2023

Smart Home – Temperature Prediction

PRROJECT REPORT

TEAM - 592148

Table of Content

INTRODUCTION

1.1. Project Overview

The gadgets in a smart home are interconnected and may be accessed from a single hub, such as a laptop, gaming console, tablet, or smartphone. One home automation system may operate door locks, TVs, thermostats, cameras, lights, cameras, cameras, cameras, and even appliances like the refrigerator. The purpose of smart Wi-Fi thermostats has evolved significantly from its initial design, which was to regulate indoor temperature. Additionally, they are now able to learn from the actions of their inhabitants and provide them remote control over their comfort. Thermostats have been used for many years to control thermal comfort in buildings. In its most basic form, a thermostat is a device that lets residents set a desired indoor temperature, sense the actual temperature inside the thermostat housing, and send signals to the heating and/or cooling devices to turn on or off. These functions help residents control the HVAC system, which regulates the temperature of the room to the desired temperature. Thermostats monitor temperature using sensors like thermistors or thermal diodes. They also frequently have humidity sensors and microprocessor-based circuitry that controls the HVAC system and runs on user-specified set point schedules. By using smart Wi-Fi thermostat data from homes to create dynamic predictive models for room temperature and cooling/heating needs, this project aims to surpass the current state of the art. Global energy consumption is rising even as efforts are made everywhere to reduce greenhouse gas emissions and move towards a more sustainable society. Heating, ventilation, and air conditioning (HVAC) accounts for around half of the building energy consumption, which accounts for 20–40% of the world's overall energy consumption. Therefore, a crucial first step in lowering global energy consumption is the implementation of energy efficiency-related methods and optimisation techniques in buildings.

1.2. Purpose

We shall only use the sensor data from The University of CEU Cardenal Herrera (CEU-UCH) in Spain for this study. The data will be pre-processed before being sent to regression methods including Xgboost, Random Forest, LightGDBM, and Linear Regression.

We will use these methods to train and test the data. The best model is chosen from this group and stored in PKL format. We'll be deploying IBM and integrating Flask

LITERATURE SURVEY

2.1. Existing Problem

W Jin, I Ullah, (2019), stated that Occupant comfort management is crucial in smart homes, aiming to achieve high comfort levels while minimizing energy consumption. Learning models can predict factors of a mathematical model for optimal results without expensive experiments. This paper proposes occupant comfort management based on energy optimization using an environment prediction model. The model provides optimal power consumption based on the objective function, requiring temperature and comfort index data as input parameters. A recurrent neural network with a pre-collected dataset is used for temperature and humidity prediction. The comfort index model derives the predicted mean vote value for the energy optimization model. Experimental results show an 8.43% reduction in optimized power consumption compared to actual power consumption using mean absolute percentage error. The emulation of an indoor environment using optimal energy consumption presents approximately similar to actual data. An improved energy optimization model is proposed to recommend optimal power consumption for operating heaters, which requires changing indoor environmental factors like temperature and humidity. Indoor temperature and humidity are used as input parameters, and RNN and LSTM are used for prediction models. The experiment exploits MAPE to predict indoor temperature and humidity values, resulting in an 8.43% reduction in optimized power consumption for 10,400 records. The indoor temperature data results in a 0.42% reduction, and indoor humidity data results in a 9.99% difference. The proposed optimization scheme saves energy and provides a comfortable environment by using the optimal power consumption to operate the heater.

H Xu, Y He (2020), mentioned that Globally, residential structures and commercial shops consume a substantial amount of energy, with an increasing helpful energy demand ratio of 1:3. This is a result of energy's limited supply, which is mostly dictated by the power rating and pattern of energy use between the building system and its inhabitant. Turning loads is used to calculate the design specifications for the heating and cooling systems in order to ensure thermal comfort in the conditioned area .Complex energy simulation technologies like as Energy Plus, Open Studio, and

Autodesk Revit are used to estimate the energy consumption of buildings; however, because of domain-specific knowledge, these findings are extremely reliant on complexity, dataset distribution, and time consumption. Web-based technologies have been revolutionised by machine learning classifiers, which offer a comprehensive grasp of energy-saving building and cost-effectiveness.

I Priyadarshini, S Sahu,(2022), quoted in their paper that Smart homes are equipped with automation gadgets that simplify our lives by enabling homeowners to remotely operate appliances and other equipment via the Internet of Things (IoT) platform. Energy consumption analysis is essential for convenience and financial savings since these gadgets are able to learn the schedules of the homes and adjust accordingly. An greater carbon footprint, a higher danger of climate change, and an increase in supply demand are all caused by rising energy usage. Keeping an eye on energy usage is crucial for cost management and identifying areas for savings .Using machine learning models such as Decision Trees (DT), Random Forest (RF), eXtreme Gradient Boosting (XGBoost), and k-Nearest Neighbour (KNN), this article conducts an overall analysis of energy usage in smart homes. An ensemble model based on DT-RF-XG Boost is proposed in the paper to analyse consumption and compare it with baseline techniques. The suggested ensemble model performs better than all existing baseline methods in a variety of datasets and disciplines. We want to use other machine learning approaches, such as neural networks and optimisation algorithms, to track energy usage in the future. Deep learning, time series analysis, and other sophisticated algorithms could be adopted to solve issues with energy usage globally as machine learning techniques continue to gain traction. We may look forward to developing dependable solutions for consistent energy usage after consumption has been examined.

P Hietaharju, M Ruusunen, (2018), said that By offering forecast data on heat demand, a unique dynamic model for interior temperature prediction and management has been created to increase energy efficiency. Real measurement data from five distinct building types were used to analyse the model, making use of readily accessible data from many sources. The number of input variables and parameters was kept to a minimum by selecting a model structure with few free parameters

and input variables. Using the same model framework and actual data from five buildings, simulations revealed an average modelling error of less than 5% throughout the course of the 28-hour forecast horizon. Large buildings' interior temperatures may be predicted using this standardised and simple-to-use model framework, allowing for demand side management and predictive optimisation of heat demand at the city level. The model's accuracy and simplicity make it a valuable tool for managing indoor temperature in buildings.

B Thomas, & M Soleimani-Mohseni, (2007), put forward that The issue of finding interior climate forecast models for buildings is covered in the article. Through identification trials, two buildings were recognised using non-linear artificial neural network models (ANN-models) of varying orders in addition to linear ARX-, ARMAX-, and BJ-models. Numerous input signals, including temperature both inside and outside, heating power, wall temperatures, ventilation flow rate, time of day, and solar radiation, were employed by the models. It was shown that ANN-models could forecast temperatures more accurately than linear models. The first building demonstrated that the time of day and sun radiation interact nonlinearly to affect inside temperature prediction. The second building demonstrated that the relationship between ventilation flow rate and inside temperature is nonlinear. The paper comes to the conclusion that non-linear ANN-models are a fascinating model structure for indoor climate in buildings. They can be obtained through identification experiments of almost the same type as linear models. The neural network's more flexible structure allows it to better deal with non-linear systems.

K Yao, K C., Huang (2018), Their study aimed to create a smart home using an AI model integrated with the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) application for environment control. The input data included outdoor temperature, indoor temperature, humidity, illumination, and indoor person count. The output control decisions included air conditioners, dehumidifiers, power curtains, and lights. An artificial neural network was used to process the data for machine learning, aiming to achieve a comfortable environment. The control decisions were analysed for model loss and accuracy. LabVIEW was used for designing the sensing component, data display, and control interface, while Python was used to establish the intelligent model.

Remote sensing and control were fulfilled using the web publishing tool built into LabVIEW. The study successfully achieved home environment sensing and prediction, achieving smart home control. The system can be applied to various systems, including robots and vehicles, to enhance their AI capabilities. The combination of LabVIEW's excellent monitoring and control interface and Python's AI algorithm calculation capabilities offers advantages in this area.

FMJ Shamrat, SM Allayear,(2018), Proposed that In the modern world, room temperature prediction in air conditioners is a difficult and unclear task. Hardware like a Raspberry Pi Zero, IR sensor, heat sensor, microcontroller, and the room's AC with an existing remote are utilised to construct a system. Python programming is used to create an embedded system that implements the suggested system. The data may be used to create a mathematical formula, and an algorithm is created to forecast temperature data using the numbers from the two sensors. The AC switches on or off automatically based on temperature readings from the sensors. Any smart AC space, including those for the disabled, private rooms, conference rooms, hallways, schools and transportation where manual control is impractical, can use this system. As it is challenging to adjust temperature sporadically for those who are disabled, during sleeping hours, or during meetings, temperature prediction in air conditioners is becoming more and more significant in the Internet of Things space. It might be more effective if temperature prediction is based on object temperature because many studies deal with room temperature rather than object temperature .These days, automation and room temperature prediction are difficult problems. Using Python programming, a smart AC system room with temperature prediction and mechanisation has been planned, produced, and put into use. The method, which uses machine learning, artificial intelligence, and expert systems to transform any manual AC system room into a smart AC system environment, has been the subject of several trials and analyses.

H Yar, .AS Imran,(2021), put forward that The home sector has undergone a transformation thanks to the Internet of Things (IoT), which has improved convenience, efficiency, and security. But there's a demand for intelligent home apps that address several parts of the house at once, such automation, safety, security, and energy conservation with less bandwidth, processing, and

expense. Based on the Internet of Things and Edge Computing paradigm, this research paper suggests an affordable integrated system for smart homes that ensures security and safety while offering remote and automated control over household equipment .The suggested method protects client privacy by using edge computing to store sensitive data in a local cloud. Data from scalar and visual sensors are processed and stored on the edge device (RPI), which lowers the costs associated with compute, bandwidth, and storage. In order to create intelligent predictions and improve current functionality, such as human fall detection, interior fire and smoke detection, and suspicious behaviour recognition, the system gathers data from a variety of sensors, including cameras and temperature. Deep learning models may be used to provide personalised resource management by understanding how different home gadgets are used by individuals. In order to provide greater functionality and lower storage and bandwidth costs, the researchers intend to incorporate effective, deeply integrated vision-enabled technology into the smart home in the future. They also intend to employ the RPI as a processing unit to analyse multimedia data, such as surveillance footage.

2.2. Reference

Jin, W., Ullah, I., Ahmad, S., & Kim, D. (2019). Occupant comfort management based on energy optimization using an environment prediction model in smart homes. *Sustainability*, *11*(4), 997.

Xu, H., He, Y., Sun, X., He, J., & Xu, Q. (2020). Prediction of thermal energy inside smart homes using IoT and classifier ensemble techniques. *Computer Communications*, *151*, 581-589.

Priyadarshini, I., Sahu, S., Kumar, R., & Taniar, D. (2022). A machine-learning ensemble model for predicting energy consumption in smart homes. *Internet of Things*, *20*, 100636

Hietaharju, P., Ruusunen, M., & Leiviskä, K. (2018). A dynamic model for indoor temperature prediction in buildings. *Energies*, *11*(6), 1477

Thomas, B., & Soleimani-Mohseni, M. (2007). Artificial neural network models for indoor temperature prediction: investigations in two buildings. *Neural Computing and Applications*, *16*, 81-89.

Yao, K. C., Huang, W. T., Wu, C. C., & Chen, T. Y. (2021). Establishing an AI Model on Data Sensing and Prediction for Smart Home Environment Control Based on LabVIEW. *Mathematical Problems in Engineering*, *2021*, 1-18.

Shamrat, F. J. M., Allayear, S. M., & Jabiullah, M. I. (2018). Implementation of a smart AC automation system with room temperature prediction. *Journal of the Bangladesh Electronic Society*, *18*(1-2), 1816-1510

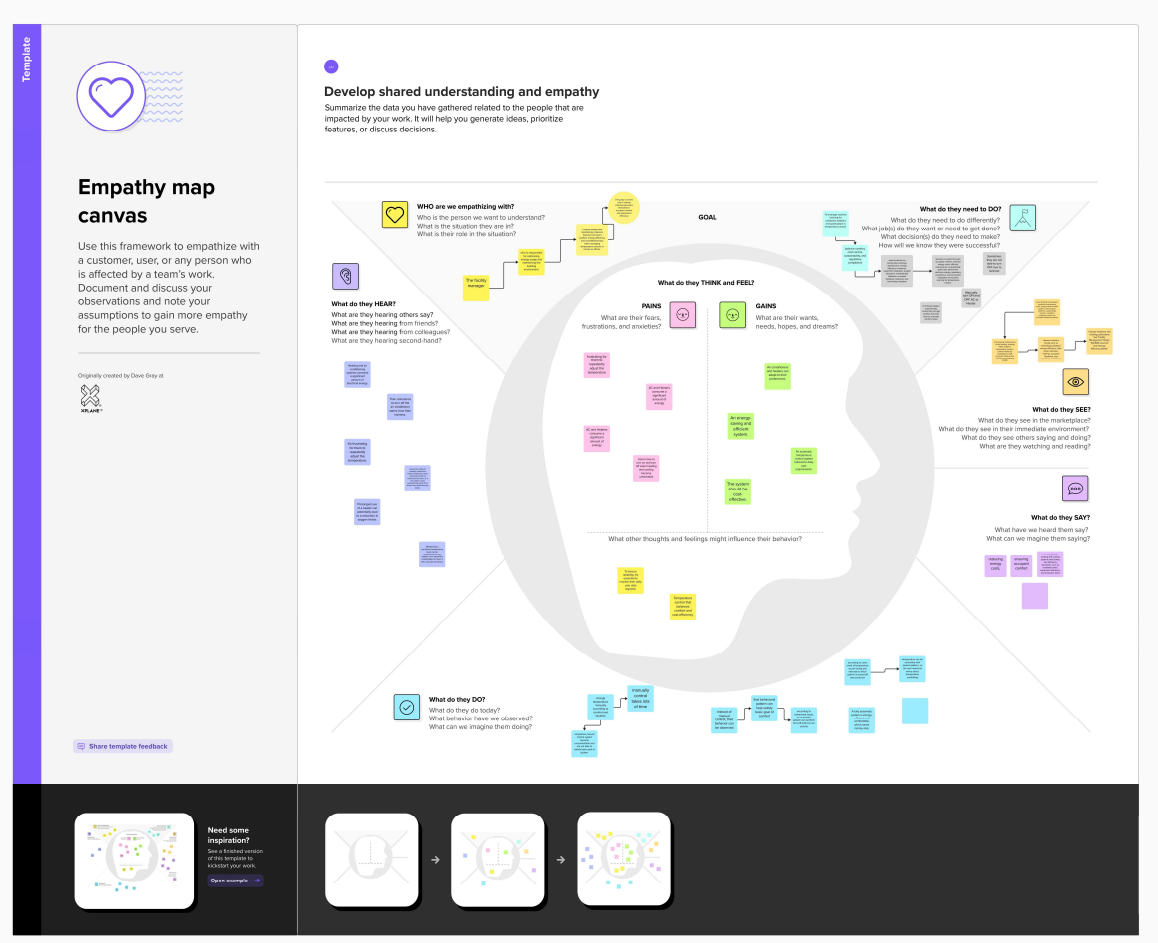
Yar, H., Imran, A. S., Khan, Z. A., Sajjad, M., & Kastrati, Z. (2021). Towards smart home automation using IoT-enabled edge-computing paradigm. *Sensors*, *21*(14), 4932

2.3. Problem Statement Definition

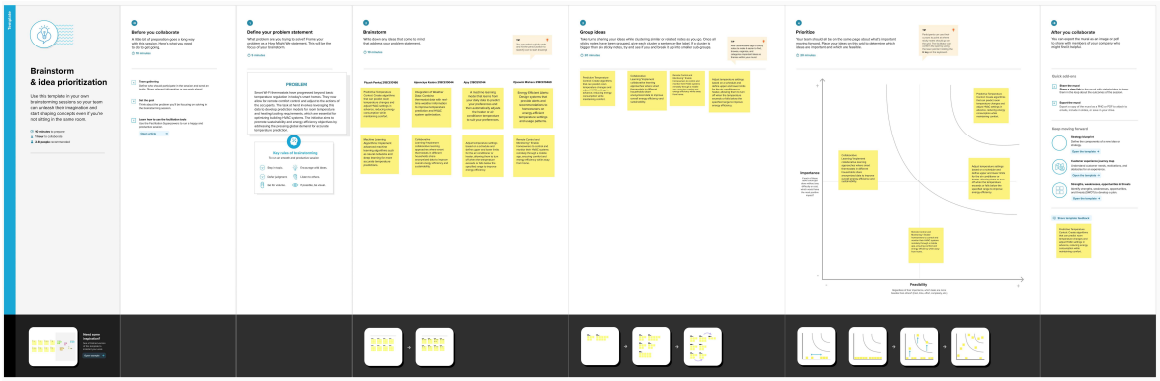
Smart Wi-Fi thermostats have progressed beyond basic temperature regulation in today's smart homes. They now allow for remote comfort control and adjust to the actions of the occupants. The task at hand involves leveraging this data to develop prediction models for room temperature and heating/cooling requirements, which are essential for optimizing building HVAC systems. The initiative aims to promote sustainability and energy efficiency objectives by addressing the pressing global demand for accurate temperature prediction.

IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map and Canvas



3.2. Ideation & Brainstorming



REQUIREMENT ANALYIS

4.1. Functional Analysis

Forecast Data Input:

In order to anticipate room temperature, the system should let users enter a variety of environmental parameters (CO2 levels, humidity, illumination, weather, etc.).

Forecast Results:

The system needs to offer an estimated room temperature after input data has been entered.

Getting Around:

The home page, prediction page, and contact page should all be easy for users to move between.

Regarding this page:

An "About" page with details about the project and its goals ought to be included in the system.

Contact Details:

Users should be able to contact the system through a "Contact" page that contains their contact information.

Designing with responsiveness:

To guarantee a positive user experience across several platforms (desktop, tablet, and mobile), the user interface has to be responsive.

Validation of Data:

To make sure that the input data complies with the specifications and format, the system should validate it.

Integrating Models:

For the purpose of predicting temperature, the prediction page and the machine learning model should work together.

4.2. Non-Functional Analysis

Achievement:

It should be possible for the system to generate predictions in an acceptable amount of time, even when several users are using it at once.

Scalability:

To accommodate more users and data input, the programme has to be scalable.

Dependability:

The predictive system must to be dependable, regularly yielding precise forecasts.

Usability:

A user-friendly interface should have an intuitive design, clear instructions, and both.

Accessible:

There should be less downtime for maintenance and maximum availability and accessibility for users.

Adaptability:

It is imperative for the application to be interoperable with several web browsers in order to provide a consistent user experience.

Sustainability:

It should be simple for developers to maintain and update the codebase if it is well-organized and documented.

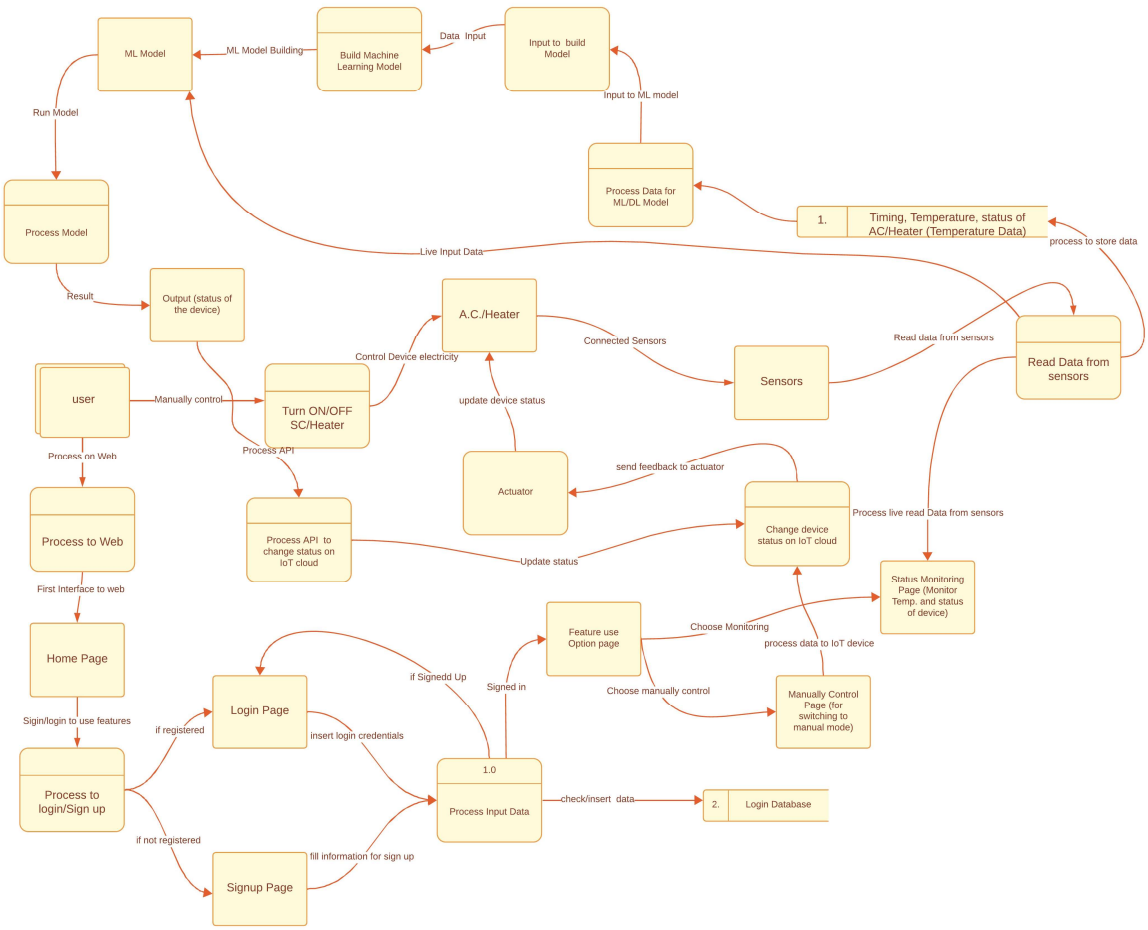
Record-keeping:

Give developers thorough documentation that includes code comments, installation guidelines, and API documentation.

Project Design

5.1. Data Flow Diagram & User Stories

Data Flow Diagram



User Stories

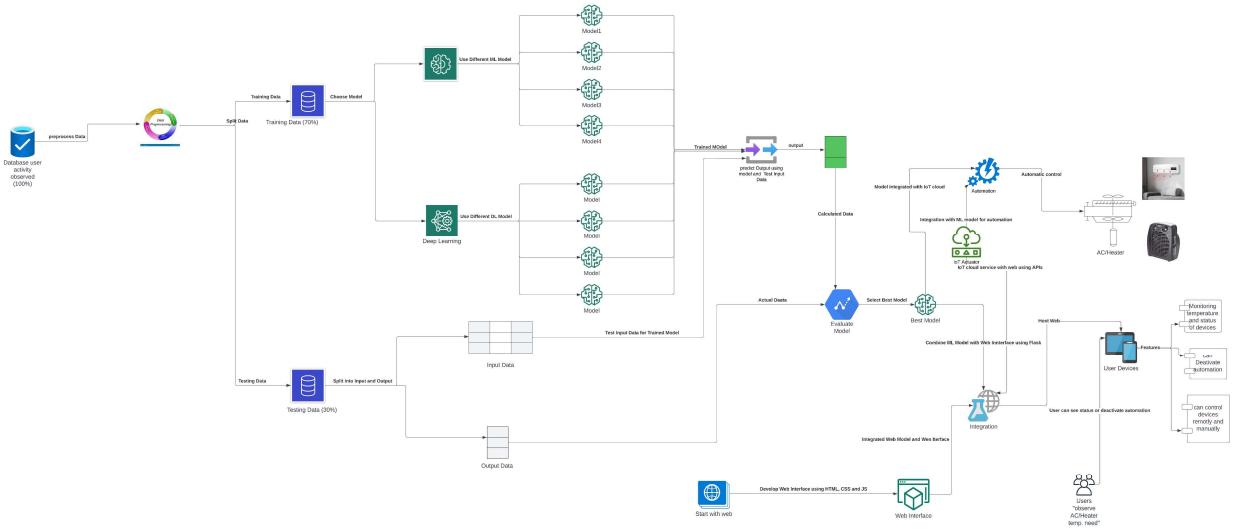
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Visitor to web	Pages Interaction	USN-1	As a Visitor to the Website, I want to be greeted with a welcoming home page that provides an overview of the smart home temperature prediction project.	Upon visiting the home page, I should see a clear and concise overview of the smart home temperature prediction project. The information on the home page should be presented in an engaging and easy-to-read format.	High	Sprint-1
		USN-2	As a Visitor, I want to navigate easily to different sections of the website using a user-friendly navigation menu.	The navigation menu should be prominently displayed on each page. Clicking on menu items should smoothly navigate to the corresponding sections.	High	Sprint-1
		USN-3	As a Visitor, I want to learn more about the project by visiting the "About" page, which provides information about the objectives and purpose of the smart home temperature prediction.	The "About" page should provide detailed information about the project's objectives, features, and the technology used. The content on the "About" page should be informative and engaging.	High	Sprint-1
		USN-4	As a Visitor, I want to access the "Contact Us" page to find contact details for inquiries and support.	The "Contact Us" page should display contact details, including email address and phone number.	Medium	Sprint-1

				The contact details should be accurate and up-to-date.		
User	Prediction	USN-5	As a User, I want to click on a button or link to navigate to the prediction page where I can input various environmental factors for temperature prediction.	The prediction page should be accessible by clicking on a designated button or link. The navigation to the prediction page should be smooth and intuitive.	High	Sprint-1
		USN-6	As a User Inputting Data, I want to input data such as CO2 levels, humidity, lighting, and weather conditions into the system for predicting the room temperature.	The input fields for environmental factors (CO2 levels, humidity, lighting, etc.) should be clearly labelled. Users should be able to enter valid numerical values into the input fields.	High	Sprint-1
		USN-7	As a User Submitting Data, I want to submit the input data and receive a predicted room temperature based on the smart home temperature prediction model.	After submitting the input data, the system should process the information without errors. Users should receive a clear and accurate prediction of room temperature.	High	Sprint-1
		USN-8	As a User Viewing Prediction, I want to see the predicted room temperature displayed clearly on the prediction page.	The predicted room temperature should be displayed prominently on the prediction page. The result should be presented in a format that is easy to understand.	High	Sprint-1
		USN-9	As a User, I want the system to provide feedback or confirmation	After submitting data and viewing the prediction, the system	Low	Sprint-1

			after submitting the input data and receiving the prediction.	should provide visual feedback or confirmation. Users should be able to easily identify the success or failure of their prediction request.		
		USN-10	As a User Navigating, I want to have a consistent and intuitive user interface design across different pages, ensuring a positive user experience.	The user interface design should be consistent across different pages. The design should be responsive, providing a positive user experience on various devices.	high	Sprint-1
		USN-11	As a User, I want the system to be responsive so that I can access and use it on various devices, such as desktops, tablets, and mobile phones.	The system should be responsive, adapting to different screen sizes and orientations. Users should have a consistent and enjoyable experience regardless of the device used.	High	Sprint-1
Developer	Sustainability	USN-12	As a Developer, I want the codebase to be well-documented and organized, making it easy for me to understand and maintain the application.	The codebase should include comprehensive documentation, including a README file, inline comments, and API documentation if applicable. Code should be organized and follow best practices for maintainability.	High	Sprint-1
		USN-13	As a Developer, I want the system to handle errors gracefully and provide informative error messages to users.			Sprint-1
		USN-14	As a Developer, I want to ensure that user data is securely transmitted over the network, protecting user privacy.			Sprint-1
		USN-15	As a Developer, I want the application to be scalable, handling an			Sprint-1

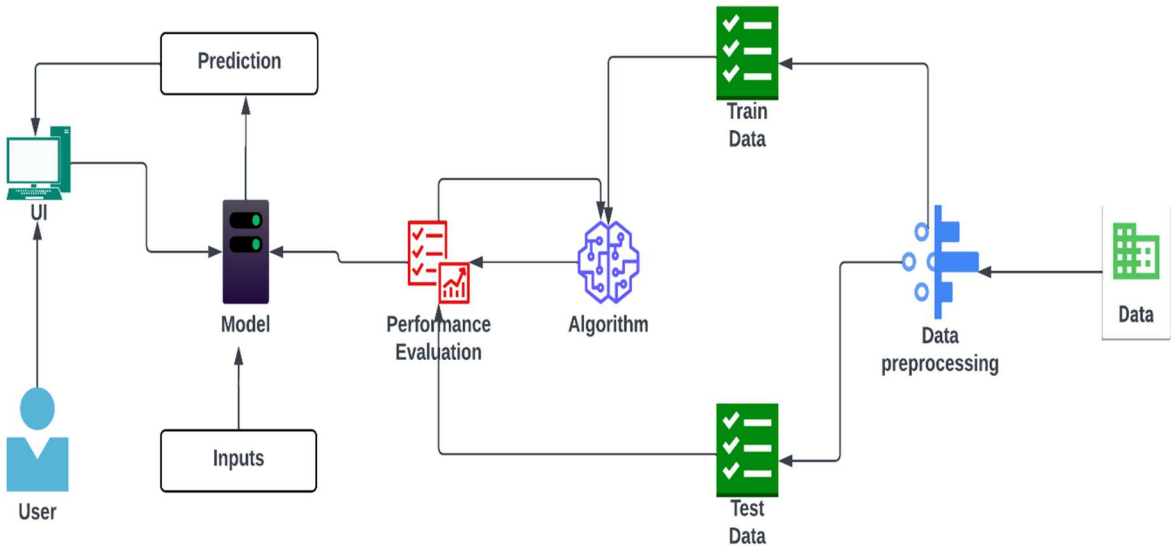
			increasing number of users and input data without compromising performance.			

5.2. Solution Architecture



Project Planning and Scheduling

6.1. Technical Architecture



6.2. Sprint Planning & Estimation

Components & Technologies:

S.No	Component	Description	Technology
1.	Home Page	The homepage includes a navigation menu for easy access to different sections.	HTML& CSS
2.	Contact Page	Users can reach out for inquiries or support.	HTML& CSS
3.	Prediction Page	The prediction page includes input fields for various environmental factors, such as CO2 levels, humidity, lighting, etc.	HTML& CSS
4.	Prediction Algorithm	The system incorporates a predictive model to generate room temperature predictions based on the provided input data.	Python, Numpy, Pandas, Scikit-learn, Matplotlib, Scipy, Pickle-mixin, seaborn
5.	Resulting Display	System displays the predicted room temperature prominently on the prediction page.	Python, Flask, pickle

6.	Responsive Design	UI is designed to be responsive, providing a positive user experience on various conditions.	HTML,CSS, Js & Python
7.	Navigation	Navigation menu allows users to easily move between different pages.	HTML,CSS, Js & Python
8.	User Interaction	Users can interact with the system by inputting data, submitting the form, and receiving predictions.	HTML,CSS & Js
9.	Machine Learning Model	To predict indoor temperature using different parameters.	Python, Numpy, Pandas, Scikit-learn, Matplotlib, Scipy, Pickle-mixin, seaborn

Application Characteristics:

S.No	Characteristics	Description	Technology
1.	User interface	The user interface provides an intuitive and responsive platform for users to input environmental parameters and receive predictions for room temperature in a smart home.	HTML, CSS, VS Code, GitHub
2.	ML model	The machine learning model predicts room temperature based on various environmental factors in a smart home.	Python, Numpy, Pandas, Scikit-learn, Matplotlib, Scipy, Pickle-mixin, seaborn, jupyter-notebook, anaconda navigator
3.	Integration	The system seamlessly integrates Flask web framework with a machine learning model, allowing users to predict and control room temperature in a smart home environment.	Flas & Spyder
4.	Availability	The complete project, including codes and the machine learning model, is available on GitHub for easy access and collaboration.	GitHub
5.	Scalability	The system demonstrates scalability by efficiently handling user interactions and data processing, ensuring its adaptability to increased user load or future feature expansions.	GitHub

6.3. Sprint Delivery Schedule

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint -1	Pages Interaction	USN-1	As a Visitor to the Website, I want to be greeted with a welcoming home page that provides an overview of the smart home temperature prediction project.	2	High	• Piyush Pankaj • Ajay
Sprint -1		USN-2	As a Visitor, I want to navigate easily to different sections of the website using a user-friendly navigation menu.	1	High	• Ajay
Sprint -1		USN-3	As a Visitor, I want to learn more about the project by visiting the "About" page, which provides information about the objectives and purpose of the smart home temperature prediction.	2	High	• Ajeencky a Kaabra • Ojaswini Mishra
Sprint -1		USN-4	As a Visitor, I want to access the "Contact Us" page to find	2	Medium	• Piyush Pankaj • Ojaswini Mishra

			contact details for inquiries and support.			
Sprint -2	Prediction	USN-5	As a User, I want to click on a button or link to navigate to the prediction page where I can input various environmental factors for temperature prediction.	1	High	• Ajay • Ajeencky a Kaabra • Ojaswini Mishra
Sprint -2		USN-6	As a User Inputting Data, I want to input data such as CO2 levels, humidity, lighting, and weather conditions into the system for predicting the room temperature.	1	High	• Piyush Pankaj • Ajeencky a Kaabra
Sprint -2		USN-7	As a User Submitting Data, I want to submit the input data and receive a predicted room temperature based on the smart home temperature prediction model.	2	High	• Ajay • Piyush Pankaj • Ojaswini Mishra
Sprint -2		USN-8	As a User Viewing Prediction, I want to see the predicted room temperature displayed clearly on the	2	High	• Ajay • Ojaswini Mishra

			prediction page.			
Sprint -2		USN-9	As a User, I want the system to provide feedback or confirmation after submitting the input data and receiving the prediction.	1	Low	• Piyush Pankaj
Sprint -2		USN-10	As a User Navigating, I want to have a consistent and intuitive user interface design across different pages, ensuring a positive user experience.	2	high	• Ajeencky a Kaabra • Piyush Pankaj • Ojaswini Mishra
Sprint -2		USN-11	As a User, I want the system to be responsive so that I can access and use it on various devices, such as desktops, tablets, and mobile phones.	2	High	• Piyush Pankaj • Ajay • Ajeencky a Kaabra • Ojaswini Mishra
All	Sustainability	USN-12	As a Developer, I want the codebase to be well-documented and organized, making it easy for me to understand and maintain the application.	1	High	• Piyush Pankaj • Ajay • Ajeencky a Kaabra • Ojaswini Mishra

All		USN-13	As a Developer, I want the system to handle errors gracefully and provide informative error messages to users.	1		• Piyush Pankaj • Ajay • Ajeencky a Kaabra • Ojaswini Mishra
All		USN-14	As a Developer, I want to ensure that user data is securely transmitted over the network, protecting user privacy.	1		• Piyush Pankaj • Ajay • Ajeencky a Kaabra • Ojaswini Mishra
All		USN-15	As a Developer, I want the application to be scalable, handling an increasing number of users and input data without compromising performance.	1		• Piyush Pankaj • Ajay • Ajeencky a Kaabra • Ojaswini Mishra

Project Tracker, Velocity & Burn down Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	11	6 Days	1 Nov 2023	6 Nov 2023	11	8 Nov 2023
Sprint-2	15	6 Days	6 Nov 2023	12 Nov 2023	15	15 Nov 2023

Velocity:

Sprint- 1: 1.833
Sprint- 2: 2.5

CODING & SOLUTIONING

7.1. Home Page Overview

- The system has a homepage that provides an overview of the smart home temperature prediction project.
- The homepage includes a navigation menu for easy access to different sections.
- The "About/Home" page contains detailed information about the project, including its objectives and features.
- It highlights the evolution of smart Wi-Fi thermostats and the goals of the current project.

Code-

```
<!DOCTYPE html>
<html>
  <head>
    <title>smart home temperature</title>
    <!-- <link rel="stylesheet" type="text/css" href="static/css/index.css" /> --
>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{ url_for('static', filename='css/index.css') }}"
    />
  </head>
  <body>
    <nav>
      <label>Smart Home Temperature</label>
      <ul>
        <li>
          <a href="/">About</a>
          <a href="predict">Predict</a>
          <a href="/contact">Contact Us</a>
        </li>
      </ul>
    </nav>
    <div class="container">
      <h3>Welcome to Smart Home Temperature</h3>
      <button type="button" onclick="redirectToNewPage()">
        Click Here to predict
      </button>
    </div>
```

```
    <h4>About</h4>
    <p>
      Smart WI-FI thermostats have moved well beyond the function they were
      originally designed for;<br />namely,controlling heating and cooling
      comfort in buildings.
    </p>

    <ul>
      <li>
        * They are now also learning form occupant behaviours and permit
        occupants to control their comfort remotely.
      </li>
      <br />
      <li>
        * This project seeks to go beyound this state of the art by utilizing
        smart Wi-Fi thermostat data in residences to develop dynamic predictive
        models for room temperature.
      </li>
    </ul>

    <div class="navbar">
      <h2>Develeoped by @team-592148</h2>
    </div>
    <script>
      function redirectToNewPage() {
        // Redirect to a new HTML page
        window.location.href = "predict";
      }
    </script>
  </body>
</html>
```

7.2. Contact Page

- The "Contact Us" page displays contact details, including email address and phone number.
- Users can reach out for inquiries or support.

Code-

```
<!DOCTYPE html>
<html>
  <head>
```

```
    <title>smart home temperature</title>
    <!-- <link rel="stylesheet" type="text/css" href="static/css/index.css" /> --
>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{ url_for('static', filename='css/index.css') }}"
    />
  </head>
  <body>
    <nav>
      <label>Smart Home Temperature</label>
      <ul>
        <li>
          <a href="/">About</a>
          <a href="/predict">Predict</a>
          <a href="/contact">Contact</a>
        </li>
      </ul>
    </nav>
    <div class="container">
      <h3>Welcome to Smart Home Temperature</h3>
    </div>

    <h3>Contact Us</h3>
    <h4>
      VIT Bhopal University,<br />
      Bhopal-Indore Highway Kothrikalan, Sehore<br />
      Madhya Pradesh - 466114.<br />
      Email add:- ajay.2021@vitbhopal.ac.in<br />
      Contact No.:- 7742519654
      <br />
    </h4>
  </body>
</html>
```

7.3. Prediction Page

- Users can access the prediction functionality by clicking on a designated button or link.
- The prediction page includes input fields for various environmental factors, such as CO2 levels, humidity, lighting, etc.

- The prediction page features a form with labeled input fields for different environmental parameters.
- Users can input numerical values for CO2 levels, humidity, lighting, and other factors influencing room temperature.

Code-

```
<!DOCTYPE html>
<html>
  <head>
    <title>smart home temperature</title>
    <!-- <link rel="stylesheet" type="text/css" href="static/css/predict.css" />
-->
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='css/predict.css') }}">
  </head>
  <body>
    <nav>
      <label>Smart Home Temperature</label>
      <ul>
        <li>
          <a href="/">Home</a>
          <a href="/predict">Predict</a>
          <a href="/contact">Contact Us</a>
        </li>
      </ul>
    </nav>
    <div class="container">
      <h3>Welcome to Smart Home Temperature</h3>
    </div>

    <center>
      <div class="login-box">
        <h2>INPUTS</h2>
        <form action="/output" method="POST">
          <div class="user-box">
            <label>CO2(dinning room)</label>
            <input
              type="number"
              placeholder="CO2 in dinning room"
              name="CO2_(dinning-room)"
              required=""
              step="0.001"
            />
```

```
          </div>

          <div class="user-box">
            <label>CO2 room</label>
            <input
              type="number"
              placeholder="CO2 in room"
              name="CO2_room"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Humidity(dinning room)</label>
            <input
              type="number"
              name="Relative_humidity_(dinning-room)"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Humidity room</label>
            <input type="number" name="Relative_humidity_room" required=""
step="0.001" />
          </div>

          <div class="user-box">
            <label>lighting(dinning room)</label>
            <input type="number" name="Lighting_(dinning-room)" required=""
step="0.001" />
          </div>

          <div class="user-box">
            <label>lighting room</label>
            <input type="number" name="Lighting_room" required="" step="0.001" />
          </div>

          <div class="user-box">
            <label>Rain</label>
            <input type="number" name="Meteo_Rain" required="" step="0.001" />
          </div>
```

```
          <div class="user-box">
            <label>Sun Dusk</label>
            <input type="number" name="Meteo_Sun_dusk" required="" step="0.001"
/>
          </div>

          <div class="user-box">
            <label>Wind</label>
            <input type="number" name="Meteo_Wind" required="" step="0.001" />
          </div>

          <div class="user-box">
            <label>Sunlight at west</label>
            <input
              type="number"
              name="Meteo_Sun_light_in_west_facade"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Sunlight at east</label>
            <input
              type="number"
              name="Meteo_Sun_light_in_east_facade"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Sunlight at south</label>
            <input
              type="number"
              name="Meteo_Sun_light_in_south_facade"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Sun Irridance</label>
```

```
            <input type="number" name="Meteo_Sun_irradiance" required=""
step="0.001" />
          </div>

          <div class="user-box">
            <label>Outdoor Humidity</label>
            <input
              type="number"
              name="Outdoor_relative_humidity_Sensor"
              required=""
              step="0.001"
            />
          </div>

          <a>
            <span></span>
            <span></span>
            <span></span>
            <span></span>
            round off upto 3 decimals like 3=3.000, 3.1425=3.143, 2.1423=3.142,
0=0.000
          </a>
          <a href="/output">
            <span></span>
            <span></span>
            <span></span>
            <span></span>
            <input type="submit">Submit</input>
          </a>

          <h4>{{result}}</h4>
        </form>

      </div>
    </center>
  </body>
</html>
```

7.4. Prediction Algorithm

- The system incorporates a predictive model (loaded from a pickled file) to generate room temperature predictions based on the provided input data.

Code-

In [1]:
```
!pip install xgboost
!pip install lightgbm
```

```
Requirement already satisfied: xgboost in d:\image processing softwares\lib\site-packages (2.0.2)
Requirement already satisfied: numpy in d:\image processing softwares\lib\site-packages (from xgboost) (1.23.5)
Requirement already satisfied: scipy in d:\image processing softwares\lib\site-packages (from xgboost) (1.10.0)
Requirement already satisfied: lightgbm in d:\image processing softwares\lib\site-packages (4.1.0)
Requirement already satisfied: numpy in d:\image processing softwares\lib\site-packages (from lightgbm) (1.23.5)
Requirement already satisfied: scipy in d:\image processing softwares\lib\site-packages (from lightgbm) (1.10.0)
```

Import Libraries

In [2]:
```
#importing the libraries
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
```

import Data Set

In [3]:
```
#load the dataset
df = pd.read_csv(r"D:\Git Hub\SI-GuidedProject-603567-1697618783\Project Development Phase\Regression\Dataset\train.csv")
```

In [4]:
```
#displaying the first five rows
df.head()
```

Out[4]:

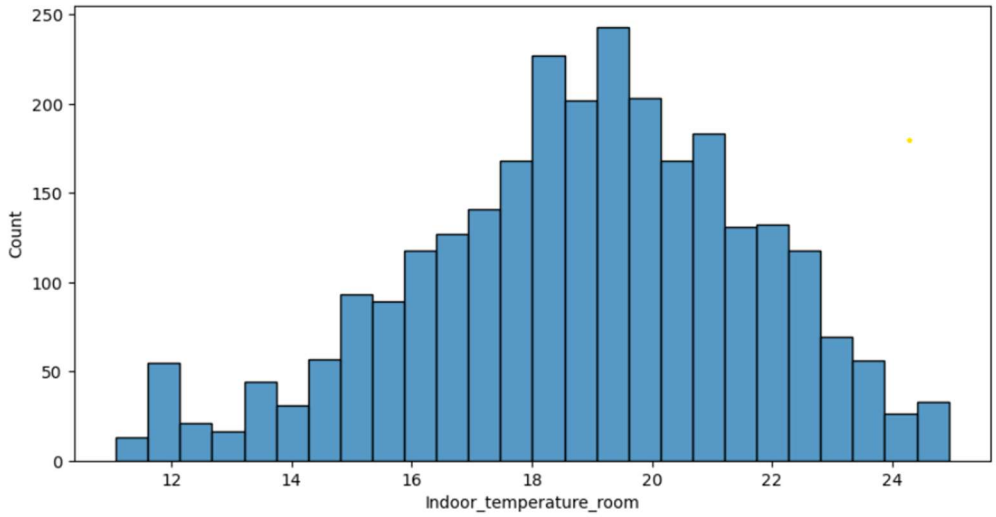
	Id	Date	Time	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_
0	0	13/03/2012	11:45	216.560	221.920	39.9125	42.4150	81.6650	113.520	0.0	
1	1	13/03/2012	12:00	219.947	220.363	39.9267	42.2453	81.7413	113.605	0.0	
2	2	13/03/2012	12:15	219.403	218.933	39.7720	42.2267	81.4240	113.600	0.0	
3	3	13/03/2012	12:30	218.613	217.045	39.7760	42.0987	81.5013	113.344	0.0	
4	4	13/03/2012	12:45	217.714	216.080	39.7757	42.0686	81.4657	113.034	0.0	

—

Data Visualisation

In [5]:
```
#univariate analysis
plt.figure(figsize=(10,5))
sns.histplot(data=df, x='Indoor_temperature_room',)
```

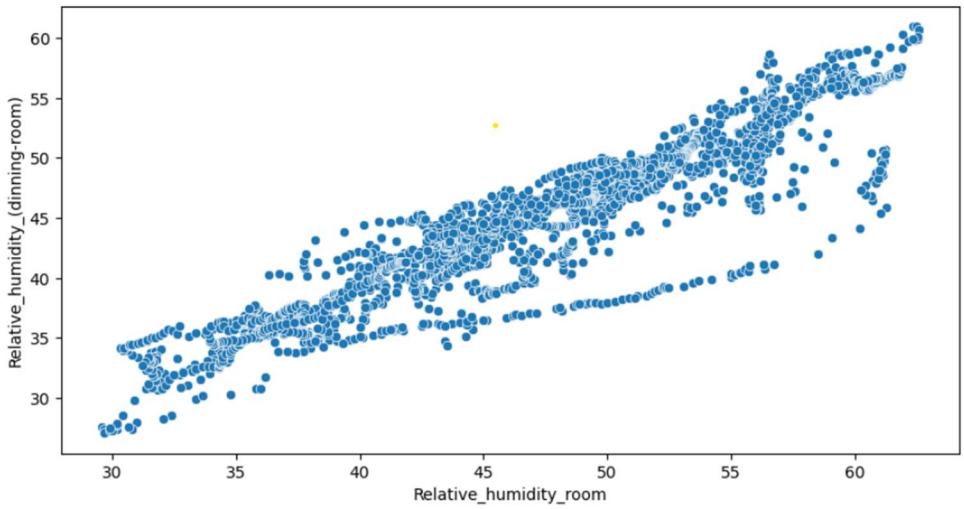
Out[5]: <Axes: xlabel='Indoor_temperature_room', ylabel='Count'>



Above graph shows that the Indoor_temperature_room feature shows normal distribution

In [6]:
```
#Bivariate analysis
plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x='Relative_humidity_room', y='Relative_humidity_(dinning-room)')
```

Out[6]: <Axes: xlabel='Relative_humidity_room', ylabel='Relative_humidity_(dinning-room)'>



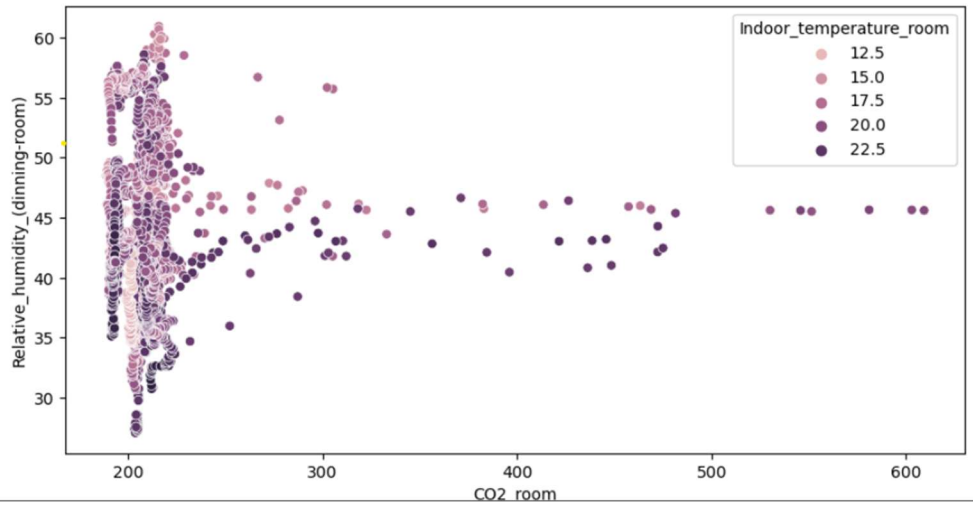
Independent variable = Relative_humidity_room

dependent variable = Relative_humidity_(dinning-room)

above graph represents good linear relation between above two varables

In [7]:
```
#multivariate analysis
plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x='CO2_room', y='Relative_humidity_(dinning-room)', hue='Indoor_temperature_room')
```

Out[7]: <Axes: xlabel='CO2_room', ylabel='Relative_humidity_(dinning-room)'>



According to above graph as CO2 increase in the room, indoor temperature also increases, room_temperature is low when co2 is arround 200 and humidity is between 30-43

In [8]:
```
#Descriptive Analysis
df.describe()
```

Out[8]:

	Id	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_Sur
count	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.
mean	1381.500000	208.479123	211.065844	44.878420	47.321220	26.745381	40.732571	0.047033	325.
std	798.042397	27.032686	28.469144	6.587440	7.557795	23.298441	42.326087	0.206705	305.
min	0.000000	187.339000	188.907000	27.084000	29.594700	10.740000	11.328000	0.000000	0.
25%	690.750000	200.893250	202.682750	40.351975	42.531325	11.588700	13.265300	0.000000	0.
50%	1381.500000	207.045500	209.408000	45.434650	47.534700	11.801300	17.690000	0.000000	611.
75%	2072.250000	211.245500	213.218750	49.352675	52.685975	31.224000	52.057350	0.000000	619.
max	2763.000000	594.389000	609.237000	60.957300	62.594700	110.693000	162.965000	1.000000	624.

above we get total_count as count, mean, standard deviation, quantriles bu using describe function

In [9]:
```
#correlation between columns
corr = df.corr()
corr
```

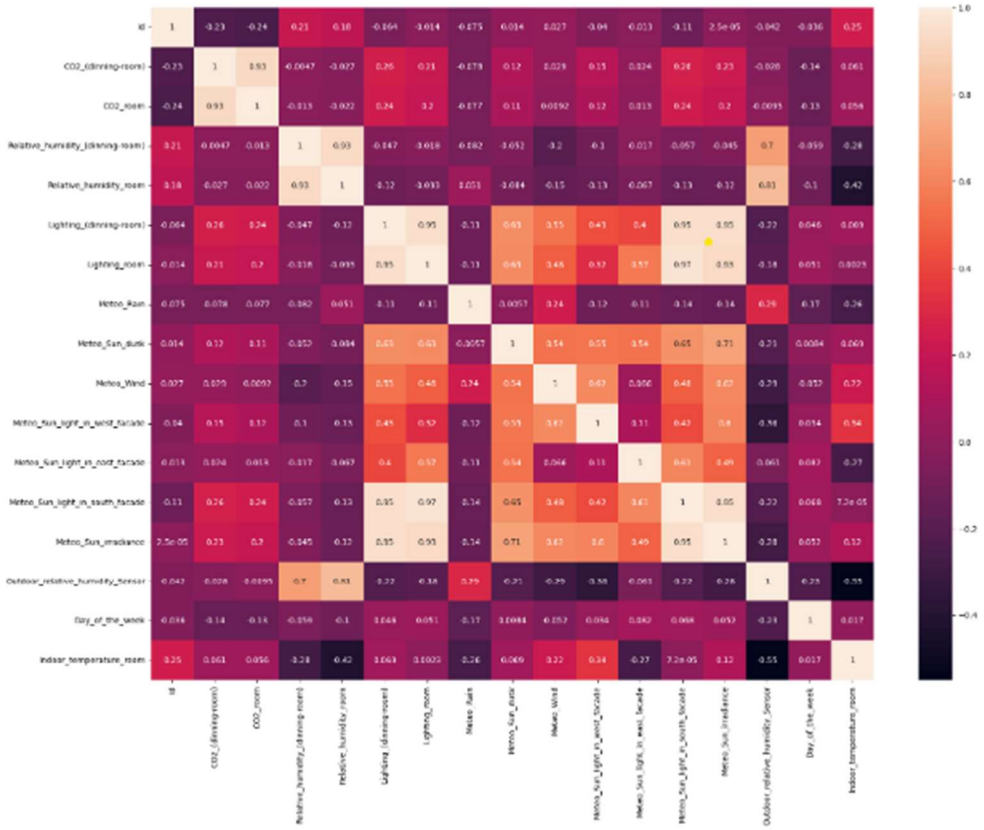
```
C:\Users\ajayp\AppData\Local\Temp\ipykernel_23920\2195921255.py:2: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  corr = df.corr()
```

Out[9]:

	Id	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room
Id	1.000000	-0.226771	-0.242380	0.207623	0.177251	-0.064395	-0.014088
CO2_(dinning-room)	-0.226771	1.000000	0.925522	-0.004698	-0.027170	0.255059	0.214517
CO2_room	-0.242380	0.925522	1.000000	-0.013242	-0.022244	0.241727	0.196714
Relative_humidity_(dinning-room)	0.207623	-0.004698	-0.013242	1.000000	0.931267	-0.046600	-0.017654
Relative_humidity_room	0.177251	-0.027170	-0.022244	0.931267	1.000000	-0.116434	-0.092876
Lighting_(dinning-room)	-0.064395	0.255059	0.241727	-0.046600	-0.116434	1.000000	0.948650
Lighting_room	-0.014088	0.214517	0.196714	-0.017654	-0.092876	0.948650	1.000000
Meteo_Rain	-0.075370	-0.078120	-0.077395	-0.081597	0.051322	-0.108220	-0.112234
Meteo_Sun_dusk	0.013842	0.121910	0.108002	-0.052456	-0.083595	0.631730	0.629562
Meteo_Wind	0.027212	0.029190	0.009194	-0.197537	-0.153276	0.545034	0.475906
Meteo_Sun_light_in_west_facade	-0.040328	0.146489	0.115046	-0.100505	-0.130920	0.426375	0.315060
Meteo_Sun_light_in_east_facade	-0.012653	0.023633	0.012565	-0.016503	-0.067384	0.404444	0.565679
Meteo_Sun_light_in_south_facade	-0.112416	0.256665	0.242475	-0.057019	-0.133493	0.949340	0.965505
Meteo_Sun_irradiance	0.000025	0.226493	0.204212	-0.045240	-0.120310	0.948145	0.932237
Outdoor_relative_humidity_Sensor	-0.042246	-0.028235	-0.009462	0.696973	0.809993	-0.220635	-0.182360
Day_of_the_week	-0.036023	-0.135281	-0.129085	-0.058968	-0.103301	0.045991	0.051008
Indoor_temperature_room	0.246559	0.061361	0.056195	-0.275193	-0.417901	0.069264	0.002253

In [10]:
```
#plot above correlation
plt.subplots(figsize=(20,15))
sns.heatmap(corr, annot=True)
```

Out[10]: <Axes: >



In above we can see that similar feature like lighting, humidity, CO2 for room and dinning room are highly correlative

and Meteo_Sun_light_in_south_facade and Meteo_Sun_irradian are also highly correlativwe with lighting for both room and dinning room

Data Preprocessing

In [11]:
```
#information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 19 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Id                                 2764 non-null   int64
 1   Date                               2764 non-null   object
 2   Time                               2764 non-null   object
 3   CO2_(dinning-room)                 2764 non-null   float64
 4   CO2_room                           2764 non-null   float64
 5   Relative_humidity_(dinning-room)   2764 non-null   float64
 6   Relative_humidity_room             2764 non-null   float64
 7   Lighting_(dinning-room)            2764 non-null   float64
 8   Lighting_room                      2764 non-null   float64
 9   Meteo_Rain                         2764 non-null   float64
 10  Meteo_Sun_dusk                     2764 non-null   float64
 11  Meteo_Wind                         2764 non-null   float64
 12  Meteo_Sun_light_in_west_facade     2764 non-null   float64
 13  Meteo_Sun_light_in_east_facade     2764 non-null   float64
 14  Meteo_Sun_light_in_south_facade    2764 non-null   float64
 15  Meteo_Sun_irradiance               2764 non-null   float64
 16  Outdoor_relative_humidity_Sensor   2764 non-null   float64
 17  Day_of_the_week                    2764 non-null   float64
 18  Indoor_temperature_room            2764 non-null   float64
dtypes: float64(16), int64(1), object(2)
memory usage: 410.4+ KB
```

In [12]:
```
#checking for null values
#get sum of null values
df.isnull().sum()
```

Out[12]:
```
Id                                 0
Date                               0
Time                               0
CO2_(dinning-room)                 0
CO2_room                           0
Relative_humidity_(dinning-room)   0
Relative_humidity_room             0
Lighting_(dinning-room)            0
Lighting_room                      0
Meteo_Rain                         0
Meteo_Sun_dusk                     0
Meteo_Wind                         0
Meteo_Sun_light_in_west_facade     0
Meteo_Sun_light_in_east_facade     0
Meteo_Sun_light_in_south_facade    0
Meteo_Sun_irradiance               0
Outdoor_relative_humidity_Sensor   0
Day_of_the_week                    0
Indoor_temperature_room            0
dtype: int64
```

above it is seen that no null values in our data and now there is no need to handle them

In [13]:
```
#Handling categorical Data
#again through indormation
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Id                                  2764 non-null   int64
 1   Date                                2764 non-null   object
 2   Time                                2764 non-null   object
 3   CO2_(dinning-room)                  2764 non-null   float64
 4   CO2_room                            2764 non-null   float64
 5   Relative_humidity_(dinning-room)    2764 non-null   float64
 6   Relative_humidity_room              2764 non-null   float64
 7   Lighting_(dinning-room)             2764 non-null   float64
 8   Lighting_room                       2764 non-null   float64
 9   Meteo_Rain                          2764 non-null   float64
 10  Meteo_Sun_dusk                      2764 non-null   float64
 11  Meteo_Wind                          2764 non-null   float64
 12  Meteo_Sun_light_in_west_facade      2764 non-null   float64
 13  Meteo_Sun_light_in_east_facade      2764 non-null   float64
 14  Meteo_Sun_light_in_south_facade     2764 non-null   float64
 15  Meteo_Sun_irradiance                2764 non-null   float64
 16  Outdoor_relative_humidity_Sensor    2764 non-null   float64
 17  Day_of_the_week                     2764 non-null   float64
 18  Indoor_temperature_room             2764 non-null   float64
dtypes: float64(16), int64(1), object(2)
memory usage: 410.4+ KB
```

from above we can see data and time is only object data type and that data we are not including in our training so we don't need to handle categorical data. Other data types are float.

In [14]:
```
#Seperate depedent and independent variables
x=df.iloc[:,3:17] #INDEPENDENT VARIABLE
y=df.iloc[:,18:] #DEPENDENT VARIABLE
```

In [15]:
```
x.head()
```

Out[15]:

	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_Sun_dusk	Meteo_Wind
0	216.560	221.920	39.9125	42.4150	81.6650	113.520	0.0	623.360	1.42625
1	219.947	220.363	39.9267	42.2453	81.7413	113.605	0.0	623.211	1.59200
2	219.403	218.933	39.7720	42.2267	81.4240	113.600	0.0	622.656	1.89133
3	218.613	217.045	39.7760	42.0987	81.5013	113.344	0.0	622.571	1.82800
4	217.714	216.080	39.7757	42.0686	81.4657	113.034	0.0	622.400	2.38071

In [16]:
```
y.head()
```

Out[16]:

	Indoor_temperature_room
0	17.8275
1	18.1207
2	18.4367
3	18.7513
4	19.0414

In [17]:
```
#splitting data into train and test
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)
```

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

Model Building

4 classification algorithms are used in model building

Linear Regression Model
Random Forest model
Light Gradient Boost Model
Xgboost model

Linear Regression Model

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

Out[19]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

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

Out[21]:
```
0.17425288988529142
```

Random Forest model

In [22]:
```
rf = RandomForestRegressor()
```

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

```
C:\Users\ajayp\AppData\Local\Temp\ipykernel_23920\1149647727.py:1: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rf.fit(x_train,y_train)
```

Out[23]:
```
RandomForestRegressor()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [24]: 1 pred1 = rf.predict(x_test)

In [25]: 1 pred1

Out[25]: array([21.618334, 21.27269 , 18.909898, 15.396623, 15.288172, 18.462552,
                18.204108, 16.202799, 18.322967, 18.155539, 22.021283, 16.58721 ,
                22.697447, 21.566378, 17.959353, 19.616625, 20.294848, 17.980272,
                21.621567, 16.256448, 19.670394, 16.147082, 16.783248, 20.391394,
                18.980112, 18.977458, 23.096354, 22.551678, 12.337911, 22.208924,
                17.938796, 20.168189, 20.216059, 21.008815, 18.018247, 15.705856,
                17.758723, 16.323511, 15.072934, 21.735057, 20.847159, 21.016636,
                22.55878 , 15.80513 , 16.095872, 16.313321, 17.598209, 19.787262,
                20.251053, 22.764732, 18.696833, 21.68169 , 19.4337  , 18.772987,
                22.171107, 22.857391, 17.831792, 17.5866  , 22.158708, 16.503597,
                22.838108, 19.484439, 19.497019, 19.784071, 22.503241, 17.07812 ,
                18.745744, 20.694292, 20.093232, 21.255181, 19.529829, 17.89106 ,
                17.402334, 22.201753, 22.010371, 16.270639, 20.117584, 22.671754,
                19.602795, 16.679414, 24.322784, 18.335331, 19.024312, 20.241291,
                18.410515, 19.442757, 17.542119, 22.302715, 19.638932, 17.476433,
                20.807519, 20.576808, 17.788778, 18.677864, 23.66118 , 16.202548,
                22.286528, 16.016571, 18.383636, 18.607819, 20.013217, 18.394084,
                19.919494, 19.502399, 17.802137, 18.299756, 16.971486, 17.110558,
                21.593663, 17.645954, 18.771928, 20.736208, 20.573099, 16.283229,

In [26]: 1 r2_score(y_test, pred1)

Out[26]: 0.9194343408777197
```

Light Gradient Boost Model

```
In [27]: 1 lg = lgb.LGBMRegressor()

In [28]: 1 lg.fit(x_train,y_train)

[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000268 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 3303
[LightGBM] [Info] Number of data points in the train set: 1934, number of used features: 14
[LightGBM] [Info] Start training from score 18.826209

Out[28]: LGBMRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [29]: 1 pred2 = lg.predict(x_test)

In [30]: 1 r2_score(y_test, pred2)

Out[30]: 0.9360756052934541
```

Xgboost Model

In [31]:
```
xg = xgb.XGBRegressor()
```

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

Out[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, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [33]:
```
pred3 = xg.predict(x_test)
```

In [34]:
```
r2_score(y_test,pred3)
```

Out[34]:
```
0.9312711740450739
```

In [35]:
```
xg1 = xgb.XGBRegressor()
xg1.fit(x_train_scaled,y_train)
```

Out[35]:
```
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, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Model	*r2_score*
lir	17.4%
rf	91.7%
lg	93.6%
xg	93.1%

Here above lg is the best model, hence we will export the model

In [36]:
```
import pickle
pickle.dump(lg,open('temperature.pkl','wb'))
```

More Evaluations

In [55]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [39]:
```
#for Light Gredient Model
mse = mean_squared_error(y_test, pred2)
mse
```

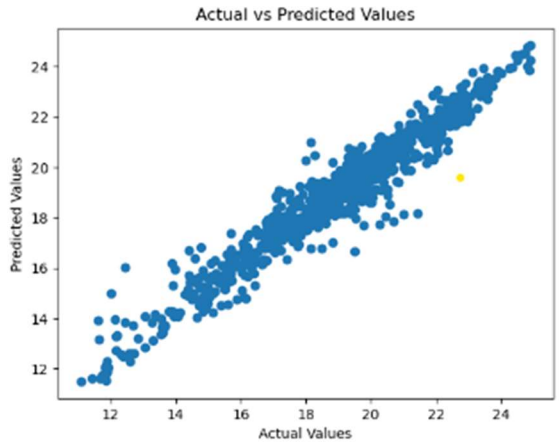
Out[39]: 0.5007536664270105

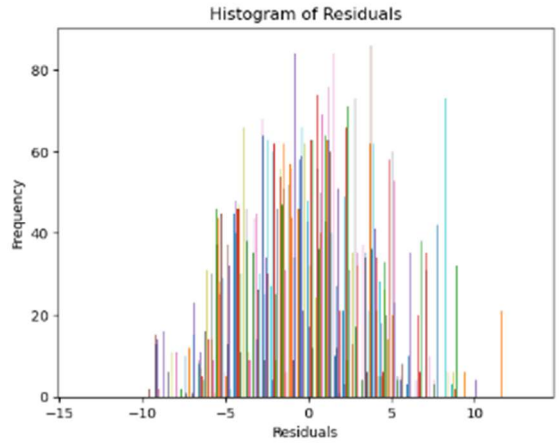
In [52]:
```
y_test_np = y_test.values
y_test_np
```

```
[18.068 ],
[22.3067],
[16.7953],
[23.588 ],
[21.7973],
[16.9173],
[19.6067],
[19.7733],
[17.856 ],
[21.188 ],
[15.684 ],
[18.836 ],
[16.2967],
[17.0253],
[20.8147],
[19.    ],
[19.0353],
[23.9307],
[22.6173],
```

In [53]:
```
#Scatter plot of actual vs predicted values
plt.scatter(y_test, pred2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()

# Residuals (difference between actual and predicted values)
residuals = y_test_np - pred2

# Histogram of residuals
plt.hist(residuals, bins=50)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Histogram of Residuals')
plt.show()
```

Actual vs Predicted Values





In [56]:
```
#for Light Gredient Model
mae = mean_absolute_error(y_test, pred2)
mae
```

Out[56]: 0.5089989459469727

```
In [71]:
#for Linear Regression Model
mae = mean_absolute_error(y_test, pred)
print(mae)
mse1 = mean_squared_error(y_test, pred)
print(mse)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred)

1.572677907866914
3.7427887580824297
1.934628842461114

Out[71]: 0.5222091780535016
```

```
In [73]:
#for Random Forest Model
mae = mean_absolute_error(y_test, pred1)
print(mae)
mse1 = mean_squared_error(y_test, pred1)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred1)

0.5211940722891568
0.6311135111844771
0.7944265297587166

Out[73]: 0.9194343408777197
```

```
In [76]:
#for Light Gradient Boost Model
mae = mean_absolute_error(y_test, pred2)
print(mae)
mse1 = mean_squared_error(y_test, pred2)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred2)

0.5089989459469727
0.5007536664270105
0.7076395031560989

Out[76]: 0.9360756052934541
```

```
In [77]:
#for XgBoost Model
mae = mean_absolute_error(y_test, pred3)
print(mae)
mse1 = mean_squared_error(y_test, pred3)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred3)

0.5326548784995941
0.5383893229516795
0.7337501774798283

Out[77]: 0.9312711740450739
```

7.5. Resulting Display

- After submitting the input data, the system displays the predicted room temperature prominently on the prediction page.
- The result is presented in a readable format for user understanding.

7.6. Responsive Design

- The user interface is designed to be responsive, providing a positive user experience on various devices and screen sizes.

7.7. Model Integration

- The prediction page should integrate with the machine learning model for temperature prediction.

Python Code on Spyder-

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 21 09:25:35 2023

@author: ajaypoonia
"""
#import the libraries
from flask import Flask, request, render_template
import pickle
import numpy as np
import pandas as pd

# load model
app = Flask(__name__, static_url_path='/static')

model = pickle.load(open(r'temperature.pkl', 'rb'))
app = Flask(__name__)

# Render HTML pages
@app.route("/")
def home():
    return render_template("index.html")

#contact page
@app.route("/contact")
def contact():
    return render_template("contact.html")

# Change the route to render predict.html
@app.route("/predict")
def predict():
    return render_template("predict.html")

# Retrieve value from UserInterface
@app.route('/output', methods=['post', 'get'])
def output():
    # reading the inputs given by the user
    input_feature = [float(x) for x in request.form.values()]
    input_feature = [np.array(input_feature)]
    print(input_feature)
    names = ['CO2_(dinning-room)', 'CO2_room', 'Relative_humidity_(dinning-
room)', 'Relative_humidity_room',
```

```
            'Lighting_(dinning-room)', 'Lighting_room', 'Meteo_Rain',
'Meteo_Sun_dusk', 'Meteo_Wind',
            'Meteo_Sun_light_in_west_facade', 'Meteo_Sun_light_in_east_facade',
'Meteo_Sun_light_in_south_facade',
            'Meteo_Sun_irradiance', 'Outdoor_relative_humidity_Sensor']
    print(names)
    data = pd.DataFrame(input_feature, columns=names)
    print(data)
    prediction = model.predict(data)
    print(prediction)
    return render_template('predict.html', result="Your room temperature will
be:  " + str(np.round(prediction[0])))

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

7.8. Navigation

- The navigation menu allows users to easily move between different pages, including the homepage, prediction page, and contact page.

7.9. User Interaction

- Users can interact with the system by inputting data, submitting the form, and receiving predictions.
- Visual feedback or confirmation is provided to indicate the success or failure of the prediction request.

7.10. Code Documentation

- The codebase includes comprehensive documentation, including a README file, inline comments, and potentially API documentation.
- The code follows best practices for maintainability.

7.11. Styling with CSS

- The system uses CSS stylesheets for styling, including separate stylesheets for the index page, prediction page, and potentially other pages.

Code Index CSS-

```
*{
```

```
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  box-sizing: border-box;
}

body{
  font-family: montserrat;
}

nav{
  background:black;
  height: 80px;
  width: 100%;
}

label{
  color: white;
  font-size: 35px;
  line-height: 80px;
  padding: 0 100px;
  font-weight: bold;
}

nav ul{
  float: right;
  margin-right: 20px;
}

nav ul li{
  display: inline-block;
  line-height: 80px;
  margin: 0 10px;
}

nav ul li a{
  color: white;
  font-size: 18px;
  padding: 7px 13px;
  border-radius: 3px;
  text-transform: uppercase;
}
```

```
h3{
  text-align: center;
  margin-top: 50px;
  font-size: 40px;
  font-weight: bold;


}

.container {
      border: 1px solid;
      background-color: lightblue;
      height: 200px;
      position: relative;
}

button{
      position: absolute;
      left: 50%;
      transform: translateX(-50%);
      background-color: lightgreen;
      margin-top: 30px;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
      font-size: 16px;
      border-radius: 10px;
}


h4{
font-weight: bold;
margin-left: 100px;
font-size: 30px;

margin-top:50px;
}

p{
  margin-top:20px;
  margin-left: 100px;
  color:gray;
  font-size: 20px;
```

```
}

ul{
  margin-top: 30px;
  margin-left: 120px;
  list-style-type: disc;
  font-size: 20px;

}

.navbar {
  background-color: #333;
  overflow: hidden;
  position: fixed;
  bottom: 0;
  width: 100%;
  height: 50px;
}

.navbar h2{
  color:white;
  margin-top: 10px;
  text-align: center;
}
```

Code Predict CSS-

```
*{
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  box-sizing: border-box;
}

body{
  font-family: montserrat;
}

nav{
  background:black;
  height: 80px;
  width: 100%;
}
```

```
label{
  color: white;
  font-size: 35px;
  line-height: 80px;
  padding: 0 100px;
  font-weight: bold;
}

nav ul{
  float: right;
  margin-right: 20px;
}

nav ul li{
  display: inline-block;
  line-height: 80px;
  margin: 0 10px;
}

nav ul li a{
  color: white;
  font-size: 18px;
  padding: 7px 13px;
  border-radius: 3px;
  text-transform: uppercase;
}

h3{
  text-align: center;
  margin-top: 50px;
  font-size: 40px;
  font-weight: bold;


}

.container {
      border: 1px solid;
      background-color: lightblue;
      height: 200px;
      position: relative;
}
```

```
html {
  height: 100%;
}
/*body {
  margin:0;
  padding:0;
  font-family: sans-serif;
  background: linear-gradient(#141e30, #243b55);
}*/

.login-box {
  position: absolute;
  top: 85%;
  left: 50%;
  width: 400px;
  padding: 40px;
  transform: translate(-50%, -50%);
  background: rgba(0,0,0,.5);
  box-sizing: border-box;
  box-shadow: 0 15px 25px rgba(0,0,0,.6);
  border-radius: 10px;
}

.login-box h2 {
  margin: 0 0 30px;
  padding: 0;
  color: #fff;
  text-align: center;
}

.login-box .user-box {
  position: relative;
}

.login-box .user-box input {
  width: 100%;
  padding: 10px 0;
  font-size: 16px;
  color: white;
  margin-bottom: 30px;
  border: none;
  border-bottom: 1px solid #fff;
  outline: none;
  background: transparent;
```

```
}
.login-box .user-box label {
  position: absolute;
  top:0;
  left: 0;
  padding: 10px 0;
  font-size: 16px;
  color: white;
  pointer-events: none;
  transition: .5s;
}

.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
  top: -20px;
  left: 0;
  color: white;
  font-size: 12px;
}

.login-box form a {
  position: relative;
  display: inline-block;
  padding: 10px 20px;
  color: #03e9f4;
  font-size: 16px;
  text-decoration: none;
  text-transform: uppercase;
  overflow: hidden;
  transition: .5s;
  margin-top: 40px;
  letter-spacing: 4px
}

.login-box a:hover {
  background: #03e9f4;
  color: #fff;
  border-radius: 5px;
  box-shadow: 0 0 5px #03e9f4,
              0 0 25px #03e9f4,
              0 0 50px #03e9f4,
              0 0 100px #03e9f4;
}
```

```
.login-box a span {
  position: absolute;
  display: block;
}

.login-box a span:nth-child(1) {
  top: 0;
  left: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(90deg, transparent, #03e9f4);
  animation: btn-anim1 1s linear infinite;
}

@keyframes btn-anim1 {
  0% {
    left: -100%;
  }
  50%,100% {
    left: 100%;
  }
}

.login-box a span:nth-child(2) {
  top: -100%;
  right: 0;
  width: 2px;
  height: 100%;
  background: linear-gradient(180deg, transparent, #03e9f4);
  animation: btn-anim2 1s linear infinite;
  animation-delay: .25s
}

@keyframes btn-anim2 {
  0% {
    top: -100%;
  }
  50%,100% {
    top: 100%;
  }
}

.login-box a span:nth-child(3) {
  bottom: 0;
```

```
  right: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(270deg, transparent, #03e9f4);
  animation: btn-anim3 1s linear infinite;
  animation-delay: .5s
}

@keyframes btn-anim3 {
  0% {
    right: -100%;
  }
  50%,100% {
    right: 100%;
  }
}

.login-box a span:nth-child(4) {
  bottom: -100%;
  left: 0;
  width: 2px;
  height: 100%;
  background: linear-gradient(360deg, transparent, #03e9f4);
  animation: btn-anim4 1s linear infinite;
  animation-delay: .75s
}

@keyframes btn-anim4 {
  0% {
    bottom: -100%;
  }
  50%,100% {
    bottom: 100%;
  }
}
```

Code contact CSS –

```
*{
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  box-sizing: border-box;
```

```
}

body{
  font-family: montserrat;
}

nav{
  background:black;
  height: 80px;
  width: 100%;
}

label{
  color: white;
  font-size: 35px;
  line-height: 80px;
  padding: 0 100px;
  font-weight: bold;
}

nav ul{
  float: right;
  margin-right: 20px;
}

nav ul li{
  display: inline-block;
  line-height: 80px;
  margin: 0 10px;
}

nav ul li a{
  color: white;
  font-size: 18px;
  padding: 7px 13px;
  border-radius: 3px;
  text-transform: uppercase;
}



.container {
      border: 1px solid;
```

```
      background-color: lightblue;
      height: 200px;
      position: relative;
}

h4{

text-align: center;
margin-left: 100px;
font-size: 30px;
margin-top:50px;
}
```

PERFORMANCE TESTING

8.1. Performance Evaluation

Regression Model:

For Light Gradient Boost Model (Best Model, selected for project)

- MAE– 0.5090
- MSE – 0.5008
- RMSE – 0.7076
- R2 score – 0.9361

```
: 1 #for Light Gradient Boost Model
  2 mae = mean_absolute_error(y_test, pred2)
  3 print(mae)
  4 mse1 = mean_squared_error(y_test, pred2)
  5 print(mse1)
  6 nrmse = np.sqrt(mse1)
  7 print(nrmse)
  8 r2_score(y_test,pred2)
```

```
0.5089989459469727
0.5007536664270105
0.7076395031560989

: 0.9360756052934541
```

For Linear Regression Model

- MAE– 1.5727
- MSE – 3.7428
- RMSE – 1.9346
- R2 score – 0.5222

```
1 #for Linear Regression Model
2 mae = mean_absolute_error(y_test, pred)
3 print(mae)
4 mse1 = mean_squared_error(y_test, pred)
5 print(mse)
6 nrmse = np.sqrt(mse1)
7 print(nrmse)
8 r2_score(y_test,pred)
```

```
1.572677907866914
3.7427887580824297
1.934628842461114

0.5222091780535016
```

For Random Forest Model

- MAE– 0.5212
- MSE – 0.6311
- RMSE – 0.7944
- R2 score – 0.9194

```
#for Random Forest Model
mae = mean_absolute_error(y_test, pred1)
print(mae)
mse1 = mean_squared_error(y_test, pred1)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred1)
```

```
0.5211940722891568
0.6311135111844771
0.7944265297587166

0.9194343408777197
```

For Xgboost Model

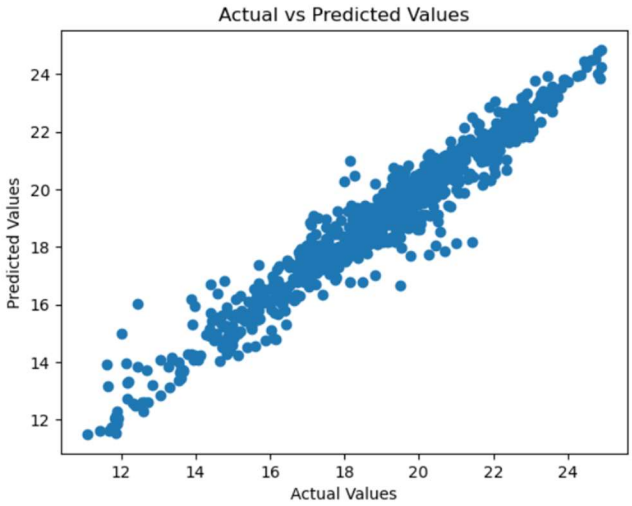
- MAE– 0.5327
- MSE – 0.5384
- RMSE – 0.7338
- R2 score – 0.9313

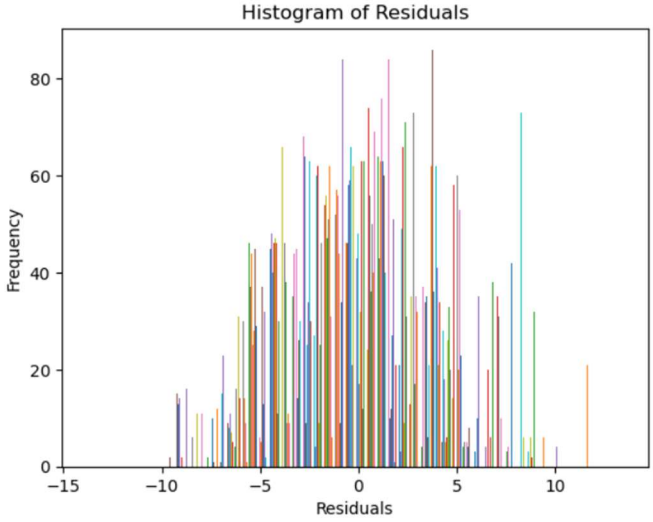
```
#for XgBoost Model
mae = mean_absolute_error(y_test, pred3)
print(mae)
mse1 = mean_squared_error(y_test, pred3)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred3)
```

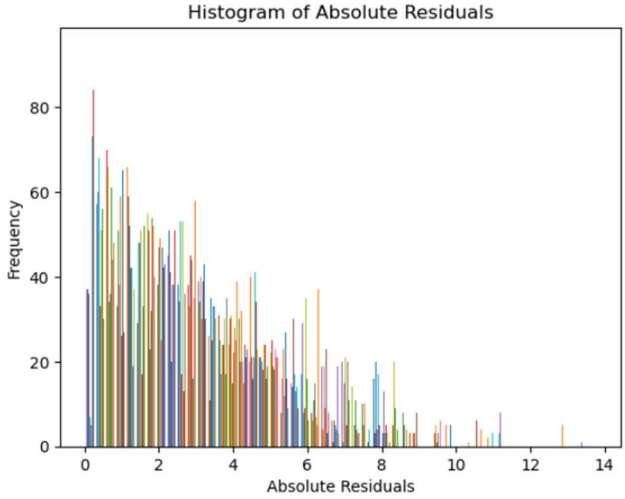
```
0.5326548784995941
0.5383893229516795
0.7337501774798283

0.9312711740450739
```

Here best performance model is Light Gradient Boost Model; hence we are selecting it.

Some more visualisation on this is as follow:


Actual vs Predicted Values
Predicted Values
Actual Values
12
14
16
18
20
22
24

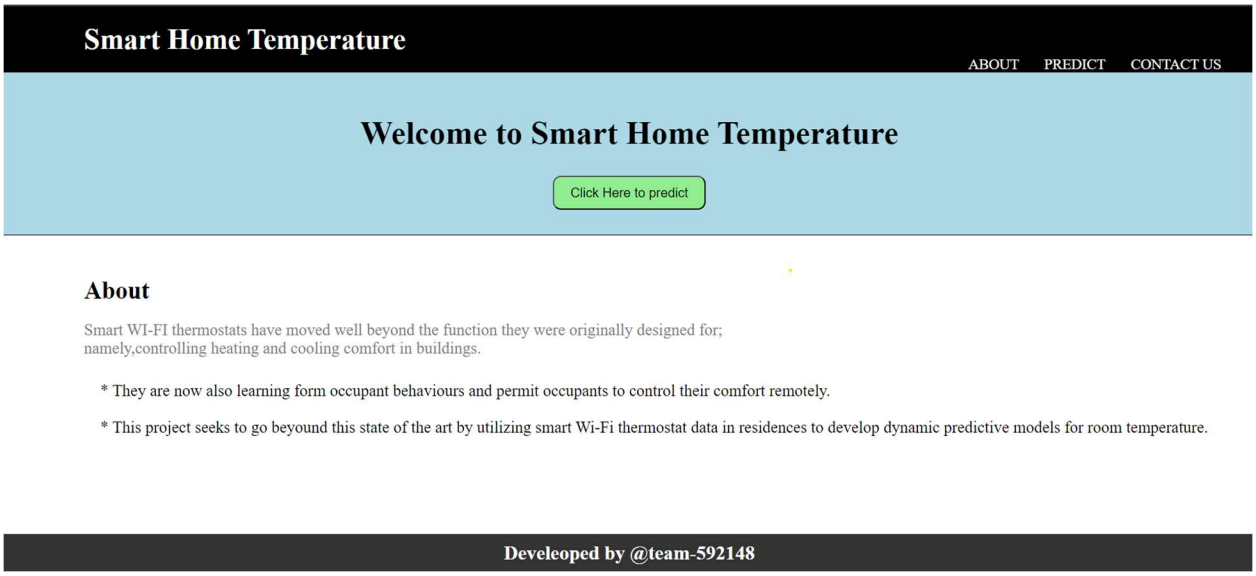

Histogram of Residuals
Frequency
Residuals
0
20
40
60
80
−15
−10
−5
0
5
10


Histogram of Absolute Residuals
Frequency
Absolute Residuals
0
20
40
60
80
0
2
4
6
8
10
12
14

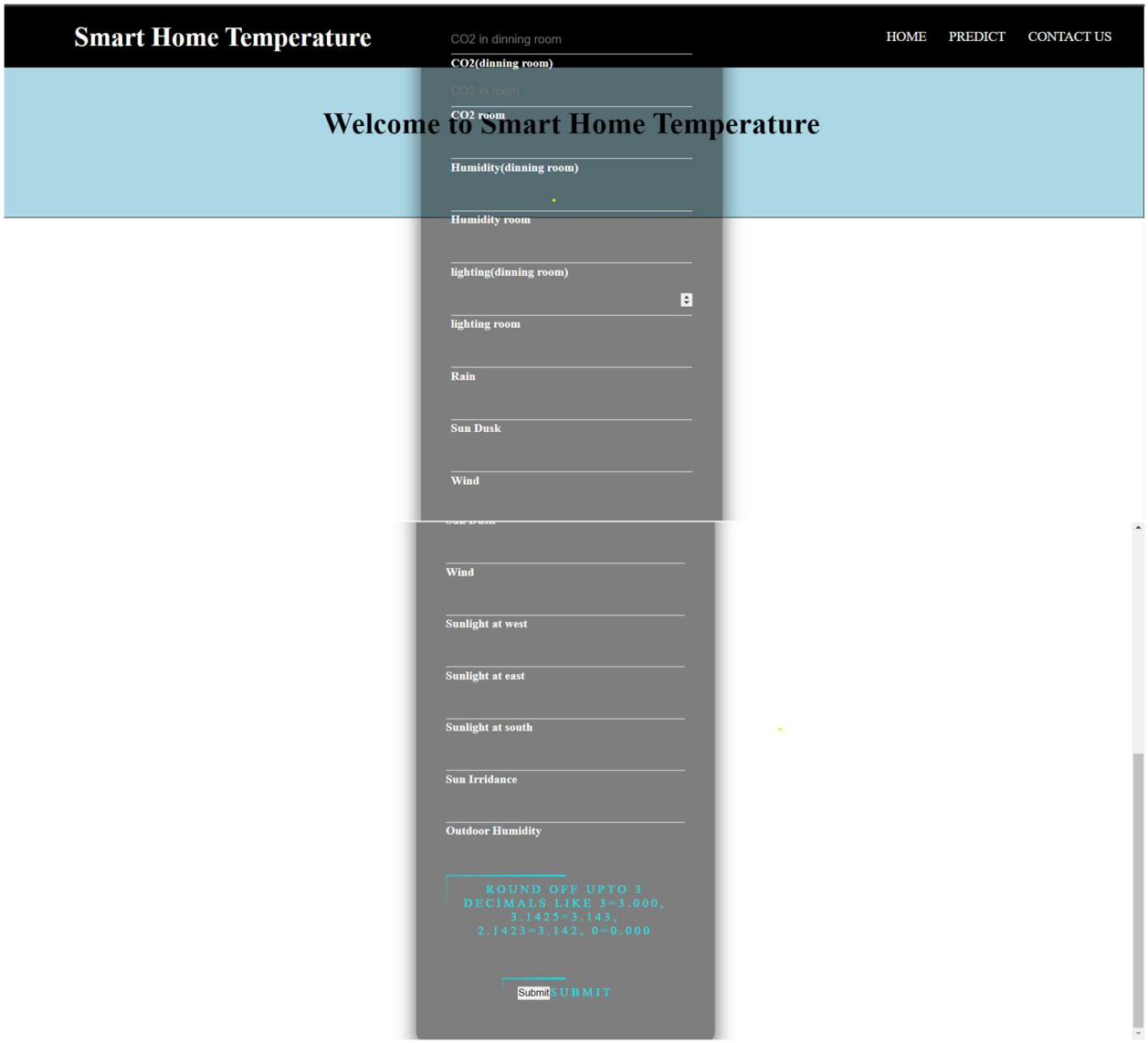
RESULTS

9.1. Output Screenshots

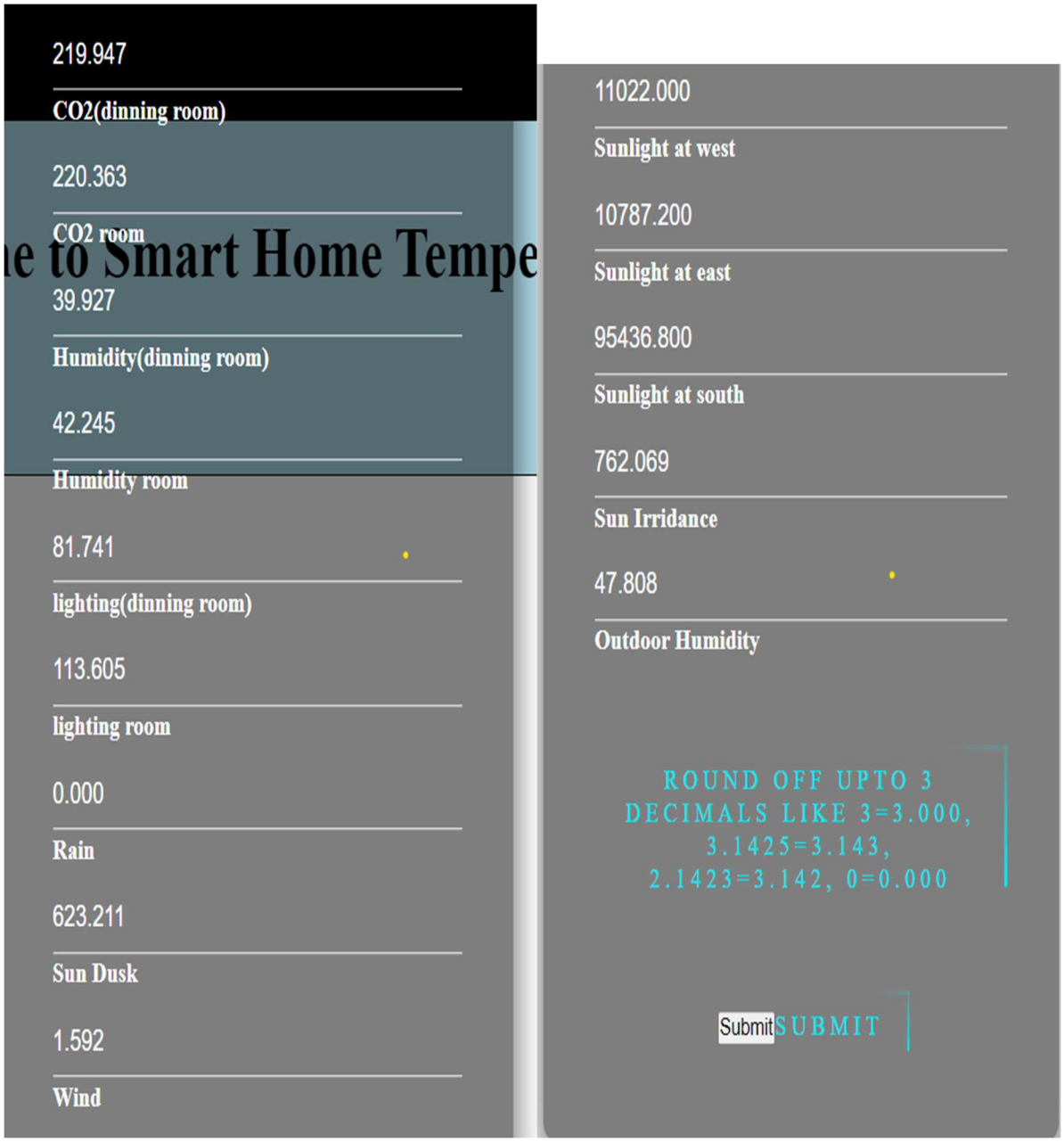
Home Page

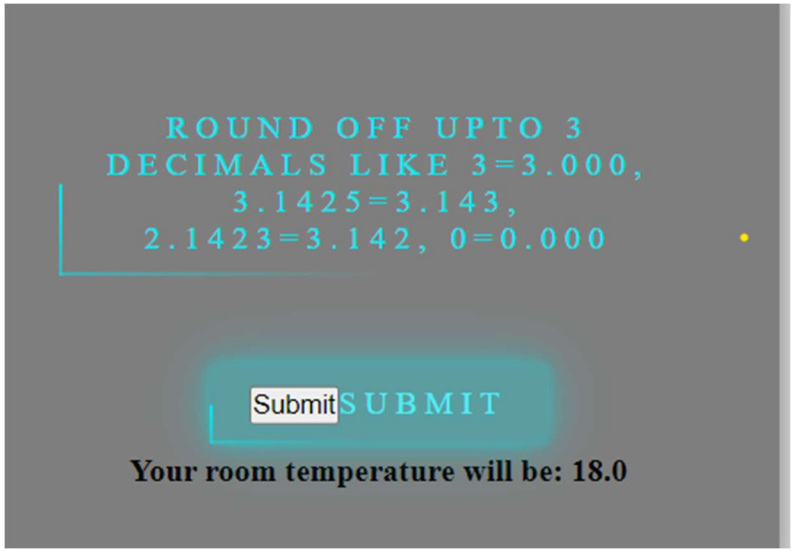


Predict Page



Predicted result


219.947
CO2(dinning room)
220.363
CO2 room
e to Smart Home Tempe
39.927
Humidity(dinning room)
42.245
Humidity room
81.741
lighting(dinning room)
113.605
lighting room
0.000
Rain
623.211
Sun Dusk
1.592
Wind
11022.000
Sunlight at west
10787.200
Sunlight at east
95436.800
Sunlight at south
762.069
Sun Irridance
47.808
Outdoor Humidity
ROUND OFF UPTO 3 DECIMALS LIKE 3=3.000, 3.1425=3.143, 2.1423=3.142, 0=0.000
SubmitSUBMIT



Contact Page

Contact Us

VIT Bhopal University,
Bhopal-Indore Highway Kothrikalan, Sehore
Madhya Pradesh - 466114.
Email add:- ajay.2021@vitbhopal.ac.in
Contact No.:- 7742519654

ADVANTAGES AND DISADVANTAGES

10.1. Advantages

- **Energy Efficiency:** By predicting room temperature based on various environmental factors, the system can optimize heating and cooling systems for energy efficiency, reducing energy consumption.

- **Remote Control:** Users can remotely control and monitor their home temperature, providing convenience and flexibility.

- **User-Friendly Interface:** The system offers a user-friendly interface with easy navigation and clear input forms, enhancing user experience.

- **Predictive Analytics:** Utilizing predictive models allows for proactive temperature management, adapting to changing conditions and user preferences.

- **Smart Home Integration:** The project aligns with the trend of smart home technologies, contributing to the broader ecosystem of connected devices.

- **Educational Value:** The project's "About" page educates users on the evolution of smart Wi-Fi thermostats and the project's goals, fostering understanding and awareness.

- **Contact and Support:** The "Contact Us" page provides a channel for users to seek assistance or clarification, enhancing user support.

- **Responsive Design:** A responsive design ensures a consistent and optimal user experience across various devices, including smartphones and tablets.

10.2. Disadvantages

- **Limited Environmental Factors:** The system may not consider all relevant environmental factors influencing room temperature, potentially leading to less accurate predictions.

- **Dependence on Model Accuracy:** The accuracy of temperature predictions heavily relies on the quality of the predictive model. If the model is not well-trained or lacks data, predictions may be less reliable.

- **Data Privacy Concerns:** Collecting and using data related to home environments may raise privacy concerns. Users might be hesitant to share detailed information about their living spaces.

- **Complexity for Users:** The input form requires users to input multiple environmental parameters, which might be overwhelming for some users or lead to errors if not properly guided.

- **Limited Interactivity:** The system's interaction is currently limited to predicting room temperature. Adding features like historical data analysis or user feedback could enhance interactivity.

- **Potential Security Risks:** Depending on how user data is handled, there might be security risks associated with the system, especially if it involves remote control of home devices.

- **Dependency on Flask:** The system is built using Flask, which might limit scalability for larger applications. Consideration of alternative frameworks could be beneficial for future expansion.

- **Static Content Loading:** The system currently loads static content (CSS) from files. Using Content Delivery Networks (CDNs) or optimizing static content loading could improve performance.

Conclusion

In conclusion, the smart home temperature prediction project successfully combines machine learning with a user-friendly web interface to enhance energy efficiency and user control in home temperature management. The integration of Flask provides a robust foundation, and the availability of the project on GitHub facilitates collaboration and further development. While the system showcases strengths in energy optimization and remote control, considerations for model accuracy, user complexity, and privacy concerns should be addressed for continued improvement. Overall, the project lays a foundation for smart home applications and highlights the potential for future enhancements in predictive analytics and user engagement.

FUTURE SCOPE

The future scope for the above project encompasses several key aspects aimed at further enhancing its functionality and user experience. Firstly, there is a potential for continued refinement and improvement of the machine learning models, with a focus on increasing accuracy and accommodating a wider array of environmental factors.

Additionally, the project can benefit from the implementation of a user feedback mechanism, allowing users to provide input on predicted temperatures and facilitating iterative development. Integration with popular smart home ecosystems is another avenue for future development, enabling seamless interoperability with various devices and platforms.

Analysing historical temperature data trends could provide valuable insights for users, and the development of dedicated mobile applications for iOS and Android platforms would enhance accessibility. Regular updates to machine learning models to reflect evolving user behavior, environmental conditions, and advancements in predictive modelling techniques are crucial for maintaining effectiveness.

Energy consumption analytics could be integrated to help users make informed decisions about energy-efficient temperature control. Establishing a community around the project can foster collaboration, feature contributions, and the development of plugins or extensions, expanding its potential impact.

Enhancements in security and privacy controls are essential to address concerns related to user data and smart home device interactions. Integration of additional IoT sensors and devices can provide more diverse and granular data for improved temperature predictions.

The implementation of user profiles would allow for personalized temperature preferences, creating a tailored and adaptive smart home experience. Incorporating explainability features into the machine learning model can provide users with insights into the decision-making process, enhancing transparency and trust.

Global weather integration through APIs could further optimize predictions by considering real-time weather updates. Scalability enhancements will be crucial to support larger deployments,

ensuring the system's effectiveness as user numbers increase. Overall, the future scope for the project is expansive, with a focus on continual improvement, adaptation to emerging technologies, and a commitment to enhancing the overall smart home experience.

APPENDIX

13.1. Source Code

HTML

Home

```
<!DOCTYPE html>
<html>
  <head>
    <title>smart home temperature</title>
    <!-- <link rel="stylesheet" type="text/css" href="static/css/index.css" /> --
>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{ url_for('static', filename='css/index.css') }}"
    />
  </head>
  <body>
    <nav>
      <label>Smart Home Temperature</label>
      <ul>
        <li>
          <a href="/">About</a>
          <a href="predict">Predict</a>
          <a href="/contact">Contact Us</a>
        </li>
      </ul>
    </nav>
    <div class="container">
      <h3>Welcome to Smart Home Temperature</h3>
      <button type="button" onclick="redirectToNewPage()">
        Click Here to predict
      </button>
    </div>

    <h4>About</h4>
    <p>
      Smart WI-FI thermostats have moved well beyond the function they were
      originally designed for;<br />namely,controlling heating and cooling
      comfort in buildings.
```

```
    </p>

    <ul>
      <li>
        * They are now also learning form occupant behaviours and permit
        occupants to control their comfort remotely.
      </li>
      <br />
      <li>
        * This project seeks to go beyound this state of the art by utilizing
        smart Wi-Fi thermostat data in residences to develop dynamic predictive
        models for room temperature.
      </li>
    </ul>

    <div class="navbar">
      <h2>Develeoped by @team-592148</h2>
    </div>
    <script>
      function redirectToNewPage() {
        // Redirect to a new HTML page
        window.location.href = "predict";
      }
    </script>
  </body>
</html>
```

Predict page

```
<!DOCTYPE html>
<html>
  <head>
    <title>smart home temperature</title>
    <!-- <link rel="stylesheet" type="text/css" href="static/css/predict.css" />
-->
    <link rel="stylesheet" type="text/css" href="{{ url_for('static',
filename='css/predict.css') }}">
  </head>
  <body>
    <nav>
      <label>Smart Home Temperature</label>
      <ul>
        <li>
          <a href="/">Home</a>
```

```
        <a href="/predict">Predict</a>
        <a href="/contact">Contact Us</a>
      </li>
    </ul>
  </nav>
  <div class="container">
    <h3>Welcome to Smart Home Temperature</h3>
  </div>

  <center>
    <div class="login-box">
      <h2>INPUTS</h2>
      <form action="/output" method="POST">
        <div class="user-box">
          <label>CO2(dinning room)</label>
          <input
            type="number"
            placeholder="CO2 in dinning room"
            name="CO2_(dinning-room)"
            required=""
            step="0.001"
          />
        </div>

        <div class="user-box">
          <label>CO2 room</label>
          <input
            type="number"
            placeholder="CO2 in room"
            name="CO2_room"
            required=""
            step="0.001"
          />
        </div>

        <div class="user-box">
          <label>Humidity(dinning room)</label>
          <input
            type="number"
            name="Relative_humidity_(dinning-room)"
            required=""
            step="0.001"
          />
        </div>
```

```
          <div class="user-box">
            <label>Humidity room</label>
            <input type="number" name="Relative_humidity_room" required=""
step="0.001" />
          </div>

          <div class="user-box">
            <label>lighting(dinning room)</label>
            <input type="number" name="Lighting_(dinning-room)" required=""
step="0.001" />
          </div>

          <div class="user-box">
            <label>lighting room</label>
            <input type="number" name="Lighting_room" required="" step="0.001" />
          </div>

          <div class="user-box">
            <label>Rain</label>
            <input type="number" name="Meteo_Rain" required="" step="0.001" />
          </div>

          <div class="user-box">
            <label>Sun Dusk</label>
            <input type="number" name="Meteo_Sun_dusk" required="" step="0.001"
/>
          </div>

          <div class="user-box">
            <label>Wind</label>
            <input type="number" name="Meteo_Wind" required="" step="0.001" />
          </div>

          <div class="user-box">
            <label>Sunlight at west</label>
            <input
              type="number"
              name="Meteo_Sun_light_in_west_facade"
              required=""
              step="0.001"
            />
          </div>
```

```
          <div class="user-box">
            <label>Sunlight at east</label>
            <input
              type="number"
              name="Meteo_Sun_light_in_east_facade"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Sunlight at south</label>
            <input
              type="number"
              name="Meteo_Sun_light_in_south_facade"
              required=""
              step="0.001"
            />
          </div>

          <div class="user-box">
            <label>Sun Irridance</label>
            <input type="number" name="Meteo_Sun_irradiance" required=""
step="0.001" />
          </div>

          <div class="user-box">
            <label>Outdoor Humidity</label>
            <input
              type="number"
              name="Outdoor_relative_humidity_Sensor"
              required=""
              step="0.001"
            />
          </div>

          <a>
            <span></span>
            <span></span>
            <span></span>
            <span></span>
            round off upto 3 decimals like 3=3.000, 3.1425=3.143, 2.1423=3.142,
0=0.000
          </a>
```

```
          <a href="/output">
            <span></span>
            <span></span>
            <span></span>
            <span></span>
            <input type="submit">Submit</input>
          </a>

          <h4>{{result}}</h4>
        </form>

      </div>
    </center>
  </body>
</html>
```

Contact Page

```
<!DOCTYPE html>
<html>
  <head>
    <title>smart home temperature</title>
    <!-- <link rel="stylesheet" type="text/css" href="static/css/index.css" /> --
>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{ url_for('static', filename='css/index.css') }}"
    />
  </head>
  <body>
    <nav>
      <label>Smart Home Temperature</label>
      <ul>
        <li>
          <a href="/">About</a>
          <a href="/predict">Predict</a>
          <a href="/contact">Contact</a>
        </li>
      </ul>
    </nav>
    <div class="container">
      <h3>Welcome to Smart Home Temperature</h3>
    </div>
```

```
    <h3>Contact Us</h3>
    <h4>
      VIT Bhopal University,<br />
      Bhopal-Indore Highway Kothrikalan, Sehore<br />
      Madhya Pradesh - 466114.<br />
      Email add:- ajay.2021@vitbhopal.ac.in<br />
      Contact No.:- 7742519654
      <br />
    </h4>
  </body>
</html>
```

CSS

Home page

```
*{
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  box-sizing: border-box;
}

body{
  font-family: montserrat;
}

nav{
  background:black;
  height: 80px;
  width: 100%;
}

label{
  color: white;
  font-size: 35px;
  line-height: 80px;
  padding: 0 100px;
  font-weight: bold;
}
```

```
nav ul{
  float: right;
  margin-right: 20px;
}

nav ul li{
  display: inline-block;
  line-height: 80px;
  margin: 0 10px;
}

nav ul li a{
  color: white;
  font-size: 18px;
  padding: 7px 13px;
  border-radius: 3px;
  text-transform: uppercase;
}

h3{
  text-align: center;
  margin-top: 50px;
  font-size: 40px;
  font-weight: bold;


}

.container {
      border: 1px solid;
      background-color: lightblue;
      height: 200px;
      position: relative;
}

button{
      position: absolute;
      left: 50%;
      transform: translateX(-50%);
      background-color: lightgreen;
      margin-top: 30px;
      padding: 10px 20px;
      text-align: center;
      text-decoration: none;
```

```
      font-size: 16px;
      border-radius: 10px;
}


h4{
font-weight: bold;
margin-left: 100px;
font-size: 30px;

margin-top:50px;
}

p{
  margin-top:20px;
  margin-left: 100px;
  color:gray;
  font-size: 20px;

}

ul{
  margin-top: 30px;
  margin-left: 120px;
  list-style-type: disc;
  font-size: 20px;

}

.navbar {
  background-color: #333;
  overflow: hidden;
  position: fixed;
  bottom: 0;
  width: 100%;
  height: 50px;
}

.navbar h2{
  color:white;
  margin-top: 10px;
  text-align: center;
}
```

Predict Page

```css
*{
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  box-sizing: border-box;
}

body{
  font-family: montserrat;
}

nav{
  background:black;
  height: 80px;
  width: 100%;
}

label{
  color: white;
  font-size: 35px;
  line-height: 80px;
  padding: 0 100px;
  font-weight: bold;
}

nav ul{
  float: right;
  margin-right: 20px;
}

nav ul li{
  display: inline-block;
  line-height: 80px;
  margin: 0 10px;
}

nav ul li a{
  color: white;
  font-size: 18px;
  padding: 7px 13px;
  border-radius: 3px;
  text-transform: uppercase;
}
```

```
h3{
  text-align: center;
  margin-top: 50px;
  font-size: 40px;
  font-weight: bold;


}

.container {
      border: 1px solid;
      background-color: lightblue;
      height: 200px;
      position: relative;
}


html {
  height: 100%;
}
/*body {
  margin:0;
  padding:0;
  font-family: sans-serif;
  background: linear-gradient(#141e30, #243b55);
}*/

.login-box {
  position: absolute;
  top: 85%;
  left: 50%;
  width: 400px;
  padding: 40px;
  transform: translate(-50%, -50%);
  background: rgba(0,0,0,.5);
  box-sizing: border-box;
  box-shadow: 0 15px 25px rgba(0,0,0,.6);
  border-radius: 10px;
}

.login-box h2 {
  margin: 0 0 30px;
  padding: 0;
```

```
  color: #fff;
  text-align: center;
}

.login-box .user-box {
  position: relative;
}

.login-box .user-box input {
  width: 100%;
  padding: 10px 0;
  font-size: 16px;
  color: white;
  margin-bottom: 30px;
  border: none;
  border-bottom: 1px solid #fff;
  outline: none;
  background: transparent;
}
.login-box .user-box label {
  position: absolute;
  top:0;
  left: 0;
  padding: 10px 0;
  font-size: 16px;
  color: white;
  pointer-events: none;
  transition: .5s;
}

.login-box .user-box input:focus ~ label,
.login-box .user-box input:valid ~ label {
  top: -20px;
  left: 0;
  color: white;
  font-size: 12px;
}

.login-box form a {
  position: relative;
  display: inline-block;
  padding: 10px 20px;
  color: #03e9f4;
  font-size: 16px;
```

```
  text-decoration: none;
  text-transform: uppercase;
  overflow: hidden;
  transition: .5s;
  margin-top: 40px;
  letter-spacing: 4px
}

.login-box a:hover {
  background: #03e9f4;
  color: #fff;
  border-radius: 5px;
  box-shadow: 0 0 5px #03e9f4,
              0 0 25px #03e9f4,
              0 0 50px #03e9f4,
              0 0 100px #03e9f4;
}

.login-box a span {
  position: absolute;
  display: block;
}

.login-box a span:nth-child(1) {
  top: 0;
  left: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(90deg, transparent, #03e9f4);
  animation: btn-anim1 1s linear infinite;
}

@keyframes btn-anim1 {
  0% {
    left: -100%;
  }
  50%,100% {
    left: 100%;
  }
}

.login-box a span:nth-child(2) {
  top: -100%;
  right: 0;
```

```
  width: 2px;
  height: 100%;
  background: linear-gradient(180deg, transparent, #03e9f4);
  animation: btn-anim2 1s linear infinite;
  animation-delay: .25s
}

@keyframes btn-anim2 {
  0% {
    top: -100%;
  }
  50%,100% {
    top: 100%;
  }
}

.login-box a span:nth-child(3) {
  bottom: 0;
  right: -100%;
  width: 100%;
  height: 2px;
  background: linear-gradient(270deg, transparent, #03e9f4);
  animation: btn-anim3 1s linear infinite;
  animation-delay: .5s
}

@keyframes btn-anim3 {
  0% {
    right: -100%;
  }
  50%,100% {
    right: 100%;
  }
}

.login-box a span:nth-child(4) {
  bottom: -100%;
  left: 0;
  width: 2px;
  height: 100%;
  background: linear-gradient(360deg, transparent, #03e9f4);
  animation: btn-anim4 1s linear infinite;
  animation-delay: .75s
}
```

```
@keyframes btn-anim4 {
  0% {
    bottom: -100%;
  }
  50%,100% {
    bottom: 100%;
  }
}
```

Contact Page

```
*{
  padding: 0;
  margin: 0;
  text-decoration: none;
  list-style: none;
  box-sizing: border-box;
}

body{
  font-family: montserrat;
}

nav{
  background:black;
  height: 80px;
  width: 100%;
}

label{
  color: white;
  font-size: 35px;
  line-height: 80px;
  padding: 0 100px;
  font-weight: bold;
}

nav ul{
  float: right;
  margin-right: 20px;
}

nav ul li{
  display: inline-block;
```

```
  line-height: 80px;
  margin: 0 10px;
}

nav ul li a{
  color: white;
  font-size: 18px;
  padding: 7px 13px;
  border-radius: 3px;
  text-transform: uppercase;
}




.container {
      border: 1px solid;
      background-color: lightblue;
      height: 200px;
      position: relative;
}

h4{

text-align: center;
margin-left: 100px;
font-size: 30px;
margin-top:50px;
}
```

Machine Learning Model Building Code

In [1]:
```
!pip install xgboost
!pip install lightgbm
```

```
Requirement already satisfied: xgboost in d:\image processing softwares\lib\site-packages (2.0.2)
Requirement already satisfied: numpy in d:\image processing softwares\lib\site-packages (from xgboost) (1.23.5)
Requirement already satisfied: scipy in d:\image processing softwares\lib\site-packages (from xgboost) (1.10.0)
Requirement already satisfied: lightgbm in d:\image processing softwares\lib\site-packages (4.1.0)
Requirement already satisfied: numpy in d:\image processing softwares\lib\site-packages (from lightgbm) (1.23.5)
Requirement already satisfied: scipy in d:\image processing softwares\lib\site-packages (from lightgbm) (1.10.0)
```

Import Libraries

In [2]:
```
#importing the libraries
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
```

import Data Set

In [3]:
```
#Load the dataset
df = pd.read_csv(r"D:\Git Hub\SI-GuidedProject-603567-1697618783\Project Development Phase\Regression\Dataset\train.csv")
```

In [4]:
```
#displaying the first five rows
df.head()
```

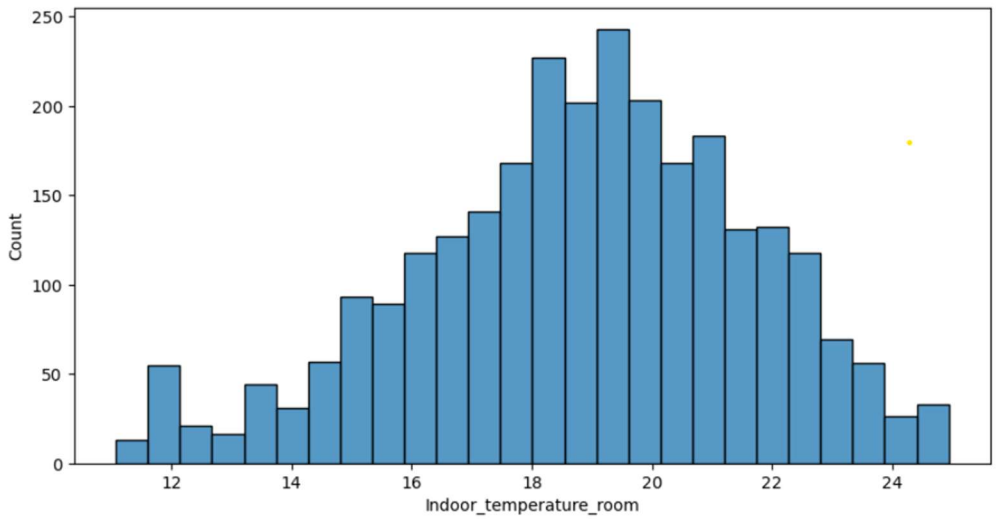
Out[4]:

	Id	Date	Time	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_
0	0	13/03/2012	11:45	216.560	221.920	39.9125	42.4150	81.6650	113.520	0.0	
1	1	13/03/2012	12:00	219.947	220.363	39.9267	42.2453	81.7413	113.605	0.0	
2	2	13/03/2012	12:15	219.403	218.933	39.7720	42.2267	81.4240	113.600	0.0	
3	3	13/03/2012	12:30	218.613	217.045	39.7760	42.0987	81.5013	113.344	0.0	
4	4	13/03/2012	12:45	217.714	216.080	39.7757	42.0686	81.4657	113.034	0.0	

Data Visualisation

In [5]:
```
#univariate analysis
plt.figure(figsize=(10,5))
sns.histplot(data=df, x='Indoor_temperature_room',)
```

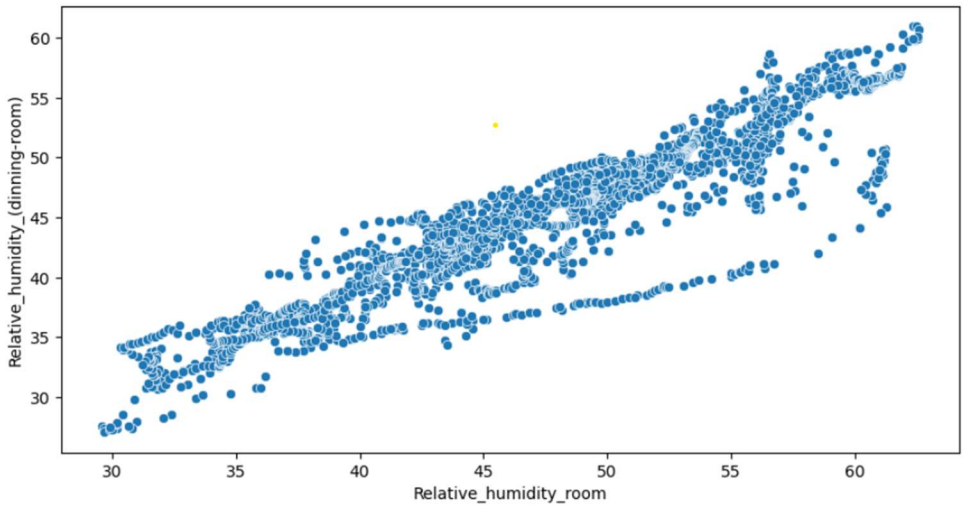
Out[5]:
```
<Axes: xlabel='Indoor_temperature_room', ylabel='Count'>
```



Above graph shows that the Indoor_temperature_room feature shows normal distribution

In [6]:
```
#Bivariate analysis
plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x='Relative_humidity_room', y='Relative_humidity_(dinning-room)')
```

Out[6]: <Axes: xlabel='Relative_humidity_room', ylabel='Relative_humidity_(dinning-room)'>



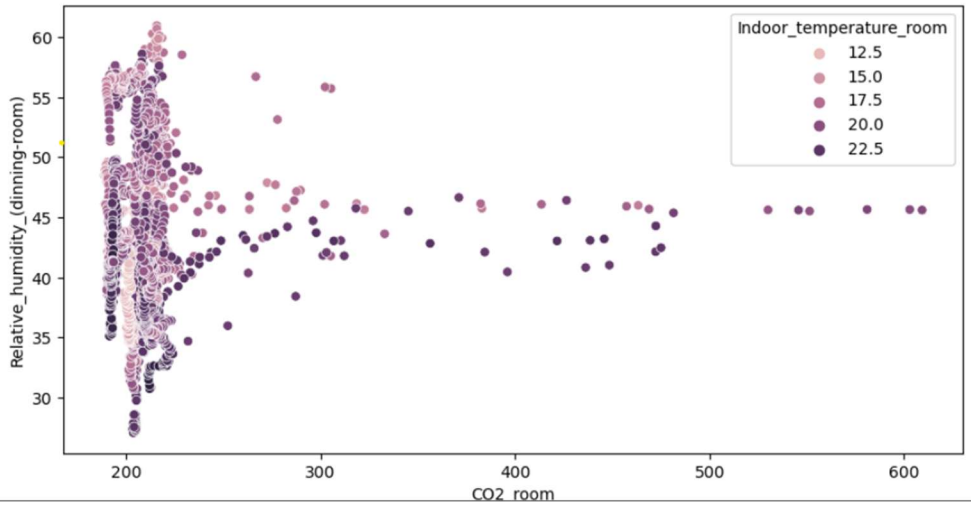
Independent variable = Relative_humidity_room

dependent variable = Relative_humidity_(dinning-room)

above graph represents good linear relation between above two varables

In [7]:
```
#multivariate analysis
plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x='CO2_room', y='Relative_humidity_(dinning-room)', hue='Indoor_temperature_room')
```

Out[7]: <Axes: xlabel='CO2_room', ylabel='Relative_humidity_(dinning-room)'>



According to above graph as CO2 increase in the room, indoor temperature also increases, room_temperature is low when co2 is arround 200 and humidity is between 30-43

In [8]:
```
#Descriptive Analysis
df.describe()
```

Out[8]:

	Id	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_Sur
count	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	2764.
mean	1381.500000	208.479123	211.065844	44.878420	47.321220	26.745381	40.732571	0.047033	325.
std	798.042397	27.032686	28.469144	6.587440	7.557795	23.298441	42.326087	0.206705	305.
min	0.000000	187.339000	188.907000	27.084000	29.594700	10.740000	11.328000	0.000000	0.
25%	690.750000	200.893250	202.682750	40.351975	42.531325	11.588700	13.265300	0.000000	0.
50%	1381.500000	207.045500	209.408000	45.434650	47.534700	11.801300	17.690000	0.000000	611.
75%	2072.250000	211.245500	213.218750	49.352675	52.685975	31.224000	52.057350	0.000000	619.
max	2763.000000	594.389000	609.237000	60.957300	62.594700	110.693000	162.965000	1.000000	624.

above we get total_count as count, mean, standard deviation, quantriles bu using describe function

In [9]:
```
#correlation between columns
corr = df.corr()
corr
```

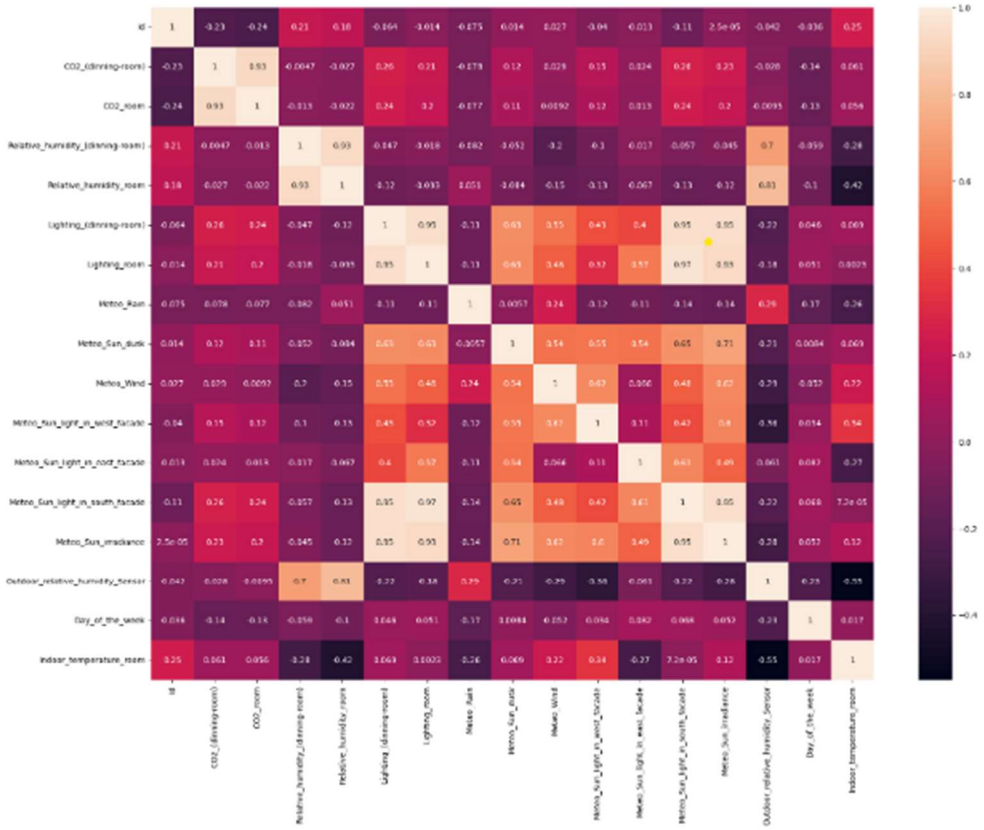
```
C:\Users\ajayp\AppData\Local\Temp\ipykernel_23920\2195921255.py:2: FutureWarning: The default value of numeric_only in DataFram
e.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_
only to silence this warning.
  corr = df.corr()
```

Out[9]:

	Id	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room
Id	1.000000	-0.226771	-0.242380	0.207623	0.177251	-0.064395	-0.014088
CO2_(dinning-room)	-0.226771	1.000000	0.925522	-0.004698	-0.027170	0.255059	0.214517
CO2_room	-0.242380	0.925522	1.000000	-0.013242	-0.022244	0.241727	0.196714
Relative_humidity_(dinning-room)	0.207623	-0.004698	-0.013242	1.000000	0.931267	-0.046600	-0.017654
Relative_humidity_room	0.177251	-0.027170	-0.022244	0.931267	1.000000	-0.116434	-0.092876
Lighting_(dinning-room)	-0.064395	0.255059	0.241727	-0.046600	-0.116434	1.000000	0.948650
Lighting_room	-0.014088	0.214517	0.196714	-0.017654	-0.092876	0.948650	1.000000
Meteo_Rain	-0.075370	-0.078120	-0.077395	-0.081597	0.051322	-0.108220	-0.112234
Meteo_Sun_dusk	0.013842	0.121910	0.108002	-0.052456	-0.083595	0.631730	0.629562
Meteo_Wind	0.027212	0.029190	0.009194	-0.197537	-0.153276	0.545034	0.475906
Meteo_Sun_light_in_west_facade	-0.040328	0.146489	0.115046	-0.100505	-0.130920	0.426375	0.315060
Meteo_Sun_light_in_east_facade	-0.012653	0.023633	0.012565	-0.016503	-0.067384	0.404444	0.565679
Meteo_Sun_light_in_south_facade	-0.112416	0.256665	0.242475	-0.057019	-0.133493	0.949340	0.965505
Meteo_Sun_irradiance	0.000025	0.226493	0.204212	-0.045240	-0.120310	0.948145	0.932237
Outdoor_relative_humidity_Sensor	-0.042246	-0.028235	-0.009462	0.696973	0.809993	-0.220635	-0.182360
Day_of_the_week	-0.036023	-0.135281	-0.129085	-0.058968	-0.103301	0.045991	0.051008
Indoor_temperature_room	0.246559	0.061361	0.056195	-0.275193	-0.417901	0.069264	0.002253

In [10]:
```
#plot above correlation
plt.subplots(figsize=(20,15))
sns.heatmap(corr, annot=True)
```

Out[10]: <Axes: >



In above we can see that similar feature like lighting, humidity, CO2 for room and dinning room are highly correlative

and Meteo_Sun_light_in_south_facade and Meteo_Sun_irradian are also highly correlativwe with lighting for both room and dinning room

Data Preprocessing

In [11]:
```
#information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Id                                  2764 non-null   int64
 1   Date                                2764 non-null   object
 2   Time                                2764 non-null   object
 3   CO2_(dinning-room)                  2764 non-null   float64
 4   CO2_room                            2764 non-null   float64
 5   Relative_humidity_(dinning-room)    2764 non-null   float64
 6   Relative_humidity_room              2764 non-null   float64
 7   Lighting_(dinning-room)             2764 non-null   float64
 8   Lighting_room                       2764 non-null   float64
 9   Meteo_Rain                          2764 non-null   float64
 10  Meteo_Sun_dusk                      2764 non-null   float64
 11  Meteo_Wind                          2764 non-null   float64
 12  Meteo_Sun_light_in_west_facade      2764 non-null   float64
 13  Meteo_Sun_light_in_east_facade      2764 non-null   float64
 14  Meteo_Sun_light_in_south_facade     2764 non-null   float64
 15  Meteo_Sun_irradiance                2764 non-null   float64
 16  Outdoor_relative_humidity_Sensor    2764 non-null   float64
 17  Day_of_the_week                     2764 non-null   float64
 18  Indoor_temperature_room             2764 non-null   float64
dtypes: float64(16), int64(1), object(2)
memory usage: 410.4+ KB
```

In [12]:
```
#checking for null values
#get sum of null values
df.isnull().sum()
```

Out[12]:
```
Id                                  0
Date                                0
Time                                0
CO2_(dinning-room)                  0
CO2_room                            0
Relative_humidity_(dinning-room)    0
Relative_humidity_room              0
Lighting_(dinning-room)             0
Lighting_room                       0
Meteo_Rain                          0
Meteo_Sun_dusk                      0
Meteo_Wind                          0
Meteo_Sun_light_in_west_facade      0
Meteo_Sun_light_in_east_facade      0
Meteo_Sun_light_in_south_facade     0
Meteo_Sun_irradiance                0
Outdoor_relative_humidity_Sensor    0
Day_of_the_week                     0
Indoor_temperature_room             0
dtype: int64
```

above it is seen that no null values in our data and now there is no need to handle them

In [13]:
```
#Handling categorical Data
#again through indormation
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Id                                  2764 non-null   int64
 1   Date                                2764 non-null   object
 2   Time                                2764 non-null   object
 3   CO2_(dinning-room)                  2764 non-null   float64
 4   CO2_room                            2764 non-null   float64
 5   Relative_humidity_(dinning-room)    2764 non-null   float64
 6   Relative_humidity_room              2764 non-null   float64
 7   Lighting_(dinning-room)             2764 non-null   float64
 8   Lighting_room                       2764 non-null   float64
 9   Meteo_Rain                          2764 non-null   float64
 10  Meteo_Sun_dusk                      2764 non-null   float64
 11  Meteo_Wind                          2764 non-null   float64
 12  Meteo_Sun_light_in_west_facade      2764 non-null   float64
 13  Meteo_Sun_light_in_east_facade      2764 non-null   float64
 14  Meteo_Sun_light_in_south_facade     2764 non-null   float64
 15  Meteo_Sun_irradiance                2764 non-null   float64
 16  Outdoor_relative_humidity_Sensor    2764 non-null   float64
 17  Day_of_the_week                     2764 non-null   float64
 18  Indoor_temperature_room             2764 non-null   float64
dtypes: float64(16), int64(1), object(2)
memory usage: 410.4+ KB
```

from above we can see data and time is only object data type and that data we are not including in our training so we don't need to handle categorical data. Other data types are float.

In [14]:
```
#Seperate depedent and independent variables
x=df.iloc[:,3:17] #INDEPENDENT VARIABLE
y=df.iloc[:,18:] #DEPENDENT VARIABLE
```

In [15]:
```
x.head()
```

Out[15]:

	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_Sun_dusk	Meteo_Wind
0	216.560	221.920	39.9125	42.4150	81.6650	113.520	0.0	623.360	1.42625
1	219.947	220.363	39.9267	42.2453	81.7413	113.605	0.0	623.211	1.59200
2	219.403	218.933	39.7720	42.2267	81.4240	113.600	0.0	622.656	1.89133
3	218.613	217.045	39.7760	42.0987	81.5013	113.344	0.0	622.571	1.82800
4	217.714	216.080	39.7757	42.0686	81.4657	113.034	0.0	622.400	2.38075

In [16]:
```
y.head()
```

Out[16]:

	Indoor_temperature_room
0	17.8275
1	18.1207
2	18.4367
3	18.7513
4	19.0414

In [17]:
```
#splitting data into train and test
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)
```

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

Model Building

4 classification algorithms are used in model building

```
Linear Regression Model
Random Forest model
Light Gradient Boost Model
Xgboost model
```

Linear Regression Model

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

Out[19]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

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

Out[21]:
```
0.17425288988529142
```

Random Forest model

In [22]:
```
rf = RandomForestRegressor()
```

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

```
C:\Users\ajayp\AppData\Local\Temp\ipykernel_23920\1149647727.py:1: DataConversionWarning: A column-vector y was passed when a 1
d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  rf.fit(x_train,y_train)
```

Out[23]:
```
RandomForestRegressor()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [24]: 1 pred1 = rf.predict(x_test)

In [25]: 1 pred1

Out[25]: array([21.618334, 21.27269 , 18.909898, 15.396623, 15.288172, 18.462552,
                18.204108, 16.202799, 18.322967, 18.155539, 22.021283, 16.58721 ,
                22.697447, 21.566378, 17.959353, 19.616625, 20.294848, 17.980272,
                21.621567, 16.256448, 19.670394, 16.147082, 16.783248, 20.391394,
                18.980112, 18.977458, 23.096354, 22.551678, 12.337911, 22.208924,
                17.938796, 20.168189, 20.216059, 21.008815, 18.018247, 15.705856,
                17.758723, 16.323511, 15.072934, 21.735057, 20.847159, 21.016636,
                22.55878 , 15.80513 , 16.095872, 16.313321, 17.598209, 19.787262,
                20.251053, 22.764732, 18.696833, 21.68169 , 19.4337  , 18.772987,
                22.171107, 22.857391, 17.831792, 17.5866  , 22.158708, 16.503597,
                22.838108, 19.484439, 19.497019, 19.784071, 22.503241, 17.07812 ,
                18.745744, 20.694292, 20.093232, 21.255181, 19.529829, 17.89106 ,
                17.402334, 22.201753, 22.010371, 16.270639, 20.117584, 22.671754,
                19.602795, 16.679414, 24.322784, 18.335331, 19.024312, 20.241291,
                18.410515, 19.442757, 17.542119, 22.302715, 19.638932, 17.476433,
                20.807519, 20.576808, 17.788778, 18.677864, 23.66118 , 16.202548,
                22.286528, 16.016571, 18.383636, 18.607819, 20.013217, 18.394084,
                19.919494, 19.502399, 17.802137, 18.299756, 16.971486, 17.110558,
                21.593663, 17.645954, 18.771928, 20.736208, 20.573099, 16.283229,

In [26]: 1 r2_score(y_test, pred1)

Out[26]: 0.9194343408777197
```

Light Gradient Boost Model

```
In [27]: 1 lg = lgb.LGBMRegressor()

In [28]: 1 lg.fit(x_train,y_train)

[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000268 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 3303
[LightGBM] [Info] Number of data points in the train set: 1934, number of used features: 14
[LightGBM] [Info] Start training from score 18.826209

Out[28]: LGBMRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [29]: 1 pred2 = lg.predict(x_test)

In [30]: 1 r2_score(y_test, pred2)

Out[30]: 0.9360756052934541
```

Xgboost Model

In [31]:
```
1 xg = xgb.XGBRegressor()
```

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

Out[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, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [33]:
```
1 pred3 = xg.predict(x_test)
```

In [34]:
```
1 r2_score(y_test,pred3)
```

Out[34]:
```
0.9312711740450739
```

In [35]:
```
1 xg1 = xgb.XGBRegressor()
2 xg1.fit(x_train_scaled,y_train)
```

Out[35]:
```
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, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Model	*r2_score*
lir	17.4%
rf	91.7%
lg	93.6%
xg	93.1%

Here above lg is the best model, hence we will export the model

In [36]:
```
1 import pickle
2 pickle.dump(lg,open('temperature.pkl','wb'))
```

More Evaluations

In [55]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [39]:
```
#for Light Gredient Model
mse = mean_squared_error(y_test, pred2)
mse
```

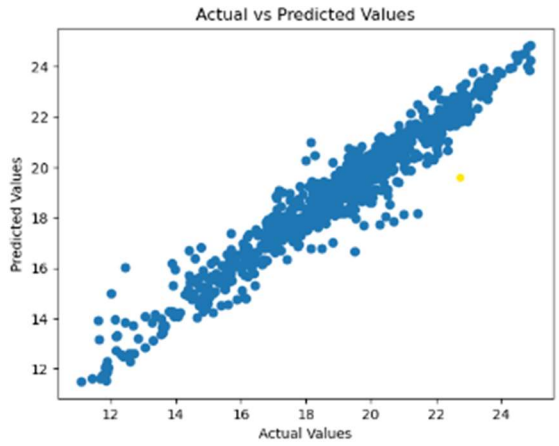
Out[39]: 0.5007536664270105

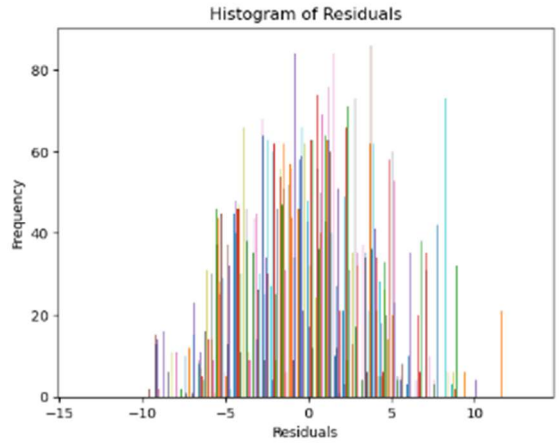
In [52]:
```
y_test_np = y_test.values
y_test_np
```

```
[18.068 ],
[22.3067],
[16.7953],
[23.588 ],
[21.7973],
[16.9173],
[19.6067],
[19.7733],
[17.856 ],
[21.188 ],
[15.684 ],
[18.836 ],
[16.2967],
[17.0253],
[20.8147],
[19.    ],
[19.0353],
[23.9307],
[22.6173],
```

In [53]:
```
#Scatter plot of actual vs predicted values
plt.scatter(y_test, pred2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()

# Residuals (difference between actual and predicted values)
residuals = y_test_np - pred2

# Histogram of residuals
plt.hist(residuals, bins=50)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Histogram of Residuals')
plt.show()
```

Actual vs Predicted Values





In [56]:
```
#for Light Gredient Model
mae = mean_absolute_error(y_test, pred2)
mae
```

Out[56]: 0.5089989459469727

```
In [71]:
#for Linear Regression Model
mae = mean_absolute_error(y_test, pred)
print(mae)
mse1 = mean_squared_error(y_test, pred)
print(mse)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred)

1.572677907866914
3.7427887580824297
1.934628842461114

Out[71]: 0.5222091780535016
```

```
In [73]:
#for Random Forest Model
mae = mean_absolute_error(y_test, pred1)
print(mae)
mse1 = mean_squared_error(y_test, pred1)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred1)

0.5211940722891568
0.6311135111844771
0.7944265297587166

Out[73]: 0.9194343408777197
```

```
In [76]:
#for Light Gradient Boost Model
mae = mean_absolute_error(y_test, pred2)
print(mae)
mse1 = mean_squared_error(y_test, pred2)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred2)

0.5089989459469727
0.5007536664270105
0.7076395031560989

Out[76]: 0.9360756052934541
```

```
In [77]:
#for XgBoost Model
mae = mean_absolute_error(y_test, pred3)
print(mae)
mse1 = mean_squared_error(y_test, pred3)
print(mse1)
nrmse = np.sqrt(mse1)
print(nrmse)
r2_score(y_test,pred3)

0.5326548784995941
0.5383893229516795
0.7337501774798283

Out[77]: 0.9312711740450739
```

Integration Code in Python (Flask Frame Work)

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 21 09:25:35 2023

@author: ajaypoonia
"""
#import the libraries
from flask import Flask, request, render_template
import pickle
import numpy as np
import pandas as pd

# load model
```

```
app = Flask(__name__, static_url_path='/static')

model = pickle.load(open(r'temperature.pkl', 'rb'))
app = Flask(__name__)

# Render HTML pages
@app.route("/")
def home():
    return render_template("index.html")

#contact page
@app.route("/contact")
def contact():
    return render_template("contact.html")

# Change the route to render predict.html
@app.route("/predict")
def predict():
    return render_template("predict.html")

# Retrieve value from UserInterface
@app.route('/output', methods=['post', 'get'])
def output():
    # reading the inputs given by the user
    input_feature = [float(x) for x in request.form.values()]
    input_feature = [np.array(input_feature)]
    print(input_feature)
    names = ['CO2_(dinning-room)', 'CO2_room', 'Relative_humidity_(dinning-
room)', 'Relative_humidity_room',
             'Lighting_(dinning-room)', 'Lighting_room', 'Meteo_Rain',
'Meteo_Sun_dusk', 'Meteo_Wind',
             'Meteo_Sun_light_in_west_facade', 'Meteo_Sun_light_in_east_facade',
'Meteo_Sun_light_in_south_facade',
             'Meteo_Sun_irradiance', 'Outdoor_relative_humidity_Sensor']
    print(names)
    data = pd.DataFrame(input_feature, columns=names)
    print(data)
    prediction = model.predict(data)
    print(prediction)
    return render_template('predict.html', result="Your room temperature will
be:  " + str(np.round(prediction[0])))

# Main Function
if __name__ == '__main__':
```

```
app.run(debug=True)
```

13.2. GitHub & Project Demo Link

Link - https://github.com/smartinternz02/SI-GuidedProject-603567-1697618783

Demo Video Link - https://youtu.be/KHjW4W3AdX0?si=vmEQG-wI6bdRaHNU