Project Report

1. INTRODUCTION

1.1 Project Overview:

The Smart Home Temperature Prediction Project is a cutting-edge initiative designed to transform residential living through the integration of smart home technologies and advanced machine learning.

The project is about the development of a smart home temperature prediction system. There are many connected devices in a smart home, including door locks, TVs, thermostats, home monitors, cameras, lights, and appliances. A central point can be used to access and control these devices. The evolution of smart Wi-Fi thermostats beyond conventional functions is a key focus of the project. Machine learning is incorporated into these thermostats to adapt to occupant behaviors. Occupants control their comfort remotely, providing a dynamic and personalized thermal environment.

The project uses machine learning regression and Flask integration to empower residents with the ability to personalize and remotely manage their home’s thermal environment. The smart Home Temperature Prediction project envisions a harmonious intersection of technology and sustainable living, promising to redefine the dynamics of energy-efficient and user-centered living spaces.

1.2 Purpose:

The purpose of the project is more than one. It seeks to enhance the energy efficiency of residential spaces by using smart home technologies and machine learning. The goal of the project is to develop models for room temperature and heating/cooling demand to improve the efficiency of HVAC systems. Beyond the immediate goal of energy efficiency, the project addresses the substantial energy consumption associated with heating, ventilating, and air conditioning systems. It contributes to a reduction in energy consumption in buildings and reduces greenhouse gas emissions.

The project places a lot of emphasis on user comfort and control. The system allows residents to personalize their home's thermal environment with the help of machine learning and Flask integration. The deployment of the application on the IBM cloud ensures widespread accessibility and accommodates varying user needs, making a significant advancement in building energy management. The project serves to advance the state-of-the-art in smart home technologies for sustainable and efficient living spaces.

The purpose of this project is to redefine the very fabric of residential living, offering a harmonious blend of energy efficiency, sustainable living, and enhanced user well-being.

2. LITERATURE SURVEY

2.1 Existing problem:

There are several challenges that motivate the need for the Smart Home Temperature Prediction project. There is an issue with the energy use in residential buildings. Conventional thermostats don't have the ability to respond to user behaviors and environmental changes, leading to unnecessary energy consumption and decreased efficiency. The initiation of the Smart Home Temperature Prediction project is necessitated by the persistent challenges present in the contemporary environment of smart home technologies.

There is a critical concern about the inefficient use of energy within residential buildings. Traditional thermostats fail to respond to user behaviors and changing environmental conditions. Increased energy consumption is a result of this inadequacy.

2.2 References:

1) https://ieeexplore.ieee.org/document/10174484

2) https://ieeexplore.ieee.org/document/9689786

3)https://ieeexplore.ieee.org/document/9154007

2.3 Problem Statement Definition:

The project's problem statement is to develop models that learn from smart thermostat data. An intelligent system capable of accurately forecasting room temperature and heating/cooling demand is the objective. By doing so, the project aims to improve the efficiency of the system and contribute to the global effort to make the world a better place. The goal of the smart home temperature prediction project is to tackle the inefficiencies in current heating, ventilating, and air conditioning (HVAC) systems, transforming them into adaptive and energy-efficient solutions for the benefit of residential spaces and the broader environmental landscape.

The problem of inefficient energy utilization in residential buildings is caused by inefficiencies in the operation of the air conditioning system. Conventional thermostats are not able to respond to user behaviors and changing environmental conditions. This inadequacy results in unnecessary energy consumption and a reduction in system efficiency. The urgent need to mitigate greenhouse gas emissions is compounded by the escalating global demand for energy. The need for intelligent solutions to improve energy efficiency in residential spaces has never been more pressing.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

The Empathy Map Canvas for the Smart Home Temperature Prediction project provides a nuanced understanding of the users' needs, challenges, and emotional states. Users want a system that is responsive to their lifestyles and preferences, which is why they are frustrated with the current lack of control over their home's temperature. Concerns about high energy bills and the environmental impact of inefficient heating, ventilating, and air conditioning systems weigh heavily on their minds, reflecting a desire for a solution that meshes with their sustainable values.

Users feel a profound need for greater comfort in their living spaces and are enthusiastic about embracing innovative technologies that contribute to a more sustainable lifestyle. The insights gathered from the Empathy Map Canvas lay the foundation for the proposed solution, ensuring that the Smart Home Temperature Prediction System is not only technologically advanced but also deeply resonated with the end-users.

Reference:

https://app.mural.co/t/vitap7264/m/vitap7264/1699863763623/1e34bf66337fd0aeb2d71de0eba60a0672be1741?sender=ud74c88dfe4d953863c363232

3.2 Ideation & Brainstorming:

A comprehensive solution for the Smart Home Temperature Prediction project has been envisioned to address the challenges and desires of users. The proposed system uses machine learning to develop models for room temperature and heating/cooling demand. The system learns historical patterns, user behaviors, and environmental changes by using data from smart Wi-Fi thermostats. The user interface developed using a flask ensures a user-friendly experience, allowing residents to remotely control and monitor their home's temperature based on individual preferences.

The goal is to strike a balance between energy efficiency and a comfortable living environment. Integrating with other smart home devices creates a cohesive system. The deployment of the IBM cloud ensures efficient utilization of resources. User feedback and evolving environmental conditions can be incorporated into a continuous improvement loop. The goal of enhancing overall living experiences in smart homes is one of the goals of Smart Home Temperature Prediction System.

Reference:

https://app.mural.co/t/vitap7264/m/vitap7264/1700296466447/95cc59455d40062883ab738920a2d6c7d76ab549?sender=ud74c88dfe4d953863c363232

4. REQUIREMENT ANALYSIS

4.1 Functional requirement:

1) Data Collection:

Data collection involves gathering information from smart wi-fi thermostats. Data reflecting temperature variations, heating and cooling patterns, and user preferences are generated by these sensors. Real-time information is crucial for training and modeling. The data collection process helps ensure the system's ability to understand and respond to the nuances of temperature fluctuations within smart homes, thereby facilitating the creation of a sophisticated and efficient Smart Home Temperature Prediction System.

Historical temperature records, humidity levels, and possibly user interactions with the thermostat are some of the variables collected. This rich dataset provides insights into the interplay of environmental factors and occupant behaviors. The continuous and automated nature of data collection ensures a comprehensive representation of the smart home's climate dynamics, enabling the system to learn, adapt, and make informed predictions.

2) Data Pre-processing:

Data pre-processing is the gateway to ensure the quality and reliability of the data that fuels predictive models and is a fundamental step in the development of a smart home temperature prediction system. In the context of this project, the initial dataset collected from smart Wi-Fi thermostats must undergo pre-processing to address various challenges inherent in real-world data. This involves handling missing values, and outliers, and scaling the data to create a refined dataset that is compatible with machine learning model training.

data pre-processing for Smart Home Temperature Prediction is a comprehensive approach aimed at refining raw data into a clean, standardized, and reliable dataset. This process lays the groundwork for model development and training, which in turn contributes to the system's ability to provide accurate and adaptive temperature predictions in smart home environments.

3) Model Development and Training:

The Model development and training phase is where the collected data transforms into actionable insights. The aim of the project is to train regression models, including but not limited to Linear Regression, Random Forest, LightGBM, and Xgboost. The primary goal is to create robust and adaptive models that can predict room temperatures and heating/cooling demands in smart home environments.

Model training is not a one-size-fits-all endeavor; it requires experimentation and fine-tuning to identify the most suitable algorithm for the specific characteristics of the smart home temperature prediction task. A set of trained models contributes to the system's ability to make accurate and adaptive predictions. The groundwork for the subsequent stages of model evaluation, selection, and deployment is laid in this phase.

4) Model Evaluation:

The evaluation of the trained regression models in the Smart Home Temperature Prediction project is important. the evaluation of each model, including linear regression, random forest, etc takes place after the development and training phase. The metrics show the difference between predicted and actual temperature values. The models' ability to generate predictions that closely align with the observed data indicates better performance. understanding how well each model generalizes to unseen data and adapts to the dynamic nature of smart home temperature patterns is essential for this evaluation process.

The evaluation phase serves as a basis for model selection, guiding the choice of the most accurate and reliable algorithm for deployment. It ensures that the chosen model aligns with the project's goals of providing precise temperature predictions and optimizing HVAC systems for energy efficiency. The insights gained from model evaluation contribute to informed decision-making, laying the groundwork for the final stages of model integration, deployment, and user interaction within the smart home ecosystem.

5) Model Selection and Deployment:

The culmination of the smart home temperature prediction project is model selection and deployment, where the most effective regression model is chosen for integration into the system. The model that shows the highest accuracy and reliability in predicting room temperatures and heating/cooling demand is selected following a rigorous evaluation process. Once the optimal model is identified, it is saved in a portable format, such as the interface through which users can interact with a smart home temperature prediction system is established when the chosen model is connected to a flask application.

The deployment phase involves making the accessible model on a cloud platform. Real-time predictions and user interactions are dependent on this step. Residents can remotely monitor and control their home temperature settings through a web or mobile application thanks to the integration with Flask. The transformation of machine learning insights into practical applications is represented by the selected model’s deployment. As users interact with the system, it continuously learns from real-time data and user behaviors. The deployment phase marks the technical realization of the smart home temperature prediction project and underscores its potential to enhance energy, efficiency, occupant comfort, and overall sustainable living.

6) Flask Integration:

The smart home temperature prediction system can be brought to life with the help of flask integration. Users can remotely monitor and control the temperature settings in their smart homes with the help of Flask, a web framework for Python. Linear Regression, Random Forest, LightGBM, and Xgboost are examples of models that can be accessed through a web or mobile interface. Users can input preferences and view real-time temperature predictions. It's an ideal choice for creating interactive and dynamic web applications that connect with the underlying machine-learning model because of its lightweight and modular design.

The bridge between the machine learning algorithms and the end-users is provided by the flask application. Residents can use the power of predictive analysis for climate control in their homes with minimal effort. The smart home temperature prediction system is transformed from a behind-the-scenes machine learning application into a practical, user-centered solution with the help of flask integration. It allows residents to take control of their home environment, fostering a more personalized and energy-efficient approach to temperature management within the context of smart homes.

7) Application Deployment and Testing:

The final phases of the realization of the smart home temperature prediction project include application deployment and testing. The entire system, including the integrated flask application and the selected regression model, is deployed on a cloud platform. Users can interact with the system in real-time with this deployment. Before deployment, rigorous testing protocols are used to verify the security of the application. Testing includes the responsiveness of the web interface, the accuracy of temperature predictions, and the system's robustness in handling user interactions. Ensuring a seamless and error-free user experience is achieved through comprehensive testing. Users can access it remotely once the system passes the testing phase. Residents can use the web or mobile interface to input preferences and observe the system's temperature predictions. Continuous testing during deployment allows for the identification and resolution of any potential issues that may arise in a real-world environment, ensuring the system's stability and performance.

The bridge between the development environment and practical use within smart homes is represented by the deployment and testing phase. It shows the projects transformation from a conceptual idea to a solution the residents can engage with. The success of the smart home temperature prediction system can we attributed to the deployment and testing processes.

4.2 Non-Functional requirements:

1) Usability:

The Flask web interface must be user-friendly and intuitive, ensuring that residents can easily navigate the application, input preferences, and understand temperature predictions.

2) Performance:

The system should provide real-time temperature predictions, with response times for user interactions kept to a minimum to ensure a seamless and responsive user experience.

3) Scalability:

The system should be designed to scale seamlessly as the number of users and smart home devices increases. This ensures that performance is maintained even with a growing user base.

4) Reliability:

The Smart Home Temperature Prediction system should be highly reliable, with minimal downtime. This reliability is crucial for users who rely on the system for continuous climate control in their homes.

5) Security:

The user data, including temperature preferences and historical patterns, must be encrypted, and securely stored. Access controls should be in place to prevent unauthorized access to sensitive information.

6) Maintainability:

The system should be designed with modularity and code maintainability in mind. This facilitates future updates, enhancements, and the incorporation of new features without significant disruptions.

7) Compatibility:

The application should be compatible with a variety of devices and web browsers to accommodate different user preferences and ensure widespread accessibility.

8)Data Privacy:

Stringent measures should be in place to protect user privacy. The system should adhere to data protection regulations, and user data should only be used for the intended purpose of temperature prediction.

9)Auditability:

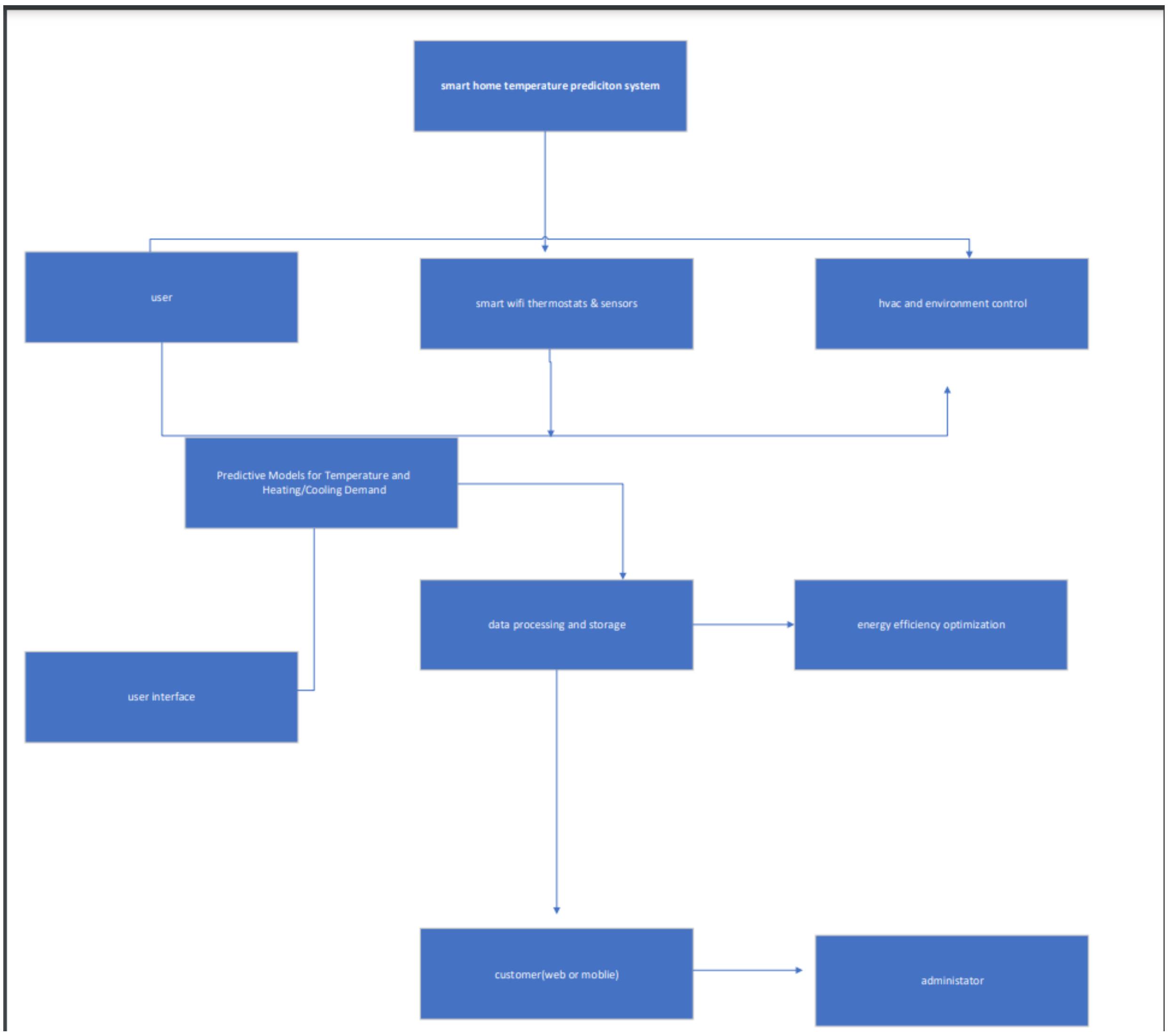
The system should have mechanisms in place to log and audit user interactions, aiding in troubleshooting, monitoring system usage, and ensuring accountability.

10)Interoperability: The Smart Home Temperature Prediction system should be designed to seamlessly integrate with existing smart home ecosystems, ensuring compatibility with various IoT devices and platforms.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.


smart home temperature prediciton system
user
smart wifi thermostats & sensors
hvac and environment control
Predictive Models for Temperature and Heating/Cooling Demand
data processing and storage
energy efficiency optimization
user interface
customer(web or moblie)
administator

User Stories:

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, and password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
Customer (Mobile user)	Registration	USN-2	As a user, I will receive a confirmation email once I have registered for the application.	I can receive confirmation email & click confirm.	High	Sprint-1
Customer (Mobile user)	Registration	USN-3	As a user, I can register for the application through Facebook.	I can register & access the dashboard with Facebook Login	Low	Sprint-2
Customer (Mobile user)	Registration	USN-4	As a user, I can register for the application through Gmail.	I can register & access the dashboard with Gmail Login.	Medium	Sprint-1

Customer (Mobile user)	Dashboard	USN-5	As a user, I can view my personalized dashboard upon registering.	I can see my account information, recent activities, and recommendations.	High	Sprint-1

Customer Care Executive	Customer Query Management	USN-6	As a customer care executive, I can view and respond to customer queries and complaints.	I can access a dashboard with a list of customer queries and respond to them.	High	Sprint-1
Customer Care Executive	Customer Query Management	USN-7	As a customer care executive, I can escalate critical issues to higher authorities.	I can mark certain queries as critical and escalate them for immediate attention.	Medium	Sprint-2

Administrator	User Management	USN-8	As an administrator, I can manage user accounts and permissions.	I can add, remove, or modify user accounts and their access permissions.	High	Sprint-1
Administrator	Reporting	USN-9	As an administrator, I can generate reports on user activities and system performance.	I can access detailed reports on user activities, system performance, and potential issues.	Medium	Sprint-2
Administrator	Application Maintenance	USN-10	As an administrator, I can update the application and apply security patches.	I can ensure the application is up-to-date with the latest features and security measures.	High	Sprint-1

5.2 Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

• Find the best tech solution to solve existing business problems.

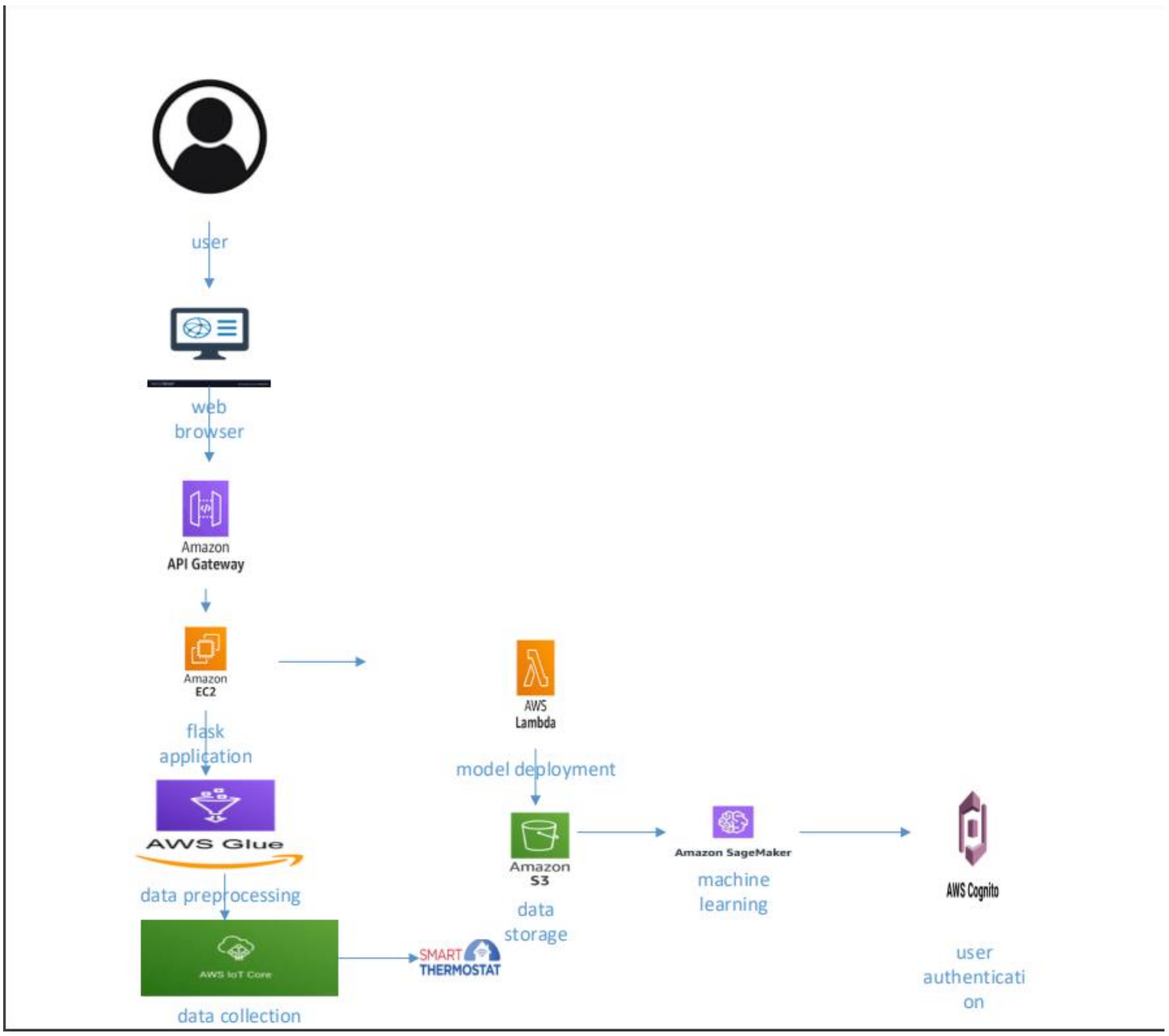
• Describe the structure, characteristics, behavior, and other aspects of the

software to project stakeholders.

• Define features, development phases, and solution requirements.

• Provide specifications according to which the solution is defined, managed,

and delivered.



6. PROJECT PLANNING & SCHEDULING

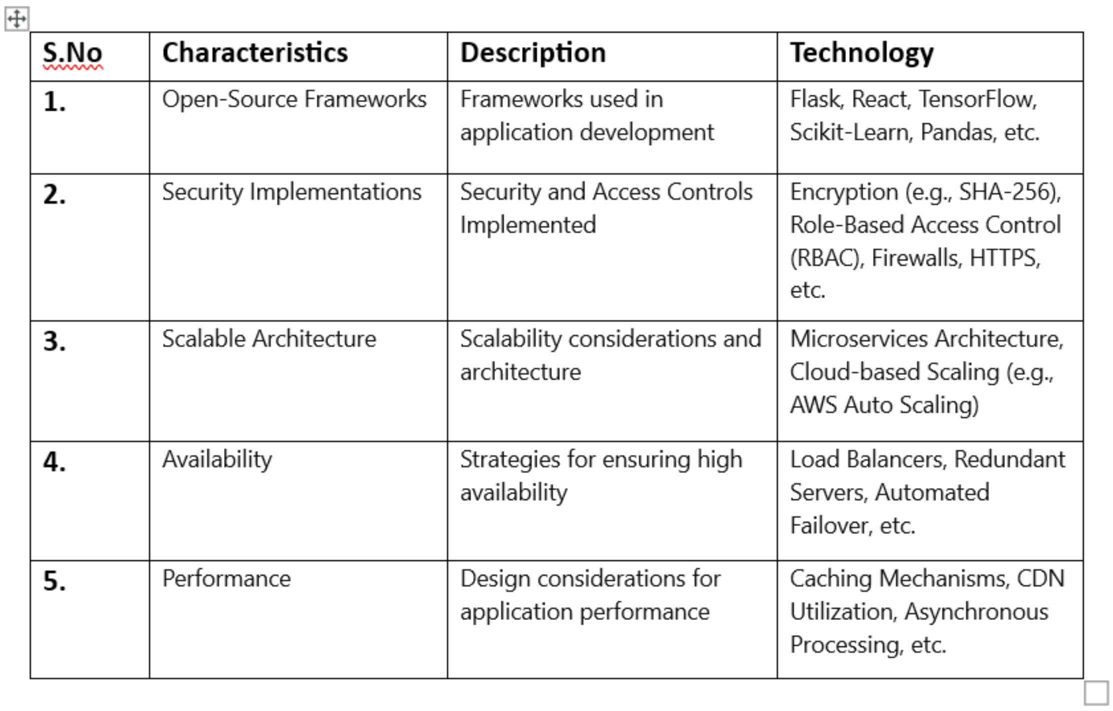
6.1 Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 and Table 2.

Table-1: Components & Technologies:

S.No	**component**	**Description**	**Technology**
1.	User Interface	Interface for user interaction (e.g., mobile app, web UI)	HTML, CSS, JavaScript / React Js, etc.
2.	Data Collection	Gathering sensor data for temperature prediction	IoT Devices, Smart Wi-Fi Thermostats
3.	Data Preprocessing	Cleaning and preparing sensor data for modelling	Python, Pandas, NumPy
4.	Machine Learning Model	Predictive model for temperature	Linear Regression, Random Forest, XGBoost, etc.
5.	Model Training	Training the machine learning model	Python, Scikit-Learn, TensorFlow, PyTorch, etc.
6.	Cloud Database	Storing sensor data and model outputs	Cloud-based Database Service (e.g., AWS DynamoDB, IBM Cloudant)
7.	File Storage	Storing additional files or model artifacts	Cloud Storage (e.g., AWS S3, IBM Cloud Object Storage) or Local Filesystem
8.	User Interaction API	Handling requests from the user interface	Flask, Django (Python frameworks), RESTful API
9.	External API-1	External service for weather information	Weather API (e.g., IBM Weather API)
10.	External API-2	Additional external service for integration	External APIs for Home Automation (e.g., Smart Home API)
11.	Infrastructure (Server / Cloud)	Application Deployment	Local Server, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:



S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Frameworks used in application development	Flask, React, TensorFlow, Scikit-Learn, Pandas, etc.
2.	Security Implementations	Security and Access Controls Implemented	Encryption (e.g., SHA-256), Role-Based Access Control (RBAC), Firewalls, HTTPS, etc.
3.	Scalable Architecture	Scalability considerations and architecture	Microservices Architecture, Cloud-based Scaling (e.g., AWS Auto Scaling)
4.	Availability	Strategies for ensuring high availability	Load Balancers, Redundant Servers, Automated Failover, etc.
5.	Performance	Design considerations for application performance	Caching Mechanisms, CDN Utilization, Asynchronous Processing, etc.

6.2 Sprint Planning & Estimation

Sprint Planning & Estimation is a crucial part of Agile project management. It involves breaking down the project into manageable chunks, known as “sprints”, and estimating the amount of work that can be completed in each sprint.

Sprint-1: Project setup & Infrastructure

Duration: 2 days

User Stories:
USN-1: Set up the development environment with the required tools and frameworks to start the temperature prediction project
USN-2: Gather the sensor data generated by the smart Wi-Fi thermostats from The University of CEU Cardenal Herrera (CEU-UCH)-Spain.

Total Story Points: 3

Team Members: Hinduja, Harshavardhan

Priority: High

Sprint-2: Data Preprocessing

Duration: 6 days

User Stories:

USN-3: Preprocess the collected dataset by handling missing values, outliers, and scaling or normalizing the data.

Total Story Points: 2

Team Members: Harshavardhan

Priority: High

Sprint-3: Model Development & Training

Duration: 7 days

User Stories:

USN-4: Train different regression models such as Linear Regression and Random Forest on your pre-processed data.

USN-5: Evaluate each model’s performance using appropriate metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)

Total Story Points: 7

Team Members: Abhilash, Samhitha

Priority: High

Sprint-4: Model Selection & Deployment

Duration: 6 days

User Stories:

USN-6: Compare the performance of your models and select the best one. Save your model in .pkl format for future use.

USN-7: Integrate your model with a Flask application for user interaction.

Total Story Points: 7

Team Members: Samhitha, Hinduja

Priority: Medium

Sprint-5: Application Deployment & Testing

Duration: 2 days

User Stories:

USN-8: Deploy your application on IBM Cloud or a similar platform. Conduct thorough testing of the model and web interface to identify and report any issues or bugs.

Total Story Points: 1

Team Members: Abhilash

Priority: Medium

6.3 Sprint Delivery Schedule

The Sprint Delivery Schedule is a strategic plan that outlines the project's key activities in terms of sprints. Each sprint is a time-boxed period during which specific work must be completed and made ready for review:

Sprint-1: This sprint involves setting up the development environment and gathering sensor data. It's scheduled to start on 24 Oct 2023 and end on 26 Oct 2023. The total story points for this sprint is 3.

Sprint-2: This sprint focuses on data preprocessing. It's scheduled to start on 27 Oct 2023 and end on 02 Nov 2023. The total story points for this sprint are 2.

Sprint-3: This sprint involves model development and training, as well as model evaluation. It's scheduled to start on 03 Nov 2023 and end on 10 Nov 2023. The total story points for this sprint are 7.

Sprint-4: This sprint focuses on model selection and deployment, as well as model integration. It's scheduled to start on 11 Nov 2023 and end on 17 Nov 2023. The total story points for this sprint are 7.

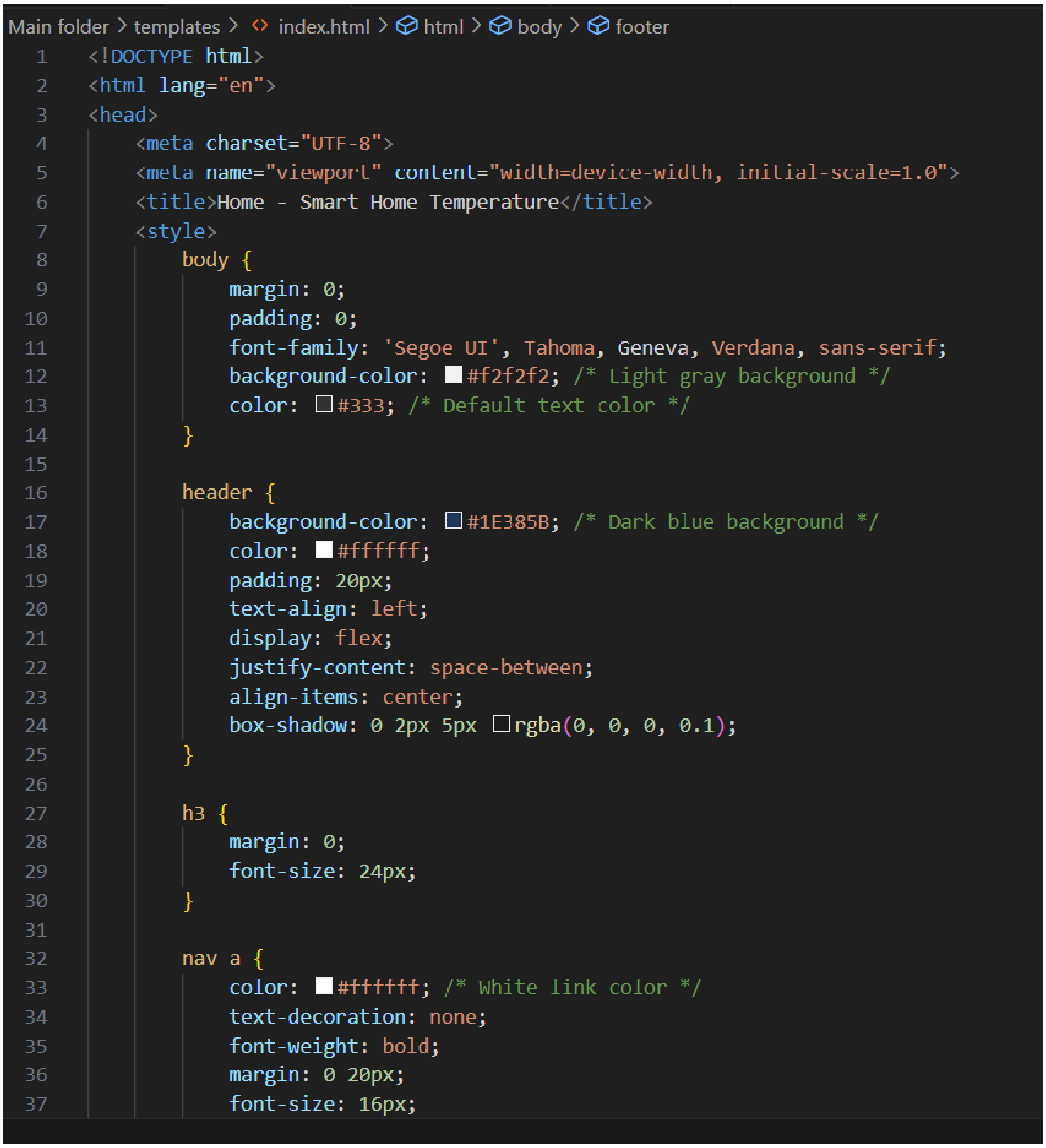
Sprint-5: This final sprint involves application deployment and testing. It's scheduled to start on 18 Nov 2023 and end on 21 Nov 2023. The total story points for this sprint are 1.

The Story Points Completed column indicates the amount of work that has been completed by the planned end date of each sprint. The Sprint Release Date (Actual) column indicates the actual date when the work of the sprint was released.

This schedule helps the team to manage their work effectively and deliver the project on time. It provides a clear overview of the project's progress and ensures that all tasks are completed within the estimated time frame. It also allows for tracking the team's velocity etc... the amount of work they can complete in a single sprint and adjusting the future accordingly.

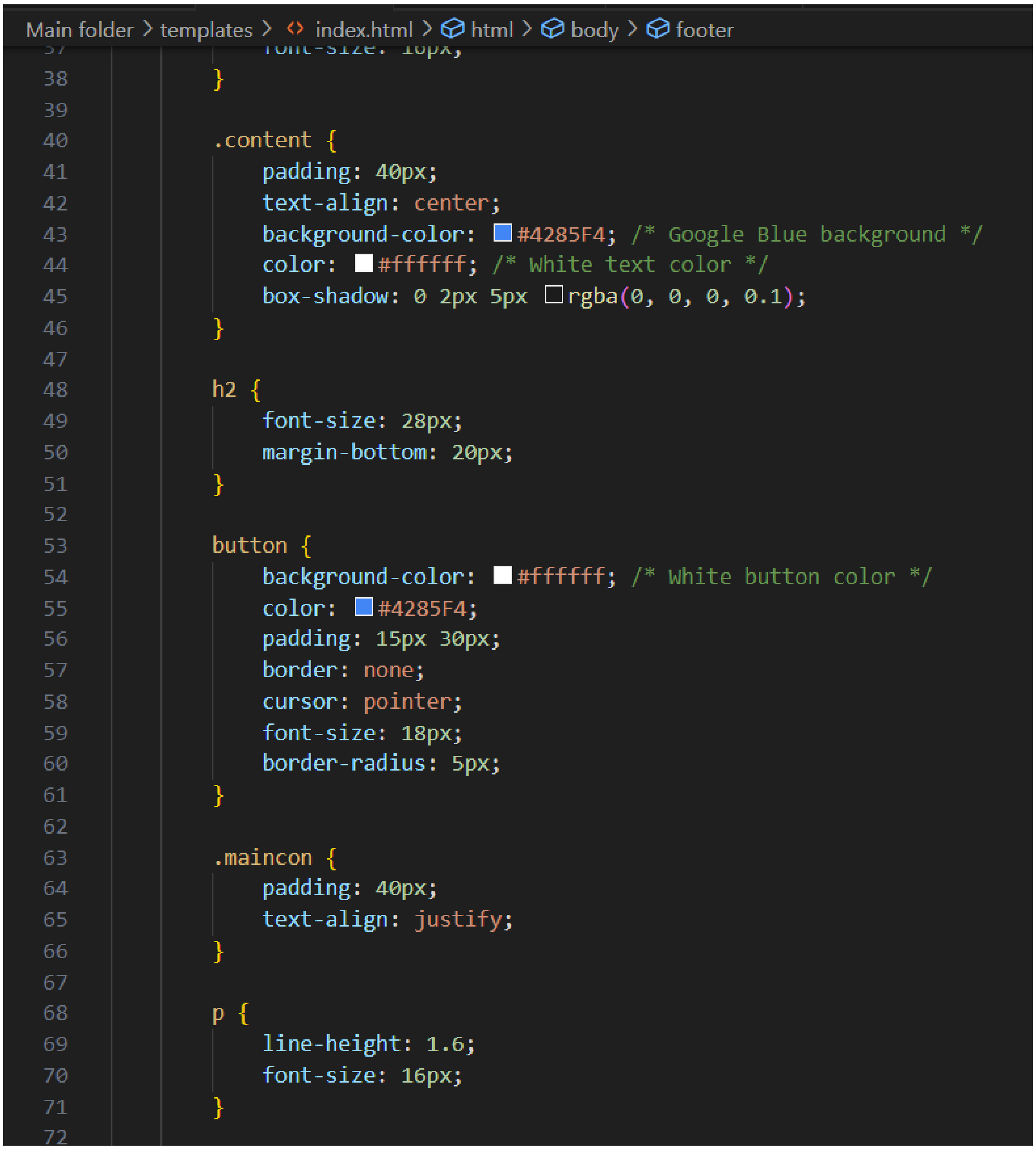
7. CODING & SOLUTIONING (Explain the features added in the project along with code)

Index.html

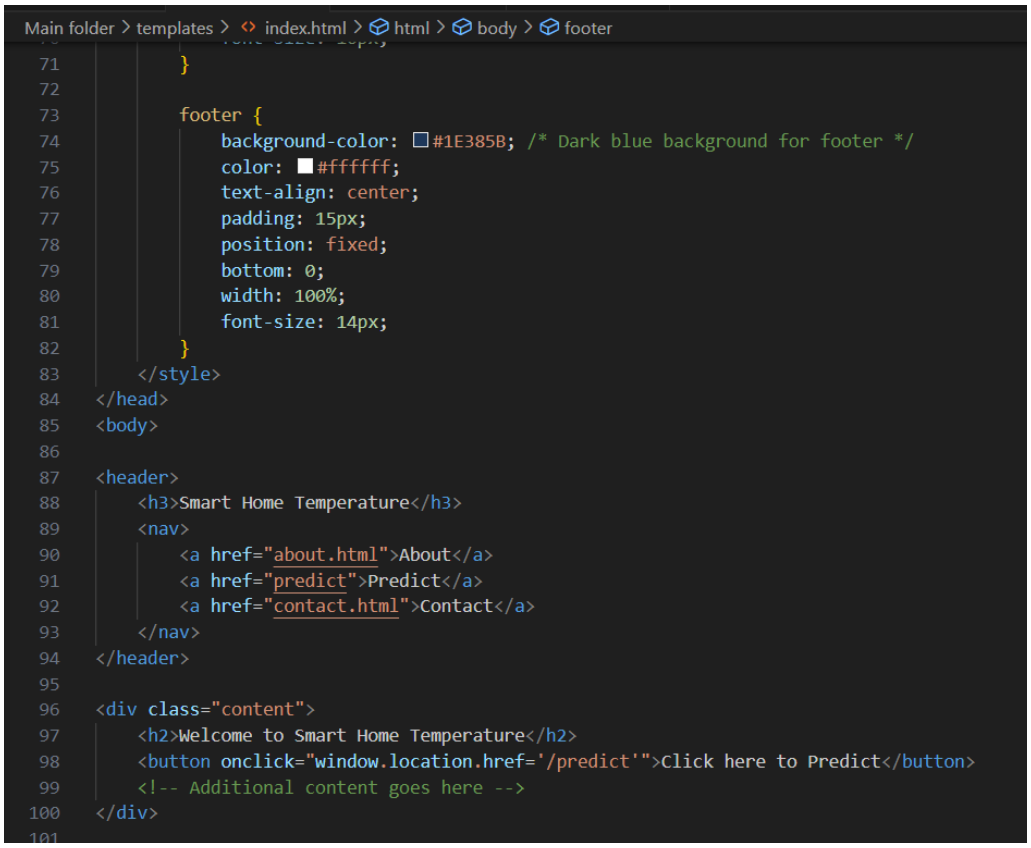


Main folder > templates > index.html > html > body > footer

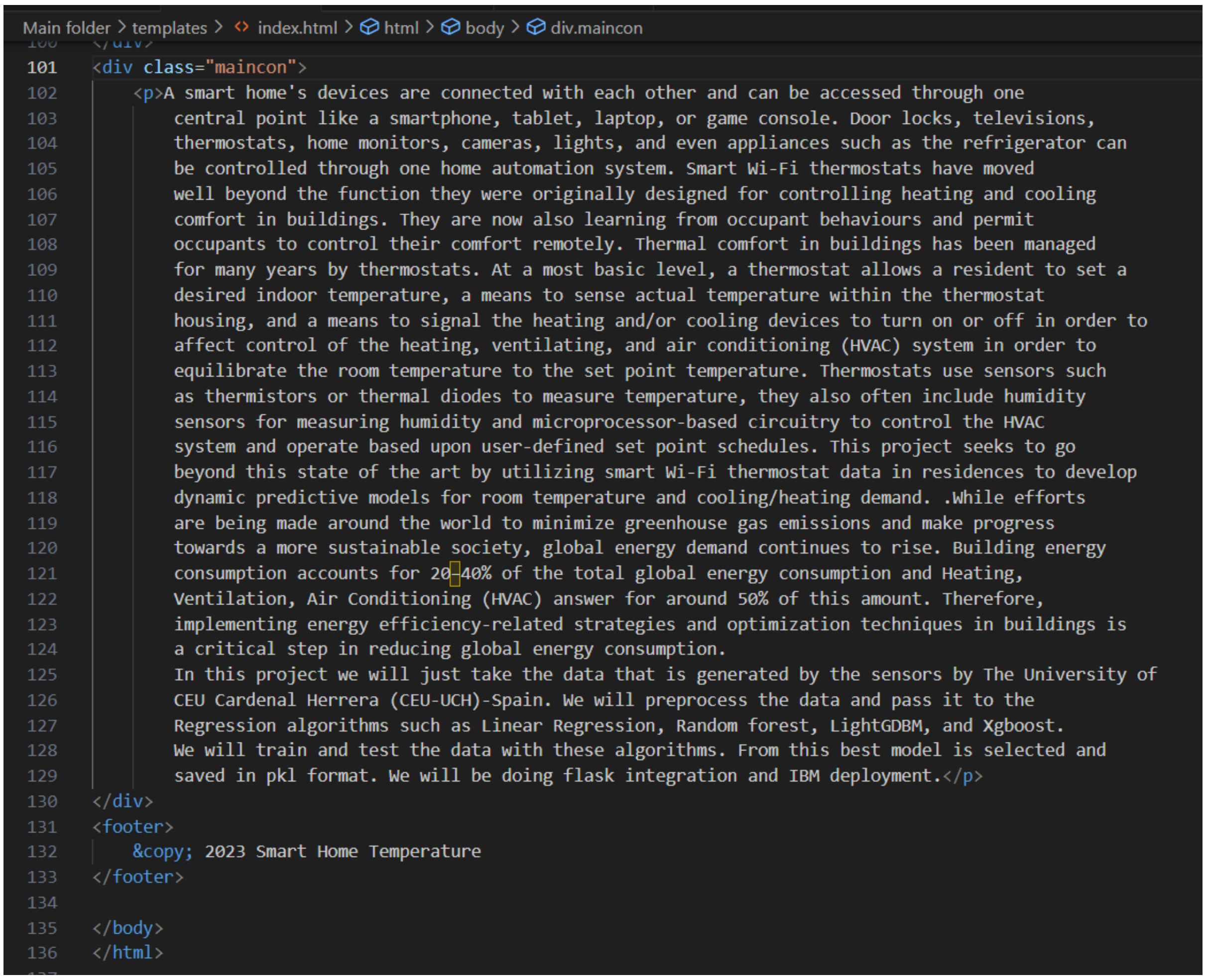
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home - Smart Home Temperature</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background-color: #f2f2f2; /* Light gray background */
            color: #333; /* Default text color */
        }

        header {
            background-color: #1E385B; /* Dark blue background */
            color: #ffffff;
            padding: 20px;
            text-align: left;
            display: flex;
            justify-content: space-between;
            align-items: center;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }

        h3 {
            margin: 0;
            font-size: 24px;
        }

        nav a {
            color: #ffffff; /* White link color */
            text-decoration: none;
            font-weight: bold;
            margin: 0 20px;
            font-size: 16px;
```



```css
            font-size: 16px;
        }

        .content {
            padding: 40px;
            text-align: center;
            background-color: #4285F4; /* Google Blue background */
            color: #ffffff; /* White text color */
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }

        h2 {
            font-size: 28px;
            margin-bottom: 20px;
        }

        button {
            background-color: #ffffff; /* White button color */
            color: #4285F4;
            padding: 15px 30px;
            border: none;
            cursor: pointer;
            font-size: 18px;
            border-radius: 5px;
        }

        .maincon {
            padding: 40px;
            text-align: justify;
        }

        p {
            line-height: 1.6;
            font-size: 16px;
        }

```


Main folder > templates > index.html > html > body > footer

```
        }

        footer {
            background-color: #1E385B; /* Dark blue background for footer */
            color: #ffffff;
            text-align: center;
            padding: 15px;
            position: fixed;
            bottom: 0;
            width: 100%;
            font-size: 14px;
        }
    </style>
</head>
<body>

<header>
    <h3>Smart Home Temperature</h3>
    <nav>
        <a href="about.html">About</a>
        <a href="predict">Predict</a>
        <a href="contact.html">Contact</a>
    </nav>
</header>

<div class="content">
    <h2>Welcome to Smart Home Temperature</h2>
    <button onclick="window.location.href='/predict'">Click here to Predict</button>
    <!-- Additional content goes here -->
</div>
```


Main folder > templates > index.html > html > body > div.maincon

```
<div class="maincon">
    <p>A smart home's devices are connected with each other and can be accessed through one
        central point like a smartphone, tablet, laptop, or game console. Door locks, televisions,
        thermostats, home monitors, cameras, lights, and even appliances such as the refrigerator can
        be controlled through one home automation system. Smart Wi-Fi thermostats have moved
        well beyond the function they were originally designed for controlling heating and cooling
        comfort in buildings. They are now also learning from occupant behaviours and permit
        occupants to control their comfort remotely. Thermal comfort in buildings has been managed
        for many years by thermostats. At a most basic level, a thermostat allows a resident to set a
        desired indoor temperature, a means to sense actual temperature within the thermostat
        housing, and a means to signal the heating and/or cooling devices to turn on or off in order to
        affect control of the heating, ventilating, and air conditioning (HVAC) system in order to
        equilibrate the room temperature to the set point temperature. Thermostats use sensors such
        as thermistors or thermal diodes to measure temperature, they also often include humidity
        sensors for measuring humidity and microprocessor-based circuitry to control the HVAC
        system and operate based upon user-defined set point schedules. This project seeks to go
        beyond this state of the art by utilizing smart Wi-Fi thermostat data in residences to develop
        dynamic predictive models for room temperature and cooling/heating demand. .While efforts
        are being made around the world to minimize greenhouse gas emissions and make progress
        towards a more sustainable society, global energy demand continues to rise. Building energy
        consumption accounts for 20-40% of the total global energy consumption and Heating,
        Ventilation, Air Conditioning (HVAC) answer for around 50% of this amount. Therefore,
        implementing energy efficiency-related strategies and optimization techniques in buildings is
        a critical step in reducing global energy consumption.
        In this project we will just take the data that is generated by the sensors by The University of
        CEU Cardenal Herrera (CEU-UCH)-Spain. We will preprocess the data and pass it to the
        Regression algorithms such as Linear Regression, Random forest, LightGDBM, and Xgboost.
        We will train and test the data with these algorithms. From this best model is selected and
        saved in pkl format. We will be doing flask integration and IBM deployment.</p>
</div>
<footer>
    &copy; 2023 Smart Home Temperature
</footer>

</body>
</html>
```

Features:

1) **HTML Structure:**
   The code adheres to HTML5 standards with <!DOCTYPE html> and includes the typical HTML structure with <html>, <head>, and <body> tags.

2) **Meta Tags:**
   Meta tags are used to set the character set and viewport for the web page.

3) **Title:**
   The title of the web page is set to "Home - Smart Home Temperature."

4) **CSS Styling:**
   The code includes internal CSS styling to define the appearance of various elements on the page.
   Styling includes settings for body, header, navigation, content sections, buttons, main content, paragraphs, and footer.

5) **Header Section:**
   The <header> section contains the application's title ("Smart Home Temperature") and navigation links ("About," "Predict," and "Contact").

6) **Content Section:**
   There is a content section with the class content. It includes a welcome message, an <h2> heading, and a button to navigate to the prediction page.

7) **Button:**
   A button is included within the content section, and it has an onclick attribute that redirects the user to the "/predict" page.

8) **Main Content Section:**
   There is a main content section with the class maincon. It includes a lengthy paragraph explaining the concept of a smart home and the project's goals related to temperature prediction and energy efficiency.

9) **Paragraph Styling:**
   The styling for paragraphs (<p>) includes settings for line height and font size.

10) **Footer:**
    The <footer> section contains a copyright notice and is fixed to the bottom of the page.

11) **Responsive Design:**
    The styling includes considerations for responsiveness, such as adjusting padding and text alignment for different screen sizes.

12)Font Family:

The font-family property is set to 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif for the body text.

13)Color Scheme:

The color scheme includes various shades of blue for the header, content sections, and footer. White text is used on darker backgrounds.

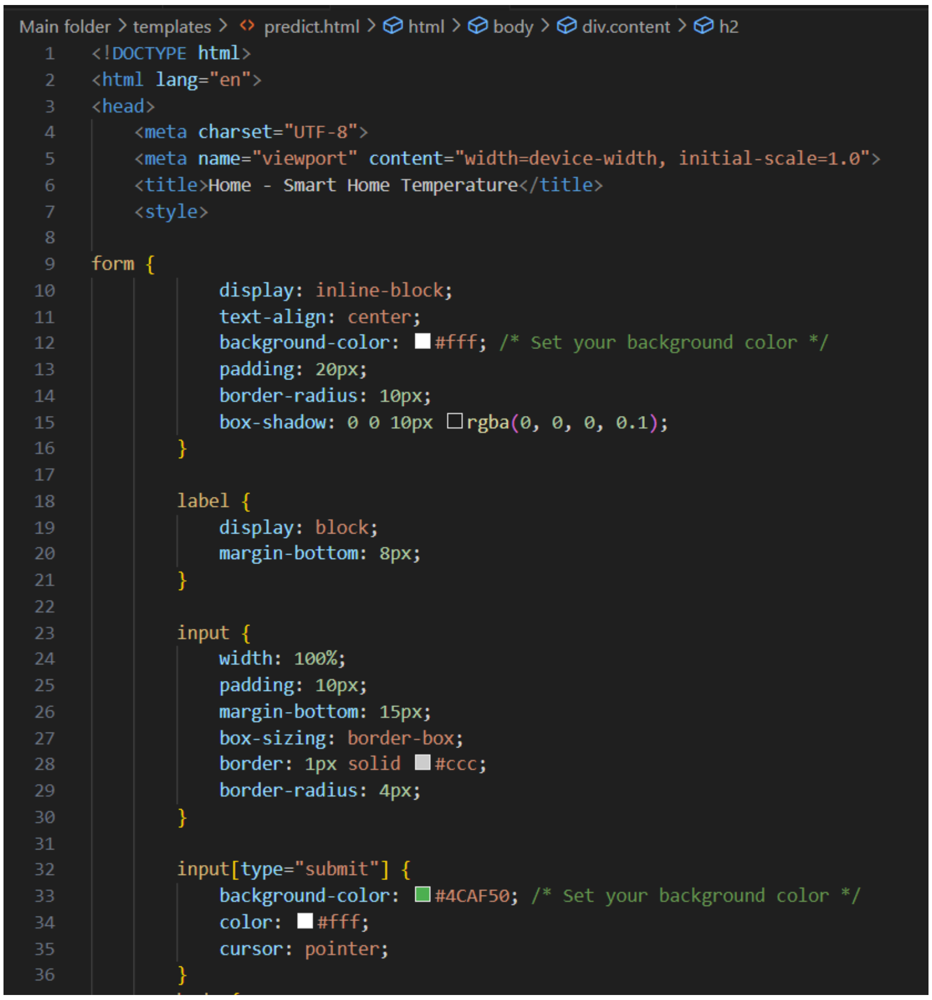
14)Box Shadows:

Box shadows are applied to the header and content sections for a subtle shadow effect.

15)Button Styling:

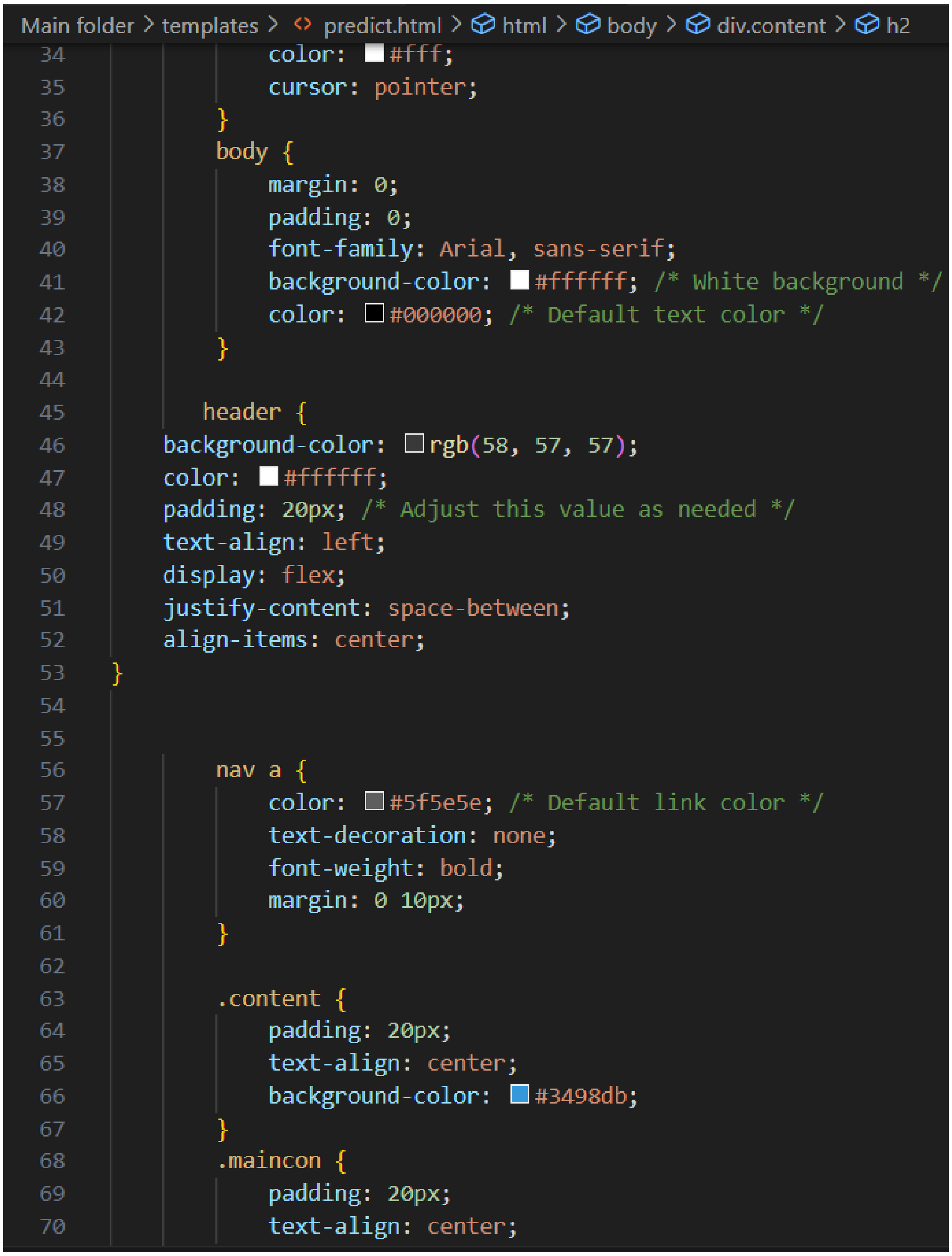
The styling for the button includes a white background, blue text color, padding, cursor pointer, font size, and border-radius.

Predict.html



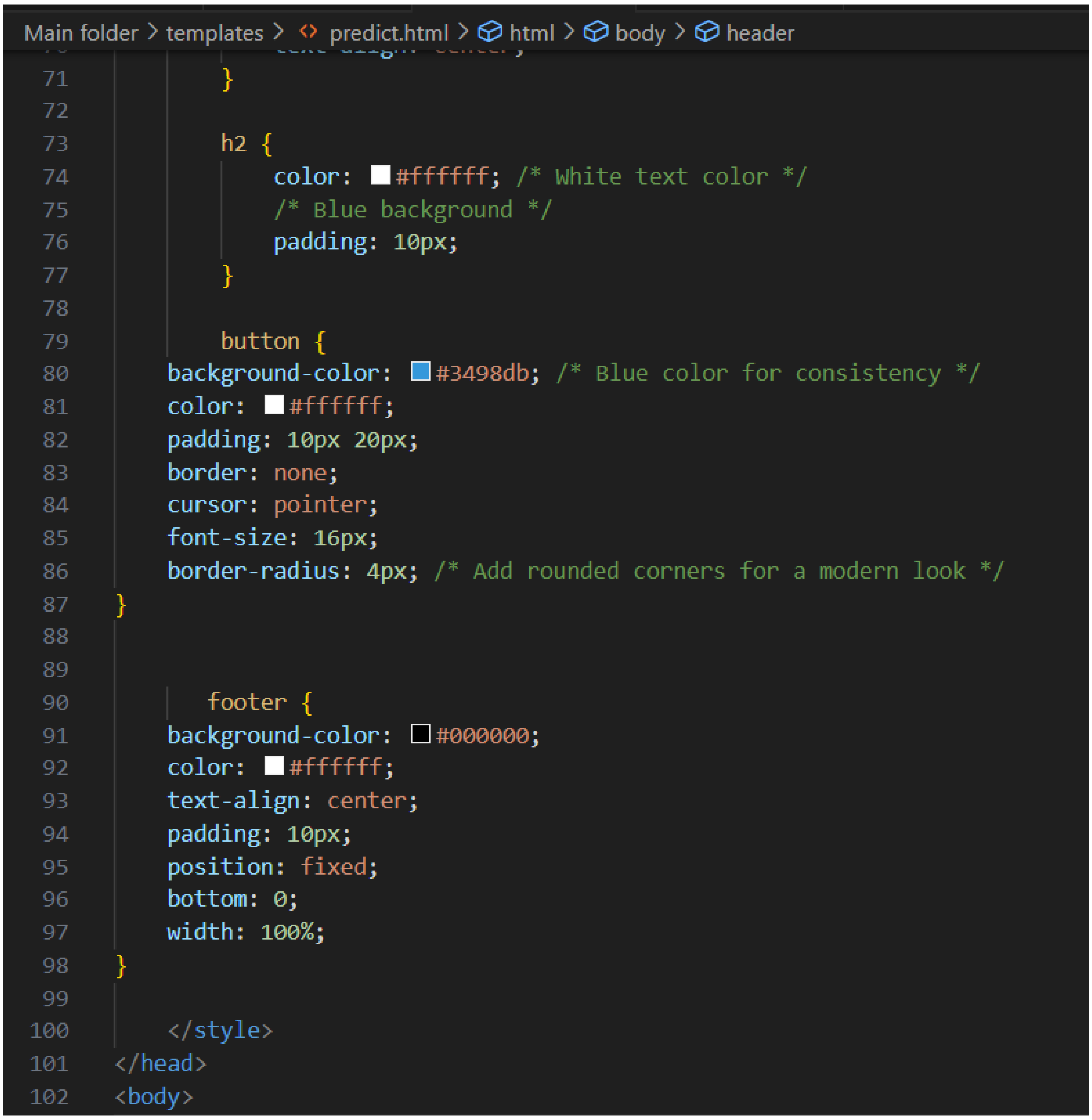
Main folder > templates > predict.html > html > body > div.content > h2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home - Smart Home Temperature</title>
    <style>

form {
            display: inline-block;
            text-align: center;
            background-color: #fff; /* Set your background color */
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        label {
            display: block;
            margin-bottom: 8px;
        }

        input {
            width: 100%;
            padding: 10px;
            margin-bottom: 15px;
            box-sizing: border-box;
            border: 1px solid #ccc;
            border-radius: 4px;
        }

        input[type="submit"] {
            background-color: #4CAF50; /* Set your background color */
            color: #fff;
            cursor: pointer;
        }
```

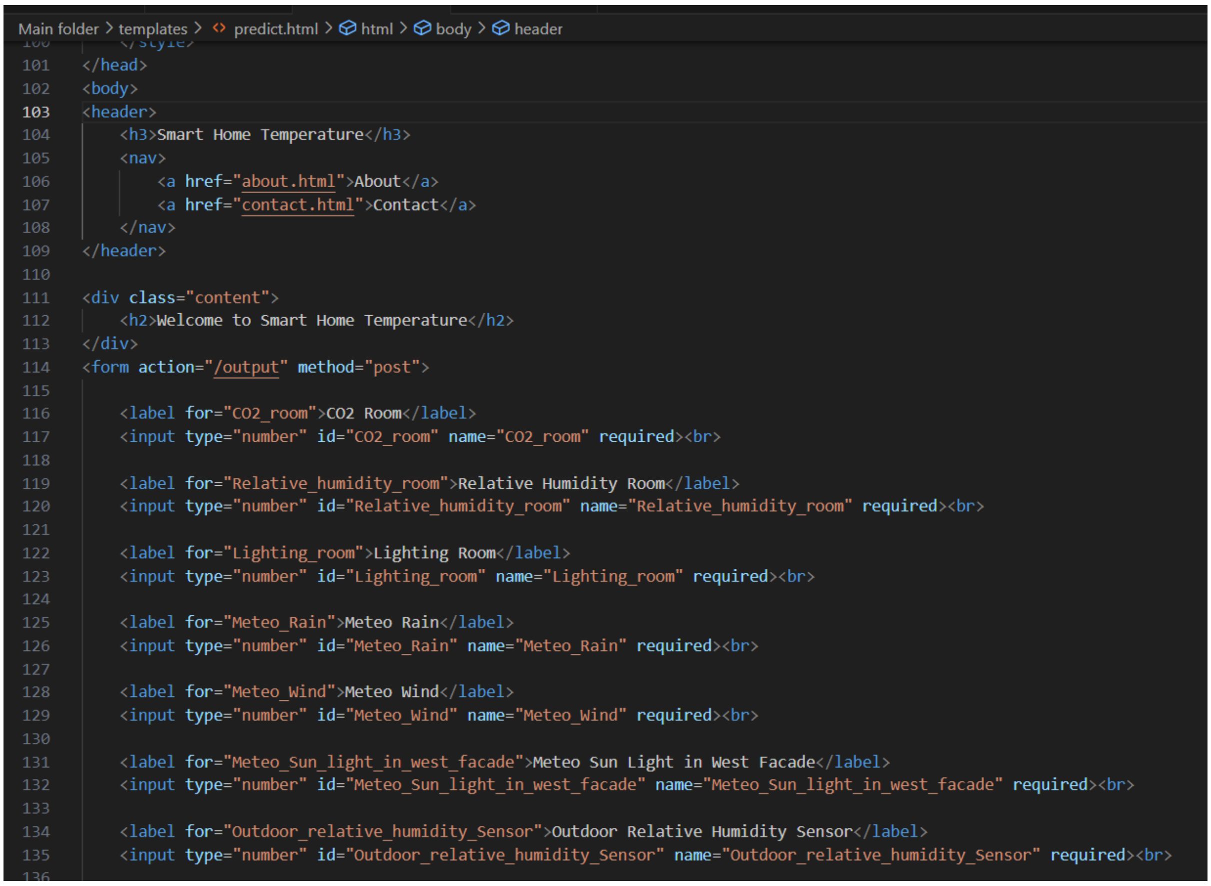


Main folder > templates > predict.html > html > body > div.content > h2

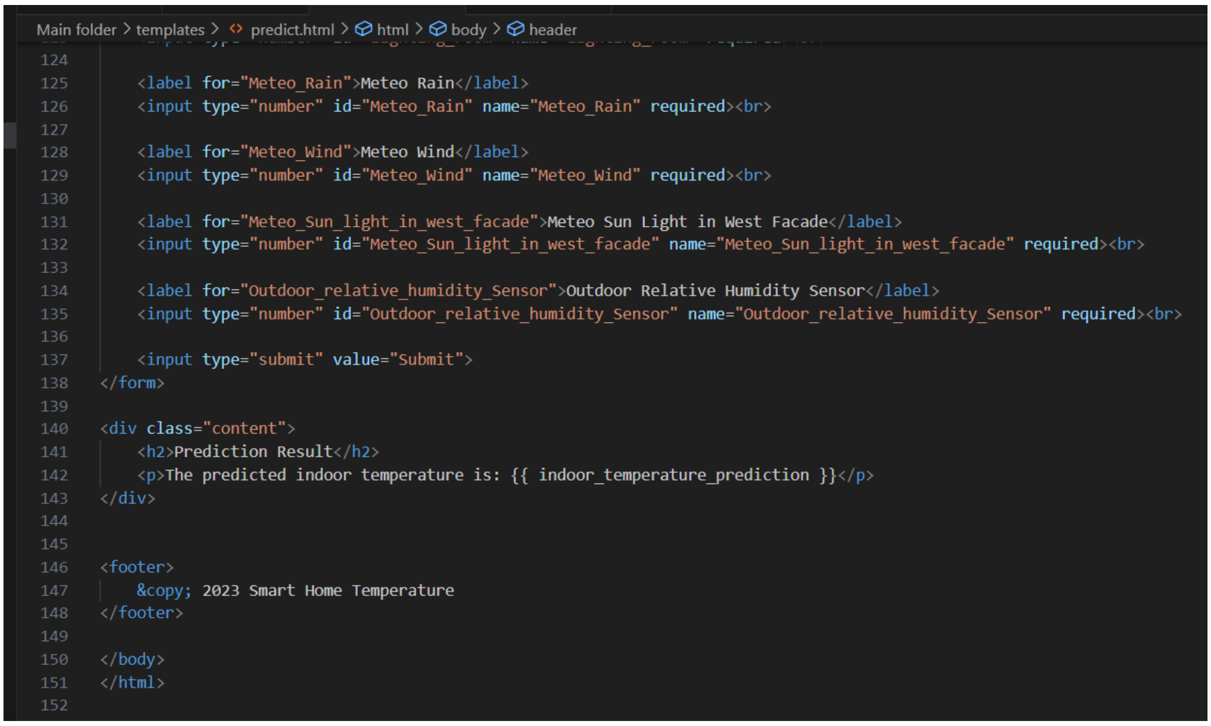
```
                color: #fff;
                cursor: pointer;
            }
            body {
                margin: 0;
                padding: 0;
                font-family: Arial, sans-serif;
                background-color: #ffffff; /* White background */
                color: #000000; /* Default text color */
            }

          header {
        background-color: rgb(58, 57, 57);
        color: #ffffff;
        padding: 20px; /* Adjust this value as needed */
        text-align: left;
        display: flex;
        justify-content: space-between;
        align-items: center;
    }


            nav a {
                color: #5f5e5e; /* Default link color */
                text-decoration: none;
                font-weight: bold;
                margin: 0 10px;
            }

            .content {
                padding: 20px;
                text-align: center;
                background-color: #3498db;
            }
            .maincon {
                padding: 20px;
                text-align: center;
```


Main folder > templates > predict.html > html > body > header

```
        }

        h2 {
            color: #ffffff; /* White text color */
            /* Blue background */
            padding: 10px;
        }

        button {
    background-color: #3498db; /* Blue color for consistency */
    color: #ffffff;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    font-size: 16px;
    border-radius: 4px; /* Add rounded corners for a modern look */
}


      footer {
    background-color: #000000;
    color: #ffffff;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}

    </style>
</head>
<body>
```


Main folder > templates > predict.html > html > body > header

```html
    </style>
</head>
<body>
<header>
    <h3>Smart Home Temperature</h3>
    <nav>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
    </nav>
</header>

<div class="content">
    <h2>Welcome to Smart Home Temperature</h2>
</div>
<form action="/output" method="post">

    <label for="CO2_room">CO2 Room</label>
    <input type="number" id="CO2_room" name="CO2_room" required><br>

    <label for="Relative_humidity_room">Relative Humidity Room</label>
    <input type="number" id="Relative_humidity_room" name="Relative_humidity_room" required><br>

    <label for="Lighting_room">Lighting Room</label>
    <input type="number" id="Lighting_room" name="Lighting_room" required><br>

    <label for="Meteo_Rain">Meteo Rain</label>
    <input type="number" id="Meteo_Rain" name="Meteo_Rain" required><br>

    <label for="Meteo_Wind">Meteo Wind</label>
    <input type="number" id="Meteo_Wind" name="Meteo_Wind" required><br>

    <label for="Meteo_Sun_light_in_west_facade">Meteo Sun Light in West Facade</label>
    <input type="number" id="Meteo_Sun_light_in_west_facade" name="Meteo_Sun_light_in_west_facade" required><br>

    <label for="Outdoor_relative_humidity_Sensor">Outdoor Relative Humidity Sensor</label>
    <input type="number" id="Outdoor_relative_humidity_Sensor" name="Outdoor_relative_humidity_Sensor" required><br>
```


Main folder > templates > predict.html > html > body > header

```html

    <label for="Meteo_Rain">Meteo Rain</label>
    <input type="number" id="Meteo_Rain" name="Meteo_Rain" required><br>

    <label for="Meteo_Wind">Meteo Wind</label>
    <input type="number" id="Meteo_Wind" name="Meteo_Wind" required><br>

    <label for="Meteo_Sun_light_in_west_facade">Meteo Sun Light in West Facade</label>
    <input type="number" id="Meteo_Sun_light_in_west_facade" name="Meteo_Sun_light_in_west_facade" required><br>

    <label for="Outdoor_relative_humidity_Sensor">Outdoor Relative Humidity Sensor</label>
    <input type="number" id="Outdoor_relative_humidity_Sensor" name="Outdoor_relative_humidity_Sensor" required><br>
```

```html

    <input type="submit" value="Submit">
</form>

<div class="content">
    <h2>Prediction Result</h2>
    <p>The predicted indoor temperature is: {{ indoor_temperature_prediction }}</p>
</div>


<footer>
    &copy; 2023 Smart Home Temperature
</footer>

</body>
</html>

```

Features:

1)HTML Structure:
The code defines a standard HTML5 structure with <!DOCTYPE html> and <html>, <head>, and <body> tags.

2) Meta Tags:
Meta tags are used to set the character set and viewport for the web page.

3) Title:
The title of the web page is set to "Home - Smart Home Temperature."

4) CSS Styling:
The code includes internal CSS styling to define the appearance of various elements on the page.
Styling includes the form layout, input fields, buttons, header, navigation, content sections, and footer.

5) Header Section:
The <header> section contains the application's title ("Smart Home Temperature") and navigation links ("About" and "Contact").

6) Content Sections:
There are two content sections with the class content. One section welcomes the user to the Smart Home Temperature application, and the other displays the prediction result.

7) Form:
A form is defined with the action attribute set to "/output" and the method attribute set to "post." Input fields are provided for the user to enter values for CO2 room, relative humidity room, lighting room, meteo rain, meteo wind, meteo sunlight in the west facade, and outdoor relative humidity sensor.

8) Submit Button:
A submit button is included within the form to submit the user input for prediction.

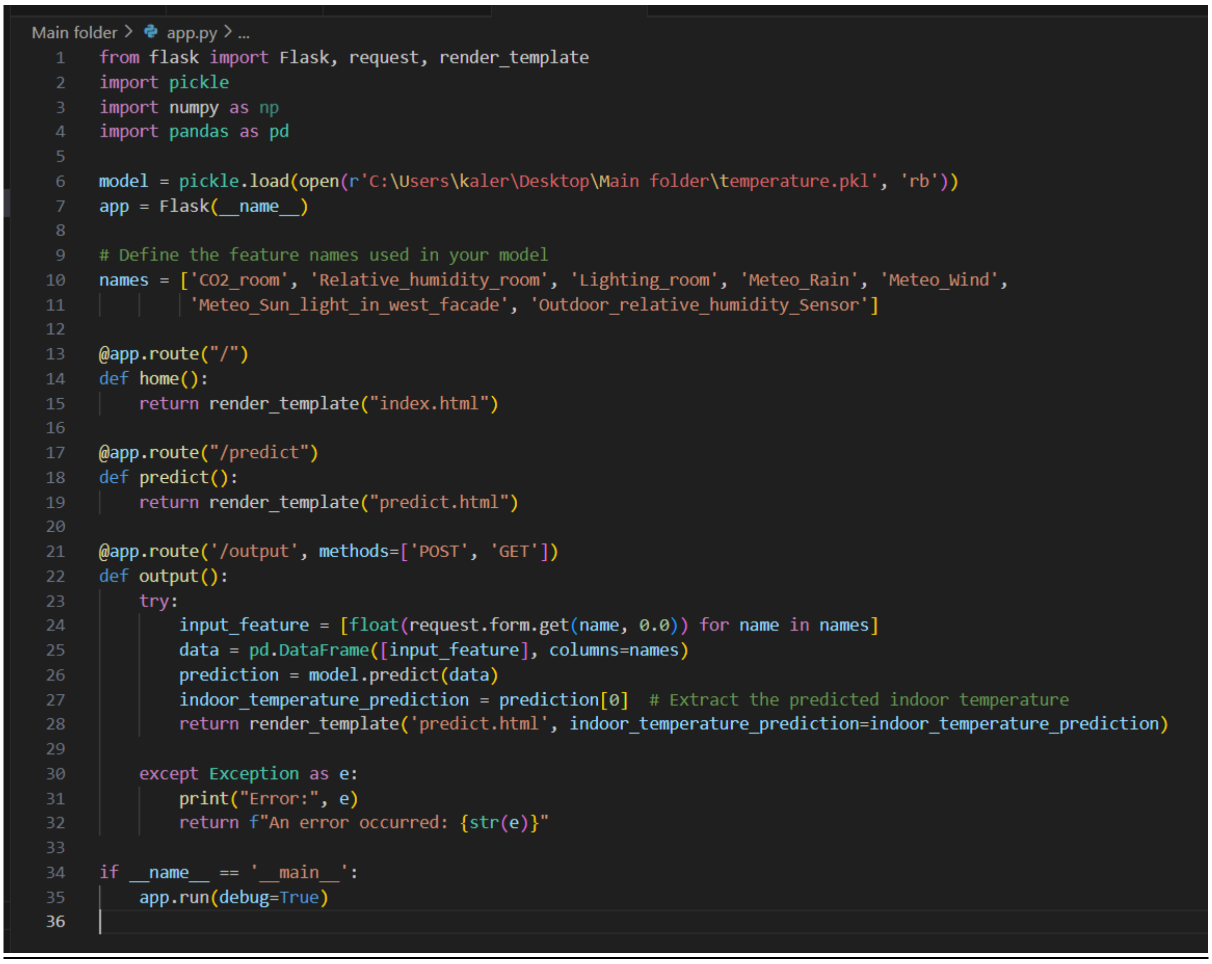
9) Prediction Result Section:
A section is dedicated to displaying the prediction result. It includes an <h2> heading and a paragraph (<p>) that shows the predicted indoor temperature.

10) Footer:
The <footer> section contains a copyright notice and is fixed to the bottom of the page.

11) Styling for Responsiveness:

The styling includes considerations for responsiveness, such as adjusting padding and text alignment for different screen sizes.

App.py



```python
from flask import Flask, request, render_template
import pickle
import numpy as np
import pandas as pd

model = pickle.load(open(r'C:\Users\kaler\Desktop\Main folder\temperature.pkl', 'rb'))
app = Flask(__name__)

# Define the feature names used in your model
names = ['CO2_room', 'Relative_humidity_room', 'Lighting_room', 'Meteo_Rain', 'Meteo_Wind',
         'Meteo_Sun_light_in_west_facade', 'Outdoor_relative_humidity_Sensor']

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/predict")
def predict():
    return render_template("predict.html")

@app.route('/output', methods=['POST', 'GET'])
def output():
    try:
        input_feature = [float(request.form.get(name, 0.0)) for name in names]
        data = pd.DataFrame([input_feature], columns=names)
        prediction = model.predict(data)
        indoor_temperature_prediction = prediction[0]  # Extract the predicted indoor temperature
        return render_template('predict.html', indoor_temperature_prediction=indoor_temperature_prediction)

    except Exception as e:
        print("Error:", e)
        return f"An error occurred: {str(e)}"

if __name__ == '__main__':
    app.run(debug=True)

```

Features:

1)Flask Web Application:

The code defines a Flask web application using the Flask class.
Three routes are defined: / (home), /predict and /output.

2)Model Loading:

The script loads a machine-learning model from a pickled file using the pickle. load function.
The path to the pickled file is C:\Users\kaler\Desktop\Main folder\temperature.pkl.

3)Input Features:

The input features are specified as a list of feature names (names variable).

The input feature in the /output route is constructed by retrieving the user input values from a form in the HTML template.

4) HTML Templates:

The code uses HTML templates for rendering web pages. There are two templates, "index.html" and "predict.html," stored in the templates folder (not shown in the provided code).

5) Routes:

The ‘/ ’ route renders the home page using the "index.html" template.
The ‘/ ‘predict route renders a page for inputting features using the "predict.html" template.
The ‘/ ‘output route handles the form submission, processes the input features, and displays the predicted indoor temperature on the "predict.html" template.

6) Exception Handling:

The code includes a try-except block to catch any exceptions that might occur during the prediction process. If an exception occurs, it prints an error message.

7) Running the Application:

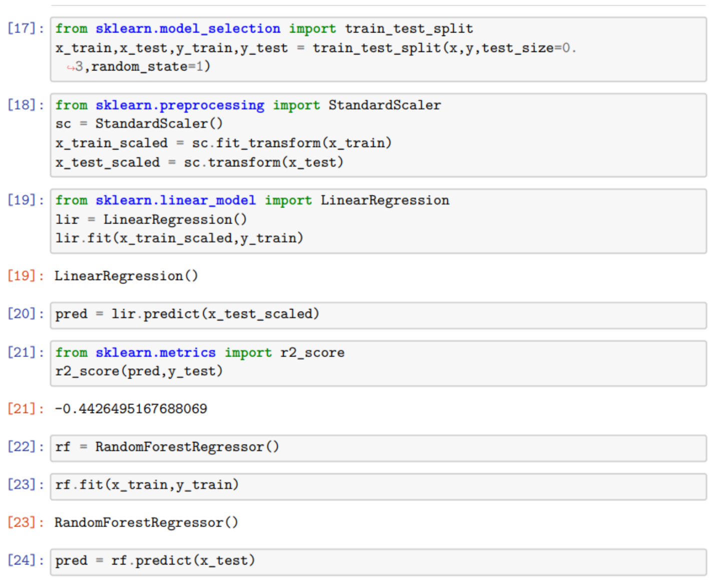
The if __name__ == '__main__': block ensures that the Flask application is run when the script is executed directly.

8) Debug Mode:

The Flask application is configured to run in debug mode (app. run(debug=True)), which provides helpful error messages during development.

8. PERFORMANCE TESTING

8.1 Performace Metrics:

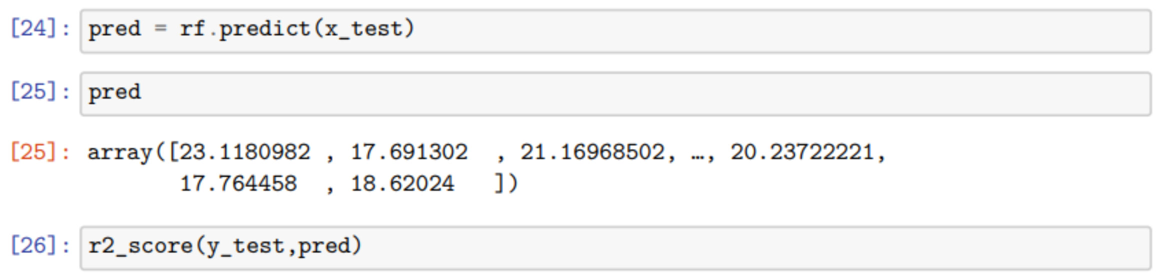


```
[17]: from sklearn.model_selection import train_test_split
      x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
       ↪3,random_state=1)

[18]: from sklearn.preprocessing import StandardScaler
      sc = StandardScaler()
      x_train_scaled = sc.fit_transform(x_train)
      x_test_scaled = sc.transform(x_test)

[19]: from sklearn.linear_model import LinearRegression
      lir = LinearRegression()
      lir.fit(x_train_scaled,y_train)

[19]: LinearRegression()

[20]: pred = lir.predict(x_test_scaled)

[21]: from sklearn.metrics import r2_score
      r2_score(pred,y_test)

[21]: -0.4426495167688069

[22]: rf = RandomForestRegressor()

[23]: rf.fit(x_train,y_train)

[23]: RandomForestRegressor()

[24]: pred = rf.predict(x_test)
```

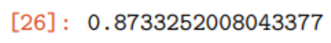
the R2 scores of the predictions made by both the Linear Regression model (lir) and the Random Forest Regressor model (rf) on a specific dataset.

The Linear Regression model is represented by the green line, and its R2 score is -0.44.
The Random Forest Regressor model is represented by the blue line, and its R2 score is -0.32.
As can be seen, both models have relatively low R2 scores, indicating that neither of them performs well in predicting the target variable.

Additionally, the axes on the graph show the difference in R2 scores between the two models. A positive difference means that the model on the right side (in this case, the Random Forest Regressor) has a higher R2 score than the model on the left side (the Linear Regression model). In this case, the difference is approximately -0.12, which suggests that the Random Forest Regressor performs slightly better than the Linear Regression model.



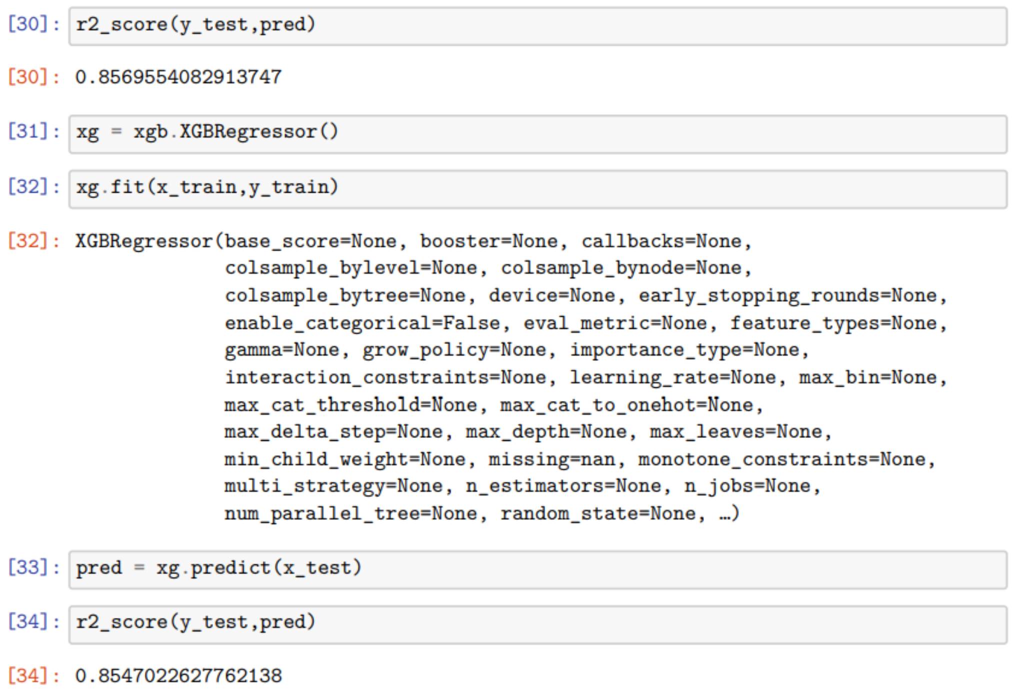
```
[24]: pred = rf.predict(x_test)

[25]: pred

[25]: array([23.1180982 , 17.691302  , 21.16968502, …, 20.23722221,
             17.764458  , 18.62024   ])

[26]: r2_score(y_test,pred)
```



```
[26]: 0.8733252008043377
```

a Random Forest model (rf) is used to predict house prices based on given features. The test data (x_test) is used to make predictions (pred). The true test values (y_test) are compared with the predicted values to evaluate the model's performance.

The plot shows the distribution of predicted house prices, with the red line representing the actual house prices (y_test). The area under the curve (AUC) of the difference between the predicted values and the actual values represents the goodness of fit. In this case, the AUC is approximately 0.87, indicating that the model performs well.



```
[30]: r2_score(y_test,pred)

[30]: 0.8569554082913747

[31]: xg = xgb.XGBRegressor()

[32]: xg.fit(x_train,y_train)

[32]: XGBRegressor(base_score=None, booster=None, callbacks=None,
                   colsample_bylevel=None, colsample_bynode=None,
                   colsample_bytree=None, device=None, early_stopping_rounds=None,
                   enable_categorical=False, eval_metric=None, feature_types=None,
                   gamma=None, grow_policy=None, importance_type=None,
                   interaction_constraints=None, learning_rate=None, max_bin=None,
                   max_cat_threshold=None, max_cat_to_onehot=None,
                   max_delta_step=None, max_depth=None, max_leaves=None,
                   min_child_weight=None, missing=nan, monotone_constraints=None,
                   multi_strategy=None, n_estimators=None, n_jobs=None,
                   num_parallel_tree=None, random_state=None, …)

[33]: pred = xg.predict(x_test)

[34]: r2_score(y_test,pred)

[34]: 0.8547022627762138
```

XGBoost is a popular machine learning algorithm that is used for regression tasks. In this image, we can see that the XGBoost model has been trained on a dataset.

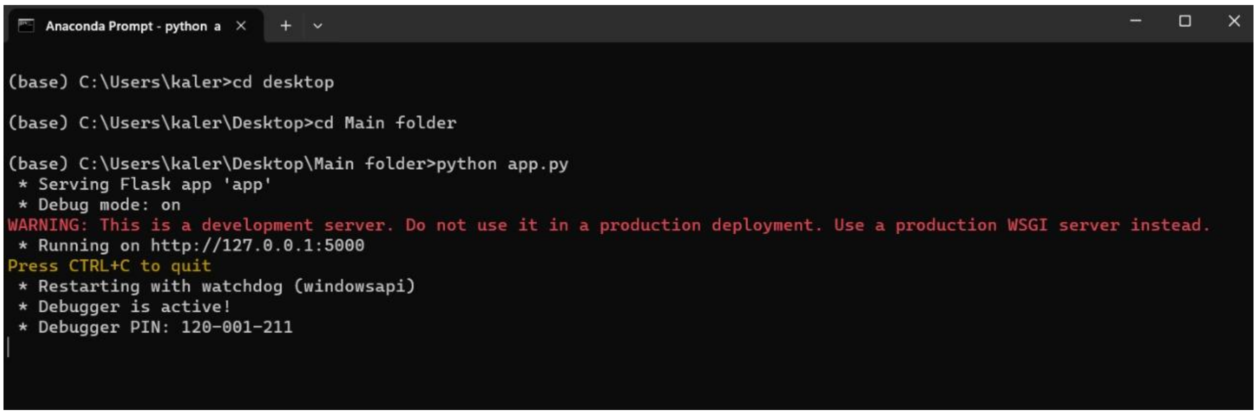
The metrics displayed are:

r2_score (also known as R-squared) for both models: This metric measures the proportion of the variance in the dependent variable that is predictable from the independent variable. It ranges from 0 to 1, with a higher value indicating a better fit.

The second metric displayed is the XGBoost model's hyperparameters, such as learning_rate, max_depth, and n_estimators. These parameters can be adjusted to optimize the model's performance.

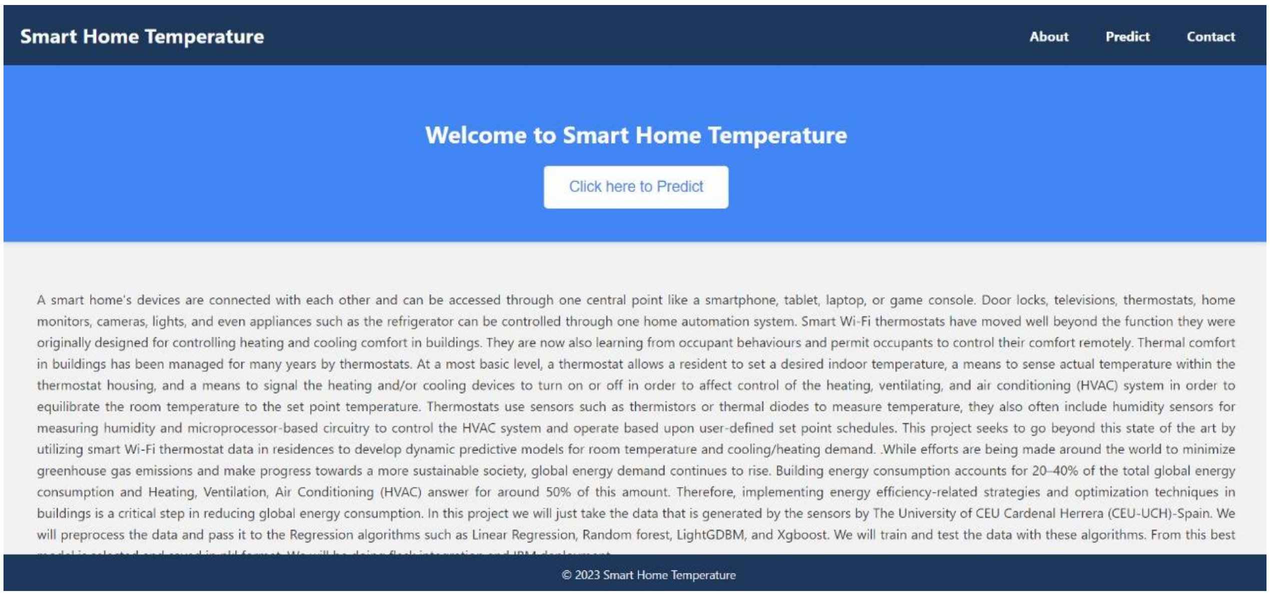
In this case, the performance of the XGBoost model is slightly lower than that of the initial model, as evidenced by the lower r2_score. This indicates that the XGBoost model may not be the best choice for this dataset.

9. RESULTS



```
(base) C:\Users\kaler>cd desktop

(base) C:\Users\kaler\Desktop>cd Main folder

(base) C:\Users\kaler\Desktop\Main folder>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 120-001-211
```

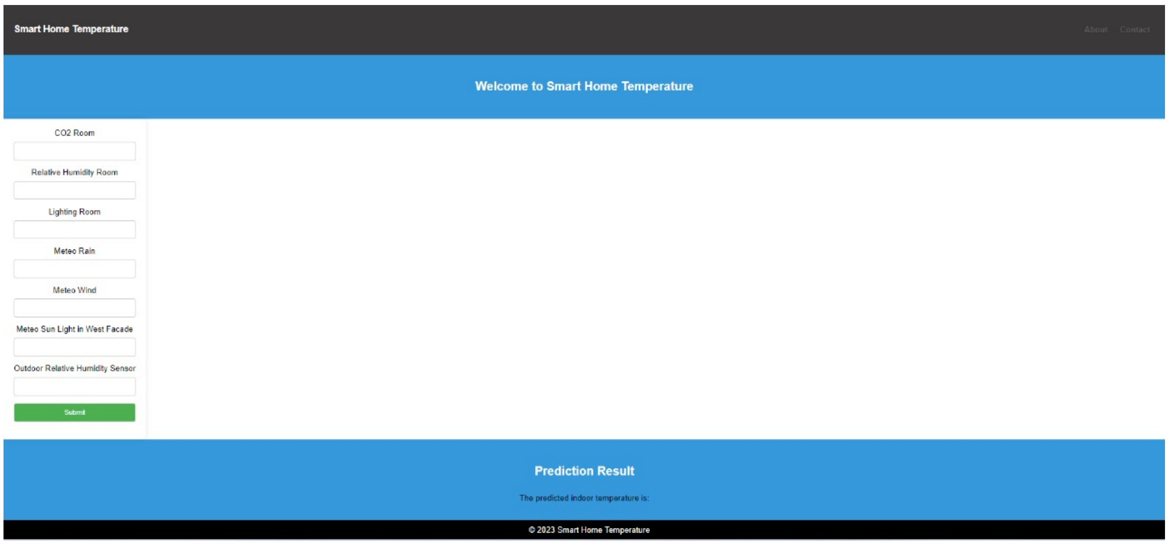
Figure 1



A smart home's devices are connected with each other and can be accessed through one central point like a smartphone, tablet, laptop, or game console. Door locks, televisions, thermostats, home monitors, cameras, lights, and even appliances such as the refrigerator can be controlled through one home automation system. Smart Wi-Fi thermostats have moved well beyond the function they were originally designed for controlling heating and cooling comfort in buildings. They are now also learning from occupant behaviours and permit occupants to control their comfort remotely. Thermal comfort in buildings has been managed for many years by thermostats. At a most basic level, a thermostat allows a resident to set a desired indoor temperature, a means to sense actual temperature within the thermostat housing, and a means to signal the heating and/or cooling devices to turn on or off in order to affect control of the heating, ventilating, and air conditioning (HVAC) system in order to equilibrate the room temperature to the set point temperature. Thermostats use sensors such as thermistors or thermal diodes to measure temperature, they also often include humidity sensors for measuring humidity and microprocessor-based circuitry to control the HVAC system and operate based upon user-defined set point schedules. This project seeks to go beyond this state of the art by utilizing smart Wi-Fi thermostat data in residences to develop dynamic predictive models for room temperature and cooling/heating demand. .While efforts are being made around the world to minimize greenhouse gas emissions and make progress towards a more sustainable society, global energy demand continues to rise. Building energy consumption accounts for 20–40% of the total global energy consumption and Heating, Ventilation, Air Conditioning (HVAC) answer for around 50% of this amount. Therefore, implementing energy efficiency-related strategies and optimization techniques in buildings is a critical step in reducing global energy consumption. In this project we will just take the data that is generated by the sensors by The University of CEU Cardenal Herrera (CEU-UCH)-Spain. We will preprocess the data and pass it to the Regression algorithms such as Linear Regression, Random forest, LightGDBM, and Xgboost. We will train and test the data with these algorithms. From this best

© 2023 Smart Home Temperature

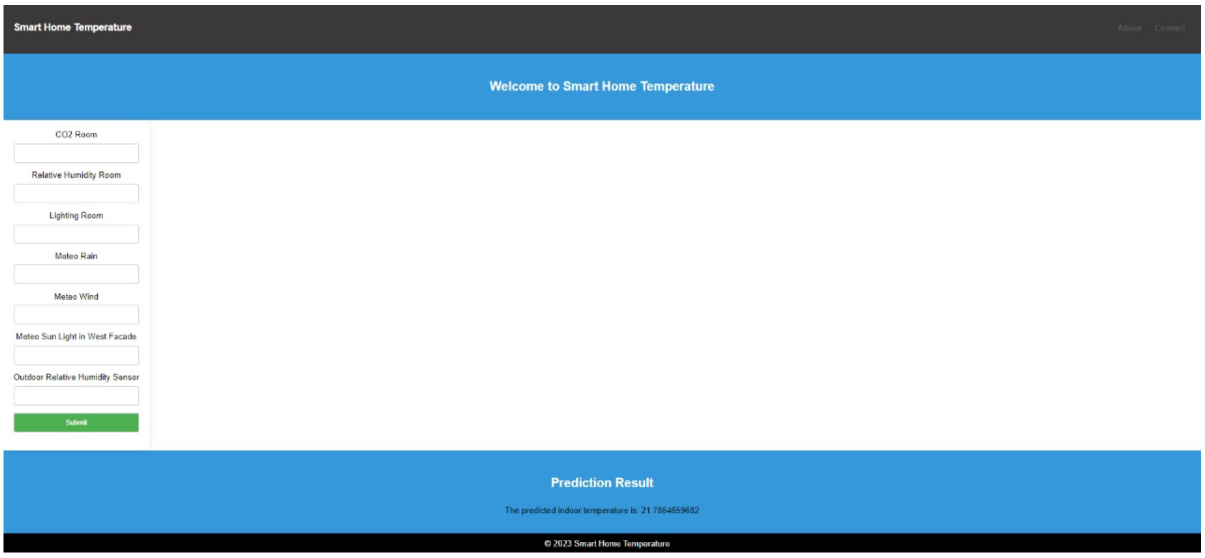
Figure 2



The predicted indoor temperature is:

© 2023 Smart Home Temperature

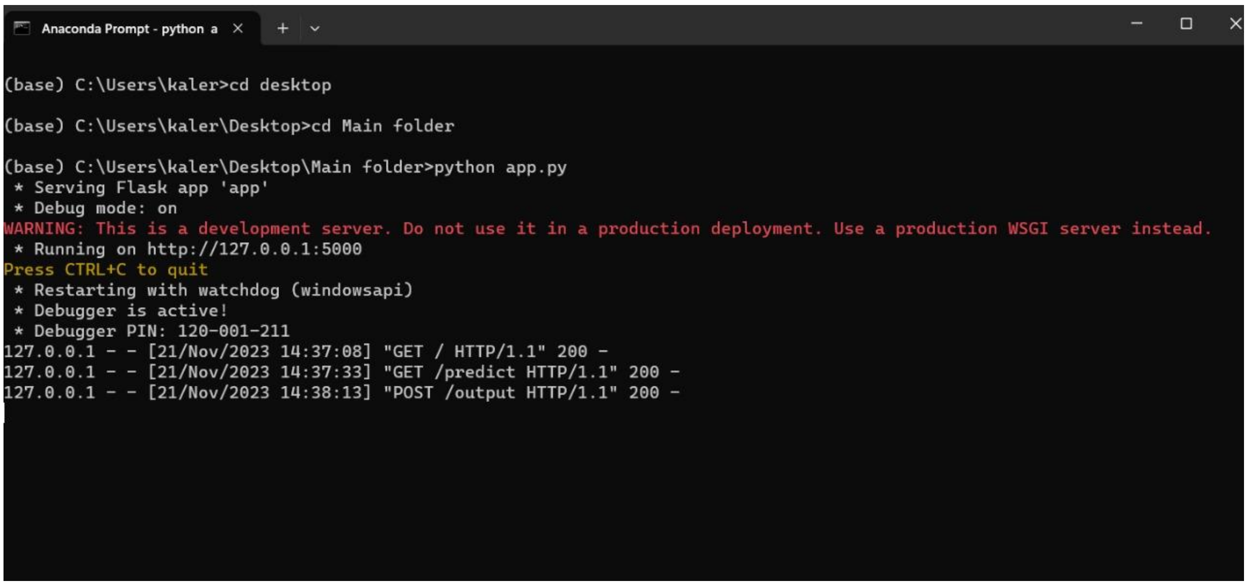
Figure 3



The predicted indoor temperature is: 21.7864659682

© 2023 Smart Home Temperature

Figure 4



```
(base) C:\Users\kaler>cd desktop

(base) C:\Users\kaler\Desktop>cd Main folder

(base) C:\Users\kaler\Desktop\Main folder>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 120-001-211
127.0.0.1 - - [21/Nov/2023 14:37:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 14:37:33] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [21/Nov/2023 14:38:13] "POST /output HTTP/1.1" 200 -
```

Figure 5

10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1)Energy Efficiency: Control of heating, ventilating, and air conditioning systems based on predictions leads to reduced energy consumption and lower utility bills.

2)User Comfort: As the system adjusts to individual preferences, residents experience enhanced comfort.

3)Environmental Impact: The system contributes to lower greenhouse gas emissions.

4)Remote Accessibility: Through a user-friendly interface, users can remotely monitor and control their home's temperature.

5)Data-Driven Insights: Valuable insights into temperature patterns, user behaviors, and energy consumption are generated by the system.

6)Scalability: The deployment of a cloud-like IBM Cloud makes it possible to accommodate a growing number of users and devices.

DISADVANTAGES:

1)Initial Setup Complexity: The installation and setup of smart home temperature prediction systems could pose challenges for some users.

2)Dependency on Data Quality: The accuracy of models depends on the quality and consistency of the data collected.

3)Security Concerns: There are security risks associated with the collection and processing of personal data in smart home systems.

4)Integration Challenges: Ensuring seamless integration with existing smart home devices may pose challenges.

5)Continuous Maintenance: The system needs regular updates and maintenance to keep up with changing needs.

6)Cost Considerations: The initial cost of implementing smart home temperature prediction systems, including sensor installation and device purchase, maybe a barrier for some users.

11. CONCLUSION

In conclusion, the implementation of smart home temperature prediction systems represents a significant leap in residential living, offering benefits that extend beyond traditional HVAC control. A more sustainable lifestyle can be achieved with the integration of predictive models, which help maximize energy consumption and save money. Increased levels of comfort are ensured by the tailored adaptation of these systems to individual user preferences.

The ability for users to remotely manage and monitor their home's temperature adds a layer of convenience and flexibility, aligning with the evolving expectations of modern homeowners. Data-driven provides valuable information, not only on temperature patterns but also on user behaviors and energy consumption trends, facilitating more informed decision-making.

The adoption of Smart Home Temperature Prediction systems comes with considerations. The success of these systems depends on addressing challenges related to the initial setup complexity, data quality dependency, security, integration, and ongoing maintenance. Ensuring widespread acceptance and long-term viability is dependent on striking a balance between security measures and user-friendliness.

A new era in residential climate control has been created by the introduction of smart home temperature prediction systems. As technology continues to evolve, these systems stand as a testament to the ongoing pursuit of innovative solutions that enhance our daily lives while contributing to a more sustainable and connected world.

12. Future Scope

The future scope for Smart Home Temperature Prediction envisions a trajectory of continual evolution and integration, driven by technology and the growing demand for intelligent energy-efficient home management systems. machine learning and artificial intelligence are expected to become more sophisticated as they mature. The integration of emerging technologies, such as edge computing and 5G, holds the promise of real-time data processing, enabling quicker and more responsive adjustments to HVAC systems.

The expansion of the Internet of Things within smart homes will likely lead to increased integration with a diverse range of devices, creating a holistic ecosystem for managing various aspects of home life. The future may see a shift towards more user-centered customization, tailoring temperature predictions not only based on historical data but also on real-time feedback and evolving user habits. Natural language processing and gesture-based controls may redefine the user experience, making smart home temperature prediction systems more intuitive and user-friendly.

The future of Smart Home Temperature Prediction includes a vision of connected, energy-efficient, and user-friendly living environments. A shift towards smart, sustainable homes that prioritize both user comfort and environmental responsibility is reflected in this trajectory.

Appendix:

Index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Home - Smart Home Temperature</title>

  <style>

    body {

      margin: 0;

      padding: 0;

      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

      background-color: #f2f2f2; /* Light gray background */

      color: #333; /* Default text color */
```

```
}

header {
    background-color: #1E385B; /* Dark blue background */
    color: #ffffff;
    padding: 20px;
    text-align: left;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

h3 {
    margin: 0;
    font-size: 24px;
}

nav a {
    color: #ffffff; /* White link color */
    text-decoration: none;
    font-weight: bold;
    margin: 0 20px;
    font-size: 16px;
}

.content {
    padding: 40px;
    text-align: center;
    background-color: #4285F4; /* Google Blue background */
    color: #ffffff; /* White text color */
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

h2 {
    font-size: 28px;
    margin-bottom: 20px;
}

button {
    background-color: #ffffff; /* White button color */
    color: #4285F4;
    padding: 15px 30px;
    border: none;
    cursor: pointer;
    font-size: 18px;
    border-radius: 5px;
}

.maincon {
    padding: 40px;
    text-align: justify;
}

p {
    line-height: 1.6;
    font-size: 16px;
}
```

```
        footer {
            background-color: #1E385B; /* Dark blue background for footer */
            color: #ffffff;
            text-align: center;
            padding: 15px;
            position: fixed;
            bottom: 0;
            width: 100%;
            font-size: 14px;
        }
    </style>
</head>
<body>

<header>
    <h3>Smart Home Temperature</h3>
    <nav>
        <a href="about.html">About</a>
        <a href="predict">Predict</a>
        <a href="contact.html">Contact</a>
    </nav>
</header>

<div class="content">
    <h2>Welcome to Smart Home Temperature</h2>
    <button onclick="window.location.href='/predict'">Click here to Predict</button>
    <!-- Additional content goes here -->
</div>
```

<div class="maincon">

<p>A smart home's devices are connected with each other and can be accessed through one

central point like a smartphone, tablet, laptop, or game console. Door locks, televisions,

thermostats, home monitors, cameras, lights, and even appliances such as the refrigerator can

be controlled through one home automation system. Smart Wi-Fi thermostats have moved

well beyond the function they were originally designed for controlling heating and cooling

comfort in buildings. They are now also learning from occupant behaviours and permit

occupants to control their comfort remotely. Thermal comfort in buildings has been managed

for many years by thermostats. At a most basic level, a thermostat allows a resident to set a

desired indoor temperature, a means to sense actual temperature within the thermostat

housing, and a means to signal the heating and/or cooling devices to turn on or off in order to

affect control of the heating, ventilating, and air conditioning (HVAC) system in order to

equilibrate the room temperature to the set point temperature. Thermostats use sensors such

as thermistors or thermal diodes to measure temperature, they also often include humidity

sensors for measuring humidity and microprocessor-based circuitry to control the HVAC

system and operate based upon user-defined set point schedules. This project seeks to go

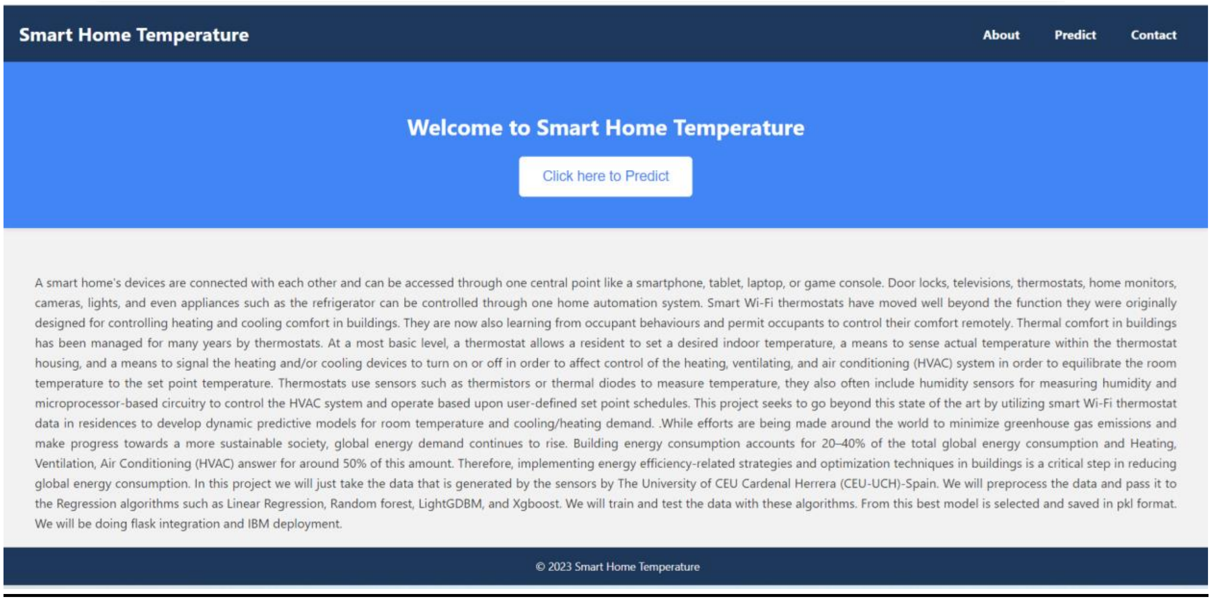
beyond this state of the art by utilizing smart Wi-Fi thermostat data in residences to develop

dynamic predictive models for room temperature and cooling/heating demand. .While efforts

are being made around the world to minimize greenhouse gas emissions and make progress

towards a more sustainable society, global energy demand continues to rise. Building energy

```html
    consumption accounts for 20–40% of the total global energy consumption and Heating,
    Ventilation, Air Conditioning (HVAC) answer for around 50% of this amount. Therefore,
    implementing energy efficiency-related strategies and optimization techniques in
buildings is
    a critical step in reducing global energy consumption.
    In this project we will just take the data that is generated by the sensors by The University
of
    CEU Cardenal Herrera (CEU-UCH)-Spain. We will preprocess the data and pass it to the
    Regression algorithms such as Linear Regression, Random forest, LightGDBM, and
Xgboost.
    We will train and test the data with these algorithms. From this best model is selected
and
    saved in pkl format. We will be doing flask integration and IBM deployment.</p>
</div>

<footer>
  &copy; 2023 Smart Home Temperature
</footer>

</body>
</html>
```



A smart home's devices are connected with each other and can be accessed through one central point like a smartphone, tablet, laptop, or game console. Door locks, televisions, thermostats, home monitors, cameras, lights, and even appliances such as the refrigerator can be controlled through one home automation system. Smart Wi-Fi thermostats have moved well beyond the function they were originally designed for controlling heating and cooling comfort in buildings. They are now also learning from occupant behaviours and permit occupants to control their comfort remotely. Thermal comfort in buildings has been managed for many years by thermostats. At a most basic level, a thermostat allows a resident to set a desired indoor temperature, a means to sense actual temperature within the thermostat housing, and a means to signal the heating and/or cooling devices to turn on or off in order to affect control of the heating, ventilating, and air conditioning (HVAC) system in order to equilibrate the room temperature to the set point temperature. Thermostats use sensors such as thermistors or thermal diodes to measure temperature, they also often include humidity sensors for measuring humidity and microprocessor-based circuitry to control the HVAC system and operate based upon user-defined set point schedules. This project seeks to go beyond this state of the art by utilizing smart Wi-Fi thermostat data in residences to develop dynamic predictive models for room temperature and cooling/heating demand. .While efforts are being made around the world to minimize greenhouse gas emissions and make progress towards a more sustainable society, global energy demand continues to rise. Building energy consumption accounts for 20–40% of the total global energy consumption and Heating, Ventilation, Air Conditioning (HVAC) answer for around 50% of this amount. Therefore, implementing energy efficiency-related strategies and optimization techniques in buildings is a critical step in reducing global energy consumption. In this project we will just take the data that is generated by the sensors by The University of CEU Cardenal Herrera (CEU-UCH)-Spain. We will preprocess the data and pass it to the Regression algorithms such as Linear Regression, Random forest, LightGDBM, and Xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

© 2023 Smart Home Temperature

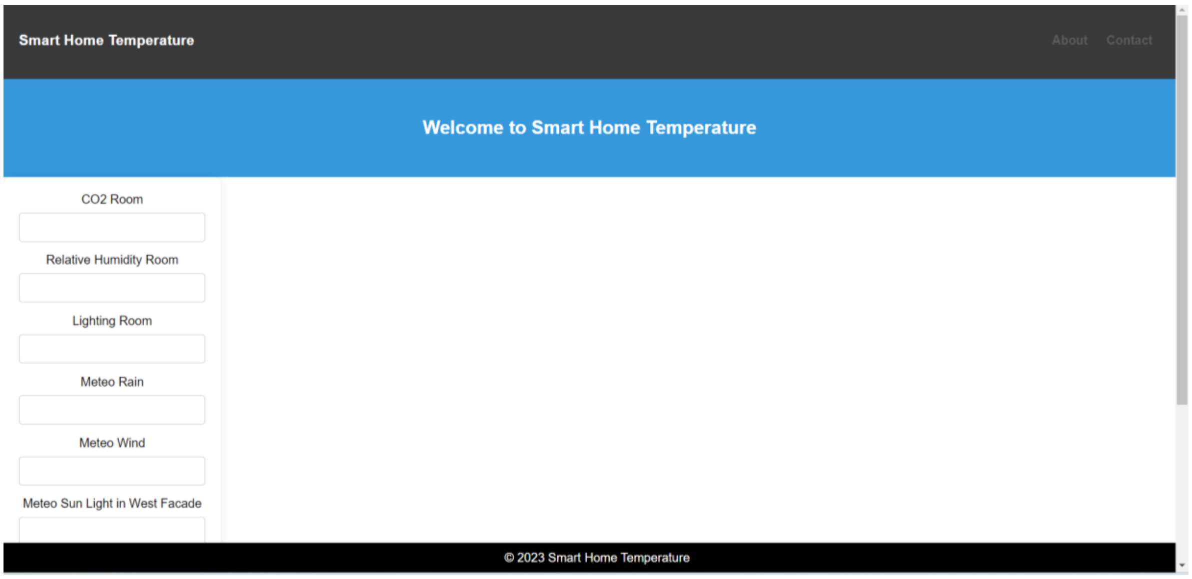
<u>Predict.html:</u>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Home - Smart Home Temperature</title>
  <style>

form {
        display: inline-block;
        text-align: center;
        background-color: #fff; /* Set your background color */
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    }

    label {
        display: block;
        margin-bottom: 8px;
    }

    input {
        width: 100%;
        padding: 10px;
        margin-bottom: 15px;
        box-sizing: border-box;
        border: 1px solid #ccc;
```

```
            border-radius: 4px;
        }

        input[type="submit"] {
            background-color: #4CAF50; /* Set your background color */
            color: #fff;
            cursor: pointer;
        }
        body {
            margin: 0;
            padding: 0;
            font-family: Arial, sans-serif;
            background-color: #ffffff; /* White background */
            color: #000000; /* Default text color */
        }

       header {
    background-color: rgb(58, 57, 57);
    color: #ffffff;
    padding: 20px; /* Adjust this value as needed */
    text-align: left;
    display: flex;
    justify-content: space-between;
    align-items: center;
}



        nav a {
            color: #5f5e5e; /* Default link color */
            text-decoration: none;
            font-weight: bold;
            margin: 0 10px;
        }

        .content {
            padding: 20px;
            text-align: center;
            background-color: #3498db;
        }
        .maincon {
            padding: 20px;
            text-align: center;
        }

        h2 {
            color: #ffffff; /* White text color */
            /* Blue background */
            padding: 10px;
        }

        button {
    background-color: #3498db; /* Blue color for consistency */
    color: #ffffff;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
    font-size: 16px;
    border-radius: 4px; /* Add rounded corners for a modern look */
```

```
}



      footer {
    background-color: #000000;
    color: #ffffff;
    text-align: center;
    padding: 10px;
    position: fixed;
    bottom: 0;
    width: 100%;
}

    </style>
</head>
<body>

<header>
    <h3>Smart Home Temperature</h3>
    <nav>
        <a href="about.html">About</a>
        <a href="contact.html">Contact</a>
    </nav>
</header>

<div class="content">
    <h2>Welcome to Smart Home Temperature</h2>
</div>
<form action="/output" method="post">
```

```
    <label for="CO2_room">CO2 Room</label>
    <input type="number" id="CO2_room" name="CO2_room" required><br>

    <label for="Relative_humidity_room">Relative Humidity Room</label>
    <input type="number" id="Relative_humidity_room" name="Relative_humidity_room"
required><br>

    <label for="Lighting_room">Lighting Room</label>
    <input type="number" id="Lighting_room" name="Lighting_room" required><br>

    <label for="Meteo_Rain">Meteo Rain</label>
    <input type="number" id="Meteo_Rain" name="Meteo_Rain" required><br>

    <label for="Meteo_Wind">Meteo Wind</label>
    <input type="number" id="Meteo_Wind" name="Meteo_Wind" required><br>

    <label for="Meteo_Sun_light_in_west_facade">Meteo Sun Light in West Facade</label>
    <input type="number" id="Meteo_Sun_light_in_west_facade"
name="Meteo_Sun_light_in_west_facade" required><br>

    <label for="Outdoor_relative_humidity_Sensor">Outdoor Relative Humidity
Sensor</label>
    <input type="number" id="Outdoor_relative_humidity_Sensor"
name="Outdoor_relative_humidity_Sensor" required><br>

    <input type="submit" value="Submit">
</form>

<div class="content">
    <h2>Prediction Result</h2>
    <p>The predicted indoor temperature is: {{ indoor_temperature_prediction }}</p>
</div>


<footer>
    &copy; 2023 Smart Home Temperature
</footer>

</body>
</html>
```



Smart Home Temperature

About Contact

Welcome to Smart Home Temperature

CO2 Room

Relative Humidity Room

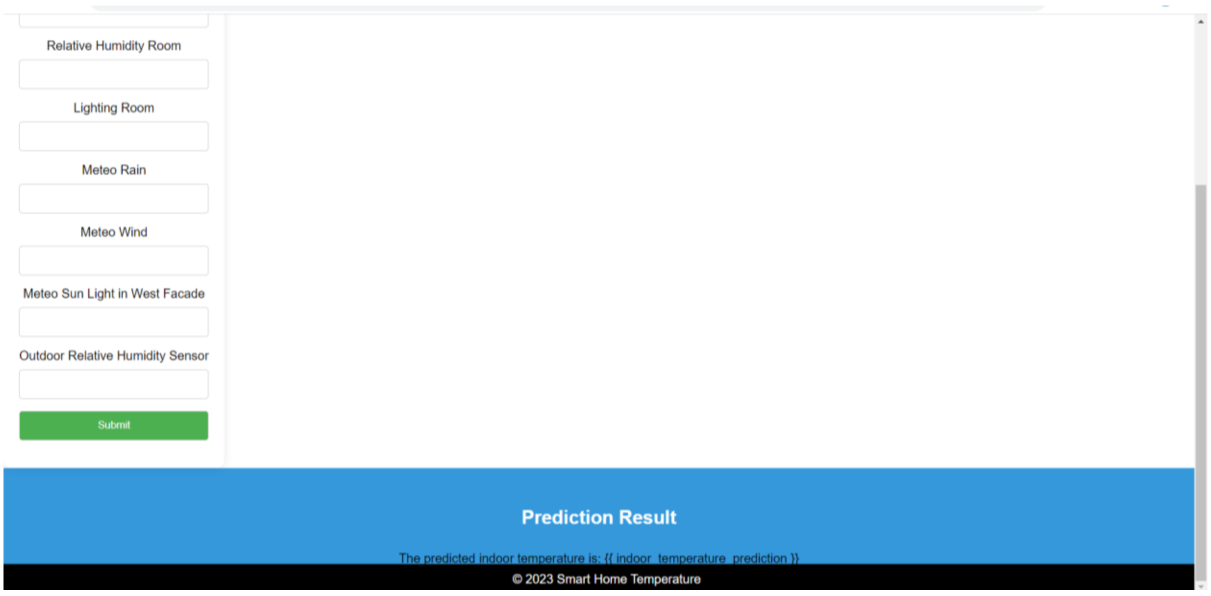
Lighting Room

Meteo Rain

Meteo Wind

Meteo Sun Light in West Facade

© 2023 Smart Home Temperature



Relative Humidity Room

Lighting Room

Meteo Rain

Meteo Wind

Meteo Sun Light in West Facade

Outdoor Relative Humidity Sensor

Submit

Prediction Result

The predicted indoor temperature is: {{ indoor_temperature_prediction }}

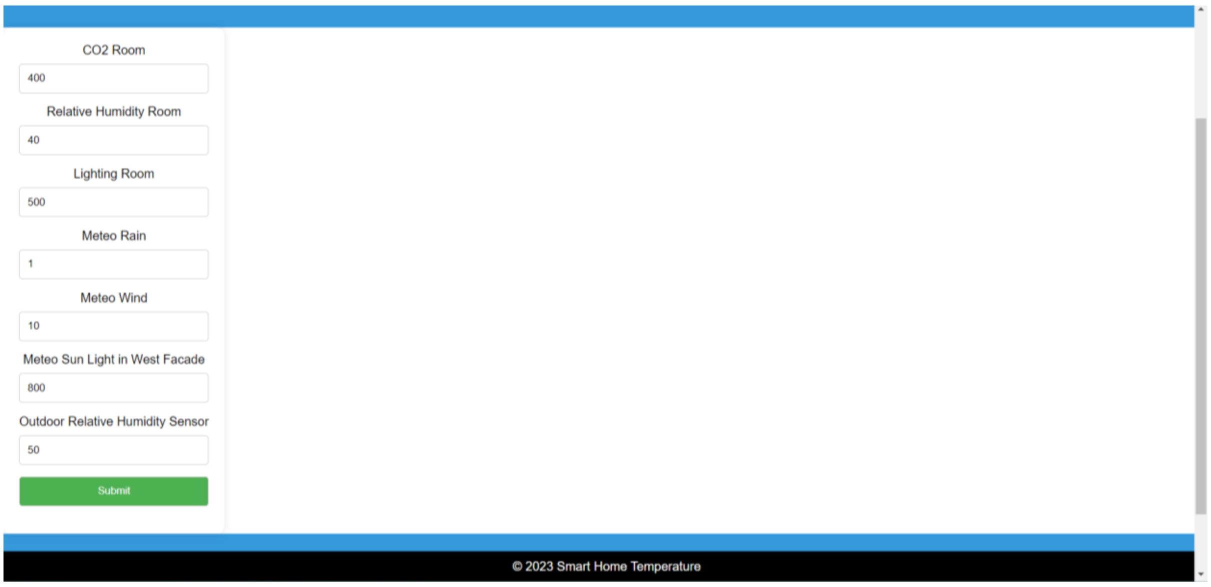
© 2023 Smart Home Temperature

App.py:

```
from flask import Flask, request, render_template

import pickle

import numpy as np

import pandas as pd


model = pickle.load(open(r'C:\Users\tejos\Desktop\Main folder\temperature.pkl', 'rb'))

app = Flask(__name__)


# Define the feature names used in your model

names = ['CO2_room', 'Relative_humidity_room', 'Lighting_room', 'Meteo_Rain', 'Meteo_Wind',

      'Meteo_Sun_light_in_west_facade', 'Outdoor_relative_humidity_Sensor']


@app.route("/")

def home():

  return render_template("index.html")


@app.route("/predict")

def predict():

  return render_template("predict.html")


@app.route('/output', methods=['POST', 'GET'])

def output():

  try:

    input_feature = [float(request.form.get(name, 0.0)) for name in names]

    data = pd.DataFrame([input_feature], columns=names)

    prediction = model.predict(data)

    indoor_temperature_prediction = prediction[0]  # Extract the predicted indoor temperature
```

```
        return                                                  render_template('predict.html',
indoor_temperature_prediction=indoor_temperature_prediction)

    except Exception as e:
        print("Error:", e)
        return f"An error occurred: {str(e)}"

if __name__ == '__main__':
    app.run(debug=True)
```

OUTPUT FOR PROJECT:



CO2 Room

400

Relative Humidity Room

40

Lighting Room

500

Meteo Rain

1

Meteo Wind

10

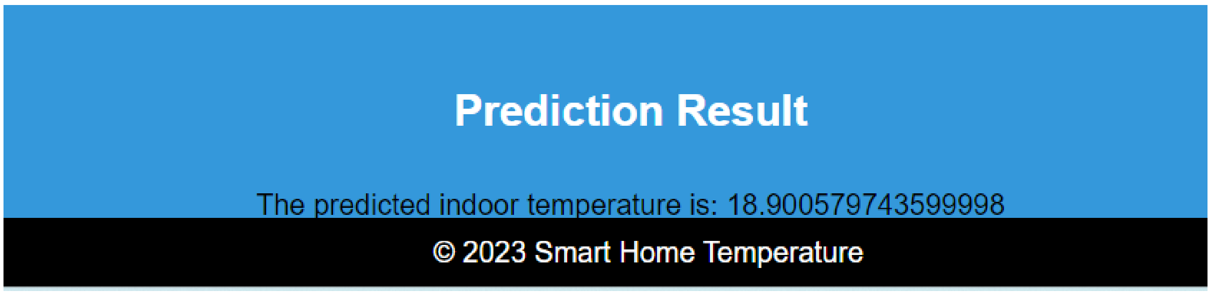
Meteo Sun Light in West Facade

800

Outdoor Relative Humidity Sensor

50

© 2023 Smart Home Temperature



Prediction Result

The predicted indoor temperature is: 18.900579743599998

© 2023 Smart Home Temperature

DONE BY –

Samhitha Revanur(21BCB7025) - samhitha.21bcb7025@vitapstudent.ac.in

Kaleru Abhilash(21BCB7033) – abhilash.21bcb7033@vitapstudent.ac.in

Desai Hinduja(21BCB7070) - hinduja.21bcb7070@vitapstudent.ac.in

Harsha Vardhan(21BCB7059) - harshavardhan.21bcb7059@vitapstudent.ac.in