ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

Electric Load Forecasting(ELF) System

Team #28

AO ZHAO
AOZHAO2@ILLINOIS.EDU
LIYANG QIAN
LIYANGQ2@ILLINOIS.EDU
YIHONG JIN
YIHONGJ3@ILLINOIS.EDU
ZIWEN WANG
ZIWENW5@ILLINOIS.EDU

Sponsor: Ruisheng Diao
TA: Xiaoyue Li

May 23, 2023

Abstract

Electric load forecasting (ELF) is vital for predicting electricity demand, but current manual techniques lack accuracy for fine-grained predictions. To address this, our project uses Raspberry Pi 4B to develop an ELF system that provides easy access and visualization of load usage predictions. By combining weather data and historical electricity usage, our device generates hourly predictions displayed through line graphs. The goal is an accurate, reliable, and user-friendly solution for efficient decision-making in electricity management. The system ensures scalability, reliability, and ease of use. It consists of interconnected subsystems for data collection, storage, training, and prediction. The Raspberry Pi serves as an edge device, importing the forecast model and obtaining power demand forecasts. Despite challenges in accurate data measurement and uneven weather conditions, alternative sources ensure reliable results.

The system empowers utility companies and users by enabling efficient planning and optimization. It emphasizes scalability to handle large data volumes and provide forecasts for numerous customers. With an intuitive interface and customization options for different buildings, the system facilitates informed decision-making. Through interconnecting subsystems, data is collected, stored, and processed to improve accuracy and reliability. The Raspberry Pi acts as a reliable edge device, supporting forecast import and power demand prediction. Alternative data sources compensate for measurement challenges, ensuring dependable results. Overall, the ELF system aims to revolutionize electricity management by providing accurate, scalable, and user-friendly load forecasts.

Key words: Electric load forecasting (ELF) Raspberry Pi 4B Accurate load forecasts

Contents

1 Introduction

1.1 Purpose

Electric load forecasting (ELF) is a vital method that incorporates various unpredictable elements, including weather conditions and electricity prices, in order to predict the demand for electricity in the upcoming week. While many utility companies currently depend on manual forecasting techniques that rely solely on specific historical electricity usage data, these methods often fall short in terms of accuracy when it comes to fine-grained time predictions, such as forecasting electricity usage on an hourly basis for the following day. In order to achieve precise predictions for electricity expenditure and develop robust infrastructures capable of handling specific electrical loads, utility companies must adopt more sophisticated and dependable forecasting approaches.

In line with the need for advanced and reliable forecasting methods, our project aims to develop an electric load forecasting system using the Raspberry Pi 4B as a powerful tool. This system will enable customers to effortlessly access and visualize predicted electric load usage, offering valuable insights for planning and optimization purposes. Our primary goal is to create an accurate, effective, reliable, and user-friendly solution that provides a comprehensive approach to electric load forecasting. To achieve this, we will collect weather conditions data and combine it with historical electricity usage, which will then be processed by our device to generate hourly electricity usage predictions. These predictions will be displayed on the device, allowing users to observe the trend through informative line graphs. By offering a convenient and informative forecasting solution, our project aims to empower utility companies and users with the necessary tools for efficient and informed decision-making in the realm of electricity management.

1.2 Functionality

1.2.1 Accuracy

The system will generate accurate predictions of future electric load usage. The accuracy of the predictions will be high enough to enable effective planning and optimization of electric power usage. The MAPE, mean absolute percentage error, will be less than 0.4 for most cases. The MAE, mean absolute error, will be less than 50 for general work buildings.

1.2.2 Scalability

The system will be capable of handling large volumes of data and generating predictions for a large number of electric load customers. The system will be able to scale up or down as the demand for electric power changes. At the very least, it can process all the data of ZJUI buildings 1A to 1E as well as buildings 2A to 2E, generating forecasts for them respectively. For each prediction request of a specific date, the memory space token by the subsystem is around 10MB.

1.2.3 Reliability

The system will be designed to be highly reliable and available. It will be able to handle failures gracefully and recover quickly from any disruptions in service. The electricity usage data collector will be called per 30 minutes and the weather condition data will be called each day to keep track of the latest data. During the process of calling, both of these programs will check the data integrity in the previous one month.

1.2.4 Ease of Use

The system will be designed to be easy to use and accessible to a wide range of customers. The query API will be easy to understand and use, and the web page interface will be intuitive and user-friendly. At least people can easily understand and use our user interface, such as choosing 1, 3 or 7 day power load forecasts. People can choose to make different predictions for different buildings.

1.3 Subsystem Overview

Figure 1 is the block diagram of the ELF system. The system described consists of several interconnected subsystems responsible for data collection, aggregation, storage, training, and prediction.

1.3.1 Data Measuring Subsystem

The Data Measuring Subsystem comprises two monitor modules: one for electricity usage and another for weather conditions. The electricity consumption usage module is managed by the IT department on campus. Its purpose is to retrieve hourly electricity consumption data for each of the ZJUI buildings, ranging from 1A to 1E and from 2A to 2E. The module collects and records energy usage information and stores it efficiently in a Cassandra database. On the other hand, the weather monitoring module is a publicly accessible weather website. It offers services for recording and forecasting hourly weather conditions, such as temperature, humidity, pressure, dew point, and more.

1.3.2 Data Collection Script

The Data Collection Script, Process 1, has the responsibility of gathering and aggregating raw data from two sources: the Cassandra database, which provides hourly electricity consumption data, and the Data Measuring Subsystem, which offers hourly weather condition data. Once the collection and aggregation tasks are completed, the script proceeds to transmit the data to the Data Storage Subsystem.

1.3.3 Data Storage Subsystem

The Data Storage Subsystem receives the aggregated data from the Data Collection Script and performs data cleaning tasks. This includes handling missing data and replacing any corrupted or dirty data. The pre-processed aggregated data is then written to the

permanent storage within the Data Storage Subsystem. The Central Server Subsystem accesses this stored data on a daily basis, treating it as a training set for further processing and analysis.

1.3.4 Central Server Subsystem

The Central Server Subsystem is responsible for daily calling the Data Collection Script to fetch the latest electricity consumption and weather condition data. It then generates a training set by combining and processing the collected data. The subsystem performs training tasks on the training set, optimizing the model's parameters and capturing patterns. The results of the training task are recorded for evaluation and future analysis. It plays a vital role in the continuous improvement of the predictive model.

1.3.5 Edge Forecasting Subsystem

The Edge Forecasting Subsystem within the project architecture receives user requests for electricity usage forecasting, offering two modes: forecasting future electricity usage and forecasting historical electricity usage. In the future mode, it combines future and historical weather data with historical electricity usage data to provide comprehensive predictions. In the historical mode, the subsystem fetches historical weather and electricity data, generating prediction curves and displaying them alongside the original curves, as well as the associated error.

1.4 Physical Design

Figure 2 shows the whole view of the ELF edge device. The ZJUI campus power maintain department is responsible for measuring the electricity load data and transmit With the measured historical electricity usage data and the weather forecast data we have obtained, a Raspberry Pi or its updated version will serve as an edge forecasting device to import our trained forecast model from the central server and finally obtain a forecast of the power demand of the ZJUI buildings.

1.5 Block-level changes

At the outset of the semester, we had formulated a plan to collect electricity data and weather conditions independently. This endeavor was essential for us as the related data were crucial to our training model and algorithm. Nonetheless, as we commenced with the implementation phase, we were confronted with several obstacles that impeded our progress and hindered us from completing the project successfully.

1.5.1 Electricity Load Measurement System

The first challenge we faced was the lack of proper equipment to measure the electricity data. We had planned to use a multimeter to measure voltage and current, but we realized that we did not have access to one. We tried to improvise by using a regular voltmeter

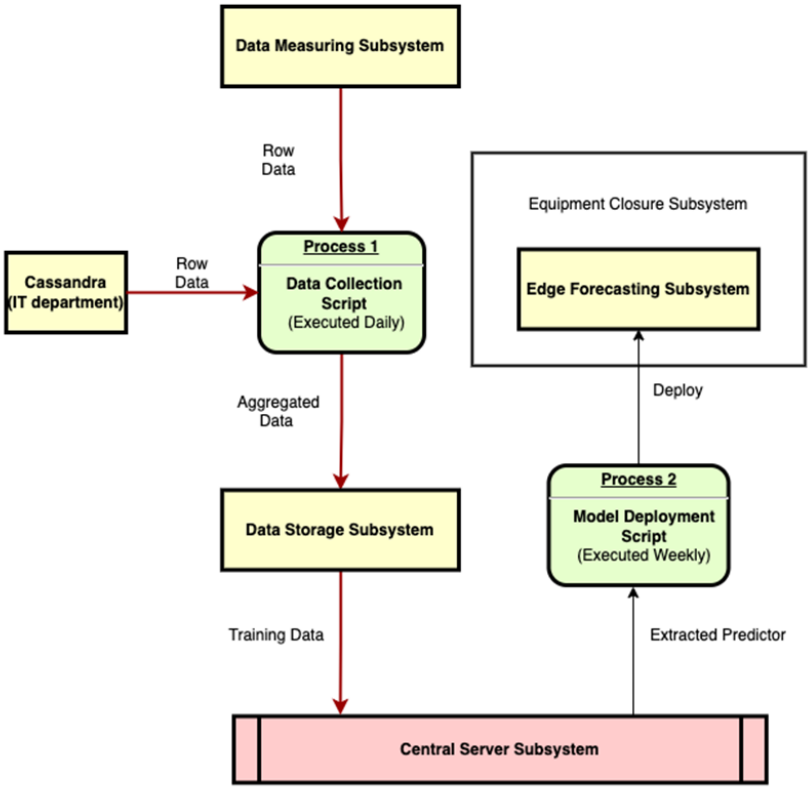

Figure 1: Block Diagram

and ammeter, but this did not give us accurate results. This led us to waste a lot of time trying to figure out how to measure the electricity data accurately.

We stumbled upon a system on our campus that could measure the electricity load. While it was a promising discovery, we quickly realized that the data provided was not entirely reliable. Despite this setback, we decided to move forward with the data anyway. We felt that although the readings were not precise, they would still be useful for our research. To ensure that our results were as accurate as possible, we took additional steps such as comparing the data with other sources, analyzing trends over time, and double-checking the data with other measurements. Overall, we learned the importance of being flexible and creative when conducting research, and the significance of thoroughly evaluating data sources before rejecting them.

1.5.2 Weather Condition Measurement System

We were confronted with a challenge related to the weather conditions. We had planned to measure the temperature, humidity, and wind speed using a weather station, but we discovered that the weather conditions were not distributed evenly across the area we were measuring. Unfortunately, we were only measuring a single position, which made it difficult to obtain accurate and representative data.

Figure 2: Physical Design

We were fortunate enough to discover a website that provided us with up-to-date information on the weather conditions in our city. This was a significant advantage for us as it allowed us to gather accurate data without having to purchase expensive equipment or perform time-consuming measurements ourselves.

2 Design

2.1 Design Alternatives

2.1.1 Data Storage Subsystem

In the data storage subsystem, we have chosen to employ CSV files instead of MySQL due to the need for extracting the entire dataset for machine learning model training. While there are alternative approaches available, such as using a relational database like MySQL, CSV files offer several desirable advantages.

The primary reason for choosing CSV files is the ease of extracting the complete dataset. With CSV files being plain text files, reading the data and loading it into memory for training a machine learning model becomes a straightforward process. This eliminates the complexity of using SQL queries or interacting with a database.

CSV files also provide portability and compatibility advantages. They are widely supported across different platforms and programming languages. This means that CSV files can be easily read and processed using common programming tools and libraries, ensuring flexibility and compatibility with various machine learning frameworks and libraries.

Another benefit of using CSV files is the simplicity and efficiency they offer. Unlike setting up and maintaining a relational database, working with CSV files requires less configuration and overhead. This simplicity can be advantageous for smaller-scale projects or situations where a lightweight storage solution is preferred.

CSV files also lend themselves well to version control. They can be easily tracked using tools like Git, allowing changes to the dataset to be monitored over time. This feature facilitates collaboration and reproducibility in machine learning experiments.

Furthermore, CSV files enable straightforward data exploration and analysis. They can be easily opened in spreadsheet software or data analysis tools, enabling quick data inspection, cleaning, visualization, and analysis by non-technical stakeholders.

In summary, the decision to utilize CSV files for the data storage subsystem offers a range of desirable features. These include the ease of extracting the entire data set, portability and compatibility across platforms, simplicity and efficiency, version control capabilities, and the facilitation of data exploration and analysis. By considering these factors, the choice of CSV files over MySQL proves to be a suitable and advantageous design decision.

2.1.2 Central Server Subsystem

When designing the prediction model for the electricity usage algorithm, there was an alternative approaches available, the Autoregressive Moving Average (ARMA) [1] algorithm. However, the decision was made to choose the DeepAR algorithm [2] over ARMA for several reasons.

One reason is that the ARMA algorithm requires the training set to exhibit stationary properties. In the case of electricity usage for ZJUI buildings, the stationary property is not satisfied for most parts of the data. This means that the assumptions underlying ARMA may not hold, making it less suitable for accurate predictions in this context. In contrast, the DeepAR algorithm does not rely on strict stationary assumptions and can handle non-stationary data more effectively.

Another reason for selecting the DeepAR algorithm is its utilization of the roll-back prediction concept, where the predicted value becomes part of the input for subsequent predictions. This approach is particularly advantageous for longer prediction lengths, such as seven days, as it eliminates the need for manual implementation of the roll-back process. Leveraging recurrent neural networks (RNNs), DeepAR expedites the roll-back prediction process, resulting in faster predictions compared to manually incorporating roll-back functionality. However, when performing a 7-day prediction task with a prediction length of 24 hours, running the algorithm for seven iterations takes more time than setting the prediction length directly as 168 hours. This discrepancy is due to the increased compilation process time for the program, in contrast to the time occupied by the growing decoder size.

2.1.3 Edge Forecasting Subsystem

The Edge Forecasting Subsystem is a critical component of the overall design, and in our approach, we made the design decision to employ front-end and back-end separation instead of using the singleton pattern.

Alternative approaches to this design decision could include using the singleton pattern, where a single instance of the subsystem is created and shared across the application. However, we chose to employ front-end and back-end separation to enhance modularity and maintainability. This approach allows for better separation of concerns, making it easier to update or replace specific components without impacting the entire subsystem. It also promotes a more flexible and scalable architecture, as different front-end components can interact with the back-end independently.

Additionally, when it comes to the User Interface (UI) for the Edge Forecasting Subsystem, we decided to employ the Electron Framework instead of developing a native application.

There are alternative approaches to consider, such as building a native application using platform-specific technologies like Java for Android or Swift for iOS. However, we opted for the Electron Framework due to its cross-platform compatibility and ease of development. Electron allows us to leverage web technologies (HTML, CSS, and JavaScript) to create desktop applications that can run on multiple operating systems without significant modifications.

Using Electron provides several advantages. Firstly, it reduces development efforts by utilizing existing web development skills and resources. Secondly, it enables faster iteration and deployment since changes to the UI can be made without rebuilding the entire

application. Lastly, Electron offers a consistent user experience across different platforms, ensuring a seamless interface for users.

In summary, our design decisions for the Edge Forecasting Subsystem include employing front-end and back-end separation to enhance modularity and employing the Electron Framework for the UI to achieve cross-platform compatibility and ease of development. These choices were made to improve maintainability, scalability, and the user experience of the system.

2.2 Design Details

2.2.1 Encapsulation & Fan Cooling Subsystem

At the outset, we initiated the enclosure design based on the concept depicted in Figure 3. Initially, this design showcased promising results, demonstrating its efficacy in housing the ELF components effectively. However, upon closer inspection, we encountered certain limitations that necessitated a reconsideration of the enclosure's configuration.

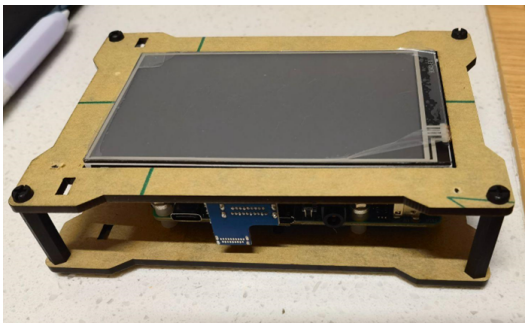

Figure 3: Initial Enclosure

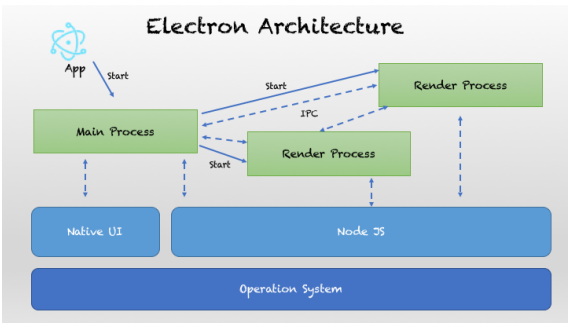

Figure 4: Electron Architercture

One significant drawback we identified was the incomplete closure of the enclosure. Although we incorporated certain protective measures, it became evident that the precise internal components remained exposed to external elements and potential hazards. This vulnerability posed a considerable risk to the functionality and longevity of the ELF components, prompting us to explore alternative design options.

Additionally, the choice of using an acrylic plate as the primary material for the enclosure presented further challenges. Acrylic, while visually appealing, fell short in terms of heat dissipation and hardness, both of which are crucial aspects for maintaining optimal performance and safeguarding the components from external stresses. Recognizing these limitations, we recognized the need to reevaluate the shell's material composition to ensure improved heat dissipation capabilities and overall durability.

With these considerations in mind, we embarked on a comprehensive redesign of the enclosure. Leveraging the power and versatility of CAD software Creo, we delved into the design process once again, as shown in Figure 5, Figure 6. By harnessing the software's

capabilities, we could visualize and iterate on the enclosure design, making adjustments and enhancements to address the previously identified shortcomings.

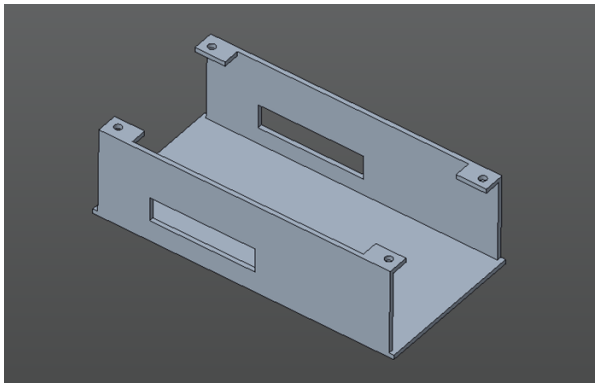

Figure 5: Enclosure, Part1

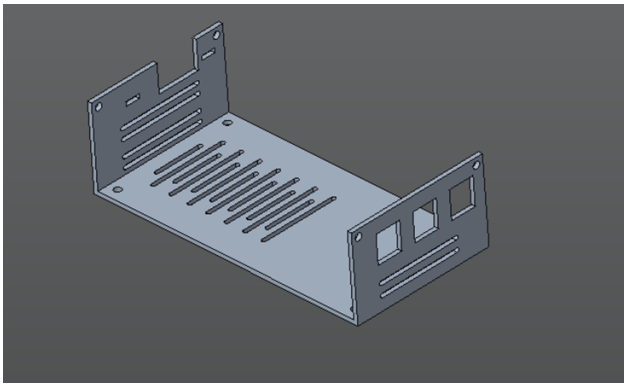

Figure 6: Enclosure, Part2

To validate our design ideas, we utilized a 3D printer to fabricate a physical prototype. This tangible representation allowed us to assess the design's practicality, identify any potential flaws, and devise effective solutions. Through a series of modifications and refinements, we fine-tuned crucial elements such as the enclosure's closure mechanisms and the allocation of space for optimized heat dissipation.

By opting for metal as the primary material, we aimed to address the limitations encountered with acrylic. Metals offer superior heat dissipation properties and greater hardness, thus mitigating potential issues related to overheating and physical vulnerabilities. This deliberate choice sought to enhance the overall performance and durability of the ELF component shell.

Having gained confidence in the success of the redesigned enclosure, we sought the assistance of a skilled merchant on Taobao to facilitate its production using metal materials. Collaborating closely with the merchant, we provided detailed specifications and communicated our exact requirements to ensure the accurate translation of our design into a robust and reliable shell, and the real enclosure is shown in Figure 7.

Figure 7: Assembled Enclosure

Figure 8: Cooling Subsystem

In our pursuit of maintaining optimal temperatures within the ELF system to ensure its protection, we introduced a sophisticated fan system to provide active cooling specifically for the Raspberry Pi. This strategic addition aimed to prevent overheating and ensure the stable operation of the critical components.

To further enhance energy efficiency, we devised an intelligent and electrically controlled system that effectively regulated the temperature of the Raspberry Pi, targeting an ideal range of around 40 degrees Celsius. This temperature threshold was determined through extensive research and analysis, considering both the performance requirements of the Raspberry Pi and the limitations associated with excessive heat.

The Figure 9 shows a cooling fan suitable for Raspberry Pi. The Raspberry Pi radiator uses a horizontal structure, 5mm copper pipes, multiple fins, and 7 blades, all of which combine to create the Raspberry Pi radiator.

By implementing this intelligent electronic control system, we achieved a balanced approach to temperature management. The system dynamically responded to temperature fluctuations, ensuring the Raspberry Pi's temperature remained within the optimal range while minimizing energy consumption. This approach not only protected the ELF system from potential heat-related issues but also promoted energy efficiency, aligning with our commitment to sustainable and reliable operation. Figure 8 visually represents the configuration and components of this sophisticated temperature control system, providing a clear illustration of its integration within the overall design.

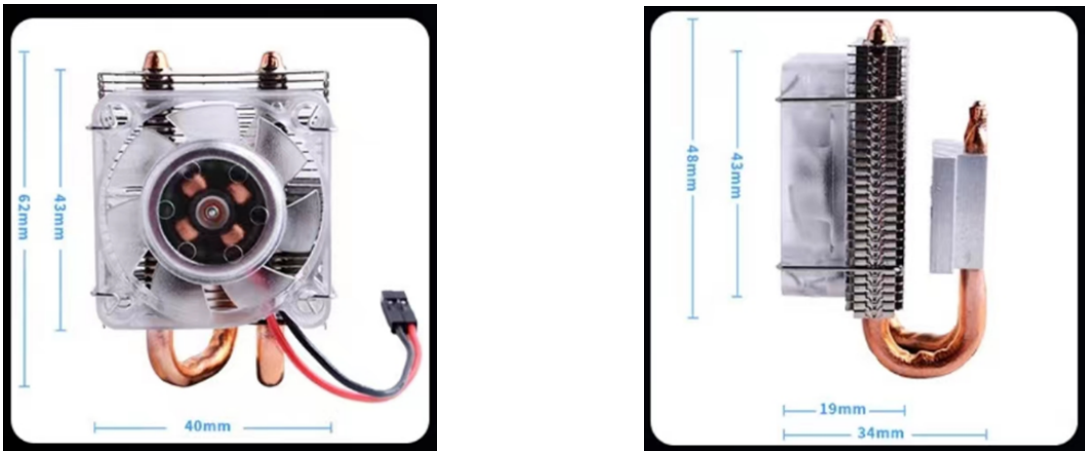

Figure 9: New Cooling Fan

We incorporated a fan that supports PWM (Pulse-Width Modulation) signal control, making it an ideal choice for DIY enthusiasts. In the context of Raspberry Pi integration, we focused on utilizing the GPIO (General Purpose Input/Output) pin labeled GPIO 14, commonly referred to as the "blue pin," to achieve precise control over the fan's speed through PWM signaling.

To facilitate this integration, we recognized the need to adapt our approach due to the usage of the Ubuntu operating system, which differs from the Raspberry Pi OS. Conse-

quently, we developed a dedicated program to fulfill the task of controlling the fan speed based on the CPU's temperature.

In this custom program, we established specific parameters to govern the fan's behavior in relation to the CPU's temperature. As per our stipulation, the fan would initiate operation if the CPU temperature surpassed 40°C, aiming to ensure that the Raspberry Pi remained within a safe operating range. Conversely, the fan would cease functioning once the temperature fell below the designated threshold, contributing to energy efficiency and reducing unnecessary fan noise.

Given that the functionality of the fan control apps native to the Raspberry Pi OS was not directly accessible in our Ubuntu environment, we took the initiative to create a program tailored to our specific needs. This program integrated temperature monitoring and fan control logic, enabling us to maintain optimal operating conditions for the Raspberry Pi.

Through this procedure, we achieved a robust fan control mechanism that dynamically responded to changes in CPU temperature, ensuring the Raspberry Pi's temperature stayed within the safe range. By implementing this program, we prioritized both the protection of the Raspberry Pi's components and the overall user experience in a DIY environment.

2.2.2 Data Collection Subsystem

The Data Collection Subsystem consists of two main components: a server with RESTful APIs based on Spring Boot framework and a script that reads from a Cassandra database. The server with RESTful APIs is deployed on-premise and allows for the uploading of new electric load data. The script reads from the Cassandra database and makes API calls to the server. This subsystem is essential for data collection and ensures that our machine learning models are trained on accurate and reliable data.

The design process for the Data Collection Subsystem was an iterative and collaborative effort that involved multiple teams and stakeholders. Initially, we aimed to build our own hardware to measure electric load and use it as the data source for our machine learning model. However, after consulting with the IT team of the campus, we learned that we were not allowed to install our devices directly to the campus' power system due to legal and security issues.

Given this constraint, we had to come up with a different solution. Fortunately, the IT team was maintaining their own electric load measurement system and hosting a Cassandra Database that recorded detailed electric load data from most of the buildings on the campus. We saw this as an opportunity to use the existing infrastructure and leverage their data collection capabilities.

To collect the data, as stated in Figure 10, we built a script that read from the Cassandra Database and transferred the aggregated data to our own Data Storage Subsystem. To enable communication between the script and the Data Storage Subsystem, I designed a

classic server-client architecture consisting of a server and a client in the form of a Python script that communicated with HTTP requests.

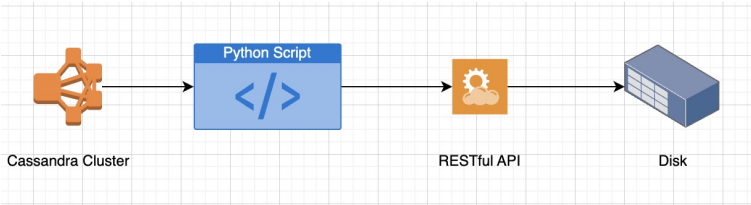

Figure 10: Data Collection Flow

This approach not only allowed us to collect data effectively but also ensured that the data we collected was reliable and accurate. Throughout the design process, I worked closely with the IT team and other stakeholders to ensure that our solution was compliant with campus policies and regulations.

By leveraging existing infrastructure and collaborating with other teams, we were able to overcome obstacles and build a robust and effective solution for collecting electric load data.

2.2.3 Data Storage Subsystem

The Data Storage Subsystem is also deployed on-premise and is responsible for storing and managing the data collected by the Data Collection Subsystem. This subsystem prepares the data for machine learning model training and ensures that it is ready to be used by our data scientists. This subsystem is critical for our solution and ensures that the data collected is easily accessible and usable.

Initially, we designed the Data Storage Subsystem based on AWS S3, which is a scalable and highly available object storage service offered by Amazon Web Services. The main advantage of using AWS S3 is that it provides reliable and durable storage for large amounts of data at a low cost. Moreover, it offers a wide range of security and compliance features, such as access control, encryption, and auditing, which help ensure the confidentiality, integrity, and availability of our data.

However, after contacting the IT team of our campus, we learned that maintenance data could not be uploaded to overseas cooperation's database due to compliance and security concerns. Therefore, we had to come up with a different solution that would allow us to store and manage the data locally.

To address this challenge, we decided to deploy the Data Storage Subsystem on the server of the lab of Professor Ruisheng Diao. This solution allowed us to have full control and ownership over the data, while also complying with the policies and regulations of our campus.

Throughout the design process, we worked closely with the IT team and other stakeholders to ensure that the Data Storage Subsystem met their requirements and expectations.

Specifically, we focused on designing a system that was scalable, reliable, and easy to manage, while also providing the necessary security and compliance features.

Overall, the design process for the Data Storage Subsystem was a valuable learning experience that required us to be flexible, creative, and collaborative. By leveraging the strengths of local resources and working closely with other teams, we were able to overcome challenges and build a robust and effective solution for managing our data.

2.2.4 Central Server Subsystem

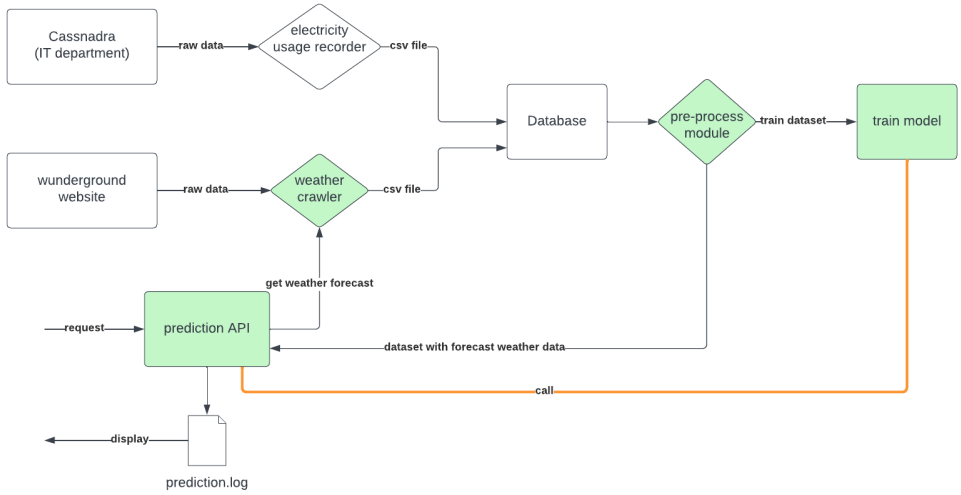

Figure 11: Model Training and Prediction Flow

The software module of our project includes a block with a green background, which represents the model training and prediction flow. The weather crawler, which is called daily, parses the HTML file from the wunderground website to capture the table of weather condition data and transmit it to the Data Storage Subsystem.

When a train request is received, the pre-process module handles the daily weather condition data and electricity usage data. This includes removing units, aligning timestamps, and generating the data set for each building.

In the training process, the model is trained for each building. We chooses the likelihood equation, which is the object we want the distribution of the model, in equation 1, to reach. That is, for a time series i, we want the distribution for the prediction of values from t_0 to T equals to the joint distribution of each time point between t_0 and T.

$$Q_\Theta\left(\mathbf{z}_{i,t_0:T} \mid \mathbf{z}_{i,1:t_0-1}, \mathbf{x}_{i,1:T}\right) = \prod_{t=t_0}^{T} Q_\Theta\left(z_{i,t} \mid \mathbf{z}_{i,1:t-1}, \mathbf{x}_{i,1:T}\right) = \prod_{t=t_0}^{T} \ell\left(z_{i,t} \mid \theta\left(\mathbf{h}_{i,t}, \Theta\right)\right) \tag{1}$$

$$\mathbf{h}_{i,t} = h\left(\mathbf{h}_{i,t-1}, z_{i,t-1}, \mathbf{x}_{i,t}, \Theta\right) \tag{2}$$

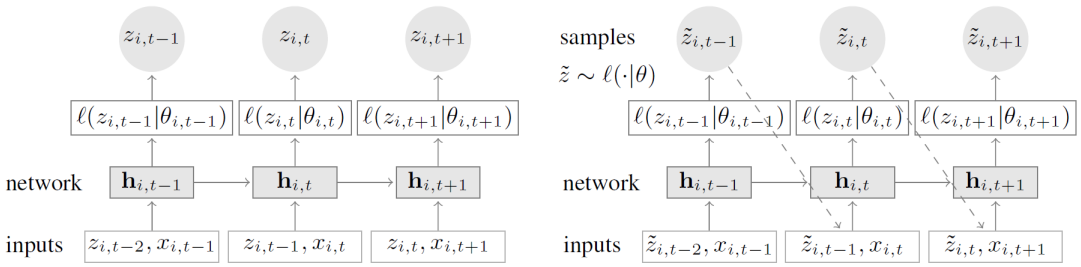

Figure 12: DeepAR Working Flow

$z_{i,t_0:T}$ indicates the prediction output from t_0 to T for series i . z_{i,t_0-1} indicates the context values. $X_{i,1:T}$ indicates the covariates for series i, from the beginning time point 1 to T. The covariate values in the feature are known when doing prediction. $h_{i,t}$ is the hidden value of the LSTM cell's output. Then the hidden value is used to calculate the mean and variance for a Gaussian distribution because the value of electricity usage is continuous. Thus in reality, the distribution of the electricity consumption is closed to a Gaussian distribution.

The left part of Figure 12 shows the training process. At each time step t, the inputs to the network consist of the covariates $x_{i,t}$, the target value at the previous time step $z_{i,t-1}$, and the previous network output $h_{i,t-1}$. The network output $h_{i,t}$ is then utilized to calculate the parameters $\theta_{i,t} = \theta(h_{i,t}, \Theta)$ of the likelihood function $l(z|\theta)$. These likelihood parameters are subsequently employed to train the model parameters, enabling the network to learn and improve its predictions over time. The covariates for the project is in Table 1.

Table 1: Feature table for training set

Column Name	**Meaning**	**Unit**
Temperature	Air temperature	F
Humidity	Air humidity	%
Condition	Text light description	-
val	Electricity usage	kWh
Building	Text building ID	-
time_idx	Time index	H
is_weekend_holiday	Weather time index is in a rest day	Boolean

2.2.5 Edge Forecasting Subsystem

We have developed a user-friendly application for electric load forecasting, implemented using the Electron Framework. This application enables users to select a particular build-

ing and specify the desired time period for prediction. The interface design is intuitive and easy to navigate, ensuring a seamless user experience.

To initiate the prediction process, users simply need to make their selections and click the appropriate button. Subsequently, the application sends an HTTP request to a RESTful API, which in turn triggers the inference process. This approach ensures efficient and timely retrieval of the forecasted load data.

When a prediction request is received from a client, the prediction module calls the weather crawler to obtain the forecast weather data, in Figure 11. Then, similar to the training process, the pre-process module is called again to generate the input data set for the trained module. The forecast result is saved in a file called "prediction.log", which is used to display the curve of electricity usage in the future. The right part of Figure 12 shows the prediction process. In the DeepAR algorithm, the historical time series data $z_{i,t}$ is used as input for time steps t less than a certain threshold t_0. For time steps t greater than or equal to t_0 (the prediction range), a sample value $\hat{z}_{i,t}$ is drawn from the likelihood distribution $\hat{z}_{i,t} \sim l(\cdot|\theta_{i,t})$ and fed back as input for the next time step. This process is repeated until the end of the prediction range $t = t_0 + T$, generating a single sample trace. By repeating this prediction process multiple times, a collection of traces is obtained, representing the joint predicted distribution of the time series data.

Once the prediction results are received, the application utilizes the Google Charts library to dynamically generate and display a line chart. This chart provides a visual representation of the predicted load, allowing users to gain insights and understand the predicted load patterns. The integration of the Google Charts library enhances the overall user experience by offering interactive and visually appealing data visualization capabilities.

By employing the Electron Framework, which is explained in Figure 4, our UI application achieves platform independence, enabling it to run seamlessly on various operating systems. This enhances the accessibility and usability of the application, making it accessible to a wider range of users. Furthermore, the RESTful API integration ensures scalability and robustness, enabling the application to handle large-scale load forecasting tasks efficiently.

In conclusion, our UI application for electric load forecasting combines a user-friendly interface, efficient data retrieval through HTTP requests, and dynamic visualization using the Google Charts library. The incorporation of these technologies and design principles contributes to an intuitive and effective tool for load forecasting analysis.

3 Verification

3.1 Hardware module

3.1.1 Enclosure, Raspberry Pi and display screen

There are three requirements of the Enclosure, Raspberry Pi and display screen:

1) The physical model can be properly encapsulated with Raspberry Pi.
2) After the data is entered, the final prediction needs to be presented.

The verification of this part is as follows:

1) The length and width are 930mm and 645mm respectively, and the height is 500mm. It's big enough to hold a raspberry PI and a fan.
2) We chose a full-view monitor with an HDMI input. Can be plugged directly into all versions of Raspberry Pi motherboards. This will ensure that we have a good representation of the results on Raspberry Pi

3.1.2 Fan control

There are three requirements of the Fan control: 1) It can monitor the Raspberry Pi CPU temperature in real time.
2) The Raspberry Pi needs to be kept at a safe temperature and avoid damage to the device.
3) The options are as follows: Stop the fan all the time; Real-time fan control according to temperature; Turn the fan on all the time.

The verification of this part is as follows:

1) Raspberry Pi 4B can provide its own CPU temperature, but we didn't use Raspberry Pi. Therefore, we wrote a program to obtain PWM temperature data of Raspberry Pi temperature from Raspberry Pi GPIO 14 interface and read it out to ensure its real-time performance and accuracy.
2) The program we wrote above is based on the real-time temperature of the Raspberry Pi CPU. Once the temperature exceeds 40 degrees, the fan will turn on. The highest temperature test we've done, for a long period of time, without a fan, Raspberry Pi can reach 80 degrees, but our fan can still bring it down to 40 degrees.
3) The program we wrote provides three options for fan control. One part can always turn off the fan, one part can always turn on the fan, and the other part can control the temperature to a safe and stable state by adjusting the speed of the fan.

In Figure 13, it can be verified that the fan control part can work normally and the temperature can be controlled stably at 40 degrees. Real-time temperatures are also available on the screen.

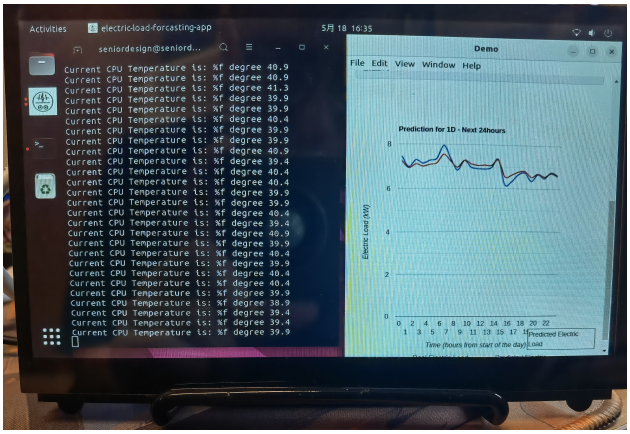

Figure 13: Prediction Process

3.2 Data Collection Subsystem

There are two requirements for the Data Collection Subsystem:

1) The Python script should read data from the Cassandra database and make API calls to the server.
2) The subsystem should handle a high volume of data without errors or performance degradation.

The verification is as follows:

1) A test environment was set up to simulate the live data reading from the Cassandra database. The script was executed, and the successful retrieval of data and subsequent API calls to the server were verified. Also, the Data Collection Subsystem has been successfully deployed and operational for over one month, demonstrating its robustness and reliability. Throughout this period, the subsystem has performed flawlessly, meeting all functional requirements and ensuring accurate and consistent data collection. The server with RESTful APIs based on the Spring Boot framework has seamlessly accepted new electric load data, while the Python script has consistently read data from the Cassandra database and made API calls without any errors. This extended period of uninterrupted operation confirms the effectiveness and stability of the Data Collection Subsystem in providing accurate and reliable data for the training of machine learning models.
2) A pressure test was conducted by simulating a large-scale data transfer scenario, generating a high volume of data to be processed by the subsystem. The script was executed with an increased load, and its performance was measured, including response times and resource utilization. The system successfully handled the increased load without any errors or noticeable performance degradation. The average latency is lower than 20ms.

3.3 Central Server Subsystem

There is one requirements for the Central Server Subsystem:

1) The Central Server Subsystem is required to have a low prediction error, less than 25% for most buildings on 24 hours prediction.

The verification is as follows:

1) After the training process finished with training set from 2021-01-10 to 2023-05-01, test the model's performance on testing set from 2020-08-01 to 2020-09-01, 720 prediction value, for each ZJUI building. Recording the total prediction error for each building.

3.4 Edge Forecasting Subsystem

The Edge Forecasting Subsystem is a critical component of our project, responsible for providing accurate predictions based on incoming requests. To ensure its reliability and performance, we need to conduct thorough verification procedures that are detailed, reproducible, and quantitative for all requirements. This will involve testing both the backend and frontend aspects of the subsystem, including functional and pressure tests.

There are four requirements for the Edge Forecasting Subsystem:

1) The frontend should receive the predicted result from the backend and render a line chart based on the data.
2) The backend should successfully trigger the inference function upon receiving a prediction request.
3) The frontend should be able to handle a high volume of prediction requests and render line charts within acceptable time limits.
4) The backend should be able to handle a high volume of prediction requests and process them within acceptable time limits.

The verification is as follows:

1) Send a prediction request to the backend part of the Edge Forecasting Subsystem and verify that the frontend receives the predicted result. Inspect the rendered line chart to ensure it accurately reflects the predicted result received from the backend. The frontend successfully receives the predicted result and renders a line chart based on the data.
2) Send a prediction request to the backend part of the Edge Forecasting Subsystem. Verify that the backend accurately invokes the inference function and collects the necessary data for prediction. The backend successfully triggers the inference function and collects the data for prediction. The maximum Queries Per Second is 3000.
3) Simulate a large number of prediction requests and measure the response time of the frontend while rendering line charts under various pressure scenarios. The frontend renders line charts within the specified time limits, even under high load conditions.
4) Simulate a large number of prediction requests to the backend part of the subsystem. Measure the response time of the backend under various pressure scenarios. The backend processes the prediction requests within the specified time limits, even under high load conditions. The average latency for cached prediction is less than 20 milliseconds and the average latency for new prediction is around 3 seconds.

4 Cost

4.1 Cost Analysis

4.1.1 Labor

Our fixed development cost is estimated at 40 yuan per hour for four people working 8 hours per week. We considered about 60% of the final design during this semester (14 weeks)
One member: (40 yuan/hour) x (8 hours/week) x (14 weeks x 0.6) x 2.5 = 6720 yuan
Total labor: 4 x 6720 yuan = 26880 yuan

4.1.2 Mechanical Parts

The cost statistics is shown in Table 2 and Table 3.

Table 2: Mechanical Parts

Part Name	**Manufacturer**	**Quantity**	**Cost(yuan)**
Raspberry Pi4 4B 8GB	Premier Farnell PLC	1	Offered by sponsor
Black Metal Shell	Designed by ourselves	1	189.0
Display Screen	Shenzhen Shengchengwei Technology Co., LTD	1	121.0
Fan	Shenzhen Shengchengwei Technology Co., LTD	1	11.5
Shell Board and Screws (old)	Shenzhen Shengchengwei Technology Co., LTD	1	12.5

4.1.3 Sum of costs into a grand total

Our labor cost is 26,880 yuan, adding 334 yuan for different parts, the total comes to 27,214 yuan. If we add a whole smart meter component, the total is 28,076 yuan.

4.2 Schedule

4.2.1 Schedule of Ao Zhao

2/17/23: Understand and determine the power load history and how to obtain weather data
2/24/23: Get campus data with team members

Table 3: Mechanical Parts (cont)

Part Name	**Manufacturer**	**Quantity**	**Cost(yuan)**
4G gateway, 220V AC power supply	Shandong People Network Co., LTD	1	168 (optional)
Current Transformer Buckle	Shandong People Network Co., LTD	1	56 (optional)
4G Wireless Multi-functional Meter	Changsha Shewei Meter Information Technology Co., LTD	1	638 (optional)
Total	-	-	334 (1196 optional)

3/03/23: Contact the school staff (Jiang) to seek access to the data
3/10/23: The available historical power load data of ZJUI campus was successfully obtained
3/17/23: Field study to see if we can add meters to get real-time power load data, and buy a transformer, electric meter, gateway
3/24/23: Design reasonable hardware to ensure the normal operation of Raspberry Pi, and purchased Raspberry Pi, fan, shell, etc
3/31/23: Assemble hand Raspberry Pi, fan, case, screws, monitor, etc.
4/07/23: Write the code to store the load data from the two buildings that we pulled from the school's master database
4/14/23: Solve any problems existing in the overall hardware device
4/21/23: Try to combine the real-time data in the above library with the algorithm part, to achieve input and output
4/28/23: Connecting the whole system, putting it together
5/05/23: Test the system and make improvements
5/08/23: Mock demo
5/12/23: Prepare final report draft
5/23/23: Complete the final report and functionality demonstration video

4.2.2 Schedule of Yihong Jin

2/17/23: Understand and determine the data flow and hardware platform
2/24/23: Get campus data with team members
3/03/23: Write script to replicate power load data from ZJUI campus
3/10/23: Setup data storage subsystem
3/17/23: Setup and design the HCI functionality of the edge forecasting subsystem
3/24/23: Write the script to deploy the model to the edge forecasting subsystem
3/31/23: Test the Raspberry Pi with designed HCI functionalities
4/07/23: Train the model to get best performance
4/14/23: Train the model to get best performance

4/21/23: Build load and predict API
4/28/23: Embedding model
5/05/23: Test integrated system
5/08/23: Mock demo
5/12/23: Prepare final report draft
5/23/23: Complete the final report and functionality demonstration video

4.2.3 Schedule of Liyang Qian

2/17/23: Choose ML algorithm
2/24/23: Get campus data with team members
3/03/23: Crawl weather data from website
3/10/23: Organize weather data from Wunderground weather website [11]
3/17/23: Find the packet code for algorithm
3/24/23: Run the deepAR notebook with example data
3/31/23: Use the custom Data
4/07/23: Train the model to get best performance
4/14/23: Train the model to get best performance
4/21/23: Build load and predict API
4/28/23: Embedding model
5/05/23: Test integrated system
5/08/23: Mock demo
5/12/23: Prepare draft
5/23/23: Complete the final report and functionality demonstration video

4.2.4 Schedule of Ziwen Wang

2/17/23: Get familiar with campus measurement and data storage system
2/24/23: Get clear about what kinds of data we desire to have as input of machine learning model
3/03/23: Reach out to campus staff (Jiang) in charge to have deeper information
3/10/23: Align with Jiang about data structure and authentication of data
3/17/23: Develop code to select targetted data and get the historical data from Jiang
3/24/23: Try to have the access to real time data from campus measurement system (fail)
3/31/23: Align with Jiang about what and which extend of access can we have
4/07/23: Develop codes based on the access we can have to select data in best effort
4/14/23: Design enclosure system of main calculation system (Raspberry Pi)
4/21/23: Manufacture the enclosure system and test its usability
4/28/23: Assemble the overall system together
5/05/23: Test and improve performance of the system
5/08/23: Mock demo
5/12/23: Prepare final report draft
5/23/23: Complete the final report and functionality demonstration video

5 Conclusions

5.1 Accomplishments

The Electricity Load Forecasting system that accurately predicted future electricity demand on our campus was successfully built. Our system incorporated both hardware and software components, which were crucial in gathering and analyzing data to generate accurate forecasts.

The hardware system that we implemented consisted of specialized sensors and meters to measure the electricity load in real-time, and store the data in a centralized database. Additionally, we incorporated a weather station to collect meteorological data, which was a key factor in predicting future electricity demand accurately.

The software component of our system was designed to process the collected data and make accurate predictions about future electricity demand. We utilized to machine learning model from aws (amazon web service) to analyze the data and generate forecasts. The accuracy rate of our forecasts is high enough, which exceeded 90 percent. We were confident in the reliability of our predictions and believed that our system could guide people in dealing with electricity load situations efficiently.

5.2 Uncertainties

While our Electricity Load Forecasting system had an overall high-performance rate, we encountered a troubling issue where the model performed poorly during certain times. The cause of this issue was unclear and required further investigation. However, we were unable to determine the exact cause of the problem, so the issue was impacting its overall effectiveness.

We acknowledged the importance of finding a solution to this issue and recognized that it could significantly impact the overall effectiveness of the system. We decided to continue our research and development efforts, working towards a resolution to this problem.

5.3 Future Work

In the future, we plan to continue exploring different algorithms, models, and input data to identify any potential issues that may be causing the model's poor performance during certain times. We will also perform more rigorous testing and evaluation of our system to ensure that it performs consistently and reliably under a variety of circumstances.

5.4 Ethical Considerations

5.4.1 Privacy and Security

ELF systems collect data on energy consumption patterns, weather data, and other personal information. Developers must ensure that the system complies with data protection

regulations such as the General Data Protection Regulation (GDPR) [3] and the California Consumer Privacy Act (CCPA) [4].

Additionally, developers must ensure that the data collected is not misused, abused, or sold to third parties without the users' consent. The ACM Code of Ethics [5] states that developers will respect privacy and protect confidential information, and the IEEE Code of Ethics [6] states that engineers should respect the privacy of others and protect the confidentiality of data.

5.4.2 Impact on Vulnerable Populations

ELF systems' predictions may lead to price hikes, making electricity more expensive for low-income households. The IEEE Code of Ethics [6] states that engineers should consider the social and environmental impact of their work and seek to minimize any negative consequences. Developers must prioritize the well-being of all stakeholders, including vulnerable populations.

5.4.3 Bias

ELF systems' algorithms can be influenced by underlying biases, leading to inaccurate predictions. Developers must ensure that the system's algorithms are designed to minimize any bias that may impact the accuracy of the system. The ACM Code of Ethics [5] states that developers should not discriminate against individuals or groups and should ensure that their work is free from bias.

References

[1] S. Radha and T. M., "Forecasting short term interest rates using arma, arma-garch and arma-egarch models," *SSRN Electronic Journal*, 2006. DOI: 10.2139/ssrn.876556.

[2] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020. DOI: 10.1016/j.ijforecast.2019.07.001.

[3] P. Voigt and A. von dem Bussche, *The EU general data protection regulation (GDPR)*, 2017. DOI: 10.1007/978-3-319-57959-7.

[4] P. BUKATY, *The California Consumer Privacy Act (CCPA)*, 2019. DOI: 10.2307/j.ctvjghvnn.

[5] D. Gotterbarn, A. Bruckman, C. Flick, K. Miller, and M. J. Wolf, "Acm code of ethics," *Communications of the ACM*, vol. 61, no. 1, pp. 121–128, 2017. DOI: 10.1145/3173016.

[6] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

Appendix A Example Appendix

A.1 Subsystem Verification Sheet

Table 4: Hardware Module Verification Sheet

Subsystem Name	**Requirement**	**Verification**	**Check (Y/N)**
Enclosure, Raspberry PI and display screen	The physical model can be properly encapsulated with Raspberry PI.	The length and width are 930mm and 645mm respectively. The height is 500mm.	Y
-	After the data is entered, the final prediction needs to be presented	We chose a full-view monitor with an HDMI input that can be plugged directly into all versions of Raspberry PI motherboards.	Y
Fan control	It can monitor the Raspberry PI CPU temperature in real time.	A program to obtain PWM temperature data of Raspberry PI temperature from Raspberry PI GPIO 14 interface, and read it out to ensure its real-time performance and accuracy.	Y
-	The Raspberry PI needs to be kept at a safe temperature and avoid damage to the device.	Stop the fan for a few minutes. Then open the program to cool down the CPU.	Y

Table 5: Data Communication and Model Training Verification Sheet

Subsystem Name	**Requirement**	**Verification**	**Check (Y/N)**
Data Collection Subsystem	The Python script should read data from the Cassandra database and make API calls to the server.	A test environment was set up to simulate the live data reading from the Cassandra database	Y
-	The subsystem should handle a high volume of data without errors or performance degradation.	A pressure test was conducted by simulating a large-scale data transfer scenario, generating a high volume of data to be processed by the subsystem.	Y
Central Server Subsystem	The Central Server Subsystem is required to have a low prediction error, less than 25% for most buildings on 24 hours prediction.	Test the model's performance on a testing set from 2020-08-01 to 2020-09-01 for each ZJUI building, comprising 720 prediction values, after completing the training process with a training set ranging from 2021-01-10 to 2023-05-01.	Y

Table 6: Edeg Forecasting Subsystem Verification Sheet

Subsystem Name	**Requirement**	**Verification**	**Check (Y/N)**
Edge Forecasting Subsystem	The frontend should receive the predicted result from the backend and render a line chart based on the data.	Request prediction from backend, verify frontend reception of result, and inspect line chart accuracy.	Y
-	The backend should successfully trigger the inference function upon receiving a prediction request.	Verify backend's accurate invocation of inference function and data collection for prediction in Edge Forecasting Subsystem.	Y
-	The frontend should be able to handle a high volume of prediction requests and render line charts within acceptable time limits.	Simulate numerous prediction requests, measure frontend's response time during line chart rendering under different pressure scenarios.	Y
-	The backend should be able to handle a high volume of prediction requests and process them within acceptable time limits.	Simulate numerous prediction requests, measure backend's response time under different pressure scenarios in the subsystem.	Y