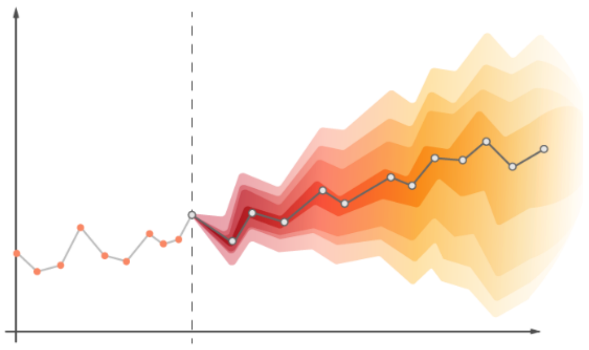


Streaming Data Management and Time Series Analysis Project

19-01-2025

Andrea D'Amicis

Contents

1 Introduction

This report describes the steps taken to make predictions of the provided time series, which contains data about the **hourly traffic congestion indicator** for a freeway in a large U.S. city. The provided data covered a **period from 2015-01-01 to 2016-11-30**, and the goal was to predict the hourly values of the time series of the last month from 2016-12-01 to 2016-12-31.

First, the phase of data exploration is described: relevant features of the provided data are discussed and leveraged to perform some useful data preprocessing operations. Then, in the following three sections, respectively, regarding the development of an ARIMA model, a UCM model, and a ML model, the steps taken towards the selection of the best performing model for each of the categories are described.

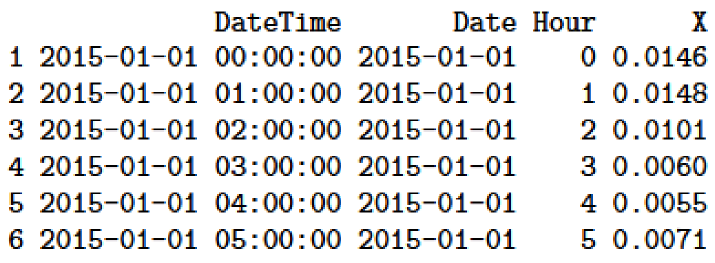
2 Data Exploration and Preprocessing

This section focuses on exploring the dataset's content, identifying any potential characteristics that might lead to issues during the modeling phase, and addressing them proactively.

2.1 Dataset

Time serie is table of 17544 observations and 4 columns having the following variables:

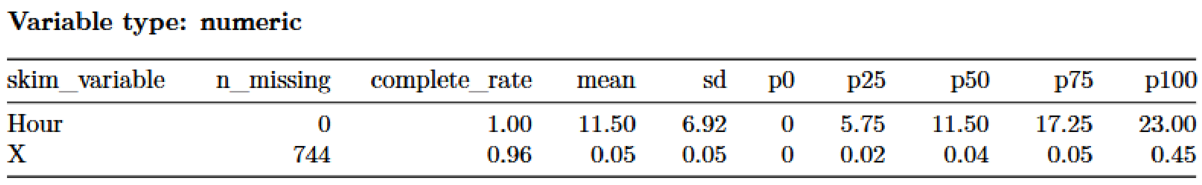
- **DateTime**, in the format YYYY-MM-DD : HH:MM:SS, from 2015-01-01 to 2016-11-30.
- **Date**, in format YYYY-MM-DD.
- **Hour**, from 0 to 23.
- **X**, referred to the value of time serie.



```
             DateTime       Date Hour      X
1 2015-01-01 00:00:00 2015-01-01    0 0.0146
2 2015-01-01 01:00:00 2015-01-01    1 0.0148
3 2015-01-01 02:00:00 2015-01-01    2 0.0101
4 2015-01-01 03:00:00 2015-01-01    3 0.0060
5 2015-01-01 04:00:00 2015-01-01    4 0.0055
6 2015-01-01 05:00:00 2015-01-01    5 0.0071
```

Figure 1: The first 6 observations of the provided dataset

Below is provided a little further inspection of the numeric variables in order to understand the distribution of the data. In fact, excluding time variables, the time series is approximately 95 per cent complete, indicating that there are missing data corresponding to 744 observations but these regard the hourly predictions that we have



Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Hour	0	1.00	11.50	6.92	0	5.75	11.50	17.25	23.00
X	744	0.96	0.05	0.05	0	0.02	0.04	0.05	0.45

Figure 2: Distribution of the numeric variables

to provide in the final part of project. The values of the series are between 0 and 0.45 with a mean of 0.046 and a rather moderate standard deviation of 0.049. The 50th percentile is 0.037 indicating that 50% of the series has values above this value. In addition, the 75th percentile, with a value of 0.054, is also rather low compared to the highest value in the series. I expect to see the series remaining mostly below this value and reaching with several peaks the value of the maximum.

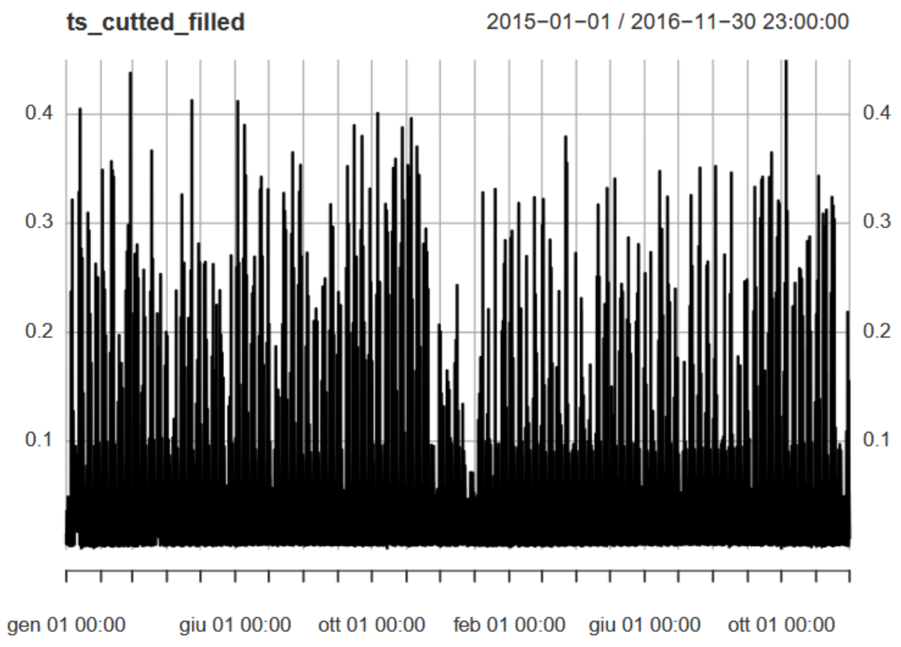


Figure 3: Plot of the time serie

As mentioned from the previous comment, the data exhibits significant variability with frequent spikes, suggesting potential seasonality or cyclical patterns. The majority of observations are below 0.1 and there are severeal peaks across the entire time serie. There is a decrease of traffic in the final part of the year 2015 suggesting a seasonality related to holidays.

2.2 Zero values and related preprocessing

It's conducted a thorough investigation into the presence of NA and zero values within the time series dataset. The results of this analysis revealed that there were no NA values present; however, are identified a total of **107 instances** where the values were equal to zero.

After careful consideration, it is reasonable to concluded that these zero values could potentially distort the analysis and decided to address them appropriately.

To ensure continuity and preserve the integrity of the data, these zero values are replaced using a **moving median method**. This approach involves calculating the median value within a 24-hour window, consisting of 12 hours before and 12 hours after each zero value. This method was selected because it effectively smooths out anomalies while retaining the temporal patterns in the data.

The filled gaps (likely from missing data imputation) seem to integrate well with the general structure, maintaining continuity in the series.

2.3 Outliers and related preprocessing

To investigate the presence of outliers, a histogram was created to analyze the frequency distribution across different classes. The analysis showed that the majority of the values are concentrated in the two classes ranging from 0 to 0.1. Based on this analysis, it was concluded that the values belonging to the very low-frequency class are likely to be considered **outliers**. Replacing these outlier values with the median was deemed reasonable. Therefore, after determining an appropriate quantile threshold, the replacement of these values was carried out accordingly.

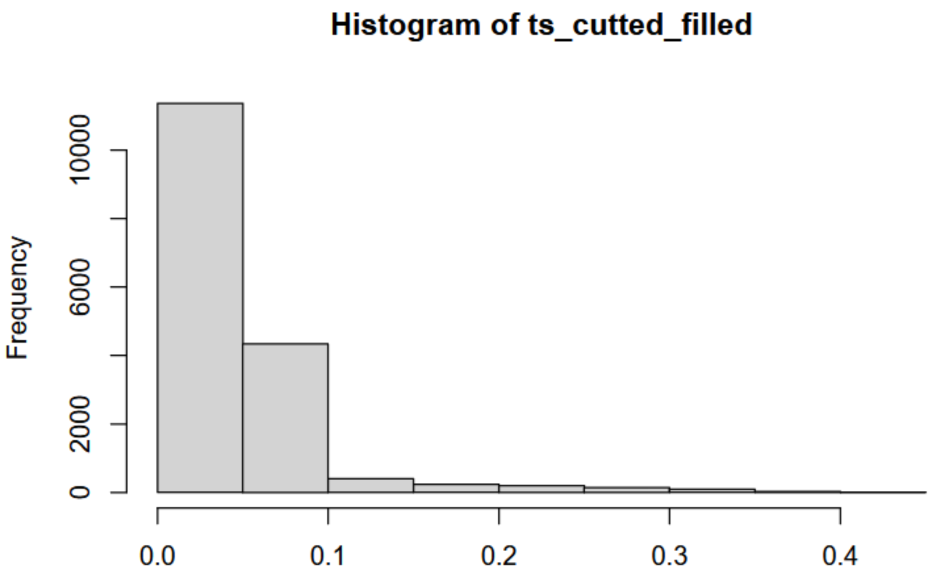


Figure 4: Histogram of the time serie

The **99th percentile** of the time series was calculated to identify the threshold above which values can be considered outliers. By counting the observations that exceeded this threshold, it was found that there were only **168 such instances**. This number is very small compared to the total number of observations in the dataset. Given their limited presence, handling these outliers by replacing them with the median value was deemed reasonable, ensuring the dataset remains robust and free from very extreme anomalies that could skew the analysis.

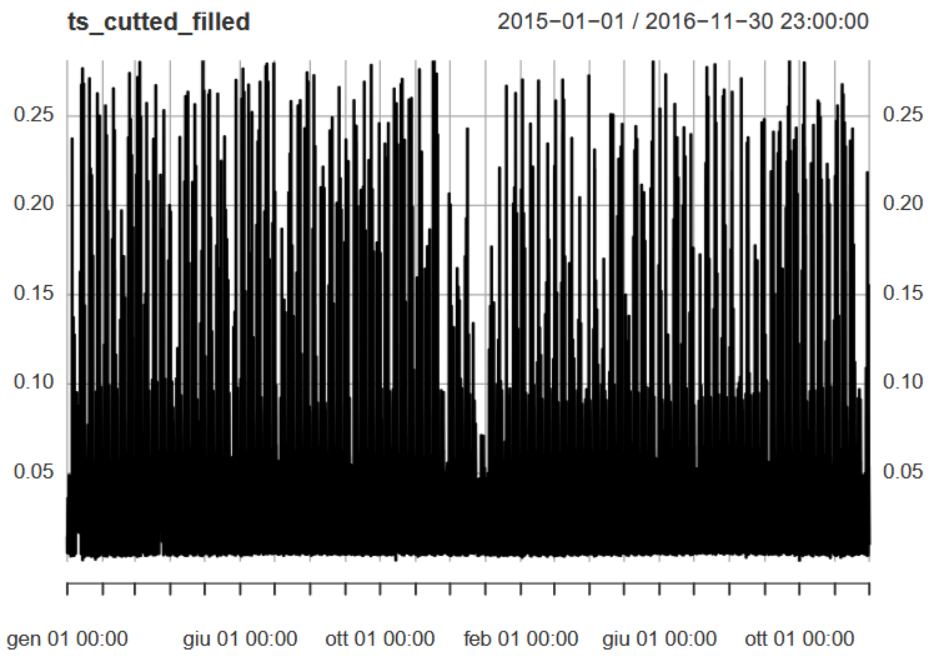


Figure 5: Final plot of the entire time serie after preprocessing

2.4 Split in 24 time series

The hourly time series was divided into **24 daily time series** with each series containing observations recorded at the **same hour of the day across all days**. This approach was adopted to facilitate a more detailed analysis of seasonality across multiple levels, such as weekly and annual patterns.

This decomposition allows the behavior of each specific hour to be studied independently, enabling the identification of daily recurring patterns that might be obscured when analyzing the entire time series as a whole. By isolating each hour, it becomes easier to detect variations in trends and patterns that occur consistently at specific times of the day.

Additionally, this method provides a clearer perspective on how seasonality interacts across different time scales. For instance, weekly patterns, such as differences between weekdays and weekends, and annual trends, such as seasonal fluctuations, can be analyzed with greater precision. This granular approach enhances the ability to identify and model complex temporal dynamics, ultimately improving the quality of insights and forecasts.

2.5 Stationarity and Box-Cox

To examine the stationarity of the 24 time series, the relationship between the mean and the standard deviation was analysed by plotting their trend to investigate whether they were correlated. The observed correlation between the mean and standard deviation suggests that the time series are not stationary. As a result, applying a variance-stabilizing transformation becomes a logical step to address this issue. Stationarity is a crucial property for time series analysis because it ensures the consistency of statistical properties over time, which is necessary for reliable modeling and forecasting. A time serie, in essence, represents a single realization of a random process, and stationarity helps to ensure that meaningful inferences can be drawn about the underlying variable generating the data.

To address this, the **Box-Cox transformation** was applied to each of the 24 daily time series, and the optimal lambda value was calculated for each series. The obtained lambda values are as follows:

Optimal Lambdas					
-0.99995568	-0.99995223	-0.99992471	-0.28183339	-0.03410270	-0.40158558
0.07960928	-0.14448631	-0.30520352	-0.33984977	-0.22154101	1.99992425
1.99992425	1.99992425	0.48873312	0.10206000	0.08864715	1.09296178
0.05761952	0.87154550	0.64040828	-0.75452759	-0.25330359	-0.91224536

Table 1: Optimal Lambdas

The diversity in the optimal lambda values highlights the varying characteristics of the individual time series. Some series require transformations close to a log transformation (lambda near zero), while others require a different transformations (lambda near 1, 2 or -1). This variation indicates that each time series exhibits distinct statistical properties, necessitating tailored adjustments to achieve stationarity. The differences also emphasize the complexity of the dataset and the importance of addressing these variations individually to ensure accurate and robust modeling. For a clear understanding, I report below a table with a corresponding transformations related to lambdas.

λ	**Transformation**
-2	$\frac{1}{x^2}$
-1	$\frac{1}{x}$
-0.5	$\frac{1}{\sqrt{x}}$
0	$\log(x)$
0.5	$\sqrt{x}$
1	x
2	x^2

2.6 Augmented Dickey–Fuller test (ADF)

The Augmented Dickey–Fuller (ADF) test is applyed to check if the preprocessing I performed was consistent and if all 24 series were stationary. The ADF test evaluates whether a time series is stationary by comparing the tau statistic to a critical value at a given significance level (in this case, 5%).

In the results, if the tau statistic is more negative than the critical value at the 5% level, the null hypothesis of non-stationarity is rejected, indicating that the series is stationary. From the table, most of the series have a tau statistic that is more negative than the critical value of -2.86 at the 5% level, and their corresponding p-values are less than 0.05, further confirming stationarity. Please note that R, for convenience, doesn't report too small values replacing them with 0.010. For this reason there is a large presence of this value.

Hour	IsStationary	P-Value	Tau Statistic	Critical Value (5%)
1	TRUE	0.01000000	-4.917948	-2.86
2	TRUE	0.01000000	-5.837128	-2.86
3	TRUE	0.01000000	-6.839808	-2.86
4	TRUE	0.01000000	-6.895324	-2.86
5	TRUE	0.01000000	-6.445611	-2.86
6	TRUE	0.01000000	-7.580303	-2.86
7	TRUE	0.01000000	-6.412932	-2.86
8	TRUE	0.01000000	-7.458907	-2.86
9	TRUE	0.01000000	-7.478475	-2.86
10	TRUE	0.01000000	-7.588359	-2.86
11	TRUE	0.01000000	-7.111670	-2.86
12	TRUE	0.01000000	-4.679636	-2.86
13	TRUE	0.01000000	-5.266351	-2.86
14	TRUE	0.01000000	-5.490569	-2.86
15	TRUE	0.01000000	-6.621222	-2.86
16	TRUE	0.01000000	-7.116080	-2.86
17	TRUE	0.01000000	-8.229424	-2.86
18	TRUE	0.01000000	-6.939559	-2.86
19	TRUE	0.01000000	-7.507875	-2.86
20	TRUE	0.01821757	-3.815609	-2.86
21	TRUE	0.04207562	-3.445131	-2.86
22	TRUE	0.04758142	-2.875880	-2.86
23	TRUE	0.01000000	-4.254819	-2.86
24	TRUE	0.01000000	-5.195442	-2.86

Table 2: Augmented Dickey-Fuller Test Results for Stationarity

2.7 Split into Train, Validation and Test

To provide a brief recap and clarify all dimensions, the initial dataset consisted of 17,544 observations of a time serie. The last **744 observations had no values**, as they represent the predictions we aim to make. To proceed, the dataset was divided into training, validation, and test sets for each of the 24 daily time series that were previously created.

Each of these **24 time series** consisted of **731 values**, of which the **last 31 were unknown** and are considered as "test set dimension", representing the values to be predicted. The remaining **700 observations** were split into a training set and a validation set. Specifically, **630 instances** (90% of the dataset) were allocated for the **training set** to fit the models, while the remaining **70 observations** (10%) were reserved for the **validation set** to fine-tune the hyperparameters and evaluate the models during development.

This structured division ensures that the training set is sufficiently large to capture the underlying patterns in the data while leaving enough instances for validation to assess model generalization. The test set, comprising the unknown future values, remains untouched during the training and validation phases, ensuring unbiased evaluation of the forecasting performance. By adhering to these principles, the approach minimizes data leakage and establishes a robust framework for developing and testing models tailored to each time series.

2.8 Dummy variables

To enhance the predictive capability of the model, are introduced a series of dummy variables corresponding to specific dates and holidays, transforming them into one-hot encoded variables. These variables represent special events and holidays that could potentially influence the behavior of the time series. The selected dates and holidays include:

- **Dec24** (Christmas Eve)
- **Dec25** (Christmas Day)
- **Dec26** (Boxing Day)
- **Jan1** (New Year's Day)
- **Jan6** (Epiphany)
- **EasterSat** (Saturday before Easter)
- **Easter** (Easter Sunday)
- **EasterMon** (Easter Monday)
- **EasterTue** (Tuesday after Easter)
- **Aug15** (Assumption of Mary)
- **EndYear** (New Year's Eve)
- **Valentine** (Valentine's Day)

Each of these dates was encoded as a binary variable, where a value of 1 indicates that the observation falls on the specific holiday or date, and 0 otherwise (one-hot encoded variables) to allow the model to handle categorical features effectively, ensuring that no implicit ordinal relationship is introduced between these events.

This transformation is particularly valuable in capturing the potential seasonality or irregular patterns in the data associated with holidays. For instance, traffic behavior around Christmas (Dec24–Dec26) or New Year's Eve (EndYear) often differs significantly from regular days. Similarly, Easter-related variables accommodate the shifting nature of this holiday in the calendar, ensuring its effects are properly modeled.

By including these one-hot encoded variables, the model can better account for the unique impact of holidays and special events on the series' dynamics. This step was critical to refining the predictive performance and ensuring that the model captures both temporal patterns and event-driven variations.

2.9 Evaluation Metric: Mean Absolute Error (MAE)

To evaluate the performance of the forecasting models, the **Mean Absolute Error (MAE)** was used as the primary metric. MAE measures the average magnitude of errors between predicted and actual values, providing an interpretable measure of model accuracy. It is defined mathematically as:

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

Where:

- n is the number of observations in the test set,
- y_i represents the actual observed value at time step i,
- $\hat{y}_i$ is the predicted value at time step i,
- $|y_i - \hat{y}_i|$ is the absolute error for each prediction.

The MAE is expressed in the same units as the original data, making it straightforward to interpret. **A lower MAE indicates better model performance**, as it signifies that the predictions are closer to the actual observations on average.

The choice of MAE as the evaluation metric was motivated by its simplicity and robustness. Unlike metrics such as Mean Squared Error (MSE), MAE is less sensitive to outliers, as it treats all errors with equal weight. This property is particularly important for time series data, where extreme deviations may occur due to irregular events but should not disproportionately affect the evaluation of the model's overall performance.

By focusing on the average absolute deviation, MAE provides a clear and intuitive measure to compare the forecasting accuracy across different models and time series.

3 ARIMA (Auto Regressive Integrated Moving Average) models

The Autoregressive Integrated Moving Average (ARIMA) model is a widely used statistical method for time series forecasting. ARIMA combines three key components:

- **Autoregressive (AR)**: Models the dependency between an observation and a specified number of lagged observations.
- **Integrated (I)**: Applies differencing to make the time series stationary.
- **Moving Average (MA)**: Captures the relationship between an observation and the residual errors from a moving average model applied to lagged observations.

This approach is particularly effective for time series data that exhibit trends, seasonality, or other forms of non-stationarity, provided these patterns can be transformed into a stationary format through preprocessing.

After testing several combinations of ARIMA parameters, I identified three models that showed the most promising performance. These models were then further refined by incorporating dummy variables corresponding to specific dates and holidays, as described in the preprocessing section.

The results obtained from each enriched model were compared to determine the most effective approach. This evaluation highlights the impact of incorporating domain-specific features (such as dummy variables) alongside classical time series modeling techniques.

As they are very similar to each other, I decided to bring only a few graphs of the residuals of the 24 series throughout the paper and not all of them in order to get a general idea of the workflow.

3.1 Arima (3,1,1) (0,1,1)(7)

The ARIMA $(3,1,1)(0,1,1)_7$ model is a seasonal ARIMA configuration specifically designed for time series data with weekly seasonality. The components of the model are as follows:

- **Non-seasonal part** $(p = 3, d = 1, q = 1)$**:**
  - $p = 3$: The model includes three autoregressive (AR) terms, meaning the current value depends on the three most recent lagged observations.
  - $d = 1$: Differencing is applied once.
  - $q = 1$: One moving average (MA) term is included to account for the dependency on the lagged forecast errors.
- **Seasonal part** $(P = 0, D = 1, Q = 1)_7$**:**
  - $P = 0$: No seasonal autoregressive terms are included.
  - $D = 1$: One seasonal differencing is applied to remove seasonal trends.
  - $Q = 1$: One seasonal moving average term is included.
  - The subscript (7) indicates that the seasonal cycle has a period of 7, reflecting weekly seasonality.

The residual results are the following for each of 24 time series.

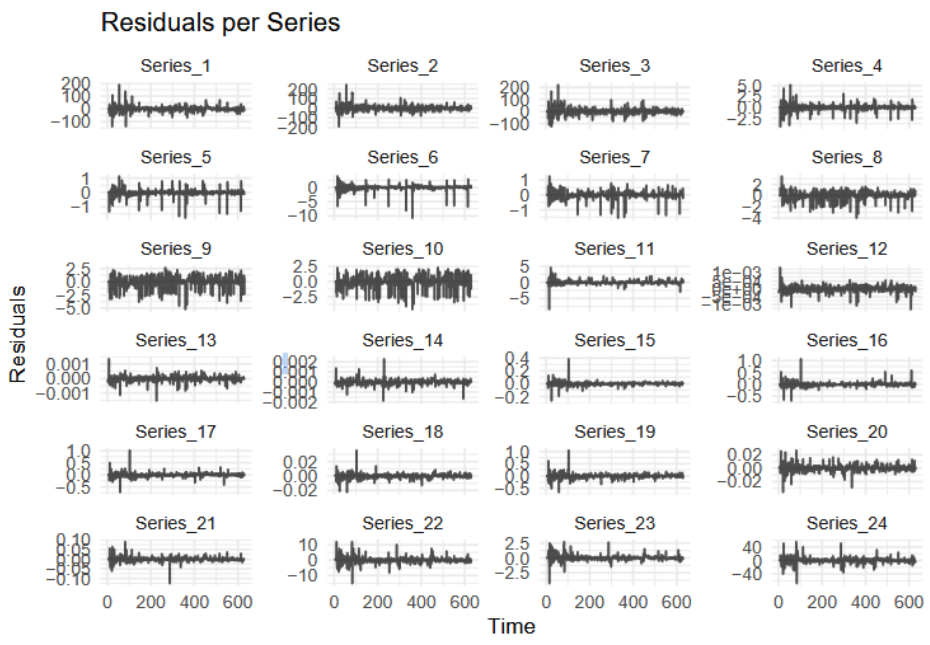


Figure 6: Residuals ARIMA (3,1,1) (0,1,1)(7)

The residues of the series seem to be acceptable. However, it is reasonable to take into account that at peak times there is more traffic and therefore more variation in the time series. In fact, during the hours from 8 to 10 a.m. and around 8 p.m., higher residuals are obtained. The following plot was generated by making predictions for the 24 different time series and then re-packaging the entire series into a single hourly series to visualize the plot with prediction and validation data as ground truth. Regarding the results, the predictions seem to accurately follow the general trend of the series, but at several points, the peaks are underestimated, while towards the end, they are overestimated for just one week. Overall, the model appears to be a good fit, with a **MAE of 0.009563**.

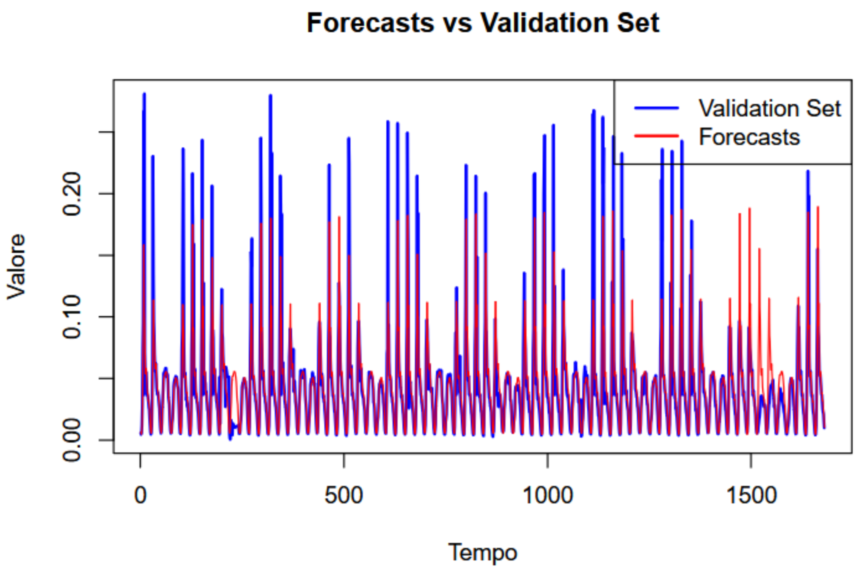


Figure 7: Forecasts plot of ARIMA (3,1,1) (0,1,1)(7)

3.2 Arima (3,1,1) (0,1,1)(7) with dummies

The ARIMA(3,1,1)(0,1,1)(7) model with dummies incorporates additional explanatory variables, known as dummy variables, to capture the effects of specific events or irregularities, such as holidays or other external factors, that might influence the time series outside the regular seasonal and trend patterns.

The residuals of the series seem to be acceptable are very very similar to previous ones. However as previously said, it is reasonable to take into account that at peak times there is more traffic and therefore more variation in the time series.

The following plot was generated in same manner of the previous one, by making predictions for the 24 different time series and then re-packaging the entire series into a single hourly series to visualize the plot with prediction and validation data as ground truth. Regarding the results, the predictions are very similar to the previous one with no significant differences but in some points, it seems to be more accurate.

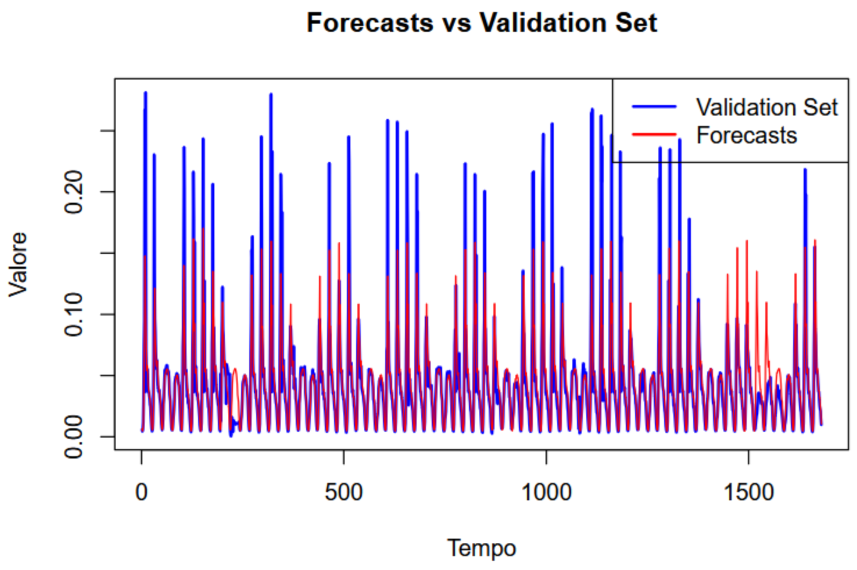


Figure 8: Forecasts plot of ARIMA (3,1,1) (0,1,1)(7) with dummies

Overall, the model appears to be a good fit, with a **MAE of 0.009546** which is a little lower than previous one without dummies.

3.3 Arima (3,1,1) (0,1,2)(7)

Then, I decided to experiment with the ARIMA(3,1,1)(0,1,2)(7) model to see if increasing the order of the Moving Average (MA) component would lead to better results in capturing the underlying patterns of the dataset. By enhancing the MA term, I aimed to improve the model's ability to account for short-term fluctuations and noise in the data, potentially providing a more accurate forecast.

The residual results are the following for each of 24 time series.

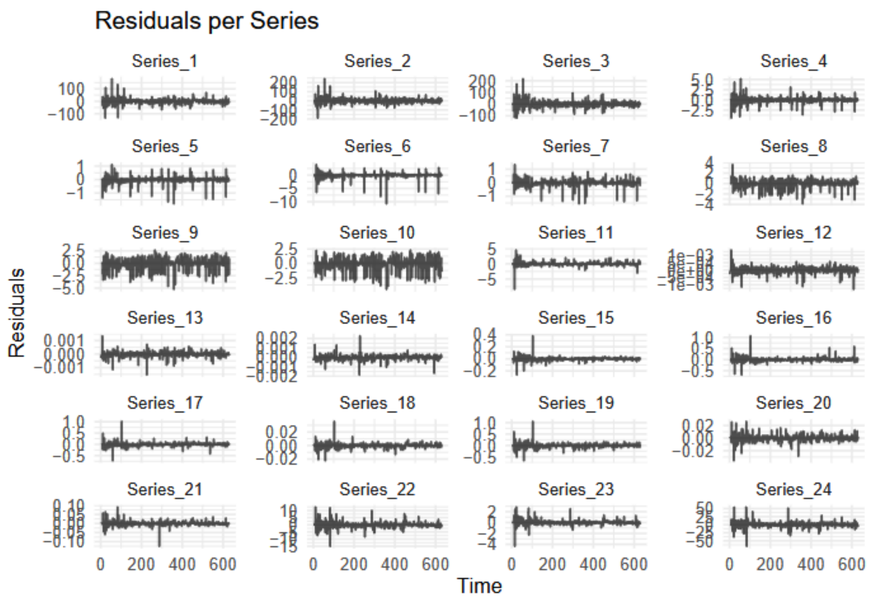


Figure 9: Residuals ARIMA(3,1,1) (0,1,2)(7)

The residuals of the series appear to be acceptable and are quite similar to the previous ones. However, as mentioned earlier, it is important to consider that during peak times, there is increased traffic, leading to greater variability in the time series.

The following plot was generated in same manner of the previous ones. Regarding the results, the predictions are very similar to the previous one with no significant differences but in some points, it seems to be less accurate in comparison with previous dummies model.

Overall, the model appears to be a good fit, with a **MAE of 0.009558** which is a little lower than previous one without dummies.

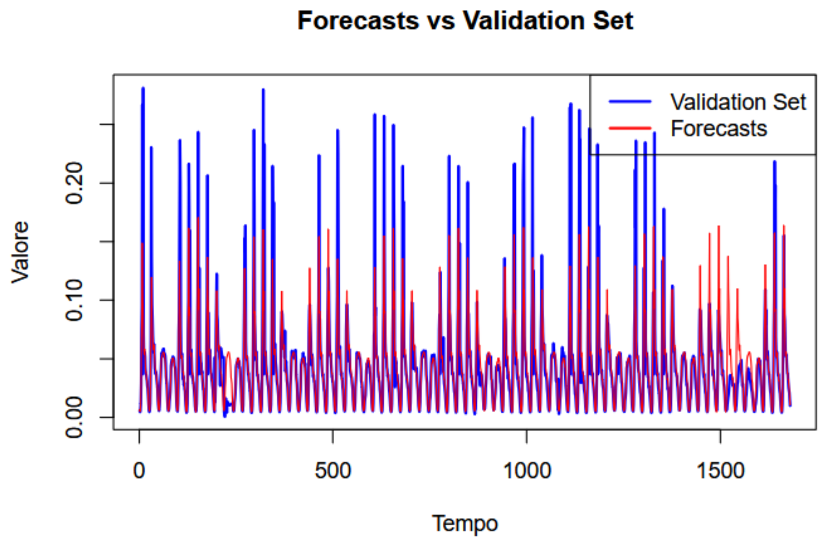


Figure 10: Forecasts plot of ARIMA (3,1,1) (0,1,2)(7)

3.4 Arima (3,1,1) (0,1,2)(7) with dummies

The ARIMA(3,1,1)(0,1,2)(7) model with dummies, as you will see, doesn't bring any particular improvements.

The residual results are very very similar and this suggests that increasing the order of the Moving Average (MA) component did not significantly improve the model's performance. Despite the higher-order MA term, the model still struggles to capture the short-term fluctuations and noise more effectively, indicating that other factors may play a more crucial role in improving the model's accuracy. I tried adding the dummy variables anyway and these are the results.

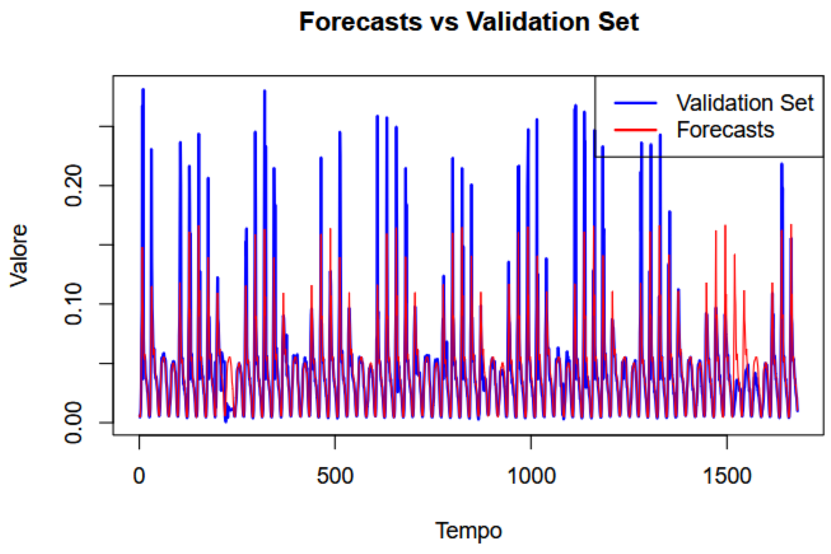


Figure 11: Forecasts plot of ARIMA (3,1,1) (0,1,2)(7) with dummies

The result achieved with this model using dummies is **MAE of 0.009562**.

3.5 Arima (3,1,1) (1,1,1)(7)

The last experiment is ARIMA(3,1,1)(1,1,1)(7) model to see if increasing the order of the Moving Average (MA) component would lead to better results. By raising the AR term, I aimed to allow the model to better capture the longer-term dependencies in the data, potentially improving its ability to forecast future values based on the past observations.

The residual results are the following for each of 24 time series.

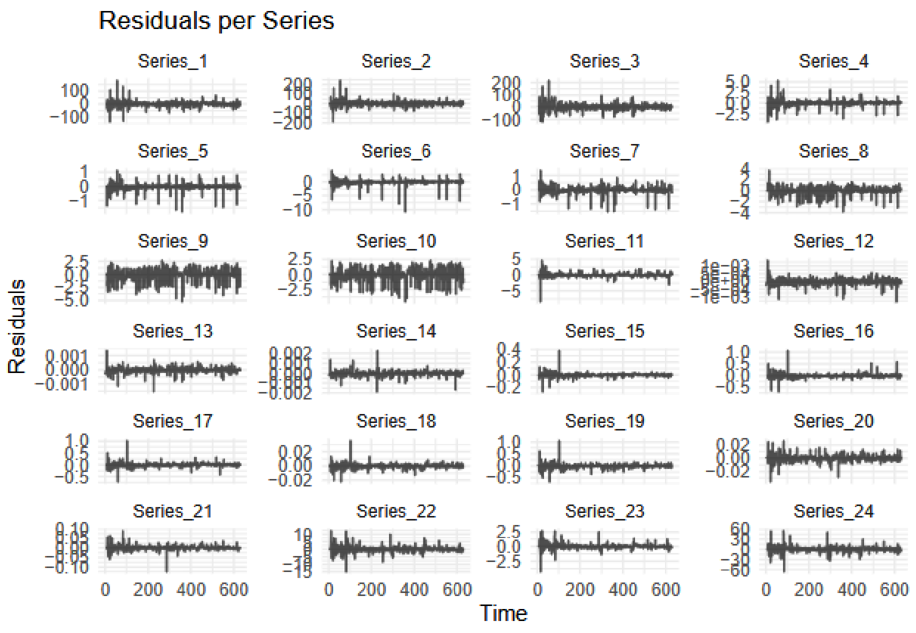


Figure 12: Residuals ARIMA (3,1,1) (1,1,1)(7)

The residuals of the series appear to be acceptable and are very similar to the previous ones. However, as mentioned earlier, it is reasonable to consider that during peak periods, there is an increase in traffic, which leads to greater variation in the time series.

The following plot was generated using the same approach as the previous ones. In terms of results, the predictions closely resemble those of the previous model, with no major differences. However, at certain points, the model appears to be less accurate when compared because produces peak forecasts that are more evenly distributed but still worse than the first category with a greater difference.

Overall, the model appears to be a good fit, with a **MAE of 0.009552** which is a little better than previous one but very very close.

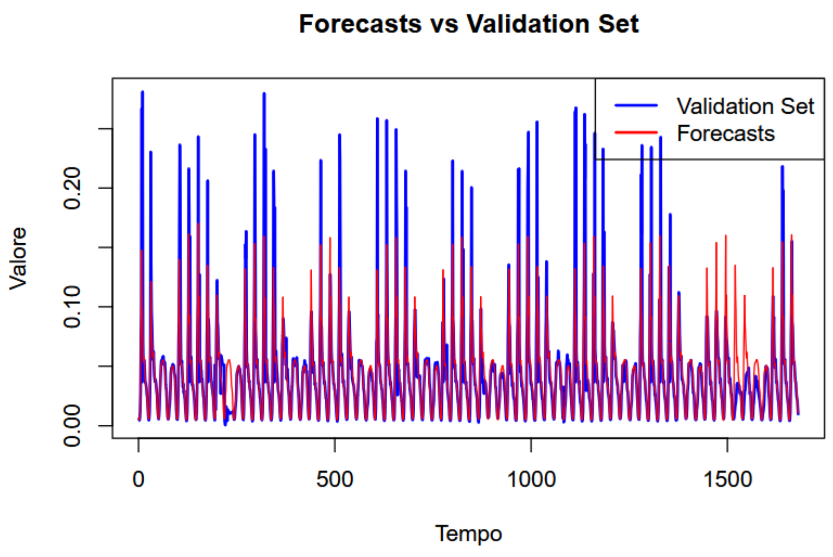


Figure 13: Forecasts plot of ARIMA (3,1,1) (1,1,1)(7)

3.6 Arima (3,1,1) (1,1,1)(7) with dummies

The ARIMA(3,1,1)(1,1,1)(7) model with dummies, as you will notice, does not provide any notable improvements.

The residual results are very similar, which suggests that increasing the order of the Autoregressive (AR) component did not significantly improve the model's performance because they are inline with others and changes are not significant.

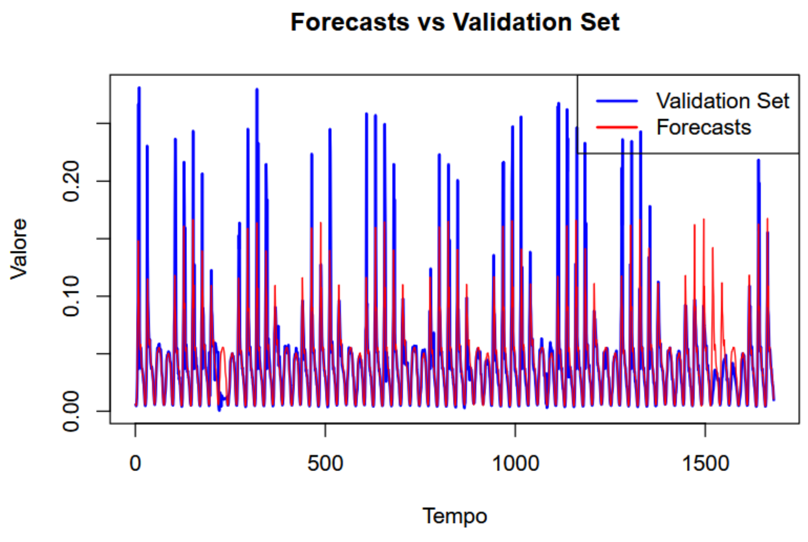


Figure 14: Forecasts plot of ARIMA (3,1,1) (1,1,1)(7) with dummies

Overall, the model seems to provide a good fit, with a **MAE of 0.009554**, which is slightly better than the previous model, though the difference is minimal.

3.7 Deployment

To sum up the results achieved in this section, it's provided a comparison of the performance of different ARIMA models, evaluated with and without the inclusion of dummy variables. The models, with varying (p,d,q) and (P,D,Q) configurations, are assessed based on their error values. The results indicate that the inclusion of dummy variables does not lead to significant improvements in model performance.

Table 3: Model Performance Comparison of ARIMA models

Model	**Normal**	**With Dummies**
ARIMA (3,1,1) (0,1,1)(7)	0.009563	0.009546
ARIMA (3,1,1) (0,1,2)(7)	0.009558	0.009562
ARIMA (3,1,1) (1,1,1)(7)	0.009552	0.009554

The model with the best predictive performance is the first one, ARIMA(3,1,1)(0,1,1)[7], enhanced with the dummies. It achieved an MAE of 0.009546 on the validation set. The model was then retrained using the complete dataset, applying the same approach of training 24 daily time series and then combining them into a single hourly time series. The training data includes information up to 2016-11-30, and the goal was to predict the final month of hourly data for 2016. The plot below shows the last section of the actual time series, followed by the predicted part from the model.

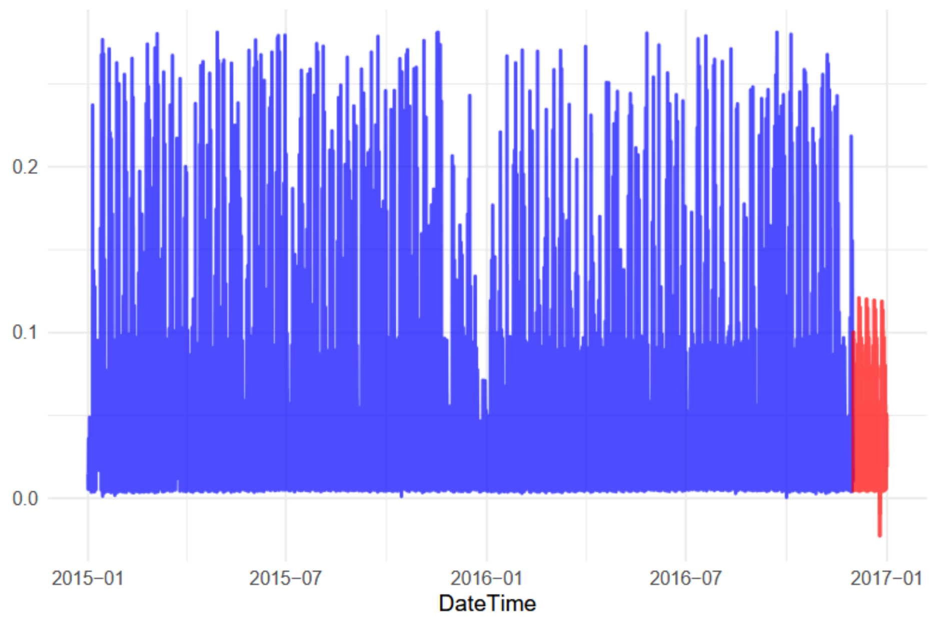


Figure 15: Plot of entire time with ARIMA forecasts

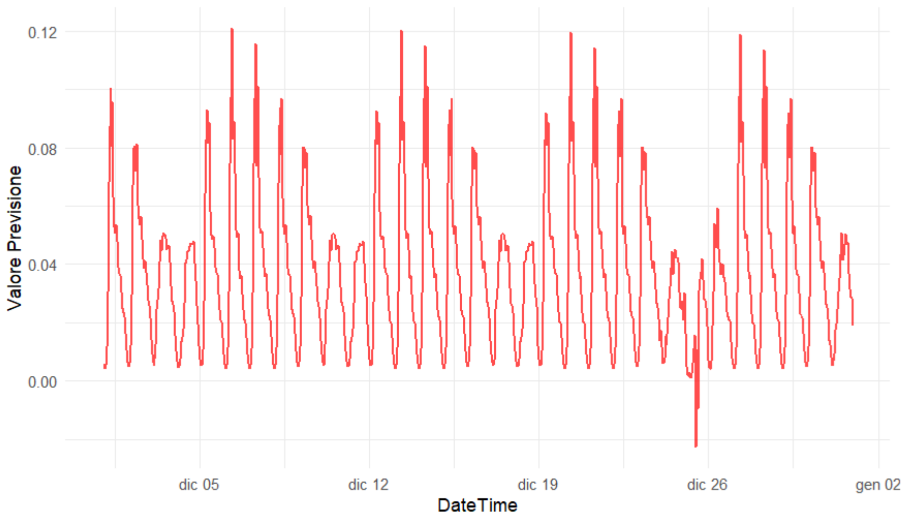


Figure 16: Forecasts plot of ARIMA (3,1,1) (0,1,1)(7) with dummies

4 Unobserved Components Models (UCM)

The Unobserved Components Model (UCM) is a flexible framework for time series analysis that decomposes a series into interpretable components such as trend, seasonality, and irregular effects. This decomposition allows UCMs to model complex dynamics while maintaining a clear understanding of the underlying patterns in the data. UCMs are particularly effective for time series with identifiable structure and components that evolve over time.

In this section, I will discuss the process I followed for selecting the best-performing UCM. The approach involved starting with a simpler baseline model and progressively enriching it with additional components, such as seasonal terms or trend adjustments to evaluate whether these enhancements improved performance.

The data used for these models underwent the same preprocessing steps as described for the ARIMA models, ensuring consistency and comparability between the approaches. However, the execution of the Kalman filter was performed to estimate the unobserved components, refine the model's accuracy and smooth the entire time serie. Each variation of the UCM was evaluated against predefined metrics, and the results guided the selection of the most effective configuration.

This iterative process ensured that the final model was both interpretable and robust, striking a balance between complexity and performance.

Since the residuals of the 24 series are quite similar to one another, only a selection of their graphs is included in the paper to provide a general overview of the workflow, rather than presenting all of them.

4.1 LLT, seasonal dummy(7)

The first proposed model has a LLT (Local Linear Trend) component and a Seasonal Component modeled with dummy variables with a period of 7 days. This was chosen because of the considerations done in the exploration i notice a repeated pattern every week. Typically a model in a state-space model is used for time series analysis because it adapts dynamically over time to capture changes in the underlying level of the series. With a period of 7, these dummies represent the recurring effects associated with each day of the week, capturing consistent daily variations within a weekly cycle.

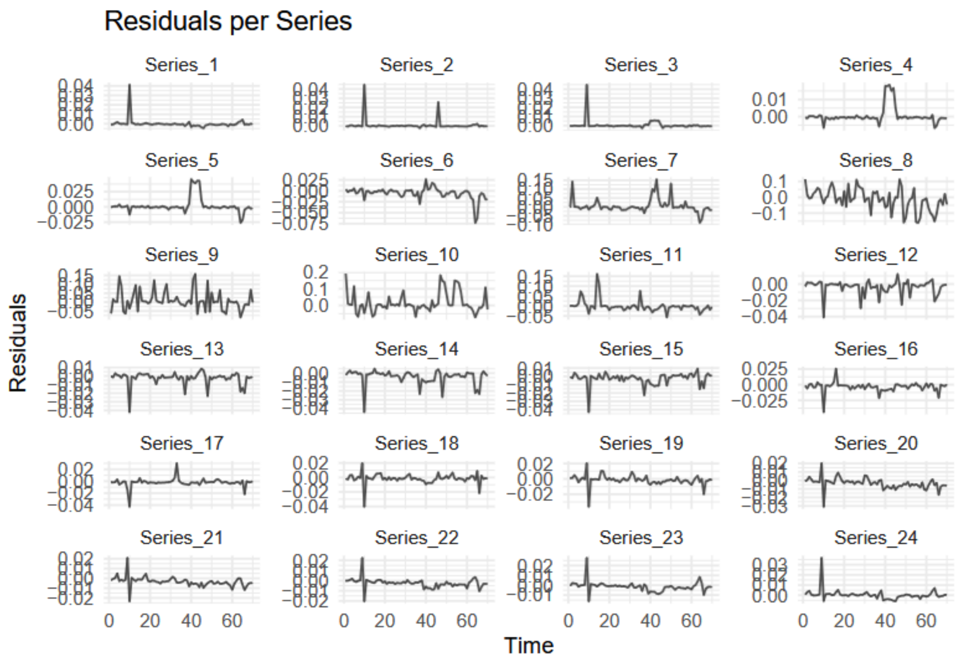


Figure 17: Residuals UCM with LLT, seasonal dummy(7)

The residues of the various series are acceptable and with little error. Compared to the UCMs there seems to be much less noise from the various plots. In some graphs, it can be seen that the model only tends to underestimate some values, while at the busiest times the error starts to rise in positive and negative values, indicating that in some cases it underestimates and in others overestimates the forecast. As mentioned earlier, it is reasonable to consider that during peak periods, there is an increase in traffic, which leads to greater variation in the time series.

The overall prediction is quite good. The plot indicates that the forecasts are closely aligned with the actual data in the validation set, though there are some discrepancies, particularly in the peaks. This suggests the model captures the general pattern but may underperform in predicting extreme values. Despite this, the peaks of forecast are better in comparison with the UCMs model. In this case I obtained a **MAE of 0.009738**.

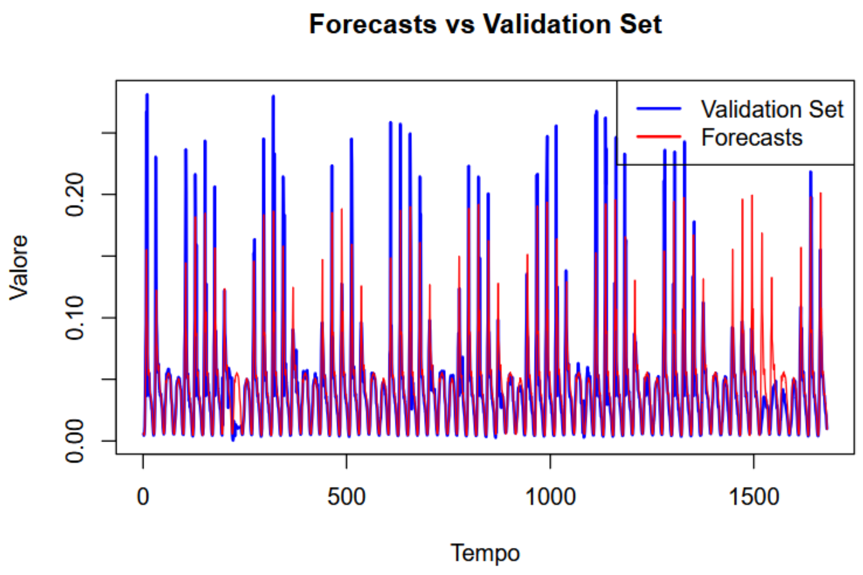


Figure 18: Forecasts plot of UCM with LLT, seasonal dummy(7)

4.2 LLT, seasonal dummy(7) with dummies

I make an attempt with the use of dummy variables. So, the LLT (Local Linear Trend) model with seasonal dummy(7) incorporates dummy variables to capture weekly seasonality, in particular on special days of the year.

The residuals remain approximately the same probably also due to the very low scale of values but there are no significant changes. Again, the forecasts improve but the peaks are further underestimated despite the forecast still being good with a **MAE of 0.009588**.

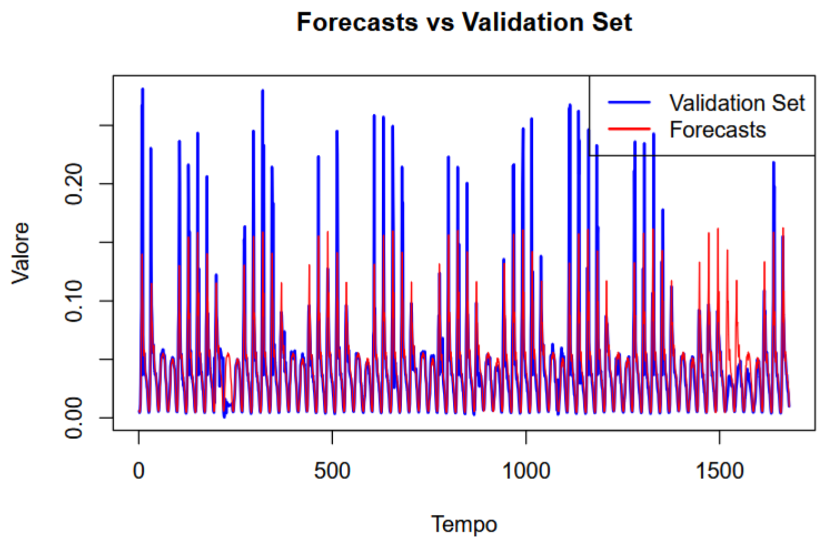


Figure 19: Forecasts plot of UCM with LLT, seasonal dummy(7) and dummies

4.3 LLT, trigonometric seasonality(7)

The second proposed model has a a LLT (Local Linear Trend) component and, again, a seasonal component with a period of 7 days, but this time of trigonometric type, modeled with sinusoids.

The residues of the various series are acceptable and with little error but are approximately the same of the previous. This suggest that are not very different. Again, compared to the UCMs there seems to be much less noise from the various plots. In some graphs, it can be seen that the model underestimates and in others overestimates the forecast.

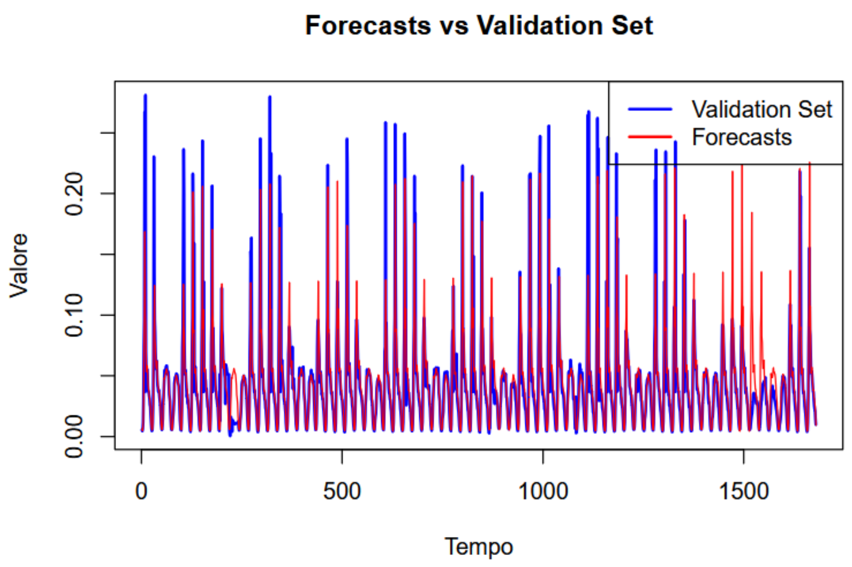


Figure 20: Forecasts plot of UCM with LLT, trigometric seasonality (7)

The overall prediction is satisfactory. The graph shows that the forecasts closely follow the actual data in the validation set, although some differences are noticeable, particularly around the peaks. In this case, the performance is reflected by a **MAE of 0.009792** which is little worse than previous model.

4.4 LLT, trigonometric seasonality(7) with dummies

I explore the use of trigonometric seasonality. The LLT (Local Linear Trend) model with trigonometric seasonality(7) represents weekly patterns through sine and cosine functions, allowing for a smooth and continuous representation of cyclic variations over the weekly period.

The residuals remain largely unchanged, likely due to the very small scale of the values, with no significant differences observed. While the forecasts show improvement, the peaks are still underestimated. Nonetheless, the overall prediction remains strong, achieving a **MAE of 0.00958**.

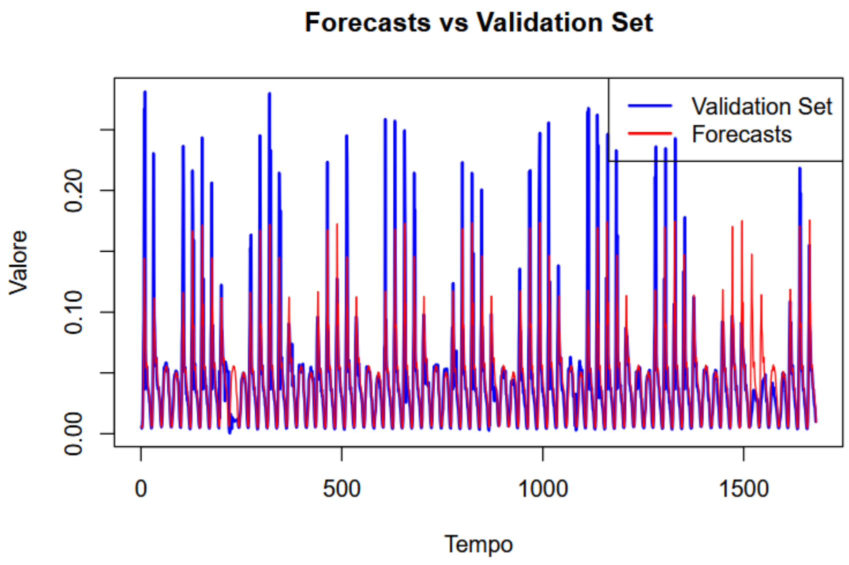


Figure 21: Forecasts plot of UCM with LLT, trigonometric seasonality(7) and dummies

4.5 LLT, seasonal dummy (7) and one trigonometric harmonic (365)

Since the model with the lowest mae was the first one with LLT, seasonal dummy(7) I decided to add a trigonometric seasonal component with a 365-day period. I repeated the test with different numbers of harmonics until I arrived at the final result. For convenience i report every results that i obtained.

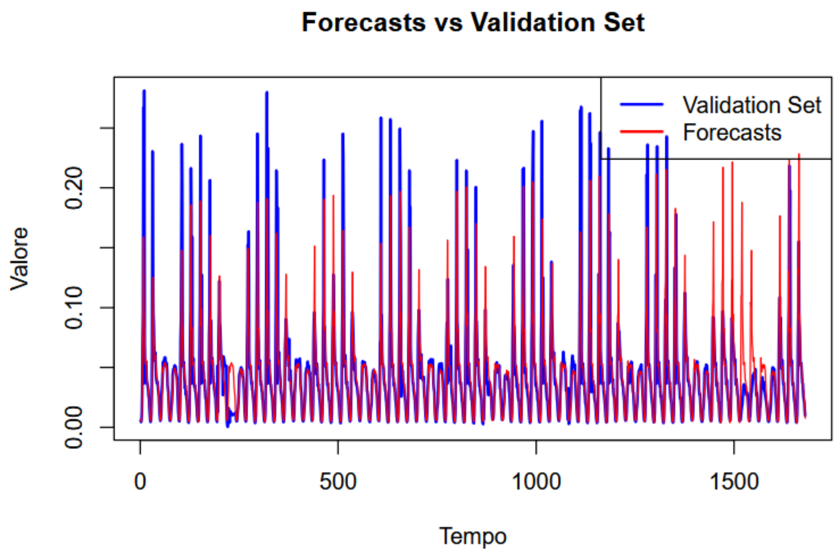


Figure 22: Forecasts plot of UCM with LLT, seasonal dummy(7), one trigonometric harmonic (365)

At first glance, it may seem that the result is not bad at all. However, one can see that there is a sort of upward trend of the peaks tending to increase, which makes me think that this is not the best solution for our series. The result is a **MAE of 0.01** which is greater than everyones.

4.6 LLT, seasonal dummy (7) and 2 harmonics (365)

I decide to investigate further with an another attempt with 2 harmonics. The forecasts obtained are the following:

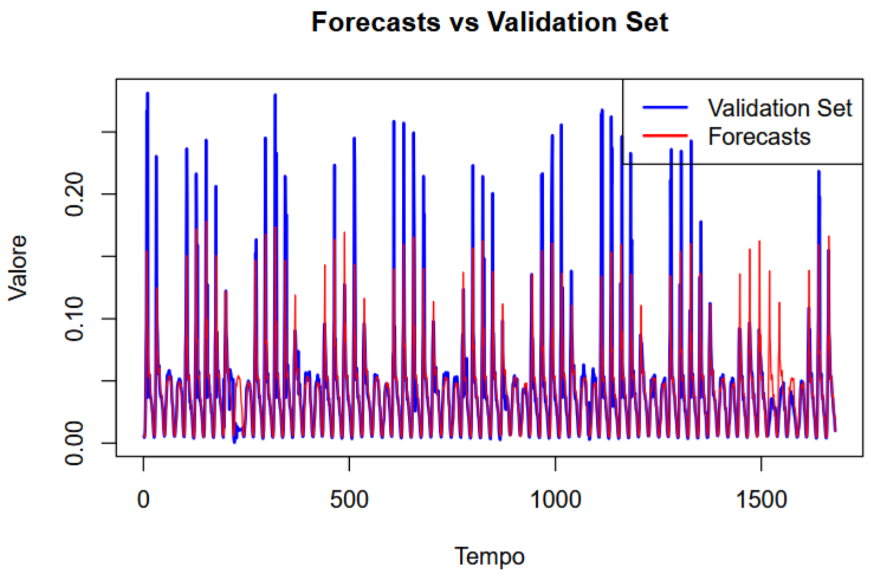


Figure 23: Forecasts plot of UCM with LLT, seasonal dummy(7), 2 trigonometric harmonic (365)

Using the 2 harmonics to model annual seasonality seems to achieve a good result compared to the previous model. No particular trends are noticeable, but the forecast trend seems to be correct despite the peaks always being underestimated. With this model I achieve a **MAE of 0.009514** which is the lowest so far.

4.7 LLT, seasonal dummy (7) and 3 harmonics (365)

The last but not least attempt I made was to further increase the number of harmonics arriving to 3 harmonics. The predictions obtained are as follows:

As in the first case, we have that the trend takes over everything else in the forecasts, resulting in completely wrong forecasts with an erroneous trend increasing over time and increasing the amplitude of the estimate causing larger and larger errors. The result is the worst obtained until now: **MAE of 0.0145**

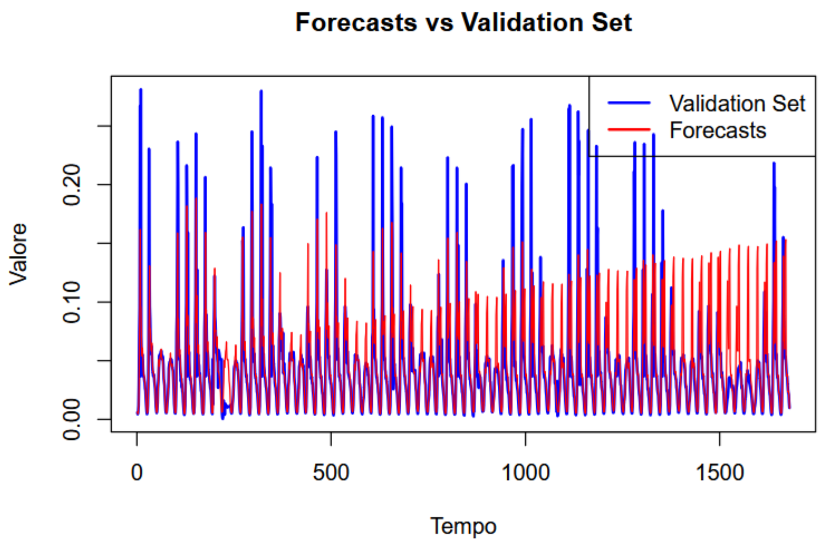


Figure 24: Forecasts plot of UCM with LLT, seasonal dummy(7), 3 trigonometric harmonic (365)

4.8 LLT, seasonal dummy (7) and 2 harmonics (365) with dummies

The very last attempt I made once I found the correct number of harmonics was to insert dummy variables into the model to see if the predictions improved further.

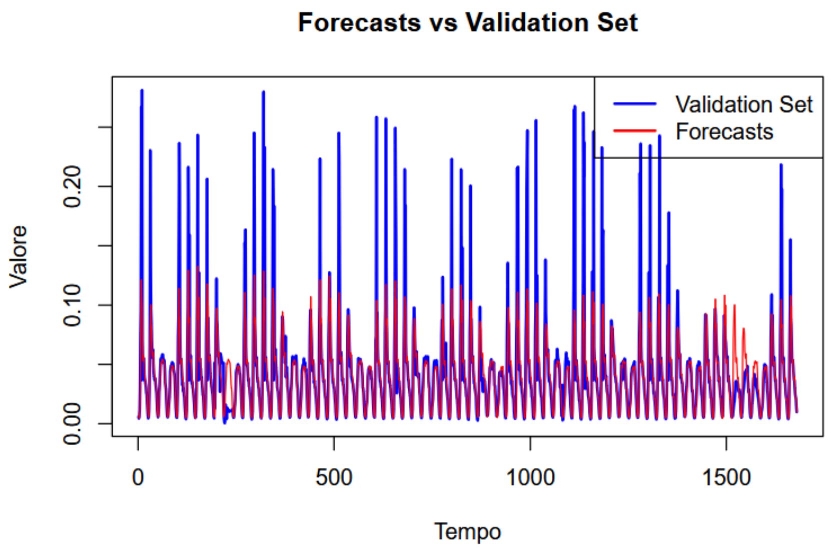


Figure 25: Forecasts plot of UCM with LLT, seasonal dummy(7), 3 trigonometric harmonic (365) and dummies

The result is better so far with a **MAE of 0.0095**. The model fits the series well. Although it underestimates the peaks it commits less error due to the presence of the dummy variables. In the final part of the series, the series does not have the peaks that it usually has, so we get less error.

4.9 Deployment

To sum up the results achieved in this section, it's provided a comparison of the performance of different UCM models, evaluated with and without the inclusion of dummy variables. The models, utilizing various seasonal components such as seasonal dummies and trigonometric seasonality with different harmonics, are assessed based on their error values. The analysis shows that the inclusion of dummy variables results in minor improvements for some models, while certain configurations, such as the use of three harmonics, lead to higher errors, and others do not provide any valid results.

Table 4: Model Performance Comparison of UCM models

Model	**Normal**	**With Dummies**
LLT, seasonal dummy (7)	0.009738	0.009588
LLT, trigonometric seasonality (7)	0.009792	0.00958
LLT, seasonal dummy (7) and one trigonometric harmonic (365)	0.01	NONE
LLT, seasonal dummy (7) and 2 harmonics (365)	0.009514	0.0095
LLT, seasonal dummy (7) and 3 harmonics (365)	0.0145	NONE

The model with the best predictive performance is the last one, LLT, seasonal dummy (7) and 2 harmonics (365), enhanced with the dummies. It achieved an MAE of 0.0095 on the validation set. The model was subsequently retrained using the full dataset, following the same method of training 24 daily time series and then merging them into one continuous hourly time series. The training data extends up to 2016-11-30, with the objective of forecasting the final month of hourly data for 2016. The plot below illustrates the last portion of the actual time series, followed by the predicted segment from the model.

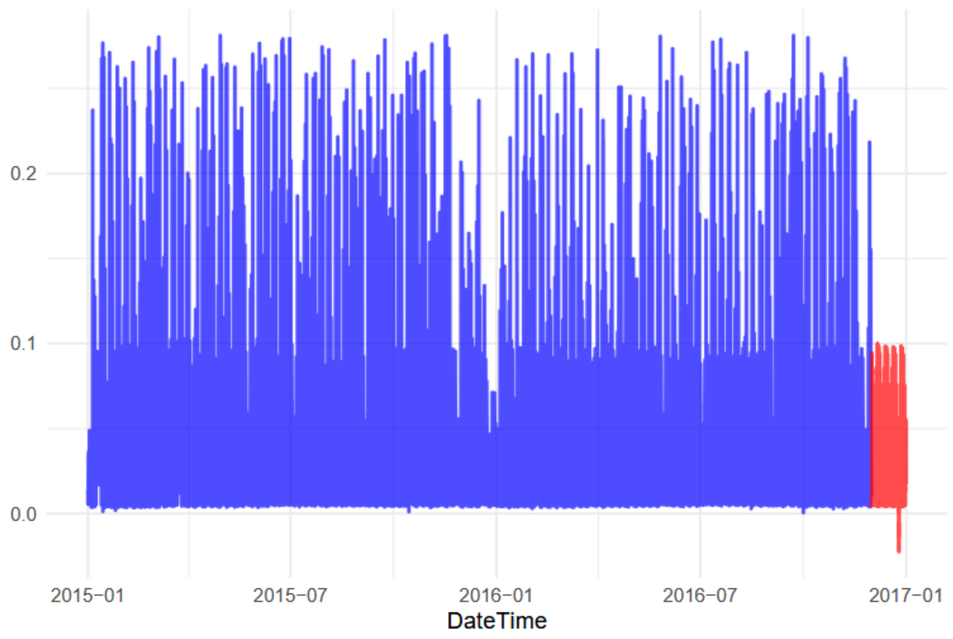


Figure 26: Plot of entire time with UCM forecasts

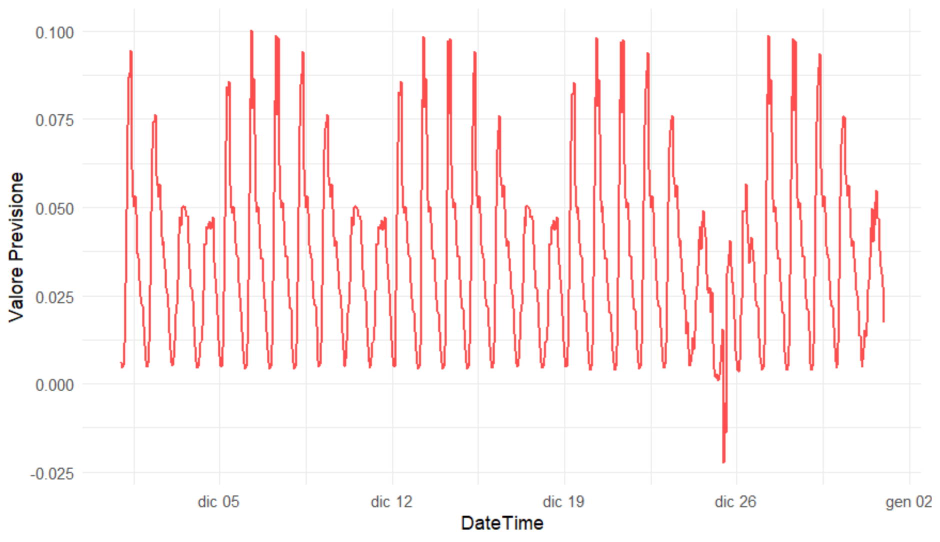


Figure 27: Forecasts plot of UCM model with LLT, seasonal dummy (7) and 2 harmonics (365) with dummies

5 Machine Learning (ML) Models

Machine learning in time series analysis focuses on leveraging algorithms to identify patterns, trends, and dependencies in sequential data collected over time. It involves tasks such as forecasting future values, detecting anomalies, or classifying sequences. Unlike traditional statistical methods, machine learning models can handle complex, non-linear relationships and integrate diverse features like seasonality, external variables, and lags. Non-parametric estimation techniques are often employed to make fewer assumptions about the underlying data distribution, providing flexibility in modeling diverse time series patterns for applications in finance, healthcare, energy, and beyond. In my project I decided to use RandomForest, XGBoost and K-Nearest Neighbors (KNN) models. Unlike the statistical models seen above, machine learning models **need feature engineering** to perform adequately. In fact, I decided to add some variables to the dataset such as:

- **Lag-X**: variable containing the time series value of the previous lag
- **Lag-X7**: variable containing the value of the time series of the previous week
- **Diff-X**: difference between the current value and the previous lag
- **Diff-X7**: difference between the current value and the previous week's value
- **DayOfWeek**: ordinal numeric value referring to the day of the week (from 1 to 7)
- **DayOfYear**: ordinal numeric value referring to the day of the year (from 1 to 365)

The logarithmic transformation was applied to the series to make them as additive as possible.

5.1 Random Forest

The first model used is Random Forest, which is an ensemble learning method primarily used for classification and regression tasks. It builds multiple decision trees during training and combines their outputs to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the data using a technique called bootstrap sampling, and features are randomly selected at each split to ensure diversity among trees. By aggregating predictions (majority vote for classification or averaging for regression), Random Forest achieves robust and reliable results, making it effective for handling complex, non-linear relationships and large datasets with minimal tuning.

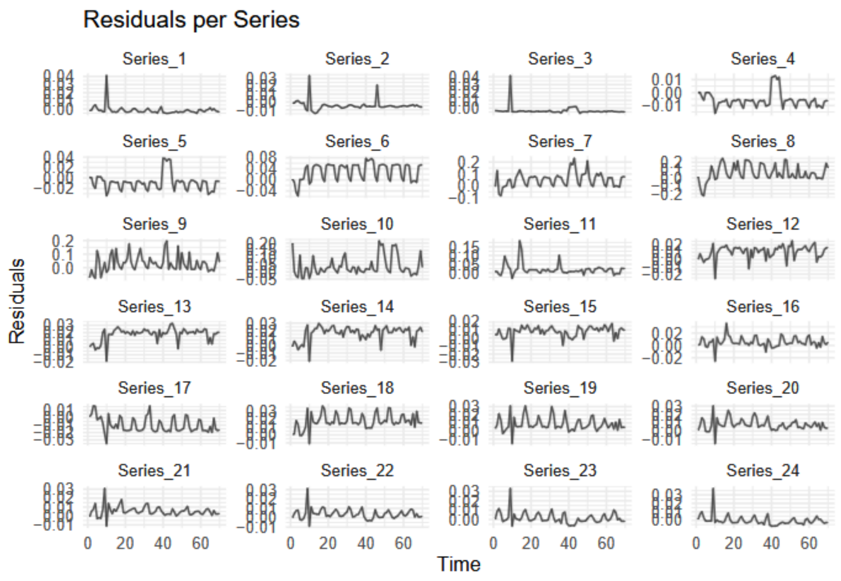


Figure 28: Residuals of Random Forest

The residuals seem a little more scattered in comparison with the UCM models. At peak hours, traffic is underestimated as can be seen from the graphs above, showing a fluctuating effect for each series. In short, a good model was not obtained.

In fact, the forecast initially overestimates the traffic and then continues to make forecasts with the same magnitude throughout the rest of the series. The model probably failed to understand the seasonality of the series. This model achieved a **MAE of 0.019579** which is very large in comparison with the previous models.

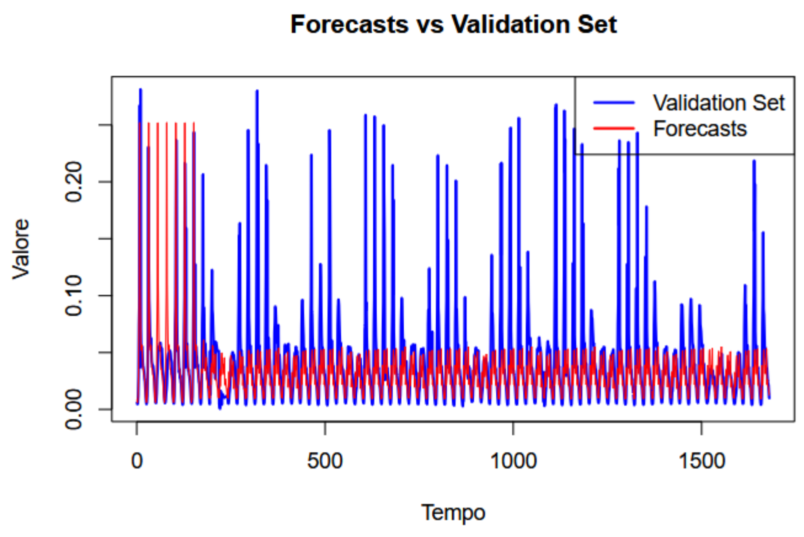


Figure 29: Forecasts plot of Random Forest

5.2 Random Forest with dummies

I decided to give it a try with the dataset containing the dummy variables to see if it can learn better.

The residuals are approximately the same previously seen without dummies. In the following there are the results of the forecasts but the situation doesn't change. Probably this is not the good model due to the fact that dummies are practically useless despite the **MAE of 0.01922** increases a little bit.

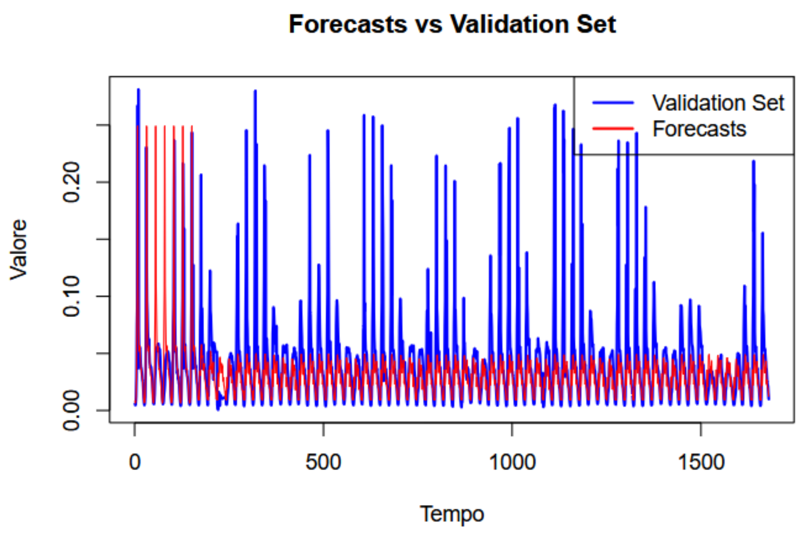


Figure 30: Forecasts plot of Random Forest with dummies

5.3 Extreme Gradient Boosting (XGBoost)

XGBoost (Extreme Gradient Boosting) is a powerful and scalable machine learning algorithm designed for supervised learning tasks like classification and regression. It is based on the gradient boosting framework, where models are built sequentially, and each new model corrects the errors of the previous ones. XGBoost introduces enhancements such as regularization to prevent overfitting, efficient handling of missing data, and parallelized tree construction, making it faster and more robust than traditional gradient boosting methods.

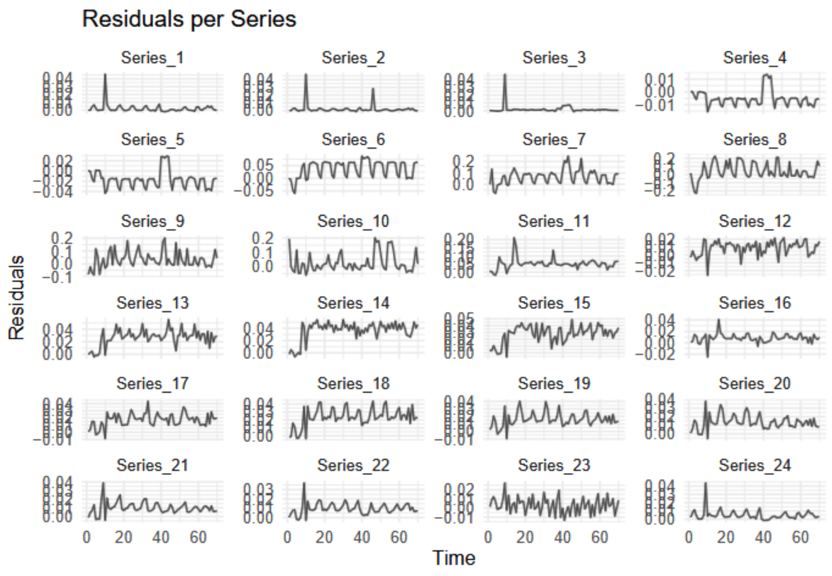


Figure 31: Residuals of XGBoost

The residuals seem approximately the same of the previous model due to the fact that together are based on tree construction. Again, during peak hours, traffic is underestimated. In summary, the model did not achieve satisfactory performance.

The forecast initially overestimates traffic and then maintains predictions with a consistent magnitude for the remainder of the series. This suggests that the model likely failed to capture the seasonality inherent in the data. This model delivered the poorest performance, with a **MAE of 0.024698**, which is significantly higher compared to the previous models.

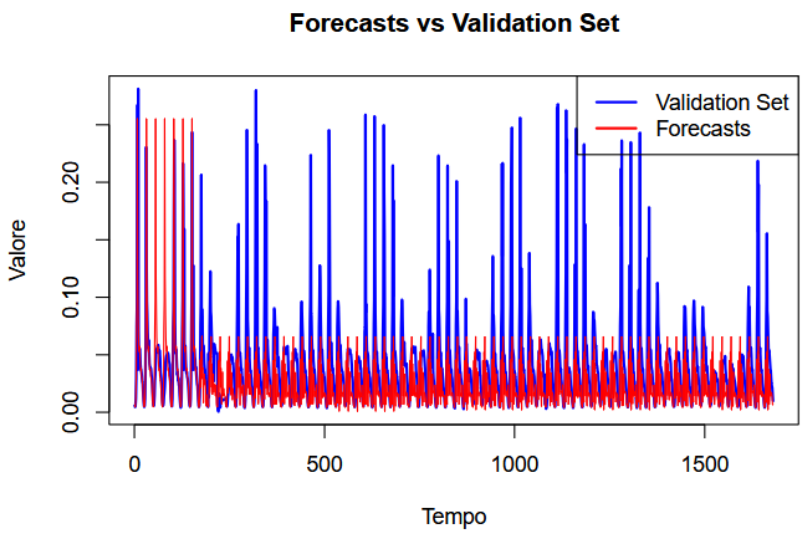


Figure 32: Forecasts plot of XGBoost

5.4 Extreme Gradient Boosting (XGBoost) with dummies

I decided to experiment with the dataset containing dummy variables to determine if it could improve the model's learning.

However, the residuals are similar to those observed without the dummy variables. The forecast results, shown below, indicate that the situation remains unchanged. This is likely not an optimal model, as the dummy variables appear to be largely ineffective. Additionally, the **MAE of 0.01931** shows a slight increase, further confirming its limitations.

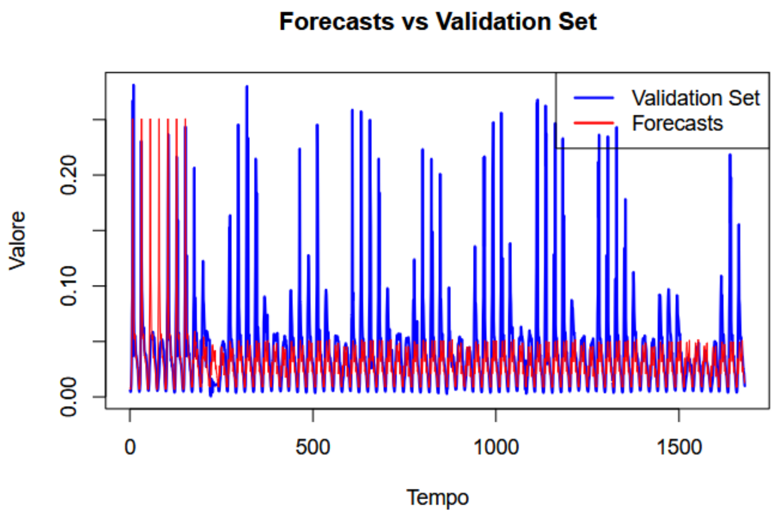


Figure 33: Forecasts plot of XGBoost with dummies

5.5 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) model is a simple yet effective machine learning algorithm used for classification and regression tasks. It operates by identifying the k closest data points (neighbors) to a given input based on a chosen distance metric, such as Euclidean or Manhattan distance. For classification, the model assigns the majority class among the neighbors, while for regression, it averages their values. KNN is non-parametric, making it flexible for various data distributions, but it can be computationally expensive for large datasets due to the need to calculate distances for each prediction.

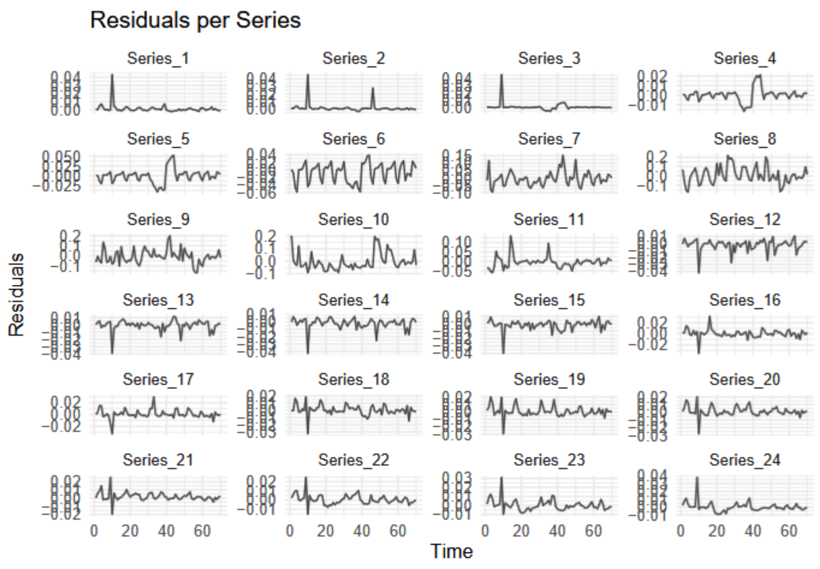


Figure 34: Residuals of KNN

The residuals turn out to be better than all other Machine Learning models seen so far. The magnitude is small. However, like all previous models, during peak hours, the traffic is underestimated as can be seen from the graphs above. Nevertheless, it seems that residuals are less scattered. Let's see what the forecast looks like.

The graph shows how high peaks are initially predicted and then smoothen out depending on seasonality. Compared to previous models, it seems to follow a certain seasonality typical of this time series. This model obtained a **MAE of 0.013559** which is the lowest so far for machine learning models.

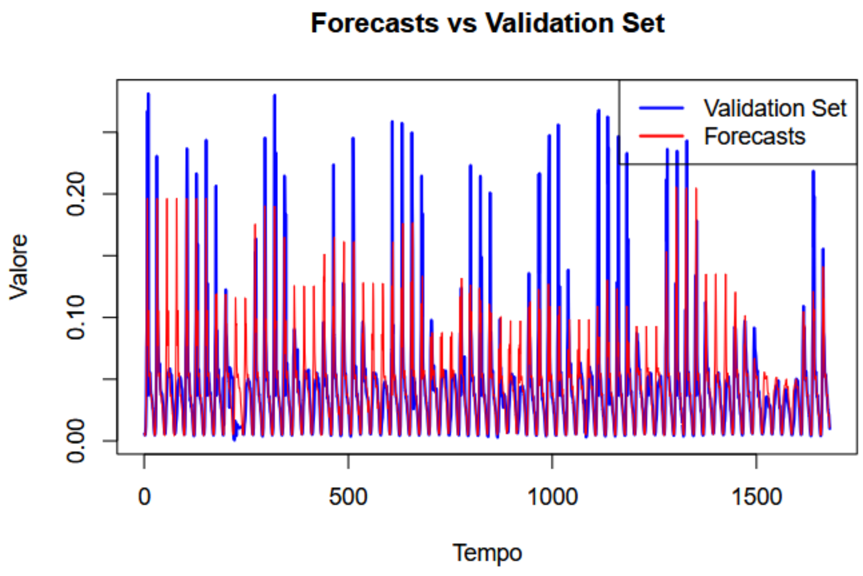


Figure 35: Forecasts plot of KNN

5.6 K-Nearest Neighbors (KNN) with dummies

I decided to experiment with the dataset containing the dummy variables to determine whether it would improve model learning.

However, the residuals are similar to those observed without the dummy variables. The results of the predictions, shown below, indicate that the situation remains roughly the same but with slight improvement in some places. The **MAE of 0.01353** also says that we have a small improvement in the predictions.

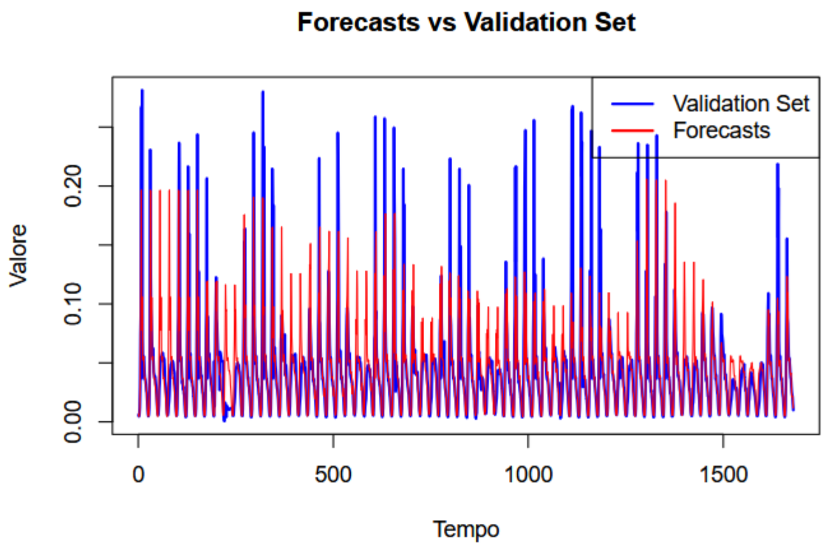


Figure 36: Forecasts plot of KNN with dummies

5.7 Deployment

To sum up the results achieved in this section, it's provided a comparison of the performance of different machine learning models, evaluated with and without the inclusion of dummy variables. The models, including Random Forest, XGBoost, and KNN, are assessed based on their error values. The results indicate that the inclusion of dummy variables leads to improvements in the performance of XGBoost and Random Forest, while the effect on KNN is minimal.

Table 5: Model Performance Comparison of ML models

Model	**Normal**	**With Dummies**
Random Forest	0.019579	0.01922
XGBoost	0.024698	0.01931
KNN	0.013555	0.01353

The model with the best predictive performance is the last, K-Nearest Neighbors (KNN), enhanced with dummies. It achieved an MAE of 0.01353 on the validation set. The model was retrained using the complete dataset, employing the same approach of training 24 individual daily time series and combining them into a unified hourly time series. The training data spans up to 2016-11-30, aiming to predict the hourly data for the final month of 2016. The plot below depicts the final segment of the actual time series, followed by the model's predicted values.

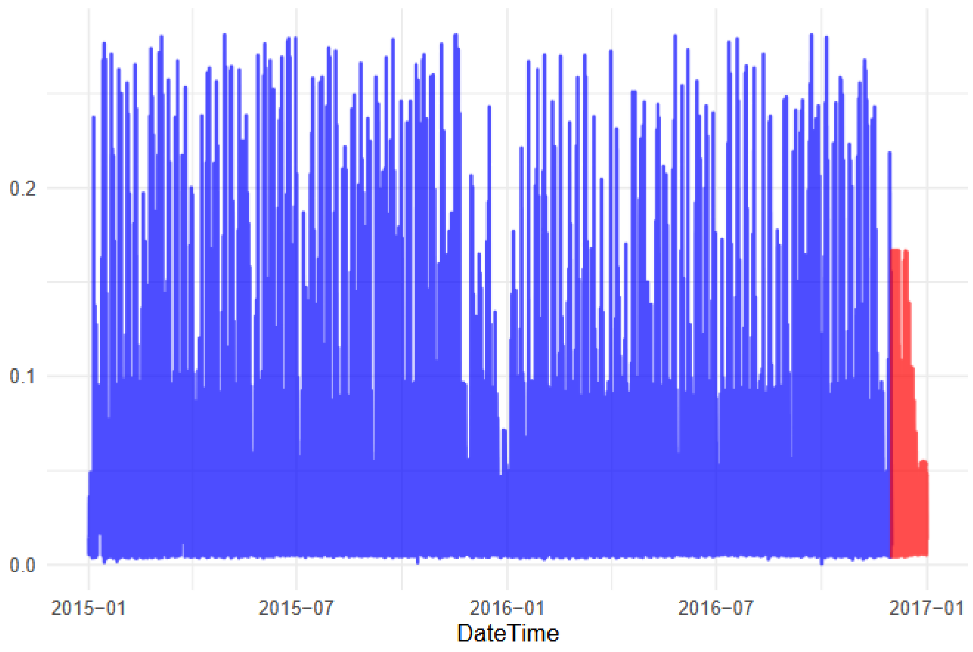


Figure 37: Plot of entire time with KNN forecasts

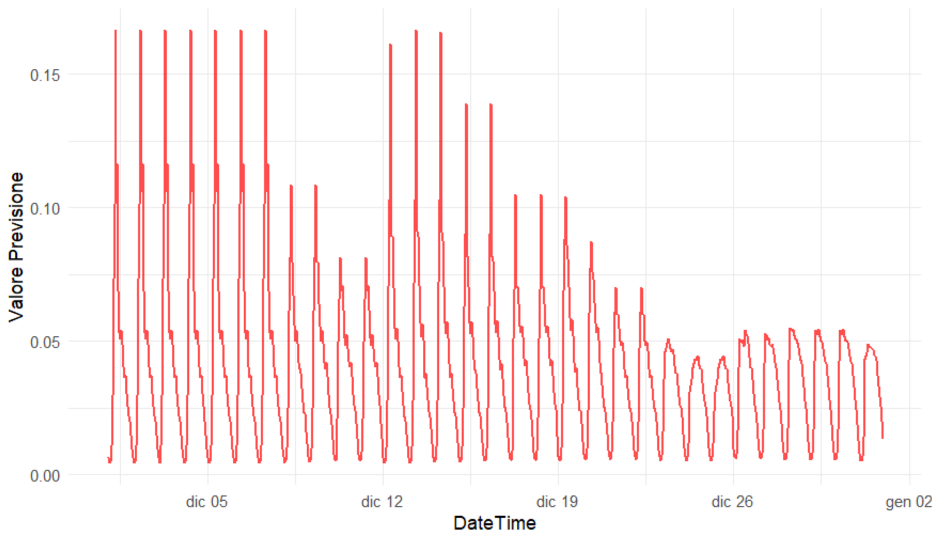


Figure 38: Forecasts plot of ML model with KNN and dummies

6 Conclusions

To summarise, the objective of the project was the selection of the best performing ARIMA, UCM and Machine Learning models and their application to make predictions on time series values over the required time period.

In the exploration phase, I carried out an initial inspection of the data, ensuring its consistency by removing outliers and zeros. To model multi-seasonality, I decided to divide the hourly series into 24 daily series. For each of them, I verified and applied the stationarity of the process through Box-Cox transformations and according to the Augmented Dickey-Fuller test. Through various plots and autocorrelograms made during the exploration and development phase, I noticed a relevant feature of the time series, which emphasised its weekly seasonality.

After developing the models and plotting their predictions, we can still conclude that all models were able to successfully capture this form of seasonality. The error measure taken into account for the comparison between the models was the Mean Absolute Error (MAE). For all models, I tested different parameter combinations and described the three most promising, which were then calculated from the validation set data (10%) I had available. The most promising configurations were tested again with the addition of the main holiday dummy variables.

The three best models are the following:

- **ARIMA(3,1,1)(0,1,1)[7]** with dummies and with a **MAE of 0.009546**
- UCM: **LLT, seasonal dummy (7) and 2 harmonics (365)**, with the dummies with a **MAE of 0.0095**
- ML: **K-Nearest Neighbors (KNN)** with dummies, with a **MAE of 0.01353**

In conclusion, the models demonstrate reasonably good performance overall. They effectively capture the general trend and seasonal patterns of the time series in all cases.

However, a common limitation across all models is their inability to accurately predict the peaks in the original series. These discrepancies suggest that while the models are capable of understanding the broader structure of the data, they may lack the granularity needed to account for abrupt changes or extreme values.

Future improvements could involve exploring advanced techniques, such as hybrid models or models specifically designed to handle peak detection, and incorporating additional features that might better explain these variations. Despite these limitations, the models provide valuable insights and serve as a strong foundation for further refinement.