Time series forecasting of picking tasks in warehouse zones
a case study

Braspenning, Steven J.C.

Award date:
2023

Department of Industrial Engineering and Innovation Sciences
Information Systems Research Group

Time series forecasting of picking tasks in warehouse zones: a case study

Master Thesis in partial fulfillment of the requirements for the degree of Master of Science in Operations Management and Logistics

Author:
S.J.C. (Steven) Braspenning - 1002943

Supervisors:
Dr. L. Genga - TU/e
Dr. Z. Bukhsh - TU/e
C. Tegelaers - Pipple

March 6, 2023

Preface

This thesis presents my master graduation project conducted at Pipple, for DHL. This thesis also marks the end of my studies at the University of Technology Eindhoven (TU/e) which I thoroughly enjoyed. I would like to take this opportunity to show my gratitude to the people who have supported me along the way.

First of all, I would like to thank my TU/e supervisors who have supported me during my studies. In particular, I would like to thank Ms. Laura Genga, who was my mentor during my master's Operations, Management, and Logistics (OML) and supervised me during my master thesis. I really enjoyed the discussions we had which gave me valuable insights on the project. Furthermore, I admire how you were always available to discuss every topic, and appreciate the support and guidance you gave me during the length of my master thesis.

I would also like to show my gratitude to the people at Pipple. Your open and inclusive culture made me feel at home the moment I started my graduate internship. I would like to especially thank Ms. Claudia Tegelaers for supervising my project and Ms. Vera van der Lelij for helping me find my way inside Pipple. Claudia, I highly value the insights and different perspectives you provided me with during the project. Moreover, I admire your ability to come to the core of problems and to never be convinced until someone has come up with valid argumentation, or you have figured it out yourself. Vera, I want to thank you for checking in on me during my project and for the flexibility you provide to graduate interns. I also want to thank DHL, especially Mr. Giel Stevens, for providing the context and data and being open to discuss and answer questions about the project.

Lastly, I want to thank everyone else that has supported me or contributed to my academic career until now. Especially, the most important people in my life, my family and friends. In particular, I would like to thank my parents. Without your unconditional support and encouragement in both good and bad times, I would not have nearly been where I am now. I will be forever grateful for that.

I have been able to develop myself, both academically and personally, during my studies at the TU/e, as well as, during my master thesis project. I am proud of my academic journey, including the master thesis project that you are currently reading. I hope you enjoy the rest of it.

Steven Braspenning, March 2023

Abstract

Accurate workload forecasts for order pickers in warehouses are essential when basing capacity planning of order pickers on these forecasts. Accurate forecasts provide the opportunity to tailor the capacity planning to the present as well as to the future. This research presents a case study on multi-step-ahead time series forecasting of incoming picking tasks in a warehouse, for picking zones and departure buckets, on zero-inflated time series data. To forecast the number of incoming picking tasks, three statistical models: exponential smoothing (ES), autoregressive integrated moving average (ARIMA), and Croston method, and two machine learning models: multilayer perceptron (MLP), and long short-term memory (LSTM), are compared to a benchmark. Features are engineered from the time series data and included in the input for the forecasting models that allow this. Furthermore, hyperparameter tuning and data preprocessing are performed for each individual forecasting model. The forecasts of the models are evaluated by making use of three different performance metrics. Overall, it is found that all the tested forecasting models outperform the benchmark model in this thesis. Furthermore, statistical forecasting models that include a seasonal component overall perform best on zero-inflated time series data. ARIMAX is the forecasting model that performs best when one model is applied to all time series together. When the best-performing forecasting models are applied to time series groups, based on performance on aggregation levels in the data, a small improvement in forecasting performance is achieved.

Keywords: Time series forecasting, Multi-step-ahead forecasting, Exponential smoothing, ARIMA, Croston method, Neural networks, MLP, LSTM

Executive summary

This master thesis compares different time series forecasting models that forecast the number of incoming picking tasks in the warehouse of DHL. The time series data that is forecasted on, has different aggregation levels which are the picking zones and departure buckets in the warehouse. These aggregations cause the time series data to be zero-inflated.

Problem statement

Effective capacity planning of order pickers in warehouses is essential for delivery services to thrive in current times. Including forecasts on the expected workload of order pickers, which is equal to the number of incoming picking tasks, in the capacity planning can yield more effective capacity planning as the planning can be tailored to the present as well as the future. However, to achieve a more effective capacity planning when the forecasts are included rather than making the capacity planning on the present workload, the forecasts on which the capacity planning is based should be accurate. Currently, in the warehouse of DHL, forecasts of the incoming number of picking tasks are made. This forecast model serves as the benchmark of this thesis. The forecasts of this model are however not yet Incorporated in the capacity planning as DHL wants to evaluate if the benchmark makes good quality forecasts or if other forecasting models could provide better forecasting performance before including the forecasts in the capacity planning. This thesis aims to improve the accuracy of the benchmark forecasting model by testing different forecasting models on the time series data of incoming picking tasks. The main research question of this thesis therefore is:

How can picking tasks in the warehouse of DHL be more accurately predicted by using time series forecasting?

To give a grounded answer to this research question, a research methodology is constructed. This involves all the steps needed such as reviewing literature, data preparation, analysis of the time series, data preprocessing, and the selection and testing of forecasting models, that in the end lead to answering the main research question.

Time series analysis

The need to preserve essential information in the forecasts causes the time series to be split up into different aggregations. This causes the total number of time series on which is forecasted to be 165. These time series are analyzed before the forecasting models are implemented to forecast on them. This is done to identify any patterns or characteristics in the time series which can require data preprocessing or alternating the setup of the forecasting models. The time series are tested on stationarity which showed that some time series are non-stationary, which is handled accordingly. Moreover, the time series showed seasonality on the hour per day and day of the week, especially for time series with high mean number of incoming picking tasks. This is included in the setup of the models that have a seasonal component. Furthermore, features are engineered that represent this seasonality or that are believed to affect the number of incoming picking tasks.

Forecasting models

A literature review is performed to identify forecasting models for time series forecasting. This resulted in the selection of three statistical models: exponential smoothing (ES), autoregres-

sive integrated moving average (ARIMA), and Croston method, and two machine learning models: multilayer perceptron (MLP), and long short-term memory (LSTM), to be tested and compared to the benchmark in this thesis. Furthermore, the features engineered from the time series are included in the forecasting models that allow these features to be included as input.

Results

As mentioned before, exponential smoothing (ES), autoregressive integrated moving average (ARIMA), and Croston method, multilayer perceptron (MLP), and long short-term memory (LSTM) models are applied to the zero-inflated time series data. In Figure 1, the RMSLE, which is a performance metric to measure the forecasting error, for all the individual forecasting models can be seen. In this figure, MLP_wf is the MLP without features, MLP_wsf is the MLP with features, LSTM_wf is the LSTM without features, and LSTM_wsf is the LSTM with features.

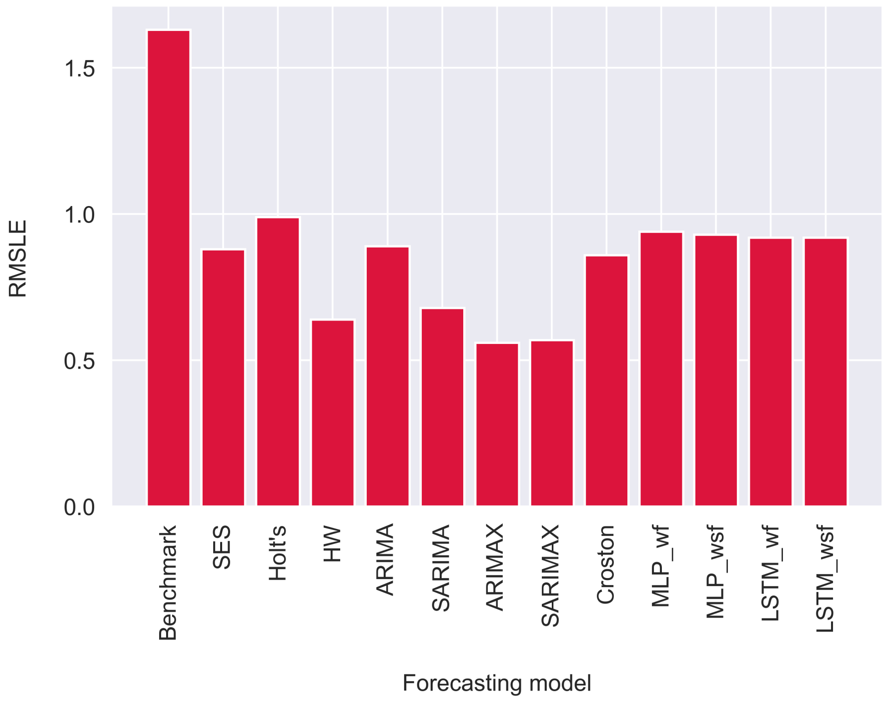


Figure 1: RMSLE of forecasting models

From the results, it can be observed that all the forecasting models that are tested outperform the benchmark. Furthermore, the ARIMAX model, which is an ARIMA model which includes features, performs best when the time series models are applied to all time series together. Moreover, more generally, it is observed that statistical time series models that include a seasonal component perform best on the time series data. After evaluating the results on the different aggregation level it is observed that for these levels, which consists of picking zones and departure buckets, there is not one specific model that performs best across all these picking zones or departure buckets. Therefore, with the aim to increase forecasting performance, the best-performing forecasting models per picking zone or departure bucket are used for the group of time series corresponding to that picking zone or departure bucket. This yielded a small improvement in the forecasting performance in comparison to the ARIMAX model being used

on all time series together.

Conclusions and recommendations

As can be concluded from the results, all forecasting models outperform the benchmark forecasting model that is currently in place in the warehouse of DHL. Specifically, ARIMAX is the best-performing model when models are applied to all time series together. A small improvement can be made when applying forecasting models on time series groups that are divided by the aggregation levels of the time series.

It is therefore recommended to DHL to evaluate the performance of the forecasting models in this thesis and test the forecasting model that is best fitting to their needs by evaluating, apart from the forecasting performance, the running time of the models. Furthermore, it has to be noted that there are limitations to this research. These limitations should be considered when deciding to test or even implement the forecasting models that are tested in this thesis. Additionally, it is recommended to DHL to continue exploring different models and approaches that may lead to an increase in forecasting performance.

Contents

List of Abbreviations

List of Figures

List of Tables

1 Introduction

In this Chapter, the motivation for potentially improving the accuracy of forecasting incoming picking tasks in the warehouse of DHL is given. After that, more information on DHL, the picking process in the warehouse, and how the forecasts are computed, is provided. Then, the problem statement and the research methodology are presented. Lastly, an overview which explains how the rest of the thesis is structured is provided.

1.1 Motivation

Effective capacity planning of order pickers, which is the planning of order pickers based on the workload in a warehouse, is essential for delivery services in current times. This is due to customers expecting short-term delivery of their products, which means that orders have to be picked efficiently. Additionally, in current times, the recruitment of order pickers is very hard due to the tight labor market. This means that the workload in the warehouse has to be divided among fewer order pickers. This is another reason why having an effective capacity planning is highly valuable. One of the big players in the package delivery market is DHL. DHL has many warehouses in which packages are stored and order pickers pick these packages for delivery to customers. This thesis is performed in one of the warehouses of DHL. In this warehouse, order pickers operate in different picking zones, in which different products are stored. The picking zones are divided based on product characteristics. In total, there are 11 different picking zones in the warehouse of DHL. The order pickers are distributed to pick orders in only one of the picking zones which can then be packaged and sent to the customer.

In the warehouse, around 15,000 packages are shipped per working day. This implies that sub-optimal capacity planning on the operational side can have large effects on the cost savings in absolute amounts of money for DHL. Consequently, having more order pickers on the floor than needed results in unnecessary costs. On the other hand, having too few order pickers on the floor when the workload is high can result in lower rates of on-time delivery of products. This means that the percentage of on-time delivery in the warehouse can decrease as a result of sub-optimal capacity planning. Furthermore, sub-optimal capacity planning can result in an additional drawback. This is observed in the warehouse of DHL where order pickers often experience (too) high workload due to bad planning of order pickers among picking zones which causes stress and has a negative effect on work satisfaction.

Currently, in the warehouse of DHL, forecasts for the future workload are made on the number of incoming picking tasks that are expected during the coming hours, per picking zone. These forecasts are however not yet used to base the capacity planning on. This means that the capacity planning in the warehouse of DHL is solely based on the workload that is known. DHL plans to, in the future, make the capacity planning of order pickers based both on the current workload, as well as the expected future workload. Including the future workload can result in more efficient capacity planning due to being able to better cope with changes in workload in the future. When the capacity planning is partially based on forecasts of expected workload, however, forecasting errors may arise which can lead to the capacity planning of order pickers being made on the wrong number of expected incoming picking tasks. This can lead to sub-optimal capacity planning of which the negative effects have been discussed before. Therefore, improving the forecast accuracy of the current forecasts, which means forecasting closer to the real number of incoming picking tasks coming in per picking zone, can lead to improved effectiveness of the capacity planning, when DHL implements the capacity planning on both current and future workload. This thesis aims to improve the accuracy of the current forecasts of the expected future workload in the warehouse of DHL, which will in the future be used to plan the capacity of order pickers.

1.2 Company description

DHL, which is part of the Deutsche Post DHL Group, is one of the largest shipping companies worldwide. In 2021, their revenue exceeded 81 billion euros. They have about 380,000 employees in more than 220 countries and provide global sustainable trade flows, connecting people and businesses all over the world. Furthermore, the Deutsche Post DHL Group makes a positive impact on the planet by making use of sustainable business practices and by making commitments to society and the environment. This comes forward in their aim to provide net-zero emission logistics by 2050. Within the portfolio of DHL are services that include, but are not limited to, national and international parcel delivery, e-commerce shipping and fulfillment solutions, international express, road, air, and ocean transport, and supply chain management solutions.

The warehouse of DHL, in which this research was conducted, provides warehousing and shipping of electronic devices in a Business-to-Business (B2B) environment. In this warehouse, everything from picking and packaging, to shipping the orders is done. This research focuses on the process of picking the orders in the warehouse. In Section 1.3, the picking process is discussed in more detail.

1.3 Picking process

In the warehouse of DHL, the picking process starts when an order is released from the planning department to the warehouse. This order is received by the order pickers who will then go into the warehouse to pick (a part) of that order, depending on if the order can be fully picked from the picking zone to which the order picker is assigned, or not. If the order cannot be fully picked in the zone of the order picker, another order picker in another zone also has to pick a part of that order. After that (part of) the order is picked, the items are brought to the packaging zone which further processes the order.

The order pickers in the warehouse of DHL work in specific zones which are divided based on characteristics of products. For example, fast-moving or small products are stored in picking zones located closer to the packaging zone and are often smaller in size. On the other hand, slower-moving or larger products are grouped in picking zones that are further away from the packaging zone but often bigger in size. This minimizes the total distance that order pickers have to cover as they are less likely to pick an item from the zones that are further away. An order picker works in one picking zone at a time but can be reassigned to another picking zone at any time of the day.

The capacity planning of the order pickers in the warehouse of DHL is done per picking zone. As mentioned before, order pickers can only be assigned to one zone. As the forecasts are planned to be used in the capacity planning, for which the picking zone needs to be known, it is essential to also include information on the picking zone in the forecasts. Switching between picking zones can happen at any time, but costs valuable time that could be used for picking. Furthermore, acting in this reactive way rather than planning capacity proactively, which is desired, results in, on the one hand, often switching too late to be able to handle an increase in workload in a picking zone. On the other hand, acting in this reactive way means sometimes having more order pickers than needed in a picking zone for the current and future workload, which leads to unnecessary costs of employment.

Apart from these zones, orders also have specific time windows, which are referred to as departure buckets, in which they need to be completed. Knowing when an order has to leave is vital information for the capacity planning as picking tasks are prioritized based on the time when the order, in which the picking tasks are, has to leave the warehouse. As DHL is planning to include the forecasts in the capacity planning, this information also has to be included in the forecasts on the picking tasks. More information on the aggregation levels of the forecasts that are made is given in Section 1.4.

1.4 Forecasting of picking tasks

Currently, DHL has a forecasting model in production, of which the technical details are provided later on in this thesis. This forecasting model has been implemented without any insights on how good the forecast performance is in comparison to other forecasting models. As mentioned before, having forecasting errors can lead to sub-optimal capacity planning, when forecasts are used to plan the capacity on. Therefore, DHL is interested in investigating and comparing the forecast performance of the current model to other forecasting models to test how well it performs, before basing the capacity planning on the forecasts of the current model. The forecasting model currently in production is the benchmark to which the performance of the other models implemented in this research are compared. This research seeks to improve the performance of the existing model, by implementing different time series forecasting techniques to the data of DHL. Time series forecasting models are used because the data of DHL can be classified as time series data. This is due to the fact that the data points are collected in a sequence, and in a constant time interval, in the past (George E. P. et al. (2016)). The forecasting of picking tasks is done on a short forecasting horizon, which consists of 2 days. Additionally, the forecasts of picking tasks made in the warehouse of DHL are not used in the weekends, which implies the forecasts are not used on Saturdays and Sundays. This is because the order pickers do not work on these days as there are almost no incoming picking tasks. Intuitively this is caused by the fact that this warehouse operates in a B2B environment, which causes customers to order during work hours and therefore picking tasks to mostly come in during working days (Monday until Friday) and during regular working hours (8 AM until 5 PM). It is however possible for a picking tasks to come in at any time and any day of the week.

As described in Section 1.3, information on in which picking zone the picking task falls is essential to preserve in the forecasts, as an order picker can only be assigned to one picking zone in the capacity planning. Consequently, for the forecasts to be useful to include in the capacity planning, the forecasts should be made per picking zone. Furthermore, for each order, the time at which the order needs to leave the warehouse is collected in the data. This information is also essential to preserve in the forecasts because when the tasks should be performed is largely dependent on when it has to be completed. It can be understood that a picking task that is part of an order that has to leave the warehouse within two hours has a higher priority than a task that is part of an order that has to leave in two days. Therefore, to deal with the prioritization of picking tasks well, information on the deadline for each picking task should be included in the forecasts. This implies that the time series of picking tasks on which is forecasted should be categorized in departure times. In the warehouse of DHL, to categorize the deadlines of picking tasks, so-called ”departure buckets” have been constructed. The deadline of a picking task is categorized within one of the departure buckets that span 4 hours of time. Both the picking zone and the departure bucket of picking tasks are present in the data. This implies that for each unique picking zone and departure bucket combination, the time series is individually analyzed and forecasted, in order to preserve this information in the forecasts. In reality, this amounts to 11 × 15 (picking zones × departure buckets) = 165 time series that are analyzed and forecasted on. In Figure 2, the two aggregation levels, of the picking zones and departure buckets are depicted.

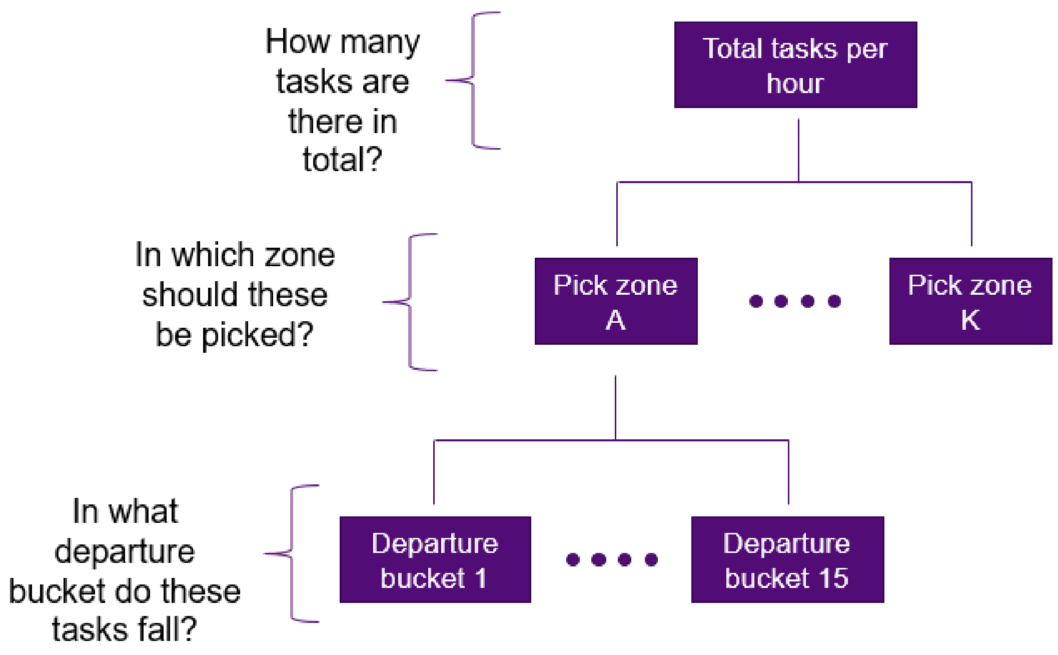

How many tasks are there in total?
Total tasks per hour
In which zone should these be picked?
Pick zone A
Pick zone K
In what departure bucket do these tasks fall?
Departure bucket 1
Departure bucket 15

Figure 2: Aggregation levels of picking tasks time series

In this research, the focus is on the forecasting of picking tasks in the warehouse of DHL. The actual capacity planning of order pickers that is discussed in this Section is out of scope. However, the context of the plans of DHL to include the forecasts in the capacity planning to make them more efficient, and the importance of capacity planning in warehouses, puts emphasis on the importance of possibly improving the accuracy of the forecasts for incoming picking tasks in the warehouse of DHL.

1.5 Problem statement

The goal of the thesis is to investigate what time series forecasting methods could improve the performance of picking task forecasting in the warehouse of DHL. The main research question of this thesis is:

How can picking tasks in the warehouse of DHL be accurately predicted by using time series forecasting?

In order to be able to answer the main research question provided above, research sub-questions have to be formulated. The insights and answers on these research sub-questions are the underlying fundament on which the main research question is answered. First of all, the way in which the dataset should be prepared to make it useful for training and testing the forecasting models is determined. After that, data analysis on the time series is performed to identify underlying characteristics of the time series. Insights from this analysis assist in making choices for the time series models that are decided to be implemented. Additionally, to evaluate the performance of the forecasting models, the performance metrics that best suit the case of DHL are determined. Lastly, given the results of the performance metrics, the best-performing individual forecast models are discussed.

All in all, the research sub-questions that are formulated to structure the thesis, and answer the main research question are:

1. How should the dataset be prepared to make forecasts on?
2. What characteristics can be attributed to the underlying time series on different aggregation levels and what does this imply for data transformations of time series?
3. Which time series models are available and which ones best suit the problem at hand?
4. Which performance evaluation metrics are relevant to be used?
5. What forecast models provide the best performance and how can the results on different aggregation levels of the forecasts be explained?

1.6 Methodology

In this section, the research design that is used to structure this thesis is discussed.

The research design can be seen in Figure 3. This research design was constructed based on the CRoss Industry Standard Process for Data Mining (CRISP-DM) framework which was first introduced by Chapman et al. (2000). In Figure 3, the colors indicate what part of the CRISP-DM framework relates to the research design constructed for this thesis.

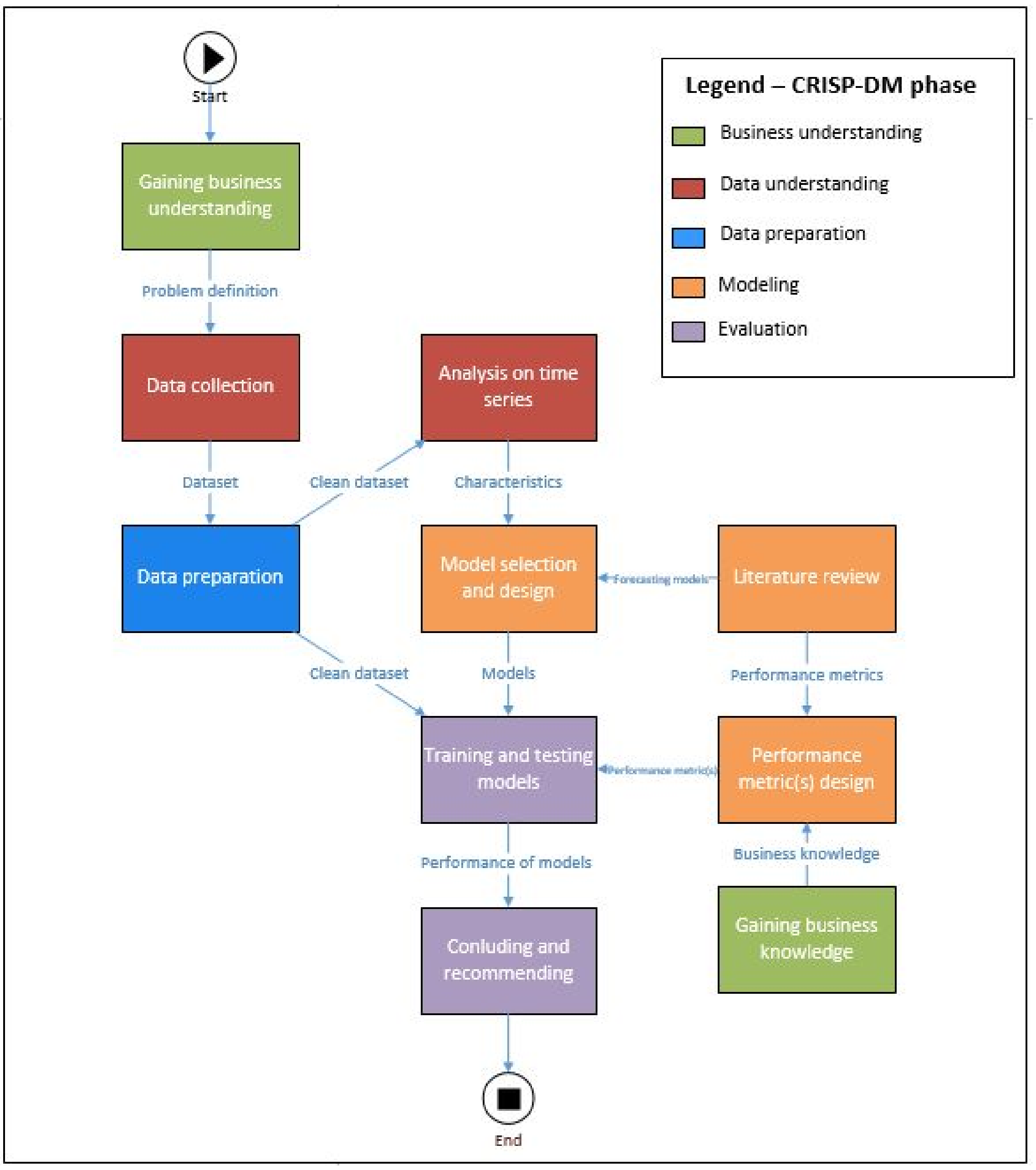


Figure 3: Research design

First of all, the business understanding phase of the CRISP-DM framework in the research design model of this thesis corresponds to gaining business understanding. This was essential throughout the project as constructing the problem statement, making modeling decisions, and interpreting the results to draw conclusions, were all based on specific domain knowledge that indicated what would be best to do in order to provide the most valuable results.

The next phase of the CRISP-DM framework, data understanding, consisted of two parts in this thesis. These parts were the data collection and analysis on the time series. The data collection phase ensured that the data on which the forecasting models would be tested, and the results would be generated, was collected from DHL. Furthermore, the analysis on the time series that was done provided necessary insights which were used to tailor the time series forecasting to the specific time series of incoming picking tasks and its properties. Moving on to the third phase of the CRISP-DM framework, data preparation, which is named the same in the research design constructed for the thesis. In this phase preparation, which involved cleaning as well as transforming the dataset, was done which was vital in getting a clean and usable dataset on which the forecasting models could be trained and tested.

The next phase of the CRISP-DM framework, the modeling phase, in the thesis consists of the literature review, as well as the forecasting model selection and design, and the performance metric design. The literature review provides the theoretical background on which the selection and design of forecasting models and performance metrics is made. This implies that based on the insights gained from the literature review, combined with the insights from the analysis of the time series, performed in the previous phase, the selection and design of the forecasting models and the performance metrics is done. Furthermore, apart from selecting the fitting forecasting models, the model selection and design phase also involves designing how the model is implemented. This phase therefore also includes, performing model-specific data transformations, implementing the feature engineering, and designing the training and testing process.

The last phase of the CRISP-DM framework that is used in this thesis is the evaluation phase. After the forecasting models had been selected and designed to fit the properties of the time series of incoming picking tasks, the models were trained and tested on the dataset. After implementing the different forecasting models in the training and testing process that is designed, results on the performance of these models were computed by making use of the performance metrics designed in the previous steps. After the results were obtained, conclusions and recommendations were given in the last sections of the thesis.

1.7 Scientific contribution

In the case of DHL, for the forecasts to be useful, it is needed to keep the relevant information about the picking zone and departure buckets in the data. Due to the information that is preserved, the forecasts are made on highly granular data which causes the data to be zero-inflated, and for multiple time series to be forecasted. Zero-inflated data implies that the data contains a high percentage of zero values. The combination of a high percentage of zero values, combined with scattered positive values might be hard to model for the forecasting models. The different time series also have different statistics in terms of number of picking tasks coming in. Some time series have a higher mean value of incoming picking tasks, with a lower amount of zero values, while other time series have lower mean with higher amount of zero values. Furthermore, in the case of DHL, the forecasts are made, for every hour, two days ahead, because short forecast horizons are most useful. This research, therefore, involves forecasting a large amount of time series, which are zero-inflated, on short time horizons. In most literature about time series forecasting, the data contains only one or a few time series, with (almost) no zero values but strictly positive values, which are forecasted over longer time horizons. Evaluating how the different well-known individual forecast models perform when forecasting on many different time series at the same time, with zero-inflated data, on short time horizons is a valuable scientific contribution of this thesis. The thesis provides insights into how well the forecasting models perform on the large amount of zero-inflated time series provided by DHL, including which forecasting model performs best. Resulting from this, conclusions about how well the time series forecasting models perform, and what forecasting model performs best, in the domain of forecasting picking tasks in warehouses, and other domains with the same zero-inflated data structure and large amount of time series, can be drawn.

1.8 Outline thesis

The rest of this thesis is structured as follows. In Chapter 2, the theoretical background of this thesis is provided. In this Chapter, the literature that is used to theoretically base this thesis on is discussed. Furthermore, this provides information on the available time series forecasting models and performance metrics used in literature. This information is used to answer sub-research questions 2 and 3 provided in Section 1.5. In Chapter 3, the input that is used to base the forecasts on, and the structure of the forecasts themselves are provided. In Chapter 4, the dataset that is used for the time series forecasting is presented, including all preparation that is performed. Furthermore, in this section, also data analysis is performed to gain an understanding of the properties of the time series. Chapter 4 forms the basis to answer sub-research questions 1 and 2. Then, in Chapter 5, the modeling and design of the forecasting models that are used is explained. This includes the data preprocessing per specific forecasting model, the training and testing process, and the performance metrics that are used. Furthermore, in this part of the thesis, the results of the forecasting performance of the different individual forecasting models are provided. All in all, this Chapter provides the information with which sub-research questions 3, 4, and 5 can be answered. After the results have been presented and discussed, in Chapter 6 the conclusions are given. This includes revisiting and answering the research questions, providing scientific contributions, giving recommendations to DHL, and possible limitations and future research of the thesis.

2 Theoretical background

In this Chapter, a literature review on time series forecasting is provided. The first part of the literature review goes into the analysis of time series data. After that, the main body of the literature review goes into the various forecasting models that are used in literature to predict time series data. Lastly, performance measurements are discussed. In this part, the characteristics of these metrics, including in which situations these metrics can best be used, is explained. In Chapter 2, only the high-level principles about the analysis, models, and performance metrics are provided. This implies that no detailed, mathematical, elements of any of these components are discussed. In the literature review that is performed as part of this master thesis, more detailed information on any of the components in this Chapter, including mathematical theories underlying the forecasting models, can be found (Braspenning (2022)).

A systematic literature review protocol was applied to find relevant literature on the topics of the literature review. The literature review that is performed as part of this master thesis is referred to for further details on the literature search process (Braspenning (2022)).

2.1 Time series data analysis

The first part of the literature review will dive deeper into the data analysis that can be performed on the time series. The goal of this data analysis is to identify characteristics underlying the time series. These characteristics can then be modeled and used to predict future values of picking tasks (Avishek and Pks (2017)).

2.1.1 Components of time series

In this section, the underlying components that make up a time series are given and further explained. There are four main components of which a time series can consist. These are:

- general trend
- seasonality
- cyclical movements
- unexpected variations

(Avishek and Pks (2017), Lazzeri (2020), Li et al. (2019)).

A general trend is an upwards or downwards trend over a period of time. General trends are often caused by fundamental shifts or systematic changes in a process. An example of a time series with general trend is seen in Figure 4. In this figure, over the long term, a general upward-sloping movement is seen. This is defined as an upward trend (Avishek and Pks (2017), Lazzeri (2020)).

In Figure 4, also seasonality can be observed. Seasonality is the repetitive and periodic fluctuations seen in the time series. Seasonality can be observed as the repetition of certain patterns corresponding to periods (Schaffer et al. (2021)). This implies seasonality comes back every period, season, quarter, etc. When trends are also present in the time series data, seasonality can be harder to identify as it is hard to distinguish what part of the variation belongs to the trend and what part belongs to the seasonality (Avishek and Pks (2017), Lazzeri (2020)). Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots can be used to quite easily spot seasonality (Li et al. (2019)). These plots will be discussed later in on. In Figure 4, apart from an upward trend, the seasonality is visualized in the yearly fluctuations that can be seen in the plot. This plot shows that the winter periods have higher CO2 concentrations than the summer periods.

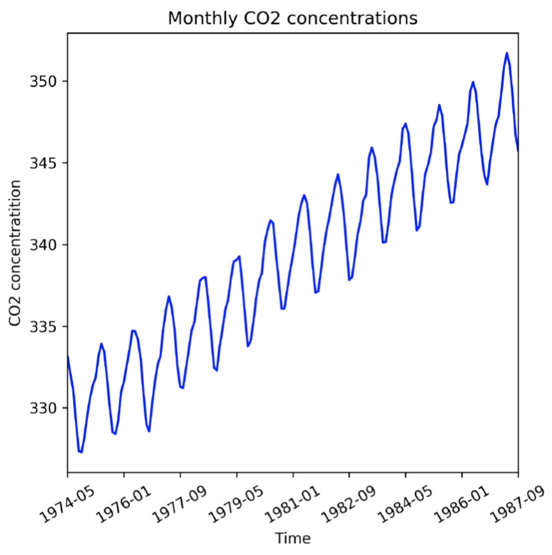


Figure 4: Trend and seasonality (Avishek and Pks (2017))

Another component that can cause variations in time series data is cyclical changes. Cyclical changes are comparable to seasonality but do not occur periodically. This implies that they do not have a fixed time interval in which they occur but the interval can spread over longer or shorter time periods. Furthermore, cyclical changes generally occur less frequently than seasonality which means they need time series that spans over larger amounts of time to be observed. A well-known example of cyclic change occurs in economics where there are macroeconomic periods of regression followed by periods of boom (Avishek and Pks (2017), Lazzeri (2020)).

All variation that cannot be attributed to a systematic dependency on the time index, and is random, is seen as unexpected variations. This unexpected variation is stochastic and cannot be captured in a mathematical model. Unexpected variations are either due to unknown knowledge about exploratory variables, which implies that it is not known that a variable has an effect on a certain time series, or random noise in the time series data (Avishek and Pks (2017), Lazzeri (2020)). Furthermore, time series without any seasonality, trends, or cyclical movements are also called time series with white noise, where white noise corresponds to the unexpected variance in the series (Hyndman and Athanasopoulos (2018), Schaffer et al. (2021)). When a time series only consists of unexpected variance, it is assumed to be stationary. The concept of stationary time series is discussed later on.

Types of plots that can help identify seasonality, as well as trends, are ACF and PACF plots. These plots show the autocorrelations and partial autocorrelations between the value at a given time and the so-called ”lagged” values in the time series. Just as normal correlation describes the relationship between two variables, autocorrelation (or partial autocorrelation) describes the relationships between the value at time y and y_{t-k} where the values at time y_{t-k} are the ”lagged” values (Hyndman and Athanasopoulos (2018), Schaffer et al. (2021)). Furthermore, in contrast to the autocorrelations in the ACF plot, the partial autocorrelation in the PACF plot controls for values of time series at all lags shorter than the lagged value (Schaffer et al. (2021)). In other words, partial autocorrelation provides the relationship between a value at a given time and a value of a lagged time while removing all the relationships between these values, or the correlation of the residuals. In Figure 5, an ACF and PACF plot can be seen.

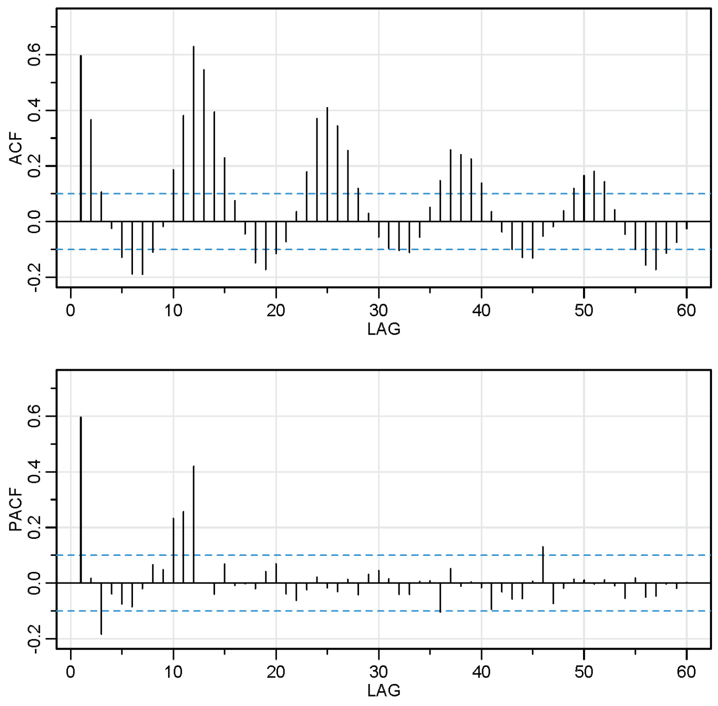


Figure 5: ACF and PACF plots

As can be seen most clearly in the ACF plot on top, there is a seasonal pattern present in the data. This can be seen because the correlation between the lagged values and the observed value displays a clear period pattern. Furthermore, when autocorrelation in the ACF tend to decrease, this can be an indicator of trend in the time series. Lastly, the dashed line indicates if the correlation is significantly different from zero (Hyndman and Athanasopoulos (2018)). The seasonal pattern that is observed in the ACF is not observed in the PACF plot. This is because the relationships between the lagged values that are compared are removed.

2.1.2 Stationary time series

Stationary time series are assumed to only consist of unexpected variance. An important assumption in statistical estimation theory is that to be able to analyze the data, the mean, variance, and autocorrelation of the data does not shift over time. If the data is non-stationary, these assumptions may not hold, and the results can be unreliable, which is why the time series have to be made stationary to be used for forecasts. Stationary requires mean, variance, and autocorrelation of the time series to not be dependent on the time of the observation. Furthermore, there can be no trend or seasonality in the data, mean and variance has to be constant over time, and autocorrelations need to have a constant structure (Avishek and Pks (2017), Schaffer et al. (2021)). Cyclical fluctuations are not included because they do not occur in fixed intervals and can therefore be explained by exogenous variables. A time series without predictable patterns is, generally, considered a stationary time series. All in all, first, the stationarity of time series has to be analyzed, and if concluded that they are not stationary, then data transformations can be applied.

Firstly, visualizations of time series can give an indication of whether the time series includes trends, seasonality, and if they are stationary or not. For example, the decomposition of time series or the ACF and PACF plots can provide information on the presence of trend or seasonality (Li et al. (2019)), the structure of the autocorrelations in the time series (Schaffer et al. (2021)), and if the time series is stationary or not (Zeroual et al. (2020), Schaffer et al. (2021)). Furthermore, statistical tests can be done to verify if a time series is stationary, and thus its mean, variance, and autocorrelations are not affected by trends or seasonality. Statistical tests such as the Augmented Dickey-Fuller (ADF) test or the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) can be done to detect stationarity. In short, for the ADF test, the null hypothesis is that the time series is non-stationary whereas the alternative hypothesis is the time series is stationary. This implies that for the time series analysis, it is desired that the ADF test (and thus the p-value) is significant as this would indicate a stationary time series. For the KPSS test, however, the null hypothesis is that the time series is stationary. This implies that for the time series analysis, it is desired that the KPSS test (and thus the p-value) is not significant as this would indicate a stationary time series (Atwan (2022)). According to Afriyie et al. (2020), when there is disagreement between the ADF and the KPSS tests, the KPSS test should be used, based on the performance of the tests.

When time series are non-stationary, different data transformations can be performed to make the time series stationary. Different transformation techniques can be used to transform the time series into a stationary time series. The two most commonly used ones are differencing and transforming. The idea of differencing is taking the difference between successive points of a time series ($\delta x_t = x_t - x_{t-1}$) which have constant mean and variance such that this time series can be treated as a stationary time series after transformation (Avishek and Pks (2017)). This means that a time series of $(x_1, x_2, x_3, ..., x_n)$ becomes $(x_2 - x_1, x_3 - x_2, ..., x_n - x_{n-1})$ for first-order differencing. Based on the order of the trend (linear or non-linear), higher order differencing might be needed. In practice, a higher order than two is almost never needed. Furthermore, if there is seasonality this can be made stationary with seasonal differencing ($x_t^{'} = x_t - x_{t-m}$) where m is the period of seasonality (Avishek and Pks (2017)). Furthermore, transforming relates to applying transformations to the variable that is displayed in the time series to test if it is possible to make it stationary. Examples of these transformations are logarithmic transformations or root transformations.

2.1.3 Multi-step-ahead time series forecasting

Forecasting on time series data is often done in a one-step-ahead approach. In some cases, however, forecasts which forecast multiple steps ahead are desired. This is known as multi-step-ahead time series forecasting and can be defined as an estimation on the future time series $\phi_{N+h}, (h = 1, 2, ...H)$, where H is an integer number higher than one, given the current and previous observation which are given by $\phi_t, (t = 1, 2, ..., N)$ (Bao et al. (2014)). The forecasts in multi-step time series forecasting can be made in different ways. The five strategies that are extracted from literature are: Direct (Dir), Recursive (Rec), Direct Recursive (DR), Multiple Input Multiple Output (MIMO), and a combination of Dir and MIMO, called Direct Multiple Output (DirMO) (Shu et al. (2022), Ahani et al. (2019)). In Dir strategy, for every estimation of a future value, ϕ_{N+h}, a new model is constructed. This means that for a forecasting horizon H periods ahead, H models should be constructed which independently make a prediction for every forecast horizon at a certain point in time. In Rec strategy, a model is built to perform one-step ahead forecasting. This model is then used to predict one step ahead, and the prediction is put back as input into the model to predict the following value until all desired steps of the forecast horizon are predicted. In comparison to the Rec strategy, the Dir strategy avoids the accumulation of prediction errors, as every forecast horizon has its own model. However, it can be computationally more costly, as more models have to be constructed (Ferreira and da Cunha (2020), Shu et al. (2022)). Moreover, a combination of Dir and Rec strategies, known as DR, can also be used. In DR, for every new forecast horizon that is forecasted at a certain point in time, a new model is constructed, just like in Dir. However, in contrast to Dir, the predicted values of the previous models are used as input into the new model, in DR, which is a feature of Rec strategy. This makes DR, a combination of Dir and Rec strategies (Shu et al. (2022)).

In addition to these strategies, MIMO, can also be applied to compute multi-step-ahead time series forecasts. In this method, all the values for ϕ_{N+h} for all H are predicted, as a vector of the predicted values is given as output. MIMO overcomes the limitations of both the direct and iterated strategies because it avoids the accumulation of prediction errors while only needing to construct one model. Furthermore, it also preserves the stochastic dependency which can characterize a forecasted time series. However, a limitation of MIMO is that this strategy is not implementable for all forecasting models, but only for ones that can output multiple values (Ferreira and da Cunha (2020)). Lastly, a combination of Dir and MIMO strategies, DirMO can be applied. In this strategy, a middle approach between Dir and MIMO is chosen. This implies that, when there is a H step-ahead forecasting task, this is decomposed into n multiple output forecasting tasks ($n = H/s$). Each with an output size of s ($s \in (1, 2, ...H)$) (Ahani et al. (2019)). Consequently, in the DirMO strategy, multiple different models can be constructed, like in Dir, with multiple output values, like in MIMO.

2.2 Time series forecasting models

To form an overview of the forecasting models that are present in literature, the literature of Wang et al. (2018), Hyndman and Athanasopoulos (2018), Gasparin et al. (2022), which all give an overview of different forecasting models were examined. From this literature, a division between statistical methods, and machine learning models can be observed. This division is depicted in Figure 6. Furthermore, hybrid models, that combine different models, are seen as a separate category of forecasting models. First, in Section 2.2 the statistical methods and machine learning techniques that can be applied to forecast time series are discussed. After that, in Section 2.3, information from literature about hybrid models in time series forecasting is given.

Statistical time series forecasting models

2.2.1 Exponential smoothing models

Exponential smoothing is one of the most well-known and has motivated some of the most successful forecasting techniques. Furthermore, it offers a robust technique that is widely used in many business applications (Kahraman and Akay (2022)). Exponential smoothing predicts future values by weighing past observations, which decay over time (Hyndman and Athanasopoulos (2018)). There are three different types of exponential smoothing, Simple Exponential Smoothing (SES), Holt's exponential smoothing (Holt's), and Holt's-Winters exponential smoothing (Holt's-Winters). First of all, SES is suited for time series that do not include any trend or seasonality. In SES This setting the smoothing parameter ensures decaying weights are assigned to the observations as they are further in the past.

To be able to also take into account trend and seasonality components, the Holt's linear trend method and Holt's-Winters method of exponential smoothing were constructed. In the Holt's method, a trend component is added to the SES method to be able to deal with an upward or downward trend. Additionally, this trend component can be taken as additive or damped which ensures that forecasts on longer time horizons are not over-forecasted.

In the Holt's-Winters approach, both trend and seasonality components are included to be able to capture both these components in the time series data. Just like in Holt's the trend components can be taken as no trend, additive trend, or damped trend. Furthermore, the seasonal component can be taken as additive or multiplicative. The additive seasonal component should be used when the seasonal variations are roughly constant. However, when the size of the seasonal component is proportional to the trend level of the forecast, the multiplicative method should be used.

2.2.2 Regression models

First of all, regression models can be used to predict the dependent variable. In regression models, predictor variables, also known as independent variables, are used to predict the dependent variable. In single linear regression, a single variable is used to predict the variable. However, often there are multiple variables that partially predict the dependent variable. In multiple linear regression, multiple predictors are formulated that predict the dependent variable. Furthermore, relationships between the predictor variables and the predicted variable may not be linear. When nonlinear relationships are present, nonlinear regression models are probably more suitable to model the relationship between the predictors and the predicted variable. Nonlinear regression models can simply be modeled by applying transformations to the variables in the regression model.

With regards to using regression models for forecasting, two ways in which data on the predictors can be used are possible. Ex-ante forecasts are predictions that are made with the data that is available in advance. This implies that, for the regression models to make predictions, also values of the predictors have to be predicted. This is due to the fact that the values of the predictors at time t, which is the time for which the predicted value is predicted, are not known as well. The other way in which the dependent variable can be predicted is when the value of the predictors at time t is already known. Logically, this type of prediction can not be done for future values in a model where the values of the predictors are not constant or predetermined. However, this type of prediction can provide more insights into the relationships between the predictors and the dependent variable. This latter method is called ex-post forecasting (Hyndman and Athanasopoulos (2018)).

2.2.3 Autoregressive and moving average models

In Autoregressive (AR) models, past values of the predicted variable are used in a linear regression model, to predict the value of the predicted variable at time t (Ivanovski et al. (2018)). The prediction is made by using a multiple regression of the lagged values of the predicted variable (Pena-Sanchez et al. (2018)). Alternatively to AR models, Moving Average (MA) models use the past error terms of the regression models instead of the past values of predictions, to predict the next forecast. Just like in AR models however, this is done in a regression model type manner where the past error terms are used as predictor variables (Hyndman and Athanasopoulos (2018), Ivanovski et al. (2018)). In this model, the error terms are equal to the white noise in the time series data (Ivanovski et al. (2018)). The term has previously been described in Section 2.1.

Autoregressive Integrated Moving Average (ARIMA) models are a combination of AR and MA models and use differencing if the time series on which it is used is non-stationary. This means that in ARIMA models, the prediction is made on both the lagged values and the error terms of the forecast. Furthermore, ARIMA models make predictions on stationary time series, which are differenced if needed. The concept of differencing has been discussed in Section 2.1.2. Furthermore, ARIMA models are also able to model a wide range of seasonal data. These models are called Seasonal Autoregressive Integrated Moving Average (SARIMA) models. In these models, there is a non-seasonal part just like in the ARIMA model, however, there is also a seasonal component added which takes into account the prediction made on both the lagged value and error of forecast of the last period and the lagged value and error of the forecast of the period a number of seasonal cycles ago. The number of seasonal cycles that are taken into account can be altered.

Lastly, exogenous variables can be included in both ARIMA, and SARIMA models. These models are called Autoregressive Integrated Moving Average with Exogenous Variables (ARIMAX) and Seasonal Autoregressive Integrated Moving Average with Exogenous Variables (SARIMAX), respectively (Wanishsakpong and Owusu (2020)). As can be seen above, in the ARIMA and SARIMA models, the lagged values and the forecast errors are combined. In ARIMAX, and SARIMAX, the exogenous variables are added to the models in a regression-like manner. This implies that these models are a combination of lagged values, forecasting errors, and exogenous, which makes them a combination of regression models, autoregressive models, and moving average models.

2.2.4 Croston method

The Croston method is a method that is specially tailored for dealing with zero-inflated data. The method combines a prediction on the next non-zero value, with a prediction on the time until the next arrival of a non-zero value to compute a forecast. Both these forecasts are made in a way that is similar to simple exponential smoothing. This implies that, for the next value of the non-zero occurrence, the past values of non-zero occurrences are used to base the forecast on. In the same way, to predict the time until the next non-zero arrival, the past times between non-zero arrivals are used. Furthermore, just like in SES, the smoothing parameter α should be specified. This smoothing parameter is used to determine the weights of the past values for the two forecasts that are computed. After the two forecasts on the value of the next non-zero occurrence and the time until the next non-zero occurrence are made, the ratio between these forecasts is the forecast that is computed for the next period by the Croston method. This implies that the prediction of the Croston method is the forecast of the value of the next non-zero occurrence divided by the forecast on the time until the next non-zero arrival (Hyndman and Athanasopoulos (2018)).

Machine learning time series forecasting models

2.2.5 Grey theory

Grey theory is best used in uncertain systems with limited or poor data. Furthermore, it can also work well in situations with limited sample size (Zhao et al. (2012), Zhao and Guo (2016)). It requires a limited amount of data to identify patterns and behaviors in unknown systems. Furthermore, grey theory has proven to work best on forecasting medium to long-term forecasts (Wang et al. (2018)). In practice, there are a lot of systems that are uncertain or have limited or poor data available to use for forecasting. This makes grey theory useful for a lot of applications (Liu et al. (2011)). A grey model can be described as $G(n, m)$, where n is the order of the differential equation, while m is the number of variables in the model (Kayacan et al. (2010)). In stochastic processes, grey theory introduces a so-called "grey variable" which varies in a certain range and space (Zhao and Guo (2016)). The $G(1, 1)$, which thus has first-order differential equations and one grey variable is most studied in research (Kayacan et al. (2010)).

2.2.6 Fuzzy time series

Fuzzy time series have been created to address the problem of dealing with vague, imprecise, or linguistic historical time series data. In fuzzy time series, fuzzy relational equations and approximate reasoning are used to make a forecast on time series data (Liu (2007)). To compute forecasts by using fuzzy time series forecasting methods, first, the intervals of the fuzzy model should be formulated. After that, the real value of the time series is fuzzified by making use of these intervals. The predicted fuzzy value is computed based on the rules and fuzzy relationships of the model. Lastly, the predicted fuzzy value is fuzzified to get a real value for the forecast at time t (Liu (2007)).

2.2.7 Support vector regression

Support Vector Regression (SVR) is a non-parametric method for creating regression functions. SVR are on average good at generalization performance which means that these models do not tend to overfit the data set. In other words, often, SVR tend to perform well on the test set (Wu and Akbarov (2011)). This is among the reasons why SVR models are found to be a viable contender among various time series modeling techniques (Bao et al. (2014)). In SVR, a kernel function maps the data points in a higher-dimensional space. Then, the SVR method tries to create a hyperplane on the mapping of the data into a feature space with the higher-dimensional space (Cankurt and SUBAŞI (2016)). When the SVR model can separate the data point in the higher dimensional space in a rather simple way, then the hyperplane can easily be created. When this is not the case, the problem becomes more complex and overfitting might occur.

2.2.8 Neural networks

Neural networks are models that are based on simple mathematical models of the brain. These models allow for complex non-linear relationships between the predicted variable and the predictors (Hyndman and Athanasopoulos (2018)). There are different types of Neural Network (NN) models that can be used. In the next section, the NN that are commonly used for time series forecasting, will be discussed. First, Feed-forward

Neural Networks (FNN), which only uses information in a forward-going manner, will be discussed. Afterward, the different kinds of Recurrent Neural Networks (RNN), will be explained. These kinds of NN can store memory which gives them the ability to also take into account information of previous inputs.

A FNN, which is also known as Multilayer Perceptron (MLP), consists of a network of neurons, and connections between neurons are organized in layers to transform input into output. FNN consist of an input layer, hidden layer(s), and an output layer. The number of hidden layers determines the depth of the FNN, while the number of neurons per layer defines the complexity per layer. In this network of neurons and connections, the output of each neuron is computed by the sum of the input into that neuron and the transformation that is done in that neuron by using a non-linear activation function. Furthermore, connections also can have weights that can strengthen or weaken the influence of the output of the previous neuron on the input of the next neuron. These weights are parameters of the model and are usually learned by using backpropagation (Gasparin et al. (2022)). NN with multiple hidden layers are known as deep neural networks, these will be discussed more elaborately later on. FNN can only have direct forward connections between layers, which means there is no recursive feedback as in RNN (Gasparin et al. (2022)), which is discussed in the next section.

As mentioned in the previous section, unlike FNN, RNN allows for previous outputs to also be used as inputs into the neurons (Lazzeri (2020)). In this section, first, a basic RNN, which is also known as Elman Recurrent Neural Network (ERNN), will be described. After this, Long Short-term Memory (LSTM) and Gated Recurrent Units (GRU), which are well-known extensions of the basic RNN will be discussed (Bianchi et al. (2017)). ERNN allows for weighted connections within the same hidden layer. In ERNN, the weighted connections in the same layer are only allowed in hidden layers, and thus not in the input or output layers. Because these loops are included, the ERNN can store information while also processing new input. This gives the ERNN the ability to store internal memory, which makes it possible for the model to identify temporal behavior. Among other applications, like natural language processing, this attribute can be very useful for making time series forecasts.

An extension of the simple RNN is LSTM. RNN suffers from vanishing gradients when solving problems that require learning longer temporal relationships. LSTM deals with the problem of vanishing gradient, which can occur in simple RNN, by having a memory cell structure. In LSTM, the same input and output as an ERNN is provided, but internally, the LSTM implements a gated system that controls the processing of this information. As LSTM can store internal memory unaltered for longer periods, by making use of this gated system, it can deal with these vanishing gradients (Gasparin et al. (2022)). In general LSTM makes use of the memory cell, also known as cell state. In this memory cell, information flows unchanged. However, information can be added or removed from this flow. This information adding or removing is regulated by the gates. These gates decide which information enters and is removed from the memory flow (Ahmed et al. (2022)). There are in total three gates in the LSTM. The forget gate, first decides what information should be removed from the previous cell state, which is the memory of the cell. Then, the input gate, decides how much of the new input should be added to the memory of the cell. Lastly, the output cell decides what part of the state in the cell should be provided as output. In this way, the LSTM only makes modifications to the existing information on the data, whereas, ERNN completely changes the existing data by applying a function.

Another RNN, namely GRU, is discussed. GRU makes use, like LSTM, of a gated structure that can add or remove information from the memory of the cell, also known as the state. There are two gates in the cell structure. First of all, there is an update cell, that deals with how the current information in the cell should be updated with the new input information. Furthermore, there is a reset gate, which can reset all the information in the memory of the cell, and makes the unit act as if the next input is the first input into the model. As there are only two gates, and there is no output gate in GRU, the full state is always exposed as output, because this is not controlled by a gate (Bianchi et al. (2017)). Because GRU uses two gates instead of three gates in LSTM, it is seen as the simplified version of LSTM as it has fewer gates and fewer training parameters. This simplification allows GRU to use less memory, and train faster, than LSTM (Gasparin et al. (2022)).

An NN with more than three layers, which implies multiple hidden layers as there is only one input and output layer, is known as a deep neural network. Deep neural networks are able to extract features directly from the data set without having to define these features by making use of these hidden layers. This implies that deep neural networks can be very effective for time series forecasting as they eliminate the need for performing feature engineering, data scaling, or making the data stationary. On the other hand, to be able to do this, deep neural networks need a lot of data to train, which means large datasets are desired, and running time can be long (Lazzeri (2020)).

2.2.9 Supervised machine learning models

The models discussed until now are all frequently used in time series forecasting in literature (Wang et al. (2018), Hyndman and Athanasopoulos (2018), Gasparin et al. (2022)). These models use time series sequence data, time series data features, or both, to predict the future values of the time series. However, it is also possible to extract (a lot of) features from time series data, as well as use the lagged values as features. Feature engineering on time series data can result in a data set in which there are a lot of features engineered per observation in the time series. Consequently, machine learning models that use features as input, rather than time series sequence data, can be used to make predictions on the future values of the observation of which the features are put in. When the features that come from the time series data, and are put into the models, are specified, the time series forecasting problem can be transformed into a supervised machine learning problem that has to predict the next value based on features of the current observation. This transformation gives room for a range of supervised machine-learning models to be implemented to predict these values. This approach is used less frequently in comparison to the models that are discussed until now but is an interesting area to investigate. In this next section, examples of supervised machine learning models that have been used in literature to predict on features that are engineered from time series data will be described. This implies that not all supervised learning models will be discussed, but only the ones that are used in combination with time series data in literature.

First of all, in the paper of Bou-Hamad and Jamali (2020) a Random Forest (RF) model is used to predict on features of time series. RF combines the output of randomly constructed trees to get better performance than when a single tree is constructed. In the case of RF regression, the ensemble learning of RF boosts the predictive performance by averaging the output (Bou-Hamad and Jamali (2020)). Furthermore, in literature, mostly types of Gradient Boosted Machines (GBM) are implemented to solve supervised learning problems for time series data. GBM are, just like RF, an ensemble-based regression method, that combines weak learners, which are very often decision trees, to arrive at an accurate model. The main difference between GBM and RF, is that in RF, the decision trees are randomly generated while in GBM, the trees are built one at a time and not random. GBM minimizes the inaccuracies of weak models by minimizing the loss function. This means that the GBM tries to reduce the error of the weak learners in each iteration to arrive at an accurate model. In GBM, the learning rate of each underlying decision tree determines the predictive performance. This implies that a higher learning rate of the underlying decision trees means fewer trees are needed to obtain the same predictive power of the ensemble model.

In the paper of Zhang et al. (2021), a version of GBM, a Extreme Gradient Boosting (XGBoost) machine learning model, is implemented on time series data. XGBoost is an implementation of gradient boosting, where weak classifiers are combined into a strong classifier in a linear way (Zhang et al. (2021)). Furthermore, compared to other gradient boosting techniques, XGBoost can find a strong classifier among weak classifiers, while also being good at dealing with missing value, avoiding overfitting, and reducing running time (Luo et al. (2021)). In the paper of Zhang et al. (2020), Light Gradient Boosting Machine (lightGBM), which is a gradient boosting technique just like XGBoost, has been used to forecast. Due to its quickness and relatively high performance, it is often used to solve regression, classification, and other machine learning tasks, and has also been used a lot in data competitions in recent years (Zhang et al. (2020)). Compared to XGBoost, lightGBM results in smaller and faster models.

2.3 Hybrid time series forecasting models

As the name suggests, hybrid models combine different forecasting models into one framework to come to a single prediction. This is different from ensemble models because ensemble models combine the predictions of multiple forecasting models to reduce the variability of the forecast. The goal of hybrid models is to combine the strengths of the forecasting models into one framework to overcome the limitations of the single models and enhance forecasting accuracy. What is often observed is that hybrid models work well in time series where there are both linear and non-linear components. As hybrid models often consist of statistical and machine learning methods to handle linearity and non-linearity, hybrid models perform better in these time series than other forecasting models that have difficulty predicting either the linear or non-linear components (Xu et al. (2019), Smyl (2020), Xiao et al. (2012)). A situation in which a hybrid model would work better than a single forecast model is for example when the level and seasonality of a time series are linear and can be predicted by the equations of a statistical model. The trend observed in the time series, however, is not linear and can therefore not be captured accurately by the equations of the statistical model. This can then be solved by implementing another forecasting model, for instance, a NN that can capture the non-linear behavior more accurately, to estimate the trend component of the time series. This results in a better forecast because all the components of the time series are captured accurately, whether these are linear or non-linear. Moreover, apart from using different models to predict the components of the time series, different models can also be used to capture both information from the individual time series, as well as from the whole dataset, which can enhance forecasting accuracy (Smyl (2020)).

A common procedure of the current hybrid models being implemented for time series data in literature is to first implement a statistical model to estimate the linear information in the time series, and then use the residuals to train a machine learning models that can deal with the non-linear information in the time series data, just like explained above. These results are then combined to obtain enhanced forecasting results compared to the single models that are implemented. In all these papers, the hybrid model that was proposed, outperformed the single models in a time series forecasting problem because it combines linear and non-linear information of the time series data (Xu et al. (2019), Smyl (2020), Xiao et al. (2012)).

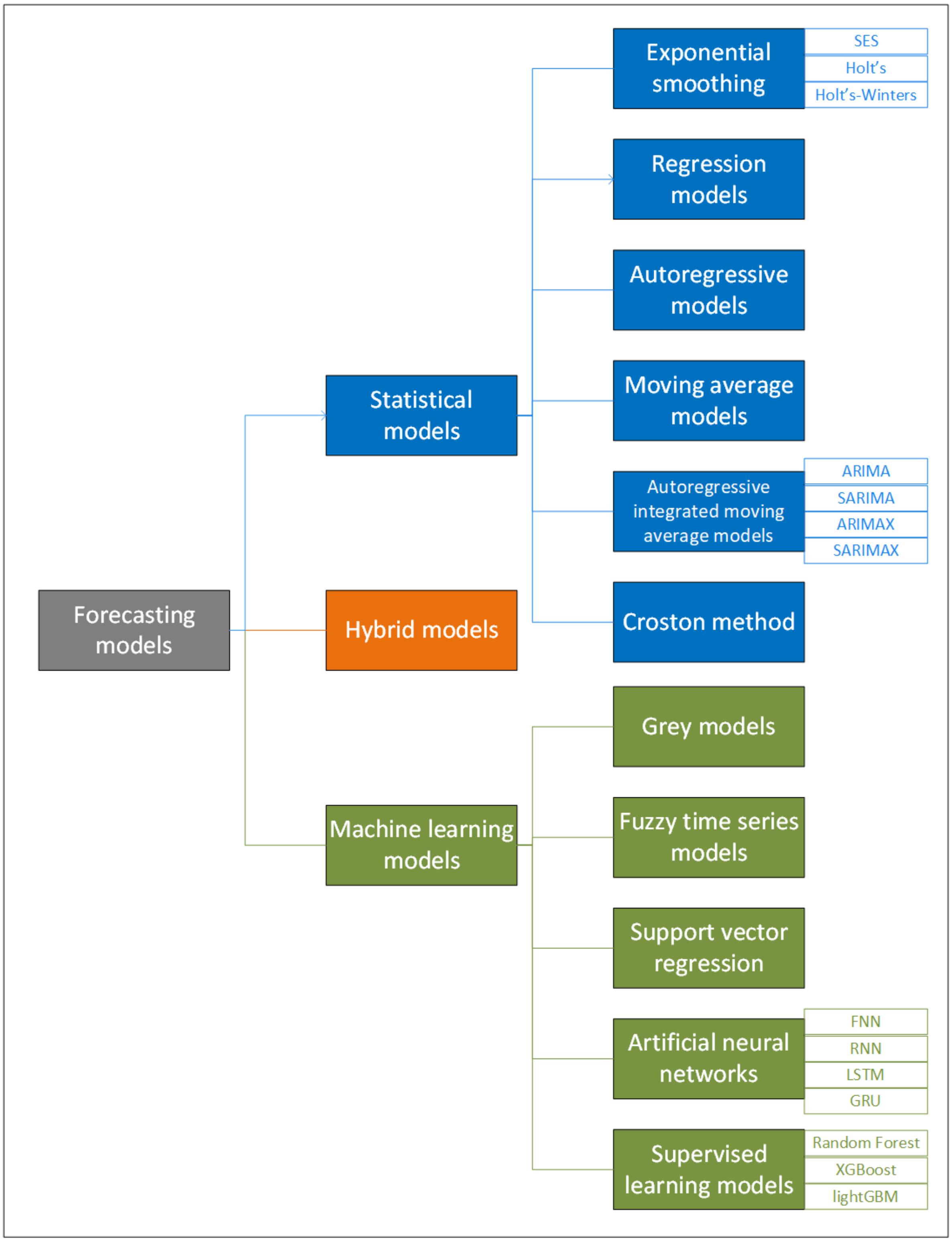


Figure 6: Overview of forecasting models discussed in the literature review

Table 1: Overview with input, strengths, and weaknesses of forecasting models (Wang et al. (2018))

Forecasting model	**Input**	**Strength(s)**	**Weakness(es)**
Exponential smoothing (ES)	The input consist of lagged values with corresponding weights; it can include trends or seasonality	Forecasts in a simple but effective way; more or less importance to more recent lagged values can easily be given	Lags behind the trend; it does not respond too well to dynamic environments
Regression models	Predictor variables that predict the dependent variable	Provides insights into factors that influence the dependent variable and their weight; it is easy to implement	Does consider the understandability of certain factor that influence the dependent variable; it cannot deal with periodicity
Autoregressive models (AR)	A linear combination of the lagged values of a variable	Flexible at handling a lot of different time series patterns	The time series data needs to be stationary
Moving average models (MA)	Past forecasting errors	Good at separating out random errors; less prone to moving up and down a lot due to temporal changes	Does not respond quick to very sudden changes in the time series
Autoregressive integrated moving average (ARIMA)	A combination of lagged values and past forecasting errors; can be extended with exogenous variables	Combines the AR and MA models to make use of both their inputs to improve performance	Requires stationary data; it cannot deal with non-linear data
Grey models	Small and incomplete information	Can make high accuracy predictions on small sample data	Does not take into account the intrinsic mechanism; cannot dynamically show system changes
Fuzzy time series	Can be numerical or linguistic and is computed by the fuzzy rules to get output	Good in situations of uncertainty or when qualitative knowledge is used	Lack of prediction formulas; cannot show the relationship between predictions and historical data
Support vector regression (SVR)	Lagged values and time series features	Can solve machine learning and non-linear problems with small samples; has high generalization performance; small amount of parameters to solve	Cannot deal well with missing data; hard to implement on big training samples
Neural networks	Lagged values and time series features	Can extract complex non-linear relationships from the data by itself; can handle multiple inputs (and outputs); can deal with long-term dependencies	Does not provide insights into the process of computing output or the reasoning; cannot work with small data samples
Supervised learning models	Time series features	Can work well in situations where there is little data on the independent time series; can take into account variables that might influence the time series other than lagged values	Might not be able to extract the sequence information from the time series

2.4 Performance measurements

In the paper of Hyndman and Koehler (2006), an adequate overview of accuracy measurements is given, and the measurements are discussed and compared with respect to time series forecasting. Consequently, indications on in which situation to best use which accuracy measurement are provided. Furthermore, the paper proposes a new method that is argued to also work well for measuring forecasting accuracy. In this section, the accuracy measurements that exist in the four different measurement categories are described, and their characteristics are given. These four different measurement categories are scale-dependent measures, measures based on percentage errors, measures based on relative errors, and relative measures. After that, the new measurement metric proposed by Hyndman and Koehler (2006), will be discussed.

First of all, the accuracy measurements which belong to the scale-dependent measures will be discussed. These include, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Median Absolute Error (MdAE). These are measurements whose scale depends on the scale of the data. Consequently, these measurements work well when comparing the accuracy of models which use the same data set, but should not be used when using different data sets with different scales (Hyndman and Koehler (2006)). MSE and RMSE have been very popular in the past. RMSE is often preferred out of these two, as this is on the same scale as the data. However, MSE and RMSE are more sensitive to outliers than MAE and MdAE which led to some argumentation against using MSE and RMSE for measuring forecasting accuracy (Hyndman and Koehler (2006)).

Moving on to the second category of accuracy measurements, measures based on percentage errors. The first four metrics in this category are Mean Absolute Percentage Error (MAPE), Median Absolute Percentage Error (MdAPE), Root Mean Squared Percentage Error (RMSPE), and Root Median Squared Percentage Error (RMdSPE). These measures have the big advantage that they are scale independent, unlike the metrics discussed above, and can thus be used to compare the accuracy of models that use different scales of data. However, these measures also have some disadvantages. The first disadvantage is that they are infinite or undefined when the observed value is zero. Furthermore, they have a very skewed distribution when the observed value is close to zero. Consequently, in data where the observed value is often zero, these measurements should not be used. Moreover, these metrics assume a meaningful zero which means they cannot be used when forecasting temperature on a Celsius or Fahrenheit scale (Hyndman and Koehler (2006)). Moreover, MAPE and MdAPE put heavier penalties on positive errors than on negative errors. To overcome this problem, two new metrics have been defined, Symmetric Mean Absolute Percentage Error (sMAPE), and Symmetric Median Absolute Percentage Error (sMdAPE). Smaller values for the observed value in the data set may have a less severe effect on the sMAPE and sMdAPE, but are still likely to include division by zero, as the forecasted value is also expected to be close to zero. Furthermore, these two metrics are formulated such that they can take negative values. Moreover, they are not as symmetric as their name suggests as they still give heavier penalties to forecasts that are low compared to forecasts that are high. Lastly, in general, measures based on percentage error are often skewed, which means they might require transformations to make them a stable accuracy measurement (Hyndman and Koehler (2006)).

Thirdly, measures based on relative errors are discussed. These measurements divide the error by another error that is computed by a standard forecasting method to make the error relative. Often, this method is a random walk, which is that the last observation is taken as the new forecast. The measures based on relative errors include Mean Relative Absolute Error (MRAE), Median Relative Absolute Error (MdRAE), and Geometric Mean Relative Absolute Error (GMRAE). A disadvantage of measures based on relative errors is that the relative error can be small or even zero. This means that to compute the relative error, the forecast error has to be divided by zero which is not possible, or by a very small number which might cause the measures to penalize the forecast error heavier than when working with bigger numbers (Hyndman and Koehler (2006)).

The last category of measures that is discussed in the paper of Hyndman and Koehler (2006), is the relative measures. This category does not involve a certain amount of metrics but is rather an idea that can be implemented to all the metrics described in the sections above. The idea of relative measures is that the metric that is computed is divided by the same metric computed for a benchmark method to provide a relative measure instead of an individual metric that is computed. This can be done for all the metrics described. This method can be used well in situations where comparisons to a benchmark method are desired.

The last two categories, measures based on relative errors, and relative measures, both try to remove the scale of the data. Both however have their disadvantages, as mentioned above. The paper of Hyndman and Koehler (2006), proposes a new performance metric, Mean Absolute Scaled Error (MASE), which gets rid of these disadvantages by scaling the error based on the in-sample MAE. This method makes use of a scaled error.

Lastly, another metric that was not mentioned in the paper of Hyndman and Koehler (2006), but can prove to be useful for evaluating time series forecasting models is Root Mean Squared Log Error (RMSLE). RMSLE can be especially useful in situations when it is desirable to avoid a huge effect on large differences in the predicted and actual values, in the case when these values are higher (Mir et al. (2022)). Furthermore, this metric provides performance in percentage errors but can still deal with zero values, unlike some of the other metrics above.

3 Forecast definition

In this Chapter, the forecasts that are made in this thesis, are defined. First, the input that is used to base the forecasts on, is discussed. After that, the forecasts that are computed for DHL are defined. This includes information on what the target variable that is being forecasted is, what the forecast horizon is, on what interval is forecasted, and how the information about the picking zone and departure bucket is preserved in the forecasts.

As mentioned in Section 1.4, DHL wants to forecast the number of incoming picking tasks, while preserving the information on the picking zones and departure buckets. To be able to do this, the dataset does not only contain data on the number of incoming picking tasks per hour but also information on the time at which the incoming task has to leave and the picking zone that it falls in. Consequently, the time series are divided per picking zone, and days to departure and departure time bucket combination. In Figure 7, an overview of the different levels on which the time series are divided can be seen. In this figure, the first level is all of the DHL data, the second level is the picking zones, the third level are the days to departure, and the fourth level is the departure time buckets.

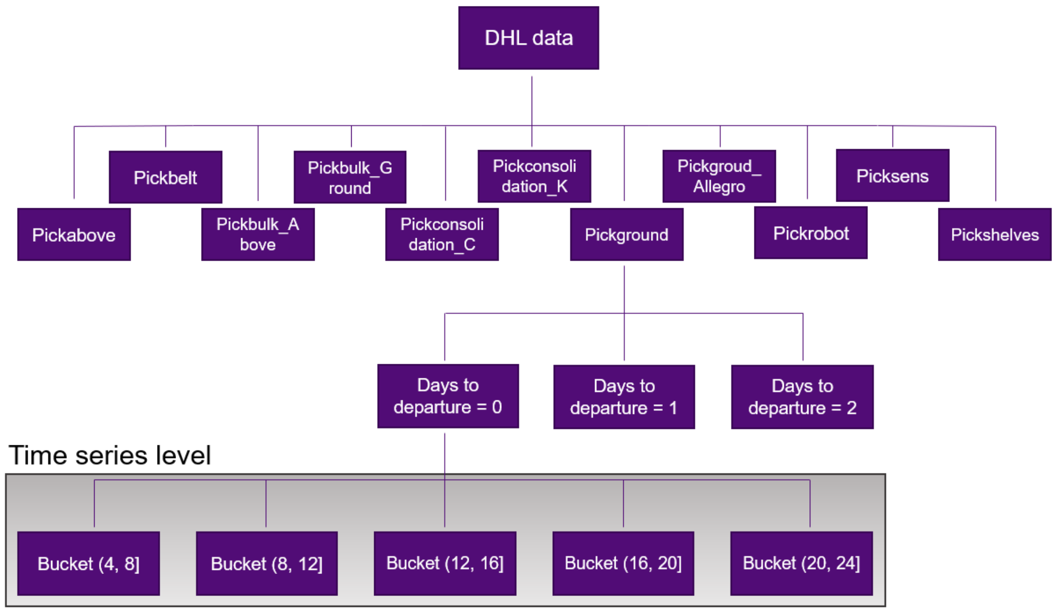


Figure 7: Different levels of time series

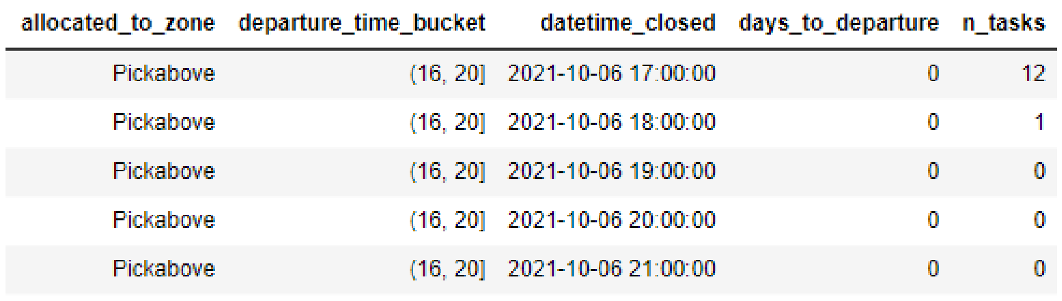
In this figure, it can be seen that the time series are used per picking zone, and per days to departure and departure time bucket combination. In this figure, the lowest level is the level on which the time series are used and the forecast is made. Doing this ensures that the information on the picking zones and departure buckets is preserved. Additionally, it is good to note that there are 11 picking zones, that the days to departure are either 0, 1, or 2, and that the departure buckets are divided in 4 hours time spans. More information about these aggregation levels is given later on in this Chapter.

3.1 Input of forecasts

A snapshot of the data that is used as input to forecast on can be seen in Figure 8. In the dataset of DHL, the number of picking tasks that come in are aggregated on one-hour intervals. This implies that every whole hour, the number of picking tasks that have come in during the past hour is added up and stored in the dataset.

In this snapshot, n_tasks corresponds to the number of picking tasks that have come in during the one-hour interval. This is the target variable that is forecasted in this thesis.

Furthermore, Datetime_closed corresponds to the hour at which the picking tasks have come in. As mentioned before, these are added up in hourly intervals. This implies that the hour in Datetime_closed is equal to the end hour of the one-hour interval at which the picking tasks are added up. For example, for the picking tasks that come in between 7 AM and 8 AM, these picking tasks are added up and stored with the hour 8 AM in the Datetime_closed feature, with the corresponding date. Allocated_to_zone is the picking zone in which the picking task falls. The combination of departure_time_bucket and days_to_departure indicates at which time range the picking task has to be completed. The time at which the picking task has to be completed depends on the time at which the order has to leave the warehouse, which depends on the time at which the order has to be delivered to the customer. The combination of the values for these features in the dataset defines the departure buckets that are used to categorize the time at which the picking tasks have to be completed. First of all, every departure bucket contains a days_to_departure value, as well as, a departure_time_bucket, which corresponds to a range of 4 hours during the day. Every picking task that comes in also has a departure time at which the task has to be completed. In which departure bucket a picking task is placed depends on both the time at which the picking task comes in and when it has to depart. If the picking tasks have to be completed on the same day as the picking task came in, then the days to departure variable is 0 if it has to leave the following day, it becomes 1, and if it has to leave 2 days later, it becomes 2. Furthermore, the time at which the picking tasks have to be completed is then placed in one of the time buckets that span 4 hours. The combination of the days to departure and the departure time bucket in which the picking tasks are placed makes the unique combination of the departure bucket in which the picking task is placed. For example, if the task that comes in has to be completed on the same day at 6 PM, then this task has days_to_departure value 0 and departure_time_bucket value (16, 20]. A task that has to be completed the day after it comes in at 11 AM has days_to_departure value 1 and departure_time_bucket value (8, 12].



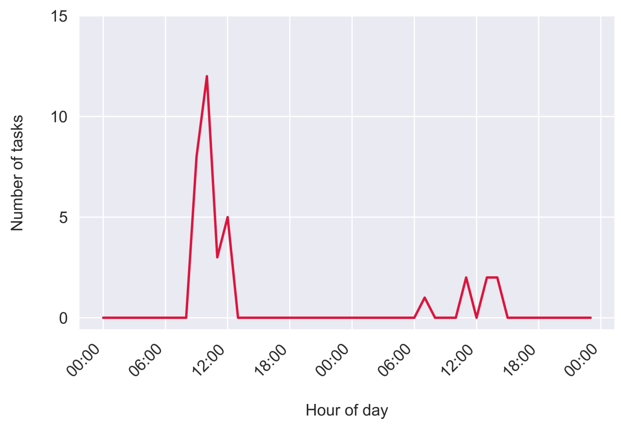
allocated_to_zone	departure_time_bucket	datetime_closed	days_to_departure	n_tasks
Pickabove	(16, 20]	2021-10-06 17:00:00	0	12
Pickabove	(16, 20]	2021-10-06 18:00:00	0	1
Pickabove	(16, 20]	2021-10-06 19:00:00	0	0
Pickabove	(16, 20]	2021-10-06 20:00:00	0	0
Pickabove	(16, 20]	2021-10-06 21:00:00	0	0

Figure 8: Snapshot of dataset

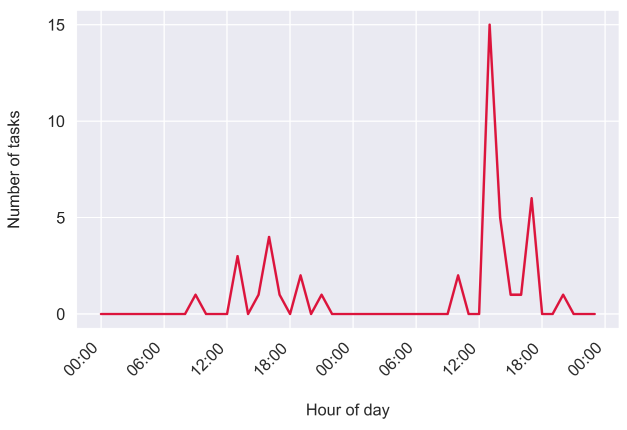
The way in which the departure bucket work can more clearly be seen in Figure 8. In this snapshot of the dataset it can be seen that on 2021-10-06, at 5 PM, in picking zone ”Pickabove”, there are 12 tasks that have to be completed on that same day between 4 PM and 8 PM. Furthermore, it can be seen that on 2021-10-06, at 9 PM, in picking zone ”Pickabove”, there are no tasks that have to be completed on that same day between 4 PM and 8 PM. In the dataset there are no tasks for which the last hour of the hour interval at which the task has come in, is equal to or later than the last hour of the departure_time_bucket with days_to_departure is 0 (has to be completed on the same day). This is the case because it is not possible for a task to be completed at the same time, or earlier than the picking task has come into the warehouse. Consequently, for these hours, where the departure buckets are earlier than these hours, the number of picking tasks is equal to 0.

The departure time buckets that exist in the dataset are (4, 8], (8, 12], (12, 16], (16, 20], (20, 24], which correspond to the time periods each day from 4 AM till 8 AM, 8 AM till 12 PM, 12 PM till 4 PM, 4 PM till 8 PM, and 8 PM till 12 AM. Furthermore, the days to departure can range from 0 until 2 which implies that tasks that have to be completed in two days or less are included in the dataset. The tasks that fall in the departure time bucket (0, 4], or that have days to departure longer than 2 days are not included in the dataset as there are almost no picking tasks that have to leave between 12 AM and 4 PM or that have longer than 2 days to be completed which why these are not interesting to analyze and forecast on for DHL.

The information on the departure time and the picking zone is preserved by dividing the different departure buckets and picking zone combinations into individual time series. As mentioned in Section 1.4, this amounts to 165 time series that need to be analyzed and forecasted on. The input on which the forecasts are made are the past values of the number of incoming picking tasks, per departure bucket, and per picking zone. The input consists of all the past values of incoming picking tasks, per departure bucket, per picking zone, that are available until the point in time where the forecasts for the incoming picking tasks are made from. In Figure 9, two snapshots of the time series of past values of incoming picking tasks for a specific departure bucket, picking zone combination can be seen.



(a) (0, (12, 16]), Pickabove



(b) (1, (12, 16]), Pickabove

Figure 9: Snapshots of time series

In Figure 9, the plot (0, (12, 16]) represents a snapshot of two days from the time series with days to departure 0 (it has to leave on the same day as the picking task has come in), departure time bucket (12, 16] (it has to leave between 12 PM and 4 PM), in the picking zone called "Pickabove". The other plot, (1, (12, 16]), represents a snapshot of two days from the time series with days to departure 1 (it has to leave on the next day with respect to the day at which the picking task has come in), and departure time bucket (12, 16] (it has to leave between 12 PM and 4 PM), in the picking zone called "Pickabove". What can be concluded from these plots is, just like mentioned before, it is not possible for a task to be completed at the same time, or earlier than the picking task has come into the warehouse. As you can see from the plot with days to departure 0 and departure time bucket (12, 16], there are no picking tasks that come in on or after 4 PM. This is not possible because there cannot be tasks coming in at or after 4 PM that have to leave before or at 4 PM. You can see that on the second day, there are picking tasks coming in before 4 PM that have to leave between 12 PM and 4 PM on the same day. This implies that, as mentioned before, when days to departure is 0, this corresponds to the tasks having to leave in the respective departure bucket on the same day. On the other hand, for the plot with days to departure 1 and departure time bucket (12, 16], there are picking tasks that come in on, or after 4 PM. This is possible because this days to departure and departure time bucket combination means that the picking tasks have to be completed between 12 PM and 4 PM on the next day, with respect to the day that the picking task has come in. Therefore, for this time series, it is possible to have picking tasks coming in after 4 PM because these picking tasks have to be done on the next day between 12 PM and 4 PM. Furthermore, what can be concluded from the plots is that virtually all the picking tasks come in during working hours, between 8 AM and 5 PM. The distribution of incoming picking tasks for hours of the day and days of the week is extensively discussed in Section 4.4.

3.2 Forecasts

As mentioned in Section 3.1, the number of incoming picking tasks in the warehouse is the target variable that is forecasted. The forecasts that are made predict the number of incoming picking tasks, per departure bucket, per zone, on a forecast horizon of 2 days. Furthermore, the number of picking tasks that come in is forecasted per hour. Therefore, the forecast for one departure bucket and picking zone combination, from one point in time, consists of 48 values, which correspond to the 48 hours that are forecasted ahead.

As discussed in Section 3.1, the departure buckets are defined with respect to when the picking task comes into the warehouse. For the forecasts, this implies that the departure bucket is defined with respect to the time for which the forecast is made, and not the time from which is forecasted. To further clarify this, a few examples are given. Suppose that the number of incoming picking tasks is forecasted from 1 PM on a Monday, 48 hours ahead. For the number of picking tasks predicted for 4 PM on a Monday, with days to departure equal to 0 and departure time bucket (8, 12], in a certain picking zone, this implies that the departure bucket corresponds to tasks that have to be completed between 8 AM and 12 PM on Monday for which the forecast is made. This value, when forecasted correctly, should be 0, as there cannot be any tasks that arrive at 4 PM but have to be completed between 8 AM and 12 PM. As the forecast is made from 1 PM on a Monday, 48 hours ahead, also values on Tuesday and Wednesday are forecasted. Furthermore, when still forecasting from 1 PM on a Monday, for the number of picking tasks predicted for 4 PM on that Monday, with days to departure being 1 and departure time bucket being (8, 12], in a certain picking zone, this implies that the departure bucket corresponds to tasks that have to be completed between 8 AM and 12 PM on the next day, which is Tuesday. This forecast can be higher than 0 as it is possible for picking tasks to come in at 4 PM on a Monday and be completed between 8 AM and 12 PM on the next day, which is Tuesday.

However, as the forecasts are made 48 hours ahead, when the forecasts are made from 1 PM on a Monday, this implies that also values for the number of incoming picking tasks are predicted for Tuesday and Wednesday. When predicting the number of picking tasks for 2 PM on a Tuesday, with days to departure 0 and departure time bucket (8, 12], in a certain picking zone, this implies that the departure bucket corresponds to tasks that have to be completed between 8 AM and 12 PM on the day for which the forecast is made, which is on Tuesday. Furthermore, when predicting the number of picking tasks for 2 PM on a Tuesday, with days to departure 1 and departure time bucket (8, 12], in a certain picking zone, this implies that the departure bucket corresponds to tasks that have to be completed between 8 AM and 12 PM on the next day with respect to the day for which the forecast is made. This implies that this forecast corresponds to the number of picking tasks coming in at 2 PM on a Tuesday, which have to be completed between 8 AM and 12 PM on a Wednesday. As mentioned before, this implies that the departure bucket, which consists of a combination of the departure time bucket and the days to departure, is defined with respect to the time for which the forecast is made, and not for the point in time from which is forecasted.

As mentioned in Section 1.4, there are 11 picking zones and 15 departure buckets in which a picking task can fall. This implies that a total of 165 ($11 * 15$) individual time series are forecasted on, as every picking zone and departure bucket combination is an individual forecast. Furthermore, the forecasts consist of 48 values, corresponding to the 48 hours that are forecasted. Consequently, for each point in time that is forecasted from, a total of 7920 ($165 * 48$) values have to be forecasted to provide a forecast spanning 2 days for every picking zone and departure bucket combination.

4 Data preparation and understanding

In Chapter 4, the data preparation that is done on the initial dataset is discussed. Furthermore, the dataset is analyzed to provide insights on the characteristics of the data that is forecasted on. In more detail, this implies that first the initial dataset and the preparation are discussed. After this, the descriptive statistics of the dataset are provided. Moreover, stationarity and seasonality in the time series data are analyzed. Lastly, the feature engineering that is performed to extract features from the time series data, including how this is used, is explained.

4.1 Dataset and preparation

A dataset including the picking tasks that have arrived in each hourly interval is provided. In this dataset, per picking task, the hour at which the picking tasks have come in is included. Furthermore, for every picking task, data on the time at which it has to be completed (in whole hours), and the picking zone of the warehouse in which the picking task has to be executed, is available.

The initial dataset should be prepared in order to get a dataset on which the forecasts are made. First of all, the data before 1 July 2020 is removed. This is done because DHL indicated that the data before this date was of low quality. Furthermore, tasks that have a negative time to be completed, or more than 2 days time to be completed are also removed from the dataset. Here, time to be completed is the difference between the time the picking tasks come in and the time it has to be completed. A negative time to completion is not possible for DHL to process and is therefore removed from the dataset. This only occurs in approximately 30 rows, which is very minimal compared to the around 2.5 million rows in the dataset. The cause of this is that the departure time is wrongly inserted in these cases. Moreover, the tasks that have more than 2 days to be completed are also removed from the forecasting. This is because these tasks make up a small portion of all the picking tasks. Around 15% of all the picking tasks have days to departure longer than 2 days. DHL wants to focus on forecasting the bulk of the incoming picking tasks which are the tasks that have 2 days or less to be completed. This is done to prevent including too many time series that have a higher proportion of zero incoming tasks per hour, as the number of incoming picking tasks gets lower when the time to completion rises, which is discussed in Section 4.2. Consequently, choosing a cutoff point allows focusing on forecasting the biggest part of the incoming picking tasks as accurately as possible, without giving too much importance to a small portion of the data in which the non-zero values are very sparse.

In the dataset of DHL, the picking zones are aggregations of different areas of the warehouse of DHL. In the initial dataset, the labels of these individual areas are given. These areas are aggregated into the 11 picking zones that are described earlier. This is done to limit the number of time series on which has to be forecasted while retaining the needed information on picking zones for DHL. As the order pickers are also assigned to these 11 different picking zones, for the capacity planning, the needed information is retained in the forecasts that are made on these aggregated picking zones. Later on in this Section, the effect of this aggregation on data loss is discussed. Another aggregation is performed on departure time that is included in the dataset. The departure time is aggregated into departure buckets that span 4 hours. With this operation, the number of time series is reduced which improves the understandability of the forecasts, while retaining the needed information level for DHL. Furthermore, in this process, the departure time bucket with departure times between midnight and 4 AM is removed because there are virtually no tasks that fall in these buckets. Like the reason for the cutoff point in the completion time, this departure time bucket is not included to focus on the time series in which most picking tasks arrive. The introduction of the departure buckets results in the transformation of some features in the dataset. The features of the resulting dataset can be seen in Table 2.

Table 2: Features in initial dataset

Feature	**Description**	**Format**	**Example**
datetime_closed	The date and last whole hour of the one-hour interval in which the picking tasks that have come in, are counted.	Datetime	14/01/2021 01:00:00
days_to_departure	Value that indicates which day the task has to be completed in regard to when it comes in (0 = today, 1 = tomorrow, 2 = in two days)	Integer	0, 1, or 2
departure_time_bucket	4-hour time span in which the task has to be completed	String	(4, 8]
allocated_to_zone	The picking zone of the warehouse in which the picking task has to be executed.	String	Pickabove
n_tasks	The number of picking tasks that have come in, in the corresponding hourly interval, for this departure time, and picking zone.	Integer	5

The dataset of DHL only includes an observation if the number of incoming tasks is not zero. This implies that, in the hour intervals, per zone, per departure time, where no picking task came in, there is no observation in the data. In time series data, every interval on which the data is collected, which is hourly in the case of DHL, should be included in the data in order to have a complete time series. This is why, for every hour, zone, and departure bucket combination, where there is no observation, an observation with zero number of incoming tasks is imputed.

Lastly, when aggregating the zones in one of the previous preparation steps, some observations are mapped to the labels for picking zones "Other" and "-". This indicates that these individual areas are not part of any of the aggregated picking zones in the warehouse according to the aggregation mapping that is made at that moment in time. This is likely caused by these individual areas being new in the warehouse, and therefore not being mapped to the aggregated picking zones at that moment in time. Because the forecasts are made on the aggregated picking zones for DHL, and these individual areas are not yet part of an aggregated picking zone, these individual areas are removed from the data. Furthermore, as can be seen in Table 3, the removal of these areas results in a data loss of 0.01% which indicates that the areas that are not yet included in the picking zone aggregations have a very small portion of the incoming picking tasks in the dataset. In Table 3, the preparation steps, including what effect each of the steps has on the number of observations in the dataset, are provided. Furthermore, between brackets, the number of observations with respect to the initial dataset is given.

Table 3: Preparation steps and effect on number of observations in dataset

Preparation step	**Remaining observations**
Initial dataset	1,626,039 (100%)
Filtering on data after 1st of July 2020	1,428,300 (87.8%)
Remove tasks with negative or >2 completion time	1,160,311 (71.4%)
Mapping of zones and removing zones ”Other” and ”-”	1,159,833 (71.3%)
Data loss with respect to the original dataset	28.7%
Introduction of departure buckets	377,925 (23.2%)
Impute observations with 0 tasks	2,628,210 (161.6%)

As can be seen in Table 3, due to the preparation steps, 28.7% of the initial data in the dataset is lost. After the preparation steps that result in a loss of data, the data loss is given in Table 3. Underneath the data loss percentage, the last two steps, that result in the final dataset that is going to be used in this thesis, are provided. The reduction of observations in the dataset due to the introduction of the departure buckets is not included in the data loss calculation. This operation is not included because the data is aggregated, which means that no observations are ”lost” but multiple observations are aggregated into one observation. Furthermore, due to the aggregation and imputation of zero values to complete the dataset, the dataset that is used for the time series forecasting is bigger than the initial dataset. This is because of the large portion of zero values that has to be imputed to complete the time series.

As explained in Section 1.4, it is important that the information on the departure bucket and the picking zone of the picking tasks is conserved in the time series forecasts. Consequently, the number of picking tasks per hour is forecasted per zone and per departure bucket. This implies that the forecasts are made on every specific combination of picking zones and departure bucket. After the preparation, there are 11 picking zones, and 15 departure buckets in total, which results in 11×15 (picking zones $\times$ departure buckets) $= 165$ time series that are forecasted.

4.2 Descriptive statistics

In Table 4, the descriptive statistics for the whole dataset of incoming number of picking tasks, per hour, per picking zone, and departure bucket, are given. Apart from the statistics for the whole dataset, the descriptive statistics per picking zone are also provided. In this table, the number of observations, mean, median, standard deviation, minimum value, first quartile, second quartile, third quartile, maximum value, skewness, and kurtosis, are given. All these statistics are computed for the incoming number of picking tasks, per hour.

Table 4: Descriptive statistics of number of picking tasks for whole dataset, and per zone

Zone	**Count**	**Mean**	**Med**	**STD**	**Min**	**25%**	**50%**	**75%**	**Max**	**% 0**	**Skew**	**Kurt**
Overall	2628210	2.03	0	11.83	0	0	0	0	1033	85.62	14.93	406.29
Pickabove	238965	0.82	0	7.80	0	0	0	0	895	88.78	42.68	2991.55
Pickbelt	238950	4.88	0	18.71	0	0	0	0	750	77.32	7.46	86.20
Pickbulk_Above	238815	0.43	0	2.58	0	0	0	0	480	90.70	38.97	5310.47
Pickbulk_Ground	238830	0.38	0	2.05	0	0	0	0	140	91.24	13.74	409.35
Pickconsolidation_C	239025	3.84	0	18.14	0	0	0	0	1033	82.92	10.21	175.18
Pickconsolidation_K	238830	0.60	0	2.88	0	0	0	0	124	89.36	9.42	142.80
Pickground	238995	4.69	0	19.42	0	0	0	0	974	78.17	8.71	141.34
Pickground_Allegro	238995	3.63	0	17.51	0	0	0	0	848	81.43	12.16	249.23
Pickrobot	238950	1.34	0	5.79	0	0	0	0	416	86.40	9.40	231.72
Picksens	238830	0.51	0	2.91	0	0	0	0	312	90.12	21.34	1209.83
Pickshelves	239025	1.14	0	4.62	0	0	0	0	215	85.39	6.96	80.27

From Table 4, it can be observed that there are big differences in the mean number of incoming picking tasks per hour, for the different zones. Furthermore, the zones with higher mean number of incoming picking tasks per hour, also have higher standard deviation. This implies that for the zones with higher mean number of incoming picking tasks per hour, these values are more spread out, or on average further away from the mean, than the values in the time series with picking zones that have lower mean number of incoming picking tasks per hour.

From the third quartile statistics in Table 4, it can be observed that there are a lot of observations that have zero incoming picking tasks. The third quartile is 0 for the whole dataset as well as for all the separate zones. This implies that, in the whole dataset, and in every zone, less than 25% has a value for the incoming picking tasks that is higher than 0. In more detail, as can be seen in Table 4, in the whole dataset 85.62 % of the values are zero. Furthermore, also the percentage of zero values per zone can be seen in the % 0 column.

Moreover, the high values for both the skewness and kurtosis on the whole dataset, as well as per zone, indicate that the distributions are heavily skewed to the right, and have heavy tails. The fact that the distribution is highly skewed to the right is because the number of tasks cannot be negative and is zero for most of the observations, with the rest of the values all being higher than zero (or to the right when plotting). Furthermore, the high kurtosis indicates that the distributions have heavy tails. The maximum value of every distribution indicates that, although there are a lot of zero values in the data, there are also some high values that are present in each of the distributions, for the whole dataset, and per zone.

It can be observed that the four zones, Pickbelt, Pickconsolidation_C, Pickground, and Pickground_Allegro have higher mean incoming picking tasks than the other 7 picking zones. Additionally, the zones with higher mean number of incoming picking tasks also generally have a lower percentage of zero values in the time series, as can be seen in Table 4. In Figure 12, the mean number of incoming picking tasks per hour, per zone is visualized. The mean number of picking tasks per hour, per zone is computed by aggregating the data per picking zone. In this figure, it can also clearly be observed that the four picking zones, Pickbelt, Pickconsolidation_C, Pickground, and Pickground_Allegro have significantly higher mean number of incoming picking tasks per hour than the other 7 zones. These four zones are highlighted in red.

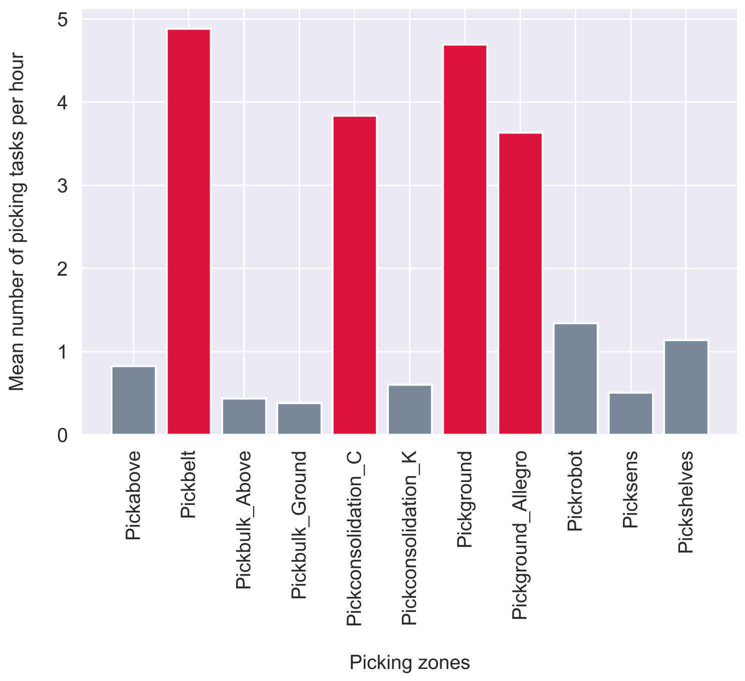


Figure 10: Mean number of picking tasks per hour, per picking zone

Next to analyzing the mean number of picking tasks per hour on aggregation level of picking zones, the differences in mean number of picking tasks per hour on the other aggregation level, the departure buckets, are also analyzed. In Figure 11, the mean picking tasks per hour, per departure bucket are visualized. The departure time buckets on the x-axis represent buckets that span over 4 hours. As described in Section 4.1, only tasks that have to leave within two days after coming into the warehouse are included in the data. Consequently, on the x-axis in Figure 11, the departure buckets in which the tasks in the data fall are included.

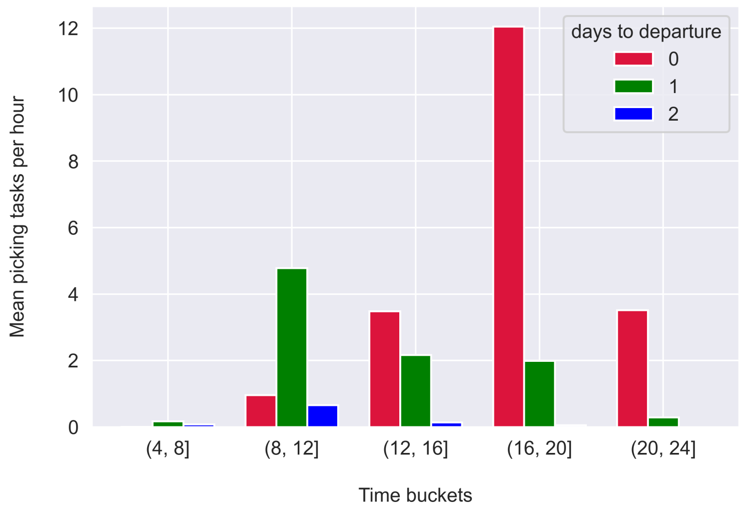


Figure 11: Mean picking tasks per hour, per departure time bucket

From Figure 11, it can be seen that the time series with the highest mean number of picking tasks fall into the departure time bucket of (16, 20] with 0 days to departure, which implies most tasks that come in, have to be done between 4 PM and 8 PM on the same day. What else can be deducted from the figure, is that almost all picking tasks have to be done either on the day they arrive or between 8 AM and 8 PM the following day. The fact that from 8 AM till 8 PM the following day contains quite some picking tasks is very likely because these are the picking tasks that come in late afternoon or in the evening and have to leave the next day in the morning or afternoon, which makes them get days to departure is equal to 1. Moreover, it can be seen that between 4 AM and 8 AM, almost no tasks have to be completed, especially when days to departure is equal to 0. This makes sense as it is very unlikely that a picking task will come in before 8 AM, and has to leave at or before 8 AM again (especially when recalling that DHL serves B2B). Lastly, it can also be seen that the probability of a task having to be done on the same day is the highest, and this probability goes down as the days go on. This implies that the number of tasks that come in goes down as the completion time of these incoming tasks increases. As discussed in Section 3.1, only the picking tasks that come in, and have to be completed within 2 days are analyzed and forecasted. In other words, only picking tasks that fall in the buckets that are formed, are forecasted. The reason for this is that there is only a very small amount of picking tasks that come in and have longer than 2 days to be completed. This is in line with the conclusions drawn from Figure 11, as the number of tasks that come in which have to be completed in 2 days is already very small, and this amount only goes down further as the completion time increases.

4.3 Time series stationarity

As explained in Section 2.1.2, stationary time series requires the mean, variance, and autocorrelation of the time series to not be dependent on the time of the observation. An assumption of ARIMA models, is that the time series should be stationary in order to provide reliable results (Lewinson (2020)). Consequently, to prevent unreliable results in the ARIMA models, possible non-stationarity in the time series should be handled correctly. Exponential Smoothing (ES) and Croston method are suitable for non-stationary data

(Lewinson (2020)). In this section, first, an assessment of the stationarity of the time series is made. After that, how the non-stationary time series are handled is discussed.

In Section 2.1.2, different methods for reviewing if time series are stationary or not, are discussed. One of these methods is inspecting the ACF and PACF plots for every time series to see if it is stationary or not. However, for the number of time series in the case of DHL this is a very time-consuming task and is therefore not executed. Alternatively, a statistical test is used to examine if the time series are stationary or not. As described in Section 2.1.2, according to Afriyie et al. (2020), when there is disagreement between the ADF and the KPSS tests, the KPSS test should be used, because this test performs better. Consequently, to assess whether or not the time series is stationary, the KPSS test is used.

In Table 5, the results of the KPSS test for each of the 165 time series can be seen. As can be deducted from this table, for the KPSS test, some time series have a significant p-value, a p-value lower than 0.05, which for this test indicates that the time series is non-stationary. Most of the time series that are indicated to be non-stationary by the KPSS test are time series with departure buckets that have days to departure of 0 or 1, or are in one of the four picking zones that have higher mean picking tasks per hour. This is potentially caused by the fact that these time series also have a lower number of zero values which means that the hourly and weekday seasonality discussed before is more apparent by the non-zero values in these time series. This causes the KPSS test to identify the time series as non-stationary.

Table 5: P-values of KPSS test for every time series

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	0.1	0.1	0.1	**0.02**	0.06	0.1	0.1	0.1	0.1	0.1	0.1	0.1	**0.02**	0.1	0.1
Pickbelt	**0.01**	**0.01**	**0.01**	**0.02**	**0.01**	**0.02**	**0.01**	0.1	**0.01**	**0.02**	**0.01**	0.1	0.1	0.09	0.1
Pickbulk_Above	0.1	**0.01**	**0.01**	**0.01**	**0.01**	0.1	**0.01**	0.09	**0.02**	**0.01**	0.1	0.1	0.1	0.1	0.1
Pickbulk_Ground	0.1	**0.01**	0.08	**0.01**	**0.01**	0.1	0.1	**0.01**	**0.01**	**0.01**	**0.02**	0.1	0.06	0.1	0.1
Pickconsolidation_C	**0.02**	**0.01**	**0.01**	**0.01**	**0.02**	**0.02**	0.1	0.1	**0.01**	0.1	**0.01**	0.1	**0.01**	**0.02**	0.1
Pickconsolidation_K	0.1	**0.01**	**0.01**	0.1	**0.02**	0.1	**0.01**	**0.01**	**0.02**	0.1	**0.02**	0.1	0.1	0.06	0.1
Pickground	0.1	0.08	**0.01**	**0.01**	**0.01**	**0.02**	**0.01**	**0.01**	**0.01**	0.1	**0.01**	**0.01**	**0.02**	0.1	0.1
Pickground_Allegro	0.1	**0.01**	**0.01**	0.1	**0.01**	0.1	**0.02**	**0.01**	**0.01**	0.06	0.1	0.1	0.1	0.1	0.1
Pickrobot	0.1	**0.01**	**0.01**	**0.01**	0.1	0.1	0.1	0.1	**0.01**	0.1	0.1	**0.01**	**0.02**	0.1	0.1
Picksens	**0.01**	**0.01**	0.1	**0.01**	0.1	0.1	0.1	0.09	**0.01**	0.1	0.1	0.1	**0.02**	0.08	0.1
Pickshelves	0.08	**0.01**	0.1	**0.02**	0.1	0.1	0.1	0.07	0.1	0.1	0.1	0.1	**0.01**	0.1	0.1

As described in Section 2.1.2, differencing can be used to make these time series stationary. The implementation of differencing only has to be applied to the ARIMA models as this is the only type of model that assumes stationarity of the time series (Lewinson (2020)). Differencing is applied to the time series which are non-stationary, according to the KPSS test, to attempt to make these time series stationary as well. After applying first-order differencing to the time series that were non-stationary before differencing, the KPSS test was done again to evaluate if the first-order differencing was sufficient to make these time series stationary. This showed that the time series that were non-stationary all became stationary after first-order differencing. In Appendix A, the p-values of the KPSS test, after the non-stationary time series are first-order differenced, can be seen. As can be observed from this table, this operation makes all the time series stationary, as all p-values are higher than 0.05. Consequently, setting the differencing parameter (d) in the ARIMA models to perform first-order differencing when the time series is non-stationary originally, and to no differencing when the time series is already stationary originally, proves to be sufficient to handle non-stationarity in the time series data for the ARIMA models. More information on how this is implemented in the ARIMA models is provided in Section 5.7.

4.4 Seasonality

After analyzing the time series data on seasonal patterns the following two insights were extracted. First of all, the mean number of picking tasks per hour was plotted for every hour of the day to investigate whether or not there is a pattern in incoming picking tasks per hour of the day. In Figure 12, the mean number of picking tasks per hour, for each hour of the day, for the whole dataset, are plotted.

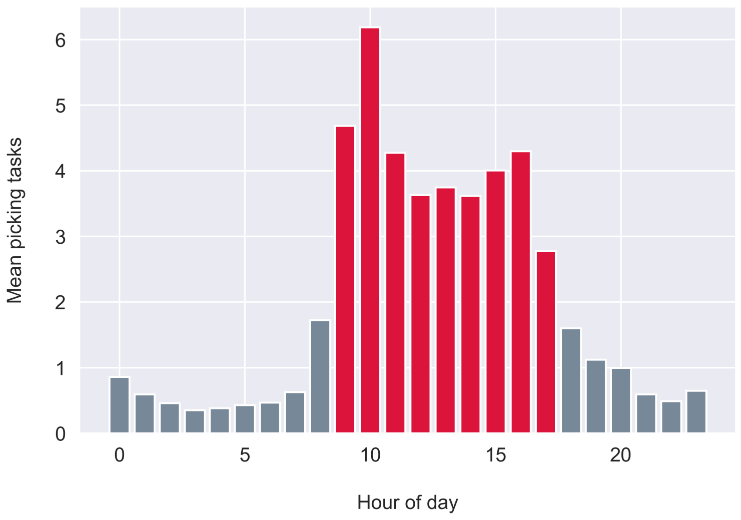


Figure 12: Mean number of picking tasks per hour of the day

As can be seen in Figure 12, there are significant differences in incoming picking tasks per hour of the day. In the figure, the bars for work hours, between 8 AM and 5 PM, are colored in red. As can be deducted from Figure 12, most picking tasks come in during these work hours, between 8 AM and 5 PM, especially in the morning. Consequently, this repetitive pattern implies hourly seasonality in a day. This can be explained by the fact that the warehouse of DHL serves B2B customers, which causes the customer to mostly order between work hours, and therefore the picking tasks for these orders to come in during working hours, as is also described in Section 1.4. However, as can be seen in Figure 12, not all picking tasks come in during working hours. Customers can order at any time which also makes it possible that picking tasks arrive at the warehouse of DHL outside of working hours.

Furthermore, in the data analysis, also the mean number of picking tasks per hour, for each day of the week was plotted. In Figure 13, the mean number of picking tasks per hour, for each day of the week, for the whole dataset, are plotted.

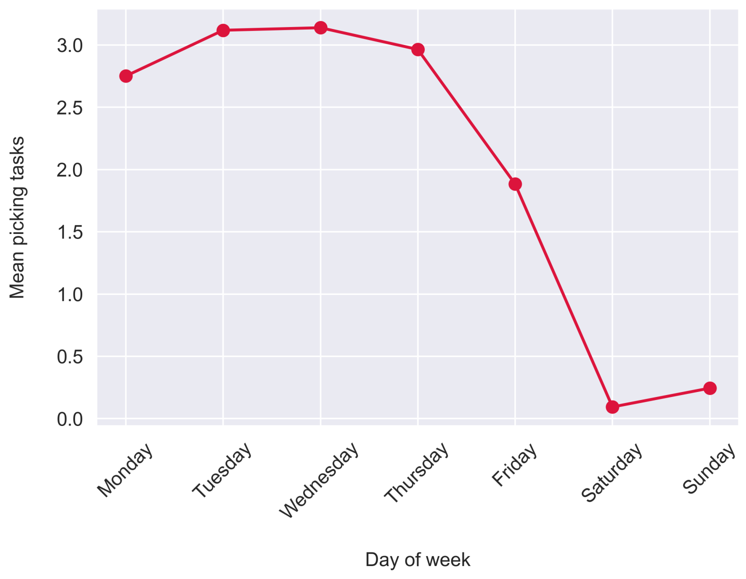


Figure 13: Mean number of picking tasks per hour, per day of the week

As can be seen in Figure 13, there are differences in the mean number of picking tasks per weekday. What can be extracted from the figure is that virtually all picking tasks come in during working days, which are Monday until Friday. Furthermore, the number of incoming picking tasks already begins to decrease on Friday, after which it goes to almost no incoming picking tasks in the weekend. This daily seasonality in the week, just like the hourly seasonality, is likely caused by the warehouse of DHL serving in a B2B environment, which makes the bulk of the picking tasks come in during working days. As can be seen in the Figure 13 however, there is still a very small amount of picking tasks that arrive during weekends. Furthermore, as a lot of businesses are less active on Fridays, this can explain the decrease in incoming picking tasks on Fridays in comparison to the other working days.

In Figure 14, the two figures above are combined in one plot. In Figure 14, the mean number of picking tasks is plotted for every hour of the week, for the whole dataset. This implies that the plot starts on Monday at 12 AM and ends on Sunday at 11 PM. Furthermore, the x-axis is divided into 24-hour periods so that the different days can be separated.

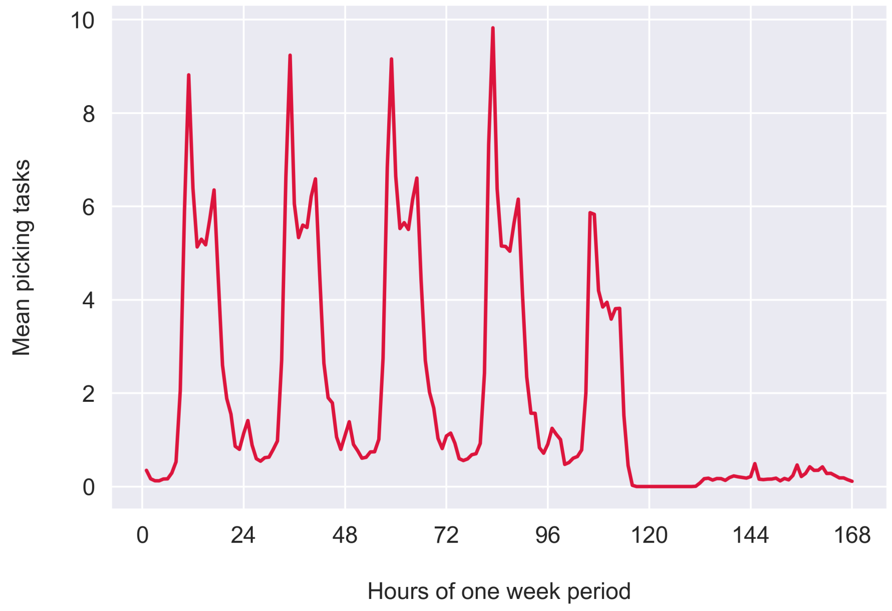


Figure 14: Mean number of picking tasks for every hour during the whole week

From Figure 14, it can be concluded that, as was also extracted from the other two plots, the mean number of picking tasks per hour is highest in working hours and working days. Additionally, from this plot, the mean number of picking tasks for every hour and weekday combination can be observed. This clearly shows that the combination of hours which are both in working hours and in working days result in the highest mean number of picking tasks per hour, and not the hours which are in working hours or in working days. Hours that are in working hours, but in weekends, or that are during working days, but not during working hours, result in significantly lower mean number of picking tasks coming in. Consequently, from this section it can be concluded that the mean number of tasks per hour coming in are highest during work hours, which is between 8 AM and 5 PM, on working days, especially from Monday until Thursday.

The behavior that is seen in Figure 14, is non-stationary due to the seasonal pattern. However, many of the time series underlying this plot are stationary as can be seen in Table 5. This is because Figure 14 is an aggregation of all the individual time series, which consist of stationary and non-stationary time series. It is possible that many of the time series are stationary as individual time series, but the aggregated time series is non-stationary. The main reason for this is that the time series with higher mean number of incoming picking tasks have a bigger impact on this figure, as these time series include a lot more picking tasks. As mentioned before, the time series with higher mean number of incoming picking tasks are more often indicated to be non-stationary by the KPSS test due to the seasonality being displayed by the non-zero values in these time series. This combined with the fact that these time series have a much bigger impact on the plot in Figure 14 than time series with low mean number of incoming picking tasks, makes this plot show seasonal behavior and thus be non-stationary. After reviewing many different individual time series, it was also concluded that the seasonal pattern seen in Figure 14 is far more often present in time series with higher mean number of incoming picking tasks. Therefore, the plot in Figure 14 is more representative of time series with higher mean number of picking tasks than for time series with lower mean number of picking tasks.

4.5 Feature engineering

Feature engineering is applied to extract additional variables from the dataset with the aim to improve the forecasting performance. These additional variables in the input of the forecasting models can enhance performance because the variables that are engineered are believed to influence the number of picking tasks coming in. In Table 6, the features that are engineered from the time series are provided. Furthermore, in this table, the features are described. Furthermore, the variable type, minimum, maximum, mean, and Pearson's correlation coefficient with respect to the variable that is forecasted, of the feature are given.

Table 6: Features engineering in dataset

Feature	**Description**	**Type**	**Min**	**Max**	**Mean**	**Corr**
Weekday	The weekday of the week represented by an integer.	Integer	0	6	3	-0.089
Hour	The hour of the observation.	Integer	0	23	11.50	0.024
Weekday_hour _mean	Mean number of tasks for that weekday and hour combination.	Float	0	535	30.01	0.767
Holiday	Binary variable indicating if the day of the observation is a holiday or not. This includes national holidays of the Netherlands, Germany, Great Britain, France, Italy, and Spain.	Binary	0	1	0.09	-0.010
Holiday_school	Binary variable indication if the day of the observation is a holiday for the schools in the south of the Netherlands or not.	Binary	0	1	0.26	-0.022

First of all, the first three features, Weekday, Hour, and Weekday_hour _mean are engineered because, as shown in Section 4.4, the hour of the day, weekday, and especially the combination between these two have an influence on the number of picking tasks coming in. The correlation of the hour variable with the number of picking tasks does not provide too much information because the relationship between the hour of the day and the number of picking tasks coming in is not linear. As discussed in Section 4.4, there are more picking tasks coming in during working hours, however, there is no linear relationship as it is not necessarily the case that the latter or earlier the hour in the day, the higher or lower the number of incoming picking tasks. The fact that the working hours show higher number of picking tasks is not shown by the correlation but is the reason for engineering this feature. Furthermore, the correlation of the weekday variable with the number of picking tasks is negative as the days later in the week have lower number of incoming picking tasks than the weekdays earlier in the week. This correlation is however not that strong which could be caused by the fact that it is the combination between weekday and hour of the day that causes the higher number of incoming picking tasks. This is shown in the Weekday_hour _mean variable which contains the combination of these variables. This variable shows a strong correlation with the number of incoming picking tasks. Including these features in the input of the forecasting models is believed to be able to enhance forecasting performance as the models have more information on variables that influence the forecasted variable.

Furthermore, the last two features in Table 6, are related to holidays. As can be seen in the table, the correlations of these features with the forecasted variable are very low. Nevertheless, these features are added to investigate whether or not these could help enhance forecasting performance. The intuition behind engineering these features comes from the fact that DHL serves customers in a B2B environment. Consequently, the bulk of the picking tasks come in during work hours on working days, when the businesses are operational. However, the behavior of the bulk of picking tasks coming in during work hours on working days might be influenced when the businesses are not operational during holidays. To capture this behavior, these two features, capturing the holidays during the year, are included in the feature engineering. Testing the forecasting models with and without using these features as input into the models provides the opportunity to assess whether or not adding these features enhances forecasting performance in the warehouse of DHL. This provides an indication of the potential of using features that are engineered from the time series data to forecast picking tasks that are coming in.

The first two and last two features in Table 6 are engineered from the timestamp that is available as datetime_closed in the dataset (this variable is explained in Table 2). These features, which can directly be engineered from the timestamp data, are also known as datetime features. Moreover, the third feature in Table 6, Weekday_hour _mean, is an expanding window feature. Expanding window statistics compute the statistics based on all the information that is available until that point in time. This means as we move further in time, more data on these statistics becomes available. For the weekday_hour_mean variable, this implies that the mean is calculated on all data that is available until that point in time.

All the features that are engineered, which are shown in Table 6, are used in the forecasting models that allow for features as input. More information on how this is implemented in the forecasting models is discussed per individual model in Chapter 5.

5 Forecast models

In Chapter 5, the implementation of the forecasting models, and the results of the forecasts that the models produce, are discussed. First, the explanation on selecting forecasting models for implementation is given. Then, the performance metrics that are used to evaluate the quality of the forecasts are given. In the hyperparameter tuning and testing section, information on the setup of the hyperparameter tuning and testing is provided. This includes the way in which the time series data is used, the types of hyperparameter tuning that are used for the specific models, and how the forecasts are made. In the sections after that, the implementation forecasting models are discussed and the results of the forecasts made by the models are presented. Lastly, the performance of the individual forecasting models is discussed and the forecast models are implemented per zone or per departure bucket, instead of for all the time series together, with the aim to enhance forecasting performance.

5.1 Model selection

In Chapter 2, the forecasting models that are used in literature are discussed. This resulted in Figure 6, in which all the forecasting models that were extracted from literature are shown. From these forecasting models, a selection on which models are implemented is made. Forecasting models from the statistical forecasting model category and the machine learning forecasting model category are selected to be able to provide insights into the performance of these model categories on the case of DHL. From the statistical forecasting models, the ES models, SES, Holt's, and Holt's-Winters, the ARIMA models, ARIMA, SARIMA, ARIMAX, SARIMAX, and the Croston method have been selected to be implemented. ES has been chosen because it is one of the most well-known forecasting techniques and it offers a robust technique that is widely used in many business applications. Furthermore, ARIMA models have been chosen because this is another renowned category of forecasting models and it incorporates all other statistical methods, regression models, autoregressive models, and moving average models, into one model, which means it can use information on the past predicted values, past error terms, and predictive variables, to produce forecasts. Lastly, the Croston method was selected because this method is tailored to forecast on zero-inflated datasets with intermittent demand. As discussed before, the dataset on which is forecasted is zero-inflated.

From the machine learning models, NN models have been selected to implement because these models can deal with complex and non-linear data which makes them suitable to deal with seasonal and multivariate data. Furthermore, NNs are the most widely used machine learning model in time series forecasting and show promising results in literature. Two types of NNs models are implemented, MLP and LSTM. MLP is chosen because it is a more simple NN, which is easy to implement, and widely used in time series forecasting. Moreover, LSTM is chosen because, by making use of the gates system, LSTM has control over the processing of information. Consequently, LSTM can include long-term dependencies which makes it suitable for time series forecasting.

5.2 Performance metrics

To evaluate the forecasting performance of the models, performance metrics have to be selected. In Section 2.4, performance metrics that are used in literature are explained. From these metrics RMSLE is selected and used as the most important performance metric because it provides advantages that are desired in the case of forecasting picking tasks in the warehouse of DHL. In RMSLE, bigger differences are less penalized when both the predicted and the actual values are big, and underestimates are penalized slightly more than overestimates. This is desirable as a forecast can be quite good, even when the absolute error is far off, when the actual values are also big. Furthermore, this discourages the forecasting models from only predicting zero values for the zero-inflated data. On top of that, RMSLE can also deal with zero values. This is essential as the dataset is zero-inflated. How the RMSLE is calculated is shown in Equation 1. In the calculation, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i+1) - \log(\hat{y}_i+1))^2} \tag{1}$$

To evaluate the models more extensively, the RMSE is also used. The RMSE is computed to gain more insight into the performance of the forecasting models using a metric that does not differentiate between underestimation and overestimation and does not take into account the magnitude of the actual values. Furthermore, in the results the MAE is also provided. MAE is used to gain a better understanding of the forecasting error in absolute terms. The way in which RMSE and MAE are calculated is shown in Equation 2 and Equation 3.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

5.3 Forecasting

To perform the hyperparameter tuning and testing, cross-validation for time series with expanding window is implemented. This means that all the data, until the point that is forecasted from, can be used for training the forecasting model. As the point from which is forecasted goes further in time, the models are trained again on the data that is available until that forecast point. This implies that for every point that is forecasted from, and that is further in time than the previous point from which was forecasted, the training data of the forecasting model consists of data that was already available for the previous forecast point and newly available data that has become available because the new point from which is forecasted is further in time. In Figure 15, cross-validation for time series with expanding window is depicted.

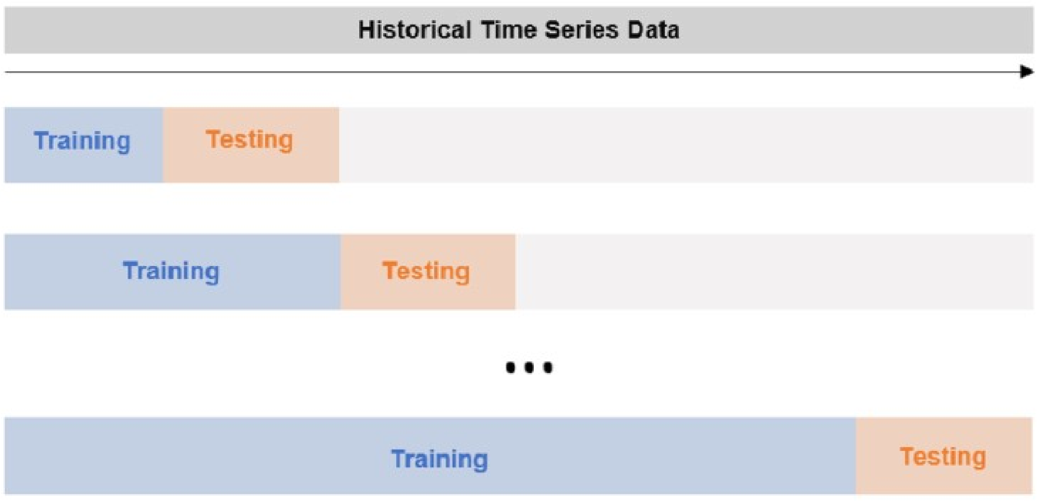


Figure 15: Cross-validation for time series with expanding window (Vien et al. (2021))

In Figure 15, the training part relates to the training, which is all the data until the point that is forecasted from. The testing part corresponds to the real number of picking tasks for the 48-hour forecast time period for which the forecast is made. Comparing the 48 forecasts of the forecasting models with the real values of the test set allows to evaluate the model performance. In cross-validation for time series, unlike in other types of cross-validation, it is essential to conserve the time dependency of the data. This is done to prevent training the forecasting models on data that has a timestamp after the point from which is forecasted. This is not allowed as the training data may contain information that is not known at the time of forecasting.

Furthermore, the forecast that is made consists of a forecast, 48 hours ahead, with a one-hour interval. This implies that the forecast consists of 48 values which means that a multi-step-ahead time series forecasting method is required. As described in Section 2.1.3, there are different methods available to perform multi-step-ahead time series forecasting. The multi-step-ahead time series forecasting method implemented in the forecasting models is MIMO. In MIMO the number of forecast values that is desired is forecasted at once by the model. This method was chosen because it does not have accumulation of forecasting errors, as in the Rec method. Furthermore, it also allows for one forecasting model, instead of 48 different models for every individual forecast value, as in the Dir method, which makes the training process a lot less cumbersome.

5.4 Hyperparameter tuning and testing

Hyperparameter tuning is performed to find the optimized hyperparameters for each forecasting model. Finding the optimal hyperparameters is essential for the forecasting models to be able to produce the most accurate forecasts. As explained in Section 5.3, the forecasts are made from a specific point, 48 hours ahead, and can be evaluated by comparing the forecasts to the test set of 48 actual values for the number of incoming picking tasks. Each time series is tuned on either three or five test sets in the cross-validation, depending on the running time of the forecasting models. These five (or three) test sets, and corresponding points from which is forecasted, are selected in the following way. First of all, the last twenty weeks of the dataset are reserved for testing the models. The five weeks (or three weeks in some models with longer running time) prior to these twenty weeks are selected for the parameter tuning. This leaves a bit more than 1 year of data to train the models on. In these five (or three) weeks, which are reserved for the parameter tuning, one point is randomly selected between Monday 00.00 AM and Thursday 00.00 AM to forecast from. These forecast points, which are randomly chosen, are uniform for all forecasting models. This interval is chosen to ensure that the models do not have to forecast on weekends which is not desired, as described in Section 1.4. These five (or three) points from which is forecasted make up the five (or three) cross-validation points on which the hyperparameter tuning is done. This implies that, for the five (or three) points from which is forecasted, the forecasting model is trained every time, with every hyperparameter combination that is tested and makes forecasts from these points. The forecasts are then reviewed corresponding to the actual values in the test set and the hyperparameter combination with the best performance over the five (or three) forecasts is chosen to be used in the testing phase. To decide what hyperparameter combination provides the performance in the forecasting models, the hyperparameter combination with the lowest RMSLE for that time series is chosen.

The hyperparameters of the models are tuned per individual time series and not on all time series aggregated. The result of this is that each individual time series has its own tuned hyperparameters which is assumed to enhance the performance of the forecasts per time series. Furthermore, per forecasting model category, different hyperparameter tuning methods are applied which fit the characteristics of the forecasting models. In Table 7, every forecasting model, including what type of hyperparameter tuning was applied, and on how many cross-validation test sets it was tuned on, is given.

Table 7: Parameter tuning of forecasting models

Model	**Parameter tuning**	**Forecasting points**
SES	Grid search	5
Holt's	Grid search	5
Holt's-Winters	Grid search	5
ARIMA	Grid search	5
SARIMA	Grid search	5
ARIMAX	Grid search	3
SARIMAX	Grid search	3
Croston	Grid search	5
MLP	Bayesian optimization	5
LSTM	Bayesian optimization	3

In Table 7, it can be seen that for the ES, ARIMA, and Croston models, grid search is used as hyperparameter optimization method. Grid search is chosen because either there are not too many hyperparameters to be tuned in the model, or the hyperparameters do not have many different values that are appropriate to test. Therefore, for these models, the grid search, which tests all the possible combinations of the values of hyperparameters that are put in, does not become too large. For the NNs however, Bayesian optimization is applied to tune the hyperparameters because in the NNs a lot of hyperparameters, which can take a lot of different possible values, are tuned. Bayesian optimization can tune hyperparameters more efficiently and can reach good results without having to do an exhaustive search of the hyperparameter combinations, which is why it is implemented for the NN models. Furthermore, as can be seen in Table 7, for the ARIMAX, SARIMAX, and LSTM models, only three points from which is forecasted are used in the parameter tuning instead of five. This is done to keep the running time of the parameter tuning within acceptable limits.

After the best hyperparameter combinations are found in the hyperparameter tuning, for each time series, for every model, the forecasting models are tested on unseen data. The testing of the forecasting models is done in the same way, by making use of cross-validation with expanding window, as the hyperparameter tuning. However, to ensure that the data is tested on unseen data, the test set is constructed from the last twenty weeks of the data, which is after the test points on which the parameter tuning is done. From these twenty weeks, the first ten weeks are used to construct the test set for the individual forecasting models, and the last ten weeks are used to test the implementation of the forecast models per zone or departure bucket. In this last phase, the forecast models are implemented per zone or departure bucket based on the performance of the individual models on the picking zone and departure bucket aggregation levels. Consequently, to ensure that there is no data leakage between the individual forecasting models and the implementation on the aggregation levels, these two test sets are separated. For both the test sets, just like in the parameter tuning, in each of the ten weeks, one point is randomly selected between Monday 00.00 AM and Thursday 00.00 AM to forecast from. These ten points are used to make the cross-validation test sets, just like in Figure 15. The testing set consists of two times 10 test points which is quite large relative to the parameter tuning which consists of 3 or 5 forecasting points. This is due to the fact that for testing the models already have tuned hyperparameters. Consequently, the model only needs to run once, with these hyperparameters, instead of running for different hyperparameter combinations. Therefore, the model can be tested on more forecasting points than the hyperparameter tuning while keeping the running times or the models acceptable.

5.5 Benchmark

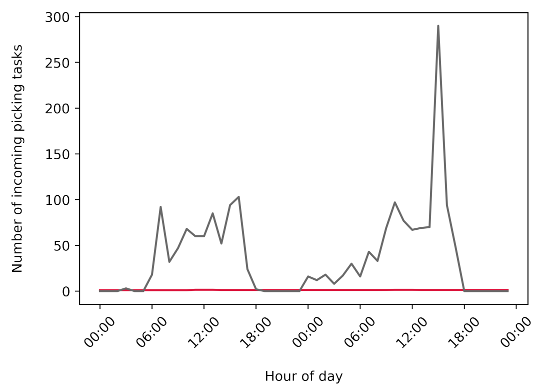
Currently, in the warehouse of DHL, a forecasting model is already in production to forecast the number of picking tasks coming in. This forecasting model is used as benchmark method to compare the chosen forecast models to, in terms of performance. The current forecasting model that is implemented is a lightGBM model that is also discussed as one of the supervised machine learning models in Section 2.2.9. This supervised machine learning model uses a combination of weak learners, which are often decision trees, to get an accurate forecasting model. Furthermore, this model predicts the number of picking tasks based on the input features that are engineered from the time series data, without directly taking the past values of the target variable, which is the number of incoming picking tasks, into account. Indirectly, features like the mean over a certain period, or lagged values, provided as features, can (partly) be used instead of having the past values of the target variable as input. In Appendix B, the features that are engineered, and used as input to the benchmark forecasting model, are described. Apart from these engineered features, also the departure bucket, and picking zone, which were already available as features in the dataset, are used as input features.

The benchmark method thus aims to forecast the number of picking tasks based on time series features, engineered from the time series data, without directly using the past values of number of picking tasks. Furthermore, the data that is used as input to the benchmark model consists of all the data for all the different picking zones and departure buckets, which are added as features in the dataset. This is a fundamental difference to the approach of the forecasting models that are implemented with the aim to improve the forecasting performance of the benchmark. These forecasting models, all use the past values of number of picking tasks as a sequence, as input, either with or without selected time series features that are engineered. The data that is used as input for these forecasting models consist of the data on past values, and optionally the engineered features, only on the picking zone and departure bucket for which the forecasts are made. This implies that these two features are not used in the forecast of the implemented forecasting models as the picking zone and departure bucket are the same for all the input data on which is forecasted. The input data of the implemented forecasting models are filtered per picking zone and departure bucket in the data preprocessing of each model. Consequently, the performance of the different approaches in use of time series data can also be compared. In Table 8, the performance metrics RMSLE, MAE, and RMSE, on the whole dataset, for the benchmark model, can be observed.

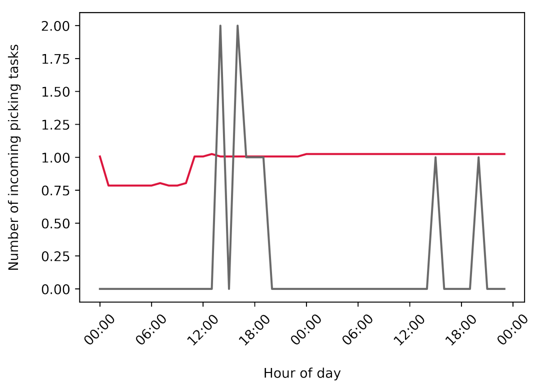
Table 8: Performance metrics of Benchmark

Model	**RMSLE**	**MAE**	**RMSE**
Benchmark	1.63	6.65	14.82

Furthermore, in Table 33, in Appendix D, the results of the benchmark per picking zone, and per departure bucket can be seen. This is done to gain more insights into the performance of the different models across different picking zones and departure buckets. As can be concluded from these results, the benchmark performs quite consistently across zones and departure buckets by reviewing the RMSLE. However, there are differences in zones and departure buckets when reviewing the MAE and RMSE. Noticeable is that the RMSLE seem to be consistent across zones and buckets, but the MAE and RMSE seem to be higher for time series with higher mean number of picking tasks. This can be explained by the fact that RMSLE evaluates the performance by making use of a log that punishes differences when actual values are bigger, less compared to MAE and RMSE. To gain more insights into how the benchmark makes the forecasts, a wide range of different individual forecasts for different time series were evaluated to see how these perform with respect to the actual values. The insights that were obtained from this analysis are explained by the two forecasts that are depicted in Figure 16. One of these forecasts represents the insights gained for time series with high mean number of picking tasks and one of these forecasts represents the insights gained for time series with low mean number of picking tasks. In this figure, the grey line represents the actual number of incoming picking tasks and the red line represents the forecast made by the model.



(a) Benchmark, Pickbelt, (0,(16, 20])



(b) Benchmark, Pick_Above, (1,(16, 20])

Figure 16: Forecasts of Benchmark

First of all, the plot with Pickbelt, (0,(16, 20]), is a forecast for the time series in zone ”Pickbelt”, with days to departure equal to 0 and departure time bucket (16, 20]. the plot with Pick Above, (1,(16, 20]), is a forecast for the time series in zone ”Pick_Above”, with days to departure equal to 1 and departure time bucket (16, 20]. The former is a time series with high mean number of tasks and the latter is a time series with lower mean number of picking tasks. This can also be seen by the y-axis of these plots which are of different magnitudes.

As can be seen in Figure 16a, the benchmark does not forecast the seasonality in this time series, with high mean number of tasks, well. In the more extensive analysis, it came forward that for some time series with high mean number of incoming picking tasks, the benchmark was sometimes able to forecast the seasonality reasonably well, but in other instances was not able to do this. For the biggest portion of the time series with high mean number of incoming picking tasks, the benchmark was unable to forecast this seasonality well. Furthermore, as can be seen in Figure 16b, for this time series, with low mean number of picking tasks, the forecast is too high relative to the actual values. In the extensive analysis, for almost all time series with low mean number of incoming picking tasks, the benchmark overestimated the actual number of incoming picking tasks as in this plot. Both these insights indicate why the benchmark can be improved.

5.6 Exponential smoothing

Three ES forecasting models are chosen to be implemented in this thesis. These are SES, Holt’s, and Holt’s-Winters. In Section 2.2.1, these three ES models are defined. To optimize the performance of the models, hyperparameter tuning is performed as described in Section 5.4. As mentioned in this section, the hyperparameter tuning method used for the ES techniques is grid search. The values that were tested in the grid search were set with the aim to test a range of possible values, which are reasoned to possibly provide good results, while also keeping the running time of the grid search acceptable. Grid search tests each of the hyperparameter combinations that can be made by the values that are set to be tested for each hyperparameter. Consequently, a model with a smaller number of hyperparameters that are tested can have more test values per hyperparameter to arrive at an equal level of hyperparameter combinations that need to be run. Due to this the SES model has more values that are tested for its only hyperparameter, α, than the other two models. Furthermore, not all hyperparameters that can be set in the models are tuned in the grid search. Certain hyperparameters are set by reviewing the time series to limit the running time of the grid search. In Table 9, the values that are tested, per hyperparameter, per ES model, are provided.

Table 9: Test values of hyperparameters that are tuned for every ES model

Model	**Hyperparameters tuned**	**Test values**
SES	α	0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9
Holt's	α	0.05, 0.9
	β	0.01, 0.2, 0.9
	Trend	Additive, None
HW	α	0.05, 0.2, 0.9
	γ	0.01, 0.2, 0.9

For SES, the only hyperparameter that needs to be specified is α. The values that are tested in the grid search for this hyperparameter can be seen in Table 9. In Holt's, the hyperparameters that can be set are α, β, ϕ, trend parameter, and damping parameter. The hyperparameters ϕ and the damping parameter are related to damping the trend in the model. As there is no indication that this is needed in the time series data, these hyperparameters are set to no damping. The other hyperparameters, α for the level, and β and the trend parameter for the trend, are tuned by the grid search, as can be seen in Table 9. The trend hyperparameter is set to none or additive for the model to be able to test whether or not adding a trend component provides better results for individual time series. As there is no clear indication of a trend in the time series data however, apart from setting the trend component to additive, there is also a possibility for the Holt's model to specify no trend, which would make it equal to SES, if this provides the best results. Lastly, for Holt's-Winters, the hyperparameters α, β, ϕ, γ, trend parameter, damping parameter, seasonal parameter, and seasonal period can be set. As there is no clear indication of a trend in the time series data, the trend components are set to no trend, to test the Holt's-Winters without trend, but only with the seasonal component, that is present in the dataset as discussed in Section 4.4. This implies that β, ϕ, the trend parameter, and the damping parameter do not have to be included in the grid search. Furthermore, it is not observed in the data that the magnitude of the seasonal components change over time. Therefore, the seasonal parameter is set to additive. Moreover, the seasonal period is set to 168 ($24 * 7$) which corresponds to the seasonal pattern which comes back on the same hour on the same weekday, which means that the seasonal period comes back every 168 ($24 * 7$) hours for hourly demand. This seasonal period relates to the hourly and daily seasonality discussed in Section 4.2. Setting this hyperparameter to 24 would indicate hourly seasonality, but would not include the fact that the day of the week is also important to take into account in the seasonality. The α and γ variables are included in the grid search as can be observed in Table 9. In Appendix C, the tuned parameters per forecasting models are given. In Table 10, the results on the performance metrics for the models SES, Holt's, and Holt's-Winters, on the whole dataset, are provided.

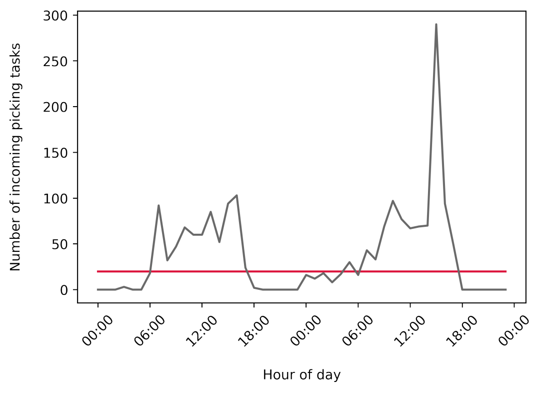
Furthermore, the results of the benchmark are added to be able to compare the results of the ES models to the benchmark.

Table 10: Performance metrics of ES models

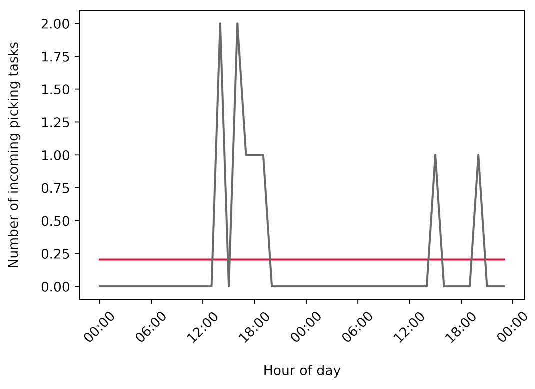
Model	RMSLE	MAE	RMSE
SES	0.88	3.28	13.62
Holt's	0.99	5.94	98.30
Holt's-Winters	0.64	2.24	10.89
Benchmark	1.63	6.65	14.82

What can be seen from the results on the performance metrics of the three ES models in Table 10 is that for all three performance metrics, the Holt's-Winters model performs the best. After the Holt's-Winters model, the SES model performs best on all three metrics. When comparing to the benchmark, it can be seen that the RMSLE and MAE are significantly better for all three ES models, when compared to the benchmark. Only the RMSE for the Holt's model is higher when compared to the benchmark. This is due to the fact that Holt's model on average has a lower absolute error than the benchmark, but has some errors that are very large compared to the benchmark which gets punished harder in the RMSE metric.

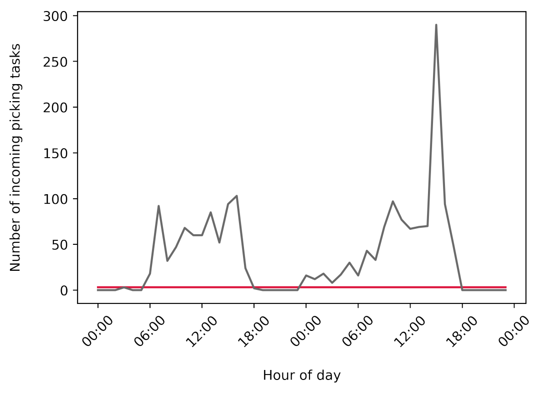
In Table 34, Table 35, and Table 36, in Appendix D, the results of the forecasting models, per picking zone, and per departure bucket can be seen. This is done to gain more insights into the performance of the different models across different picking zones and departure buckets. Unlike in the benchmark, the RMSLE is significantly higher for time series with higher mean number of picking tasks. Higher errors in time series with higher mean number of picking tasks can also be seen in the MAE and RMSE of all three ES models. An anomaly to this is the results for the departure bucket (2, (8,12)) which are high for all three models, and all three performance metrics, while the mean number of picking tasks in this departure bucket is relatively low. As for the benchmark, an extensive analysis of the individual forecasts was done for different time series. The insights that were obtained from this analysis are explained by the two forecasts of every model that are depicted in Figure 17. One of these forecasts represents the insights gained for time series with high mean number of picking tasks and one of these forecasts represents the insights gained for time series with low mean number of picking tasks. In this figure, the grey line represents the actual number of incoming picking tasks and the red line represents the forecast made by the model.



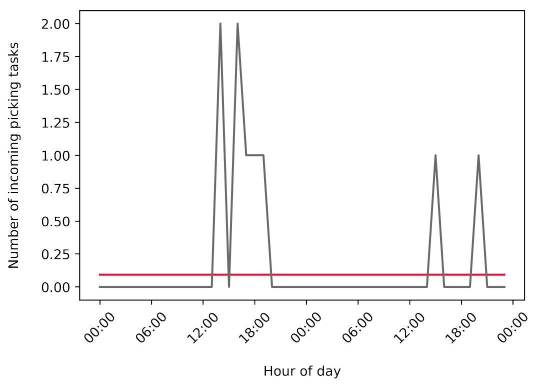
(a) SES, Pickbelt, (0,(16, 20])



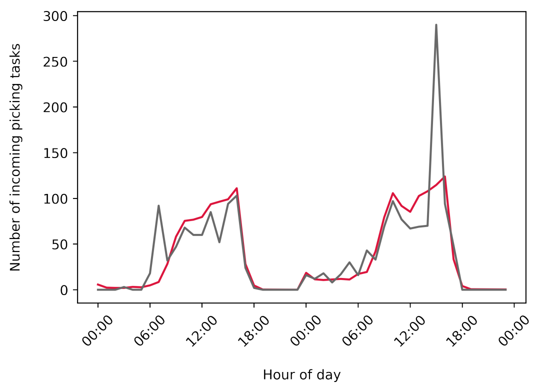
(b) SES, Pick_Above, (1,(16, 20])



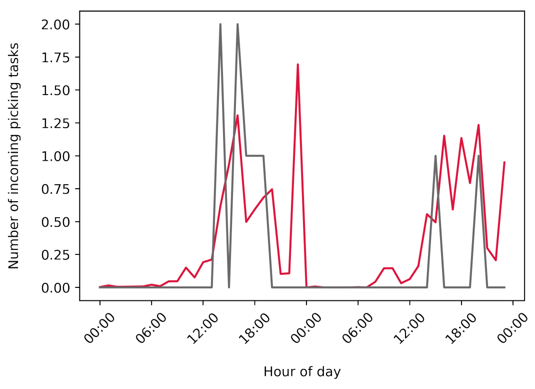
(c) Holt's, Pickbelt, (0,(16, 20])



(d) Holt's, Pick_Above, (1,(16, 20])



(e) Holt's-Winters, Pickbelt, (0,(16, 20])



(f) Holt's-Winters, Pick_Above, (1,(16, 20])

Figure 17: Forecasts of Exponential smoothing models

The plots of the forecasts were made on the same time series as for the benchmark in Section 5.5. What can be concluded from these plots is that SES and Holt's forecast (almost) the same value for each of the 48 hours ahead. This implies that these models are not able to deal with the seasonality which can be seen, especially in the time series with high mean number of tasks. This potentially causes these models to perform worse than the Holt's-Winters model that can deal with seasonality by forecasting different values for the 48 hours ahead. As can be seen in Figure 17e, the Holt's-Winters forecasts the seasonality in the time series with high mean number of tasks a lot better than the SES and Holt's models. Furthermore, in Figure 17f, it can be seen that the Holt's-Winters models also try to forecast the seasonality in the time series with low mean number of tasks. It distinguishes the times of the day in which the number of incoming picking tasks is likely to be zero or non-zero pretty well, but forecasting this accurately is hard since it is only very few picking tasks that come in. Being able to forecast the seasonality in the time series is the main reason why the Holt's-Winters performs better than the other two ES models.

5.7 ARIMA

Four ARIMA models are selected to be tested in this thesis. These are ARIMA, SARIMA, ARIMAX, and SARIMAX. In Section 2.2.3, these models are explained. To tune the hyperparameters of these models, hyperparameter tuning, as discussed in Section 5.4, is performed. To tune the hyperparameters of the ARIMA models, grid search is implemented. As in the ES models, the values that were tested in the grid search were set with the aim to test a range of possible values, which are reasoned to may be able to provide good results, while also keeping the running time of the grid search acceptable. As a result, a model with a smaller number of hyperparameters that are tested can have more test values per hyperparameter to arrive at an equal level of hyperparameter combinations that need to be run. Due to this, the ARIMA and ARIMAX models have more values that are tested for the two hyperparameters that are tuned in these models, than the other two models, SARIMA and SARIMAX for the four hyperparameters that are tuned in these models. Furthermore, not all hyperparameters that can be set in the models are tuned in the grid search. Certain hyperparameters are set by reviewing the time series to limit the running time of the grid search. In Table 11, the values that are tested, per hyperparameter, per ARIMA model, are provided. In Appendix C, the tuned parameters per forecasting models are given.

Table 11: Test values of hyperparameters that are tuned for every ARIMA model

Model	Hyperparameters tuned	Test values
ARIMA	p	0, 1, 2
	q	0, 1, 2
SARIMA	p	0, 1
	q	0, 1
	P	0, 1
	Q	0, 1
ARIMAX	p	0, 1, 2
	q	0, 1, 2
SARIMAX	p	0, 1
	q	0, 1
	P	0, 1
	Q	0, 1

Apart from the tuned parameters, for all of the ARIMA models, the differencing order d needs to be specified. As discussed in Section 4.3, the KPSS test showed that not all time series are stationary without differencing. However, after applying first-order differencing (i.e. $d = 1$) to the time series that are non-stationary, these become stationary. Consequently, in all ARIMA models, the d parameter is set to 0, if the time series is stationary according to the KPSS test, and the d parameter is set to 1, if the time series is non-stationary according to the KPSS test, making this time series stationary after first-order differencing.

Additionally, in the seasonal ARIMA models, SARIMA and SARIMAX, the seasonal differencing parameter D and the seasonal period s need to be specified. First of all, the seasonal differencing order D is set to 0 for both models, as the differencing parameter d has already ensured stationarity. Furthermore, the seasonal period s is set to 168 $(24 * 7)$ which corresponds to the seasonal pattern which comes back on the same hour, on the same weekday, which means that the seasonal period comes back every 168 $(24 * 7)$ hours for hourly demand. This seasonal period relates to the hourly and daily seasonality discussed in Section 4.2.

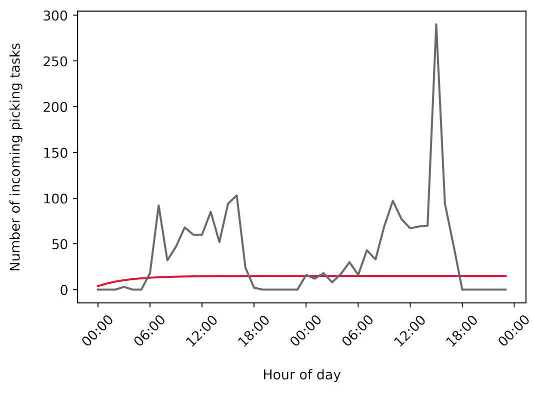
Lastly, for the ARIMAX and SARIMAX models, selected features are also included in a regression-like manner. The features that are included, including why these are included, is explained in Section 4.5. These features are: weekday, hour, holiday, school holiday, and mean number of picking tasks for the specific weekday and hour combination until that point in time. In Table 12, the results of the ARIMA models on the whole dataset, compared to the benchmark, are shown.

Table 12: Performance metrics of ARIMA models

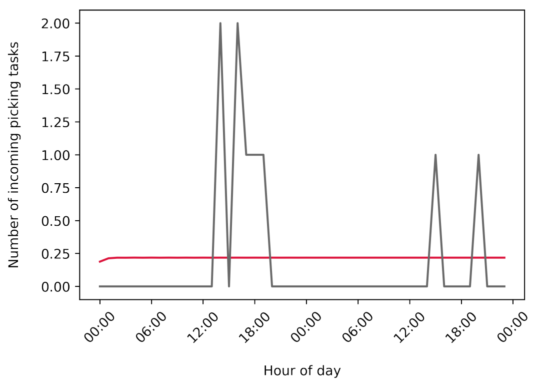
Model	**RMSLE**	**MAE**	**RMSE**
ARIMA	0.89	3.23	14.15
SARIMA	0.68	2.46	12.04
ARIMAX	0.56	1.96	10.33
SARIMAX	0.57	2.02	10.40
Benchmark	1.63	6.65	14.82

As can be seen in Table 12, all the ARIMA models outperform the benchmark on all three performance metrics. Furthermore, the ARIMAX model performs best, followed by the SARIMAX model, the SARIMA model, and the ARIMA model. This indicates that including the features in the ARIMAX and SARIMAX models improves performance significantly. Furthermore, the seasonal effect included in the SARIMA models also provides a significant enhancement in comparison to the ARIMA model. In Table 37, Table 38, Table 39, and Table 40, in Appendix D, the results of the ARIMA models can be observed, per zone and departure bucket. From these results, it can be concluded that all the ARIMA models have higher errors for the time series with high mean number of picking tasks. The ARIMAX and SARIMAX models, however, outperform the SARIMA model, especially on the time series with high mean number of incoming picking tasks. Additionally, the ARIMA model scores poorly in comparison to SARIMA, ARIMAX, and SARIMAX models on all time series.

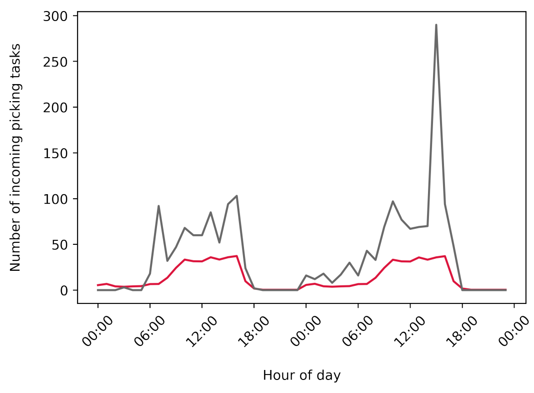
To evaluate the forecasts made by the ARIMA models more extensively, an analysis of the individual forecasts was done for different time series. The insights that were obtained from this analysis are explained by the two forecasts of every model that are depicted in Figure 18. These plots are of the same two time series as in the previous sections. In this figure, the grey line represents the actual number of incoming picking tasks and the red line represents the forecast made by the model.



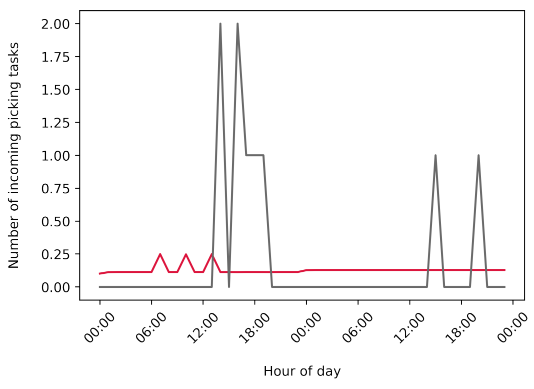
(a) ARIMA, Pickbelt, (0,(16, 20])



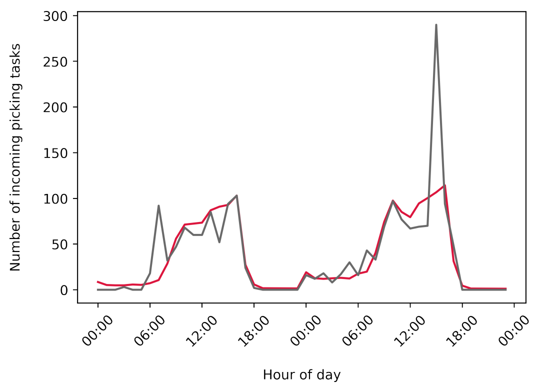
(b) ARIMA, Pick_Above, (1,(16, 20])



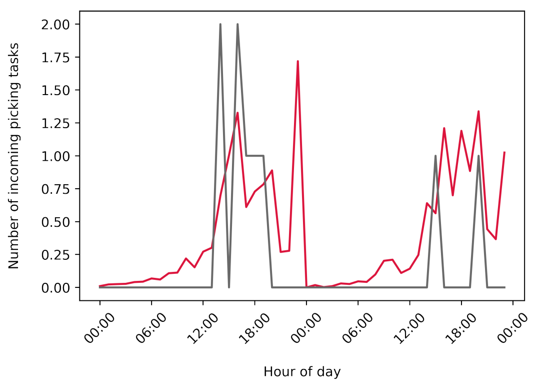
(c) SARIMA, Pickbelt, (0,(16, 20])



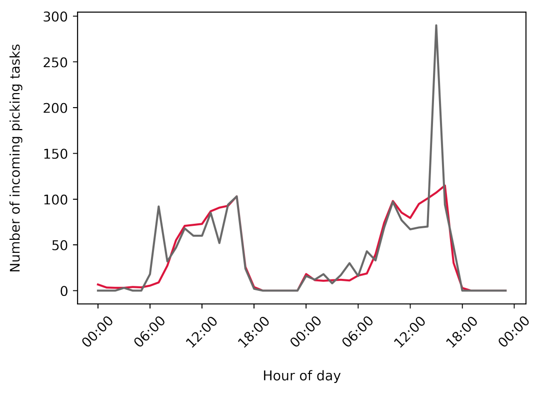
(d) SARIMA, Pick_Above, (1,(16, 20])



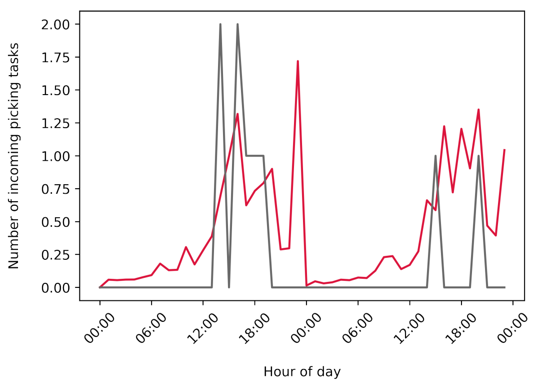
(e) ARIMAX, Pickbelt, (0,(16, 20])



(f) ARIMAX, Pick_Above, (1,(16, 20])



(g) SARIMAX, Pickbelt, (0,(16, 20])



(h) SARIMAX, Pick_Above, (1,(16, 20])

Figure 18: Forecasts of ARIMA models

What can be concluded from these plots is that the ARIMA models cannot deal with the seasonality in the time series with picking zone "Pickbelt", days to departure equal to 0, and departure time bucket (16, 20]. The SARIMA does forecast the seasonality but cannot fully estimate the magnitude of the peak values in the seasonality. Furthermore, the ARIMAX and SARIMAX models do forecast the seasonality but are able to forecast the magnitude of the seasonality better. This phenomenon is observed across all the time series with high mean number of picking tasks. As mentioned before, in the results per zone and per departure bucket, the ARIMAX and SARIMAX especially outperform the SARIMA model on the zones and departure bucket with high mean number of tasks. This confirms the claim that ARIMAX and SARIMAX are better able to forecast the magnitude of the seasonality in the time series with high mean number of tasks than the SARIMA model. Adding the features to the ARIMAX and SARIMAX models causes these models to be able to forecast the magnitude of the actual values better.

In the time series with picking zone "Pick_Above", days to departure equal to 1, and departure time bucket (16, 20], it can be seen that the ARIMA and SARIMA model predict a value close to zero for all 48 values in the forecast. On the other hand, the ARIMAX and SARIMAX models still try to predict the seasonality in this time series with a lot less non-zero actual values. As observed before this is more difficult than forecasting the seasonality in the time series with high mean number of tasks. This can also be seen in the results because the performance of the SARIMA models is comparable to the performance of the ARIMAX and SARIMAX models for time series with low mean number of incoming picking tasks. Therefore, forecasting values close to zero for all 48 hours ahead, as the SARIMA model or trying to forecast the seasonality, which is hard as non-zero values do not occur frequently, does not matter too much in terms of performance, in the time series with low mean number of incoming picking tasks. The ARIMA model seems to forecast a value a little bit further away from zero for the time series with low mean number of incoming picking tasks which is why, for these time series, the performance of this model is quite poor.

5.8 Croston method

Apart from the well-known statistical models, ES and ARIMA, also a statistical method that is specially made for forecasting zero-inflated data is implemented. In Section 2.2.4, the theoretical background on this method is provided. As described in Section 5.4, the hyperparameter tuning for the Croston method is done by grid search. The only hyperparameter that needed to be tuned was the smoothing parameter α. In Table 13, the values that were tested in the grid search to tune the smoothing parameter are shown.

Table 13: Test values of hyperparameter that are tuned for Croston method

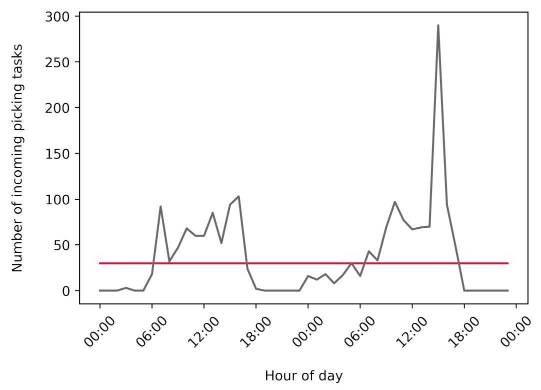
Model	**Hyperparameters tuned**	**Test values**
Croston	α	0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9

In Appendix C, the tuned smoothing parameter per time series of the Croston method are given. Furthermore, in Table 14, the results of the performance metrics for the Croston method, on the whole dataset, are provided.

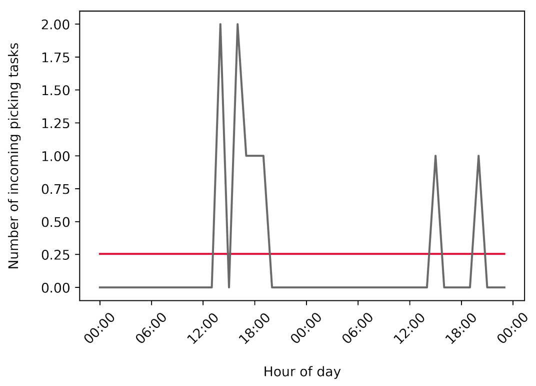
Table 14: Performance metrics of Croston method

Model	RMSLE	MAE	RMSE
Croston	0.86	3.24	13.30
Benchmark	1.63	6.65	14.82

As can be seen in Table 14, the Croston performs considerably better on all three performance metrics compared to the benchmark. In Table 41, in Appendix D, the results of the Croston method can be seen per zone and departure bucket. From these results can be conducted that the Croston method performs better on all three performance metrics for zones and departure buckets with lower mean number of incoming picking tasks. In Figure 19, two examples of forecasts that are made by the Croston method are depicted. In Figure 19a, a forecast for a time series with higher mean number of tasks is shown. In Figure 19b, a forecast for a time series with lower mean number of tasks is shown. In these figures, the grey line is the actual number of incoming picking tasks for each hour, whereas the red line is the forecasted number of incoming picking tasks per hour. Furthermore, the y-axis, which represents the number of incoming picking tasks, are on different scales. To gain more insights into how the Croston method makes the forecasts, several different forecasts for different time series were evaluated. The insights that were obtained from this analysis are explained by the two forecasts of every model that are depicted in Figure 19. Just like in the previous sections, One of these forecasts represents the insights gained for time series with high mean number of picking tasks and one of these forecasts represents the insights gained for time series with low mean number of picking tasks. In this figure, the grey line represents the actual number of incoming picking tasks and the red line represents the forecast made by the model.



(a) Croston, Pickbelt, (0,(16, 20])



(b) Croston, Pick_Above, (1,(16, 20])

Figure 19: Forecasts of Croston method

The same two time series are evaluated as in previous sections. As can be seen in Figure 19, the forecasts that are made by Croston method are the same for all 48 values of the forecast. As can be seen from Figure 19a, the Croston method does not forecast the seasonality that can be seen during the day in the plot for the high mean number of picking tasks. Furthermore, for time series with a low mean number of incoming picking tasks, like the one in Figure 19b, the Croston method forecasts values really close to 0 for all the 48 values. As the actual value is often zero, and sometimes a low number, this prediction is often close to the actual value.

5.9 Neural networks

Four NN models are selected to be tested for the case of DHL. These are MLP and LSTM, both with and without features. In Section 2.2.8, these models are explained. As discussed in Section 5.4, the method that was used to tune the hyperparameters of these models is Bayesian optimization. Bayesian optimization tries to select the hyperparameters with the best performance in a range of hyperparameter values by making use of the performance of the previous hyperparameter set. Bayesian optimization is often more efficient than grid search in terms of running time as it selects the next set of hyperparameters to evaluate based on the performance of the previous hyperparameter set, reducing the number of evaluations needed, while grid search evaluates all combinations of hyperparameters exhaustively. In Table 15, the hyperparameters that are tuned for each of the NN models are given, including the range of values that can be tested. In Appendix C, the tuned parameters per forecasting models are given.

Table 15: Test values of hyperparameters that are tuned for every ARIMA model

Model	**Hyperparameters tuned**	**Test value range**
MLP	Dropout	0.01 - 0.5
	Learning rate	0.00001 - 0.005
	Neurons	10 - 50
	Epochs	5 - 20
	Layers	1 - 2
LSTM	Dropout	0.01 - 0.5
	Learning rate	0.00001 - 0.000011
	Neurons	10 - 50
	Epochs	5 - 10
	Layers	1 - 2

First of all, all the test value ranges that are shown for the MLP model, are used for both the MLP with features and the MLP without features. In the same way, the test value ranges that are shown for the LSTM model, are used for both the LSTM with features and the LSTM without features. As can be seen in Table 15, the test value ranges of the MLP and LSTM are different for the hyperparameters learning rate and epochs. The lower range of test values for the learning rate in LSTM is to avoid that the model predicts nan values, as was sometimes the case with higher learning rate. Furthermore, the range of test values for the number of epochs is lower for LSTM to keep the running time of the Bayesian optimization reasonable. Moreover, the number of iterations that the Bayesian optimization runs for, needs to be set. One iteration corresponds to one hyperparameter combination that can be tested for the models. The number of iterations is set to 10 for the MLP models and 3 for the LSTM models, again with the aim to keep the running time reasonable.

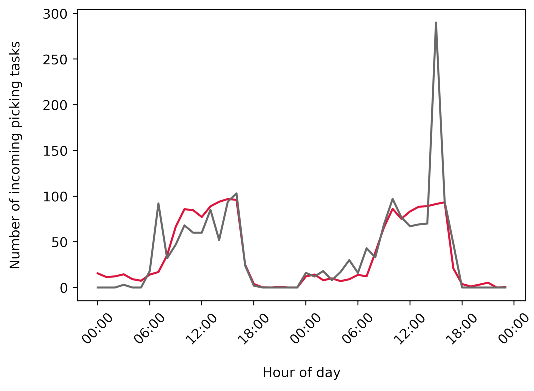
To help the learning process of the NN models, normalization is applied to the input data. This implies that all the values in the input data were scaled between 0 and 1. Normalization is chosen over standardization as the data on the number of incoming picking tasks does not follow a Gaussian distribution. Furthermore, in the NN models, vectors with past values are put in, on which the NN models predict the number of picking tasks for the next 48 hours. To train the NN models, multiple input vectors and corresponding output vectors (of 48 values which correspond to the 48 hours ahead) have to be constructed. As mentioned in Section 5.4, there is a bit more than 1 year of data. To train the NN models, vectors with past values of the last 70 days are made. The first vector is of the first 70 days of the train set, the next vector then shifts one day further to create a new vector on which is trained. This results in a total of 333 vectors on which can be trained. 70 days of data is chosen as the length of the vector because it is believed that it provides enough data per vector for the NN model to make decent predictions, while also being able to construct enough vectors to train the NN models on.

For the MLP and LSTM models without features, the vectors of past values for the last 70 days are the input that is provided to these models. For the MLP and LSTM models with features however, also the values of the features are used as input. The features that are used as input can be found in Section 4.5. The values of the features that are desired as input for these models are the values that correspond to the 48 hours for which the incoming tasks are predicted. For example, it is relevant for the model to know for which days of the week, or hours of the day, the predictions are made, instead of knowing the day of the week, or hour of the day, for the past values in the input vectors. Consequently, the length of the input vectors, which contain the values of the features, is 48 values, corresponding to the 48 hours that are forecasted. However, as described before, the input vector for the past values of incoming picking tasks contains values of the last 70 days. As the values are on an hourly interval, the length of this input vector is 1680 (70*24). However, to use these values as input to the NN, a matrix has to be formed. To form this matrix, the length of the vectors of past values of incoming picking tasks, and feature values, need to be equal to each other. To make this possible, padding and masking are used. First of all, nan values are added to the vectors that contain the features values to make the length of these vectors equal to the length of the vector of past values of incoming picking tasks. This is also known as padding. After that, the input matrix is made by combining these equally sized vectors. Lastly, a masking layer is added to the NN models that takes out the nan values. This makes sure that only the feature values are used in the NN, and that the nan values are not included in the rest of the calculations. This is known as masking. In Table 16, the results of the NN models on the whole dataset, including the results on the benchmark model, are shown.

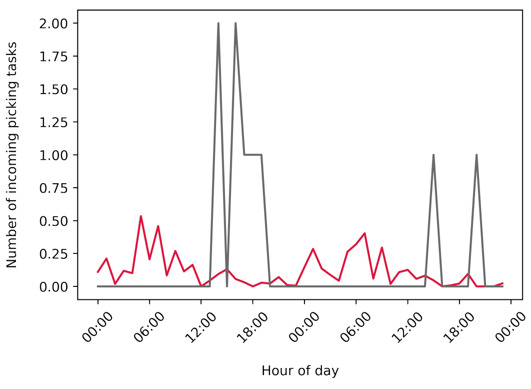
Table 16: Performance metrics of NN models

Model	RMSLE	MAE	RMSE
MLP without features	0.94	3.35	14.26
MLP with features	0.93	3.22	13.93
LSTM without features	0.92	2.90	14.67
LSTM with features	0.92	2.91	14.67
Benchmark	1.63	6.65	14.82

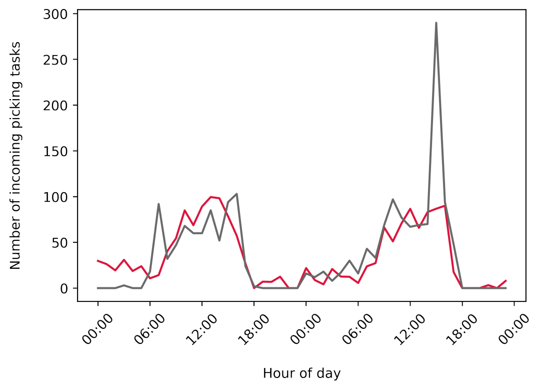
As can be seen in Table 16, the NN models outperform the benchmark on all three performance metrics. Especially for the RMSLE and MAE, the differences in performance between the NN models and the benchmark model are large. Furthermore, it can be observed that the LSTM models outperform the MLP models on RMSLE and MAE but the MLP models outperform the LSTM on the RMSE. This indicates that on average the error in the LSTM is lower, however, the LSTM models more often have bigger differences between the prediction and the actual value. Furthermore, it can be concluded that adding the features to the MLP model improves the forecasting performance by a small margin. In the LSTM model, however, adding the features did not improve the forecasting performance. The fact that the addition of features does not provide a substantial improvement for the NN models is potentially caused by the NN models not being able to link the added features to the number of incoming picking tasks. This can be caused by the fact that the NN are unable to detect relations between the features and the number of incoming picking tasks, or the feature value set on which the NN are trained being too small, as there are only 48 feature values trained on every time In Table 42, Table 43, Table 44, and Table 45, in Appendix D, the results of the NN models are given per zone and departure bucket. From these results, it can be concluded that, as for the other forecasting models, the NN models perform better in terms of performance metrics on time series with lower mean number of picking tasks. Like for the other forecasting models, an extensive analysis of the forecasts of the NN models for different time series was done. The insights that were obtained from this analysis are explained by the two forecasts of every model that are depicted in Figure 20. The plots of the forecasts that are shown are made on the same time series as was done for the other forecasting models. In this figure, the grey line represents the actual number of incoming picking tasks and the red line represents the forecast made by the model.



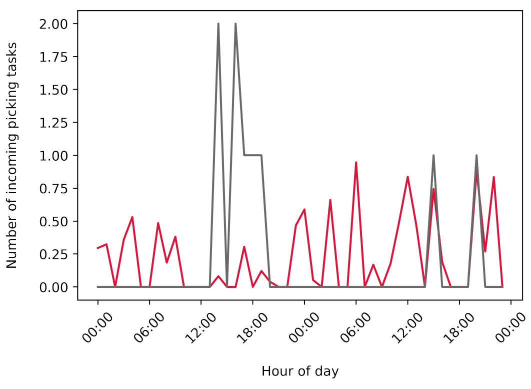
(a) MLP wihout features, Pickbelt, (0,(16, 20])



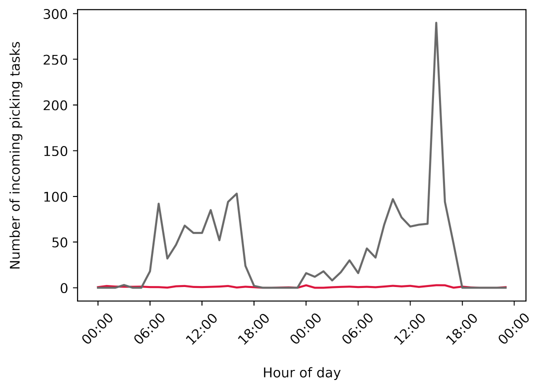
(b) MLP wihout features, Pick_Above, (1,(16, 20])



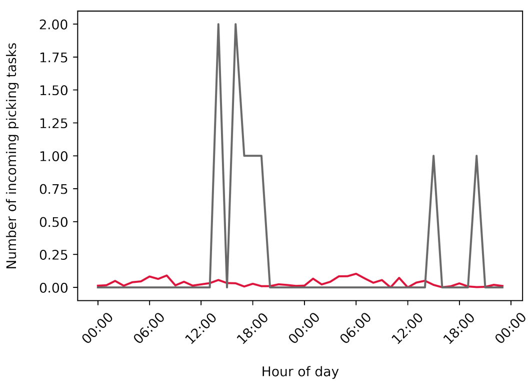
(c) MLP with features, Pickbelt, (0,(16, 20])



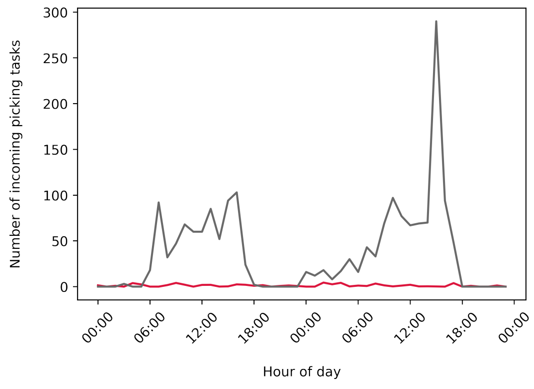
(d) MLP with features, Pick_Above, (1,(16, 20])



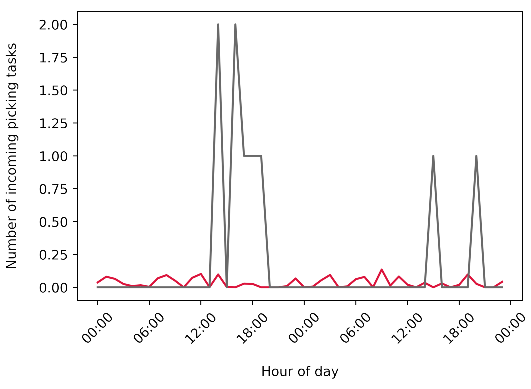
(e) LSTM without features, Pickbelt, (0,(16, 20])



(f) LSTM without features, Pick_Above, (1,(16, 20])



(g) LSTM with features, Pickbelt, (0,(16, 20])



(h) LSTM with features, Pick_Above, (1,(16, 20])

Figure 20: Forecasts of Neural network models

What can be concluded from Figure 20 is that there are big differences between the MLP models and the LSTM model when making forecasts. First of all, when reviewing the forecasts made for the time series with picking zone ”Pickbelt”, days to departure equal to 0, and departure time bucket (16, 20], it can be seen that the MLP models forecast the seasonality in this time series with high mean number of incoming picking tasks well, opposed to the forecast of the LSTM model which forecasts only values close to zero. This is also reflected by the results in Appendix D because on the zone and departure bucket with the highest mean number of tasks, the MLP models perform better in terms of RMSLE than the LSTM models. The fact that the LSTM cannot identify the seasonal pattern well could be caused by several factors. In some time series problems certain NN models have shown to be unable to model the seasonality (Zhang and Qi (2005)). Furthermore, the size of the training set can also cause the NN models to perform poorly on multi-step ahead forecasting (Sánchez-Sánchez et al. (2020)).

When evaluating the time series with picking zone ”Pick_Above”, days to departure equal to 1, and departure time bucket (16, 20], again the MLP aims to forecast the seasonality in the time series, whereas the LSTM predicts a value close to zero every time. Due to the little portion of non-zero values in this time series, it is hard for the MLP to predict the times when the number of incoming picking tasks is going to be non-zero as shown in the plots. For this forecast, LSTM provides better accuracy because the values are often zero.

The higher MAE for the MLP models compared to the LSTM model indicate that the MLP models have on average a higher absolute error. This is potentially caused by the fact that there is a larger number of time series with low mean number of incoming picking tasks, that have actual values which are often zero. As shown in Figure 20, the LSTM is better at predicting these because the LSTM often predicts values close to zero. On the other hand, the lower RMSE for the MLP models compared to the LSTM model indicates that the MLP can handle the variability in the time series better and has less large errors, as RMSE punishes larger errors harder. This can also be observed in Figure 20 because the MLP models can forecast the variability of large values in the time series with picking zone ”Pickbelt”, days to departure equal to 0, and departure time bucket (16, 20], better.

5.10 Discussion on performance of forecast models

In this section of the report, the performance of the forecasting models is discussed and compared. In Figure 21, the RMSLE for all the forecasting models, per picking zone, and per departure bucket is visualized. RMSLE is displayed because this performance metric best suits the case of DHL as explained before in Section 5.2. In this figure, the different forecasting models are displayed as different colored line plots, and the zone and departure buckets are on the x-axis. Furthermore, the MLP_wf and LSTM_wf are the MLP and LSTM models without engineered features. The MLP_wsf and LSTM_wsf are the MLP and LSTM models with selected features.

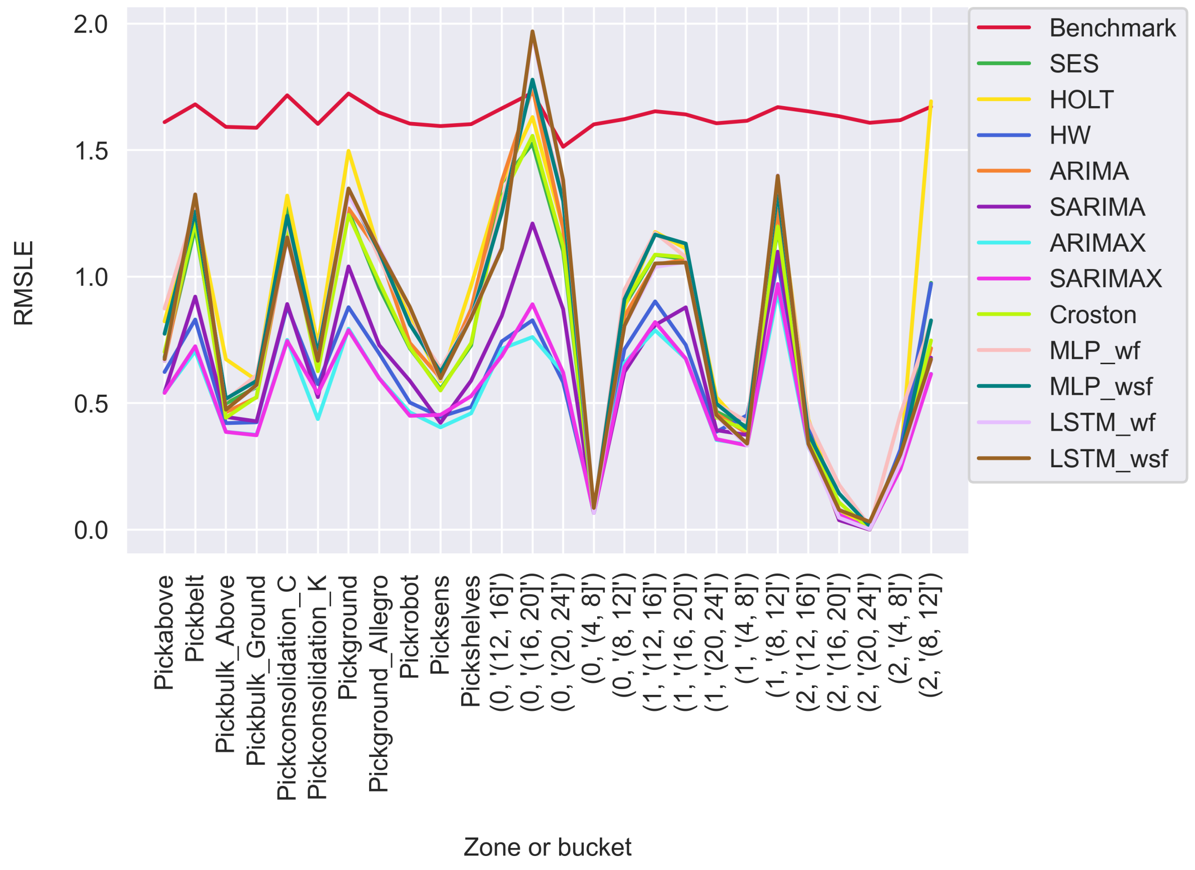


Figure 21: RMSLE of all forecasting models, per zone, and departure bucket

As can be seen in Figure 21, for virtually all picking zones and departure buckets, in terms of RMSLE, the implemented forecasting models outperform the benchmark. The forecasting models especially outperform the benchmark in zones and departure buckets with low mean number of incoming picking tasks. The forecasting models have lower RMSLE for the time series in zones and departure bucket with low mean number of incoming picking tasks, whereas the RMSLE of the benchmark stays reasonably constant. This is potentially caused by the approach which is used to make the forecasts. The benchmark treats the forecasting as a regression on the different picking zones and departure buckets while the forecasting models implemented in this thesis approach it more as a classical time series forecast. Because of this approach, the predictions of the implemented forecast models are made on data that is divided per individual time series, instead of the benchmark, which uses all the data of all zones and departure buckets together to make forecasts. Because of this, the benchmark can have a harder time distinguishing between the different zones and departure buckets compared to the other forecasting models, for which this division is already made in the data preprocessing.

Furthermore, it can be observed that the RMSLE is for which this is already done higher for the zones and departure buckets that have higher mean number of tasks. It is not the case that certain models clearly perform better on certain zones or departure buckets. Instead, it is observed that the forecasting models that perform best overall, perform among the best on all zones and departure buckets, both with low and high mean number of incoming picking tasks. What can also be observed is that the difference in performance on the RMSLE is highest on the zones and departure buckets with high mean number of incoming picking tasks. This is caused by the fact that some forecasting models cannot deal with the seasonality that is present in the time series for these zones and departure buckets which implies these forecast models perform poorly on forecasting these time series. Moreover when actual values are higher, as on average in the time series with zones and departure buckets with higher mean number of picking tasks, being a certain percentage away from the actual value results in a higher error than for time series with a lower average actual value.

When evaluating the performance of the implemented forecast models in more detail, it becomes clear that the best-performing models are the models that can deal with the seasonality in the time series. Being able to forecast the weekday and hourly seasonality, which is discussed more extensively in Section 4.4, accurately proved to be an important factor for good forecasting performance, especially in time series with high mean number of incoming tasks. Additionally, the statistical methods that included a seasonal component outperformed the NN in terms of forecasting performance. The poor performance of the NN models compared to the statistical methods that included a seasonal component is potentially due to the fact that in some time series problems NN models are unable to model the seasonality (Zhang and Qi (2005)). This can especially be seen in the predictions of the LSTM models in time series with high mean number of incoming picking tasks. Furthermore, the size of the training set can also cause the NN models to perform poorly on multi-step ahead forecasting (Sánchez-Sánchez et al. (2020)).

Additionally, adding the features to the forecasting models significantly improved the forecasting performance of the ARIMA models. For the MLP and LSTM model, including the features provided little to no improvement in forecasting performance. For the MLP model, adding the features provided a very minimal improvement in the forecasting performance. The fact that the addition of features does not provide an improvement for the NN models can be caused by the fact that the NN are unable to detect relations between the features and the number of incoming picking tasks, or the feature value set on which the NN are trained being too small. For the ARIMA models, adding the features in the ARIMAX and SARIMAX models provided significant improvements in forecasting performance compared to the ARIMA and SARIMA models. The ARIMAX model provided the best overall forecast performance, closely followed by the SARIMAX model.

Apart from the performance, the running time of the forecasting models is also given as this might be a factor when implementing forecasting models. In Table 17, the running time of all the forecasting models, including the benchmark, is provided.

Table 17: Running time of forecasting models

Model	**Running time**
Benchmark (lightGBM)	31 minutes, and 52 seconds
SES	30 minutes and 30 seconds
Holt's	42 minutes and 26 seconds
Holt's-Winters	21 minutes and 16 seconds
ARIMA	43 minutes and 3 seconds
SARIMA	1 hour, 32 minutes, and 17 seconds
ARIMAX	7 hours, 13 minutes, and 42 seconds
SARIMAX	20 hours, 54 minutes, and 38 seconds
Croston	38 minutes and 18 seconds
MLP without features	49 minutes and 39 seconds
MLP with features	1 hour, 23 minutes, and 47 seconds
LSTM without features	3 days, 10 hours, 12 minutes, and 29 seconds
LSTM with features	4 days, 12 hours, 8 minutes, and 42 seconds

As can be seen in Table 17, there are differences in the running time of the forecasting models. The benchmark, ES models, ARIMA and SARIMA, Croston, and MLP models, all have reasonably short running times. The running time of the ARIMAX and SARIMAX models consist of multiple hours. However, the models that take by far the longest to run are the LSTM models. Both with and without features these models take multiple days to run. This should be taken into consideration when deciding on what model to put into production in the environment of DHL.

5.11 Applying forecast models to time series groups

In Appendix D, the results of the forecast models per zone and per departure bucket are shown. From these results, it can be seen that there is not one forecast model that performs best for every zone and every departure bucket. Rather, for specific zones and departure buckets there are different forecasting models that provide the lowest RMSLE. RMSLE is used because it provides the best fit for the case of DHL as also discussed in Section 5.2. Because of this, applying the best performing model per specific zone, or per specific departure bucket, on the time series that includes this picking zone or departure bucket, instead of applying one forecast model on all the time series, could improve forecasting performance. That is why, in this section of the report, two cases are tested that apply different forecasting models per zone or per departure bucket. In one case, the best-performing forecast model per zone is applied to all the time series that include that zone. In the other case, the best-performing forecast model per departure bucket is applied to all the time series that include that departure bucket. Which model performs best on the zone or departure bucket is selected by evaluating which forecasting model has the lowest RMSLE for that zone or departure bucket. Which models are applied per zone or per departure bucket in these two cases can be seen in Appendix E.

To test these two cases, the same tuned hyperparameters that were used in the forecast models before, when one forecast model was applied to all time series, are used again for each specific forecast model in these cases. Moreover, the two cases are tested on new data which has a later timestamp than the data on which the individual forecast models were tested to ensure there is no data leakage. This is also described before in Section 5.4. In Table 18, the three performance metrics of the tested cases, along with the best-performing individual model and the performance metrics of the benchmark can be seen. Apart from that the running time when the models are applied per zone is 11 hours, 49 minutes, and 10 seconds. When the models are applied per departure bucket, the running time is equal to 12 hours, 58 minutes, and 38 seconds.

Table 18: Performance metrics of tested cases

Model	**RMSLE**	**MAE**	**RMSE**
Models applied per zone	0.56	2.06	10.04
Models applied per departure bucket	0.55	2.01	9.91
ARIMAX	0.56	1.96	10.33
Benchmark	1.63	6.65	14.82

As can be seen in this table, in terms of RMSLE, there is a minor improvement in the case in which the forecasting models are applied per departure bucket. In terms of MAE, both the new cases do worse than the ARIMAX model, however, in terms of RMSE, the new cases both perform better than the ARIMAX model. This implies that the new cases are worse in terms of absolute errors, but overall have fewer errors that are large. These differences can be attributed to the different forecasting models that are applied per case. It should also be noted that these new running times have slightly more running time than the ARIMAX model. In Table 46 and Table 47, in Appendix D, the results of the new cases per zone, and per departure bucket can be seen.

6 Conclusions

In this section of the report, the conclusions that can be drawn from the thesis are given. First of all, the research questions that are given in Section 1.5, are revisited, and answers to these research questions are provided. After that, recommendations that can be given based on the results of the report are described. Lastly, the limitations of the research and indications for future research are discussed.

6.1 Answering the research questions

As mentioned before, in this section answers to the research questions, that are defined at the start of the thesis, are given. The research questions can be found in Section 1.5. First, the five research sub-questions are answered. After that, the main research question of this thesis is answered.

RQ1: How should the dataset be prepared to make forecasts on?

In order to make the dataset appropriate to make forecasts on, preparation steps had to be taken. These preparation steps were needed due to different reasons, with the aim to get a clean dataset on which the forecasts can be made. First of all, several cleaning steps were performed to ensure the data quality of the dataset. Furthermore, departure buckets were introduced in the dataset. With this operation, the number of time series is reduced, which improves the understandability of the forecasts, while retaining the needed information level for DHL. Lastly, to complete the time series, the zero observations, which were not in the initial dataset, had to be imputed. This implies that, for every picking zone and departure bucket combination, for every hour where there is no observation, a zero value had to be imputed.

RQ2: What characteristics can be attributed to the underlying time series on different aggregation levels and what does this imply for data transformations of time series?

In Chapter 4, apart from describing the data preparation, data analysis was performed on the dataset to identify characteristics of the time series data. First of all, the descriptive statistics showed that the dataset was zero-inflated, which means that most of the values in the dataset are zero. Furthermore, this analysis showed that there were big differences between picking zones and departure buckets in terms of mean number of incoming picking tasks and the percentage of zero values for incoming picking tasks.

Furthermore, the KPSS test was used to investigate whether or not the time series were stationary. This showed that a considerable number of the time series were non-stationary which implied that these needed to be differenced for them to be appropriate to use in the ARIMA models. For the other statistical and machine learning models this was not needed (Lewinson (2020)). Moreover, the time series were analyzed to investigate if seasonal patterns were detectable. From this analysis was concluded that the time series of incoming picking tasks have seasonality in the combination of the hour of the day and day of the week. This led to the engineering of time series features that were believed to capture this seasonality. These features were included in various models that allowed for the inclusion of time series features. Apart from the features that were engineered based on the seasonality analysis, two features that indicated if it is a holiday or a school holiday by making use of a binary variable were also engineered and used as input into the forecasting models.

RQ3: Which time series models are available and which ones best suit the problem at hand?

To identify time series models that are available, a literature study on time series models was performed. The most vital information of this literature study on time series forecast models can be found in Section 2.2. An overview of the forecasting models that were found in literature and are discussed in the theoretical background of the thesis can be found in Figure 6. The forecasting models that were selected to be implemented were the ES models, SES, Holt's, and Holt's-Winters, the ARIMA models, ARIMA, SARIMA, ARIMAX, and SARIMAX, the Croston method, and the NN models, MLP and LSTM, both with and without added time series features. First of all, the ES models were chosen to be implemented because these models are very well-known to offer robust performance in many business applications. Furthermore, the ARIMA models were chosen because these are also very renowned and incorporate the statistical methods, regression models, autoregressive models, and moving average models, into one model. This implies that the ARIMA models use information on the past predicted values, past error terms, and predictive variables, to produce forecasts. The last statistical method that was chosen was the Croston method. This method was chosen because it is designed to be able to handle zero-inflated datasets like the one used in this thesis.

Apart from the statistical methods, machine learning methods were implemented to forecast the number of picking tasks in the warehouse of DHL. These consisted of the NN models MLP and LSTM, both with and without features. MLP was chosen because it is one of the simpler NN models, with decent computing speed, which is often used in time series forecasting problems. Furthermore, LSTM was chosen because this NN model can include long-term dependencies which makes it suitable for many time series forecasting applications.

RQ4: Which performance evaluation metrics are relevant to be used?

The three metrics that are used to measure forecast performance are RMSLE, MAE, and RMSE. RMSLE best suits the case of DHL to measure forecasting performance. This is why RMSLE is chosen as the metric on which the hyperparameters are tuned, and on which the modification of the individual models is based. RMSLE best fits the case of DHL because it penalizes bigger differences less when both the predicted and the actual values are big, and penalizes underestimates slightly more than overestimates. These properties fit the case of DHL because although the absolute error is relatively big, the forecast can still be quite good relative to the magnitude of the actual value. Furthermore, penalizing underestimates slightly more than overestimates discourages the forecasting models to only predict zero values, especially in time series with low mean number of incoming picking tasks. On top of that, the RMSLE allows for zero values in the prediction and actual value which is essential as the dataset is zero-inflated.

Additionally, RMSE and MAE are calculated to measure the performance of the forecasting models. RMSE is used to gain more insight into the performance of the forecasting models using a metric that does not differentiate between underestimation and overestimation and does not take into account the magnitude of the actual values when calculating the error. Furthermore, the MAE is provided to gain a better understanding of the forecasting error in absolute terms, which is easy to understand.

RQ5: What forecast models provide the best performance and how can the results on different aggregation levels of the forecasts be explained?

The forecast model that overall provides the best forecast performance is the ARIMAX model, closely followed by the SARIMAX model. Furthermore, all the forecasting models that were chosen to be tested in this thesis outperformed the benchmark in terms of RMSLE. The cause of this might be the fact that the models that were tested all approached the forecasting problem as a time series problem, in which the time series data is divided and then used as input. The benchmark, however, approaches this forecasting problem as a regression in which the data on all the time series together is used to make predictions.

The forecast models show higher errors for time series which have higher mean number of picking tasks. The time series with higher mean number of picking tasks are more important for DHL as these are the time series where the bulk of the incoming picking tasks are in. Overall, the best performing models are the statistical models that include a seasonal component because these models showed to be able to deal with the seasonality in the time series well. These models outperformed the NN models, which in literature often perform really well in time series forecasting. Lastly, adding the engineered features to the input data improved the forecasting performance of the NN models by a small margin. In the ARIMA models however, adding the engineered features to the input data improved the forecasting performance by a considerable amount, which lead to the best-performing forecasting model, ARIMAX.

In the results of the forecasting models that were applied to all time series, it can be observed that there is not one forecasting model that performs best across all picking zones and departure buckets. Therefore, with the aim to improve forecast performance, the best-performing forecast model for each picking zone, or each departure bucket, were applied to the time series that included that picking zone or departure bucket. This resulted in two new cases that were tested. One case in which different forecasting models were applied to groups of time series based on picking zone, and one case in which different forecasting models were applied to groups of time series based on departure bucket. This showed that a minor improvement to the forecast performance, in terms of RMSLE, can be realized when applying the best performing forecast models per departure bucket, in comparison to the ARIMAX model, which is the best performing forecast model when models are applied to all time series together.

How can picking tasks in the warehouse of DHL be more accurately predicted by using time series forecasting?

To conclude, all the forecasting models that were tested in this thesis outperformed the benchmark method of DHL. This is potentially caused by the approach of the time series models that were tested, which use the time series per picking zone, and per departure time, as input data. Furthermore, including a seasonal component in the forecasting models or using engineered time series features improved forecasting performance in the case of forecasting incoming picking tasks in the warehouse of DHL. All in all, ARIMAX was the forecasting model that provided the best performance when one forecasting model was applied to all time series. This performance was enhanced by a small margin when the best forecasting models were implemented on time series that were grouped by departure bucket.

6.2 Conclusions on scientific contribution

As described in Section 1.7, the scientific contribution of this thesis is to show how different well-known forecasting techniques perform on short-term time series forecasting with zero-inflated time series data. From the results, it can be concluded that the models that include a seasonal component perform best because these models can forecast the time and magnitude of the seasonality in the time series data well. It can be concluded that these models perform well on the zero-inflated data because the occurrence of the zero and non-zero values are largely predictable by the seasonality that is present in the time series data and is therefore not random. Furthermore, because the seasonality is on the hour in the day and weekday interval, the forecasting models with the seasonal components can use this seasonality to their advantage when performing short-term forecasting that spans 48 hours. All in all, based on the results of this thesis, when performing short-term time series with zero-inflated data it is important to assess if the non-zero and zero values in the zero-inflated data can be predicted or are random. When these values can be predicted, for example by seasonal effects, the predictors of the zero values and non-zero values should be included in the time series models to get satisfactory results.

6.3 Recommendations for DHL

As mentioned above, the forecast models in this thesis outperform the benchmark of DHL in forecasting performance. It is therefore recommended to evaluate if there would be a possibility to test the approach followed in this thesis, in the warehouse of DHL, to forecast the incoming picking tasks. If this provides satisfying results, the implementation of this approach, instead, or along side, the benchmark could be the next step. It has to be noted that there are limitations in this thesis that are discussed in Section 6.4. These limitations should be considered when deciding to test or implement the forecasting models tested in this thesis. Furthermore, in Section 5.10, the running times of the forecasting models are provided. These running times also should be taken into account when choosing which model would best fit the case of forecasting picking tasks in the warehouse of DHL.

Another recommendation for DHL is to continue to investigate ways in which the forecasting performance can be improved. Improving forecasting performance might be possible through the exploration of new forecasting models, trying different approaches in the use of data, or using different multi-step-ahead forecasting techniques.

6.4 Limitations and future research

First of all, the main limitations of this research are caused by the many time series that are forecasted. Consequently, to keep the running time reasonable, only a few different points from which the forecasts are made were trained and tested for every time series. This limited number of forecasting points for training and testing might cause these points to not fully represent other forecasting points in the time series. Consequently, this could cause the hyperparameter tuning to not be able to find the optimal hyperparameters for the total time series based on these forecasting points. Furthermore, the results of testing the forecasting models on these forecasting points might not fully represent forecasting from other points in time in the time series.

Another limitation of this research is that only the multi-step-ahead forecasting technique MIMO is applied to make forecasts. Other multi-step-ahead forecast techniques, which are discussed in Section 2.1.3, could provide better forecasting performance. An interesting topic for future research would be testing whether or not different multi-step-ahead forecasting techniques would provide better results than the MIMO technique applied in this thesis.

Thirdly, when grouping the time series to apply different forecast models to each of these groups, the time series were grouped by picking zone or departure bucket. However, making a different division of the groups of time series, for example by combinations of both picking zone and departure bucket, could provide better results. It would be interesting to further research if there are different groupings of time series, on which forecasting models can be applied, that provide better results in terms of forecasting performance.

Lastly, the exploration of other forecasting models and other features that can potentially enhance forecasting performance on zero-inflated time series data would be an interesting topic for future research. When exploring other models to forecast on zero-inflated time series, it would be especially interesting to explore other forecasting models that allow for the predictors of the zero and non-zero values to be included. Furthermore, exploring and including other features that provide more information on the structure of the time series can also potentially enhance the forecasting performance of the models that are implemented in this thesis.

References

Afriyie, J. K., Twumasi-Ankrah, S., Gyamfi, K. B., Arthur, D., and Pels, W. A. (2020). Evaluating the performance of unit root tests in single time series processes. 37

Ahani, I. K., Salari, M., and Shadman, A. (2019). Statistical models for multi-step-ahead forecasting of fine particulate matter in urban areas. *Atmospheric Pollution Research*, 10(3):689–700. 12, 13

Ahmed, D. M., Hassan, M. M., and Mstafa, R. J. (2022). A review on deep sequential models for forecasting time series data. *Applied Computational Intelligence and Soft Computing*, 2022. 17

Atwan, T. A. (2022). *Time Series Analysis with Python Cookbook.* Packt Publishing, Birmingham, England. 12

Avishek, P. and Pks, P. (2017). *Practical Time Series Analysis : Step by Step Guide Filled with Real World Practical Examples.* Packt Publishing. xi, 9, 10, 11, 12

Bao, Y., Xiong, T., and Hu, Z. (2014). Multi-step-ahead time series prediction using multiple-output support vector regression. *Neurocomputing*, 129:482–493. 12, 16

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., and Jenssen, R. (2017). Recurrent neural networks for short-term load forecasting: an overview and comparative analysis. 17, 18

Bou-Hamad, I. and Jamali, I. (2020). Forecasting financial time-series using data mining models: A simulation study. *Research in International Business and Finance*, 51:101072. 18, 19

Braspenning, S. (2022). Literature review on time series forecasting. 9

Cankurt, S. and SUBAŞI, A. (2016). Tourism demand modelling and forecasting using data mining techniques in multivariate time series: a case study in turkey. *Turkish Journal of Electrical Engineering and Computer Sciences*, 24(5):3388–3404. 16

Chapman, P., Clinton, J., Kerber, R., Khabaz, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). Crisp-dm 1.0: Step-by-step data mining guide. the crisp-dm consortium. spss. 5

Ferreira, L. B. and da Cunha, F. F. (2020). Multi-step ahead forecasting of daily reference evapotranspiration using deep learning. *Computers and electronics in agriculture*, 178:105728. 13

Gasparin, A., Lukovic, S., and Alippi, C. (2022). Deep learning for time series forecasting: The electric load case. *CAAI Transactions on Intelligence Technology*, 7(1):1–25. 13, 17, 18

George E. P., B., Gwilym M., J., Gregory C., R., and Greta M., L. (2016). *Time Series Analysis : Forecasting and Control.*, volume Fifth edition George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Greta M. Ljung of *Wiley Series in Probability and Statistics*. Wiley. 3

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts. 10, 11, 13, 14, 15, 16, 18

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688. 23, 24

Ivanovski, Z., Milenkovski, A., and Narasanov, Z. (2018). Time series forecasting using a moving average model for extrapolation of number of tourist. *UTMS Journal of Economics*, 9(2). 14, 15

Kahraman, E. and Akay, O. (2022). Comparison of exponential smoothing methods in forecasting global prices of main metals. *Mineral Economics*, pages 1–9. 13

Kayacan, E., Ulutas, B., and Kaynak, O. (2010). Grey system theory-based models in time series prediction. *Expert systems with applications*, 37(2):1784–1789. 16

Lazzeri, F. (2020). *Machine learning for time series forecasting with Python*. John Wiley & Sons. 9, 10, 17, 18

Lewinson, E. (2020). *Python for Finance Cookbook: Over 50 recipes for applying modern Python libraries to financial data analysis*. Packt Publishing Ltd. 36, 37, 69

Li, X., Zhang, C., Zhang, B., and Liu, K. (2019). A comparative time series analysis and modeling of aerosols in the contiguous united states and china. *Science of The Total Environment*, 690:799–811. 9, 10, 12

Liu, H.-T. (2007). An improved fuzzy time series forecasting method using trapezoidal fuzzy numbers. *Fuzzy Optimization and Decision Making*, 6(1):63–80. 16

Liu, S., Forrest, J., and Yang, Y. (2011). A brief introduction to grey systems theory. In *Proceedings of 2011 IEEE International Conference on Grey Systems and Intelligent Services*, pages 1–9. IEEE. 16

Luo, J., Zhang, Z., Fu, Y., and Rao, F. (2021). Time series prediction of covid-19 transmission in america using lstm and xgboost algorithms. *Results in Physics*, 27:104462. 19

Mir, A. A., Çelebi, F. V., Alsolai, H., Qureshi, S. A., Rafique, M., Alzahrani, J. S., Mahgoub, H., and Hamza, M. A. (2022). Anomalies prediction in radon time series for earthquake likelihood using machine learning-based ensemble model. *IEEE Access*, 10:37984–37999. 24

Pena-Sanchez, Y., Mérigaud, A., and Ringwood, J. V. (2018). Short-term forecasting of sea surface elevation for wave energy applications: The autoregressive model revisited. *IEEE Journal of Oceanic Engineering*, 45(2):462–471. 14

Sánchez-Sánchez, P. A., García-González, J. R., and Coronell, L. H. P. (2020). Encountered problems of time series with neural networks: Models and architectures. *Recent Trends in Artificial Neural Networks-from Training to Prediction*. 63, 65

Schaffer, A. L., Dobbins, T. A., and Pearson, S.-A. (2021). Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: a guide for evaluating large-scale health interventions. *BMC Medical Research Methodology*, 21(1). 10, 11, 12

Shu, X., Peng, Y., Ding, W., Wang, Z., and Wu, J. (2022). Multi-step-ahead monthly streamflow forecasting using convolutional neural networks. *Water Resources Management*, pages 1–16. 12, 13

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85. 19, 20

Vien, B. S., Wong, L., Kuen, T., Rose, L., and Chiu, W. K. (2021). A machine learning approach for anaerobic reactor performance prediction using long short-term memory recurrent neural network. *Struct. Health Monit. 8apwshm*, 18:61. xi, 45

Wang, Q., Li, S., and Li, R. (2018). Forecasting energy demand in china and india: Using single-linear, hybrid-linear, and non-linear time series forecast techniques. *Energy*, 161:821–831. xi, 13, 16, 18, 22

Wanishsakpong, W. and Owusu, B. E. (2020). Optimal time series model for forecasting monthly temperature in the southwestern region of thailand. *Modeling Earth Systems and Environment*, 6(1):525–532. 15

Wu, S. and Akbarov, A. (2011). Support vector regression for warranty claim forecasting. *European Journal of Operational Research*, 213(1):196–204. 16

Xiao, Y., Xiao, J., and Wang, S. (2012). A hybrid model for time series forecasting. *Human Systems Management*, 31(2):133–143. 19, 20

Xu, W., Peng, H., Zeng, X., Zhou, F., Tian, X., and Peng, X. (2019). A hybrid modelling method for time series forecasting based on a linear regression model and deep learning. *Applied Intelligence*, 49(8):3002–3015. Copyright - Applied Intelligence is a copyright of Springer, (2019). All Rights Reserved; Last updated - 2021-06-21. 19, 20

Zeroual, A., Harrou, F., Dairi, A., and Sun, Y. (2020). Deep learning methods for forecasting covid-19 time-series data: A comparative study. *Chaos, Solitons Fractals*, 140:110121. 12

Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514. 63, 65

Zhang, L., Bian, W., Qu, W., Tuo, L., and Wang, Y. (2021). Time series forecast of sales volume based on xgboost. In *Journal of Physics: Conference Series*, volume 1873, page 012067. IOP Publishing. 19

Zhang, Y., Zhu, C., and Wang, Q. (2020). Lightgbm-based model for metro passenger volume forecasting. *IET Intelligent Transport Systems*, 14(13):1815–1823. 19

Zhao, H. and Guo, S. (2016). An optimized grey model for annual power load forecasting. *Energy*, 107:272–286. 16

Zhao, Z., Wang, J., Zhao, J., and Su, Z. (2012). Using a grey model optimized by differential evolution algorithm to forecast the per capita annual net income of rural households in china. *Omega*, 40(5):525–532. 16

Appendices

A KPSS test for time series

Table 19: P-values of KPSS with non-stationary time series differenced

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	0.1	0.1	0.1	0.1	0.06	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Pickbelt	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09	0.1
Pickbulk_Above	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Pickbulk_Ground	0.1	0.1	0.08	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.06	0.1	0.1
Pickconsolidation_C	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Pickconsolidation_K	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.06	0.1
Pickground	0.1	0.08	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Pickground_Allegro	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.06	0.1	0.1	0.1	0.1	0.1
Pickrobot	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Picksens	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.1	0.1	0.08	0.1
Pickshelves	0.08	0.1	0.1	0.1	0.1	0.1	0.1	0.07	0.1	0.1	0.1	0.1	0.1	0.1	0.1

In Table 19, the p-values of the KPSS test for all the time series, after the time series that were non-stationary, were first-order differenced. As can be seen in the table, now all p-values of the KPSS test are higher than 0.05 which indicates stationarity for all the time series.

B Features for benchmark

Table 20: Features engineered and used as input to benchmark

Feature	**Description**	**Feature type**	**Mean**	**Min**	**Max**
Year	The year of the observation as integer	Datetime	2020.9	2020	2022
Quarter	The quarter of the year of the observation as integer	Datetime	2.56	1	4
Month	The month of the year as integer (1 = January, etc.)	Datetime	6.64	1	12
Day	The day of the month as integer	Datetime	15.63	1	31
Weekday	The weekday of the week as integer (0 = Monday, etc.)	Datetime	3	0	6
Hour	The hour of the observation	Datetime	11.50	0	23
Weekend	Binary variable indicating if it is weekend (Saturday or Sunday) at the time of the observation (0 = not weekend, 1 = weekend)	Datetime	0.29	0	1
Holiday	Binary variable indicating if it is a holiday or not (0 = not holiday, 1 = holiday). This includes national holidays of the Netherlands, Germany, Great Britain, France, Italy, and Spain	Datetime	0.09	0	1
Holiday_school	Binary variable indication if it is a holiday for the schools in the south of the Netherlands (0 = not holiday, 1 = holiday)	Datetime	0.26	0	1
Weekday_mean	Mean number of tasks for that weekday	Expanding window	1.99	0	84
Day_mean	Mean number of tasks for that day of the month	Expanding window	1.97	0	114
Hour_mean	Mean number of tasks for that hour in the day	Expanding window	2.00	0	441

Month_mean	Mean number of tasks in that month	Expanding window	2.02	0	78.67
Quarter_mean	Mean number of tasks in that quarter	Expanding window	1.97	0	78.67
Weekday_hour_mean	Mean number of tasks for that weekday and hour combination	Expanding window	2.01	0	441
Day_hour_mean	Mean number of tasks for that day of the month and hour combination	Expanding window	2.01	0	487
Month_day_mean	Mean number of tasks for that month and day of the month combination	Expanding window	1.70	0	409
Month_weekday_mean	Mean number of tasks for that month and weekday combination	Expanding window	1.93	0	185
Quarter_weekday_mean	Mean number of tasks for that quarter and weekday combination	Expanding window	1.97	0	87
Sum_today	Sum of tasks that have come in, in that day	Rolling window	21.33	0	2731
Mean_last_7_days	Mean number of tasks of the last 7 days	Rolling window	2.028	0	70.8
Mean_last_30_days	Mean number of tasks of the last 30 days	Rolling window	2.03	0	70.8
Mean_last_365_days	Mean number of tasks of the last 365 days	Rolling window	2.00	0	70.8
lag_t-hour	Lagged values for t-1 (one hour earlier) until t-24 (24 hours / one day earlier) with an interval of one hour are engineered	Lag	2.03	0	1033

C Tuned parameters of forecasting models

Table 21: Tuned hyperparameters SES

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	0.3	0.01	0.9	0.01	0.01	0.5	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.1	0.01
Pickbelt	0.01	0.01	0.7	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.1	0.9
Pickbulk_Above	0.9	0.01	0.9	0.01	0.01	0.01	0.01	0.01	0.01	0.5	0.05	0.9	0.01	0.3	0.05
Pickbulk_Ground	0.5	0.7	0.9	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.9	0.7	0.5	0.01
Pickconsolidation_C	0.01	0.5	0.7	0.01	0.01	0.01	0.01	0.01	0.05	0.01	0.01	0.9	0.01	0.3	0.1
Pickconsolidation_K	0.3	0.5	0.7	0.01	0.01	0.01	0.01	0.01	0.1	0.3	0.05	0.9	0.05	0.3	0.01
Pickground	0.01	0.01	0.9	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.3	0.1
Pickground_Allegro	0.01	0.01	0.05	0.01	0.01	0.05	0.01	0.01	0.01	0.01	0.5	0.01	0.01	0.3	0.9
Pickrobot	0.01	0.05	0.1	0.01	0.01	0.01	0.01	0.3	0.01	0.01	0.01	0.1	0.05	0.5	0.9
Picksens	0.9	0.1	0.01	0.01	0.01	0.01	0.01	0.1	0.01	0.3	0.7	0.01	0.01	0.01	0.01
Pickshelves	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.9

Table 22: Tuned hyperparameters Holt's

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0.9, 0.9, add)	(0.05, 0.01, add)	(0.9, 0.01, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.2, add)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, add)
Pickbelt	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, add)	(0.9, 0.01, add)
Pickbulk_Above	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, add)
Pickbulk_Ground	(0.05, 0.01, add)	(0.9, 0.01, add)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.9, add)	(0.9, 0.01, add)	(0.9, 0.01, None)	(0.9, 0.01, add)	(0.05, 0.01, add)
Pickconsolidation_C	(0.05, 0.2, add)	(0.9, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, add)	(0.05, 0.01, add)
Pickconsolidation_K	(0.05, 0.01, add)	(0.9, 0.9, add)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, add)	(0.9, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, add)	(0.05, 0.01, add)
Pickground	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.9, 0.01, add)	(0.05, 0.01, add)
Pickground_Allegro	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, add)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.9, 0.01, add)	(0.05, 0.01, add)
Pickrobot	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.05, 0.2, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.9, 0.9, add)	(0.05, 0.01, add)	(0.9, 0.01, add)	(0.05, 0.01, add)
Picksens	(0.05, 0.01, add)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.2, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, add)	(0.05, 0.01, add)
Pickshelves	(0.05, 0.01, add)	(0.9, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.9, add)	(0.05, 0.01, None)	(0.05, 0.01, None)	(0.9, 0.01, add)

In Table 22, for each time series three values are given. The first value corresponds to the tuned α parameter, the second value corresponds to the tuned β parameter, and the last value corresponds to the tuned trend parameter, for each time series.

Table 23: Tuned hyperparameters Holt's-Winters

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0.2, 0.01)	(0.05, 0.2)	(0.9, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.2, 0.2)	(0.05, 0.9)	(0.05, 0.01)	(0.05, 0.9)	(0.05, 0.01)
Pickbelt	(0.9, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.05, 0.01)	(0.9, 0.01)	(0.2, 0.2)	(0.05, 0.2)	(0.05, 0.01)	(0.9, 0.01)	(0.2, 0.01)	(0.2, 0.01)	(0.05, 0.9)
Pickbulk_Above	(0.05, 0.9)	(0.05, 0.2)	(0.05, 0.01)	(0.05, 0.01)	(0.2, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.05, 0.2)	(0.9, 0.01)	(0.05, 0.2)	(0.05, 0.2)	(0.05, 0.01)
Pickbulk_Ground	(0.05, 0.9)	(0.2, 0.2)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.05, 0.2)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.9)	(0.9, 0.01)	(0.2, 0.01)	(0.2, 0.01)	(0.05, 0.01)
Pickconsolidation_C	(0.9, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.01)
Pickconsolidation_K	(0.9, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.05, 0.2)	(0.05, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.2, 0.01)	(0.05, 0.2)	(0.05, 0.01)
Pickground	(0.05, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.2, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.2, 0.01)	(0.05, 0.9)
Pickground_Allegro	(0.05, 0.01)	(0.05, 0.01)	(0.2, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.05, 0.9)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.9)	(0.05, 0.01)	(0.2, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.2)	(0.05, 0.2)
Pickrobot	(0.05, 0.01)	(0.05, 0.2)	(0.2, 0.01)	(0.05, 0.2)	(0.05, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.2)	(0.2, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.2)	(0.9, 0.01)	(0.05, 0.2)
Picksens	(0.05, 0.01)	(0.2, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.2, 0.01)	(0.9, 0.01)	(0.2, 0.01)	(0.05, 0.01)	(0.2, 0.2)	(0.05, 0.2)	(0.9, 0.01)	(0.9, 0.01)	(0.9, 0.01)	(0.05, 0.01)
Pickshelves	(0.05, 0.9)	(0.05, 0.2)	(0.05, 0.2)	(0.2, 0.01)	(0.05, 0.01)	(0.9, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.2, 0.01)	(0.05, 0.2)	(0.05, 0.01)	(0.2, 0.01)	(0.05, 0.01)	(0.05, 0.01)	(0.2, 0.2)

In Table 23, for each time series two values are given. The first value corresponds to the tuned α parameter, and the second value corresponds to the tuned γ parameter, for each time series.

Table 24: Tuned hyperparameters ARIMA

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0, 0)	(1, 1)	(0, 0)	(2, 2)	(1, 1)	(0, 0)	(1, 2)	(1, 2)	(1, 2)	(1, 2)	(1, 2)	(1, 2)	(1, 2)	(0, 0)	(0, 0)
Pickbelt	(1, 2)	(2, 2)	(2, 0)	(2, 2)	(2, 2)	(0, 2)	(2, 0)	(2, 2)	(2, 2)	(2, 2)	(0, 2)	(1, 1)	(2, 1)	(0, 0)	(0, 0)
Pickbulk_Above	(0, 0)	(0, 1)	(0, 0)	(2, 2)	(2, 2)	(1, 1)	(2, 2)	(1, 2)	(1, 2)	(2, 2)	(1, 2)	(1, 1)	(2, 1)	(0, 0)	(0, 0)
Pickbulk_Ground	(0, 0)	(0, 0)	(0, 0)	(2, 2)	(2, 2)	(2, 2)	(2, 2)	(2, 2)	(2, 2)	(2, 2)	(1, 2)	(1, 1)	(0, 0)	(0, 0)	(0, 0)
Pickconsolidation_C	(2, 2)	(1, 2)	(1, 0)	(2, 2)	(2, 2)	(2, 2)	(1, 2)	(1, 2)	(0, 2)	(2, 1)	(0, 2)	(2, 0)	(2, 2)	(0, 0)	(0, 0)
Pickconsolidation_K	(0, 0)	(0, 0)	(0, 1)	(0, 2)	(2, 2)	(1, 2)	(2, 1)	(2, 2)	(2, 2)	(1, 2)	(2, 1)	(1, 1)	(2, 1)	(0, 0)	(0, 0)
Pickground	(0, 0)	(1, 2)	(1, 0)	(2, 2)	(2, 2)	(0, 1)	(2, 1)	(2, 2)	(2, 2)	(1, 2)	(2, 2)	(1, 1)	(2, 2)	(0, 0)	(0, 0)
Pickground_Allegro	(0, 0)	(2, 1)	(1, 2)	(2, 1)	(2, 2)	(2, 2)	(2, 1)	(2, 2)	(2, 2)	(2, 0)	(0, 0)	(2, 1)	(1, 1)	(0, 0)	(0, 0)
Pickrobot	(0, 0)	(2, 1)	(1, 2)	(1, 1)	(2, 1)	(0, 0)	(2, 1)	(0, 0)	(2, 2)	(0, 0)	(0, 0)	(0, 2)	(0, 0)	(0, 0)	(0, 0)
Picksens	(0, 0)	(0, 2)	(0, 0)	(2, 2)	(0, 1)	(2, 2)	(2, 2)	(1, 2)	(2, 2)	(2, 2)	(0, 0)	(2, 2)	(1, 1)	(0, 0)	(0, 0)
Pickshelves	(0, 0)	(2, 2)	(0, 1)	(2, 1)	(0, 2)	(2, 2)	(2, 1)	(1, 2)	(2, 1)	(1, 2)	(1, 0)	(1, 2)	(1, 2)	(0, 0)	(0, 0)

In Table 24, for each time series two values are given. The first value corresponds to the tuned p parameter, and the second value corresponds to the tuned q parameter, for each time series.

Table 25: Tuned hyperparameters SARIMA

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0, 0, 0, 0)	(1, 1, 1, 1)	(0, 1, 1, 1)	(0, 1, 1, 1)	(1, 0, 1, 1)	(0, 0, 0, 0)	(0, 1, 1, 1)	(0, 0, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 0)	(1, 1, 1, 1)	(1, 1, 0, 0)	(0, 1, 0, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickbelt	(0, 1, 0, 0)	(0, 0, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 1, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 0, 0)	(1, 0, 1, 1)	(1, 1, 1, 1)	(0, 0, 1, 1)	(0, 0, 0, 0)
Pickbulk_Above	(0, 0, 0, 0)	(0, 1, 0, 1)	(0, 0, 1, 1)	(1, 0, 1, 1)	(0, 0, 1, 1)	(1, 1, 0, 0)	(1, 0, 1, 1)	(1, 0, 1, 1)	(1, 1, 0, 0)	(0, 1, 0, 0)	(1, 1, 0, 0)	(1, 1, 0, 0)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickbulk_Ground	(0, 0, 0, 0)	(0, 0, 0, 0)	(1, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 0, 1, 1)	(0, 1, 1, 1)	(0, 1, 1, 1)	(1, 1, 0, 1)	(0, 1, 0, 1)	(0, 0, 0, 0)	(1, 1, 0, 0)	(0, 0, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickconsolidation_C	(0, 1, 0, 0)	(0, 0, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 1, 1, 0)	(1, 0, 1, 1)	(0, 0, 1, 1)	(0, 1, 1, 1)	(0, 1, 1, 1)	(1, 1, 0, 1)	(0, 0, 0, 0)	(1, 1, 0, 0)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickconsolidation_K	(0, 0, 0, 0)	(0, 0, 0, 0)	(1, 0, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 0, 1)	(0, 1, 1, 1)	(1, 1, 0, 0)	(0, 1, 1, 1)	(0, 0, 1, 1)	(1, 1, 0, 0)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickground	(0, 0, 0, 0)	(0, 1, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 1, 1, 1)	(0, 1, 0, 0)	(1, 1, 0, 0)	(0, 1, 0, 0)	(0, 0, 1, 1)	(0, 0, 0, 0)
Pickground_Allegro	(0, 0, 0, 0)	(0, 0, 1, 1)	(0, 1, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 0, 1, 1)	(0, 0, 0, 0)	(1, 0, 1, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickrobot	(0, 0, 0, 0)	(0, 1, 1, 1)	(0, 1, 1, 0)	(1, 1, 1, 1)	(0, 0, 1, 1)	(0, 0, 0, 0)	(1, 0, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 1)	(0, 1, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)	(0, 0, 0, 0)
Picksens	(0, 0, 0, 0)	(0, 1, 1, 0)	(1, 0, 1, 1)	(1, 1, 1, 1)	(0, 1, 1, 1)	(0, 0, 0, 0)	(1, 0, 1, 1)	(0, 1, 1, 1)	(0, 1, 1, 1)	(0, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 1, 1)	(0, 1, 0, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickshelves	(0, 0, 0, 0)	(0, 1, 1, 1)	(0, 0, 1, 1)	(0, 0, 1, 1)	(0, 0, 1, 1)	(0, 0, 1, 1)	(1, 0, 1, 1)	(1, 0, 1, 1)	(1, 0, 1, 1)	(0, 0, 1, 1)	(1, 0, 0, 0)	(1, 1, 1, 1)	(0, 1, 1, 1)	(0, 0, 1, 1)	(0, 0, 0, 0)

In Table 25, for each time series four values are given. The first value corresponds to the tuned p parameter, the second value corresponds to the tuned q parameter, the third value corresponds to the tuned P parameter, and the fourth value corresponds to the tuned Q parameter, for each time series.

Table 26: Tuned hyperparameters ARIMAX

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0, 0)	(0, 0)	(1, 0)	(0, 2)	(1, 2)	(1, 2)	(2, 2)	(0, 0)	(2, 2)	(1, 2)	(2, 2)	(1, 1)	(1, 2)	(1, 0)	(1, 0)
Pickbelt	(0, 0)	(1, 2)	(0, 0)	(1, 2)	(2, 1)	(1, 0)	(0, 1)	(1, 2)	(2, 0)	(1, 1)	(1, 2)	(0, 0)	(2, 2)	(0, 0)	(0, 2)
Pickbulk_Above	(0, 0)	(1, 2)	(2, 1)	(2, 2)	(0, 0)	(1, 1)	(1, 2)	(1, 0)	(2, 2)	(0, 1)	(0, 0)	(1, 1)	(1, 2)	(0, 0)	(1, 1)
Pickbulk_Ground	(0, 0)	(0, 1)	(0, 0)	(2, 1)	(0, 1)	(0, 0)	(0, 0)	(1, 2)	(1, 2)	(2, 1)	(0, 1)	(2, 1)	(2, 1)	(1, 2)	(2, 0)
Pickconsolidation_C	(1, 0)	(2, 0)	(0, 0)	(1, 1)	(1, 0)	(1, 2)	(0, 0)	(1, 2)	(0, 2)	(2, 0)	(0, 2)	(1, 1)	(2, 2)	(1, 2)	(0, 0)
Pickconsolidation_K	(2, 1)	(0, 0)	(0, 0)	(0, 0)	(2, 2)	(0, 0)	(2, 1)	(2, 2)	(2, 1)	(2, 2)	(1, 1)	(0, 1)	(1, 1)	(2, 2)	(2, 0)
Pickground	(2, 1)	(1, 2)	(0, 0)	(0, 1)	(1, 2)	(0, 2)	(0, 2)	(1, 2)	(1, 1)	(2, 2)	(1, 2)	(1, 1)	(1, 2)	(0, 0)	(0, 0)
Pickground_Allegro	(0, 0)	(2, 2)	(2, 1)	(2, 2)	(2, 1)	(0, 0)	(1, 2)	(1, 2)	(0, 1)	(2, 1)	(0, 2)	(2, 2)	(2, 1)	(0, 0)	(0, 0)
Pickrobot	(1, 0)	(1, 2)	(0, 2)	(2, 2)	(2, 2)	(0, 0)	(2, 1)	(1, 1)	(2, 2)	(2, 1)	(0, 0)	(0, 1)	(0, 0)	(2, 1)	(0, 2)
Picksens	(0, 1)	(1, 2)	(2, 2)	(2, 2)	(2, 1)	(0, 1)	(0, 0)	(1, 2)	(0, 2)	(1, 2)	(0, 0)	(2, 2)	(0, 0)	(2, 2)	(2, 0)
Pickshelves	(1, 2)	(0, 2)	(2, 2)	(0, 1)	(2, 2)	(1, 2)	(2, 0)	(1, 2)	(1, 1)	(2, 2)	(2, 2)	(2, 1)	(1, 2)	(2, 1)	(0, 0)

In Table 26, for each time series two values are given. The first value corresponds to the tuned p parameter, and the second value corresponds to the tuned q parameter, for each time series.

Table 27: Tuned hyperparameters SARIMAX

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0, 0, 0, 0)	(0, 0, 1, 1)	(0, 1, 1, 0)	(0, 1, 0, 1)	(1, 1, 0, 0)	(0, 0, 1, 0)	(1, 1, 0, 0)	(0, 0, 0, 1)	(1, 1, 0, 0)	(1, 1, 0, 0)	(0, 0, 0, 0)	(0, 1, 1, 0)	(0, 1, 0, 0)	(0, 0, 1, 1)	(1, 0, 0, 0)
Pickbelt	(0, 0, 0, 1)	(1, 1, 1, 1)	(0, 0, 1, 1)	(0, 1, 0, 0)	(1, 1, 0, 0)	(1, 0, 0, 0)	(0, 1, 0, 0)	(1, 1, 0, 0)	(1, 0, 1, 0)	(0, 1, 1, 1)	(0, 1, 1, 0)	(0, 0, 0, 0)	(1, 1, 0, 1)	(0, 0, 0, 0)	(0, 0, 0, 1)
Pickbulk_Above	(0, 0, 0, 0)	(0, 1, 1, 1)	(0, 0, 1, 1)	(0, 1, 0, 0)	(0, 0, 0, 1)	(1, 1, 0, 1)	(0, 1, 1, 1)	(1, 0, 1, 0)	(0, 1, 0, 0)	(0, 1, 0, 0)	(0, 0, 0, 0)	(1, 1, 0, 0)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 1)
Pickbulk_Ground	(0, 0, 0, 0)	(1, 1, 1, 0)	(0, 1, 1, 1)	(0, 1, 1, 1)	(0, 1, 0, 0)	(0, 0, 1, 0)	(0, 0, 0, 0)	(0, 1, 0, 0)	(0, 1, 0, 1)	(1, 1, 0, 0)	(0, 1, 0, 0)	(0, 0, 1, 1)	(1, 0, 0, 1)	(1, 1, 1, 1)	(1, 0, 0, 0)
Pickconsolidation_C	(1, 0, 0, 1)	(1, 0, 1, 0)	(0, 0, 1, 1)	(1, 1, 0, 0)	(1, 0, 1, 0)	(0, 1, 0, 0)	(0, 0, 1, 0)	(1, 0, 0, 0)	(1, 1, 1, 0)	(1, 1, 1, 0)	(0, 1, 0, 1)	(1, 1, 0, 0)	(0, 1, 0, 0)	(0, 1, 1, 0)	(0, 0, 0, 0)
Pickconsolidation_K	(0, 0, 1, 1)	(0, 0, 0, 0)	(0, 0, 1, 0)	(0, 0, 1, 1)	(0, 1, 0, 0)	(0, 0, 0, 0)	(0, 0, 1, 1)	(1, 0, 1, 0)	(1, 1, 0, 0)	(1, 1, 0, 0)	(1, 1, 0, 1)	(0, 1, 0, 0)	(1, 1, 1, 0)	(1, 1, 0, 0)	(1, 0, 0, 0)
Pickground	(0, 0, 0, 0)	(1, 0, 1, 1)	(0, 0, 0, 0)	(0, 1, 0, 1)	(0, 1, 1, 1)	(0, 1, 1, 1)	(0, 1, 1, 0)	(0, 1, 0, 1)	(1, 1, 1, 1)	(1, 1, 1, 1)	(0, 1, 1, 0)	(0, 1, 0, 1)	(0, 1, 0, 0)	(0, 0, 1, 0)	(0, 0, 0, 0)
Pickground_Allegro	(0, 0, 0, 0)	(1, 1, 1, 0)	(1, 1, 1, 1)	(1, 0, 0, 0)	(0, 1, 1, 1)	(0, 0, 0, 0)	(0, 1, 1, 0)	(0, 1, 1, 1)	(0, 1, 0, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 1, 0, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 0, 0, 0)
Pickrobot	(1, 0, 0, 0)	(1, 1, 1, 1)	(1, 1, 1, 1)	(1, 1, 0, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(1, 1, 1, 0)	(1, 0, 0, 1)	(1, 1, 0, 1)	(1, 1, 1, 1)	(0, 0, 0, 0)	(0, 1, 0, 0)	(0, 0, 1, 1)	(1, 1, 1, 1)	(0, 0, 0, 1)
Picksens	(0, 1, 0, 1)	(0, 1, 1, 0)	(1, 1, 0, 1)	(0, 0, 1, 1)	(0, 0, 1, 1)	(1, 0, 1, 0)	(0, 0, 0, 0)	(1, 1, 0, 1)	(0, 1, 0, 1)	(1, 1, 1, 0)	(1, 1, 1, 0)	(1, 0, 1, 1)	(0, 0, 1, 1)	(1, 0, 0, 0)	(1, 0, 0, 0)
Pickshelves	(0, 1, 0, 0)	(0, 1, 0, 1)	(0, 1, 1, 1)	(0, 0, 1, 1)	(1, 1, 1, 1)	(1, 0, 0, 0)	(1, 1, 1, 1)	(0, 0, 1, 0)	(1, 1, 0, 1)	(1, 1, 0, 0)	(1, 1, 0, 0)	(1, 1, 1, 1)	(0, 1, 1, 0)	(0, 0, 1, 0)	(0, 0, 0, 0)

In Table 27, for each time series four values are given. The first value corresponds to the tuned p parameter, the second value corresponds to the tuned q parameter, the third value corresponds to the tuned P parameter, and the fourth value corresponds to the tuned Q parameter, for each time series.

Table 28: Tuned hyperparameters Croston method

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	0.7	0.7	0.01	0.01	0.01	0.01	0.05	0.01	0.01	0.3	0.5	0.05	0.01	0.9	0.01
Pickbelt	0.9	0.05	0.9	0.01	0.01	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Pickbulk_Above	0.9	0.1	0.05	0.01	0.05	0.01	0.05	0.01	0.01	0.05	0.01	0.01	0.05	0.7	0.01
Pickbulk_Ground	0.01	0.05	0.05	0.01	0.05	0.1	0.01	0.01	0.01	0.1	0.3	0.01	0.05	0.01	0.01
Pickconsolidation_C	0.5	0.01	0.9	0.01	0.01	0.01	0.01	0.01	0.1	0.01	0.01	0.05	0.01	0.05	0.5
Pickconsolidation_K	0.3	0.01	0.05	0.01	0.05	0.01	0.1	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.01
Pickground	0.1	0.05	0.05	0.01	0.01	0.05	0.01	0.01	0.01	0.01	0.01	0.05	0.01	0.05	0.9
Pickground_Allegro	0.01	0.05	0.3	0.01	0.01	0.05	0.01	0.01	0.01	0.01	0.7	0.05	0.01	0.01	0.1
Pickrobot	0.01	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.01	0.01	0.01
Picksens	0.01	0.9	0.01	0.01	0.01	0.3	0.01	0.01	0.01	0.05	0.3	0.01	0.3	0.05	0.01
Pickshelves	0.9	0.01	0.05	0.01	0.01	0.05	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.9

In Table 28, for each time series one value is given. This values corresponds to the smoothing parameter, α, for each time series.

Table 29: Tuned hyperparameters MLP without features

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0.39, 0.004243, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.14, 0.004382, 27, 11, 2)	(0.02, 0.002669, 40, 12, 2)	(0.01, 0.00001, 27, 7, 2)	(0.28, 0.004914, 27, 8, 2)	(0.01, 0.005, 10, 12, 2)	(0.5, 0.005, 27, 7, 1)	(0.01, 0.00001, 29, 10, 2)	(0.06, 0.001734, 26, 8, 2)	(0.24, 0.00066, 38, 6, 1)	(0.5, 0.005, 25, 8, 2)	(0.5, 0.005, 33, 14, 2)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)
Pickbelt	(0.06, 0.001734, 26, 8, 2)	(0.05, 0.004008, 38, 18, 2)	(0.05, 0.004995, 41, 12, 2)	(0.02, 0.00232, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.03, 0.000549, 23, 6, 2)	(0.5, 0.005, 31, 11, 1)	(0.28, 0.004914, 27, 8, 2)	(0.08, 0.004255, 28, 9, 2)	(0.05, 0.004008, 38, 18, 2)	(0.28, 0.004914, 27, 8, 2)	(0.05, 0.004008, 38, 18, 2)	(0.46, 0.001107, 27, 11, 2)	(0.5, 0.005, 24, 7, 1)	(0.28, 0.004914, 27, 8, 2)
Pickbulk_Above	(0.47, 0.003685, 33, 9, 2)	(0.08, 0.004255, 28, 9, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.03, 0.001135, 27, 9, 1)	(0.21, 0.000011, 22, 16, 1)	(0.06, 0.003471, 26, 7, 2)	(0.05, 0.004008, 38, 18, 2)	(0.02, 0.00232, 26, 8, 2)	(0.06, 0.003377, 25, 6, 1)	(0.12, 0.004421, 10, 7, 2)	(0.28, 0.003588, 37, 16, 2)	(0.21, 0.000011, 22, 16, 1)	(0.15, 0.000889, 33, 16, 1)
Pickbulk_Ground	(0.41, 0.002219, 18, 5, 1)	(0.01, 0.003649, 27, 6, 2)	(0.05, 0.004008, 38, 18, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.04, 0.00269, 26, 8, 1)	(0.08, 0.004255, 28, 9, 2)	(0.2, 0.004245, 33, 19, 1)	(0.05, 0.004008, 38, 18, 2)	(0.5, 0.005, 33, 5, 2)	(0.03, 0.000415, 28, 9, 1)	(0.49, 0.003478, 25, 8, 2)	(0.39, 0.004243, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)
Pickconsolidation_C	(0.5, 0.005, 40, 10, 2)	(0.5, 0.005, 19, 9, 1)	(0.18, 0.004508, 26, 8, 2)	(0.26, 0.000271, 26, 6, 2)	(0.06, 0.001734, 26, 8, 2)	(0.39, 0.004243, 26, 8, 2)	(0.46, 0.002651, 26, 9, 2)	(0.16, 0.002935, 40, 12, 1)	(0.16, 0.002935, 40, 12, 1)	(0.05, 0.004008, 38, 18, 2)	(0.06, 0.001734, 26, 8, 2)	(0.39, 0.003664, 25, 9, 2)	(0.33, 0.001313, 35, 13, 1)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)
Pickconsolidation_K	(0.45, 0.003772, 41, 9, 2)	(0.46, 0.004502, 23, 15, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.01, 0.00001, 20, 9, 2)	(0.08, 0.004255, 28, 9, 2)	(0.01, 0.00001, 34, 11, 2)	(0.21, 0.000011, 22, 16, 1)	(0.33, 0.002581, 22, 6, 1)	(0.01, 0.00001, 40, 16, 2)	(0.46, 0.001391, 24, 9, 2)	(0.21, 0.000011, 22, 16, 1)	(0.01, 0.005, 30, 16, 2)	(0.21, 0.000011, 22, 16, 1)
Pickground	(0.5, 0.004924, 47, 17, 1)	(0.16, 0.002935, 40, 12, 1)	(0.18, 0.003586, 26, 8, 2)	(0.24, 0.00066, 38, 6, 1)	(0.08, 0.004255, 28, 9, 2)	(0.39, 0.003664, 25, 9, 2)	(0.5, 0.005, 37, 11, 1)	(0.08, 0.004255, 28, 9, 2)	(0.08, 0.004255, 28, 9, 2)	(0.01, 0.00001, 20, 16, 2)	(0.21, 0.000011, 22, 16, 1)	(0.2, 0.004245, 33, 19, 1)	(0.08, 0.004255, 28, 9, 2)	(0.37, 0.000488, 37, 15, 2)	(0.21, 0.000011, 22, 16, 1)
Pickground_Allegro	(0.21, 0.000011, 22, 16, 1)	(0.16, 0.002935, 40, 12, 1)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.01, 0.00001, 20, 20, 2)	(0.5, 0.005, 34, 5, 2)	(0.06, 0.001734, 26, 8, 2)	(0.08, 0.004255, 28, 9, 2)	(0.07, 0.003822, 40, 12, 1)	(0.39, 0.004243, 26, 8, 2)	(0.39, 0.004243, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.39, 0.004243, 26, 8, 2)	(0.28, 0.004914, 27, 8, 2)	(0.5, 0.005, 27, 7, 2)
Pickrobot	(0.21, 0.000011, 22, 16, 1)	(0.05, 0.004008, 38, 18, 2)	(0.39, 0.004243, 26, 8, 2)	(0.33, 0.001313, 35, 13, 1)	(0.2, 0.004245, 33, 19, 1)	(0.21, 0.000011, 22, 16, 1)	(0.27, 0.002171, 21, 5, 2)	(0.5, 0.005, 39, 14, 2)	(0.32, 0.001347, 22, 16, 1)	(0.07, 0.003822, 40, 12, 1)	(0.21, 0.000011, 22, 16, 1)	(0.47, 0.000023, 26, 9, 2)	(0.21, 0.000011, 22, 16, 1)	(0.34, 0.000199, 25, 8, 2)	(0.5, 0.005, 10, 5, 2)
Picksens	(0.39, 0.004243, 26, 8, 2)	(0.24, 0.00066, 38, 6, 1)	(0.08, 0.004255, 28, 9, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 23, 9, 2)	(0.5, 0.005, 32, 15, 2)	(0.12, 0.004421, 10, 7, 2)	(0.08, 0.004255, 28, 9, 2)	(0.05, 0.004008, 38, 18, 2)	(0.03, 0.000414, 28, 7, 2)	(0.5, 0.004723, 41, 17, 1)	(0.28, 0.004914, 27, 8, 2)	(0.33, 0.001313, 35, 13, 1)	(0.21, 0.000011, 22, 16, 1)
Pickshelves	(0.31, 0.001869, 25, 5, 2)	(0.21, 0.000011, 22, 16, 1)	(0.16, 0.002935, 40, 12, 1)	(0.24, 0.00066, 38, 6, 1)	(0.24, 0.00066, 38, 6, 1)	(0.29, 0.001641, 50, 20, 2)	(0.46, 0.000774, 24, 9, 2)	(0.08, 0.004255, 28, 9, 2)	(0.21, 0.000011, 22, 16, 1)	(0.01, 0.002024, 27, 7, 2)	(0.32, 0.001347, 22, 16, 1)	(0.2, 0.004245, 33, 19, 1)	(0.39, 0.004243, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.5, 0.005, 22, 5, 2)

In Table 29, for each time series five values are given. The first value corresponds to the tuned dropout parameter, the second value corresponds to the tuned learning rate parameter, the third value corresponds to the tuned number of neurons, the fourth value corresponds to the tuned number of epochs, and the fifth value corresponds to the tuned number of layers in the LSTM, for each time series.

Table 30: Tuned hyperparameters MLP with selected features

	0					1					2				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0.41, 0.001471, 19, 5, 2)	(0.5, 0.005, 42, 20, 1)	(0.38, 0.001711, 15, 5, 2)	(0.21, 0.000011, 22, 16, 1)	(0.05, 0.004008, 38, 18, 2)	(0.5, 0.005, 39, 11, 2)	(0.39, 0.004243, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.42, 0.000317, 27, 7, 1)	(0.5, 0.005, 48, 18, 2)	(0.39, 0.004243, 26, 8, 2)	(0.39, 0.004243, 26, 8, 2)	(0.5, 0.005, 41, 20, 2)	(0.3, 0.000412, 46, 5, 1)	(0.5, 0.005, 40, 20, 2)
Pickbelt	(0.02, 0.001305, 50, 15, 2)	(0.05, 0.004008, 38, 18, 2)	(0.21, 0.000011, 22, 16, 1)	(0.24, 0.00066, 38, 6, 1)	(0.24, 0.00066, 38, 6, 1)	(0.5, 0.005, 42, 9, 1)	(0.39, 0.004243, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 15, 5, 1)	(0.21, 0.000011, 22, 16, 1)	(0.44, 0.005, 43, 16, 2)	(0.01, 0.005, 25, 8, 2)	(0.11, 0.003317, 44, 17, 2)
Pickbulk_Above	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.06, 0.001734, 26, 8, 2)	(0.41, 0.002948, 45, 8, 1)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.27, 0.001921, 49, 12, 2)	(0.17, 0.003672, 10, 6, 1)
Pickbulk_Ground	(0.47, 0.005, 31, 8, 1)	(0.5, 0.005, 33, 5, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.1, 0.004926, 28, 10, 2)	(0.24, 0.00066, 38, 6, 1)	(0.24, 0.00066, 38, 6, 1)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 42, 15, 2)	(0.5, 0.005, 10, 5, 2)	(0.06, 0.001734, 26, 8, 2)	(0.2, 0.002057, 50, 10, 1)	(0.3, 0.000412, 46, 5, 1)	(0.5, 0.005, 10, 20, 2)
Pickconsolidation_C	(0.01, 0.005, 36, 20, 1)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.06, 0.001734, 26, 8, 2)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 33, 5, 2)	(0.32, 0.001347, 22, 16, 1)	(0.05, 0.004008, 38, 18, 2)	(0.08, 0.004255, 28, 9, 2)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 39, 19, 2)	(0.5, 0.005, 35, 20, 2)	(0.26, 0.001346, 32, 8, 1)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)
Pickconsolidation_K	(0.01, 0.005, 26, 20, 2)	(0.01, 0.00001, 10, 5, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 33, 5, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.39, 0.004243, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.22, 0.00025, 24, 7, 2)	(0.3, 0.000412, 46, 5, 1)	(0.5, 0.005, 34, 5, 2)	(0.5, 0.005, 10, 20, 2)
Pickground	(0.5, 0.004449, 50, 20, 2)	(0.5, 0.005, 33, 5, 2)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.03, 0.004559, 50, 20, 1)	(0.5, 0.005, 23, 9, 2)	(0.06, 0.001734, 26, 8, 2)	(0.02, 0.00232, 26, 8, 2)	(0.03, 0.000024, 10, 19, 2)	(0.5, 0.005, 43, 18, 2)	(0.06, 0.001734, 26, 8, 2)	(0.01, 0.005, 10, 17, 2)	(0.39, 0.004243, 26, 8, 2)	(0.5, 0.005, 49, 5, 1)
Pickground_Allegro	(0.5, 0.005, 10, 20, 2)	(0.05, 0.004008, 38, 18, 2)	(0.21, 0.000011, 22, 16, 1)	(0.24, 0.00066, 38, 6, 1)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 10, 20, 2)	(0.5, 0.005, 32, 12, 2)	(0.39, 0.004243, 26, 8, 2)	(0.05, 0.004008, 38, 18, 2)	(0.49, 0.004112, 33, 10, 1)	(0.45, 0.004037, 32, 11, 2)	(0.5, 0.005, 35, 5, 2)	(0.39, 0.004243, 26, 8, 2)	(0.06, 0.001734, 26, 8, 2)	(0.4, 0.000165, 42, 8, 2)
Pickrobot	(0.46, 0.005, 10, 13, 2)	(0.05, 0.004008, 38, 18, 2)	(0.01, 0.005, 50, 12, 2)	(0.05, 0.004008, 38, 18, 2)	(0.24, 0.00066, 38, 6, 1)	(0.32, 0.00311, 39, 20, 2)	(0.06, 0.001734, 26, 8, 2)	(0.5, 0.002645, 31, 14, 1)	(0.08, 0.004255, 28, 9, 2)	(0.05, 0.004008, 38, 18, 2)	(0.13, 0.004288, 50, 20, 2)	(0.21, 0.000011, 22, 16, 1)	(0.42, 0.004112, 50, 20, 1)	(0.42, 0.000317, 27, 7, 1)	(0.5, 0.005, 10, 5, 2)
Picksens	(0.14, 0.00128, 50, 18, 2)	(0.5, 0.005, 38, 20, 2)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 38, 10, 2)	(0.5, 0.005, 33, 5, 2)	(0.39, 0.004243, 26, 8, 2)	(0.18, 0.001878, 50, 20, 1)	(0.21, 0.000011, 22, 16, 1)	(0.5, 0.005, 33, 5, 2)	(0.41, 0.001218, 27, 5, 2)	(0.5, 0.005, 34, 5, 2)	(0.06, 0.001734, 26, 8, 2)	(0.18, 0.001878, 50, 20, 1)
Pickshelves	(0.41, 0.001471, 19, 5, 2)	(0.05, 0.004008, 38, 18, 2)	(0.21, 0.000011, 22, 16, 1)	(0.24, 0.00066, 38, 6, 1)	(0.24, 0.00066, 38, 6, 1)	(0.32, 0.002172, 41, 20, 1)	(0.24, 0.00066, 38, 6, 1)	(0.24, 0.00066, 38, 6, 1)	(0.01, 0.005, 29, 18, 2)	(0.24, 0.000869, 23, 6, 2)	(0.5, 0.001431, 38, 14, 2)	(0.43, 0.003421, 48, 19, 2)	(0.28, 0.002757, 50, 16, 1)	(0.24, 0.004841, 50, 13, 1)	(0.06, 0.001734, 26, 8, 2)

In Table 30, for each time series five values are given. The first value corresponds to the tuned dropout parameter, the second value corresponds to the tuned learning rate parameter, the third value corresponds to the tuned number of neurons, the fourth value corresponds to the tuned number of epochs, and the fifth value corresponds to the tuned number of layers in the LSTM, for each time series.

Table 31: Tuned hyperparameters LSTM without features

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.06, 0.00001, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.43, 0.00001, 25, 5, 2)	(0.16, 0.000011, 40, 7, 1)	(0.09, 0.000011, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)
Pickbelt	(0.37, 0.00001, 37, 8, 2)	(0.16, 0.000011, 40, 7, 1)	(0.5, 0.000011, 24, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.37, 0.00001, 37, 8, 2)	(0.32, 0.00001, 22, 9, 1)	(0.5, 0.000011, 23, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.09, 0.000011, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)
Pickbulk_Above	(0.16, 0.000011, 25, 6, 1)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.37, 0.00001, 37, 8, 2)	(0.09, 0.000011, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.23, 0.000011, 26, 6, 2)
Pickbulk_Ground	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.5, 0.000011, 23, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.01, 0.000011, 28, 5, 1)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)
Pickconsolidation_C	(0.16, 0.000011, 40, 7, 1)	(0.37, 0.00001, 37, 8, 2)	(0.32, 0.00001, 22, 9, 1)	(0.11, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.5, 0.000011, 26, 10, 1)	(0.09, 0.000011, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.22, 0.000011, 26, 5, 1)
Pickconsolidation_K	(0.43, 0.00001, 22, 8, 1)	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.01, 0.00001, 43, 9, 2)	(0.32, 0.00001, 22, 9, 1)	(0.22, 0.000011, 26, 6, 1)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)
Pickground	(0.19, 0.00001, 22, 9, 1)	(0.01, 0.000011, 20, 10, 2)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.5, 0.000011, 25, 10, 1)	(0.06, 0.00001, 26, 6, 2)	(0.01, 0.00001, 21, 10, 2)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.1, 0.00001, 22, 9, 2)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.00001, 27, 8, 2)	(0.09, 0.000011, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)
Pickground_Allegro	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.5, 0.000011, 24, 9, 1)	(0.01, 0.00001, 24, 10, 2)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)
Pickrobot	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.16, 0.00001, 26, 5, 1)	(0.16, 0.000011, 40, 7, 1)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.24, 0.00001, 26, 6, 1)	(0.16, 0.000011, 40, 7, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.01, 0.00001, 24, 5, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)
Picksens	(0.09, 0.000011, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.5, 0.000011, 18, 10, 1)	(0.37, 0.00001, 37, 8, 2)	(0.37, 0.00001, 37, 8, 2)	(0.13, 0.00001, 26, 6, 1)	(0.06, 0.00001, 26, 6, 2)	(0.15, 0.00001, 26, 6, 1)	(0.31, 0.00001, 24, 5, 1)	(0.21, 0.00001, 22, 9, 1)
Pickshelves	(0.44, 0.000011, 29, 5, 2)	(0.43, 0.00001, 22, 8, 1)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 21, 9, 2)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.00001, 23, 9, 2)	(0.37, 0.00001, 37, 8, 2)	(0.5, 0.00001, 24, 9, 1)	(0.24, 0.00001, 22, 8, 1)	(0.09, 0.000011, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)

In Table 31, for each time series five values are given. The first value corresponds to the tuned dropout parameter, the second value corresponds to the tuned learning rate parameter, the third value corresponds to the tuned number of neurons, the fourth value corresponds to the tuned number of epochs, and the fifth value corresponds to the tuned number of layers in the LSTM, for each time series.

Table 32: Tuned hyperparameters LSTM with selected features

	0					**1**					**2**				
	4, 8	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**	**4, 8**	**8, 12**	**12, 16**	**16, 20**	**20, 24**
Pickabove	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.5, 0.00001, 24, 9, 1)	(0.01, 0.00001, 22, 9, 1)	(0.09, 0.000011, 26, 6, 2)	(0.23, 0.000011, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)
Pickbelt	(0.23, 0.000011, 26, 6, 2)	(0.37, 0.00001, 37, 8, 2)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.01, 0.000011, 13, 10, 2)	(0.32, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.37, 0.00001, 37, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.000011, 25, 5, 2)
Pickbulk_Above	(0.06, 0.00001, 26, 6, 2)	(0.37, 0.00001, 37, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.13, 0.000011, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.16, 0.000011, 40, 7, 1)	(0.06, 0.00001, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)
Pickbulk_Ground	(0.37, 0.00001, 37, 8, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.16, 0.000011, 40, 7, 1)	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.24, 0.000011, 26, 5, 2)	(0.09, 0.000011, 26, 6, 2)	(0.28, 0.000011, 27, 5, 2)
Pickconsolidation_C	(0.23, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.46, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.16, 0.000011, 40, 7, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.37, 0.00001, 37, 8, 2)	(0.01, 0.00001, 28, 8, 2)
Pickconsolidation_K	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.15, 0.00001, 28, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)
Pickground	(0.06, 0.00001, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.000011, 21, 9, 2)	(0.37, 0.00001, 37, 8, 2)	(0.5, 0.000011, 24, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.09, 0.000011, 26, 6, 2)	(0.21, 0.00001, 22, 9, 1)	(0.09, 0.000011, 26, 6, 2)
Pickground_Allegro	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.5, 0.000011, 22, 9, 1)	(0.32, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.01, 0.000011, 20, 10, 2)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.000011, 20, 10, 2)	(0.16, 0.000011, 40, 7, 1)	(0.01, 0.00001, 26, 7, 2)	(0.17, 0.000011, 31, 5, 1)	(0.09, 0.000011, 26, 6, 2)	(0.01, 0.00001, 29, 7, 2)	(0.09, 0.000011, 26, 6, 2)
Pickrobot	(0.11, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.22, 0.000011, 26, 6, 2)	(0.32, 0.00001, 22, 9, 1)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.00001, 29, 7, 2)	(0.16, 0.000011, 40, 7, 1)	(0.09, 0.000011, 26, 6, 2)	(0.2, 0.00001, 38, 5, 2)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.00001, 27, 8, 2)	(0.09, 0.000011, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.25, 0.00001, 26, 6, 2)
Picksens	(0.05, 0.000011, 26, 6, 2)	(0.37, 0.00001, 37, 8, 2)	(0.06, 0.00001, 26, 6, 2)	(0.37, 0.00001, 37, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.09, 0.000011, 26, 6, 2)	(0.22, 0.000011, 26, 6, 1)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.06, 0.00001, 26, 6, 2)	(0.01, 0.00001, 26, 7, 2)	(0.06, 0.00001, 26, 6, 2)
Pickshelves	(0.01, 0.00001, 28, 8, 2)	(0.21, 0.00001, 22, 9, 1)	(0.01, 0.00001, 22, 8, 1)	(0.16, 0.000011, 40, 7, 1)	(0.21, 0.00001, 22, 9, 1)	(0.37, 0.00001, 37, 8, 2)	(0.16, 0.000011, 40, 7, 1)	(0.37, 0.00001, 37, 8, 2)	(0.01, 0.00001, 20, 10, 2)	(0.21, 0.00001, 22, 9, 1)	(0.06, 0.00001, 26, 6, 2)	(0.16, 0.000011, 40, 7, 1)	(0.37, 0.00001, 37, 8, 2)	(0.37, 0.00001, 37, 8, 2)	(0.26, 0.00001, 26, 5, 2)

In Table 32, for each time series five values are given. The first value corresponds to the tuned dropout parameter, the second value corresponds to the tuned learning rate parameter, the third value corresponds to the tuned number of neurons, the fourth value corresponds to the tuned number of epochs, and the fifth value corresponds to the tuned number of layers in the LSTM, for each time series.

D Extended results of forecasting models

Table 33: Performance metrics of benchmark

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	1.63	6.66	14.82
Zone				
Pickabove	0.83	1.61	5.52	14.53
Pickbelt	4.88	1.68	9.12	18.74
Pickbulk_Above	0.43	1.59	4.77	6.60
Pickbulk_Ground	0.38	1.59	4.85	6.59
Pickconsolidation_C	3.83	1.72	8.77	20.38
Pickconsolidation_K	0.6	1.6	5.16	7.27
Pickground	4.69	1.72	10.73	24.81
Pickground_Allegro	3.63	1.65	8.15	22.4
Pickrobot	1.34	1.60	5.68	8.34
Picksens	0.51	1.59	4.98	7.01
Pickshelves	1.14	1.60	5.49	7.85
Departure bucket				
(0, (4, 8))	0.005	1.60	4.48	6.04
(0, (8, 12))	0.96	1.62	5.64	9.44
(0, (12, 16))	3.49	1.66	8.05	18.56
(0, (16, 20))	12.05	1.73	16.49	36.01
(0, (20, 24))	3.52	1.51	6.87	10.48
(1, (4, 8))	0.17	1.62	4.82	6.67
(1, (8, 12))	4.78	1.67	9.68	21.8
(1, (12, 16))	2.17	1.65	7.22	15.45
(1, (16, 20))	1.99	1.64	6.86	12.20
(1, (20, 24))	0.29	1.61	4.88	6.76
(2, (4, 8))	0.08	1.62	4.86	13.73
(2, (8, 12))	0.66	1.67	5.76	12.07
(2, (12, 16))	0.14	1.65	5.01	7.09
(2, (16, 20))	0.05	1.63	4.71	6.42
(2, (20, 24))	0.001	1.61	4.51	6.10

Table 34: Performance metrics of SES

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.88	3.28	13.62
Zone				
Pickabove	0.83	0.68	1.57	13.25
Pickbelt	4.88	1.19	6.83	17.93
Pickbulk_Above	0.43	0.50	0.61	2.16
Pickbulk_Ground	0.38	0.56	0.81	2.44
Pickconsolidation_C	3.84	1.27	6.91	21.13
Pickconsolidation_K	0.60	0.72	1.36	4.29
Pickground	4.69	1.26	8.41	23.62
Pickground_Allegro	3.63	0.96	4.92	21.08
Pickrobot	1.34	0.72	1.96	5.62
Picksens	0.51	0.56	0.90	3.30
Pickshelves	1.14	0.73	1.79	4.95
Departure bucket				
(0, (4, 8))	0.005	0.07	0.01	0.15
(0, (8, 12))	0.96	0.88	2.24	7.51
(0, (12, 16))	3.49	1.37	6.72	20.76
(0, (16, 20))	12.05	1.53	15.94	34.23
(0, (20, 24))	3.52	1.10	4.71	8.77
(1, (4, 8))	0.17	0.41	0.42	1.79
(1, (8, 12))	4.78	1.19	7.03	20.73
(1, (12, 16))	2.17	1.09	4.20	14.12
(1, (16, 20))	1.99	1.07	3.95	10.50
(1, (20, 24))	0.29	0.47	0.57	1.82
(2, (4, 8))	0.08	0.28	0.34	12.36
(2, (8, 12))	0.66	0.98	2.62	11.96
(2, (12, 16))	0.14	0.40	0.42	2.50
(2, (16, 20))	0.06	0.06	0.01	0.08
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 35: Performance metrics of Holt's

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.99	5.94	98.3
Zone				
Pickabove	0.83	0.82	2.34	16.43
Pickbelt	4.88	1.28	7.16	18.13
Pickbulk_Above	0.43	0.67	1.48	10.37
Pickbulk_Ground	0.38	0.59	0.81	2.53
Pickconsolidation_C	3.84	1.32	6.94	20.98
Pickconsolidation_K	0.60	0.73	1.33	4.18
Pickground	4.69	1.50	32.03	322.52
Pickground_Allegro	3.63	1.10	6.07	23.88
Pickrobot	1.34	0.84	2.58	9.63
Picksens	0.51	0.59	0.91	3.34
Pickshelves	1.14	0.97	3.66	20.76
Departure bucket				
(0, (4, 8))	0.005	0.07	0.02	0.15
(0, (8, 12))	0.96	0.88	2.10	7.45
(0, (12, 16))	3.49	1.37	6.36	20.14
(0, (16, 20))	12.05	1.63	16.70	35.12
(0, (20, 24))	3.52	1.19	4.85	9.03
(1, (4, 8))	0.17	0.39	0.36	1.76
(1, (8, 12))	4.78	1.35	7.79	21.10
(1, (12, 16))	2.17	1.18	4.46	14.35
(1, (16, 20))	1.99	1.11	4.05	10.61
(1, (20, 24))	0.29	0.52	0.63	1.94
(2, (4, 8))	0.08	0.27	0.33	12.36
(2, (8, 12))	0.66	1.69	40.93	377.14
(2, (12, 16))	0.14	0.44	0.45	2.53
(2, (16, 20))	0.06	0.08	0.01	0.12
(2, (20, 24))	0.001	0.01	0.00	0.01

Table 36: Performance metrics of Holt's-Winters

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	0.64	2.24	10.89
Zone				
Pickabove	0.83	0.62	1.55	13.61
Pickbelt	4.88	0.83	4.57	13.72
Pickbulk_Above	0.43	0.42	0.53	2.09
Pickbulk_Ground	0.38	0.42	0.59	1.95
Pickconsolidation_C	3.83	0.88	4.41	16.87
Pickconsolidation_K	0.60	0.57	1.06	3.96
Pickground	4.69	0.88	5.17	16.12
Pickground_Allegro	3.63	0.70	3.58	18.08
Pickrobot	1.34	0.50	1.38	4.20
Picksens	0.51	0.45	0.68	2.64
Pickshelves	1.14	0.48	1.06	3.28
Departure bucket				
(0, (4, 8))	0.05	0.08	0.02	0.18
(0, (8, 12))	0.96	0.71	1.86	6.91
(0, (12, 16))	3.49	0.74	3.77	13.93
(0, (16, 20))	12.05	0.83	8.36	23.44
(0, (20, 24))	3.52	0.58	2.29	5.18
(1, (4, 8))	0.17	0.45	0.53	3.98
(1, (8, 12))	4.78	1.06	6.57	19.26
(1, (12, 16))	2.17	0.90	3.65	13.57
(1, (16, 20))	1.99	0.73	2.55	8.96
(1, (20, 24))	0.29	0.39	0.46	1.64
(2, (4, 8))	0.08	0.32	0.43	12.72
(2, (8, 12))	0.66	0.97	2.63	11.77
(2, (12, 16))	0.14	0.40	0.43	2.40
(2, (16, 20))	0.05	0.06	0.02	0.07
(2, (20, 24))	0.001	0.01	0.00	0.01

Table 37: Performance metrics of ARIMA

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	0.89	3.23	14.15
Zone				
Pickabove	0.83	0.67	1.38	13.27
Pickbelt	4.88	1.23	7.44	20.61
Pickbulk_Above	0.43	0.46	0.52	2.11
Pickbulk_Ground	0.38	0.52	0.70	2.38
Pickconsolidation_C	3.84	1.16	5.95	20.36
Pickconsolidation_K	0.60	0.71	1.22	4.33
Pickground	4.69	1.27	8.50	23.72
Pickground_Allegro	3.63	1.10	5.13	22.76
Pickrobot	1.34	0.74	1.93	5.74
Picksens	0.51	0.60	0.85	3.40
Pickshelves	1.14	0.87	1.84	5.78
Departure bucket				
(0, (4, 8))	0.05	0.07	0.01	0.15
(0, (8, 12))	0.96	0.84	2.05	7.33
(0, (12, 16))	3.49	1.38	6.61	20.79
(0, (16, 20))	12.05	1.75	17.10	37.03
(0, (20, 24))	3.52	1.19	4.68	8.85
(1, (4, 8))	0.17	0.38	0.35	1.77
(1, (8, 12))	4.78	1.26	7.48	21.82
(1, (12, 16))	2.17	1.05	3.74	14.43
(1, (16, 20))	1.99	1.07	3.86	10.45
(1, (20, 24))	0.29	0.46	0.45	1.85
(2, (4, 8))	0.08	0.26	0.31	12.36
(2, (8, 12))	0.66	0.72	1.38	10.46
(2, (12, 16))	0.14	0.36	0.36	2.49
(2, (16, 20))	0.06	0.04	0.00	0.06
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 38: Performance metrics of SARIMA

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	0.68	2.46	12.04
Zone				
Pickabove	0.83	0.54	1.19	13.21
Pickbelt	4.88	0.92	5.66	17.17
Pickbulk_Above	0.43	0.45	0.53	2.06
Pickbulk_Ground	0.38	0.43	0.59	2.08
Pickconsolidation_C	3.83	0.89	4.55	17.46
Pickconsolidation_K	0.60	0.52	0.93	3.62
Pickground	4.69	1.04	6.42	19.34
Pickground_Allegro	3.63	0.73	3.56	19.25
Pickrobot	1.34	0.59	1.57	5.04
Picksens	0.51	0.42	0.65	2.99
Pickshelves	1.14	0.59	1.42	4.59
Departure bucket				
(0, (4, 8))	0.05	0.07	0.01	0.15
(0, (8, 12))	0.96	0.62	1.54	7.02
(0, (12, 16))	3.49	0.84	3.96	14.43
(0, (16, 20))	12.05	1.21	12.00	29.72
(0, (20, 24))	3.52	0.87	3.51	7.34
(1, (4, 8))	0.17	0.37	0.33	1.77
(1, (8, 12))	4.78	1.10	6.85	20.93
(1, (12, 16))	2.17	0.81	3.17	13.46
(1, (16, 20))	1.99	0.88	3.04	9.39
(1, (20, 24))	0.29	0.39	0.43	1.77
(2, (4, 8))	0.08	0.26	0.31	12.36
(2, (8, 12))	0.66	0.68	1.39	10.42
(2, (12, 16))	0.14	0.37	0.36	2.49
(2, (16, 20))	0.05	0.04	0.00	0.06
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 39: Performance metrics of ARIMAX

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	0.56	1.97	10.33
Zone				
Pickabove	0.83	0.54	1.32	13.10
Pickbelt	4.88	0.71	3.88	12.52
Pickbulk_Above	0.43	0.39	0.48	2.02
Pickbulk_Ground	0.38	0.37	0.54	1.84
Pickconsolidation_C	3.83	0.75	3.76	16.04
Pickconsolidation_K	0.60	0.44	0.78	3.23
Pickground	4.69	0.79	4.62	15.25
Pickground_Allegro	3.63	0.60	3.27	17.45
Pickrobot	1.34	0.46	1.30	4.07
Picksens	0.51	0.40	0.64	2.60
Pickshelves	1.14	0.46	1.03	3.28
Departure bucket				
(0, (4, 8))	0.05	0.07	0.01	0.15
(0, (8, 12))	0.96	0.66	1.70	6.83
(0, (12, 16))	3.49	0.71	3.57	13.12
(0, (16, 20))	12.05	0.76	7.61	22.54
(0, (20, 24))	3.52	0.62	2.36	5.27
(1, (4, 8))	0.17	0.34	0.35	1.71
(1, (8, 12))	4.78	0.94	5.86	18.64
(1, (12, 16))	2.17	0.79	3.10	12.96
(1, (16, 20))	1.99	0.68	2.30	8.08
(1, (20, 24))	0.29	0.35	0.43	1.62
(2, (4, 8))	0.08	0.24	0.33	12.35
(2, (8, 12))	0.66	0.62	1.46	10.13
(2, (12, 16))	0.14	0.33	0.37	2.35
(2, (16, 20))	0.05	0.06	0.03	0.08
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 40: Performance metrics of SARIMAX

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.57	2.02	10.40
Zone				
Pickabove	0.83	0.54	1.32	13.10
Pickbelt	4.88	0.72	4.12	13.17
Pickbulk_Above	0.43	0.39	0.48	1.96
Pickbulk_Ground	0.38	0.37	0.53	1.80
Pickconsolidation_C	3.83	0.75	3.65	15.61
Pickconsolidation_K	0.60	0.54	1.02	4.10
Pickground	4.69	0.79	4.62	15.30
Pickground_Allegro	3.63	0.60	3.26	17.48
Pickrobot	1.34	0.45	1.28	4.05
Picksens	0.51	0.45	0.75	2.95
Pickshelves	1.14	0.53	1.20	3.58
Departure bucket				
(0, (4, 8))	0.005	0.07	0.01	0.15
(0, (8, 12))	0.96	0.64	1.67	6.82
(0, (12, 16))	3.49	0.69	3.25	11.96
(0, (16, 20))	12.05	0.89	8.49	23.44
(0, (20, 24))	3.52	0.62	2.37	5.29
(1, (4, 8))	0.17	0.33	0.34	1.71
(1, (8, 12))	4.78	0.97	6.05	18.86
(1, (12, 16))	2.17	0.82	3.21	13.01
(1, (16, 20))	1.99	0.68	2.29	8.04
(1, (20, 24))	0.29	0.36	0.44	1.62
(2, (4, 8))	0.08	0.24	0.33	12.35
(2, (8, 12))	0.66	0.62	1.46	10.13
(2, (12, 16))	0.14	0.33	0.37	2.35
(2, (16, 20))	0.05	0.06	0.03	0.08
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 41: Performance metrics of Croston method

	Mean tasks	RMSLE	MAE	RMSE
overall				
	2.03	0.86	3.24	13.3
Zone				
Pickabove	0.83	0.70	1.66	13.25
Pickbelt	4.88	1.22	6.95	17.53
Pickbulk_Above	0.43	0.44	0.55	2.06
Pickbulk_Ground	0.38	0.52	0.78	2.34
Pickconsolidation_C	3.83	1.20	6.35	20.03
Pickconsolidation_K	0.60	0.63	1.17	3.90
Pickground	4.69	1.24	8.41	23.08
Pickground_Allegro	3.63	0.98	4.99	20.97
Pickrobot	1.34	0.72	2.01	5.53
Picksens	0.51	0.55	0.91	3.27
Pickshelves	1.14	0.74	1.81	4.91
Departure bucket				
(0, (4, 8))	0.05	0.07	0.02	0.15
(0, (8, 12))	0.96	0.90	2.27	7.37
(0, (12, 16))	3.49	1.31	6.61	19.32
(0, (16, 20))	12.05	1.56	16.1	33.86
(0, (20, 24))	3.52	1.12	4.77	8.65
(1, (4, 8))	0.17	0.39	0.43	1.77
(1, (8, 12))	4.78	1.20	7.06	20.62
(1, (12, 16))	2.17	1.09	4.22	14.07
(1, (16, 20))	1.99	1.08	3.99	10.46
(1, (20, 24))	0.29	0.44	0.56	1.8
(2, (4, 8))	0.08	0.27	0.35	12.36
(2, (8, 12))	0.66	0.75	1.73	10.38
(2, (12, 16))	0.14	0.35	0.38	2.49
(2, (16, 20))	0.05	0.11	0.07	0.13
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 42: Performance metrics of MLP without features

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	0.94	3.35	14.26
Zone				
Pickabove	0.83	0.87	2.01	13.49
Pickbelt	4.88	1.27	6.83	18.75
Pickbulk_Above	0.43	0.51	0.58	2.16
Pickbulk_Ground	0.38	0.61	0.83	2.93
Pickconsolidation_C	3.83	1.22	6.08	20.85
Pickconsolidation_K	0.60	0.71	1.19	4.11
Pickground	4.69	1.35	8.65	25.28
Pickground_Allegro	3.63	1.07	5.60	22.77
Pickrobot	1.34	0.82	2.17	6.3
Picksens	0.51	0.64	0.94	3.48
Pickshelves	1.14	0.84	1.90	5.28
Departure bucket				
(0, (4, 8))	0.05	0.07	0.01	0.15
(0, (8, 12))	0.96	0.95	2.23	7.68
(0, (12, 16))	3.49	1.29	6.06	20.16
(0, (16, 20))	12.05	1.77	17.48	37.66
(0, (20, 24))	3.52	1.32	4.88	9.51
(1, (4, 8))	0.17	0.43	0.42	1.79
(1, (8, 12))	4.78	1.34	7.67	21.23
(1, (12, 16))	2.17	1.17	4.34	15.00
(1, (16, 20))	1.99	1.08	3.67	10.87
(1, (20, 24))	0.29	0.50	0.55	1.86
(2, (4, 8))	0.08	0.46	0.59	12.48
(2, (8, 12))	0.66	0.82	1.76	10.55
(2, (12, 16))	0.14	0.43	0.43	2.56
(2, (16, 20))	0.05	0.18	0.08	0.30
(2, (20, 24))	0.001	0.01	0.00	0.01

Table 43: Performance metrics of MLP with features

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.93	3.22	13.93
Zone				
Pickabove	0.83	0.77	1.68	13.36
Pickbelt	4.88	1.26	6.60	18.22
Pickbulk_Above	0.43	0.52	0.58	2.17
Pickbulk_Ground	0.38	0.59	0.76	2.51
Pickconsolidation_C	3.83	1.24	5.97	20.33
Pickconsolidation_K	0.60	0.70	1.14	4.09
Pickground	4.69	1.33	8.36	24.75
Pickground_Allegro	3.63	1.10	5.42	22.10
Pickrobot	1.34	0.81	2.13	6.57
Picksens	0.51	0.62	0.91	3.43
Pickshelves	1.14	0.84	1.86	5.25
Departure bucket				
(0, (4, 8))	0.05	0.07	0.01	0.15
(0, (8, 12))	0.96	0.91	2.06	7.61
(0, (12, 16))	3.49	1.25	5.31	18.61
(0, (16, 20))	12.05	1.78	16.93	36.77
(0, (20, 24))	3.52	1.30	4.79	9.49
(1, (4, 8))	0.17	0.40	0.38	1.77
(1, (8, 12))	4.78	1.34	7.66	21.21
(1, (12, 16))	2.17	1.17	4.17	14.76
(1, (16, 20))	1.99	1.13	3.89	11.01
(1, (20, 24))	0.29	0.5	0.57	1.84
(2, (4, 8))	0.08	0.29	0.35	12.36
(2, (8, 12))	0.66	0.83	1.76	10.48
(2, (12, 16))	0.14	0.38	0.35	2.51
(2, (16, 20))	0.05	0.14	0.06	0.21
(2, (20, 24))	0.001	0.01	0.00	0.01

Table 44: Performance metrics of LSTM without features

	Mean tasks	**RMSLE**	**MAE**	**RMSE**
Overall				
	2.03	0.92	2.90	14.67
Zone				
Pickabove	0.83	0.67	1.37	13.35
Pickbelt	4.88	1.33	6.36	20.01
Pickbulk_Above	0.43	0.47	0.46	2.15
Pickbulk_Ground	0.38	0.57	0.65	2.53
Pickconsolidation_C	3.83	1.15	5.07	20.72
Pickconsolidation_K	0.60	0.67	0.97	4.15
Pickground	4.69	1.32	7.68	26.43
Pickground_Allegro	3.63	1.11	4.99	23.36
Pickrobot	1.34	0.87	1.95	6.63
Picksens	0.51	0.59	0.78	3.48
Pickshelves	1.14	0.84	1.62	5.58
Departure bucket				
(0, (4, 8))	0.05	0.07	0.01	0.15
(0, (8, 12))	0.96	0.81	1.64	7.56
(0, (12, 16))	3.49	1.12	4.43	19.00
(0, (16, 20))	12.05	1.96	16.50	39.9
(0, (20, 24))	3.52	1.38	4.64	10.21
(1, (4, 8))	0.17	0.33	0.27	1.74
(1, (8, 12))	4.78	1.40	6.95	22.18
(1, (12, 16))	2.17	1.04	3.45	14.78
(1, (16, 20))	1.99	1.06	3.29	11.35
(1, (20, 24))	0.29	0.45	0.43	1.85
(2, (4, 8))	0.08	0.26	0.31	12.36
(2, (8, 12))	0.66	0.67	1.30	10.45
(2, (12, 16))	0.14	0.33	0.26	2.51
(2, (16, 20))	0.05	0.05	0.01	0.06
(2, (20, 24))	0.001	0.00	0.00	0.00

Table 45: Performance metrics of LSTM with features

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.92	2.91	14.67
Zone				
Pickabove	0.83	0.67	1.39	13.35
Pickbelt	4.88	1.32	6.37	20.01
Pickbulk_Above	0.43	0.47	0.46	2.15
Pickbulk_Ground	0.38	0.57	0.66	2.53
Pickconsolidation_C	3.83	1.15	5.09	20.71
Pickconsolidation_K	0.60	0.67	0.98	4.14
Pickground	4.69	1.35	7.70	26.47
Pickground_Allegro	3.63	1.10	5.00	23.33
Pickrobot	1.34	0.88	1.98	6.62
Picksens	0.51	0.60	0.78	3.48
Pickshelves	1.14	0.84	1.63	5.57
Departure bucket				
(0, (4, 8))	0.005	0.09	0.03	0.16
(0, (8, 12))	0.96	0.81	1.65	7.55
(0, (12, 16))	3.49	1.11	4.43	18.99
(0, (16, 20))	12.05	1.97	16.49	39.92
(0, (20, 24))	3.52	1.38	4.65	10.22
(1, (4, 8))	0.17	0.34	0.29	1.74
(1, (8, 12))	4.78	1.40	6.95	22.16
(1, (12, 16))	2.17	1.05	3.48	14.79
(1, (16, 20))	1.99	1.06	3.30	11.34
(1, (20, 24))	0.29	0.45	0.43	1.85
(2, (4, 8))	0.08	0.30	0.36	12.36
(2, (8, 12))	0.66	0.67	1.31	10.45
(2, (12, 16))	0.14	0.34	0.29	2.50
(2, (16, 20))	0.05	0.08	0.03	0.10
(2, (20, 24))	0.001	0.03	0.01	0.03

Table 46: Performance metrics of case where models are applied per zone

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.56	2.06	10.04
Zone				
Pickabove	0.83	0.53	1.30	9.59
Pickbelt	4.88	0.68	3.88	12.31
Pickbulk_Above	0.43	0.40	0.52	1.93
Pickbulk_Ground	0.38	0.38	0.56	1.98
Pickconsolidation_C	3.83	0.74	4.03	17.06
Pickconsolidation_K	0.60	0.43	0.82	3.37
Pickground	4.69	0.84	5.22	16.63
Pickground_Allegro	3.63	0.60	3.22	15.53
Pickrobot	1.34	0.49	1.26	4.05
Picksens	0.51	0.42	0.75	3.03
Pickshelves	1.14	0.45	1.08	3.40
Departure bucket				
(0, (4, 8))	0.005	0.07	0.01	0.15
(0, (8, 12))	0.96	0.68	1.54	5.88
(0, (12, 16))	3.49	0.75	3.67	14.71
(0, (16, 20))	12.05	0.67	7.36	22.19
(0, (20, 24))	3.52	0.62	2.73	7.29
(1, (4, 8))	0.17	0.35	0.41	3.43
(1, (8, 12))	4.78	0.97	6.65	20.25
(1, (12, 16))	2.17	0.80	3.52	12.86
(1, (16, 20))	1.99	0.62	2.25	7.21
(1, (20, 24))	0.29	0.41	0.53	1.98
(2, (4, 8))	0.08	0.21	0.14	0.66
(2, (8, 12))	0.66	0.62	1.36	5.94
(2, (12, 16))	0.14	0.41	0.67	5.93
(2, (16, 20))	0.05	0.07	0.03	0.09
(2, (20, 24))	0.001	0.01	0.00	0.01

Table 47: Performance metrics of case where models are applied per departure bucket

	Mean tasks	RMSLE	MAE	RMSE
Overall				
	2.03	0.55	2.01	9.91
Zone				
Pickabove	0.83	0.54	1.30	9.59
Pickbelt	4.88	0.66	3.77	12.07
Pickbulk_Above	0.43	0.38	0.49	1.85
Pickbulk_Ground	0.38	0.38	0.55	1.93
Pickconsolidation_C	3.83	0.71	3.78	16.43
Pickconsolidation_K	0.60	0.43	0.80	3.37
Pickground	4.69	0.82	5.18	16.62
Pickground_Allegro	3.63	0.59	3.18	15.54
Pickrobot	1.34	0.48	1.22	4.07
Picksens	0.51	0.43	0.76	3.02
Pickshelves	1.14	0.45	1.07	3.37
Departure bucket				
(0, (4, 8))	0.005	0.07	0.01	0.15
(0, (8, 12))	0.96	0.61	1.32	5.59
(0, (12, 16))	3.49	0.74	3.43	13.88
(0, (16, 20))	12.05	0.67	7.36	22.20
(0, (20, 24))	3.52	0.58	2.53	6.53
(1, (4, 8))	0.17	0.35	0.34	3.45
(1, (8, 12))	4.78	0.97	6.65	20.25
(1, (12, 16))	2.17	0.80	3.51	12.86
(1, (16, 20))	1.99	0.62	2.25	7.21
(1, (20, 24))	0.29	0.41	0.53	1.98
(2, (4, 8))	0.08	0.21	0.14	0.66
(2, (8, 12))	0.66	0.62	1.35	5.93
(2, (12, 16))	0.14	0.41	0.67	5.93
(2, (16, 20))	0.05	0.04	0.00	0.07
(2, (20, 24))	0.001	0.01	0.00	0.01

E Models applied per zone or departure bucket

Table 48: Forecast models applied per zone

Zone	**Forecasting model applied**
Pickabove	SARIMAX
Pickbelt	ARIMAX
Pickbulk_Above	SARIMAX
Pickbulk_Ground	SARIMAX
Pickconsolidation_C	SARIMAX
Pickconsolidation_K	ARIMAX
Pickground	SARIMAX
Pickground_Allegro	SARIMAX
Pickrobot	SARIMAX
Picksens	ARIMAX
Pickshelves	ARIMAX

Table 49: Forecast models applied per departure bucket

Departure bucket	**Forecasting model applied**
(0, (4, 8))	ARIMAX
(0, (8, 12))	SARIMA
(0, (12, 16))	SARIMAX
(0, (16, 20))	ARIMAX
(0, (20, 24))	Holt's-Winters
(1, (4, 8))	LSTM without features
(1, (8, 12))	ARIMAX
(1, (12, 16))	ARIMAX
(1, (16, 20))	ARIMAX
(1, (20, 24))	ARIMAX
(2, (4, 8))	ARIMAX
(2, (8, 12))	SARIMAX
(2, (12, 16))	ARIMAX
(2, (16, 20))	ARIMA
(2, (20, 24))	ARIMA