Streaming Data Management and Time Series Analysis

Individual Report

Forecasting Project:

forecasts evaluation using ARIMA, UCM, and Machine Learning models

Author:
Tariq Baghrous - 904027- t.baghrous@campus.unimib.it

Master's Degree in Data Science
Università degli Studi di Milano-Bicocca
2023/2024 Academic Year

Contents

1 Introduction

In the realm of time series analysis and predictive modeling, the exploration and understanding of historical data play a crucial role in extracting meaningful insights and making informed decisions. This project is dedicated to test a diverse range of predictive models available for time series analysis on R in order to effectively estimate the average number of days needed to close the requests that are closed in a specific day.

The dataset provided underwent a preparation procedure, ensuring its efficacy and optimization in alignment with the project's goals. Indeed The models employed require a specific type of data modeling to utilize them effectively and generate accurate forecasts.

Following the data preprocessing, the project shifted its focus towards testing predictive models from the following categories:**ARIMA**, **UCM** and **Machine Learning**. The analysis period encompass the timeframe from April 1^{st}, 2007 to March 31^{st}, 2015, while the forecast projected values up to November 7^{th}, 2015.

Lastly, the best forecasting model for each category is assested by the computation of their MAE, providing a comprehensive understanding of how to interpret and compare models with each other effectively.

2 Dataset and Preprocessing

The dataset consists of a "*comma-separated values*" file that contains 3009 observations and three attributes:

- **date:** string with date in format yyyy-mm-dd;
- **weekday:** string with the name of the weekday;
- **ave_days:** floating point number representing the average number of days needed to close the requests that were closed that day.

After loading the dataset, I conducted an initial analysis to check for duplicates and missing values (NAs). The analysis revealed that there were no duplicate entries in the dataset. However, I identified 202 missing values in the column "ave_days". Notably, the other two columns in the dataset did not contain any missing values.

To further understand the missing data, I examined the annual distribution of the missing values in the "ave_days" column. The distribution is as follows:

Year	Number of Missing Values
2007	33
2008	23
2009	14
2010	20
2011	27
2012	26
2013	24
2014	29
2015	6

Table 1: Annual distribution of missing values in the "ave_days" column

From this distribution, it is evident that the missing values are uniformly distributed across the years (for 2015 there ar eonly 3 months), so the next step is to analyze the distribution of missing values by weekday:

Weekday	**Number of Missing Values**
Thursday	4
Friday	4
Saturday	6
Sunday	167
Monday	20
Tuesday	1
Wednesday	0

Table 2: Distribution of missing values by day of the week in the "ave_days" column

We can observe that the majority of the missing values (167 out of 202) occur on Sundays. This disproportionate number of missing values on Sundays suggests a systemic issue related to data collection that could be related to company closures.

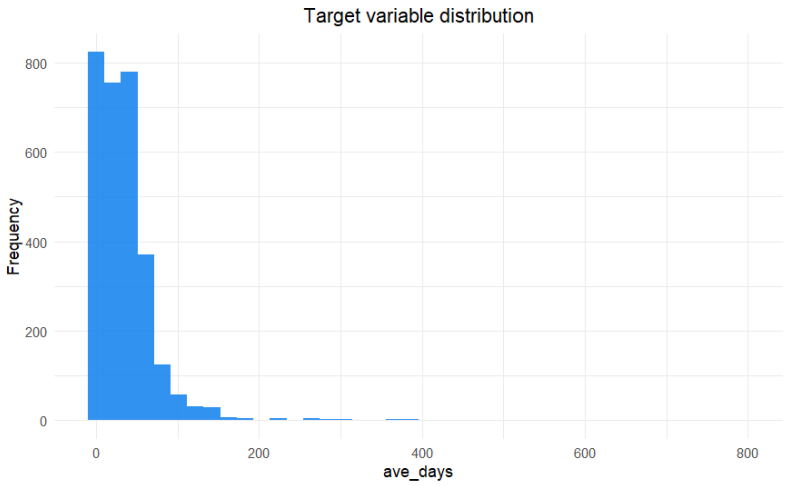
In conclusion, missing values were addressed by replacing them with zeros. This imputation method ensures that the dataset remains complete and allows for further analysis without the distortion caused by missing data.

In order to mitigate the impact of missing data on Sundays, I introduced a new column named **dummy_sunday** into the dataset. This column serves as a binary indicator: it takes the value 1 when the corresponding weekday is Sunday, and 0 otherwise. The purpose of *dummy_sunday* is to give more weight to the *ave_days* values recorded on that day.

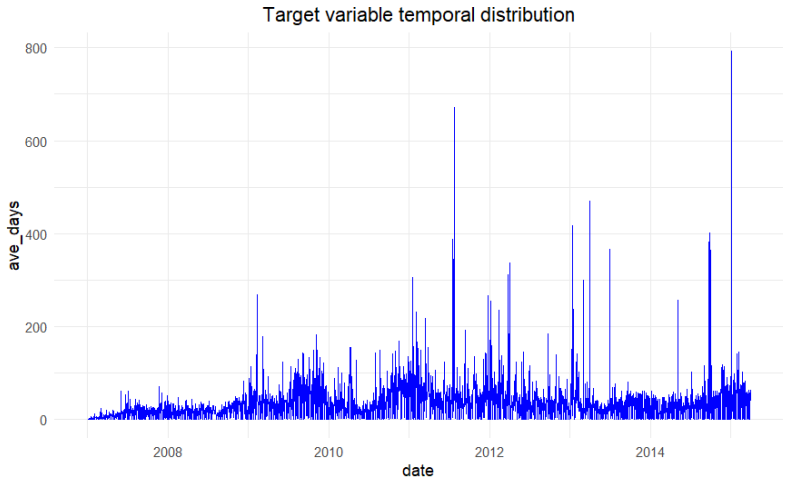
3 Data Exploration

After preprocessing, a phase of data exploration is essential to better understand the dataset and its characteristics. This phase includes visualizing variable distributions, analyzing temporal trends and seasonal patterns.

First we analyze the distribution of the target variable *ave_days*:

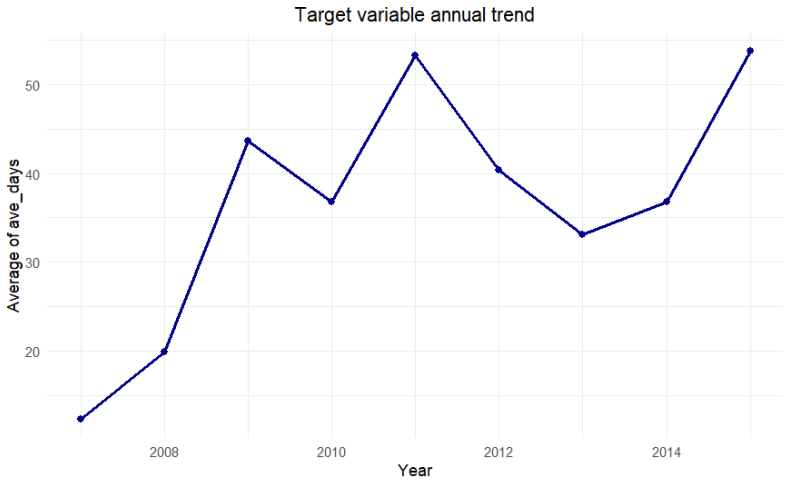


A significant proportion of these values are clustered close to 0, which suggests that the data may benefit from further transformation to avoid negative forecasts.



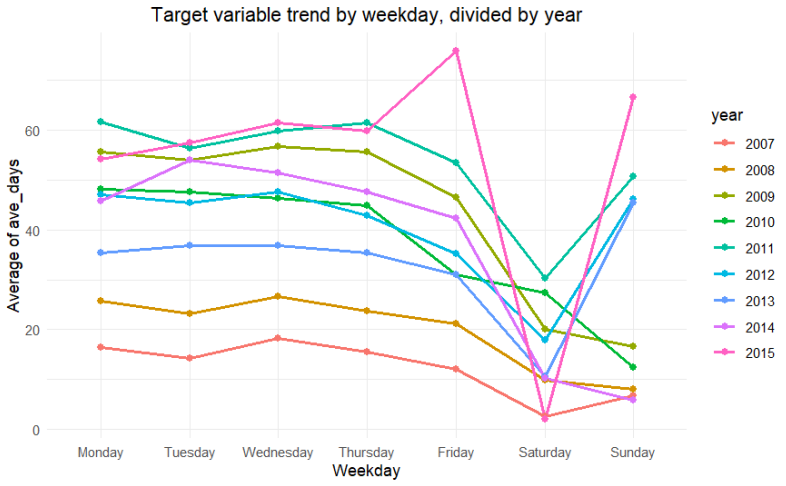
The overall temporal graph in the figure above reveals the presence of sporadic spikes, which may indicate the presence of outliers in the data.

Furthermore, to get more seasonal details we plot the data annually aggregated:



We can notice a general increase until 2011, followed by a decline until 2013 and then a subsequent rise till 2015.

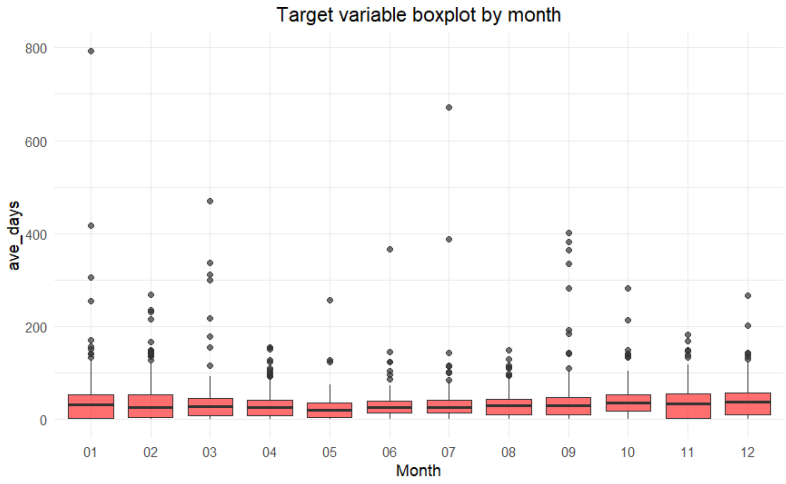
Moreover, analyzing the *ave_days* trend by day of the week segmented by year, highlights notable observations:

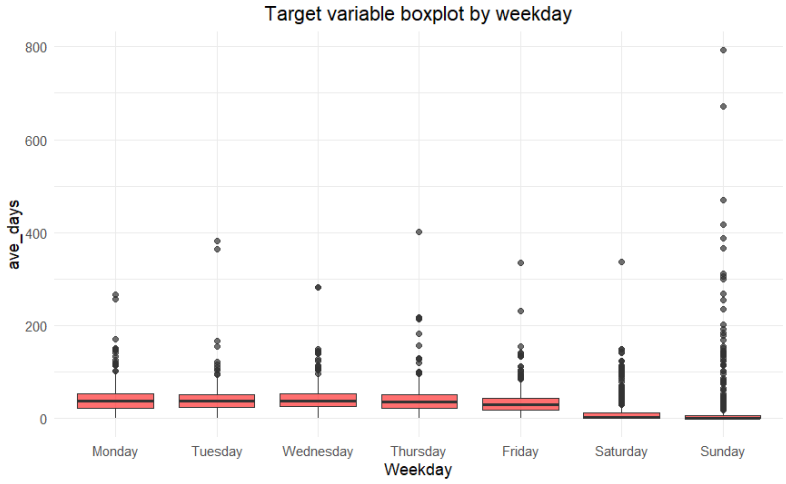


Specifically, Saturdays consistently exhibit the lowest values throughout the years, suggesting a predictable decrease in *ave_days* towards the end of the week. On the other hand, Sundays display high variability, characterized by fluctuating *ave_days* values, which further underlines the particularity of this day.

3.1 Outliers

To further understand the outliers in the *ave_days* variable, boxplots were generated based on months and weekdays:

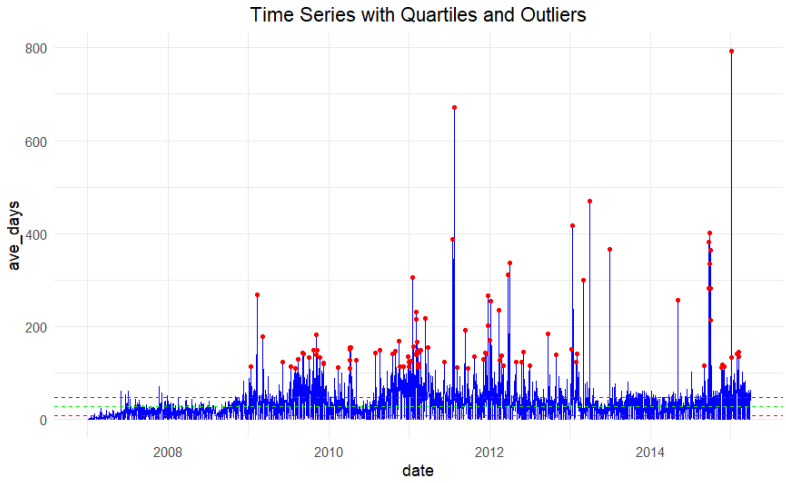




The boxplots categorized by months did not reveal any significant pattern between the months, suggesting that monthly factors do not significantly impact the *ave_days* values.

In contrast, the boxplots categorized by weekdays shows that the majority of the outliers are concentrated on weekends, particularly on Sundays. This high concentration of outliers on Sundays aligns with the earlier observation of high variability in *ave_days* values on this day.

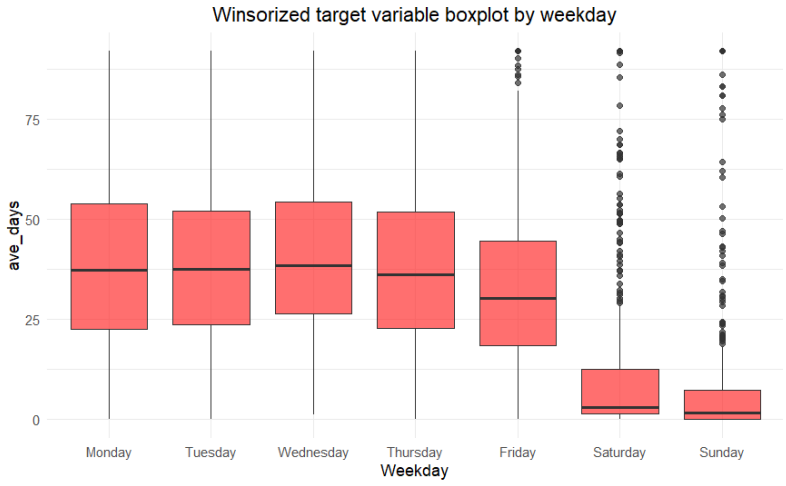
In order to identify the outliers, we compute the quartiles and determine the upper/lower bounds:



Values falling outside these bounds were classified as outliers. Through this process, we identified 102 outliers in the dataset.

To address these outliers, we applied **Winsorization**, a technique used to limit the influence of extreme values in the data. In this case, values in the *ave_days* column that exceed the 95th percentile are replaced with the 95th percentile value itself, while values below (0th percentile) remain unchanged.

The post-treatment boxplot below indicate that the previously identified outliers have been successfully treated, as evidenced by the reduced presence of extreme values.



3.2 Stationarity

To assess the stationarity of the time series data, I performed two statistical tests: the **Augmented Dickey-Fuller (ADF)** test and the **Kwiatkowski-Phillips-Schmidt-Shin (KPSS)** test.

- **ADF:** test used to check the null hypothesis that a unit root is present in a time series dataset. In other words, it tests whether the time series is non-stationary. The alternative hypothesis is that the time series is stationary;
- **KPSS:** test used to check the null hypothesis that a time series is stationary around a deterministic trend (level stationarity). The alternative hypothesis is that the series is non-stationary.

Statistic	**Value**
Dickey-Fuller	-5.9139
p-value	0.01
Alternative hypothesis	stationary
KPSS Level	5.8836
p-value	0.01

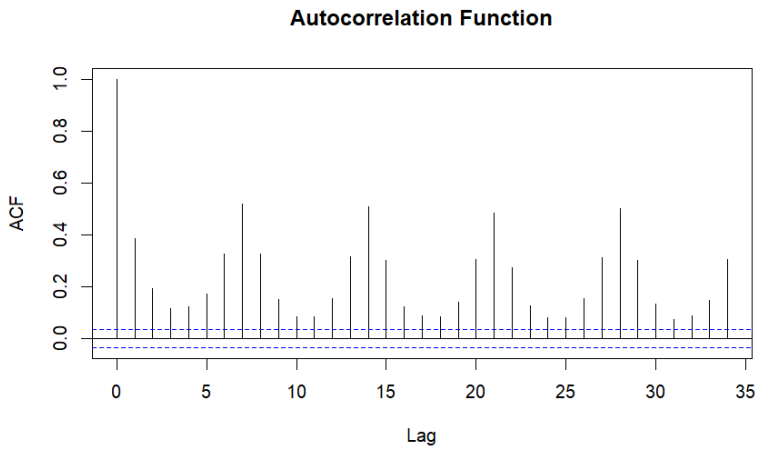
Table 3: Test statistics for Dickey-Fuller and KPSS tests

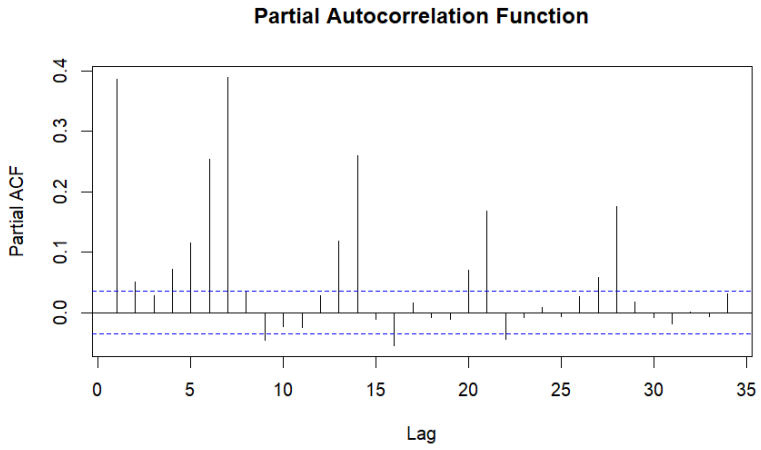
The ADF test shows a p-value of 0.01 (which is less than the typical significance level of 0.05), so we reject the null hypothesis of a unit root. This suggests that *ave_days* is stationary.

On the other hand, the KPSS test shows non-stationarity in variance since the p-value is 0.01 (lower than 0.05), However we might not completely reject the null hypothesis of stationarity at the 5% significance level.

3.3 Stagionality

To further investigate the discrepancy between the ADF and KPSS test results we analyzed the **autocorrelation function (ACF)** and **partial autocorrelation function (PACF)** of the data to uncover potential seasonal patterns.





The ACF plot shows the correlation of the time series with its own lagged values. The analysis of the ACF for the variable reveals a cyclic pattern with significant peaks at lags 7, 14, 21, and 28.

The PACF plot shows the correlation of the time series with its own lagged values after removing the effects of shorter lags. The PACF for the variable shows a significant peak at lag 7.

This plots suggest a weekly seasonal pattern where each week's value is significantly influenced by the value from the previous week. This information will be crucial for the application of the forecasting models.

3.4 Data Transformation

3.4.1 Differencing

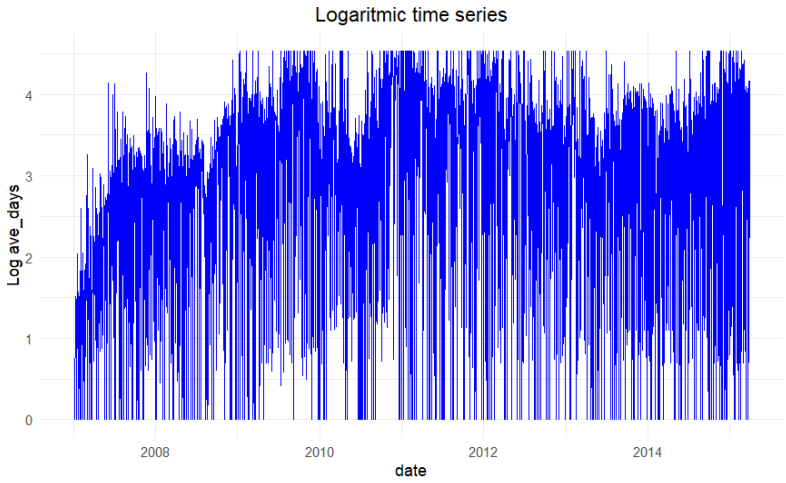
Given the identified weekly seasonality through ACF and PACF analyses, we perform a first-order differencing of the series with a lag of 7. This approach subtracts each value from the value of the same day in the previous week.

After differencing the series, the ADF test continues to indicate stationarity with a p-value of 0.01, while the KPSS test now shows a p-value of **0.1**, which is above the typical significance level of 0.05. This suggests that the series is likely stationary in variance after accounting for the weekly seasonality.

3.4.2 Log transformation

Before proceding with the forecast modeling, I applied a log transformation to the *ave_days* data (adding 1 to avoid log(0). This transformation is beneficial for several reasons:

1. **Avoiding Negative Forecasts:** Many values are close to zero. Applying a log transformation prevents negative forecasts, which are not meaningful in this context;
2. **Stabilizing Variance:** stabilize the variance helps improving the reliability of statistical analyses and models.



The log transformation additionally returned slightly better ADF and KPSS values, as shown in the table below:

Statistic	**Value**
Dickey-Fuller	-7.4289
p-value	0.01
Alternative hypothesis	stationary
KPSS Level	5.3662
p-value	0.01

Table 4: Test statistics for Dickey-Fuller and KPSS tests

To conclude the dataset preparation phase, the data was partitioned into training and test sets using an 80-20 split, useful for the computation of the MAE and evaluation of predictive models.

4 ARIMA

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

- **Autoregressive (AR):** refers to the use of past values in the regression equation for the series;
- **Integrated (I):** involves differencing the raw observations to make the time series stationary;
- **Moving Average (MA):** incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

The ARIMA model is denoted as ARIMA(p,d,q), where:

- **p:** number of lag observations in the model (autoregressive part);
- **d:** number of times that the raw observations are differenced (integrated part);
- **q:** size of the moving average window (moving average part).

Given the considerations of weekly seasonality and the significant influence of Sundays on the variable, I decided to use the Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors (**SARIMAX**) model. SARIMAX extends ARIMA by including seasonal components to capture seasonality patterns and exogenous variables to incorporate additional predictors. In this case, the seasonality is set to 7 and *dummy_sunday* is included as an exogenous variable to account for the unique influence of Sundays.

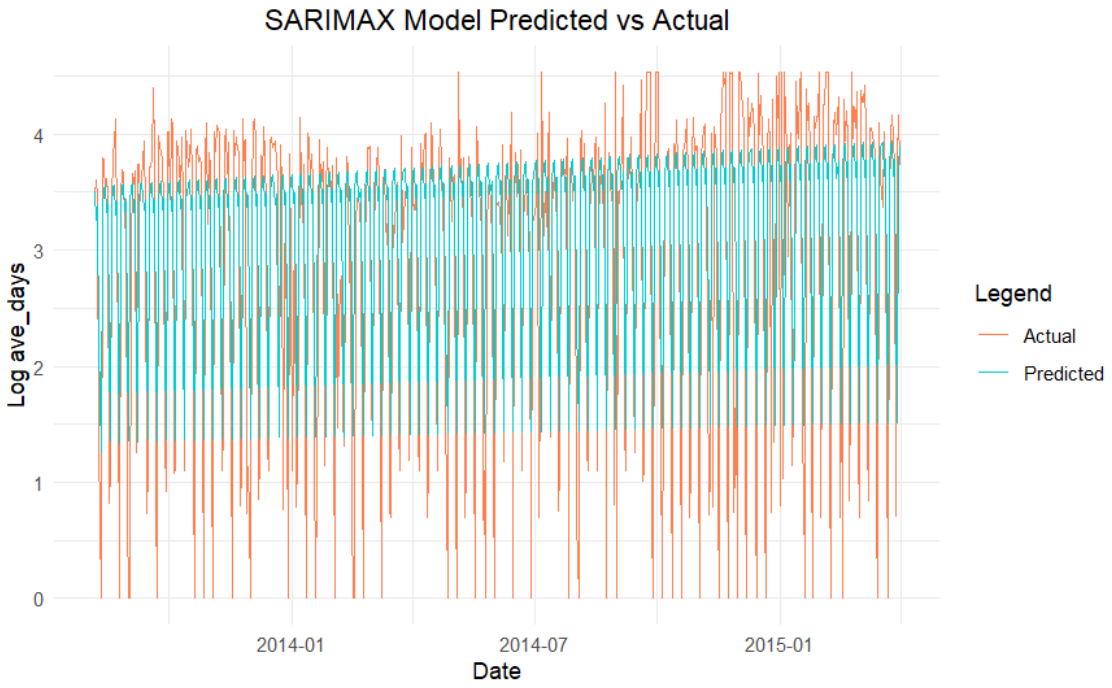
To identify the optimal parameters for the SARIMAX model, I used the *auto.arima* function as a starting point. However, to fine-tune the model, I modified the initial parameters based on the Mean Absolute Error (MAE). The general idea was to increase the seasonal MA terms to capture short-term and seasonal fluctuations more effectively.

In the table below we can see the MAE computed for each model, starting from the *auto.arima* parameters:

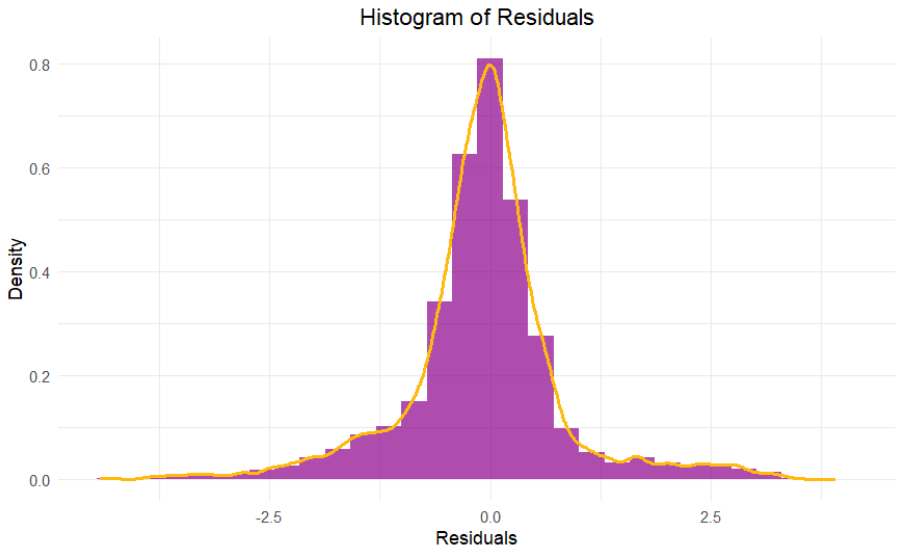
Model	**MAE**
(0,0,3)(1,1,1)[7]	0.680
(0,0,3)(1,2,2)[7]	0.665
(1,0,3)(1,2,3)[7]	0.643
(1,0,3)(1,2,4)[7]	0.637
(0,0,4)(1,2,3)[7]	0.638

Table 5: SARIMAX parameters models and corresponding MAE

The iteration ultimately found the best set of parameters as (1,0,3)(1,2,4)[7]. below we can compare the predicted values in relation to the test data:

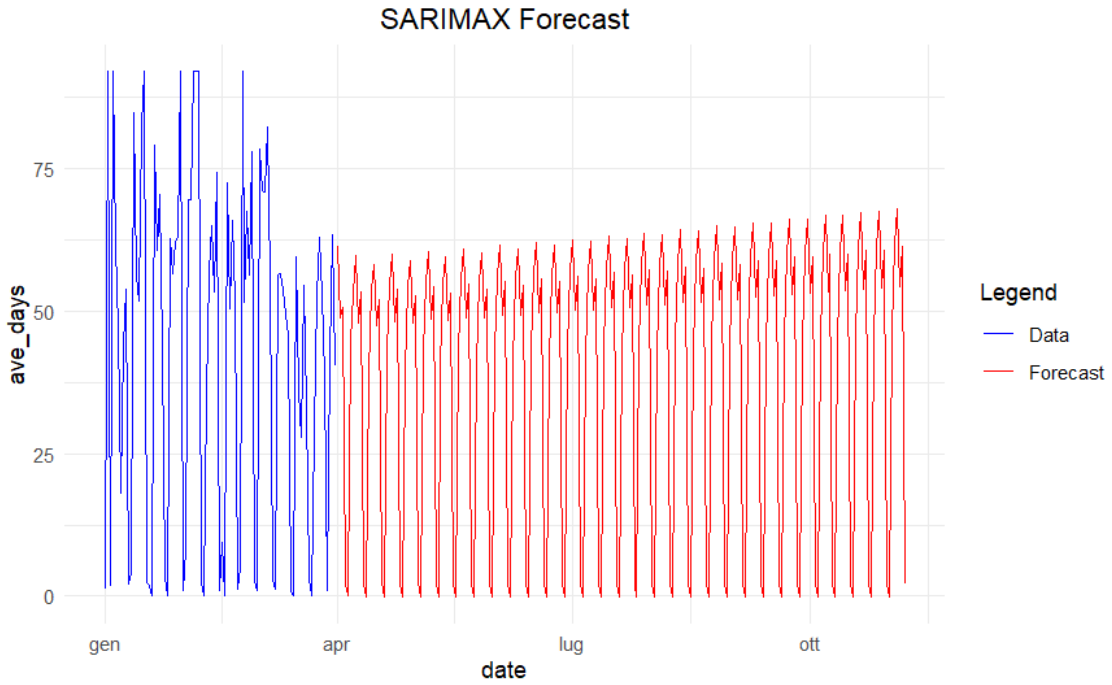


To confirm the reliability of the model, we conduct a residual analysis:



The residuals' histogram reveals that they follow a distribution close to a normal one. This is an important indication that the model captures most of the patterns in the data, leaving residuals that resemble white noise.

The optimal SARIMAX model was finally adopted to forecast the variable values from 2015-04-01 through 2015-11-07:



The forecasted values show an increasing trend, indicating that the *ave_days* variable is expected to rise over the train period.

5 UCM

The Unobserved Components Model (UCM) is a flexible time series modeling approach that decomposes a series into various unobserved components, such as trends and seasonality. This method allows for a more granular understanding of the underlying processes driving the time series data.

For this time series, the following components were considered:

- **Local Level:** captures the overall level of the series, allowing for slow changes over time;
- **Random Walk:** a non-stationary model where the value at each time point is the previous value plus a random shock, useful for series with unpredictable movements;
- **Random Walk with Drift:** extends the random walk by adding a constant drift term, suitable for series exhibiting a steady trend over time;
- **Local Linear Trend:** incorporates both level and slope, allowing for a changing trend over time, which can capture more complex movements in the data;
- **Seasonal Trend:** captures regular seasonal patterns, essential for this series given its identified weekly seasonality.

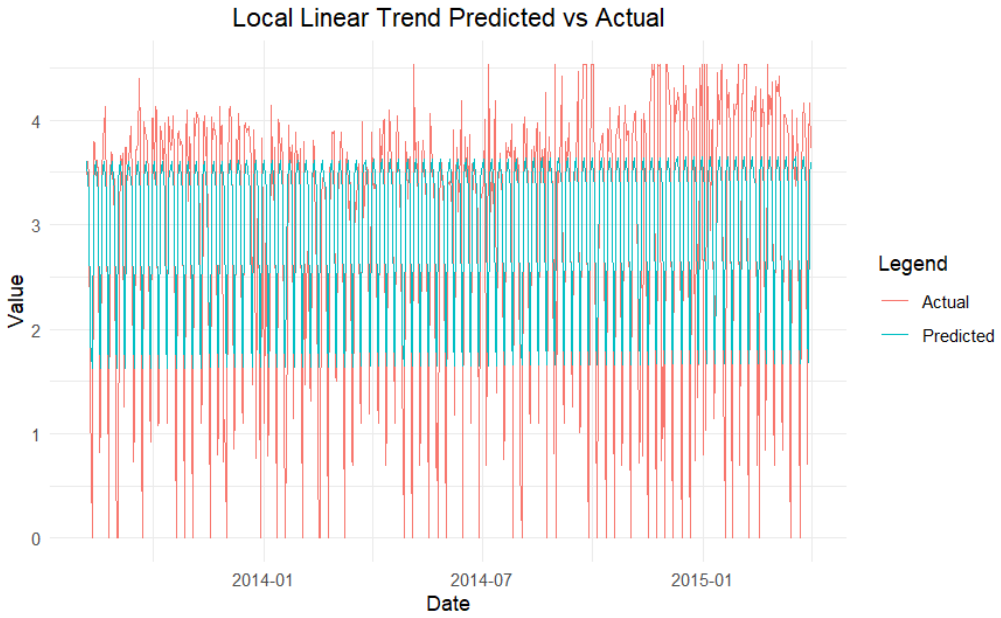
To determine the most effective UCM configuration for the variable, I compared the performance of different models by computing the Mean Absolute Error (MAE) on the test set. The results are reported in the following table:

Model	**MAE**
llevel	0.657
rwalk	1.04
rwdrift	1.23
lltrend	0.652
strend	0.960

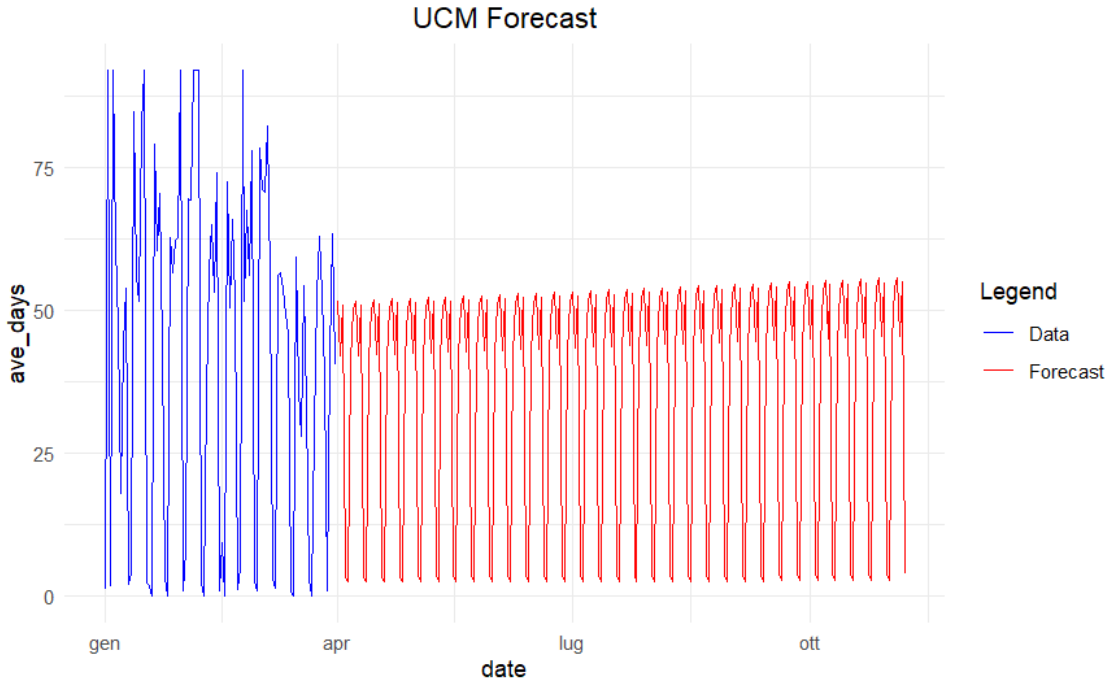
Table 6: UCM models and corresponding MAE

Among the different UCM configurations, the Local Linear Trend model exhibited the best performance with the lowest MAE. An explanation could be that incorporating both a local level and a linear trend, this model effectively adapts to long-term patterns in the data while avoiding overfitting complications.

below we can compare the predicted values in relation to the test data:



Using the Local Linear Trend model, which demonstrated the best performance based on MAE, we forecasted the variable values up to 2015-11-07:



Also in this case, the forecasted values show a slight increasing trend.

6 Machine Learning

Machine learning models have become increasingly popular for time series forecasting due to their ability to handle complex patterns and large datasets. In this analysis, I applied three different machine learning models to forecast the *ave_days* time series: **XGBoost**, **LightGBM**, and **Random Forest**.

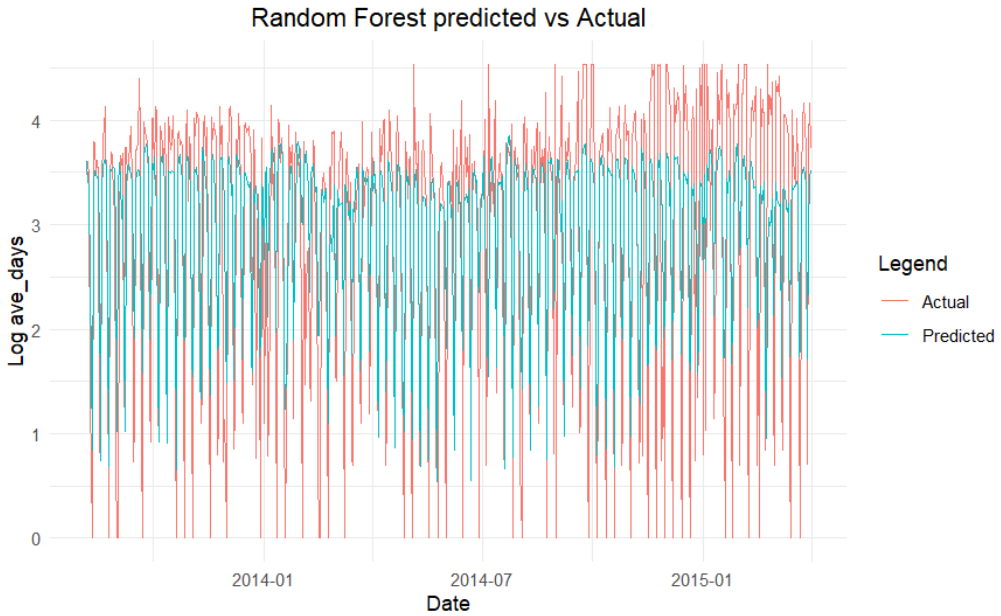
- **XGBoost (Extreme Gradient Boosting):** machine learning method based on gradient boosting. It builds multiple weak learners (typically decision trees) sequentially, where each new model corrects the errors of the previous ones;
- **LightGBM (Light Gradient Boosting Machine):** gradient boosting framework that is optimized for speed and efficiency. It uses a histogram-based approach to find the best splits and is designed to handle large datasets with high-dimensional features;
- **Random Forest:** machine learning model that constructs multiple decision trees during training and outputs the mean prediction (regression) of the individual trees.

To evaluate the performance of these models, we compare their Mean Absolute Error on the test set. The results are shown below:

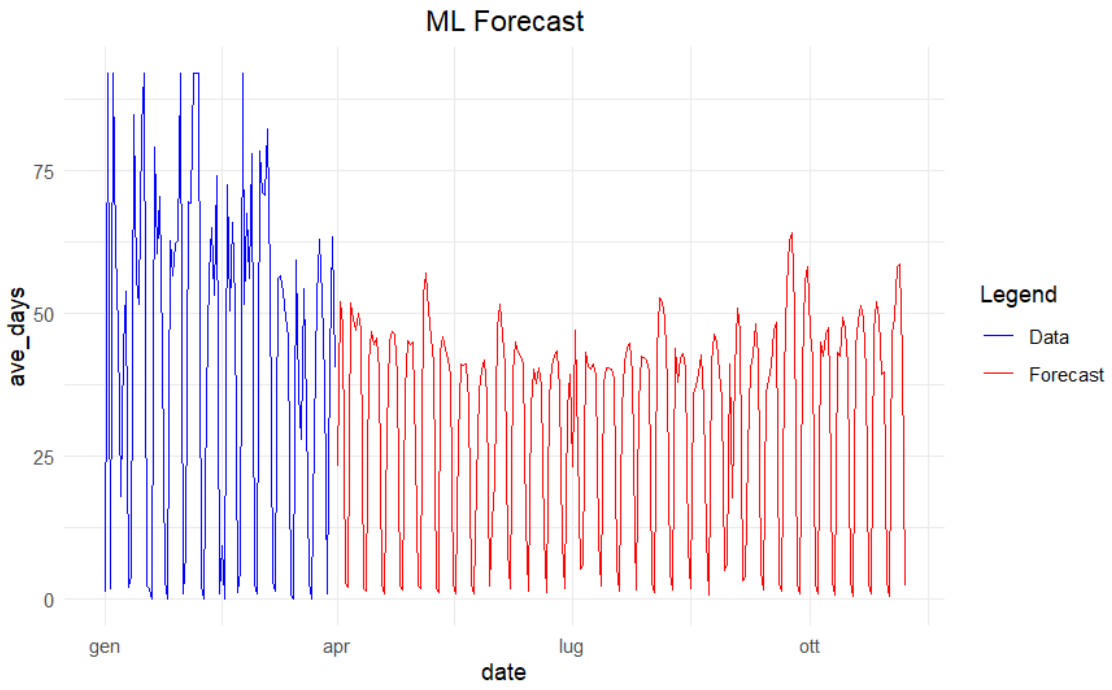
Model	**MAE**
XGBoost	0.698
LightGBM	0.693
RF	0.677

Table 7: ML models and corresponding MAE

Among the three models, the Random Forest exhibited the best performance with the lowest MAE. This can be attributed to its robustness to overfitting, since it averages multiple decision trees.



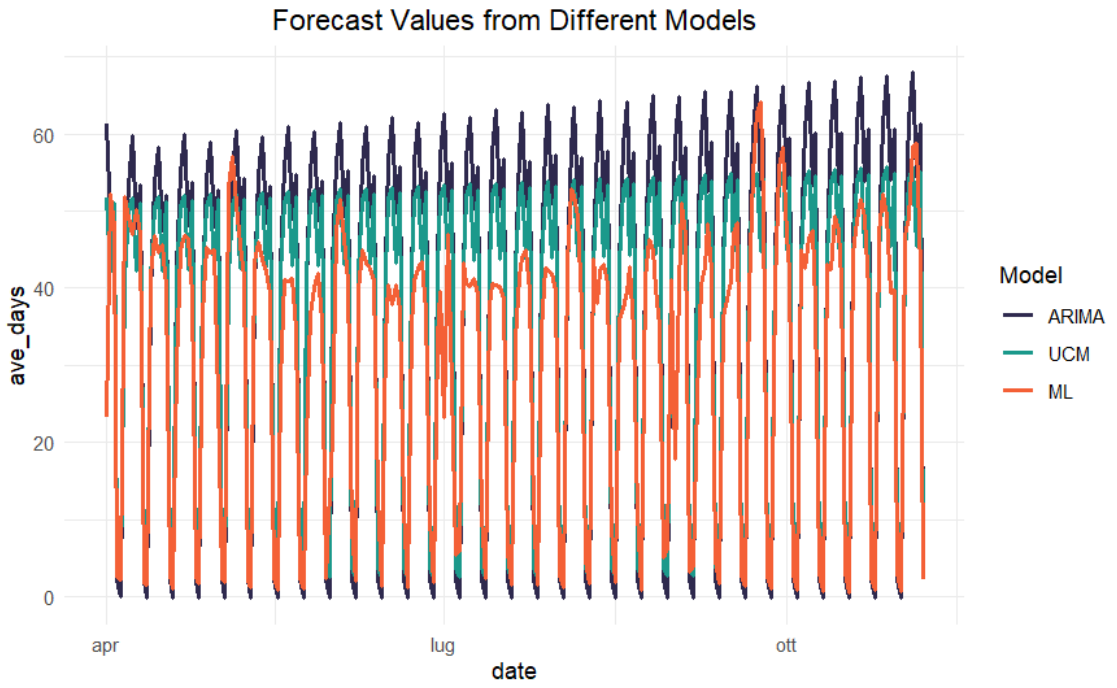
The model is ultimately adopted to forecast the variable values from 2015-04-01 through 2015-11-07:



The predicted values from the Random Forest model do not follow a simple linear pattern. Instead, they exhibit a notable seasonal trend, characterized by a decline during the summer months followed by an increase in autumn.

7 Conclusions

In conclusion, the comprehensive analysis and model comparison demonstrated that a combination of traditional time series models and advanced machine learning techniques provides a robust framework for accurate forecasting. Here we can see a comparison from the three best model from each class:



All three models capture the seasonal patterns, but The ARIMA model could be noisy, since it exhibits the most variability and pronounced spikes. UCM offers the smoothest predictions and Random Forest strikes a balance with detailed seasonal patterns, including intra-week variations.

Each model has its strengths and weaknesses, and depending on the specific forecasting needs different models may be preferred.

References

[1] UCM in python, https://www.statsmodels.org/stable/generated/statsmodels.tsa.statespace.structural.UnobservedComponents.html

[2] COMPARISION STUDY OF ADF vs KPSS TEST, https://medium.com/@tannyasharma21/comparision-study-of-adf-vs-kpss-test-c9d8dec4f62a

[3] Matteo Maria Pelagatti, Streaming Data Management and Time Series Analysis course material on unimib e-learning.