

# **ELECTRICITY PRICE FORECASTING FOR CLOUD COMPUTING USING EXTREME GRADIENT BOOST (XGBOOST) MODEL**

A report submitted in partial fulfillment of the requirements

for the Degree of

**Bachelor of Technology**  
in  
**Data Science**

BY

B.TANUJA (2011cs030021)

G.SRI PRIYA (2011cs030065)

JP.SAHITHI (2011cs030075)

Under the esteemed guidance of

Ms.B.PRASHANTHI

(Assistant professor)



**Department of BRANCH**

**School of Engineering**

**MALLA REDDY UNIVERSITY**

Maisammaguda, Dulapally, Hyderabad, Telangana 500100

**2024**



# MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

## Department of Data Science

### CERTIFICATE

This is to certify that the project report entitled “**ELECTRICITY PRICE FORECASTING FOR CLOUD COMPUTING USING EXTREME GRADIENT BOOST (XGBOOST) MODEL**”, submitted by B.Tanuja (2011CS030021), G.Sri Priya (2011CS030065), Jp.Sahithi (2011CS030075) towards the partial fulfillment for the award of Bachelor’s Degree in (Computer Science) from the Department of (Data Science), Malla Reddy University, Hyderabad, is a record of bonafide work done by him/ her. The results embodied in the work are not submitted to any other University or Institute for award of any degree or diploma.

**Internal Guide:**

**Ms.B.PRASHANTHI**

**Designation: Assistant Professor**

**Head of the Department**

**Dr.G.S.Naveen Kumar**

## DECLARATION

We hereby declare that the project report entitled “**ELECTRICITY PRICE FORECASTING FOR CLOUD COMPUTING USING EXTREME GRADIENT BOOST (XGBOOST) MODEL**” has been carried out by us and this work has been submitted to the Department of Data Science, Malla Reddy University, Hyderabad in partial fulfillment of the requirements for the award of degree of Bachelor of Technology. We further declare that this project work has not been submitted in full or part for the award of any other degree in any other educational institutions.

Place: Hyderabad

Date:

B.Tanuja	2011CS030021
G.Sri Priya	2011CS030065
JP.Sahithi	2011CS030075

## ACKNOWLEDGEMENT

We extend our sincere gratitude to all those who have contributed to the completion of this project report. Firstly, We would like to extend our gratitude to Dr. V. S. K Reddy, Vice-Chancellor, for his visionary leadership and unwavering commitment to academic excellence. We would also like to express my deepest appreciation to our project guide Ms.B.Prashanthi, Assistant Professor, whose invaluable guidance, insightful feedback, and unwavering support have been instrumental throughout the course of this project for successful outcomes.

We are also grateful to Dr. G.S. Naveen Kumar, Head of the Department of Data Science, for providing us with the necessary resources and facilities to carry out this project.

We would like to thank Dr. Kasa Ravindra, Dean, School of Engineering, for his encouragement and support throughout my academic pursuit.

My heartfelt thanks also go to Dr. Harikrishna Kamatham, Associate Dean School of Engineering for his guidance and encouragement.

We are deeply indebted to all of them for their support, encouragement, and guidance, without which this project would not have been possible.

B.TANUJA (2011cs030021)

G.SRI PRIYA (2011cs030065)

JP.SAHITHI (2011cs030075)

## ABSTRACT

This project focuses on implementing XGBoost, a powerful machine learning algorithm, for electricity price forecasting in cloud computing environments. The objective is to create an accurate and adaptable predictive model that aids decision-making processes in resource allocation and cost management. Leveraging historical data, the model incorporates relevant features, ensuring precision in predicting short and long-term fluctuations. Emphasizing interpretability, the XGBoost framework provides insights into key driving factors. The model's adaptability to diverse scenarios and real-time forecasting capabilities enhance its practical utility. Rigorous validation against benchmarks, risk assessment integration, scalability, comprehensive documentation, and continuous improvement further contribute to its effectiveness. This project aims to advance forecasting accuracy, providing valuable insights for efficient decision-making in the dynamic landscape of cloud computing. By examining the model's performance, we aim to showcase its effectiveness in capturing complex patterns inherent in the dynamics of electricity prices. The ultimate goal is to provide cloud service providers with a valuable tool for making informed decisions, thereby contributing to the optimization of their resources, reduction in operational expenditures, and an overall enhancement of sustainability within cloud computing environments. The findings of this project shed light on the potential of enhanced machine learning techniques to address the unique challenges associated with electricity price forecasting in the context of cloud computing.

## CONTENTS:

S.NO	Contents	Page No.
	<b>Abstract</b>	v
	<b>List of Figures</b>	vii-viii
<b>1.</b>	<b>Introduction</b>	1
	1.1 About the project	1
	1.2 Objective	1
<b>2.</b>	<b>Literature survey</b>	4
<b>3.</b>	<b>Proposed</b>	6
	3.1 Module description	9
	3.2 System design	12
	3.3 Modular design	20
	3.4 Implementaion	23
<b>4.</b>	<b>Result Analysis</b>	31
<b>5.</b>	<b>Conclusion and Future Scope</b>	38
	5.1 Conclusion	38
	5.2 Future Scope	38
<b>6.</b>	<b>References</b>	47
	6.1 Book References	40
	6.2 Website References	40
	6.3 Technical Publication References	40
<b>7.</b>	<b>Appendices</b>	41
	7.1 SW Used	41
	7.2 Methodologies	4.2
	7.3 Testing Methods Used	44
<b>8.</b>	<b>Code</b>	46

## LIST OF FIGUERS

Figure No.	Figure Name	Page No.
3.1.	Proposed model	15
3.2	Class diagram	21
3.3	Activity diagram	22
3.4	Use Case Diagram	23
3.5	Sequence Diagram	24
3.6	Er diagram	25
3.7	Df diagram	26
3.8	Flow chart of modular design	27
3.9	User table	30
4..1	Home Page	31
4.2	Registration successful	31

4.3	Login Page	32
4.4	Post login screen	32
4.5	Load Dataset	33
4.6	Dataset Uploade d Successfully	33
4.7	View Data	34
4.8	Model Selection	34
4.9	Score for XGBoost	35
4.10	Regressor Model Pridiction	35
4.11	Prediction Values	36
4.12	Predicted Price	36
4.13	Logout page	37



# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 ABOUT THE PROJECT**

In developing our electricity forecasting project for cloud computing using XGBoost, we began by collecting historical electricity consumption data, considering factors like time of day, day of the week, and seasonal variations. Additional data such as weather conditions and cloud usage patterns were also incorporated. Subsequently, we preprocessed the data to handle missing values, outliers, and normalize features. The dataset was then split into training and testing sets for model evaluation, and the XGBoost model was trained on the training data, with parameters fine-tuned for optimal performance. Validation of the model using the testing set ensured accurate predictions. Once satisfied with the model's performance, we implemented it into the cloud infrastructure, enabling continuous electricity demand forecasting. Feedback loops were integrated to periodically update the model, ensuring adaptability to changing consumption patterns. Beyond the lines of code and algorithms, our project resonates with a broader narrative—the narrative of responsible innovation. By optimizing energy usage, we're contributing to a greener, more sustainable future. It's a testament to the power of collaboration, where the synergy of diverse skills coalesces into a project that transcends the boundaries of conventional problem-solving. In essence, our project is not just about forecasting electricity demand; it's a journey of collective exploration, innovation, and a commitment to shaping a more sustainable and efficient future for cloud computing.

### **1.2 OBJECTIVE**

The primary goals include accurate short-term and long-term predictions, feature engineering for relevant variables, model interpretability, adaptability to diverse environments, real-time forecasting capabilities, rigorous validation against benchmarks, risk assessment integration, scalability for growing datasets, effective documentation, and continuous improvement for enhanced decision-making in cloud environments. The primary objective of employing XGBoost for electricity forecasting in the context of cloud computing is to develop a robust and accurate predictive model that can effectively anticipate future electricity prices. The overarching goal is to

enhance decision-making processes within cloud environments by providing reliable forecasts, enabling optimized resource allocation, and facilitating cost-effective management strategies.

- **Accurate Prediction** Develop a predictive model that leverages the strengths of XGBoost to accurately forecast electricity prices. The focus is on achieving high precision in predicting short-term and longterm fluctuations in prices, considering the dynamic nature of electricity markets.
- **Feature Engineering** Identify and incorporate relevant features that significantly impact electricity prices in cloud computing scenarios. This involves a thorough analysis of historical data and an understanding of the factors influencing price dynamics, such as demand patterns, market trends, and external factors affecting cloud services.
- **Model Interpretability** Ensure the XGBoost model's interpretability to provide insights into the key features driving electricity price forecasts. This involves visualizing feature importance, understanding the contribution of each variable, and communicating these findings in a comprehensible manner to stakeholders.
- **Adaptability** Design the model to be adaptable to different electricity markets and cloud computing environments. Consider the diverse characteristics of electricity data and cloud service demands, ensuring that the XGBoost model can generalize well and maintain predictive accuracy across various scenarios.
- **Real-time Forecasting** Develop capabilities for real-time forecasting to enable timely decision-making in cloud computing. This involves optimizing the model for efficiency, allowing it to process and update predictions swiftly based on the latest data, ensuring relevance in dynamic electricity markets.
- **Validation and Benchmarking** Rigorously validate the XGBoost model's performance against historical data, using appropriate metrics to assess accuracy, precision, and reliability. Benchmark the model against traditional forecasting approaches to showcase its superiority in terms of predictive power.
- **Risk Assessment** Integrate a risk assessment component to identify and quantify uncertainties associated with electricity price forecasts. This involves incorporating probabilistic forecasting

techniques within the XGBoost framework to provide decision-makers with a more nuanced understanding of potential variations in predictions.

- **Scalability** Ensure that the forecasting model is scalable, allowing it to handle increasing volumes of data and adapt to the evolving dynamics of cloud computing and electricity markets. Consider optimizations to enhance computational efficiency and accommodate growing datasets.
- **Continuous Improvement** Establish a framework for continuous model improvement by incorporating feedback, monitoring model performance over time, and integrating new data sources or features that enhance forecasting accuracy. By achieving these specific objectives, the XGBoost model for electricity forecasting in cloud computing aims to empower decision-makers with reliable insights, contributing to efficient resource management, cost optimization, and overall improved performance in cloud environments

## **CHAPTER 2**

### **LITERATURE SURVEY**

[1] Tungpimolrut et al. (2020) Tungpimolrut and team propose a hybrid model that combines the wavelet transform and neural networks for electricity price forecasting in cloud computing. The wavelet transform is employed as a preprocessing step to decompose the time series data into different frequency components, allowing the model to capture both short-term fluctuations and long-term trends effectively. Neural networks, known for their ability to learn complex patterns, are then applied to the decomposed components for accurate prediction. The testing phase, conducted with data from the Australian electricity market, demonstrated the model's superiority over traditional forecasting approaches. This integration of wavelet transform and neural networks showcases a holistic approach to handling the intricacies of electricity price fluctuations.

[2] Saleh Albahli, Muhammad Shiraz, Nasir Ayub (2020) Electricity Price Forecasting for Cloud Computing Using An Enhanced Machine Learning Model. The model achieves high accuracy, scoring 85.66 for Mean Squared Error (MSE) and 6.66 for Mean Absolute Error (MAE), leading to a substantial 25.32% reduction in electricity costs. The article underscores the effectiveness of the optimized model in data storage optimization, highlighting its potential for significant electricity cost savings. It suggests avenues for future research, including exploring additional parameters, refining constraints, and applying the model to diverse geographical areas using various data types such as energy, load forecasts, and weather data. Furthermore, the study emphasizes the role of machine learning in predicting energy forecasts for data centers, without specifying a particular classifier. It suggests expanding the research scope by exploring alternative classifiers like neural networks and incorporating clustering techniques to identify and address forecasting errors related to electricity price spikes.

[3] Maheen Zahid et al. (2019) Electricity price and load forecasting using enhanced convolutional neural network and enhanced support vector regression in smart grid. In: Electronics 8.2 (2019), p. 122. The paper introduces two models for electricity load and price prediction with a focus on enhancing forecasting accuracy. The models employ data pre-processing, including feature selection and extraction, coupled with classifier parameter tuning through grid search (GS) and cross-validation to optimize accuracy. Simulation results and performance metric values demonstrate the superior accuracy of the proposed models compared to benchmark schemes. A

hybrid feature selection technique, incorporating Recursive Feature Elimination (RFE), is utilized to enhance the model's robustness. RFE reduces data redundancy and dimensionality. The models leverage Enhanced Classifiers (ECNN and ESVR) for predicting electricity price and load, effectively addressing over-fitting and minimizing execution and computation time. Stability analysis confirms the reliability of the proposed models across different data sizes. The comprehensive approach, integrating feature selection, enhanced classifiers, and stability analysis, contributes to the improved accuracy and efficiency of the electricity load and price prediction models.

[4] Li et al. (2019) Li and colleagues present a sophisticated model for electricity price forecasting in cloud computing, leveraging a combination of deep belief network (DBN) and long short-term memory (LSTM). DBN is utilized to capture hierarchical features within the data, enabling the model to understand complex relationships at different levels. LSTM, known for its ability to capture temporal dependencies, complements the DBN by considering the sequential nature of time series data. The testing, conducted with data from the PJM electricity market, demonstrated improved accuracy compared to traditional models. This integration of DBN and LSTM reflects a nuanced approach to understanding and predicting electricity price dynamics in cloud computing environments.

[5] Wang et al. (2018) Wang, Zhang, and Gao propose a hybrid model for electricity price forecasting in cloud computing, incorporating extreme learning machine (ELM) and support vector regression (SVR). ELM is utilized for efficient and rapid learning of complex relationships in the data, while SVR contributes to capturing intricate patterns that might not be apparent through traditional methods. The hybrid model, tested with data from the Australian electricity market, exhibited enhanced forecasting performance compared to traditional models. This amalgamation of ELM and SVR highlights the importance of leveraging complementary strengths in machine learning techniques for accurate and efficient electricity price predictions in cloud computing scenario.

## **CHAPTER 3**

### **PROPOSED MODEL**

We propose this system to investigate a specific problem of whether it is valuable or not to use machine learning techniques to leverage a dramatic spike in electricity prices to offload data storage to minimize the energy consumption in cloud data centers using XGBoost Regressor, Random Forest Regressor and Support Vector Regressor. The proposed system leveraging XGBoost offers several advantages over the existing system:

- **Reduced Computational Time** XGBoost is known for its efficiency, providing faster computations compared to the existing MLNN and Ensemble techniques. This ensures improved real-time responsiveness in electricity load estimation.
- **Simplicity and User-Friendliness** XGBoost models are often simpler to implement and require less expertise for design, making the proposed system more user-friendly for deployment compared to the complexity associated with MLNN-based models.
- **Mitigated Overfitting** XGBoost incorporates regularization techniques, reducing the risk of overfitting. This addresses a potential drawback of the Ensemble techniques in the existing system, enhancing the accuracy of load estimation and the reliability of predictions.
- **Enhanced Robustness** XGBoost is robust in handling diverse data patterns and unforeseen anomalies, contributing to a more resilient system compared to the limited robustness of the existing approach.
- **Improved Scalability** XGBoost is designed for efficiency and scalability, making it more suitable for handling large datasets and accommodating increasing computational demands. This addresses the scalability challenges faced by the existing system.
- **Simplified Integration** XGBoost's simplicity facilitates easier integration into existing systems, potentially minimizing integration challenges compared to the coordination complexities of MLNN and Ensemble techniques.
- **Adaptability to Changes** The proposed system using XGBoost is more adaptable to changes in electricity consumption patterns, as it can quickly learn and adjust based on evolving

data, reducing dependency on historical data. The proposed system with XGBoost offers a streamlined, efficient, and adaptable approach, addressing the limitations of the existing system. It provides a balance between computational efficiency, model simplicity, and robust performance, making it a compelling alternative for electricity load estimation and consumption forecasting.

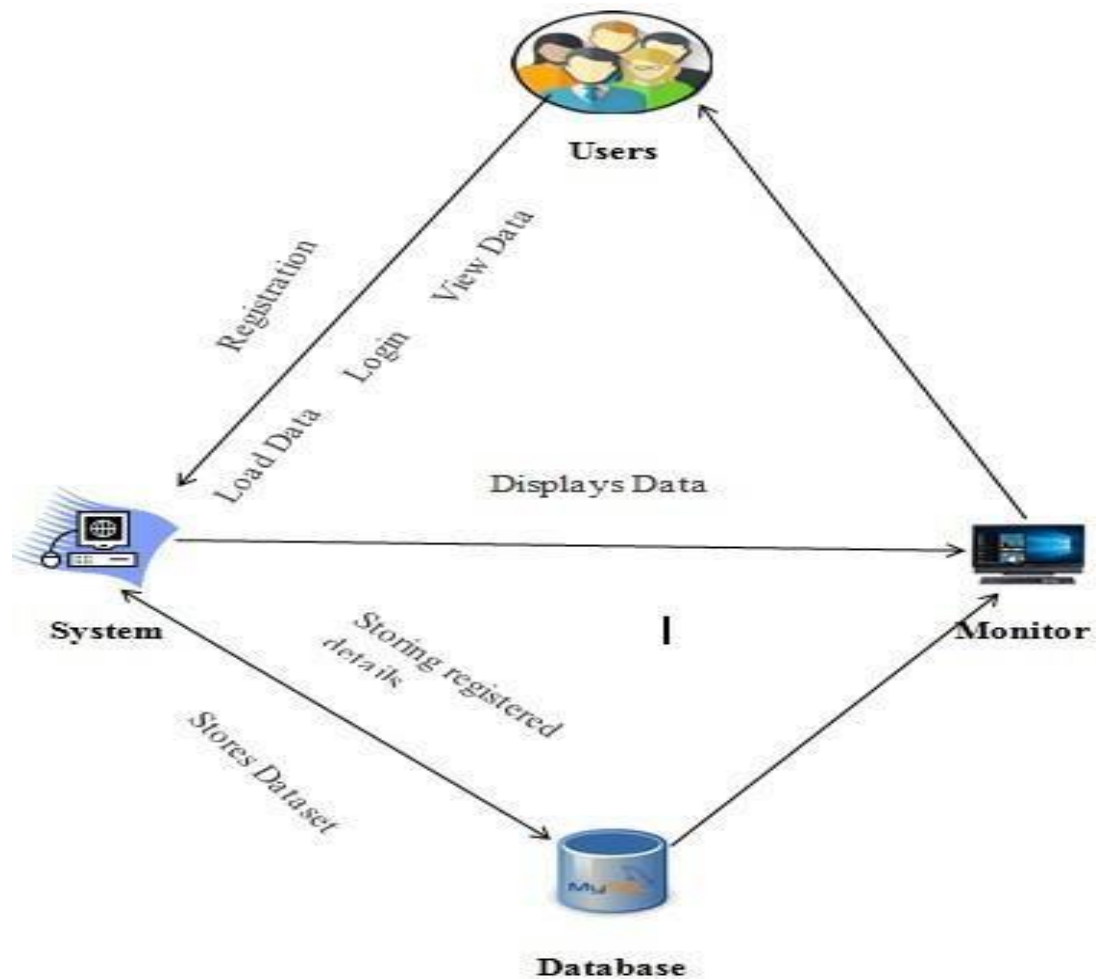


Fig 3.1:proposed model

Forecasting energy costs for cloud computing using the XGBoost technique involves several components in the system architecture. Here's an outline of the system architecture

## **1. Data Collection**

- Real-time and historical data related to cloud computing energy consumption and costs need to be collected. This includes data such as server utilization, workload patterns, energy prices, weather data, and other relevant factors.
- APIs, sensors, monitoring tools, and data warehouses can be utilized for data collection.

## **2. Data Preprocessing**

- Raw data collected needs to be preprocessed before feeding it into the XGBoost model.
- This step involves cleaning the data, handling missing values, encoding categorical variables, scaling numerical features, and possibly feature engineering to extract useful features.

## **3. Feature Selection**

- Selecting relevant features that have a significant impact on energy costs is crucial for model performance and interpretability.
- Techniques such as correlation analysis, feature importance from tree-based models, or domain knowledge can be used for feature selection.

## **4. Model Training with XGBoost**

- XGBoost (Extreme Gradient Boosting) is a popular machine learning technique known for its performance and scalability.
- The preprocessed data is used to train the XGBoost regression model to predict energy costs based on historical patterns and other relevant features.
- Hyperparameter tuning may be performed using techniques like grid search or randomized search to optimize the model's performance.

## **5. Model Evaluation**

The trained model needs to be evaluated using appropriate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).



Cross-validation techniques can be employed to assess the model's generalization performance.

## **6. Deployment**

- Once the model is trained and evaluated satisfactorily, it needs to be deployed in a production environment.
- Deployment can be done using containerization technologies like Docker, along with orchestration tools like Kubernetes for scalability and manageability.
- APIs can be created to serve predictions in real-time or on-demand.

## **7. Monitoring and Maintenance**

- Continuous monitoring of the deployed model's performance is essential to ensure it continues to provide accurate predictions.
- Monitoring tools can be employed to track model drift, data quality issues, and overall system health.
- Periodic retraining of the model with new data is necessary to keep it up-to-date and maintain its accuracy over time.

## **8. Integration with Decision Support Systems**

- The forecasted energy costs can be integrated into decision support systems used by cloud service providers or consumers to optimize resource allocation, workload scheduling, and cost management strategies. By following this architecture, we can develop a robust system for forecasting energy costs for cloud computing using the XGBoost technique.

### **3.1 MODULE DESCRIPTION**

The XGBoost model is responsible for utilizing the trained machine learning model to make accurate predictions of electricity prices based on input data. It acts as the interface between the prediction model and external systems or users, providing real-time or batch predictions as required.

## **1. User Interface Module**

- Description: Provides a graphical interface for users to interact with the system.
- Functionalities
- Displays real-time electricity consumption forecasts.
- Allows users to input preferences or adjustments for forecasting parameters.
- Presents visualizations of historical consumption trends.

## **2. Authentication and Authorization Module**

- Description: Manages user access to the system, ensuring secure and authorized interactions.
- Functionalities
- User login and authentication.
- Defines user roles and permissions for system access.

## **3. Dashboard Module**

- Description: Offers a centralized dashboard for monitoring and control.
- Functionalities
- Aggregates key performance metrics.
- Enables quick access to forecast summaries and historical data.
- 

## **4. Forecasting Engine Module**

- Description: Core module responsible for electricity consumption forecasting.
- Functionalities
- Utilizes the XGBoost model for accurate predictions.

- Receives input from the user interface and adapts forecasting parameters accordingly.

## **5. Notification Module**

- Description: Manages communication with users and system administrators.
- Functionalities
- Sends alerts for significant changes in electricity consumption patterns.
- Notifies users of system updates or maintenance schedules.

## **6. Data Management Module**

- Description: Handles the storage and retrieval of historical and real-time data.
- Functionalities
- Manages the database for efficient data storage.
- Supports data retrieval for model training and real-time forecasting.

## **7. Integration and Deployment Module**

- Description: Ensures smooth integration with existing cloud computing infrastructure.
- Functionalities
- Manages the deployment process to minimize downtime.
- Coordinates with other system components for seamless integration.

## **8. Monitoring and Logging Module**

- Description: Monitors system performance and logs relevant events.
- Functionalities
- Records system activities and user interactions for auditing.

- Generates performance reports and alerts for system administrators.

## **9. Maintenance Module**

- Description: Handles continuous system updates and maintenance.
- Functionalities
- Deploys patches and updates to the forecasting model.
- Ensures compatibility with evolving technologies.

## **10. Scalability and Performance Optimization Module**

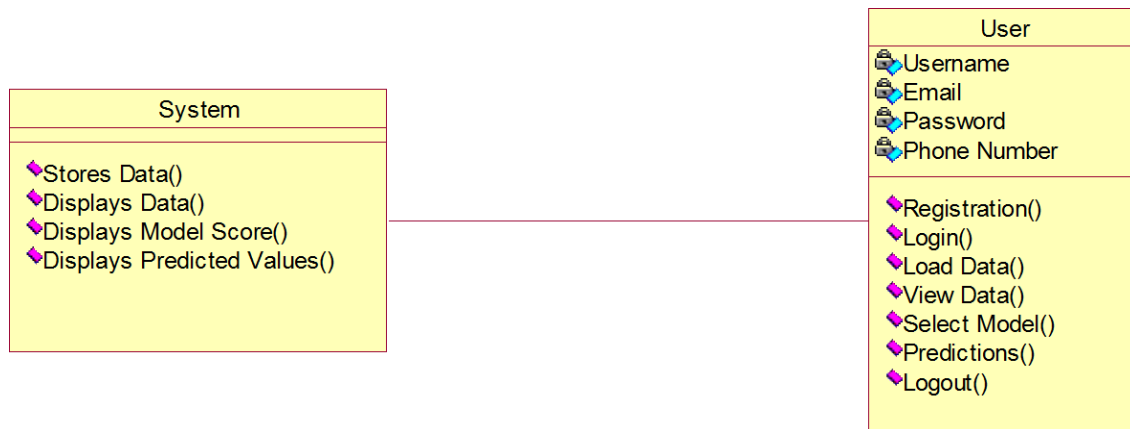
- Description: Addresses scalability challenges and optimizes system performance.
- Functionalities
- Implements strategies for handling larger datasets.
- Optimizes computational efficiency for enhanced responsiveness.

These modules collectively form a comprehensive architecture, ensuring a user-friendly experience, secure access, robust forecasting, and efficient system management in the proposed XGBoost-based electricity forecasting system for cloud computing.

## **3.2 SYSTEM DESIGN**

### **3.2.1. CLASS DIAGRAM:**

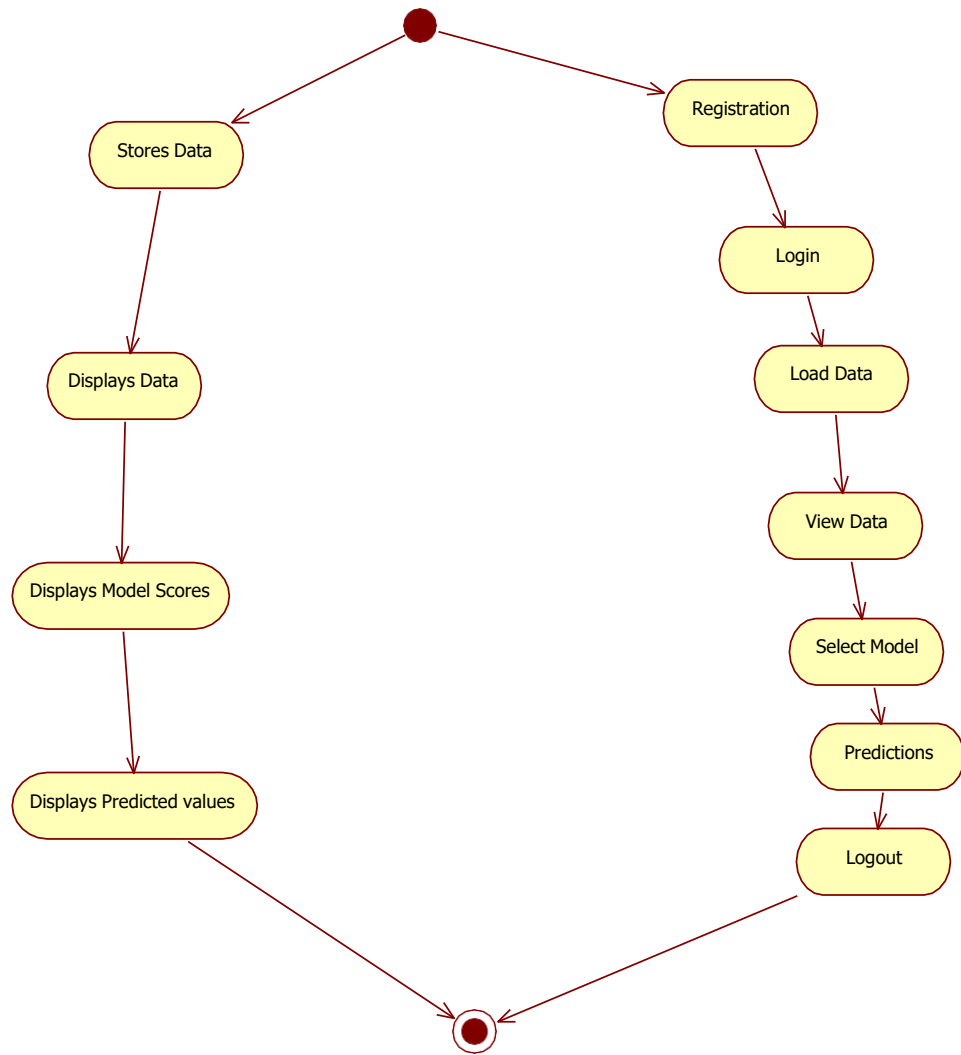
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains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



**Fig3.2:class diagram**

### **3.2.2 ACTIVITY DIAGRAM:**

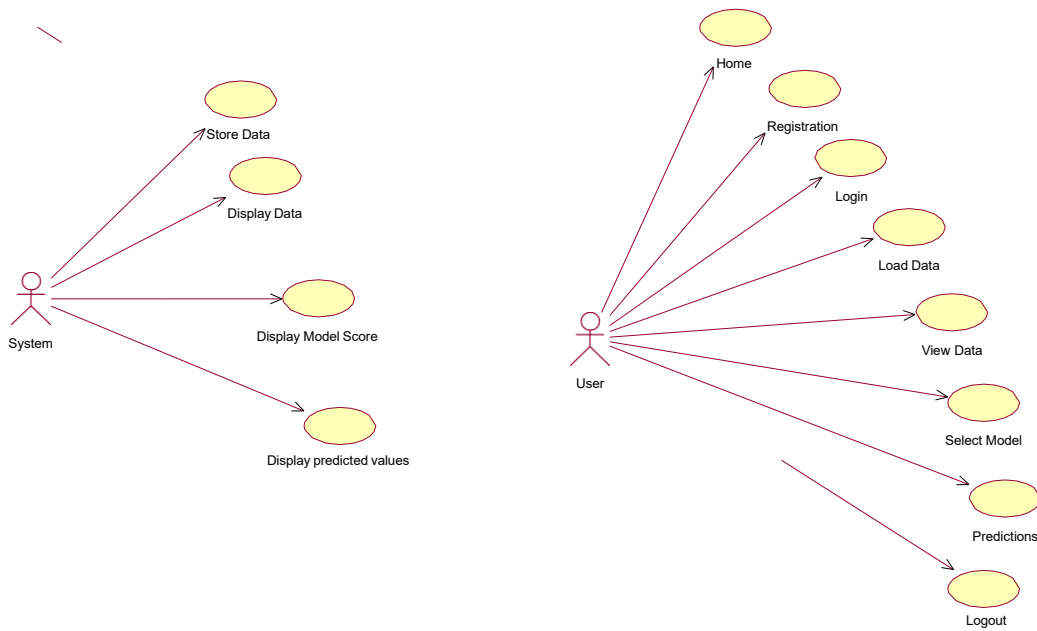
Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-bystep workflows of components in a system.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



**Fig3.3:activity diagram**

### 3.2.3 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.



**Fig3.4:use case diagram**

### **3.2.4 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.



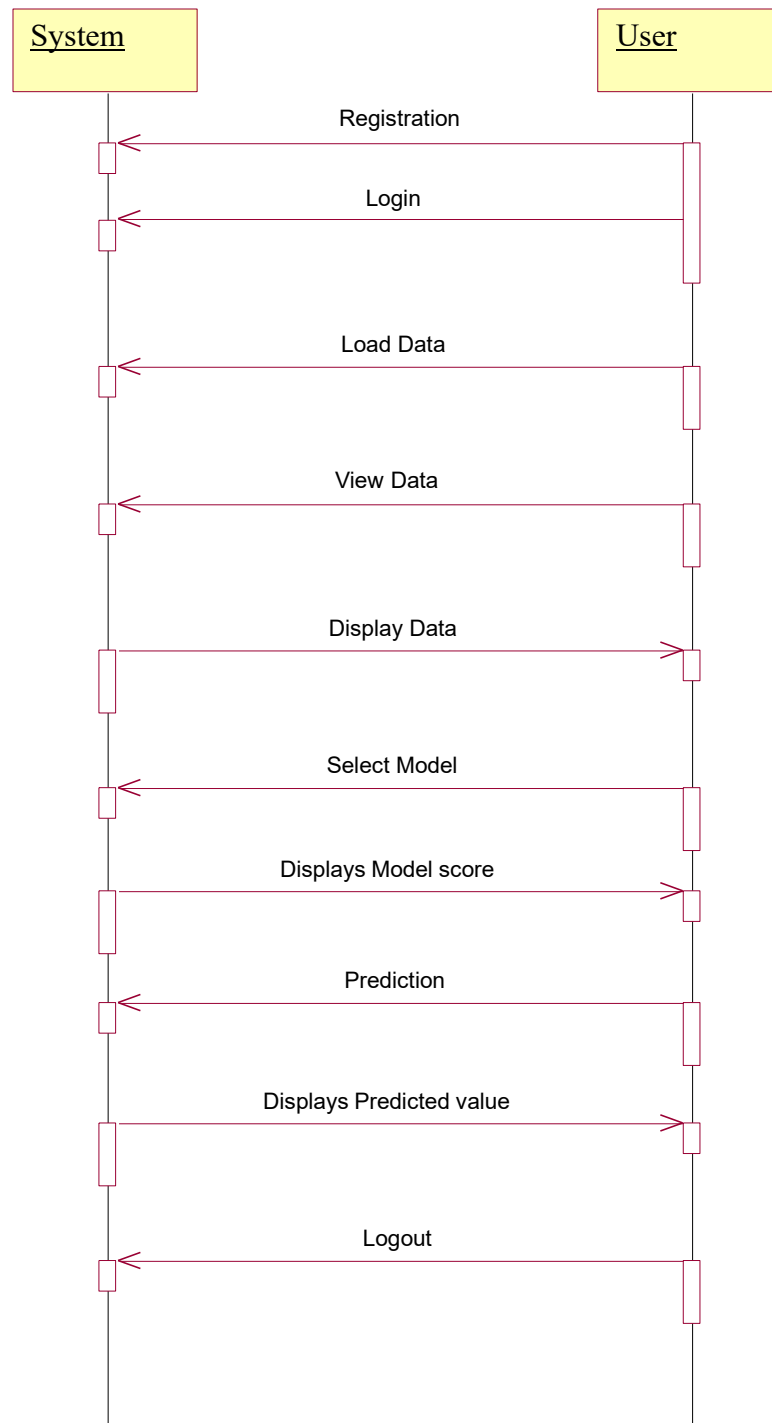


Fig3.5:sequence diagram

### 3.2.5 ER DIAGRAM:

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

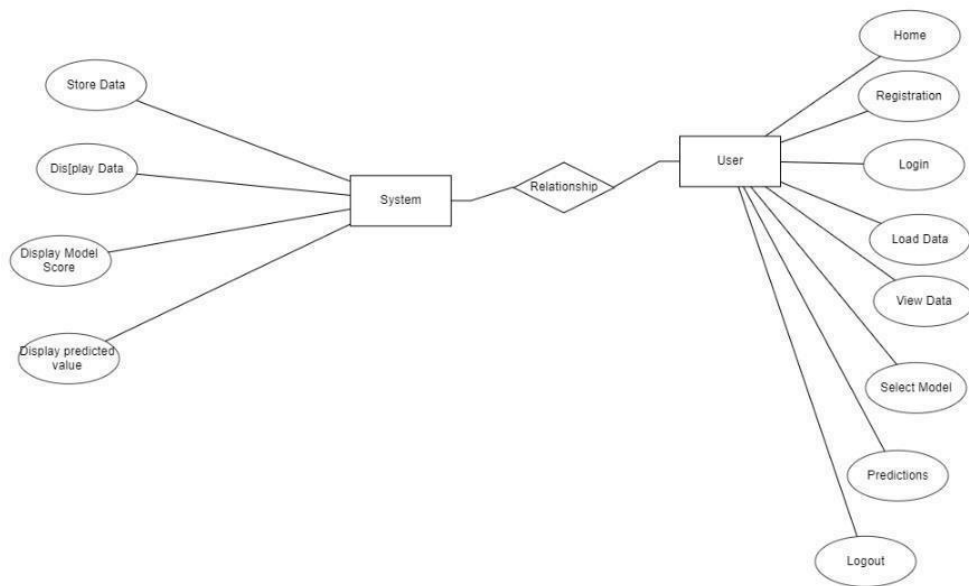


Fig3.6: ER diagram

### 3.2.6 DFD:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

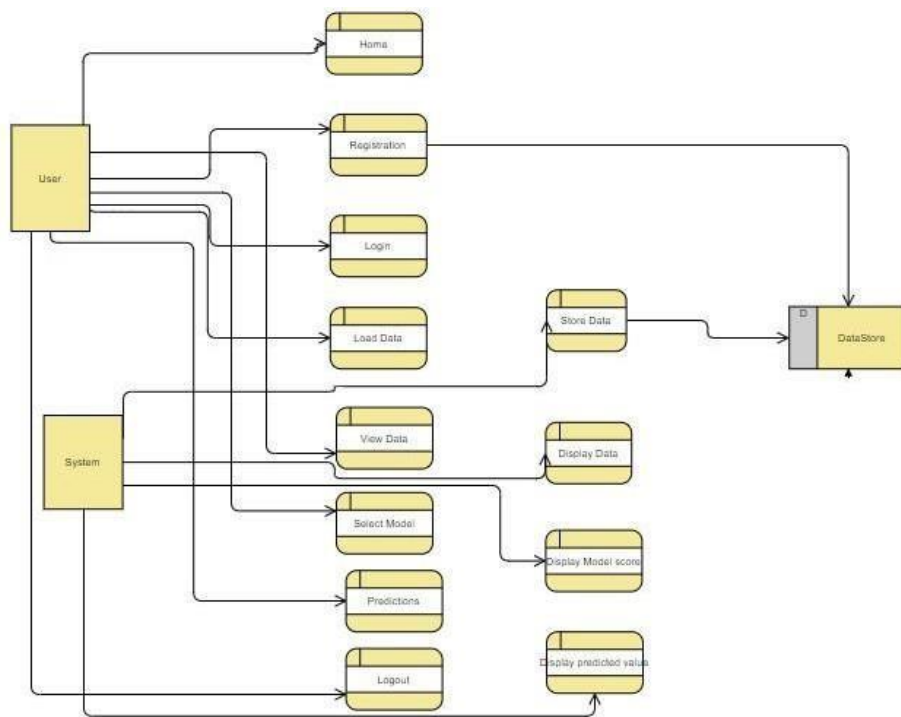


Fig3.7:dfd

### 3.3. Modular Design

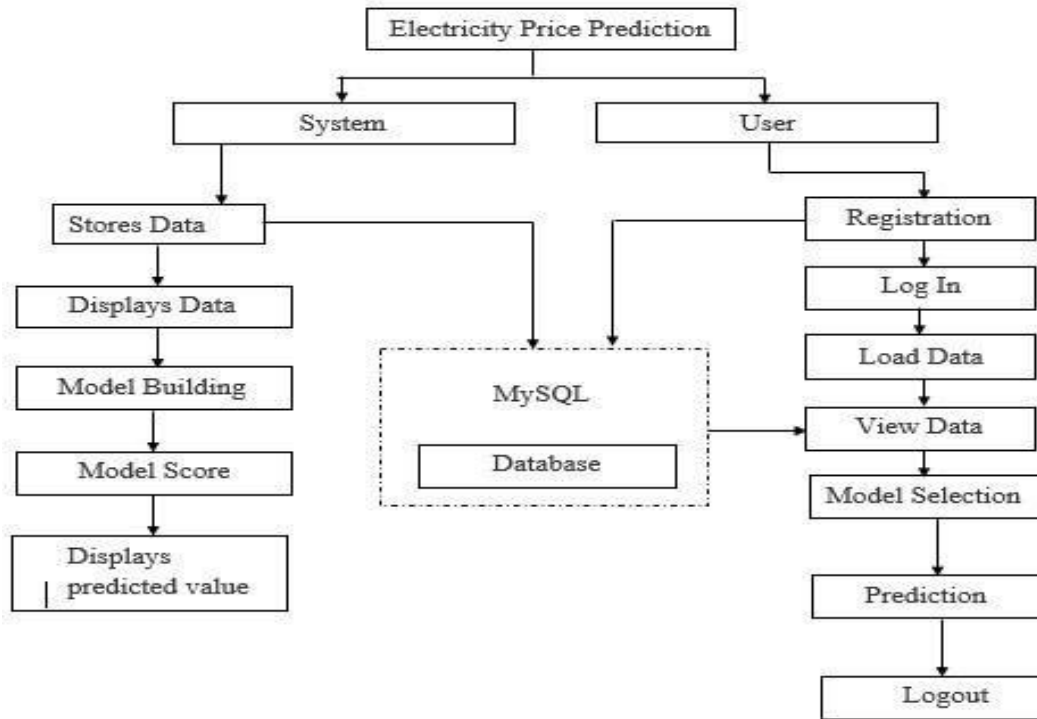


Fig:3.8:flow chart of modular design

The module aims to provide a comprehensive solution for forecasting energy costs in cloud computing environments. It utilizes the XGBoost (Extreme Gradient Boosting) technique, a powerful machine learning algorithm known for its efficiency and accuracy in regression and classification tasks. By integrating XGBoost into the forecasting process, the module offers a robust method for predicting energy consumption and associated costs within cloud computing infrastructures.

#### 1. Users

Represented as individuals interacting with the system. Users perform various actions such as registration, loading data, logging in, and viewing data.

## **2. System**

Acts as the central hub for processing user requests and interacting with other components.

It facilitates registration, data loading, login authentication, and data viewing functionalities.

## **3. Monitor**

Displays information to users and administrators. It receives data from the system for display purposes and provides visual feedback to users about their interactions with the system.

## **4. Database**

Stores both user-related information and datasets. It maintains registered user details, including authentication credentials, and stores datasets for the system to process and display.

### **3.3.2 Database Design**

Designing a database for forecasting energy costs involves structuring tables to store relevant data such as historical prices, weather conditions, market trends, and any additional features used in the prediction model. Below is the basic database design using SQL Yog Enterprise:

#### **1. Database Creation**

We begin by creating a new database in SQL Yog Enterprise, naming it something like "electricity".

#### **2. Table Design**

In this step we identify the entities and attributes needed for the prediction system.

Electricity Prices - Stores historical electricity price data.

#### **3. Generation Table (Dataset)**

Stores data related to electricity generation. Below are the attributes generation\_id: Primary Key fossil\_gas: Generation from fossil gas fossil\_hardcoal: Generation from hard coal hydro\_pumped\_storage\_consumption: Generation from hydro pumped storage consumption hydro\_water\_reservoir: Generation from hydro water reservoir other\_renewable:

Generation from other renewable sources waste: Generation from waste timestamp: Date and time of the measurement

#### 4. User Table (Database)

Stores user login details for authentication.

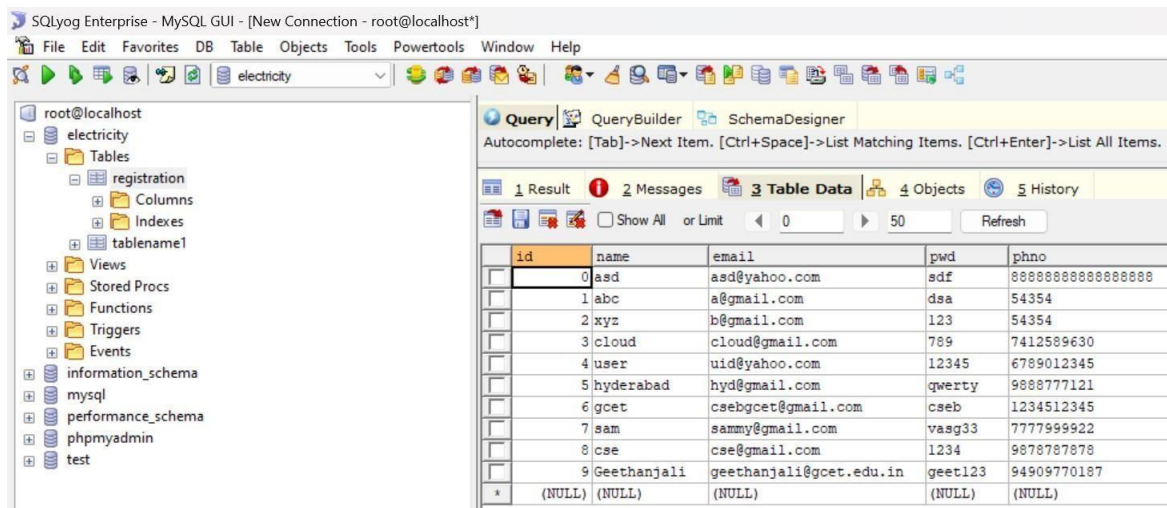
Below are the attributes user\_id: Primary Key username:

User's username password: Encrypted password for security email:

User's email address phone number: User's phone number

#### SQLyog Enterprise View

This view represents the database schema as visualized within the SQL Yog Enterprise graphical user interface (GUI). Each column name corresponds to a specific attribute within the database tables, providing a structured representation of the data fields stored in the database. The SQL Yog Enterprise GUI facilitates intuitive navigation and management of database objects, allowing users to interact with and manipulate the database schema efficiently.



The screenshot shows the SQLyog Enterprise MySQL GUI. On the left, a tree view displays the database structure for 'electricity', including tables like 'registration', 'Columns', 'Indexes', 'tablename1', 'Views', 'Stored Procs', 'Functions', 'Triggers', 'Events', 'information\_schema', 'mysql', 'performance\_schema', 'phpmyadmin', and 'test'. The main window shows the 'Table Data' tab for a table with columns: id, name, email, pwd, and phno. The table contains 9 rows of user data, with the first row having id 0 and name 'asd'.

id	name	email	pwd	phno
0	asd	asd@yahoo.com	sdf	88888888888888888888
1	abc	a@gmail.com	dsa	54354
2	xyz	b@gmail.com	123	54354
3	cloud	cloud@gmail.com	789	7412589630
4	user	uid@yahoo.com	12345	6789012345
5	hyderabad	hyd@gmail.com	qwerty	9888777121
6	gcet	csebgcet@gmail.com	cseb	1234512345
7	sam	sammy@gmail.com	vasg33	7777999922
8	cse	cse@gmail.com	1234	9878787878
9	Geethanjali	geethanjali@gcet.edu.in	geet123	94909770187
*	(NULL)	(NULL)	(NULL)	(NULL)

Fig3.9:user table

## 3.4 IMPLEMENTATION

### 3.4.1 System Implementation

- **Login**

The system allows users to authenticate their identity and access the system. Validates user credentials (username/password) against stored records in the system's database. Grants authorized users access to system functionalities and data.

- **Store Dataset**

The system stores datasets provided by users for forecasting energy cost analysis.

Accepts dataset files uploaded by users, validates file format, and stores data securely.

Provides the necessary data for training and testing machine learning models.

- **Model Selection**

The system facilitates the selection of the XGBoost model for forecasting energy costs. Presents users with available machine learning models, including XGBoost, Random forest and Support Vector Regressors. Allows users to choose the most suitable model for accurate predictions based on their requirements and dataset characteristics.

- **Model Predictions**

The system applies the selected XGBoost model to make predictions on electricity prices. Takes input data provided by users and generates predictions using the trained XGBoost model. Provides forecasted electricity prices for decision-making and planning purposes.

### 3.4.2. User Implementation

- **Registration**

Users can create accounts within the system to access its functionalities. Allows users to register with unique usernames and email addresses. Enables personalized access to system features and data analysis capabilities.

- **Login**

Registered users can log in to the system using their credentials. Provides a secure mechanism for user authentication. Grants authorized users access to their accounts and system functionalities.

- **View Modules**

After logging in, users are presented with available modules for forecasting energy costs analysis. Provides a user-friendly dashboard or interface displaying accessible features and workflows. Facilitates easy navigation and access to different functionalities within the system.

- **Load Dataset**

Users can upload datasets containing historical electricity cost data. Allows users to select and upload dataset files from their local storage. Initiates the data analysis process by providing relevant data for model training and evaluation.

- **View Dataset**

Users can visualize and explore the uploaded dataset. Presents a summary of dataset attributes, statistics, and possibly basic visualizations. Helps users understand the structure and content of the data for informed analysis and preprocessing decisions.

- **Select Model**

Users can choose the XGBoost model for electricity price prediction from available options. Allows users to explore model options and select the most suitable one based on their requirements. Empowers users to customize their analysis and choose models that best fit their dataset and prediction needs.



- **Make Predictions**

Users can input new data to generate predictions for future electricity prices using the selected XGBoost model. Provides a form or interface for users to input relevant variables and obtain predictions

### **3.4.3 XGBOOST REGRESSOR**

XGBoost (Extreme Gradient Boosting) Regressor is an advanced implementation of gradient boosting algorithm, specifically designed for regression tasks. Extreme Gradient Boosting, often abbreviated as XGBoost, is a powerful and versatile machine learning algorithm that has gained widespread popularity and is widely used in various fields such as data science, finance, healthcare, and more. XGBoost is particularly renowned for its ability to deliver exceptional predictive accuracy and robustness across diverse datasets. At its core, XGBoost is an ensemble learning technique that combines the predictions of multiple decision trees to make more accurate and robust predictions. It leverages the strengths of both bagging and boosting techniques, aiming to minimize bias and variance while enhancing overall model performance. Here are some key features and aspects of XGBoost

**Gradient Boosting:** XGBoost employs a gradient boosting framework, which involves Training decision trees to correct errors made by the previous ones. This iterative process allows it to continually refine its predictions and improve accuracy.

- **Regularization:** XGBoost includes L1 (Lasso) and L2 (Ridge) regularization terms in its objective function. This helps prevent overfitting and enhances the model's generalization capabilities.
- **Tree Pruning:** The algorithm uses a depth-first approach to grow trees and then prunes them backward to reduce overfitting, resulting in more efficient and interpretable models.
- **Handling Missing Values:** XGBoost has built-in mechanisms to handle missing data, making it robust in scenarios where data quality is a concern.
- **Parallel Processing:** XGBoost is optimized for speed and can take full advantage of multi-core processors, making it efficient for large-scale datasets.
- **Flexibility:** It can be used for both regression and classification tasks, making it a versatile choice for a wide range of problems.

### 3.4.4 RANDOM FOREST

Random Forest Regressor is a popular machine learning algorithm that belongs to the ensemble learning family. A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms. The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

Here's an explanation of its features

## **Ensemble Learning**

Random Forest is an ensemble learning method that combines multiple individual decision trees to make predictions. Each tree in the forest is trained independently on a random subset of the training data.

- **Decision Trees**

At its core, Random Forest is composed of decision trees. Each decision tree is built by recursively splitting the data based on feature values, with the goal of minimizing impurity or maximizing information gain at each split.

- **Bootstrap Aggregation (Bagging)**

Random Forest uses a technique called bootstrap aggregation or bagging to create multiple random subsets of the training data. This helps to reduce overfitting and improves the overall generalization of the model.

- **Random Feature Selection**

In addition to using random subsets of the training data, Random Forest also selects a random subset of features at each split in the decision tree. This further enhances the diversity among the individual trees in the forest.

- **Prediction**

For regression tasks, Random Forest predicts the output by averaging the predictions of all the individual trees in the forest. This ensemble approach typically results in more robust and accurate predictions compared to a single decision tree.

- **Parallelization**

Random Forest can be easily parallelized, as each tree in the forest can be trained independently of the others. This makes it suitable for training on large datasets and taking advantage of multi-core processors.

- **Robustness to Overfitting**

Random Forest tends to be less prone to overfitting compared to individual decision trees, especially when the number of trees in the forest is sufficiently large and the hyperparameters are well-tuned.

- **Feature Importance**

Random Forest provides a measure of feature importance, indicating the relative contribution of each feature to the model's predictions. This can be useful for feature selection and understanding the underlying relationships in the data.

### 3.4.5 SUPPORT VECTOR REGRESSOR

Support Vector Regressor (SVR) is a machine learning algorithm used for regression tasks. Unlike traditional regression techniques that focus on minimizing the error between predicted and actual values, SVR takes a different approach. It aims to create a hyperplane that best captures a specified fraction of the data points within a margin, known as the epsilon-tube, while minimizing the hyperplane's norm. This approach allows SVR to handle both linear and nonlinear relationships between variables effectively. SVR is particularly useful when dealing with complex, high-dimensional datasets, as it can reduce overfitting by controlling the margin's width and incorporating the kernel trick to transform data into a higherdimensional space where linear separation is more feasible.

It is based on the Support Vector Machine (SVM) algorithm, which is primarily used for the for classification. Here are its key features:

- **Non-Linear Regression**

SVR can capture non-linear relationships between features and the target variable by using appropriate kernel functions. This makes it versatile and applicable to a wide range of regression tasks.

- **Kernel Trick**

- SVR uses the kernel trick to transform the input features into a higher-dimensional space, allowing it to capture complex relationships between features and the target variable.

Common kernel functions include linear, polynomial, and radial basis function (RBF) kernels.

- **Margin Maximization**

Similar to SVM for classification, SVR aims to maximize the margin between the predicted values and the regression line, while still ensuring that a predefined fraction of training instances (support vectors) lies within a margin of tolerance.

- **Regularization**

SVR incorporates regularization parameters ( $C$  for regularization strength and  $\epsilon$  for the width of the epsilon-insensitive tube) to control the trade-off between model complexity and accuracy. This helps prevent overfitting and improves generalization ability.

- **Sparsity**

SVR typically uses only a subset of training instances (support vectors) to define the regression function. This results in a sparse solution, making SVR memory-efficient and computationally tractable, especially for large-scale regression problems.

- **Tuning Parameters**

SVR requires tuning of hyperparameters such as the choice of kernel function, kernel parameters (e.g., degree for polynomial kernel,  $\gamma$  for RBF kernel), and regularization parameters ( $C$  and  $\epsilon$ ). Cross-validation is often used to find the optimal values for these parameters. Overall, SVR is a powerful regression algorithm that is effective in capturing complex relationships in the data, especially in high-dimensional feature spaces with non-linear relationships. However, it may be sensitive to the choice of kernel function and the tuning of hyperparameters, requiring careful experimentation and validation. Common kernel functions include linear, polynomial, and radial basis function (RBF) kernels.

- **Margin Maximization**

Similar to SVM for classification, SVR aims to maximize the margin between the predicted values and the regression line, while still ensuring that a predefined fraction of training instances (support vectors) lies within a margin of tolerance.

- **Regularization**

SVR incorporates regularization parameters ( $C$  for regularization strength and epsilon for the width of the epsilon-insensitive tube) to control the trade-off between model complexity and accuracy. This helps prevent overfitting and improves generalization ability.

- **Sparsity**

SVR typically uses only a subset of training instances (support vectors) to define the regression function. This results in a sparse solution, making SVR memory-efficient and computationally tractable, especially for large-scale regression problems.

- **Tuning Parameters**

SVR requires tuning of hyperparameters such as the choice of kernel function, kernel parameters (e.g., degree for polynomial kernel, gamma for RBF kernel), and regularization parameters ( $C$  and epsilon). Cross-validation is often used to find the optimal values for these parameters.

Overall, SVR is a powerful regression algorithm that is effective in capturing complex relationships in the data, especially in high-dimensional feature spaces with non-linear relationships. However, it may be sensitive to the choice of kernel function and the tuning of hyperparameters, requiring careful experimentation and validation

## CHAPTER 4

### RESULTS ANALYSIS

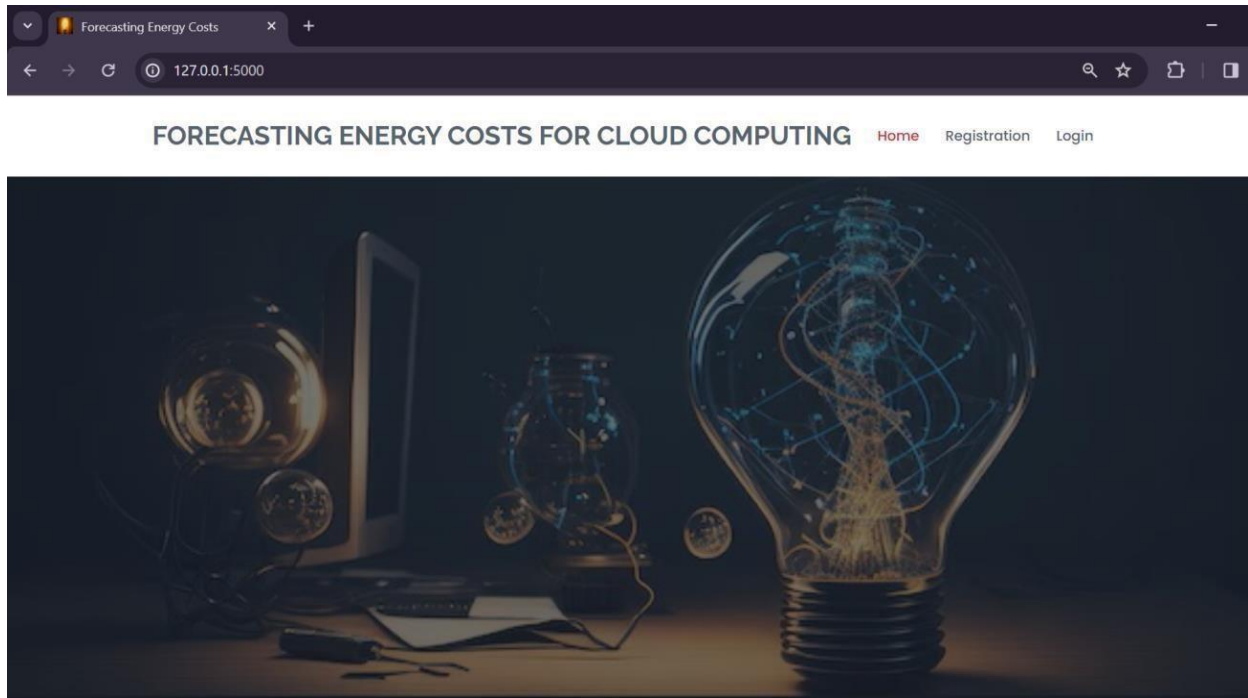


Fig 4.1 Home Page

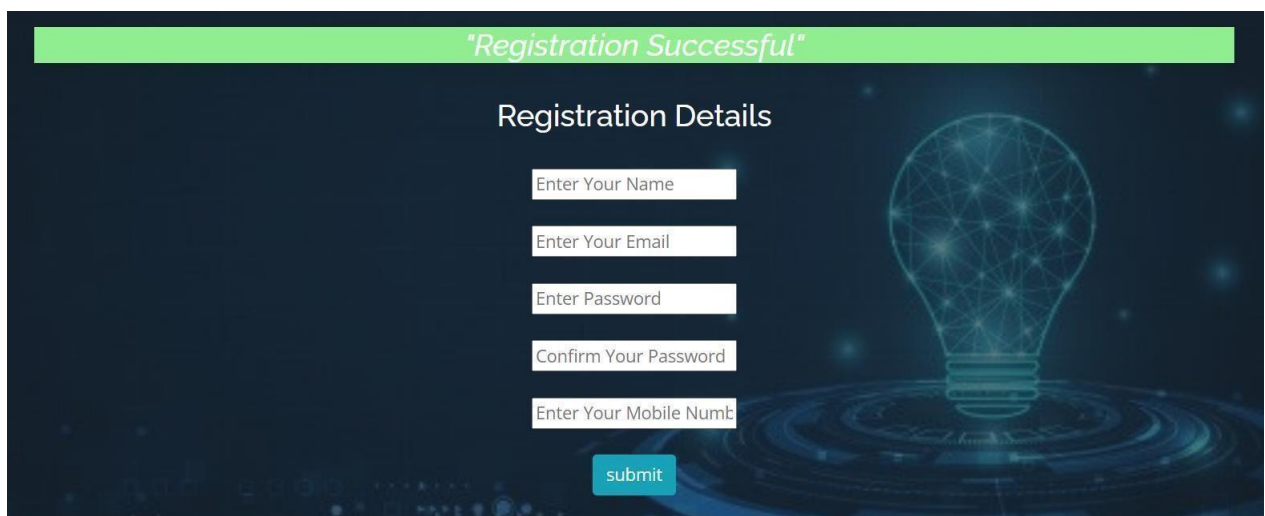
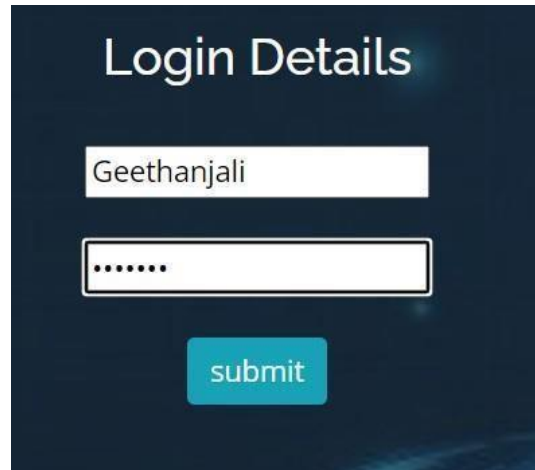


Fig 4.2 Registration Successful



A login form titled "Login Details" on a dark blue background. It features two white input fields: the first contains the text "Geethanjali", and the second contains seven dots representing a password. Below the fields is a teal "submit" button.

Fig 4.3 Login Page

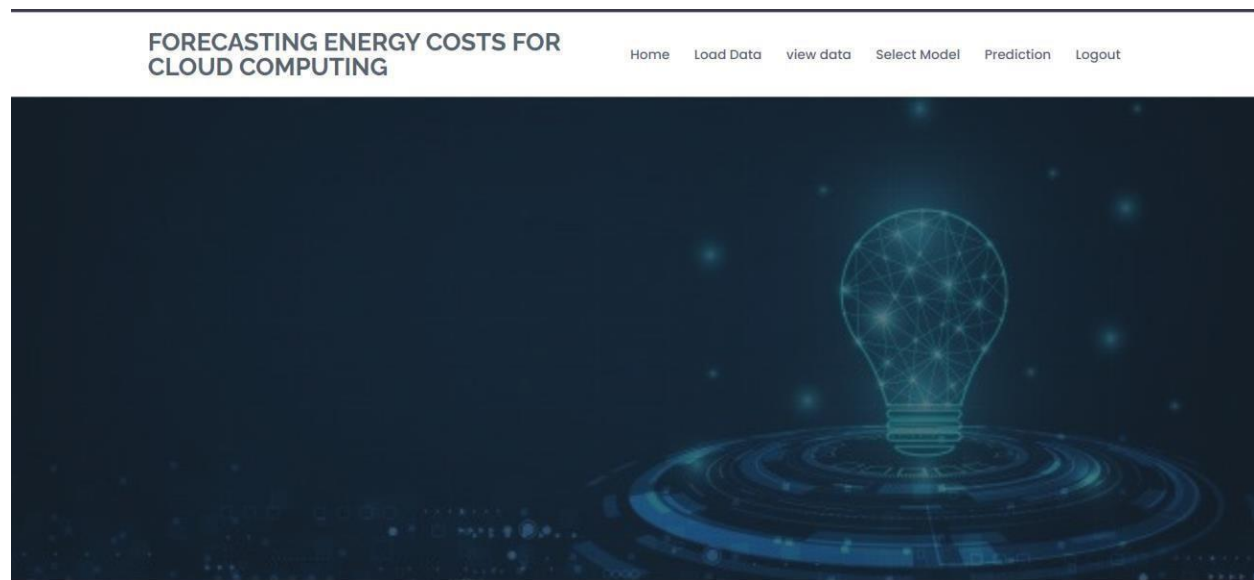


Fig 4.4 Post Login Screen



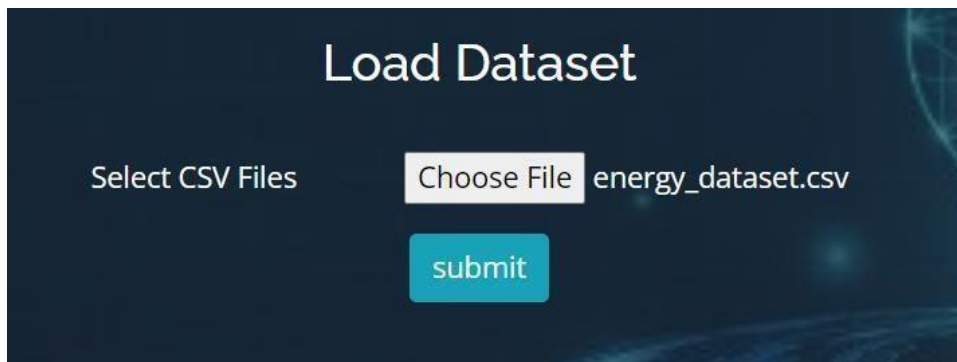


Fig 4.5 Load Dataset

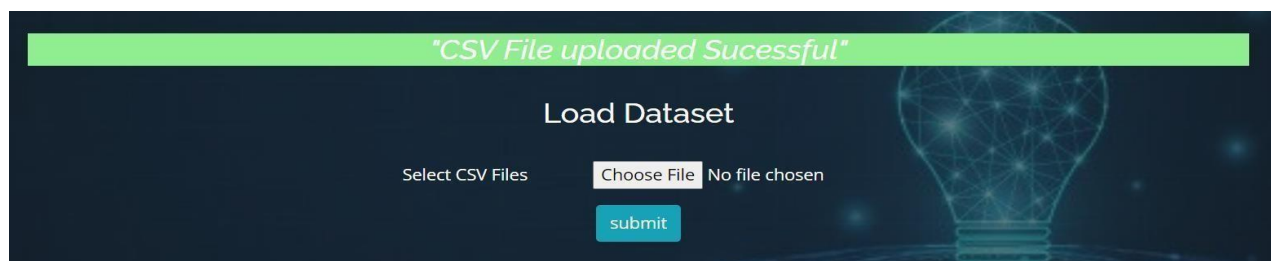


Fig 4.6 Dataset Uploaded Successfully

FORECASTING ENERGY COSTS FOR CLOUD COMPUTING						Home	Load Data	view data	Select Model	Prediction	Logout
S/N	generationfossilgas	generationfossilhardcoal	generationhydropumpedstorageconsumption	generationhydrowaterreservoir	generationoth						
1	4844	4821	863	1899	73						
2	5196	4755	920	1658	71						
3	4857	4581	1164	1371	73						
4	4314	4131	1503	779	75						
5	4130	3840	1826	720	74						
6	4038	3590	2109	743	74						
7	4040	3368	2108	848	74						
8	4030	3208	2031	1012	72						
9	4052	3335	2119	1015	73						
10	4137	3437	2170	1357	74						
11	4059	3516	2020	1817	72						
12	3931	3845	1183	1516	73						
13	3784	4220	972	1204	75						
14	3754	4404	922	1286	74						
15	3779	4256	941	1027	76						

Fig 4.7 View Data

## Model Selection

Select Testsize

Select Model

- Select an option
- Random Forest Regressor
- XGBoost Regressor
- Support Vector Regressor

Fig 4.8 Model Selection

**"R2 SCORE" OF OUR XGBOOST REGRESSOR model is 89.34**

### Model Selection

Select Testsize:

Select Model:

Fig 4.9 Score for XGBoost Regressor Model

### Prediction

Fig 4.10 Prediction

### Prediction

48
38
55
34
52
55
60
57
30

predict

Fig 4.11 Prediction Values

**PREDICTED PRICE IS [55.785]**

### Prediction

Generation fossil gas
Generation fossil hard
Generation hydro pump
Generation hydro water
Generation other rene

Fig 4.12 Predicted Price



Fig 4.13 Logout Page

## **CHAPTER 5**

### **CONCLUSION AND FUTURE SCOPE**

#### **5.1 Conclusion**

In conclusion, employing XGBoost for forecasting energy costs in cloud computing offers a robust and efficient solution. By leveraging the power of this advanced machine learning algorithm, organizations can accurately predict energy expenditures, enabling better resource allocation and cost optimization strategies. XGBoost's ability to handle complex data sets and nonlinear relationships makes it well-suited for modeling the intricate dynamics of energy consumption in cloud environments. Through meticulous feature engineering and hyperparameter tuning, XGBoost can capture subtle patterns and fluctuations in energy demand, resulting in more reliable forecasts. Additionally, its scalability ensures that it can handle large-scale datasets typically encountered in cloud computing environments. Furthermore, XGBoost provides interpretable insights into the factors influencing energy costs, empowering decision-makers to identify opportunities for efficiency improvements and cost savings. By integrating XGBoost-based forecasting into their operational workflows, businesses can make data-driven decisions that enhance sustainability, reduce environmental impact, and optimize financial performance.

#### **5.2 FUTURE SCOPE**

In this project, the primary emphasis on refining the XGBoost model for enhanced electricity forecasting precision is complemented by a diverse range of innovative initiatives. The incorporation of advanced machine learning techniques and real-time data integration signifies a significant stride towards achieving heightened predictive accuracy. Yet, the project's aspirations transcend predictive algorithms, extending towards addressing pivotal sustainability challenges within cloud data centers. The integration of renewable energy sources is not merely a feature but a strategic maneuver towards cultivating environmentally conscious computing practices. The project aims to seamlessly embed these sustainable energy solutions, contributing to a tangible reduction in the ecological footprint of cloud operations. Concurrently, the implementation of predictive maintenance strategies

underscores a dedication to operational efficiency, mitigating downtime and extending the operational lifespan of hardware components.

Furthermore, the exploration of energy-efficient hardware solutions constitutes a proactive step towards optimizing resource utilization and curbing operational expenditures. This holistic approach encapsulates the entire life cycle of cloud data center operations, with the intent of revolutionizing both the technological and environmental dimensions of computing.

The envisioned future of this project extends beyond numerical precision; it anticipates a paradigm shift in our approach to cloud computing. It is about establishing not merely advanced data centers but environmentally aware hubs of innovation that spearhead responsible and efficient computing practices. The amalgamation of cutting-edge technology with sustainability principles is set to redefine the landscape of cloud data centers, aligning them with the evolving expectations of a greener and more progressive digital era.

## REFERENCES

### 6.1 BOOK REFERENCES

1. Modeling and Forecasting Electricity Loads and Prices - A Statistical Approach by Rafal Weron
2. Energy Price Risk by Tom James
3. Forecasting Models of Electricity Prices by Javier Contreras (Edited)

### 6.2 WEBSITE REFERENCES

1. Electricity Price Forecasting –  
[https://en.m.wikipedia.org/wiki/Electricity\\_price\\_forecasting#:~:text=Electricity%20price%20forecasting%20is%20the,will%20be%20in%20the%20future](https://en.m.wikipedia.org/wiki/Electricity_price_forecasting#:~:text=Electricity%20price%20forecasting%20is%20the,will%20be%20in%20the%20future)
2. Electricity Price Volatility: How to trust AI based forecasts – <https://energy.n-side.com/blog/electricity-price-volatility-how-to-trust-ai-based-forecasts>

### 6.3 TECHNICAL PUBLICATION REFERENCES

- [1] "Electricity Price Forecasting for Cloud Computing Using a Hybrid Model of Wavelet Transform and Neural Networks," in IEEE Access, vol. 8, pp. 214947-214959, 2020.K. Tungpimolrut, S. Pholboon, and S. Kulsomboon
- [2] "An Electricity Price Forecasting Model for Cloud Computing based on Deep Belief Network and Long Short-Term Memory," in IEEE Access, vol. 7, pp. 126568-126576, 2019.X. Li, Q. Wang, and Y. Liu
- [3] "Electricity price and load fore-casting using enhanced convolutional neural network and enhanced support vector regression in smart grid". In: Electronics 8.2 (2019), p. 122Maheen Zahid et al.



## **APPENDIX**

### **7.1 SOFTWARE USED**

To forecast energy prices for cloud computing using XGBoost, the following software requirements are needed

- **Python**

XGBoost is implemented in Python, so, need to have Python installed on system. One can download Python from the official website (<https://www.python.org/>) and follow the installation instructions.

- **XGBoost**

Install the XGBoost library, which provides the implementation of gradient boosting algorithms, including XGBoost. One can install XGBoost using pip, a package manager for Python, by running the command `pip install xgboost`.

- **Data Analysis Libraries**

Utilize libraries such as Pandas and NumPy for data manipulation and analysis. These libraries provide powerful tools for handling datasets and performing data preprocessing tasks.

- **Scikit-learn**

Scikit-learn is a popular machine learning library in Python that provides various algorithms for classification, regression, clustering, and more. XGBoost can be easily integrated with scikit-learn for building and evaluating predictive models.

#### **SQLyog Enterprise**

This tool provides graphical user interface (GUI) for managing MySQL databases. We can use them to interact with databases, execute queries, and perform data analysis tasks related to energy price forecasting.

- **XAMPP**

XAMPP is a cross-platform web server solution package that includes Apache HTTP Server, MySQL database, and PHP scripting language. It's useful for hosting web applications and accessing MySQL databases locally.

- **Visual Studio Code (VS Code)**

VS Code is a lightweight yet powerful source code editor developed by Microsoft. It's highly customizable and supports various programming languages and extensions, making it suitable for writing Python code and analyzing data.

- **Chrome Browser**

The Chrome browser can be used for accessing webpages, online resources, and APIs relevant to energy price forecasting. It's also useful for viewing web-based data and dashboards.

## **7.2 METHODOLOGIES USED**

XGBoost, short for Extreme Gradient Boosting, is an ensemble learning algorithm based on decision trees. It works by sequentially combining multiple weak learners (decision trees) to create a strong predictive model. Here's a step-by-step explanation of how XGBoost works:

### **Initialization**

XGBoost initializes with an initial prediction value, typically the mean of the target variable for regression problems or the probability of the most frequent class for classification problems.

#### **1. Building Trees**

XGBoost sequentially builds decision trees, where each tree attempts to correct the errors made by the previous trees. It does this by fitting each tree on the residuals (the differences between the predicted values and the actual values) of the previous model.

## **2. Tree Construction**

Each decision tree in XGBoost is constructed using a greedy algorithm. It iteratively splits the data into partitions based on the features that result in the greatest reduction in the loss function, often using techniques like depth-first search.

## **3. Regularization**

XGBoost incorporates regularization techniques to prevent overfitting. Regularization penalties are applied to the complexity of the trees, such as L1 (Lasso) and L2 (Ridge) regularization on the weights of the tree nodes.

## **4. Gradient Boosting**

XGBoost employs gradient boosting, which is a gradient descent optimization algorithm. It minimizes a specific loss function by iteratively improving the model's predictions in the direction that reduces the loss the most.

## **5. Combining Trees**

XGBoost combines the predictions from all the trees to make the final prediction. The predictions from each tree are weighted based on their performance, with more accurate trees having higher weights.

## **6. Learning Rate**

XGBoost introduces a learning rate parameter that controls the contribution of each tree to the final prediction. A smaller learning rate makes the model more robust to overfitting but requires more trees to be added to the ensemble.

## **7. Stopping Criteria**

XGBoost uses stopping criteria to determine when to stop adding trees to the ensemble. This prevents overfitting and reduces computational costs by terminating tree construction when the model's performance no longer improves.

## 8. Parallelization

XGBoost supports parallelization, enabling efficient training on multicore CPUs. It splits the data and constructs trees in parallel to speed up the training process.

## 9. Regular Monitoring

During training, XGBoost monitors the performance on a separate validation dataset to detect overfitting and adjust hyperparameters accordingly. XGBoost combines the strengths of gradient boosting with regularized tree learning to create a robust and highly accurate predictive model, making it one of the most popular and effective algorithms for supervised learning tasks.

### 7.3 TESTING METHODS USED

**7.3.1 UNIT TEST.** It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

### 7.3.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### 7.3.3 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### 7.3.4 FUNCTIONAL TESTING

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. Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

## 8. CODE

### 8.1 app.py

```
import csv

import re

from flask import Flask, render_template, request, redirect, url_for, session from
werkzeug.utils import secure_filename import pymysql db =
pymysql.connect(host='localhost', port=3307, user='root', password='', db='electricity')
cursor = db.cursor()

from sklearn.model_selection import train_test_split from sklearn.ensemble import
RandomForestRegressor import xgboost as xgb from sklearn.svm import SVR from
sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score import os
import pandas as pd

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads'
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/load_data', methods=["POST", "GET"])
def load_data():
    if request.method == "POST":

        f = request.files['file']

        filetype = os.path.splitext(f.filename)[1]

if filetype == '.csv':
```

```

        mypath = os.path.join(app.config['UPLOAD_FOLDER'], f.filename)
        f.save(mypath)        df = pd.read_csv(mypath)
df.drop(['Unnamed: 0'], axis=1, inplace=True)

```

```

        s = mypath        sql =
"Truncate    table    tablename1"
cursor.execute(sql)
db.commit() sql = "INSERT INTO
tablename1    ("    for    col    in
df.columns.values:
        query = f"{col}, "        sql = sql +
query        sql = sql[:-2] + ") values (" + ('%s, ' *
10) sql = sql[:-2] + ")"    for row in
df.iterrows(): val    =    [str(row[1][i]) for
        i    in
range(len(row[1]))]        cursor.execute(sql,
tuple(val))        db.commit()

```

```

        return    render_template('load    data.html',
msg='success')    elif filetype != '.csv':
        return    render_template('load    data.html',
msg='invalid')    return render_template('load data.html')

```

```

@app.route('/view
data')    def view_data():

```

```

data = pd.read_sql_query('SELECT * FROM tablename1', db)
data.drop(['id'], axis=1,
inplace=True) data1 = data.head(100)

```

```

        return render_template('view_data.html', msg='data', data=data,
                                col_name=data1.columns.values, row_val=data1.values.tolist())

@app.route('/model', methods=["POST",
                                "GET"]) def model():
    if request.method == 'POST':

        selected =
        int(request.form['selected'])
        testsize =
        int(request.form['testing'])
        testsize = testsize / 100

        filename = os.listdir(app.config['UPLOAD_FOLDER']) df =
        pd.read_csv(os.path.join(app.config['UPLOAD_FOLDER'], filename[0]))

        df.drop("Unnamed: 0", axis=1, inplace=True)
        X =
        df.drop(['priceactual'], axis=1) y =
        df['priceactual']

        x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=testsize,
                                                                random_state=10)

    if (selected == 1):

        rfr = RandomForestRegressor(n_estimators=50,
                                    max_depth=14) model1 = rfr.fit(x_train, y_train) pred1 =
        model1.predict(x_test) score = r2_score(y_test, pred1)

```



```

return render_template('model.html', msg='accuracy', result=round(score, 4),
selected='RANDOM FOREST REGRESSOR') elif (selected == 2):
xgbr = xgb.XGBRegressor(learning_rate=0.4, n_estimators=200)

model2 = xgbr.fit(x_train, y_train) pred2 = model2.predict(x_test)
score = r2_score(y_test, pred2) return render_template('model.html',
msg='accuracy', result=round(score, 4), selected='XGBOOST REGRESSOR') elif
(selected == 3):
svr = SVR(kernel='rbf') x_train = x_train[:15000] y_train =
y_train[:15000] model3 = svr.fit(x_train[:15000], y_train[:15000]) pred3 =
model3.predict(x_test[:15000]) score = r2_score(y_test[:15000], pred3) return
render_template('model.html', msg='accuracy', result=round(score, 4),
selected='SUPPORT VECTOR REGRESSOR')

return render_template('model.html')

```

```

@app.route('/registration', methods=["POST",
"GET"]) def registration(): if
request.method == "POST": name =
request.form['name'] email =
request.form['email'] pwd =
request.form['pwd'] cpwd =
request.form['cpwd'] phno =
request.form['phno'] if pwd == cpwd:
sql = "select * from registration where name = '%s' and email='%s'" % (name,
email) a = cursor.execute(sql) if(a > 0):

return render_template('registration.html', msg='invalid') else:
sql = "insert into registration(name,email,pwd,phno) values
(%s,%s,%s,%s)" val = (name, email, pwd, phno)

```

```

cursor.execute(sql, val) db.commit() render_template('registration.html',return
msg='success') else:
    return render_template('registration.html',
msg='mismatch') return render_template('registration.html')

@app.route('/login', methods=["POST", "GET"]) def login(): if
request.method == 'POST': name = request.form['name'] pwd =
request.form['pwd'] sql = "select * from registration where name = '%s' and pwd='%s'" %
(name, pwd) a = cursor.execute(sql) if a > 0: return render_template('index1.html')
else:
    return render_template('login.html',
msg='invalid') return render_template('login.html')

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

@app.route('/index1') def index():
    return render_template('index1.html')

@app.route('/prediction', methods=["POST", 'GET']) def prediction(): if
request.method == "POST": gfg = request.form['gfg'] gfhc = request.form['gfhc']
ghpsc = request.form['ghpsc'] ghwr = request.form['ghwr'] gor = request.form['gor']
gw = request.form['gw'] tlf = request.form['tlf'] tla = request.form['tla'] time =
request.form['time'] values = [[float(gfg), float(gfhc), float(ghpsc), float(ghwr),
float(gor), float(gw), float(tlf), float(tla), float((time))]] xgbr =
xgb.XGBRegressor(learning_rate=0.4, n_estimators=200) model = xgbr.fit(x_train,

```

```

y_train) df_pred = pd.DataFrame(values[0], index=x_test.columns).transpose() pred
= model.predict(df_pred) return render_template('prediction.html', msg='success',
result=pred)    return render_template('prediction.html')

```

```

@app.route('/log
out')    def
logout():
    return render_template('logout.html')

```

```

@app.route('/view users') def view_users(): sql1 = "select *
from    registration"    cursor.execute(sql1)    data    =
cursor.fetchall() df = pd.read_sql_query(sql1, con=db) g =
re.split(',', data) d = pd.DataFrame(g, columns=['id',
'name', 'email', 'pwd', 'phno'])    return
render_template('view users.html', table=d) if __name__ == ('
main_'):
    app.run(debug=True)

```

**8.2 main.py** import pandas as pd import numpy as  
np import datetime import  
xgboost as xgb from sklearn.model\_selection  
import  
GridSearchCV from sklearn.svm import SVR

```

df
=
pd.read_csv('C:/Users/YMTS0356/Downloads/Datasets/electricity
price prediction/energy_dataset.csv') df.head()

```

```
df['price day ahead'] df.isnull().sum() x = df.drop(['price day ahead','generation
hydro pumped storage aggregated','forecast wind offshore eday ahead'],axis = 1)
x.shape y = df['priceactual']
```

```
x.isnull().sum()
```

```
x.info()
```

```
x.fillna(x.median(),inplace=True)
```

```
x.isnull().sum()
```

```
from sklearn.preprocessing import
LabelEncoder le = LabelEncoder() a =
le.fit_transform(x['time']) print(a) x.shape
x.drop(['time'],axis =
1,inplace=True) x['Time'] = a x.head()
x.drop(['generation fossil coal-derived gas','generation fossil oil shale','generation fossil
peat','generation geothermal','generation marine','generation wind
offshore'],axis = 1,inplace=True)
X
```

```
=x[['generationfossilgas','generationfossilhardcoal','generationhydropumpedstorage
consumption','generationhydrowaterreservoir','generationotherrenewable','generationwaste',
'totalloadforecast','totalloadactual','Time','priceactual']]
X.to_csv('cleaned_data.csv') from sklearn.ensemble import
RandomForestRegressor rfr = RandomForestRegressor() from
sklearn.model_selection import train_test_split x_train,x_test,y_train,y_test =
train_test_split(X,y,test_size=0.3,random_state=0) model = rfr.fit(x_train,y_train)
```

```

from sklearn.metrics import
mean_absolute_error,mean_squared_error,r2_score
pred =
model.predict(x_test) print(pred) mean_squared_error(y_test,pred)
model.score(x_test,y_test) mean_absolute_error(y_test,pred)

```

```

r2_score(y_test,pred)

```

```

xgbr = xgb.XGBRegressor(learning_rate=0.1, max_depth=3,
n_estimators=25) model2 = xgbr.fit(x_train, y_train) pred2 =
model2.predict(x_test) score = r2_score(y_test, pred2) a =
model2.feature_importances_ pd.Series(a)

```

```

parameters = [{'kernel': ['rbf','poly'], 'gamma': [1e-4, 1e-3, 0.01, 0.1, 0.2],'C': [1, 10,
100]}] c = GridSearchCV(SVR(),param_grid=parameters,cv=5)
c.fit(x_train,y_train) print(c.best_params_) d = [4,5,6]
e = ['a','b','c'] f={} for i in range(len(e)
): g=e[i]

```

```

f.append(g,d[i])
print(f)

```

### 8.3 index.html

```

<!DOCTYPE html>

```

```

<html lang="en">

```

```

<head>

```

```

<meta charset="utf-8">

```

```

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

<title>Forecasting Energy Costs</title>

<meta content="" name="description">

<meta content="" name="keywords">


<!-- Favicons -->

<link href="static/assets/img/logo.jpg" rel="icon">

<link href="static/assets/img/icon.jpg" rel="apple-touch-icon">


<!-- Google Fonts -->

<link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,7
00,700i|Raleway:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,
500,500i,600,600i,700,700i" rel="stylesheet">


<!-- Vendor CSS Files -->

<link href="static/assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
<link href="static/assets/vendor/icofont/icofont.min.css" rel="stylesheet">

<link href="static/assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">

<link href="static/assets/vendor/animate.css/animate.min.css" rel="stylesheet">

<link href="static/assets/vendor/remixicon/remixicon.css" rel="stylesheet">

<link href="static/assets/vendor/venobox/venobox.css" rel="stylesheet">

```

```
<link href="static/assets/vendor/owl.carousel/assets/owl.carousel.min.css"
rel="stylesheet">
```

```
<!-- Template Main CSS File -->
```

```
<link href="static/assets/css/style.css" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<!-- ===== Header ===== -->
```

```
<header id="header" class="fixed-top">
```

```
<div class="container d-flex align-items-center">
```

```
<h1 class="logo"><a href="index.html">forecasting energy costs for
cloud computing</a></h1>
<a href="index.html" class="logo"></a>
<nav class="nav-menu d-none d-lg-block">
```

```
<ul>
```

```
<li class="active"><a href="/">Home</a></li>
```

```

        <li><a href="/registration">Registration</a></li>

        <li><a href="/login">Login</a></li>

    </ul>

</nav>

</div>

</header>

<!-- ===== Hero Section ===== -->

<section id="hero">

    <div id="heroCarousel" class="carousel slide carousel-fade" data-ride="carousel">

        <ol class="carousel-indicators" id="hero-carousel-indicators"></ol>

        <div class="carousel-inner" role="listbox">

            <!-- Slide 1 -->
            <div class="carousel-item active" style="background-image:
            url(static/assets/img/slide/image2.jpg)">
                <div class="carousel-container">

<div class="container">

                </div>

```



</div>

</div>

</div>

</div>

</section>

<!-- Vendor JS Files -->

<script src="static/assets/vendor/jquery/jquery.min.js"></script>

<script src="static/assets/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<script src="static/assets/vendor/jquery.easing/jquery.easing.min.js"></script>

<script src="static/assets/vendor/php-email-form/validate.js"></script>

<script src="static/assets/vendor/isotope-layout/isotope.pkgd.min.js"></script>

<script src="static/assets/vendor/venobox/venobox.min.js"></script>

<script src="static/assets/vendor/waypoints/jquery.waypoints.min.js"></script>

<script src="static/assets/vendor/owl.carousel/owl.carousel.min.js"></script>

<!-- Template Main JS File -->

<script src="static/assets/js/main.js"></script>

</body>

</html>

