plots of CH2_CIKIS_SIC (Differenced Data)

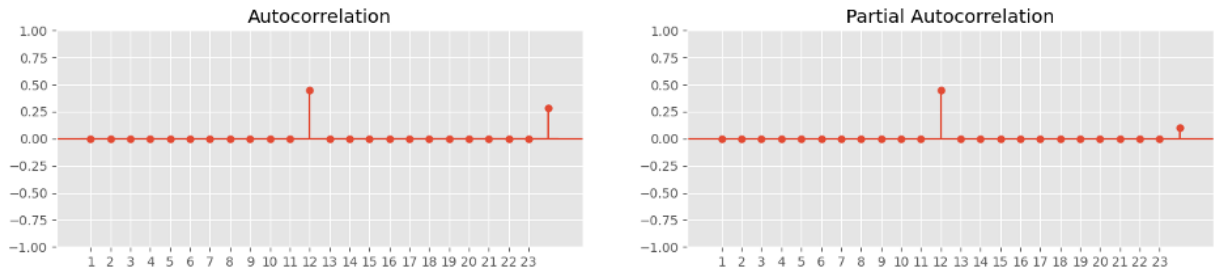


Figure 28: Autocorrelation and Partial Autocorrelation Plots of CH2_GIRIS_SIC (Differenced Data)

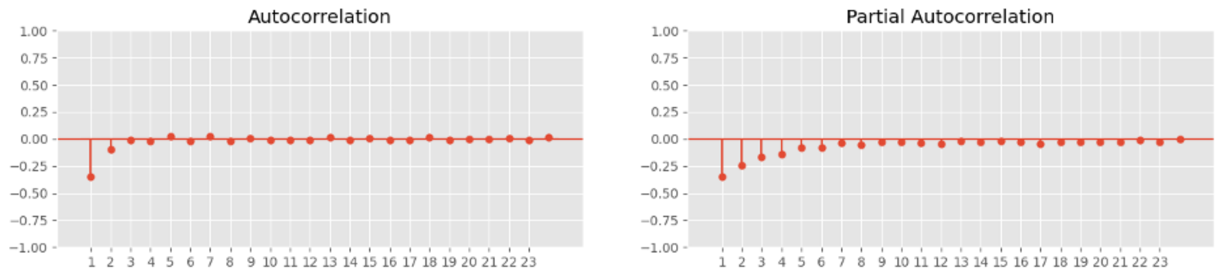


Figure 29: Autocorrelation and Partial Autocorrelation Plots of CH3_CIKIS_SIC (Differenced Data)

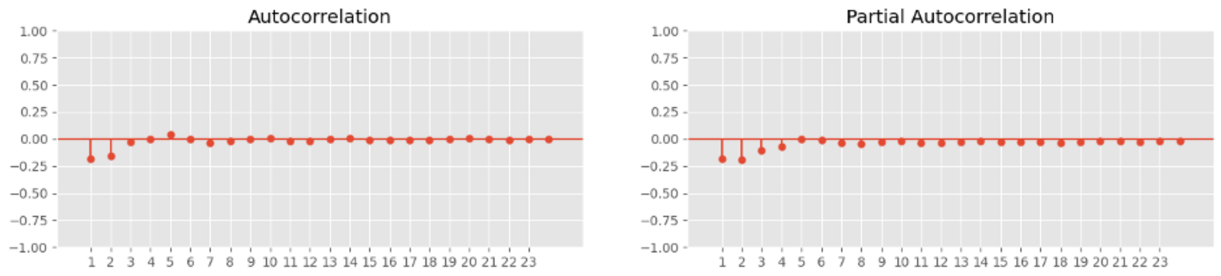
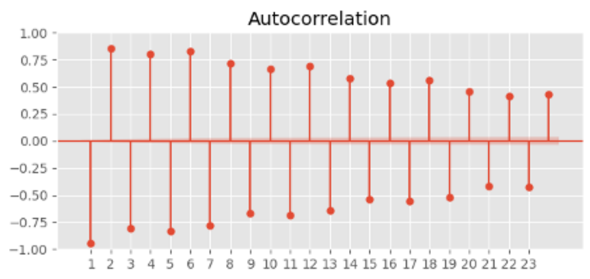


Figure 30: Autocorrelation and Partial Autocorrelation Plots of CH3_GIRIS_SIC (Differenced Data)



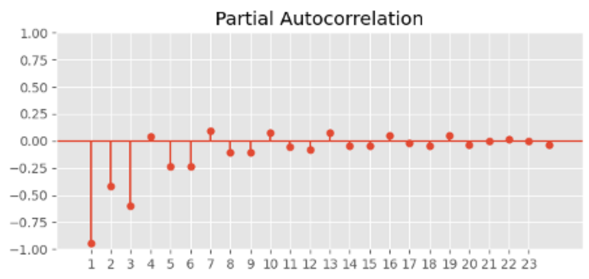


Figure 31: Autocorrelation and Partial Autocorrelation Plots of KS10_UDP_TUKETIM (Differenced Data)