A

Major Project Report On

# DECENTRALIZED SECURE FILE SHARING SYSTEM USING BLOCKCHAIN AND IPFS

Submitted to JNTU HYDERABAD

In Partial Fulfilment of the requirements for the Award of Degree of

**BACHELOR OF TECHNOLOGY IN**

**INFORMATION TECHNOLOGY**

Submitted By

**BANDARI SHIVAMANI (218R1A1208) ARIKELA NANDINI (218R1A1204) BATHINI JAHNAVI (218R1A1210)**

**ULLENGULA SAI TEJA (218R1A1215)**

Under the Esteemed guidance of

## Mr. K. Anil

Assistant Professor, Department of IT



# Department of Information Technology

CMR ENGINEERING COLLEGE

# (UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU Hyderabad)(Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

**(2024-2025)**

CMR ENGINEERING COLLEGE

# (UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU Hyderabad)(Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

**Department of Information Technology**

****

**CERTIFICATE**

This is to certify that the project entitled **“DECENTRALIZED SECURE FILE SHARING SYSTEM USING BLOCKCHAIN AND IPFS”** is a bonafide work carried out by

**BANDARI SHIVAMANI (218R1A1208) ARIKELA NANDINI (218R1A1204) BATHINI JAHNAVI (218R1A1210)**

**ULLENGULA SAI TEJA (218R1A1215)**

in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **INFORMATION TECHNOLOGY** from CMR Engineering College, affiliated to JNTU Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

|  |  |
| --- | --- |
| Internal Guide | Head of the Department |
| **Mr. K. ANIL** | **Dr. MADHAVI PINGILI** |
| Assistant Professor | Professor & HOD |
| Department of IT | Department of IT |
| CMR Engineering College Hyderabad | CMR Engineering College Hyderabad |

Project External Examiner

**DECLARATION**

This is to certify that the work reported in the present project entitled **“DECENTRALIZED SECURE FILE SHARING SYSTEM USING BLOCKCHAIN AND IPFS”** is a record of bonafide work done by us in the Department of Information Technology, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submittedto any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

**BANDARI SHIVAMANI (218R1A1208) ARIKELA NANDINI (218R1A1204) BATHINI JAHNAVI (218R1A1210)**

**ULLENGULA SAI TEJA (218R1A1215)**

**ACKNOWLEDGEMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, HOD, **Department of IT, CMR Engineering College** for their constant support.

We are extremely thankful to **Mr. K. Anil,** Assistant Professor, Internal Guide, Department of IT, for his constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

**BANDARI SHIVAMANI (218R1A1208) ARIKELA NANDINI (218R1A1204) BATHINI JAHNAVI (218R1A1210)**

**ULLENGULA SAI TEJA (218R1A1215)**

# CONTENTS

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO** |
| **ABSTRACT** | **I** |
| **LIST OF FIGURES** | **II** |
| **LIST OF TABLES** | **III** |
| **1. INTRODUCTION** | **1** |
| 1.1. Introduction | 1 |
| 1.2. Project Objectives | 3 |
| 1.3. Purpose of the project | 3 |
| 1.4. Need | 3 |
| 1.5. Significance | 4 |
| 1.6. Existing System with Disadvantages | 4 |
| 1.7. Proposed System With Advantages | 7 |
| 1.8. Input and Output Design | 9 |
| **2. LITERATURE SURVEY** | **14** |
| **3. SOFTWARE REQUIREMENT ANALYSIS** | **19** |
| 3.1. Problem Specification | 19 |
| 3.2. Modules and their Functionalities | 19 |
| 3.3. Functional Requirements | 24 |
| 3.4. Non-Functional Requirements | 24 |
| 3.5. Feasibility Study | 25 |
| **4. SOFTWARE & HARDWARE REQUIREMENTS** | **26** |
| 4.1. Software requirements | 26 |
| 4.2. Hardware requirements | 26 |
| **5. SOFTWARE DESIGN** | **27** |
| 5.1. System Architecture | 27 |
| 5.2. Dataflow Diagrams | 27 |
| 5.3. UML Diagrams | 29 |

|  |  |  |
| --- | --- | --- |
| **6. CODING AND IMPLEMENTATION** | | **34** |
| 6.1. Source code | | 34 |
| 6.2. Implementation | | 40 |
| **7. SYSTEM TESTING** | | **64** |
| 7.1. | Types of System Testing | 64 |
| 7.2. | Test Cases | 65 |
| **8. OUTPUT SCREENS** | | **66** |
| **9. CONCLUSION** | | **72** |
| **10. FUTURE ENHANCEMENTS** | | **73** |
| **11. REFERENCES** | | **74** |

**ABSTRACT**

In today's digital landscape, secure and efficient file-sharing mechanisms are essential to protect data integrity and prevent unauthorized access. Traditional cloud-based file-sharing systems suffer from centralization risks, data breaches, and high operational costs, making them vulnerable to cyberattacks and censorship. To address these challenges, this research proposes a Decentralized Secure File Sharing System using Blockchain and IPFS. The system integrates Ethereum smart contracts to manage file access permissions and authentication, ensuring a tamper-proof and transparent process. Additionally, IPFS (InterPlanetary File System) provides decentralized storage, enabling fault tolerance, data integrity, and enhanced accessibility. Users authenticate via Web3 wallets like MetaMask, eliminating the need for conventional password-based authentication, further strengthening security. By leveraging blockchain’s immutability and IPFS’s distributed storage capabilities, this approach minimizes reliance on third-party intermediaries, enhances privacy, and ensures a scalable, trustless environment for file sharing. This documentation presents the system’s architecture, implementation methodology, security mechanisms, and performance evaluation, demonstrating its effectiveness as a reliable and censorship-resistant alternative to traditional cloud-based file-sharing platforms.

**Keywords –** Decentralized File Sharing, Blockchain Security, IPFS Storage, Smart Contracts, Web3 Authentication, Data Integrity.

# LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIGURE NO** | **DESCRIPTION** | **PAGENO** |
| 1 | 1.7.1 | Proposed methodology of ML-based  energy consumption prediction in smart homes. | **7** |
| 2 | 3.2.1 | Importing Libraries | 20 |
| 3 | 3.2.2 | Feature scaling | 21 |
| 4 | 3.2.3 | Splitting the dataset | 21 |
| 5 | 3.2.4 | Working of RFR model. | 23 |
| 6 | 5.1.1 | System Architecture | 27 |
| 7 | 5.2.1 | Dataflow diagram | 28 |
| 8 | 5.3.1 | Sequence diagram | 30 |
| 9 | 5.3.2 | Use case diagram | 31 |
| 10 | 5.3.3 | Class diagram | 32 |
| 11 | 5.3.4 | Activity diagram | 33 |
| 12 | 6.2.1 | Downloading Python | 49 |
| 13 | 6.2.2 | Selecting Latest Version | 50 |
| 14 | 6.2.3 | Python versions | 50 |
| 15 | 6.2.4 | Selecting windows version | 51 |
| 18 | 6.2.5 | Installing Python | 51 |
| 19 | 6.2.6 | Adding Path | 52 |
| 20 | 6.2.7 | Installation Successful | 52 |
| 21 | 6.2.8 | Verifying installation | 53 |
| 22 | 6.2.9 | Checking installation | 53 |
| 23 | 6.2.10 | Launching Program | 54 |
| 24 | 6.2.11 | Saving files | 54 |
| 25 | 6.2.12 | Example program | 54 |
| 26 | 8.1 | Presents the GUI of Smart Homes. | 66 |
| 27 | 8.2 | Presents the Data Preprocessing and R2 Score of Linear Regression model. | 67 |
| 28 | 8.3 | Presents the Linear Regression Model Prediction on Test data. | 67 |
| 29 | 8.4 | Presents the R2 Score of Decision  Tree Regression model. | 68 |

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | 8.5 | Presents Plot of Decision Tree  Regression Model Prediction on Test data. | 69 |
| 31 | 8.6 | Presents the R2 Score of Random  Forest Regression model. | 69 |
| 32 | 8.7 | Presents Plot of Random Forest Regression Model Prediction on Test data. | 69 |
| 33 | 8.8 | Comparison Graph of each model Performance. | 70 |
| 34 | 8.9 | Proposed Model Prediction on test data. | 71 |
|  |  | **LIST OF TABLES** |  |
| 1 | 7.2 | Test cases | 65 |

1. **INTRODUCTION**

## Introduction

The ability to securely share files across networks is essential in modern digital interactions. Traditional file-sharing systems, such as cloud-based solutions, rely on centralized servers that introduce several challenges, including security risks, data breaches, unauthorized access, and dependency on third-party service providers. Additionally, these systems are prone to censorship and high operational costs, making them less reliable and more susceptible to cyberattacks. The increasing demand for privacy, data integrity, and decentralization has led to the exploration of blockchain-based file-sharing solutions.

This project presents a Decentralized Secure File Sharing System using Blockchain and IPFS, which overcomes the limitations of traditional file-sharing mechanisms by leveraging blockchain technology for access control and IPFS (InterPlanetary File System) for decentralized storage. The integration of Ethereum smart contracts ensures that access permissions are managed transparently, preventing unauthorized access and ensuring that stored files remain tamper-proof. Since blockchain transactions are immutable, any access or modification attempts are recorded permanently, providing users with full visibility into file-sharing activities.

Centralized file storage systems have several disadvantages. A single point of failure exists because if a centralized server goes down, users lose access to their files. Security vulnerabilities arise as cloud storage services are often targeted by hackers, leading to data breaches and leaks. High operational costs are another concern, as storing large amounts of data on centralized cloud services incurs significant expenses. Additionally, censorship risks exist because governments or organizations may impose restrictions on certain files, limiting access.

Decentralized file-sharing addresses these challenges by removing dependency on central authorities and distributing data across multiple nodes, ensuring availability, security, and transparency. By integrating IPFS for decentralized storage and blockchain for access control, this system eliminates vulnerabilities associated with centralized storage models. It ensures that files remain accessible even if some network nodes go offline, reducing risks related to downtime and centralized control.

This decentralized file-sharing system operates on three main components. The blockchain layer, using Ethereum smart contracts, governs access control and authentication, manages file permissions, and ensures the immutability and transparency of file transactions. The storage layer, using IPFS, stores encrypted files in a decentralized manner, ensuring data availability even if individual nodes go offline while providing content-addressable storage for quick file retrieval. The application layer consists of a frontend developed using React.js for a user-friendly interface, and a backend using Node.js and Express.js to handle authentication and interactions with smart contracts. A MySQL database securely stores user credentials with bcrypt encryption. The system ensures high security and privacy through multiple mechanisms. Blockchain immutability guarantees that file-sharing records cannot be altered or deleted. End-to-end encryption ensures that files are encrypted before being stored on IPFS, preventing unauthorized access. Web3 authentication allows users to authenticate with MetaMask, eliminating the need for traditional passwords and reducing the risks associated with password-based attacks. Smart contract-based access control ensures that users can precisely determine who has access to their files, using blockchain-based permission management.

This decentralized approach offers several key advantages over traditional file-sharing solutions. Increased security is achieved by preventing unauthorized modifications and ensuring data integrity through blockchain technology. Cost efficiency is enhanced as users avoid high cloud storage costs by utilizing IPFS’s distributed storage model. Censorship resistance is ensured as decentralized storage prevents any single entity from controlling or blocking access to files. Enhanced user control allows individuals to maintain full ownership of their files, deciding who can access them without relying on a central authority. Additionally, the system is designed to handle large-scale file sharing without performance degradation, making it highly scalable.

Cloud-based file-sharing services are widely used due to their accessibility, but they pose risks like centralized control, data breaches, and privacy concerns. Blockchain ensures verifiable, tamper-resistant access control, while IPFS provides distributed storage, eliminating reliance on centralized servers.By integrating these technologies, this research explores a secure file-sharing system that enhances user autonomy, prevents unauthorized access, and maintains data integrity. The proposed model reduces dependency on intermediaries and offers a censorship-resistant method for sharing files securely. Through blockchain-based authentication and smart contracts, this decentralized approach ensures privacy, resilience, and efficient file distribution. This study examines how blockchain and IPFS redefine file sharing, addressing limitations of traditional cloud storage.

## Project Objectives

This project aims to develop a decentralized and secure file-sharing system using blockchain and IPFS, eliminating risks of centralized storage. Ethereum smart contracts manage access permissions transparently, ensuring data integrity and security. Web3 authentication via MetaMask replaces traditional passwords, enhancing security. IPFS enables decentralized storage, reducing reliance on central servers and preventing failures. The system enhances scalability and cost-efficiency while ensuring censorship resistance. By integrating blockchain and distributed storage, this project provides a reliable, transparent, and trustless file-sharing solution, empowering users with full ownership and control while addressing limitations of conventional cloud storage models.

## Purpose of the Project

The purpose of this project is to create a highly secure and efficient file-sharing system that overcomes the limitations of traditional centralized storage solutions. In conventional cloud storage, data is managed by third-party providers, leading to concerns related to security, privacy, and control. This project aims to eliminate these concerns by implementing a decentralized architecture that allows users to store and share files without relying on intermediaries. Blockchain technology ensures that access control mechanisms are transparent and tamper-proof, while IPFS guarantees that files remain accessible even in the event of server failures. Another key purpose is to enhance user authentication through Web3 login, eliminating the vulnerabilities of traditional username-password mechanisms. The system also provides an immutable record of file-sharing activities, increasing transparency and trust among users. By addressing issues such as data breaches, censorship, and high operational costs, this project offers a practical and scalable alternative for individuals and organizations looking for secure file-sharing solutions.

## Need

The need for a decentralized secure file-sharing system arises from concerns over data privacy, security risks, and reliance on centralized providers. Traditional cloud storage is vulnerable to hacking, unauthorized access, and data breaches. Centralized systems also face single points of failure and censorship risks. By leveraging blockchain and IPFS, this project ensures secure, transparent, and censorship-resistant file-sharing. Data is distributed across multiple nodes, enhancing resilience. Smart contracts manage access permissions, giving users full control. With rising cyber threats, this decentralized approach is crucial for secure file-sharing across industries.

## Significance

This project is significant as it revolutionizes file sharing by leveraging blockchain and decentralized storage. Unlike traditional systems reliant on third-party providers, it grants users complete control over files and access permissions. Ethereum smart contracts enhance security by preventing unauthorized modifications and ensuring immutable records. IPFS enables cost-effective, scalable storage while reducing dependency on centralized servers. Web3 authentication strengthens security by replacing traditional logins. Additionally, this censorship-resistant system promotes digital freedom, addressing modern challenges in data privacy and security for individuals and industries.

## Existing System with Disadvantages Centralized File-Sharing Systems

Traditional file-sharing systems rely on centralized cloud storage providers such as Google Drive, Dropbox, and OneDrive. These platforms store user files on dedicated servers managed by a single entity. Users upload their files to a centralized server, where access is controlled through permissions set by the provider. While these systems offer convenience, they come with significant security, privacy, and reliability challenges.

## Functionality of Centralized File Storage

Step 1: Users upload files to a cloud server.

Step 2: The server stores the files and maintains metadata such as owner details, access permissions, and timestamps.

Step 3: Users access their files through authentication mechanisms such as email and passwords.

Step 4: File sharing is controlled by the provider, allowing users to grant or revoke access to specific people.

**Limitations of the Centralized System**

* **Trust Dependency:** Users must trust a third party to manage their files securely.
* **Data Breach Risks:** Centralized storage is prone to hacking attempts and security breaches.
* **Limited Data Ownership**: Users do not have full control over their files, as service providers regulate access**.**
* **Censorship Issues:** Governments or organizations can impose content restrictions or remove files.
* **High Costs:** Large-scale file storage and premium security features require costly subscriptions.

## Disadvantages

The existing centralized file-sharing systems come with several critical drawbacks. A single point of failure means that server crashes or cyberattacks can lead to data loss or inaccessibility. Security vulnerabilities make these platforms prime targets for hackers, risking sensitive information. Users also face privacy concerns as companies may access, analyze, or share their data. Censorship and content restrictions limit user freedom, while high storage costs make these services expensive. Additionally, the lack of transparent access control leaves users uncertain about who can view or modify their files, increasing security risks:

* + **Single Point of Failure**: Centralized systems depend on a single server or data center for storing and managing user files. If the server experiences technical failures, cyberattacks, or power outages, users may lose access to their files. In extreme cases, server crashes or data corruption could lead to permanent loss of data.
  + **Security Vulnerabilities**: Since all data is stored on a single central entity, these platforms become prime targets for cybercriminals. Hackers can exploit vulnerabilities to gain unauthorized access, steal sensitive information, or inject malware into cloud systems. High-profile data breaches have exposed millions of users’ private information, leading to severe financial and reputational damages.
  + **Unauthorized Access and Privacy Concerns**: Users entrust their personal and confidential data to cloud providers without full transparency over how their information is handled. In many cases, companies have access to stored files and may scan, analyze, or share data for advertising and business purposes. Additionally, government agencies can demand access to user files, raising concerns over privacy and surveillance.
  + **Censorship and Content Restrictions**: Cloud storage providers must comply with government regulations and content policies, leading to censorship and restricted access. If authorities or companies decide that certain content violates terms of service, they can delete files or block access without user consent. This creates challenges for individuals relying on these platforms for unrestricted access to information.
* **High Storage and Subscription Costs:** Most cloud services provide limited free storage, requiring users to pay for additional space. Storing large volumes of data or using advanced security features can lead to high operational costs. Moreover, businesses that require high-performance cloud storage must invest in expensive enterprise solutions, making it an impractical option for many users.

## Proposed System

The proposed system leverages blockchain technology and the InterPlanetary File System (IPFS) to create a secure, transparent, and decentralized file-sharing platform. Unlike traditional centralized cloud storage, this system eliminates the reliance on a single authority, ensuring better data privacy, security, and control. By integrating Ethereum smart contracts and IPFS, the platform allows users to store, share, and manage files securely while maintaining ownership and access control.

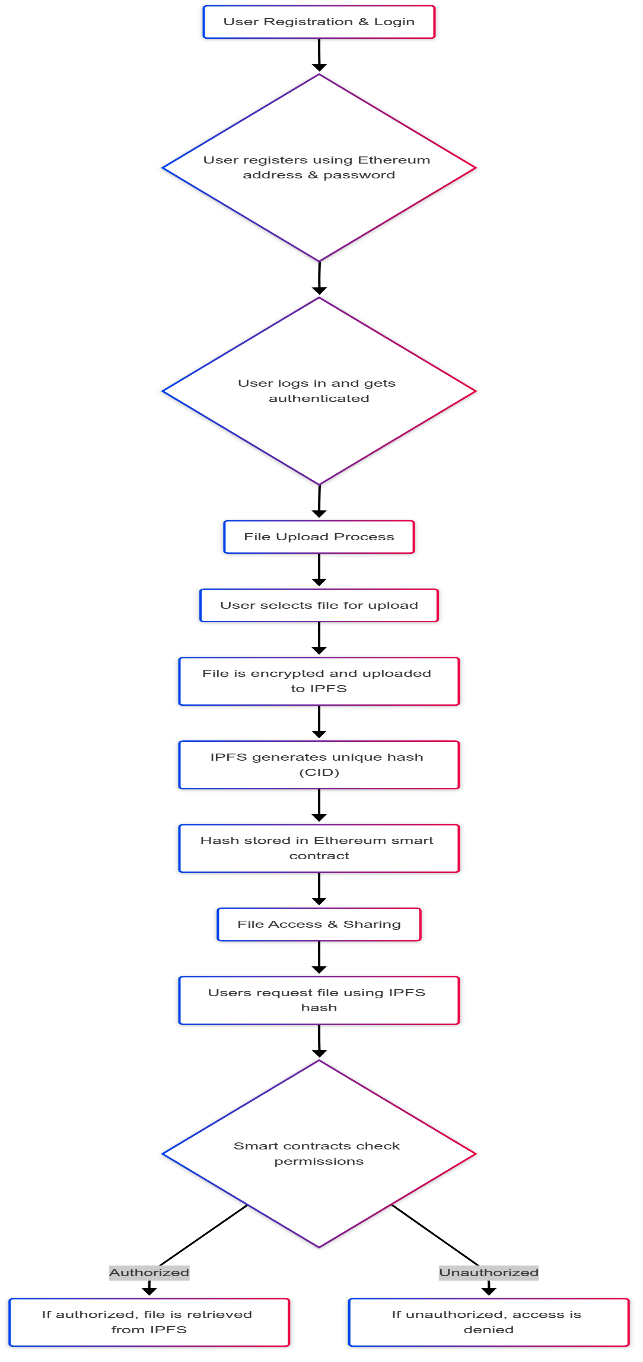


Figure 1.7.1: Proposed methodology of ML-based energy consumption prediction in smart homes.

## Advantages

The proposed system overcomes the limitations of centralized storage by leveraging blockchain and IPFS for secure, decentralized file sharing. By distributing data across multiple nodes, it eliminates single points of failure and ensures files remain accessible even during server outages. Enhanced security is achieved through encrypted file storage and smart contracts that automate access control, preventing unauthorized modifications. The system guarantees data integrity by storing file hashes on the blockchain, making tampering easily detectable. Additionally, it is cost-efficient, eliminating subscription fees, while providing full transparency and ownership control, making it a robust alternative to traditional cloud storage.

* + - **Decentralization & Elimination of Single Point of Failure:** Unlike centralized storage, the proposed system distributes file data across multiple nodes in IPFS, ensuring that files remain accessible even if individual nodes fail. This eliminates the risk of data loss due to server failures, a common issue in traditional cloud storage. Additionally, blockchain technology ensures that no central authority can manipulate or delete user files, providing a trustless and censorship-resistant file-sharing environment.
    - **Enhanced Security & Privacy:** Security is significantly enhanced by storing files as encrypted hashes, reducing the risk of hacking or unauthorized access. Since file access is managed by Ethereum smart contracts, only authorized users can view or download files. This eliminates reliance on third-party service providers and minimizes risks associated with data breaches, unauthorized modifications, or privacy violations.
    - **Data Integrity & Immutability:** The system ensures data integrity by storing file hashes on the blockchain, making any modification or tampering easily detectable. Unlike traditional cloud storage, where data can be altered without user knowledge, blockchain immutability guarantees that files remain unchanged once uploaded. Users can verify the authenticity of their files without needing to trust any central authority, ensuring a transparent and secure system.
    - **Cost-Efficiency & No Subscription Fees:** Unlike traditional cloud services that require costly subscription fees for storage and access, the proposed system leverages IPFS to distribute storage across multiple peers. This eliminates recurring costs and makes file sharing more affordable. Users only pay minimal transaction fees (gas fees) when interacting with the blockchain, ensuring a cost-effective solution without compromising security or accessibility.
    - **Transparency & Ownership Control:** The blockchain ledger maintains a clear and immutable history of all file transactions, ensuring complete transparency. Users have full ownership and control over their files, with the ability to grant or revoke access at any time. This level of control enhances privacy and security, empowering users with a decentralized file-sharing solution that eliminates reliance on centralized authorities.
  1. **INPUT AND OUTPUT DESIGN**

#### INPUT DESIGN

The input design serves as the interface between users and the Decentralized Secure File Sharing System. It ensures efficient and secure data entry, focusing on minimizing errors, reducing processing delays, and maintaining privacy. In this project, users will interact with the system through a web-based interface to upload, retrieve, and manage files securely on the blockchain.

Key considerations for input design include:

* + - What data should be given as input: Users provide files for upload, along with metadata such as file name, description, and access permissions.
    - How should the data be structured: Files are converted into hashes and stored on IPFS, with the corresponding hash recorded on the blockchain.
    - User guidance for input: The system provides an intuitive UI for file selection, permission settings, and blockchain-based authentication.
    - Error handling and validation: Input validation ensures only supported file formats and valid Ethereum addresses are accepted.

#### OBJECTIVES

* + - Ensure a user-friendly, efficient, and error-free input process for file uploads and access permissions.
    - Provide a secure mechanism to store and retrieve files via blockchain and IPFS.
    - Implement validation and authentication mechanisms to prevent unauthorized access and incorrect data entries.
    - Maintain decentralization while ensuring seamless user interaction.

#### Input Stages

The main input stages can be listed as below:

* **User Authentication** – Users log in using their Ethereum address and password stored in MySQL.
* **File Upload** – Users select a file, provide metadata, and set access permissions.
* **IPFS Storage** – The file is uploaded to IPFS, and its hash is generated.
* **Blockchain Recording** – The file hash and access permissions are stored on the Ethereum blockchain
* **File Access & Retrieval** – Users retrieve files by querying the blockchain and fetching data from IPFS.
* **Permission Management** – Users can update sharing settings and grant/revoke access through smart contract interactions.

#### Input Types

* User Inputs: File uploads, descriptions, access permissions, Ethereum address.
* System Inputs: File hash generation, smart contract interactions, access logs.

#### Input Media

#### Web-based UI for direct user interactions.

#### Blockchain transactions for secure recording of file metadata.

#### IPFS storage for decentralized file handling.

#### MySQL database for user authentication and account management.

Till here completed……………………

#### Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

#### Error Detection

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

#### Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

#### User Interface Design

It is essential to consult the system users and discuss their needs while designing the user interface:

#### User Interface Systems Can Be Broadly Classified As:

* + - * User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
      * Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

#### User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

* + - * Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
      * Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

#### Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

* + - * The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
      * Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

#### Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

#### Performance Requirements

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* + - * The system should be able to interface with the existing system
      * The system should be accurate
      * The system should be better than the existing system
      * The existing system is completely dependent on the user to perform all the duties.

#### OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* + - External Outputs, whose destination is outside the organization
    - Internal Outputs whose destination is within organization and they are the
    - User’s main interface with the computer.
    - Operational outputs whose use is purely within the computer department.
    - Interface outputs, which involve the user in communicating directly.

#### Output Definition

The outputs should be defined in terms of the following points:

* + - Type of the output
    - Content of the output
    - Format of the output
    - Location of the output
    - Frequency of the output
    - Volume of the output
    - Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

# LITERATURE SURVEY

## Title: Changes in hourly electricity consumption under COVID mandates: a glance to future hourly residential power consumption pattern with remote work in Arizona.

### **Author:** Ku, A. L., Qiu, Y., Lou, J., Nock, D., and Xing, B

Changes in hourly electricity consumption under COVID-19 mandates, particularly focusing on future residential power consumption patterns with the prevalence of remote work in Arizona, reveals a multifaceted landscape of factors. Existing research underscores the potential for shifts in peak consumption hours, increased daytime usage due to remote work activities, and alterations in weekend consumption habits.

**Title:The diverse impacts of COVID-19 on electricity demand: the case of Chile**. **Author:** Sánchez-López, M., Moreno, R., Alvarado, D.

Diverse impacts of COVID-19 on electricity demand, focusing specifically on the case of Chile, as documented in the International Journal of Electrical Power & Energy Systems, illuminates a range of nuanced effects. Research indicates that the pandemic induced significant fluctuations in electricity demand patterns, influenced by various factors such as government-imposed lockdown measures, changes in commercial and industrial activities, and shifts in consumer behavior.

## Title: Empirical grid impact of in-home electric vehicle charging differs from predictions.

### **Author:** Qiu, Y. L., Wang, Y. D., Iseki, H., Shen, X., Xing, B., and Zhang, H.

Empirical grid impact of in-home electric vehicle (EV) charging, as explored in the journal Resource and Energy Economics, reveals significant disparities between theoretical predictions and observed outcomes. Research in this domain highlights the complexities inherent in modeling the interaction between EV charging behaviors and grid infrastructure.

## Title:Impact of battery storage on residential energy consumption: an Australian case study based on smart meter data.

### **Author:** Al Khafaf, N., Rezaei, A. A., Moradi Amani, A.

Impact of battery storage on residential energy consumption, focusing on an Australian case study utilizing smart meter data, reveals significant benefits and insights. Studies indicate that residential battery storage systems can substantially reduce energy costs, peak demand, and grid reliance.

## Title:Heterogeneous changes in electricity consumption patterns of residential distributed solar consumers due to battery storage adoption.

### **Author:** Qiu, Y., Xing, B., Patwardhan, A., Hultman, N., and Zhang, H.

Heterogeneous changes in electricity consumption patterns among residential distributed solar consumers resulting from battery storage adoption highlights significant findings. Research indicates that the integration of battery storage systems alongside residential solar installations leads to diverse impacts on consumption behaviors.

## Title: Impacts of electric-driven heat pumps on residential electricity consumption: an empirical analysis from Arizona, USA. Clean.

### **Author:** Liang, J., Qiu, Y., and Xing, B.

Electric-driven heat pumps in Arizona highlights their potential to reduce residential electricity consumption while promoting clean and responsible consumption. Empirical evidence suggests significant energy savings compared to traditional systems, aligning with sustainability objectives. This analysis underscores the role of heat pumps in advancing cleaner energy adoption and informing policy decisions for a more sustainable future in residential heating and cooling.

## Title: Impacts of the co-adoption of electric vehicles and solar panel systems: empirical evidence of changes in electricity demand and consumer behaviors from household smart meter data.

### **Author:** Liang, J., Qiu, Y., and Xing, B.

Co-adoption of electric vehicles and solar panel systems explores their impact on electricity demand and consumer behavior using household smart meter data. This analysis underscores the potential for synergies between electric vehicles and solar panels to reduce carbon footprint and reshape energy usage behaviors.

## Title: Benchmarking of load forecasting methods using residential smart meter data.

**Author:** Sousa, J. C., and Bernardo, H.

Load forecasting methods utilizes residential smart meter data for benchmarking purposes. It evaluates various forecasting techniques' accuracy and efficiency in predicting household electricity demand. This analysis provides insights into the most effective methods for improving load forecasting accuracy, aiding in efficient energy resource allocation and grid management.

## Title: Data-driven based prediction of the energy consumption of residential buildings in Oshawa.

### **Author:** Yildiz, B., Bilbao, J. I., Dore, J., and Sproul

Data-driven prediction of residential building energy consumption in Oshawa reviews methodologies and models utilizing empirical data. It assesses the effectiveness of predictive algorithms in accurately estimating energy usage patterns. This research informs strategies for optimizing energy efficiency and promoting sustainable practices in residential settings.

## Title: COVID-19 pandemic ramifications on residential smart homes energy use load profiles.

### **Author:** Chinthavali, S., Tansakul, V., Lee, S., Whitehead, M., Tabassum, A., Bhandari, M., et al.

COVID-19 pandemic effects on residential smart homes examines changes in energy load profiles due to altered occupancy patterns and remote work. It assesses how pandemic-related behaviors impact energy consumption and efficiency in smart home environments. This research provides insights for adapting energy management strategies and infrastructure to evolving lifestyle norms.

## Title: Global sustainability, innovation and governance dynamics of nationalsmart electricity meter transitions.

**Author:** Sovacool, B. K., Hook, A., Sareen, S., and Geels

Global sustainability and innovation dynamics in national smart electricity meter transitions evaluates governance structures and innovation pathways. It examines the impact of smart meter adoption on energy efficiency, grid management, and sustainability goals worldwide. This research informs policy frameworks and technological advancements for effective smart meter deployment and sustainable energy transitions.

## Title: Electricity consumption of Singaporean households reveals proactive community response to COVID-19 progression.

**Author:** Raman, G., and Peng, J. C.-H.

Electricity consumption of Singaporean households during COVID-19 highlights a proactive community response amidst the pandemic. It explores shifts in energy usage patterns and initiatives to promote efficiency and resilience during varying stages of the outbreak. This research provides insights for adaptive energy management strategies and community engagement in times of crisis.

## Title: Impacts of COVID-19 related stay-at-home restrictions on residential electricity use and implications for future grid stability.

### **Author:** Li, L., Meinrenken, C. J., Modi, V., and Culligan, P. J.

COVID-19-related stay-at-home restrictions assesses their impact on residential electricity consumption and grid stability. It examines shifts in demand patterns, the effectiveness of demand-side management strategies, and implications for future grid resilience. This research informs proactive measures for adapting energy infrastructure to evolving societal dynamics and potential future disruptions.

## Title: Inequitable and heterogeneous impacts on electricity consumption from COVID-19 mitigation measures.

### **Author:** Lou, J., Qiu, Y., Ku, A. L., Nock, D., and Xing

Inequitable impacts of COVID-19 mitigation measures examines disparities in electricity consumption across socioeconomic groups. It analyzes how factors such as income, housing type, and location influence vulnerability to changes in energy use patterns. This research underscores the need for targeted policies to address disparities and ensure equitable access to energy during crises.

**Title: Cluster analysis and model comparison using smart meter data. Author:** Shaukat, M. A., Shaukat, H. R., Qadir, Z., Munawar, H. S., Kouzani, A. Z., and Mahmud, M. A. P.

Cluster analysis and model comparison utilizing smart meter data evaluates methodologies for grouping electricity consumption patterns and comparing predictive models. It explores the effectiveness of clustering techniques and model evaluation metrics in enhancing energy forecasting accuracy. This research informs decision-making processes for optimizing energy management strategies based on data-driven insights.

## Title: Deep learning for load forecasting with smart meter data: online adaptive recurrent neural network.

### **Author:** Fekri, M. N., Patel, H., Grolinger, K., and Sharma, V.

Deep learning for load forecasting with smart meter data focuses on the development and application of online adaptive recurrent neural networks. It evaluates the effectiveness of these models in accurately predicting electricity demand patterns in real-time scenarios. This research informs advancements in load forecasting techniques, enhancing energy efficiency and grid management capabilities.

**Title: Smart Home and Appliances: State of the art. Author:** Ribeiro Serrenh, T, and Bertoldi, P.

smart home and appliances provides an overview of the current state-of-the-art technologies and advancements in the field. It examines innovative solutions, integration challenges, and potential impacts on energy efficiency and consumer lifestyles. This research informs future development directions and policy considerations for the smart home ecosystem.

## Title: Big data mining of energy time series for behavioral analytics and energy consumption forecasting

**Author:** Singh, S., and Yassine, A.

Big data mining of energy time series explores methodologies for behavioral analytics and energy consumption forecasting. It assesses the effectiveness of data-driven approaches in extracting insights from large-scale energy datasets to inform predictive models. This research informs strategies for optimizing energy management and promoting efficiency in various sectors.

## Title: Recent advances in the analysis of residential electricity consumption and applications of smart meter data

### **Author:** Yildiz, B., Bilbao, J. I., Dore, J., and Sproul, A. B.

Recent advances in analyzing residential electricity consumption delves into the applications of smart meter data. It evaluates emerging techniques for extracting insights, forecasting consumption patterns, and optimizing energy use. This research informs strategies for enhancing efficiency and sustainability in residential energy management.

# SOFTWARE REQUREIMENTS ANALYSIS

## Problem Specification

Understanding the relationship between weather patterns and energy consumption is crucial in the context of smart homes and sustainable energy usage. With the rising importance of smart home technology and the increasing focus on energy efficiency, it is imperative to analyze the factors that influence energy consumption. By deciphering the impact of weather variables such as temperature, humidity, and precipitation on energy load, this study addresses a fundamental need in optimizing energy usage in smart homes. These findings are invaluable for both homeowners and energy providers, enabling them to make informed decisions to enhance energy efficiency and reduce costs.

## 2 Modules and Their Functionalities Data Preprocessing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

* + Getting the dataset
  + Importing libraries
  + Importing datasets
  + Finding Missing Data
  + Encoding Categorical Data
  + Splitting dataset into training and test set

**Importing Libraries:** To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

**Numpy**: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add

large, multidimensional arrays and matrices. So, in Python, we can import it as: import numpy as nm

Here we have used nm, which is a short name for Numpy, and it will be used in the whole program. **Matplotlib:** The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. It will be imported as below:

import matplotlib.pyplot as mpt

Here we have used mpt as a short name for this library.

**Pandas:** The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. Here, we have used pd as a short name for this library. Consider the below image:

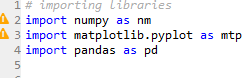


Figure 3.2.1:importing Libraries

**Handling Missing data:** The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset. There are mainly two ways to handle missing data, which are:

* + By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not soefficient and removing data may lead to loss of information which will not give the accurate output.
  + By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

**Encoding Categorical data:** Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased. Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers.

**Feature Scaling:** Feature scaling is the final step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature

scaling, we put our variables in the same range and in the same scale so that no variable dominates the other variable. A machine learning model is based on Euclidean distance, and if we do not scale the variable, then it will cause some issue in our machine learning model. Euclidean distance is given as:

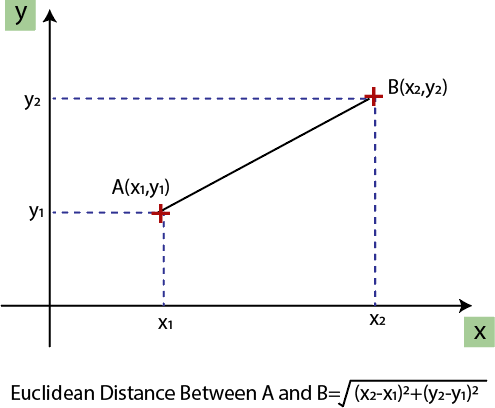


Figure 3.2.2: Feature scaling

If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So, to remove this issue, we need to perform feature scaling for machine learning.

## Splitting the Dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

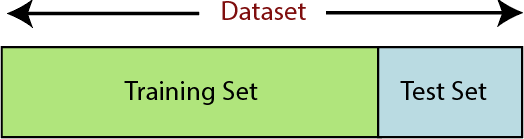


Figure 3.2.3: Splitting the dataset.

**Training Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.2, random\_state=0)

#### Explanation

* + In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
  + In the second line, we have used four variables for our output that are
  + x\_train: features for the training data
  + x\_test: features for testing data
  + y\_train: Dependent variables for training data
  + y\_test: Independent variable for testing data
  + In train\_test\_split() function, we have passed four parameters in which first two are for arrays of data, and test\_size is for specifying the size of the test set. The test\_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
  + The last parameter random\_state is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

## Random Forest Regressor

Random Forest is an ensemble learning method, meaning it combines the predictions of multiple individual algorithms (in this case, decision trees) to create a more accurate and robust model. In the case of regression tasks, where the goal is to predict a continuous numerical value (like energy consumption), the algorithm is called a Random Forest Regressor. Overall, the Random Forest Regressor plays a crucial role in accurately predicting energy consumption in smart homes based on weather patterns. Its ability to handle non-linearity, provide feature importance insights, and maintain robustness makes it a suitable choice for this complex predictive task.

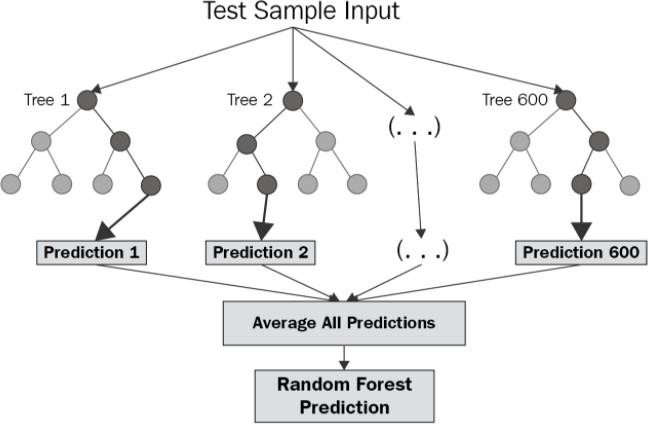


Figure 3.2.4: Working of RFR model.

## Working:

* **Decision Trees**: Random Forest starts by creating a multitude of decision trees. Each tree is trained on a random subset of the data and a random subset of features. This randomness helps the trees to be diverse and not overly reliant on a specific subset of the data.
* **Voting Mechanism**: When it's time to make a prediction, each individual tree in the forest produces its own prediction. In the case of regression, the predictions from each tree are averaged to produce the final output. This averaging process results in a more accurate and stable prediction than relying on any single decision tree.
* **Handling Complexity**: Random Forests are powerful because they can handle a large number of features and complex relationships between features and the target variable. Each tree, being a part of the forest, contributes its understanding of these relationships. When combined, they provide a comprehensive view of how different weather variables affect energy consumption.

## Advantages:

* **Handling Non-Linearity:** Random Forests excel at capturing non-linear relationships in data. Weather patterns and energy consumption are highly likely to have non-linear correlations. Traditional linear models might miss these complexities, but Random Forests can capture intricate patterns, leading to more accurate predictions.
* **Feature Importance:** Random Forests can provide insights into feature importance. In the context of this project, this means understanding which weather variables (temperature, humidity, etc.) have the most significant impact on energy consumption. This information is invaluable for homeowners and policymakers to focus on the most influential factors.
* **Robustness:** Random Forests are robust against overfitting, a common problem in machine learning where a model performs well on the training data but fails to generalize to new, unseen data. The randomization and averaging mechanisms in Random Forests help mitigate overfitting, ensuring the model's reliability.

## Functional Requirements

It provides the users a clear statement of the functions required for the system in order to solve the project information problem it contains a complete set of requirements for the applications. A requirement is condition that the application must meet for the customer to find the application satisfactory. A requirement has the following characteristics:

* + - It provides a benefit to the origination.
    - It describes the capabilities the application must provide in business terms.
    - It does not describe how the application provides that capability.
    - It is stated in unambiguous words. Its meaning is clear and understandable.
    - It is verifiable.

## Non-Functional Requirements

Career recommendation non-functional requirements, like interests he has, how hours he can work likewise, with today’s IT projects, to determine non-functional requirements, like availability, the approach requires that the designer 1 st determine the scope: does the whole solution or only part of it need to be architected to meet minimum levels?

* + - This is done through 4 steps:
    - Identify the critical areas of solutions
    - Identify the critical components within each critical area.
    - Determine each components availability and risk.
    - Model worst-case failure scenarios.

## Feasibility Study

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

* + - ECONOMICAL FEASIBILITY
    - TECHNICAL FEASIBILITY
    - OPERATIONAL FEASIBILITY

#### Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

#### Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

#### Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following:-

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits? This system is targeted to be in accordance with the above-mentioned issues.

# SOFTWARE AND HARDWARE REQUIREMENTS

## Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* + - Python IDLE 3.7 version
    - Anaconda 3.7
    - Jupiter
    - Google colab

## Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* + - Operating system :Windows, Linux
    - Processor : minimum intel i3
    - Ram : minimum 4 GB
    - Hard disk : minimum 250GB

# SOFTWARE DESIGN

## System Architecture

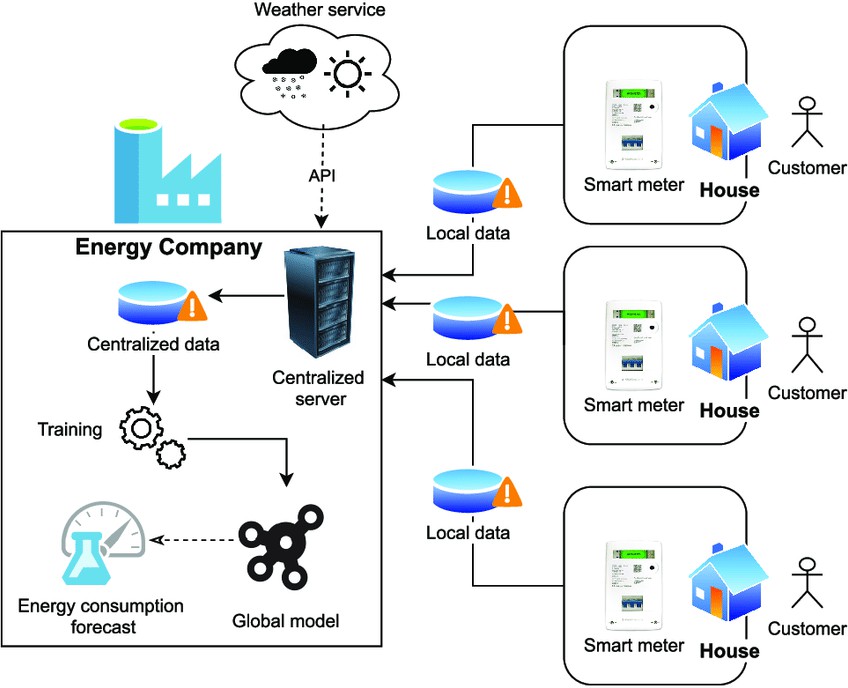
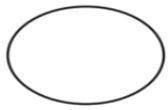
****

Figure:5.1.1 System Architecture

## Dataflow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flowgraph or bubble chart. The Basic Notation used to create a DFD’s are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.
2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.
3. Source: External sources or destination of data, which may be People, programs.



1. Data Store: Here data are stored or reference by a process in the System



Dataset

Dataset(selected features)

Feature selection

Prediction Model

Scrape feature data from URL

URL input

Result(correct sign text visible or not)

Figure: 5.2.1 Dataflow diagram

## UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non- software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

The different types are as follows:

* 1. Class Diagram
  2. Use case Diagram
  3. Sequence diagram
  4. Activity diagram

#### CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

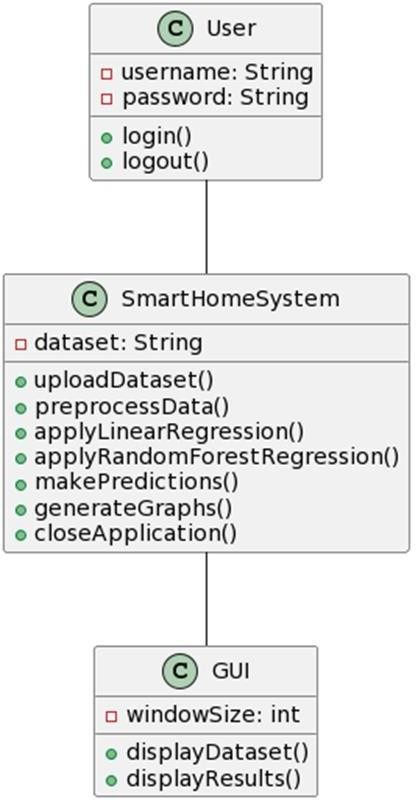


Figure 5.3.1 class diagram

#### USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

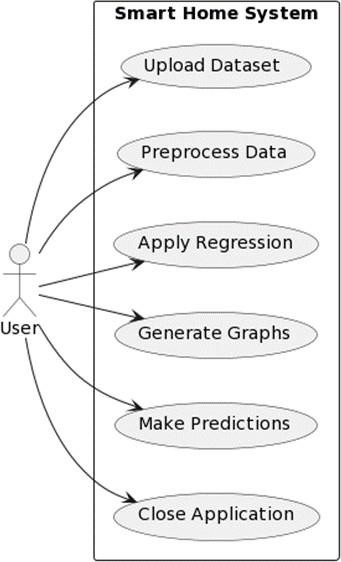


Figure 5.3.2 use case diagram

#### SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

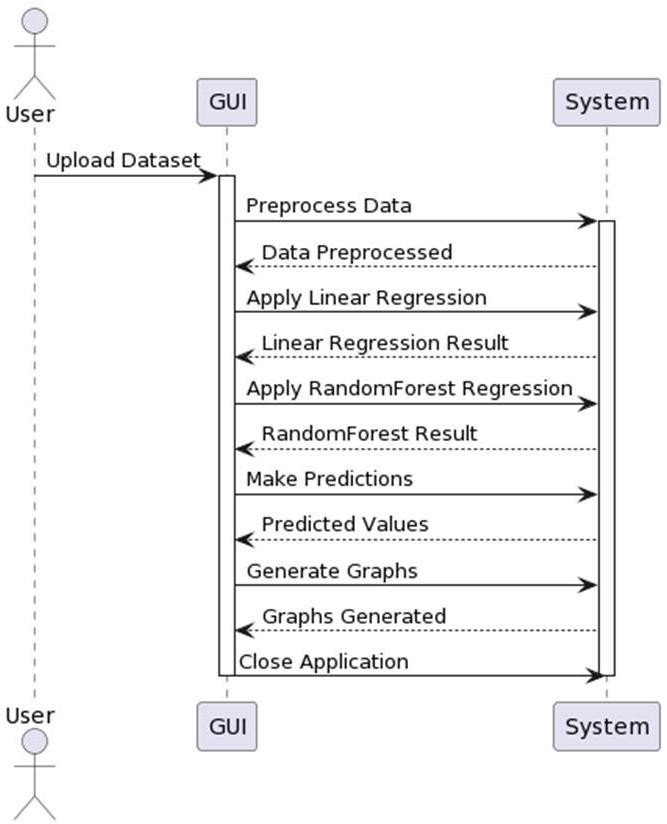


Figure 5.3.3 Sequence diagram

#### ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

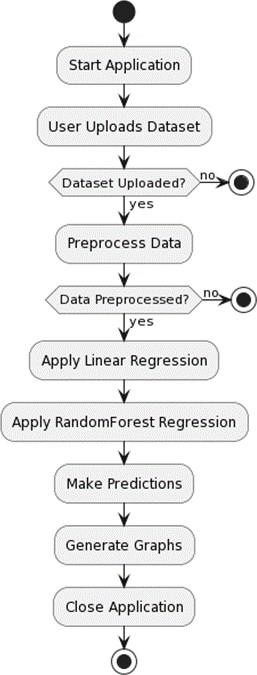


Figure 5.3.4 Activity diagram

# CODING AND ITS IMPLEMENTATION

* + 1. **Source code**

from tkinter import \* import tkinter

from tkinter import filedialog import numpy as np

from tkinter.filedialog import askopenfilename import pandas as pd

from tkinter import simpledialog import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split import pandas as pd

import numpy as np

from sklearn.tree import DecisionTreeRegressor from sklearn.preprocessing import StandardScaler from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestRegressor import os

import matplotlib.pyplot as plt import pickle

from sklearn.metrics import r2\_score,mean\_squared\_error

main = tkinter.Tk()

main.title("Exploring the Relationship Between Weather Patterns and Energy Consumption in Smart Homes: A Regression Analysis")

main.geometry("1000x1300")

global filename

global rnn\_acc,lstm\_acc global model

global X, Y,X\_train,y\_train,X\_test,y\_test,y\_pred,y\_pred1,mse,mse1,r21,r2 global dataset

global le

global r2\_score,mean\_squared\_error

def upload(): global filename global dataset

filename = filedialog.askopenfilename(initialdir = "dataset") text.delete('1.0', END)

text.insert(END,filename+' Loaded\n\n') dataset = pd.read\_csv(filename) dataset.dropna(inplace=True) text.insert(END,str(dataset.head))

def preprocess(): global dataset global X, Y

global X\_train,X\_test,y\_train,y\_test global le

text.delete('1.0', END)

X = dataset.drop("total load forecast",axis=1) Y = dataset['total load forecast']

print(Y) text.insert(END,str(X.head())) scaler = StandardScaler()

X= scaler.fit\_transform(X) X

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42) text.insert(END,"\n\nTotal Records for training : "+str(len(X\_train))+"\n")

text.insert(END,str(Y.head())) print(Y.shape)

print(X.shape)

def LinearRegression(): global model

global r2\_score

global X\_train,X\_test,y\_train,y\_test,y\_pred,mse,r2 from sklearn.linear\_model import LinearRegression model=LinearRegression() model.fit(X\_train,y\_train) y\_pred1=model.predict(X\_test)

from sklearn.metrics import r2\_score y\_pred = model.predict(X\_test)

r2 = r2\_score(y\_test, y\_pred) print(f"R2 Score: {r2}")

text.insert(END,'\n\nLinear Regression r2 score : '+str(r2)+"\n\n")

def RandomForestRegressor():

global X\_train,y\_test,X\_test,y\_train,mean\_squared\_error,mse1,r21,y\_pred1 from sklearn.ensemble import RandomForestRegressor model=RandomForestRegressor()

# Train the model model.fit(X\_train, y\_train) # Predict on the test set

y\_pred1 = model.predict(X\_test)

r21 = r2\_score(y\_test, y\_pred1) print(f"R2 Score: {r21}")

text.insert(END,'RandomForestRegressor r2 score : '+str(r21)+"\n\n") def DecisionTreeRegressor():

global X\_train,y\_test,X\_test,y\_train,mean\_squared\_error,mse2,r22,y\_pred2 from sklearn.tree import DecisionTreeRegressor model=DecisionTreeRegressor()

# Train the model model.fit(X\_train, y\_train) # Predict on the test set

y\_pred2 = model.predict(X\_test)

r22 = r2\_score(y\_test, y\_pred2) print(f"R2 Score: {r22}")

text.insert(END,'DecisionTreeRegressor r2 score : '+str(r22)+"\n\n") def predict():

global model text.delete('1.0', END)

file = filedialog.askopenfilename(initialdir="dataset") test = pd.read\_csv(file)

y\_pred = model.predict(test) for i in range(len(test)):

row\_data = test.iloc[i].values # Get the values of the row

text.insert(END, f"Row {i+1}: {row\_data}, Predicted = {y\_pred[i]} Watts\n\n") def graph():

global y\_pred,y\_test,mse1,mse,y\_pred1 # Create a scatter plot plt.figure(figsize=(8, 8)) plt.scatter(y\_test, y\_pred, color='blue')

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], linestyle='--', color='red', lw=2) # Identity line

# Set labels and title plt.xlabel('Actual Values') plt.ylabel('Predicted Values')

plt.title('LinearRegression Performance') # Create a scatter plot plt.figure(figsize=(8, 8)) plt.scatter(y\_test, y\_pred1, color='blue')

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], linestyle='--', color='red', lw=2) # Identity line

# Set labels and title plt.xlabel('Actual Values') plt.ylabel('Predicted Values')

plt.title('RandomForestRegression Performance') # Show the plot

plt.show()

# Create a scatter plot

plt.figure(figsize=(8, 8)) plt.scatter(y\_test, y\_pred2, color='blue')

plt.plot([min(y\_test),max(y\_test)],[min(y\_test),max(y\_test)],linestyle='--', color='red', lw=2) #

Identity line

# Set labels and title plt.xlabel('Actual Values') plt.ylabel('Predicted Values')

plt.title('DecisionTreeRegression Performance') plt.show()

# Create a bar chart to compare MSE

labels = [ 'LinearRegression','RandomForest','DecisiontreeRegression'] r2\_score = [r2, r21,r22]

plt.bar(labels,r2\_score) plt.xlabel('Regression Algorithm') plt.ylabel('r2\_score')

plt.title('Comparison of Regression Algorithms') plt.show()

def close():

main.destroy()

font = ('times', 15, 'bold')

title = Label(main, text='Exploring the Relationship Between Weather Patterns and Energy Consumption in Smart Homes: A Regression Analysis', justify=LEFT) title.config(bg='lavender blush', fg='DarkOrchid1')

title.config(font=font) title.config(height=3, width=120) title.place(x=100,y=5) title.pack()

font1 = ('times', 12, 'bold')

uploadButton = Button(main, text="Upload Dataset ", command=upload) uploadButton.place(x=100,y=100)

uploadButton.config(font=font1)

preprocessButton = Button(main, text="Preprocess Dataset", command=preprocess)

preprocessButton.place(x=300,y=100) preprocessButton.config(font=font1)

LRButton = Button(main, text="Linear Regression Algorithm", command=LinearRegression) LRButton.place(x=500,y=100)

LRButton.config(font=font1)

RfButton = Button(main, text="RandomForest Algorithm", command=RandomForestRegressor) RfButton.place(x=1000,y=100)

RfButton.config(font=font1)

DTButton = Button(main, text="DecisionTree Algorithm", command=DecisionTreeRegressor) DTButton.place(x=750,y=100)

DTButton.config(font=font1)

graphButton = Button(main, text="Performance Graphs", command=graph) graphButton.place(x=100,y=150)

graphButton.config(font=font1)

predictButton = Button(main, text="Prediction using Test Data", command=predict) predictButton.place(x=300,y=150)

predictButton.config(font=font1)

closeButton = Button(main, text="Close Application", command=close) closeButton.place(x=550,y=150)

closeButton.config(font=font1)font1 = ('times', 12, 'bold') text=Text(main,height=20,width=160) scroll=Scrollbar(text) text.configure(yscrollcommand=scroll.set) text.place(x=10,y=250)

text.config(font=font1) main.config(bg='light coral') main.mainloop()

## Implementation

#### Python Development Steps :

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourishedfor another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000"and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. Theemphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -

* + - * and preferably only one -- obvious way to do it."Some changes in Python 7.3:
        + Print is now a function
        + Views and iterators instead of lists
        + The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
        + There is only one integer type left, i.e. int. long is int as well.
        + The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
        + Text Vs. Data Instead Of Unicode Vs. 8-bit

#### Purpose :-

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

#### Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* + - * + Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
        + Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something abouthow much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages canpick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

#### Modules Used in Project TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machinelearning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released underthe Apache 2.0 open-source license on November 9, 2015.

#### NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multi dimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* + - * + A powerful N-dimensional array object
        + Sophisticated (broadcasting) functions
        + Tools for integrating C/C++ and Fortran code
        + Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

#### Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

#### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

#### Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

## Implementation description

This project implements a regression analysis to explore the relationship between weather patterns and energy consumption in smart homes. Below is the step-by-step explanation:

1. Importing Libraries: The code begins by importing the necessary libraries for data manipulation, analysis, and machine learning. These libraries include pandas for handling data, numpy for numerical operations, matplotlib and seaborn for data visualization, and machine learning libraries from sklearn for regression analysis.
2. Loading the Dataset: The dataset is loaded using pd.read\_csv(), assuming the dataset is stored in a CSV file named "Data.csv." The dataset is stored in the df DataFrame.
3. Data Processing

* Handling Missing Data: The code checks for missing values in the dataset using df.isnull().sum().Missing values are dropped from the dataset using df.dropna(inplace=True).
* Exploring the Dataset: The code provides an overview of the dataset using df.info(), df.head(), and df.describe() to check the data's structure, the first few rows, and basic statistics.
* Data Visualization: There is some data visualization using matplotlib and seaborn. For example, it creates a histogram of the "total load forecast" column using plt.hist().
* Correlation Analysis: The code calculates the correlation matrix between numerical variables in the dataset using df.corr(). It also displays correlations in descending order with respect to the "total load forecast" variable.
* Data Splitting: The dataset is split into training and testing sets using train\_test\_split from sklearn. The features (X) and the target variable (y) are separated, and standard scaling is applied to the features using StandardScaler.
* Decision Tree Regressor: A Decision Tree Regressor is created and trained using the training data with hyperparameters like max depth, min samples split, and min samples leaf specified.
* Random Forest Regressor: A Random Forest Regressor is created and trained using the training data.
* Model Evaluation: The code evaluates the performance of the Decision Tree and Random Forest regressors using the coefficient of determination (R-squared) as a metric, which measures the goodness of fit of the models to the data.

#### Dataset description

* + temp: Temperature (in Kelvin) column represents the temperature at the given time in Kelvin. It includes both high and low temperatures.
  + temp\_min: Minimum Temperature (in Kelvin) column Indicates the lowest temperature recorded within the specified time period, also in Kelvin.
  + temp\_max: Maximum Temperature (in Kelvin) Represents the highest temperature recorded within the specified time period, measured in Kelvin.
  + pressure: Atmospheric Pressure (in hPa) column provides information about the atmospheric pressure at the given time, measured in hectopascals (hPa).
  + humidity: Humidity Level (percentage) Indicates the amount of moisture in the air, expressed as a percentage.
  + wind\_speed: Wind Speed (in m/s) Represents the speed of the wind at the specified time, measured in meters per second.
  + wind\_deg: Wind Direction (in degrees) Indicates the compass direction from which the wind is blowing, measured in degrees.
  + rain\_1h: Amount of Rainfall in the Last Hour (in mm) Provides the amount of rainfall recorded in the last hour, measured in millimeters (mm).
  + rain\_3h: Cumulative Rainfall in the Last 3 Hours (in mm) Represents the total amount of rainfall recorded in the last three hours, measured in millimeters (mm).
  + snow\_3h: Cumulative Snowfall in the Last 3 Hours (in mm) Indicates the total amount of snowfall recorded in the last three hours, measured in millimeters (mm).
  + generation nuclear: Nuclear Power Generation (in MW) column provides the amount of electricity generated from nuclear sources, measured in megawatts (MW).
  + generation other: Other Types of Power Generation (in MW) Represents the amount of electricity generated from sources other than nuclear, solar, and wind, measured in megawatts (MW).
  + generation other renewable: Renewable Energy Generation Other Than Solar and Wind (in MW) Indicates the amount of electricity generated from renewable sources other than solar and wind, measured in megawatts (MW).
  + generation solar: Solar Power Generation (in MW) Provides the amount of electricity generated from solar sources, measured in megawatts (MW).
  + generation waste: Waste-Based Power Generation (in MW) Represents the amount of electricity generated from waste-based sources, measured in megawatts (MW).
  + generation wind offshore: Wind Power Generation from Offshore Sources (in MW) Indicates the amount of electricity generated from offshore wind sources, measured in megawatts (MW).
  + generation wind onshore: Wind Power Generation from Onshore Sources (in MW) Provides the amount of electricity generated from onshore wind sources, measured in megawatts (MW).
  + forecast solar day ahead: Solar Power Forecast for the Next Day (in MW) column provides the forecasted amount of electricity that will be generated from solar sources for the next day, measured in megawatts (MW).
  + forecast wind onshore day ahead: Onshore Wind Power Forecast for the Next Day (in MW) Represents the forecasted amount of electricity that will be generated from onshore wind sources for the next day, measured in megawatts (MW).
  + total load forecast: Total Energy Load Forecast (in MW) Indicates the forecasted total energy load, which represents the expected demand for electricity, measured in megawatts (MW).

This dataset contains a combination of meteorological data and information about energy generation and consumption. It can be used for various types of analyses, including understanding the impact of weather on energy consumption and generation or building predictive models related to energy demand.

## What is Python?

Below are some facts about Python.

* + Python is currently the most widely used multi-purpose, high-level programming language.
  + Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
  + Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
  + Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* + Machine Learning
  + GUI Applications (like Kivy, Tkinter, PyQt etc. )
  + Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  + Image processing (like Opencv, Pillow)
  + Web scraping (like Scrapy, BeautifulSoup, Selenium)
  + Test frameworks
  + Multimedia

## Advantages of Python

Let’s see how Python dominates over other languages.

#### Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

#### Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

#### Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

#### Improved Productivity

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

#### IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

#### Simple and Easy

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

#### Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

#### Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

#### Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

#### Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

#### Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

#### Advantages of Python Over Other Languages

1. **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

#### Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

#### Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn

different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

## Disadvantages of Python

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

#### Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

#### Design Restrictions

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

#### Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

#### Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

#### Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

#### How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4parts to help understand better.

#### Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: [https://www](https://www/).python.org

Figure 6.2.1:Downloading Python

Now, check for the latest and the correct version for your operating system. Step 2: Click on the Download Tab.



Figure 6.2.2: Selecting Latest Version

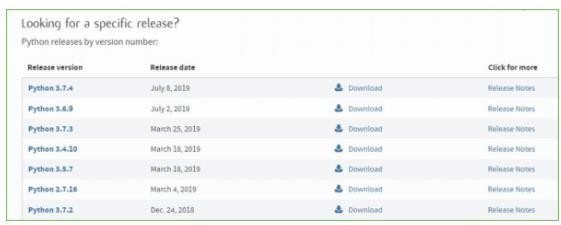
Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Figure 6.2.3: Python versions

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.



Figure 6.2.4:selecting windows version

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

## Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

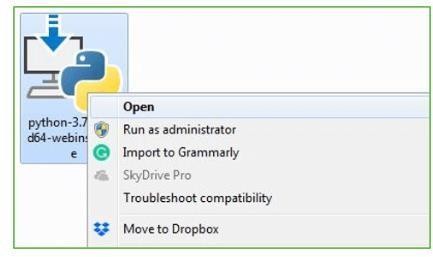


Figure 6.2.5:Installing Python

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



Figure 6.2.6:Adding Path

Step 3: Click on Install NOW After the installation is successful. Click on Close.



Figure 6.2.7:installation Successful

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

#### Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

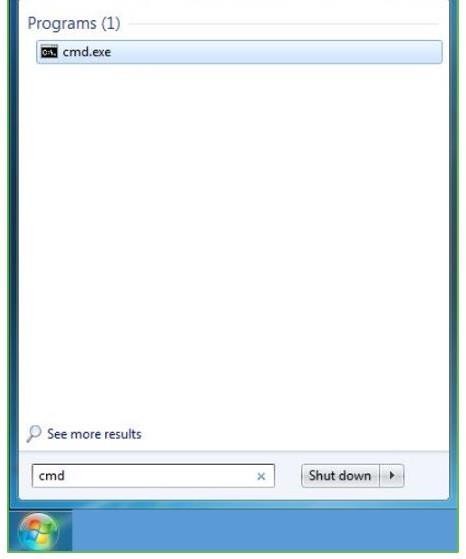


Figure 6.2.8:verifying installation

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

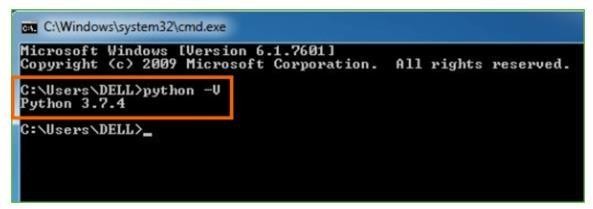


Figure 6.2.9:checking installation

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

#### Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



Figure 6.2.10:launching Program

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

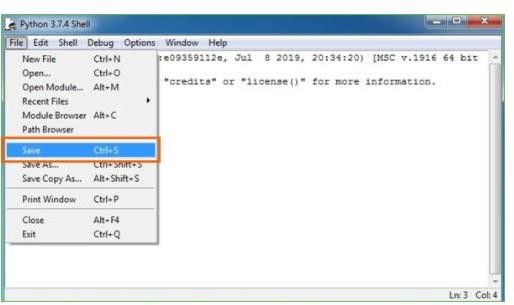


Figure 6.2.11:saving files

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

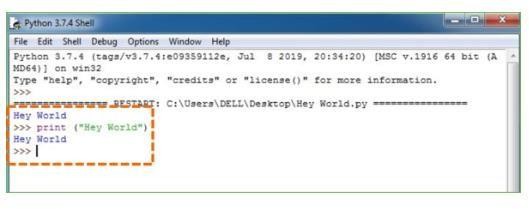


Figure 6.2.12:example program

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

## History of Python : -

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

#### What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the

human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

**Categories Of Machine Leaning :-**At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

*Supervised learning* involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

*Unsupervised learning* involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We willsee examples of both types of unsupervised learning in the following section.

## Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

## Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

**Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of overfitting & underfitting** − If the model is overfitting or underfitting, it cannot be represented well for the problem.

**Curse of dimensionality** − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

## Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

#### How to Start Learning Machine Learning?

Arthur Samuel coined the term **“Machine Learning”** in 1959 and defined it as a **“Field of study that gives computers the capability to learn without being explicitly programmed”.**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to [Indeed](http://blog.indeed.com/2019/03/14/best-jobs-2019/), Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

#### How to start learning ML?

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

#### Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

1. Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

1. Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

1. Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is [Python](https://www.geeksforgeeks.org/python-programming-language/)! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as [Keras,](https://keras.io/) [TensorFlow,](https://keras.io/) [Scikit-learn,](https://scikit-learn.org/stable/) etc.

So if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as [**Fork Python**](https://practice.geeksforgeeks.org/courses/fork-python)available Free on GeeksforGeeks.

#### Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

#### Terminologies of Machine Learning

* **Model –** A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* **Feature –** A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* **Target (Label) –** A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* **Training –** The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* **Prediction –** Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

#### Types of Machine Learning

* **Supervised Learning –** This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* **Unsupervised Learning –** This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* **Semi-supervised Learning –** This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* **Reinforcement Learning –** This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

## Advantages of Machine learning :-

#### Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

#### No human intervention needed (automation)

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

#### Continuous Improvement

As [**ML algorithms**](https://data-flair.training/blogs/machine-learning-algorithms/)gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

#### Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

#### Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

## Disadvantages of Machine Learning :-

#### Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

#### Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

#### Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

#### e. High error-susceptibility

[**Machine Learning**](https://en.wikipedia.org/wiki/Machine_learning)is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it. multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of database.

#### Jupyter Notebook

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at [Project Jupyter.](http://jupyter.org/)

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

## Anaconda :-

### What is Anaconda Python?

Together with a list of Python packages, tools like editors, Python distributions include the Python interpreter. Anaconda is one of several Python distributions. Anaconda is a new distribution of the Python and R data science package. It was formerly known as Continuum Analytics. Anaconda has more than 100 new packages.

This work environment, Anaconda is used for scientific computing, [data science,](https://www.springpeople.com/data-science-training-certification) statistical analysis, and machine learning. The latest version of Anaconda 5.0.1 is released in October 2017.

The released version 5.0.1 addresses some minor bugs and adds useful features, such as updated R language support. All of these features weren’t available in the original 5.0.0 release.

This package manager is also an environment manager, a Python distribution, and a collection of open source packages and contains more than 1000 R and [Python Data Science](https://www.springpeople.com/blog/25-python-questions-answers-for-data-science-interviews/) Packages.

### Why Anaconda for Python?

There’s no big reason to switch to Anaconda if you are completely happy with you regular python. But some people like data scientists who are not full-time developers, find anaconda much useful as it simplifies a lot of common problems a beginner runs into.

Anaconda can help with –

* [Installing Python](https://www.ics.uci.edu/~pattis/common/handouts/pythoneclipsejava/python.html) on multiple platforms
* Separating out different environments
* Dealing with not having correct privileges and
* Getting up and running with specific packages and libraries

# SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**7.1. TYPES OF SYSTEM TESTING**

#### Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application it is done after the completion of an individual unit before integration.

## Purpose:

Unit testing involves systematically verifying the correctness and reliability of individual components of the regression analysis process. This includes testing functions responsible for data preprocessing to ensure accurate handling of missing values, outliers, and data normalization. Additionally, unit tests validate algorithms for feature selection to confirm the selection of relevant weather variables influencing energy consumption

#### Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted. Invalid Input: identified classes of invalid input must be rejected. Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

## Purpose:

Functional tests ensure that the analysis accurately captures the relationship between weather patterns and energy consumption in smart homes, by verifying that the selected features effectively represent weather variables, the regression models accurately predict energy consumption, and the evaluation metrics provide meaningful insights into model performance. Furthermore, functional

testing ensures that the interpretation of results aligns with the study's objectives and theoretical framework, providing valuable insights into the impact of weather patterns on energy usage in smart homes

**7.2 TEST CASES:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Test Case** | **Excepted Result** | **Result** | **Remarks(IF Fails)** |
| 1. | Upload Dataset | Initially data need to  be uploaded. | Pass | Executed successfully |
| 2. | Preprocess Dataset | After Uploading the Data, it shows the total number of records. | Pass | Data preprocessing has executed successfully. |
| 3. | Temperature Data Missing | Regression analysis runs successfully without error | Fail | Temperature data is a critical predictor, missing values affect model accuracy and integrity. |
| 4. | Performing the Linear Regression, Random Forest,Decisio  n Tree algorithms. | It gives the regression scores of algorithms. | Pass | Executed Successfully. |
| 5. | Performan ceGraphs | It gives the Graphs | Pass | Executed Successfully |

Table no 7.2 Test Cases

* 1. **OUTPUT SCREENS**

Figure 8.1 depicts the graphical user interface (GUI) of Smart Homes, presumably showcasing the interface through which users interact with various smart home functionalities. This interface includes features such as controlling lighting, thermostats, security systems, and other connected devices remotely or through automated schedules. The GUI could also display real-time data about energy consumption, indoor air quality, or other relevant metrics, allowing users to make informed decisions about managing their home environment. Figure 8.2, it presents the data preprocessing steps undertaken before applying a linear regression model. Data preprocessing is a crucial step in machine learning pipelines as it involves cleaning, transforming, and organizing raw data to make it suitable for analysis. This figure include processes such as handling missing values, scaling features, encoding categorical variables, and splitting the data into training and testing sets. Additionally, the R2 score of the linear regression model is displayed, which indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

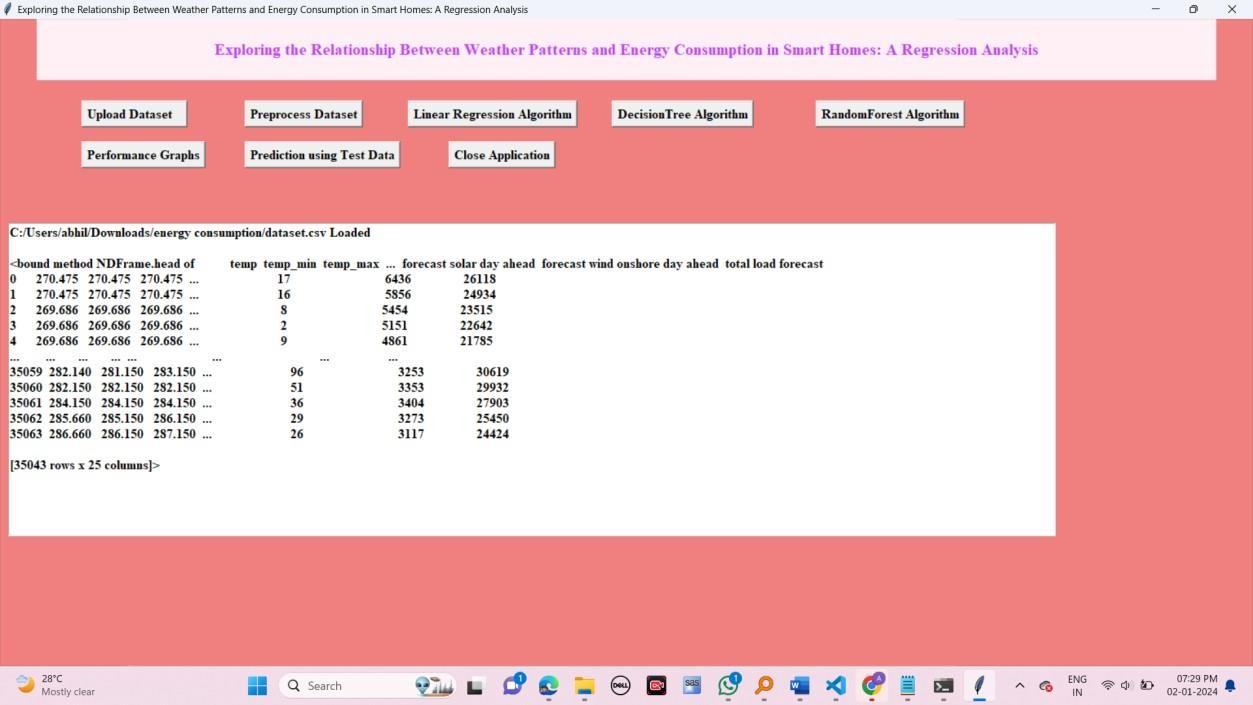


Figure 8.1: Presents the GUI of Smart Homes.

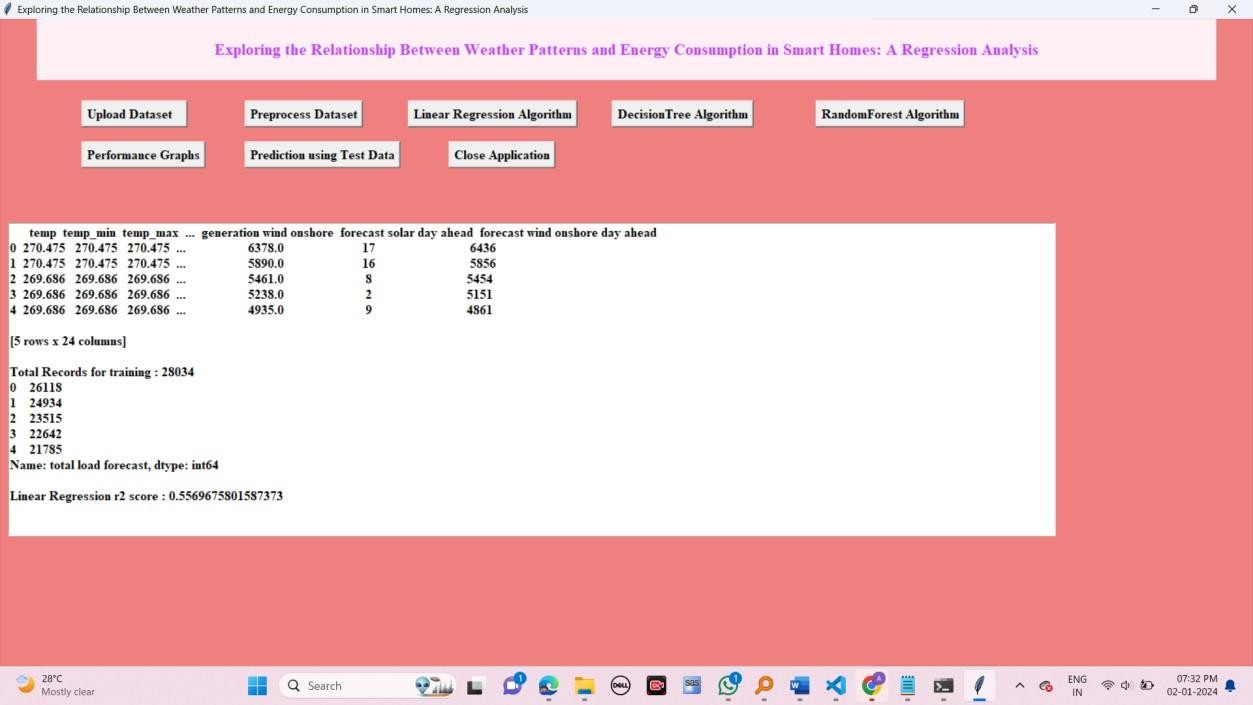


Figure 8.2: Presents the Data Preprocessing and R2 Score of Linear Regression model.

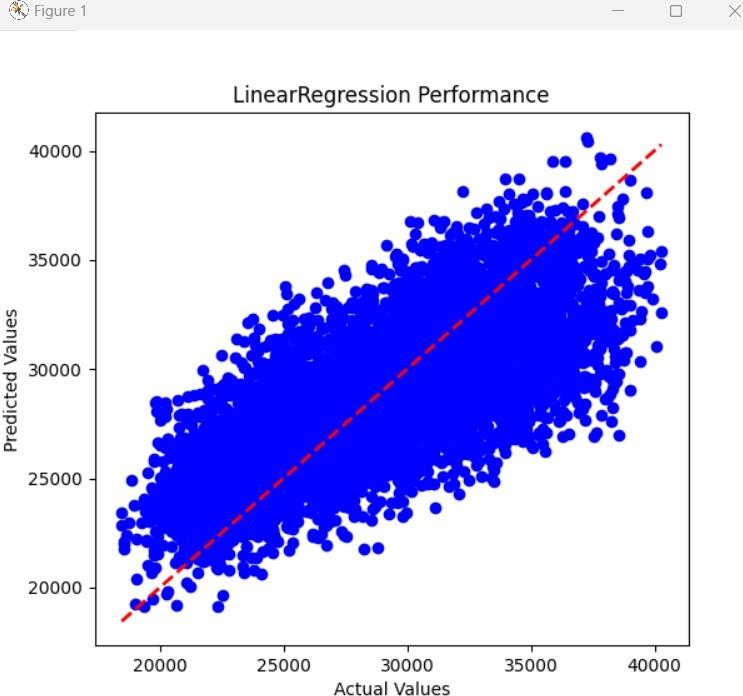


Figure8.3: Presents the Linear Regression Model Prediction on Test data.

Figure 3 shows the results of applying the linear regression model to the test data. This includes a plot comparing the actual values of the dependent variable against the predicted values generated by the model. The accuracy of the model's predictions can be evaluated visually by observing how closely the predicted values align with the actual values. Moving on to Figure 4, it displays the R2 score of a decision tree regression model. Decision tree regression is a non-parametric supervised learning method used for regression tasks. The R2 score provides insight into how well the decision tree model fits the data, with a score closer to 1 indicating a better fit.

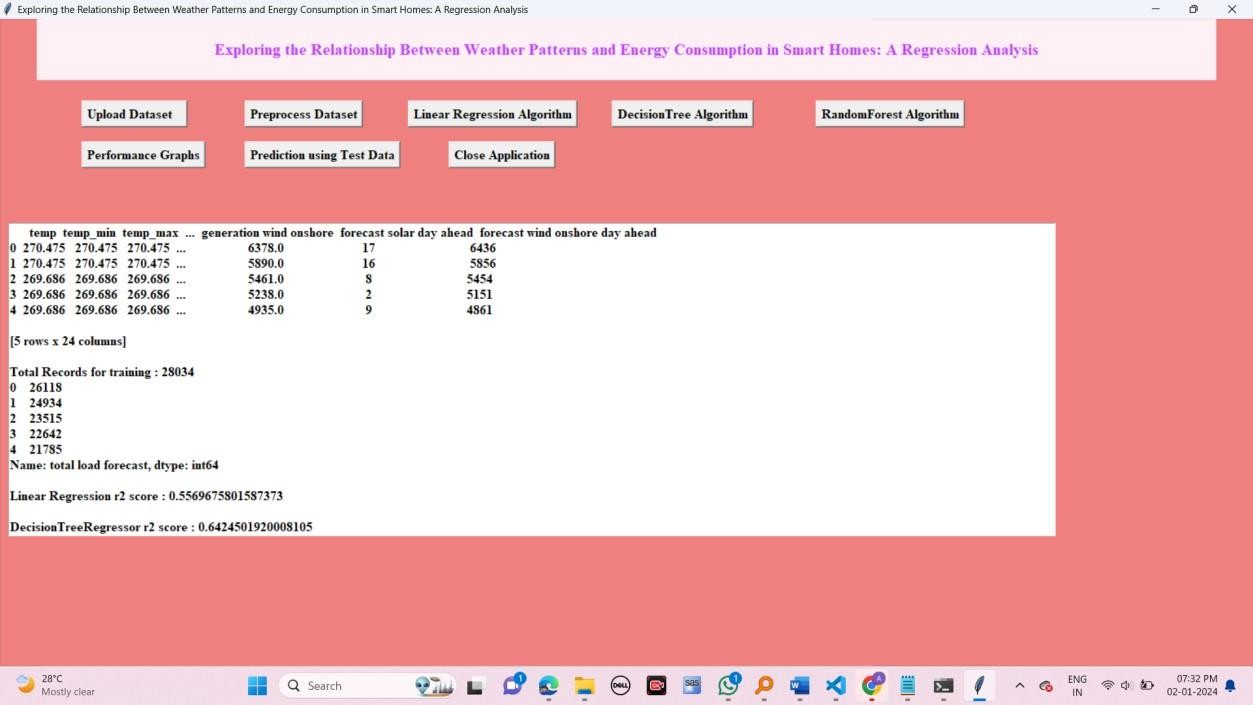


Figure 8.4: Presents the R2 Score of Decision Tree Regression model.

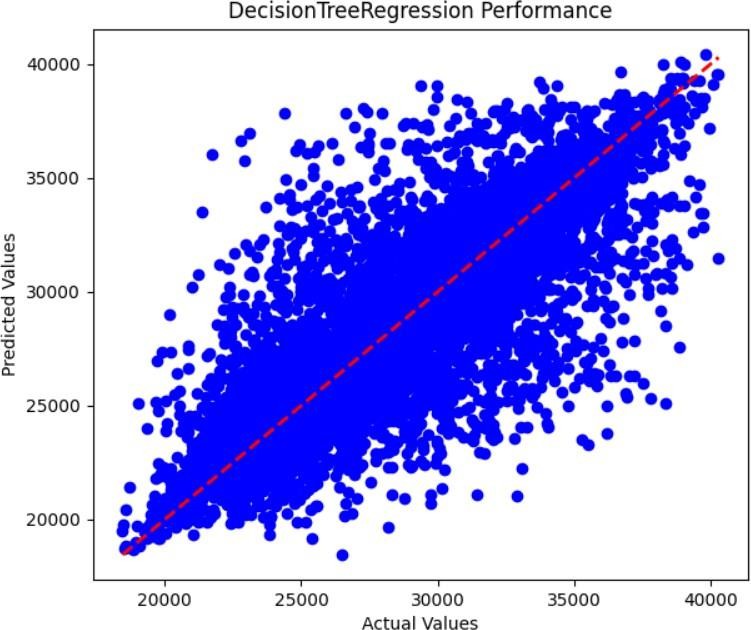


Figure 8.5: Presents Plot of Decision Tree Regression Model Prediction on Test data.

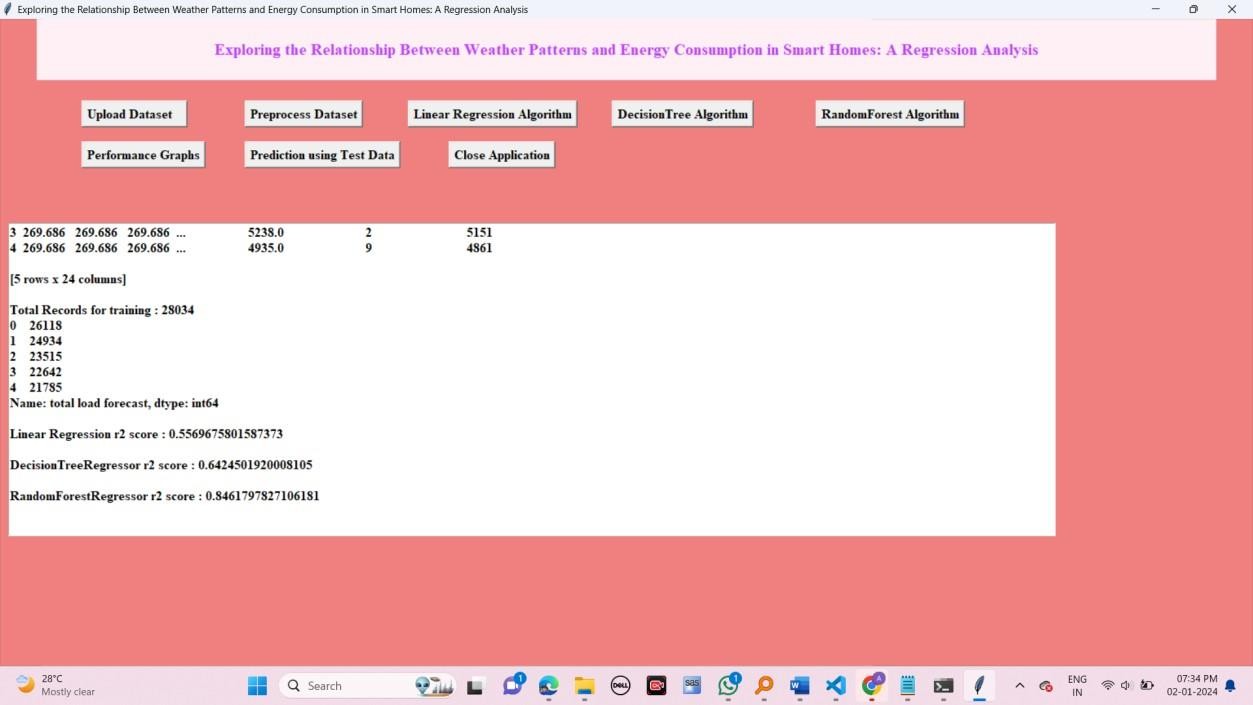


Figure 8.6: Presents the R2 Score of Random Forest Regression model.

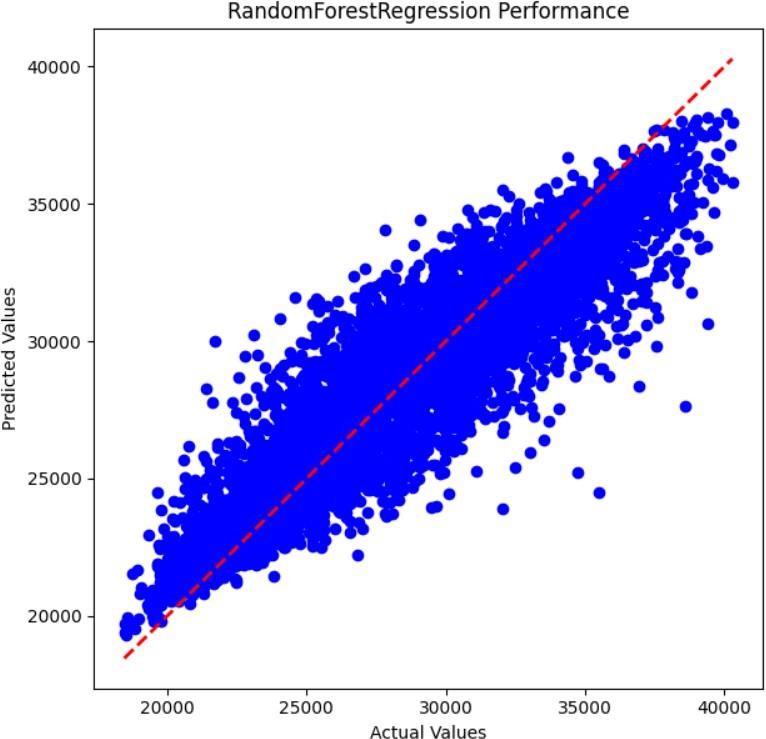


Figure 8.7: Presents Plot of Random Forest Regression Model Prediction on Test data.

Figure 5 presents a plot illustrating the predictions made by the decision tree regression model on the test data. Similar to Figure 3, this plot compares the actual values against the predicted values generated by the decision tree model. Figure 6 showcases the R2 score of a random forest regression model. Random forest is an ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of the individual trees. The R2 score in this figure provides

an indication of the random forest model's predictive performance. Figure 7 displays a plot depicting the predictions made by the random forest regression model on the test data. As with Figures 3 and 5, this plot compares the actual values against the predicted values generated by the random forest model.

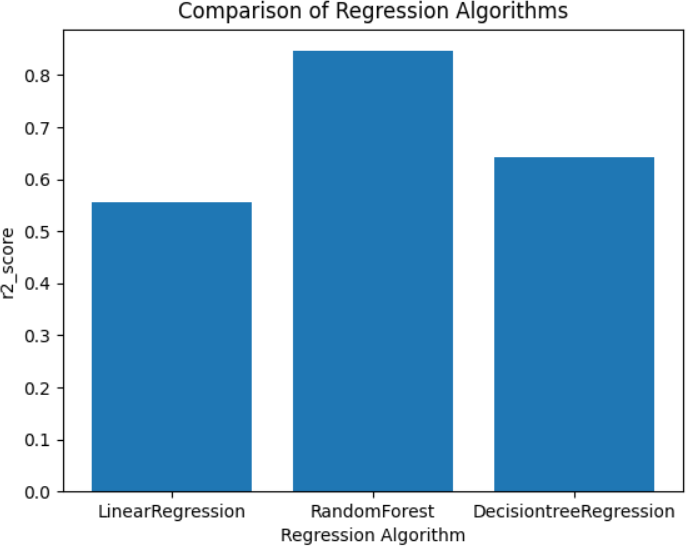


Figure 8,8: Comparison Graph of each model Performance.

Figure 8 presents a comparison graph illustrating the performance of each model (linear regression, decision tree regression, and random forest regression) based on their respective R2 scores. This graph allows for a direct comparison of how well each model fits the data and makes predictions. Finally, Figure 9 showcases the predictions made by the proposed model on the test data. This could represent a novel or improved regression model developed for a specific application, with its performance evaluated against existing models through metrics such as R2 score or visual comparison of predicted versus actual values.

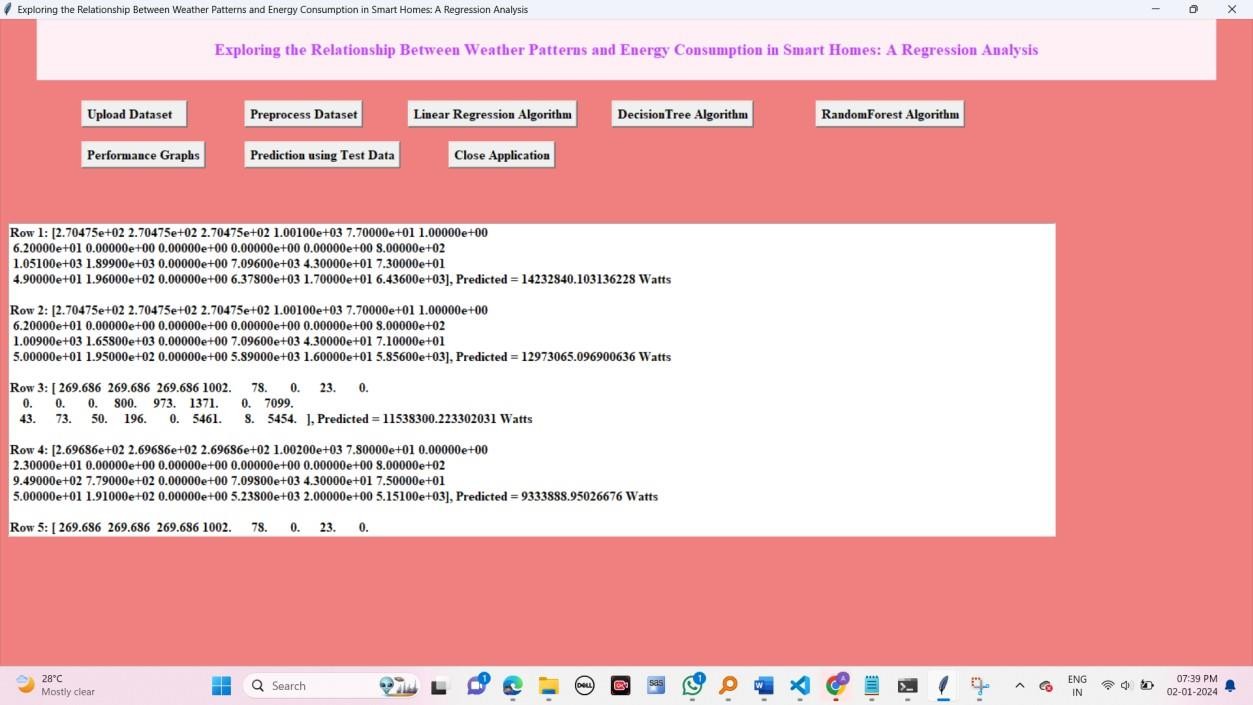


Figure8.9: Proposed Model Prediction on test data.

# CONCLUSION

In conclusion, this research has successfully delved into the intricate relationship between weather patterns and energy consumption in smart homes, employing advanced regression analysis and machine learning techniques. Through meticulous data analysis, meaningful patterns have been extracted, shedding light on the impact of weather variables such as temperature, humidity, and precipitation on energy load. The developed regression models, particularly the decision tree and random forest algorithms, have showcased promising accuracy in predicting energy consumption under varying weather conditions. These findings hold substantial implications for homeowners, energy providers, and policymakers alike. For homeowners, this study provides actionable insights into optimizing energy usage based on weather forecasts. By understanding how weather influences energy consumption, homeowners can implement targeted strategies to reduce costs and enhance efficiency. Energy providers can benefit from these insights by improving demand forecasting and management, ensuring a stable and efficient energy supply. Policymakers can integrate these findings into energy policies, fostering sustainable practices and guiding urban planning initiatives. Furthermore, this work demonstrates the power of data analytics and machine learning in addressing real-world challenges, showcasing their potential in the realm of energy management and sustainability.

# FUTURE ENHANCEMENTS

This analysis opens doors to several future avenues of research and application. Firstly, further refinement of machine learning models can enhance prediction accuracy. Exploring advanced algorithms such as deep learning neural networks might yield even more precise results, especially when dealing with vast and complex datasets from smart homes. Additionally, the integration of real- time data from IoT devices can transform these predictive models into dynamic systems. By incorporating live weather data and real-time energy consumption statistics, the models can adapt and provide instant recommendations, creating truly responsive and adaptive smart home environments.

# REFERENCES

1. Ku, A. L., Qiu, Y., Lou, J., Nock, D., and Xing, B. (2022). Changes in hourly electricity consumption under COVID mandates: a glance to future hourly residential power consumption pattern with remote work in Arizona. Appl. Energy 310:118539. doi: 10.1016/j.apenergy.2022.118539.
2. Sánchez-López, M., Moreno, R., Alvarado, D., Suazo-Martínez, C., Negrete-Pincetic, M., Olivares, D., et al. (2022). The diverse impacts of COVID-19 on electricity demand: the case of Chile.

Int. J. Electr. Power Energy Syst. 138:107883. doi: 10.1016/j.ijepes.2021.107883

1. Qiu, Y. L., Wang, Y. D., Iseki, H., Shen, X., Xing, B., and Zhang, H. (2022). Empirical grid impact of in-home electric vehicle charging differs from predictions. Resour.Energy Econ. 67:101275. doi: 10.1016/j.reseneeco.2021.101275
2. Al Khafaf, N., Rezaei, A. A., Moradi Amani, A., Jalili, M., McGrath, B., Meegahapola, L., et al. (2022). Impact of battery storage on residential energy consumption: an Australian case study based on smart meter data. Renew. Energy 182, 390–400. doi: 10.1016/j.renene.2021.10.005
3. Qiu, Y., Xing, B., Patwardhan, A., Hultman, N., and Zhang, H. (2022). Heterogeneous changes in electricity consumption patterns of residential distributed solar consumers due to battery storage adoption. iScience 25:104352. doi: 10.1016/j.isci.2022.104352
4. Liang, J., Qiu, Y., and Xing, B. (2022a). Impacts of electric-driven heat pumps on residential electricity consumption: an empirical analysis from Arizona, USA. Clean. Responsible Consum. 4:100045. doi: 10.1016/j.clrc.2021.100045
5. Liang, J., Qiu, Y., and Xing, B. (2022b). Impacts of the co-adoption of electric vehicles and solar panel systems: empirical evidence of changes in electricity demand and consumer behaviors from household smart meter data. Energy Econ. 112:106170. doi: 10.1016/j.eneco.2022.106170
6. Sousa, J. C., and Bernardo, H. (2022). Benchmarking of load forecasting methods using residential smart meter data. Appl. Sci. 12:9844. doi: 10.3390/app12199844
7. Lin, Y., Liu, J., Gabriel, K., Yang, W., and Li, C.-Q. (2022). Data-driven based prediction of the energy consumption of residential buildings in Oshawa. Buildings 12:2039.

doi: 10.3390/buildings12112039

1. Chinthavali, S., Tansakul, V., Lee, S., Whitehead, M., Tabassum, A., Bhandari, M., et al. (2022). COVID-19 pandemic ramifications on residential smart homes energy use load profiles. Energy Build. 259:111847. doi: 10.1016/j.enbuild.2022.111847
2. Sovacool, B. K., Hook, A., Sareen, S., and Geels, F. W. (2021). Global sustainability, innovation and governance dynamics of national smart electricity meter transitions. Glob. Environ.

Change 68:102272. doi: 10.1016/j.gloenvcha.2021.102272

1. Raman, G., and Peng, J. C.-H. (2021). Electricity consumption of Singaporean households reveals proactive community response to COVID-19 progression. Proc. Natl. Acad. Sci. 118:e2026596118. doi: 10.1073/pnas.2026596118
2. Li, L., Meinrenken, C. J., Modi, V., and Culligan, P. J. (2021). Impacts of COVID-19 related stay-at-home restrictions on residential electricity use and implications for future grid stability. Energy Build. 251:111330. doi: 10.1016/j.enbuild.2021.111330
3. Lou, J., Qiu, Y., Ku, A. L., Nock, D., and Xing, B. (2021). Inequitable and heterogeneous impacts on electricity consumption from COVID-19 mitigation measures. iScience 24:103231. doi: 10.1016/j.isci.2021.103231
4. Shaukat, M. A., Shaukat, H. R., Qadir, Z., Munawar, H. S., Kouzani, A. Z., and Mahmud, M. A.

P. (2021). Cluster analysis and model comparison using smart meter data. Sensors 21:3157. doi: 10.3390/s21093157

1. Fekri, M. N., Patel, H., Grolinger, K., and Sharma, V. (2021). Deep learning for load forecasting with smart meter data: online adaptive recurrent neural network. Appl. Energy 282:116177. doi: 10.1016/j.apenergy.2020.116177
2. Ribeiro Serrenh, T, and Bertoldi, P. (2019). Smart Home and Appliances: State of the art.

Luxembourg: Publications Office of the European Union. doi: 10.2760/453301

1. Singh, S., and Yassine, A. (2018). Big data mining of energy time series for behavioral analytics and energy consumption forecasting. Energies 11:452. doi: 10.3390/en11020452
2. Yildiz, B., Bilbao, J. I., Dore, J., and Sproul, A. B. (2017). Recent advances in the analysis of residential electricity consumption and applications of smart meter data. Appl. Energy 208, 402–427. doi: 10.1016/j.apenergy.2017.10.014