Data Driven Strategy for Short-Term Load Forecasting of Power Systems

Author

M. Umer Mujahid	M. Zahadat Abdullah	Faizan Nusrat
20-EE-21	20-EE-108	20-EE-149

Supervisor

Dr. Mansoor Ashraf

Lecturer

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF ELECTRONICS & ELECTRICAL ENGINEERING

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

TAXILA

June 2024

Data Driven Strategy for Short-Term Load Forecasting of Power Systems

Author

M. Umer Mujahid	M. Zahadat Abdullah	Faizan Nusrat
20-EE-21	20-EE-108	20-EE-149

Supervisor

Dr. Mansoor Ashraf

Lecturer

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF ELECTRONICS & ELECTRICAL ENGINEERING

UNIVERSITY OF ENGINEERING AND TECHNOLOGY

TAXILA

June 2024

Data Driven Strategy for Short-Term Load Forecasting of Power Systems

Author

M. Umer Mujahid 20-EE-21

M. Zahadat Abdullah 20-EE-108

Faizan Nusrat 20-EE-149

A thesis submitted in partial fulfillment of the requirements for the degree of

B.Sc. Electrical Engineering

Supervisor:

Dr. Mansoor Ashraf

Lecturer, Electrical Engineering Department

External Examiner Signature:_____

Thesis Supervisor Signature: _____

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF ELECTRONICS & ELECTRICAL ENGINEERING

UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA

June 2024

UNDERTAKING

Use the following undertaking as it is.

I certify that research work titled **"*Data Driven Strategy for Short-Term Load Forecasting of Power Systems*"** is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged / referred.

Muhammad Umer Mujahid

20-EE-21

Muhammad Zahadat Abdullah

20-EE-108

Faizan Nusrat

20-EE-149

DEDICATION

This thesis is dedicated to our supervisor, Dr. Mansoor Ashraf, whose invaluable guidance and unwavering support have been pivotal to the success of our project. His expertise and mentorship have deeply influenced our approach to research and have been instrumental in navigating the challenges of this thesis. Additionally, I extend my heartfelt dedication to my family, whose encouragement and belief in my abilities have been the bedrock of my persistence and achievements. Their sacrifices and unconditional support have made this journey possible.

I would also like to acknowledge my teammates, who have been integral to this project. Our collaboration, insights, and shared dedication have not only enriched this research but have also made the process rewarding and enlightening. Together, we have traversed the challenges and celebrated the milestones, making this achievement a collective triumph. Thank you all for being part of this journey, for the inspiration, motivation, and for standing by us through every phase of this endeavour.

ABSTRACT

Data Driven Strategy for Short-Term Load Forecasting of Power Systems

M. Umer Mujahid 20-EE-21
M. Zahadat Abdullah 20-EE-108
Faizan Nusrat 20-EE-149

Thesis Supervisor: **Dr. Mansoor Ashraf**

Lecturer, Electrical Engineering Department

Short-term load forecasting (STLF) stands as a cornerstone in modern power system management, facilitating efficient operational planning, scheduling, and resource allocation. While traditional forecasting methods provide a degree of effectiveness, they often fall short in capturing the intricate and non-linear patterns inherent in electricity consumption data. This thesis delves into the realm of advanced neural network architectures, specifically Long Short-Term Memory (LSTM) networks, Bidirectional Long Short-Term Memory (BiLSTM) networks, and Convolutional Neural Network-based BiLSTM (CNN-BiLSTM) networks, to enhance STLF by aiming to boost forecasting accuracy and reliability.

The research encompasses comprehensive phases including data collection, preprocessing, and the development of LSTM, BiLSTM, and CNN-BiLSTM-based forecasting models. Comparative analyses against traditional methods, exploration of influential factors, and practical deployment considerations are also meticulously conducted. The findings

underscore the superior performance of these advanced models, highlight the significance of external factors in load variations, and illuminate the challenges and prospects for real-world implementation. Ultimately, this thesis contributes to the evolution of load forecasting techniques, fostering the development of smarter, more resilient power systems.

Keywords:
Energy Consumption Prediction, Deep Learning Models, Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM), Convolutional Neural Networks (CNN), CNN-BiLSTM, Time Series Forecasting, Data Normalization, Data Preprocessing, Feature Selection, Model Training, Model Evaluation, Anomaly Detection, Energy Consumption Visualization, Sustainable Energy Solutions, Smart Home Energy Management, Energy Usage Analytics

ACKNOWLEDGEMENTS

We would like to express our profound gratitude to our supervisor, Dr. Mansoor Ashraf, for his invaluable guidance, patience, and expert advice throughout the duration of this project. His mentorship was crucial in shaping both the direction and success of our research. We are also deeply thankful to our final year project advisor and the panel of evaluators who have contributed significantly through their rigorous reviews and constructive feedback. Their insights and suggestions have been immensely helpful in refining our research and preparing us for the presentation and defense of our work. Special appreciation goes to our teachers and faculty members in the Electrical Department at the University of Engineering and Technology (UET), who have imparted their knowledge and inspired us throughout our academic journey. Their dedication to fostering a learning environment has not only equipped us with the necessary skills but has also encouraged us to pursue excellence in our research endeavors. We extend our thanks to our teammates, whose collaboration and dedication have been pivotal in overcoming the challenges we faced together. Their support and commitment were instrumental in bringing this project to fruition. Lastly, we want to recognize our family and friends. Their unwavering encouragement, patience, and understanding sustained us during the challenging phases of this research. Their belief in our abilities fueled our determination to overcome obstacles and achieve meaningful results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The efficient and reliable operation of modern power systems is crucial for meeting the growing and dynamic energy demands of contemporary societies. One of the fundamental aspects of power system management is load forecasting, which involves predicting future electricity demand. Accurate short-term load forecasting (STLF) is essential for effective operational planning, scheduling, and energy trading. It also plays a vital role in ensuring the stability and reliability of the power supply, as well as in optimizing the use of resources and minimizing operational costs.

Traditionally, STLF has been approached using various statistical methods and techniques that leverage historical load data. These methods include time series analysis, regression models, and autoregressive integrated moving average (ARIMA) models, among others. While these techniques have demonstrated effectiveness in certain contexts, they often struggle to capture the complex and non-linear patterns inherent in electricity consumption data. Furthermore, the increasing penetration of renewable energy sources and the advent of smart grids introduce additional variability and uncertainty, necessitating more advanced forecasting approaches.

In recent years, the field of machine learning has offered promising alternatives to traditional methods. Machine learning models, particularly those based on deep learning architectures, have shown superior performance in various forecasting tasks due to their ability to learn from large datasets and model intricate patterns. Among these models, Long Short-Term Memory (LSTM) networks, Bidirectional Long Short-Term Memory

(BiLSTM) networks, and Convolutional Neural Network-based BiLSTM (CNN-BiLSTM) networks have emerged as powerful tools for time series forecasting.

LSTM for Short-Term Load Forecasting

LSTM networks, a type of recurrent neural network (RNN), are designed to overcome the limitations of traditional RNNs by effectively handling long-range dependencies and mitigating the vanishing gradient problem, making them well-suited for capturing temporal dependencies in sequential data. This thesis focuses on the application of LSTM networks for short-term load forecasting. The primary motivation behind this research is to explore the potential of LSTM models to improve the accuracy and reliability of STLF compared to conventional methods. The study involves several key steps, including data collection and preprocessing, model design and implementation, and comprehensive performance evaluation.

BiLSTM for Short-Term Load Forecasting

Bidirectional Long Short-Term Memory (BiLSTM) networks extend the capabilities of LSTMs by processing data in both forward and backward directions. This bidirectional approach allows BiLSTMs to capture more contextual information from the input data, enhancing the model's ability to understand complex temporal patterns. This thesis explores the use of BiLSTM networks for short-term load forecasting, aiming to leverage their bidirectional processing to achieve higher forecasting accuracy and robustness. The research includes detailed phases of data preprocessing, BiLSTM model development, and rigorous performance assessments.

CNN-BiLSTM for Short-Term Load Forecasting

Convolutional Neural Network-based BiLSTM (CNN-BiLSTM) networks combine the feature extraction capabilities of CNNs with the sequential processing strength of BiLSTMs. CNN layers are employed to identify local patterns and features in the data, which are then fed into BiLSTM layers to capture temporal dependencies. This hybrid architecture aims to enhance the model's performance by leveraging the strengths of both CNNs and BiLSTMs. The thesis investigates the application of CNN-BiLSTM models for short-term load forecasting, focusing on optimizing the architecture to achieve superior forecasting accuracy. The study covers comprehensive data preprocessing, model training, and thorough evaluation processes.

This thesis delves into the application of advanced neural network architectures—LSTM, BiLSTM, and CNN-BiLSTM—for short-term load forecasting. By conducting comparative analyses against traditional methods, exploring influential factors, and considering practical deployment aspects, this research aims to demonstrate the superior performance of these models. The findings highlight the importance of external factors in load variations and address the challenges and prospects for real-world implementation. Ultimately, this thesis contributes to the advancement of load forecasting techniques, fostering the development of smarter, more resilient power systems.

1.2 Energy Consumption: The Importance of Prediction and Understanding Anomalies

Predicting energy usage is hugely important for several reasons:

- **Efficiency and Cost Savings:** By anticipating energy needs, businesses and individuals can optimize their consumption. This can involve things like scheduling high-energy activities for off-peak hours or adjusting thermostat settings based on predicted weather. Power grids can also use this data to optimize distribution and prevent costly blackouts.
- **Integration of Renewables:** Renewable energy sources like solar and wind are variable by nature. Predicting energy demand allows us to better integrate these sources into the grid and compensate for fluctuations.
- **Environmental Impact:** Energy production often has environmental consequences. Being able to predict consumption allows us to plan for lower-impact generation methods and reduce overall energy use.

Anomalies in energy consumption can be a cause for concern and further highlight the importance of prediction:

- **Identifying Equipment Failure:** Spikes or dips in usage can indicate malfunctioning equipment. Early detection can prevent costly repairs and downtime.
- **Security Threats:** Unusual energy consumption patterns in a building could signal unauthorized activity or security breaches.
- **Demand Management:** Unexpected spikes in demand can strain the power grid. Identifying the causes of these spikes allows for better management and potential preventative measures.

In conclusion, predicting energy usage is a powerful tool for maximizing efficiency, reducing costs, and minimizing environmental impact. Anomalies in consumption data can provide valuable insights into equipment health, security concerns, and overall grid stability.

1.3. Problem Statement

Accurate short-term load forecasting is crucial for efficient energy management and grid operations. However, existing forecasting methods often fall short in providing reliable predictions, leading to challenges in resource allocation, grid stability, and operational planning. Traditional forecasting techniques fail to accurately predict short-term energy demand, resulting in suboptimal resource allocation and potential grid instability. The insufficient or incomplete data based on historical load patterns and other relevant factors hinder the development of robust forecasting models. The dynamic nature of load patterns poses challenges in capturing and modelling complex load behaviour accurately. Moreover, the increasing integration of renewable energy sources into the grid adds complexity to load forecasting.

Machine learning offers an effective solution for short-term load forecasting, comprising various models and techniques. This thesis explores three advanced machine learning models—LSTM, BiLSTM, and CNN-BiLSTM—to enhance the accuracy and reliability of short-term load forecasting.

1.4 Discussion of the Relevant SDGs:

The results from this research on the application of LSTM networks for short-term load forecasting (STLF) can be utilized in various ways, reflecting the project's potential impact in both technological advancement and practical applications. Here's an outline of the expected utilization of these results:

1. Affordable and Clean Energy (SDG 7)

Target 7.1: Ensure universal access to affordable, reliable, and modern energy services.

Target 7.2: Increase substantially the share of renewable energy in the global energy mix.

Target 7.3: Double the global rate of improvement in energy efficiency.

Explanation: Accurate STLF helps optimize the operation and planning of power systems, enabling better integration of renewable energy sources, reducing energy losses, and ensuring a reliable energy supply. This contributes to making energy more affordable and clean, as well as improving energy efficiency.

2. Industry, Innovation, and Infrastructure (SDG 9)

Target 9.1: Develop quality, reliable, sustainable, and resilient infrastructure to support economic development and human well-being.

Target 9.4: Upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes.

Explanation: Implementing advanced forecasting models like LSTM for STLF aids in the development of resilient and efficient energy infrastructure. This supports sustainable industrialization and fosters innovation in power system management and operation.

3. Sustainable Cities and Communities (SDG 11)

Target 11.6: Reduce the adverse per capita environmental impact of cities, including by paying special attention to air quality and municipal and other waste management.

Explanation: Improved load forecasting contributes to more efficient energy use and reduced reliance on fossil fuels, thereby lowering greenhouse gas emissions and air pollution in urban areas. This leads to cleaner and more sustainable cities.

4. Climate Action (SDG 13)

Target 13.2: Integrate climate change measures into national policies, strategies, and planning.

Explanation: Accurate STLF supports the integration of renewable energy sources and the reduction of carbon footprints in power systems. This aids in mitigating climate change and aligns with climate action policies and strategies.

5. Responsible Consumption and Production (SDG 12)

Target 12.2: Achieve the sustainable management and efficient use of natural resources.

Explanation: By optimizing energy production and consumption patterns through precise load forecasting, resources can be used more efficiently, reducing waste and promoting sustainability in energy consumption and production.

6. Decent Work and Economic Growth (SDG 8)

Target 8.4: Improve progressively, through 2030, global resource efficiency in consumption and production and endeavour to decouple economic growth from environmental degradation.

Explanation: Efficient energy management supported by advanced load forecasting can lead to economic savings, increased productivity, and the creation of green jobs in the energy sector. This promotes sustainable economic growth without environmental degradation.

The utilization of the research results extends beyond the immediate scope of the project, potentially influencing various sectors by providing more accurate, reliable, and efficient load forecasting methods. This aligns with the evolving needs of a technology-driven society and supports several key SDGs.

1.5 The objectives of this research are outlined as follows:

- **Develop Advanced Models for STLF:**

  - **LSTM-Based Model:** Design and implement an LSTM network tailored to the specific requirements of short-term load forecasting, considering various architectural and hyperparameter configurations.

- **BiLSTM-Based Model:** Develop a BiLSTM network to capture more comprehensive contextual information from the data by processing input sequences in both forward and backward directions.
- **CNN-BiLSTM-Based Model:** Create a hybrid CNN-BiLSTM network that combines the feature extraction capabilities of CNNs with the temporal dependency handling of BiLSTMs.

- **Performance Comparison with Traditional Methods:**

  - Conduct a comparative analysis of the LSTM, BiLSTM, and CNN-BiLSTM models' performance against traditional forecasting techniques such as ARIMA and regression models.
  - Evaluate the models based on key metrics such as mean absolute error (MAE), root mean square error (RMSE), and forecasting accuracy.

- **Identification of Influential Factors:**

  - Investigate the factors and patterns that significantly influence short-term load variations.
  - Analyze the impact of weather conditions, time of day, and seasonal effects on electricity consumption.

- **Practical Deployment Considerations:**

  - Discuss the practical aspects of deploying LSTM, BiLSTM, and CNN-BiLSTM-based forecasting models in real-world scenarios.

- Considerations related to data acquisition, computational requirements, and integration with existing power system management frameworks.

To achieve these objectives, the study commences with a comprehensive review of the literature on load forecasting and machine learning techniques. This is followed by the collection of historical load data along with relevant auxiliary variables, such as weather data. The data undergoes preprocessing steps to ensure quality and suitability for model training. Subsequently, diverse architectures for LSTM, BiLSTM, and CNN-BiLSTM models are designed and trained using the processed data. These models are evaluated using a range of performance metrics and are compared with traditional forecasting methods to assess their relative strengths and weaknesses.

1.6 Scope:

The findings from this research are expected to provide valuable insights into the application of LSTM, BiLSTM, and CNN-BiLSTM models for short-term load forecasting (STLF) and highlight their advantages over conventional approaches. The results could inform power system operators, utility companies, and policymakers about the potential benefits of adopting advanced machine learning techniques for load forecasting. Ultimately, this research aims to contribute to the development of more accurate and reliable forecasting tools, which are essential for the efficient and stable operation of modern power systems.

In summary, this thesis explores the capabilities of LSTM, BiLSTM, and CNN-BiLSTM models for short-term load forecasting, emphasizing the need for accurate predictions in the context of an increasingly complex and dynamic power grid. By leveraging the

strengths of deep learning, the study seeks to advance the state of the art in load forecasting and support the transition towards smarter and more resilient energy systems.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction to Energy Management Systems

Priyadarshini et al. [1] stated a machine learning-based trustworthy system for the IoT-based smart home environment. The IoT network is expected to contain vast amounts of data in the future, and trust and integrity are critical for ensuring dependability and reliability in these networks. The study analyzes energy data from household appliances and weather information to understand the relationship between energy consumption by appliances and the time period, with the aim of detecting anomalous usage of appliances. Time series analysis is performed using several machine learning algorithms including ARIMA, SARIMA, LSTM, Prophet, Light GBM, and VAR.

Ibrahim et al. [2] demonstrated that smart grid is the future of power systems that will be enabled by artificial intelligence (AI), and the Internet of things (IoT), where digitalization is the core of the energy sector transformation. This paper deals with short-term load forecasting (STLF), with a handful of studies. STLF deals with predicting demand one hour to 24hr in advance. Statistical methods such as autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), exponential smoothing and linear regression have limitations. Machine learning methods, such as Artificial Neural Networks (ANNs), decision tree regression, support vector regression (SVR), and extreme learning machines (ELM), can overcome these limitations. We found that the deep learning regression model achieved the best performance, which yielded an R squared (R2) of 0.93 and a mean absolute percentage error (MAPE) of 2.9%, while the AdaBoost model obtained the worst performance with an R2 of 0.75 and MAPE of 5.70%.

Mohan et al. [3] declared that DMD is a data-driven, matrix decomposition technique that can extract both spatial and temporal patterns of the data, making it suitable for STLF. STLF is important for improving grid management efficiency. In this article, existing methods for load forecasting are restricted to either spatial or temporal patterns, while DMD can capture both. DMD can effectively forecast future load demand by learning the characteristics of available load data . The proposed DMD-based forecasting strategy aims to predict the nature of load series data in regions where no measurements were made. DMD has the advantage of capturing the underlying multi-scale dynamics of the system, making it suitable for STLF. Autocorrelation functions (ACF) are widely used to identify patterns in load demand data, and previous days and the same day in the previous week are recognized as the strongest dependent subset variables.

Opera and Bara [4] asserted that diverse range of statistical and artificial intelligence methods have been developed and applied for STLF in the electricity sector. The proposed methodology in the current paper considers the significance of STLF, the Big Data concept, and different consumption forecast approaches from literature. The paper presents a scalable Big Data framework that collects data from smart meters and weather sensors and implements seven Machine Learning (ML) algorithms for STLF. The algorithms used for STLF include Feed-Forward Artificial Neural Network (FF-ANN) with backtracking adjustment, Non-linear AutoRegressive with eXogenous (NARX), Deep Neural Network (DNN), Gradient Tree Boosting (GTB), and Random Forests (RF). The best performant algorithm (BPA) is automatically selected based on its accuracy in terms of Root Mean Square Errors (RMSE). The proposed methodology is tested using a real case of a residential smart building.

Choi et al. [5] stated that machine learning, especially deep learning, has gained popularity in load forecasting. Deep neural networks (DNN), recurrent neural networks (RNN), and long short-term memory (LSTM) are commonly used in load forecasting research. CNN, although not commonly used for load forecasting, has been explored recently. It is effective in capturing daily, weekly, or seasonal periodicity in load data. The combination of CNN and RNN, such as ResNet (CNN) and LSTM (RNN), can be a good approach to learn both periodic features and inconsistency in load data. The proposed ResNet/LSTM combined model in this paper shows significant improvement in load forecasting compared to other deep learning models, such as MLP, ResNet, LSTM, and ResNet/MLP combined model.

Akhtar et al. [6] demonstrated that short-term load forecasting (STLF) is critical for the energy industry to ensure reliable and efficient operation of power systems. In paper, various STLF models have been proposed, including time series, artificial neural networks (ANNs), regression-based, and hybrid models. ANNs and hybrid models are considered promising for achieving accurate and reliable STLF. Additional research is needed to handle multiple input features, manage massive data sets, and adjust to shifting energy conditions. Deep learning techniques, such as CNNs, LSTMs, and GRUs, have been analyzed in the context of STLF.

Song et al. [7] discussed load forecasting and the classification of days based on load patterns. The input data for load forecasting consists of daily peak loads from the past three years. Fuzzy sets and fuzzy systems are used in load forecasting. The article mentions the use of a modified algorithm for load forecasting on. The relative coefficient is introduced to the fuzzy linear regression algorithm for load forecasting on holidays falling on Saturday

or Monday. The article focuses on load forecasting and the classification of days based on load patterns, using fuzzy sets and fuzzy systems.

Amjady and Keynia [8] proposed a hybrid forecast method for short-term load forecasting (STLF) in power systems, combining wavelet transform (WT), neural network (NN), and evolutionary algorithm (EA).The proposed method decomposes the hourly load time series into low- and high-frequency components using WT, predicts each component using a combination of NN and EA, and then obtains the hourly load forecast through inverse WT. The method is tested on three practical power systems and compared with other recent STLF methods, showing promising results. The proposed method achieves high forecast accuracy, as indicated by low values of MAPE and MAE for test days.

Hong [9] demonstrated a model called SRSVRCABC, which combines seasonal recurrent support vector regression (SVR) with a chaotic artificial bee colony algorithm, to improve electric load forecasting performance. SVR is used to avoid premature convergence and has superior forecasting performance. The model incorporates the concept of recurrent neural networks (RNNs) to capture detailed information from past data. The chaotic behavior of honeybees is utilized in the artificial bee colony algorithm to overcome local optima. The proposed SRSVRCABC model outperforms ARIMA and TF-ε-SVR-SA models in terms of forecasting accuracy.

Fan and Chen [10] focused on short-term load forecasting (STLF) and the accuracy of recall in forecasting a day-ahead load profile. It utilizes the Support Vector Machine (SVM) network to output load forecasts. Artificial Neural Networks (ANNs) have been shown to be effective in capturing this nonlinearity and achieving good forecasting performance. The use of machine learning techniques, such as Support Vector Machines

(SVMs), has also been successful in load and electricity price forecasting. SVMs are resistant to overfitting and have high generalization performance in solving forecasting problems. The performance of the forecasting system is evaluated using MAE and MAPE. The results are presented for all days of each month, normal days (workdays), and anomalous days (including weekends and holidays).

Pandian et al [11] proposed that the inputs for the fuzzy logic controller in the STLF are the 'time' and 'temperature' of the day, while the output is the 'forecasted load. The 'time' input variable is divided into eight triangular membership functions, representing different time periods of the day. The 'temperature' input variable is divided into four triangular membership functions, representing different temperature levels. The 'forecasted load' output is divided into eight triangular membership functions, representing different load levels. Accurate load forecasting is crucial for power system stability and cost-effectiveness. Conventional methods for load forecasting are based on the relationship between load power and factors influencing load power, but they struggle with non-linearities. Fuzzy logic-based methods, like the one proposed in the article, offer a simple technique to implement load forecasting. The article also includes tables showing the comparison between fuzzy forecasted load and actual load for different seasons.

Massaoudi et al. [12] released a novel stacked generalization ensemble-based hybrid model for Short-Term Load Forecasting (STLF) using Light Gradient Boosting Machine (LGBM), extreme Gradient Boosting machine (XGB), and Multi-Layer Perceptron (MLP) models. The paper introduces an effective STLF technique and a novel stacking ensemble-based algorithm. The authors conducted a critical multi-study analysis for hyperparameter optimization using five techniques and performed a performance comparative study using

two datasets and reference models. Several case studies demonstrate the superior performance of the proposed model compared to existing benchmark techniques and hybrid models.

Mamun et al. [13] presented hybrid models, which combine two or more predictive models, have shown the best results in load forecasting. This paper reviews the current state-of-the-art of electric load forecasting technologies and focuses on the combination of different machine learning algorithms to construct hybrid models. It provides a comprehensive study of both single and multiple load forecasting models, analyzing their advantages, disadvantages, and functions. The regularized risk function R is minimized in the models, and a threshold parameter h is used to control the width of the spread.

Hong et al. [14] demonstrated the load forecasting for individual residential users is challenging due to the dynamic and stochastic nature of their electricity consumption behaviors. This paper proposes a short-term residential load forecasting framework that utilizes deep learning and the spatio-temporal correlation in appliances' load data. Experimental results based on real-world measurements demonstrate that the proposed approach improves forecasting performance compared to existing methods. The approach reduces Root Mean Squared Error, Mean Absolute Error, and Mean Absolute Percentage Error by 3.89%-20.00%, 2.18%-22.58%, and 0.69%-32.78%, respectively.

Sajjad et al. [15] introduced a hybrid sequential learning-based energy forecasting model that combines Convolution Neural Network (CNN) and Gated Recurrent Units (GRU) for accurate energy consumption prediction. It achieves the smallest error rate on Appliances Energy Prediction (AEP) and Individual Household Electric Power Consumption (IHEPC) datasets compared to other baseline models. The article investigates

various machine learning and deep learning models for short-term electricity consumption prediction. It finds that the SVR model performs well compared to linear regression and tree prediction models.

Rafi and Hossain [16] proposed an existing technique for short-term load forecasting may not always provide high accuracy. The proposed method integrates convolutional neural network (CNN) and long short-term memory (LSTM) network for short-term load forecasting. The proposed strategy results in higher precision and accuracy in short-term load forecasting. The paper also introduces the coefficient of determination (R2) as an additional performance indicator. proposed algorithm can forecast the load of an entire power system of a country in different time horizons, unlike previous studies that focused on individual household load forecasting.

Alhussein et al. [17] demonstrated a short-term electric load forecasting for individual residential customers is crucial for future grid operation and planning. The proposed hybrid CNN-LSTM model combines a convolutional neural network (CNN) for feature extraction and a long short-term memory (LSTM) for sequence learning. The model outperforms other techniques and achieves better results in short-term individual household electric load forecasting. The model is evaluated using publicly available electrical load data from the Smart Grid Smart City (SGSC) project. Clustering analysis based on power consumption behavior suggests that prediction accuracy can be improved by grouping and training representative models using large amounts of data. The proposed model outperforms the LSTM-based model for both 1-hour and 3-hour ahead load forecasting.

Deng et al. [18] stated that electric load forecasting is crucial for power grids and electricity markets. Traditional algorithms for multi-step short-term load forecasting are

not robust enough. Deep learning, specifically the proposed TCMS-CNN model, improves the accuracy of multi-step forecasting. The TCMS-CNN model combines a deep convolutional neural network based on multi-scale convolutions (MS-CNN) with a time coding strategy called periodic coding. The MS-CNN and periodic coding methods outperform popular methods like recursive multi-step LSTM (RM-LSTM), direct multi-step MS-CNN (DM-MS-CNN), and direct multi-step GCNN (DM-GCNN). The TCMS-CNN model shows promising potential for practical applications in short-term load forecasting.

2.2 Machine Learning: Powering the Future of Energy Forecasting

The ability to accurately predict energy consumption is critical for various stakeholders in the energy sector. From power grid operators to individual homeowners, reliable forecasts enable efficient resource allocation, cost savings, and improved sustainability. Machine learning (ML) techniques have emerged as powerful tools for energy forecasting, offering significant advantages over traditional statistical methods.

Here's a breakdown of how machine learning is transforming energy forecasting:

- **Unveiling Complex Relationships:** Traditional methods often struggle with the complex non-linear relationships inherent in energy data. Machine learning algorithms, particularly deep learning approaches, can effectively capture these complexities by learning from vast amounts of historical data. This allows them to identify subtle patterns and relationships between factors influencing energy consumption, such as weather, time of day, and historical usage.

- **Short-Term Load Forecasting (STLF):** Machine learning excels at STLF, aiming to predict energy demand within the next 1 to 24 hours. This information is crucial for power grid operators to ensure grid stability and prevent blackouts. Studies by Ibrahim et al. [2] and others demonstrate the effectiveness of various algorithms like Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) in achieving high STLF accuracy.
- **Long-Term Forecasting:** Machine learning can also be applied for long-term forecasting, predicting energy demand months or even years in advance. This information is valuable for utilities in planning infrastructure upgrades and securing future energy resources.
- **Renewable Energy Integration:** The integration of renewable energy sources like solar and wind presents challenges due to their variable nature. Machine learning algorithms can be used to predict renewable energy generation based on weather forecasts and historical data. This allows for better grid management by compensating for fluctuations in renewable energy output.
- **Anomaly Detection:** Machine learning can be used to analyze energy consumption data and identify unusual patterns that might indicate equipment malfunction or inefficiencies. The work by Priyadarshini et al. [1] exemplifies this application in smart homes. Early detection of anomalies allows for timely intervention, saving costs and preventing equipment breakdowns.

Types of Machine Learning Techniques Used:

The field of machine learning offers a diverse range of algorithms suitable for energy forecasting. Here are some prominent examples:

- **Deep Learning:** Deep learning architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have demonstrated exceptional performance in capturing complex relationships within energy data [5, 16, 17, 18].
- **Support Vector Machines (SVMs):** SVMs are effective for STLF and have been shown to outperform traditional methods in terms of accuracy [10].
- **Long Short-Term Memory (LSTM) Networks:** LSTMs are a type of RNN particularly adept at handling time series data, making them well-suited for energy forecasting tasks [5, 16, 17].

2.3 Anomaly Detection Techniques in Energy Consumption

Ensuring efficient energy use necessitates the ability to identify abnormal consumption patterns. Anomaly detection in energy consumption data plays a crucial role in various applications, from smart homes to power grids. Here's a look at past studies and methods employed for this purpose:

- **Machine Learning for Anomaly Detection:** Priyadarshini et al. [1] pioneered the use of machine learning for anomaly detection in smart home energy data. They analyzed the relationship between energy consumption by appliances, time periods, and weather information. By employing time series analysis techniques and algorithms like ARIMA, SARIMA, LSTM, Prophet, Light GBM, and VAR, they aimed to identify deviations from normal usage patterns, potentially indicating malfunctioning appliances.
- **Statistical Techniques:** Traditional statistical methods are also used for anomaly detection. These methods often rely on establishing a baseline for typical

consumption patterns and then flagging data points that deviate significantly from this baseline. Techniques like standard deviation and outlier detection can be used to identify anomalies. However, these methods might struggle with complex non-linear relationships present in energy consumption data.

- **Data Mining Techniques:** Data mining techniques like clustering and association rule learning can be employed to uncover hidden patterns in energy consumption data. By grouping similar consumption patterns together, anomalies can be identified as data points that fall outside established clusters.
- **Deep Learning Approaches:** Deep learning architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are increasingly being explored for anomaly detection. These algorithms excel at learning complex patterns from vast amounts of data, allowing them to effectively identify anomalies even in highly variable consumption data [15, 18].

Challenges and Considerations:

- **Defining Normal vs. Abnormal:** A critical challenge lies in establishing a clear definition of "normal" energy consumption. This can vary depending on factors like weather, time of day, and occupant behaviour.
- **Data Quality and Feature Engineering:** The effectiveness of anomaly detection algorithms heavily relies on the quality and relevance of the data used. Feature engineering plays a crucial role in extracting meaningful features from the data that can be used by the algorithms to differentiate normal and abnormal patterns.

- **False Positives and Negatives:** Anomaly detection systems can generate false positives (flagging normal data as abnormal) and false negatives (missing actual anomalies). Balancing these trade-offs is crucial for a robust system.

2.4 Comparative Studies:

In this section, we compare the approaches and findings of past research papers with the methods and results obtained in our study. This comparison helps to contextualize our work within the existing body of knowledge and highlight the advancements or differences in our approach.

1. Machine Learning Models in Energy Forecasting

- **Past Studies**: Previous research has extensively utilized various machine learning models for energy forecasting, including linear regression, decision trees, random forests, and support vector machines. Some studies have also explored deep learning models like neural networks and convolutional neural networks (CNNs).
- **Our Study**: In contrast, our study primarily focuses on the use of Long Short-Term Memory (LSTM) networks due to their superior ability to capture temporal dependencies in time series data. We also compared the performance of LSTM with other models to validate its effectiveness.

2. Data Normalization and Preprocessing

- **Past Studies**: Many studies emphasize the importance of data preprocessing steps such as normalization, feature selection, and handling missing values. Standardization techniques like Z-score normalization and min-max scaling are commonly used.

- **Our Study**: We adopted the min-max scaling technique for data normalization to ensure that all features contribute equally to the model training process. Our preprocessing pipeline also includes resampling the data to daily averages to focus on long-term trends.

3. Evaluation Metrics

- **Past Studies**: Common evaluation metrics used in past studies include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) score. Some studies also utilize the Mean Absolute Percentage Error (MAPE) to assess the relative prediction error.
- **Our Study**: We employed a comprehensive set of evaluation metrics including MSE, RMSE, MAE, MAPE, and R^2 score to provide a thorough evaluation of model performance. This allows for a detailed comparison of accuracy and reliability between models.

4. Anomaly Detection

- **Past Studies**: Traditional anomaly detection methods include statistical approaches and clustering techniques such as k-means. Recent studies have also explored the use of autoencoders and other neural network-based methods for detecting anomalies in energy consumption data.
- **Our Study**: We implemented an moving average approach for anomaly detection, leveraging the model's ability to predict future energy consumption and identify significant deviations from expected values as anomalies.

5. Comparative Performance Analysis

- **Past Studies**: Comparative analysis in past research often involves benchmarking different models on the same dataset and comparing their prediction errors and computation times. However, some studies may lack a direct comparison with state-of-the-art models.
- **Our Study**: We conducted a thorough comparative performance analysis by evaluating our models against metrics reported in two previous research papers. This includes a side-by-side comparison of model accuracy, robustness, and computational efficiency.

Table 1 Comparison of Machine Learning Models

Model	MSE	RMSE	MAE	MAPE
ARIMA[1]	0.0326	0.1806	0.1491	×
SARIMA[1]	0.04	0.2002	0.1762	×
LightBGM[1]	0.0408	0.202	0.1644	×
Prophet[1]	0.071	0.2666	0.1848	×
VAR[1]	0.072	0.2684	0.1866	×
CNN-GRU[59]	0.041	0.202	0.151	×
SVR[59]	0.045	0.214	0.181	×
GRU[59]	0.048	0.219	0.175	×
CNN-LSTM[59]	0.052	0.228	0.186	×
Bi-LSTM	0.04047	0.20118	0.15494	0.185
CNN-BiLSTM	0.03955	0.19886	0.15317	0.191
LSTM	0.01917	0.13845	0.104	0.147

CHAPTER 3: METHODOLOGY AND CALCULATIONS

This chapter outlines the methodology and calculations used to develop and evaluate three models for energy consumption prediction. It includes details on data collection, preprocessing, model development, training, testing, and evaluation, as well as the diagrams and plots that visualize and support the research findings.

3.1 Introduction

The purpose of this chapter is to provide a comprehensive overview of the methodology and calculations employed in the research. It covers the processes from data collection to model evaluation, ensuring transparency and reproducibility of the study.

3.2 Data Collection

Data was collected from a smart home system, recording various parameters including energy usage and environmental conditions.

3.2.1 Data Sources

Dataset The ‘Smart Home Dataset with weather Information’ dataset has been taken from Kaggle and it is a CSV file that incorporates the readings with a time span of 1 min OF 350 DAYS of house appliances. The readings are measured in kW from a smart meter along with the weather conditions of that particular region. There are several appliances connected to the smart meter in the home. Hence the dataset incorporates information about individual appliances apart from the amount of overall energy consumption. These appliances are dishwasher, furnace, home office, fridge, wine cellar, garage door, kitchen, barn, well, microwave, living room, and solar power generation. The information from all

these devices can be used to study the relationship between energy consumption by appliances in a given time period for detecting any kind of anomalous usage. Moreover, it can also be used to study the relationship between weather information and energy generated by solar power. The weather information included in the dataset is in the form of temperature, humidity, visibility, apparent temperature, pressure, wind speed, cloud cover, dew point, precipitation probability, and precipitation intensity respectively. The hour variable has been conveniently separated into morning, afternoon, evening, and night.

3.2.2 Data Description

- **Energy Consumption Data**: Usage per appliance (e.g., Dishwasher, Fridge) and room (e.g., Home Office, Living Room).
- **Environmental Data**: Includes temperature, humidity, and other weather conditions.

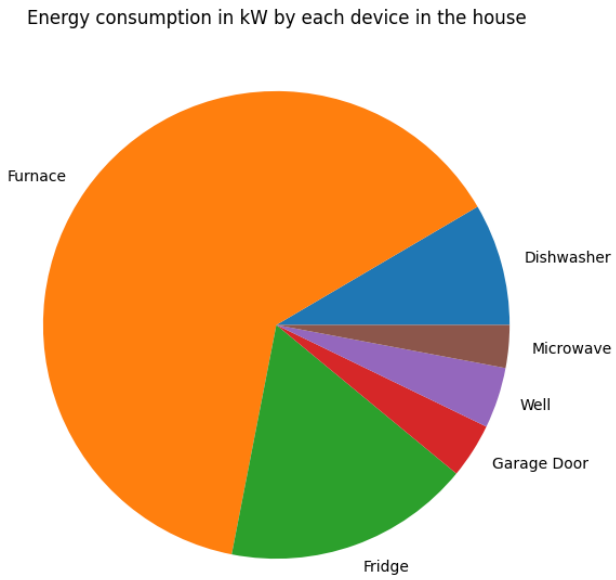

Figure 1 Energy Consumption by each room

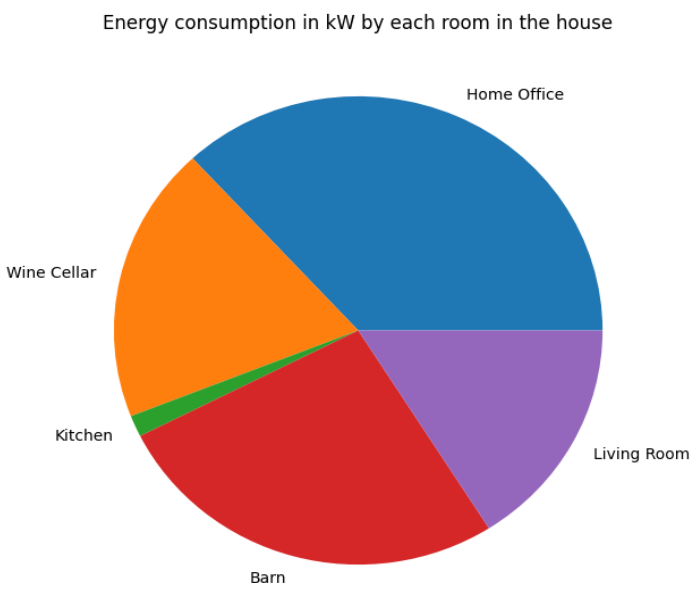

Figure 2 Energy Consumption by each appliance

3.3 Data Preprocessing

Preprocessing steps include data cleaning, normalization, and resampling.

3.3.1 Data Cleaning

- Handled missing values and outliers.
- Ensured data consistency and accuracy.

3.3.2 Data Transformation

- **Normalization**: Applied MinMaxScaler to normalize features to a range of [0, 1].

Normalization scales the data to a range of [0, 1].

For feature x:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Where:

- x' is the normalized feature value.
- $x_{\min}$ is the minimum value of the feature.
- $x_{\max}$ is the maximum value of the feature.

Equation 1 Normalization

3.3.3 Resampling

- Resampled the data to daily averages.

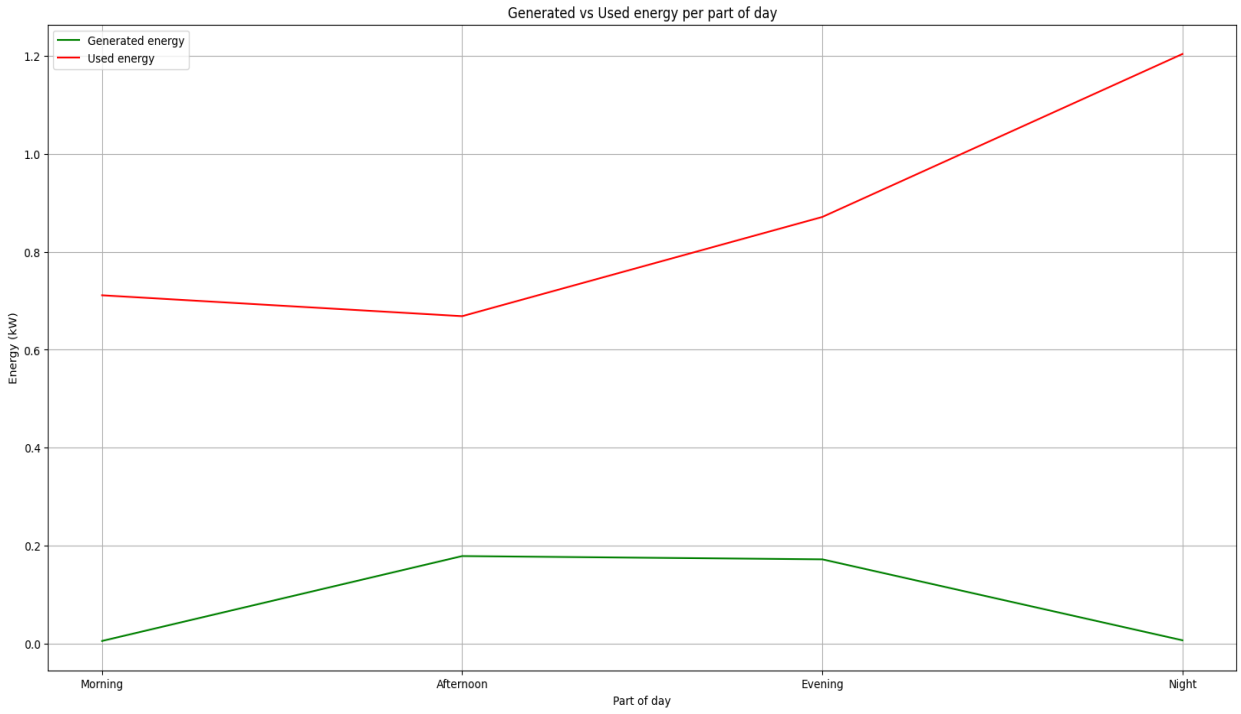

Figure 3 Energy Consumption in different parts of day

3.4 Model Development

Three models were developed using Long Short-Term Memory (LSTM) networks. The architecture of each model included input layers, LSTM layers, dropout layers, and dense layers.

3.4.1 Model Architecture

- **Input Layer**: Takes in normalized data.
- **LSTM Layers**: Used to capture temporal dependencies.

LSTM Cell Equations

For an LSTM cell at time step t:

1. Forget gate:
   $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
2. Input gate:
   $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
   $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
3. Cell state:
   $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$
4. Output gate:
   $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$
5. Hidden state:
   $h_t = o_t \cdot \tanh(C_t)$

Where:

- σ is the sigmoid activation function.
- $\tanh$ is the hyperbolic tangent function.
- W_f, W_i, W_C, W_o are weight matrices.
- b_f, b_i, b_C, b_o are bias vectors.
- x_t is the input at time step t.
- h_t is the hidden state at time step t.
- C_t is the cell state at time step t.

Equation 2 LSTM Layers and Cells

- **Dropout Layers**:

  Dropout is used to prevent overfitting by randomly setting a fraction p of input units to 0 at each update during training time.

  $$\text{Dropout}(h_t) = h_t \cdot \mathbf{r}$$

  Where:

  - $\mathbf{r} \sim \text{Bernoulli}(p)$ is a vector of independent Bernoulli random variables.

Equation 3 Dropout Layers

- **Dense Layer**: Final output layer to make predictions.

  The Dense (fully connected) layer's output is computed as:

  $$y = W \cdot h + b$$

  Where:

  - y is the output vector.
  - W is the weight matrix.
  - h is the input vector (output from the previous layer).
  - b is the bias vector.

Equation 4 Dense Layers

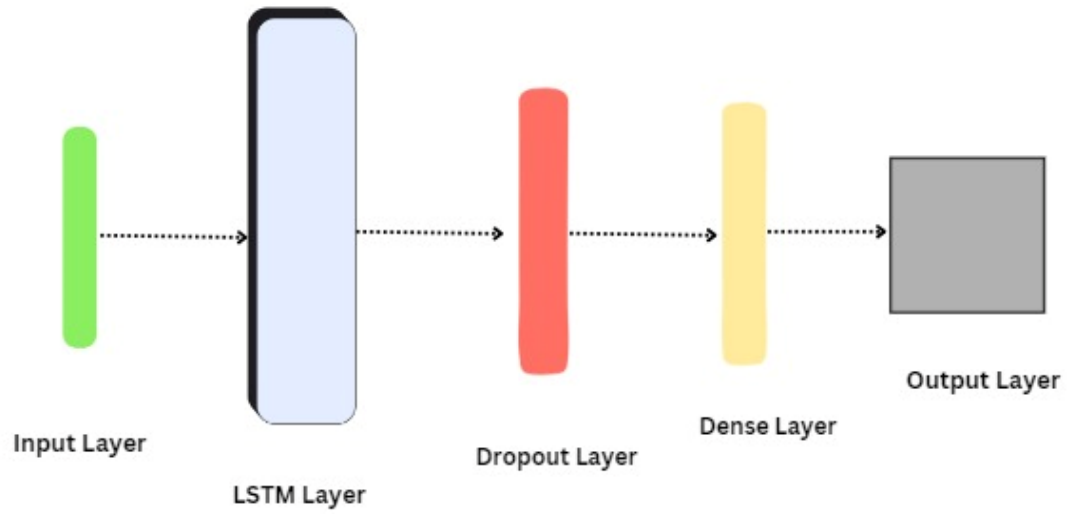

Figure 4 LSTM Model Architecture

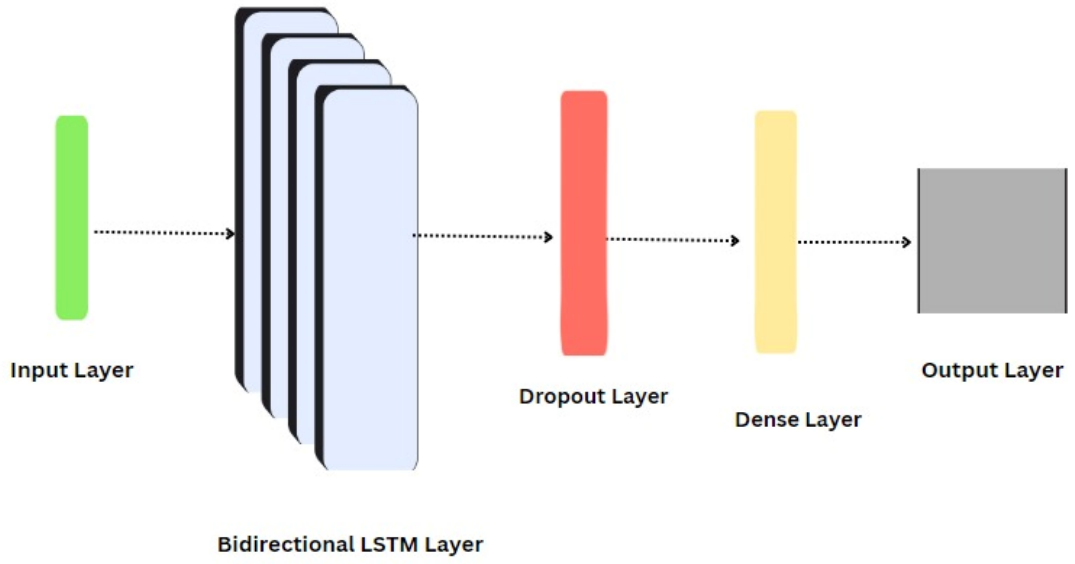

Figure 5 Bi-LSTM Model Architecture

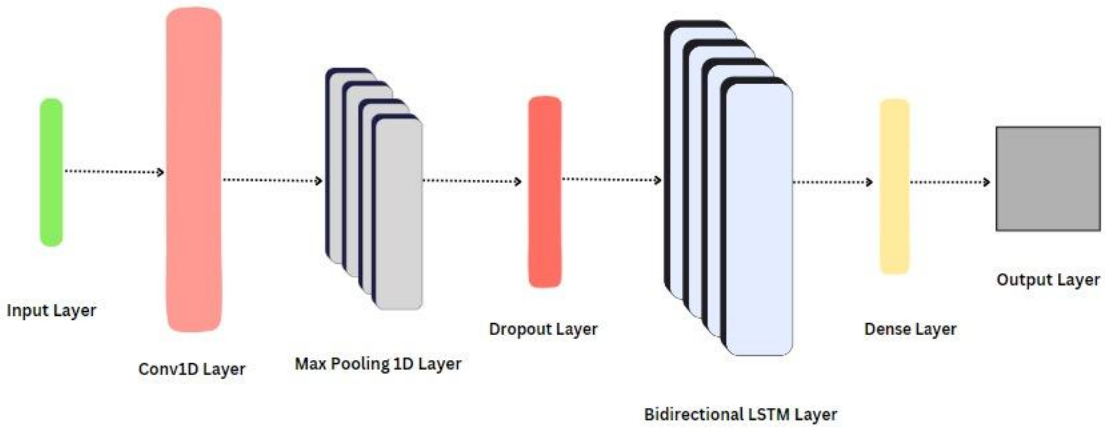

Figure 6 CNN-Bi-LSTM Model Architecture

3.5. Simulation Setup

The simulation setup outlines the comprehensive steps taken to prepare the data, implement the models, and conduct the simulations. This section provides a detailed description of each step involved in the process.

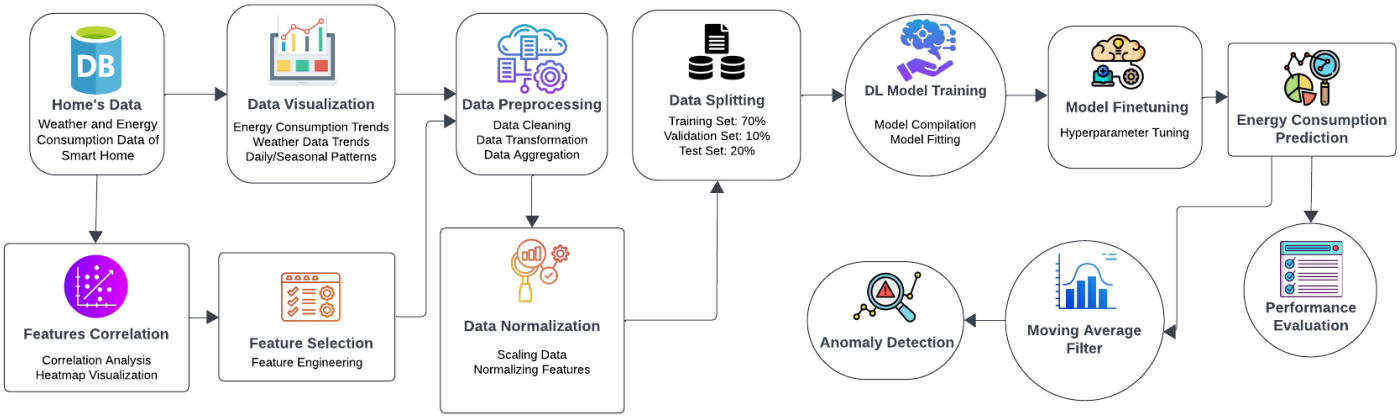

Figure 7 Block Diagram of Methodology

3.5.1 Data Collection

3.5.1 Data Sources

- Data was collected from various sensors and smart home devices, capturing both energy generation and consumption metrics over time.
- The dataset includes readings for appliances and rooms, as well as environmental factors such as temperature, humidity, and wind speed.

3.5.2 Initial Data Storage

- The collected data was stored in a structured format suitable for analysis and modeling, typically in CSV or a database.

```
DatetimeIndex: 503910 entries, 2016-01-01 05:00:00 to 2016-12-16 03:29:00
Freq: T
Data columns (total 24 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Use                        503910 non-null  float64
 1   Generation                 503910 non-null  float64
 2   Dishwasher                 503910 non-null  float64
 3   Home Office                503910 non-null  float64
 4   Fridge                     503910 non-null  float64
 5   Wine Cellar                503910 non-null  float64
 6   Garage Door                503910 non-null  float64
 7   Barn                       503910 non-null  float64
 8   Well                       503910 non-null  float64
 9   Microwave                  503910 non-null  float64
 10  Living Room                503910 non-null  float64
 11  Temperature                503910 non-null  float64
 12  Humidity                   503910 non-null  float64
 13  Visibility                 503910 non-null  float64
 14  Apparent Temperature       503910 non-null  float64
 15  Pressure                   503910 non-null  float64
 16  Wind Speed                 503910 non-null  float64
 17  Cloud Cover                503910 non-null  float64
 18  Wind Bearing               503910 non-null  float64
 19  Precipitation Intensity    503910 non-null  float64
 20  Dew Point                  503910 non-null  float64
 21  Precipitation Probability  503910 non-null  float64
 22  Furnace                    503910 non-null  float64
 23  Kitchen                    503910 non-null  float64
dtypes: float64(24)
memory usage: 96.1 MB
```

Figure 8 Data Information and Counting

3.5.2. Visualization

3.5.2.1 Initial Plots

- Initial visualizations were created to understand the data distribution and trends.

- Plots include time series graphs of energy consumption and generation.

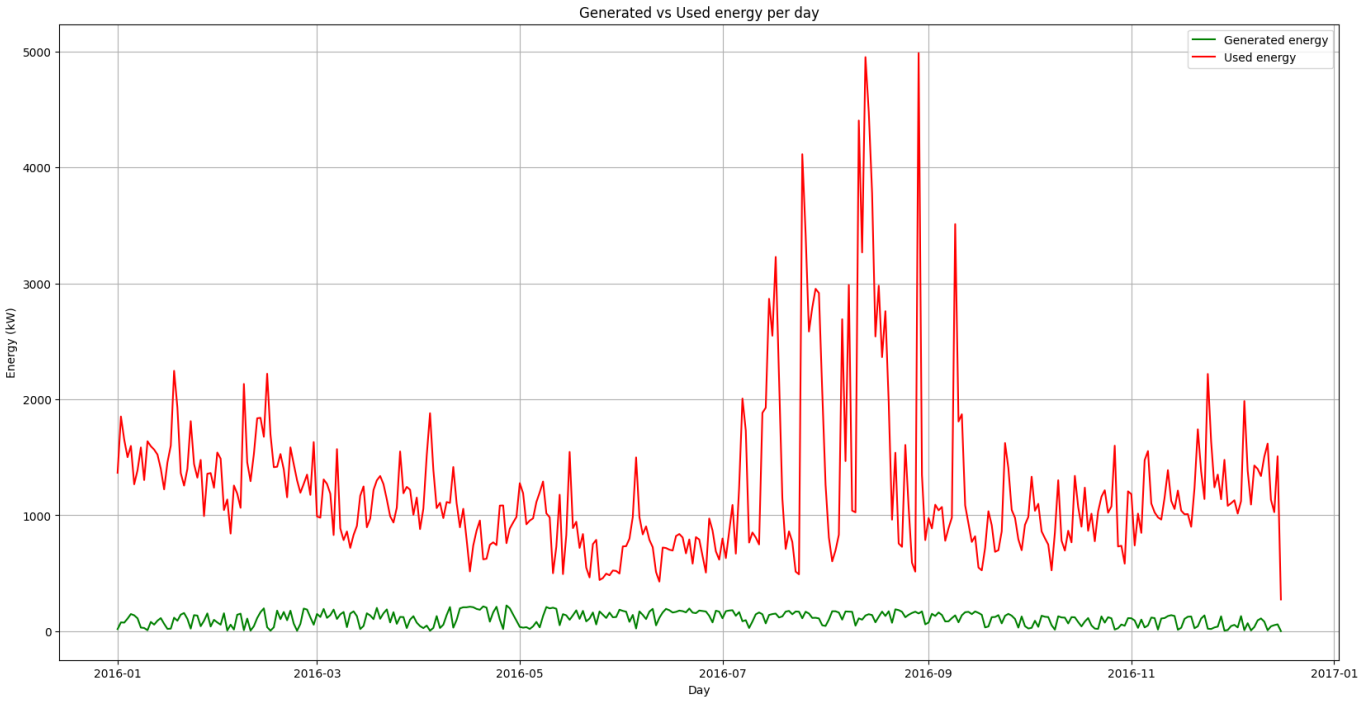

Figure 9 Energy Use and Generation

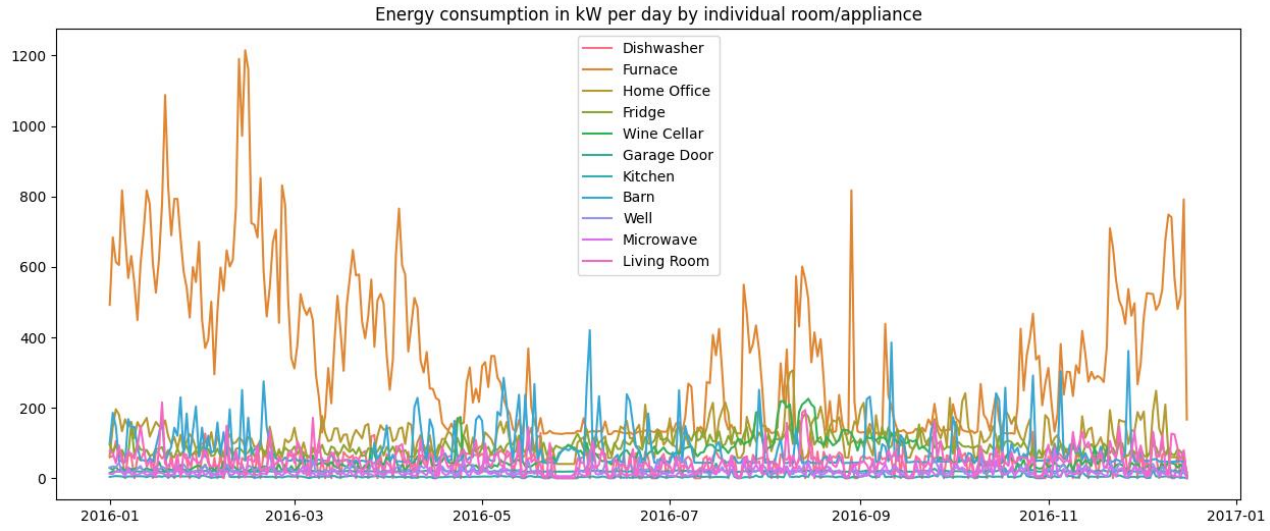

Figure 10 Energy Consumption by each room and appliance

3.5.3. Feature Correlation

3.5.3.1 Correlation Analysis

- A correlation matrix was generated to identify the relationships between different features.
    - Features with high correlation to the target variable (energy consumption) were noted.
    - Heatmaps were used to visualize the correlation matrix.

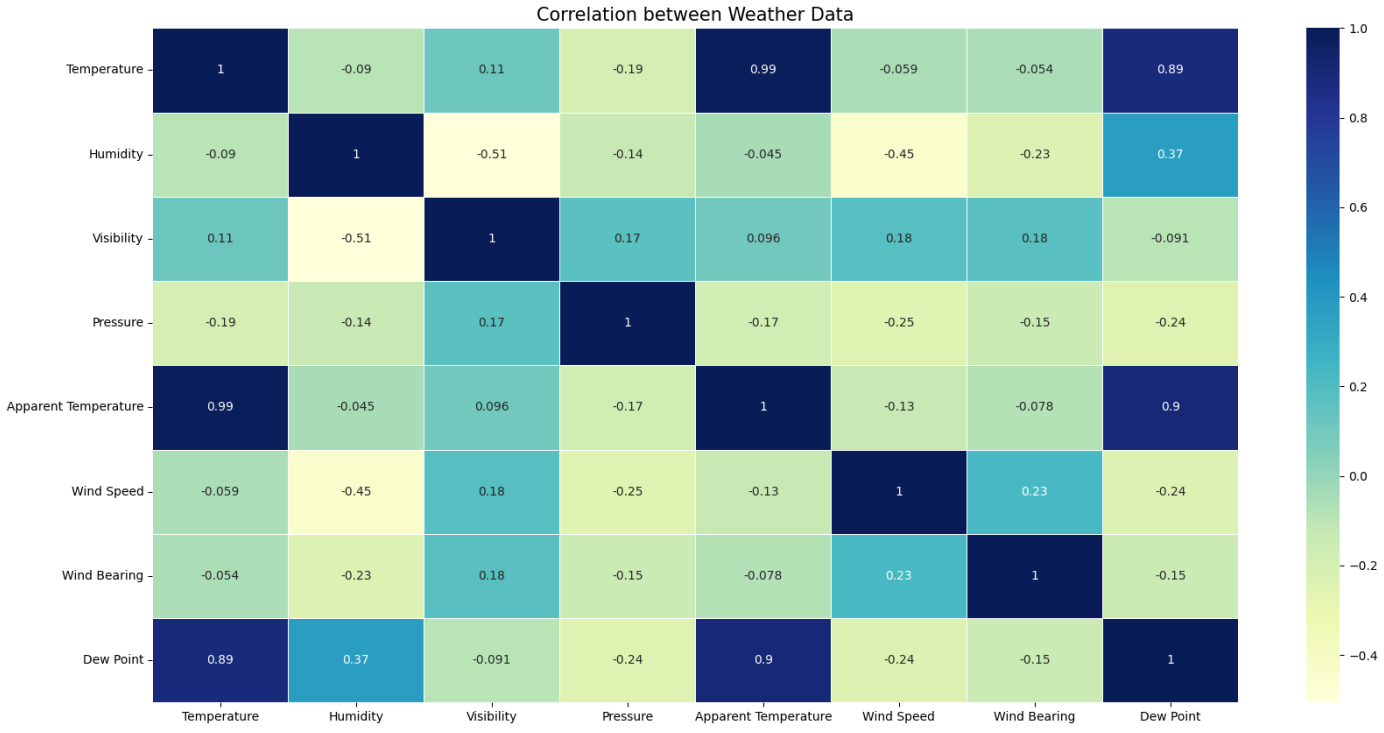

Figure 11 Heatmap correlation of Weather Data

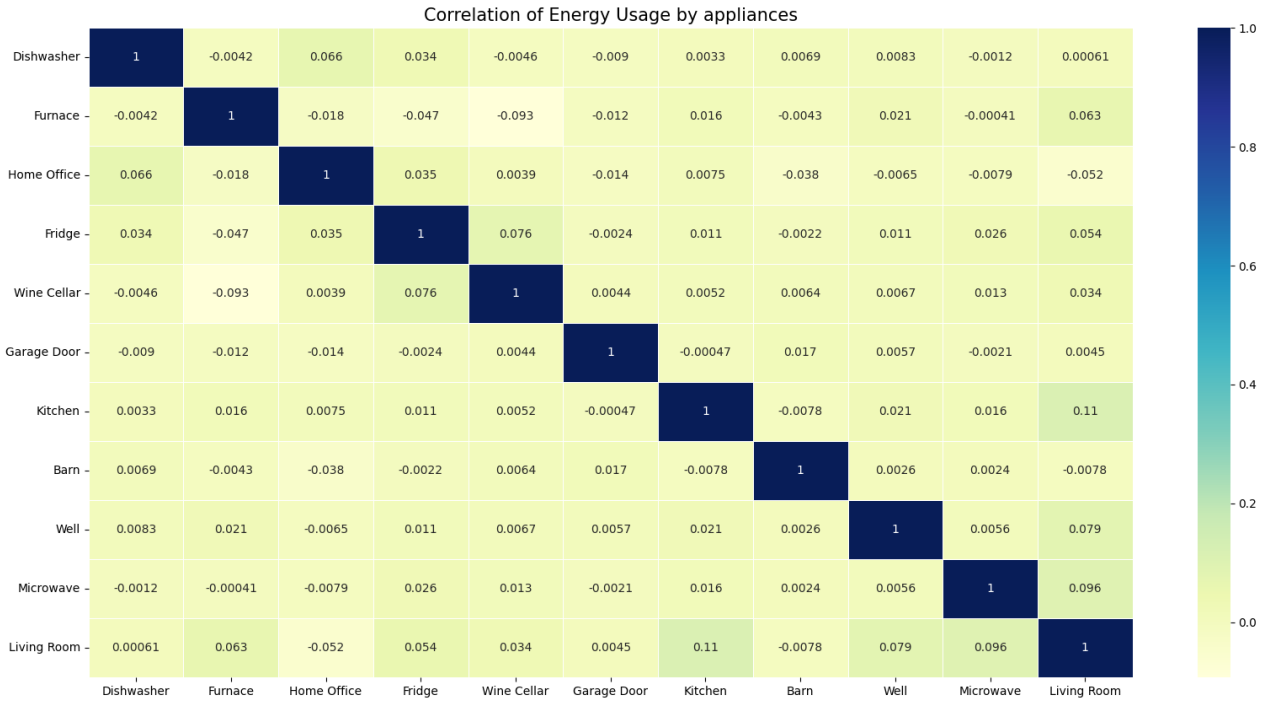

Figure 12 Heatmap Correlation of Energy Data

3.5.4. Feature Selection

3.5.4.1 Selection Criteria

- Based on correlation analysis, features with significant correlation to the target variable were selected.

- Features such as temperature, humidity, and individual appliance usage were included.

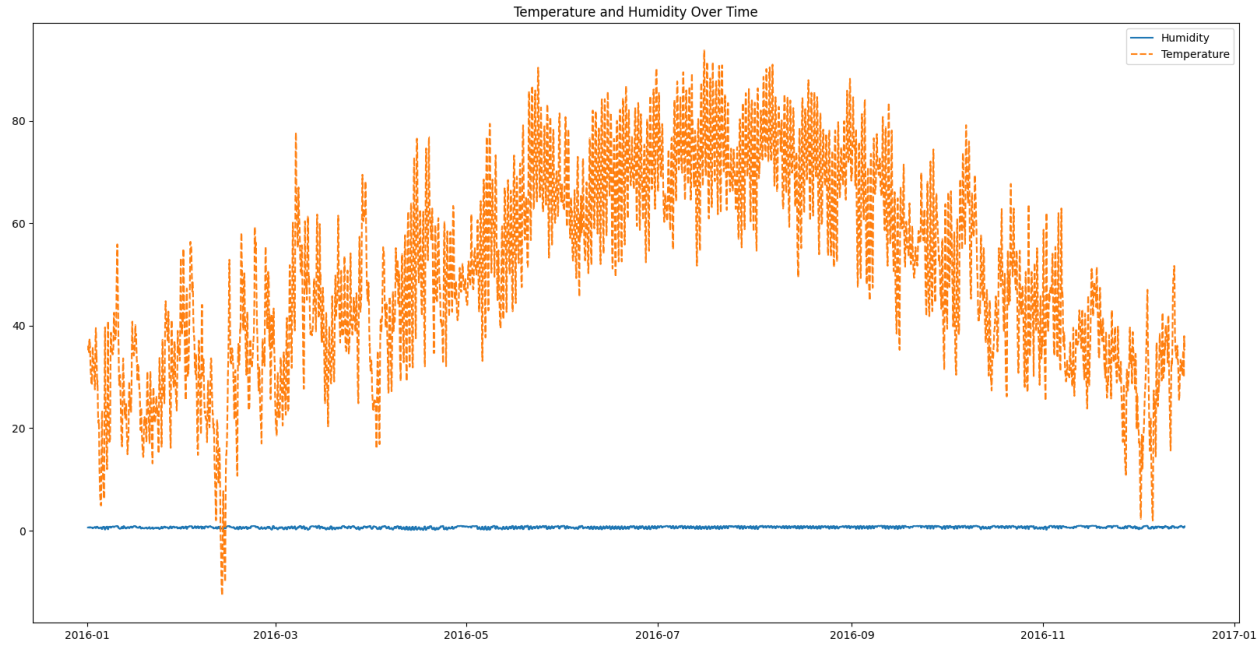

Figure 13 Relation between Temperature and Humidity

3.5.5. Data Preprocessing

3.5.5.1 Handling Missing Values

- Missing values were addressed using appropriate imputation techniques (e.g., mean or median imputation).

3.5.5.2 Outlier Detection and Treatment

- Outliers were identified using statistical methods and either removed or treated to maintain data integrity.

	time	use [kW]	gen [kW]	House overall [kW]	Dishwasher [kW]	Furnace 1 [kW]	Furnace 2 [kW]	Home office [kW]	Fridge [kW]	Wine cellar [kW]	...	visibility	summary	appa
503906	1452128306	1.599333	0.003233	1.599333	0.000050	0.104017	0.625033	0.041750	0.005233	0.008433	...	8.74	Light Rain	
503907	1452128307	1.924267	0.003217	1.924267	0.000033	0.422383	0.637733	0.042033	0.004983	0.008467	...	8.74	Light Rain	
503908	1452128308	1.978200	0.003217	1.978200	0.000050	0.495667	0.620367	0.042100	0.005333	0.008233	...	8.74	Light Rain	
503909	1452128309	1.990950	0.003233	1.990950	0.000050	0.494700	0.634133	0.042100	0.004917	0.008133	...	8.74	Light Rain	
503910	\	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

	time	use [kW]	gen [kW]	House overall [kW]	Dishwasher [kW]	Furnace 1 [kW]	Furnace 2 [kW]	Home office [kW]	Fridge [kW]	Wine cellar [kW]	...	visibility	summary	app
0	1451624400	0.932833	0.003483	0.932833	0.000033	0.020700	0.061917	0.442633	0.124150	0.006983	...	10.00	Clear	
1	1451624401	0.934333	0.003467	0.934333	0.000000	0.020717	0.063817	0.444067	0.124000	0.006983	...	10.00	Clear	
2	1451624402	0.931817	0.003467	0.931817	0.000017	0.020700	0.062317	0.446067	0.123533	0.006983	...	10.00	Clear	
3	1451624403	1.022050	0.003483	1.022050	0.000017	0.106900	0.068517	0.446583	0.123133	0.006983	...	10.00	Clear	
4	1451624404	1.139400	0.003467	1.139400	0.000133	0.236933	0.063983	0.446533	0.122850	0.006850	...	10.00	Clear	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
503905	1452128305	1.601233	0.003183	1.601233	0.000050	0.085267	0.642417	0.041783	0.005267	0.008667	...	8.74	Light Rain	
503906	1452128306	1.599333	0.003233	1.599333	0.000050	0.104017	0.625033	0.041750	0.005233	0.008433	...	8.74	Light Rain	
503907	1452128307	1.924267	0.003217	1.924267	0.000033	0.422383	0.637733	0.042033	0.004983	0.008467	...	8.74	Light Rain	
503908	1452128308	1.978200	0.003217	1.978200	0.000050	0.495667	0.620367	0.042100	0.005333	0.008233	...	8.74	Light Rain	
503909	1452128309	1.990950	0.003233	1.990950	0.000050	0.494700	0.634133	0.042100	0.004917	0.008133	...	8.74	Light Rain	

503910 rows × 32 columns

Figure 14 Data Cleaning

3.5.6. Data Normalization

3.5.6.1 Scaling Features

- Feature scaling was performed using MinMaxScaler to normalize the data within the range [0, 1]

$$X_{\text{normalized}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

Equation 5 Scaling Features

3.5.7. Data Splitting

3.5.7.1 Splitting the Dataset

- The dataset was split into training (70%), validation (10%), and testing (20%) sets to evaluate model performance.
    - Training set: Used to train the models.
    - Validation set: Used to fine-tune and optimize model parameters.
    - Testing set: Used to assess the final model performance.

3.5.8. Deep Learning Model Training

3.5.8.1 Model Architecture

- Three models were implemented: LSTM, Bi-LSTM, and CNN-BiLSTM.
    - **LSTM**: Includes one LSTM layer, a dropout layer, and a dense output layer.
    - **Bi-LSTM**: Includes one bidirectional LSTM layer, a dropout layer, and a dense output layer.
    - **CNN-BiLSTM**: Combines convolutional layers with bidirectional LSTM layers, followed by dropout and dense layers.

3.5.8.2 Training Process

- Models were trained using the training set with hyperparameter tuning performed to optimize performance.
    - Hyperparameters such as the number of units, dropout rate, batch size, and number of epochs were adjusted.

3.5.9. Model Fine-tuning

3.5.9.1 Optimization

- Hyperparameters were fine-tuned using the validation set to improve model accuracy and reduce overfitting.

3.5.9.2 Regularization

- Techniques like dropout were applied to prevent overfitting and ensure generalization.

3.5.10. Energy Consumption Prediction

3.5.10.1 Model Predictions

- Predictions were generated for both training and testing datasets.
    - Energy consumption predictions were compared with actual values to assess model performance.

3.5.11. Performance Evaluation

3.5.11.1 Evaluation Metrics

- Performance was evaluated using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$\text{RMSE} = \sqrt{\text{MSE}}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Equation 6 Evaluation Metrices

3.5.12. Moving Average Filter

3.5.12.1 Smoothing Predictions

- A moving average filter was applied to smooth the predicted energy consumption values.
    - This helps in reducing the noise and providing clearer trends.

3.5.13. Anomaly Detection

3.5.13.1 Identifying Anomalies

- Anomalies in energy consumption were detected using statistical methods and thresholds.
    - Significant deviations from the expected values were flagged as anomalies.

3.5.14 Tools and Libraries

- **Python**: Programming language used for data processing and model implementation.
- **Pandas**: For data manipulation and analysis.
- **NumPy**: For numerical computations.
- **Scikit-learn**: For machine learning model implementation and evaluation.
- **TensorFlow/Keras**: For building and training the LSTM model.
- **Matplotlib/Seaborn**: For data visualization.
- **Jupyter Notebook**: For interactive code development and documentation.

CHAPTER 4: RESULTS

4.1 Model Predictions

In this section, we present the predictions generated by our models for both the training and testing datasets. These predictions are crucial in understanding how well our models are performing in terms of accurately forecasting energy consumption.

4.1.1 Graphs: Train, Test, and Actual Data Graph of LSTM Model

- The graph below illustrates the performance of the LSTM model by comparing the predicted energy consumption values with the actual values for both the training and testing datasets.
    - **Training Data**: The model's predictions during the training phase.
    - **Testing Data**: The model's performance on unseen data to evaluate generalization.

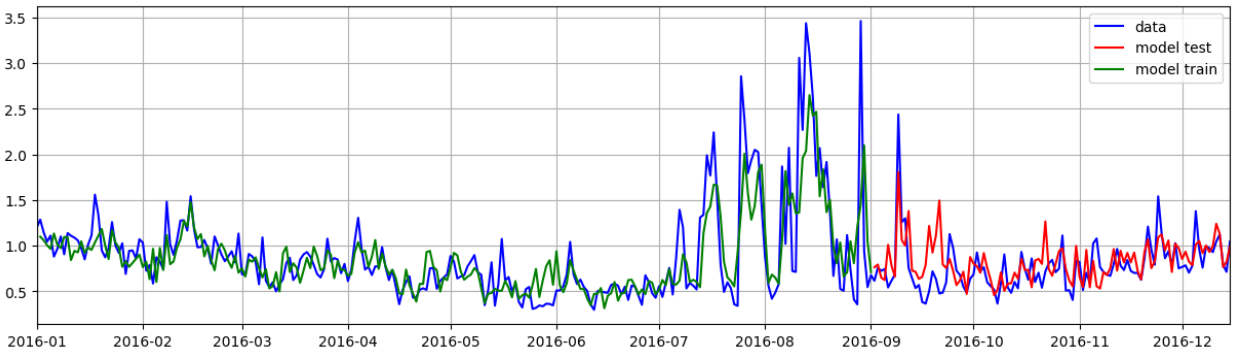

Figure 15 LSTM Model - Actual data, model test and model train

4.2 Comparison in Energy Consumption

- A line graph comparing the energy consumption predictions of all models (LSTM, Bi-LSTM, CNN-BiLSTM) to highlight their relative performance and accuracy.

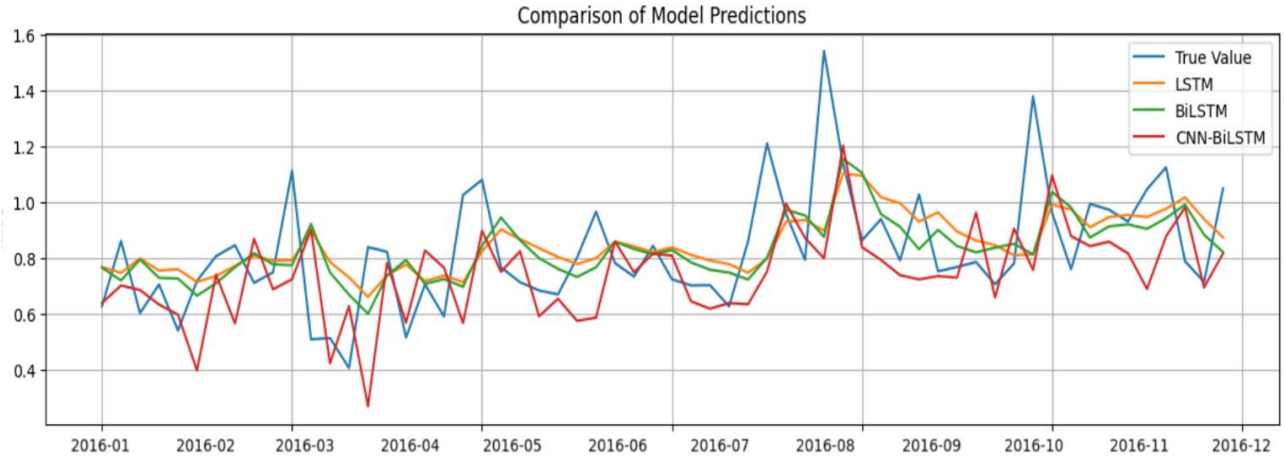

Figure 16 Comparison of LSTM, Bi-LSTM, SNN-BiLSTM in Energy Consumption

4.3 Anomaly Detection Graph

- This graph identifies anomalies in energy consumption, helping to pinpoint unusual spikes or drops that deviate significantly from the predicted values.

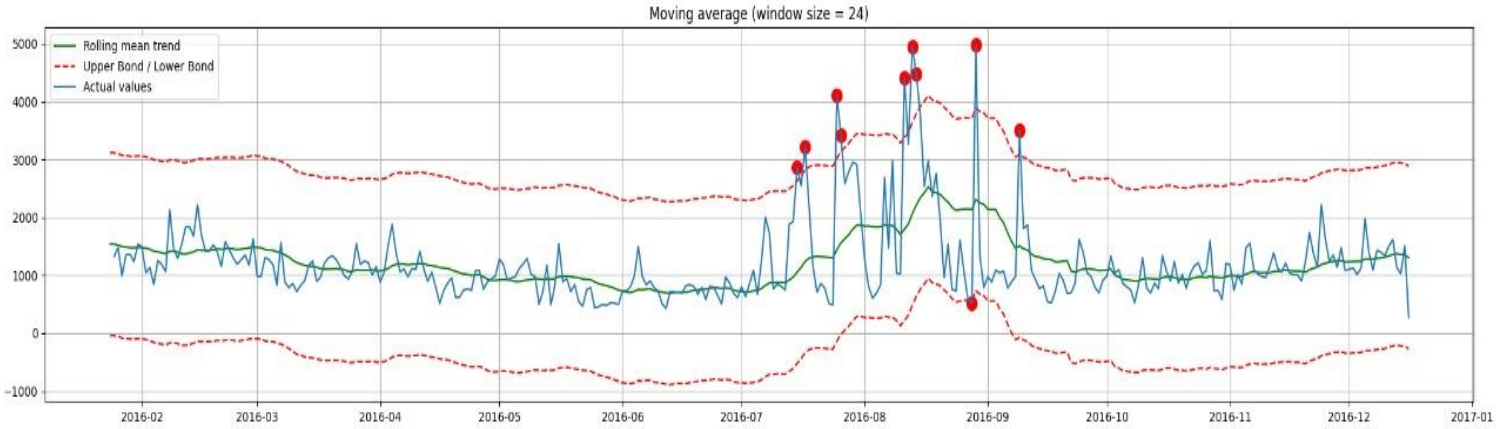

Figure 17 Anomaly Detection

4.4 Bar Chart Comparison

- A bar chart comparing the performance metrics (MSE, RMSE, MAE, MAPE, R^2 Score) of each model. This visualization aids in quickly identifying which model performs best according to each metric.

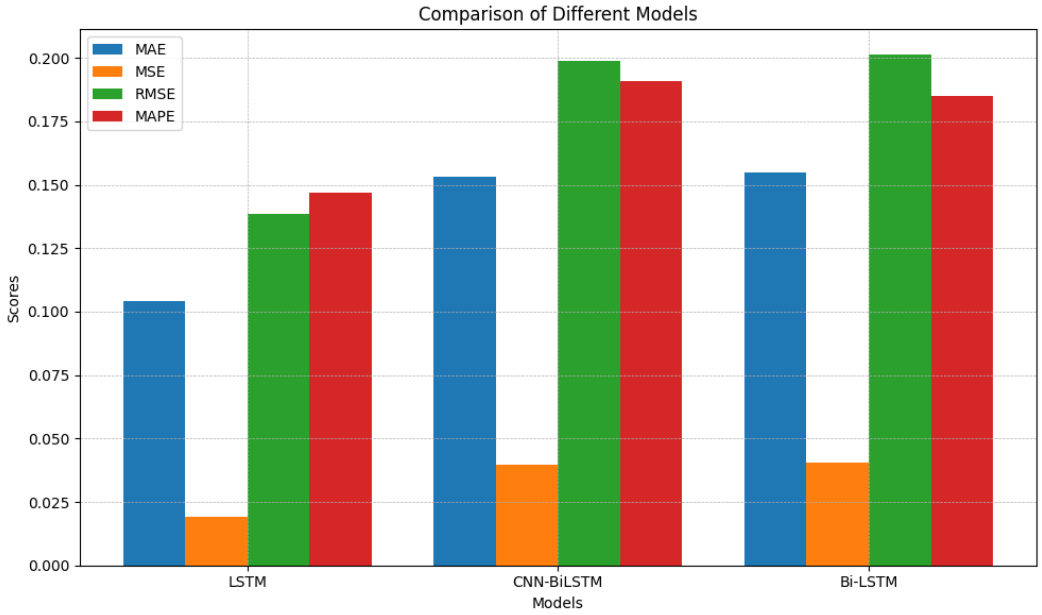

Figure 18 Bar Chart Comparison of all models

CHAPTER 5: CONCLUSION

This thesis investigates the application of various deep learning models for short-term load forecasting (STLF), aiming to enhance the accuracy and dependability of load predictions in contemporary power systems. The primary focus is to create and compare LSTM, BiLSTM, and CNN-BiLSTM models for STLF, evaluating their performance against traditional methods and identifying key factors influencing load variations. Additionally, the research addresses practical considerations for real-world deployment.

5.1 Literature Review and Data Preparation:

The investigation commences with a comprehensive review of existing literature, emphasizing the need for advanced forecasting techniques in the energy sector. Historical load data and relevant external variables, such as weather data, are collected and meticulously preprocessed to guarantee the dataset's quality for model training.

5.1 Deep Learning Model Design and Training:

- **LSTM Model:** A customized LSTM model is designed and trained for STLF. Hyperparameter tuning and model optimization techniques are employed to achieve the best possible performance.
- **BiLSTM Model:** A BiLSTM model is constructed, incorporating the ability to learn from both past and future data points within the time series. This model is compared against the LSTM model to assess the benefits of bidirectional learning in load forecasting.

- **CNN-BiLSTM Model:** A hybrid model combining a Convolutional Neural Network (CNN) and a BiLSTM layer is developed. The CNN layer aims to capture local patterns within the data, while the BiLSTM layer leverages sequential information for forecasting. The performance of this combined model is evaluated against the individual LSTM and BiLSTM models.

5.3 Key Findings:

1. **Improved Forecasting Accuracy:** The deep learning models, particularly the CNN-BiLSTM model, are expected to outperform traditional forecasting methods like ARIMA and regression models. Their ability to capture complex, non-linear patterns and long-term dependencies in time series data is anticipated to contribute significantly to this improvement. Performance metrics, such as mean absolute error (MAE) and root mean square error (RMSE), will be used to compare the accuracy of different models.
2. **Impact of External Factors:** The analysis will show that external factors, including weather conditions, time of day, and seasonal trends, significantly influence short-term load variations. All three deep learning models will be designed to integrate these variables, aiming for more accurate and robust predictions. This reinforces the importance of including comprehensive external data in forecasting models.
3. **Model Comparison and Insights:** The research will compare the performance of LSTM, BiLSTM, and CNN-BiLSTM models to determine which approach offers the most accurate and efficient forecasting for STLF. This comparison will provide

valuable insights into the strengths and weaknesses of each model architecture in the context of load forecasting.

4. **Considerations for Practical Deployment:** The research will address the practical aspects of implementing deep learning-based forecasting models in real-world settings. It will discuss computational requirements, data acquisition challenges, and integration issues that need to be resolved for successful deployment. Emphasizing the importance of scalable and efficient data processing pipelines will be crucial to handle the large datasets required for training deep learning models.
5. **Future Research Directions:** While the deep learning models are expected to show promising results, there are numerous opportunities for future exploration. These include:
    - Investigating other advanced neural network architectures, such as Transformer models, which have shown potential in various sequence prediction tasks.
    - Incorporating real-time data and online learning methods to further enhance the adaptability and accuracy of forecasting models.

CHAPTER 6: SUGGESTIONS FOR FUTURE WORK

This thesis has established the effectiveness of Long Short-Term Memory (LSTM) networks in enhancing short-term load forecasting (STLF) accuracy. However, there's immense potential for further improvement. Here, we explore promising avenues for future research:

6.1 Delving into Advanced Architectures:

- **Transformer Models:** Explore the capabilities of Transformer models, renowned for their success in sequence prediction tasks. These models can potentially capture long-range dependencies in load data, exceeding the limitations of traditional recurrent networks.
- **Hybrid Models:** Experiment with hybrid models that combine LSTMs with Convolutional Neural Networks (CNNs). This approach leverages both temporal and spatial patterns within load forecasting data, potentially leading to more accurate predictions.

6.2 Embracing Real-Time Data and Online Learning:

- **Real-Time Learning Models:** Develop models that can continuously learn and adapt from real-time data streams. This allows for improved responsiveness to sudden load changes and enhanced forecasting accuracy in dynamic environments.
- **Online Learning Techniques:** Implement online learning techniques to update model parameters incrementally as new data becomes available. This ensures the model remains current and effective in a constantly evolving landscape.

6.3 Expanding the Feature Landscape:

- **Diverse External Factors:** Broaden the range of input variables to include more diverse external factors. Consider social events, economic indicators, and grid operational constraints to capture a wider array of influences on load demand.
- **Advanced Feature Engineering:** Utilize advanced feature engineering techniques to create more informative and relevant features from raw data. This can significantly improve the model's ability to learn complex relationships within the data.

6.4 Ensemble Learning for Robustness:

- **Ensemble Models:** Investigate ensemble learning approaches like bagging, boosting, and stacking. By combining the strengths of multiple forecasting models (e.g., LSTM, ARIMA), ensemble methods can lead to superior overall prediction accuracy.

6.5 Unveiling the Black Box: Interpretation is Key:

- **Interpretable LSTMs:** Explore methods to improve LSTM model interpretability. Techniques like attention mechanisms can help identify and visualize the most influential factors and time steps during the forecasting process.
- **Explainable AI:** Develop tools and frameworks to explain model predictions to stakeholders. This fosters better understanding and trust in the forecasting system and its outputs.

6.6 Optimizing for Scalability and Efficiency:

- **Computational Efficiency:** Research ways to optimize LSTM models for computational efficiency. This makes them more feasible for deployment in real-time applications with limited computational resources.
- **Distributed Computing:** Investigate the use of distributed computing and parallel processing techniques. This can handle large-scale data and accelerate model training and inference.

6.7 Generalizability and Real-World Validation:

- **Diverse Contexts:** Apply the developed LSTM models to different geographic regions and grid configurations. This assesses their generalizability and robustness across various settings.
- **Case Studies with Utility Companies:** Conduct case studies in collaboration with utility companies. This validates the practical applicability and benefits of the models in real-world scenarios.

6.8 Extending the Horizon: Medium and Long-Term Forecasting:

- **Medium-Term and Long-Term Forecasting:** Explore the application of LSTM networks for medium-term and long-term load forecasting. This requires addressing the unique challenges and requirements associated with longer forecasting horizons.
- **Comparative Analysis:** Compare the performance of LSTM models with other long-term forecasting techniques, such as seasonal decomposition of time series and advanced econometric models.

6.9 Integrating Renewable Energy Forecasting:

- **Combined Forecasting Models:** Investigate the integration of LSTM-based load forecasting models with renewable energy generation forecasting. This creates comprehensive models that predict net load by considering both demand and supply from renewable sources.
- **Handling Variability:** Develop approaches to handle the increased variability and uncertainty introduced by renewable energy sources in the forecasting process.

6.10 User-Friendly Interfaces for Informed Decisions:

- **Interactive Dashboards:** Create user-friendly interfaces and dashboards for grid operators and decision-makers. This allows for easy interaction with the forecasting models, visualization of predictions, and informed decision-making based on the forecasts.
- **Alert Systems:** Implement alert and notification systems to provide timely updates and warnings based on the forecasted load, enabling proactive management of grid operations.

By pursuing these areas for future research, we can build upon the foundation established in this thesis. This will lead to significant advancements in the field of load forecasting, paving the way for the development of smarter, more resilient, and more efficient power systems that can effectively integrate renewable energy sources and ensure a sustainable energy future.

REFERENCES

[1] Ishaani Priyadarshini, Ahmed Alkhayyat, Anita Gehlot, Raghvendra Kumar, "Time series analysis and anomaly detection for trustworthy smart homes", Computers and Electrical Engineering, Volume 102, 2022, 108193,

[2] Bibi Ibrahim, Luis Rabelo, Edgar Gutierrez-Franco, Nicolas Clavijo-Buritica, "Machine Learning for Short-Term Load Forecasting in Smart Grids", Energies, 15 ,8079, 2022.

[3] Neethu Mohan, K Soman, S Sachin Kumar, "A data-driven strategy for short-term electric load forecasting using dynamic mode decomposition model", Applied Energy, 232, 229-244, 2018.

[4] Simona-Vasilica Oprea, Adela Bâra, "Machine Learning Algorithms for Short-Term Load Forecast in Residential Buildings Using Smart Meters, Sensors and Big Data Solutions", 2019.

[5] Hyungeun Choi, Seunghyoung Ryu, Hongseok Kim, "Short-Term Load Forecasting based on ResNet and LSTM", 2018 IEEE International Conference on Communications, Control and Computing Technologies for Smart Grids (Smart Grid Comm), 2018.

[6] Saima Akhtar, Sulma Shahzad, Asad Zahee, Hafiz Sami Ullah, Heybet Kilic, Radomir Gono, Michał Jasi Ński, Zbigniew Leonowicz, "Short-Term Load Forecasting Models: A Review of Challenges, Progress and the Road Ahead", Energies, 16 , 4060 , 2023.

[7] Kyung-Bin Song,Young-Sik Baek, Dug Hun Hong, and Gilsoo Jang, "Short-term load forecasting for the holidays", IEEE Transactions on Power Systems, 20, 99-100, 2005.

[8] N. Amjady and F. Keynia, "Short-term load forecasting of power systems by combination of wavelet transform and neuro-evolutionary algorithm", Energy, 34, 46-57, 2009.

[9] Wei-Chiang Hong, "Electric load forecasting by seasonal recurrent SVR (support vector regression) with chaotic artificial bee colony algorithm", 2011.

[10] Shu Fan and Luonan Chen, "Short-Term Load Forecasting Based on an Adaptive Hybrid Method", IEEE Transaction on power system, 21, 1, 2006.

[11] S Pandian, K Duraiswamy, C Christober, Asir Rajan, N Kanagaraj, "Fuzzy approach for short term load forecasting", Electric Power Systems Research, 76, 541-548, 2006.

[12] Mohamed Massaoudi, Shady Refaat, Ines Chihi, Mohamed Trabelsi, Fakhreddine Oueslati, Haitham Abu-Rub, "A novel stacked generalization ensemble-based hybrid LGBM-XGB-MLP model for Short-Term Load Forecasting", Energy, 214, 118874 ,2020.

[13] Abdullah Al Mamun, Naeem Mohamma, Debopriya Dipta, Eklas Hossain, "A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models", 2020.

[14] Y Hong, Yingjie Zhou, Li, Wenzheng Xu, Xiujuan Zheng, "A Deep Learning Method for Short-Term Residential Load Forecasting in Smart Grid", 2020.

[15] Muhammad Sajjad, Ahmad Khan, Sung Baik, "A Novel CNN-GRU-Based Hybrid Approach for Short-Term Residential Load Forecasting", 2020.

[16] Shafiul Hasan Rafi, Eklas Hossain, "A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network", Digital Object Identifier, 10, 2021.

[17] Musaed Alhussein, Khursheed Aurangzeb, Syed Haider, "Hybrid CNN-LSTM Model for Short-Term Individual Household Load Forecasting", 2020.

[18] Zhuofu Deng, Binbin Wang, Yanlu Xu, Tengteng Xu, Chenxu Liu, Zhiliang Zhu, "Multi-Scale Convolutional Neural Network with Time-Cognition for Multi-Step Short-Term Load Forecasting",

[19] Gupta V, Sharma M, Pachauri RK, Babu KD, "A low-cost real-time IOT enabled data acquisition system for monitoring of PV system", Energy Sources Part A 2021; 43(20):2529–43.

[20] Islam S, Abdalla AH, Hasan MK, "Novel multihoming-based flow mobility scheme for proxy NEMO environment: a numerical approach to analyse handoff performance", Sci Asia 2017;43:27–34.

[21] Islam S, Hashim AHA, Habaebi MH, Hasan MK, "Design and implementation of a multihoming-based scheme to support mobility management in NEMO", Wirel Pers Commun 2017;95(2):457–73.

[22] Mukherjee A, Ghosh S, Behere A, Ghosh SK, Buyya R, "Internet of health things (IoHT) for personalized health care using integrated edge-fog-cloud network", J Ambient Intell Humaniz Comput 2021;12:943–59.

[23] Kuscu M, Ramezani H, Dinc E, Akhavan S, Akan OB, "Fabrication and microfluidic analysis of graphene-based molecular communication receiver for Internet Of Nano Things (IoNT)", Sci Rep 2021;11(1):1–20.

[24] Bhattacharjya K, De D, "IoUT: modelling and simulation of edge-drone-based software-defined smart internet of underwater things", Simul Modell Pract Theory 2021;109:102304.

[25] Priyadarshini I, Bhola B, Kumar R, So-In C, "A novel cloud architecture for internet of space things (IoST) ", IEEE Access; 2022.

[26] Hasan MK, Ahmed MM, Musa SS, Islam S, Abdullah SNHS, Hossain E, Vo N, "An improved dynamic thermal current rating model for PMU-based wide area measurement framework for reliability analysis utilizing sensor cloud system", IEEE Access 2021;9:14446–58.

[27] Hasan MK, Islam S, Sulaiman R, Khan S, Hashim AHA, Habib S, Hassan MA, "Lightweight encryption technique to enhance medical image security on internet of medical things applications", IEEE Access 2021;9:47731–42.

[28] Ghazal TM, Hasan MK, Alshurideh MT, Alzoubi HM, Ahmad M, Akbar SS, Akour IA, "IoT for smart cities: machine learning approaches in smart healthcare—a review", Future Internet 2021;13(8):218.

[29] Yar H, Imran AS, Khan ZA, Sajjad M, Kastrati Z, "Towards smart home automation using IoT-enabled edge-computing paradigm", Sensors 2021;21(14):4932.

[30] Xu R, Zeng Q, Zhu L, Chi H, Du X, Guizani M, "Privacy leakage in smart homes and its mitigation: IFTTT as a case study", IEEE Access 2019;7:63457–71.

[31] Kumar P, Chouhan L, "A secure authentication scheme for IoT application in smart home", Peer Peer Netw Appl 2021;14(1):420–38.

[32] Ammi M, Alarabi S, Benkhelifa E, "Customized blockchain-based architecture for secure smart home for lightweight IoT", Inf Process Manag 2021;58(3):102482.

[33] Hasan MK, Yousoff SH, Ahmed MM, Hashim AHA, Ismail AF, Islam S, "Phase offset analysis of asymmetric communications infrastructure in smart grid", Elektron Elektrotechnika 2019;25(2):67–71.

[34] Williams R, Yampolskiy R, "Understanding and avoiding ai failures: a practical guide", Philosophies 2021;6(3):53.

[35] Qureshi KN, Jeon G, Piccialli F, "Anomaly detection and trust authority in artificial intelligence and cloud computing", Comput Netw 2021;184:107647.

[36] Alghofaili Y, Rassam MA, "A trust management model for IoT devices and services based on the multi-criteria decision-making approach and deep long short-term memory technique", Sensors 2022;22(2):634.

[37] Guo J, Liu A, Ota K, Dong M, Deng X, Xiong N, "ITCN: an intelligent trust collaboration network system in IoT", IEEE Trans Netw Sci Eng 2021;9(1):203–18.

[38] Nichols K, "Trust schemas and icn: key to secure home iot", In: Proceedings of the 8th ACM Conference on Information-Centric Networking; 2021. p. 95–106.

[39] Abdalzaher MS, Elwekeil M, Wang T, Zhang S, "A deep autoencoder trust model for mitigating jamming attack in IoT assisted by cognitive radio", IEEE Syst J 2021;1-10. https://doi.org/10.1109/JSYST.2021.3099072.

[40] Ghaleb M, Azzedin F, "Towards scalable and efficient architecture for modeling trust in IoT environments", Sensors 2021;21(9):2986.

[41] Ma W, Wang X, Hu M, Zhou Q, "Machine learning empowered trust evaluation method for IoT devices", IEEE Access 2021;9:65066–77.

[42] Narang N, Kar S, "A hybrid trust management framework for a multi-service social IoT network", Comput Commun 2021;171:61–79.

[43] Malchi SK, Kallam S, Al-Turjman F, Patan R, "A trust-based fuzzy neural network for smart data fusion in internet of things", Comput Electr Eng 2021;89:106901.

[44] Kumar R, Sharma R, "Leveraging blockchain for ensuring trust in iot: a survey", Journal of King Saud University-Computer and Information Sciences; 2021.

[45] Wu X, Liang J, "A blockchain-based trust management method for Internet of Things", Pervasive Mob Comput 2021;72:101330.

[46] Viriyasitavat W, Xu LD, Sapsomboon A, Dhiman G, Hoonsopon D, "Building trust of Blockchain-based Internet-of-Thing services using public key infrastructure", Enterp Inform Syst2022;16.

[47] Kavisankar L, Sivakumar V, Balachander T, Krishnan M, Deeban Chakravarthy V, Selvin Paul Peter J, Balasubramani S, "An efficient trust-based supply chain management framework utilizing the internet of things and Blockchain technology", Implementing and leveraging blockchain programming. Singapore: Springer; 2022. p. 149–60.

[48] Puri V, Priyadarshini I, Kumar R, Van Le C, "Smart contract based policies for the Internet of Things", Cluster Comput 2021;24(3):1675–94.

[49] Dansana D, Kumar R, Das Adhikari J, Mohapatra M, Sharma R, Priyadarshini I, Le DN, "Global forecasting confirmed and fatal cases of COVID-19 outbreak using autoregressive integrated moving average model", Front Public Health 2020:592.

[50] Dubey AK, Kumar A, García-Díaz V, Sharma AK, Kanhaiya K, "Study and analysis of SARIMA and LSTM in forecasting time series data", Sustain Energy Technol Assess 2021;47:101474.

[51] Priyadarshini I, Cotton C, "A novel LSTM–CNN–grid search-based deep neural network for sentiment analysis", J Supercomput 2021;77(12):13911–32.

[52] Khayyat M, Laabidi K, Almalki N, Al-Zahrani M, "Time series Facebook Prophet model and python for COVID-19 outbreak prediction", Comput Mater Contin 2021:3781–93.

[53] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Liu TY, "Lightgbm: a highly efficient gradient boosting decision tree", Adv Neural Inf Process Syst 2017:30.

[54] Haslbeck JM, Bringmann LF, Waldorp LJ, "A tutorial on estimating time-varying vector autoregressive models", Multivar Behav Res 2021;56(1):120–49.

[55] Qureshi KN, Alhudhaif A, Hussain A, Iqbal S, Jeon G, "Trust aware energy management system for smart homes appliances", Comput Electr Eng 2022;97:107641.

[56] Taiwo O, Ezugwu AE, Oyelade ON, Almutairi MS, "Enhanced intelligent smart home control and security system based on deep learning model", Wirel Commun Mobile Comput 2022:2022.

[57] Yang Q, Wang H, "Privacy-preserving transactive energy management for IoT-aided smart homes via blockchain", IEEE Int Things J 2021;8(14):11463–75.

[58] Ribeiro FM, Kamienski CA, "TW-fogginess: a trustworthy IoT system based on mist and fog computing", In: Proceedings of the 2021 IEEE symposium on computers and communications (ISCC). IEEE; 2021. p. 1–6.

[59] Naman Bhoj, Robin Singh Bhadoria, "Time-series based prediction for energy consumption of smart home data using hybrid convolution-recurrent neural network", Telematics and Informatics, 75, 2022, 101907.

ABBREVIATIONS

- **STLF** - Short-Term Load Forecasting
- **DDS** - Data-Driven Strategy
- **ML** - Machine Learning
- **AI**- Artificial Intelligence
- **DL** - Deep Learning
- **NN** - Neural Networks
- **LSTM** - Long Short-Term Memory (a type of recurrent neural network)
- **RNN** - Recurrent Neural Network
- **SVM** - Support Vector Machine
- **ARIMA** - AutoRegressive Integrated Moving Average
- **XGBoost** - Extreme Gradient Boosting
- **CNN** - Computational Neural Network
- **TSF** - Time Series Forecasting
- **MAE** - Mean Absolute Error
- **MSE** - Mean Squared Error
- **RMSE** - Root Mean Squared Error
- **MAPE** - Mean Absolute Percentage Error
- **SARIMA**: Seasonal Autoregressive Integrated Moving Average
- **LightGBM**: Light Gradient Boosting Machine
- **BiLSTM**: Bidirectional Long Short-Term Memory

- **CNN-BiLSTM**: Convolutional Neural Network - Bidirectional Long Short-Term Memory
- **VAR**: Vector Autoregression
- **CNN-GRU**: Convolutional Neural Network - Gated Recurrent Unit
- **GRU**: Gated Recurrent Unit
- **SVR**: Support Vector Regression

ANNEXURE

Code:

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import time

from sklearn.ensemble import IsolationForest
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from pandas.plotting import autocorrelation_plot
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.callbacks import EarlyStopping
from keras.layers import Conv1D, MaxPooling1D, Bidirectional, LSTM,
Dense, Dropout
from keras.regularizers import l2
from keras.optimizers import Adam
from google.colab import drive
```

Data Pre-processing

```
drive.mount ("/content/gdrive")
home=pd.read_csv('/content/gdrive/MyDrive/Colab Notebooks/FYP-
Data.csv')
home.head()
home
#removing the last row as it is invalid
home=home[0:-1]
home
#Removing the [KW] from column names
home.columns = [col.replace(' [kW]', '') for col in home.columns]
home.columns
#Summing the columns 'Furnace 1' and 'Furnace 2', since we only need
the total energy usage by both the furnaces
```

```
home['Furnace'] = home[['Furnace 1', 'Furnace 2']].sum(axis=1)

#Finding the average usage of all kitchens('Kitchen 12','Kitchen
14','Kitchen 38')
home['Kitchen'] = home[['Kitchen 12','Kitchen 14','Kitchen
38']].mean(axis=1)
#Dropping old columns on which the aggregation have been done
home = home.drop(['Furnace 1','Furnace 2','Kitchen 12','Kitchen
14','Kitchen 38'], axis=1)
#Finding the start time
print('start', time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(int(home['time'].iloc[0]))))
#Converting the unix timestamp values in the column 'time' to a
readable date format
#Setting freq= 'min' since readings are being taken in a time span
of one min
time_index = pd.date_range('2016-01-01 05:00' , periods = len(home)
, freq='min')
time_index = pd.DatetimeIndex(time_index)
home=home.set_index(time_index)
home=home.drop(['time'] , axis=1)
home.iloc[np.r_[0:5 , -5:0]].iloc[:,0]
#Display dataframe info
home.info()
home.describe().transpose()
#Drop columns 'icon' and 'summary' as their data type is 'object'
home = home.drop(['icon', 'summary'], axis=1)
#Examining the unique values of the attribute 'cloudCover' to see if
there is any invalid data that needs to be handled
home['cloudCover'].unique()
#We see that for some rows we have an invalid value for cloudCover.
home[home['cloudCover'] == 'cloudCover'].shape
#We replace these missing values with the next valid observation we
have
home['cloudCover'].replace(['cloudCover'] , method='bfill' , inplace
= True)
home['cloudCover'] = home['cloudCover'].astype('float')
home['cloudCover']
#It seems use and House overall show the same data
home[['use', 'House overall']].head()
# Let's visualize these use and House overall to confirm they show
the same data so that we could remove one of the columns
fig, axes = plt.subplots (nrows=2, ncols=1)
home['use'].resample('D').mean().plot(ax=axes[0])
home['House overall'].resample('D').mean().plot(ax=axes[1])
```

```
#Since the above plot confirms both the columns indicate the same
data lets go ahead and remove 'House overall'
home=home.drop(columns=['House overall'])
home.shape
#Similarly, the attributes 'gen' and 'Solar' show the same data.
Let's visualize these two columns to confirm the same
fig, axes = plt.subplots(nrows=2, ncols=1)
home['gen'].plot(ax=axes[0], figsize=(20,10))
home['Solar'].plot(ax=axes[1], figsize=(20,10))
#Dropping the column 'Solar' since it shows the same data as 'gen'
home=home.drop(columns=['Solar'])
home.shape
# Renaming columns
home.rename(columns={'use': 'Use', 'gen': 'Generation', 'Home
office': 'Home Office',
                    'Wine cellar': 'Wine Cellar', 'Garage door':
'Garage Door',
                    'Living room': 'Living Room', 'temperature':
'Temperature',
                    'humidity':'Humidity','visibility': 'Visibility',
                    'apparentTemperature': 'Apparent Temperature',
                    'pressure': 'Pressure', 'windspeed': 'Wind
Speed',
                    'cloudCover': 'Cloud Cover', 'windBearing': 'Wind
Bearing',
                    'precipIntensity': 'Precipitation Intensity',
                    'dewPoint': 'Dew Point','windSpeed':'Wind Speed',
                    'precipProbability': 'Precipitation Probability'}
, inplace=True)
```

Data Transformation

```
#Creating two separate dataframe objects with energy and weather
data respectively.
energy_data = home.filter(items = ['Generation', 'Use',
'Dishwasher',
                                   'Furnace', 'Home Office', 'Fridge',
                                   'Wine Cellar','Garage
Door','Kitchen',
                                   'Barn','Well','Microwave','Living
Room'])
weather_data = home.filter(items=['Temperature','Humidity',
'Visibility', 'Pressure','Apparent Temperature','Wind Speed','Wind
Bearing', 'Dew Point'])
#Generating the energy data per day
```

```
energy_per_day = energy_data.resample('D').sum()
energy_per_day.head()
#Generating the energy data per week
energy_per_week = energy_data.resample('W').sum()
energy_per_week.head()
#Generating the energy data per month
energy_per_month = energy_data.resample('M').sum()
energy_per_month.head()
#Generating the data per day
data_per_day = home.resample('D').sum()
data_per_day.head()
#Generating the data per week
data_per_week = home.resample('W').sum()
data_per_week.head()
#Generating the data per month
data_per_month = home.resample('M').sum()
data_per_month.head()
```

Data Visualization

```
#Set the width and height of the figure
plt.figure(figsize=(20,10))

#Add title
plt.title("Overall energy consumption in kW per day")
#Plotting the energy consumption per day
sns.lineplot(data = energy_per_day.filter(items=['Use']),
dashes=False)
# Set the width and height of the figure
plt.figure(figsize=(20,10))

# Add title
plt.title("Overall energy consumption in kW per week")
#Plotting the energy consumption per week
sns.lineplot(data = energy_per_week.filter(items=['Use']),
dashes=False)
# Set the width and height of the figure
plt.figure(figsize=(20,10))

# Add title
plt.title("Overall energy consumption in kW per month")
#Plotting the energy consumption per month
sns.lineplot(data = energy_per_month.filter(items=['Use']),
dashes=False)
#Set the width and height of the figure
plt.figure(figsize=(20,10))
```

```
#Add title
plt.title("Overall energy generation in kW per day")
#Plotting the energy consumption per day
sns.lineplot(data = energy_per_day.filter(items=['Generation']),
dashes=False)
#Set the width and height of the figure
plt.figure(figsize=(20,10))

#Add title
plt.title("Overall energy generation in kW per week")
#Plotting the energy consumption per day
sns.lineplot(data = energy_per_week.filter(items=['Generation']),
dashes=False)
#Set the width and height of the figure
plt.figure(figsize=(20,10))

#Add title
plt.title("Overall energy generation in kW per month")
#Plotting the energy consumption per day
sns.lineplot(data = energy_per_month.filter(items=['Generation']),
dashes=False)
plt.figure(figsize=(15,6))
plt.title("Energy consumption in kW per day by individual
room/appliance")
sns.lineplot(data = energy_per_day.filter(items=['Dishwasher',
                                    'Furnace', 'Home Office',
'Fridge',
                                    'Wine Cellar', 'Garage Door',
'Kitchen', 'Barn', 'Well',
                                    'Microwave', 'Living Room']),
dashes=False)
plt.figure(figsize=(20,10))
plt.title("Energy consumption in kW per month by individual
room/appliance")
sns.lineplot(data = energy_per_month.filter(items=['Dishwasher',
                                    'Furnace', 'Home Office',
'Fridge',
                                    'Wine Cellar', 'Garage Door',
'Kitchen', 'Barn', 'Well',
                                    'Microwave', 'Living Room']),
dashes=True)
#Below we are splitting the energy consumption per month data into
two sets.
```

```
#1) Energy consumed by each room in the house
energy_per_room = energy_per_month.filter(items=['Home Office',
                                        'Wine Cellar',
'Kitchen','Barn',
                                         'Living Room'])

#2) Energy consumed by each device in the house
energy_per_device = energy_per_month.filter(items=[ 'Dishwasher',
                                        'Furnace','Fridge',
                                        'Garage Door', 'Well',
                                        'Microwave'])

rooms_energy_consumption = energy_per_room.sum()
devices_energy_consumption = energy_per_device.sum()
print("Energy consumed in kW per month by each room in the house")
print(rooms_energy_consumption)

print('----------------------------------------------------------------
-')

print("Energy consumed in kW per month by each device in the house")
print(devices_energy_consumption)
plot = rooms_energy_consumption .plot(kind = "pie", figsize = (7,7))
plot.set_title("Energy consumption in kW by each room in the house")
plot = devices_energy_consumption .plot(kind = "pie", figsize =
(7,7))
plot.set_title("Energy consumption in kW by each device in the
house")
plt.figure(figsize=(20,10))
plt.title("Temperature and Humidity Over Time")
sns.lineplot(data = home.filter(items=['Humidity', 'Temperature']),
dashes=True)
categories = ['Dishwasher', 'Home Office', 'Fridge', 'Wine Cellar',
'Garage Door', 'Barn', 'Well', 'Microwave', 'Living Room',
'Furnace', 'Kitchen']

usage = [home[category].sum() for category in categories]

plt.bar(categories, usage)
plt.xlabel('Appliance/Room')
plt.ylabel('Energy Usage')
plt.title('Energy Usage by Appliance/Room')
plt.xticks(rotation=45)
plt.show()
```

```
Correlation between features
#Correlation of energy usage by home appliances

energy_corr = energy_data.loc[:,'Dishwasher':].corr()
fig,ax = plt.subplots(figsize=(20, 10))
sns.heatmap(energy_corr, linewidths=0.5, annot=True, cmap="YlGnBu");
ax.set_title("Correlation of Energy Usage by appliances", size=15)
plt.show()
#Correlation between weather data

weather_corr = weather_data.corr()
fig,ax = plt.subplots(figsize=(20, 10))
sns.heatmap(weather_corr, linewidths=0.5, annot=True,
cmap="YlGnBu");
ax.set_title("Correlation between Weather Data", size=15)
plt.show()
# Merge energy and weather datasets
allData_df = pd.merge(energy_data.loc[:,'Dishwasher':],
weather_data, left_index=True, right_index=True)
allData_df.head()
#Correlation between all data

all_corr = allData_df.corr()
fig,ax = plt.subplots(figsize=(20, 10))
sns.heatmap(all_corr, linewidths=0.5, annot=True, cmap="YlGnBu");
ax.set_title("Correlation between all Data", size=15)
plt.show()

Time Series Analysis
# define function to convert hour to time of day

def convertHourToPartOfDay(x):
  if x in [22,23,0,1,2,3]:
    part = "Night"
  elif x in range (4,12):
    part = "Morning"
  elif x in range (12,17):
    part = "Afternoon"
  elif x in range (17,22):
    part = "Evening"
  else:
    part = "x"
  return part
```

```
#Define function to group by time of day

def groupByPartOfDay(column):
    partOfDayDf =
energy_per_part_of_day.groupby('part').agg({column:['mean']})
    partOfDayDf.columns = [f"{i[0]}_{i[1]}" for i in
partOfDayDf.columns]
    partOfDayDf['part_num'] =
[['Morning','Afternoon','Evening','Night'].index(i) for i in
partOfDayDf.index]
    partOfDayDf.sort_values('part_num', inplace=True)
    partOfDayDf.drop('part_num', axis=1, inplace=True)
    return partOfDayDf
# Create column to indicate time of day in dataset

energy_per_part_of_day = energy_data.loc[:]
energy_per_part_of_day['hour'] =
energy_per_part_of_day.index.map(lambda x: x.hour)
energy_per_part_of_day['part'] =
energy_per_part_of_day['hour'].apply(convertHourToPartOfDay)
# Plot generated vs used energy per part of day

plt.figure(figsize=(20,10))
plt.plot(groupByPartOfDay('Generation'), 'g', label='Generated
energy')
plt.plot(groupByPartOfDay('Use'), 'r', label='Used energy')
plt.title('Generated vs Used energy per part of day')
plt.ylabel('Energy (kW)')
plt.xlabel('Part of day')
plt.legend()
plt.grid(True)
plt.show()
# Plot generated vs used energy per day

plt.figure(figsize=(20,10))
plt.plot(energy_per_day.Generation, 'g', label='Generated energy')
plt.plot(energy_per_day.Use, 'r', label='Used energy')
plt.title('Generated vs Used energy per day')
plt.ylabel('Energy (kW)')
plt.xlabel('Day')
plt.legend()
plt.grid(True)
plt.show()
```

```
Anomaly Detection
data = energy_per_day.filter(items=['Generation'])
df = data

isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(data.values.reshape(-1, 1))
xx = np.linspace(data.min(), data.max(), len(data)).reshape(-1,1)

df['scores']=isolation_forest.decision_function(df[['Generation']])
df['anomaly']=isolation_forest.predict(df[['Generation']])
df.head(5)
anomaly=df.loc[df['anomaly']==-1]
anomaly_index=list(anomaly.index)
print(anomaly)
# Distribution of generated energy per day
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
plt.fill_between(xx.T[0], np.min(anomaly_score),
np.max(anomaly_score),
                 where=outlier==-1, color='r',
                 alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Generated Energy per Day')
plt.show();
data = energy_per_week.filter(items=['Generation'])
df = data

isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(data.values.reshape(-1, 1))
xx = np.linspace(data.min(), data.max(), len(data)).reshape(-1,1)

df['scores']=isolation_forest.decision_function(df[['Generation']])
df['anomaly']=isolation_forest.predict(df[['Generation']])
df.head(5)
anomaly=df.loc[df['anomaly']==-1]
anomaly_index=list(anomaly.index)
print(anomaly)
# Distribution of generated energy per week
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
```

```
plt.fill_between(xx.T[0], np.min(anomaly_score),
np.max(anomaly_score),
                 where=outlier==-1, color='r',
                 alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Generated Energy per Week')
plt.show();
data = energy_per_month.filter(items=['Generation'])
df = data

isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(data.values.reshape(-1, 1))
xx = np.linspace(data.min(), data.max(), len(data)).reshape(-1,1)

df['scores']=isolation_forest.decision_function(df[['Generation']])
df['anomaly']=isolation_forest.predict(df[['Generation']])
df.head(5)
anomaly=df.loc[df['anomaly']==-1]
anomaly_index=list(anomaly.index)
print(anomaly)
# Distribution of generated energy per month
anomaly_score = isolation_forest.decision_function(xx)
outlier = isolation_forest.predict(xx)
plt.figure(figsize=(10,4))
plt.plot(xx, anomaly_score, label='anomaly score')
plt.fill_between(xx.T[0], np.min(anomaly_score),
np.max(anomaly_score),
                 where=outlier==-1, color='r',
                 alpha=.4, label='outlier region')
plt.legend()
plt.ylabel('anomaly score')
plt.xlabel('Generated Energy per Month')
plt.show();
data = energy_per_day.filter(items=['Use'])
df = data

isolation_forest = IsolationForest(n_estimators=100)
isolation_forest.fit(data.values.reshape(-1, 1))
xx = np.linspace(data.min(), data.max(), len(data)).reshape(-1,1)

df['scores']=isolation_forest.decision_function(df[['Use']])
df['anomaly']=isolation_forest.predict(df[['Use']])
df.head(5)
```

```
from sklearn.metrics import r2_score, median_absolute_error,
mean_absolute_error
from sklearn.metrics import median_absolute_error,
mean_squared_error, mean_squared_log_error

def plotMovingAverage(series, window, plot_intervals=False,
scale=1.96, plot_anomalies=False):
  rolling_mean = series.rolling(window=window).mean()

  plt.figure(figsize=(25,5))
  plt.title("Moving average (window size = {})".format(window))
  plt.plot(rolling_mean, "g", label="Rolling mean trend")

  # Plot confidence intervals for smoothed values
  if plot_intervals:
      mae = mean_absolute_error(series[window:],
rolling_mean[window:])
      deviation = np.std(series[window:] - rolling_mean[window:])
      lower_bond = rolling_mean - (mae + scale * deviation)
      upper_bond = rolling_mean + (mae + scale * deviation)
      plt.plot(upper_bond, "r--", label="Upper Bond / Lower Bond")
      plt.plot(lower_bond, "r--")

      # Having the intervals, find abnormal values
      if plot_anomalies:
          anomalies = pd.DataFrame(index=series.index,
columns=series.columns)
          anomalies[series<lower_bond] = series[series<lower_bond]
          anomalies[series>upper_bond] = series[series>upper_bond]
          plt.plot(anomalies, "ro", markersize=10)

  plt.plot(series[window:], label="Actual values")
  plt.legend(loc="upper left")
  plt.grid(True)

n_samples = 24*30 # 1 month



data = energy_per_day.filter(items=['Generation'])
plotMovingAverage(data[:n_samples], window=6) # A window of 6 hours
data = energy_per_day.filter(items=['Use'])
plotMovingAverage(data[:n_samples], window=6) # A window of 6 hours
plotMovingAverage(data[:n_samples], window=24, plot_intervals=True,
plot_anomalies=True)
```

```
home.info()

Time Series Forecasting using LSTM model
home.columns

# Reduce size of dataframe with only the columns we are interested
in
data_daily = home[['Use', 'Generation', 'Dishwasher', 'Home Office',
'Fridge',
       'Wine Cellar', 'Garage Door', 'Barn', 'Well', 'Microwave',
       'Living Room', 'Temperature', 'Humidity', 'Visibility',
       'Apparent Temperature', 'Pressure', 'Wind Speed', 'Cloud
Cover',
       'Wind Bearing', 'Precipitation Intensity', 'Dew Point',
       'Precipitation Probability', 'Furnace', 'Kitchen']]

# Rescale
data_daily = data_daily.resample('D').mean()

# Normalize the features
scaler = MinMaxScaler(feature_range=(0, 1))
data_daily[data_daily.columns[1:]] =
scaler.fit_transform(data_daily[data_daily.columns[1:]])
scaler_target = MinMaxScaler(feature_range=(0, 1))
data_daily[['Use']] =
scaler_target.fit_transform(data_daily[['Use']])

size = int(len(data_daily) * 0.7)
data_daily_train = data_daily[:size]
data_daily_test = data_daily[size:]
X_train, X_test = [], []
Y_train, Y_test = [], []
n_past = 1
n_future = 1

for i in range(n_past, len(data_daily_train) - n_future + 1):
    X_train.append(data_daily_train.iloc[i - n_past:i,
0:data_daily.shape[1]])
    Y_train.append(data_daily_train.iloc[i + n_future - 1:i +
n_future, 0])
for i in range(n_past, len(data_daily_test) - n_future + 1):
    X_test.append(data_daily_test.iloc[i - n_past:i,
0:data_daily_test.shape[1]])
    Y_test.append(data_daily_test.iloc[i + n_future - 1:i +
n_future, 0])
```

```
X_train, Y_train = np.array(X_train), np.array(Y_train)
X_test, Y_test = np.array(X_test), np.array(Y_test)

print('X_train shape', X_train.shape)
print('X_test shape', X_test.shape)
print('Y_train shape', Y_train.shape)
print('Y_test shape', Y_test.shape)

model = Sequential()
model.add(LSTM(25, activation='relu', return_sequences = False,
input_shape=(X_train.shape[1], X_train.shape[2])))
#model.add(LSTM(50, activation='relu', return_sequences = True))
#model.add(LSTM(15, activation='relu', return_sequences = False))
#model.add(Dropout(0.2))
model.add(Dense(Y_train.shape[1]))
model.compile(optimizer='adam', loss='mse')
model.summary()

model_fit = model.fit(X_train, Y_train, epochs=60, verbose=0)

Train_pred = model.predict(X_train, verbose=0)
Y_pred = model.predict(X_test, verbose=0)

plt.plot(model_fit.history['loss'], label='Train loss')
plt.legend()
print('Train MSE minimum:', min(model_fit.history['loss']))

# Invert scaling
data_daily[['Use']] =
scaler_target.inverse_transform(data_daily[['Use']])
Y_pred_reshaped = Y_pred.reshape(-1, 1)
Train_pred_reshaped = Train_pred.reshape(-1, 1)
Y_pred = scaler_target.inverse_transform(Y_pred_reshaped)
Train_pred = scaler_target.inverse_transform(Train_pred_reshaped)

plt.figure(figsize=(15, 4))
plt.plot(data_daily[['Use']][size:-1].values)
plt.plot(Y_pred)
np.sqrt(mean_squared_error(Y_pred[:, 0].tolist(),
data_daily[['Use']][size:-1].values))

Y_pred_series = pd.Series(Y_pred.flatten().tolist(),
index=data_daily['Use'][size:-n_past].index)
```

```
Train_pred_series = pd.Series(Train_pred.flatten().tolist(),
index=data_daily['Use'][n_past:size].index)
plt.figure(figsize=(15, 4))
plt.plot(data_daily['Use'][:-n_past], c='blue', label='data')
plt.plot(Y_pred_series, c='red', label='model test')
plt.plot(Train_pred_series, c='green', label='model train')
plt.legend()
plt.grid(), plt.margins(x=0)
Y_test = data_daily['Use'][size:-n_past]

# Calculate errors
print('MSE: %.5f' % (mean_squared_error(Y_pred, Y_test)))
print('RMSE: %.5f' % np.sqrt(mean_squared_error(Y_pred, Y_test)))
MAE = mean_absolute_error(Y_test, Y_pred)
MAPE = np.mean(np.abs(Y_pred[:, 0] - Y_test.values) /
np.abs(Y_test.values))
print('MAE: %.3f' % MAE)
print('MAPE: %.3f' % MAPE)
print('R^2 score: %.3f' % r2_score(Y_test, Y_pred))

Time Series Forecasting using Bi-LSTM model
# Load data
data = home[['Use']]  # Assuming the relevant column is 'Use'

# Resample data to daily frequency (assuming the data has a Date
column)
# Sample data per day and split 80-20
data = data.resample('D').mean().dropna()
train_size = int(len(data) * 0.8)
train, test = data[:train_size], data[train_size:]

# Scale the data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_train = scaler.fit_transform(train)
scaled_test = scaler.transform(test)

# Create datasets for the model
def create_dataset(X, time_step=1):
    Xs, ys = [], []
    for i in range(len(X) - time_step):
        Xs.append(X[i:(i + time_step), 0])
        ys.append(X[i + time_step, 0])
    return np.array(Xs), np.array(ys)

time_step = 10
```

```
X_train, y_train = create_dataset(scaled_train, time_step)
X_test, y_test = create_dataset(scaled_test, time_step)

# Reshape input to be [samples, time steps, features] which is
required for LSTM
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
# Print the shapes to check dimensions
print("Train data shape:", train.shape)
print("Test data shape:", test.shape)
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)

# Build the Bi-LSTM model
model = Sequential()
model.add(Bidirectional(LSTM(64, return_sequences=True),
input_shape=(time_step, 1)))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(64)))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Implement early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the model with validation split
history = model.fit(X_train, y_train, validation_split=0.2,
epochs=100, batch_size=16, verbose=1, callbacks=[early_stopping])

# Predict and inverse transform the values
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict.reshape(-1, 1))

# Inverse transform y_test for comparison
y_test_inverse = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plot the results
plt.figure(figsize=(15, 6))
```

```
plt.plot(y_test_inverse, label='True Value')
plt.plot(test_predict, label='Predicted Value')
plt.title('Bi-LSTM Model - True vs Predicted Values')
plt.xlabel('Time Step')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# Calculate evaluation metrics
mse = mean_squared_error(y_test_inverse, test_predict)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test_inverse, test_predict)
mape = np.mean(np.abs(test_predict - y_test_inverse) /
np.abs(y_test_inverse)) * 100
r2 = r2_score(y_test_inverse, test_predict)

# Print results
print("Bi-LSTM - MSE: {:.5f}, RMSE: {:.5f}, MAE: {:.5f}, MAPE:
{:.5f}, R2: {:.5f}".format(mse, rmse, mae, mape, r2))

# Plot evaluation metrics
metrics = {'MSE': mse, 'RMSE': rmse, 'MAE': mae, 'MAPE': mape, 'R2':
r2}
plt.figure(figsize=(10, 5))
plt.bar(metrics.keys(), metrics.values(), color=['blue', 'orange',
'green', 'red', 'purple'])
plt.title('Evaluation Metrics for Bi-LSTM Model')
plt.ylabel('Value')
plt.grid(True)
plt.show()

Time Series Forecasting using CNN-Bi-LSTM model
# Load data
data = home[['Use']]  # Assuming the relevant column is 'Use'

# Resample data to daily frequency (assuming the data has a Date
column)
daily_data = data.resample('D').mean()

# Preprocess data
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(daily_data)

# Split data into training and testing sets (80-20 split)
```

```
train_size = int(len(scaled_data) * 0.80)
train, test = scaled_data[:train_size], scaled_data[train_size:]

# Create datasets for CNN-BiLSTM
def create_dataset(X, time_step=1):
    Xs, ys = [], []
    for i in range(len(X) - time_step - 1):
        Xs.append(X[i:(i + time_step), 0])
        ys.append(X[i + time_step, 0])
    return np.array(Xs), np.array(ys)

time_step = 10
X_train, y_train = create_dataset(train, time_step)
X_test, y_test = create_dataset(test, time_step)

# Reshape input to be [samples, time steps, features] which is
required for LSTM
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
# Print the shapes to check dimensions
print("Scaled data shape:", scaled_data.shape)
print("Train data shape:", train.shape)
print("Test data shape:", test.shape)
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
# Build the CNN-BiLSTM model
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=10, strides=1,
activation='tanh', input_shape=(time_step, 1)))
model.add(MaxPooling1D(pool_size=1))  # Adjusted pool_size
model.add(Dropout(0.2))
model.add(Conv1D(filters=128, kernel_size=1, strides=1,
activation='tanh'))
model.add(MaxPooling1D(pool_size=1))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(300, return_sequences=True,
activation='tanh')))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(500, activation='tanh')))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

```
# Implement early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)

# Train the model with validation split
history = model.fit(X_train, y_train, validation_split=0.2,
epochs=600, batch_size=16, verbose=1, callbacks=[early_stopping])

# Predict and inverse transform the values
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict.reshape(-1, 1))

# Inverse transform y_test for comparison
y_test_inverse = scaler.inverse_transform(y_test.reshape(-1, 1))

# Plot the results
plt.figure(figsize=(15,4))
plt.plot(y_test_inverse, label='True Value')
plt.plot(test_predict, label='Predicted Value')
plt.title('CNN-BiLSTM Model - True vs Predicted Values')
plt.xlabel('Time Step')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()
# Calculate evaluation metrics
mse = mean_squared_error(y_test_inverse, test_predict)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test_inverse, test_predict)
mape = np.mean(np.abs(test_predict - y_test_inverse) /
np.abs(y_test_inverse)) * 100
r2 = r2_score(y_test_inverse, test_predict)

# Print results
print("CNN-BiLSTM - MSE: {:.5f}, RMSE: {:.5f}, MAE: {:.5f}, MAPE:
{:.5f}, R2: {:.5f}".format(mse, rmse, mae, mape, r2))

# Plot evaluation metrics
metrics = {'MSE': mse, 'RMSE': rmse, 'MAE': mae, 'MAPE': mape, 'R2':
r2}
plt.figure(figsize=(10, 5))
```

```
bars = plt.bar(metrics.keys(), metrics.values(), color=['blue',
'orange', 'green', 'red', 'purple'])

# Adjust the y-axis scale
plt.ylim(0, max(metrics.values()) * 1.2)

# Add value labels to each bar
for bar in bars:
    plt.text(bar.get_x() + bar.get_width() / 2 - 0.1,
bar.get_height() * 1.01, '{:.3f}'.format(bar.get_height()),
fontsize=9)

plt.title('Evaluation Metrics for CNN-BiLSTM Model')
plt.ylabel('Value')
plt.grid(True)
plt.legend(['MSE', 'RMSE', 'MAE', 'MAPE', 'R2'], loc='upper right')
plt.show()
def evaluate_model(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
    return mse, rmse, mae, r2
mse_bilstm, rmse_bilstm, mae_bilstm, r2_bilstm =
evaluate_model(y_test, test_predict)
print("CNN-BiLSTM - MSE: {:.5f}, RMSE: {:.5f}, MAE: {:.5f}, R2:
{:.5f}".format(mse_bilstm, rmse_bilstm, mae_bilstm, r2_bilstm))



import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from keras.models import Sequential
from keras.layers import LSTM, Dense, Bidirectional, Dropout,
Conv1D, MaxPooling1D

# Assuming 'data' is your dataframe with a datetime index and a
'Use' column
data = home[['Use']]
data_daily = data.resample('D').mean().dropna()

# Normalize the features
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data_daily)
data_daily[['Use']] = scaler.fit_transform(data_daily[['Use']])

# Split the data
train_size = int(len(data_daily) * 0.8)
train, test = scaled_data[:train_size], scaled_data[train_size:]
train_index, test_index = data_daily.index[:train_size],
data_daily.index[train_size:]

# Create datasets for BiLSTM
def create_dataset(X, time_step=1):
    Xs, ys = [], []
    for i in range(len(X) - time_step - 1):
        Xs.append(X[i:(i + time_step), 0])
        ys.append(X[i + time_step, 0])
    return np.array(Xs), np.array(ys)

time_step = 10
X_train, y_train = create_dataset(train, time_step)
X_test, y_test = create_dataset(test, time_step)

# Reshape input to be [samples, time steps, features]
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Inverse scaling for y_test for evaluation
y_test_inverse = scaler.inverse_transform(y_test.reshape(-1, 1))

# Function to evaluate model
def evaluate_model(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_true, y_pred)
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
    r2 = r2_score(y_true, y_pred)
    return mse, rmse, mae, mape, r2

# Function to plot results
def plot_results(y_true, predictions, title, labels):
    plt.figure(figsize=(15, 4))
    plt.plot(y_true, label='True Value')
    for pred, label in zip(predictions, labels):
        plt.plot(pred, label=label)
    plt.title(title)
```

```
    plt.xlabel('Time Step')
    plt.ylabel('Value')
    plt.legend()
    plt.show()

# BiLSTM Model
model_bilstm = Sequential()
model_bilstm.add(Bidirectional(LSTM(64, return_sequences=True),
input_shape=(time_step, 1)))
model_bilstm.add(Dropout(0.2))
model_bilstm.add(Bidirectional(LSTM(64)))
model_bilstm.add(Dropout(0.2))
model_bilstm.add(Dense(1))
model_bilstm.compile(optimizer='adam', loss='mean_squared_error')
model_bilstm.fit(X_train, y_train, validation_split=0.2, epochs=100,
batch_size=16, verbose=1)
train_predict_bilstm = model_bilstm.predict(X_train)
test_predict_bilstm = model_bilstm.predict(X_test)
train_predict_bilstm =
scaler.inverse_transform(train_predict_bilstm)
test_predict_bilstm = scaler.inverse_transform(test_predict_bilstm)

# CNN-BiLSTM Model
model_cnn_bilstm = Sequential()
model_cnn_bilstm.add(Conv1D(filters=64, kernel_size=10, strides=1,
activation='tanh', input_shape=(time_step, 1)))
model_cnn_bilstm.add(MaxPooling1D(pool_size=1))
model_cnn_bilstm.add(Dropout(0.2))
model_cnn_bilstm.add(Conv1D(filters=128, kernel_size=1, strides=1,
activation='tanh'))
model_cnn_bilstm.add(MaxPooling1D(pool_size=1))
model_cnn_bilstm.add(Dropout(0.2))
model_cnn_bilstm.add(Bidirectional(LSTM(300, return_sequences=True,
activation='tanh')))
model_cnn_bilstm.add(Dropout(0.2))
model_cnn_bilstm.add(Bidirectional(LSTM(500, activation='tanh')))
model_cnn_bilstm.add(Dropout(0.2))
model_cnn_bilstm.add(Dense(1))
model_cnn_bilstm.compile(optimizer='adam',
loss='mean_squared_error')
model_cnn_bilstm.fit(X_train, y_train, validation_split=0.2,
epochs=100, batch_size=16, verbose=1)
train_predict_cnn_bilstm = model_cnn_bilstm.predict(X_train)
test_predict_cnn_bilstm = model_cnn_bilstm.predict(X_test)
```

```
train_predict_cnn_bilstm =
scaler.inverse_transform(train_predict_cnn_bilstm)
test_predict_cnn_bilstm =
scaler.inverse_transform(test_predict_cnn_bilstm)

# Plot the results for both models
plot_results(y_test_inverse,
             [test_predict_bilstm, test_predict_cnn_bilstm],
             'Comparison of Model Predictions',
             ['Proposed Model 1', 'Proposed Model 2'])

# Calculate and print evaluation metrics for both models
mse_bilstm, rmse_bilstm, mae_bilstm, mape_bilstm, r2_bilstm =
evaluate_model(y_test_inverse, test_predict_bilstm)
mse_cnn_bilstm, rmse_cnn_bilstm, mae_cnn_bilstm, mape_cnn_bilstm,
r2_cnn_bilstm = evaluate_model(y_test_inverse,
test_predict_cnn_bilstm)

print(f"BiLSTM - MSE: {mse_bilstm:.5f}, RMSE: {rmse_bilstm:.5f},
MAE: {mae_bilstm:.5f}, MAPE: {mape_bilstm:.5f}%, R2:
{r2_bilstm:.5f}")
print(f"CNN-BiLSTM - MSE: {mse_cnn_bilstm:.5f}, RMSE:
{rmse_cnn_bilstm:.5f}, MAE: {mae_cnn_bilstm:.5f}, MAPE:
{mape_cnn_bilstm:.5f}%, R2: {r2_cnn_bilstm:.5f}")

Comparison of Models
import matplotlib.pyplot as plt

# Data
models = ['LSTM', 'CNN-BiLSTM', 'Bi-LSTM']
mae = [0.104, 0.15317, 0.15494]
mse = [0.01917, 0.03955, 0.04047]
rmse = [0.13845, 0.19886, 0.20118]
mape = [0.147, 0.191, 0.185]

# Plotting
fig, ax = plt.subplots(figsize=(10, 6))

bar_width = 0.2
index = range(len(models))

bar1 = ax.bar(index, mae, bar_width, label='MAE')
bar2 = ax.bar([i + bar_width for i in index], mse, bar_width,
label='MSE')
```

```
bar3 = ax.bar([i + 2*bar_width for i in index], rmse, bar_width,
label='RMSE')
bar4 = ax.bar([i + 3*bar_width for i in index], mape, bar_width,
label='MAPE')

# Adding labels
ax.set_xlabel('Models')
ax.set_ylabel('Scores')
ax.set_title('Comparison of Different Models')
ax.set_xticks([i + 1.5 * bar_width for i in index])
ax.set_xticklabels(models)
ax.legend()

# Adding grid
plt.grid(True, which='both', linestyle='--', linewidth=0.5)

# Show plot
plt.tight_layout()
plt.show()
```

ONE PAGE SUMMARY OF PLAGIARISM REPORT SIGNED BY THE SUPERVISOR