

SMART HOME TEMPERATURE

MINI PROJECT

Submitted to

**Jawaharlal Nehru Technological University
Hyderabad**

In partial fulfilment of the requirements for the award of the degree of

POST GRADUATION

In

MASTER OF COMPUTER APPLICATIONS

Submitted By

GADDALA VISHNU

23UK1F0006

Under the guidance of

Mrs. SHAZIA TAZEEN

Assistant Professor



DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

VAAGDEVI ENGINEERING COLLEGE(AUTONOMUS)

Affiliated to JNTUH(AUTONOMUS), HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) -506005

**DEPARTMENT OF
MASTER OF COMPUTER APPLICATIONS
VAAGDEVI ENGINEERING COLLEGE(AUTONOMOUS)**



CERTIFICATE OF COMPLETION PROJECT WORK REVIEW-I

This is to certify that the PG Project Phase-1 entitled “SMART HOME TEMPERATURE” is being submitted by GADDALA VISHNU (23UK1F0006) in partial fulfilment of the requirements for the award of the degree of Post Graduation in Master of Computer Applications to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023-2024.

Project Guide

Mrs. SHAZIA TAZEEN
(Assistant Professor)

HOD

DR. R. NAVEEN KUMAR
(Professor)

External

ACKNOWLEDGEMENT

I wish to take this opportunity to express my sincere gratitude and deep sense of respect to our beloved **Dr. Syed Musthak Ahmed**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this PG Project Phase-1 in the institute.

I extend my heartfelt thanks to **Dr. R. Naveen Kumar**, Head of the Department of MCA, Vaagdevi Engineering College for providing necessary infrastructure and thereby giving freedom to carry out the PG Project Phase-1.

I express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the PG Project Phase-1 and for their support in completing the PG Project Phase-1.

I express heartfelt thanks to the guide, **Mrs. SHAZIA TAZEEN**, Assistant professor, Department of CSE for her constant support and giving necessary guidance for completion of this PG Project Phase-1.

Finally, I express my sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

GADDALA VISHNU

(23UK1F0006)

ABSTRACT

The concept of smart homes has revolutionized modern living by integrating advanced technologies to enhance comfort and efficiency. Among the key components of a smart home, temperature control plays a crucial role in optimizing energy usage and occupant comfort. This abstract explores various aspects of smart home temperature management, focusing on the technologies, strategies, and benefits associated with these systems.

This abstract outlines a project focused on implementing a smart home temperature management system using IoT (Internet of Things) technology. The system integrates sensors for real-time environmental data collection and employs a microcontroller-based central unit for processing and controlling HVAC systems. Key objectives include optimizing indoor climate control based on user preferences and external conditions, facilitated by machine learning algorithms.

Smart home temperature systems utilize sensors, actuators, and intelligent algorithms to monitor and adjust indoor climate conditions in real-time. These systems can adapt to occupants' preferences, external weather conditions, and energy availability to maintain optimal thermal comfort while minimizing energy consumption. Integration with other smart devices, such as lighting and occupancy sensors, further enhances efficiency by coordinating activities based on real-time data.

Key features include smart thermostats equipped with sensors for real-time monitoring of temperature and humidity levels. These devices communicate with centralized controllers or cloud-based platforms, enabling remote access and control via mobile applications or voice commands. Machine learning algorithms analyse data from sensors and user interactions to predict and adjust heating, ventilation, and air conditioning (HVAC) settings automatically.

TABLE OF CONTENTS: -

1.INTRODUCTION.....	6
1.1OVERVIEW.....	6
1.2 PURPOSE	7-8
2.LITERATURE SURVEY.....	9
2.1 EXISTING PROBLEM	9
2.2 PROPOSED SOLUTION.....	10-11
3.THEORITICAL ANALYSIS.....	12
3.1 BLOCK DIAGRAM.....	12
3.2 HARDWARE /SOFTWARE DESIGNING	12-13
4.FLOWCHART.....	14
5.RESULTS.....	15-17
6.ADVANTAGES AND DISADVANTAGES.....	18-19
7.APPLICATIONS	20
8.CONCLUSION	20
9. FUTURE SCOPE.....	21
10. APPENDIX (SOURCE CODE)	22-33
11. CODE SNIPPETS.....	33-38

1.INTRODUCTION

1. 1. OVERVIEW

The concept of smart homes has transformed residential living by integrating advanced technologies to enhance comfort, convenience, and energy efficiency. Among these innovations, smart home temperature management stands out as a fundamental aspect that revolutionizes how homeowners regulate indoor climates.

Traditionally, controlling indoor temperatures relied on manual adjustments of thermostats, often leading to inefficiencies and discomfort due to inaccurate settings or uninformed decisions. However, with the advent of IoT (Internet of Things) technology, sensors, and machine learning algorithms, smart home temperature management systems have ushered in a new era of intelligent climate control.

Smart home temperature management are smart thermostats equipped with sensors that monitor not only temperature but also humidity levels and occupancy. These devices serve as the central hub for collecting real-time data and making informed decisions about HVAC (Heating, Ventilation, and Air Conditioning) system operations.

IoT connectivity enables these thermostats to communicate with other smart devices and cloud-based platforms, allowing homeowners to remotely access and control their home's temperature settings from anywhere using smartphones or computers. This connectivity not only enhances convenience but also enables energy-efficient practices by optimizing heating and cooling schedules based on real-time conditions and user preferences.

1. 2. PURPOSE

The purpose of smart home temperature management systems is multifaceted, aiming to enhance residential living through several key objectives:

1. **Comfort Optimization:** Smart temperature systems ensure that indoor environments remain consistently comfortable by dynamically adjusting heating, cooling, and ventilation settings based on real-time conditions and user preferences. This capability eliminates the need for manual adjustments and maintains an optimal climate throughout the day and night.
2. **Energy Efficiency:** By leveraging sensors, IoT connectivity, and machine learning algorithms, smart temperature systems optimize energy usage. They achieve this by analysing patterns in temperature data, occupancy, and external weather conditions to adjust HVAC operations intelligently. This optimization reduces energy waste and lowers utility costs while maintaining comfort levels.
3. **Remote Accessibility and Control:** IoT-enabled smart thermostats allow homeowners to remotely monitor and control their home's temperature settings via mobile applications or web interfaces. This feature provides flexibility and convenience, enabling adjustments from anywhere at any time, which is particularly useful for managing temperature while away from home or adjusting settings according to changing schedules.
4. **Integration with Smart Home Ecosystem:** Smart temperature management systems integrate seamlessly with other smart home devices and platforms. This interoperability enhances overall home automation capabilities, allowing for coordinated actions such as adjusting lighting, shades, or security systems based on temperature changes or occupancy status.
5. **Environmental Impact:** By reducing energy consumption and optimizing HVAC operations, smart temperature systems contribute to environmental sustainability. Lower energy usage

translates to reduced greenhouse gas emissions and a smaller carbon footprint, aligning with global efforts to mitigate climate change.

6. **User Experience and Convenience:** These systems provide a user-friendly interface for managing temperature settings, offering features such as personalized scheduling, voice control compatibility, and proactive alerts for maintenance or energy-saving tips. This enhances overall user experience by making temperature management effortless and intuitive.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

Several existing challenges and problems persist in the realm of smart home temperature management systems:

- **Compatibility and Integration:** One of the significant hurdles is ensuring seamless compatibility and integration with existing HVAC systems and other smart home devices. Different manufacturers use varied communication protocols and standards, making it challenging for devices to communicate effectively and operate in harmony.
- **Reliability and Stability:** Smart temperature systems rely heavily on connectivity to function optimally. Issues such as network outages, firmware updates, or compatibility issues between devices can lead to instability or malfunctions in temperature control, affecting user comfort and convenience.
- **User Interface Complexity:** While smart thermostats offer advanced features and capabilities, the complexity of their user interfaces can pose challenges for some users. Complicated setup processes, unintuitive controls, or overwhelming options may deter users from fully utilizing the system's capabilities.
- **Privacy and Security Concerns:** IoT devices, including smart thermostats, collect and transmit sensitive data about user habits and home occupancy. Ensuring robust data encryption, secure communication protocols, and protection against cyber threats are critical to safeguarding user privacy and preventing unauthorized access to personal information.
- **Cost and Affordability:** The initial cost of purchasing and installing smart temperature management systems, including smart thermostats and compatible devices, can be a barrier for some homeowners. Additionally, ongoing costs related to maintenance,

updates, and potential subscription fees for advanced features may impact affordability and adoption rates.

2.2 PROPOSED SOLUTION

Smart home temperature management systems involve addressing various technical, practical, and user-related aspects. Here are some proposed solutions:

1. Compatibility and Integration:

- **Standardization:** Encourage industry-wide adoption of common communication protocols and standards for IoT devices to improve interoperability.
- **Compatibility Testing:** Conduct rigorous compatibility testing between smart thermostats, HVAC systems, and other smart devices to ensure seamless integration.
- **Firmware Updates:** Provide regular firmware updates and compatibility patches to ensure devices remain compatible with evolving technologies.

2. Reliability and Stability:

- **Redundancy and Backup:** Implement redundant communication paths (e.g., Wi-Fi and Zigbee) to maintain connectivity during network outages.
- **Local Control:** Enable local control options that do not rely solely on cloud connectivity, ensuring continuous operation even when internet access is disrupted.
- **Stability Testing:** Conduct thorough stability testing during product development to identify and address potential issues before deployment.

3. User Interface Complexity:

- **Simplified Setup:** Streamline initial setup processes with guided tutorials, intuitive interfaces, and automated configuration wizards.
- **Customization Options:** Provide customizable dashboards and settings to accommodate varying user preferences and technical expertise levels.
- **User Feedback:** Gather user feedback through usability testing and surveys to continually improve the user interface and overall user experience.

4. Privacy and Security Concerns:

- **Data Encryption:** Implement robust encryption protocols (e.g., TLS, AES) to secure data transmission between devices and cloud servers.
- **Privacy Controls:** Provide clear privacy settings and options for users to control data sharing and access permissions.
- **Security Updates:** Ensure prompt security updates and patches to address vulnerabilities and protect against cyber threats.

5. Cost and Affordability:

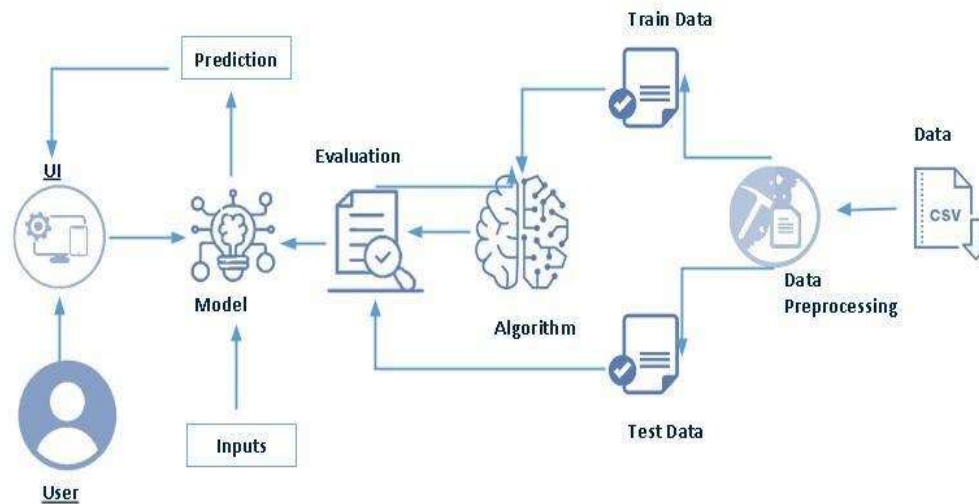
- **Price Competitiveness:** Work towards reducing manufacturing costs and offering competitive pricing for smart thermostats and related devices.
- **Energy Savings Incentives:** Partner with utilities to offer rebates or incentives for installing energy-efficient smart temperature systems.
- **Long-term Cost Analysis:** Educate consumers on the long-term cost savings potential of smart temperature systems to justify initial investments.

Implementing these proposed solutions requires collaboration among manufacturers, developers, regulators, and consumers to

improve product design, enhance user experience, ensure security, promote energy efficiency, and foster widespread adoption of smart home temperature management systems.

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



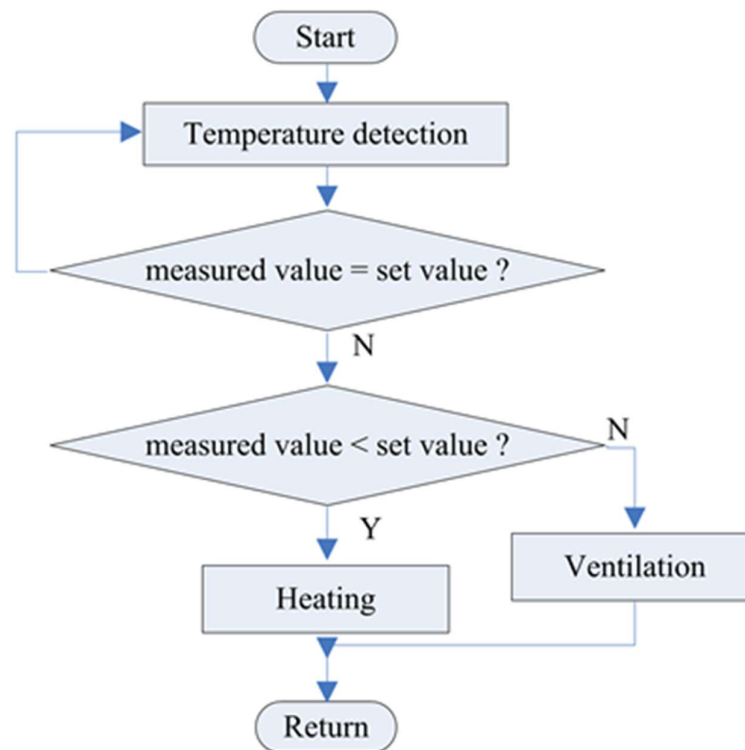
3.2. SOFTWARE DESIGNING

The following is the Software required to complete this project:

- **JUPYTER NOOTBOOK:** will serve as the development and execution environment for your predictive modelling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.
- **Dataset (CSV File):** The dataset in CSV format is essential for training and testing your predictive model. It should include historical smart home temperature data, weather information, and other relevant features.

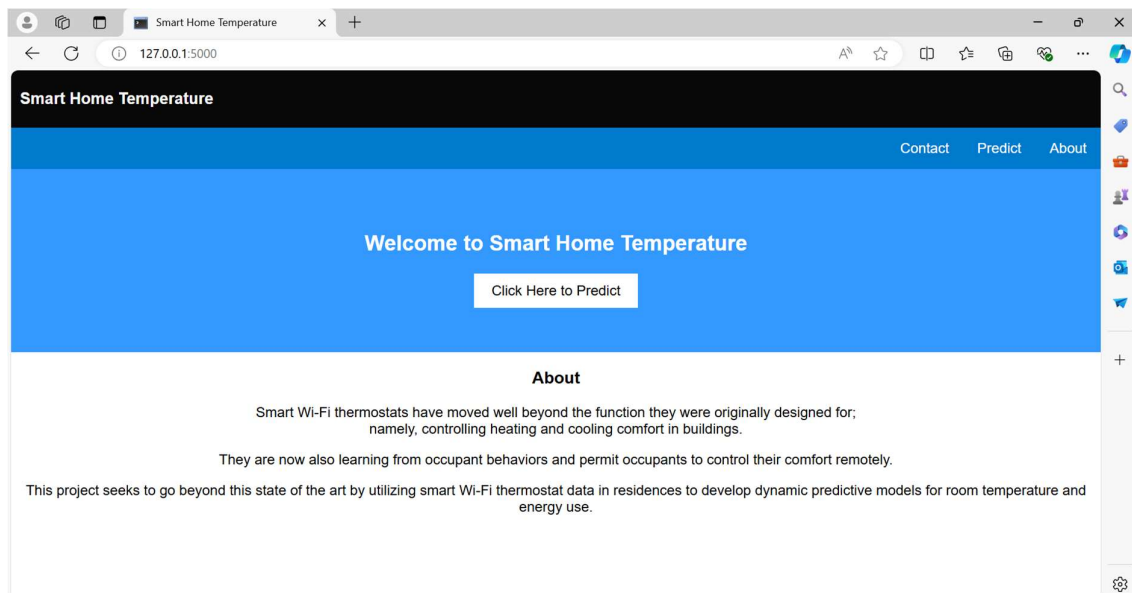
- **Data Preprocessing Tools:** Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.
- **Feature Selection/Drop:** Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.
- **Model Training Tools:** Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the SMART HOME TEMPERATURE prediction task.
- **Model Accuracy Evaluation:** After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict TEMPERATURE categories based on historical data.
- **UI Based on Flask Environment:** Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data or view TEMPERATURE predictions, and recommended precautions.
- Jupyter Notebook will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the TEMPERATURE predictions and associated TEMPERATURE information.

4.FLOWCHART



5.RESULT

HOME PAGE



PREDICTIONS

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/index". The page title is "Smart Home Temperature". The interface features a black header with the title and a blue navigation bar with "Contact" and "About" links. A large blue banner reads "Welcome to Smart Home Temperature". Below this, a central white box contains several input fields for predictions:

- co2:** CO2 in the room
- Humidity:** Humidity in the room
- Lighting:** Lighting in the room
- Rain:** Rain in Last 15 Minutes
- Wind:** Wind at outside

Below the central box, another white box contains:

- Sunlight at West:** Sunlight at west side of
- Outdoor Humidity:** Humidity at outside
- Submit** button

The browser's right sidebar shows various extension icons, and the bottom of the page has a scrollbar.

Smart Home Temperature

Contact About

Welcome to Smart Home Temperature

co2:

11

Humidity:

22

Lighting:

15

Rain:

68

Wind:

8

Sunlight at West:

9

Outdoor Humidity:

6

Submit

Smart Home Temperature

Contact About

Welcome to Smart Home Temperature

Prediction Result

The predicted temperature is: [21.70948005]

[Go back to Home](#)

6.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- **Energy Efficiency:** Smart thermostats can optimize heating and cooling based on occupancy and usage patterns, leading to energy savings by reducing unnecessary heating or cooling.
- **Cost Savings:** By using energy more efficiently, smart temperature control systems can lower utility bills over time.
- **Remote Access:** Users can control the temperature of their homes remotely via smartphone apps or web portals, allowing adjustment from anywhere.
- **Integration with Other Devices:** Smart thermostats can integrate with other smart home devices such as voice assistants (e.g., Alexa, Google Assistant) or home automation systems for seamless control.
- **Learning Capabilities:** Some smart thermostats learn your habits and adjust temperature settings automatically, optimizing comfort without manual intervention.
- **Comfort and Convenience:** Maintaining consistent temperatures and being able to adjust settings easily contributes to overall comfort and convenience.
- **Environmental Benefits:** By reducing energy consumption, smart temperature control systems contribute to a smaller carbon footprint and environmental sustainability.

DISADVANTAGES:

- **Initial Cost:** Smart thermostats and their installation can be more expensive than traditional thermostats, which might deter some homeowners.
- **Complexity:** The technology behind smart thermostats can be complex, requiring some users to spend time learning how to use and configure them properly.
- **Compatibility Issues:** Compatibility with existing HVAC systems or other smart home devices can sometimes be a concern, requiring careful research before purchase.
- **Dependence on Internet and Power:** Smart thermostats rely on internet connectivity and power to function properly. Internet outages or power failures can disrupt their operation.
- **Privacy and Security Risks:** Like any internet-connected device, smart thermostats can pose privacy and security risks if not properly secured or if data is mishandled by the manufacturer.
- **Potential for Malfunction:** As with any electronic device, there is a possibility of malfunction or software bugs that could affect performance or reliability.
- **Learning Curve:** Users who are not tech-savvy may find the setup and programming of smart thermostats challenging, which could impact their ability to fully utilize all features.
- **Constant Updates:** Smart thermostats may require regular firmware updates to fix bugs, improve functionality, or patch security vulnerabilities, which could require user attention and time.

7.APPLICATIONS

1. Energy Efficiency in Residential Settings:

Remote access allows homeowners to monitor and adjust temperatures from anywhere, ensuring comfort while minimizing energy waste when no one is home.

2. Enhanced Comfort and Convenience:

Integration with voice assistants (e.g., Alexa, Google Assistant) or smart home ecosystems enables convenient voice commands or automated routines for temperature adjustments.

3. Temperature Zoning for Optimal Comfort:

Smart temperature control systems support zoning capabilities, allowing different areas of the home to be heated or cooled independently based on occupancy or specific comfort needs.

4. Remote Monitoring and Maintenance:

Monitoring sensors can detect anomalies in temperature or HVAC system performance, alerting homeowners to potential issues like equipment malfunctions or inefficiencies.

8.CONCLUSION

In conclusion, smart home temperature control systems offer a range of benefits and applications that enhance comfort, efficiency, and convenience in both residential and commercial settings. By optimizing heating and cooling based on occupancy patterns, preferences, and external conditions, these systems contribute to significant energy savings and reduced utility costs. The ability to monitor and adjust temperatures remotely via smartphone apps or integrate with other smart devices adds further convenience and control to users' daily lives. Moreover, features such as temperature zoning and proactive maintenance alerts ensure tailored comfort and reliability, making smart temperature control a valuable investment for modern homes and businesses alike. As technology continues to advance, these systems are likely to play an increasingly integral role in creating sustainable and comfortable indoor environments.

9.FUTURE SCOPE

The future scope of smart home temperature control systems:

- 1. Enhanced Artificial Intelligence and Machine Learning:** Future systems will likely leverage AI and machine learning algorithms even more effectively to optimize temperature control based on occupants' habits, weather forecasts, and building characteristics. This could lead to further energy savings and personalized comfort.
- 2. Integration with IoT and Smart Grids:** Greater integration with the Internet of Things (IoT) and smart grid technologies will enable smart thermostats to communicate with utility providers and adjust energy usage in response to real-time electricity prices and grid demands. This can lead to more efficient energy consumption and grid stability.
- 3. Health and Wellness Features:** There is a growing interest in integrating health and wellness features into smart home systems. Future temperature control systems may incorporate sensors to monitor indoor air quality, humidity levels, and even analyze occupants' health metrics to create healthier indoor environments.
- 4. Interoperability and Standardization:** As the smart home ecosystem expands, there will be a push for interoperability and standardization among devices and platforms. This will allow different brands and types of smart temperature control systems to work seamlessly together, enhancing flexibility and user experience.
- 5. User Interface and Experience:** Future systems will likely focus on enhancing user interfaces and experiences, making them more intuitive and user-friendly. This includes advancements in app design, voice control capabilities, and personalized recommendations based on user preferences.

10.APPENDIX

Model building:

1)Dataset

2)Jupyter Notebook and VS code Application Building

1. HTML file (home file, Index file, Predict file)
2. Models in pickle format

SOURCE CODE:

Home.html

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Smart Home Temperature</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    .header {
      background-color: #070707fe;
      background-size: xx-small;
      color: rgb(249, 247, 247);
      text-align: center;
      padding: 10px;
      font-size: xx-small;
      text-align: left;
```

```

}
.nav-bar {
    background-color: #007acc;
    overflow: hidden;
}
.nav-bar a {
    float: right;
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
.banner {
    background-color: #3399ff;
    text-align: center;
    padding: 50px 0;
    color: white;
}
.predict-button {
    background-color: white;
    border: none;
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
}
.about1{
    color: black;
    text-align: center;

```

```

    }
</style>
</head>
<body>
    <div class="header">
        <h1>Smart Home Temperature</h1>
    </div>
    <div class="nav-bar">
        <a href="#about">About</a>
        <a href="/index">Predict</a>
        <a href="#contact">Contact</a>
    </div>
    <div class="banner">
        <h2>Welcome to Smart Home Temperature</h2>
        <a href="{{url_for('index')}}">
            <button class="predict-button">Click Here to Predict</button></a>
    </div>
    <div class="about1">
        <h3>About</h3>
        <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>
        <p>They are now also learning from occupant behaviors and permit occupants to
control their comfort remotely.</p>
        <p>This project seeks to go beyond this state of the art by utilizing smart Wi-Fi
thermostat data in residences to develop dynamic predictive models for room temperature
and energy use.</p>
    </div>

</body>
</html>

```


Index.html

```
<html lang="en">

<head>

  <meta charset="UTF-8">

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

  <title>Smart Home Temperature</title>

<style>

  body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-color: #f4f4f4;

  }

  .header {

    background-color: #070707fe;

    background-size: xx-small;

    color: rgb(249, 247, 247);

    text-align: center;

    padding: 10px;

    font-size: xx-small;

    text-align: left;

  }

  .nav-bar {

    background-color: #007acc;

    overflow: hidden;

  }

  .nav-bar a {

    float: right;

    display: block;

    color: white;
```

```

    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

.main {
    text-align: center;
    padding: 20px;
    background-color: #007BFF;
    color: white;
}

.form-container {
    background-color: white;
    margin: 20px auto;
    padding: 20px;
    width: 60%;
    max-width: 300px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

.form-group {
    margin: 10px 0;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
}

.form-group input {
    width: calc(100% - 20px);
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;

```

```

        border-radius: 4px;
    }
    .submit-btn {
        background-color: #007BFF;
        color: white;
        padding: 10px 20px;
        border: none;
        border-radius: 4px;
        cursor: pointer;
    }
</style>
</head>
<body>
    <div class="header">
        <h1>Smart Home Temperature</h1>
    </div>
    <div class="nav-bar a">
        <a href="#about">About</a>
        <a href="#contact">Contact</a>
    </div>
    <div class="main">
        <h2>Welcome to Smart Home Temperature</h2>
    </div>
    <div class="form-container">
        <form action="/predict" method="post">
            <div class="form-group">
                <label for="CO2_room">co2:</label>
                <input type="number" name="CO2_room" placeholder="CO2 in the room">
                <label for="Relative_humidity_room">Humidity:</label>
                <input type="number" name="Relative_humidity_room" placeholder="Humidity in the
room">

```

```

<label for="Lighting_room">Lighting:</label>
<input type="number" name="Lighting_room" placeholder="Lighting in the room">
<label for="Meteo_Rain">Rain:</label>
<input type="number" name="Meteo_Rain" placeholder="Rain in Last 15 Minutes">
<label for="Meteo_Wind">Wind:</label>
<input type="number" name="Meteo_Wind" placeholder="Wind at outside">
<label for="Meteo_Sun_light_in_west_facade">Sunlight at West:</label>
<input type="number" name="Meteo_Sun_light_in_west_facade"
placeholder="Sunlight at west side of">
<label for="Outdoor_relative_humidity_Sensor">Outdoor Humidity:</label>
<input type="number" name="Outdoor_relative_humidity_Sensor"
placeholder="Humidity at outside">
</div>
<button type="submit" class="submit-btn">Submit</button>
</form>
</div>
</body>
</html>

```

PREDICT.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Smart Home Temperature</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;

```

```

    background-color: #f4f4f4;
}

.header {
    background-color: #070707fe;
    background-size: xx-small;
    color: rgb(249, 247, 247);
    text-align: center;
    padding: 10px;
    font-size: xx-small;
    text-align: left;
}

.nav-bar {
    background-color: #007acc;
    overflow: hidden;
}

.nav-bar a {
    float: right;
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

.main {
    text-align: center;
    padding: 20px;
    background-color: #007BFF;
    color: white;
}

.form-container {

```

```
background-color: white;
margin: 20px auto;
padding: 20px;
width: 60%;
max-width: 300px;
border-radius: 8px;
box-shadow: 0 0 10px rgba(0,0,0,0.1);
}

.form-group {
margin: 10px 0;
}

.form-group label {
display: block;
margin-bottom: 5px;
}

.form-group input {
width: calc(100% - 20px);
padding: 10px;
margin-bottom: 10px;
border: 1px solid #ccc;
border-radius: 4px;
}

.submit-btn {
background-color: #007BFF;
color: white;
padding: 10px 20px;
border: none;
border-radius: 4px;
cursor: pointer;
}
```

```

    </style>
</head>
<body>
    <div class="header">
        <h1>Smart Home Temperature</h1>
    </div>
    <div class="nav-bar a">
        <a href="#about">About</a>
        <a href="#contact">Contact</a>
    </div>
    <div class="main">
        <h2>Welcome to Smart Home Temperature</h2>
    </div>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Result</title>
</head>
<body>
    <center>
        <h1>Prediction Result</h1>
        <p><h2>The predicted temperature is: {{ prediction_text }}</h2></p>
    </center>
    <a href="/">Go back to Home</a>
</body>
</html>

```

APP.PY

```
from flask import Flask,request,render_template
import pickle
import numpy as np
import pandas as pd
import os
app=Flask(__name__)
encoders_path = os.path.dirname(os.path.abspath(__file__))
model=pickle.load(open(os.path.join(encoders_path, 'temperature.pkl'),'rb'))

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

@app.route('/index',methods=['GET','POST'])
def index():
    return render_template("index.html")

@app.route('/predict', methods=['POST'])
def predict():
    co2=float(request.form['CO2_room'])
    humidity=float(request.form['Relative_humidity_room'])
    lighting=float(request.form['Lighting_room'])
    rain=float(request.form['Meteo_Rain'])
    wind=float(request.form['Meteo_Wind'])
    sunlight=float(request.form['Meteo_Sun_light_in_west_facade'])
    outdoor_humidity=float(request.form['Outdoor_relative_humidity_Sensor'])
    pred=[[co2,humidity,lighting,rain,wind,sunlight,outdoor_humidity]]
```



```

print(pred)

output=model.predict(pred)

print(output)

return render_template('predict.html',prediction_text=output)

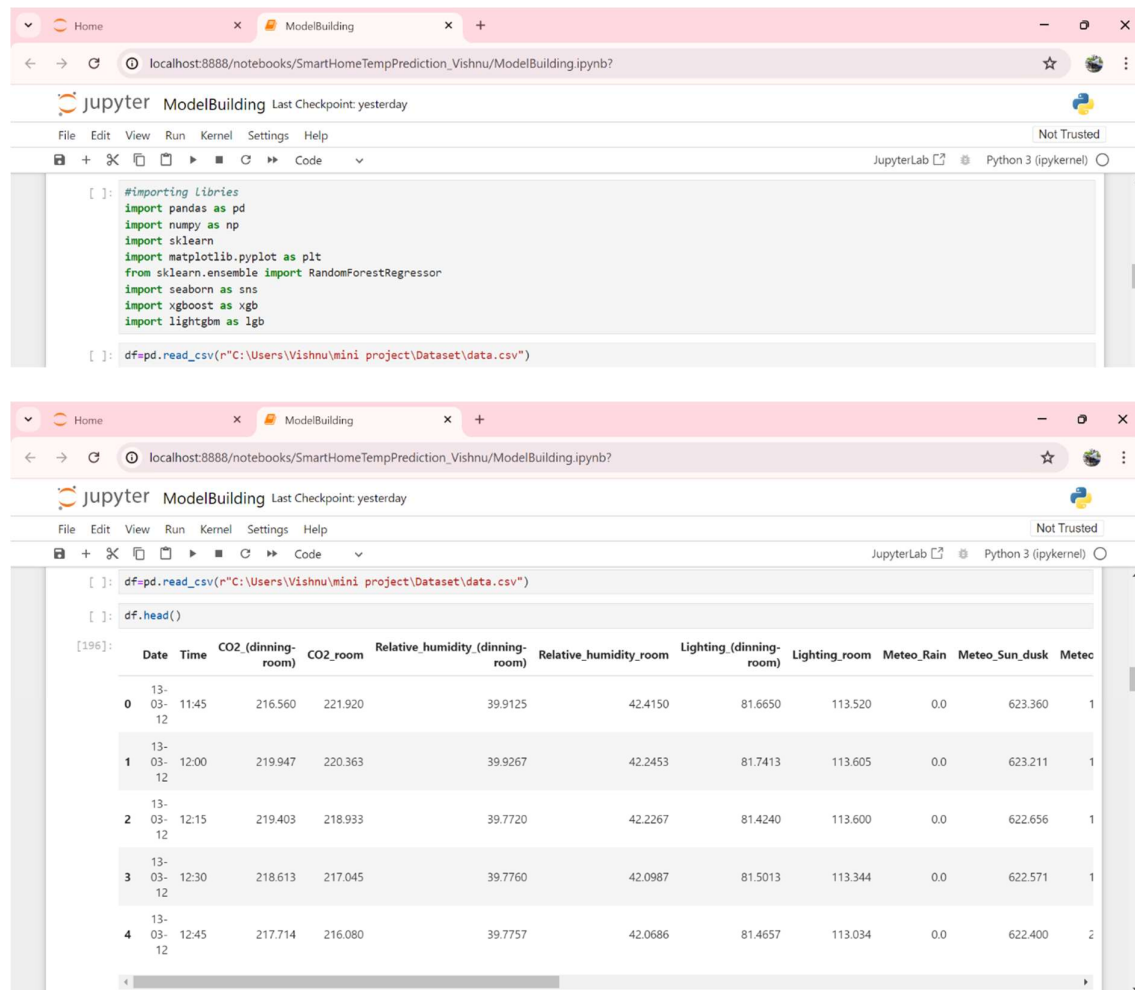
if __name__ == '__main__':

    app.run(debug=True)

```

11.CODE SNIPPETS

MODEL BUILDING



The top screenshot shows the Jupyter Notebook interface with the following code in the code cell:

```

[ ]: #importing Libraries
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestRegressor
import seaborn as sns
import xgboost as xgb
import lightgbm as lgb

[ ]: df=pd.read_csv(r"C:\Users\Vishnu\mini project\Dataset\data.csv")

```

The bottom screenshot shows the output of the code, which is a DataFrame with 196 rows and 11 columns. The output is displayed as a table with the following columns: Date, Time, CO2 (dinning-room), CO2_room, Relative_humidity (dinning-room), Relative_humidity_room, Lighting (dinning-room), Lighting_room, Meteo_Rain, Meteo_Sun_dusk, and Metec.

	Date	Time	CO2 (dinning-room)	CO2_room	Relative_humidity (dinning-room)	Relative_humidity_room	Lighting (dinning-room)	Lighting_room	Meteo_Rain	Meteo_Sun_dusk	Metec
0	13-03-12	11:45	216.560	221.920	39.9125	42.4150	81.6650	113.520	0.0	623.360	1
1	13-03-12	12:00	219.947	220.363	39.9267	42.2453	81.7413	113.605	0.0	623.211	1
2	13-03-12	12:15	219.403	218.933	39.7720	42.2267	81.4240	113.600	0.0	622.656	1
3	13-03-12	12:30	218.613	217.045	39.7760	42.0987	81.5013	113.344	0.0	622.571	1
4	13-03-12	12:45	217.714	216.080	39.7757	42.0686	81.4657	113.034	0.0	622.400	2

```
ModelBuilding Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help

JupyterLab Python 3 (ipykernel)

4 03-12:45 217.714 216.080 39.7757 42.0686 81.4657 113.034 0.0 622.400 2
12

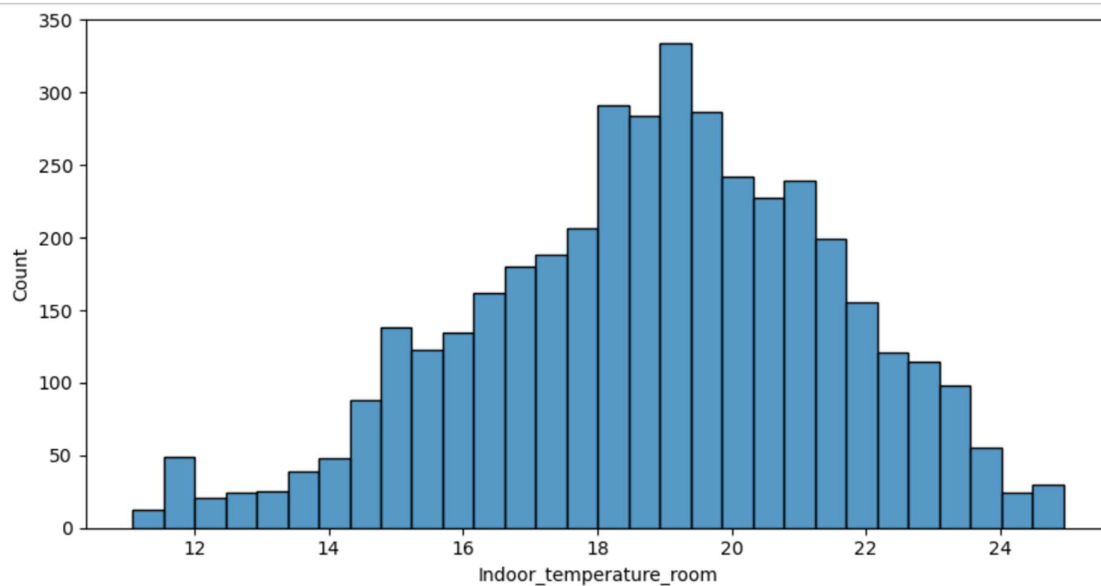
[ ]: df.shape
[197]: (4137, 18)
```

```
ModelBuilding Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help

JupyterLab Python 3 (ipykernel)

[ ]: plt.figure(figsize=(10,5))
sns.histplot(data=df, x='Indoor_temperature_room',)
[198]: <Axes: xlabel='Indoor_temperature_room', ylabel='Count'>
```



```
ModelBuilding Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help

JupyterLab Python 3 (ipykernel)

[ ]: plt.figure(figsize=(10,5))
sns.scatterplot(data=df, x='Relative_humidity_room', y='Relative_humidity_(dinning-room)')
[199]: <Axes: xlabel='Relative_humidity_room', ylabel='Relative_humidity_(dinning-room)'>
```


Home x ModelBuilding x +

localhost:8888/notebooks/SmartHomeTempPrediction_Vishnu/ModelBuilding.ipynb?

Jupyter ModelBuilding Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Not Trusted

JupyterLab Python 3 (ipykernel)

```
[ ]: df.describe()
```

```
[201]:
```

	CO2_(dinning-room)	CO2_room	Relative_humidity_(dinning-room)	Relative_humidity_room	Lighting_(dinning-room)	Lighting_room	Meteo_Rain	Meteo_Sun_dusk	Meteo_Wind
count	4137.000000	4137.000000	4137.000000	4137.000000	4137.000000	4137.000000	4137.000000	4137.000000	4137.000000
mean	206.599835	209.611623	42.389879	44.546069	28.970248	42.335496	0.038756	335.094312	1.304623
std	22.763114	24.183477	7.215405	8.297436	25.684356	42.602571	0.187128	304.513038	1.223829
min	187.339900	188.907000	26.173300	27.256000	10.740000	11.328000	0.000000	0.606667	0.000000
25%	200.228000	201.707000	36.088000	38.446700	11.540700	13.509300	0.000000	0.650000	0.168667
50%	205.131000	208.907000	42.776000	44.802700	14.126700	22.085300	0.000000	612.821000	0.962667
75%	210.016000	212.331000	47.584000	50.301300	40.034700	55.064000	0.000000	619.712000	2.225330
max	594.389000	609.237000	60.957300	62.594700	111.797000	162.965000	1.000000	625.003000	6.321330

Home x ModelBuilding x +

localhost:8888/notebooks/SmartHomeTempPrediction_Vishnu/ModelBuilding.ipynb?

Jupyter ModelBuilding Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Not Trusted

JupyterLab Python 3 (ipykernel)

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4137 entries, 0 to 4136
Data columns (total 18 columns):
#   Column                                     Non-Null Count  Dtype
---  ---
0    Date                                     4137 non-null   object
1    Time                                     4137 non-null   object
2    CO2_(dinning-room)                     4137 non-null   float64
3    CO2_room                                4137 non-null   float64
4    Relative_humidity_(dinning-room)       4137 non-null   float64
5    Relative_humidity_room                 4137 non-null   float64
6    Lighting_(dinning-room)                4137 non-null   float64
7    Lighting_room                          4137 non-null   float64
8    Meteo_Rain                             4137 non-null   float64
9    Meteo_Sun_dusk                         4137 non-null   float64
10   Meteo_Wind                             4137 non-null   float64
11   Meteo_Sun_light_in_west_facade         4137 non-null   float64
12   Meteo_Sun_light_in_east_facade         4137 non-null   float64
13   Meteo_Sun_light_in_south_facade        4137 non-null   float64
14   Meteo_Sun_irradiance                   4137 non-null   float64
15   Outdoor_relative_humidity_Sensor       4137 non-null   float64
16   Day_of_the_week                        4137 non-null   float64
17   Indoor_temperature_room                4137 non-null   float64
dtypes: float64(16), object(2)
memory usage: 581.9+ KB
```

Home x ModelBuilding x +

localhost:8888/notebooks/SmartHomeTempPrediction_Vishnu/ModelBuilding.ipynb?

Jupyter ModelBuilding Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Not Trusted

JupyterLab Python 3 (ipykernel)

```
[ ]: df.isnull().sum()
```

```
[203]:
```

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	

```
ModelBuilding

[22]: # Assuming your DataFrame is named 'df'
df.drop(['Date', 'Time'], axis=1, inplace=True)

[23]: df.drop(['CO2_(dinning-room)', 'Relative_humidity_(dinning-room)', 'Lighting_(dinning-room)', 'Meteo_Sun_dusk', 'Meteo_Sun_light_in_east_facade', 'Meteo_Sun_1',
4

[24]: x=df.iloc[:, :-1].values
y=df.iloc[:, -1].values

[25]: x.shape

[25]: (4137, 7)

[26]: 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)

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

```
ModelBuilding

[28]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
lir=LinearRegression()
lir.fit(x_train_scaled,y_train)

[28]: + LinearRegression
LinearRegression()

[29]: x_test_scaled.shape

[29]: (1242, 7)

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

[31]: r2_score(pred,y_test)

[31]: -0.4426495167688036
```

```
ModelBuilding

[32]: rf=RandomForestRegressor()

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

[33]: + RandomForestRegressor
RandomForestRegressor()

[34]: x_train.shape

[34]: (2895, 7)

[35]: x_test.shape

[35]: (1242, 7)

[36]: pred=rf.predict(x_test)

[37]: pred

[37]: array([23.16513337, 17.523718 , 21.17828828, ..., 20.2239653 ,
17.87788002, 18.733412  ])
```

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

[38]: 0.873384878414834
```

A Jupyter Notebook window titled 'ModelBuilding' with a 'Last Checkpoint: yesterday' message. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations, running, and code execution. The code cell shows the following:

```
[39]: lg=LGBMRegressor()

[40]: lg.fit(x_train,y_train)

[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.002077 seconds.
You can set 'force_row_wise=true' to remove the overhead.
And if memory is not enough, you can set 'force_col_wise=true'.
[LightGBM] [Info] Total Bins 1539
[LightGBM] [Info] Number of data points in the train set: 2895, number of used features: 7
[LightGBM] [Info] Start training from score 18.804740

[40]: LGBMRegressor()

[41]: pred=lg.predict(x_test)

[42]: r2_score(y_test,pred)

[42]: 0.8569554082913747
```

The output for cell [40] shows a detailed log from LightGBM, including information about multi-threading, memory usage, and training progress.

A Jupyter Notebook window titled 'ModelBuilding' with a 'Last Checkpoint: yesterday' message. The code cell shows the following:

```
[43]: xg=XGBRegressor()

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

[44]: XGBRegressor

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, ...)

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

[46]: r2_score(y_test,pred)

[46]: 0.8547922627762138
```

The output for cell [44] shows the full parameter list for the XGBRegressor object.

A Jupyter Notebook window titled 'ModelBuilding' with a 'Last Checkpoint: yesterday' message. The code cell shows the following:

```
[47]: import pickle
pickle.dump(rf,open('temperature.pkl','wb'))

[48]: import os
os.getcwd()

[48]: 'C:\\Users\\Vishnu\\SmartHomeTempPrediction_Vishnu'
```