TRACKS THE ELECTRIC POWER CONSUMPTION OF A HOUSEHOLD

André Teixeira
1190384

Gonçalo Ribeiro
1220493

MEI - ISEP - MINDD
Dezembro 2023

Index

Table of Figures

1. Introduction

For this project, we'll use a dataset that tracks the electric power consumption of a household. Forecasting energy consumption helps ensure the grid provides enough energy for households, allowing energy companies to plan load, produce enough during peak times, and avoid excess electricity, preventing grid imbalance and disconnection risks.

1.1 Dataset Overview

The dataset to explore consists of three years of hourly electricity load and temperature, between 2012 and 2014. It contains the following columns:

Date – the actual date the measurement was recorded;
Hout – the actual hour the measurement was recorded;
Load value – the actual consumption measured in kWh (kilowatt-hour);
Temperature – the temperature collected in degrees Celsius;

2. Data exploration and preparation

In the process of preparing the dataset for analysis, several essential steps were taken to ensure the data's quality, consistency, and readiness for further exploration.

2.1 Checking for missing values and duplicates

Next, a thorough check for missing values and duplicates was conducted. Missing values could introduce bias or affect the accuracy of analyses, and duplicates might distort statistical measures. By confirming that there were no missing values or duplicates, confidence in the dataset's integrity is established.

2.2 Searching Date and Hour columns

Checks for the existence of 'Date' and 'Hour' columns in the DataFrame. If present, it performs operations to adjust 'Hour' values, combines 'Date' and 'Hour' into a new 'DateTime' column, and sets this new column as the index for time series analysis. If the columns do not exist, a warning message is printed.

2.3 Checking correlations

Produces a graphical representation of the correlation matrix, enhancing the understanding of inter-variable relationships and contributing valuable insights to the data analysis process.

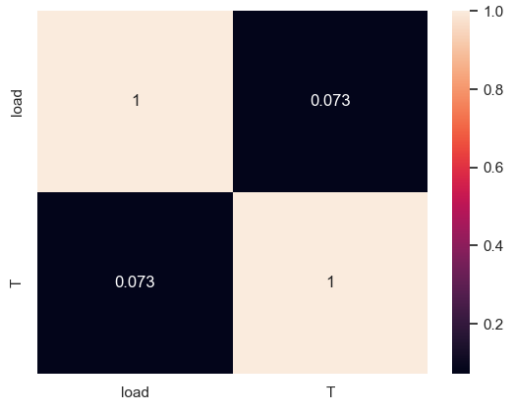

Figure 1 Checking correlations heatmap

The correlation heatmap shows a strong correlation between temperature and consumption.

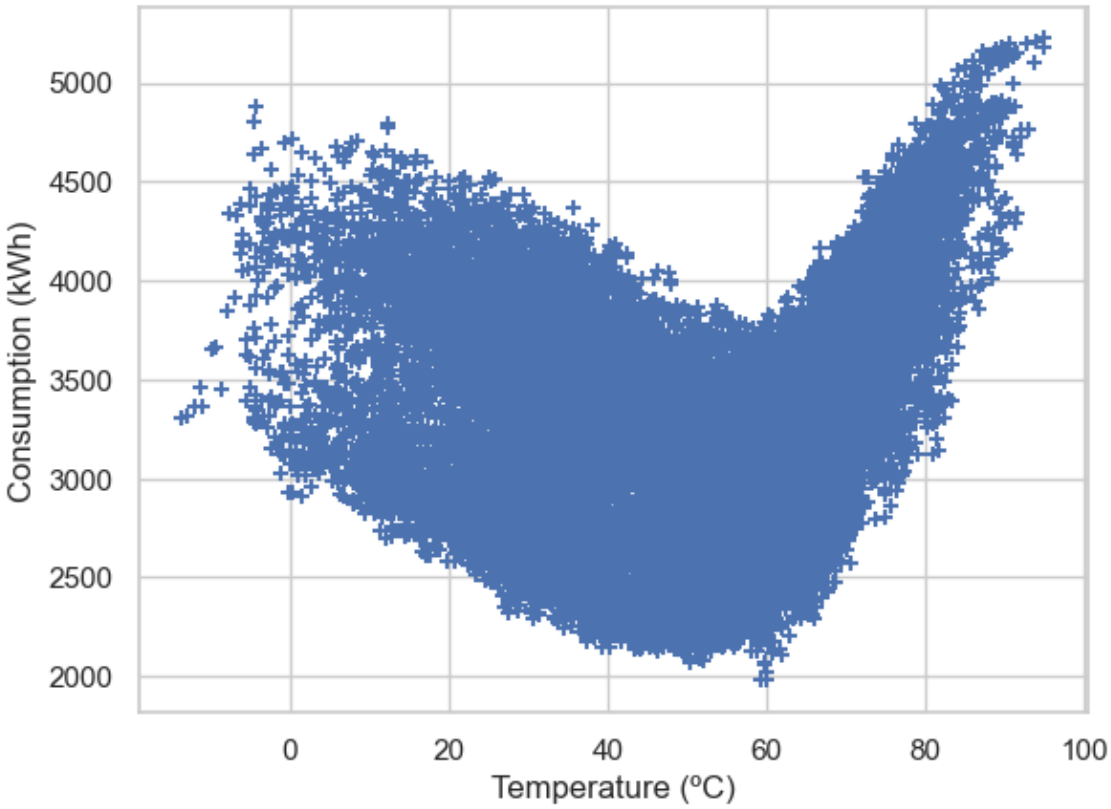

Figure 2 Checking correlations scatter plots

It's notable an increase of energy consumption when the temperatures are lower or too high.

3. Time Series Analysis

3.1 Components of a Time Series

The primary purpose to visually inspect and analyze the decomposed components of the time series. By decomposing the time series into trend, seasonal, and residual components, we can gain insights into the underlying patterns, long-term trends, and seasonality present in the data. This is particularly useful in time series analysis for understanding the contributing factors to the observed patterns and making more informed predictions or assessments.

In summary, the code performs seasonal decomposition on the 'load' time series, assumes an additive model, specifies a daily seasonality (period of 24 hours), and then plots and displays the decomposed components for visual analysis.

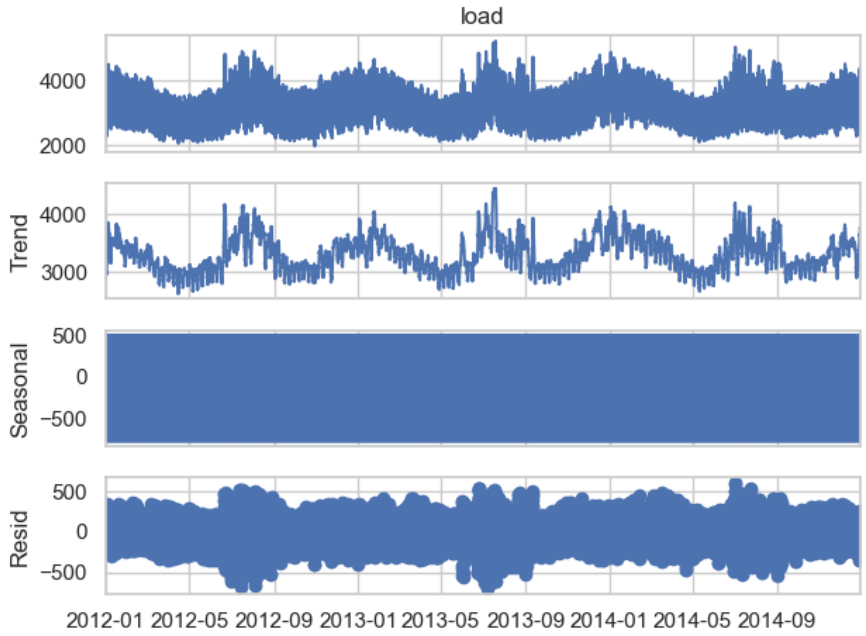

Figure 3 Components of time series

3.2 ADCF Test to check if the series is stationary

We use the ADF (Augmented Dickey-Fuller) test to evaluate the stationarity of the time series represented by the 'load' column of the DataFrame.

```
ADF Statistic: -10.406281746483792
p-value: 1.8532219576247764e-18
```

Figure 4 Augmented Dickey-Fuller (ADF) results

- ADF statistic, which is a negative number we can reject the null hypothesis
- P-value is less than 0.05, we can also reject the null hypothesis and say the series is stationary
- The Dataset is stationary

3.3 Autocorrelation function - ACF

The graphical representation of the Autocorrelation Function (ACF) serves as a fundamental tool in the analysis of time series data. It provides insights into the correlation between each value in the time series and its preceding values, commonly referred to as lags.

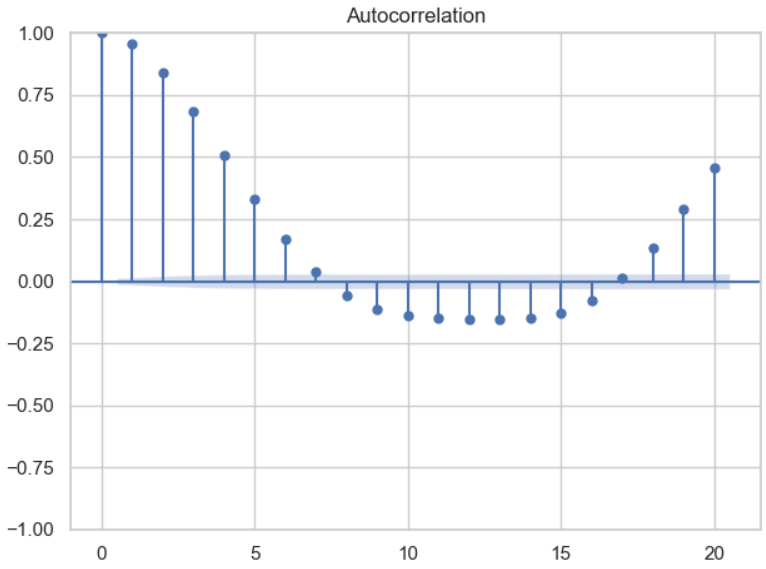

Figure 5 Autocorrelation function – ACF

3.4 Partial autocorrelation function - PACF

The graphical representation of the Autocorrelation Function (ACF) serves as a fundamental tool in the analysis of time series data. It provides insights into the correlation between each value in the time series and its preceding values, commonly referred to as lags.

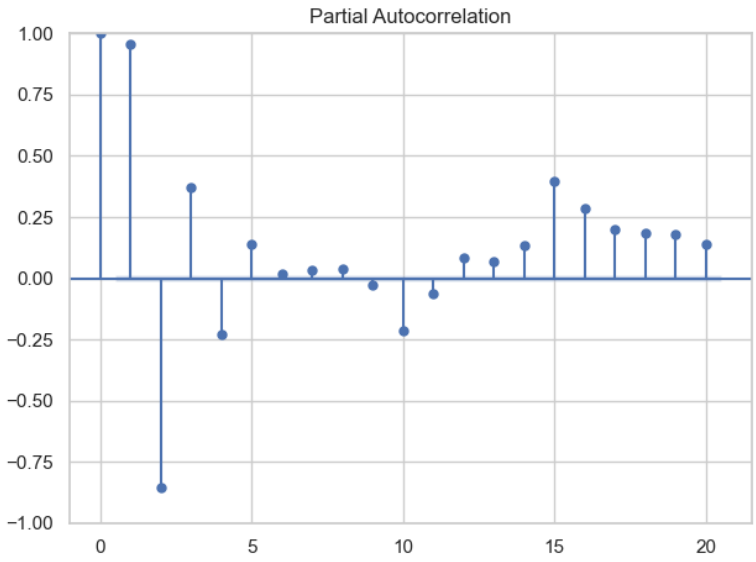

Figure 6 Partial autocorrelation fuction – PACF

There are significant coefficients after lag 0, which is not a indicator of a random walk.

4. Forecasting an Autoregressive Moving Average model ARMA(p,q)

In the context of time series analysis, the goal is to predict future values using an Autoregressive Moving Average (ARMA) model with parameters p and q. To prepare the data:

Differencing was applied to the 'load' series, creating a new series labeled df_diff. The dataset was then divided into training (train) and testing (test) subsets based on a split date of '2014-01-01'. The training set comprises 17,545 observations, and the testing set includes 8,760 observations.

Later, we generated a graph of the differenced time series (load_diff).

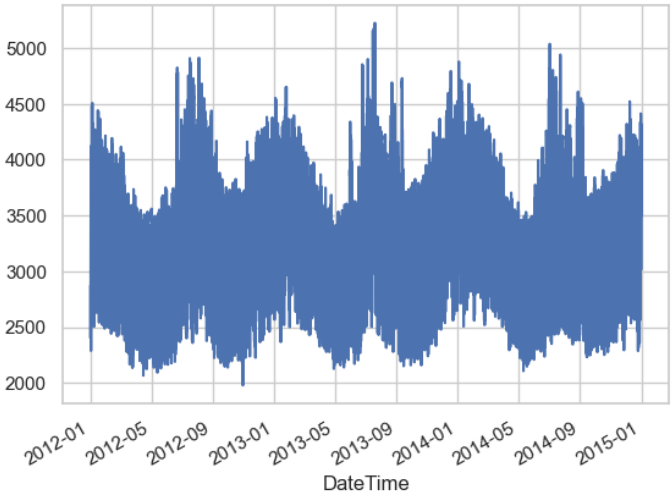

Figure 7 Load_diff

This graphical representation allows for an examination of the time series following the application of differencing. Differencing is commonly employed to achieve stationarity in the time series, a crucial step in many time series models, including ARMA.

The x-axis of the graph represents temporal indices, while the y-axis represents the values of the differenced time series. By observing the graph, one can look for indications of stationarity and identify patterns or behaviors that may influence the choice of the model.

In summary, the graph of the differenced time series provides a visual insight into the stationarity of the series after differencing, laying the groundwork for subsequent modeling.

5. Calculate Akaike information criterion (AIC)

Throughout the optimization process of an Autoregressive Moving Average (ARMA) model for forecasting, we adopted the following approach:

We defined a function called “optimize_ARMA” with the purpose of exploring various ARMA order combinations and selecting the optimal model based on the Akaike Information Criterion (AIC). This function iterates through a predefined list of order combinations, fits SARIMAX models to the training data, calculates the AIC for each model, and stores the results.

The ARMA order combinations were set as “ps” and “qs”, ranging from 0 to 3. The “order_list” was generated using the Cartesian product of “ps” and “qs”. Subsequently, the” optimize_ARMA” function was applied to fit SARIMAX models for each order combination.

The results were organized into a DataFrame named “result_df”, containing the order combinations and their corresponding AIC values. This DataFrame was sorted in ascending order of AIC, where lower AIC values indicate more suitable model fits.

```
ps range(0, 4)
qs range(0, 4)
order_list:
 [(0, 0), (0, 1), (0, 2)]

 ....
[(3, 1), (3, 2), (3, 3)]
16
```

	(p,q)	AIC
0	(3, 2)	202061.055741
1	(3, 1)	204755.625831
2	(3, 3)	204810.025405
3	(2, 3)	204817.986745
4	(2, 2)	204834.359324
5	(2, 1)	205126.701303
6	(3, 0)	205394.764274
7	(1, 3)	205401.979790
8	(1, 2)	207060.797262
9	(2, 0)	210530.287389
10	(1, 1)	213203.542950
11	(1, 0)	229763.960299
12	(0, 3)	274754.062323
13	(0, 2)	289647.379972
14	(0, 1)	310522.327095
15	(0, 0)	334505.036886

Figure 8 AIC results

In the results analysis, it was observed that the order combination (3, 2) achieved the lowest AIC, indicating a superior fit compared to other combinations. Sorting the DataFrame by AIC revealed the top three combinations as (3, 2), (3, 1), and (3, 3). Conversely, higher-order combinations, such as (0, 3), (0, 2), and (0, 1), exhibited significantly higher AIC values, suggesting less effective model fits.

6. Analize SARIMAX results

```
                               SARIMAX Results
==========================================================================
Dep. Variable:              load_diff   No. Observations:                17545
Model:               SARIMAX(3, 0, 2)   Log Likelihood           -101024.528
Date:                Sun, 31 Dec 2023   AIC                      202061.056
Time:                        17:26:53   BIC                      202107.691
Sample:                             0   HQIC                     202076.411
                                - 17545
Covariance Type:                  opg
==========================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------
ar.L1          2.6524      0.003    906.081      0.000       2.647       2.658
ar.L2         -2.4102      0.004   -606.423      0.000      -2.418      -2.402
ar.L3          0.7579      0.001    644.855      0.000       0.756       0.760
ma.L1         -0.4626      0.006    -72.768      0.000      -0.475      -0.450
ma.L2         -0.5233      0.006    -84.274      0.000      -0.535      -0.511
sigma2      5868.6998   2.47e-07   2.37e+10      0.000    5868.700    5868.700
==========================================================================
Ljung-Box (L1) (Q):                   3.89   Jarque-Bera (JB):              4768.91
Prob(Q):                              0.05   Prob(JB):                         0.00
Heteroskedasticity (H):               1.05   Skew:                            -0.40
Prob(H) (two-sided):                  0.04   Kurtosis:                         5.42
==========================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 3.06e+25. Standard errors may be unstable.
```

Figure 9 SARIMAX results

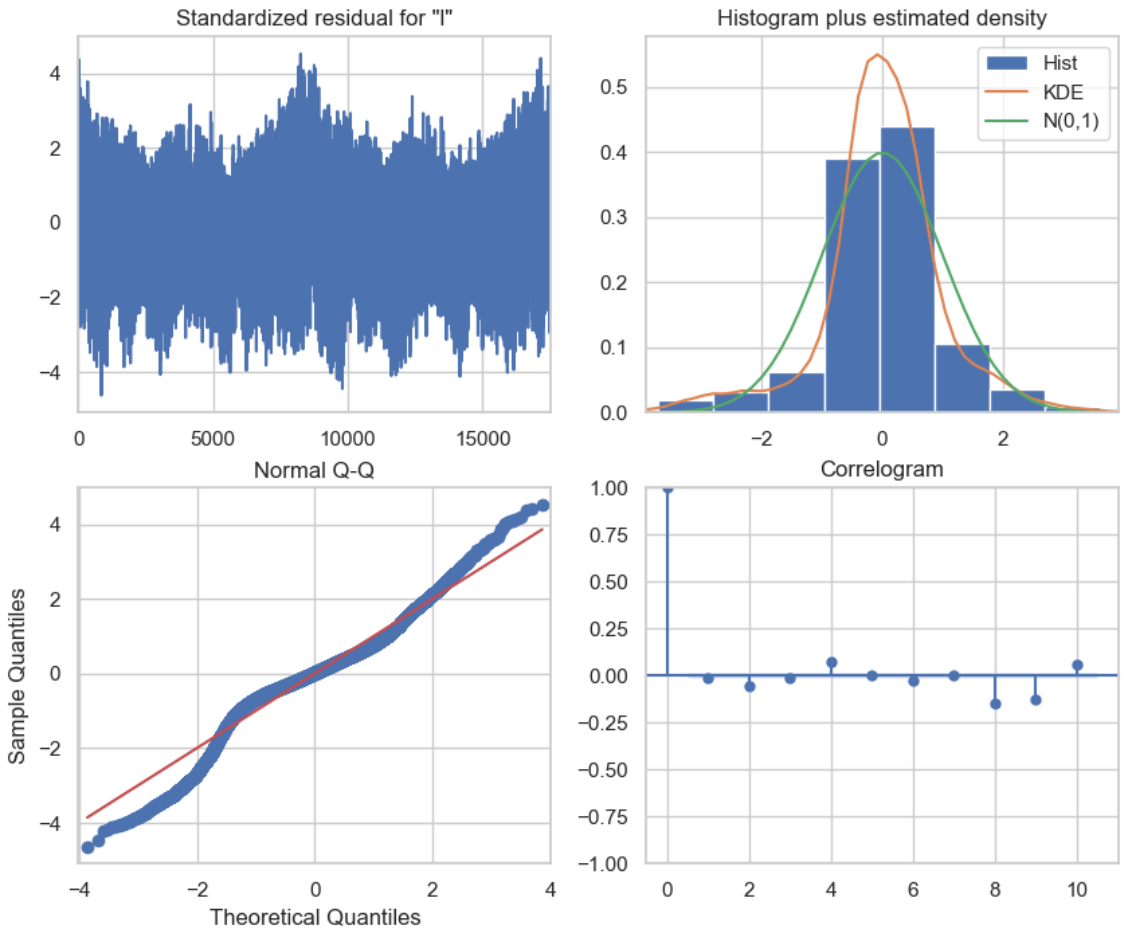

Figure 10 SARIMAX Graph

Throughout the process of fitting the SARIMAX(3, 0, 2) model to the "load_diff" time series, several key points were observed:

- **Model Orders:**

The SARIMAX model incorporates three autoregressive terms (AR) and two moving average terms (MA), denoted as (3, 0, 2).

- **Fit Metrics:**

The AIC, a metric balancing model accuracy with complexity, was calculated as 202061.056. This lower score suggests a relatively good fit, considering the trade-off between fit quality and model simplicity.

- **Model Coefficients:**

Coefficients for the AR terms (ar.L1, ar.L2, ar.L3) and MA terms (ma.L1, ma.L2) provide insights into temporal dynamics and relationships between observations. Significant values indicate the importance of these terms in modeling.

- **Residual Standard Deviation (sigma2):**

The standard deviation of residuals, presented as 5868.6998, reflects the variability of model residuals. This value is crucial for assessing the spread of residuals around the mean and understanding the model's consistency in capturing data variability.

- **Statistical Tests:**

The Ljung-Box test (Q) assessed autocorrelation of residuals. In this case, a value of 3.89 suggests that, overall, residuals do not exhibit significant autocorrelation.

The Jarque-Bera test was applied to check for normality of residuals. The elevated value of 4768.91 and associated p-value (0.00) indicate deviations from normality, emphasizing the need for attention to the distribution of residuals.

- **Heteroskedasticity and Other Statistics:**

The heteroskedasticity test (H) yielded a p-value of 0.04, suggesting a potential presence of heteroskedasticity in residuals.

A skewness (Skew) close to zero suggests a symmetric distribution of residuals, while kurtosis (Kurtosis) of 5.42 indicates a slight "heavy tail" in the distribution.

- **Warnings and Caveats:**

Warnings were issued, alerting to potential issues in the stability of results, particularly related to the covariance matrix. These warnings underscore the need to interpret results with caution.

This comprehensive analysis highlights crucial aspects of the model, from its orders to the quality of fit, providing a comprehensive view of the performance and characteristics of the SARIMAX(3, 0, 2) model concerning the "load_diff" time series.

Subsequently, we performed the following steps in the analysis of the model residuals:

We calculated the residuals of the fitted model (model_fit) using the function “model_fit.resid”.

We conducted the Ljung-Box test on the residuals using the “acorr_ljungbox“function, considering lags from 1 to 10.

The test results were extracted from the p-values associated with each lag.

The obtained results were the following p-values for lags 1 to 10:

```
1      1.613498e-02
2      4.795274e-11
3      4.146628e-11
4      6.684672e-29
5      4.194454e-28
6      2.048422e-29
7      8.434433e-29
8     2.248260e-100
9     1.803472e-151
10    2.224347e-163
Name: lb_pvalue, dtype: float64
```

Figure 11 Results of the residuals of model_fit

These p-values represent the probability of observing autocorrelations in the residuals up to the corresponding lag. Low p-values indicate evidence against the null hypothesis of no autocorrelation.

- **Lag 1 (1.613498e-02):**

  The p-value is 0.0161, indicating a probability of approximately 1.61% of observing autocorrelation in the residuals at the first lag. This relatively low value suggests possible significant autocorrelation.

- **Lag 2 (4.795274e-11):**

  The p-value is extremely low (approximately 4.80e-11), indicating a virtually nil probability of no autocorrelation in the residuals at the second lag. This suggests strong evidence of autocorrelation.

- **Lag 3 (4.146628e-11):**

  Like lag 2, the p-value is very low (approximately 4.15e-11), indicating strong evidence of autocorrelation in the residuals at the third lag.

- **Lags 4 to 10 (6.684672e-29 to 2.224347e-163):**

  The p-values for subsequent lags are extremely low, indicating consistent evidence of significant autocorrelation in the residuals. As the lag increases, the probabilities of no autocorrelation become virtually zero.

Then, we calculated the Mean Squared Error (MSE) for three different forecasting methods (historical mean, last value, and ARMA(2)) and prints the results. Here's an analysis of the obtained MSE values:

- **MSE historical Mean (301711.77):**

  The MSE for the historical mean method represents the average squared difference between the actual and predicted values based on the historical mean. A higher MSE indicates a larger deviation between predicted and actual values, suggesting that the historical mean may not capture the underlying patterns in the data well.

- **MSE last value (325566.248):**

  The MSE for the last value method measures the squared difference between the actual and predicted values based on the last observed value. A higher MSE suggests that this simplistic method may not effectively capture the variability in the data, especially if there are significant changes over time.

- **MSE ARMA(2) (196935.768):**

  The MSE for the ARMA(2) method reflects the squared difference between the actual and predicted values based on the ARMA(2) model. A lower MSE indicates a better fit of the model to the data compared to the historical mean and last value methods. The ARMA(2) method seems to provide a more accurate representation of the underlying patterns in the time series.

In summary, the MSE values suggest that the ARMA(2) method outperforms the historical mean and last value methods in terms of predictive accuracy for the given dataset. Lower MSE values indicate better model performance in capturing the observed variability.

7. Plot Prediction Load

Subsequently, a chart depicting the last 300 data points of the time series stored in the "test" DataFrame was generated. The “plot()” function was employed to visualize the temporal evolution of the data within this specific interval. This visual representation allows for the analysis of trends, patterns, or anomalies present in the time series, providing a more intuitive understanding of the data's behavior over time. The “plt.show()” command was used to display

the resulting chart, facilitating the observation and interpretation of recent patterns in the time series.

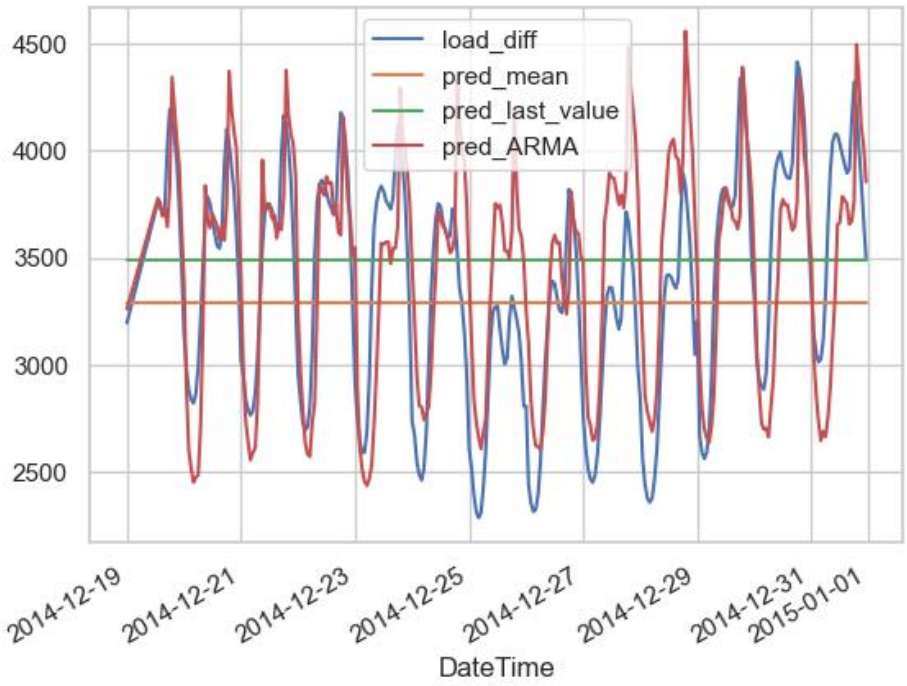

Figure 12 Plot Prediction Load

8. Time series forecasting with Machine Learning

Posteriorly, the dataset was divided into training, validation, and test sets. The training set (`train`) encompasses data points preceding the date '2014-01-01', the validation set (`validation`) includes data points from '2014-01-01' to '2014-09-01', and the test set (`test`) contains data points on or after '2014-09-01'.

Additionally, a Min-Max scaling transformation was applied to standardize the feature values within a specific range. The “MinMaxScaler()” was utilized for this purpose. The training, validation, and test sets were independently scaled using the fitted scaler. The resulting scaled datasets are named “train_scaled”, “validation_scaled”, and “test_scaled”, respectively.

The purpose of scaling is to normalize the feature values, ensuring that they fall within a consistent range. This process is often beneficial for machine learning models that are sensitive to the scale of input features, such as neural networks. The scaled training set (“train_scaled”) is provided as a glimpse of the standardized data.

Subsequently, we created lagged variables for the training, validation, and test sets. New columns were generated for each input variable (in this case, 'load') shifted over different time periods (lags). Specifically, columns 'x_1' through 'x_5' were created, representing 'load' values shifted from 1 to 5 time periods.

After the shifting process, rows with resulting NaN values were removed to ensure consistency in the datasets. The independent (X) and dependent (y) variables were then defined as follows:

1. **For the training set:**

"X_train": Variables 'x_1' through 'x_5' after removing rows with NaN values.

"y_train": The 'load' variable shifted 5 periods forward after removing rows with NaN values.

2. **For the validation set:**

"X_validation": Variables 'x_1' through 'x_5' after removing rows with NaN values.

"y_validation": The 'load' variable shifted 5 periods forward after removing rows with NaN values.

3. **For the test set:**

"X_test": Variables 'x_1' through 'x_5' after removing rows with NaN values.

"y_test": The 'load' variable shifted 5 periods forward after removing rows with NaN values.

The results indicate the shape of the datasets after processing:

```
X_train shape: (17540, 5)
X_test shape: (2923, 5)
y_train shape: (17540,)
y_test shape: (2923,)
```

Figure 13 Results

- "X_train": 17540 observations with 5 variables.
- "y_train": 17540 observations.
- "X_test": 2923 observations with 5 variables.
- "y_test": 2923 observations.

This data preparation step is common in time series problems, enabling the model to utilize historical information for forecasting future values.

8.1 Linear Regression

After data preparation, we proceeded with the application of a Linear Regression model for forecasting. Here is a summary of the process and the obtained results:

Steps Taken:

1. Linear Regression Model: We utilized the LinearRegression class from the scikit-learn package to create a Linear Regression model.

2. Model Training: The model was trained with the training sets “X_train” and “y_train”, where “X_train” contains the independent variables, and “y_train” represents the dependent variable shifted 5 periods forward.

3. Predictions: The trained model was applied to make predictions on the test set (“X_test”), generating predictions denoted as “y_pred_lr”.

4. Performance Evaluation: We calculated performance evaluation metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2).

5. Results Visualization: A graph was produced comparing actual loads (“y_test”) with loads predicted by the Linear Regression model (“y_pred_lr”) for the last 300 records of the test set.

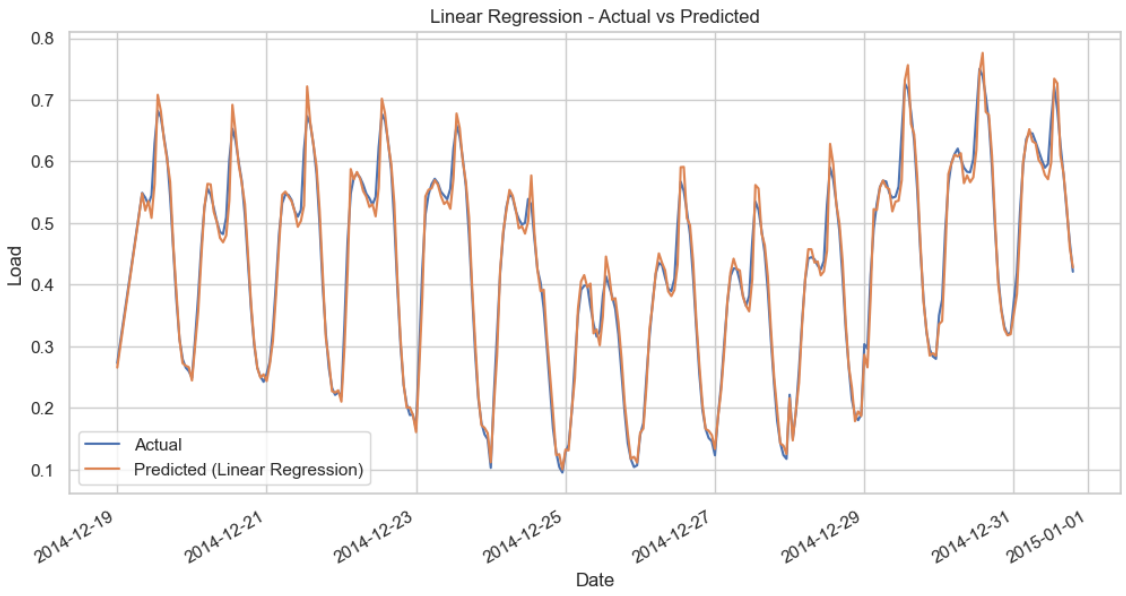

Figure 14 Linear Regression graph

```
Mean Squared Error (Linear Regression): 0.000552609589579745
Mean Absolute Error (Linear Regression): 0.01664332182479851
R-squared (Linear Regression): 0.9780635214420328
```

Figure 15 Linear Regression Results

These results indicate good performance of the Linear Regression model in the task of electricity load forecasting. The low MSE and MAE, along with a high R^2, suggest that the model's predictions are close to actual values, with a significant explanation of variability in the data. The graph provides an additional visualization of this agreement between predictions and actual values.

8.2 Random Forest

After executing the Linear Regression model, we proceeded with the application of a Random Forest Regression model for predictions. Here is an analysis of the process and the results obtained:

Steps Taken:

1. Random Forest Regression Model: We utilized the RandomForestRegressor class from scikit-learn, configuring 100 estimators and setting the random seed to 42.

2. Model Training: The model was trained with the training sets X_train and y_train, where X_train contains the independent variables, and y_train represents the dependent variable shifted 5 periods forward.

3. Predictions: We applied the trained model to make predictions on the test set (X_test), generating predictions referred to as y_pred_rf.

4. Performance Evaluation: We calculated performance evaluation metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2).

5. Results Visualization: We produced a graph comparing actual loads (y_test) with loads predicted by the Random Forest Regression model (y_pred_rf) for the last 300 records of the test set.

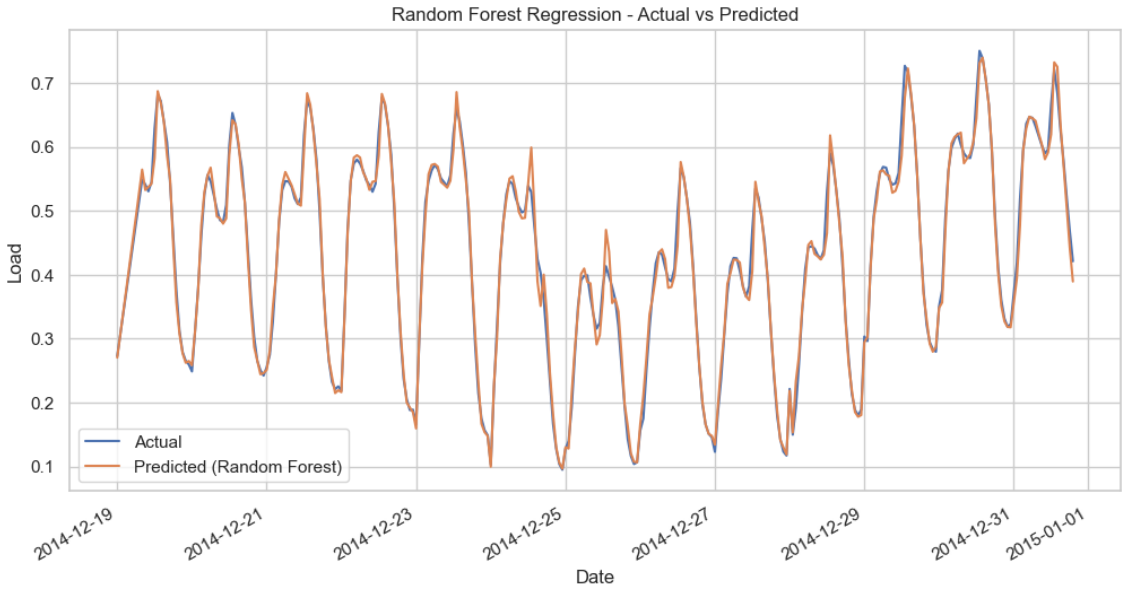

Figure 16 Random Forest Graph

```
Mean Squared Error (Random Forest): 0.00022025777008064385
Mean Absolute Error (Random Forest): 0.010001557700549332
R-squared (Random Forest): 0.991256612368464
```

Figure 17 Random Forest Results

The results indicate a notable performance of the Random Forest Regression model in the task of electric load prediction. The low MSE and MAE, along with an R^2 close to 1, suggest that the model's predictions closely align with the actual values. The graph highlights the model's effectiveness in reproducing trends and fluctuations in electric load. Compared to the Linear Regression model, the Random Forest Regression model demonstrates a superior ability to capture more complex patterns in the data.

8.3 Decision Tree

After applying the Decision Tree Regressor model, we conducted the following analysis of the process and obtained results:

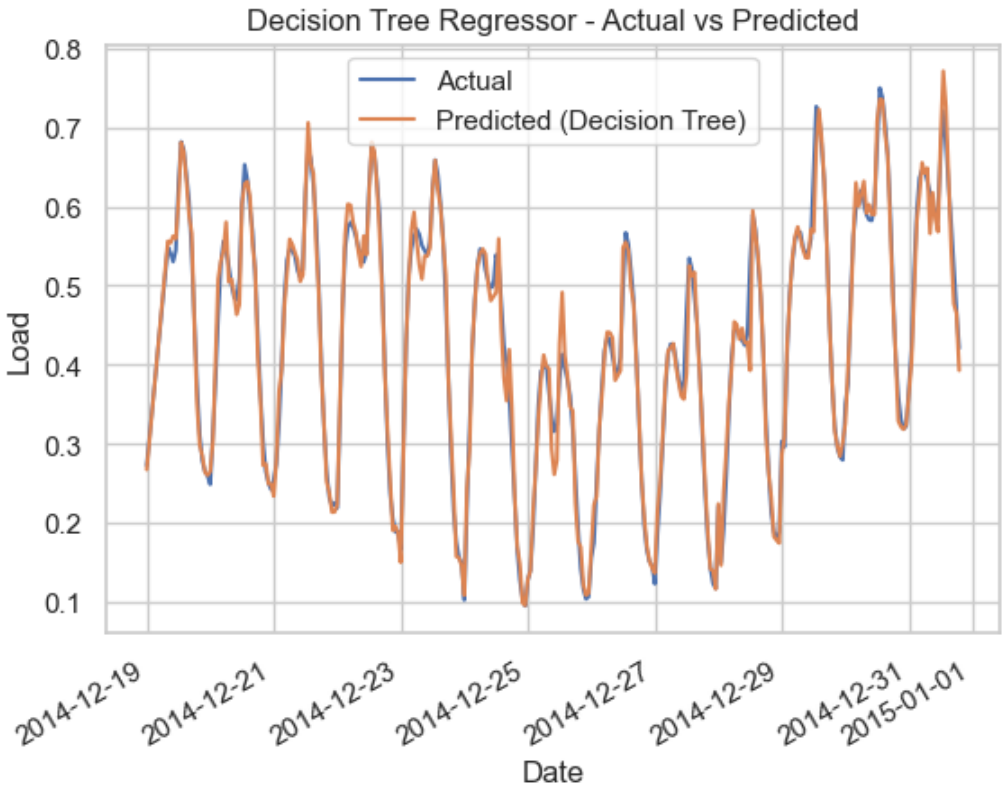

Figure 18 Decision Tree Graph

```
Mean Squared Error (Decision Tree): 0.00047897771184698837
Mean Absolute Error (Decision Tree): 0.014286248956920487
R-squared (Decision Tree): 0.9809864242246208
```

Figure 19 Decision Tree Results

These results indicate a robust performance of the Decision Tree Regressor model in the task of predicting electrical load. The low MSE and MAE, along with a high R^2, suggest that the model's predictions align well with actual values, demonstrating the model's ability to capture complexity in electrical load data. The graph provides a clear visualization of this agreement between predictions and actual values.

8.4 XGBOOST

After implementing the XGBoost (Extreme Gradient Boosting) model, we conducted an analysis of the process and obtained the following results:

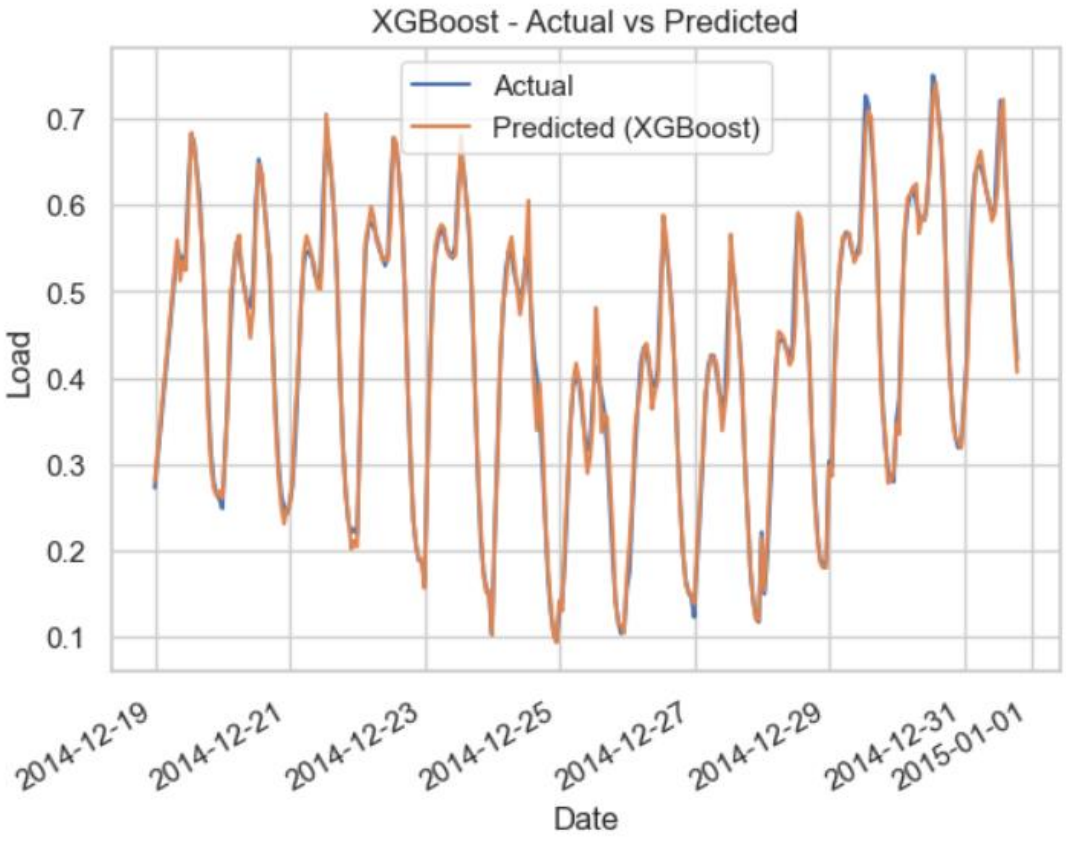

Figure 20 XGBoost Graph

```
Mean Squared Error (XGBoost): 0.0002495141947078466
Mean Absolute Error (XGBoost): 0.0110972478272591
R-squared (XGBoost): 0.990095244662186
```

Figure 21 XGBoost Results

These results indicate the exceptional performance of the XGBoost model in predicting electrical load. The low MSE and MAE values, along with a high R-squared value of 0.990095244662186, suggest that the XGBoost model provides highly accurate predictions. The graph visually demonstrates the close alignment between the XGBoost model's predictions and the actual load values, showcasing its effectiveness in capturing the underlying patterns in the data.

8.5 SVR

After applying the Support Vector Regression (SVR) model, we conducted an analysis of the process and obtained the following results:

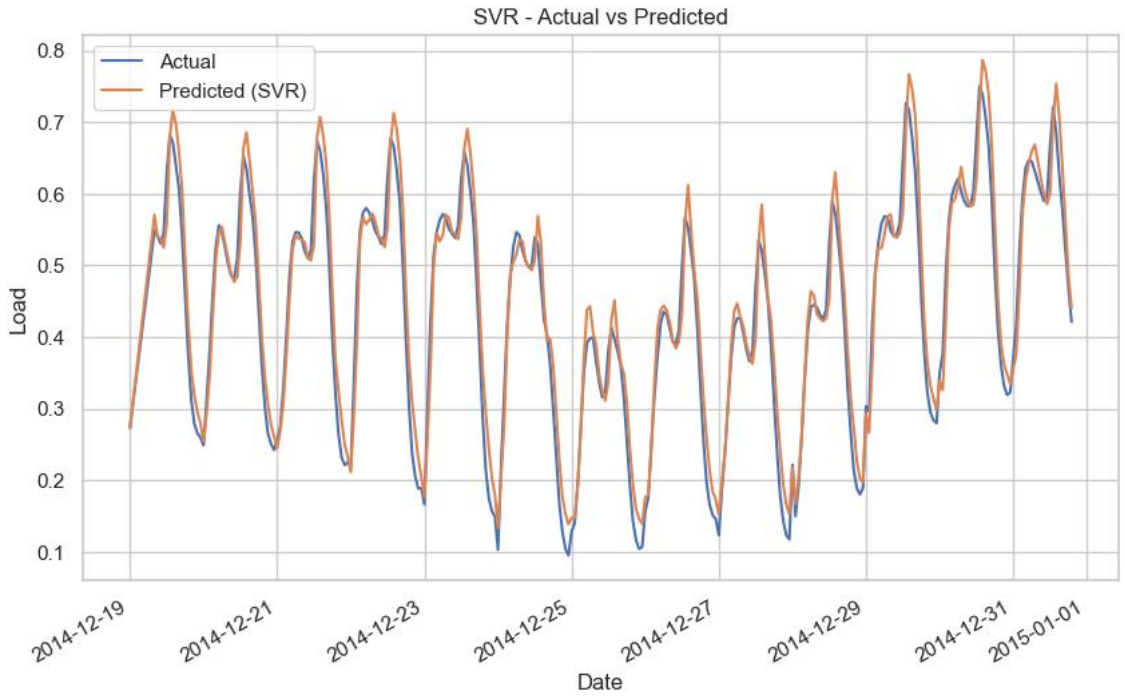

Figure 22 SVR Graph

```
Mean Squared Error (SVR): 0.0015592019305049566
Mean Absolute Error (SVR): 0.03151757653612558
R-squared (SVR): 0.9381056710541803
```

Figure 23 SVR Results

These results indicate the performance of the Support Vector Regression (SVR) model in predicting electrical load. The MSE and MAE values, although higher compared to some other models, still reflect a reasonable accuracy in load predictions. The R-squared value of 0.9381056710541803 suggests a good fit of the SVR model to the data. The graph visually demonstrates the alignment between the SVR model's predictions and the actual load values.

9. Time series forecasting with Deep Learning

9.1 LSTM

We created a plot to visually compare the actual electric load values with the predicted values for the last 300 records of the test set.

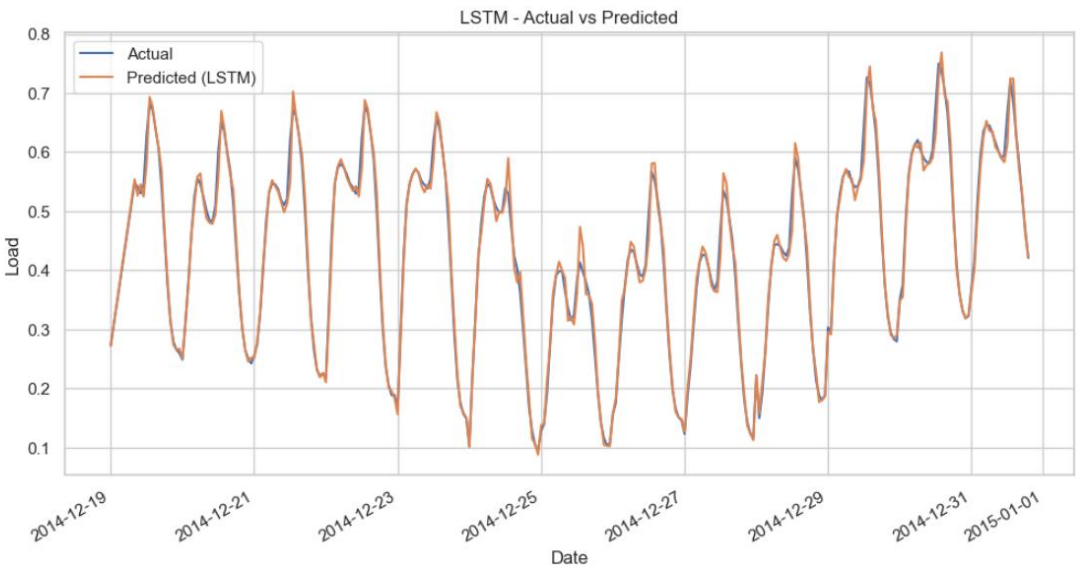

Figure 24 LSTM Graph

```
Mean Squared Error (LSTM): 0.00026625862790295485
Mean Absolute Error (LSTM): 0.011320075679349164
R-squared (LSTM): 0.9894305549668279
```

Figure 25 LSTM Results

The low MSE and MAE values suggest that the LSTM model provides accurate predictions. The high R^2 score (close to 1) indicates that the model effectively captures the variability in the electric load data. The visual representation further illustrates the alignment between the predicted and actual values.

Overall, these results indicate that the LSTM model is highly effective in forecasting electric load, showcasing its ability to capture temporal dependencies in the data.

9.2 GRU

After implementing the Gated Recurrent Unit (GRU) model for electric load forecasting, we proceeded with the evaluation of the model's performance. Here's an overview of the steps taken and the obtained results:

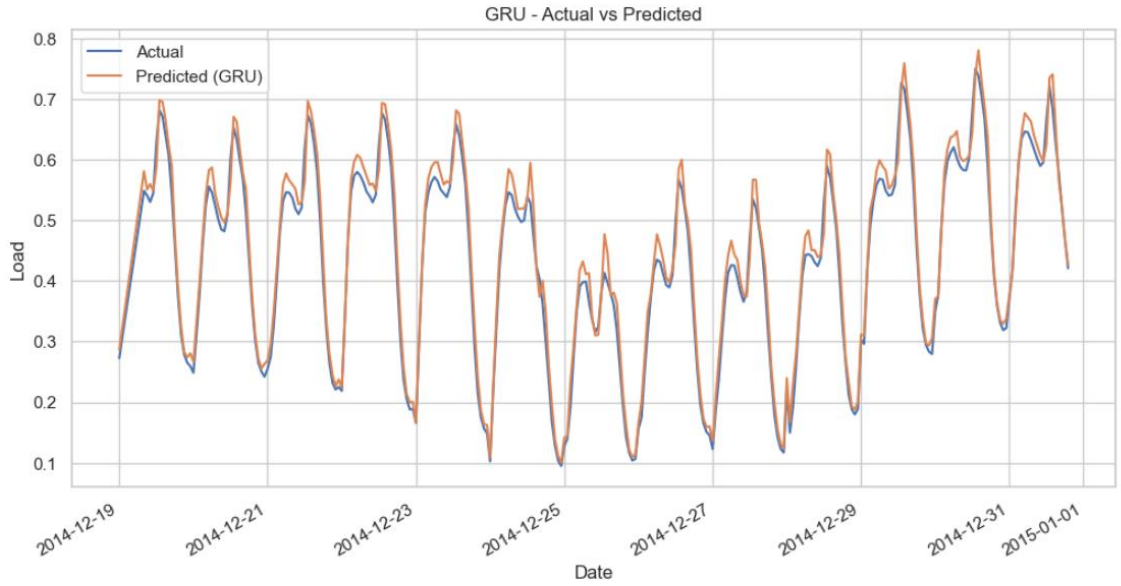

Figure 26 GRU Graph

```
Mean Squared Error (GRU): 0.0006094470828797177
Mean Absolute Error (GRU): 0.020213001070060653
R-squared (GRU): 0.9758072912271144
```

Figure 27 GRU Results

The MSE and MAE values suggest that the GRU model provides accurate predictions, with the R^2 score indicating a strong ability to capture the variability in the electric load data. The visual representation illustrates the alignment between the predicted and actual values.

In summary, these results indicate that the GRU model is effective in forecasting electric load, demonstrating its ability to capture temporal dependencies in the data, albeit with slightly higher errors compared to the LSTM model.

10. Evaluating

After applying various forecasting models, including linear regression, random forest, decision tree, XGBoost, support vector regression (SVR), LSTM, and GRU, we proceeded to compare their performance using different evaluation metrics. Here's an explanation of the process and the obtained results:

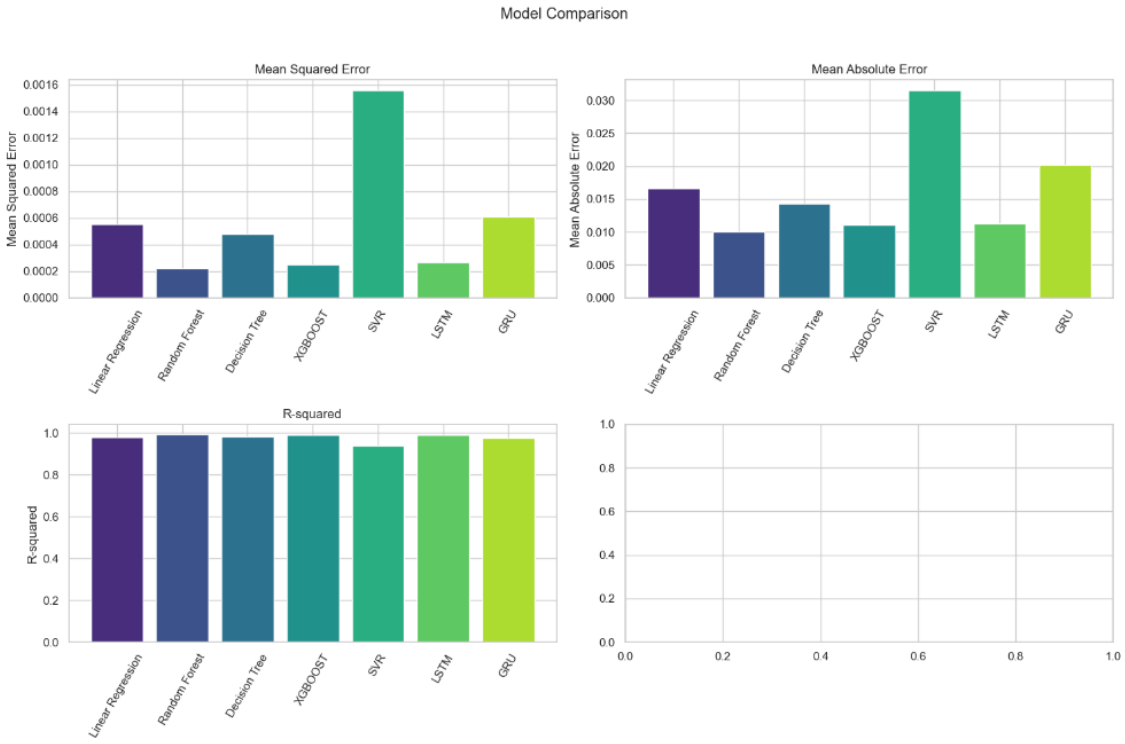

Figure 28 Model Comparison

```
MSE historical Mean 301711.77
MSE last value 325566.248
MSE ARMA(2) 196935.768
```

Figure 29 Model Comparison Results

The Random Forest model demonstrates the best overall performance, exhibiting the lowest values for MSE, MAE, and the highest R^2. XGBoost and LSTM also show robust results, with a good ability to forecast and fit the data. Support Vector Regression (SVR) performed less effectively compared to tree-based models and neural networks. The choice between LSTM and GRU may depend on specific factors, but both exhibit good performance in the forecasting task. In summary, considering the metrics used, the Random Forest model stands out as the most effective choice for electric load forecasting in this scenario.