DEVELOPMENT OF A PREDICTION SYSTEM FOR HOUSEHOLD ELECTRICITY CONSUMPTION IN DHAKA CITY

MD. MEHEDI HASAN MUAZ (SN. 0418140018)

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY
DHAKA, BANGLADESH

MARCH 2024

DEVELOPMENT OF A PREDICTION SYSTEM FOR HOUSEHOLD ELECTRICITY CONSUMPTION IN DHAKA CITY

M. Engineering Thesis

By

Md. Mehedi Hasan Muaz (0418140018)

Approved as to style and content by the Board of Examination on 31 December, 2023:

Dr. Md Akhtaruzzaman Assistant Professor of Computer Science and Engineering MIST, Dhaka	Chairman (Supervisor) Board of Examination
Dr. Muhammad Nomani Kabir Associate Professor of Computer Science and Engineering UIU, Dhaka	Member (External) Board of Examination
Dr. Nusrat Sharmin Assistant Professor of Computer Science and Engineering MIST, Dhaka	Member (Internal) Board of Examination

Department of Computer Science and Engineering MIST, Dhaka

DEVELOPMENT OF A PREDICTION SYSTEM FOR HOUSEHOLD ELECTRICITY CONSUMPTION IN DHAKA CITY

DECLARATION

I hereby declare that the study reported in this project entitled above is my own original work and has not been submitted before anywhere for any degree or other purposes. Further I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this project and sources have been acknowledged and/or cited in the reference section.

Md. Mehedi Hasan Muaz

Department of Computer Science and Engineering MIST, Dhaka

DEVELOPMENT OF A PREDICTION SYSTEM FOR HOUSEHOLD ELECTRICITY CONSUMPTION IN DHAKA CITY

A Thesis

By

Md. Mehedi Hasan Muaz

DEDICATION

Dedicated to my wife and my parents for always supporting and encouraging me

ACKNOWLEDGEMENT

First and foremost, I would like to praise Allah the Almighty, the Most Gracious, and the Most Merciful for His blessings given to me during my study and in completing this thesis. May Allah's blessing goes to His final Prophet Muhammad Sallallahu-alaihe-wasallam, his family and his companions.

He is thankful to his research supervisor, Assistant Professor Dr. M. Akhtaruzzaman, Department of Computer Science and Engineering, Military Institute of Science and Technology (MIST), for providing patient guidance, enthusiastic encouragement and useful critiques throughout this project.

The author also acknowledges the support of Department of CSE, Military Institute of Science and Technology (MIST) for successful completion of this project. The author gratefully thanks Dr. Md. Mahabubur Rahman and Lt Col Dr. Muhammad Narzul Islam for their valuable advice, criticism, and overall support. The author also expressed his gratitude to Dhaka Electric Supply Company Ltd (DESCO) for providing necessary data for this project.

Especial thanks goes to the Quantum Robotics and Automation Research Group (QRARG) and R&D section of DAZDREAM Technologies for providing technical support and insightful suggestions.

The author would like to express his sincere gratitude to his family, friends, and others who helped him, directly or indirectly for their unwavering support and patience as he worked on this project.

ABSTRACT

Development of a Prediction System for Household Electricity Consumption in Dhaka City

In today's modern world, electricity plays a crucial role in everyday life, serving as an essential resource for various activities. With uninterrupted access to electricity being a top priority for nations worldwide, accurately forecasting electricity consumption has become increasingly important. Such forecasts are instrumental in enhancing the reliability and efficiency of power supply infrastructure. A review of the relevant studies shows that, currently there is no application with which the electricity consumption of Dhaka city can be predicted. In order to fill in the need for such an application, the primary objective of this research was - conducting performance analysis among state of the art machine learning algorithms in terms of predicting power. In order to accomplish this objective, consumption data of a group of consumers from Dhaka city (specifically in Kafrul, Kallyanpur, Pallabi, Monipur and Rupnagar area) was collected for a period of three years. Then the data was used to train machine learning models for predicting power consumption with Artificial Neural Network, Support Vector Machine, XGBoost, LightGBM and CatBoost. In these models, features like monthly consumption (kilowatt-hour) for two years, sanctioned load, meter type, etc. had been selected to predict the electricity consumption of next one year. Grid search technique had been deployed to find the appropriate hyper-parameter values for each algorithm. Afterwards, the performance of each of the proposed predictive models was measured to determine their effectiveness by calculating MAE, RMSE, R2 values for each algorithm. Experimental results present that Artificial Neural Network (ANN) and Support Vector Machine (SVM) shows more accurate result compared to the other algorithms as per MAE, RMSE, and R squared method. For example, the MAE for ANN and SVM were 49.19 and 51.08 respectively, where the MAE for XGBoost, LightGBM and CatBoost were 127.22, 165.62 and 70.81 respectively. However, in terms of time required to train the model, LightGBM and CatBoost stands out among all these algorithms. The secondary objective of this research was to develop a web application using the best performing model to predict the electricity consumption of the consumers of Dhaka city. To achieve this objective, a web application had been developed using the model trained

with Artificial Neural Network (ANN) where the users can put the electricity consumption of any consumer during two years and predict the electricity consumption for the next one year. Output of the experiment shows that adopting the methods may greatly benefit power generation, transmission and distribution entities as well as the government in forecasting energy needs and plan accordingly. The methods used in this work can also be utilized to reduce system loss and prevent equipment damage to a great extent.

সারসংক্ষেপ

Development of a Prediction System for Household Electricity Consumption in Dhaka City

বর্তমানে আধুনিক জীবনযাত্রা বিভিন্ন ক্ষেত্রে বিদ্যুৎ একটি গুরুত্বপূর্ণ ভূমিকা পালন করে। বিশ্বের প্রতিটি দেশেই নিরবচ্ছিন্ন বিদ্যুৎ সরবরাহকে অধিক গুরুত্ব প্রদান করার কারণে গ্রাহকপ্রান্তে বিদ্যুৎ ব্যবহারের পরিমাণ পূর্ব থেকেই ধারণা করা অতীব জরুরী হয়ে পরছে। বিদ্যুৎ বিতরণ ব্যবস্থার নির্ভরযোগ্যতা ও সক্ষমতা বৃদ্ধিতে এ ধরনের পদক্ষেপ অপরিহার্য ভূমিকা পালন করবে। এ সম্পর্কিত বিভিন্ন গবেষণাপত্র পর্যালোচনা করে দেখা যায়, বর্তমানে ঢাকা শহরের বিদ্যুৎ ব্যবহারের পূর্ভাবাস বের করার মত কোন অ্যাপ্লিকেশন নেই। এই ঘাটতি পূরণ করতে এই গবেষণার প্রধান লক্ষ্য ছিল – কিছু আধুনিক মেশিন লার্নিং অ্যালগরিদম ব্যবহার করে বিদ্যুৎ ব্যবহারের পরিমাণ অনুমান করা এবং এবং অ্যালগরিদমগুলোর কার্যকারিতা যাচাই করা। এই লক্ষ্য বাস্তবায়নের নিমিত্তে ঢাকা শহরের (বিশেষত কাফরুল, কল্যাণপুর, পল্লবী, মনিপুর, রূপনগর এলাকার) নির্দিষ্ট সংখ্যক গ্রাহকের ৩ বছরের বিদ্যুৎ ব্যবহারের তথ্য সংগ্রহ করা হয়েছে। পরবর্তীতে এই ডাটা ব্যবহার করে পাঁচটি ভিন্ন ভিন্ন আধুনিক মেশিন লার্নিং অ্যালগরিদম (Artificial Neural Network, Support Vector Machine, XGBoost, LightGBM এবং CatBoost) ব্যবহার করে পাঁচটি মডেল তৈরী করা হয়। এই মডেলগুলোতে দুই বছরের মাসিক বিদ্যুৎ ব্যবহারের পরিমাণ, অনুমোদিত লোড, মিটারের ধরণ প্রভৃতি ব্যবহার করে পরবর্তী এক বছরের মাসিক বিদ্যুৎ ব্যবহারের পরিমাণ অনুমান করা হয়। প্রতিটি অ্যালগরিদমের জন্য সঠিক hyper-parameter বের করতে grid search পদ্ধতি ব্যবহার করা হয়েছে। অতপরঃ প্রত্যেকটি মডেলের MAE, RMSE, R2 বের করার মাধ্যমে তাদের কার্যকারিতার তুলনামূলক বিশ্লেষণ করা হয়। কার্যকারিতা নিরীক্ষার মাধ্যমে দেখা যায়, Artificial Neural Network (ANN) এবং Support Vector Machine (SVM) অ্যালগরিমের মাধ্যমে তৈরী করা মডেলগুলো MAE, RMSE, এবং R squared স্কোরে অন্য মডেলগুলোর চেয়ে এগিয়ে আছে। উদাহরণস্বরূপ, ANN এবং SVM এর MAE হচ্ছে যথাক্রমে ৪৯.১৯ ও ৫১.০৮, যেখানে XGBoost, LightGBM এবং CatBoost এর MAE যথাক্রমে ১২৭.২২, ১৬৫.৬২ এবং ৭০.৮১। তবে মডেলগুলোর প্রশিক্ষণের সময় বিবেচনায় LightGBM এবং CatBoost অ্যালগরিদম অন্যগুলোর চেয়ে এগিয়ে থাকে। এই গবেষণাকর্মের আরেকটি লক্ষ্য ছিল সবচেয়ে কার্যকর অ্যালগরিদমটি দিয়ে একটি ওয়েব অ্যাপ্লিকেশন তৈরী করা, যার মাধ্যমে ঢাকা শহরের বিদ্যুৎ গ্রাহকদের বিদ্যুৎ ব্যবহার অনুমান করতে পারবে। এই লক্ষ্যে Artificial Neural Network (ANN) এর মডেলটি ব্যবহার করে একটি ওয়েব অ্যাপ্লিকেশন তৈরী করা হয়, যেখানে ব্যবহারকারীরা কোন বিদ্যুৎ গ্রাহকের দুই বছরের মাসিক বিদ্যুৎ ব্যবহারের তথ্য প্রদান করে পরবর্তী এক বছরের মাসিক বিদ্যুৎ ব্যবহার অনুমান করতে পারেন। উক্ত নিরীক্ষার মাধ্যমে প্রতীয়মান হয় যে, এই পদ্ধতির মাধ্যমে বিদ্যুৎ ব্যবহারের পূর্বাভাস জানা গেলে

তা বিদ্যুৎ উৎপাদন, সঞ্চালন, ও বিতরণ সংস্থাসমুহের পাশাপাশি সরকারকেও বিদ্যুতের চাহিদা অনুযায়ী পরিকল্পনা প্রণয়নে সাহায্য করবে। এছাড়াও, এই গবেষণাকর্মে ব্যবহৃত পদ্ধতিসমূহ সিস্টেম লস কমাতে এবং যন্ত্রপাতি নষ্ট হওয়া রোধ করতে অনেকাংশে ভূমিকা রাখবে। এই গবেষণাপত্রে Artificial Neural Network, Support Vector Machine, XGBoost, LightGBM এবং CatBoost অ্যালগরিদমের মাধ্যমে ঢাকা শহরের (বিশেষত কাফরুল, কল্যাণপুর, পল্লবী, মনিপুর, রূপনগর এলাকার) নির্দিষ্ট সংখ্যক গ্রাহকের বিদ্যুৎ ব্যবহারের পরিমাণ ধারণা করা হয়েছে। এই মডেলগুলোতে ইনপুট হিসাবে তিন বছরের মাসিক বিদ্যুৎ ব্যবহারের তথ্য, অনুমোদিত লোড, মিটারের ধরন, ইত্যাদি ব্যবহার করার মাধ্যমে গ্রাহকদের পরবর্তী এক বছরের বিদ্যুৎ ব্যবহার নির্ণয় করা হয়েছে। অতঃপর, প্রতিটি মডেলের MAE, RMSE, R2 নির্ণয় করার মাধ্যমে অ্যালগরিদমগুলোর কার্যকারিতা দেখানো হয়েছে। কার্যকারিতা নিরীক্ষার মাধ্যমে দেখা যায়, Artificial Neural Network (ANN) এবং Support Vector Machine (SVM) অ্যালগরিমের মাধ্যমে তৈরী করা মডেলগুলো MAE, RMSE, এবং R squared স্কোরে অন্য মডেলগুলোর চেয়ে এগিয়ে আছে। তবে মডেলগুলো প্রশিক্ষণের সময় বিবেচনায় LightGBM এবং CatBoost অ্যালগরিদম অন্যগুলোর চেয়ে এগিয়ে থাকে। উক্ত নিরীক্ষার মাধ্যমে প্রতীয়মান হয় যে, এই পদ্ধতির মাধ্যমে বিদ্যুৎ ব্যবহারের পূর্বাভাস জানা গেলে তা বিদ্যুৎ উৎপাদন, সঞ্চালন, ও বিতরণ সংস্থাসমুহের পাশাপাশি সরকারকেও বিদ্যুতের চাহিদা অনুযায়ী পরিকল্পনা প্রণয়নে সাহায্য করবে। এছাড়াও, এই গবেষণাকর্মে ব্যবহৃত পদ্ধতিসমূহ সিস্টেম লস কমাতে এবং যন্ত্রপাতি নষ্ট হওয়া রোধ করতে অনেকাংশে ভূমিকা রাখবে।

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Electricity has long been an indispensable part of the modern civilization. Nowadays, ensuring uninterrupted electricity has become a priority for every country. Therefore, forecasting electricity consumption can help improving the power supply infrastructure. This study has focused on comparing power consumption prediction of a group of consumers with five machine learning algorithms. Namely, Artificial Neural Network, Support Vector Machine, XGBoost, LightGBM and CatBoost. In those models, features like monthly consumption (kilowatt-hour) for three years, sanctioned load (kw), meter type (1P/3P), tariff, etc. were used to predict the power consumption of another year forward. Grid search technique was deployed to find the appropriate hyper parameter values for each algorithm. Afterwards, the performance of each of the proposed predictive models was measured to determine their effectiveness by calculating MAE, RMSE, R2 values for each algorithm. Experimental results present that ANN and SVM shows more accurate result compared to the other algorithms as per MAE, RMSE and R2 score. However, LightGBM and CatBoost takes considerably less time while training the model compared to other three algorithms. The output of this experiment shows that adopting the methods may greatly benefit the government and other utilities in forecasting energy needs and plan accordingly. The methods used in this work can also be utilized to reduce system loss and prevent equipment damage to a great extent.

1.1 Background of the Study

Uninterrupted quality power supply can significantly contribute to improving the economy of any country. Maintaining Power Quality Indices (Watson and Miller, 2015) is a challenging task for power distributors which can become even harder with fluctuating demand patterns. Therefore, concerned entities have to forecast the demand of electricity all the time in order to facilitate generation, transmission, and distribution of sufficient amount of electricity. Failure to forecast power demand may cause catastrophic consequences like Grid Failure (CNN, 2022). Moreover, it becomes necessary to speculate power consumption if the electric meter is damaged or meter reading is unavailable.

1.1.1 Necessity of Power Consumption Forecasting

Forecasting power consumption is crucial for several reasons (Fig. 1.1):

1. Resource Planning: Accurate forecasts enable the government, power suppliers and grid operators to plan and allocate resources effectively. By predicting future demand, they can ensure an adequate supply of electricity, preventing shortages or overproduction.
2. Grid Stability: Maintaining a stable power grid is essential for preventing blackouts and ensuring reliable electricity delivery. Forecasting consumption helps operators anticipate fluctuations in demand, allowing them to adjust generation and distribution accordingly to maintain grid stability.
3. Energy Efficiency: Understanding consumption patterns over time allows for the identification of peak usage periods and trends. This information can be used to implement energy-saving measures and encourage consumers to adjust their usage habits, promoting overall energy efficiency.
4. Renewable Integration: With the increasing adoption of renewable energy sources like solar and wind, accurate consumption forecasts are essential for integrating these intermittent resources into the grid effectively. By aligning renewable generation with predicted demand, operators can optimize energy production and minimize reliance on traditional fossil fuel-based generation.
5. Nuclear Power Integration: The frequency of the power grid needs to stable when nuclear power is integrated in the grid. Otherwise grid failure can occur. To maintain a stable frequency, balance between demand and supply has to be ensured. Power demand forecast can help maintain the balance.
6. Cost Management: Forecasting consumption helps utilities manage costs associated with electricity generation, transmission, and distribution. By accurately predicting demand, they can optimize resource allocation and avoid unnecessary expenditures, ultimately leading to cost savings for both utilities and consumers.

Overall, forecasting power consumption plays a crucial role in ensuring the reliability, efficiency, and sustainability of electricity systems.

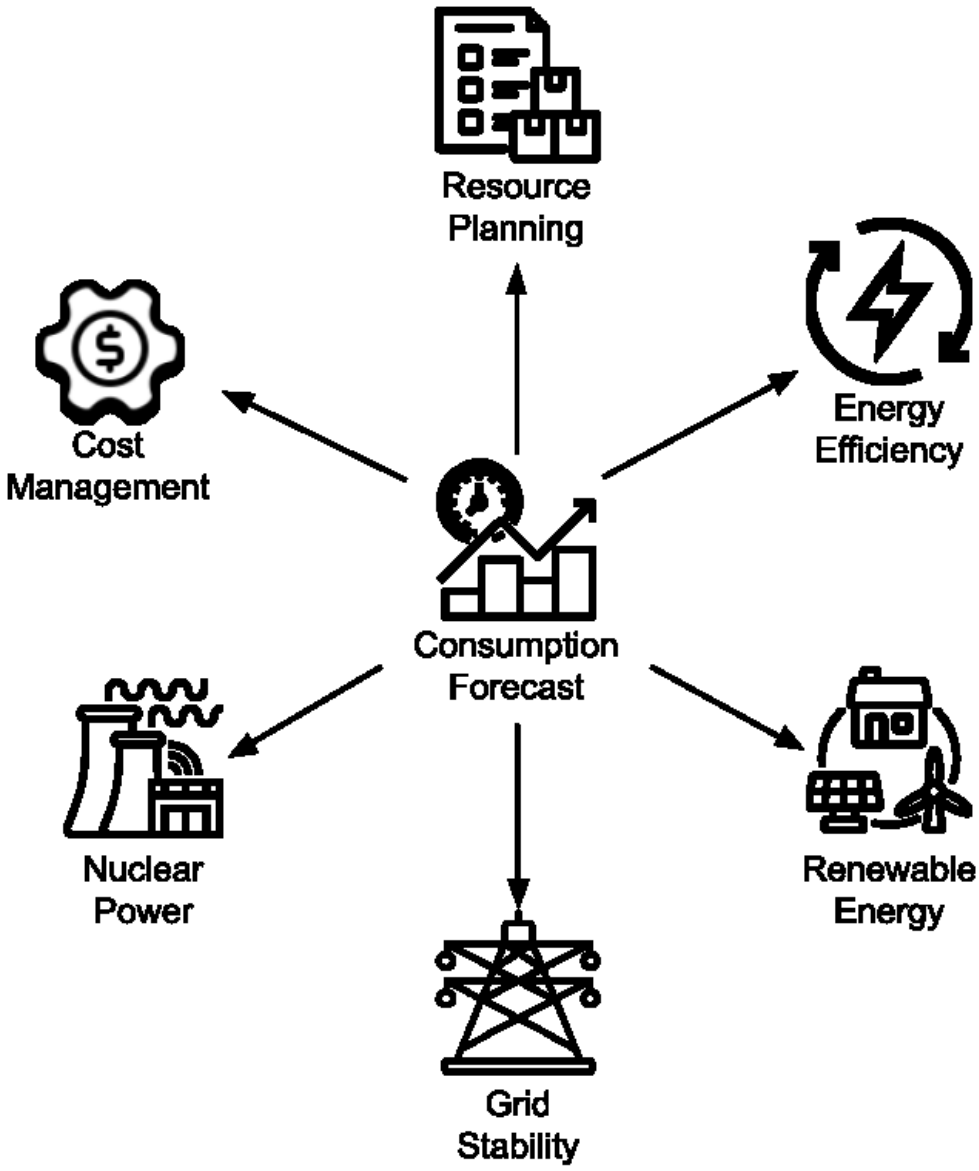

Fig. 1.1: Importance of power consumption forecast

1.2 Problem Statement

In order to predict the electricity consumption for policy making, ensuring preparedness and various other reasons, developing an acceptable model has become necessary in the context of Bangladesh. Therefore, several models need to be developed widely accepted algorithms for power consumption prediction and the performance of those models need to be compared. A forecast application for consumption prediction also needs to be developed with the best performing model.

1.3 Research Objectives

The research objectives can be described as follows:

a. To conduct performance analysis among state of the art machine learning algorithms in terms of predicting power consumption.
b. To develop a web application using the best performing model to predict electricity consumption of users.

1.4 Methodology

The whole process can be divided into a few steps, data collection, data pre-processing, predictive analysis with ANN, SVM, XGBoost, LightGBM and CatBoost, then, performance comparison with MAE, RMSE and R2. Finally, web application development using the best performing model. The following steps were followed throughout the research work in order to achieve the objectives. The process is also presented in Fig. 1.2.

1. Selection of problem statement
2. Set specific objectives for the research
3. Collect power consumption data from DESCO
4. Apply proper feature engineering (pre-processing) to prepare the dataset for building forecast models
5. Develop machine learning models for predicting power consumption
6. Evaluate the performance of different models using one or more regression performance metrics and compare them
7. Develop a web application for predicting power consumption of individual consumers with the best performing model

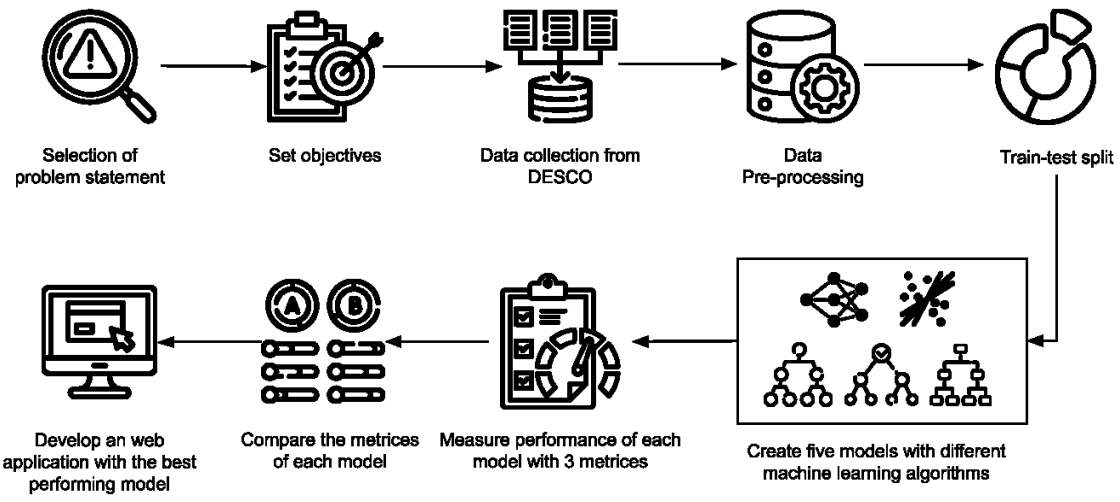

Fig. 1.2: Methodology

1.4.1 Data Collection

Power consumption data of a group of consumers from Dhaka city (specifically Kafrul, Kallyanpur, Pallabi, Monipur and Rupnagar) was collected for a period of four years from Dhaka Electric Supply Company Ltd. (DESCO). The data was collected by submitting a request for electricity consumption data to the office of managing director, DESCO.

1.4.2 Data Pre-processing

The steps which have been followed for data pre-processing are shown below:

- Feature scaling of numeric data: Numeric values were normalized to eliminate bias and to obtain better accuracy (Wan, 2019).
- Missing value management: To fill up the missing values in the dataset, k-nearest neighbor algorithm (Kaiser, 2014) has been used in this study.
- Removal of redundant features: Using Pearson correlation coefficient (Mujahid, 2017), redundant features having similar effect on the algorithm were reduced to improve execution time.
- Division of dataset: After completion of pre-processing, the whole dataset was divided into Training set and test set in 80:20 ratio.

To find the model with the best accuracy, in this case, the regressor variants of five machine learning algorithms and three evaluation methods were selected. After evaluating all the

algorithms with all the evaluation metrics, the algorithms were chosen to develop the final application.

1.4.3 Predictive Analysis with ML Algorithms

At this stage of the study, five machine learning algorithms were chosen. Among those, two algorithms are basic machine learning algorithms and rest three are advanced machine learning algorithms.

1.4.3.1 Basic Machine Learning Algorithms

a. Artificial Neural Network: Artificial neural networks (ANNs) are computational models inspired by the structure and functioning of biological neural networks in the human brain. These networks consist of interconnected nodes, or artificial neurons, organized in layers, including an input layer, one or more hidden layers, and an output layer. Through a process called training, ANNs learn to perform tasks by adjusting the strengths of connections between neurons, known as weights, often utilizing algorithms like backpropagation to minimize prediction errors (Fig. 1.3). Capable of learning complex patterns and relationships from large datasets, ANNs are powerful tools for tasks such as classification, regression, and pattern recognition. ANNs have found applications across diverse domains, including image and speech recognition, natural language processing, medical diagnosis, and autonomous vehicles. Despite their effectiveness, challenges such as overfitting persist, prompting ongoing research into architectures, optimization techniques, and interpretability. With advancements in hardware and software, artificial neural networks continue to drive innovations in artificial intelligence and machine learning applications.

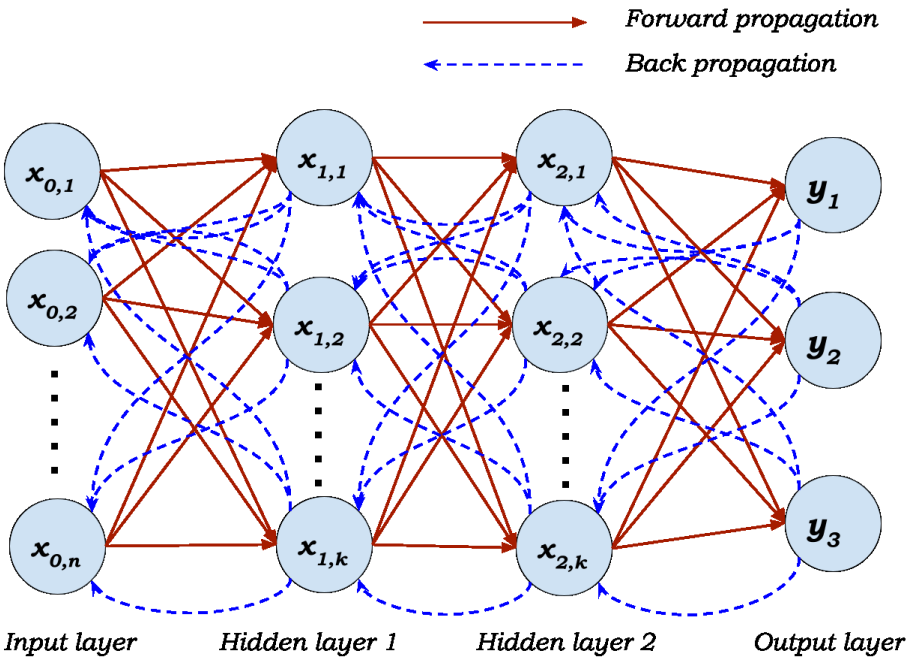

Fig. 1.3: Artificial Neural Network (ANN)

b. Support Vector Machine (SVM): Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates the different classes in the input data space (Fig. 1.4). The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, known as support vectors. SVM can handle both linear and nonlinear classification problems through the use of kernel functions, which transform the input data into higher-dimensional feature spaces where a linear separation may be possible. SVM aims to achieve the best generalization performance by maximizing the margin while minimizing classification errors. It is robust to overfitting, particularly when using a large margin or regularization parameters. SVM has found wide applications in various domains such as image classification, text categorization, and bioinformatics. Despite its effectiveness, SVM can be computationally expensive, particularly with large datasets, due to the need for solving a quadratic optimization problem. Proper selection of hyperparameters, such as the choice of kernel function and regularization parameter, is crucial for obtaining optimal performance with SVM. Overall, SVM is a powerful and versatile algorithm for binary and multi-

class classification tasks, valued for its ability to handle complex datasets with high-dimensional feature spaces.

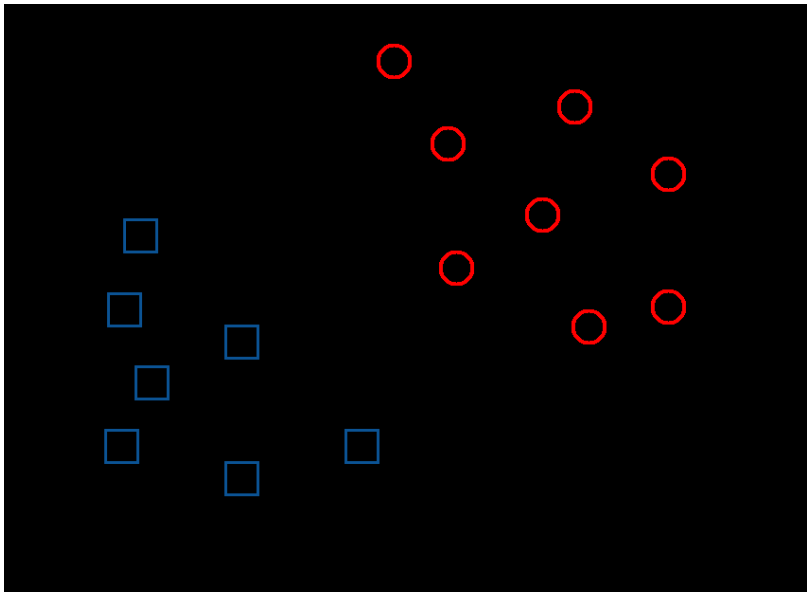

Fig. 1.4: Support Vector Machine (SVM)

1.4.3.2 Advanced Machine Learning Algorithms

a. **Extreme Gradient Boosting (XGBoost):** It is an open-source regularizing gradient-boosting framework that works in multiple languages. It is fast, scalable, and portable with good accuracy compared to other algorithms. XGBoost is an advanced implementation of gradient boosting algorithms used for supervised learning tasks, primarily in classification and regression. It is renowned for its speed, efficiency, and high performance in competitions and real-world applications. XGBoost builds a series of decision trees sequentially, with each subsequent tree aiming to correct the errors made by the previous ones. The model minimizes a loss function while also penalizing the complexity of the trees to prevent overfitting. XGBoost employs regularization techniques such as L1 and L2 regularization to control model complexity and improve generalization. It also supports parallel and distributed computing, making it scalable and capable of handling large datasets. Feature importance analysis is straightforward with XGBoost, as it provides insights into the relative importance of each feature in predicting the target variable. Additionally, XGBoost offers flexibility in handling

missing data and can automatically handle sparse data matrices. With its robustness, speed, and ability to handle diverse data types, XGBoost has become a popular choice in both academia and industry for various machine learning tasks. Its rich set of hyperparameters allows for fine-tuning to achieve optimal performance, making it a valuable tool in the data scientist's toolkit.

b. **Light Gradient Boosting Machine (LightGBM):** LightGBM, or Light Gradient Boosting Machine, is a gradient boosting framework developed by Microsoft that focuses on speed and efficiency. It is designed for distributed and efficient training of large-scale datasets. LightGBM introduces novel techniques such as Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to enhance training speed and reduce memory usage. Unlike traditional gradient boosting algorithms, LightGBM grows trees leaf-wise instead of level-wise, which can lead to faster convergence and reduced computational costs. It also implements histogram-based algorithms for computing feature splits, further boosting efficiency. LightGBM supports both classification and regression tasks and offers various objective functions and evaluation metrics to cater to different problem types. The framework allows for parallel and GPU acceleration, making it suitable for training models on large datasets. Additionally, LightGBM provides advanced regularization techniques to prevent overfitting and improve generalization performance. With its speed, scalability, and comprehensive feature set, LightGBM has gained popularity among data scientists and machine learning practitioners for its ability to deliver state-of-the-art performance across a wide range of applications.

c. **CatBoost**: CatBoost is a gradient boosting library developed by Yandex that specializes in handling categorical features effectively. It stands out for its ability to automatically handle categorical variables without the need for prior preprocessing, making it convenient for real-world datasets. CatBoost employs a novel algorithm called Ordered Boosting, which reduces overfitting by minimizing the impact of noisy features and preventing data leakage. It also utilizes symmetric decision trees, which helps improve training speed and memory efficiency. CatBoost supports both classification and regression tasks and provides various objective functions and evaluation metrics tailored to different problem domains. The library offers built-in support for handling missing data, reducing the need for

manual imputation. Additionally, CatBoost incorporates robust techniques such as robust mean encoding and ordered target statistics to further enhance its performance with categorical variables. With its focus on simplicity, speed, and robustness, CatBoost has become a popular choice for both beginners and experienced practitioners in the machine learning community. Its ability to deliver competitive performance while requiring minimal parameter tuning makes it a valuable tool for a wide range of applications.

1.5 Evaluation Methods

To compare performances among the five models, a number of evaluation methods were deployed. The evaluation methods are as follows:

Root Mean Square Error (RMSE) is a widely used metric in regression analysis to evaluate the accuracy of a predictive model. It measures the square root of the average of the squared differences between the predicted values and the actual values. RMSE provides a single numerical value that represents the average magnitude of the errors made by the model across all data points. A lower RMSE indicates better predictive performance, with values closer to zero indicating higher accuracy. RMSE is sensitive to outliers since it squares the errors before averaging them, making it important to preprocess data appropriately. It is often used alongside other evaluation metrics to provide a comprehensive assessment of model performance.

Mean Absolute Error (MAE) is a commonly used metric in regression analysis to measure the average magnitude of errors between predicted and actual values. Unlike RMSE, MAE calculates the average absolute differences between predicted and actual values, without squaring the errors. This makes MAE less sensitive to outliers compared to RMSE. A lower MAE indicates better predictive accuracy, with values closer to zero reflecting higher precision. MAE is straightforward to interpret since it represents the average absolute error in the same units as the target variable. It is often preferred in situations where outliers are present or when the focus is on understanding the average magnitude of errors rather than penalizing large errors more heavily. Additionally, MAE is useful for comparing the performance of different models and for assessing the robustness of a predictive model.

R-squared (R^2) or the coefficient of determination is a statistical measure that quantifies the proportion of variance in the dependent variable that is explained by the independent variables in a regression model. It ranges from 0 to 1, where a value closer to 1 indicates that a larger proportion of the variance in the dependent variable is explained by the independent variables. R-squared is interpreted as the percentage of the response variable variation that is explained by the model. However, it does not indicate the goodness of fit in an absolute sense and can be misleading when the model is overfitted. R-squared can be negative if the model performs worse than a model that simply predicts the mean of the dependent variable. Despite its limitations, R-squared is a valuable tool for evaluating the overall performance of regression models and comparing different models.

Training Time is also another performance metrics that denotes the time required to train the model with the training dataset. Reducing training time is essential for improving efficiency, scalability, competitiveness, and innovation in machine learning applications. Faster training times enable the scaling of machine learning systems to larger datasets and more complex models, making it feasible to tackle real-world problems with high-dimensional data. Speedy training facilitates quicker deployment of machine learning models into production environments, allowing organizations to derive value from their data more rapidly.

1.6 Organization of the Book

The organization of the next chapters in this book is as follows. Chapter 2 covers the comprehensive literature review. Development of the overall system architecture has been discussed in chapter 3. The discussion on experimental design, setup and implementation can be found in chapter 4. Chapter 5 shows system testing, result analysis and benchmarking, while chapter 6 concludes the thesis book.

CHAPTER 2
LITERATURE REVIEW

The prediction of electricity consumption has garnered significant attention in recent years due to its crucial role in energy management, sustainability, and cost optimization. Traditional methods of forecasting electricity demand often rely on statistical techniques and time series analysis, which may struggle to capture the complex patterns and dynamics inherent in consumption data. In contrast, machine learning approaches offer promising alternatives by leveraging the power of algorithms to uncover hidden patterns, nonlinear relationships, and dependencies in electricity consumption data. A comprehensive literature review is essential to explore the existing methodologies, challenges, and advancements in electricity consumption prediction using machine learning techniques. By synthesizing the findings from previous studies, this review aims to identify gaps in the current literature and propose avenues for future research to enhance the accuracy, efficiency, and applicability of machine learning-based electricity consumption forecasting models. This paper presents a thorough examination of the state-of-the-art methods, their strengths, limitations, and potential applications in predicting electricity consumption, thereby contributing to the growing body of knowledge in energy forecasting and management.

The literature review encompasses a wide range of studies that have applied machine learning techniques to electricity consumption prediction across various domains, including residential, commercial, industrial, and grid-level forecasting. Previous research has explored diverse machine learning algorithms such as artificial neural networks, support vector machines, decision trees, random forests, and gradient boosting methods for electricity consumption forecasting. Several studies have demonstrated the superiority of machine learning approaches over traditional methods in terms of accuracy, flexibility, and adaptability to complex consumption patterns and external factors. Challenges such as data quality, feature selection, model interpretability, and scalability have also been identified and addressed in the literature, highlighting the need for robust methodologies and comprehensive evaluations. Moreover, recent advancements in deep learning, ensemble methods, and hybrid models have shown promising results in improving the performance and generalization capabilities of machine learning-based electricity consumption

prediction models. By synthesizing and critically analyzing the existing literature, this review aims to provide insights into the current trends, best practices, and future directions in the field of machine learning-driven electricity consumption forecasting. Ultimately, the findings from this review will inform the development of more accurate, reliable, and efficient prediction models to support decision-making processes in energy management and policy planning.

2.1 Related Works

Several works on electricity consumption pattern have been conducted using various methods. Recently, Lin et al. (2023) and Liu et al. (2023) published two separate works on ”day ahead” residential power consumption forecasting, which may prove to be very useful for the prediction of power consumption of the next day with higher accuracy. Hino et al. (2013) explored daily consumption patterns of about 500 households in suburban area of Japan using statistical techniques like Gaussian distributions. They have also attempted to find typical patterns of electricity consumption by using generalized Kullback-Leibler divergence and hierarchical clustering. In another study, Zhou et al. (2017) deployed a fuzzy c-means clustering to group consumers with similar consumption patterns. The proposed system was used to cluster households of Jiangsu province, China. The groups formed with clustering showed significant difference in terms of various indicators of electricity consumption, thus validating the effectiveness of the algorithm.

McLoughlin et al. (2012) used multiple linear regression to find electricity consumption patterns of 4200 domestic Irish dwellings. The linear regression had four parameters, namely total electricity consumption, maximum demand, load factor and time of use (ToU) of maximum electricity demand. The study also found a strong relationship between maximum demand and a number of household appliances like dishwashers and electric cookers.

The contribution of different appliances to the total power consumption in household was studied by Farinaccio et al. (1999). The research attempted to break down electricity into major end user components. The study included daily demand profile and daily energy consumption pattern of domestic hot water heater (DHW) and refrigerator. There have also been studies on the relationship of economic improvement with access to quality electricity supply and consumption patterns. For example, Oseni (2012) studied energy consumption

pattern along with access to different forms of energy sources in Nigeria. In a different study, Keyno et al. (2009) introduced a method to pre prepare data to develop a model that can be applied to both levels of demand forecast, namely ”macroeconomic decision making” and ”engineering and middle management”.

In order to find load curves for different patterns of households, Abreu et al. (2012) used k-means clustering algorithm. The study discovered that 80 percent of the household consumptions can be explained using either persistent daily routines or consumption patterns for specific weather and daily conditions. Ramos et al. (2022) used k-nearest neighbor algorithm alongside neural network to develop two different models to forecast power consumption of an office building. The system uses a decision tree to decide which algorithm will be used in a particular context. In a recent study, Wang et al. (2019) developed an adoptive DBSCAN algorithm to create seasonal consumption pattern of individual consumers in weekdays. Then an enhanced apriori algorithm was proposed that describes the relationship between typical electricity consumption pattern and 35 factors related to dwelling characteristics, socio-demographic, etc. Another recent study (Perez-Chacon et al., 2018) used k-means clustering from apache spark machine learning library to cluster different patterns. A voting system was deployed to find the optimal number of clusters. Other than machine learning algorithms, other algorithms in artificial intelligence were used by researchers for performing optimization tasks in power consumption. For example, genetic algorithm had been used by Ahmed et al. (2020) to optimize the power consumption of IoT devices.

There have been numerous studies on predictive systems regarding other fields as well. Khan et al. (2019) developed a predictive system with BSAS clustering and Naive Bayes algorithm to assist in the diagnosis of type 2 diabetes. A mobile application was also developed for practical use and trial runs (Khan et al., 2017). In another study, an m-health application was developed using support vector machine (SVM) (Qureshi et al., 2020). Predictive systems have found their way into many other health-informatics related subjects like prediction of cesarean child birth (Khan et al., 2020) and many others (Preetha, 2020).

In recent years, there has been some progress in the development of web applications created to predict power consumption trends within specific regions. Using a diverse array of statistical methodologies, these applications offer insights into the anticipated energy

demands of various geographical areas. These web-based solutions adopt a broader perspective, analyzing data at the scale of regions, countries, or even entire continents. By harnessing vast datasets and state of the art algorithms, these platforms empower power utilities, governments and decision makers with comprehensive forecasts, enabling them to make informed decisions regarding resource allocation, infrastructure development, and energy management strategies. Through their accessible interfaces and predictive capabilities, these applications represent a significant advancement in the field of energy analytics, facilitating more efficient and sustainable utilization of power resources on a global scale.

RTE consumption forecast portal predicts daily peak electricity demand of France up to seven days in advance (Fig. 2.1). The prediction is primarily made using the weather forecast data. This website also shows hourly peak demand within next 7 days (Fig. 2.2).

Forecast from Wednesday 06 March 2024 to Tuesday 12 March 2024

Day	Hours	Peak	Reference temperatures	Expected gaps
Wednesday 06/03/2024	14:00	66 700	7.6	0.4
Thursday 07/03/2024	14:00	65 600	7.7	0.7
Friday 08/03/2024	14:00	64 100	7.8	0.9
Saturday 09/03/2024	17:30	56 600	7.8	1.4
Monday 11/03/2024	00:30	54 100	7.9	1.9
Tuesday 12/03/2024	00:30	61 000	8	2
Tuesday 12/03/2024	14:00	61 200	8.1	2

Fig. 2.1: Daily peak demand prediction of France in RTE consumption forecast

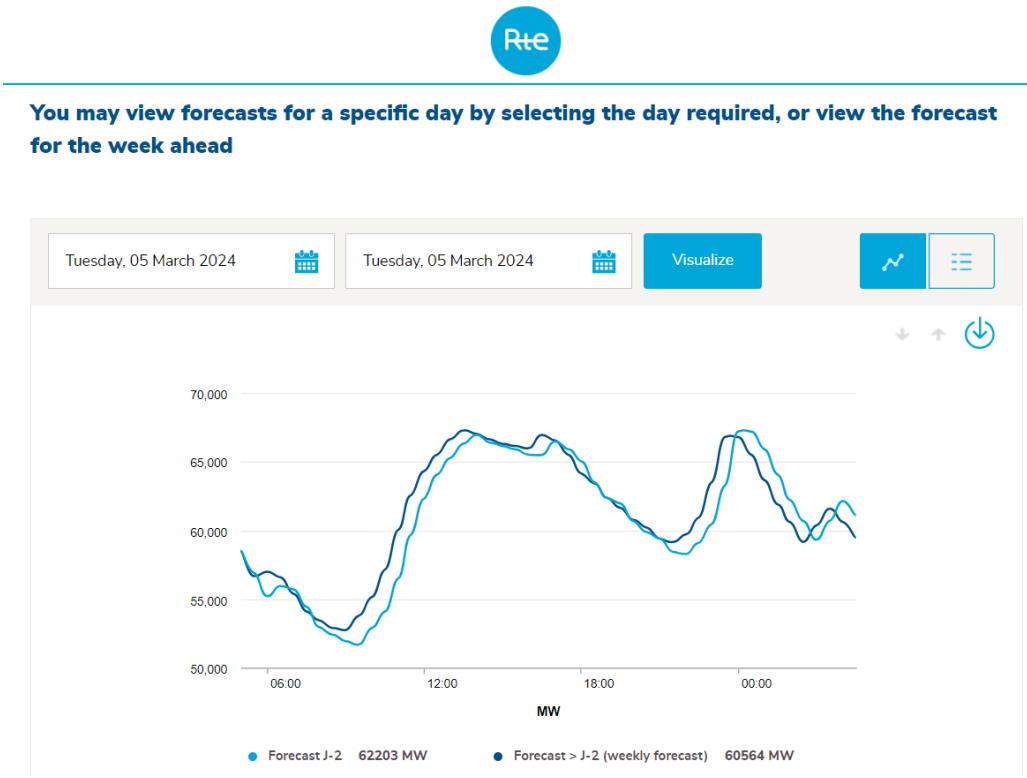

Fig. 2.2: Hourly peak demand prediction of France in RTE consumption forecast

Enerdata consumption predictor predicts the region-wise yearly energy consumption for every year up to 2050 (Fig. 2.3).

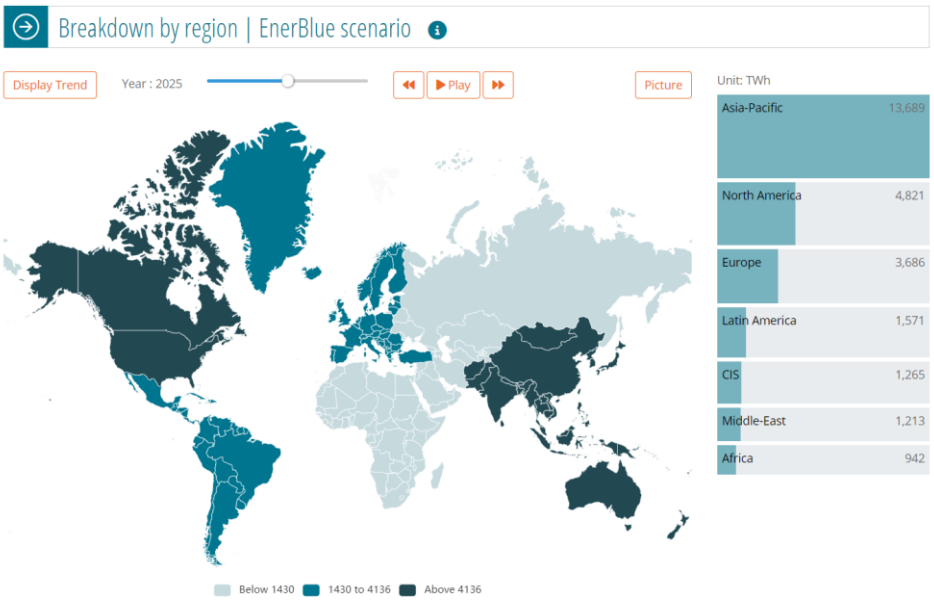

Fig. 2.3: Yearly total power consumption forecast by Enerdata

2.2 Gap Analysis

For gap analysis, the methods deployed by various researchers for prediction of electricity consumption has been shown below in Table 2.1:

Table 2.1: Related studies

Author(s)	Purpose of study	Result	Limitations
Liu et al. (2023)	Prediction of next day's household consumption	Daily electricity consumption predictor for household	The study focused solely on day-ahead predictions, potentially overlooking short-term fluctuations or longer-term trends in electricity consumption patterns.
Lin et al. (2023)	Prediction of next day's household consumption	Daily electricity consumption predictor for household	The accuracy and reliability of the electricity load scenario predictions was limited by the quality and availability of data used for model training and validation.
Hino et al. (2013)	Daily consumption pattern in Japan's household	Analysis of Daily consumption pattern in Japan's household using Gaussian distributions	The clustering method may face challenges in scalability when applied to large-scale datasets or when analyzing electricity consumption patterns across a wide range of households.

Table 2.1: Related studies (continued)

Author(s)	Purpose of study	Result	Limitations
Zhou et al.(2017)	Grouping consumers with similar consumption patterns	The consumers were grouped to different clusters with Fuzzy c-means clustering	The effectiveness of the fuzzy clustering-based model may be highly dependent on the specific algorithm and parameters chosen, which could affect the reliability and generalizability of the results.
McLoughlin et al. (2012)	Finding electricity consumption patterns of 4200 domestic Irish dwellings	By using linear regression algorithm, household power consumption was predicted in Ireland.	The accuracy and reliability of characterizing domestic electricity consumption patterns could be limited by the availability and quality of data on dwelling and occupant socio-economic variables, which may vary in completeness and accuracy.
Oseni (2012)	Energy consumption pattern along with access to different forms of energy sources in Nigeria	The access to various energy sources was studied and the energy consumption pattern was shown.	The research was based on a specific sample or subset of households in Nigeria, potentially limiting the generalizability of the findings to the broader population.

Table 2.1: Related studies (continued)

Author(s)	Purpose of study	Result	Limitations
Keyno et al. (2009)	Developing a model that can be applied to both levels of demand forecast, namely "macroeconomic decision making" and "engineering and middle management"	Two types of demand level forecasts were made using only one model.	The forecasting method was limited to short-term predictions and may not account for longer-term trends or seasonal variations in electricity consumption patterns.
Abreu et al. (2012)	Finding load curves for different patterns of households	Using k-means clustering, the load curves having similar patterns were clustered.	The analysis focused on a specific timeframe, potentially overlooking seasonal variations or longer-term trends in residential electricity consumption patterns.
Ramos et al. (2022)	Forecasting power consumption of an office building	The monthly consumption of an office premise was predicted by k-nearest neighbor algorithm.	The findings may be specific to the dataset and context used in the research, limiting their applicability to different office buildings with varying characteristics or energy usage patterns.

Table 2.1: Related studies (continued)

Author(s)	Purpose of study	Result	Limitations
Wang et al. (2018)	Creating seasonal consumption pattern of individual consumers in weekdays	By using an adoptive DBSCAN algorithm, consumption pattern for individual customers were predicted.	The effectiveness of association rule mining for analyzing household characteristics' impacts on electricity consumption patterns may be limited by the availability and quality of data, which could vary in completeness and accuracy.
Perez-Chacon (2018)	Clustering different patterns	K-means clustering of Apache spark library was deployed to group different consumption patterns according to their characteristics.	While big data analytics can uncover complex patterns in electricity consumption, the interpretability of the results may be challenging, making it difficult to extract actionable insights for energy management and policy-making.

From the literature review it is evident that there has been little research conducted on the electricity consumption pattern of any region of Bangladesh so far. There is also absence of an application that can predict power consumption in consumer level. To cover the gap, this study intends to work with power consumption pattern in the Dhaka city and develop a web portal for prediction of household electricity consumption in consumer level.

2.3 Summary

This chapter has listed notable recent studies related to electricity consumption from all over the world along with other relevant studies. The research gap regarding individual consumer level consumption prediction was discussed as well. Besides, various application related to power consumption/demand prediction were also shown. This chapter justifies the need to develop a model using a high-performing algorithm and a web application for individual consumers' consumption forecast.

CHAPTER 3
DATA COLLECTION, PREPROCESSING, AND DESIGN OF MODELS ARCHITECTURES

To facilitate the forecast of power consumption of individual consumers, a dataset of forty thousand consumers' three year's data was collected from Dhaka Electric Supply Company Ltd. (DESCO).

3.1 Data Collection

Anonymous consumption dataset was collected from Dhaka Electric Supply Company (DESCO) Limited with written approval from ICT division, DESCO. The dataset covers the area of Kafrul, Kallyanpur, Pallabi, Monipur, Rupnagar area of Dhaka city. After removing unnecessary features, the dataset contains nearly 6 features, namely peak/off-peak data consumption, month, assigned tariff, meter type, sanctioned load etc. The distribution of different types of users can be seen in Fig. 1.1. As it can be found in the figure, residential (96.92%) consumers are pre-dominant in the dataset. The dataset contains electricity consumption information of approximately 40 thousand users over the period of July 2019 to Jun 2023. It can also be seen that 92% of the meters found in this dataset are single phase meters, while the rest (8%) are three phase meters (Fig. 1.2).

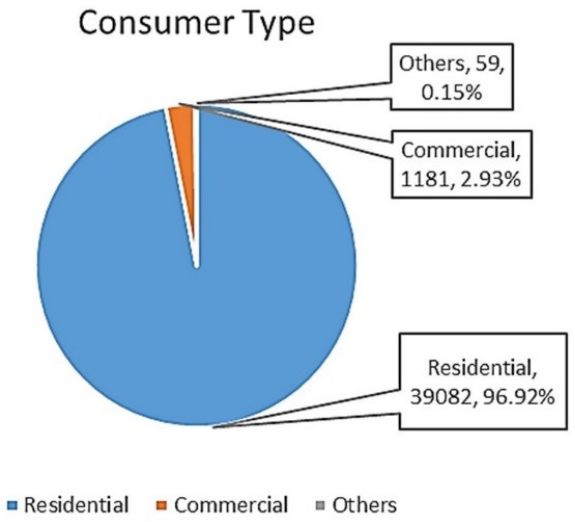

Fig. 3.1: Distribution of different consumer types in the dataset.

The box plot of monthly consumption data (in kwh) during July 2022 to June 2023 shows that Overall power consumption during November, December, January and February are considerably less than other months (Fig. 1.3). Another noticeable trait in the dataset is the presence of significant outliers in the data of February 2023.

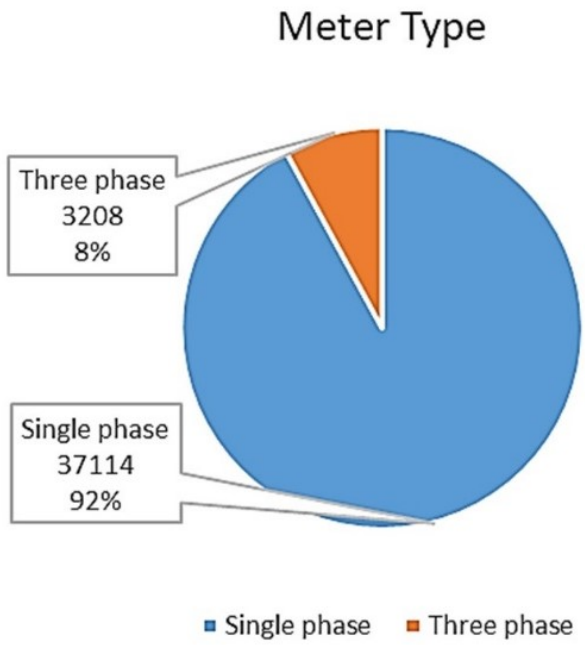

Fig. 3.2: Distribution of different meter types in the dataset.

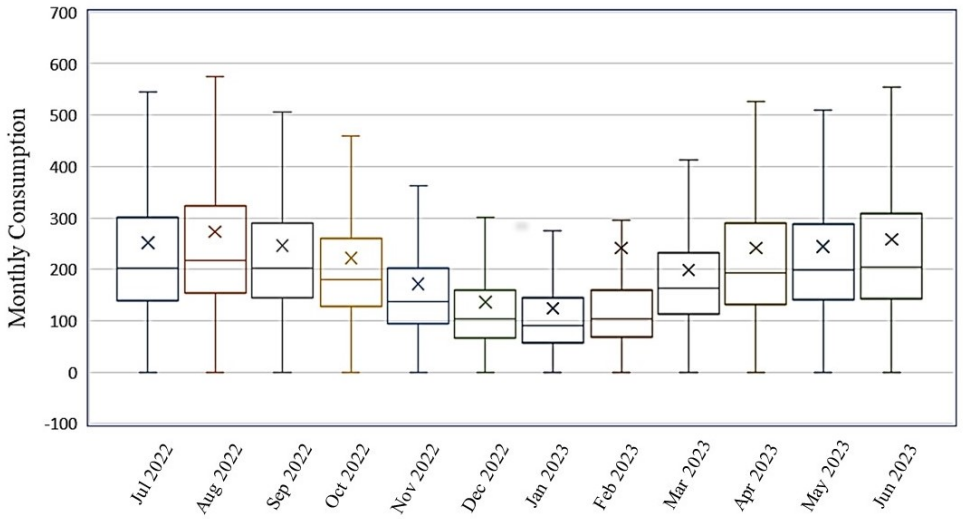

Fig. 3.3: Box plot of monthly consumption data (in kwh) during July 2022 to June 2023.

3.2 Data Preprocessing

The steps which have been followed for data pre-processing are shown below:

• Feature scaling of numeric data: Numeric values were normalized to eliminate bias and to obtain better accuracy. Normalization was performed using the following formula,

$$xi = (xi - \mu) / (max(x) - min(x)) \quad (3.1)$$

where μ is the mean value of the feature x.

• Missing value management: The columns with missing values were dropped in order to improve performance of the prediction systems.

• Non-Numerical Feature Handling: The categorical features with non-numeric values were encoded to binary features using Pandas library. Below are the non-numerical features that were binary encoded:

- Meter type
- Consumer type or tariff

• Removal of redundant features: Using Pearson correlation coefficient (Mujahid, 2017), redundant features having similar effect on the algorithm were reduced to improve execution time.

• Division of dataset: After completion of pre-processing, the dataset was ready for being used in the prediction models (Fig. 3.4). The whole dataset was randomly divided into Training set and test set with a ratio of 80:20.

	Account No.	Max Power(kW)	Meter Type_1P	Meter Type_3P	Tariff_Category-A: Residential
0	2947556	3.0	1	0	1
1	2951311	2.0	1	0	1
2	2954701	5.0	1	0	1
3	2964633	3.0	1	0	1
4	11041971	25.0	0	1	0

...

...

...

Month_2019-09	Month_2019-10	Month_2019-11	Month_2019-12	Month_2020-01	Month_2020-02
80.292	67.629	61.605	33.557	31.689	32.388
249.981	208.977	187.113	119.156	38.828	113.857
155.936	302.995	241.312	173.833	141.513	142.886
134.031	115.082	99.455	59.705	53.431	66.768
0.000	0.000	0.000	0.000	0.000	0.000

Fig. 3.4: Sample of the collected dataset.

3.3 Predictive Analysis with XGBoost

The XGBoost (eXtreme Gradient Boosting) is a popular and efficient open-source implementation of the gradient boosted trees algorithm. It's a well-known open source library with a very efficient implementation of the gradient boosting algorithm (Friedman, 2001). Soon after its initial release, XGBoost rose to prominence as the go-to technique for solving various machine learning problems and frequently became a common element in the winning solutions to many machine learning competitions. Ensemble type models like XGBoost are primarily based on decision trees (Fig. 3.5). Decision trees are added one at a time to the model which tend to correct the errors done by previous models.

Gradient boosting is a supervised learning approach that combines an ensemble of estimates from a variety of weaker and simpler models in an effort to properly forecast a target variable. Because of its capable handling of a wide range of data types, relationships, distributions, and adjustable hyperparameters, the XGBoost algorithm excels in machine learning contests. Regression, classification (binary and multiclass), and ranking issues can all be solved using XGBoost.

Regression trees serve as the weak learners when utilizing gradient boosting for regression, and each one of them associates each input data point with a leaf that holds a continuous score. With a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity, XGBoost minimizes a regularized (L1 and L2) objective function (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that forecast the residuals or errors of earlier trees, which are then integrated with earlier trees to produce the final prediction (Fig. 3.6). As the loss when introducing new models is minimized, the technique is known as gradient boosting.

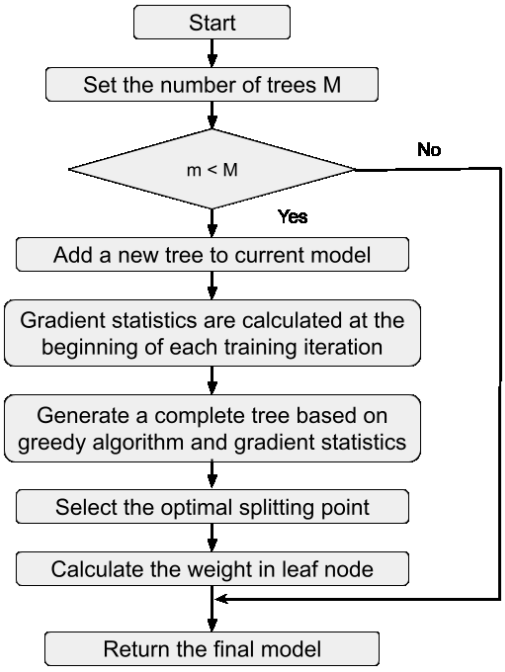

Fig. 3.5: XGBoost Flwochart

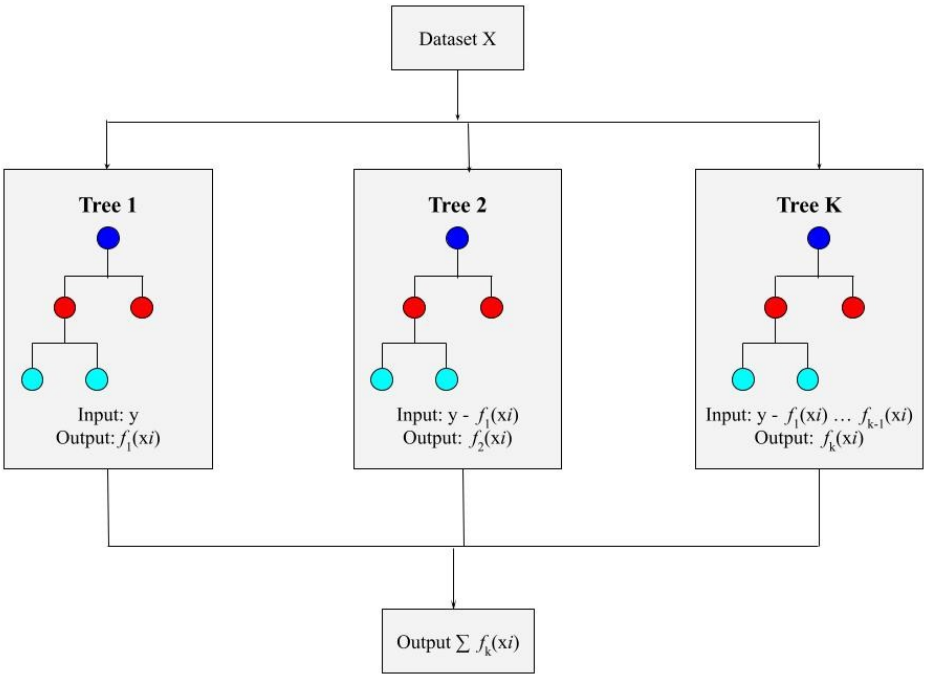

Fig. 3.6: Gradient Boosting in XGBoost

The process has been described as follows:

- Step 1: Make an Initial Prediction and Calculate Residuals. The initial prediction can be made on anything. For example, it can be set as mean, median or mode of the input dataset. After that, the residuals are calculated using the following formula,

$$\mathbf{residuals} = \mathbf{observedValues} - \mathbf{predictedValues} \quad (3.2)$$

- Step 2: Build an XGBoost tree. Each tree begins with a single leaf, which contains all the residuals. In order to build the tree, it is required to calculate the similarity score of the leaves (before and after splitting) and gains after splitting the residuals using the formulae below

$$\mathbf{similarityScore} = \frac{\mathbf{sumOfResiduals}^2}{(\mathbf{numberOfResiduals} + \boldsymbol{\lambda})} \quad (3.3)$$

where, λ is the regularization parameter

$$\mathbf{gain} = \mathbf{leftSimilarity} + \mathbf{rightSimilarity} - \mathbf{rootSimilarity} \quad (3.4)$$

- Step 3: Prune the tree. Starting from the bottom of the tree, the splits with positive gains are kept while the splits with negative gains are removed
- Step 4: Calculate the Output Values of Leaves. The output values are calculated using the following formula.

$$\mathbf{outputvalue} = \frac{\mathbf{sumOfResiduals}}{\mathbf{numberOfResiduals} + \boldsymbol{\lambda}} \quad (3.5)$$

  If the learning rate is too low, the learning process takes long time. On the other hand, if the learning rate is too high, gradient descent may not converge to the minimum cost value. For this study, multiple learning rates have been tried and the most efficient learning rate found was 0.3. Fig. 3.7 shows the convergence of residuals with respect to iterations for learning rate 0.1, 0.3, and 1.0. It can be seen that, learning rate 0.3 converges the cost faster compared to other values for learning rate.
- Step 5: Make new predictions
- Step 6: Calculate Residuals Using the New Predictions
- Step 7: Repeat step 2 to 6 until the residuals are very small or the number of iterations has reached the maximum number of iterations.
- Step 8: Evaluation of the algorithm with test dataset. The trained model was then evaluated using the test dataset. The following metrics were used to evaluate to model: MAE, RMSE, R2.

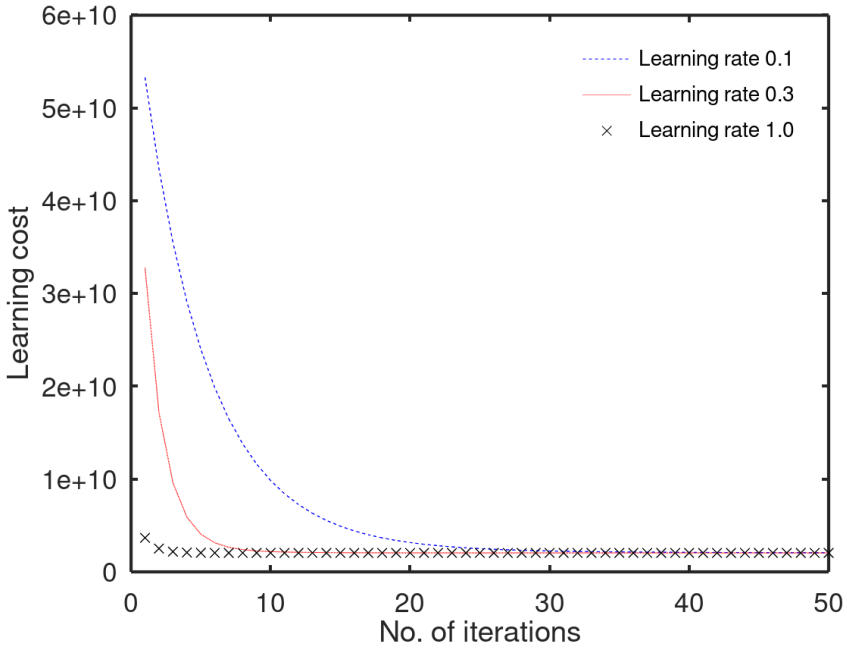

Fig. 3.7: Convergence of residuals for different learning rates.

3.4 Predictive Analysis with LightGBM

LightGBM is a popular and efficient open-source implementation of the Gradient Boosting Decision Tree (GBDT) algorithm. GBDT is a supervised learning technique that combines an ensemble of estimates from a variety of weaker and simpler models in an effort to accurately forecast a target variable. In order to dramatically increase the effectiveness and scalability of traditional GBDT, LightGBM employs additional approaches.

Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) are added to the standard Gradient Boosting Decision Tree (GBDT) algorithm by LightGBM. These methods are intended to greatly enhance the effectiveness and scalability of GBDT. GBDT is an ensemble model of decision trees, which are trained in sequence. In each iteration, GBDT learns the decision trees by fitting the negative gradients (also known as residual errors).

The main cost in GBDT lies in learning the decision trees, and the most time-consuming part in learning a decision tree is to find the best split points. One of the most popular algorithms to find split points is the pre-sorted algorithm, which enumerates all possible

split points on the pre-sorted feature values. This algorithm is simple and can find the optimal split points, however, it is inefficient in both training speed and memory consumption. Another popular algorithm is the histogram-based algorithm. Instead of finding the split points on the sorted feature values, the histogram-based algorithm buckets continuous feature values into discrete bins and uses these bins to construct feature histograms during training.

The fundamental difference of LGBM with XGBoost lies within the way a decision tree is constructed (Fig. 3.8). In case of LGBM, the tree is split leaf-wise instead of level-wise or depth-wise. It can reduce more loss compared to the level-wise algorithms. The process has been described as follows:

- The dataset is divided into a number of bins. Each of the bins contains a range of data points, which makes it faster for the algorithm to find the optimum split using these bins.
- The algorithm uses Exclusive Feature Bundling (EFB) to combine more than one features together to produce one feature, which significantly reduces the running time.
- Gradient Based One Side Sampling (GOSS) reduces training time by keeping all the examples with large gradients (errors) and conducting random sampling on the examples with small gradients. This method reduces training time by only using a subset of training data which proved to be more impactful and less time-consuming.
- The selected features are then assigned to respective slope coefficients

Because of its robust handling of a variety of data types, relationships, distributions, and the number of hyperparameters that can be fine-tuned, the LightGBM method does well in machine learning contests.

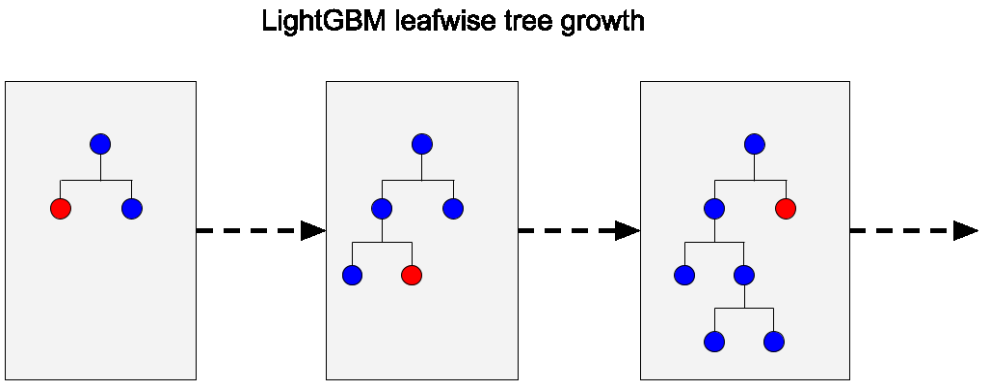

Fig. 3.8: Tree growth in LGBM

3.5 Predictive Analysis with CatBoost

CatBoost is another popular and high-performance open-source implementation of the Gradient Boosting Decision Tree (GBDT) algorithm (Fig. 3.9) (Prokhorenkova, 2018). GBDT is a supervised learning algorithm that attempts to accurately predict a target variable by combining an ensemble of estimates from a set of simpler and weaker models. While training the model with input data, a number of decision trees are built. Each successive tree has a lower loss than its predecessor. Here are some of the key factors regarding how it works-

- Overfitting detector: The change in loss value is calculated before building each tree. In order to prevent overfitting, the following condition is checked, which if satisfied, tree generation is halted-

$$\mathbf{CurrentPValue} < \mathbf{Threshold} \quad (3.6)$$

- Preliminary calculation of splits: To determine the possible ways of putting data into buckets, quantization is performed. This step is impeccable for choosing the tree structure. The method quantization and the number of buckets are set in the starting parameters.
- Transformation of features: Categorical features and text features are converted to numerical features before calculating the values of leaves.

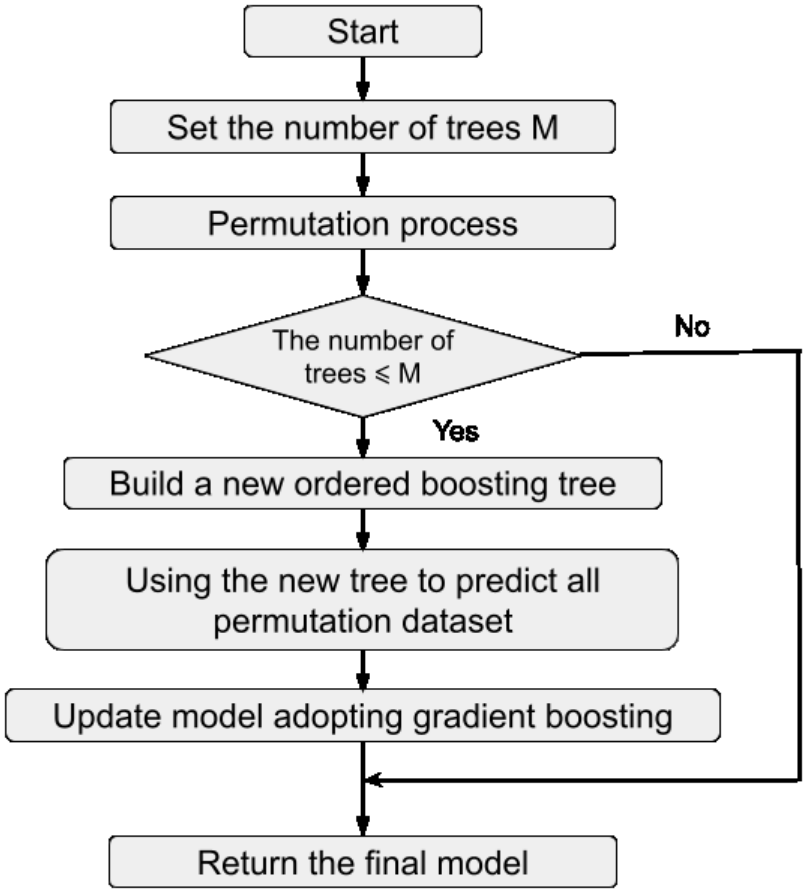

Fig. 3.9: Flowchart of CatBoost Algorithm

CatBoost brings two significant algorithmic improvements to GBDT:

1. The use of ordered boosting, a permutation-driven substitute for the conventional approach
2. An original method for handling categorical features

Both methods were developed to combat a prediction shift brought on by a particular type of target leakage that is present in all current iterations of gradient boosting algorithms.

3.6 Predictive Analysis with Neural Network

Artificial neural networks (ANNs) are computational models inspired by the structure and functioning of biological neural networks found in the human brain. These networks consist of interconnected nodes, or neurons, organized in layers, including input, hidden, and output layers. ANNs learn from data by adjusting the strengths of connections between neurons, known as weights, through a process called training. They are capable of learning

complex patterns and relationships from large datasets, making them powerful tools for tasks such as classification, regression, and pattern recognition. With ongoing advancements in research and technology, artificial neural networks continue to drive innovations in artificial intelligence and machine learning applications.

3.7 Predictive Analysis with Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that best separates different classes in the input data space. SVM aims to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, making it robust to outliers. By utilizing kernel functions, SVM can handle non-linearly separable data by transforming it into higher-dimensional feature spaces. SVM has found wide applications in various domains such as image classification, text categorization, and medical diagnosis, owing to its versatility and effectiveness in handling complex datasets.

3.8 Train-Test Split

To effectively train the models and evaluate their performance, the dataset underwent a preprocessing step: random splitting into two distinct subsets, following an 80:20 ratio. This division ensured that a substantial portion of the data, constituting 80% of the total dataset, was allocated for training purposes. Meanwhile, the remaining 20% formed a separate test dataset, designated exclusively for assessing the models' performance and generalization capabilities. This partitioning strategy allowed for rigorous validation of the trained models, enabling researchers to gauge their effectiveness in making accurate predictions on unseen data.

3.9 Hyper-parameter Tuning

To find the best algorithm with the right hyperparameter tuning, any of the following two techniques are used in general:

a. Random search

b. Grid search

3.9.1 Random Search

This process defines a search space as a bounded domain of hyperparameter values and randomly sample points in that domain. This is faster, but there remains a possibility of not getting the right hyperparameters.

3.9.2 Grid Search

Grid search defines a search space as a grid of hyperparameter values and evaluates every position in the grid (Fig. 3.10). It takes more time compared to random search, but it performs better in terms of finding the right hyperparameters.

Grid search is excellent for spotting combinations that usually produce good results. Although it frequently takes more time to complete, random search is excellent for discovering and obtaining hyperparameter combinations that you would not have predicted intuitively.

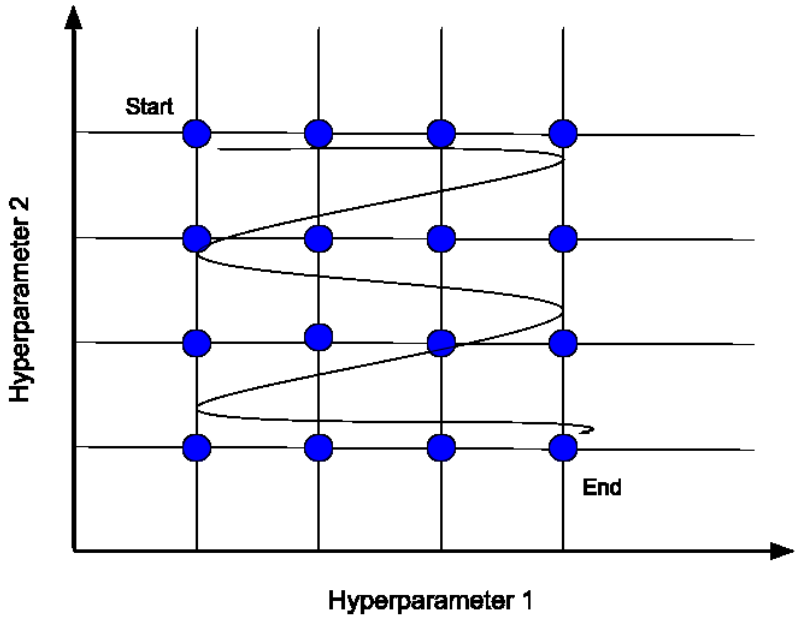

Fig. 3.10: Grid search for hyperparameter tuning and finding the best error score.

With respect to the requirement of this project, grid search technique was used with 3 different evaluation metrics:

1. Mean Absolute Error (MAE)
2. Root Mean Squared Error (RMSE)
3. R Squared (R2)

3.10 Input and Output Prediction Data

Both the train and test dataset were again split into input and output datasets. The input dataset comprises of the following fields:

- Sanctioned load
- Tariff
- Meter type (single phase or three phase)
- Monthly electricity consumption of 36 months (July 2019 to June 2022)

The output dataset comprised of monthly electricity consumption for next 12 months (July 2022 to June 2023). To handle the output of 12 separate fields, the MultiOutputRegressor class from scikit-learn library was deployed. This library helped to create 12 models for 12 months' output with each algorithm in a clean and convenient way. The average output for all the models along with the actual data can be seen in Table 3.1.

3.11 Evaluation Metrics

Table 3.2 shows different types of evaluation metrics that are used to evaluate machine learning based systems.

After comparing the performance of the five models for MAE, RMSE and R2 score, it was found that Artificial Neural Network (ANN) performs best for the given dataset. The details of performance comparison will be provided in Chapter 6.

Table 3.1: Monthly average consumption comparison among actual value and predicted values from test dataset

	Actual Average Consumption in test dataset (kwh)	XGBoost Output Average (kwh)	LightGBM Output Average (kwh)	CatBoost Output Average (kwh)	Neural Network Output Average (kwh)	SVM Output Average (kwh)
Jul-22	248.7993	248.6904	248.8185	249.3387	253.1993	239.9504
Aug-22	269.1216	269.6681	269.7427	270.008	288.9825	250.5856
Sep-22	243.1167	243.1663	243.2466	243.6559	238.0122	234.3268
Oct-22	219.4141	219.0335	219.1312	219.5191	218.152	210.9163
Nov-22	169.9105	169.3018	169.2146	169.7438	176.478	160.8697
Dec-22	134.9979	135.3366	135.0726	135.4687	143.739	126.1928
Jan-23	122.2134	123.262	123.1058	123.5054	130.6159	113.3855
Feb-23	133.2236	1157.152	480.3866	360.7525	184.0526	125.7967
Mar-23	195.7509	196.9899	197.0751	197.3967	183.0797	188.25
Apr-23	236.9899	238.2419	238.4874	238.5608	233.8378	228.3018
May-23	240.6159	241.7838	242.2302	242.3042	244.3941	231.2519
Jun-23	253.137	254.6376	255.4111	255.2988	230.3853	241.327

Table 3.2: Evaluation metrices

Metric Name	Description	Optimization Direction
validation:accuracy	Classification rate, calculated as #(right)/#(all cases).	Maximize
validation:auc	Area under the curve.	Maximize
validation:error	Binary classification error rate, calculated as #(wrong cases)/#(all cases).	Minimize
validation:f1	Indicator of classification accuracy, calculated as the harmonic mean of precision and recall.	Maximize
validation:logloss	Negative log-likelihood.	Minimize
validation:mae	Mean absolute error.	Minimize
validation:map	Mean average precision.	Maximize
validation:merror	Multiclass classification error rate, calculated as #(wrong cases)/#(all cases).	Minimize
validation:mlogloss	Negative log-likelihood for multiclass classification.	Minimize
validation:mse	Mean squared error.	Minimize
validation:ndcg	Normalized Discounted Cumulative Gain.	Maximize
validation:rmse	Root mean square error.	Minimize

3.12 Summary

This chapter shows the methodology of completion of this research starting from data collection up to development of forecast models. The comparison of forecast models and development of the web application will be discussed in the subsequent chapters.

CHAPTER 4
EXPERIMENTAL SETUP AND IMPLEMENTATION

To develop the web application for electricity consumption prediction using Artificial Neural Network (ANN), the trained model was needed to be deployed from Jupyter notebook to the web server. Flask and Pickle libraries were used to save the model to a file in the web server, load the model and run prediction based on user input in the web application.

4.1 Deploying the Model in a Web Server

In order to transition the machine learning model from its training environment—a personal computer running Jupyter Notebook—to a functional web application, deployment onto a web server was necessary. By housing the model on a web server, it became accessible to users through a web application interface, allowing for remote access and utilization. This deployment process involved configuring the model to operate within the infrastructure of the web server, ensuring compatibility with the technologies and protocols required for web-based interaction. Once deployed, the model could seamlessly process incoming requests from the web application, providing real-time predictions or analyses as needed.

4.1.1 Saving the Trained Model

The first step of model deployment is to save the trained model from Jupyter Notebook in a file. To this task, Joblib was chosen because of its simplicity and ease of operation. Joblib is a set of tools for lightweight pipelining in Python. It is robust, reliable, fast and has specific optimizations for numpy arrays. These tools can be easily installed in any stable python version. Joblib is based on BSD-license, which makes if suitable to build both open source and proprietary systems. The file extension used for pickled files for this project is ".sav".

The latest versions of python already contains Joblib, so there is no need to install it. However, if Joblib is not installed in the system, it can be installed using the command:

pip install joblib

If Joblib is installed in the system, we need to save the trained model to a ".sav" file. There are three steps to save the model:

1. Import joblib by executing
   $import\ joblib$
2. Save the model to a ".sav" file by executing
   $joblilb.dump(model,'ann\_reggressor.sav')$

Here, "model" is the model object trained using the test dataset and ANN.

4.1.2 Deployment of the Saved Model in Web Server

To deploy the saved model binary file in web server, a web framework named "Flask" was installed in the web server. Flask is a powerful and flexible micro web framework for Python that works for both small and large scaled projects. The framework provides a swift way to deploy the web project and get it running within short time. The next steps are to be included in the server-side application.

1. Import Flask
   $from\ flask\ import\ Flask, render\_template, request$
2. Import Joblib
   $import\ joblib$
3. Load the saved model
   $model\ =\ joblib.load('ann\_reggressor.sav')$
4. Run prediction from the html request data and the model
   $arr\ =\ np.array([[data1, data2, \ldots\ldots\ldots, datan]])$
   $pred\ =\ model.predict(arr)$
   $return\ render\_template('result.html', data = pred)$

The variables data1, data2, ……, datan denote the input variables like tariff, sanctioned load, meter type and three years' consumption data. The predicted data named 'pred' here contains an array of twelve numbers, each representing a months' consumption (from July to June) for the user. It is sent to the web page for displaying as consumption forecast.

4.2 Development of the Web Application

The prediction web interface is a single-page web application (Fig. 4.1 and Fig. 4.2) that allows users to insert the previous consumption records of a particular consumer and get the predicted output for the next 12 months.

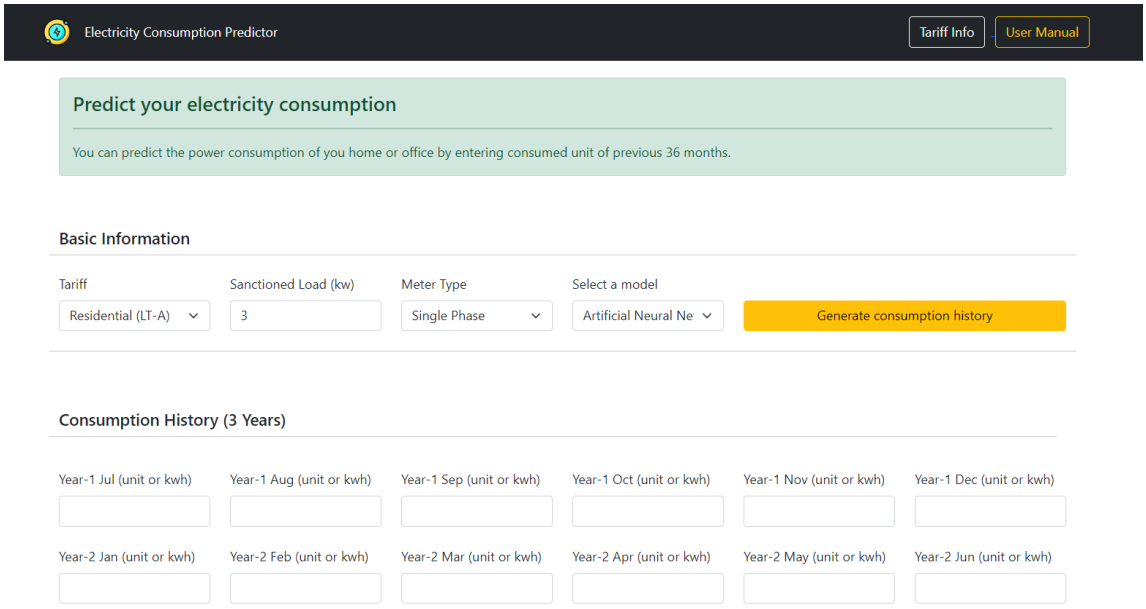

Fig 4.1: Web interface of the electricity consumption prediction system – 1

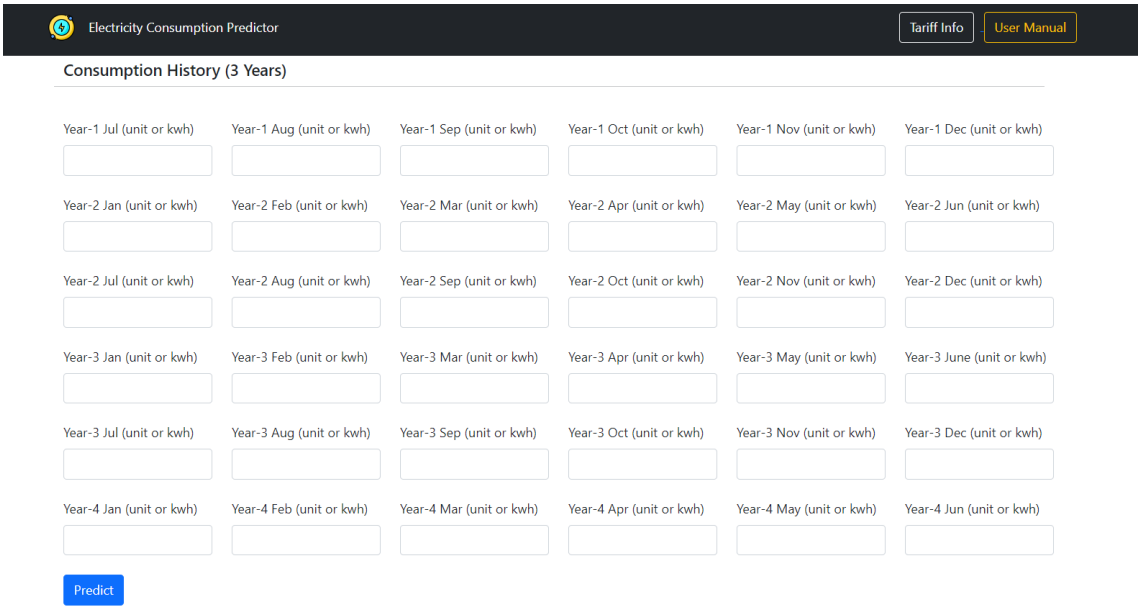

Fig 4.2: Web interface of the electricity consumption prediction system - 2

4.2.1 Basic Architecture of the Web Application

In order to develop a system to predict the household electricity consumption in Dhaka city, five prediction models have been developed using ANN, SVM, XGBoost, LightGBM and CatBoost algorithms. Later, the model trained using ANN was integrated into a web interface to be made publicly available for further use. The household consumption prediction system has the following components (Fig. 4.3):

1. **Prediction models:** When predicting the likelihood of a specific outcome, machine learning refers to prediction as the result of an algorithm that has been trained on past data and applied to current data. Using previous data, predictions in machine learning enable power utilities to make precise assumptions about the anticipated consumption demand. Regression analysis calculates correlations between variables, identifying important trends in vast, varied data sets and how those patterns relate to one another. While prediction focuses on fitting a shape that is as near to the data as possible, classification focuses on categorizing data. In this case, five regressive models were trained with ANN, SVM, XGBoost, LightGBM and CatBoost algorithms, where the hyperparameters were set to obtain the highest accuracy by grid search. The model trained by ANN was chosen to be used in the web application because of its superior performance compared to other applications.
2. **Server-side web application:** A server-side web application, also known as a back-end web application, is a type of web application where the majority of the logic and processing occurs on the server side. In this architecture, client devices, such as web browsers, interact with the server through HTTP requests and responses. Every client device sends requests to various servers while accessing and browsing the Internet, and servers can serve multiple client devices simultaneously.

   In many cases, the server generates HTML dynamically based on the request and sends it back to the client for display in the web browser. This allows for the creation of dynamic web pages that can change content based on user input or other factors. Examples of server-side technologies commonly used in web development include PHP, Java Servlets, Python with frameworks like Django or Flask, Ruby on Rails, Node.js, and ASP.NET. These technologies provide frameworks and libraries for building server-side logic, handling HTTP requests, interacting with databases, and other common tasks in web application development.

For this project, a server-side middleware application was developed with Flask (Python) to facilitate the web interface. This application receives the input data from the front-end application, calculates the predicted consumption value using the prediction model, and sends back the predicted value to the front-end application.

3. **Front-end web application:** The front-end application is a client side application. Everything in a web application that is shown or takes place on the client is referred to as "client side" in web development (end user device). This comprises everything the user sees, like text, graphics, and the rest of the user interface (UI), as well as any operations an application carries out inside the user's browser. The client-side browser interprets markup languages like HTML and CSS. Additionally, a lot of modern developers are moving away from performing everything on the server side and incorporating client-side processes into their application design. Business logic for dynamic webpages, for example, typically runs client side in a modern online application. Client-side scripting is almost often used for these types of activities. In this case, an HTML-based front-end web application was developed and linked to the server-side application that takes input from users, sends the input to the server, and shows the predicted value returned from the server. Bootstrap 5 and Google charts were also used to improve user experience. Client side processing enabled the author to implement input generation, statistical analysis of the output and data visualization of the predicted output.

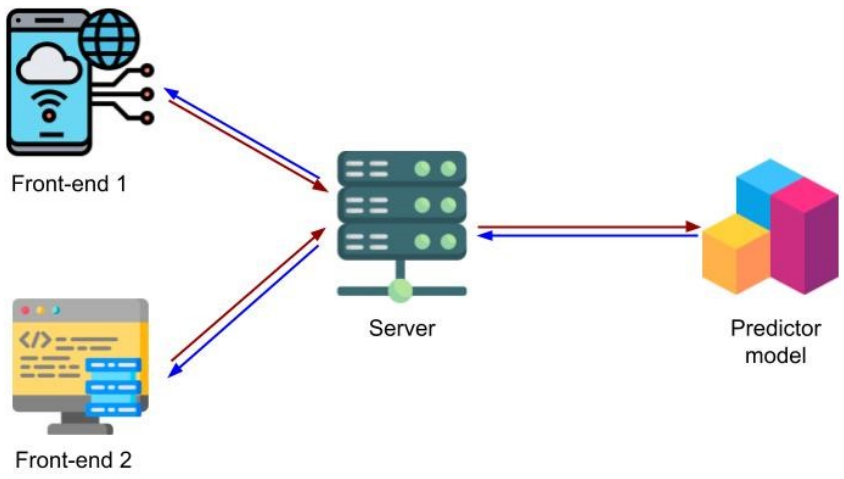

Fig. 4.3: Overview of the system architecture of the household electricity consumption prediction system.

4.3 Performance of the Web Application

The forecast web application performs very well when provided the accurate historical consumption data. For comparison, a comparison between predicted consumption value calculated by the web application and actual consumption value of a consumer during July 2022 to June 2023 has been shown in Table 4.1. It can be seen that the performance of the web portal is within satisfactory range.

Table 4.1: Comparison between forecasted consumption by web application and actual consumption

Month	Actual Consumption (kwh)	Forecasted Consumption by web application (kwh)
July 2022	154	161.24
August 2022	162	158.53
September 2022	159	158.02
October 2022	142	145.35
November 2022	129	124.17
December 2022	113	116.93
January 2023	118	122.54
February 2023	127	131.05
March 2023	139	136.57
April 2023	147	149.74
May 2023	153	151.26
June 2023	152	157.37

4.4 Proposed Components to Be Added in the Future

To be able to serve the need of mass people, a few more components can be suggested to be added in the future.

1. A load balancer in the server to manage a large number of users.
2. Training the model with updated datasets regularly, which will improve the model over time.

3. Mobile application for the convenience of mobile users.

4.5 Summary

This chapter described the design and implementation of the web application tool. It also briefly discussed how the trained model has been implemented in the web application for forecasting electricity consumption.

CHAPTER 5
SYSTEM TESTING, RESULT ANALYSIS AND BENCHMARKING

Following the implementation and rigorous evaluation of the power consumption forecast system employing five distinct algorithms, a comprehensive comparison of the performance metrics derived from each model was conducted. This comparative analysis encompassed a thorough examination of the best-performing attributes of each algorithm, with a particular emphasis on assessing their efficacy across three key evaluation metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R Squared (R2). This comparative assessment facilitated a nuanced understanding of the nuances and intricacies associated with different algorithmic approaches, thereby informing future decision-making processes and methodological refinements. Through this meticulous evaluation process, the author was able to identify and prioritize the most promising avenues for improving the accuracy, robustness, and efficiency of the power consumption forecast system.

5.1 System Testing of the Web Application

System testing involves testing design and behavior of the system and comparing with the expectations of the stakeholders. It consists of a series of black-box tests which are carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both.

5.1.1 Setting up the Test Environment

A test environment was set up to perform overall system testing. The environment comprised of the elements shown in Table 5.1. It is to be noted that, the whole environment was run in a personal computer as per the specifications provided.

Table 5.1: System test environment

Component Type	Component Name	Specification
Hardware	Processor	Intel(R) Core(TM) i5-8265U CPU (8 CPUs) @ 1.60GHz 1.80 GHz
	Memory	12 GB
	Graphics Processor	NVIDIA GeForce MX150
	Graphics Memory	8 GB
	Screen Resolution	1920 x 1080
Software	Operating System	Windows 11 Home 64 bit
	Python	3.8.2
	Flask	3.0.2
	Joblib	0.16.0
	Bootstrap	5.0.2

5.1.2 Generate Test Data

A test case generator was deployed in the front end application, where the user can generate random consumption data to test the forecast of the working model. The user had to click a button called “Generate consumption history” after selecting the tariff, meter type and sanctioned load. The button click generated a random test input dataset comprising of 36 months’ consumption data successfully (Fig. 5.1 and Fig. 5.2).

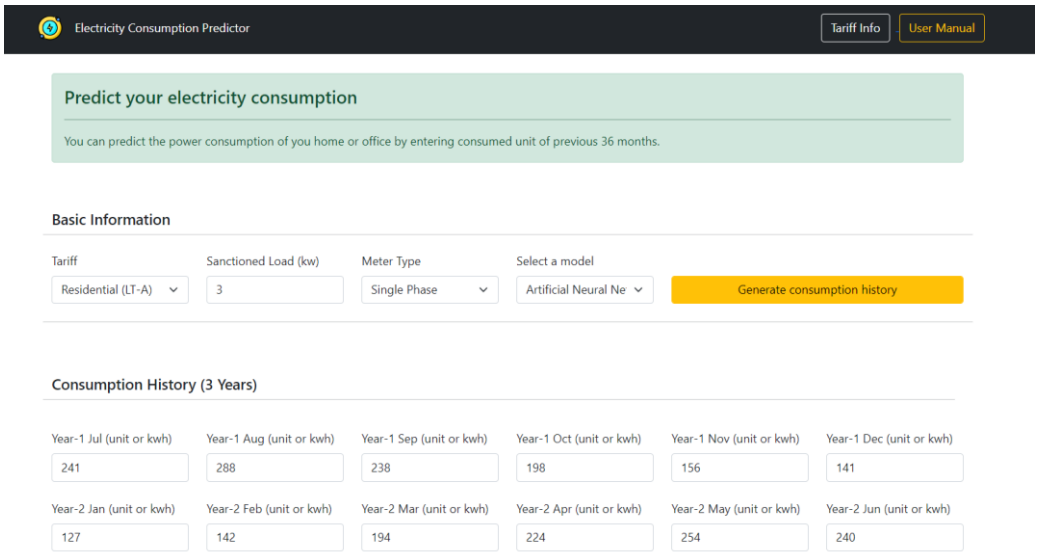

Fig. 5.1: Generating test data – 1

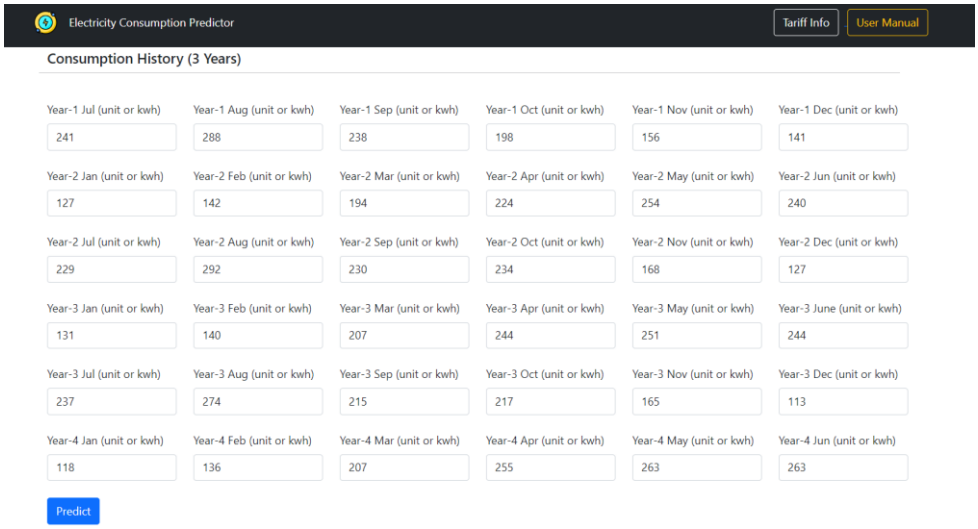

Fig. 5.2: Generating test data - 2

5.1.3 Consumption Forecast Using the Test Data

After generation of manual entry of the test data, the forecast of next twelve months' power consumption can be predicted by clicking the "Predict" button and the bottom of the forecast web application. The forecast output had three parts. The first part consisted of twelve months power consumption forecast (Fig. 5.3), while the second part contains graphical representation of the forecast data and statistical analysis (max, min, average, median, standard deviation) on the power consumption forecast (Fig. 5.4).

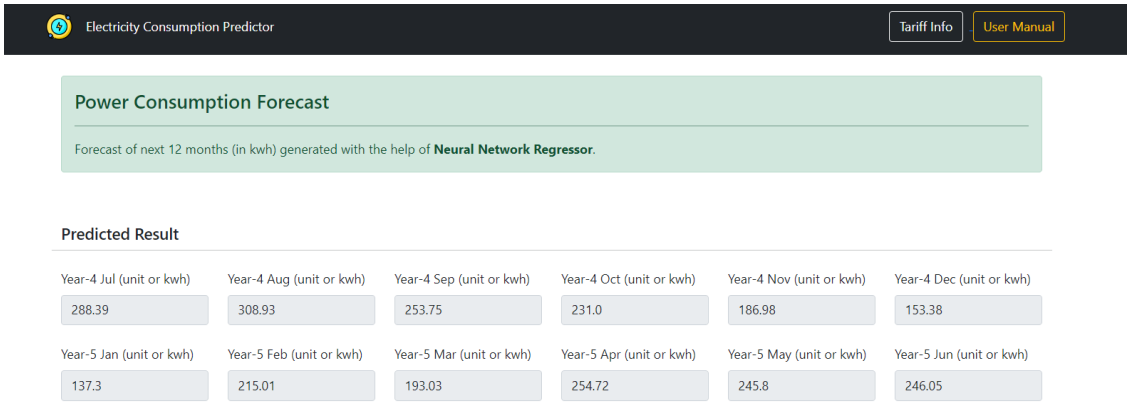

Fig. 5.3: Forecast of twelve months' power consumption in web application

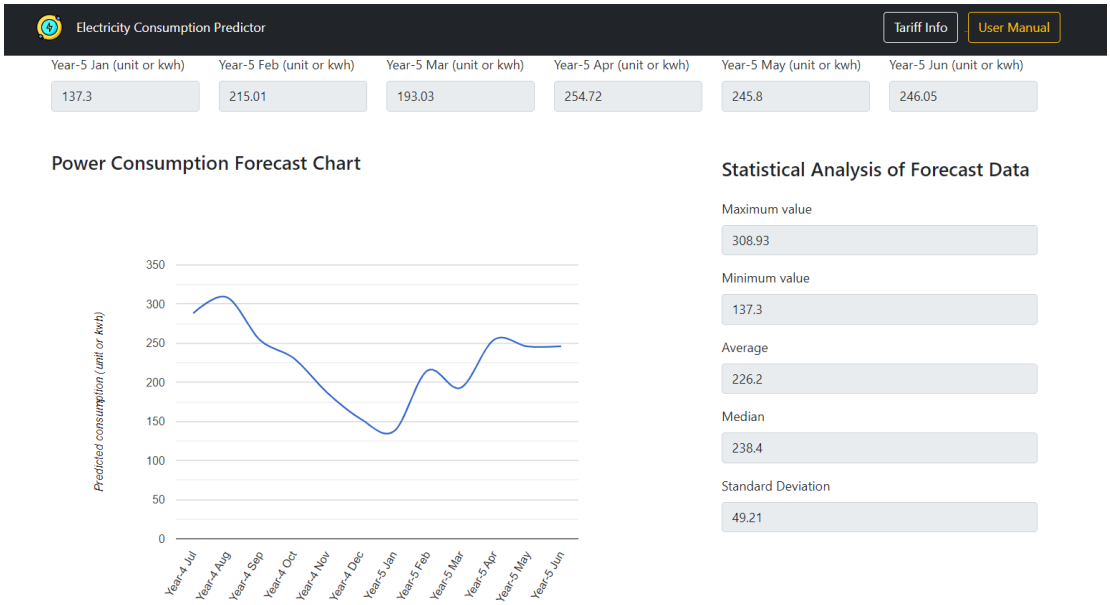

Fig. 5.4: Visualization and statistical analysis of consumption forecast

The data range of the forecast output is consistent with the generated input values and therefore, the system test can be concluded as successful.

5.2 Benchmark Testing of the Web Application

In order to run benchmarking tests on the web application, as test environment was set up with similar specifications as pointed in Table 5.1 for system test. After initial setup of the test system, the following benchmarking tests were conducted.

5.2.1 Browser Compatibility Test

A wide range of web browsers were selected to run the browser compatibility test. Below are the list of browser versions that were checked for loading the web application (Table 5.2)

Table 5.2: Browser compatibility

Browser	Version	Compatible
Google Chrome (PC)	123.0.6312.46 (latest)	Yes
Mozilla Firefox (PC)	123.0.1 (latest)	Yes
Microsoft Edge (PC)	122.0.2365.92 (latest)	Yes
Google Chrome (Android)	123.0.6312.40 (latest)	Yes
Mozilla Firefox (Android)	123.0 (latest)	Yes

Based on the browser compatibility chart, it can be deduced that the web application runs on the widely circulated web browsers in both PC and mobile platform.

5.2.2 Loading Time Test

The data page of the application loads within less than 200 milliseconds, as shown in Fig. 5.5. The HTML page takes 12 milliseconds to load, the other elements take rest of the time. However, after clicking the “Predict” button, the forecast output page takes up to 900 milliseconds (Fig. 5.6) because of the time it takes to load the external google charts library.

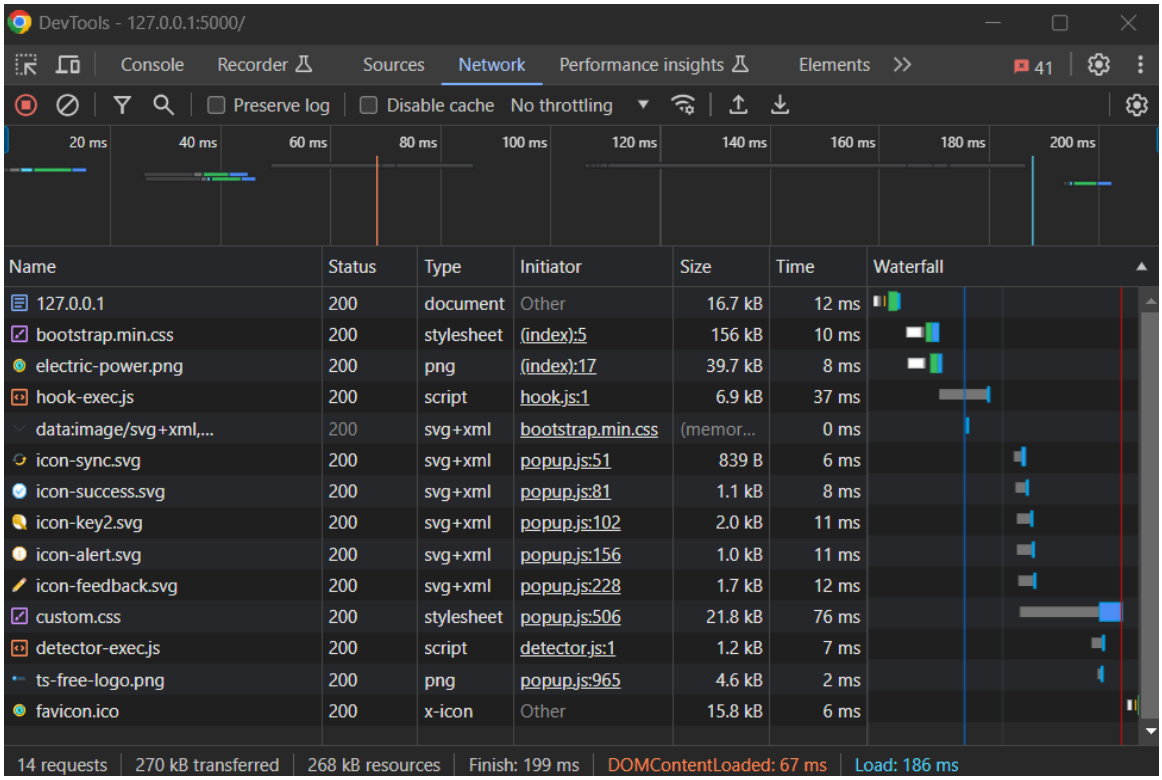

Fig. 5.5: Loading time of the data input page

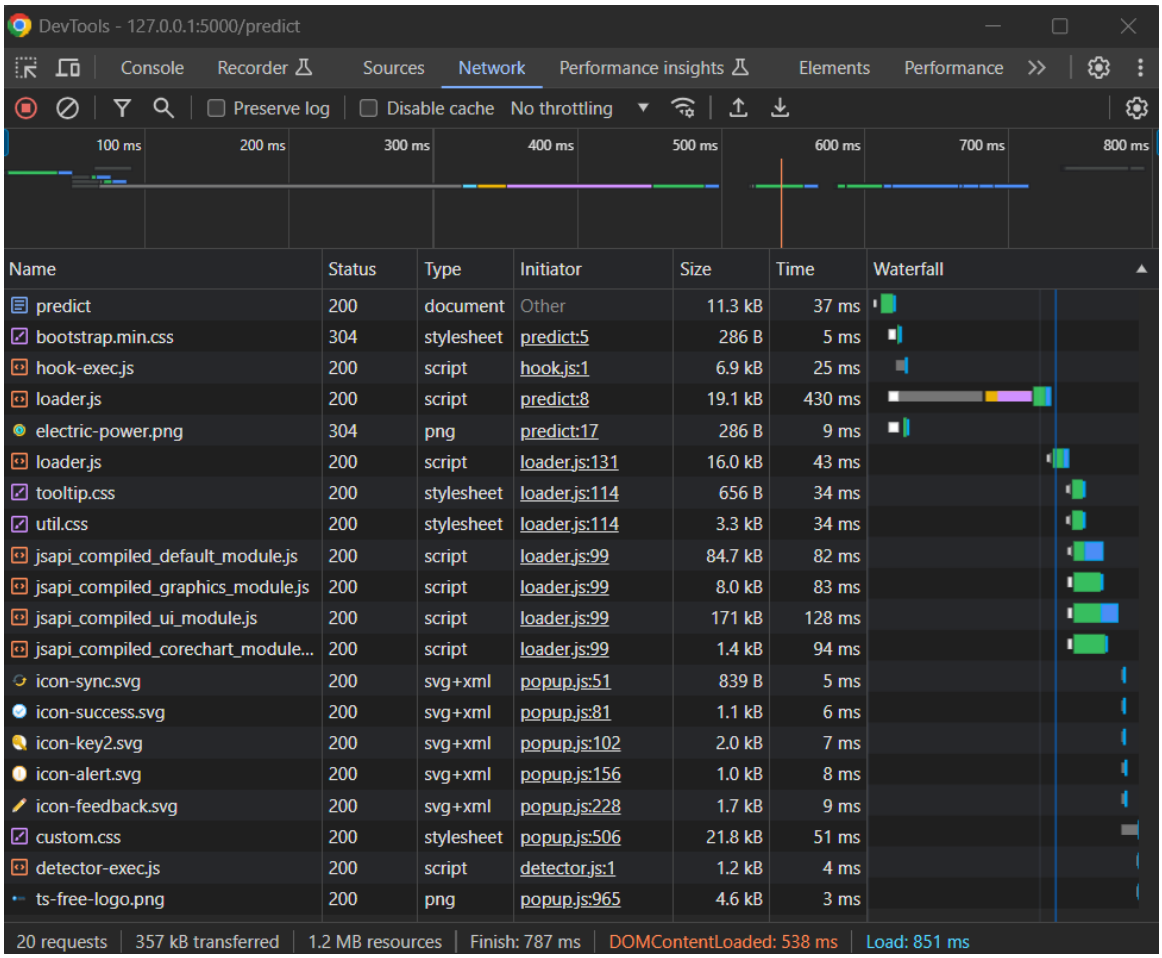

Fig. 5.6: Loading time of the forecast page

Based on the loading time of the application, it can be concluded that, the application is fast enough to maintain user satisfaction.

5.3 Performance Comparison of the Algorithms

The findings presented in Table 5.3 illustrate that both Artificial Neural Networks (ANN) and Support Vector Machines (SVM) consistently exhibit superior performance across multiple evaluation metrics compared to the other three algorithms considered. Specifically, in terms of mean average error (MAE), root mean square error (RMSE), and R-squared error, ANN and SVM consistently outshine their counterparts.

Mean average error (MAE) reflects the average magnitude of errors between predicted and actual values, while root mean square error (RMSE) provides a measure of the deviation between predicted and observed values, accounting for the square of the differences.

Additionally, R-squared error measures the proportion of variance in the dependent variable that is predictable from the independent variables.

The grid search technique was used to try out different combinations of hyperparameters to find the best accuracy with different performance metrics. From the methods employed, the resulting scores were found and put in the following table (Table 5.3) for comparison.

Table 5.3: Performance comparison among the models

Algorithm	MAE	RMSE	R2
XGBoost	127.22073	18566.90318	-16910.15947
LightGBM	165.62486	2089.41156	-213.02929
CatBoost	70.80693	1439.78913	-100.53355
Artificial Neural Network (ANN)	49.19311	114.91548	0.60710
Support Vector Machine (SVM)	51.08429	135.97493	0.43701

However, in terms of the time required while training the model, LightGBM and CatBoost outperforms the other three (Table 5.4).

Table 5.4: Time required for training the models

Algorithm	Time Required for Training the Model (second)
XGBoost	482.96
LightGBM	10.44
CatBoost	35.12
Artificial Neural Network (ANN)	217.56
Support Vector Machine (SVM)	1052.01

Given their consistently superior performance across these critical evaluation metrics, ANN and SVM emerge as the most promising and recommended algorithms for the development of a household electricity consumption prediction system. These algorithms demonstrate robust predictive capabilities and offer reliable forecasts, making them well-suited for

applications where accurate and precise predictions are paramount. Thus, based on the empirical evidence presented, ANN and SVM stand out as the optimal choices for implementing an effective and efficient household electricity consumption prediction system.

After evaluation, the algorithms were ranked for each performance metrices in Table 5.5.

Table 5.5: Overall performance ranking

	MAE	RMSE	R2	Training Time
XGBoost	4th	5th	5th	4th
LightGBM	5th	4th	4th	1st
CatBoost	3rd	3rd	3rd	2nd
Artificial Neural Network (ANN)	1st	1st	2nd	3rd
Support Vector Machine (SVM)	2nd	2nd	1st	5th

It is evident from the table that, ANN and SVM outperforms other three algorithms in MAE, RMSE and R-squared, while LightGBM and Catboost performs better the then the other three in terms of training time. Therefore, the ANN and SVM libraries were chosen to be the recommended algorithms of the household electricity prediction system.

Other Libraries Required to Train the Model:

- Pandas
- Numpy
- Sklearn (scikit-learn)

To implement the system, python was selected as the server-side language because of its compatibility with machine learning libraries.

5.4 Visualization of Output Comparison

In order to validate the accuracy of the algorithms, the month-wise consumption forecasts were generated for July 2022 to June 2023 (Fig. 5.7) with all five algorithms and shown along with the actual consumption data. It can be seen that, in most cases the generated forecasts matched the actual month-wise consumption data. However, the performance of XGBoost, LGBM and CatBoost was severely impacted in the prediction of February 2023

due to the presence of outliers in the dataset, while ANN and SVM were nearly unaffected by the outliers (separately shown again in Fig. 5.8). From the comparison, we can see that, ANN and SVM performs better in unprocessed data while forecasting power consumption.

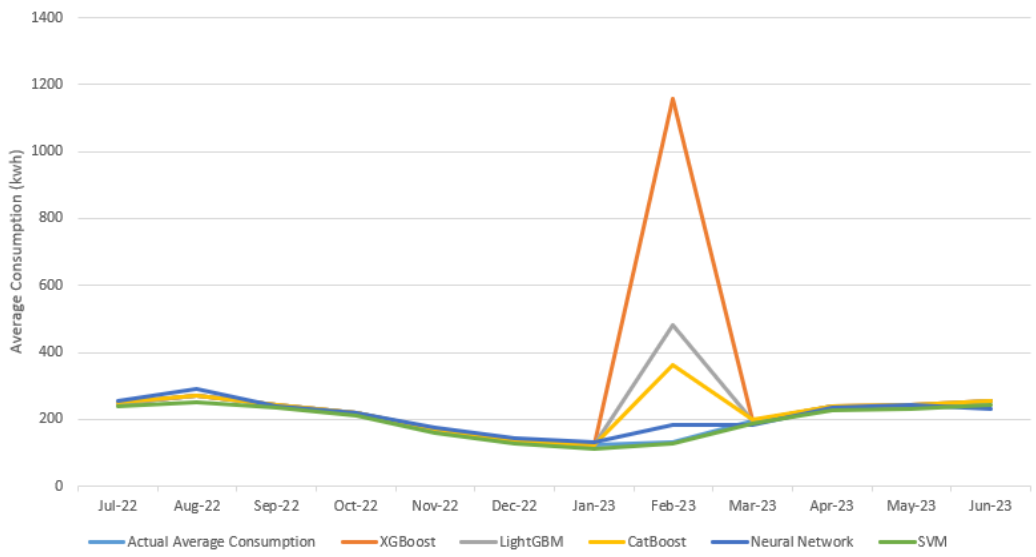

Fig. 5.7: Comparison of output from power consumption forecast models with actual consumption data for all five algorithms.

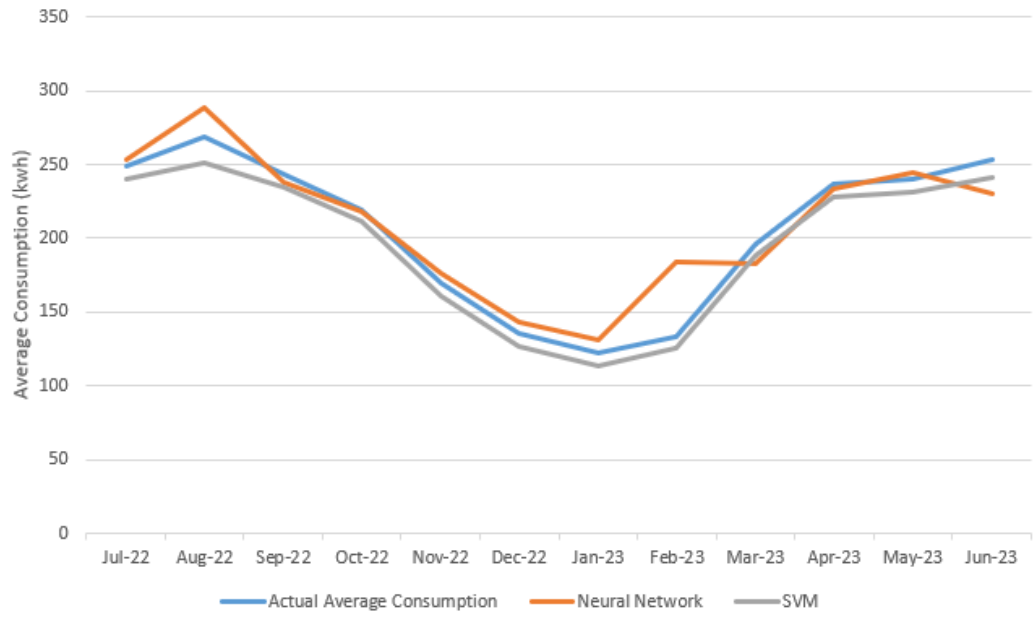

Fig. 5.8: Comparison of output from power consumption forecast models with actual consumption data for Artificial Neural Network and Support Vector Machine (SVM).

5.5 Summary

This chapter shows the comparative performance of all the five algorithms in terms of MAE, RMSE and R2 score. It also compares the algorithms in terms of the time it takes to train the models. We can deduce which algorithms are more susceptible to errors due to outliers and which algorithms are more resilient.

CHAPTER 6
CONCLUSION

6.1 Project Outcome

The primary focus and achievement of this project culminated in the development and implementation of an advanced prediction system utilizing a diverse ensemble of five distinct machine learning algorithms. This sophisticated system operates by ingesting fundamental customer data alongside two years' worth of consumption records, effectively leveraging this comprehensive dataset to generate accurate estimations of monthly consumption for the subsequent year. By harnessing the power of cutting-edge machine learning techniques, our models empower users with the ability to forecast power consumption not only for individual consumers but also for a wide array of consumers within a designated geographical region.

This predictive capability holds immense potential for informing energy management strategies, facilitating demand planning, and optimizing resource allocation across diverse sectors of the economy. Moreover, the versatility and scalability of our prediction system make it a valuable asset for utility companies, policymakers, and researchers alike, offering actionable insights into consumption trends, patterns, and variability at both the micro and macro levels. Through rigorous testing, validation, and refinement, our models have demonstrated their robustness and reliability, paving the way for their widespread adoption and utilization in real-world applications aimed at enhancing energy efficiency and sustainability.

6.2 Project Implication and Contribution

This comprehensive study has yielded significant insights, leading to the development and evaluation of a total of five distinct predictive models, each meticulously crafted and rigorously tested across a multitude of performance metrics.

6.2.1 Technological Contribution

Through systematic comparison and analysis, these models have been thoroughly scrutinized and benchmarked against various matrices, shedding light on their respective

strengths, weaknesses, and areas for improvement. This outcome marks a pivotal advancement in forecasting capabilities, poised to deliver tangible benefits to policymakers and utility companies alike.

6.2.2 Economical Contribution

By providing more accurate and reliable forecasts of power consumption, these models hold the potential to optimize resource allocation, enhance energy management strategies, and ultimately drive greater efficiency and sustainability in the energy sector.

6.2.3 Social Contribution

Furthermore, the findings of this study are expected to serve as a catalyst for further research endeavors focused on understanding and addressing the complexities of power consumption patterns within Dhaka city and the broader context of Bangladesh. Similar work can also be conducted in other utilities like gas, water, etc. As such, the impact of this study extends far beyond its immediate scope, laying the groundwork for future research initiatives aimed at tackling pressing energy-related issues and fostering sustainable development. Forecasting future consumption pattern can lead to creating awareness among consumers to limit electricity usage and help reduce carbon emission.

6.3 Limitations

This study relied on a dataset comprising monthly consumption data from a relatively small sample size of consumers, posing limitations on the breadth and generalizability of the findings. It is evident that expanding the scope of the dataset to encompass a larger and more diverse pool of consumers holds the potential to enhance the predictive capabilities of the system, enabling it to capture a broader range of consumption patterns and trends. Furthermore, a notable challenge encountered in this study pertains to the presence of outliers within the dataset, as illustrated in Figure 10. These outliers can exert a significant influence on the overall performance of the prediction system, potentially skewing results and compromising the reliability of the models. To address this limitation, future research endeavors may explore strategies for outlier detection and removal prior to model training, thereby mitigating their impact and fostering more robust and accurate predictions. Additionally, efforts to refine data preprocessing techniques, such as outlier identification

and data cleaning, could further bolster the performance and efficacy of the predictive models, ultimately advancing our ability to forecast power consumption with greater precision and reliability. By systematically addressing these challenges and refining methodologies, future studies can build upon the foundation laid by this research, paving the way for more comprehensive and insightful analyses of energy consumption dynamics.

6.4 Future Work

The future scopes of work include using daily and hourly consumption collected from smart-meters instead of only monthly consumption. The authors intend to use other algorithms and the upgraded form of the algorithms used in this research to achieve better accuracy and reduced running time.

From the result obtained, it is evident that Artificial Neural Network (ANN) performs better in terms of accuracy compared to other algorithms in the given scenario. It is expected that the consumption forecast algorithms developed can play a vital role in ensuring smooth electricity distribution, greater lifespan of valuable equipment, better policy formulation, reduce system loss, and reduce carbon emission. Power distributors and government can use the insights gained from this study to forecast energy needs and take necessary steps to reduce system loss and load shedding. As the country is gradually shifting towards implementation of smart-grid, forecasting can be an integral part of the smart-grid system. Moreover, similar predictive models can also be developed to forecast the consumptions of gas, fuel, water, etc.

REFERENCES

Abreu, J.M., Pereira, F.C., and Ferrao, P. (2012). Using pattern recognition to identify habitual behavior in residential electricity consumption, Energy and buildings, Vol. 49, pp. 479-487.

Ahmed, Z. E., Hasan, M. K., Saeed, R. A., Hassan, R., Islam, S., Mokhtar, R. A., Khan, S., and Akhtaruzzaman, M. (2020). Optimizing energy consumption for cloud internet of things, Frontiers in Physics, Vol. 8, pp. 358.

Farinaccio, L., and Zmeureanu, R. (1999). Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses, Energy and Buildings, Vol. 30, No. 3, pp. 245-259.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, pp. 1189-1232.

Hino, H., Shen, H., Murata, N., Wakao, S., and Hayashi, Y. (2013). A versatile clustering method for electricity consumption pattern analysis in households, IEEE Transactions on Smart Grid, Vol. 4, No. 2, pp. 1048-1057.

Kaiser, J. (2014). Dealing with missing values in data, Journal of systems integration, Vol. 5, No. 1, pp. 42-51.

Keyno, H.S., Ghaderi, F., Azade, A., and Razmi, J. (2009). Forecasting electricity consumption by clustering data in order to decline the periodic variable's affects and simplification the pattern, Energy Conversion and Management, Vol. 50, No. 3, pp. 829-836.

Khan, N.I., Mahmud, T., Islam, M.N., and Mustafina, S.N. (2020). Prediction of Cesarean Childbirth using Ensemble Machine Learning Methods, Proceedings of the 22nd international conference on information integration and web-based applications & services, pp. 331-339.

Khan, N.S., Muaz, M.H., Kabir, A., and Islam, M.N. (2019). A Machine Learning-Based Intelligent System for Predicting Diabetes, International Journal of Big Data and Analytics in Healthcare (IJBDAH), Vol. 4, No. 2, pp. 1- 20.

Khan, N.S., Muaz, M.H., Kabir, A., and Islam, M.N. (2017). Diabetes predicting mhealth application using machine learning, International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), IEEE, pp. 237-240.

Lin, L., Chen, C., Wei, B., Li, H., Shi, J., Zhang, J., and Huang, N. (2023), Residential electricity load scenario prediction based on transferable flow generation model. Journal of Electrical Engineering & Technology, Vol. 18, No. 1, pp. 99-109.

Liu, C., Li, F., Zhang, C., Sun, B., and Zhang, G. (2023). A dayahead prediction method for high-resolution electricity consumption in residential units, Energy, Vol. 265, pp. 125999.

McLoughlin, F., Duffy, A., and Conlon, M. (2012). Characterising domestic electricity consumption patterns by dwelling and occupant socioeconomic variables: An Irish case study, Energy and buildings, Vol. 48, pp. 240-248.

Montgomery, D.C., Peck, E.A., and Vining, G.G. (2021). Introduction to linear regression analysis, John Wiley Sons.

Mujahid, A.K., and Thirumalai, C. (2017). Pearson correlation coefficient analysis (PCCA) on adenoma carcinoma cancer, International Conference on Trends in Electronics and Informatics (ICEI), IEEE, pp. 492-495.

Oseni, M.O. (2012). Households' access to electricity and energy consumption pattern in Nigeria, Renewable and Sustainable Energy Reviews, Vol. 16, No. 1, pp. 990-995.

Paul, R., Varadhan, S. (2022), Bangladesh plunged into darkness by national grid failure. Reuters, https://www.reuters.com/world/asia-pacific/large-parts-bangladesh-without-power-after-national-grid-failure-daily-star-2022-10-04/, Accessed March 16, 2024.

Perez-Chacon, R., Luna-Romera, J.M., Troncoso, A., Martinez-Alvarez, F., and Riquelme, J.C. (2018). Big data analytics for discovering electricity consumption patterns in smart cities, Energies, Vol. 11, No. 3, pp. 683.

Preetha, S., Manohar, A., HM, A.A., and Tarikere, N.Y. (2020). A REVIEW OF MACHINE LEARNING ALGORITHMS IN HEALTHCARE, International Journal of Recent Trends in Engineering and Research, Vol. 06, No. 05, pp. 44-51.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). CatBoost: unbiased boosting with categorical features, Advances in neural information processing systems, Vol. 31.

Qureshi, K.N., Din, S., Jeon, G., and Piccialli, F. (2020). An accurate and dynamic predictive model for a smart M-Health system using machine learning, Information Sciences, Vol. 538, pp. 486-502.

Ramos, D., Faria, P., Morais, A., and Vale, Z. (2022). Using decision tree to select forecasting algorithms in distinct electricity consumption context of an office building, Energy Reports, Vol. 8, pp. 417-422.

Seber, G.A., and Lee, A.J. (2012). Linear regression analysis, John Wiley Sons, Vol. 329.

Wan, X. (2019). Influence of feature scaling on convergence of gradient iterative algorithm, In Journal of Physics: Conference Series, IOP Publishing, Vol. 1213, No. 3, pp. 032021.

Wang, F., Li, K., Duic, N., Mi, Z., Hodge, B.M., Shafie-khah, M., and Catalao, J.P. (2018). Association rule mining based quantitative analysis approach of household characteristics impacts on residential electricity consumption patterns, Energy conversion and management, Vol. 171, pp. 839- 854.

Watson, N.R., and Miller, A. (2015). Power quality indices.

Yu, H., Chen, L., Yao, J., and Wang, X. (2019). A three-way clustering method based on an improved DBSCAN algorithm, Physica A: Statistical Mechanics and its Applications, Vol. 535, p. 122289.

Zhao, S., Li, W., and Cao, J. (2018). A user-adaptive algorithm for activity recognition based on k-means clustering, local outlier factor, and multivariate gaussian distribution, Sensors, Vol. 18, No. 6, pp. 1850.

Zhou, K., Yang, S., and Shao, Z. (2017). Household monthly electricity consumption pattern mining: A fuzzy clustering-based model and a case study, Journal of cleaner production, Vol. 141, pp. 900-908.

APPENDIX – A

Python Code for Data Pre-processing

```
###### Import packages ########
import pandas as pd
import numpy as np
import glob
import os
from sklearn.model_selection import train_test_split

# File list
path = 'data'
# import data from your file path
all_files = glob.glob(os.path.join(path , "*.xlsx"))

li = []

for filename in all_files:
    df = pd.read_excel(filename, index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)

######## Separate necessary columns ########
selected_cols_consumption = ['Account No.', 'Month', '+A
Increment (kWh)  ']
selected_cols_general = ['Account No.', 'Meter Type', 'Max
Power(kW)', 'Tariff']
X_consumption_only = frame[selected_cols_consumption]
X_general_only = frame[selected_cols_general]

########### Encode by Month ##########
X_consumption_only_encoded =
(pd.get_dummies(X_consumption_only[['Month']])
   .multiply(X_consumption_only['+A Increment (kWh)  '], axis=0)
   .groupby(X_consumption_only['Account No.']).sum()
)

########## Separate max power for each account #########
X_general_only_MP = X_general_only.groupby(['Account
No.']).max(['Max Power(kW)'])
print(len(X_general_only_MP))
print(X_general_only_MP.head())

########### Keeping max value for meter type, max power and
tariff #############
X_general_only_max = X_general_only.groupby(['Account
No.']).max()
```

```
print(X_general_only_max.head())

################ One hot encoding for meter type and tariff
#############
X_general_only_encoded = pd.get_dummies(X_general_only_max)
print(X_general_only_encoded.head())
print(X_general_only_encoded.columns)

######### Convert the index to account no for smooth merging
#####
X_consumption_only_encoded['Account No.'] =
X_consumption_only_encoded.index
X_consumption_only_encoded =
X_consumption_only_encoded.reset_index(drop=True)
X_consumption_only_encoded['Account No.'].head()

############# Merge consumption and general columns together
##############
X_combined = pd.merge(X_general_only_encoded,
X_consumption_only_encoded, on = 'Account No.')
X_combined.head()

########## Drop rows with any 0 value in the consumption columns
######
X_combined = X_combined.loc[X_combined['Month_2019-07'] * ……. *
X_combined['Month_2023-06']

############# Reset index for the dataset ############33
X_combined = X_combined.reset_index(drop=True)
X_combined.head()

############# Drop invalid rows along with test data
#############
X_combined = X_combined.drop(X_combined[X_combined['Account No.']
< 10000000].index)
X_combined = X_combined.drop(X_combined[X_combined['Account No.']
> 99999999].index)
X_combined =
X_combined.drop(X_combined[X_combined['Tariff_Category-T:
Temporary'] == 1].index)
print (len(X_combined))

########## Save the processed dataset for running analysis
###########
X_combined.to_csv('final_processed_data.csv', index=False)

######## Read dataset and separate input-output, test-train
##########
X_proc =
```

```
pd.read_csv('final_processed_data.csv').select_dtypes(include=['n
umber']).copy().drop('Account No.', axis=1)
######## Separate the output columns ##########
output_columns = ['Month_2022-07', 'Month_2022-08', 'Month_2022-
09', 'Month_2022-10', 'Month_2022-11', 'Month_2022-12',
'Month_2023-01', 'Month_2023-02', 'Month_2023-03', 'Month_2023-
04', 'Month_2023-05', 'Month_2023-06']
Y = X_proc[output_columns]
X = X_proc.drop(output_columns, axis=1)
######## Split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
train_size=0.8, random_state=10)

############ Save the train and test files for further use
X_train_df.to_csv('X_train.csv', index=False)
Y_train_df.to_csv('Y_train.csv', index=False)
X_test_df.to_csv('X_test.csv', index=False)
Y_test_df.to_csv('Y_test.csv', index=False)
```

APPENDIX – B

Python Code for Predictive Model with XGBoost

```
###### Import packages ########
import pandas as pd
import numpy as np
import glob
import os
from sklearn.multioutput import MultiOutputRegressor
import xgboost as xgb
import time

####### Read test and train data saved from XGBoost Regressor
X_train = pd.read_csv('X_train.csv')
Y_train = pd.read_csv('Y_train.csv')
X_test = pd.read_csv('X_test.csv')
Y_test = pd.read_csv('Y_test.csv')
X_train.head()

############ Create and train the model
start_time = time.time()
regressor = MultiOutputRegressor(xgb.XGBRegressor(learning_rate =
0.05,
                            n_estimators  = 500,
                            max_depth     = 4,
                            eval_metric='rmsle'))
regressor.fit(X_train, Y_train)

end_time = time.time()

# Print the time elapsed
print(end_time - start_time)

### Prediction and evaluation
predictions = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

RMSE = np.sqrt(mean_squared_error(Y_test, predictions))
MAE = mean_absolute_error(Y_test, predictions)
R2_score = r2_score(Y_test, predictions)
print("RMSE score: %.5f" % RMSE)
print("MAE score: %.5f" % MAE)
print("R2 score: %.5f" % R2_score)
```

```
########### Convert numpy arrays to pandas dataframe for saving
in csv #########
X_train_df = pd.DataFrame(X_train)
Y_train_df = pd.DataFrame(Y_train)
X_test_df = pd.DataFrame(X_test)
Y_test_df = pd.DataFrame(Y_test)
predictions_df = pd.DataFrame(predictions)

########### Save data for overall analysis
predictions_df.to_csv('xgboost_predictions.csv', index=False)
```

APPENDIX – C

Python Code for Predictive Model with CatBoost

```
###### Import packages ########
import pandas as pd
import numpy as np
import glob
import os
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from catboost import CatBoostRegressor
import time

####### Read test and train data saved before
X_train = pd.read_csv('X_train.csv')
Y_train = pd.read_csv('Y_train.csv')
X_test = pd.read_csv('X_test.csv')
Y_test = pd.read_csv('Y_test.csv')
X_train.head()

############ Create and training the model
start_time = time.time()
regressor = MultiOutputRegressor(CatBoostRegressor(learning_rate
= 0.05,
                            iterations  = 500,
                            depth     = 4,
                            ))
regressor.fit(X_train, Y_train)

end_time = time.time()

# Print the time elapsed
print(end_time - start_time)

### Prediction and evaluation
predictions = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

RMSE = np.sqrt(mean_squared_error(Y_test, predictions))
MAE = mean_absolute_error(Y_test, predictions)
R2_score = r2_score(Y_test, predictions)
print("RMSE score: %.5f" % RMSE)
print("MAE score: %.5f" % MAE)
```

```
print("R2 score: %.5f" % R2_score)

########### Convert numpy arrays to pandas dataframe for saving
in csv #########
predictions_df = pd.DataFrame(predictions)

########### Save data for overall analysis
predictions_df.to_csv('catboost_predictions.csv', index=False)
```

APPENDIX – D

Python Code for Predictive Model with LightGBM

```
###### Import packages ########
import pandas as pd
import numpy as np
import glob
import os
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
import lightgbm as ltb
import time

####### Read test and train data saved before
X_train = pd.read_csv('X_train.csv')
Y_train = pd.read_csv('Y_train.csv')
X_test = pd.read_csv('X_test.csv')
Y_test = pd.read_csv('Y_test.csv')
X_train.head()

############ Create and training the model
start_time = time.time()
regressor = MultiOutputRegressor(ltb.LGBMRegressor(learning_rate
= 0.05,
                            n_estimators  = 500,
                            max_depth     = 4,
                            ))
regressor.fit(X_train, Y_train)

end_time = time.time()

# Print the time elapsed
print(end_time - start_time)

### Prediction and evaluation
predictions = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

RMSE = np.sqrt(mean_squared_error(Y_test, predictions))
MAE = mean_absolute_error(Y_test, predictions)
R2_score = r2_score(Y_test, predictions)
print("RMSE score: %.5f" % RMSE)
print("MAE score: %.5f" % MAE)
print("R2 score: %.5f" % R2_score)
```

```
########### Convert numpy arrays to pandas dataframe for saving
in csv #########
predictions_df = pd.DataFrame(predictions)

########### Save data for overall analysis
predictions_df.to_csv('lgbm_predictions.csv', index=False)
```

APPENDIX – E

Python Code for Predictive Model with Artificial Neural Network (ANN) and Saving the Model

```
###### Import packages ########
import pandas as pd
import numpy as np
import glob
import os
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.neural_network import MLPRegressor
import time
import joblib

####### Read test and train data saved from before
X_train = pd.read_csv('X_train.csv')
Y_train = pd.read_csv('Y_train.csv')
X_test = pd.read_csv('X_test.csv')
Y_test = pd.read_csv('Y_test.csv')
X_train.head()

############ Create and training the model
start_time = time.time()
regressor = MultiOutputRegressor(MLPRegressor(random_state=1,
max_iter=500))
regressor.fit(X_train, Y_train)

end_time = time.time()

# Print the time elapsed
print(end_time - start_time)

# Save the trained model
joblib.dump(regressor, "ann_regressor.sav")

### Prediction and evaluation
predictions = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

RMSE = np.sqrt(mean_squared_error(Y_test, predictions))
MAE = mean_absolute_error(Y_test, predictions)
R2_score = r2_score(Y_test, predictions)
```

```
print("RMSE score: %.5f" % RMSE)
print("MAE score: %.5f" % MAE)
print("R2 score: %.5f" % R2_score)

########### Convert numpy arrays to pandas dataframe for saving
in csv #########
predictions_df = pd.DataFrame(predictions)

########### Save data for overall analysis
predictions_df.to_csv('neural_network_predictions.csv',
index=False)
```

APPENDIX – F

Python Code for Predictive Model with Support Vector Machine (SVM)

```
###### Import packages ########
import pandas as pd
import numpy as np
import glob
import os
from sklearn.model_selection import train_test_split
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR
import time

####### Read test and train data saved from before
X_train = pd.read_csv('X_train.csv')
Y_train = pd.read_csv('Y_train.csv')
X_test = pd.read_csv('X_test.csv')
Y_test = pd.read_csv('Y_test.csv')
X_train.head()

############ Create and training the model
start_time = time.time()
regressor = MultiOutputRegressor(SVR(C=1.0, epsilon=0.2))
regressor.fit(X_train, Y_train)

end_time = time.time()

# Print the time elapsed
print(end_time - start_time)

### Prediction and evaluation
predictions = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score

RMSE = np.sqrt(mean_squared_error(Y_test, predictions))
MAE = mean_absolute_error(Y_test, predictions)
R2_score = r2_score(Y_test, predictions)
print("RMSE score: %.5f" % RMSE)
print("MAE score: %.5f" % MAE)
print("R2 score: %.5f" % R2_score)

########### Convert numpy arrays to pandas dataframe for saving
in csv #########
predictions_df = pd.DataFrame(predictions)
```

```
########### Save data for overall analysis
predictions_df.to_csv('svm_predictions.csv', index=False)
```

APPENDIX – G

Server-side Python Code for the Forecast Web Application

```
from flask import Flask, render_template, request
import joblib
import numpy as np

app = Flask(__name__)

model = joblib.load("ann_regressor.sav")

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

@app.route('/predict', methods=['POST'])
def submit():
    data1 = float(request.form['validationServerSL'])
    data2 = 0
    data3 = 0
    if request.form['validationServerMeterType'] == "1P":
        data2 = 1
    elif request.form['validationServerMeterType'] == "3P":
        data3 = 1
    data4 = 0
    data5 = 0
    - - - - - - -

    - - - - - - -
    data19 = 0
    if request.form['validationServerTariff'] == "residential":
        data4 = 1
    elif request.form['validationServerTariff'] == "commercial-
tou":
        data16 = 1
    elif request.form['validationServerTariff'] == "3P":
        data17 = 1

    data20 = float(request.form['validationServer01'])
    data21 = float(request.form['validationServer02'])
    - - - - - - -

    - - - - - - -
    data54 = float(request.form['validationServer35'])
    data55 = float(request.form['validationServer36'])
    arr = np.array([[data1, data2, data3, data4, data5, data6,
data7, data8, data9, data10, data11, data12, data13, data14,
```

```
data15, data16, data17, data18, data19, data20, data21, data22,
data23, data24, data25, data26, data27, data28, data29, data30,
data31, data32, data33, data34, data35, data36, data37, data38,
data39, data40, data41, data42, data43, data44, data45, data46,
data47, data48, data49, data50, data51, data52, data53, data54,
data55]])
    predicted_data = model.predict(arr).round(2)
    # return str(predicted_data)
    return render_template('result.html', data=predicted_data)

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

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

APPENDIX – H

Sample of the Dataset

	Account No.	Max Power(kW)	Meter Type_1P	Meter Type_3P	Tariff_Category-A: Residential
0	2947556	3.0	1	0	1
1	2951311	2.0	1	0	1
2	2954701	5.0	1	0	1
3	2964633	3.0	1	0	1
4	11041971	25.0	0	1	0

...

...

...

Month_2019-09	Month_2019-10	Month_2019-11	Month_2019-12	Month_2020-01	Month_2020-02
80.292	67.629	61.605	33.557	31.689	32.388
249.981	208.977	187.113	119.156	38.828	113.857
155.936	302.995	241.312	173.833	141.513	142.886
134.031	115.082	99.455	59.705	53.431	66.768
0.000	0.000	0.000	0.000	0.000	0.000