CONSTRUCTOR UNIVERSITY

Comparing Time Series Forecasting Models and Validation Methods for Non-Transparent Markets (Wood Waste)

by

Suraj Karakulath

a thesis for conferral of a Master of Science in Data Science for Society and Business

Prof. Dr. Christoph Lattemann, School of Business, Social & Decision Sciences

Prof. Dr. Andreas Seebeck, School of Business, Social & Decision Sciences

Date of Submission: 20 May 2024

Comparing Time Series Forecasting Models and Validation Methods for Non-Transparent Markets (Wood Waste)

Suraj Karakulath

Constructor (Jacobs) University Bremen

MDSSB-THE- 01 Master Thesis

Supervisor 1: Christoph Lattemann

Supervisor 2: Andreas Seebeck

Date: May 19, 2024

Abstract

Time series forecasting methods have been adopted to forecast prices in various sectors. However, in certain non-transparent markets like the price of wood waste, due to the limitations of availability of information and several external factors, the task is more challenging. In this thesis, various time series models and forecasting methods are compared by using them to forecast future prices of wood waste based on historical prices recorded weekly for different categories of wood waste in 10 clusters (groups of postal codes) in Germany. The standard steps of data cleaning will be followed by exploration of characteristics of the time series including ACF, PACF, decomposition, differencing and stationarity test to select the necessary parameters for the models. Five different models are compared including parametric models such as ARIMA and non-parametric ones like Decision Trees, Random Forest, XGBoost and Prophet, as well as 3 distinct methods of forecasting and validation – static (multi-step) forecasting, rolling window and walk-forward validation. When the performance metrics such as RMSE and Direction Accuracy for each method and model were compared, it was found that the static method yielded poorer results on RMSE in most cases, and that the choice of the recommended model and forecasting method depended heavily on the specific time series of the categories in the clusters.

Keywords: time series analysis, time series forecasting, ARIMA, ARIMA, Decision Trees, Random Forest, XGBoost, Prophet

Acknowledgements

The work on this thesis project and completion of this report was made possible through the guidance and supervision from several collaborators and individuals. I am grateful for all the help and support I received from them. I would especially like to express my sincere gratitude for my supervisors Prof. Dr. Christoph Lattemann and Prof. Dr. Andreas Seebeck. I am also indebted to Laura Maria Schmid, Research Associate to Prof. Latteman, for her valuable insights, learning resources and encouragement. I would also like to thank all my professors and tutors at Constructor University Bremen, that I had the pleasure to study under and collaborate with and for teaching me the essential foundational concepts and skills pertaining to time series and other data science topics.

Contents

Introduction

The increasing awareness of the negative impact of fossil fuels and the growth in favorable view of renewable raw materials (Daian and Ozarska 2009) has led to more initiatives by consumers and organizations in the judicious treatment of waste products. Consumers are exploring ways to make productive use of their waste and organizations in the renewable energy sector are discovering new opportunities to collect waste that can be used for energy production.

One of the major examples of such increased use of renewable raw materials can be seen among organizations that specialize in wood processing (Bell, Mollenkopf, and Stolze 2013). In the timber industry, where it is possible to completely utilize wood raw material, production of products can be complemented by the reusing and recycling waste or by-products. Specifically, the lumber industry is one where the by-products of wood processing can be used for energy generation purposes. In addition to by-products of wood processing, the waste material that is generated after the consumption of wood-based products are also valuable resources that can be used for generating energy, with the help of energy power plants. Various stakeholders involved in this ecosystem, including wood-processing plants, enterprises that act as intermediaries between waste suppliers and renewable energy generation and power plants will therefore be interested in the market dynamics and price fluctuations of wood waste to conduct their activities.

However, analyzing the price movements in this market is a difficult endeavor due to the "non-transparent" nature of the market. This means that there is often limited information availability and opacity in pricing. These limitations create inefficiencies and lowers confidence among stakeholders. There may also be several hidden factors and variables that could

potentially determine prices in such markets, such as the price of other commodities like oil or gas, electricity consumption and business cycles.

This is in contrast to transparent markets such as financial markets and commodity markets, where information is readily available, easily accessible, and understandable to all participants. In such markets, prices of goods, services, or financial instruments are openly disclosed, allowing buyers and sellers to make informed decisions. Relevant information about market conditions, product specifications, terms of trade, and other factors influencing transactions is also more readily accessible.

To effectively address the limitations present in forecasting of prices in non-transparent markets, one can employ time series analysis and forecasting, which makes predictions for future price points based on historical data, if such data is available. It is also critical to experiment with a variety of models and compare their evaluated performance before selecting one, as the very lack of transparency could mean that different models work for different cases.

To this end, this thesis uses historical data on wood waste price recorded and provided by the Brüning Group company, a market-leader in trading with energy-supplying bulk goods. The company supplies combined heat and power plants with the desired combustibles such as wood chips, waste wood and material from landscape conservation, after collecting them from suppliers. Their business model relies on collecting wood waste from sellers, which could be consumers or businesses, at a specific price and selling it to power plants for energy generation. In the buying stage, the price can be negative or positive, depending on whether the company was paid by the seller to collect the wood waste or if they had to pay to the seller. This makes it critical for the company to know what the price of the wood waste will be in the future, in order to understand and arrive at an appropriate number for negotiations. The historical data supplied

by Bruning Group records the weekly prices of different categories of wood waste since September 2020 for 10 “clusters” (combinations of postal codes) in Germany.

There is much literature and research done on time series analysis and forecasting and various machine learning algorithms have been introduced and explored. Some of the commonly used models like ARIMA, Random Forest as well as neural networks have been adopted for time series analysis and forecasting of different variables including those in financial markets. Some of these have also been employed for forecasting variables associated with waste, including the amount of waste and even the price of raw wood materials. However, the specific variable of wood waste prices is yet to be explored for time series forecasting.

This thesis will first cover the existing literature on time series analysis and forecasting techniques adopted for non-transparent markets to identify the gaps in research on the specific case of forecasting of wood waste prices. Then the data provided by the Brüning Group company is cleaned, analyzed and explored, before using various models to make forecasts on a test data split from the original dataset. The forecasts themselves are made using different procedures, namely static (multi-step) forecasting, rolling window and walk-forward validation, and each method for the models are evaluated for their accuracy using relevant metrics such as Root Mean Squared Error (RMSE), and Direction Accuracy. The metrics are also compared across categories and clusters to explore whether some models perform better in certain clusters than others and finally, a recommended approach for forecasting future price is provided, to enable stakeholders like the Brüning Group company, to make informed decisions about the best prices to negotiate for in their operations with suppliers and buyers, to help them conduct business effectively. Additionally, this thesis is also expected to contribute to the growing body of knowledge in forecasting prices in other non-transparent markets and to offer valuable insights

and recommendations for practitioners, researchers, and decision-makers in their respective sectors.

Literature review

The forecasting of price and other quantitative variables using time series is a common task in financial markets, economic research institutions, government-affiliated entities and business operations. In the specific sectors of renewable energy and waste management, there is significant value in being able to forecast the price or value of various variables. For instance, various models have been explored to forecast municipal solid waste (MSW), including correlation analysis (Daskalopoulos, Badr, and Probert 1998) and linear regression (Kumar and Samadder 2017).

Decision trees have been used to predict municipal solid waste (MSW) in Canada (Kannangara et al. 2018), gradient boosting for forecasting weekly MSW generation in New York City (Johnson et al. 2017), applied support vector machine (SVM) for monthly MSW generation in Australia (Abbasi and El Hanandeh 2016) and similar tasks in India (Kumar et al. 2018), a well as artificial neural networks (ANN) (Azadi and Karimi-Jashni 2016) in other cases. A comparison of 6 ML models including random forest (RF) and k-nearest neighbor (KNN) algorithms was explored by (Nguyen et al. 2021) to forecast solid waste generation in residential areas in Vietnam. This study looked at eight variables including those related to the economy, demography, consumption and waste generation and found that urban population, monthly consumption expenditure, and retail sales were important variables to consider. These studies do not analyze waste prices specifically, but the amount of waste or their correlations with other variables such as GDP and population.

Other studies have explored time series to model price related variables associated with wood. An analysis of price development of raw wood in the Czech Republic was conducted by (Hlavackova, Brezina, and Sujova 2015) using statistical methods such as price indices and trendline analysis, which suggested that the status of wood producers and organizations in the wood processing industry and global economic developments were important factors to consider.

Different variants of the ARIMA model have also been employed to forecast prices of variables adjacent to wood. For example (Leskinen and Kangas 1998) used the AR(1) model to simulate realistic future timber price scenarios in Finland. ARIMA models were also used by (Malaty, Toppinen, and Viitanen 2007) in analyzing the Nordic pine sawlog markets in Finland by using monthly prices of real stumpage from 1995 to 2005. Similarly, (Michinaka et al. 2016) used ARIMA to forecast the monthly prices of domestic logs in Japan, primarily to aid logging companies, forest owners and sawmills assess profitability, manage daily operations and for better risk management. ARIMA methods have shown to provide highly satisfactory fits with low data collection costs ("Forecasting Prices of Manufactured Pinus Spp. Using ARIMA Models," n.d.) but these types of models have also shown to perform poorly in long-term forecasts and in predicting unusual movements (Chu 1978). Another study explored the prediction prices (Górna et al. 2023) in Polish conditions of wood processing by-products in the form of biomass. This employed the ARIMA model among others to forecast trends and prices of selected wood products and discovered that supply changes due to global influence created significant uncertainty and that the choice of the model depends on market conditions. A comparison of artificial neural networks and other time series models was conducted by (Kożuch, Cywicka, and Adamowicz 2023) for forecasting prices of timber in Poland but to analyze the timber market from the point of view of production and sales.

Most of these previous studies involved forecasting the MSW generated and not the actual price related variables or that of wood waste specifically. In fact, there is not much notable literature that deals directly with the prediction of future prices for post-consumption wood. While the MSW generated may influence the price of waste and therefore that of wood waste, there may be other factors and determinants that need to be considered in order to be able to produce more accurate forecasts. This lack of transparency in the wood waste price market means that using time series analysis on the actual historical prices is likely to be a more reliable indicator of future prices.

Moreover, these previous studies only compared a few predictive ML algorithms at a time, which may therefore miss out on the versatility of different ML techniques as well as their advantages or limitations. This thesis builds on these previous studies by using historical data on wood waste prices obtained from a market-leading firm in Germany to forecast future prices using a variety of models and methods. The diversity of the models and methods will shed light on the ability and limitations of each, so that stakeholders can select the best time series forecasting technique that fits the needs of their specific task.

Time series forecasting

Time series analysis is a fundamental aspect of understanding and predicting trends in various fields, from finance to weather forecasting. It involves the use of various models to predict future values based on past observations. Over the years, there has been a lot of research and models proposed by academics, data scientists and analysts for time series forecasting, with some of them being generally applicable while others being ideally suited for specific domains.

The ARIMA models are one of the most widely used among these for modeling time series data with stationary properties. It is based on a parametric machine learning algorithm. This means that the learning model summarizes the data using a set of parameters of fixed size and does not depend on the number of examples available for training.

ARIMA is an example of an autoregressive model, which means that prediction of future values is based on previous values ("Forecasting: Principles and Practice (3rd Ed)," n.d.). These are linear models, which means that the function that maps the input variables of time to output variables, or price in this case, is often assumed to be linear or a linear combination of the input variables. Such models have the benefit of being interpretable and intuitive, since it can be seen how a past time step is weighted and how this input influences the prediction.

In contrast, there are other algorithms that assume anything about the form of the mapping function. In this way, they are capable of freely learning whatever functional form is present from the training process. These are called nonparametric machine learning algorithms. Models based on Decision Trees, including RandomForest, XGBoost and Propet are non-parametric models.

Decision Tree models make predictions taking the averages of subsets of the training dataset. The subsets in this case are formed by partitioning the space of input data into different regions and then for each region, the average of all observation outputs inside those regions is taken as a prediction. In the case of time series, the regions are simply splits of time intervals, which are mutually exclusive and completely exhaustive.

For Decision Tree based models, it is required to have both input and output variables for the model to learn from as per the supervised machine learning approach. In a univariate time-series problem, however, there is only the time series as the target. To work around this

issue, the time-series needs to be prepared in the form X values and y values to become suitable for tree models.

Random Forest and XGBoost models build on the Decision Trees in their own way. Random Forest constructs a number of decision trees, each of which is trained on a random subset of the training data and a random subset of the input features before averaging the predictions from the trees. XGBoost builds a collection of decision trees sequentially, with each subsequent tree attempting to correct the errors made by the previous ones in an iterative approach.

Finally, Prophet is another non-parametric model that implements an additive time series forecasting model, comprising trend, seasonality, and holiday effects.

In the analysis for this thesis project, the historical price data of wood waste is used to predict the price at future time steps using the ARIMA, Decision Tree, Random Forest, XGBoost and Prophet models.

Data collection and preprocessing

The data used for this analysis and forecasting was obtained from the client Brüning Group. It includes the price of waste as recorded weekly from around Sep 2020 until May 2024. The price value is sometimes recorded as negative, which is when the client was paid for taking the waste from the supplier and at other times as positive, in which case the client made payments to suppliers to take the waste from them.

There are 5773 observations in total in the data set, with seven columns for distinct features. These include:

- An index column of integer values
- A “week” column in object type

- “wPreis” which contains the price recorded in float format
- “Plz” which is a string object that denotes the cluster (combination of Postal Codes)
- “Category” – a string object that denotes one category of the wood waste
- “Behandelt” – a string object that denotes another category of the wood waste
- “full” – a string object that denotes the combination of the Category and Behandelt categories

The data is recorded for 10 unique clusters which are collections of Postal Codes (Postleitzahl) in Germany. These are recorded as follows:

1. "['25', '24']"
2. "['80', '81', '82', '83', '84', '85', '93', '94']"
3. "['70', '71', '73', '74', '75', '76']"
4. "['50', '51', '52', '53']"
5. "['40', '41', '42', '44', '45', '46', '47']"
6. "['26', '27', '28']"
7. "['48', '49']"
8. "['20', '21', '22', '23']"
9. "['10', '11', '12', '13', '14', '15', '16']"
10. "['29', '30', '31', '38', '39']"

There are also different product categories recorded, a total of 4 unique ones namely:

1. 'A2 - geschreddert'
2. 'A1 & A2 - geschreddert'
3. 'A2 & A3 - geschreddert'
4. 'A3 - geschreddert'

These are different categories of the wood waste as recorded by the client. This suggests that time series analysis and prediction must be conducted for values across time for a single

category at a time, as mixing up multiple categories is not meaningful. There are no null values in the data set so it is not required to drop any rows.

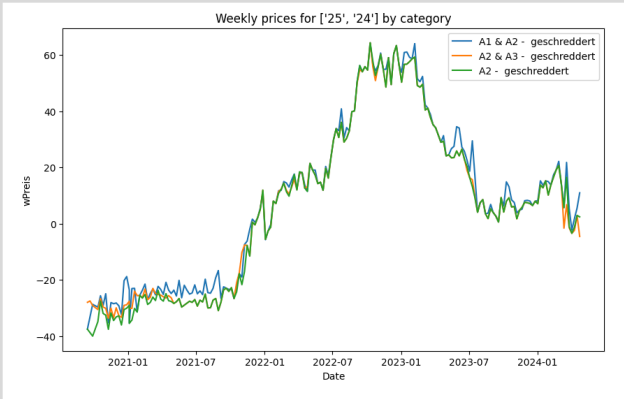
Data cleaning and exploration

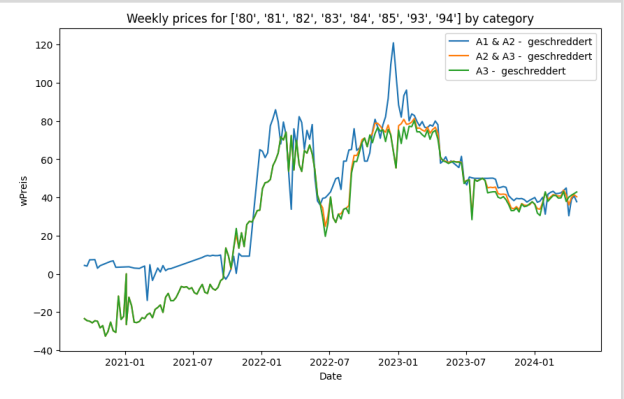
The data is cleaned according to certain basic standard steps. This included some string edits and transformations to simplify the category column and conversion of the 'week' column to datetime format.

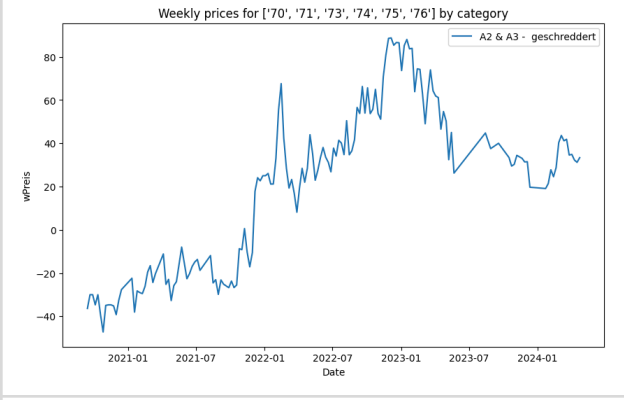
The “full” category is simply the combination of the values in the Category and Behandelt columns, which makes the latter two columns redundant and these can be removed.

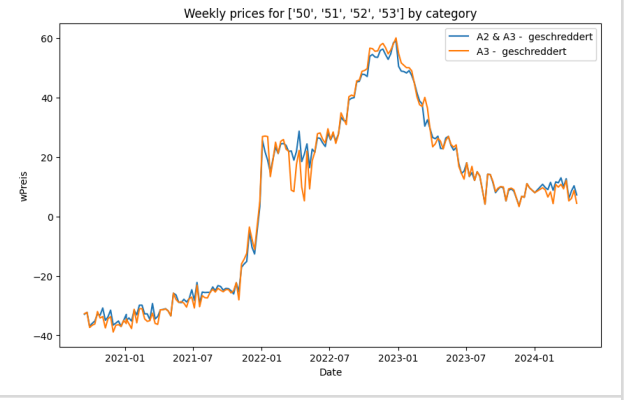
Since the week variable contained only week numbers in the format “YYYY-WW” denoting the number of the week in the year, this is converted into a datetime variable in the format “YYYY-MM-DD”. As a general best practice for time series analysis and prediction, the datetime variable is also set as the index of the dataframe.

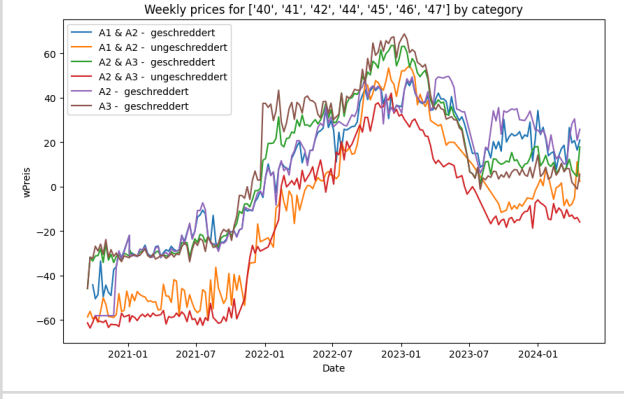
When the wPreis values are plotted, it is seen that for all 3 categories in the same cluster these values follow a somewhat similar trend, with occasional deviations. The number of categories in each cluster also varies, as some clusters have only one waste category, such as 'A1 & A2 - geschreddert' for the cluster [1, 4, 6, 7, 8, 9] while others can have up to 6 categories like the cluster ['40', '41', '42', '44', '45', '46', '47'].

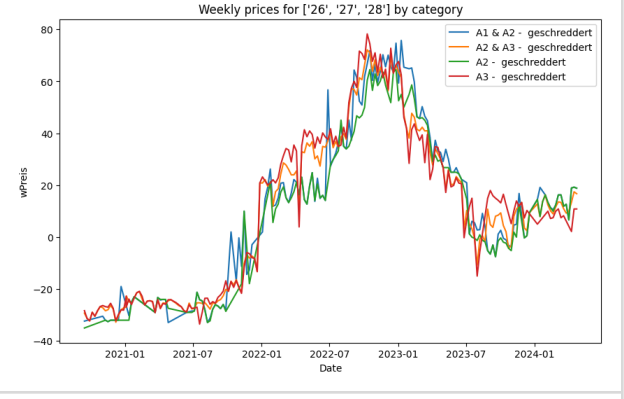

Weekly prices for ['25', '24'] by category
A1 & A2 - geschreddert
A2 & A3 - geschreddert
A2 - geschreddert
wPreis
Date

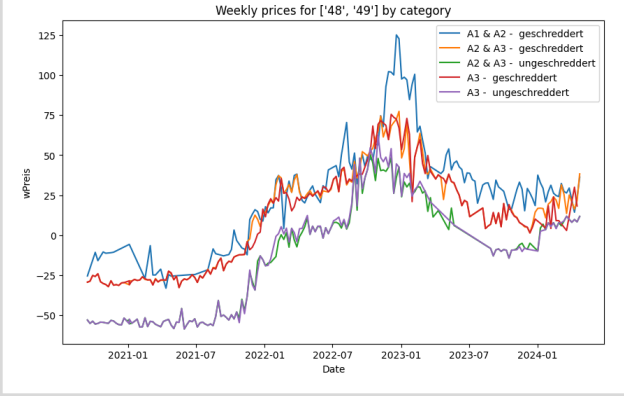

Weekly prices for ['80', '81', '82', '83', '84', '85', '93', '94'] by category
A1 & A2 - geschreddert
A2 & A3 - geschreddert
A3 - geschreddert
wPreis
Date

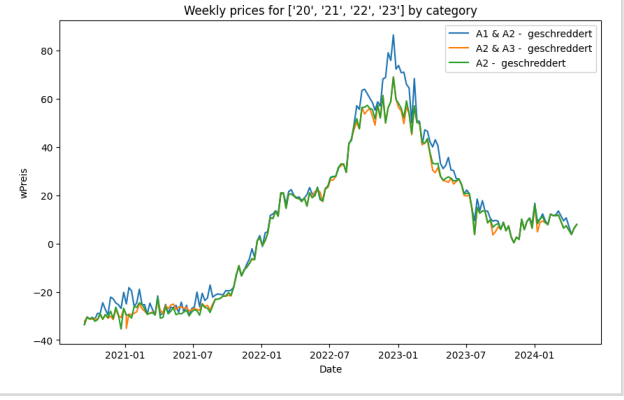

Weekly prices for ['70', '71', '73', '74', '75', '76'] by category
A2 & A3 - geschreddert
wPreis
Date


Weekly prices for ['50', '51', '52', '53'] by category
A2 & A3 - geschreddert
A3 - geschreddert
wPreis
Date


Weekly prices for ['40', '41', '42', '44', '45', '46', '47'] by category
A1 & A2 - geschreddert
A1 & A2 - ungeschreddert
A2 & A3 - geschreddert
A2 & A3 - ungeschreddert
A2 - geschreddert
A3 - geschreddert
wPreis
Date


Weekly prices for ['26', '27', '28'] by category
A1 & A2 - geschreddert
A2 & A3 - geschreddert
A2 - geschreddert
A3 - geschreddert
wPreis
Date


Weekly prices for ['48', '49'] by category
A1 & A2 - geschreddert
A2 & A3 - geschreddert
A2 & A3 - ungeschreddert
A3 - geschreddert
A3 - ungeschreddert
wPreis
Date


Weekly prices for ['20', '21', '22', '23'] by category
A1 & A2 - geschreddert
A2 & A3 - geschreddert
A2 - geschreddert
wPreis
Date

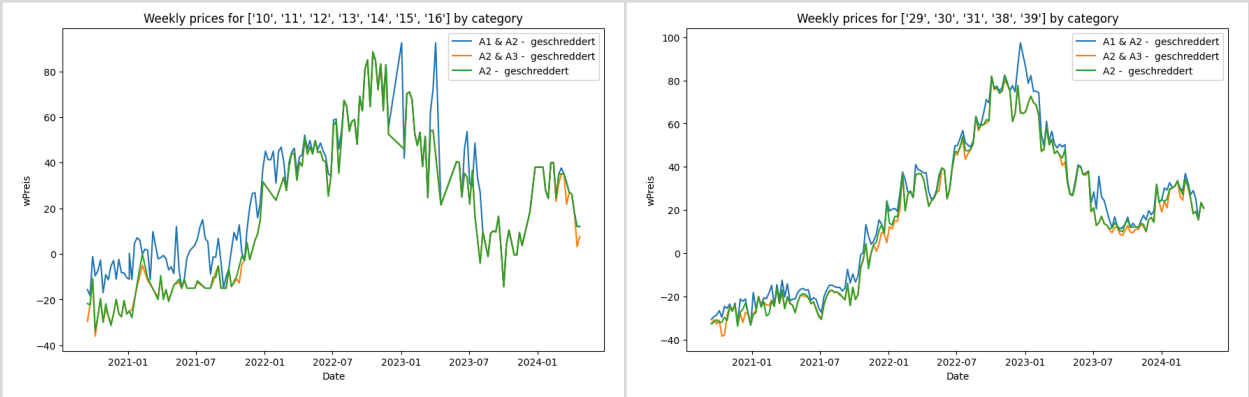


Fig. 1. The weekly wPreis values from September 2020 to April 2024 for each category in all 10 clusters

As can be seen from the plots in Fig. 1., the price values tend to remain relatively stable with a slight increase until the end of 2021 after which they start rising until the middle of 2022. This is followed by a noticeable spike in prices in late 2022 and then a sharp decline throughout 2023 until it stabilizes again with a gradual downward trend towards the end of 2023, followed by an increasing trend in early 2024.

There could be a number of factors behind this structure for the time series. The gradual increase in 2020 could be in part due to the post-COVID-19 recovery and resumption of economic activities, which may also have had an effect on the demand for wood-based products. Continued supply chain disruptions may also have reduced the availability of wood waste, contributing to the increase in prices. The sharp rise in early to mid 2020 may have been driven by the energy crisis, exacerbated by geopolitical tensions, and the increased demand for alternative energy sources, including biomass and wood waste, not to mention the rising inflation rates. The subsequent drops in prices from the middle of 2022 may have resulted in part due to the stabilization of supply chains as producers and suppliers adjusted to the increased demand, as well as the economic slowdown or recession in 2023.

All the categories in all the clusters exhibit a similar pattern as well in their respective time series as well, which suggests that the prices for the different categories generally tend to

behave in similar ways, even if they may have slightly different influencing factors, and that it makes sense to test the same time series models for forecasting across all of them.

Methodology

The methodology adopted for this analysis involves comparison of forecasts from the different models described. But crucially, we also explore different procedures of forecasting using each of these models, wherever possible. These are Static (multi-step), Rolling window, and Walk-forward validation. Each approach offers unique advantages and considerations, influencing the accuracy and reliability of forecasts.

Static (Multi-step) Forecasting: This involves fitting the model on a training subset of historical data and generating forecasts for multiple future time steps corresponding to a test subset in one go. The model is trained using a fixed historical window once, and forecasts are generated for a predefined horizon. The forecasts are compared with the test data set values and evaluation metrics such as Root Mean Squared Error (RMSE) are used to assess how good the accuracy is.

Rolling Window Forecasting: In this method, the model is fit to the training data set in a rolling window of pairs of X and y values. However, for the forecasting, this differs from the static method as it continues to forecast one step at a time based on the input from the immediately preceding window taken as predictors. The window rolls forward incorporating new observations from the test data set to predict for the next step, while discarding older data points, unlike the static multi-step prediction, where the forecasts for the test set are not made through

such a moving window. The same evaluation metrics such as RMSE can be used to compare how far off the predictions are from each step in the test data set.

It is important to note that this procedure of forecasting may not be applicable for all the models under consideration due to various reasons, including the fact that some models do not take a sequence as input or in the case of Prophet, the design of the model as publicly released by Facebook.

Walk-forward Validation: In the walk-forward validation approach, the model is trained with an increasing historical window and used to generate forecasts for the next single time step. The forecasting for each step in the test data set is still done based on a moving window of preceding input but unlike the rolling window approach, the historical window is not fixed but continues to grow larger with each new observation added from the new time step, and the process is repeated iteratively. The accuracy of the predictions are still assessed using the common evaluation metrics.

By comparing these forecasting approaches, we will explore the best model and which specific procedure using that model can be best suited for prediction.

Auto-correlation and Partial Auto-correlation

Before testing the different models on the time series, especially that of autoregressive models like ARIMA, it is important to consider whether there are any correlations in the structure of the data. The Auto-correlation Function (ACF) and Partial Auto-correlation Function (PACF) are useful tools for analyzing the temporal dependencies present in time series data. This

allows for the selection of appropriate lags and inputs that models need to be supplied with as parameters for prediction.

The ACF measures how correlated the values of a time series are with those at a different time in the series, known as lagged values. A strong autocorrelation at a particular lag suggests that there could be a repetitive pattern or trend in the data which can be taken advantage of for prediction purposes. By examining the ACF plot, and looking for lags where the correlation exceeds a set confidence interval, different models can be informed of parameters such as the number of time steps to be considered for input.

The PACF is a similar metric, which also examines the correlation between values in a time series and its lags while also taking into account the effects of the time steps in between. It essentially captures how much influence each lag has on the current observation, excluding the indirect effects mediated by other lags. Similar to the ACF, the PACF helps in identifying the lag that can be used for specifying the order of autoregressive terms in forecasting models such as ARIMA.

The plotting functions plot_acf, plot_pacf from the statsmodels.graphics.tsaplots package can be used to plot both the ACF and the PACF. When these plotting functions are called on the price values for the categories in the cluster ['25', '24'] the following results can be observed.

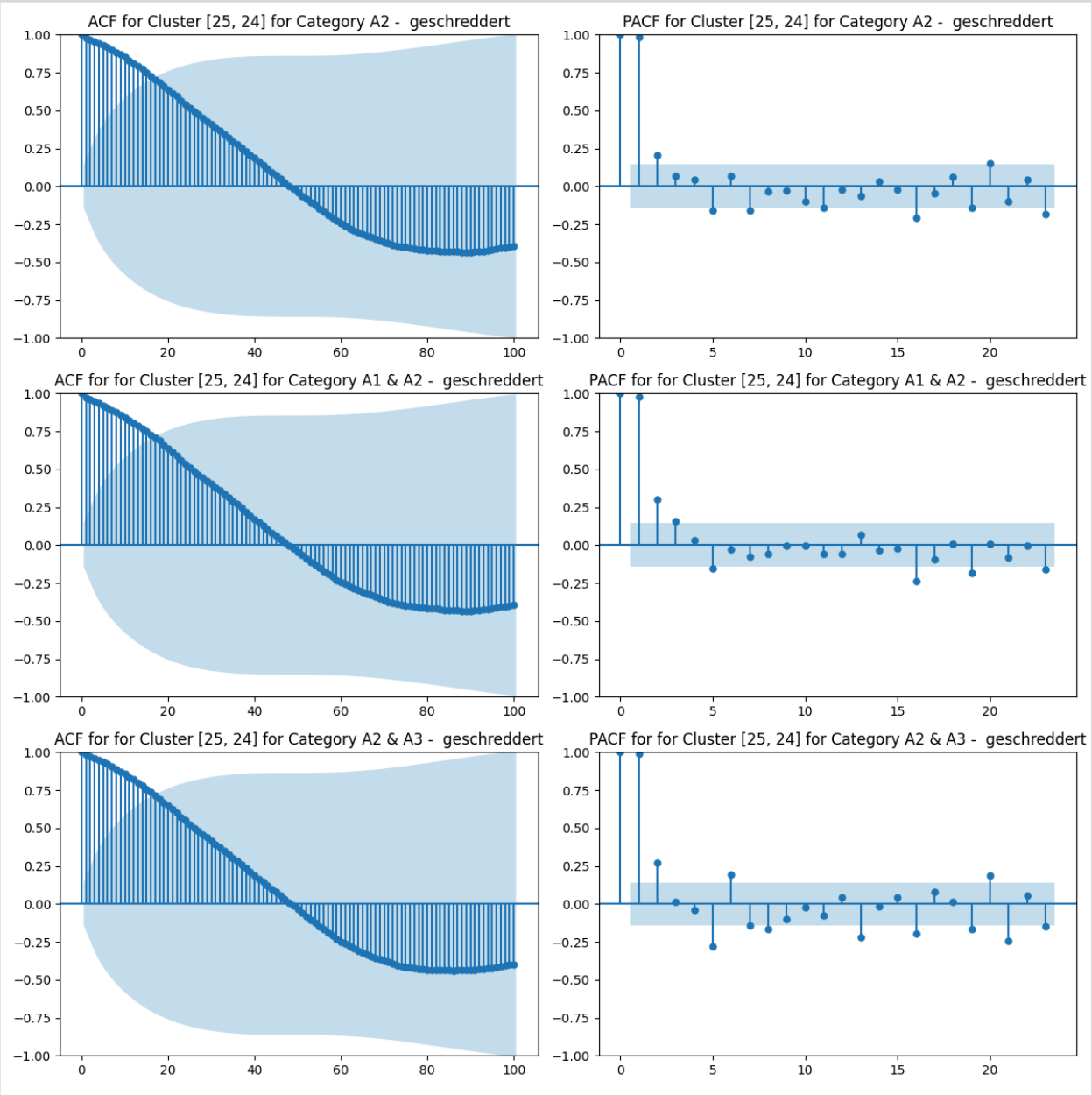


Fig. 2. ACF and PCF plots of time series for the three categories in the cluster [25,25]

The ACF plots show that there is a positive correlation with the first 10-to-15 lags and it decreases backwards in time until it becomes less significant after around 15 weeks.

The PACF plots show high correlation for 1 week lag but that is expected. There are also significant correlations for lags up to and around 5, as well as further back in time, around 20 weeks.

While these are useful metrics to use for the modeling, for the purpose of this analysis, they are only taken as suggestions for potential correlations. The reason is that the "category" field is provided from the stakeholder after combining multiple categories and this may cause some of the lags to be missed in the plot. For the purpose of this analysis, a lag of 4 time steps (weeks) is taken for the modeling tasks.

Seasonal decomposition

In addition to the ACF and PACF, it is also useful to decompose the time series into various components through seasonal decomposition. This is a crucial technique used in time series analysis to decompose a series into its constituent components, namely trend, seasonality, and residuals. The decomposition process involves separating the original time series into three main components:

Trend: This component captures the long-term trajectory or underlying trend of the time series data. It suggests the overall direction of the series, illustrating gradual changes and movements over time.

Seasonality: The seasonality component captures the repetitive patterns or fluctuations that occur at regular intervals. Seasonality could be seen as daily, weekly, monthly, or yearly cycles, depending on the nature of the data and the underlying factors driving the fluctuations.

Residuals: The residual component shows the noise present in the data after removing the trend and seasonal components. It captures the unexplained variation in the time series that cannot be attributed to the trend or seasonality.

Seasonal decomposition allows for the identification of recurring patterns and trends that may be present in the data, which helps further in developing accurate forecasting models. Similar to the ACF and PACF, the functions seasonal_decompose from the from statsmodels.tsa.seasonal can be used to extract the trend, seasonal and residual components.

The seasonal_decompose function also has a parameter “period” that can be specified. This determines the length of the seasonal component that the function will attempt to extract from the time series data. For example, in the waste price time series, the data is recorded weekly, and the period parameter can be set to a value where it is expected to see some repeating patterns. Since the ACF and PCF plots suggested a correlation of around 4 to 5 weeks of lag, the period parameter can be set to 4 to observe the results.

When applied on the price values for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' we observe the following components.

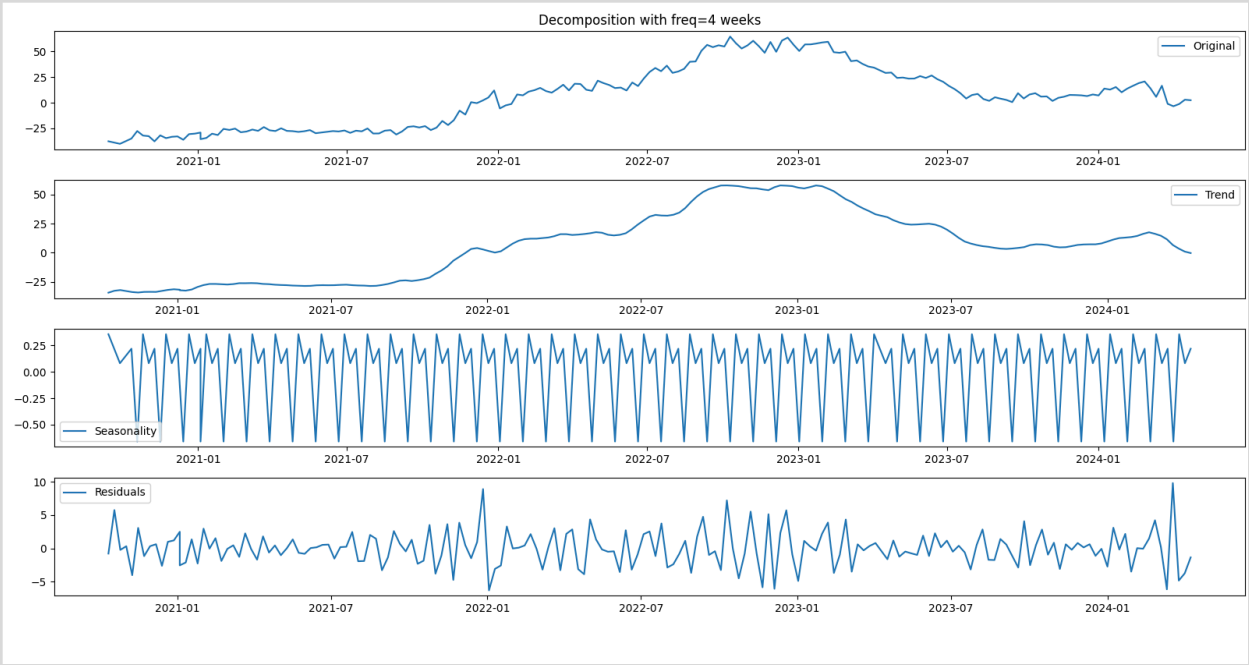


Fig. 3. Seasonal decomposition results for category 'A1 & A2 - geschreddert' in cluster ['25', '24'] with frequency of 4 weeks.

The results of the seasonal decomposition suggests that the trend is one that remains somewhat steady from 2020 until early 2022, with an increase that continues until late 2022 after which a decrease is seen until the end of the time period. The seasonality also shows that there is a clear seasonal pattern around 4 weeks, although the scale of the seasonal changes are small compared to that of the trend. The residuals indicate that there are some random fluctuations that increase in magnitude, and which are of comparable scale to the trend values at some points but remain approximately constant over time.

Stationarity – Testing and Converting

Another important step that must be taken before time series modeling, especially in the case of the ARIMA model, is the check for stationarity. A stationary time series is one in which the statistical properties such as mean and variance stay constant over time. In a stationary time

series, the underlying data does not exhibit systematic changes or trends, making it easier to model and forecast future values. Assessing stationarity can be either by a visual examination of the time series plot, analyzing summary statistics such as mean and variance, or through statistical tests such as the Augmented Dickey-Fuller (ADF) test.

Stationarity analysis is crucial for determining the appropriate modeling approach. While not all models require the conversion of a non-stationary time series to stationary such as Decision Trees, RandomForest, XGBoost, and Prophet for, others like ARIMA assume stationarity. If the time series is non-stationary, it's typically necessary to difference the series to achieve stationarity (i.e., remove trends and seasonality) before fitting an ARIMA model by subtracting the series from its lagged values.

Dickey-Fuller test

To determine whether a time series is stationary or not, the Augmented Dickey-Fuller (ADF) test can be employed. This is a statistical test based on the null hypothesis that the time series has a unit root, which indicates that it is non-stationarity. The alternative hypothesis is that the time series is stationary. The test statistic is compared to critical values from the Dickey-Fuller distribution to determine whether the null hypothesis can be rejected.

The ADF test statistic is negative, and more negative values indicate stronger evidence against the null hypothesis of non-stationarity. If the test statistic is less than the critical value at a chosen significance level, the null hypothesis is rejected, and the series is considered stationary. Conversely, if the test statistic is greater than the critical value, the null hypothesis cannot be rejected, and the series is considered non-stationary.

The ADF test can be applied to both raw time series data and differenced data, to check if a time series can be made stationary after differencing. The ADF test is performed by using the adfuller function from the statsmodels.tsa.stattools package and delivers the following results for each of the categories in the cluster ['25', '24'].

Table 1. Results of Dickey-Fuller Test for the categories in the cluster ["['25', '24']"]

```
Results of Dickey-Fuller Test for ['A2 -  geschreddert'] in ["['25', '24']"]:
Test Statistic                  -1.378535
p-value                          0.592522
#Lags Used                       4.000000
Number of Observations Used    182.000000
Critical Value (1%)             -3.466800
Critical Value (5%)             -2.877555
Critical Value (10%)            -2.575308
dtype: float64
Failed to Reject Ho - Time Series is Non-Stationary
```

```
Results of Dickey-Fuller Test for ['A1 & A2 -  geschreddert'] in ["['25', '24']"]:
Test Statistic                  -1.420392
p-value                          0.572462
#Lags Used                       4.000000
Number of Observations Used    182.000000
Critical Value (1%)             -3.466800
Critical Value (5%)             -2.877555
Critical Value (10%)            -2.575308
dtype: float64
Failed to Reject Ho - Time Series is Non-Stationary
```

```
Results of Dickey-Fuller Test for ['A2 & A3 -  geschreddert'] in ["['25', '24']"]:
Test Statistic                  -1.503353
p-value                          0.531852
#Lags Used                       7.000000
Number of Observations Used    181.000000
Critical Value (1%)             -3.467005
Critical Value (5%)             -2.877644
Critical Value (10%)            -2.575355
dtype: float64
Failed to Reject Ho - Time Series is Non-Stationary
```

For all 3 categories, it can be seen that the test statistic is negative, but not less than the critical values, which implies that the null hypothesis is not rejected and that all 3 time series are non-stationarity. Thus for all three categories, the time series requires conversion to stationary before it can be used for ARIMA model prediction.

Differencing

Converting a time series to stationary can be achieved with differencing. This involves computing the differences between consecutive observations. The result is a stabilization of the mean of the time series by removing changes in the level, and therefore reducing trend and seasonality. Converting the time series for one of the categories in the cluster ['25', '24'] yields the following plot.

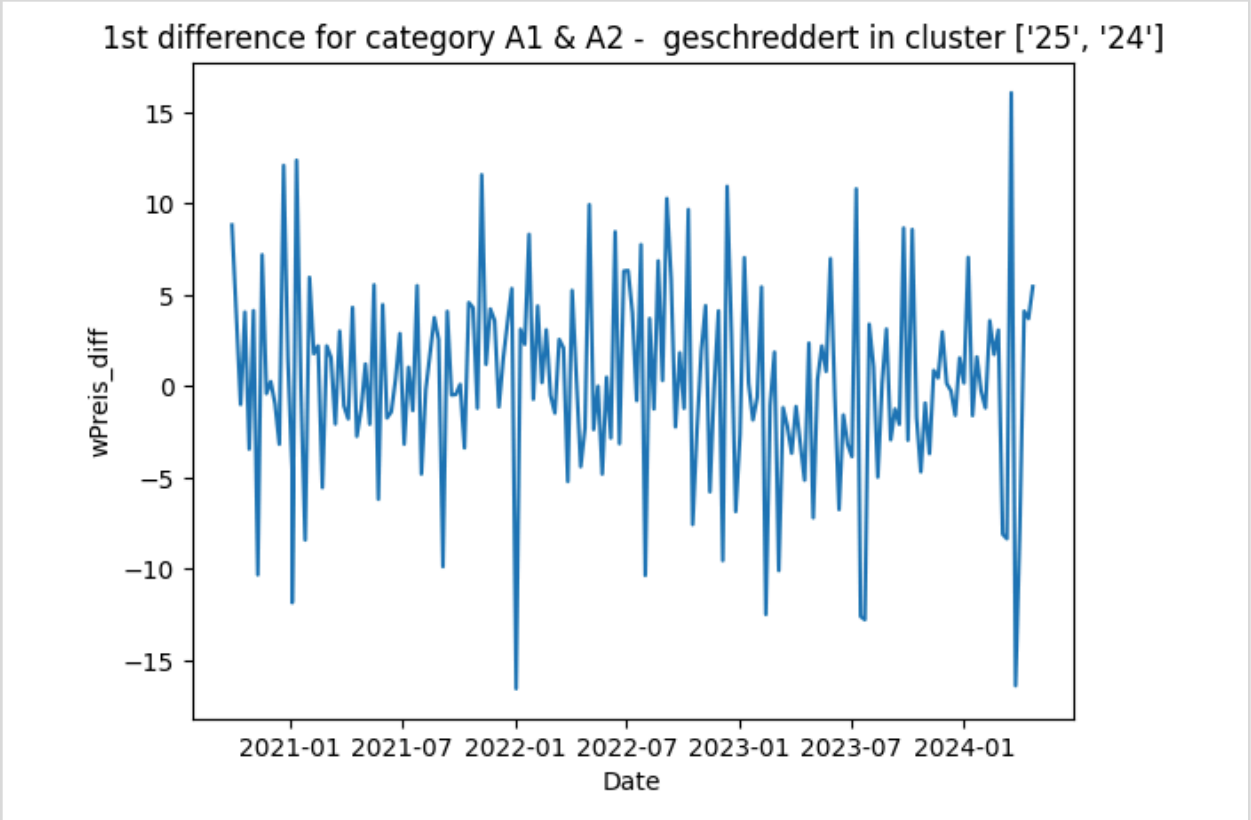


Fig. 4. Plot of 1st order differenced time series for category A1 & A2 - geschreddert in cluster ['25', '24']

The plot of the 1st differenced time series for the same cluster can be seen to have no clear trend or seasonality, which hints towards stationarity. Performing the ADF test again on this 1st differenced time series gives the following results.

Table 2. Results of Dickey-Fuller Test for the 1st differenced time series for category A1 & A2 - geschreddert in the cluster ["['25', '24']"]

```
Results of Dickey-Fuller Test for 1st differenced time series of A1 & A2 -
geschreddert in ['25', '24']:
Test Statistic                -6.348625e+00
p-value                        2.644583e-08
#Lags Used                     3.000000e+00
Number of Observations Used    1.820000e+02
Critical Value (1%)           -3.466800e+00
Critical Value (5%)           -2.877555e+00
Critical Value (10%)          -2.575308e+00
dtype: float64
Reject Ho - Time Series is Stationary
```

Since the test statistic in this case is much less than all the critical values, it means that the null hypothesis has been rejected and that this time series is stationary.

Modeling

To model the time series and make predictions, a specific time series is chosen first to observe the results. Since all of the categories across 10 clusters seem to follow similar time series with some minor deviations, the modeling is first done on one specific category in one cluster to view the results before applying across all the categories and clusters. For an initial analysis and comparison of metrics for the different models, the category of 'A1 & A2 - geschreddert' in the cluster ['25', '24'] is used as the dataset.

Train-test split

Since the time series for the different categories and clusters may have different observations, it may not be ideal to set a specific number of observations for training and test data to compare the performance of the models uniformly. To account for this possible difference in the number of observations, the training test split is based on a ratio of 95-5 % ratio. Thus for each of the models and methods that will be explored, the dataset chosen is split into training and test sets with a split ratio of 0.05. Wherever possible in each model, the model is fit to the

training data set and used to predict for the time steps in the test dataset in the three methods of static multi-step forecasting, rolling window and walk-forward validation. Note that for ARIMA and Prophet only the static and walk-forward options are feasible.

Evaluation

The predictions are compared with the actual values in the test dataset and various error metrics are calculated to evaluate the performance of the method and model. To quantify the accuracy of the model's prediction, the parameters below are evaluated:

- **mae**: Mean Absolute Error, the mean of the absolute value of the difference between the actual values in the test set and that predicted by the model.
- **me**: Mean Error, simple the mean of the difference between the actual values in the test set and that predicted by the model.
- **rmse**: Root Mean Squared Error, the square root of the square of the difference between between the actual values in the test set and that predicted by the model.

The primary metric is the RMSE as it penalizes large errors more heavily compared to MAE because of the squared term in the calculation. This means that RMSE gives higher weight to larger deviations from the actual values, making it more sensitive to outliers or large prediction errors. The square root also scales the errors back to the original units of the forecasted variable, giving us an interpretable measure of the average magnitude of errors.

Direction accuracy

The above error metrics only indicate the magnitude by which the predicted values deviate from that which are expected. They do not indicate whether the direction of prediction,

i.e. whether the price was correctly predicted to be going up or down was correct. This can often be a more useful insight for stakeholders who are interested in conducting transactions and business operations based on price movements in such non-transparent markets. Therefore, an additional metric of direction accuracy is also evaluated, which quantifies the fraction of the number of times that the model predictions were in the right direction (up or down) as the change in the expected values.

The different methods of forecasting are put to test for all the categories in all the clusters to compare the performance metrics. Comparing across clusters can give an insight into whether the price in some geographical regions are more difficult to predict. However, the comparison across clusters has to be performed for the same category present in those clusters.

ARIMA

ARIMA models are widely used in various domains for time series prediction tasks. One common application is in financial forecasting, where ARIMA models are used to predict stock prices, market trends, and economic indicators. They are also used in areas such as demand forecasting, sales prediction, and resource allocation. The model combines three key components: Autoregression (AR), Integration (I), and Moving Average (MA). Each component represents a different aspect of the time series data and contributes to the overall predictive power of the model. Autoregression refers to the dependence of the current value of a variable on its past values. In an ARIMA model, the AR component captures this dependency by regressing the current value of the series on its lagged values. In an autoregression model, the target is calculated using a linear combination of its past values. An autoregressive model of order p can be written as

(1)

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \varepsilon_t,$$

where ε_t is white noise. This can be thought of as a multiple regression with lagged values of y_t as predictors. This is referred to as an AR(p) model – an autoregressive model of order p.

Integration represents the differencing operation applied to the time series data to make it stationary. The Moving Average (MA) component of ARIMA captures the dependency between the current value of the series and the residual errors from previous predictions. It models the short-term fluctuations in the data. Instead of using past values in a regression, the moving average model uses past forecast errors.

(2)

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q},$$

where ε_t is white noise. This is referred to as an MA(q) model, a moving average model of order q .

Combining differencing with autoregression and a moving average model will result in a non-seasonal ARIMA model. The full model can be written as

(3)

$$y_t = c + \phi_1 y'_{t-1} + \ldots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t,$$

This mathematical formulation of ARIMA involves specifying the orders of the AR, I, and MA components, denoted by p, d, and q respectively. The ARIMA model is denoted as ARIMA(p,d,q).

Choice of p and d for ARIMA

The ARIMA model function allows for the explicit specification of the three important parameters – lag (p), differencing (d) and residual error (q). While the 1st order differencing was seen to convert the time series of some of the categories to stationary, when the ADF test is applied for all categories in all the clusters, it is seen that for some categories such as A2 & A3 - geschreddert in ['50', '51', '52', '53'], A2 & A3 - ungeschreddert in ['40', '41', '42', '44', '45', '46', '47'] and A3 - geschreddert in ['40', '41', '42', '44', '45', '46', '47'] are not stationary even after 1st differencing. However, one second order differencing, they are all detected as stationary.

Table 3. Comparison of Dickey-Fuller Test results for the 1st and 2nd differenced time series for all categories in all clusters

Time series	Stationary/ Non-Stationary	1st difference	2nd difference
A2 - geschreddert in ['25', '24']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['25', '24']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['25', '24']	Non-Stationary	Stationary	Stationary
A3 - geschreddert in ['80', '81', '82', '83', '84', '85', '93', '94']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['80', '81', '82', '83', '84', '85', '93', '94']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['80', '81', '82', '83', '84', '85', '93', '94']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['70', '71', '73', '74', '75', '76']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['50', '51', '52', '53']	Non-Stationary	**Non-Stationary**	Stationary
A3 - geschreddert in ['50', '51', '52', '53']	Non-Stationary	Stationary	Stationary
A2 & A3 - ungeschreddert in ['40', '41', '42', '44', '45', '46', '47']	Non-Stationary	**Non-Stationary**	Stationary
A1 & A2 - ungeschreddert in ['40', '41', '42', '44', '45', '46', '47']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['40', '41', '42', '44', '45', '46', '47']	Non-Stationary	Stationary	Stationary
A3 - geschreddert in ['40', '41', '42', '44', '45', '46', '47']	Non-Stationary	**Non-Stationary**	Stationary
A1 & A2 - geschreddert in ['40', '41', '42', '44', '45', '46', '47']	Non-Stationary	Stationary	Stationary
A2 - geschreddert in ['40', '41', '42', '44', '45', '46', '47']	Non-Stationary	Stationary	Stationary

A2 - geschreddert in ['26', '27', '28']	Non-Stationary	Stationary	Stationary
A3 - geschreddert in ['26', '27', '28']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['26', '27', '28']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['26', '27', '28']	Non-Stationary	Stationary	Stationary
A2 & A3 - ungeschreddert in ['48', '49']	Non-Stationary	Stationary	Stationary
A3 - ungeschreddert in ['48', '49']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['48', '49']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['48', '49']	Non-Stationary	Stationary	Stationary
A3 - geschreddert in ['48', '49']	Non-Stationary	Stationary	Stationary
A2 - geschreddert in ['20', '21', '22', '23']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['20', '21', '22', '23']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['20', '21', '22', '23']	Non-Stationary	**Non-Stationary**	Stationary
A2 & A3 - geschreddert in ['10', '11', '12', '13', '14', '15', '16']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['10', '11', '12', '13', '14', '15', '16']	Non-Stationary	Stationary	Stationary
A2 - geschreddert in ['10', '11', '12', '13', '14', '15', '16']	Non-Stationary	Stationary	Stationary
A2 & A3 - geschreddert in ['29', '30', '31', '38', '39']	Non-Stationary	Stationary	Stationary
A1 & A2 - geschreddert in ['29', '30', '31', '38', '39']	Non-Stationary	Stationary	Stationary
A2 - geschreddert in ['29', '30', '31', '38', '39']	Non-Stationary	Stationary	Stationary

Thus for using ARIMA model for forecasting the prices for all clusters in general, it is better to use the second order differenced version of all the time series to be able to compare their performances. In other words, for all the ARIMA model functions the parameter of d, for differencing will be set as 2.

Since it was observed that the time series may have correlations for up to around 5 lags, a range of values for p from 1 to 10 are tried for one cluster. When the model is fit on the training set for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert', for the values p = 1, d = 2 the following result is obtained.

Table 4. ARIMA model results for category 'A1 & A2 - geschreddert' in cluster ['25', '24'] for the values p = 1, d = 2

```
  SARIMAX Results
==========================================================================================
Dep. Variable:                    wPreis   No. Observations:                  178
Model:                 ARIMA(1, 0, 0)   Log Likelihood                -543.126
Date:                Thu, 16 May 2024   AIC                           1092.252
Time:                        12:53:32   BIC                           1101.797
Sample:                             0   HQIC                          1096.123
                                  - 178
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8228     17.367      0.047      0.962     -33.215      34.861
ar.L1          0.9854      0.011     87.352      0.000       0.963       1.008
sigma2        25.6555      2.417     10.614      0.000      20.918      30.393
===================================================================================
Ljung-Box (L1) (Q):                  13.83   Jarque-Bera (JB):                 4.71
Prob(Q):                              0.00   Prob(JB):                         0.09
Heteroskedasticity (H):               0.99   Skew:                            -0.24
Prob(H) (two-sided):                  0.96   Kurtosis:                         3.63
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
AIC: 1092.2518563602057
BIC: 1101.797207011082
```

In the results above, the key metrics to note is the AIC (Akaike Information Criterion). This is a metric that measures how well the model has been fit to the data. Lower AIC values means a better fit for the model, and it can be used to compare different models.

To arrive at a single value of p for standardizing the predictions and comparing across all categories and clusters, the AIC values for p values up to 10 are compared as below.

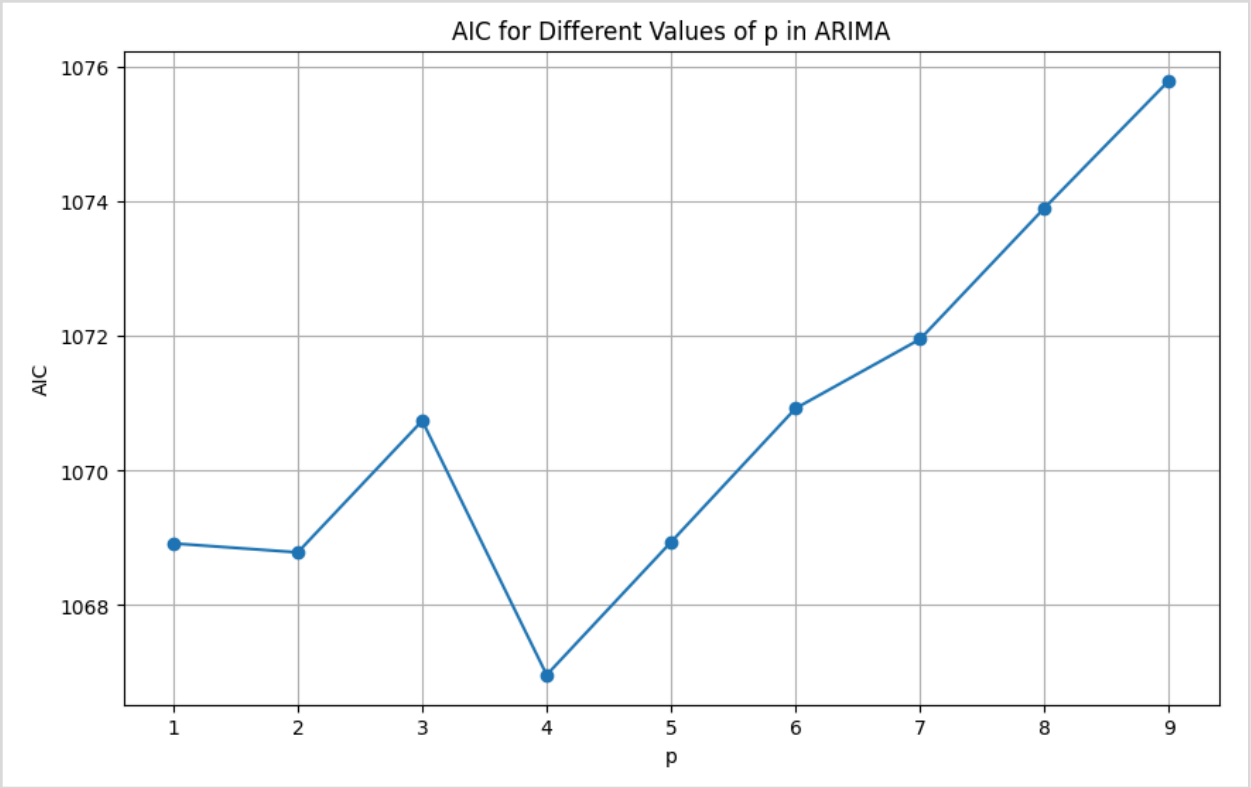


Fig. 5. Plot of AIC for different p values for category A1 & A2 - geschreddert in cluster ['25', '24']

It is seen that a p-value of 4 produces the lowest AIC value and thus p is set to 4 for all the ARIMA predictions. Generating the results of the ARIMA model for p = 4, d = 1 reveals a slightly lesser AIC value (1089.76) than the previous case of p = 1, d = 0, as shown below

Table 5. ARIMA model results for category 'A1 & A2 - geschreddert' in cluster ['25', '24'] for the values p = 4, d = 1

SARIMAX Results

Dep. Variable:	wPreis	No. Observations:	178
Model:	ARIMA(4, 2, 0)	Log Likelihood	-539.879
Date:	Thu, 16 May 2024	AIC	1089.759
Time:	13:13:37	BIC	1105.611
Sample:	0	HQIC	1096.189
	- 178		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-1.1276	0.078	-14.513	0.000	-1.280	-0.975
ar.L2	-0.9595	0.109	-8.815	0.000	-1.173	-0.746
ar.L3	-0.6097	0.106	-5.758	0.000	-0.817	-0.402
ar.L4	-0.1785	0.081	-2.212	0.027	-0.337	-0.020
sigma2	26.7889	2.420	11.068	0.000	22.045	31.533

```
===================================================================================
Ljung-Box (L1) (Q):                   0.08   Jarque-Bera (JB):                 8.10
Prob(Q):                              0.77   Prob(JB):                         0.02
Heteroskedasticity (H):               1.21   Skew:                            -0.32
Prob(H) (two-sided):                  0.47   Kurtosis:                         3.83
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
AIC: 1089.758922870018
BIC: 1105.6113428452088
```

Static (multi-step) forecasting: To use the ARIMA model for static, multi-step prediction, the ARIMA model from the statsmodels package is used with a lag of 4 and differencing of 2. The dataset is divided into training and testing sets with a split ratio of 0.05 for the latter. This trained model is then used to forecast for steps equal to the number of observations in the test set.

The rolling window approach as described in the methodology is not possible for ARIMA as it is not dependent on an input sequence for the forecast. Moreover, the ARIMA forecast function does not require an "input" other than the steps, so the only options are either to do a static forecast, or to conduct a walk-forward validation.

Walk-forward validation: To apply the walk-forward method of forecasting, the same ARIMA model from the statsmodels package can be used with a lag of 4 and differencing of 2 are applied. The model is trained with a “history” dataset which is initialized as the original training dataset containing the 0.95 split of the data. Then in a loop, each step trains the model on the history, predicts one output step based on that training model and appends the new observation from the test set into the history, essentially elongating the history at every step.

Decision Tree

Decision trees have been widely used in various machine learning tasks, including time series forecasting. These are non-parametric supervised learning methods which can be used for both classification and regression tasks. In the case of time series forecasting, decision trees can still be used in an auto-regressive manner, where the future of a random variable is viewed as dependent on its past values.

Decision trees offer a flexible and interpretable option for modeling and predicting future trends in cases where the data may exhibit complex patterns and irregularities. They work by partitioning the feature space into a set of disjoint regions based on recursive binary splitting of the feature space, where at each step, the algorithm selects the feature and the split point that optimally separates the data according to some criterion, such as maximizing information gain or minimizing impurity. The goal is to create splits that lead to accurate predictions by following the decision rules at each split based on the values of the input features. For time series forecasting, decision trees can be adapted by incorporating lagged variables as features.

Static (multi-step) forecasting: To use decision trees for a static (multi-step) forecasting, the ForecasterAutoreg class from the SKforecast package can be used. This class changes any regressor that is normally used with the scikit-learn API into a recursive autoregressive forecaster that can be used for multi-step forecasting. The ForecasterAutoreg class has an option to set a regressor parameter as DecisionTreeRegressor and also another parameter lag which can be set to 4.

Rolling window forecasting: To use decision tree-based machine learning for rolling, the data has to be restructured to look like a supervised learning problem. This means given a sequence of values in the time series dataset, a set number of previous time steps are used as input variables (X)and the next time step (y) as the output variable. The order of the observations has to be preserved, and must continue to be done in the training.

For the number of the previous steps, the window, the lag of 4 is used based on correlation plots and also observing the lag with the lowest AIC value for the ARIMA results. The training set in this case will consist of a feature variable X which is a list of sequences of 4 values and the target variable y, which is the next value after the sequence. The target values will begin with the 5th value in the original training dataset and end with the last one in the same set.

The model is created using the DecisionTreeRegressor from the sklearn.tree package and is trained on this X and y once. Then in each step of a loop that runs the length of the test dataset, the predict function is called on the feature variable testX containing the preceding 4 values as a rolling window to get a corresponding predicted y value. This is repeated until predicted y values are forecast to compare with each of the steps in the original test dataset.

Walk-forward validation: Similar to the rolling window method, for using decision trees in the walk-forward validation methods of forecasting, the data is restructured with the set number of previous time steps as input variables and the next time step as the output variable to prepare for supervised learning. The difference is that in each step of the loop that runs the length of the test data set, not only are predictions made on the rolling window of 4 preceding values but the model itself is trained on the entire history updated to include the new observation from the test data.

Random Forest

Random Forest is an ensemble learning technique that combines the predictions of multiple decision trees to improve overall performance and robustness. Developed by Brieman (Breiman 2001), Random Forest works by constructing a collection of decision trees, each trained on a random subset of the training data and a random subset of the input features. It is based on the bootstrap aggregation (bagging) (James et al., n.d.) of decision trees. Bagging involves the creation of decision trees, where each is constructed from a different bootstrap sample in the training dataset. A bootstrap sample is one where an observation may appear more than once in the sample. This type of approach using bagging is an effective ensemble algorithm as each decision tree is fit on a slightly different training dataset, which gives slightly different performance for each. Predictions from the trees are averaged across all decision trees, to produce better performance than that given by any single tree.

One of the major strengths of Random Forest is its ability to work with high-dimensional data and noisy features effectively. But Random Forest models may also suffer from limitations such as computational overhead and the potential for overfitting, especially when dealing with imbalanced data or highly correlated features.

Static (multi-step) forecasting: Similar to the case of Decision Tree, to use the Random Forest model for a static (multi-step) forecasting, the ForecasterAutoreg class from the SKforecast package is used, with the regressor parameter set as RandomForestRegressor and also the parameter lag set to 4. The model is then trained on the original training dataset once to predict for the time steps corresponding to the test data set.

Rolling window forecasting: The use of the Random Forest Model to forecast in the rolling window method, a process similar to that of Decision Trees is adopted. The training data is restructured with a list of sequences of 4 values and the target variable y, which is the next value after the sequence in a supervised learning problem, preserving the order. The model is created using the RandomForestRegressor from the sklearn.ensemble package and is trained on this X and y once. Then in each step of a loop that runs the length of the test dataset, the predict function is called on the feature variable testX containing the preceding 4 values as a rolling window to get a corresponding predicted y value, repeating until forecasts are made to compare with each of the steps in the test dataset.

Walk-forward validation: For walk-forward validation too, a process similar to that of Decision Trees is adopted. The data is restructured to prepare for supervised learning and in each step of the loop predictions made on the rolling window of 4 preceding values with the model being retrained on the entire history updated to include the new observation from the test data.

XGBoost

XGBoost, which stands for eXtreme Gradient Boosting, is a machine learning algorithm known for its high performance in a variety of predictive modeling tasks, including time series forecasting. It is an efficient implementation of gradient boosting for classification and regression problems and is an ensemble learning technique that builds a collection of decision trees sequentially, with each subsequent tree focusing on correcting the errors made by the previous ones in an iterative approach. By employing gradient boosting techniques, XGBoost

optimizes the learning process by iteratively adding new trees that minimize a specified loss function. This results in a model that can handle large datasets and high-dimensional feature spaces efficiently.

Static (multi-step) forecasting: Similar to the case of Decision Tree and Random Forest, to use XGBoost model for a static (multi-step) forecasting, the ForecasterAutoreg class from the SKforecast package is used, with the regressor parameter set as XGBRegressor and the parameter lag set to 4. The model is then trained on the original training dataset once to predict for the time steps corresponding to the test data set.

Rolling window forecasting: The use of the XGBoost model to forecast in the rolling window method also follows a process similar to that of Decision Trees and Random Forests. The data is prepared for supervised learning by restructuring it as a list of sequences of 4 previous time steps as the input (X) and the next time step is the output (y). The model is created using the XGBRegressor from the xgboost package and is trained on this X and y once. Then in each step of a loop that runs the length of the original test dataset in a rolling window, the predict function is called on the feature variable testX containing the preceding 4 values as sequence to get a corresponding predicted y value. This is repeated as the window of fixed size of 4 moves along until predicted y values are forecast to compare with each of the steps in the original test dataset.

Walk-forward validation: For walk-forward validation too, a process similar to that of Decision Trees and Random Forest is adopted. The data is restructured to prepare for supervised

learning and in each step of the loop predictions made on the rolling window of 4 preceding values with the model being retrained on the entire history updated to include the new observation from the test data.

Prophet

Prophet is a time series forecasting model developed by Facebook that is based on an open-source library. It is designed to automatically find a good set of hyperparameters in order to make accurate forecasts for data with trends and seasonal structure. According to the documentation developed by Facebook, Prophet implements what is known as an “additive time series forecasting model”, where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. (Package ‘prophet’, 2019). Thus it is capable of forecasting data with strong seasonal patterns, multiple seasonality, and holidays.

Prophet is used in time series forecasting due to its simplicity and ease of use, with the automatic feature selection allowing for quick building and evaluation of forecasting models without the need for extensive parameter tuning. Prophet also offers uncertainty estimation, which means that it is able to quantify the uncertainty associated with each forecasted value through confidence intervals around the predicted values.

Static (multi-step) forecasting: The Prophet model effectively does a static prediction by default. It does not have an option to predict using a rolling window approach as the model simply outputs the predictions for the number of steps specified after taking in a set input. The recommended approach with Prophet is to create a new dataframe using the "*make_future_dataframe*" method with the number of periods specified (which can be assigned

to the length of the test data set). To get the exact dates matching with that of the test data set, the frequency has to be specified as weekly, with a start of Monday. Predicting on the newly created dataframe outputs a standard Prophet dataframe with various metrics for the prediction. The important metric to consider is "*yhat*", which is the actual prediction value for the period steps.

Walk-forward validation: For the walk-forward validation approach, the same process can be applied by applying the prophet model to predict just one step ahead and by ensuring that the model is retrained at every step by appending the new observation to the training data set.

Results

For each of the models, the actual and predicted prices were compared using the different methods to identify the errors. The key results are as follows:

ARIMA Static Forecast

The ARIMA model from the statsmodels package is used, with a lag of 4 and differencing of 2, trained on the training dataset and forecasts are made for steps equal to the number of observations in the test set. Getting the predictions for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following plot and metrics.

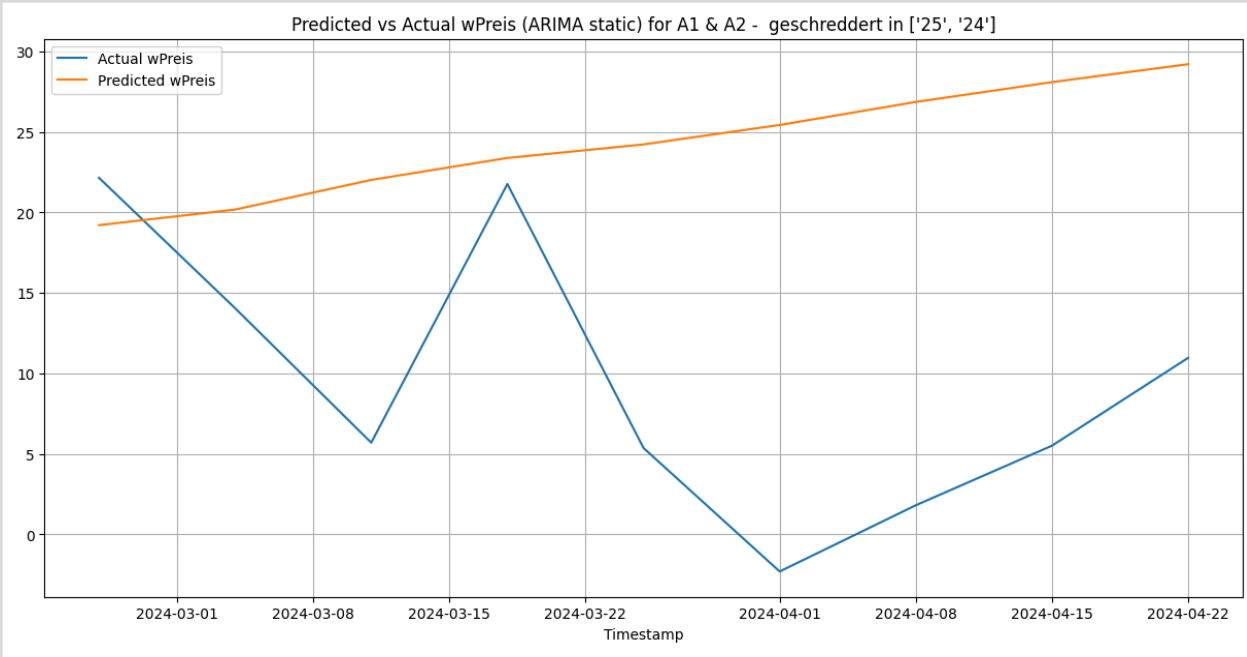


Fig. 6. Predicted vs. Actual wPreis values using ARIMA (static multi-step forecasting) A1 & A2 - geschreddert in cluster ['25', '24']

Table 6. ARIMA static results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (ARIMA static) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 15.497
ME: 14.843
RMSE: 17.983
Direction Accuracy: 0.500
```

The static forecasts produce an RMSE of 17.983 which is comparable to the scale of the expected values. The direction accuracy suggests that the forecasts are in the right direction 50% of the time which is not ideal and better accuracies are desired. These metrics will need to be compared with that produced by other models to see which ones produce more accurate forecasts

ARIMA Walk-forward Validation

To predict the waste price using ARIMA through the walk-forward validation method, the same ARIMA model from the statsmodels package is used with the parameters of 4 for lag

and differencing of 1. When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

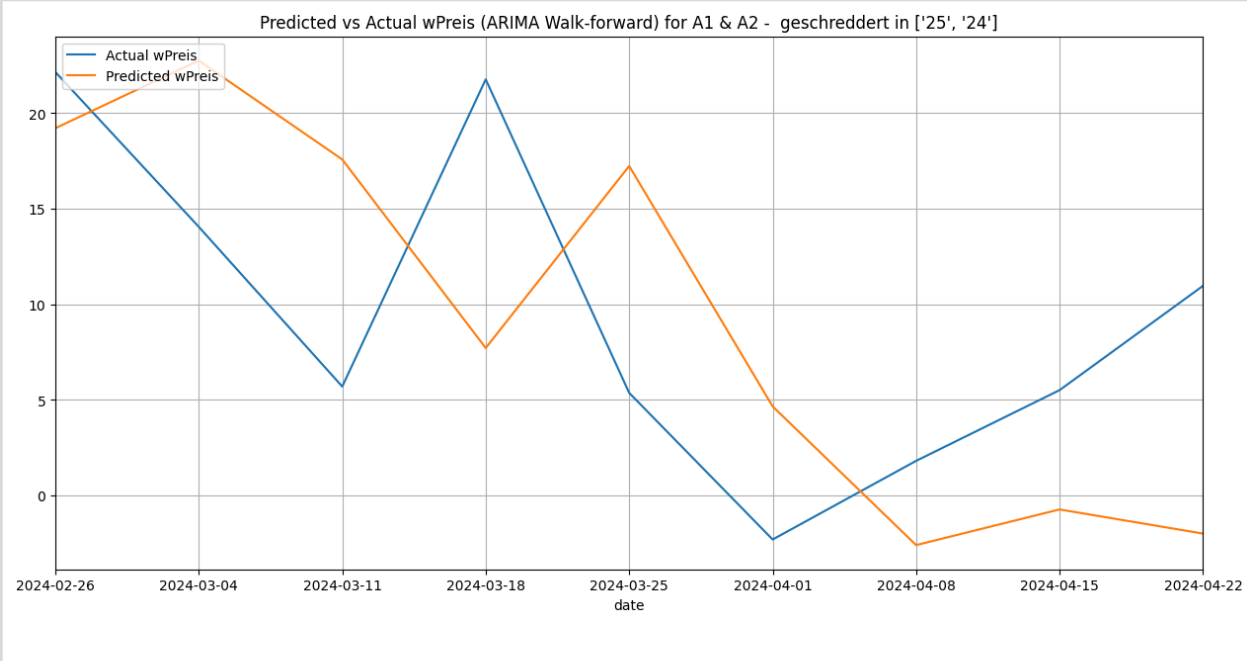


Fig. 7. Predicted vs. Actual wPreis values using ARIMA (walk-forward) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 7. ARIMA walk-forward results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (ARIMA Walk-forward) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 8.889
ME: -0.134
RMSE: 9.652
Direction Accuracy: 0.375
```

As can be seen from the RMSE of 9.652 for the walk-forward validation method compared to the RMSE value of 17.983 from the static method, the former is more accurate in terms of how far off the forecasts are from the actual observed values. However, the walk-forward validation performs worse than the static approach in predicting the direction with an accuracy of 37.5% for the former compared to 50% for that of the latter. It should also be noted that there are only 9 time steps in this particular series, and other series will also have

approximately the same number of time steps due to the 0.05 ratio for the test data set. This means that the direction accuracy percentage is based only on these time steps and may improve with more observations in the test data set.

The same two methods of forecasting are put to test for all the categories in all the clusters to compare the performance metrics. Comparing the metrics for the same category across clusters can give an insight into whether some geographical regions are easier to predict and the specific models that may be employed for prediction there.

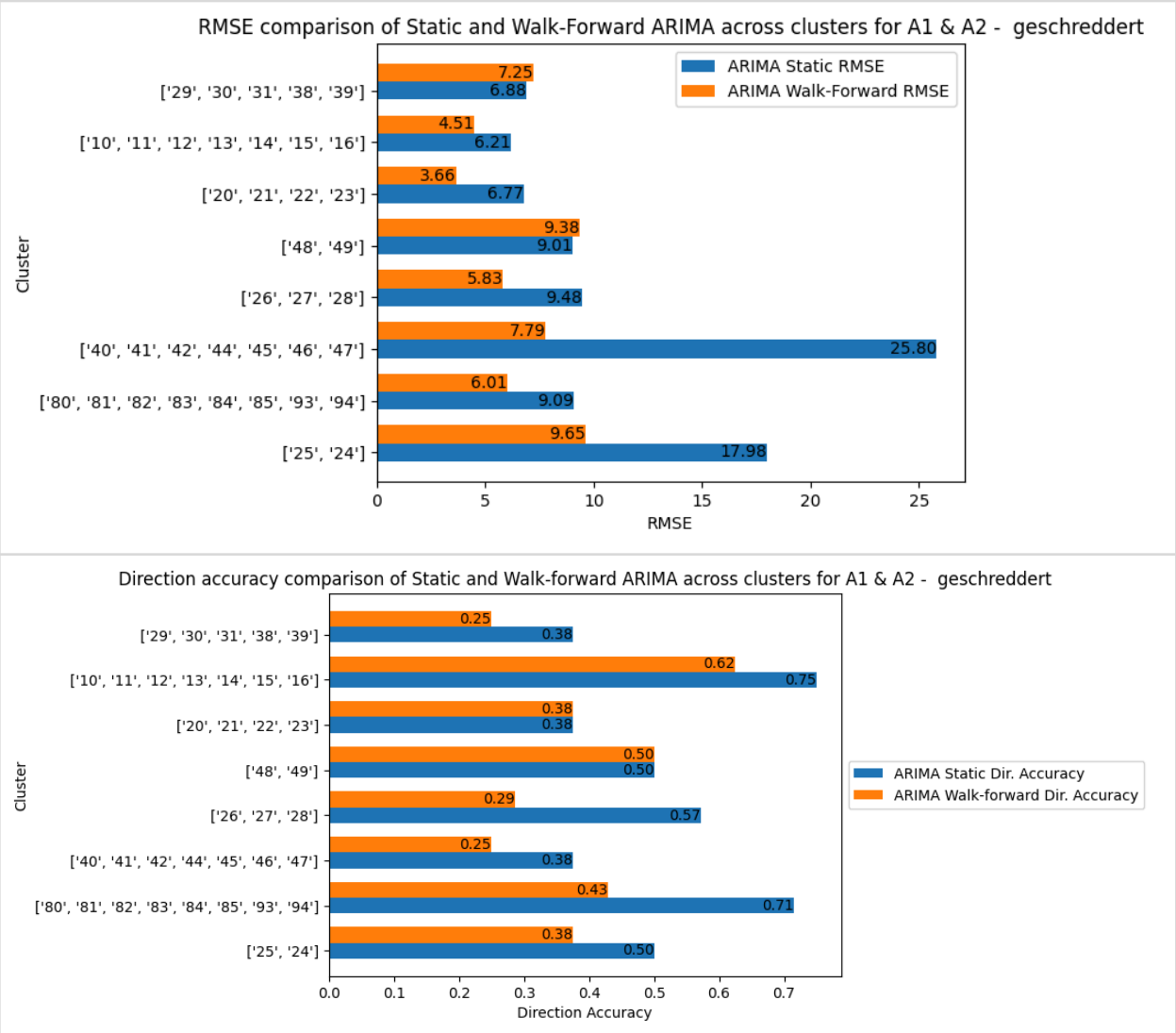


Fig. 8. Comparison of RMSE and Direction Accuracy between Static and Walk-forward methods using ARIMA model across all clusters for category A1 & A2 - geschreddert.

The comparison reveals that the walk-forward validation method of forecasting has lower RMSE in almost all cases except for 2 clusters. The actual magnitude of the error is the lowest in the cluster ['20', '21', '22', '23'] at 3.66. Direction accuracy tends to be higher with static forecast prediction, except in 2 clusters, with the highest being 75% for the cluster ['10', '11', '12', '13', '14', '15', '16'].

Decision Tree Static Forecast

Using the DecisionTreeRegressor in the ForecasterAutoreg class from the SKforecast package and setting the lag parameter set as 4, the model is trained once on the training set and forecasts are generated for the test set. When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

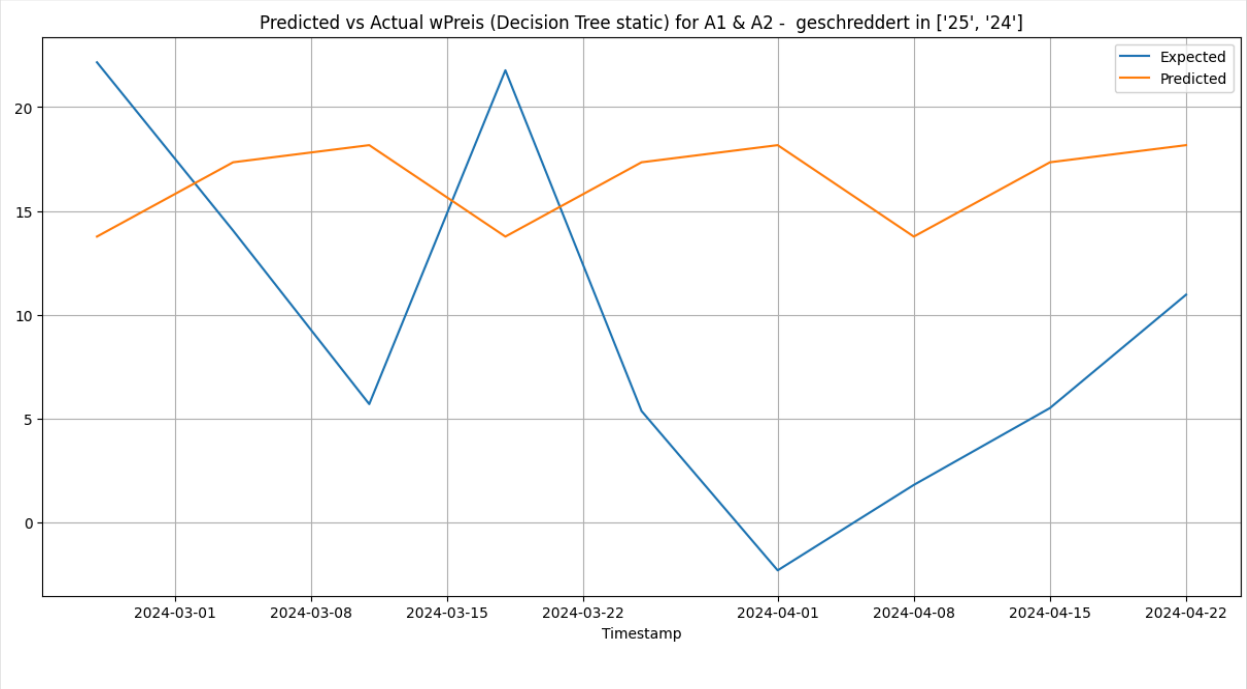


Fig. 9. Predicted vs. Actual wPreis values using Decision Tree (static) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 8. Decision Tree static results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Decision Tree static) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 10.617
ME: 6.975
RMSE: 11.534
Direction Accuracy: 0.250
```

The RMSE value of 11.534 remains comparable to the scale of the wPreis values while the direction accuracy is less than desired at 25%.

Decision Tree Rolling Window

In the case of the rolling window approach, a sequence of previous time steps is used as the input (X) and the next time step is the output (y) in a supervised learning problem. The order of the observations should be maintained, when the data set is used in training the supervised model.

For the number of the previous steps, the window, the lag of 4 is used based on what was observed in the ACF and PACF plots and the ARIMA model. The training set in this case will consist of a feature variable X which is a list of sequences of 4 values and the target variable y, which is the next value after the sequence. The target values will begin with the 5th value in the original training dataset and end with the last one in the same set.

The model is trained on this X and y once. Then in each step of a loop that runs the length of the original test dataset, the predict function is called on the feature variable testX containing the preceding 4 values as sequence to get a corresponding predicted y value. This is repeated until we have a predicted y value to compare with each of the steps in the original test dataset.

When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

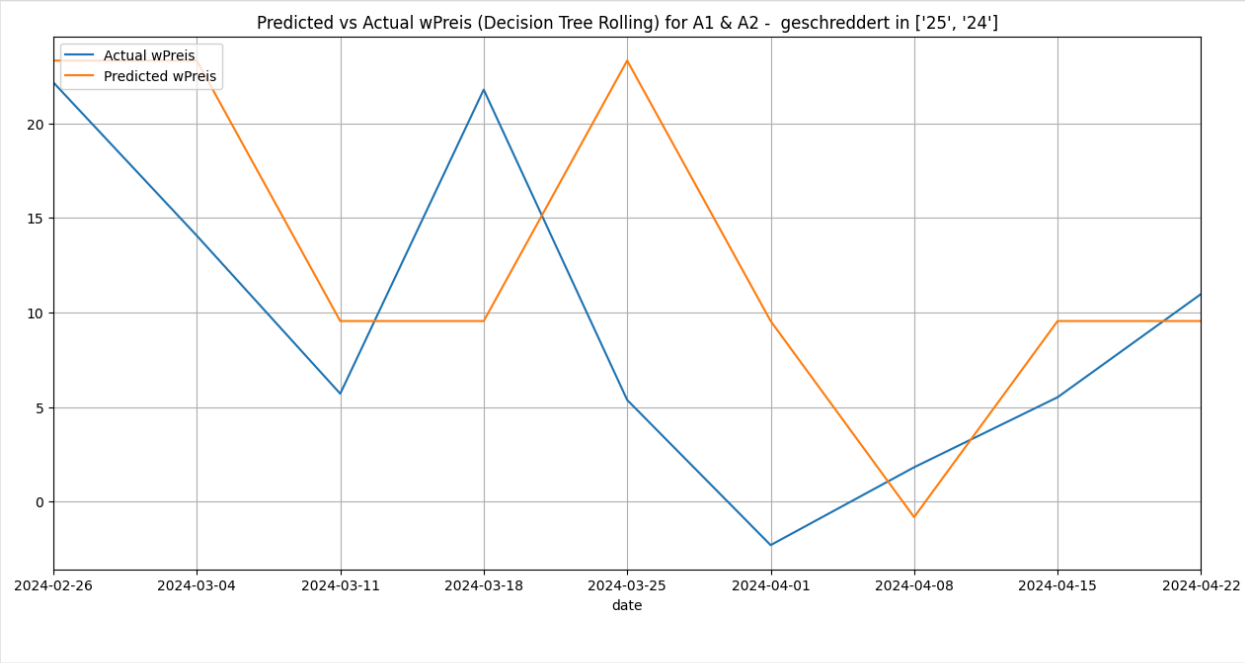


Fig. 10. Predicted vs. Actual wPreis values using Decision Tree (Rolling) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 9. Decision Tree Rolling results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Decision Tree Rolling) for A1 & A2 - geschreddert in ['25',
'24']
MAE: 7.148
ME: 3.527
RMSE: 9.057
Direction Accuracy: 0.375
```

As can be observed, both the RMSE of 9.057 and Direction Accuracy of 37.5% are improvements over the static method.

Decision Tree Walk-forward Validation

In the case of the walk-forward validation approach, the key difference is that the training of the model on the feature variable X and target variable y happens at every step of the loop,

with both being updated with the new observation from the original test data set. This will still include the lag of 4. The predict function is also called at each step of the loop on the updated feature variable testX. When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

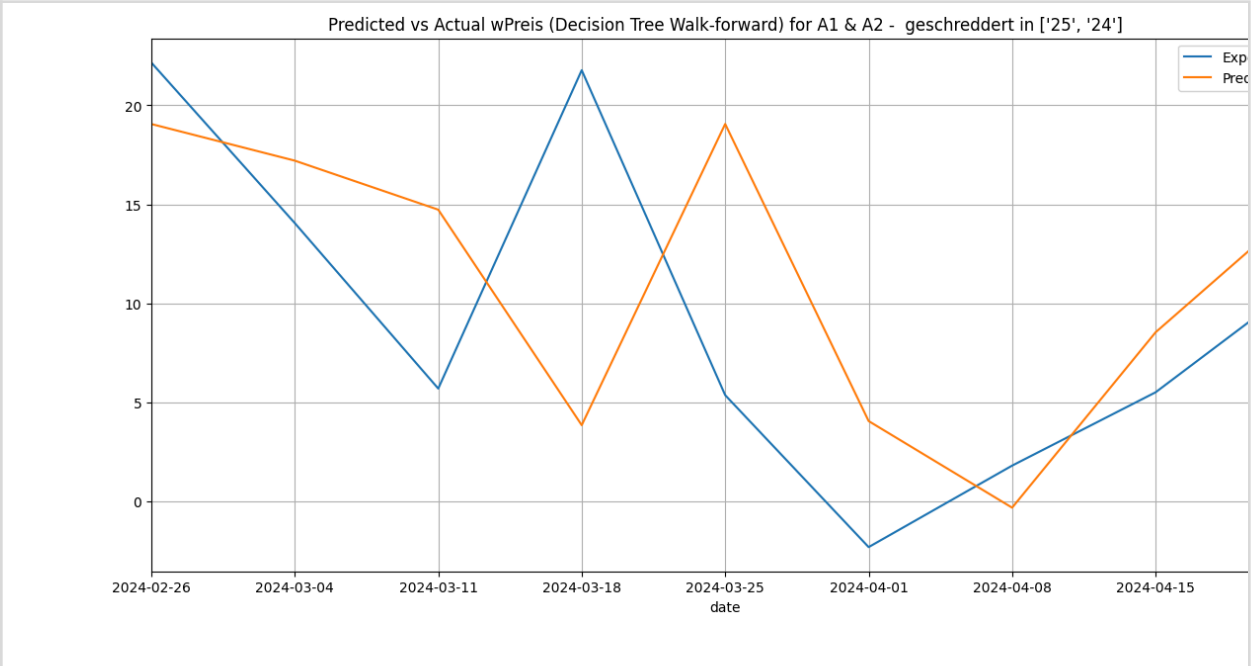


Fig. 11. Predicted vs. Actual wPreis values using Decision Tree (Walk-forward) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 10. Decision Tree Walk-forward results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Decision Tree Walk-forward) for A1 & A2 - geschreddert in ['25',
'24']
MAE: 6.921
ME: 1.779
RMSE: 8.684
Direction Accuracy: 0.625
```

The RMSE of 8.684 and Direction Accuracy of 62.5% are both the best out of the 3 methods for this particular time series. Comparing the metrics for the same category across clusters gives the following results.

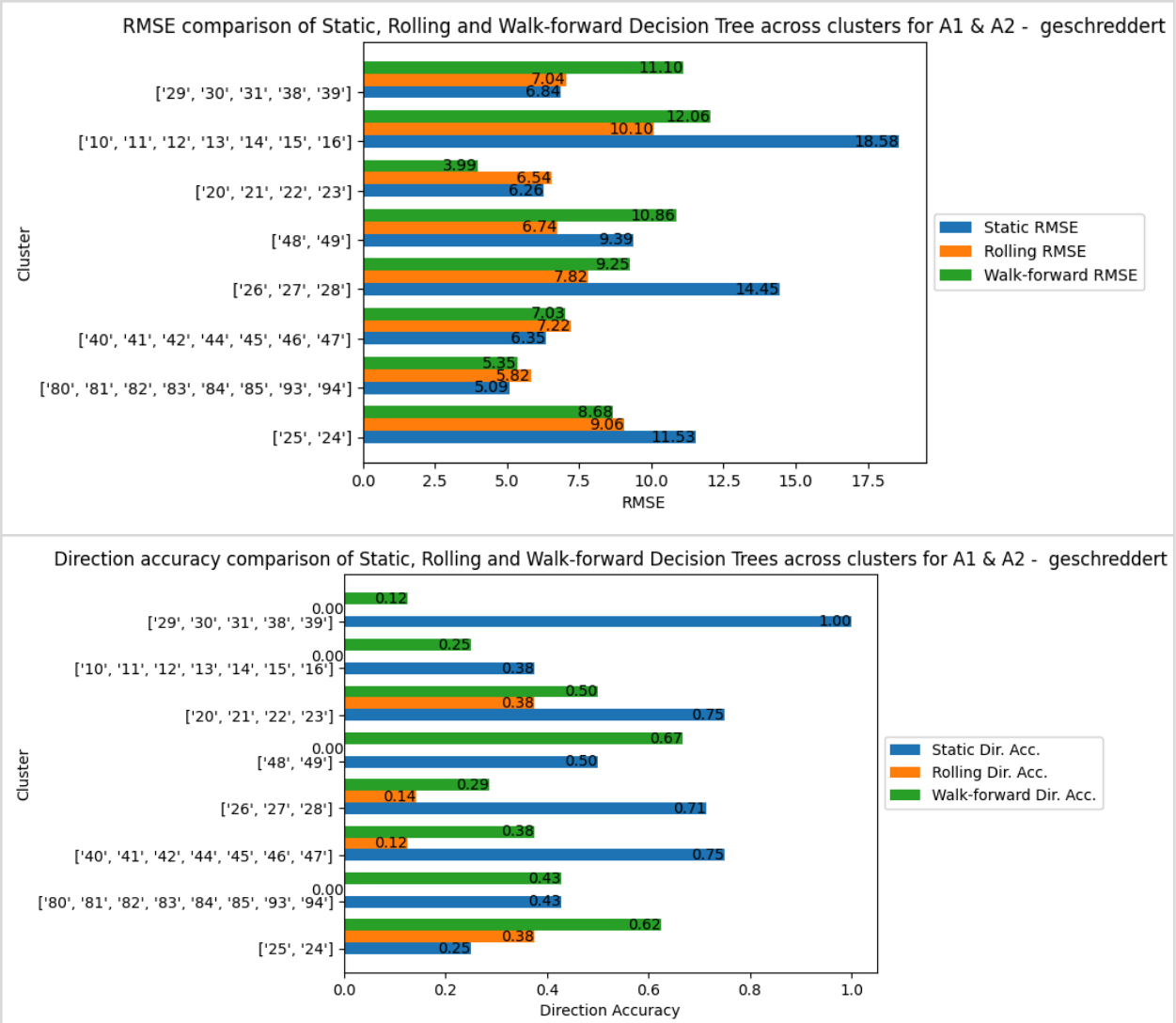


Fig. 12. Comparison of RMSE and Direction Accuracy between Static, Rolling and Walk-forward methods using Decision Trees across all clusters for category A1 & A2 - geschreddert.

The comparison does indicate that any one method produces the lowest RMSE in general. The lowest RMSE obtained is 3.99 for the walk-forward validation method of forecasting in the cluster "['20', '21', '22', '23']". Direction accuracy tends to be higher with static forecast prediction, except in some clusters, with the highest being for the cluster "['29', '30', '31', '38', '39']" where the forecasts are 100% correct in terms of direction. In the forecasts for some time

series, the direction was wrong 100% of the time, which is the reason for the missing bars in the plot above.

Random Forest Static Forecast

Similar to the case of Decision Tree, the RandomForest model will need to be trained on the original training dataset once to predict for the time steps corresponding to the test data set. The same ForecasterAutoreg class from the SKforecast package is used with the regressor being set as RandomForestRegressor and with a lag of 4. When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following results.

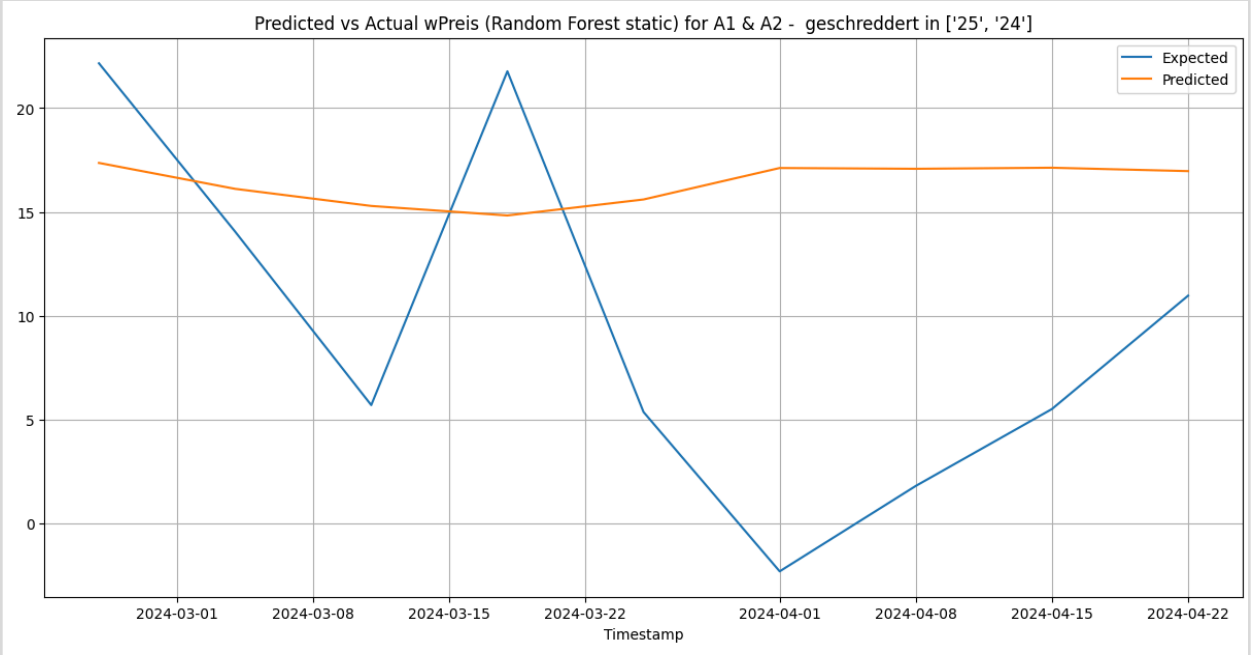


Fig. 13. Predicted vs. Actual wPreis values using Random Forest (Static) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 11. Random Forest Static results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Random Forest static) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 9.543
ME: 6.936
RMSE: 10.814
Direction Accuracy: 0.375
```

The RMSE value of 10.814 is reasonable and within the same scale as the wPreis values while the direction accuracy of 37.5% is less than what is desired.

Random Forest Rolling Window

Similar to the case for Decision Tree, in the rolling window approach, the training set in this case will consist of a feature variable X which is a list of sequences of 4 values and the target variable y, which is the next value after the sequence. The model is trained on this X and y once and a loop runs the length of the original dataset, in each step of which the predict function is called on the feature variable testX containing the preceding 4 values as sequence to get a corresponding predicted y value. This is repeated until we have a predicted y value to compare with each of the steps in the original test dataset. When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following results.

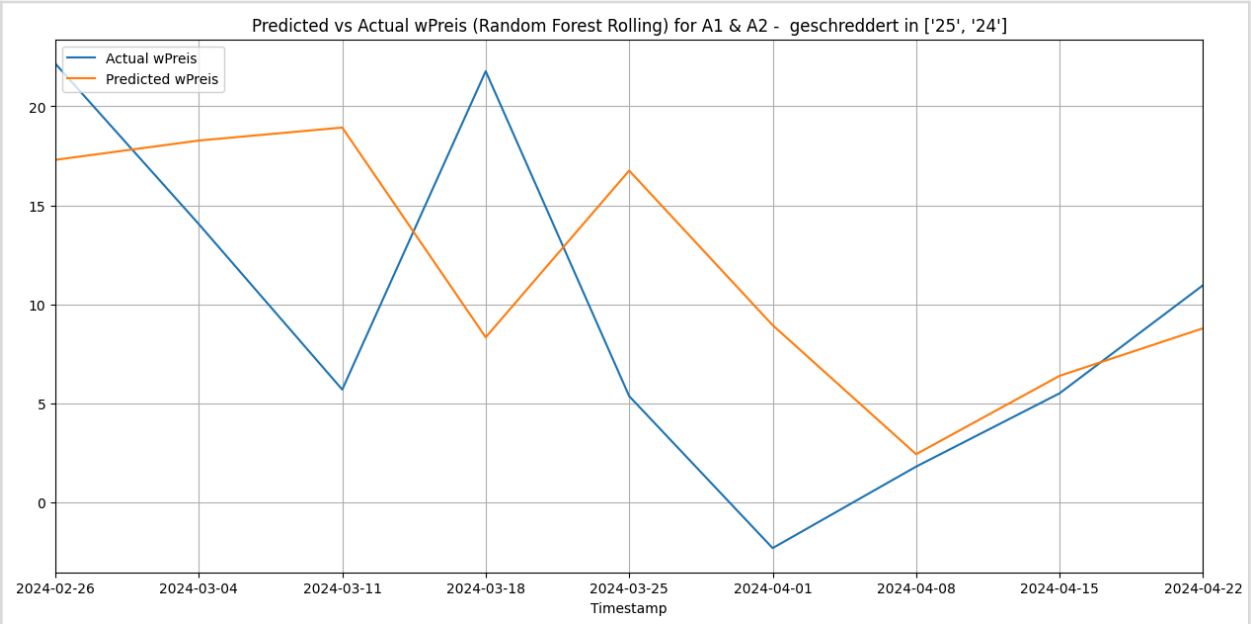


Fig. 14. Predicted vs. Actual wPreis values using Random Forest (Rolling) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 12. Random Forest Rolling results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Random Forest Rolling) for A1 & A2 -  geschreddert in ['25',
'24']
MAE: 6.894
ME: 2.346
RMSE: 8.555
Direction Accuracy: 0.375
```

The RMSE value of 8.555 is an improvement over the static method while the direction accuracy of 37.5% is not.

Random Forest Walk-forward Validation

Similar to the same procedure for the Decision Tree, in the walk-forward validation approach, the training of the model on the feature variable X and target variable y happens at every step of the loop, with both being updated with the new observation from the original test data set. This will still include the lag of 4. The predict function is also called at each step of the loop on the updated feature variable testX. When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

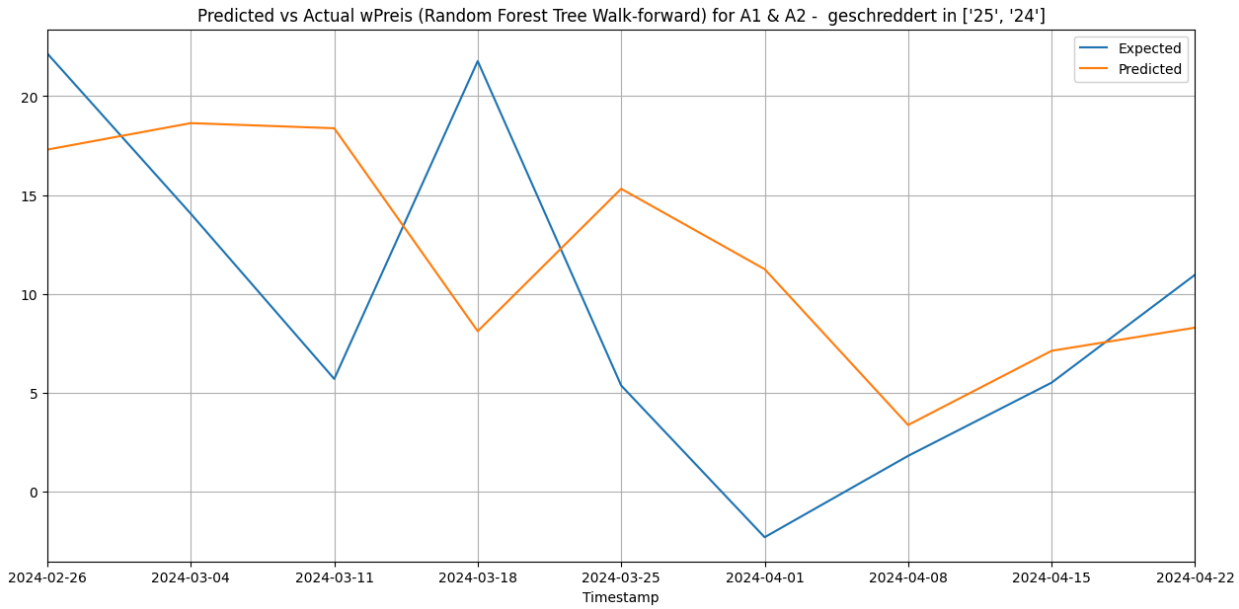


Fig. 15. Predicted vs. Actual wPreis values using Random Forest (Walk-forward) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 13. Random Forest Walk-forward results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Random Forest Walk-forward) for A1 & A2 - geschreddert in ['25',
'24']
MAE: 7.234
ME: 2.526
RMSE: 8.733
Direction Accuracy: 0.500
```

The RMSE value of 8.733 from the walk-forward validation method is better than that produced by the static method not as good as the one from the rolling method. The Direction Accuracy of 50% however is the best out of the three methods, although better accuracies than this are desired.

When the three methods of forecasting are compared across clusters the following result is obtained.

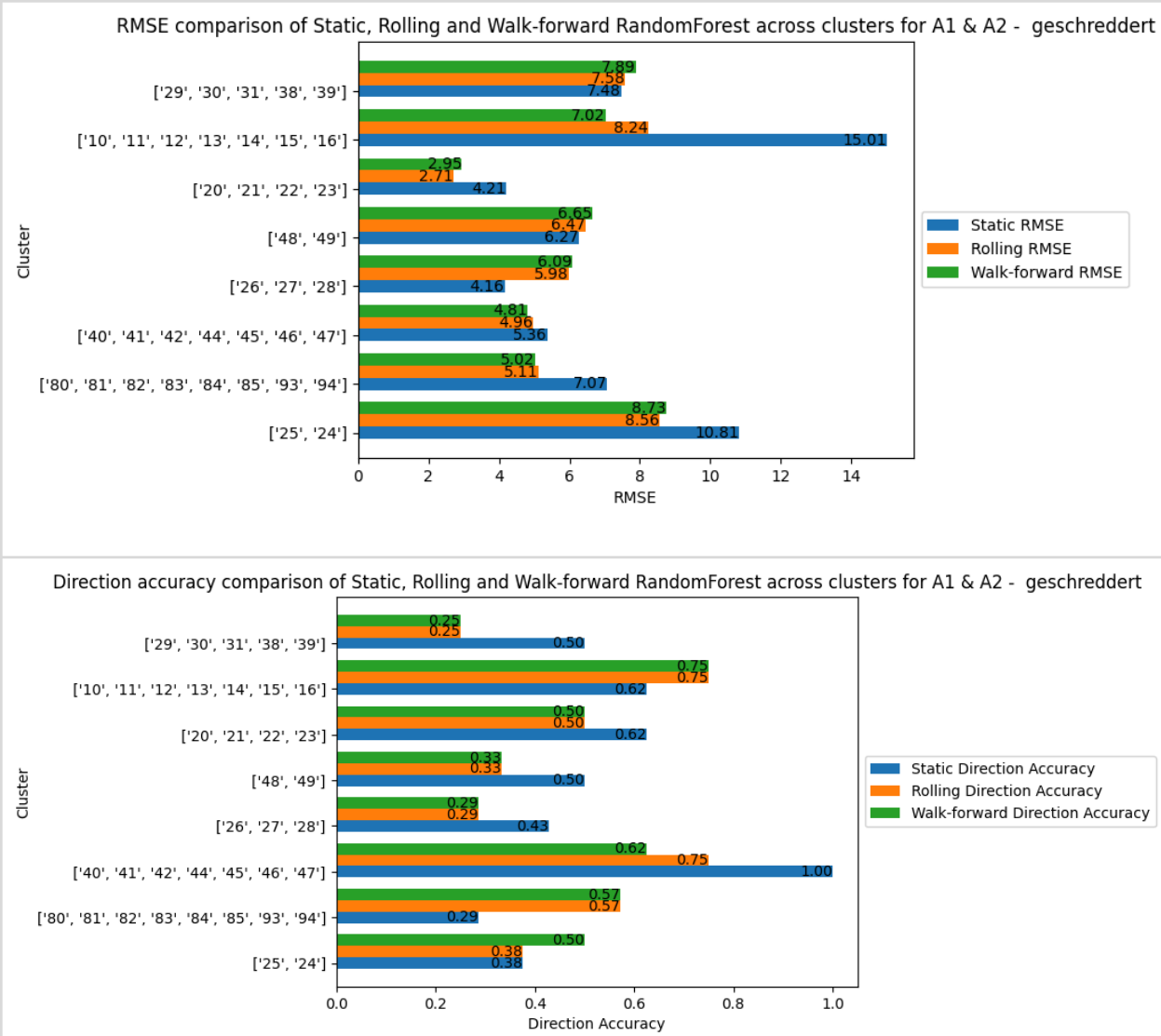


Fig. 16. Comparison of RMSE and Direction Accuracy between Static, Rolling and Walk-forward methods using Random Forest across all clusters for category A1 & A2 - geschreddert.

Again, there does not seem to be one single method that produces the lowest RMSE in general. The lowest RMSE of 2.71 is obtained for the rolling method of forecasting in the cluster "['20', '21', '22', '23']". Direction accuracy tends to be higher with static forecast prediction, except in some clusters, with the highest being for the cluster "['40', '41', '42', '44', '45', '46', '47']" where the forecasts where 100% in the right direction.

XGBoost Static Forecast

Using the same ForecasterAutoreg class from the SKforecast package is used with the regressor being set as XGBRegressor and with a lag of 4, forecasts are made for the number of steps in the test data set.

When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

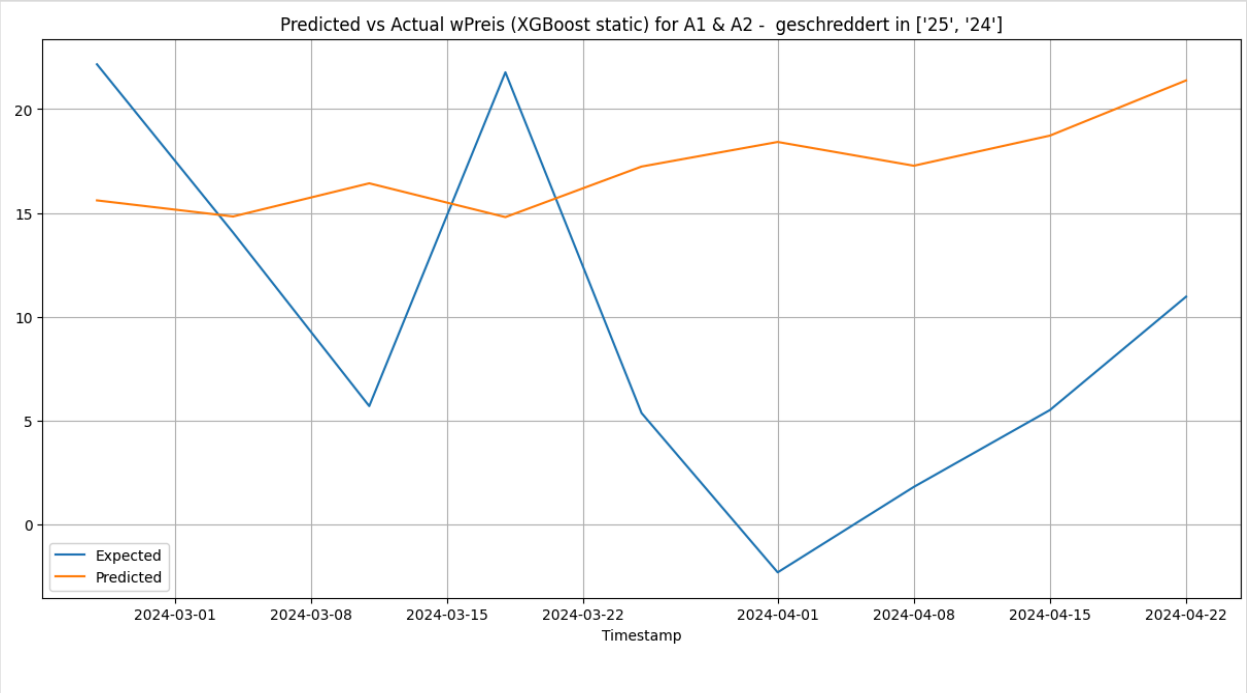


Fig. 17. Predicted vs. Actual wPreis values using XGBoost(static) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 14. XGBoost static results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (XGBoost static) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 10.741
ME: 7.736
RMSE: 12.012
Direction Accuracy: 0.375
```

The RMSE value of 12.012 is comparable to the scale of the wPreis values and the direction accuracy of 37.5% suggests that better models or methods are desired to be able to predict the direction with more confidence.

XGBoost Rolling Window

As in the case for Decision Tree and Random Forest, in the rolling window approach for XGBoost, the training set will consist of a feature variable X which is a list of sequences of 4 values and the target variable y, which is the next value after the sequence. The model is trained on this X and y once and a loop runs the length of the original dataset, in each step of which the predict function is called on the feature variable testX containing the preceding 4 values as sequence to get a corresponding predicted y value. This is repeated until predicted y values are obtained to compare with each of the steps in the original test dataset.

When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

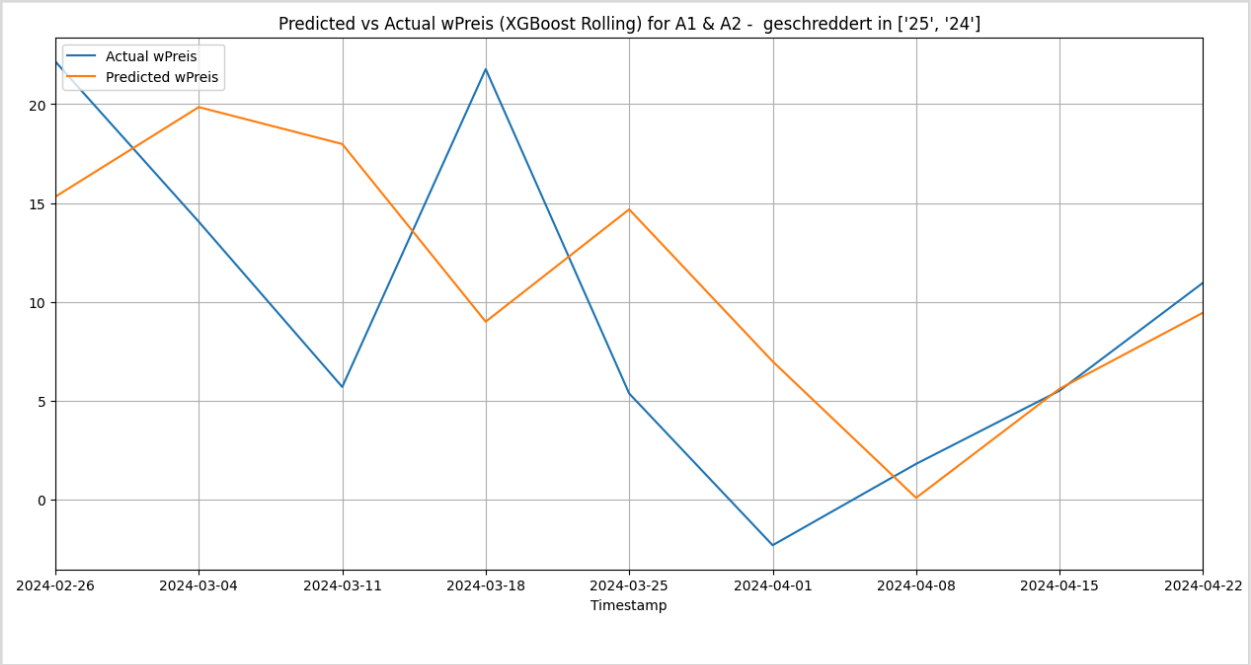


Fig. 18. Predicted vs. Actual wPreis values using XGBoost(rolling) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 15. XGBoost rolling results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (XGBoost Rolling) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 6.623
ME: 1.549
RMSE: 7.975
Direction Accuracy: 0.500
```

The RMSE value of 7.975 is an improvement over that from the static method and so is the direction accuracy of 50% even though more accurate direction forecasts are needed.

XGBoost Walk-forward Validation

Similar to the same procedure for the Decision Tree and RandomForest, in the walk-forward validation approach, the training of the model on the feature variable X and target variable y happens at every step of the loop, with both being updated with the new observation from the original test data set. This will still include the lag of 4. The predict function is also called at each step of the loop on the updated feature variable testX.

When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

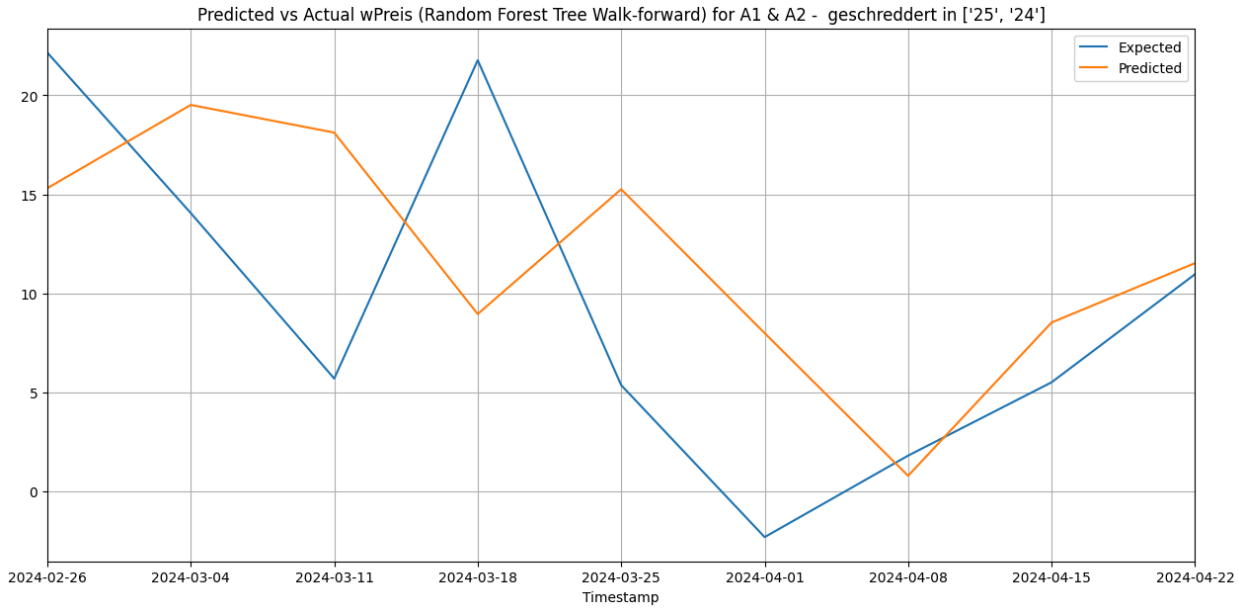


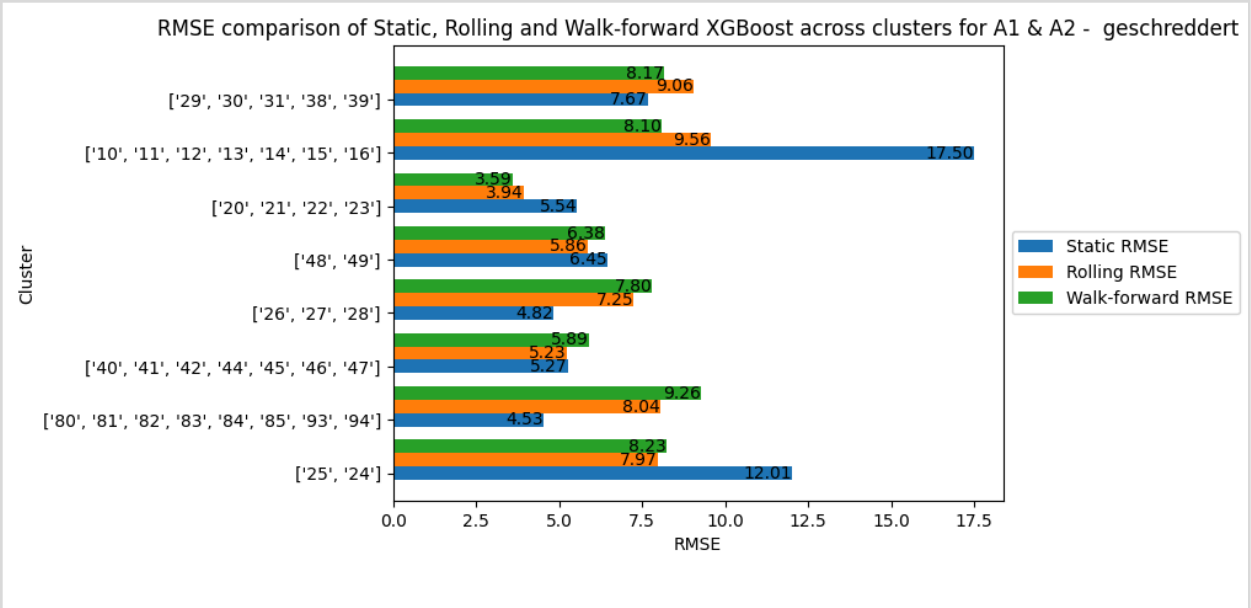
Fig. 19. Predicted vs. Actual wPreis values using XGBoost(walk-forward) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 16. XGBoost walk-forward results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (XGBoost Walk-forward) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 6.923
ME: 2.328
RMSE: 8.226
Direction Accuracy: 0.500
```

The RMSE value of 8.226 is an improvement over the static method but still not as good as that seen with the rolling window method. The Direction Accuracy pf 50% is on par with that from the rolling window method.

When the three methods of forecasting for XGBoost are compared across clusters the following results are obtained .



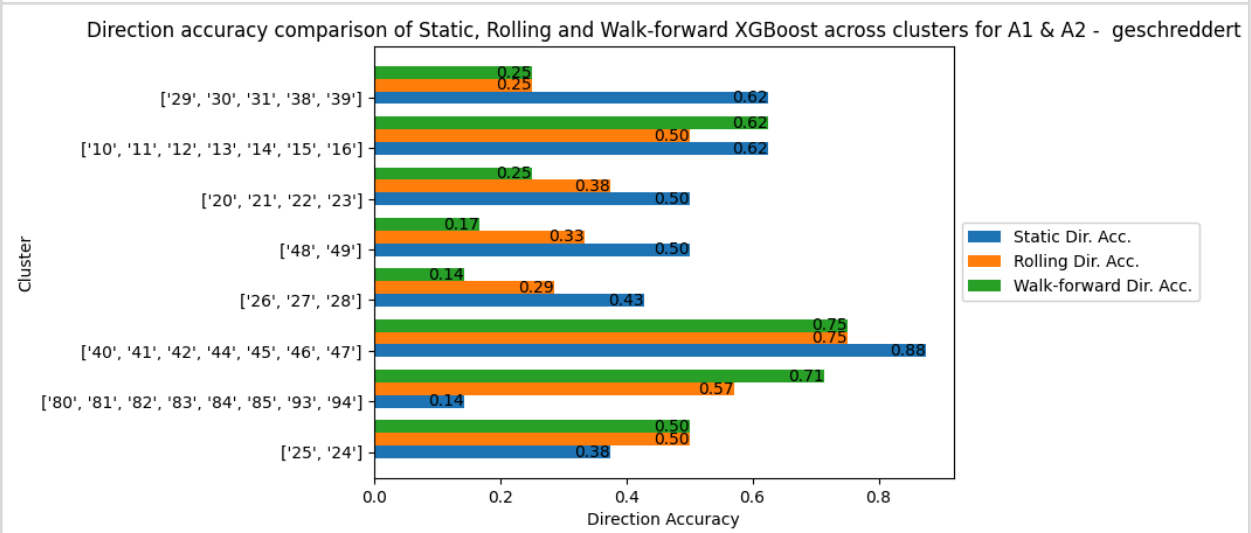


Fig. 20. Comparison of RMSE and Direction Accuracy between Static, Rolling and Walk-forward methods using XGBoost across all clusters for category A1 & A2 - geschreddert.

Once again, it can be seen that there is no single method that performs better than the others for all the clusters. The lowest RMSE of 3.59 is obtained for the walk-forward validation method of forecasting in the cluster "['20', '21', '22', '23']". Direction accuracy tends to be the highest with static forecasting with some exceptions, and the highest value being 88% for the cluster "['40', '41', '42', '44', '45', '46', '47']".

Prophet Static Forecast

Using Prophet's recommended approach of creating a new dataframe using the "make_future_dataframe" method with the number of periods specified as the length of the test data set, and setting frequency to be weekly, with a start of Monday, forecasts were made with the following results.

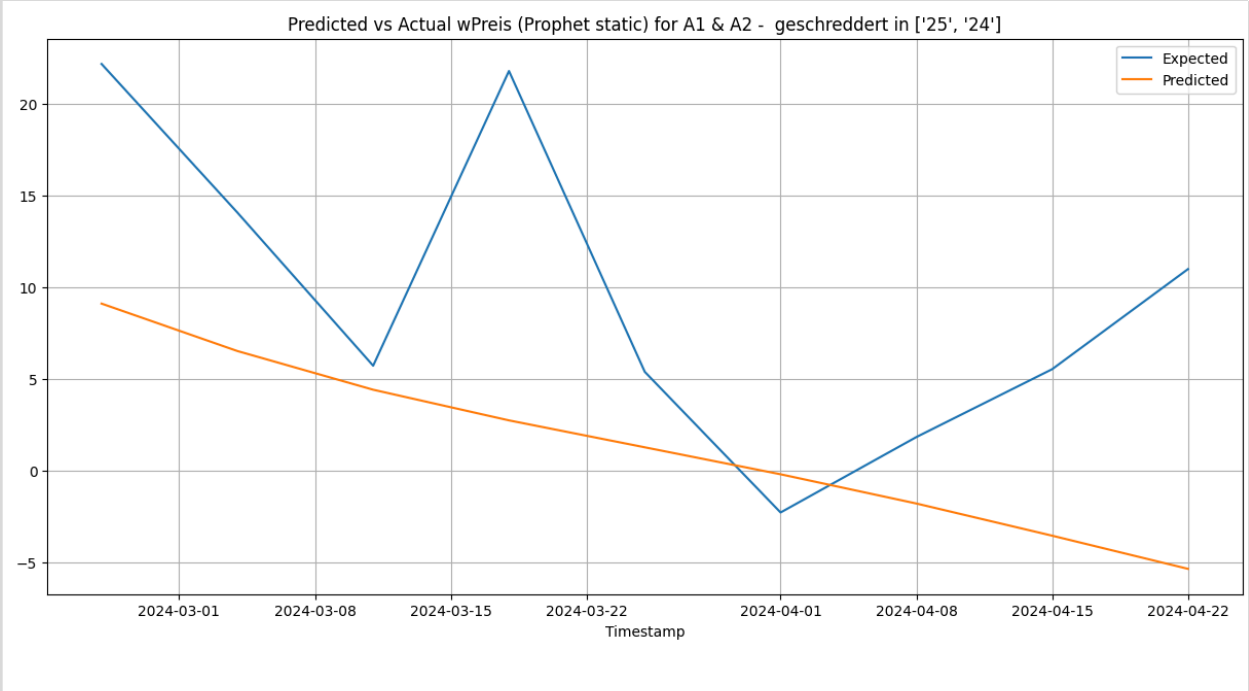


Fig. 21. Predicted vs. Actual wPreis values using Prophet(static) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 17. Prophet static results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Prophet static) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 8.468
ME: -8.006
RMSE: 10.414
Direction Accuracy: 0.500
```

Prophet Walk-forward Validation

Similar to the same procedure for the Decision Tree and RandomForest, in the walk-forward validation approach, the training of the model on the feature variable X and target variable y happens at every step of the loop, with both being updated with the new observation

from the original test data set. This will still include the lag of 4. The predict function is also called at each step of the loop on the updated feature variable testX.

When this is done for the cluster ['25', '24'] and for the category 'A1 & A2 - geschreddert' within it gives the following comparison plot and metrics.

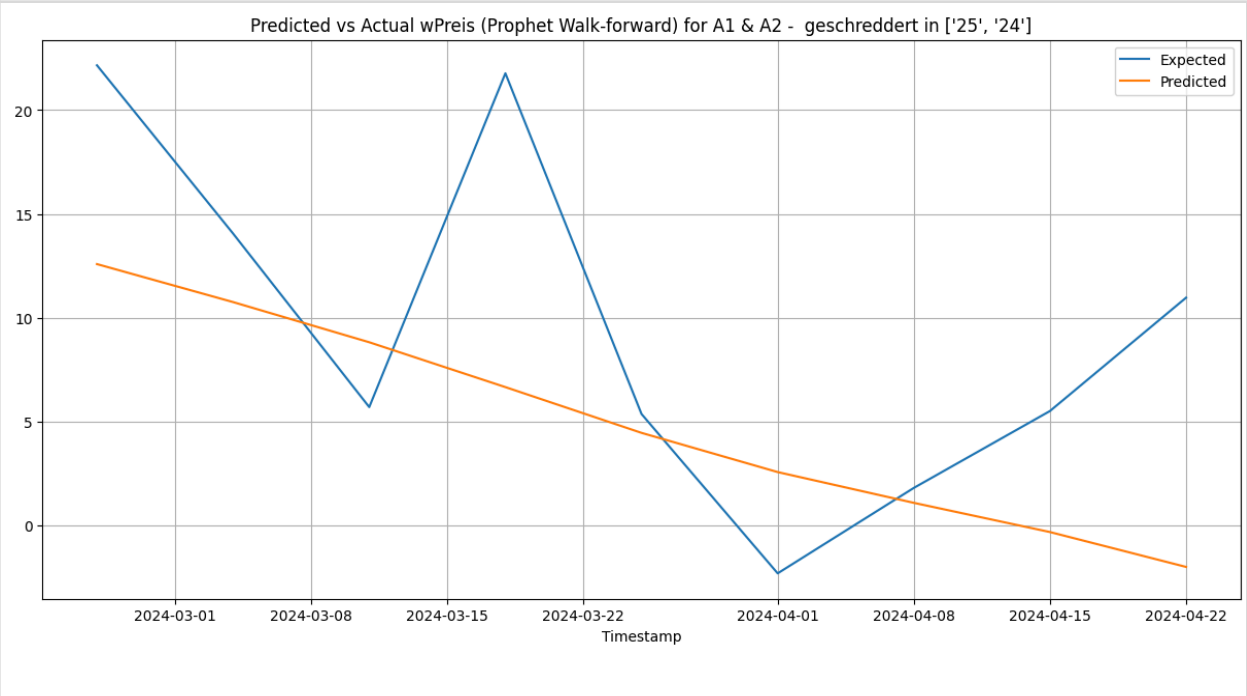


Fig. 22. Predicted vs. Actual wPreis values using Prophet(static) for A1 & A2 - geschreddert in cluster ['25', '24']

Table 18. Prophet walk-forward results for category 'A1 & A2 - geschreddert' in cluster ['25', '24']

```
Forecast Accuracy (Prophet Walk-forward) for A1 & A2 -  geschreddert in ['25', '24']
MAE: 6.264
ME: -4.487
RMSE: 7.939
Direction Accuracy: 0.500
```

When the two methods of forecasting for Prophet are compared across clusters we get the following result.

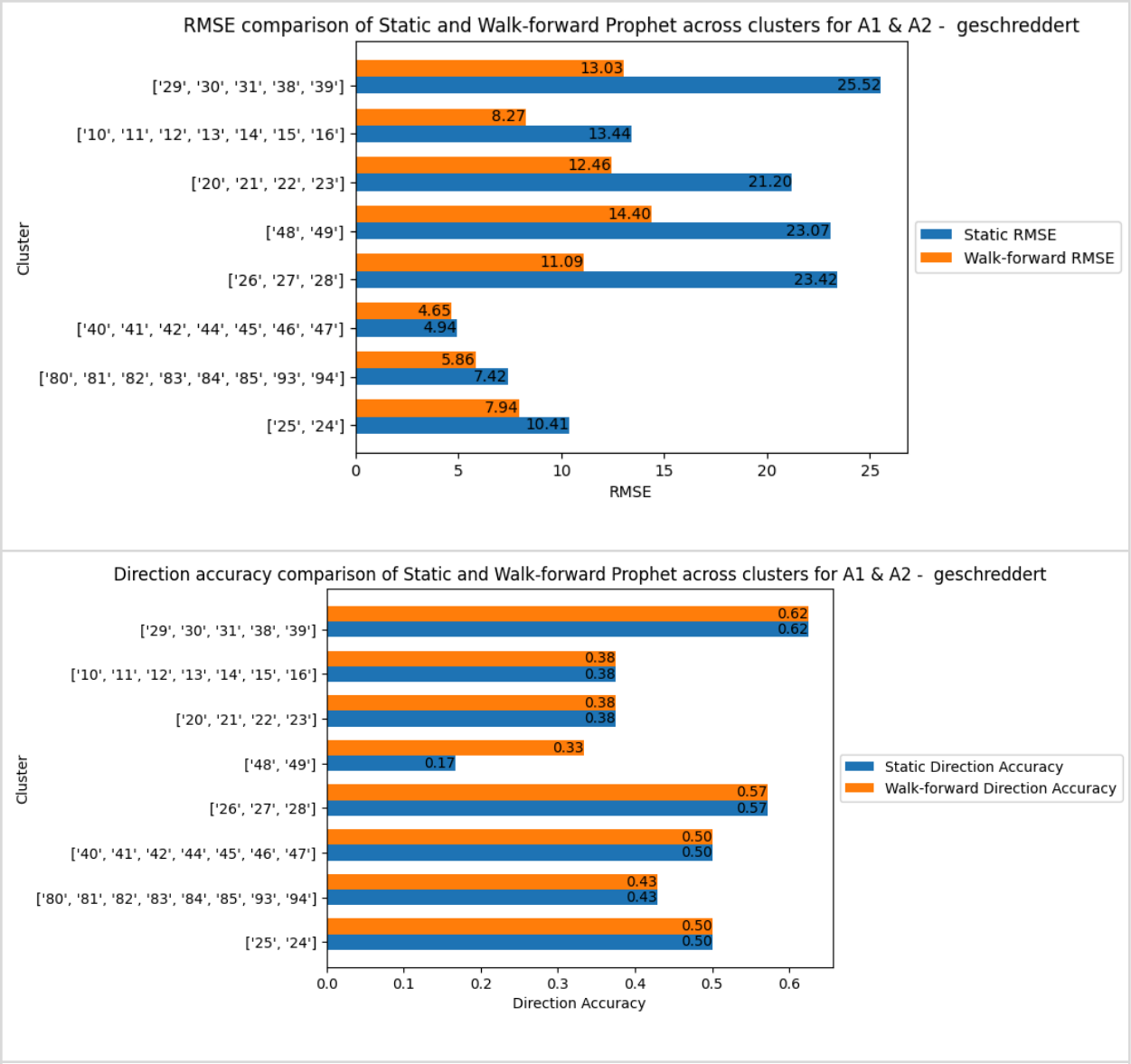


Fig. 23. Comparison of RMSE and Direction Accuracy between Static, Rolling and Walk-forward methods using Prophet across all clusters for category A1 & A2 - geschreddert.

In the case of Prophet, it can be seen that the walk-forward validation produces the lower RMSE for all clusters with the lowest being 4.65 for the cluster ['40', '41', '42', '44', '45', '46', '47']. This is also observed for all other categories, suggesting that when using Prophet, adopting the walk-forward validation method can be the better option. Direction accuracy also follows a similar pattern, with the walk-forward validation being either equal to the static method in how

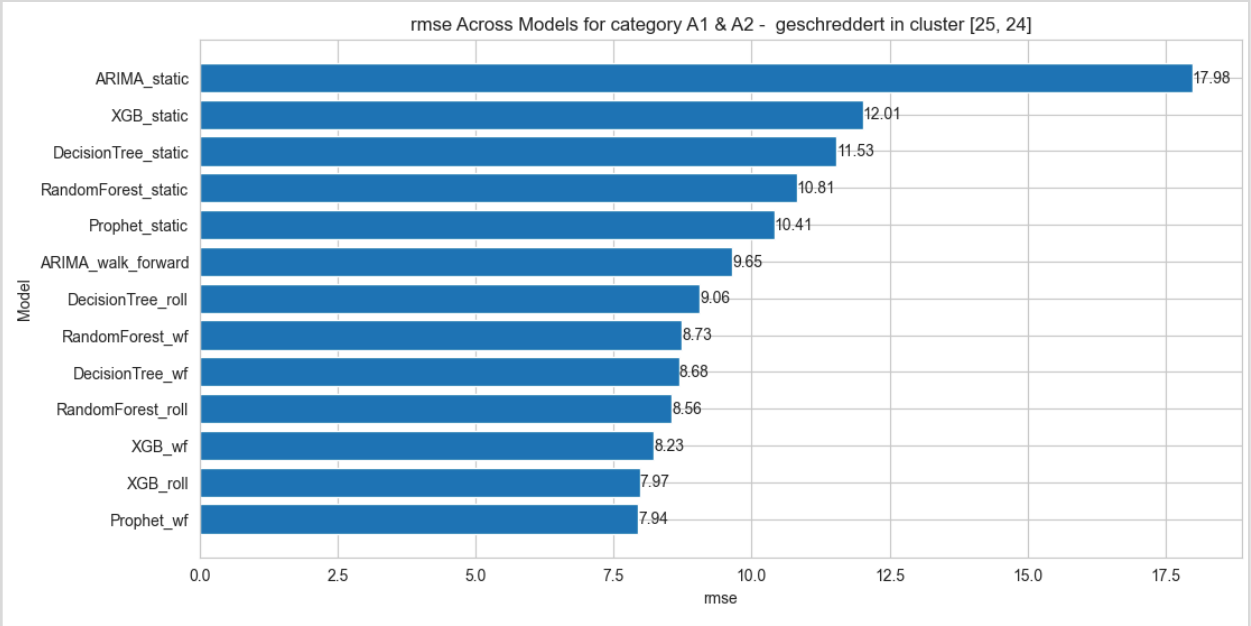
accurate it is to predict the direction, with the highest being 62% for the cluster "['29', '30', '31', '38', '39']".

Discussion

The results of forecasting using each of the available methods for ARIMA, Decision Tree, Random Forest, XGBoost and Prophet models illustrate the following points.

Static forecasting methods usually perform worse than others

Comparing the different models and methods of forecasting within each for the predictions for the specific category of A1 & A2 - geschreddert for the cluster ['25', '24'], it is observed that the Prophet model with the walk-forward validation forecasting has the lowest RMSE among all (7.94) followed closely by the XGBoost model with the rolling window forecasting (7.97) and the XGBoost model with the walk-forward validation method (8.23). In all models, the static forecasting approach had higher RMSE, suggesting that either a rolling window or walk-forward validation method is likely to be preferable in general if the objective is to forecast values of wPreis as closely as possible to the real values that will be observed.



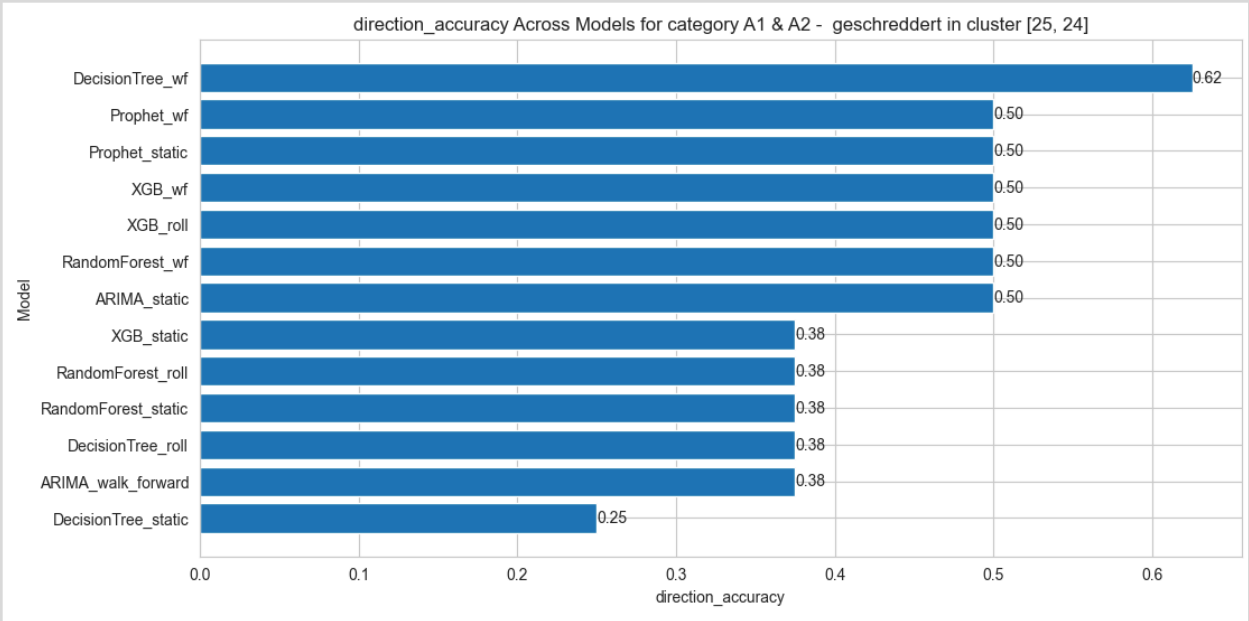


Fig. 24. Comparison of RMSE and Direction Accuracy for each of the methods under each of the models for the category A1 & A2 - geschreddert in cluster [25,24]

Direction accuracy, which can often be a more useful metric for stakeholders interested in planning ahead for price increases or decreases, is the highest for Decision Tree model when forecasting is done with the walk-forward validation 62% and all others are accurate at most 50% of the time.

Different time series may require different methods and models

As seen from the evaluation of the different methods for each of the models ARIMA, Decision Trees, RandomForest, XGBoost and Prophet and the various methods of forecasting using each, there is no clear winner that applies for all the categories and clusters. The complete comparison of RMSE values for each of the model-method combination for each of the time series(which can be viewed in the Appendix) shows this clearly.

In fact, counting the number of times each model-method combination produced the lowest RMSE in all of the 33 time series, the following counts can be obtained.

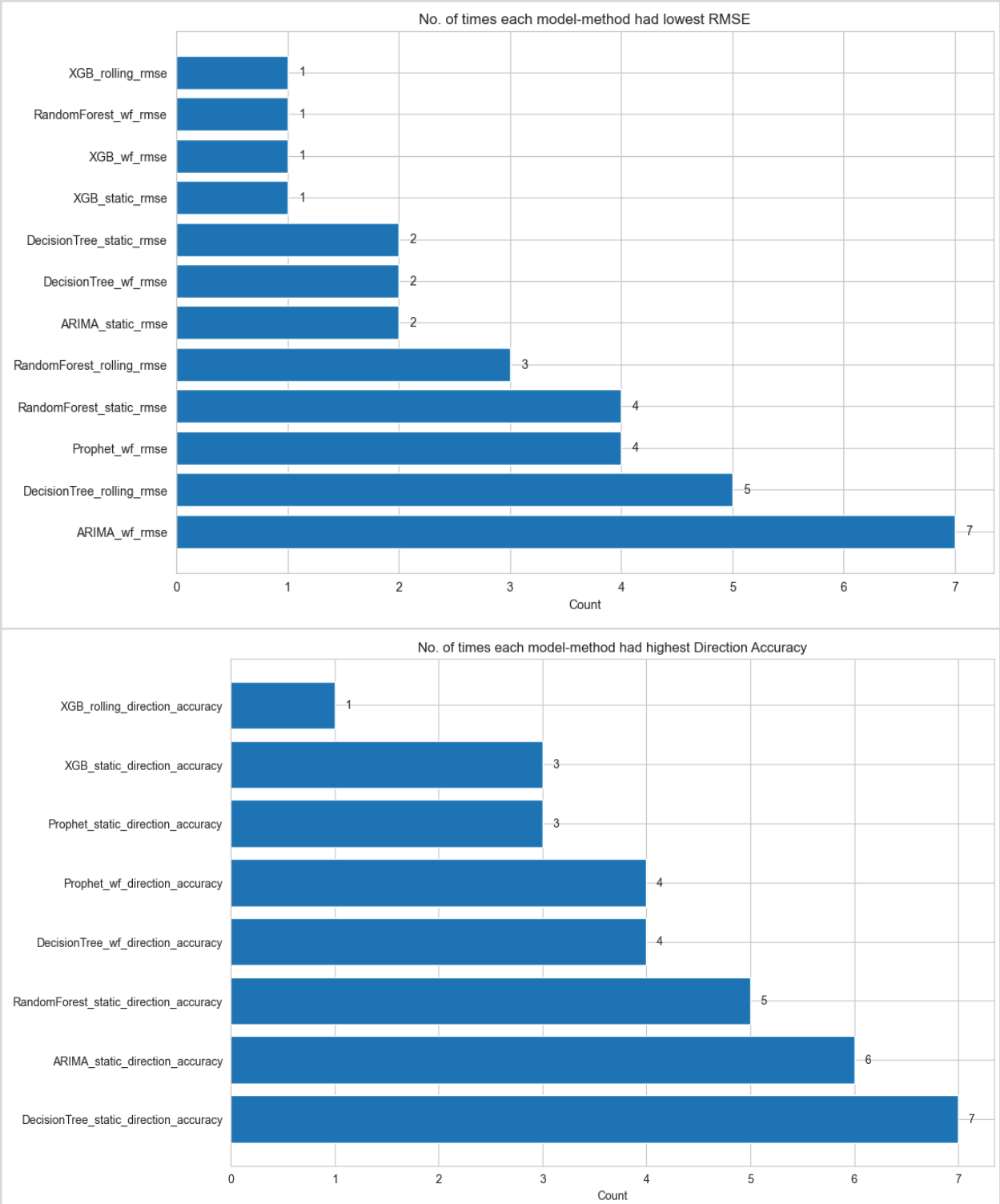


Fig. 25. Count of the number of times each model-method combination had the lowest RMSE and highest Direction Accuracy across all 33 time series.

As seen from the above plot, the walk-forward validation method using ARIMA model produced the lowest RMSEs most times (7) out of all the 33 time series, followed by the rolling forecast method using Decision Tree (5) and the walk-forward method using Prophet (4) times. Similarly, in the case of the highest direction accuracy among all model-method combination, the static forecasting using Decision Tree was the best in 7 out of the 33 time series, followed by the static forecasting using ARIMA (6) and Random Forest (5). Thus there is no single best model or indeed, the specific forecasting method within each model that can be applied universally across all time series and different categories and clusters may require specific choices of models and methods for the best predictive capabilities.

Lastly, even in the use of a model, different parameters may yield different results for different categories and clusters. For example, in the case of ARIMA, the p-value used was 4 as it suggested the lowest AIC values for one particular series. Other series may perform better when different p-values are used. This means that the model training and fitting was based on the assumption that the 4 lagged values immediately preceding a time step are important predictors in the forecasting of the next step. However, for other categories and clusters, there may be more previously lagged values that influence the forecast at every step, or there may be fewer lagged time steps that have predictive value, even just the immediate previous time step. Thus to explore whether accuracy improves, such parameters may need to be customized for each time series to find the predictions that yield the best metrics.

Conclusion

Predicting prices of wood waste is a challenging task due to the non-transparent nature of the market. The lack of readily accessible information and the possibility of hidden factors such as price of other commodities like oil or gas, electricity consumption and business cycles influencing the price of wood waste compounds the problem. Using historical data on the prices of wood waste is the option pursued in this analysis, with the data being provided by The Bruning Group for weekly observations from September 2020 until April 2024.

The analysis presented in this thesis explored various models for forecasting including ARIMA, Decision Trees, RandomForest, XGBoost, and Prophet and three key methods for predictions were employed wherever applicable, namely static (multi-step), rolling window and walk-forward validation. The choice of selecting these models were influenced by the specific benefits they each offer. ARIMA model was considered due to the popularity of its use in time series forecasting as well as its effectiveness in capturing linear dependencies and seasonality in the data. Decision Trees, RandomForest, and XGBoost were considered due to their flexibility, especially in handling non-stationary data. These models showed the ability to capture complex relationships and interactions between variables without requiring explicit stationarity conversion. Prophet was also explored due to its specialization as a time series forecasting tool and its ability to capture seasonality and holiday effects inherent in the data.

The findings suggest that different models and methods yield different performance metrics and that no single approach can be adopted as a general-purpose solution for prediction. Depending on the specific category of wood waste and the geographical region in question, the appropriate forecasting model to be adopted will be the one that provides the best performance metrics. For each individual time series, different parameters, such as the p-values for ARIMA,

or the lags and input time steps for the other models may be unique as there may be more or less lagged values that may be important in the forecasting.

Further work that could be explored to build on this analysis includes going beyond univariate time series. Multivariate time series analysis that incorporates additional variables such as economic indicators, demographic data, environmental factors and other financial markets may allow for a more comprehensive understanding of the underlying dynamics of the wood waste market and potentially improve prediction accuracy. Additionally, other forecasting models such as Long Short-Term Memory (LSTM) neural networks can also be explored and compared with the performance metrics found in this analysis.

This thesis contributes to the understanding of time series forecasting methodologies and their applications in price prediction for non-transparent markets. By evaluating the performance of various models across different datasets and forecasting horizons, valuable insights were gained into the strengths and limitations of each approach. Moving forward, continued research in this area will be helpful to address emerging challenges and enhance the predictive capabilities of time series forecasting models.

Code repository and notebooks

The analysis for this thesis was conducted using Python, with the use of all the essential packages for data cleaning, transformation, data visualization, machine learning and time series analysis and forecasting. The notebooks detailing the code can be found in the public GitHub repository at https://github.com/surajkarak/thesis .

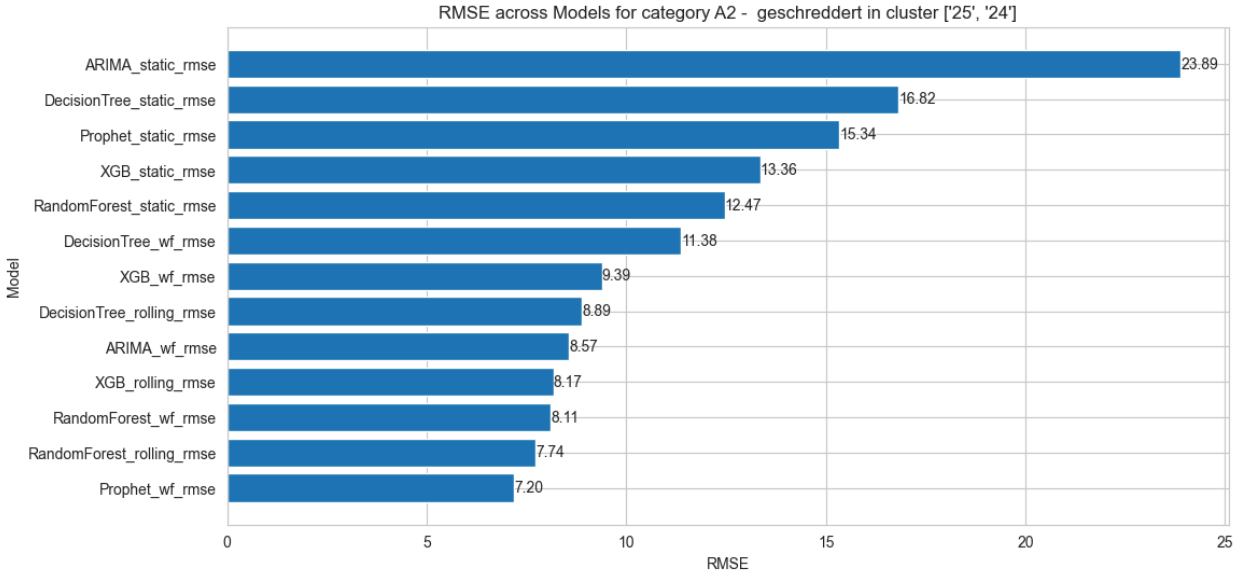
References

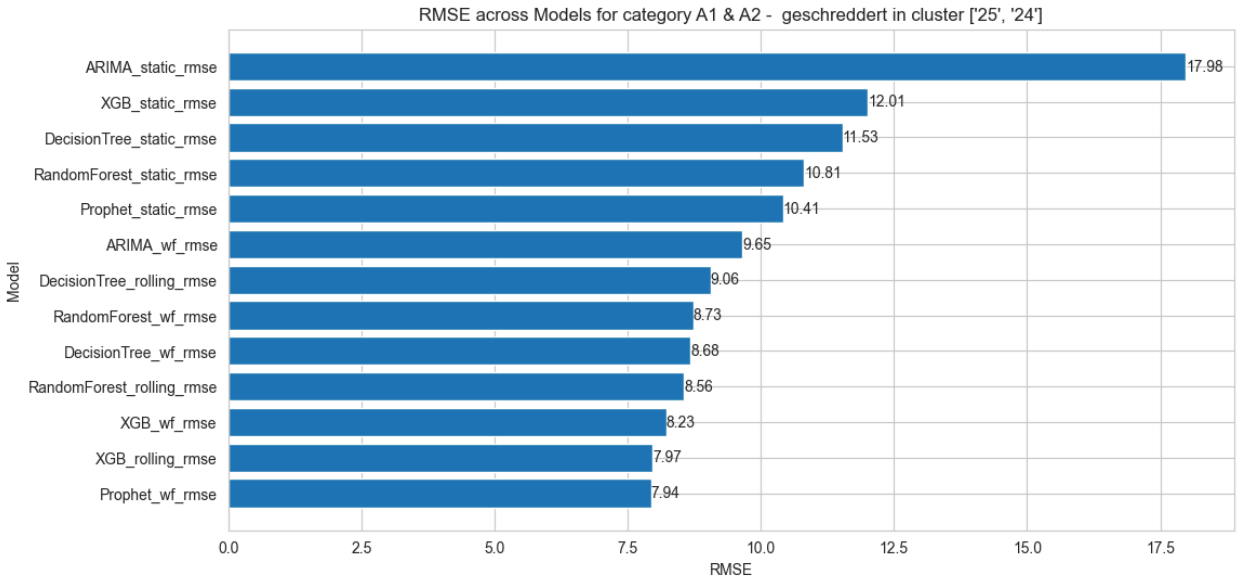
1. Abbasi, Maryam, and Ali El Hanandeh. 2016. "Forecasting Municipal Solid Waste Generation Using Artificial Intelligence Modelling Approaches." *Waste Management* 56 (October): 13–22.
2. Azadi, Sama, and Ayoub Karimi-Jashni. 2016. "Verifying the Performance of Artificial Neural Network and Multiple Linear Regression in Predicting the Mean Seasonal Municipal Solid Waste Generation Rate: A Case Study of Fars Province, Iran." *Waste Management* 48 (February): 14–23.
3. Bell, John E., Diane A. Mollenkopf, and Hannah J. Stolze. 2013. "Natural Resource Scarcity and the Closed‐loop Supply Chain: A Resource‐advantage View." *International Journal of Physical Distribution & Logistics Management* 43 (5/6): 351–79.
4. Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.
5. Chu, Ke-Young. 1978. "Short-Run Forecasting of Commodity Prices: An Application of Autoregressive Moving Average Models (Prévisions à Court Terme Des Prix Des Produits de Base: Application de Modèles Autorégressifs de Moyennes Mobiles) (El Pronóstico a Corto Plazo de Los Precios de Productos Básicos: Aplicación de Modelos de Media Móvil Autorregresiva)." *Staff Papers - International Monetary Fund. International Monetary Fund* 25 (1): 90–111.
6. Daian, G., and B. Ozarska. 2009. "Wood Waste Management Practices and Strategies to Increase Sustainability Standards in the Australian Wooden Furniture Manufacturing Sector." *Journal of Cleaner Production* 17 (17): 1594–1602.
7. Daskalopoulos, E., O. Badr, and S. D. Probert. 1998. "Municipal Solid Waste: A Prediction Methodology for the Generation Rate and Composition in the European Union Countries and the United States of America." *Resources, Conservation and Recycling* 24 (2): 155–66.
8. "Forecasting Prices of Manufactured Pinus Spp. Using ARIMA Models." n.d. https://www.scielo.org.mx/scielo.php?pid=S1405-04712014000100004&script=sci_abstract&tlng=en.
9. "Forecasting: Principles and Practice (3rd Ed)." n.d. Accessed May 19, 2024. https://otexts.com/fpp3/.
10. Górna, Aleksandra, Alicja Szabelska-Beręsewicz, Marek Wieruszewski, Monika Starosta-Grala, Zygmunt Stanula, Anna Kożuch, and Krzysztof Adamowicz. 2023. "Predicting Post-Production Biomass Prices." *Energies* 16 (8): 3470.
11. Hlavackova, Petra, David Brezina, and Andrea Sujova. 2015. "The Price Formation of Raw Wood in the Czech Republic and a Comparison with the Neighbor States." *Procedia*

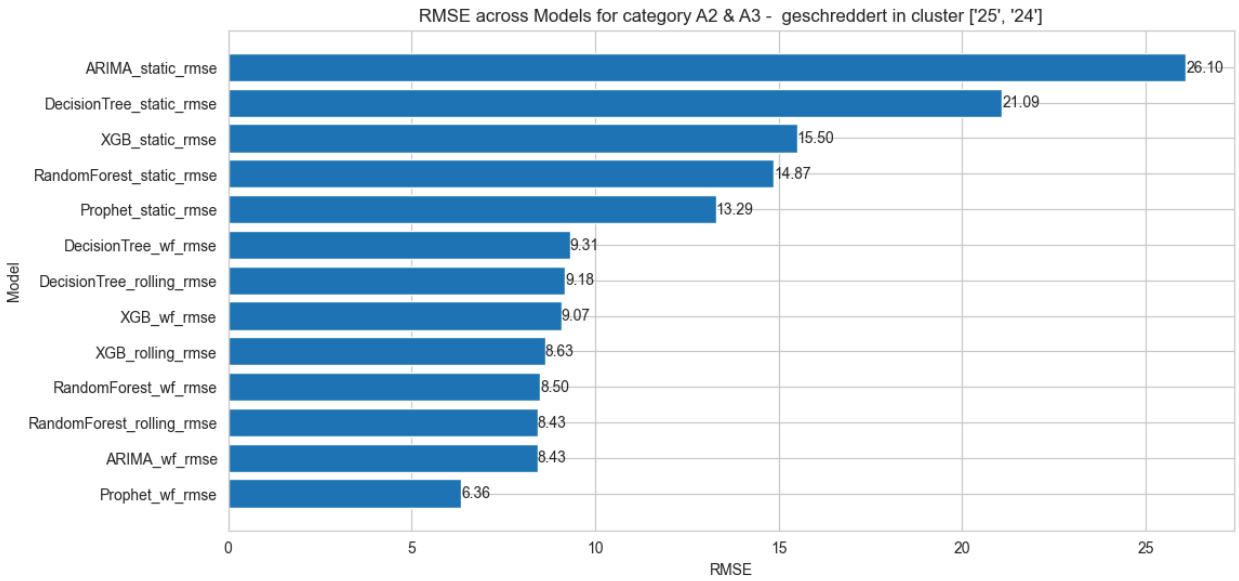
Economics and Finance 26 (January): 389–95.
12. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. n.d. *An Introduction to Statistical Learning*. Springer US. Accessed May 19, 2024.
13. Johnson, Nicholas E., Olga Ianiuk, Daniel Cazap, Linglan Liu, Daniel Starobin, Gregory Dobler, and Masoud Ghandehari. 2017. "Patterns of Waste Generation: A Gradient Boosting Model for Short-Term Waste Prediction in New York City." *Waste Management* 62 (April): 3–11.
14. Kannangara, Miyuru, Rahul Dua, Leila Ahmadi, and Farid Bensebaa. 2018. "Modeling and Prediction of Regional Municipal Solid Waste Generation and Diversion in Canada Using Machine Learning Approaches." *Waste Management* 74 (April): 3–15.
15. Kożuch, Anna, Dominika Cywicka, and Krzysztof Adamowicz. 2023. "A Comparison of Artificial Neural Network and Time Series Models for Timber Price Forecasting." *Forests, Trees and Livelihoods* 14 (2): 177.
16. Kumar, Atul, and S. R. Samadder. 2017. "An Empirical Model for Prediction of Household Solid Waste Generation Rate – A Case Study of Dhanbad, India." *Waste Management* 68 (October): 3–15.
17. Kumar, Atul, S. R. Samadder, Nitin Kumar, and Chandrakant Singh. 2018. "Estimation of the Generation Rate of Different Types of Plastic Wastes and Possible Revenue Recovery from Informal Recycling." *Waste Management* 79 (September): 781–90.
18. Leskinen, Pekka, and Jyrki Kangas. 1998. "Modelling and Simulation of Timber Prices for Forest Planning Calculations." *Scandinavian Journal of Forest Research / Issued Bimonthly by the Nordic Forest Research Cooperation Committee* 13 (1-4): 469–76.
19. Malaty, Ramses, Anne Toppinen, and Jari Viitanen. 2007. "Modelling and Forecasting Finnish Pine Sawlog Stumpage Prices Using Alternative Time-Series Methods." *Canadian Journal of Forest Research. Journal Canadien de La Recherche Forestiere* 37 (1): 178–87.
20. Michinaka, Tetsuya, Hirofumi Kuboyama, Kazuya Tamura, Hiroyasu Oka, and Nobuyuki Yamamoto. 2016. "Forecasting Monthly Prices of Japanese Logs." *Forests, Trees and Livelihoods* 7 (5): 94.
21. Nguyen, X. Cuong, T. Thanh Huyen Nguyen, D. Duong La, Gopalakrishnan Kumar, Eldon R. Rene, D. Duc Nguyen, S. Woong Chang, W. Jin Chung, X. Hoan Nguyen, and V. Khanh Nguyen. 2021. "Development of Machine Learning - Based Models to Forecast Solid Waste Generation in Residential Areas: A Case Study from Vietnam." *Resources, Conservation and Recycling* 167 (April): 105381.

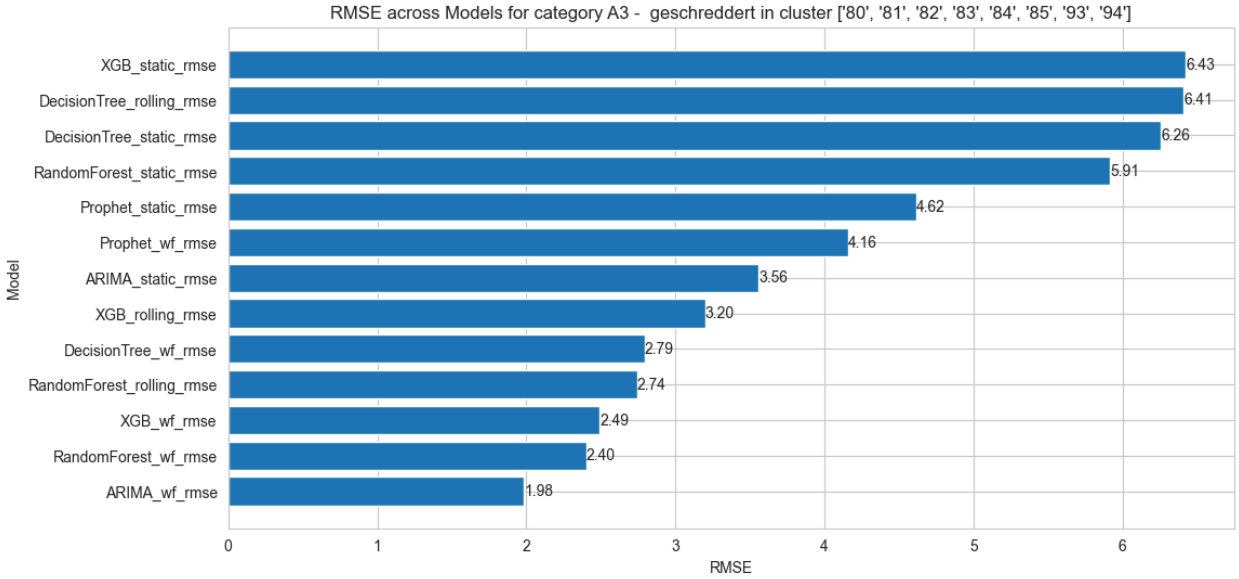
Appendix

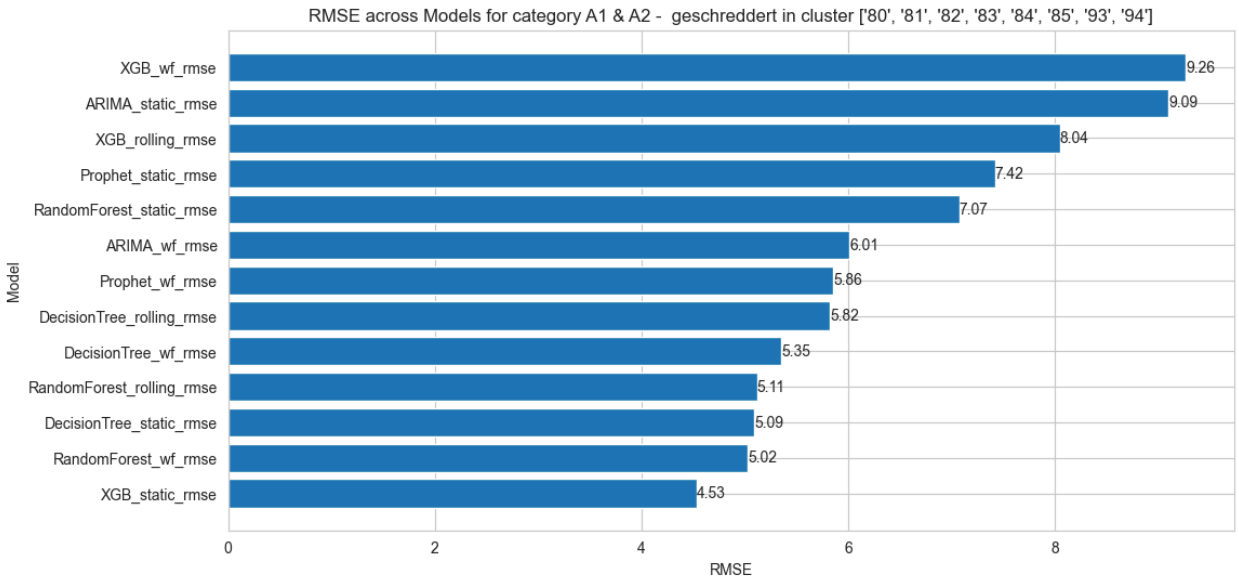
The following are comparison of RMSE values for each of the model-method combination for each of the 33 time series.

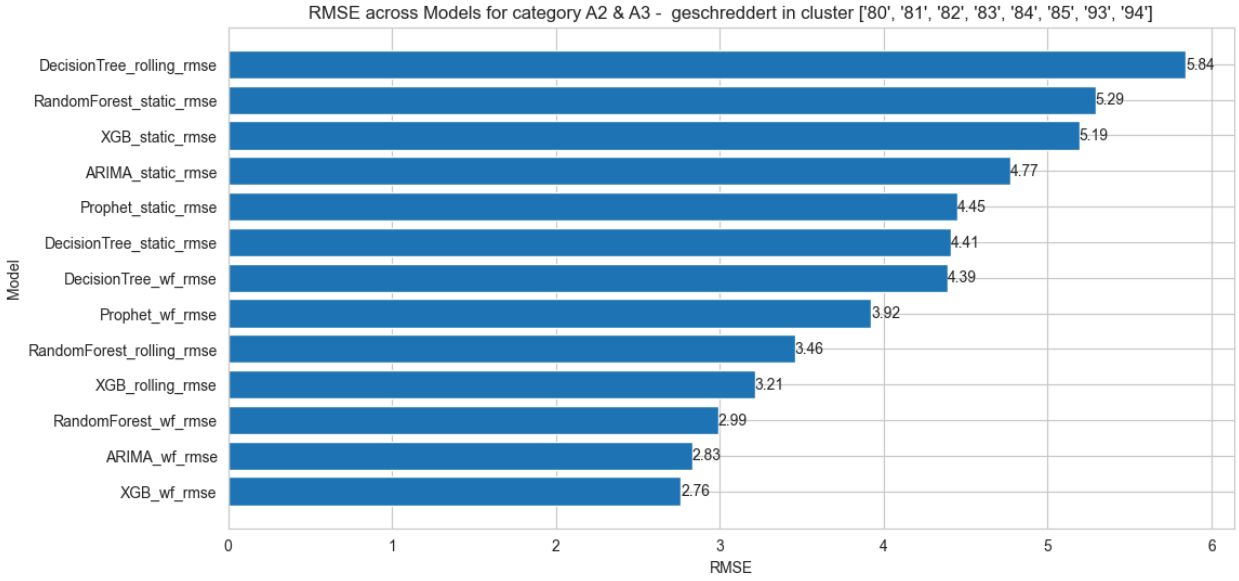


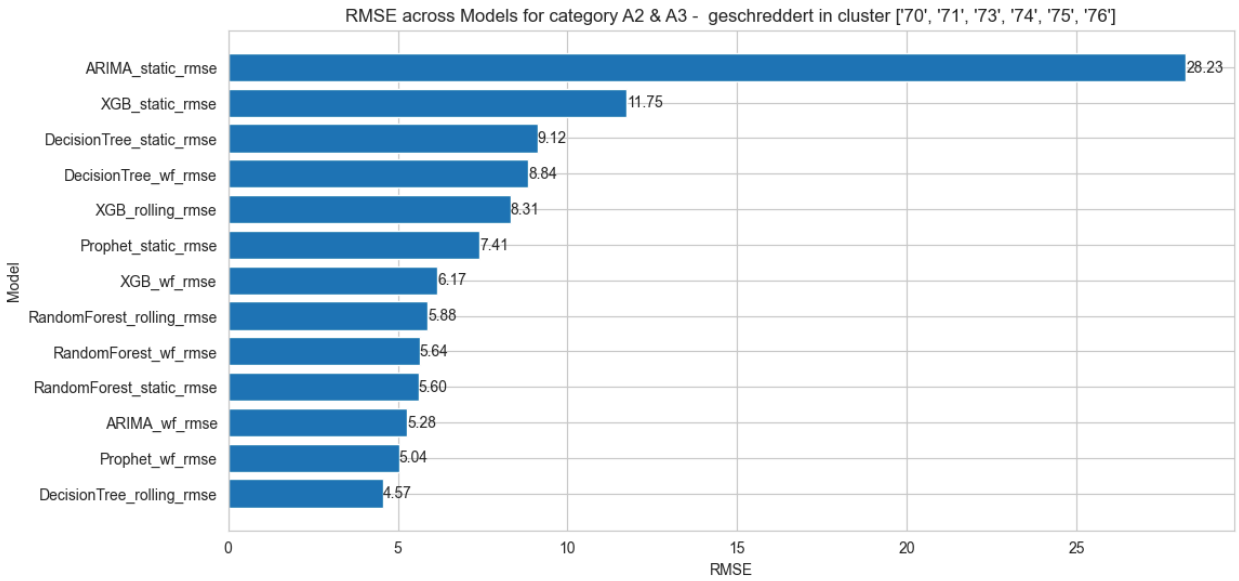


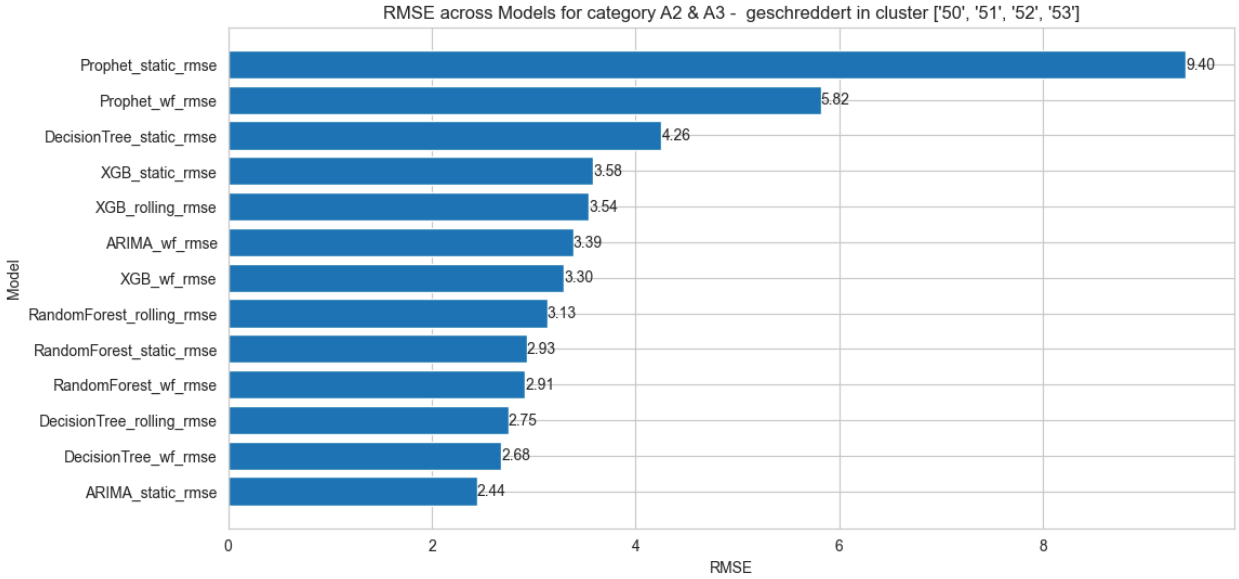

RMSE across Models for category A2 & A3 - geschreddert in cluster ['25', '24']
ARIMA_static_rmse
DecisionTree_static_rmse
XGB_static_rmse
RandomForest_static_rmse
Prophet_static_rmse
DecisionTree_wf_rmse
DecisionTree_rolling_rmse
XGB_wf_rmse
XGB_rolling_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
ARIMA_wf_rmse
Prophet_wf_rmse
26.10
21.09
15.50
14.87
13.29
9.31
9.18
9.07
8.63
8.50
8.43
8.43
6.36
Model
0
5
10
15
20
25
RMSE

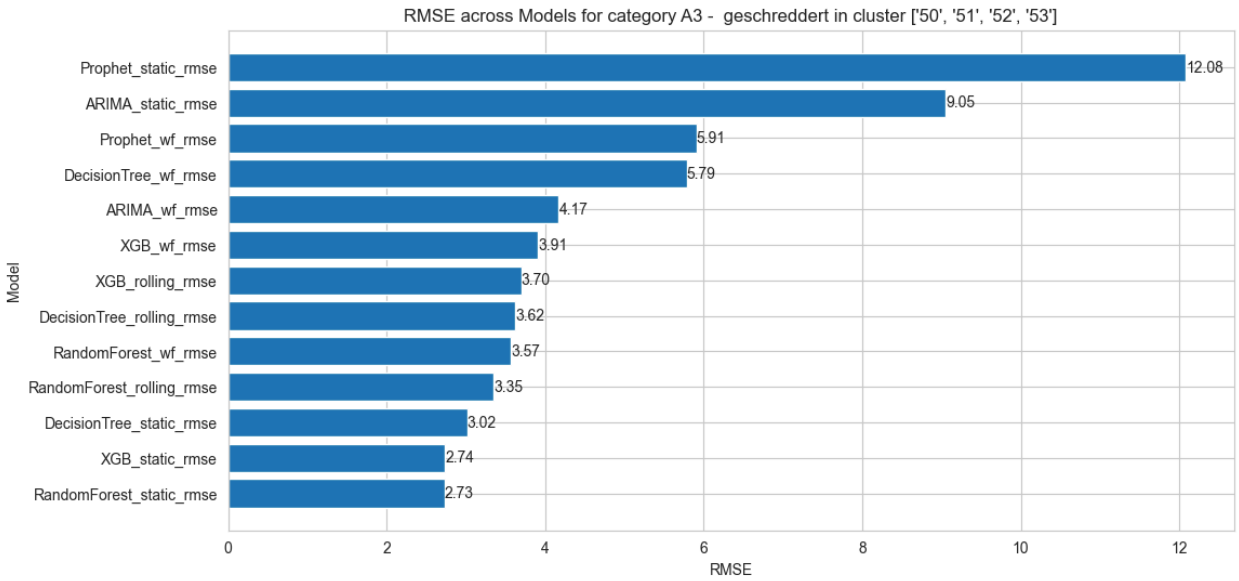

RMSE across Models for category A3 - geschreddert in cluster ['80', '81', '82', '83', '84', '85', '93', '94']
XGB_static_rmse
DecisionTree_rolling_rmse
DecisionTree_static_rmse
RandomForest_static_rmse
Prophet_static_rmse
Prophet_wf_rmse
ARIMA_static_rmse
XGB_rolling_rmse
DecisionTree_wf_rmse
RandomForest_rolling_rmse
XGB_wf_rmse
RandomForest_wf_rmse
ARIMA_wf_rmse
6.43
6.41
6.26
5.91
4.62
4.16
3.56
3.20
2.79
2.74
2.49
2.40
1.98
Model
0
1
2
3
4
5
6
RMSE

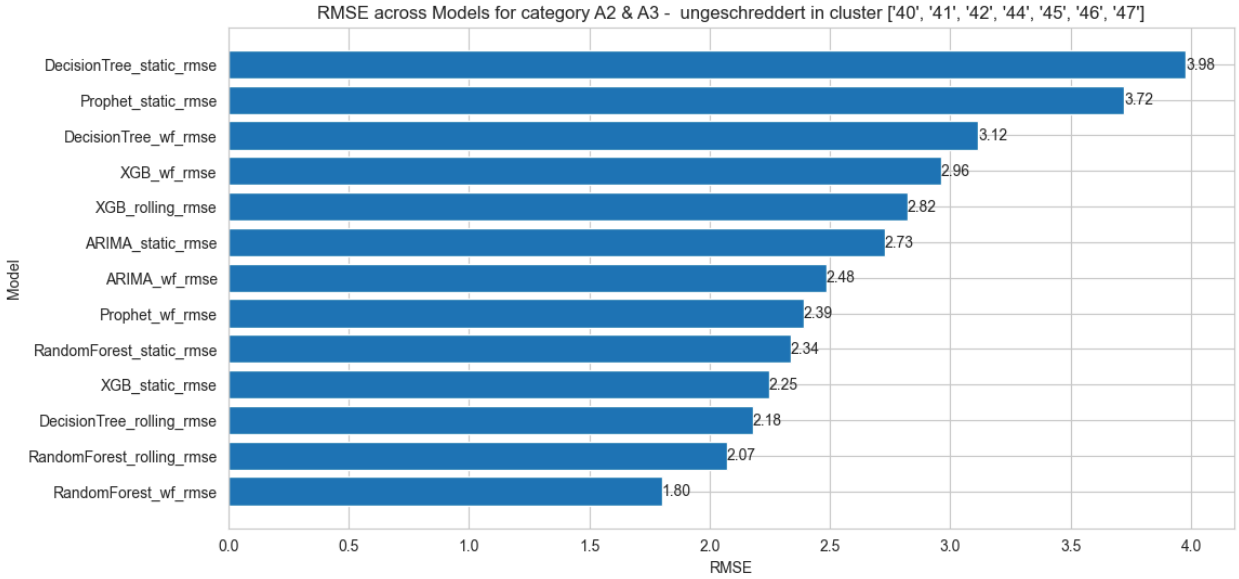

RMSE across Models for category A1 & A2 - geschreddert in cluster ['80', '81', '82', '83', '84', '85', '93', '94']
XGB_wf_rmse
ARIMA_static_rmse
XGB_rolling_rmse
Prophet_static_rmse
RandomForest_static_rmse
ARIMA_wf_rmse
Prophet_wf_rmse
DecisionTree_rolling_rmse
DecisionTree_wf_rmse
RandomForest_rolling_rmse
DecisionTree_static_rmse
RandomForest_wf_rmse
XGB_static_rmse
9.26
9.09
8.04
7.42
7.07
6.01
5.86
5.82
5.35
5.11
5.09
5.02
4.53
Model
0
2
4
6
8
RMSE

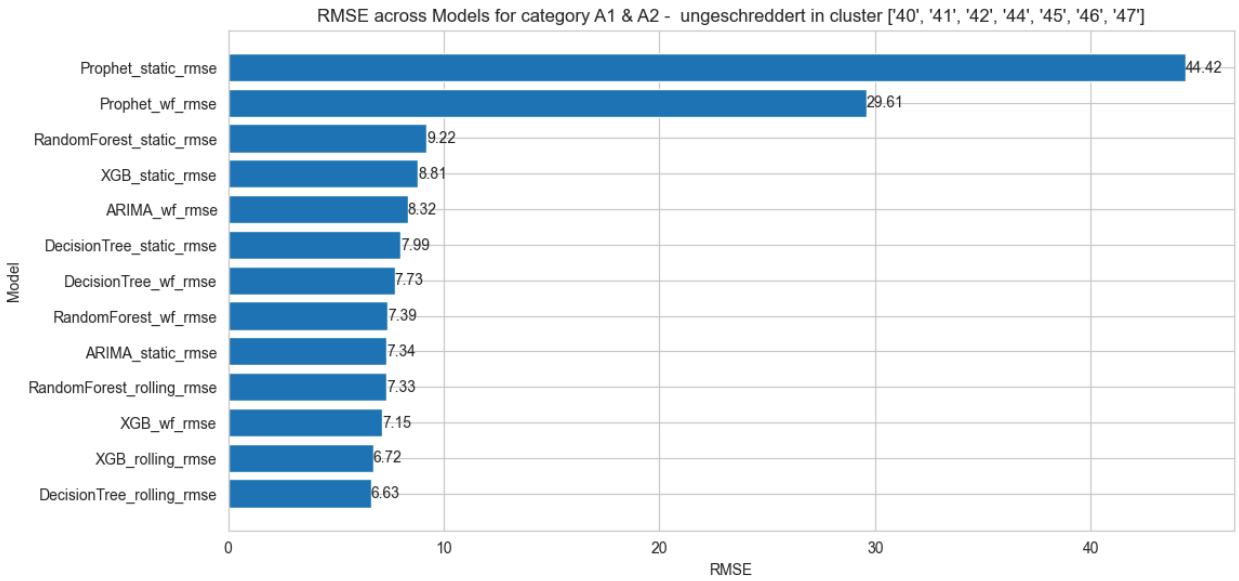

RMSE across Models for category A2 & A3 - geschreddert in cluster ['80', '81', '82', '83', '84', '85', '93', '94']
DecisionTree_rolling_rmse
RandomForest_static_rmse
XGB_static_rmse
ARIMA_static_rmse
Prophet_static_rmse
DecisionTree_static_rmse
DecisionTree_wf_rmse
Prophet_wf_rmse
RandomForest_rolling_rmse
XGB_rolling_rmse
RandomForest_wf_rmse
ARIMA_wf_rmse
XGB_wf_rmse
5.84
5.29
5.19
4.77
4.45
4.41
4.39
3.92
3.46
3.21
2.99
2.83
2.76
Model
0
1
2
3
4
5
6
RMSE

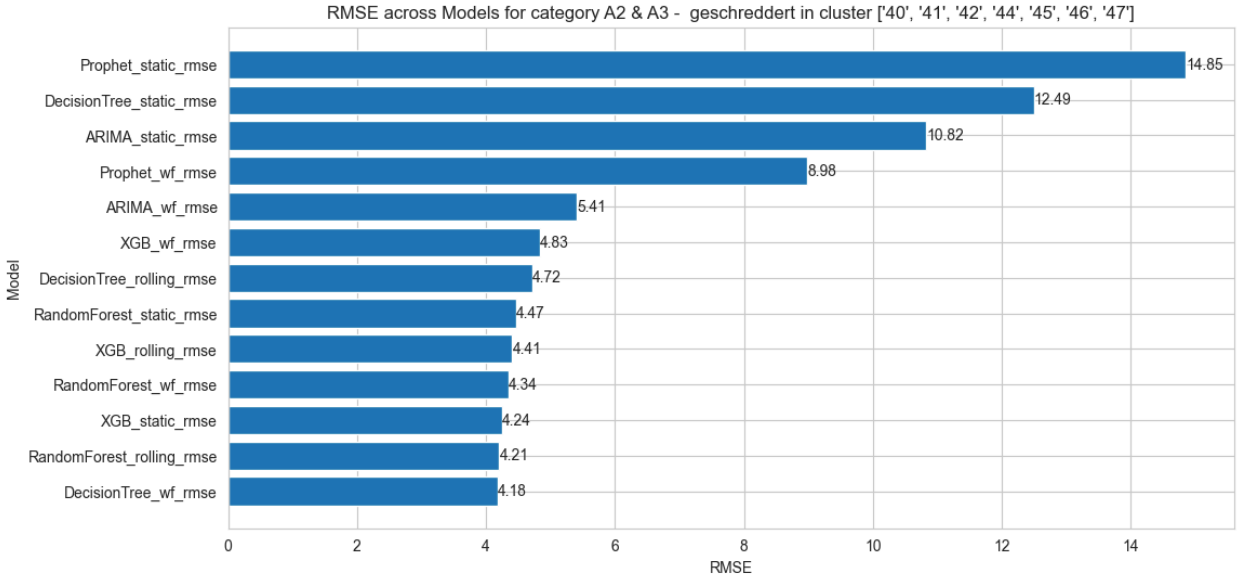

RMSE across Models for category A2 & A3 - geschreddert in cluster ['70', '71', '73', '74', '75', '76']
ARIMA_static_rmse
XGB_static_rmse
DecisionTree_static_rmse
DecisionTree_wf_rmse
XGB_rolling_rmse
Prophet_static_rmse
XGB_wf_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
RandomForest_static_rmse
ARIMA_wf_rmse
Prophet_wf_rmse
DecisionTree_rolling_rmse
28.23
11.75
9.12
8.84
8.31
7.41
6.17
5.88
5.64
5.60
5.28
5.04
4.57
Model
0
5
10
15
20
25
RMSE

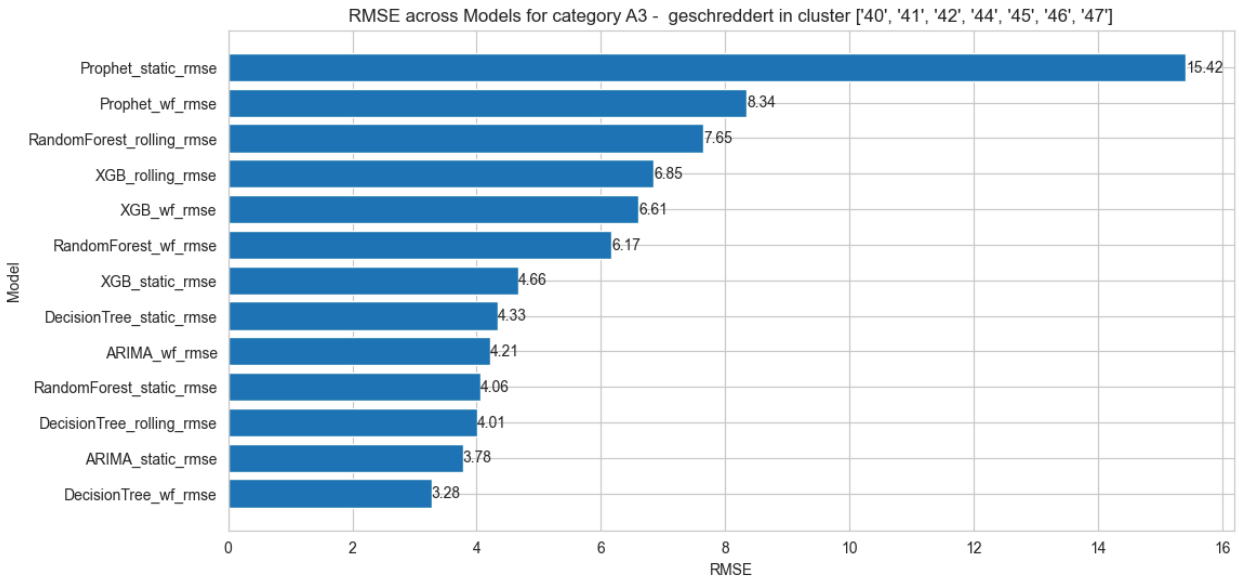

RMSE across Models for category A2 & A3 - geschreddert in cluster ['50', '51', '52', '53']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_static_rmse
XGB_static_rmse
XGB_rolling_rmse
ARIMA_wf_rmse
XGB_wf_rmse
RandomForest_rolling_rmse
RandomForest_static_rmse
RandomForest_wf_rmse
DecisionTree_rolling_rmse
DecisionTree_wf_rmse
ARIMA_static_rmse
9.40
5.82
4.26
3.58
3.54
3.39
3.30
3.13
2.93
2.91
2.75
2.68
2.44
Model
0
2
4
6
8
RMSE

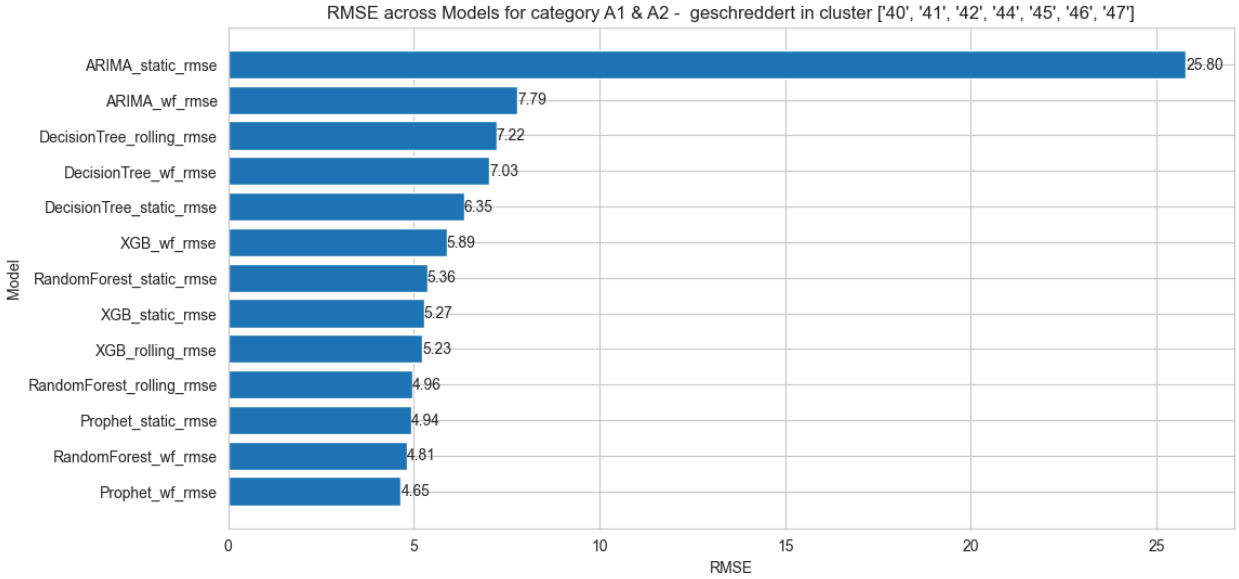

RMSE across Models for category A3 - geschreddert in cluster ['50', '51', '52', '53']
Prophet_static_rmse
ARIMA_static_rmse
Prophet_wf_rmse
DecisionTree_wf_rmse
ARIMA_wf_rmse
XGB_wf_rmse
XGB_rolling_rmse
DecisionTree_rolling_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
DecisionTree_static_rmse
XGB_static_rmse
RandomForest_static_rmse
12.08
9.05
5.91
5.79
4.17
3.91
3.70
3.62
3.57
3.35
3.02
2.74
2.73
Model
0
2
4
6
8
10
12
RMSE

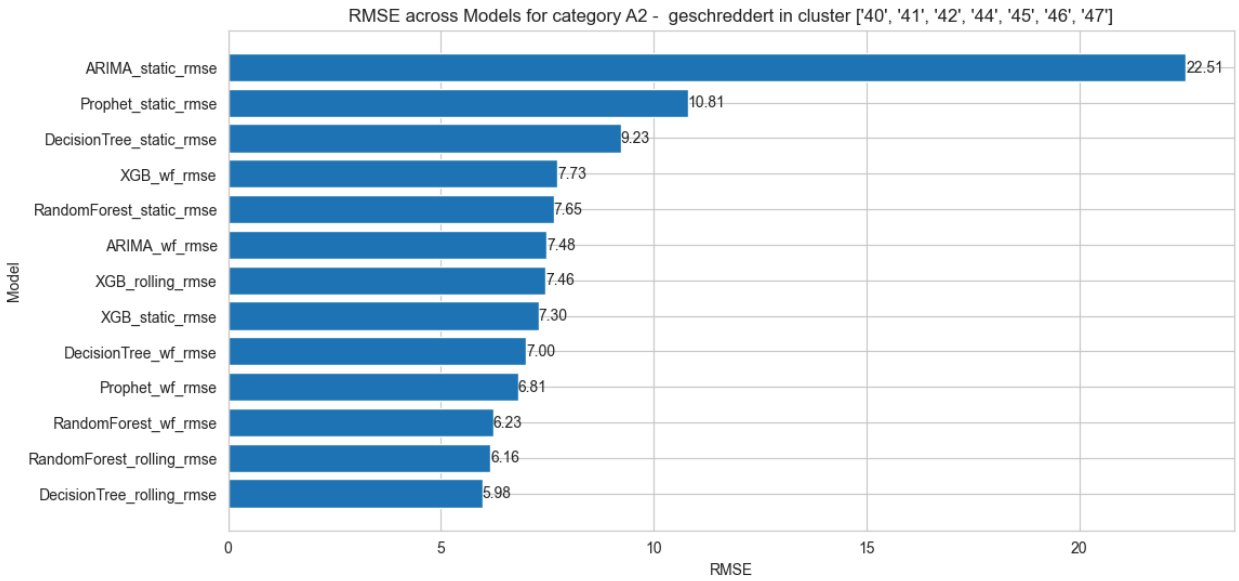

RMSE across Models for category A2 & A3 - ungeschreddert in cluster ['40', '41', '42', '44', '45', '46', '47']
DecisionTree_static_rmse
Prophet_static_rmse
DecisionTree_wf_rmse
XGB_wf_rmse
XGB_rolling_rmse
ARIMA_static_rmse
ARIMA_wf_rmse
Prophet_wf_rmse
RandomForest_static_rmse
XGB_static_rmse
DecisionTree_rolling_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
3.98
3.72
3.12
2.96
2.82
2.73
2.48
2.39
2.34
2.25
2.18
2.07
1.80
Model
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
RMSE

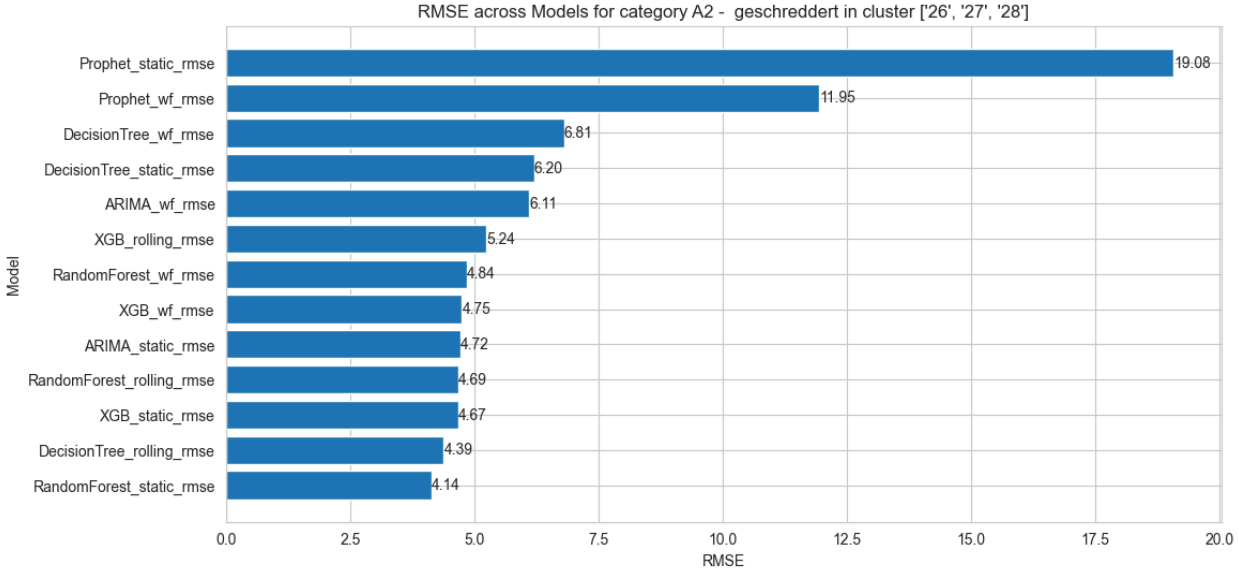

RMSE across Models for category A1 & A2 - ungeschreddert in cluster ['40', '41', '42', '44', '45', '46', '47']
Prophet_static_rmse
Prophet_wf_rmse
RandomForest_static_rmse
XGB_static_rmse
ARIMA_wf_rmse
DecisionTree_static_rmse
DecisionTree_wf_rmse
RandomForest_wf_rmse
ARIMA_static_rmse
RandomForest_rolling_rmse
XGB_wf_rmse
XGB_rolling_rmse
DecisionTree_rolling_rmse
44.42
29.61
9.22
8.81
8.32
7.99
7.73
7.39
7.34
7.33
7.15
6.72
6.63
Model
0
10
20
30
40
RMSE

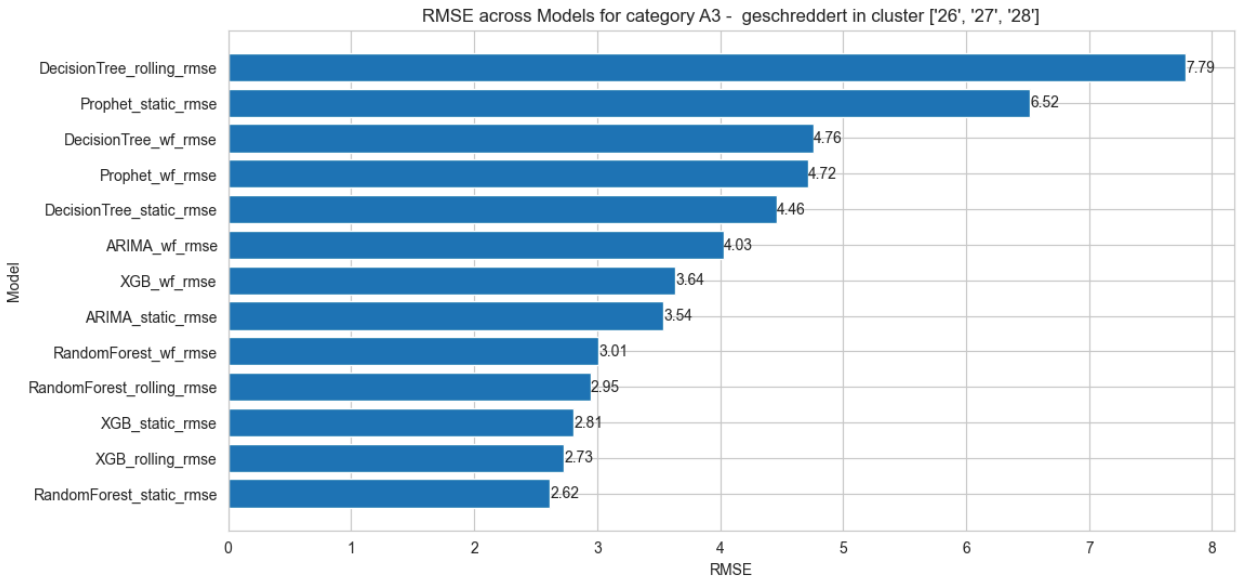

RMSE across Models for category A2 & A3 - geschreddert in cluster ['40', '41', '42', '44', '45', '46', '47']
Prophet_static_rmse
DecisionTree_static_rmse
ARIMA_static_rmse
Prophet_wf_rmse
ARIMA_wf_rmse
XGB_wf_rmse
DecisionTree_rolling_rmse
RandomForest_static_rmse
XGB_rolling_rmse
RandomForest_wf_rmse
XGB_static_rmse
RandomForest_rolling_rmse
DecisionTree_wf_rmse
14.85
12.49
10.82
8.98
5.41
4.83
4.72
4.47
4.41
4.34
4.24
4.21
4.18
Model
0
2
4
6
8
10
12
14
RMSE

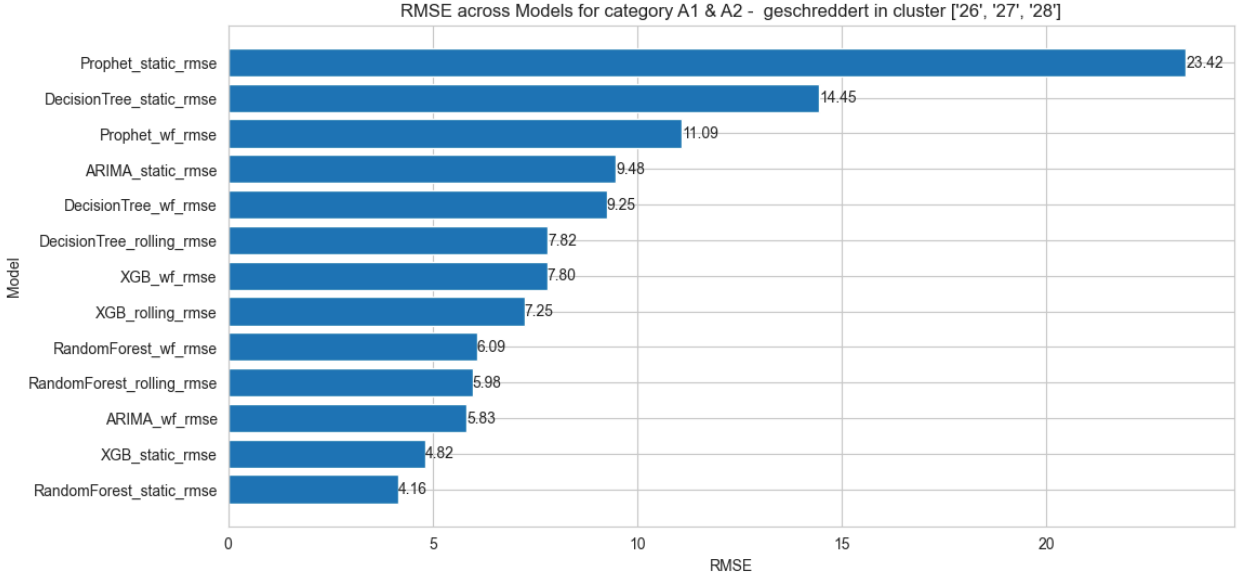

RMSE across Models for category A3 - geschreddert in cluster ['40', '41', '42', '44', '45', '46', '47']
Prophet_static_rmse
Prophet_wf_rmse
RandomForest_rolling_rmse
XGB_rolling_rmse
XGB_wf_rmse
RandomForest_wf_rmse
XGB_static_rmse
DecisionTree_static_rmse
ARIMA_wf_rmse
RandomForest_static_rmse
DecisionTree_rolling_rmse
ARIMA_static_rmse
DecisionTree_wf_rmse
15.42
8.34
7.65
6.85
6.61
6.17
4.66
4.33
4.21
4.06
4.01
3.78
3.28
Model
0
2
4
6
8
10
12
14
16
RMSE

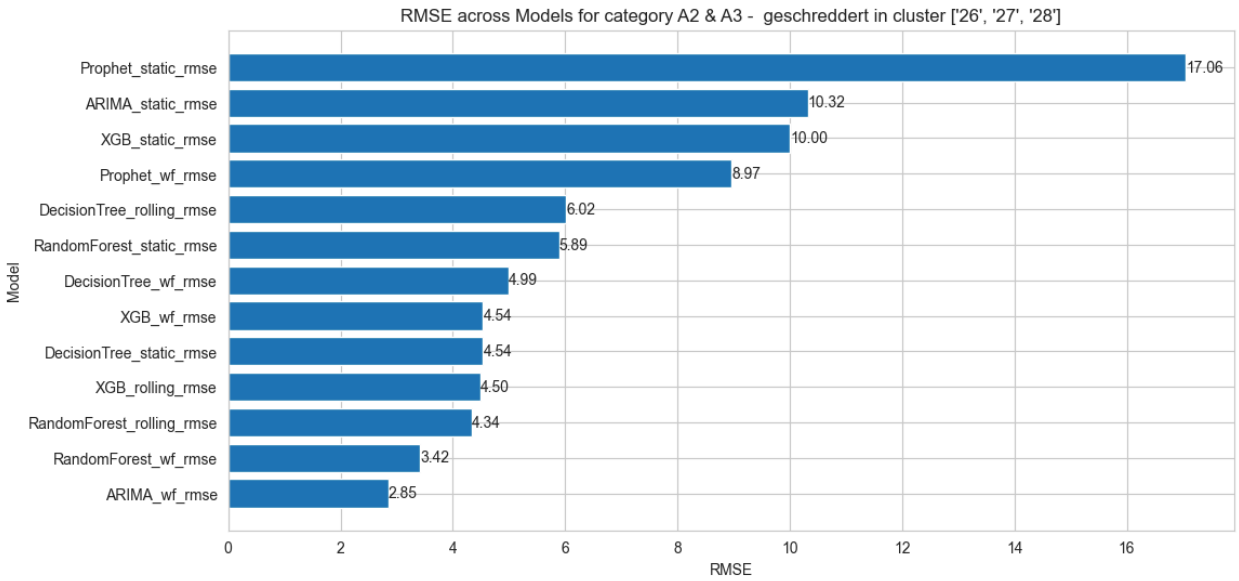

RMSE across Models for category A1 & A2 - geschreddert in cluster ['40', '41', '42', '44', '45', '46', '47']
ARIMA_static_rmse
ARIMA_wf_rmse
DecisionTree_rolling_rmse
DecisionTree_wf_rmse
DecisionTree_static_rmse
XGB_wf_rmse
RandomForest_static_rmse
XGB_static_rmse
XGB_rolling_rmse
RandomForest_rolling_rmse
Prophet_static_rmse
RandomForest_wf_rmse
Prophet_wf_rmse
25.80
7.79
7.22
7.03
6.35
5.89
5.36
5.27
5.23
4.96
4.94
4.81
4.65
Model
0
5
10
15
20
25
RMSE

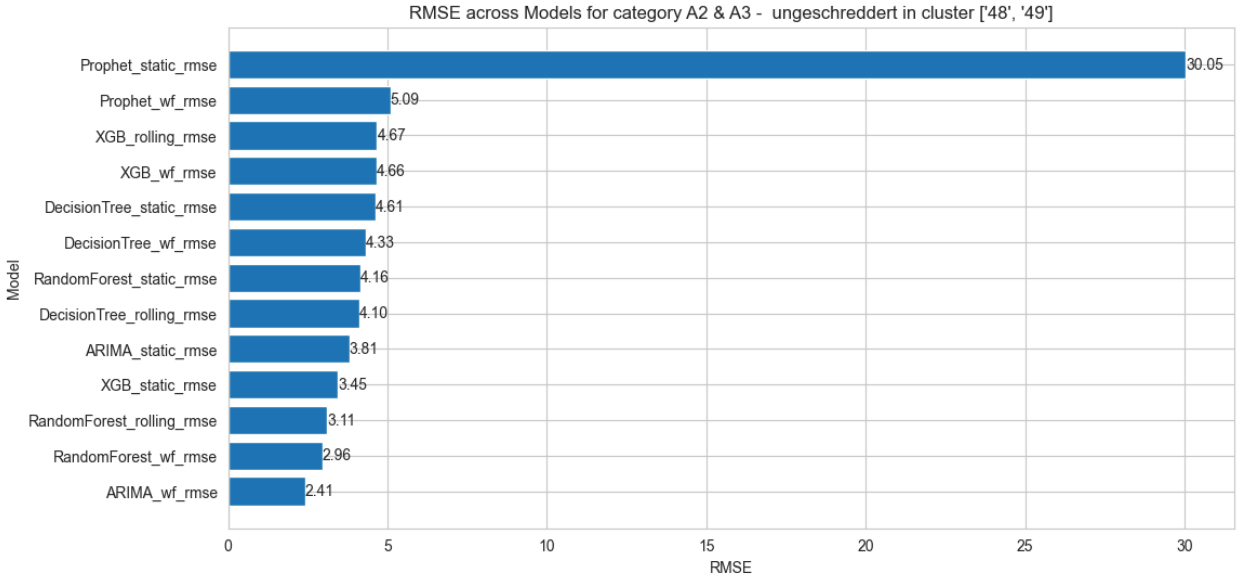

RMSE across Models for category A2 - geschreddert in cluster ['40', '41', '42', '44', '45', '46', '47']
ARIMA_static_rmse
Prophet_static_rmse
DecisionTree_static_rmse
XGB_wf_rmse
RandomForest_static_rmse
ARIMA_wf_rmse
XGB_rolling_rmse
XGB_static_rmse
DecisionTree_wf_rmse
Prophet_wf_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
DecisionTree_rolling_rmse
22.51
10.81
9.23
7.73
7.65
7.48
7.46
7.30
7.00
6.81
6.23
6.16
5.98
Model
0
5
10
15
20
RMSE

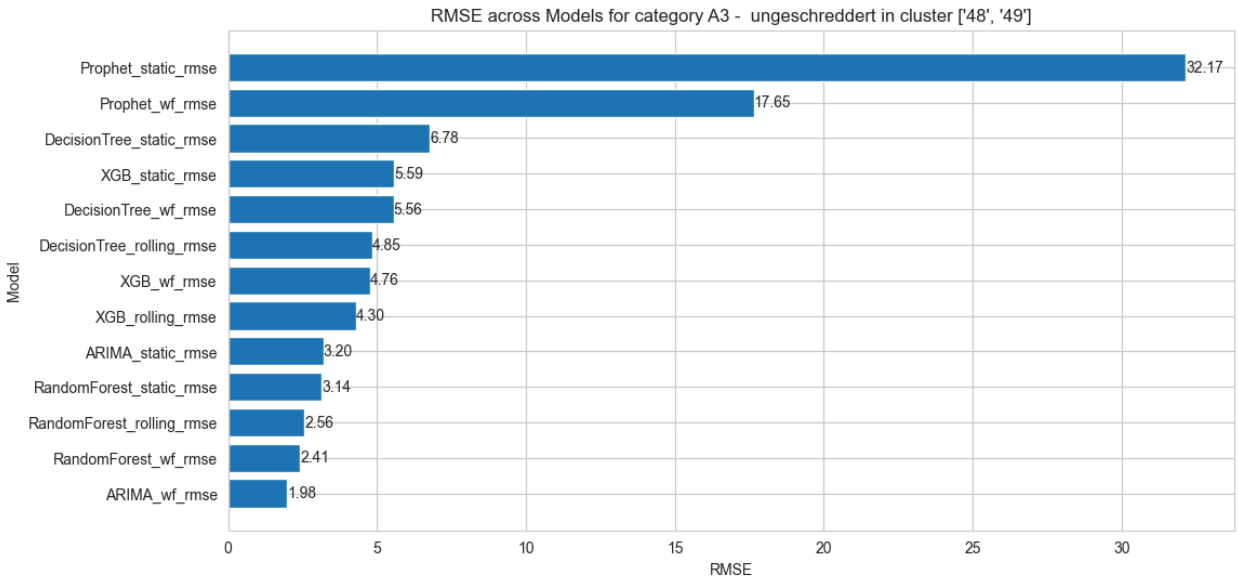

RMSE across Models for category A2 - geschreddert in cluster ['26', '27', '28']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_wf_rmse
DecisionTree_static_rmse
ARIMA_wf_rmse
XGB_rolling_rmse
RandomForest_wf_rmse
XGB_wf_rmse
ARIMA_static_rmse
RandomForest_rolling_rmse
XGB_static_rmse
DecisionTree_rolling_rmse
RandomForest_static_rmse
19.08
11.95
6.81
6.20
6.11
5.24
4.84
4.75
4.72
4.69
4.67
4.39
4.14
Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
RMSE

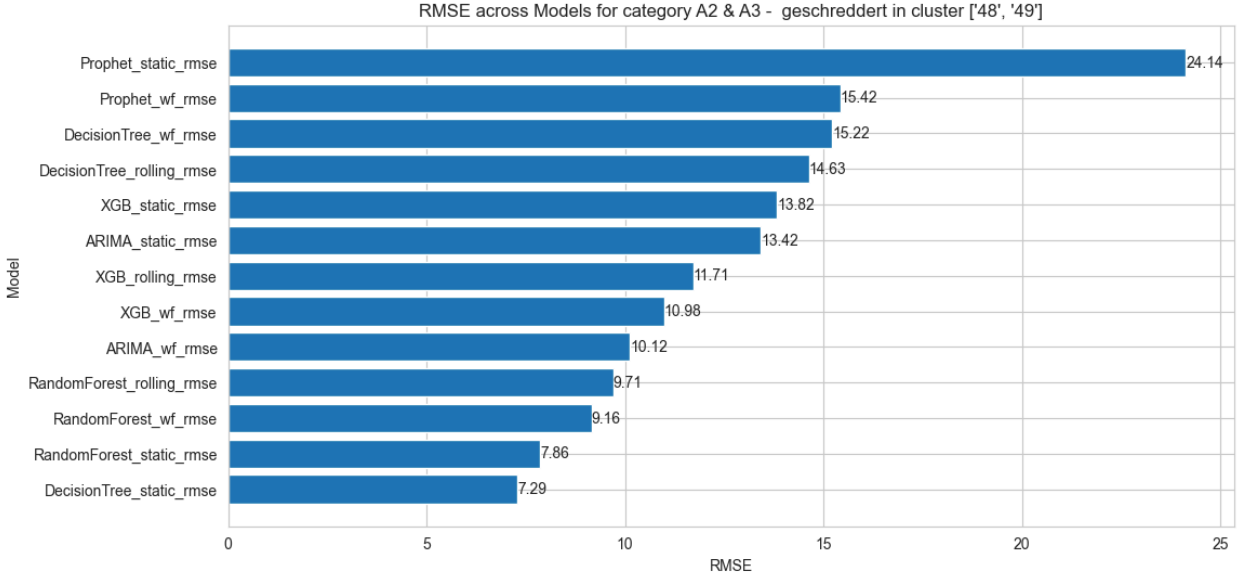

RMSE across Models for category A3 - geschreddert in cluster ['26', '27', '28']
DecisionTree_rolling_rmse
Prophet_static_rmse
DecisionTree_wf_rmse
Prophet_wf_rmse
DecisionTree_static_rmse
ARIMA_wf_rmse
XGB_wf_rmse
ARIMA_static_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
XGB_static_rmse
XGB_rolling_rmse
RandomForest_static_rmse
7.79
6.52
4.76
4.72
4.46
4.03
3.64
3.54
3.01
2.95
2.81
2.73
2.62
Model
RMSE
0
1
2
3
4
5
6
7
8

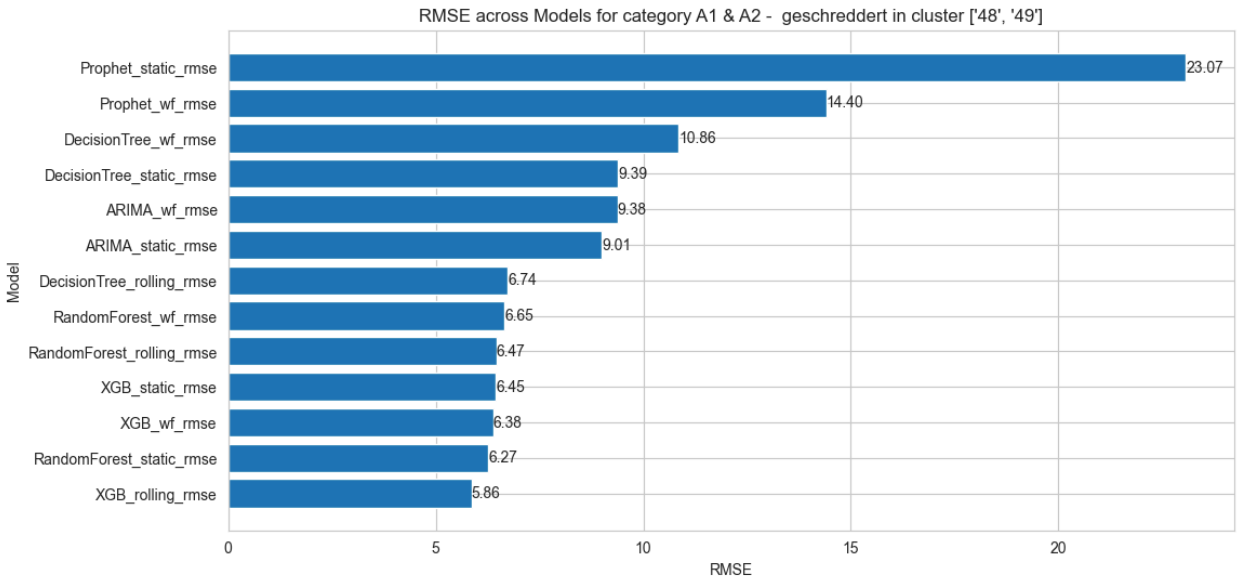

RMSE across Models for category A1 & A2 - geschreddert in cluster ['26', '27', '28']
Prophet_static_rmse
DecisionTree_static_rmse
Prophet_wf_rmse
ARIMA_static_rmse
DecisionTree_wf_rmse
DecisionTree_rolling_rmse
XGB_wf_rmse
XGB_rolling_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
ARIMA_wf_rmse
XGB_static_rmse
RandomForest_static_rmse
23.42
14.45
11.09
9.48
9.25
7.82
7.80
7.25
6.09
5.98
5.83
4.82
4.16
Model
RMSE
0
5
10
15
20

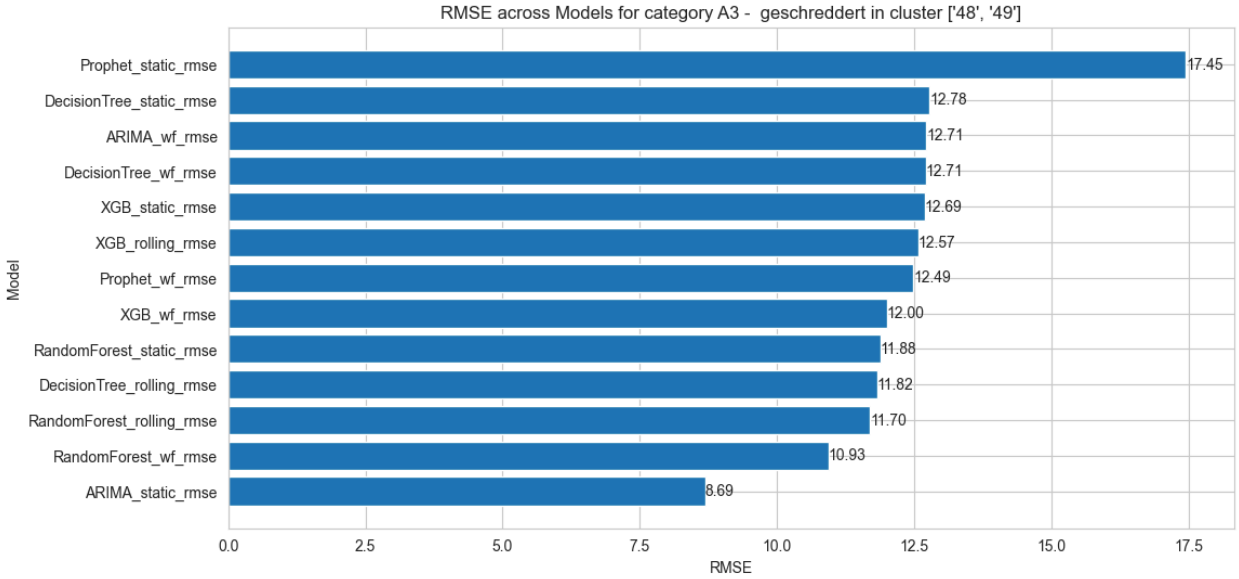

RMSE across Models for category A2 & A3 - geschreddert in cluster ['26', '27', '28']
Prophet_static_rmse
ARIMA_static_rmse
XGB_static_rmse
Prophet_wf_rmse
DecisionTree_rolling_rmse
RandomForest_static_rmse
DecisionTree_wf_rmse
XGB_wf_rmse
DecisionTree_static_rmse
XGB_rolling_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
ARIMA_wf_rmse
17.06
10.32
10.00
8.97
6.02
5.89
4.99
4.54
4.54
4.50
4.34
3.42
2.85
Model
0
2
4
6
8
10
12
14
16
RMSE

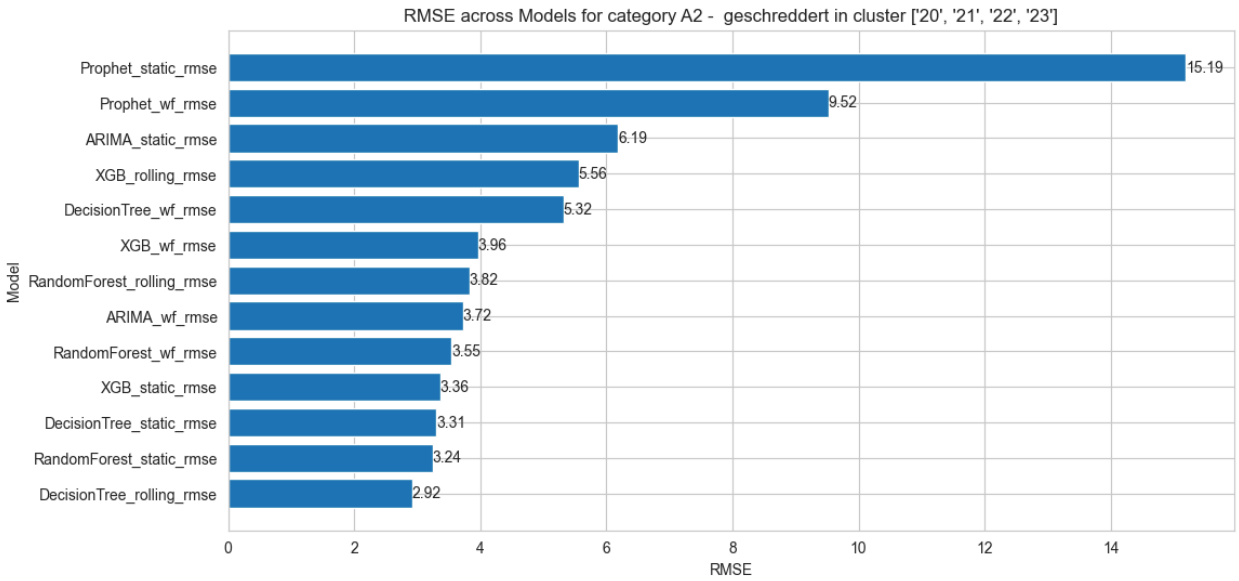

RMSE across Models for category A2 & A3 - ungeschreddert in cluster ['48', '49']
Prophet_static_rmse
Prophet_wf_rmse
XGB_rolling_rmse
XGB_wf_rmse
DecisionTree_static_rmse
DecisionTree_wf_rmse
RandomForest_static_rmse
DecisionTree_rolling_rmse
ARIMA_static_rmse
XGB_static_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
ARIMA_wf_rmse
30.05
5.09
4.67
4.66
4.61
4.33
4.16
4.10
3.81
3.45
3.11
2.96
2.41
Model
0
5
10
15
20
25
30
RMSE

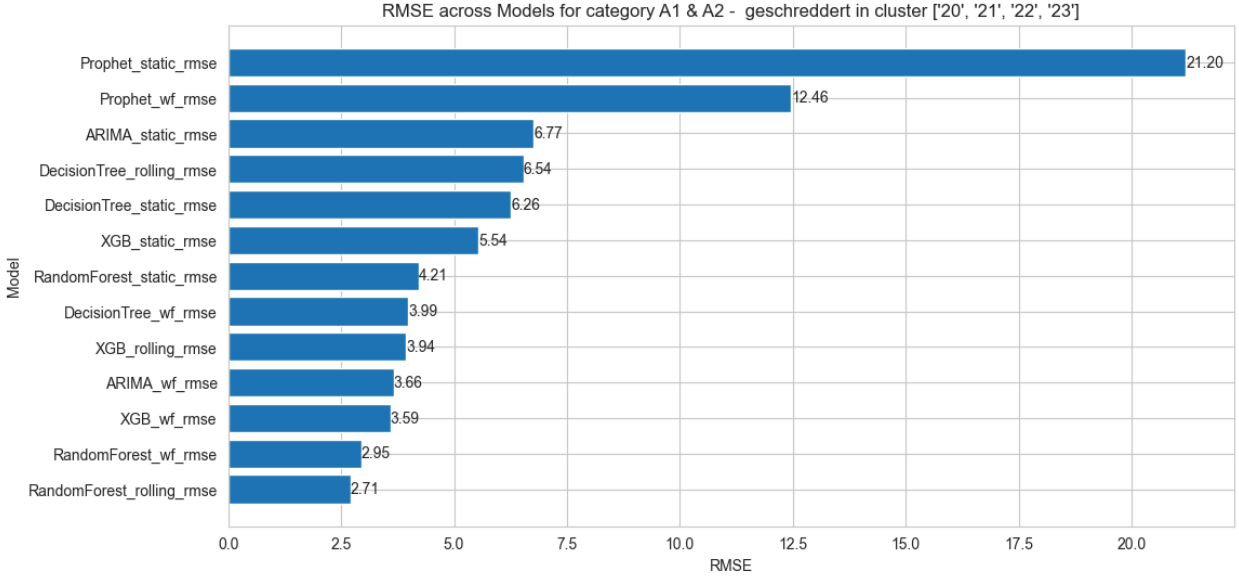

RMSE across Models for category A3 - ungeschreddert in cluster ['48', '49']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_static_rmse
XGB_static_rmse
DecisionTree_wf_rmse
DecisionTree_rolling_rmse
XGB_wf_rmse
XGB_rolling_rmse
ARIMA_static_rmse
RandomForest_static_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
ARIMA_wf_rmse
32.17
17.65
6.78
5.59
5.56
4.85
4.76
4.30
3.20
3.14
2.56
2.41
1.98
Model
RMSE
0
5
10
15
20
25
30

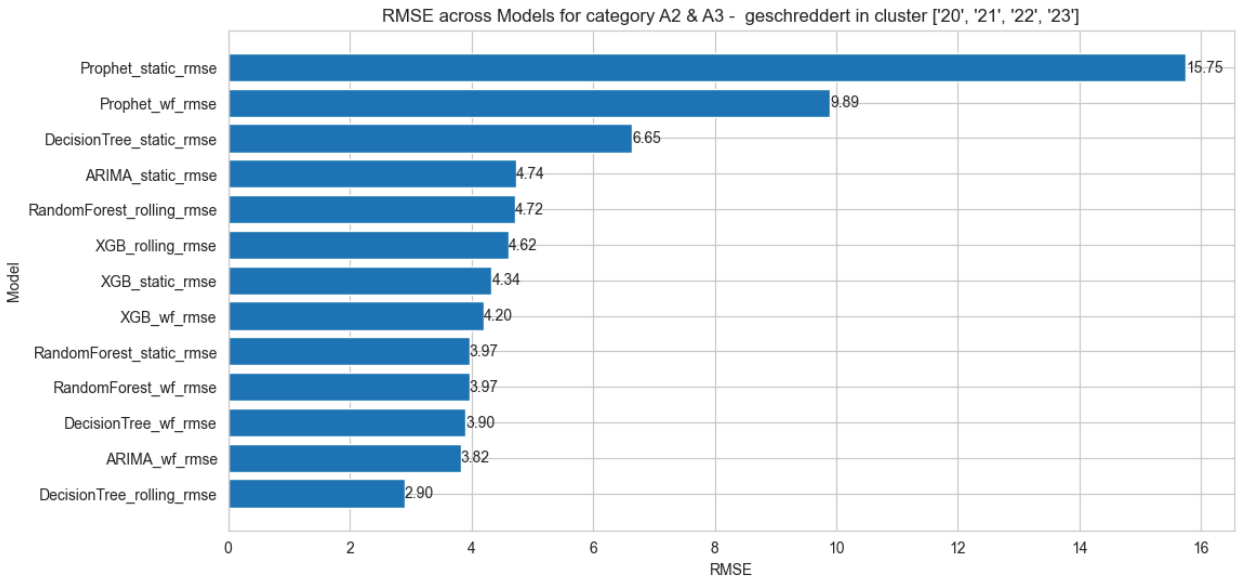

RMSE across Models for category A2 & A3 - geschreddert in cluster ['48', '49']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_wf_rmse
DecisionTree_rolling_rmse
XGB_static_rmse
ARIMA_static_rmse
XGB_rolling_rmse
XGB_wf_rmse
ARIMA_wf_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
RandomForest_static_rmse
DecisionTree_static_rmse
24.14
15.42
15.22
14.63
13.82
13.42
11.71
10.98
10.12
9.71
9.16
7.86
7.29
Model
RMSE
0
5
10
15
20
25

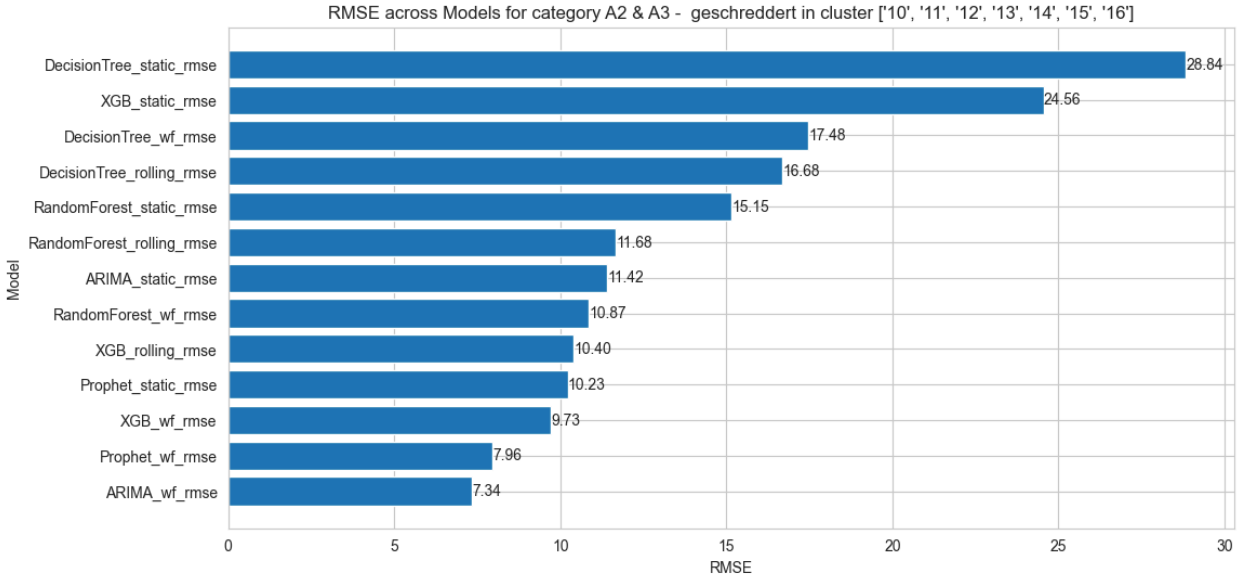

RMSE across Models for category A1 & A2 - geschreddert in cluster ['48', '49']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_wf_rmse
DecisionTree_static_rmse
ARIMA_wf_rmse
ARIMA_static_rmse
DecisionTree_rolling_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
XGB_static_rmse
XGB_wf_rmse
RandomForest_static_rmse
XGB_rolling_rmse
23.07
14.40
10.86
9.39
9.38
9.01
6.74
6.65
6.47
6.45
6.38
6.27
5.86
Model
0
5
10
15
20
RMSE

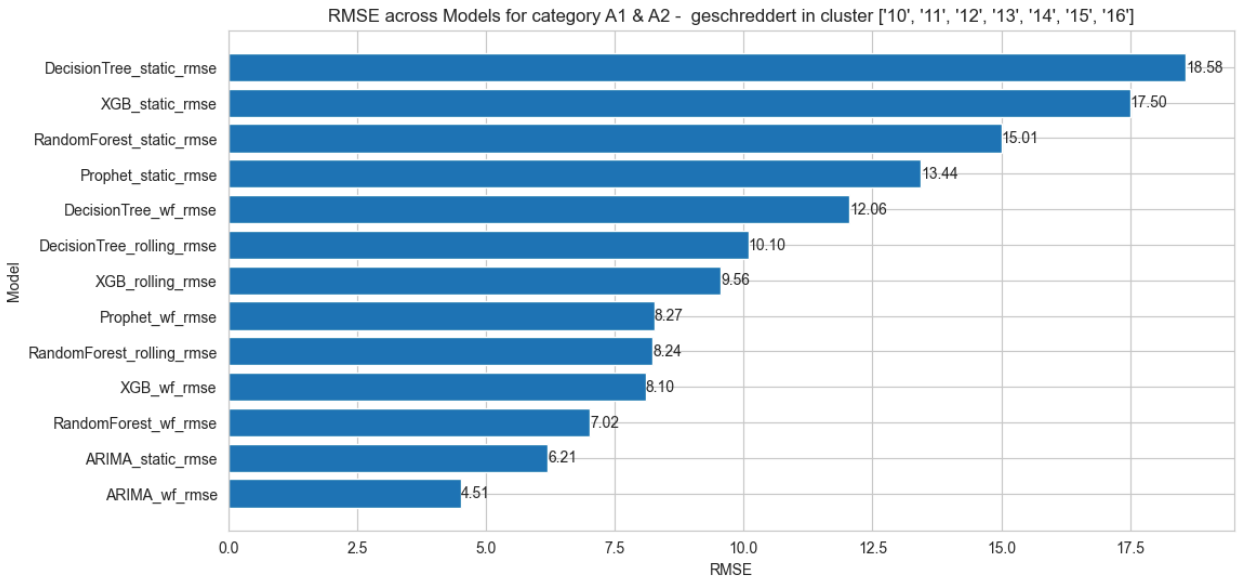

RMSE across Models for category A3 - geschreddert in cluster ['48', '49']
Prophet_static_rmse
DecisionTree_static_rmse
ARIMA_wf_rmse
DecisionTree_wf_rmse
XGB_static_rmse
XGB_rolling_rmse
Prophet_wf_rmse
XGB_wf_rmse
RandomForest_static_rmse
DecisionTree_rolling_rmse
RandomForest_rolling_rmse
RandomForest_wf_rmse
ARIMA_static_rmse
17.45
12.78
12.71
12.71
12.69
12.57
12.49
12.00
11.88
11.82
11.70
10.93
8.69
Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
RMSE

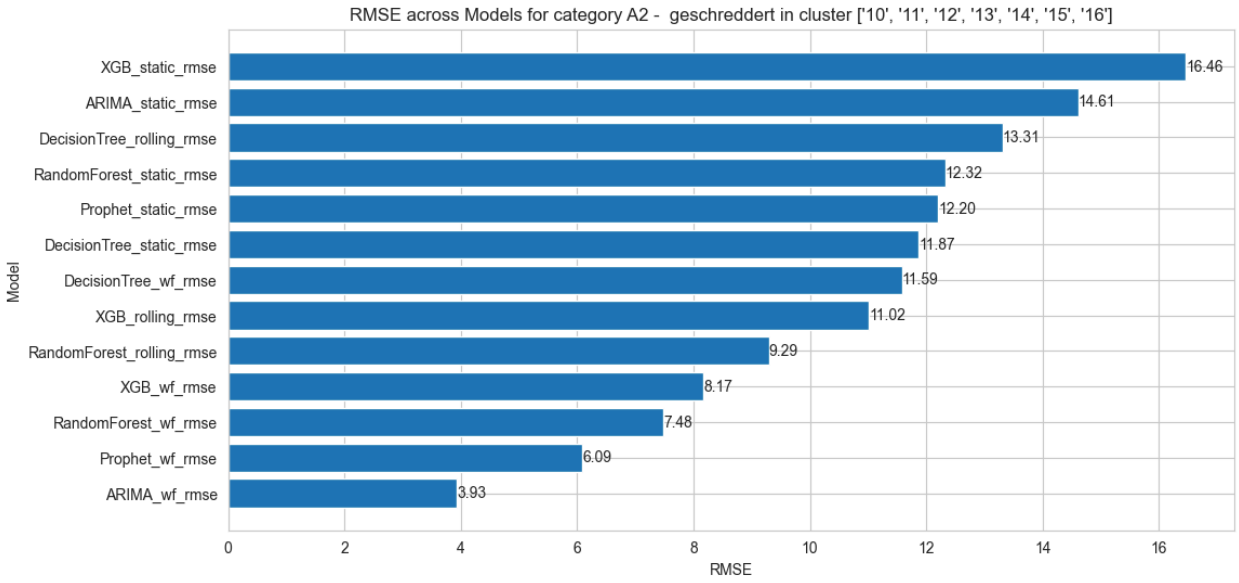

RMSE across Models for category A2 - geschreddert in cluster ['20', '21', '22', '23']
Prophet_static_rmse
Prophet_wf_rmse
ARIMA_static_rmse
XGB_rolling_rmse
DecisionTree_wf_rmse
XGB_wf_rmse
RandomForest_rolling_rmse
ARIMA_wf_rmse
RandomForest_wf_rmse
XGB_static_rmse
DecisionTree_static_rmse
RandomForest_static_rmse
DecisionTree_rolling_rmse
15.19
9.52
6.19
5.56
5.32
3.96
3.82
3.72
3.55
3.36
3.31
3.24
2.92
Model
0
2
4
6
8
10
12
14
RMSE

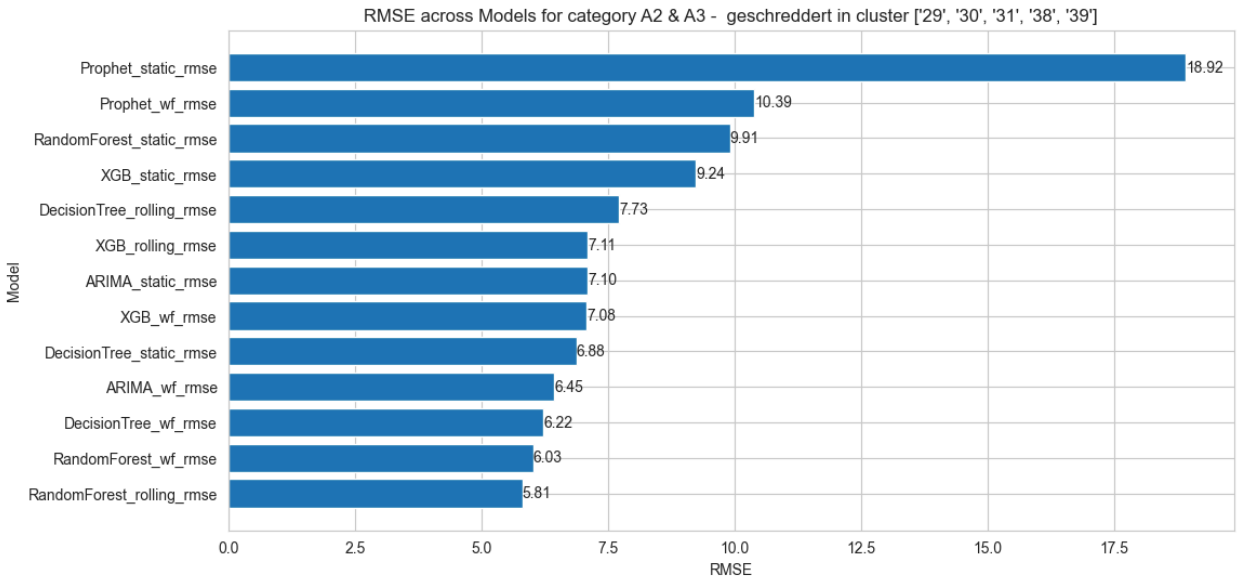

RMSE across Models for category A1 & A2 - geschreddert in cluster ['20', '21', '22', '23']
Prophet_static_rmse
Prophet_wf_rmse
ARIMA_static_rmse
DecisionTree_rolling_rmse
DecisionTree_static_rmse
XGB_static_rmse
RandomForest_static_rmse
DecisionTree_wf_rmse
XGB_rolling_rmse
ARIMA_wf_rmse
XGB_wf_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
21.20
12.46
6.77
6.54
6.26
5.54
4.21
3.99
3.94
3.66
3.59
2.95
2.71
Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
RMSE

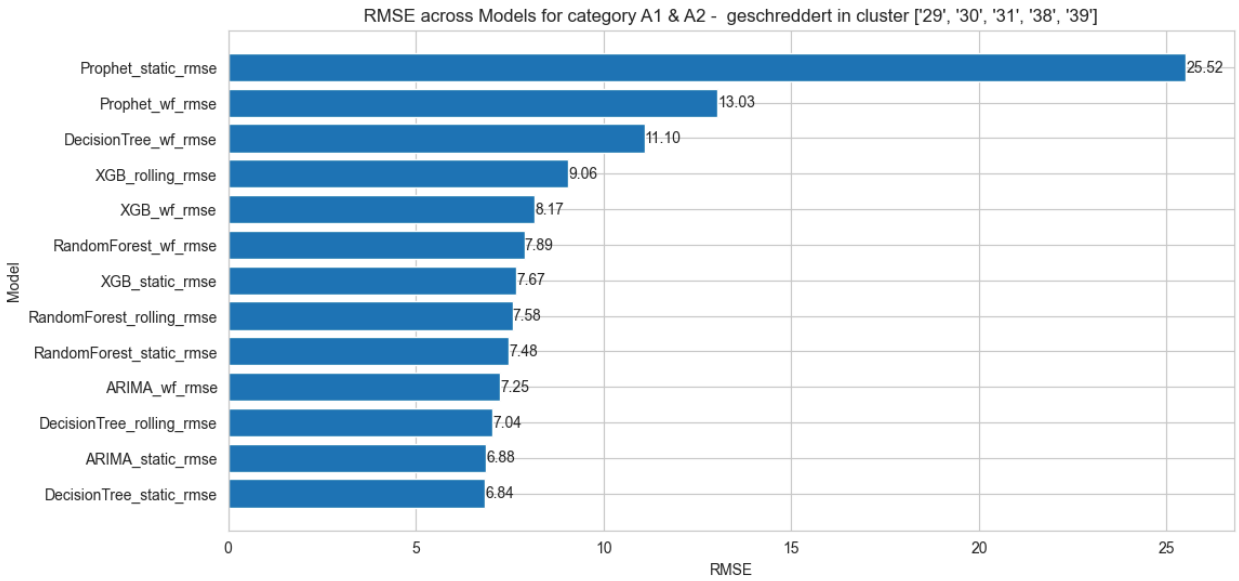

RMSE across Models for category A2 & A3 - geschreddert in cluster ['20', '21', '22', '23']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_static_rmse
ARIMA_static_rmse
RandomForest_rolling_rmse
XGB_rolling_rmse
XGB_static_rmse
XGB_wf_rmse
RandomForest_static_rmse
RandomForest_wf_rmse
DecisionTree_wf_rmse
ARIMA_wf_rmse
DecisionTree_rolling_rmse
15.75
9.89
6.65
4.74
4.72
4.62
4.34
4.20
3.97
3.97
3.90
3.82
2.90
Model
0
2
4
6
8
10
12
14
16
RMSE

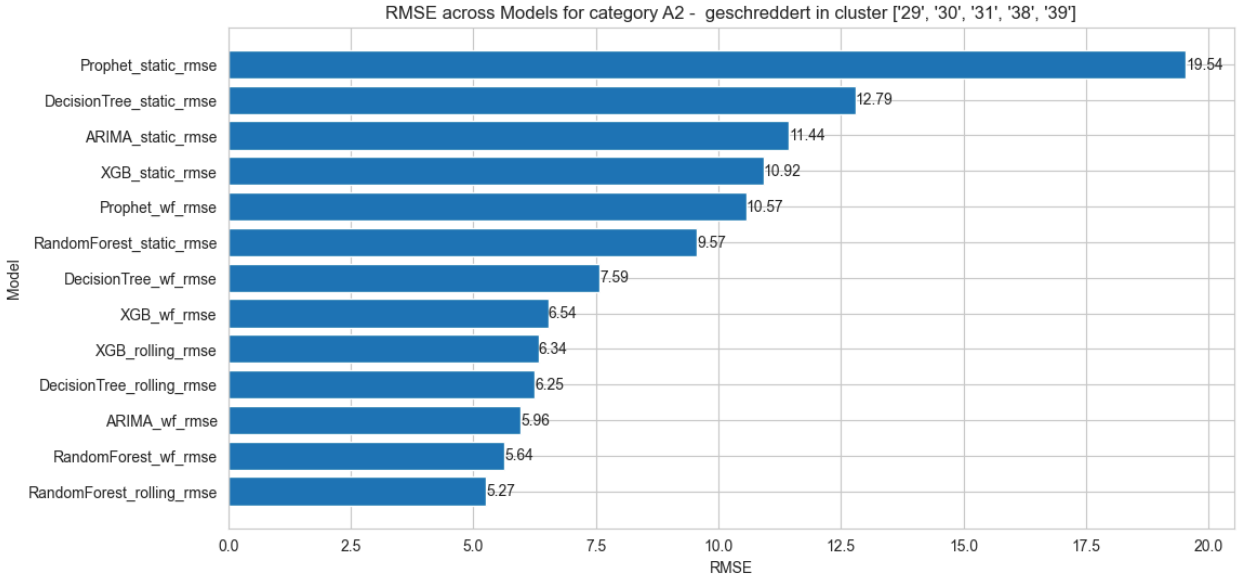

RMSE across Models for category A2 & A3 - geschreddert in cluster ['10', '11', '12', '13', '14', '15', '16']
DecisionTree_static_rmse
XGB_static_rmse
DecisionTree_wf_rmse
DecisionTree_rolling_rmse
RandomForest_static_rmse
RandomForest_rolling_rmse
ARIMA_static_rmse
RandomForest_wf_rmse
XGB_rolling_rmse
Prophet_static_rmse
XGB_wf_rmse
Prophet_wf_rmse
ARIMA_wf_rmse
28.84
24.56
17.48
16.68
15.15
11.68
11.42
10.87
10.40
10.23
9.73
7.96
7.34
Model
0
5
10
15
20
25
30
RMSE


RMSE across Models for category A1 & A2 - geschreddert in cluster ['10', '11', '12', '13', '14', '15', '16']
DecisionTree_static_rmse
XGB_static_rmse
RandomForest_static_rmse
Prophet_static_rmse
DecisionTree_wf_rmse
DecisionTree_rolling_rmse
XGB_rolling_rmse
Prophet_wf_rmse
RandomForest_rolling_rmse
XGB_wf_rmse
RandomForest_wf_rmse
ARIMA_static_rmse
ARIMA_wf_rmse
18.58
17.50
15.01
13.44
12.06
10.10
9.56
8.27
8.24
8.10
7.02
6.21
4.51
Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
RMSE


RMSE across Models for category A2 - geschreddert in cluster ['10', '11', '12', '13', '14', '15', '16']
XGB_static_rmse
ARIMA_static_rmse
DecisionTree_rolling_rmse
RandomForest_static_rmse
Prophet_static_rmse
DecisionTree_static_rmse
DecisionTree_wf_rmse
XGB_rolling_rmse
RandomForest_rolling_rmse
XGB_wf_rmse
RandomForest_wf_rmse
Prophet_wf_rmse
ARIMA_wf_rmse
16.46
14.61
13.31
12.32
12.20
11.87
11.59
11.02
9.29
8.17
7.48
6.09
3.93
Model
0
2
4
6
8
10
12
14
16
RMSE


RMSE across Models for category A2 & A3 - geschreddert in cluster ['29', '30', '31', '38', '39']
Prophet_static_rmse
Prophet_wf_rmse
RandomForest_static_rmse
XGB_static_rmse
DecisionTree_rolling_rmse
XGB_rolling_rmse
ARIMA_static_rmse
XGB_wf_rmse
DecisionTree_static_rmse
ARIMA_wf_rmse
DecisionTree_wf_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
18.92
10.39
9.91
9.24
7.73
7.11
7.10
7.08
6.88
6.45
6.22
6.03
5.81
Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
RMSE


RMSE across Models for category A1 & A2 - geschreddert in cluster ['29', '30', '31', '38', '39']
Prophet_static_rmse
Prophet_wf_rmse
DecisionTree_wf_rmse
XGB_rolling_rmse
XGB_wf_rmse
RandomForest_wf_rmse
XGB_static_rmse
RandomForest_rolling_rmse
RandomForest_static_rmse
ARIMA_wf_rmse
DecisionTree_rolling_rmse
ARIMA_static_rmse
DecisionTree_static_rmse
25.52
13.03
11.10
9.06
8.17
7.89
7.67
7.58
7.48
7.25
7.04
6.88
6.84
Model
0
5
10
15
20
25
RMSE


RMSE across Models for category A2 - geschreddert in cluster ['29', '30', '31', '38', '39']
Prophet_static_rmse
DecisionTree_static_rmse
ARIMA_static_rmse
XGB_static_rmse
Prophet_wf_rmse
RandomForest_static_rmse
DecisionTree_wf_rmse
XGB_wf_rmse
XGB_rolling_rmse
DecisionTree_rolling_rmse
ARIMA_wf_rmse
RandomForest_wf_rmse
RandomForest_rolling_rmse
19.54
12.79
11.44
10.92
10.57
9.57
7.59
6.54
6.34
6.25
5.96
5.64
5.27
Model
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
RMSE