ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Time Series Forecasting: A Data Mining Approach

by
João Figueiredo, 1230194
João Araújo 1200584

Course: Masters in Informatics Engineering
Subject: Data Mining

Supervisors: Maria de Fátima Coutinho Rodrigues, Catarina Figueiredo

Date: Sunday 31st December, 2023

Academic Year: 2023/2024

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 1
- 1.3 Goals . . . 1
- 1.4 Document Structure . . . 2
  - 1.4.1 Chapter 2 - Data Exploration and Preprocessing . . . 2
  - 1.4.2 Chapter 3 - Time Series Decomposition . . . 2
  - 1.4.3 Chapter 4 - Statistical Models . . . 2
  - 1.4.4 Chapter 5 - Machine Learning Models . . . 2
  - 1.4.5 Chapter 6 - Deep Learning Models . . . 2
  - 1.4.6 Chapter 7 - Conclusions, Constraints, and Future Work . . . 3

2 Data Exploration and Preprocessing **4**
- 2.1 Data Exploration . . . 4
- 2.2 Data Preprocessing . . . 5

3 Time Series Decomposition **8**
- 3.1 Components . . . 8
  - 3.1.1 Trend . . . 8
  - 3.1.2 Seasonality . . . 9
  - 3.1.3 Residuals . . . 9

4 Statistical Models **11**
- 4.1 Baseline Models . . . 11
- 4.2 Time Series Stationarity . . . 12
- 4.3 SARIMA/SARIMAX . . . 12
  - 4.3.1 ACF and PACF . . . 12
  - 4.3.2 SARIMA Model Parameters . . . 14
  - 4.3.3 Defining SARIMA/SARIMAX Parameters . . . 15
  - 4.3.4 Residual Analysis . . . 15
  - 4.3.5 Sarima and SARMAX forecasting . . . 16
  - 4.3.6 SARIMA vs. SARIMAX Forecasting . . . 17
- 4.4 ARIMA Model . . . 18
- 4.5 Baseline vs SARIMA . . . 18
- 4.6 Model Evaluation . . . 19
  - 4.6.1 Rolling Forecast Evaluation . . . 19
  - 4.6.2 Normal Forecast . . . 19

5 Machine Learning Models **20**
- 5.1 Data Preparation . . . 20
- 5.2 Models Used . . . 21
  - 5.2.1 Linear Regression . . . 21
  - 5.2.2 Random Forest . . . 21
  - 5.2.3 LightGB . . . 22
  - 5.2.4 Gradient Boosting . . . 22
  - 5.2.5 XGBoost . . . 22
  - 5.2.6 Neural Network . . . 23

5.2.7 Support Vector Regression . . . 23
5.2.8 K-Nearest Neighbours . . . 23
5.3 Single-Step Approach . . . 24
5.3.1 Sliding Window and train/test split . . . 24
5.3.2 Models forecasts . . . 25
5.3.3 Model evaluation . . . 29
5.4 Multi-Step Approach . . . 31
5.4.1 Sliding Window and train/test split . . . 31
5.4.2 Models forecasts . . . 31
5.4.3 Model evaluation . . . 35

6 Deep Learning Models **38**
6.1 Data Preparation . . . 38
6.2 Models Used . . . 38
6.2.1 Long Short-Term Memory (LSTM) . . . 38
6.2.2 Gated Recurrent Unit (GRU) . . . 39
6.3 Forecasting approach . . . 40
6.3.1 Single-step . . . 40
6.3.2 Multi-step . . . 41
6.3.3 Single-step Multivariate . . . 42
6.3.4 Multi-step Multivariate . . . 43
6.4 Model Evaluation . . . 44
6.4.1 Mean square error . . . 45
6.4.2 Mean absolute error . . . 45
6.4.3 Root mean square error . . . 46
6.4.4 Model's performance overview . . . 46
6.4.5 Conclusion . . . 46

7 Conclusion, Constraints and Future Work **47**
7.1 Conclusion . . . 47
7.2 Constraints . . . 47
7.3 Future Work . . . 47

List of Figures

2.1 Time series plot of electric power consumption (load). 4
2.2 Time series plot of temperature. 5
2.3 Combined time series plot of load and temperature. 5
2.4 Dataset with no changes. 6
2.5 Date and Hour merged 6
2.6 Correlation between Load and Temperature 7

3.1 trend 9
3.2 Seasonal Component (first 1000 data points) 9
3.3 residuals 10

4.1 Baseline Models 11
4.2 ADFuler check 12
4.3 ACF plot 13
4.4 PACF plot 14
4.5 SARIMA 14
4.6 q-q plot 15
4.7 Residuals Histogram 16
4.8 Sarima Forecasting 17
4.9 Sarimax Forecasting 17
4.10 SARIMA vs SARIMAX 18
4.11 ARIMA 18
4.12 Rolling Forecast Sarima vs Baseline 19
4.13 Rolling Forecast Sarima vs Baseline MSE 19

5.1 Sliding Window example 20
5.2 Sliding Window function 21
5.3 Linear Regression 21
5.4 Random Forest 21
5.5 LGB 22
5.6 GB 22
5.7 XGB 22
5.8 Neural Networks 23
5.9 SVR 23
5.10 Multiple Neighbours 24
5.11 24
5.12 Single-Step Data preparation 25
5.13 Linear Regression Forecast 25
5.14 Random Forest Forecast 26
5.15 LGB Forecast 26
5.16 GB Forecast 27
5.17 XGB Forecast 27
5.18 Neural Networks Forecast 28
5.19 SVR Forecast 28
5.20 KNN Forecast 29
5.21 Single-Step MSE 29
5.22 Single-Step MAE 30

5.23 Single-Step RMSE . . . 30
5.24 Multi-Step Data preparation . . . 31
5.25 Linear Regression Forecast . . . 32
5.26 Random Forest Forecast . . . 32
5.27 LGB Forecast . . . 33
5.28 GB Forecast . . . 33
5.29 XGB Forecast . . . 34
5.30 Neural Networks Forecast . . . 34
5.31 SVR Forecast . . . 35
5.32 KNN Forecast . . . 35
5.33 Multi-Step MSE . . . 36
5.34 Multi-Step MAE . . . 36
5.35 Multi-Step RMSE . . . 37

6.1 DL data Transformation example . . . 38
6.2 LSTM architecture . . . 39
6.3 GRU architecture . . . 40
6.4 Single-Step LSTM Forecasts . . . 41
6.5 Single-Step Gru Forecasts . . . 41
6.6 Multi-Step LSTM Forecasts . . . 42
6.7 Multi-Step Gru Forecasts . . . 42
6.8 Single-Step Multivariate LSTM Forecasts . . . 43
6.9 Single-Step Multivariate Gru Forecasts . . . 43
6.10 Multi-Step LSTM Multivariate Forecasts . . . 44
6.11 Multi-Step Gru Multivariate Forecasts . . . 44
6.12 Deep Learning MSE Comparison . . . 45
6.13 Deep Learning MAE Comparison . . . 45
6.14 Deep Learning RMSE Comparison . . . 46

Listings

2.1 DateTime Conversion. 7

Chapter 1

Introduction

1.1 Context

In recent years, the demand for electric power has surged, driven by the ever-increasing reliance on electronic devices and a growing global population. This escalating demand poses significant challenges for energy providers who must balance the delicate equation of supplying enough energy to meet peak demands while avoiding overproduction that can strain the grid and lead to disconnection risks. The need for accurate forecasting of energy consumption has never been more critical, providing a proactive solution to grid management and ensuring a reliable and sustainable energy supply for households.

1.2 Motivation

The motivation behind this project stems from the pressing necessity to address the challenges faced by energy companies in effectively planning and managing the distribution of electricity. The traditional energy grid is no longer sufficient in handling the complexities introduced by fluctuating consumption patterns and dynamic environmental factors. Accurate forecasting of electric power consumption emerges as a pivotal tool to empower energy providers in adapting to the evolving landscape of energy demands, optimizing load distribution, and ultimately fostering a resilient and efficient energy infrastructure.

1.3 Goals

This project aims to delve into the intricacies of electric power consumption forecasting using a rich dataset spanning three years (2012-2014). By leveraging statistical models, machine learning algorithms, and deep learning networks, we aspire to develop robust one-step and multi-step forecasting models. The goals can be summarized as follows:

1. **Statistical Modeling:** Selecting and deriving a subset of the dataset suitable for application with statistical models. Employing an appropriate statistical function to model the electric power consumption series effectively.
2. **Machine Learning Models:** Employing various regression algorithms to capture the nuanced relationships within the dataset. Comparing and contrasting the effectiveness of these models in forecasting energy consumption.
3. **Deep Learning Models:** Implementing Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks to harness the power of deep learning in capturing temporal dependencies and patterns within the dataset.
4. **Model Evaluation:** Conducting a comprehensive evaluation of all models, comparing their performance against at least one baseline model. Assessing the accuracy, efficiency, and reliability of each approach.
5. **Model Selection:** Identifying and selecting the best-performing model based on the evaluation metrics. Providing insights into the strengths and limitations of each approach and offering recommendations for real-world implementation.

Through these endeavors, this project aims to contribute valuable insights and methodologies to the field of energy forecasting, facilitating advancements in grid management and paving the way for a more sustainable and resilient energy future.

1.4 Document Structure

1.4.1 Chapter 2 - Data Exploration and Preprocessing

In this chapter, the focus is on understanding and preparing the dataset for analysis. Key components include:

- **Introduction to the Dataset:** Provide a brief overview of the dataset, highlighting the variables and their significance.
- **Exploratory Data Analysis (EDA):** Discuss the exploratory analysis conducted, such as data distribution, outliers, and patterns.
- **Data Preprocessing:** Detail the steps taken for handling missing data, outliers, and feature engineering.

1.4.2 Chapter 3 - Time Series Decomposition

This chapter centers on decomposing the time series data to better understand its components:

- **Introduction to Time Series Decomposition:** Explain the concept of time series decomposition and its importance.
- **Decomposition Methods:** Discuss the methods used to decompose the time series into trend, seasonal, and residual components.
- **Visualization:** Provide visual representations of the decomposed components.

1.4.3 Chapter 4 - Statistical Models

Here, the focus shifts to implementing and evaluating statistical models:

- **Introduction to Statistical Models:** Briefly introduce statistical models like SARIMA.
- **Model Implementation:** Detail how statistical models are applied to the dataset.
- **Validation and Performance Metrics:** Discuss the validation process and the metrics used to evaluate the performance of these models.

1.4.4 Chapter 5 - Machine Learning Models

This chapter explores the application of machine learning algorithms for forecasting:

- **Introduction to Machine Learning Models:** Provide an overview of regression algorithms used in this project.
- **Model Implementation:** Describe how machine learning models are trained on the dataset.
- **Model Evaluation:** Discuss the evaluation metrics used to assess the performance of machine learning models.

1.4.5 Chapter 6 - Deep Learning Models

Here, the focus is on the implementation and evaluation of deep learning models:

- **Introduction to Deep Learning Models:** Briefly introduce LSTM and GRU networks.
- **Model Architecture:** Describe the architecture of the implemented deep learning models.
- **Training and Evaluation:** Explain the training process and the metrics used to evaluate the deep learning models.

1.4.6 Chapter 7 - Conclusions, Constraints, and Future Work

This concluding chapter wraps up the report by:

- **Summarizing Findings:** Summarize the key findings from the analysis.
- **Constraints:** Discuss any limitations or constraints faced during the project.
- **Future Work:** Propose potential directions for future research or improvements in forecasting models.

Chapter 2

Data Exploration and Preprocessing

In this chapter, we explore the dataset through various visualizations and perform necessary preprocessing steps.

2.1 Data Exploration

We conducted a detailed exploration of the dataset to understand its characteristics. Key visualizations include:

1. Time Series Plot of Load:

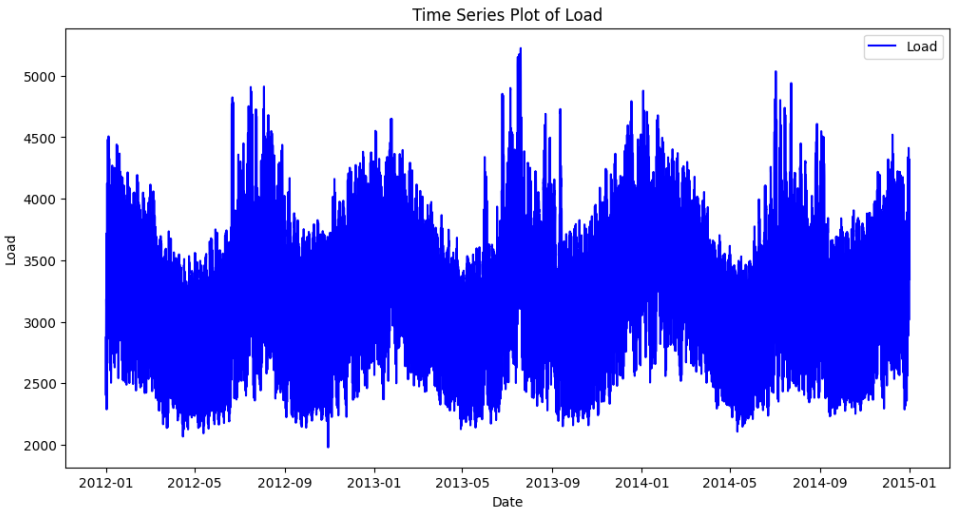

Figure 2.1: Time series plot of electric power consumption (load).

2. Plot of Temperature:

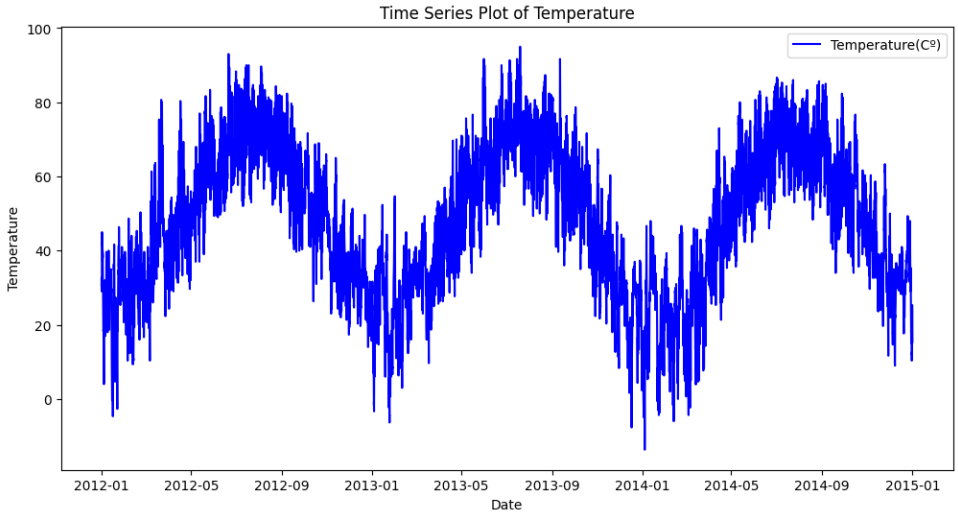

Figure 2.2: Time series plot of temperature.

3. Combined Plot of Load and Temperature:

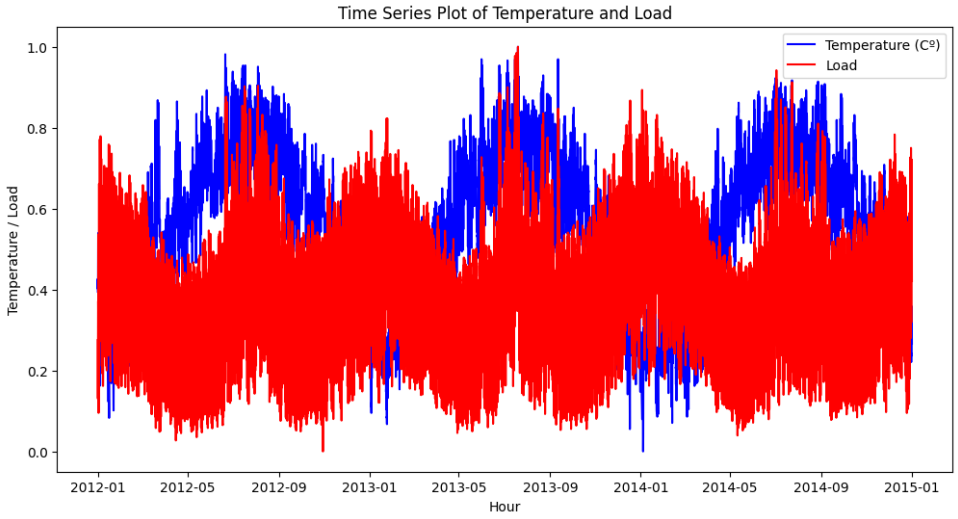

Figure 2.3: Combined time series plot of load and temperature.

2.2 Data Preprocessing

In terms of preprocessing, the following steps were taken:

1. Merging Date and Hour Columns: We merged the Date and Hour columns to create a new Date-Time column, facilitating easier temporal analysis.

	Date	Hour	load	T
0	2012-01-01	0	2872.0	30.666667
1	2012-01-01	1	2698.0	32.000000
2	2012-01-01	2	2558.0	32.666667
3	2012-01-01	3	2444.0	30.000000
4	2012-01-01	4	2402.0	31.000000
...	...	...	...	...
26300	2014-12-31	20	4012.0	18.000000
26301	2014-12-31	21	3856.0	16.666667
26302	2014-12-31	22	3671.0	17.000000
26303	2014-12-31	23	3499.0	15.333333
26304	2014-12-31	24	3345.0	15.333333

Figure 2.4: Dataset with no changes.

	load	T
Datetime		
2012-01-01 00:00:00	2872.0	30.666667
2012-01-01 01:00:00	2698.0	32.000000
2012-01-01 02:00:00	2558.0	32.666667
2012-01-01 03:00:00	2444.0	30.000000
2012-01-01 04:00:00	2402.0	31.000000
...	...	...
2012-01-13 07:00:00	3418.0	30.000000
2012-01-13 08:00:00	3681.0	29.333333
2012-01-13 09:00:00	3768.0	29.333333
2012-01-13 10:00:00	3818.0	30.333333
2012-01-13 11:00:00	3856.0	30.666667

Figure 2.5: Date and Hour merged

2. Conversion to DateTime: The merged DateTime column was converted to the DateTime data type for more straightforward time-based operations.

Listing 2.1: DateTime Conversion.

```
# Convert to datetime, considering "24" as "0" of the next day
df['Datetime'] = pd.to_datetime(df['Date'].astype(str) + ' '
+ df['Hour'].astype(str) + ':00:00', errors='coerce')

# Adjust the hour for values equal to 24 and increment the date
mask = df['Hour'] == 24
df.loc[mask, 'Datetime'] = pd.to_datetime(df.loc[mask, 'Date'].astype(str)
+ ' 00:00:00', errors='coerce') + pd.DateOffset(days=1)

# Drop the original columns if needed
df = df.drop(['Date', 'Hour'], axis=1)
df.set_index('Datetime', inplace=True)
print(df.head(300))
```

3. Correlation Analysis: We calculated the correlation between the load and temperature variables. Surprisingly, the correlation coefficient was found to be negligible, suggesting a lack of direct linear relationship.

```
#let's do the math and check if our variable are correlated

# Check the correlation between "load" and "T" columns
correlation = df_normalized['load'].corr(df['T'])

print(f'Correlation between "load" and "T": {correlation}')
```

✓ 0.0s

```
Correlation between "load" and "T": 0.0726679606476381
```

Figure 2.6: Correlation between Load and Temperature

A thorough exploration of the dataset revealed a notable observation: no missing values were identified within the time series data. This absence of missing data contributes to the robustness of our analysis, ensuring a complete and reliable foundation for subsequent modeling and forecasting endeavors. The availability of a complete dataset minimizes potential biases and enhances the credibility of the insights derived from our comprehensive exploration.

The visualizations and preprocessing steps provide valuable insights into the dataset, setting the foundation for subsequent modeling and analysis.

Chapter 3

Time Series Decomposition

The "Time Series Decomposition" chapter plays a vital role in examining the complex patterns within the electric power consumption dataset. This method involves untangling the temporal dynamics by identifying trends, seasonality, and residual variations. Using advanced decomposition techniques, the chapter aims to offer a detailed understanding of how external factors and inherent patterns interact over time.

The content comprises a thorough overview of time series decomposition concepts, methodologies, and visual representations of decomposed components. The primary objective is to improve forecasting precision by uncovering the temporal intricacies of the dataset, establishing the foundation for subsequent modeling phases in predicting electric energy consumption effectively.

3.1 Components

In this section, we systematically performed time series decomposition on the electric power consumption dataset, employing an additive approach. By isolating trend, seasonality, and residuals, we aimed to discern the fundamental components contributing to the overall temporal behavior. Notably, a period of 24 was utilized for visualization, aligning with the hourly nature of the dataset. The ensuing subsections delve into each component.

3.1.1 Trend

The trend component in time series decomposition represents the long-term direction in the data, showcasing whether the variable is increasing, decreasing, or remaining relatively constant over an extended period. Identifying the trend is crucial for understanding the underlying, persistent patterns that extend beyond short-term fluctuations. This subsection will delve into methodologies for recognizing and extracting trends within a time series, shedding light on the overarching directional behavior of the electric power consumption dataset.

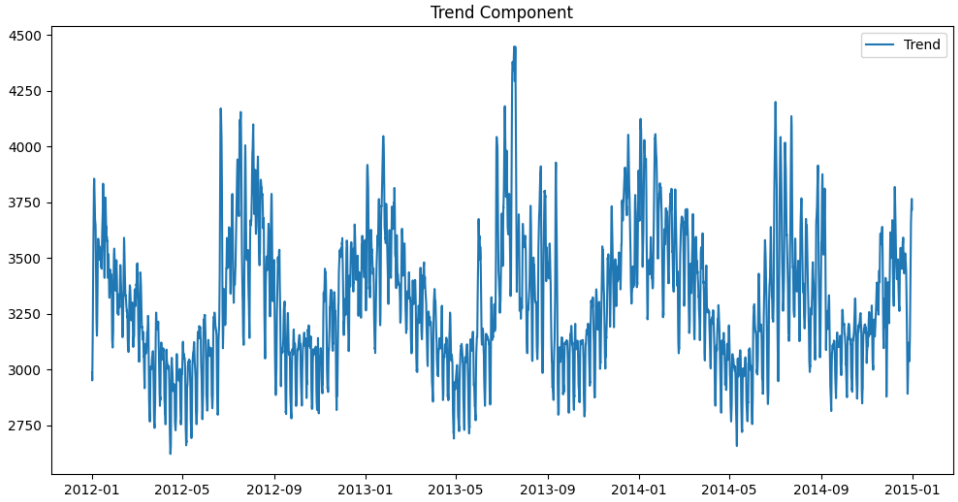

Figure 3.1: trend

3.1.2 Seasonality

Seasonality captures the repetitive, periodic patterns that occur at fixed intervals within a time series. In the context of electric power consumption, seasonality may manifest as regular fluctuations based on daily, weekly, or monthly cycles. Understanding and isolating these seasonal patterns are essential for anticipating and adapting to recurring consumption variations. This subsection will explore techniques for detecting and interpreting seasonality within the dataset, providing insights into the cyclic behavior inherent in the electric power consumption time series.

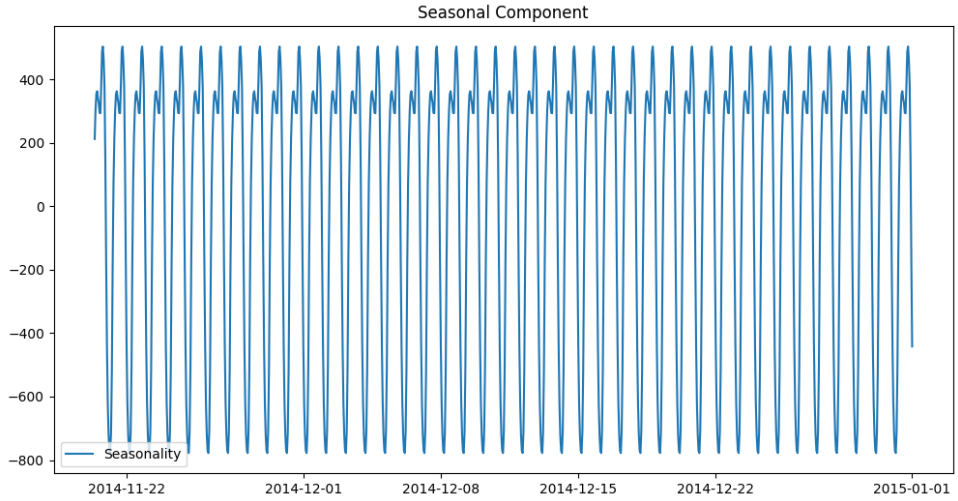

Figure 3.2: Seasonal Component (first 1000 data points)

3.1.3 Residuals

Residuals represent the unexplained or random variations in a time series after removing the trend and seasonality components. Analyzing residuals is critical for assessing the model's ability to capture the underlying patterns and for identifying any remaining irregularities. In the realm of electric power consumption forecasting, examining residuals allows us to gauge the accuracy of our decomposition and modeling efforts. This subsection will examine methods for interpreting residuals, emphasizing their significance in evaluating the effectiveness of the time series decomposition process.

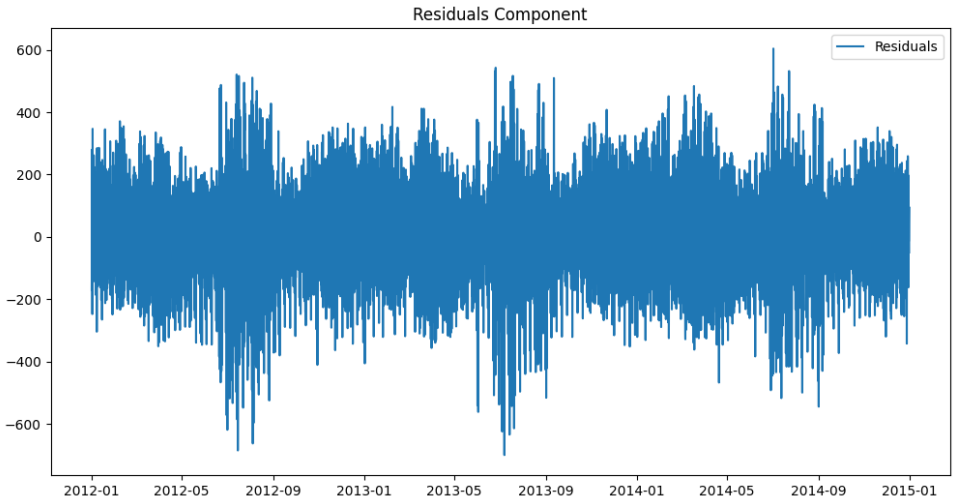

Figure 3.3: residuals

Chapter 4

Statistical Models

In this chapter, statistical models are utilized to predict electricity load and assess their forecasting accuracy. Predictions focus on the final week of the dataset, involving forecasting the last 168 hours while using preceding data for training. Baseline models are initially employed, followed by the application of SARIMA/SARIMAX models, with ARIMA also included as an additional model.

4.1 Baseline Models

Before exploring statistical models, two baseline models were applied to predict load: the historical mean and the last value method.

1. **Historical Mean Model:** This model calculates the average load value over the historical dataset and predicts future values based on this mean. It provides a simple yet informative benchmark for forecasting.

2. **Last Value Method:** The last value method predicts future load values using the most recent observed value. It assumes that the future will resemble the latest recorded pattern, offering a straightforward approach.

Both models are purely mathematical and serve as simple benchmarks. To assess their performance, a rolling forecast was employed, allowing for continuous validation and refinement of predictions.

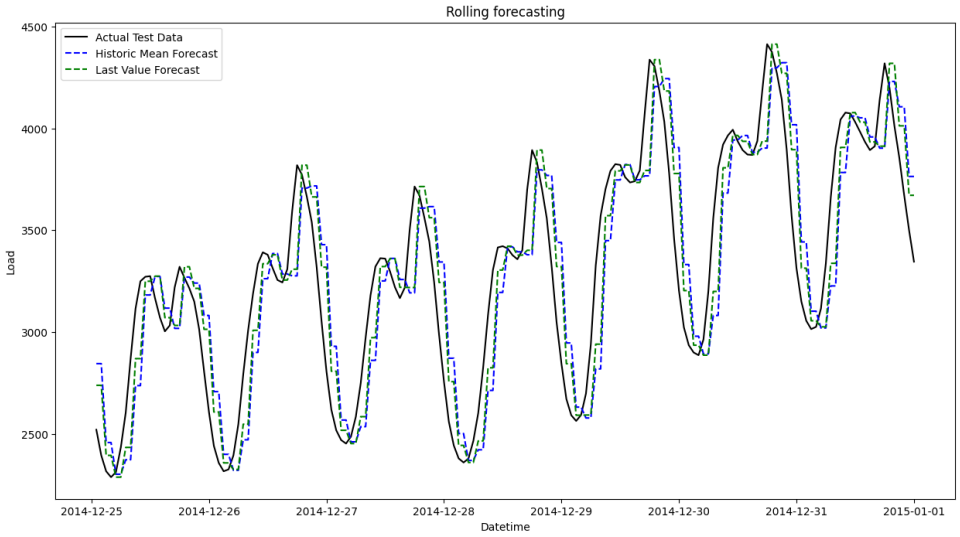

Figure 4.1: Baseline Models

4.2 Time Series Stationarity

Time series stationarity is a crucial concept in time series analysis, referring to the statistical properties of a time series remaining constant over time. A stationary time series exhibits consistent mean, variance, and autocorrelation, making it more amenable to modeling and forecasting.

Achieving stationarity often involves addressing issues like trend and seasonality present in the data. Common techniques to attain stationarity include:

- Differencing: Subtracting the current observation from the previous one helps eliminate trends and reduce non-constant variance.
- Transformation: Applying mathematical transformations such as logarithms can stabilize variance and linearize trends.
- Seasonal Differencing: Removing seasonal patterns by differencing observations at the seasonal period.
- Detrending: Removing trend components using techniques like polynomial fitting or moving averages.

In the context of this study, an Augmented Dickey-Fuller (ADF) test for stationarity was conducted on the electric power consumption time series. Surprisingly, the results indicated that the time series was already stationary. This finding eliminated the need for additional preprocessing steps, and the section discusses the significance of stationarity and its implications for modeling the time series effectively.

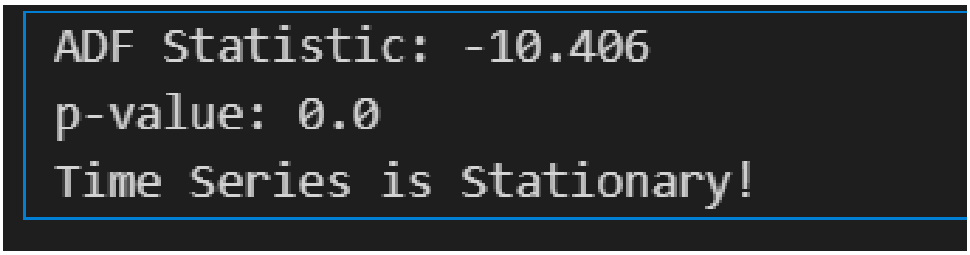

Figure 4.2: ADFuler check

4.3 SARIMA/SARIMAX

In this section, we present results from SARIMA and SARIMAX models, the latter incorporating temperature as an exogenous variable.

Two forecasting methods were employed—rolling forecast and standard forecast—allowing comparison with baseline models. SARIMA/SARIMAX was chosen due to the time series' seasonality, a characteristic effectively addressed by these models.

4.3.1 ACF and PACF

In this section, we delve into the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) as indispensable tools for comprehending the temporal dependencies inherent in the electric power consumption time series.

- **Autocorrelation Function (ACF):**
  - ACF measures the correlation between an observation and its lagged values at different time intervals. It aids in identifying the presence of serial correlation in the time series. Peaks in the ACF plot indicate significant correlations at specific lags, offering insights into potential patterns.
- **Partial Autocorrelation Function (PACF):**
  - PACF represents the correlation between two observations after accounting for the effects of all the intermediate lags. It helps discern the direct relationship between observations, providing clarity on the underlying temporal structure. Significant spikes in the PACF plot indicate the order of autoregressive terms in the time series.

By analyzing the ACF and PACF plots, valuable information about the lagged relationships within the time series is obtained, guiding the selection of appropriate parameters for models like ARIMA and SARIMA.

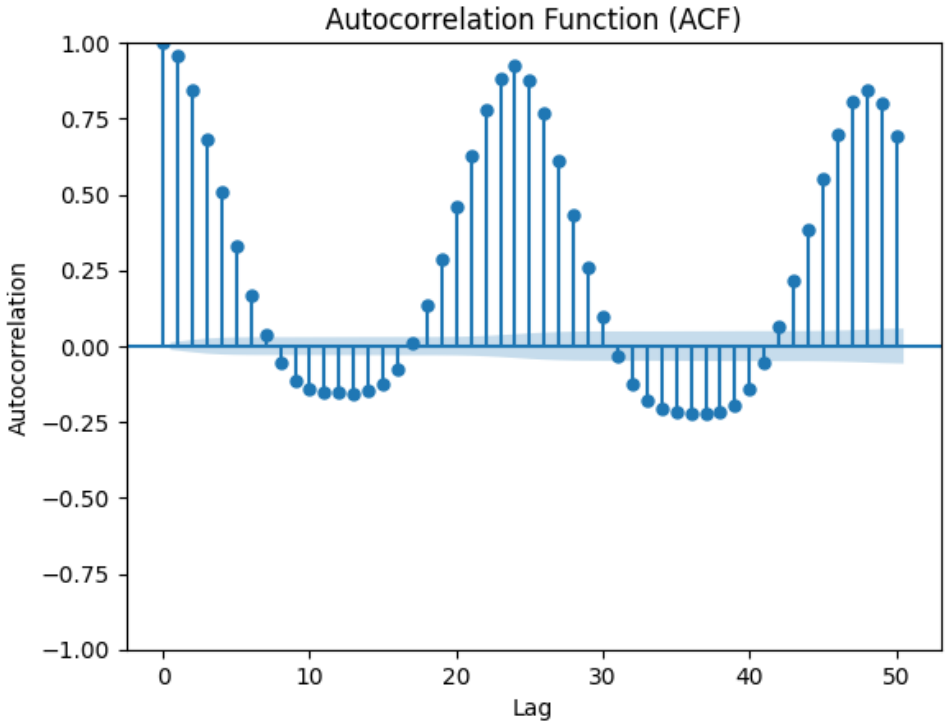

Figure 4.3: ACF plot

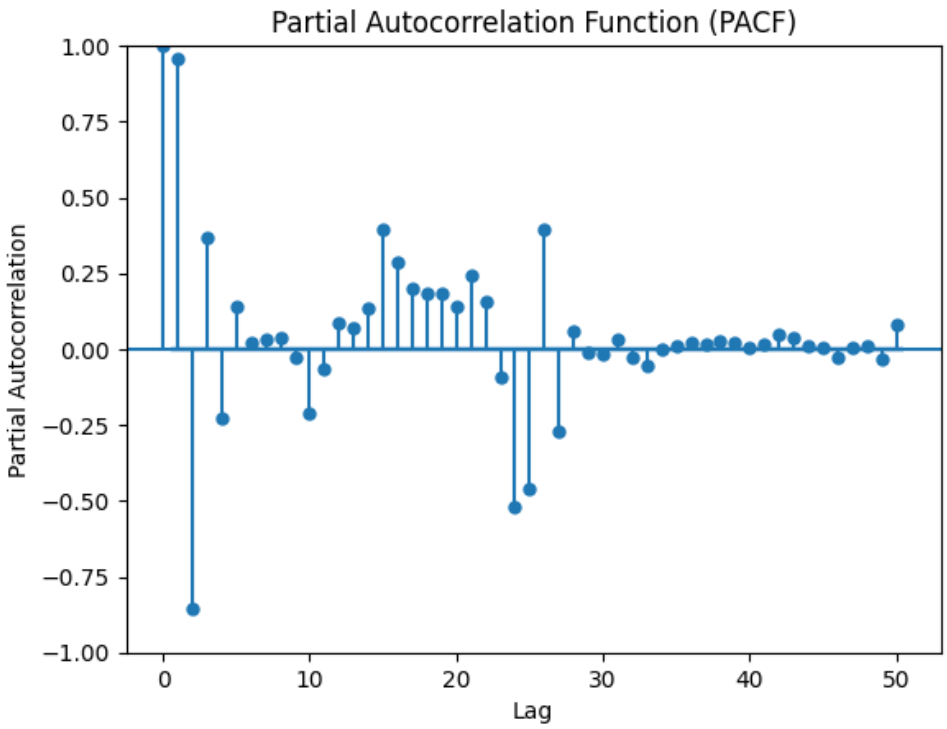

Figure 4.4: PACF plot

4.3.2 SARIMA Model Parameters

$$SARIMA \underbrace{(p,d,q)}_{non-seasonal} \underbrace{(P,D,Q)_m}_{seasonal}$$

Figure 4.5: SARIMA

The Seasonal AutoRegressive Integrated Moving Average (SARIMA) model is characterized by several parameters that collectively define its structure. These parameters include:

- **p (AR order):** The autoregressive order, denoting the number of lag observations included in the model. It captures the relationship between the current observation and its preceding values.
- **d (Integration order):** The differencing order, indicating the number of times the time series is differenced to achieve stationarity. It represents the degree of differencing required to make the series stationary.
- **q (MA order):** The moving average order, representing the number of lagged forecast errors used in the model. It accounts for the impact of past errors on the current observation.
- **P (Seasonal AR order):** The seasonal autoregressive order, analogous to 'p' but applied to the seasonal component of the time series.
- **D (Seasonal Integration order):** The seasonal differencing order, similar to 'd' but applied to the seasonal component.
- **Q (Seasonal MA order):** The seasonal moving average order, akin to 'q' but applied to the seasonal component.

- **s (Seasonal Periodicity):** The number of time steps in each seasonal period. It defines the seasonality of the time series, indicating the frequency at which the pattern repeats.

These parameters collectively configure the SARIMA model to capture the temporal patterns and seasonal variations present in the electric power consumption time series data.

4.3.3 Defining SARIMA/SARIMAX Parameters

In the process of defining SARIMA and SARIMAX parameters, an exhaustive search spanned multiple combinations of p, q, P, Q within the range of 1 to 4. Notably, given the stationarity of the electric power consumption time series ($d = D = 0$), emphasis was placed on optimizing autoregressive and moving average terms.

Ultimately, the choice of $p = 2$, $q = 3$, $P = 2$, $Q = 2$, and $m = 24$ was made based on the minimization of the Akaike Information Criterion (AIC). This combination yielded the lowest AIC among the tested configurations, indicating its superior fit to the time series data and justifying its selection as the optimal set of parameters.

Due to the computational demands and runtime considerations, a more extensive search was constrained. It was acknowledged that the SARIMA modeling process can be resource-intensive, limiting the ability to explore a broader parameter space effectively. As a compromise, the search was conducted within a reasonable range to strike a balance between computational efficiency and model effectiveness.

4.3.4 Residual Analysis

Residual analysis is a pivotal step in assessing the effectiveness of a time series model. Residuals represent the differences between observed and predicted values, offering insights into the model's ability to capture underlying patterns. By examining the residuals, we gain valuable information about the presence of systematic patterns, potential model inadequacies, and the overall reliability of the forecasting model. This section explores the outcomes of residual analysis, shedding light on the accuracy and appropriateness of the employed time series models for electric power consumption forecasting.

This section explores the outcomes of residual analysis, which includes performing the Ljung-Box test to assess residual autocorrelation. Additionally, the residuals are visualized through Q-Q plots and histograms, providing a comprehensive examination of their distribution and aiding in the evaluation of model assumptions.

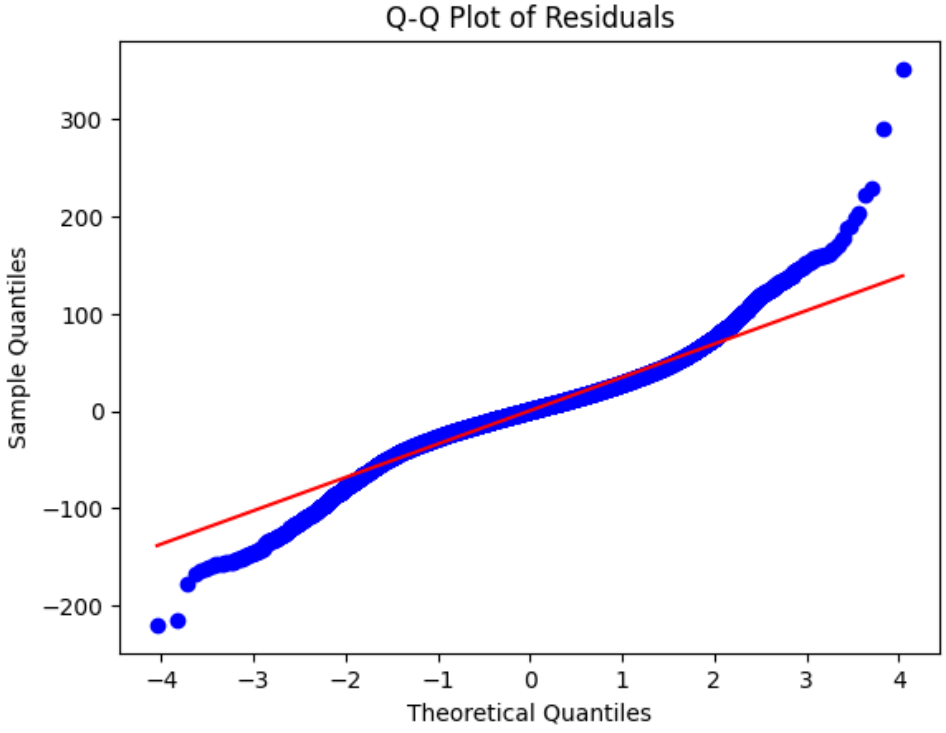

Figure 4.6: q-q plot

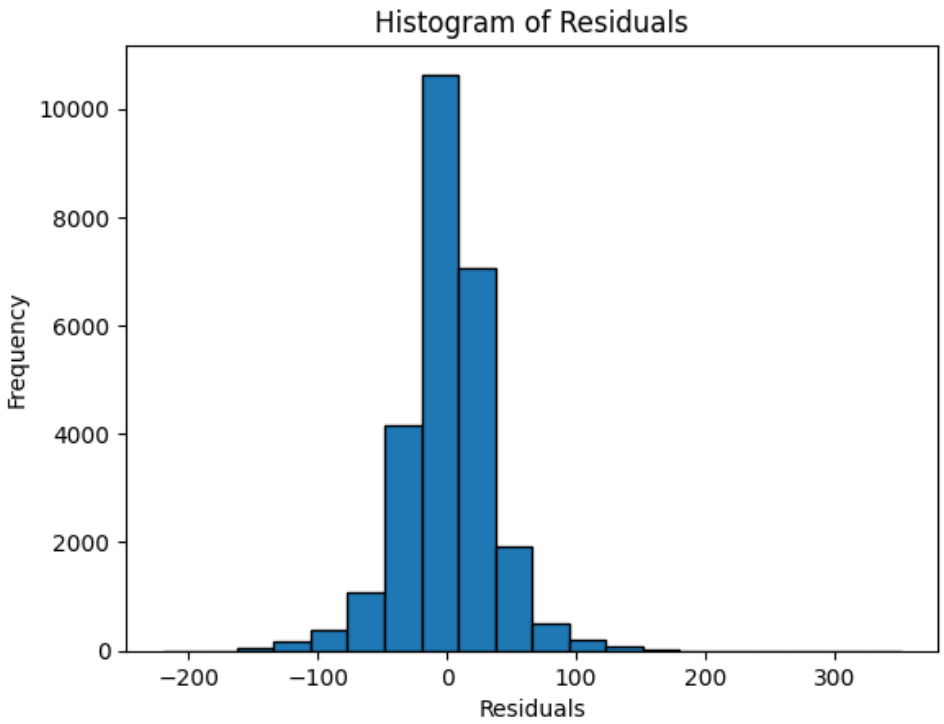

Figure 4.7: Residuals Histogram

The results from the Ljung-Box test are summarized as follows:

Count	1.00×10
Mean	4.90×10^{-2}
Std	1.54×10^{-1}
Min	1.48×10^{-39}
25%	5.33×10^{-17}
50%	8.89×10^{-7}
75%	1.89×10^{-4}
Max	4.88×10^{-1}

The results from the Ljung-Box test, with p-values consistently lower than 0.05, indicate significant residual autocorrelation. Additionally, the Q-Q plot does not exhibit a linear trend, suggesting a departure from normal distribution in the residuals.

These findings collectively imply that the residuals show correlation, signaling potential inadequacies in the model for forecasting purposes. Further refinement or exploration of alternative modeling approaches may be warranted to enhance the model's predictive performance.

4.3.5 Sarima and SARMAX forecasting

Both models were used for normal forecasting on the last week of our data (168 hours). Here are two plots showcasing their forecasting capabilities.

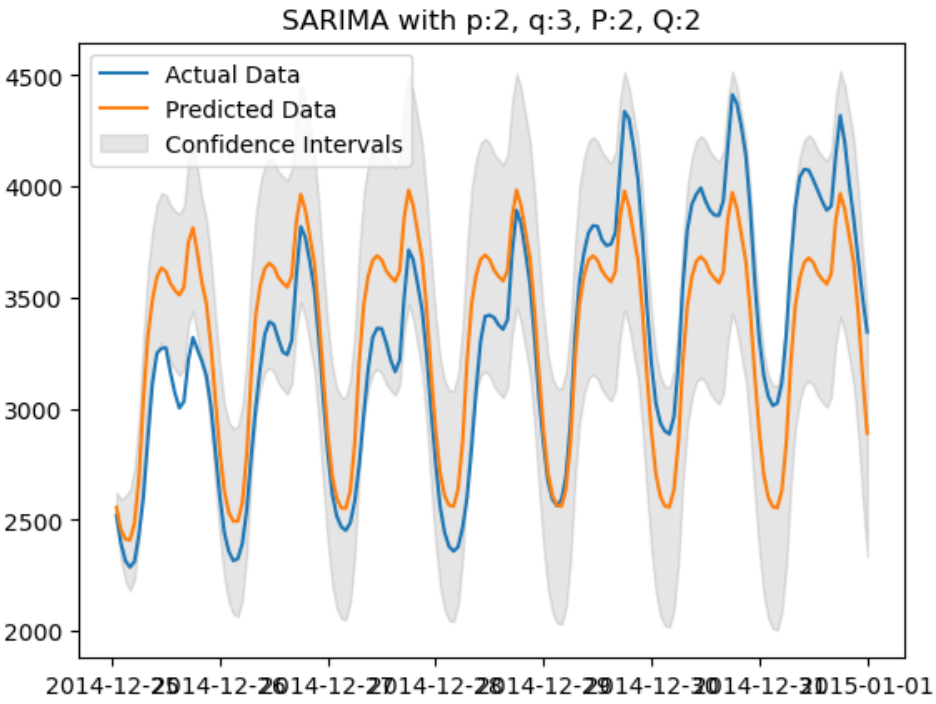

Figure 4.8: Sarima Forecasting

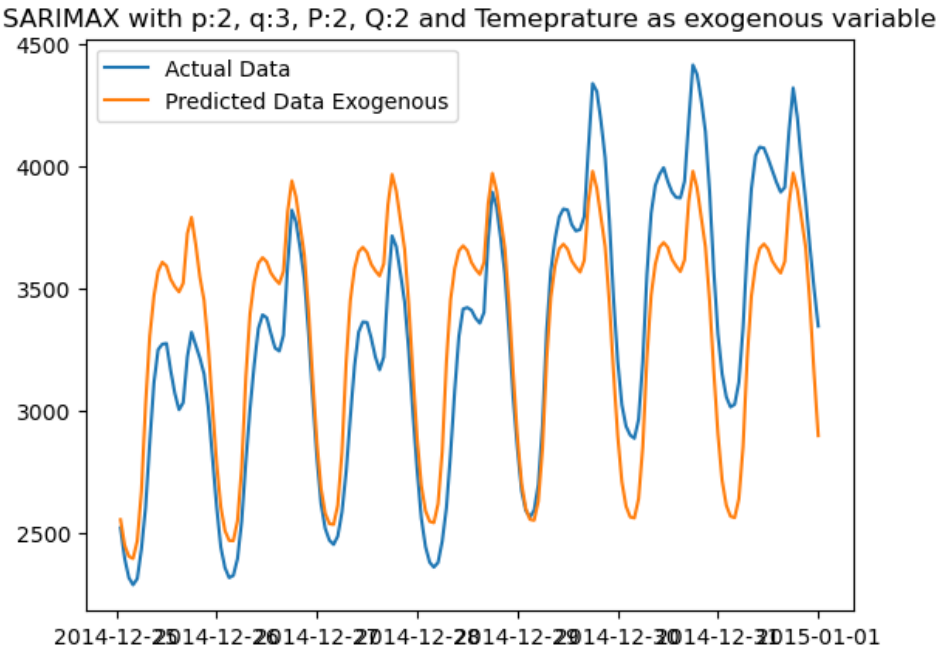

Figure 4.9: Sarimax Forecasting

4.3.6 SARIMA vs. SARIMAX Forecasting

SARIMAX outperformed SARIMA slightly, with mean squared errors (MSE) of 97k and 90k, respectively. This suggests that the inclusion of temperature (exogenous variable) had a discernible impact on the model's performance.

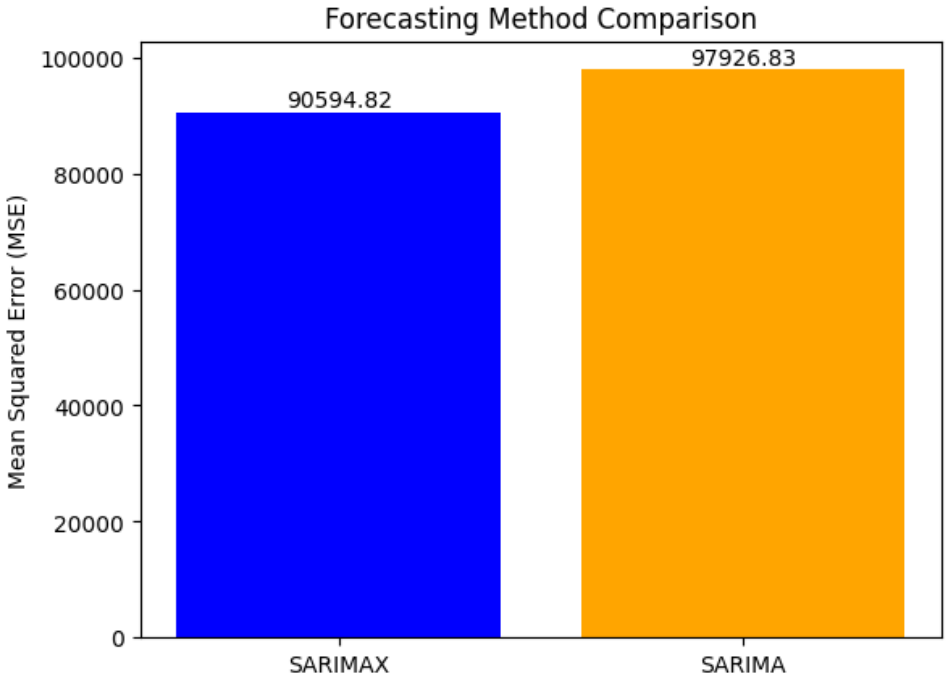

Figure 4.10: SARIMA vs SARIMAX

4.4 ARIMA Model

Despite lacking an inherent seasonal component, we incorporated the ARIMA model as an additional factor. After an extensive search for optimal parameters (p, q) with d = 0, we identified 5 and 6 as the optimal values. However, the model performed poorly in forecasting our data, evident from a high MSE of 290560. Here's the plot:

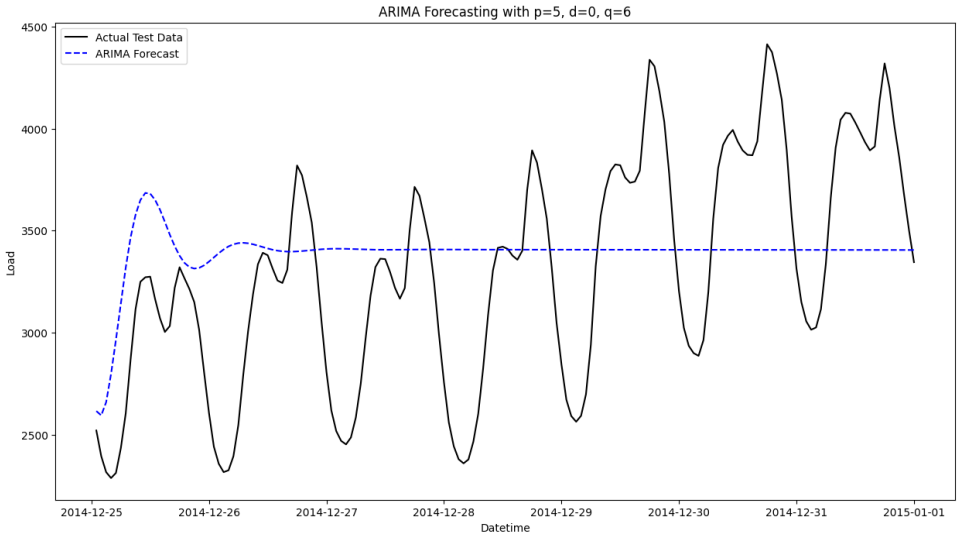

Figure 4.11: ARIMA

4.5 Baseline vs SARIMA

In a rolling forecast, SARIMA outperformed both baseline models and its own performance in normal forecasting, as evident from lower MSE and improved visualization. This superiority is attributed to

the nature of the rolling forecast.Notably, SARIMAX could not be applied due to its expected longer runtime, exceeding 900 minutes, which is impractical for debugging.

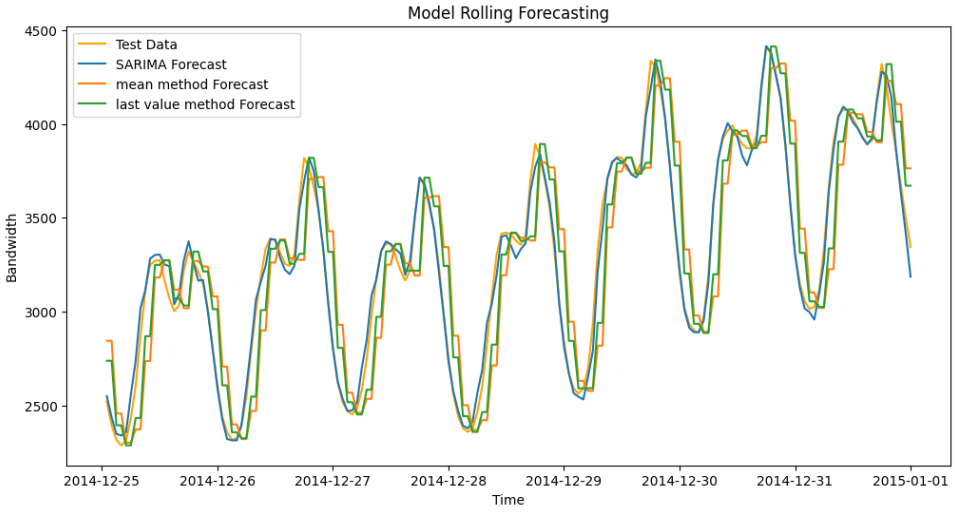

Figure 4.12: Rolling Forecast Sarima vs Baseline

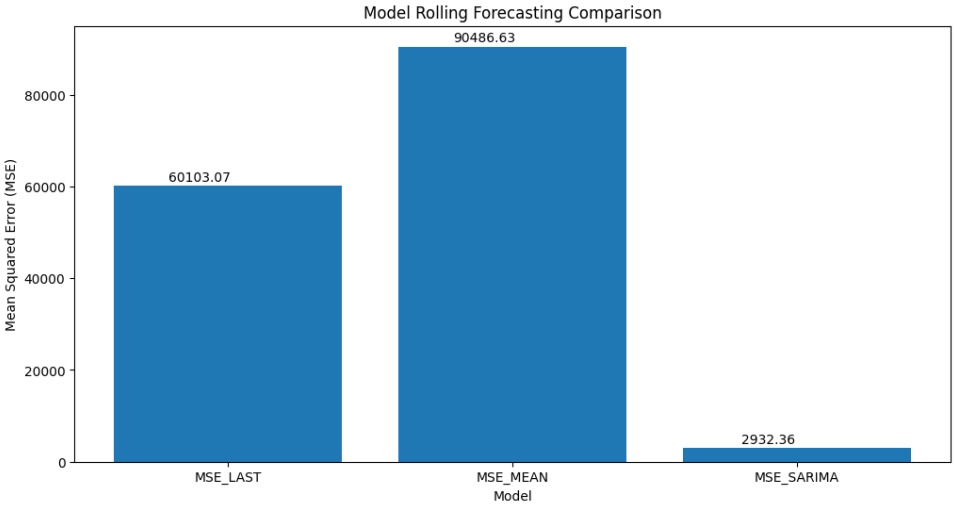

Figure 4.13: Rolling Forecast Sarima vs Baseline MSE

4.6 Model Evaluation

4.6.1 Rolling Forecast Evaluation

In the rolling forecast analysis, SARIMA demonstrated superior performance compared to both baseline models and its own normal forecasting results, as indicated by lower MSE and improved visualization. This advantage can be attributed to the nature of the rolling forecast

4.6.2 Normal Forecast

In the normal forecast assessment, SARIMA and SARIMAX exhibited similar performance, with SARIMAX showing a slight advantage. Notably, SARIMAX performed marginally better in this context.

Chapter 5

Machine Learning Models

In this chapter, we adopt a different approach by implementing various machine learning models, primarily regression algorithms, and others, employing two distinct strategies: single-step and multi-step forecasting.

5.1 Data Preparation

In the data preparation phase, we transformed the dataset into a tabular format using a sliding window approach. This involved:

1. **Sliding Window Technique:** We divided the time series data into overlapping windows, where each window represented a snapshot of the data. This technique helps capture temporal patterns and relationships within the dataset.
2. **Feature Extraction:** From each window, relevant features were extracted, such as lagged values, statistical measures, or other domain-specific indicators. These features served as input variables for the machine learning models.
3. **Target Definition:** Corresponding to each window, we defined the target variable based on the forecasting horizon—whether it's a single-step or multi-step prediction. For single-step forecasting, the target is typically the next data point, while for multi-step forecasting, it involves predicting several future data points.
4. **Tabular Structuring:** The extracted features and corresponding target variables were organized into a tabular structure. This format is conducive to training machine learning models, allowing them to learn patterns and relationships from the historical data.

By adopting this approach, we harnessed temporal dependencies for effective model training and forecasting. Here's a visual representation of the process:

Window = 6
Horizon = 3

	date	temperature
0	2009-01-01	-6.810629
1	2009-01-02	-3.360486
2	2009-01-03	5.435694
3	2009-01-04	7.283889
4	2009-01-05	12.690069

date	x1	x2	x3	x4	x5	x6	y1	y2	y3
2009-01-01	-6.810629	-3.360486	5.435694	7.283889	12.690069	15.201597	20.121875	18.864792	21.289722
2009-01-02	-3.360486	5.435694	7.283889	12.690069	15.201597	20.121875	18.864792	21.289722	11.937847
2009-01-03	5.435694	7.283889	12.690069	15.201597	20.121875	18.864792	21.289722	11.937847	3.210903
2009-01-04	7.283889	12.690069	15.201597	20.121875	18.864792	21.289722	11.937847	3.210903	3.682431
2009-01-05	12.690069	15.201597	20.121875	18.864792	21.289722	11.937847	3.210903	3.682431	-1.678194

Figure 5.1: Sliding Window example

Here's the code that performs this transformation:

```python
def sliding_window(data, window_size, horizon=1):
    """Performs a sliding window on a time series.

    Args:
      data: A Pandas DataFrame with a single column containing the time series data.
      window_size: The size of the sliding window.
      horizon: The number of steps to forecast into the future.

    Returns:
      X: Input features (2D array)
      y: Target labels (1D array)
    """

    X, y = [], []
    for i in range(0, len(data) - window_size - horizon + 1):
        window = data.iloc[i:i + window_size].values  # Extract the series values directly
        target = data.iloc[i + window_size + horizon - 1]  # Target is horizon steps ahead

        X.append(window)
        y.append(target)

    return np.array(X), np.array(y)
```

Figure 5.2: Sliding Window function

5.2 Models Used

In machine learning, we employed the following models: Linear Regression, Random Forest, LIGHTGB, Gradient Boosting (GB), XGBoost (XGB), Neural Networks, Support Vector Regression (SVR), and K-Nearest Neighbors (KNN). Each subsequent subsection will explain each model.

5.2.1 Linear Regression

Linear Regression is a simple and interpretable supervised learning algorithm used for predicting a continuous outcome by modeling a linear relationship between input features and the target variable.

```python
# Create and train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Figure 5.3: Linear Regression

5.2.2 Random Forest

Random Forest is an ensemble learning method that builds a multitude of decision trees during training. It operates by constructing a diverse set of trees and combining their predictions for robust and accurate results.

In our implementation, Random Forest was employed for regression tasks. The model builds multiple decision trees with randomly selected subsets of features and aggregates their predictions to reduce overfitting and enhance generalization. Random Forest is known for its simplicity, flexibility, and ability to handle complex relationships within the data, making it a powerful tool for regression analyses.

```python
# Train Random Forest Regressor
model_rf = RandomForestRegressor()
model_rf.fit(X_train, y_train)
```

Figure 5.4: Random Forest

5.2.3 LightGB

LightGBM is a gradient boosting framework designed for speed and efficiency. It builds decision trees in a leaf-wise manner and is known for its ability to handle large datasets and provide high-performance results.

In our implementation, we utilized LightGBM with the objective set to 'regression,' indicating that the model is configured for regression tasks. Additionally, the metric was set to Mean Squared Error (MSE), which is a measure of the average squared difference between predicted and actual values. This choice aligns with our regression analysis goal, aiming to minimize prediction errors and enhance model accuracy. LightGBM's efficiency and flexibility make it suitable for various regression tasks.

```
# Create and train a LightGBM model
model_lgb = lgb.LGBMRegressor(objective='regression', metric='mse')
model_lgb.fit(X_train, y_train)
```

Figure 5.5: LGB

5.2.4 Gradient Boosting

Gradient Boosting is an ensemble learning technique that combines the predictions of multiple weak learners, usually decision trees, to create a strong predictive model. It builds trees sequentially, with each subsequent tree addressing the errors made by the previous ones.

In our implementation, Gradient Boosting was employed for regression tasks. The model aims to minimize the residual errors through an iterative process, resulting in an aggregated predictor with improved accuracy. Gradient Boosting is versatile and effective, offering high performance in capturing complex relationships within the data.

```
# Create and train a Gradient Boosting model
model_gb = GradientBoostingRegressor()  # You
model_gb.fit(X_train, y_train)
```

Figure 5.6: GB

5.2.5 XGBoost

XGBoost is a powerful and efficient gradient boosting algorithm commonly used for regression and classification tasks. It builds an ensemble of weak learners (typically decision trees) and combines their predictions to create a robust model.

In our implementation, we used XGBoost with the objective set to 'reg:squarederror,' indicating that the model is configured for regression tasks and aims to minimize the mean squared error. This objective is suitable for predicting continuous values, aligning with our regression analysis. XGBoost offers flexibility and high performance, making it a popular choice in various machine learning applications.

```
# Create and train an XGBoost model
model = xgb.XGBRegressor(objective ='reg:squarederror')
model.fit(X_train, y_train)
```

Figure 5.7: XGB

5.2.6 Neural Network

Neural Networks, or artificial neural networks, are a class of machine learning models inspired by the human brain's structure. In our implementation, the neural network architecture consisted of three layers: one input layer, one middle hidden layer, and one output layer. Each layer contributes to the model's ability to learn complex patterns in the data.

The data was scaled to ensure consistent feature contributions. The optimizer 'adam' was employed, and the loss function 'mean_squared_error' was used to guide the training process. This configuration allows the neural network to efficiently learn and make predictions on continuous target variables.

```
# Standardize the input features
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

y_train_scaled = scaler.fit_transform(y_train.reshape(-1, 1)).flatten()
y_test_scaled = scaler.transform(y_test.reshape(-1, 1)).flatten()


# Create a neural network regressor model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)  # Output layer with 1 neuron for regression
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
```

Figure 5.8: Neural Networks

5.2.7 Support Vector Regression

Support Vector Regression is a supervised learning algorithm used for regression tasks. It utilizes the principles of Support Vector Machines (SVM) to find a hyperplane that best represents the relationship between the input features and the target variable. SVR is particularly effective in capturing non-linear patterns in the data.

In our implementation, the data was scaled because SVR is sensitive to the scale of the input features. Scaling ensures that all features contribute equally to the model. Additionally, we used the radial basis function (RBF) kernel, a popular choice for SVR, as it allows the model to handle complex relationships between variables.

```
# SVR requires feature scaling
scaler_X = StandardScaler()
scaler_y = StandardScaler()

X_train_scaled = scaler_X.fit_transform(X_train)
y_train_scaled = scaler_y.fit_transform(y_train.flatten().reshape(-1, 1)).flatten()

X_test_scaled = scaler_X.transform(X_test)

# SVR model
svr_model = SVR(kernel='rbf')  # Radial basis function (RBF) kernel
svr_model.fit(X_train_scaled, y_train_scaled)
```

Figure 5.9: SVR

5.2.8 K-Nearest Neighbours

K-Nearest Neighbors is a supervised learning algorithm used for both classification and regression tasks. It works by assigning a new data point's label or value based on the majority class or average of its k-nearest neighbors in the feature space. In our implementation, we tested multiple values for the parameter k, and the optimal configuration, yielding the best performance, was determined to be $k = 5$.

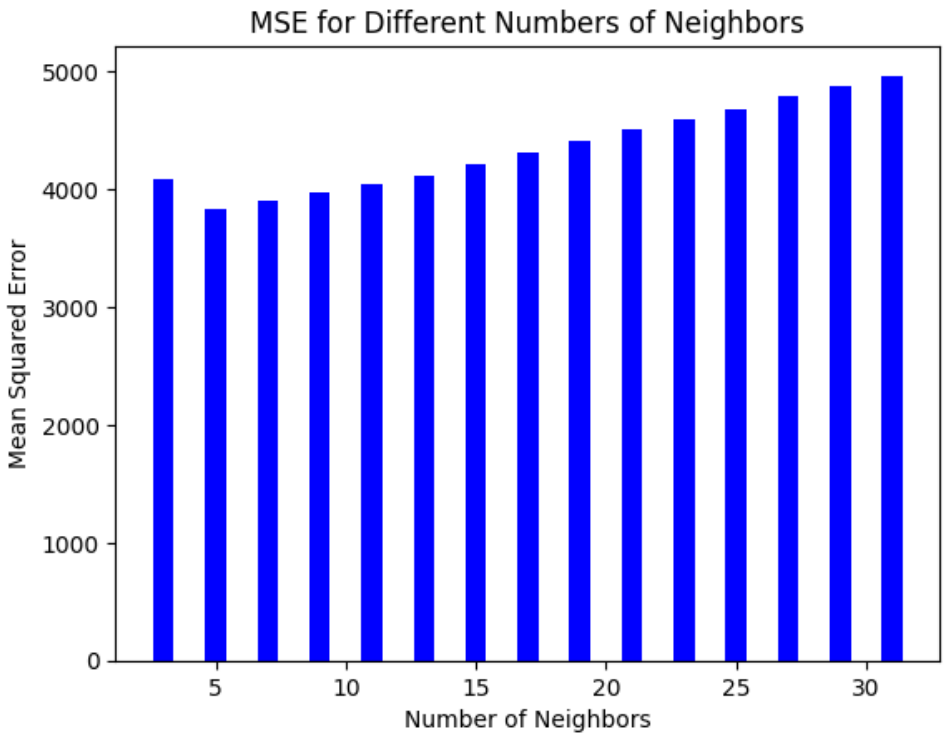

Figure 5.10: Multiple Neighbours

```
# KNN model
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

Figure 5.11

5.3 Single-Step Approach

5.3.1 Sliding Window and train/test split

In the ”single-step” sliding window approach (horizon: 1, window size: 6), a train/test split was performed, allocating the first 80% for training and the remaining 20% for testing.

Single-Step (Models)

```
#train/split

window = 6
horizon = 1

X, y = sliding_window(df['load'], window, horizon)


split_ratio = 0.8
split_index = int(split_ratio * len(df))

print(X.shape)
print(y.shape)

X_train, y_train = X[:split_index], y[:split_index]
X_test, y_test = X[split_index:], y[split_index:]
```

```
(26299, 6)
(26299,)
```

Figure 5.12: Single-Step Data preparation

5.3.2 Models forecasts

In each subsequent subsubsection, a visual comparison of the models' forecasts will be presented. The plots showcase the predictions for the last 168 hours of data, allowing for a clear comparison of each model's performance

Linear Regression

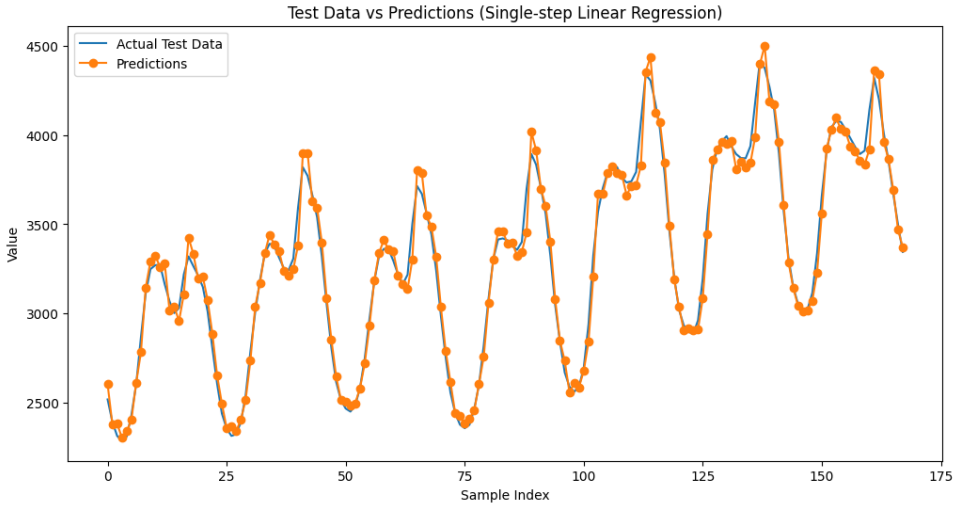

Figure 5.13: Linear Regression Forecast

Random Forest

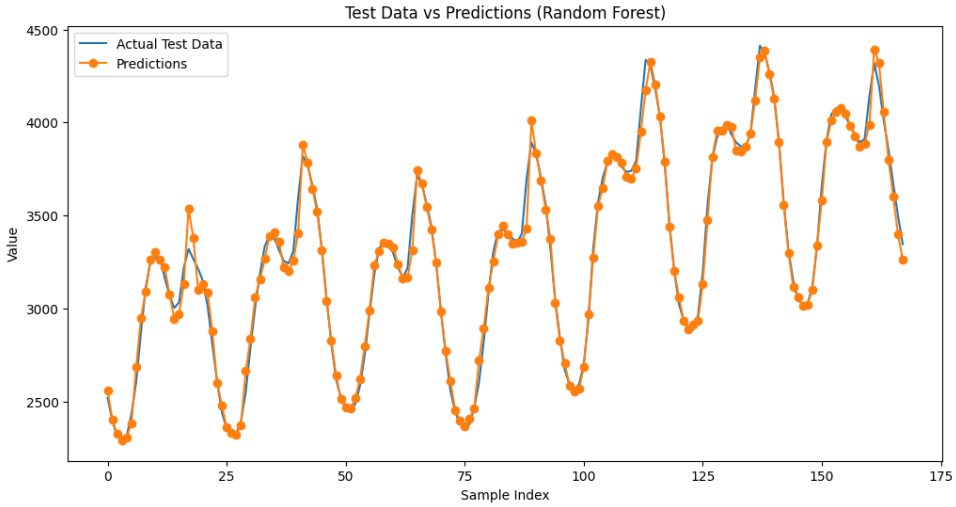

Figure 5.14: Random Forest Forecast

LightGB

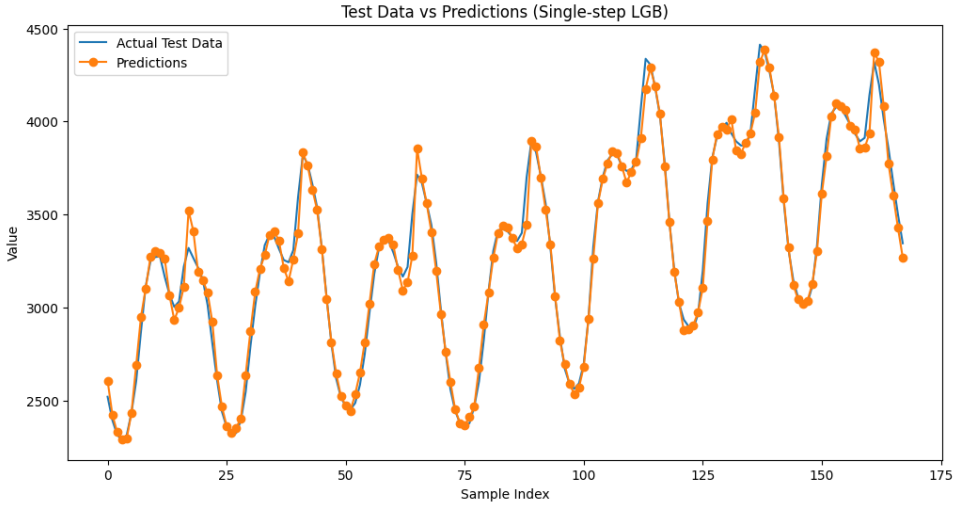

Figure 5.15: LGB Forecast

Gradient Boosting

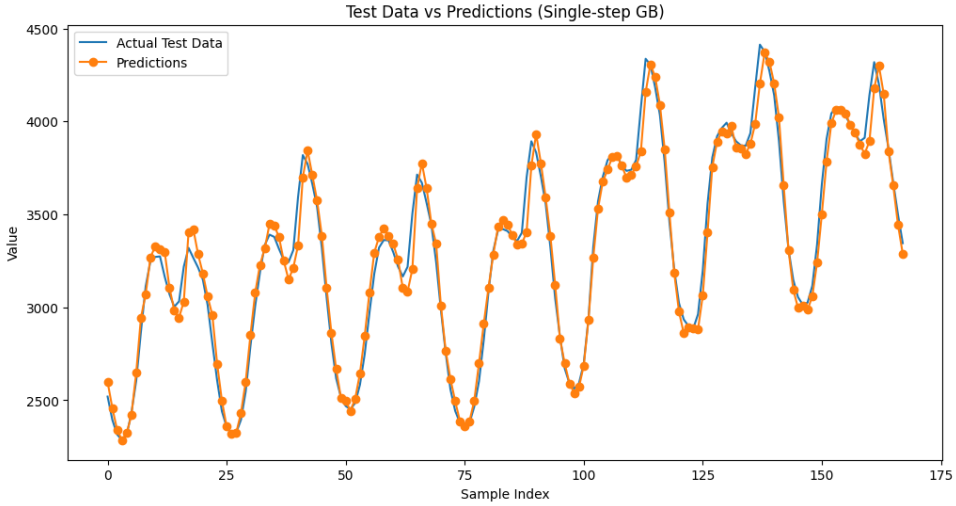

Figure 5.16: GB Forecast

XGBoost

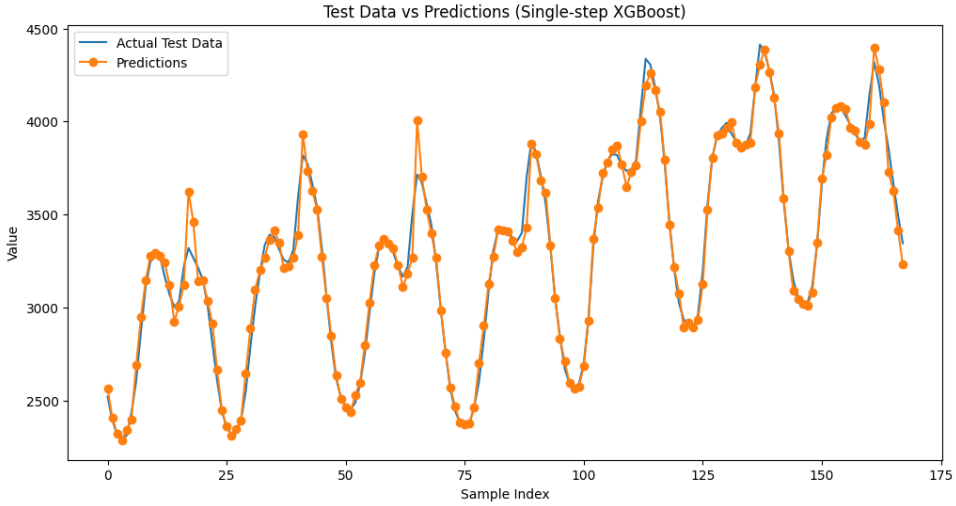

Figure 5.17: XGB Forecast

Neural Network

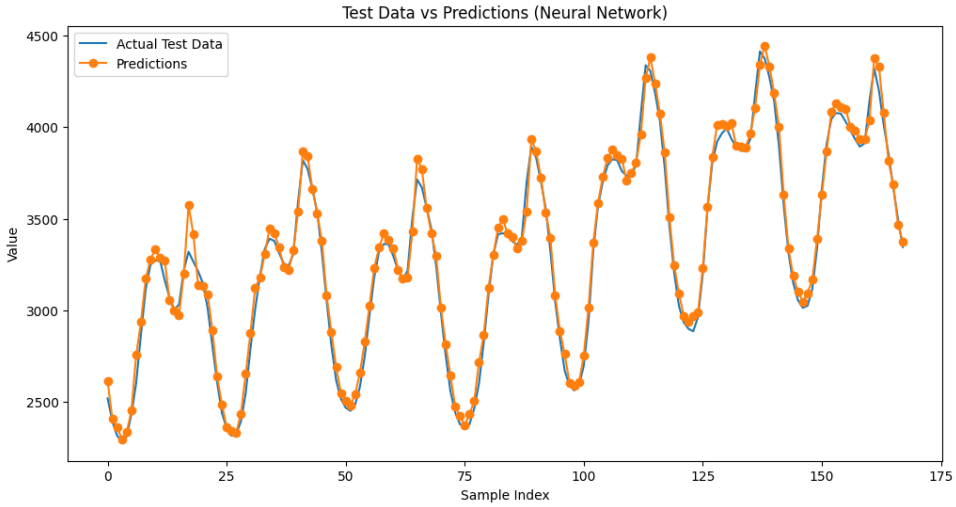

Figure 5.18: Neural Networks Forecast

Support Vector Regression

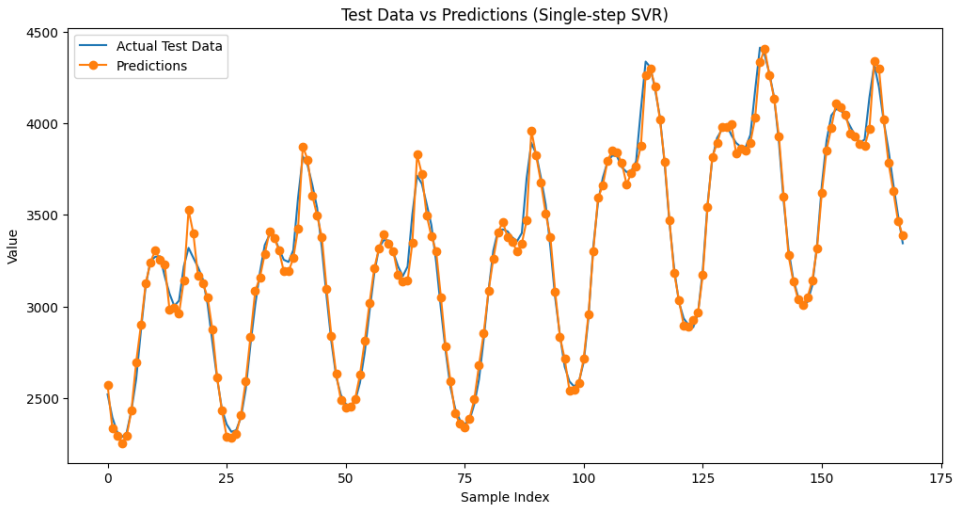

Figure 5.19: SVR Forecast

K-Nearest Neighbours

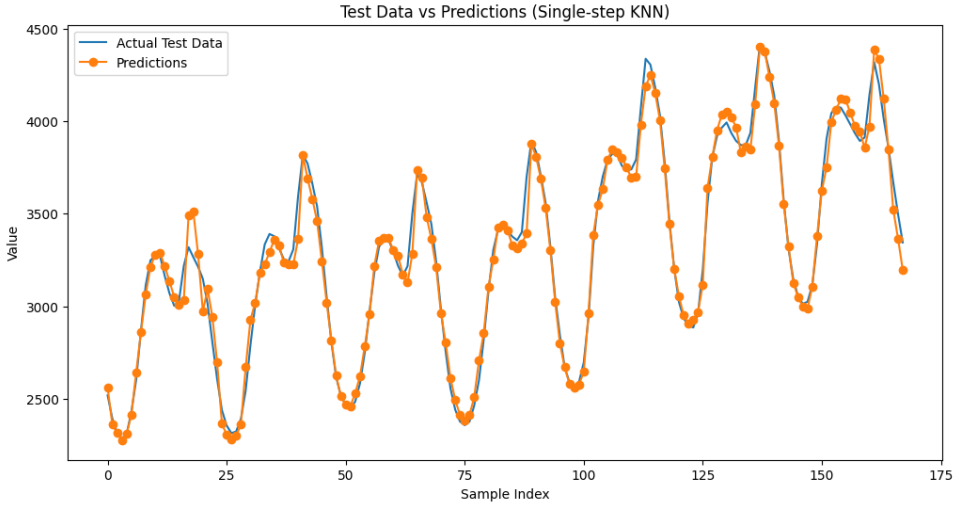

Figure 5.20: KNN Forecast

5.3.3 Model evaluation

We will compare and evaluate the models based on three key metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These metrics provide comprehensive insights into the models' performance by quantifying the accuracy and precision of their predictions.

Mean Squared Error

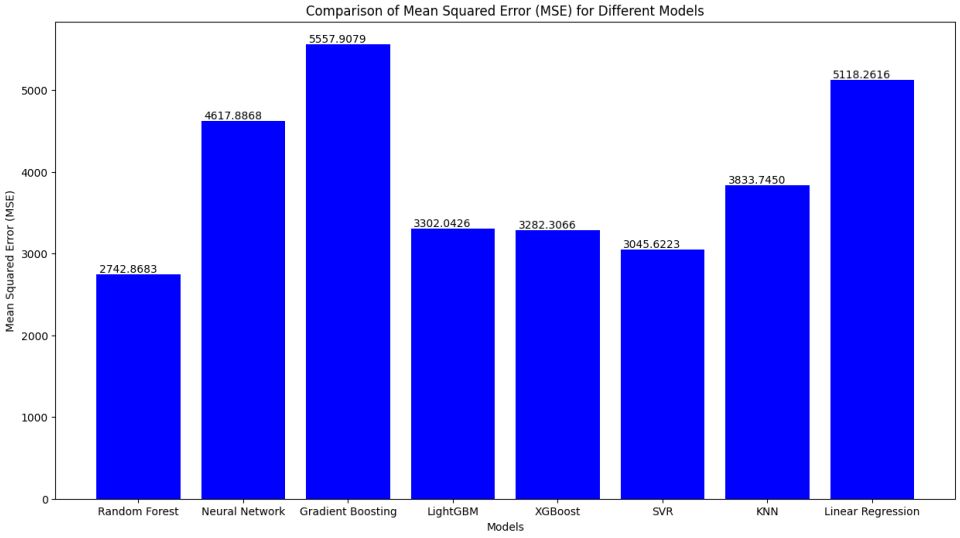

Figure 5.21: Single-Step MSE

Mean Absolute Error

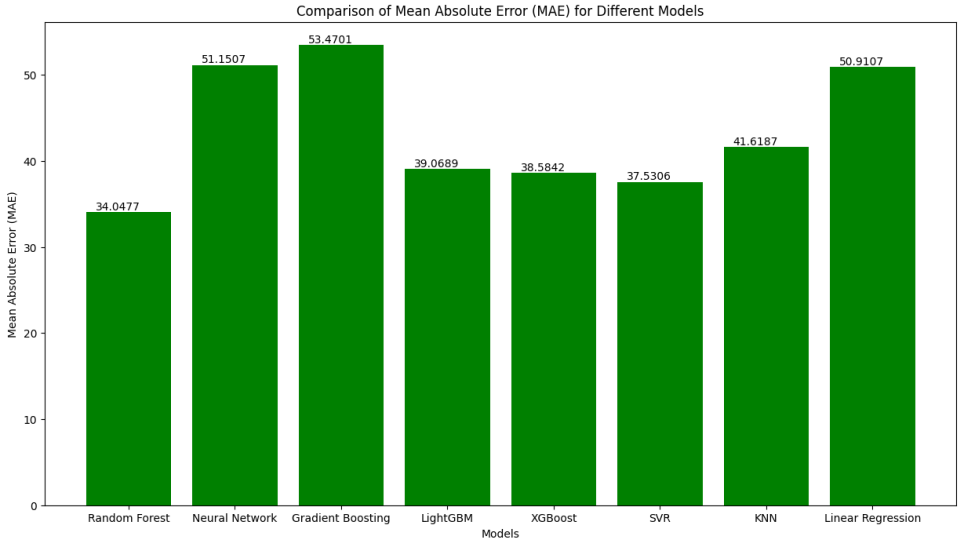

Figure 5.22: Single-Step MAE

Root mean square error

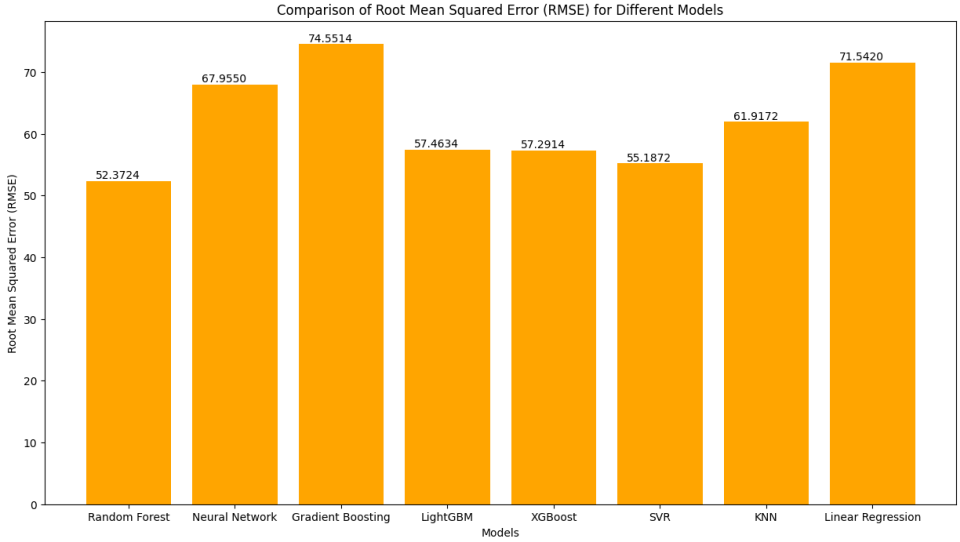

Figure 5.23: Single-Step RMSE

Conclusion

In conclusion, after evaluating the machine learning models using Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), the following observations can be made:

1. **Random Forest** consistently outperforms other models across all metrics, showcasing its robust predictive capability and effectiveness in minimizing errors.

2. **SVR** and **XGBoost** exhibit competitive performance, demonstrating relatively low values for MSE, MAE, and RMSE. These models showcase versatility and accuracy in capturing complex patterns within the data.

3. **LightGBM** and **KNN** deliver moderate performance, providing reliable predictions with a balanced trade-off between accuracy and computational efficiency.

4. **Neural Network** and **Linear Regression** show higher values across all metrics, suggesting a comparatively lower accuracy in predicting the target variable. Further fine-tuning or exploration of more complex architectures may be needed for improvement.

In summary, the Random Forest model stands out as the preferred choice for accurate and reliable predictions in this regression task, followed closely by SVR and XGBoost. The decision ultimately depends on the specific requirements, interpretability, and computational considerations for the given application.

5.4 Multi-Step Approach

5.4.1 Sliding Window and train/test split

In the "multi-step" sliding window approach (horizon: 3, window size: 6), a train/test split was performed, allocating the first 80% for training and the remaining 20% for testing.

```
#train/split

window = 6
horizon = 3

X, y = sliding_window(df['load'], window, horizon)

split_ratio = 0.8
split_index = int(split_ratio * len(df))

print(X.shape)
print(y.shape)

X_train, y_train = X[:split_index], y[:split_index]
X_test, y_test = X[split_index:], y[split_index:]
```

```
(26297, 6)
(26297,)
```

Figure 5.24: Multi-Step Data preparation

5.4.2 Models forecasts

In each subsequent subsubsection, a visual comparison of the models' forecasts will be presented. The plots showcase the predictions for the last 168 hours of data, allowing for a clear comparison of each model's performance

Linear Regression

Figure 5.25: Linear Regression Forecast

Random Forest

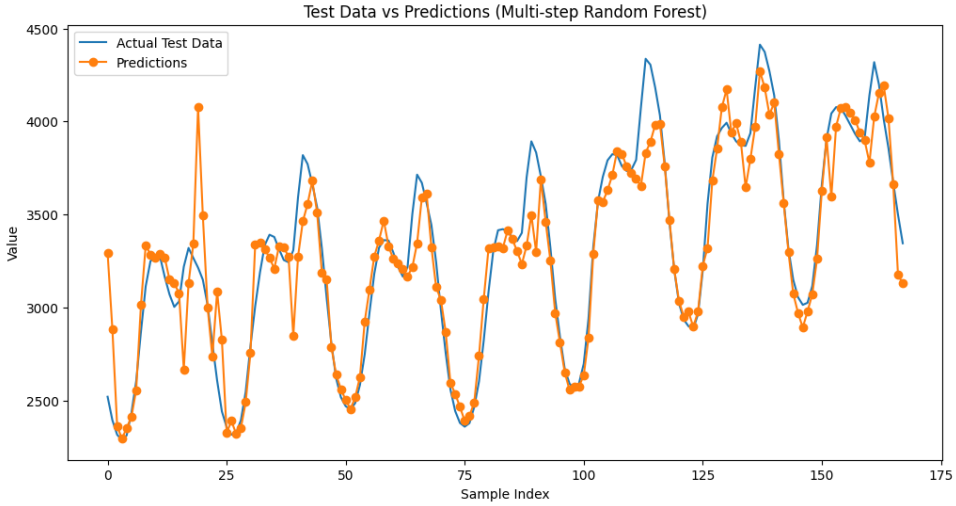

Figure 5.26: Random Forest Forecast

LightGB

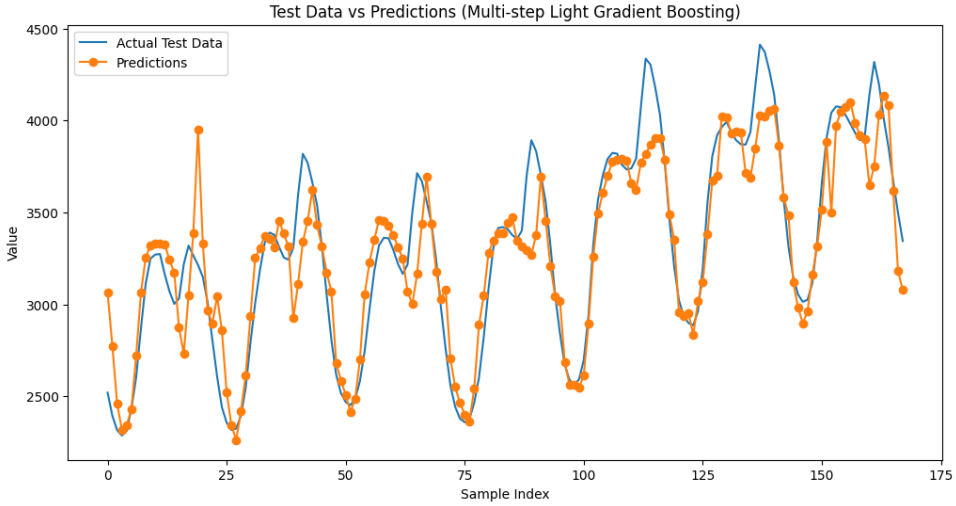

Figure 5.27: LGB Forecast

Gradient Boosting

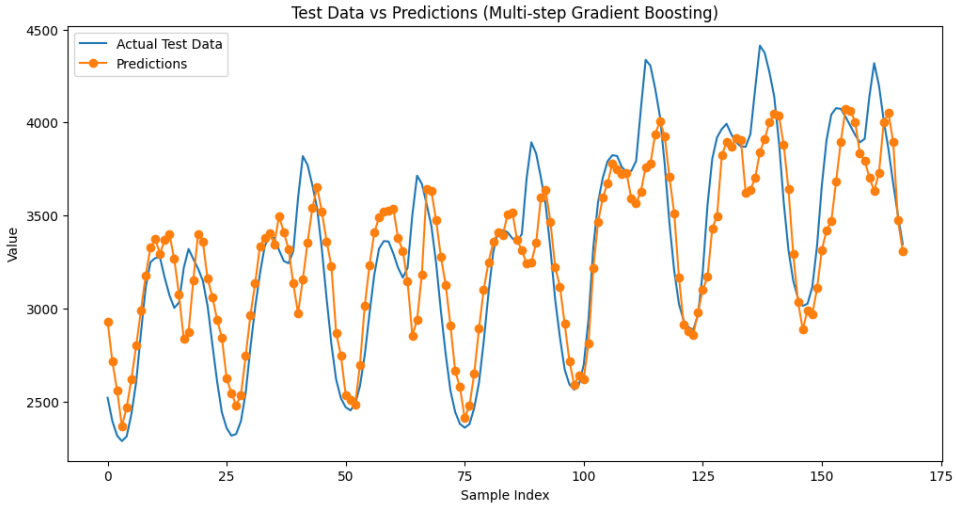

Figure 5.28: GB Forecast

XGBoost

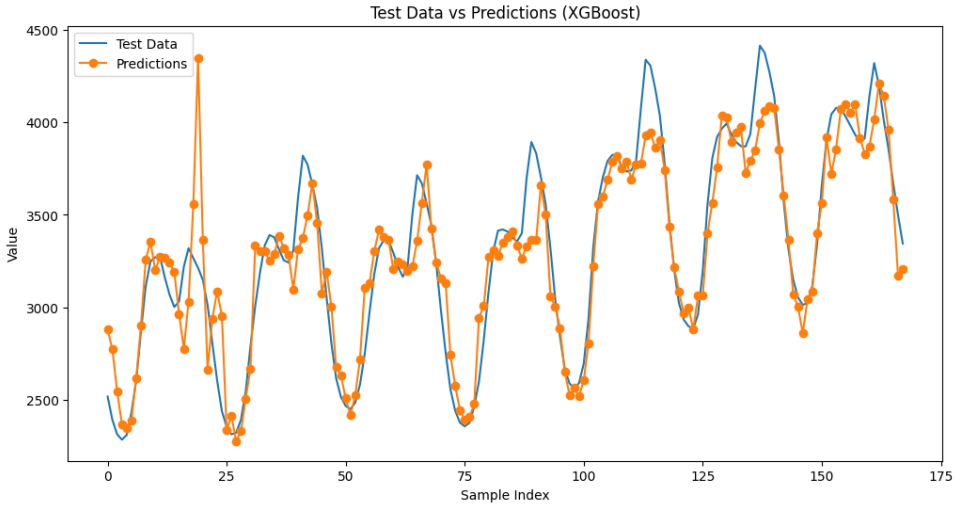

Figure 5.29: XGB Forecast

Neural Network

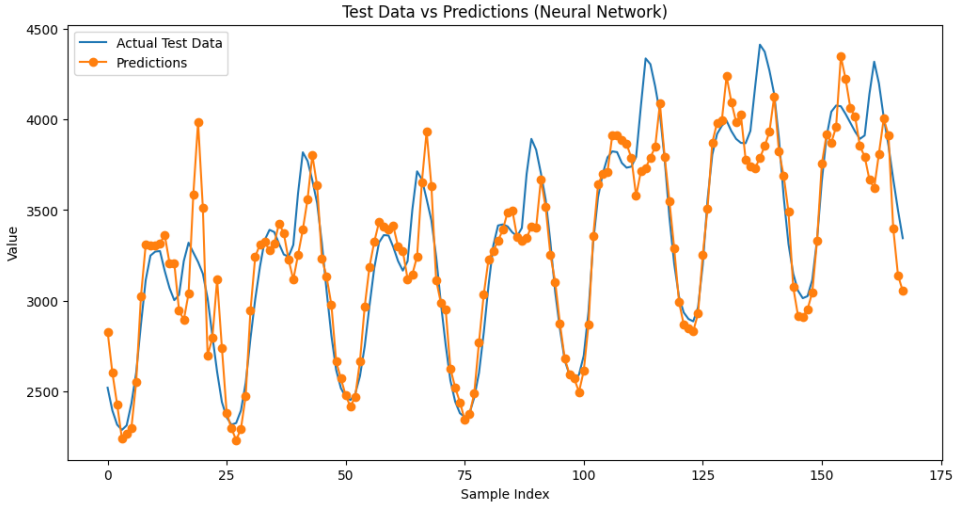

Figure 5.30: Neural Networks Forecast

Support Vector Regression

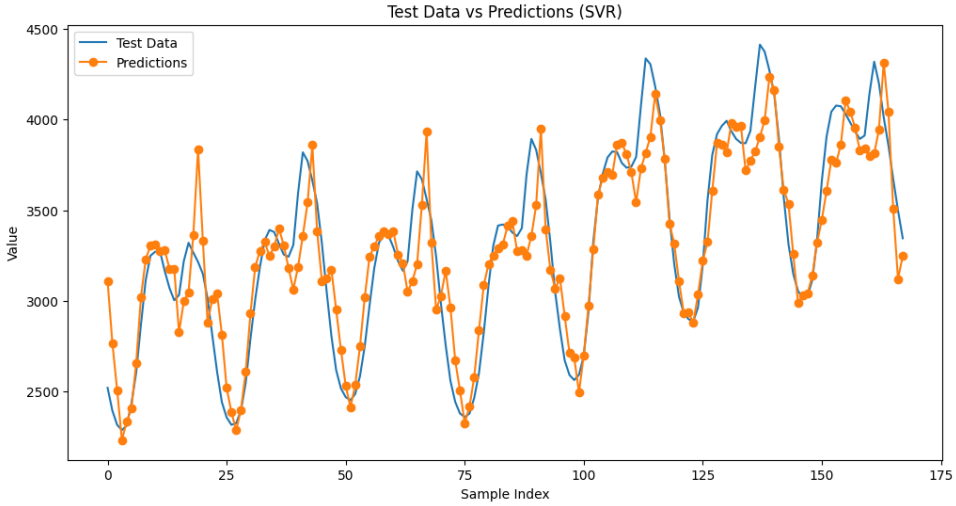

Figure 5.31: SVR Forecast

K-Nearest Neighbours

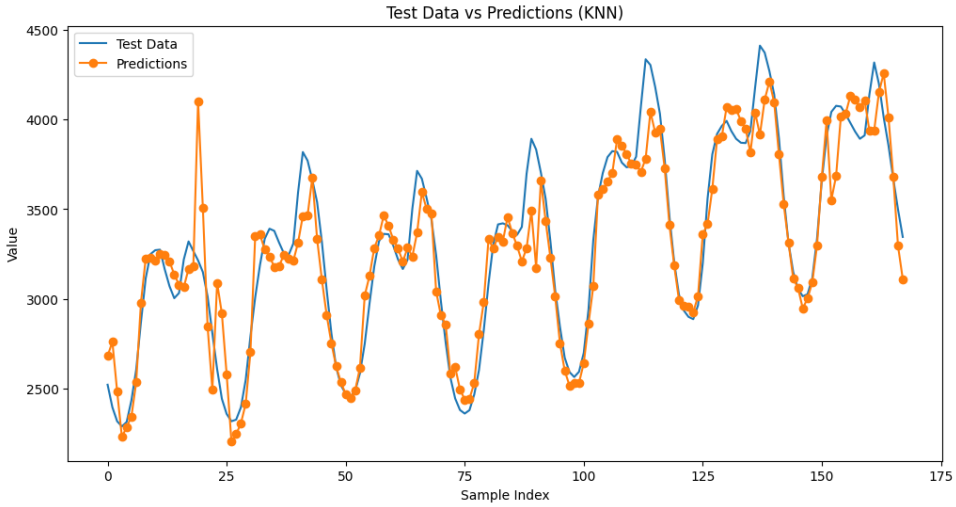

Figure 5.32: KNN Forecast

5.4.3 Model evaluation

We will compare and evaluate the models based on three key metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These metrics provide comprehensive insights into the models' performance by quantifying the accuracy and precision of their predictions.

Mean Squared Error

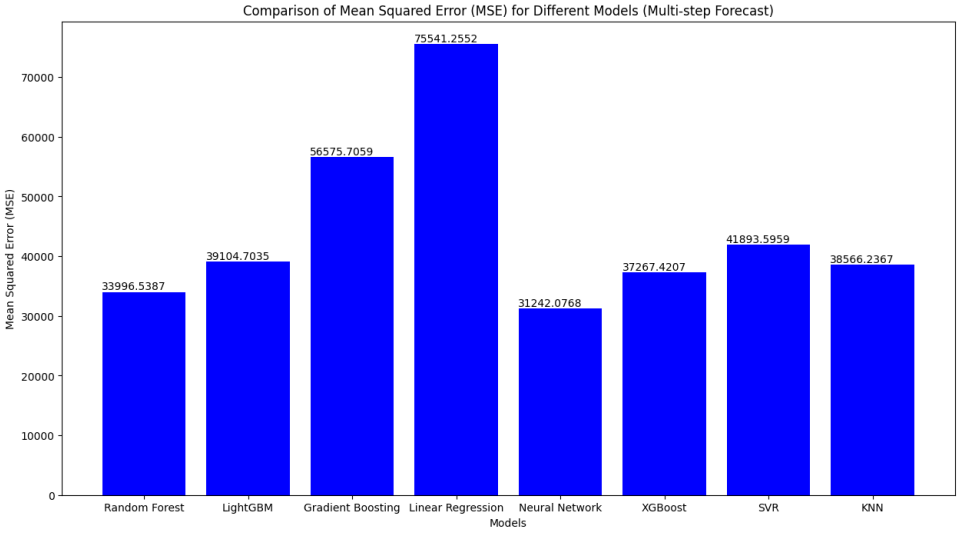

Figure 5.33: Multi-Step MSE

Mean Absolute Error

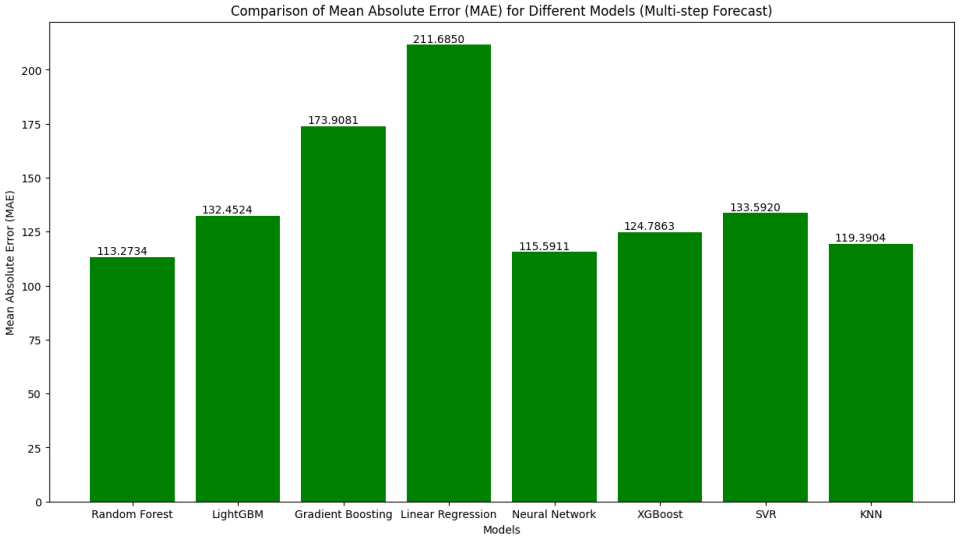

Figure 5.34: Multi-Step MAE

Root mean square error

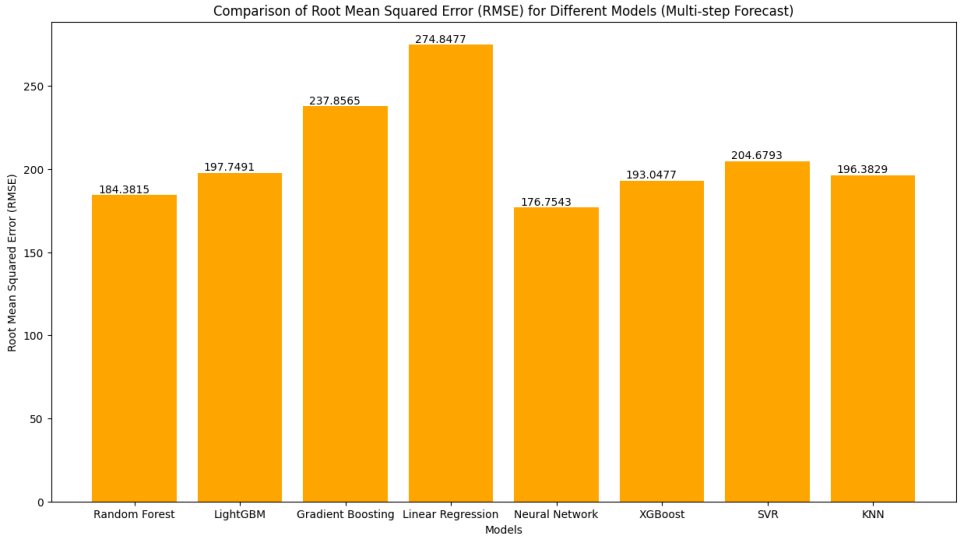

Figure 5.35: Multi-Step RMSE

Conclusion

After a thorough evaluation of various machine learning models for the multi-step forecasting task, the following conclusions can be drawn:

- **Random Forest** emerges as a robust performer, exhibiting lower Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) compared to other models. It demonstrates a high level of accuracy in predicting future values.
- **Neural Network** also stands out with competitive performance, showcasing relatively lower errors across all metrics. Its ability to capture complex patterns in the data makes it a compelling choice.
- **XGBoost** and **LightGBM** deliver moderate performance, offering a balance between accuracy and computational efficiency. They present viable alternatives with reasonable predictive capabilities.
- **Gradient Boosting**, **SVR**, and **KNN** exhibit varying degrees of performance, with higher errors compared to the top-performing models. Consideration should be given to their specific strengths and weaknesses in the context of the application.
- **Linear Regression**, while a simple and interpretable model, shows higher errors, suggesting limitations in capturing the complexity of the forecasting task.

In summary, the choice of the model depends on the specific requirements, interpretability, and computational considerations. For accurate and reliable multi-step forecasting, Random Forest and Neural Network emerge as promising candidates.

Chapter 6

Deep Learning Models

In this chapter, we explore the application of deep learning models, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), for our forecasting task. The investigation covers various scenarios, including single-step forecasting, multi-step forecasting, single-step forecasting with multivariate data (specifically temperature), and multi-step forecasting with multivariate inputs. These advanced neural network architectures are designed to capture intricate patterns in time series data, and our analysis aims to evaluate their effectiveness in improving forecasting accuracy and precision.

6.1 Data Preparation

The data underwent a crucial transformation into a format represented as X (3 dimensions) and Y (2 dimensions), as illustrated in the accompanying image. This transformation was essential to enable effective utilization of deep learning models in subsequent stages.

Figure 6.1: DL data Transformation example

In this phase, a crucial step involved splitting the dataset into training, validation, and test sets to facilitate robust model evaluation. The data was partitioned as follows: - 70% of the data was allocated for training. - The subsequent 10% (from 70% to 80%) was reserved for validation. - The remaining 20% (from 80% to 100%) constituted the test set.

Additionally, to enhance the models' stability and convergence, a MinMax scaler was applied to scale the data. This normalization process proved particularly valuable for models sensitive to variations in input feature magnitudes.

6.2 Models Used

In this section, we introduce two powerful recurrent neural network (RNN) architectures employed for the forecasting task:

6.2.1 Long Short-Term Memory (LSTM)

LSTM is a specialized type of RNN designed to overcome the limitations of traditional RNNs in capturing long-range dependencies. It incorporates memory cells and gating mechanisms, allowing it to selectively

retain or discard information over extended sequences. This makes LSTMs particularly effective in handling temporal dependencies and intricate patterns within time series data.

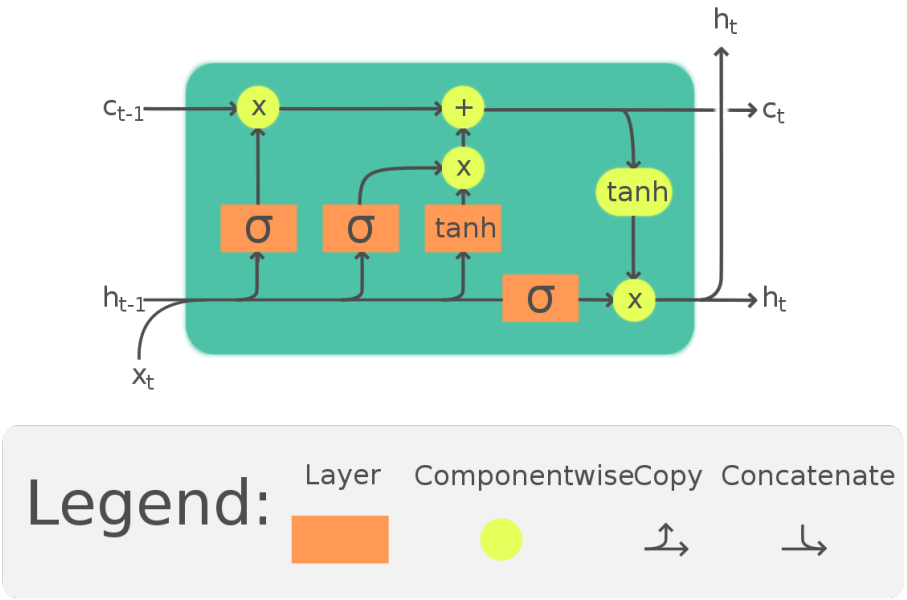

Figure 6.2: LSTM architecture

6.2.2 Gated Recurrent Unit (GRU)

Similar to LSTM, GRU is another variant of RNN that addresses some of the challenges in modeling long-term dependencies. GRU utilizes gating mechanisms to regulate the flow of information within the network. It offers a more streamlined architecture compared to LSTM, making it computationally efficient while maintaining strong performance in capturing temporal patterns.

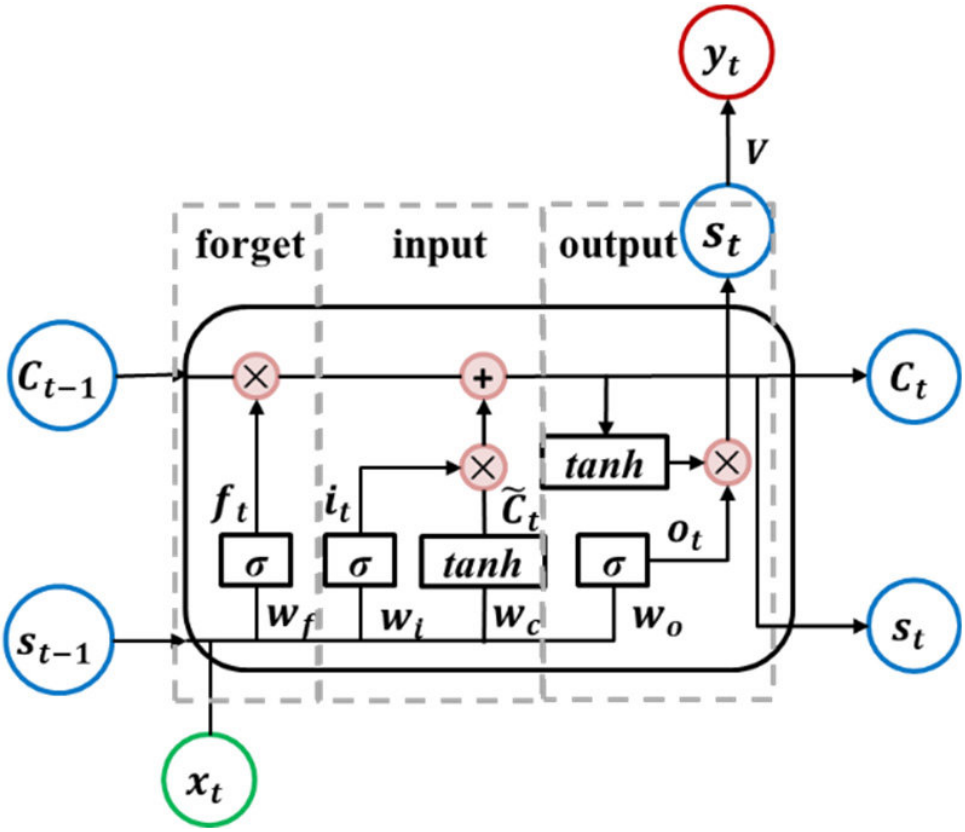

Figure 6.3: GRU architecture

These two architectures serve as the foundation for our deep learning models, contributing to their ability to learn and forecast complex temporal patterns in the time series data.

6.3 Forecasting approach

6.3.1 Single-step

For the single-step forecasting approach, a critical step in data preparation involved transforming the time series data into a tabular format using a sliding window. The window size was set to 6, with a horizon of 1. This transformation allowed the model to learn from historical observations and make predictions one step ahead, facilitating the training process for effective single-step forecasting.

In the following subsubsection, forecasts from each model will be shown.

Forecast results

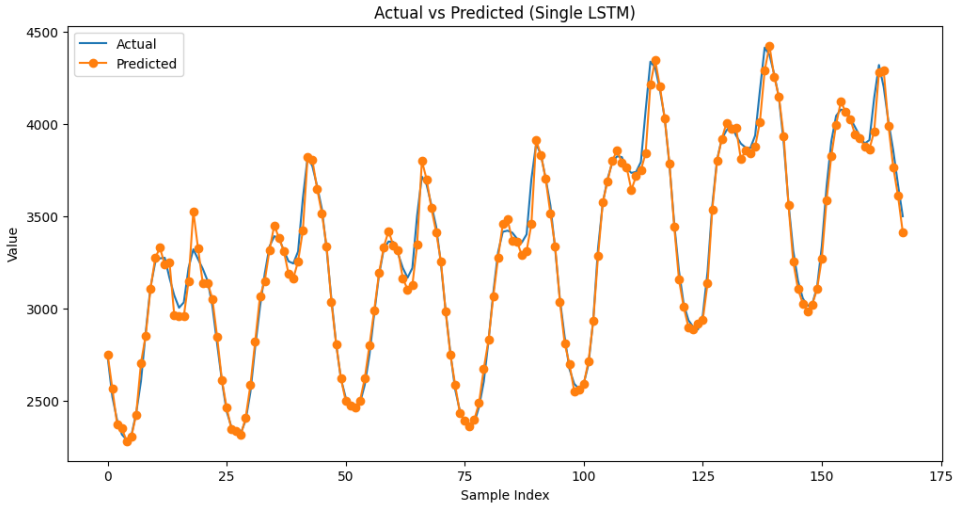

Figure 6.4: Single-Step LSTM Forecasts

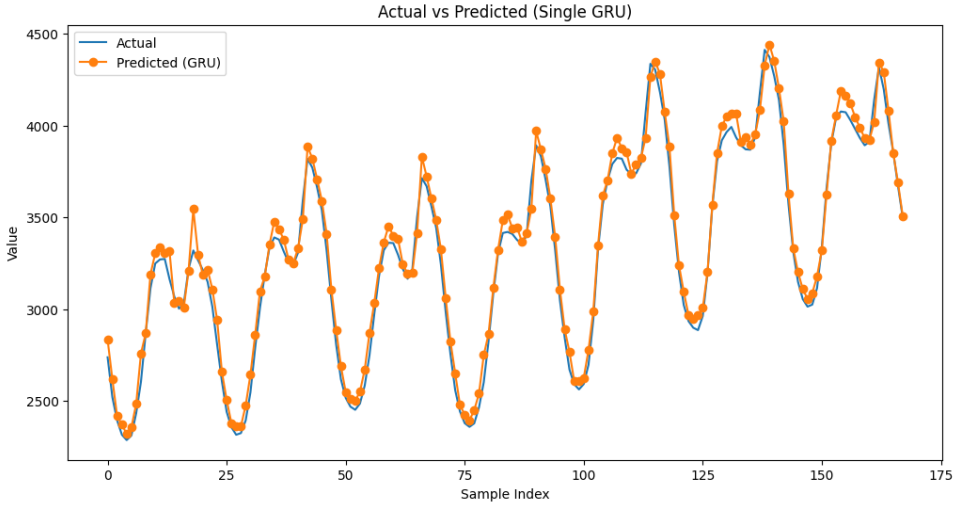

Figure 6.5: Single-Step Gru Forecasts

6.3.2 Multi-step

In contrast to the single-step forecasting approach, the data preparation for multi-step forecasting involved utilizing a sliding window with a larger window size of 8 and an extended horizon of 3. This adjustment in the window size and horizon aimed to provide the model with a more comprehensive historical context, enabling it to forecast multiple future time steps simultaneously. The larger window size enhances the model's ability to capture intricate patterns and dependencies in the time series data, contributing to more effective multi-step forecasting.

In the following subsubsection, forecasts from each model will be shown.

Forecast results

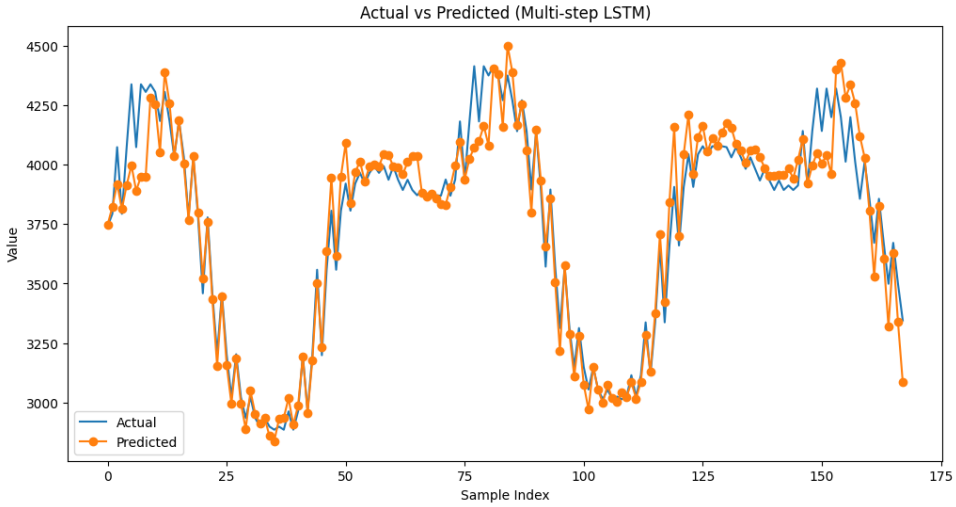

Figure 6.6: Multi-Step LSTM Forecasts

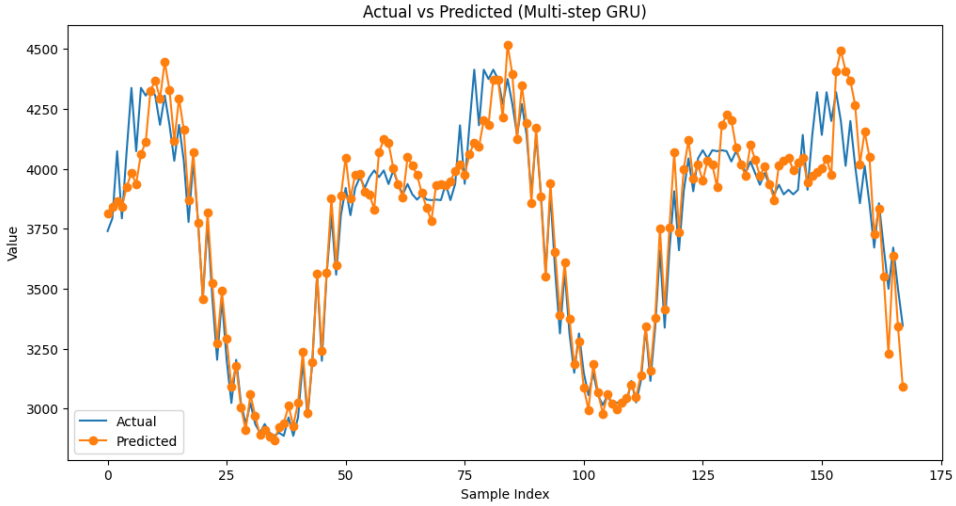

Figure 6.7: Multi-Step Gru Forecasts

6.3.3 Single-step Multivariate

Similar to the single-step univariate approach, the single-step multivariate forecasting involved transforming the time series data into a tabular format using a sliding window. The window size was maintained at 6, with a horizon of 1. This transformation allowed the model to learn from historical observations and make predictions one step ahead. Notably, in this multivariate setting, temperature was incorporated as an additional feature alongside the time series data. The inclusion of temperature as an extra feature aimed to enhance the model's understanding of the contextual factors influencing the forecasting task.

In the following subsubsection, forecasts from each model will be shown.

Forecast results

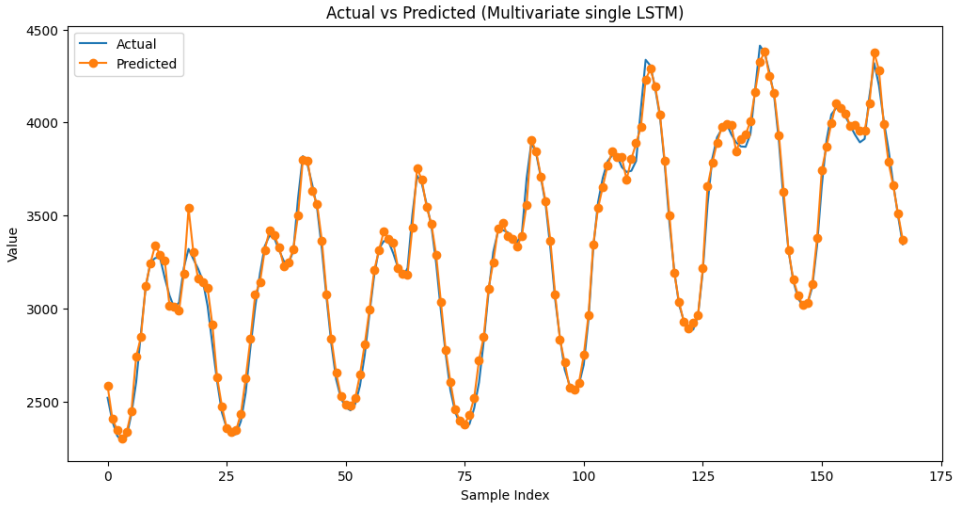

Figure 6.8: Single-Step Multivariate LSTM Forecasts

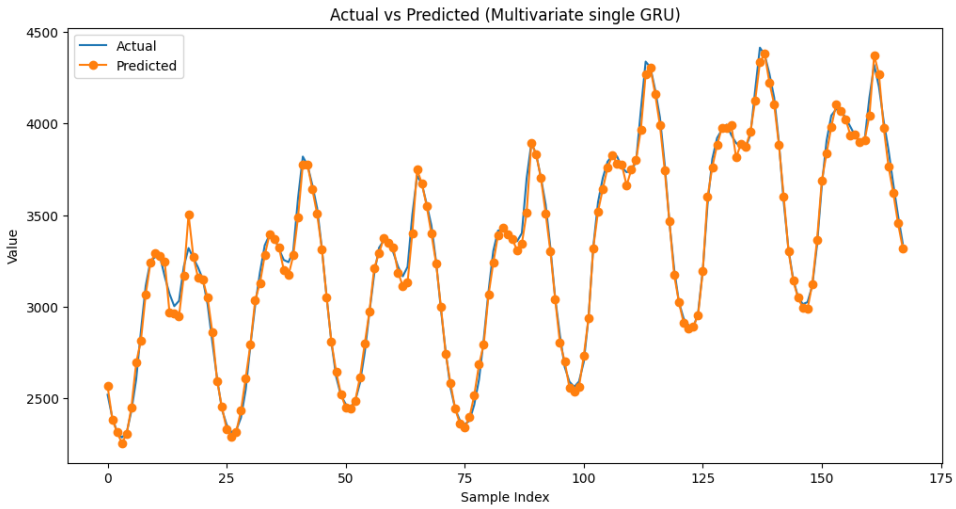

Figure 6.9: Single-Step Multivariate Gru Forecasts

6.3.4 Multi-step Multivariate

Differing from the single-step multivariate approach, the data preparation for multi-step forecasting involved utilizing a sliding window with a larger window size set to 8 and an extended horizon of 3. This adjustment aimed to afford the model a broader historical context, empowering it to forecast multiple future time steps concurrently. The increased window size enhances the model's capacity to discern intricate patterns and dependencies within the time series data. Notably, in this multivariate setting, temperature was incorporated as an additional feature, augmenting the model's understanding of contextual factors for more comprehensive multi-step forecasting.

In the following subsubsection, forecasts from each model will be shown.

Forecast results

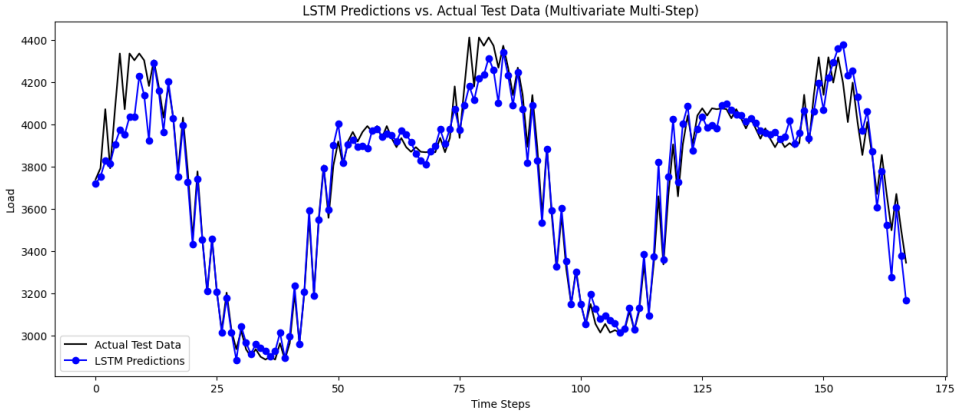

Figure 6.10: Multi-Step LSTM Multivariate Forecasts

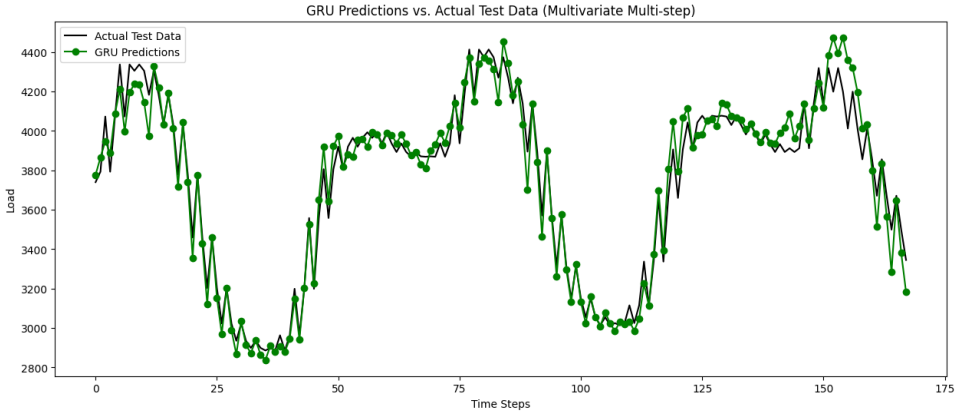

Figure 6.11: Multi-Step Gru Multivariate Forecasts

6.4 Model Evaluation

We will compare and evaluate the models based on three key metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). These metrics provide comprehensive insights into the models' performance by quantifying the accuracy and precision of their predictions.

6.4.1 Mean square error

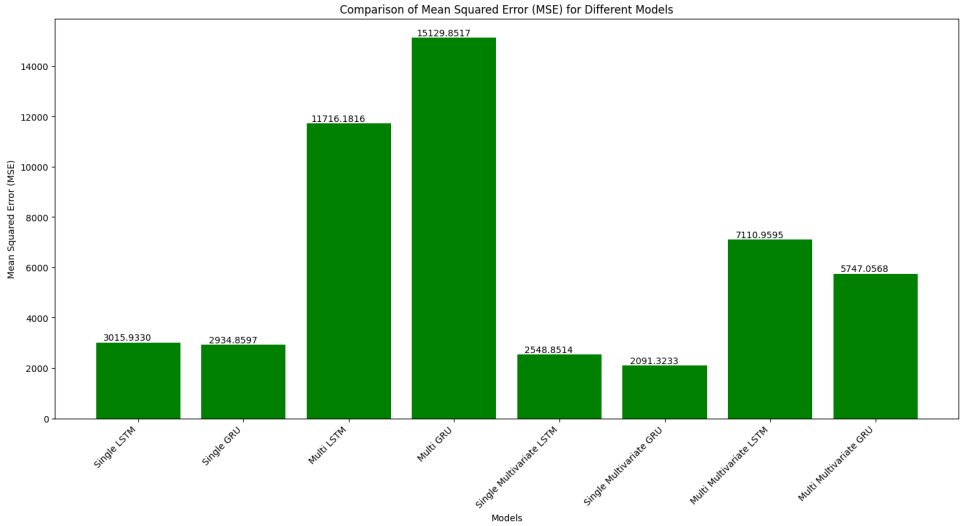

Figure 6.12: Deep Learning MSE Comparison

6.4.2 Mean absolute error

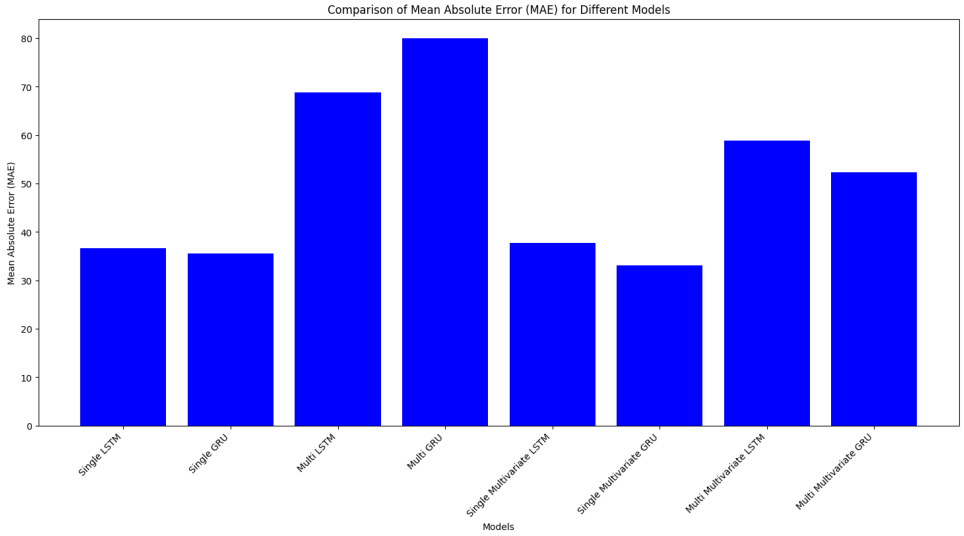

Figure 6.13: Deep Learning MAE Comparison

6.4.3 Root mean square error

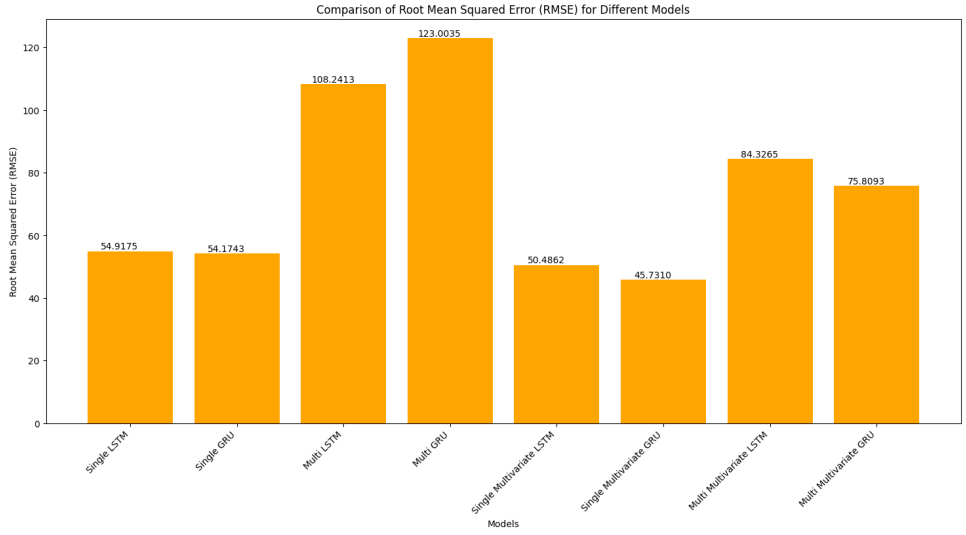

Figure 6.14: Deep Learning RMSE Comparison

6.4.4 Model's performance overview

The performance evaluation of various deep learning models for different forecasting scenarios yielded insightful results. The following conclusions can be drawn from the Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE) metrics:

Single-Step Univariate Models

Both the Single LSTM and Single GRU models demonstrated competitive performance, showcasing low errors across all metrics. The slight advantage of Single GRU in terms of lower MSE and MAE suggests its effectiveness in capturing single-step univariate patterns.

Multi-Step Univariate Models

In the multi-step univariate forecasting task, Single LSTM outperformed Multi LSTM, while Multi GRU exhibited higher errors. This indicates the challenge of capturing complex dependencies for multiple future time steps, with Single LSTM offering a more accurate prediction.

Single-Step Multivariate Models

The Single Multivariate LSTM and Single Multivariate GRU models, incorporating temperature as an extra feature, displayed impressive performance with lower errors. Single Multivariate GRU, in particular, showcased superiority in terms of lower MSE and RMSE.

Multi-Step Multivariate Models

In the multi-step multivariate context, both Multi Multivariate LSTM and Multi Multivariate GRU models demonstrated competitive performance. Multi Multivariate LSTM exhibited slightly higher accuracy, showcasing its capability to leverage additional features for improved multi-step forecasting.

6.4.5 Conclusion

In summary, the choice of the optimal model depends on the specific forecasting task and the nature of the data. The Single Multivariate GRU model stands out for single-step multivariate forecasting, while Multi Multivariate LSTM shows promise for multi-step multivariate predictions. These insights provide valuable guidance for selecting the most suitable deep learning model based on the forecasting requirements.

Chapter 7

Conclusion, Constraints and Future Work

7.1 Conclusion

This study undertook a comprehensive exploration of diverse forecasting models, spanning statistical, baseline, machine learning (ML), and deep learning (DL) approaches. The comparative analysis revealed that ML and DL models consistently outperformed statistical models, with DL exhibiting superiority in specific scenarios. Notably, the incorporation of temperature as an additional feature in DL models showcased its impact on enhancing forecasting accuracy.

7.2 Constraints

While our investigation yielded valuable insights, certain constraints were encountered during the experimentation process. Notably, the computational demands of the SARIMA and SARIMAX models posed challenges in parameter optimization due to their extended runtime. This limitation influenced the ability to efficiently fine-tune these models for optimal performance.

7.3 Future Work

To propel the field of time series forecasting forward, future research could incorporate feature selection techniques to refine the model inputs. Exploring additional granularity levels in date-related features holds promise, as electricity consumption patterns are likely to vary significantly across different times of the year. By fine-tuning feature selection, models can potentially capture more nuanced temporal dependencies and enhance their predictive capabilities.

The integration of advanced feature engineering methods, coupled with a deeper investigation into the impact of date-related features, could provide valuable insights for optimizing forecasting models. This avenue of research aims to unlock further potential in accurately predicting electricity consumption, contributing to the refinement and applicability of time series forecasting methodologies in real-world scenarios.