Scientific Paper for the
Interdisciplinary Project (IDP)
at Technische Universität München

# Titel


Betreuer: M. Sc. Alexander Döge
M. Sc. Christian Ruf

Lehrstuhl für Operations Management
Technische Universität München


Studiengang: Informatics Master


Eingereicht von: Johannes Neutze
Haager Strasse 74
85435 Erding
Tel.: +49 157 78948410
Matrikelnummer: 03611960



Eingereicht am: 14.07.2015

The food delivery market is growing very fast and everyone wants to have a piece of it. In order to gain a large share, delivery services have to improve their way of working. Volo has done this step by introducing an improved travelling salesman algorithm to improve the routes for the drivers and thus being able to lower the number of drivers needed. This gives a good advantage by reducing the amount of money spent for the fleet but it still can be improved by predicting time events which the algorithm is not capable of. This interdisciplinary project has the goal to create a reliable forecast for the preparation times of food for restaurants, so the driver can arrive right on time.

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

The food delivery market is a fast growing market with an giant volume. Rocket Internet predicts the market to have a value of 90 billion Euro by 2019. This money attracts many companies which fight for the supremacy in the market. In order to achieve this goal they have to differentiate from their competitors. Some do this by being the cheapest, others have the biggest variety of lower quality restaurants to offer and still others try to optimize the delivery process to improve the quality of service.

An example is is the Munich based startup called VOLO. The idea of VOLO is to provide a great and effective experience in food delivery. Starting with premium restaurants, an appealing online shop and topped with a fast and efficient driver fleet. The fleet is relying on an algorithm making the deliveries itself as fast as possible while being able to serve many orders at once. The algorithm is based on the travelling salesman algorithm and calculates the best routing solution for a number of drivers who have to fulfill a number of deliveries. It assigns each driver deliveries she has to fulfill as well as the best route and order to pick up and delivery orders.. This minimizes empty drives and idle time for the benefit of the driver and the company. The driver is able to earn more money from loaded times and tips and VOLO has to pay less for drivers since the idle times are reduced to a minimum.

This algorithm works optimal from the point in time when the driver receives the order information and starts driving to the restaurant as well as when she leaves the restaurant and starts driving to the customer. The problem is that the food is almost never ready at the time the algorithm sends the driver to the restaurant. It is crucial to pick up the meal at the exact right time. Being late is bad, as the food will be cold or no longer fresh. Being early is bad, as it induces waiting time for the driver as she has to wait for the order to be finished. Forecasting the optimal arrival time gives an advantage over the competitors.

This is why VOLO has the need to create a forecast, predicting preparation time for the food so the driver can be sent to the restaurant just in time.

The following chapters will explain the proceeding to solve the problem. It will start with the resources used to generate knowledge, then focus on the methodology used to forecast and it will finish with evaluating the result and a conclusion.

# 2 Review of Literature and Research

This chapter is about the sources and the information gained before starting the actual forecasting.

## 2.1 Research

In order to get a good overview over the problem, a search in Google Scholar was done. The goal was to find similar problems and approaches to it. Different words were chosen for the search, like "preparation time", "meal", "restaurant" and "forecast". The results did not give the right output for this problem. Most of the forecasting in restaurant was about the amount of staff needed, the amount of meals which will be sold over a period or the size a buffet has to be. These topics did not fit the problem given because the problem stated is pretty unique. There are not many companies who do last mile delivery with time critical materials. Very often the restaurant has its own fleet of driver which is available all the time and they send it when a meal has to be delivered. This has a low rate of management and is easy to implement but it costs money when the driver has nothing to do. Combining multiple restaurants with one delivery fleet and managing the fleet via a routing and timing algorithm is rather new. The time component could be implemented by using a static time for every order or a back channel when the order is placed. The first solution does not give enough time gain while the second requires a lot of infrastructure. This is why a custom approach has to be made in order to optimize the time part.

## 2.2 Approach

After searching for fitting materials, the book "Forecasting: Principles and Practice" by R. Hyndman and A. Athanasopoulos was chosen as a starting point, since it covers the topic of forecasting in general pretty good for starting the job. After reading the book, it was determined to follow the steps the book suggests to create the forecast. The book (Hyndman and Athanasopoulos (2013)) has five basic steps for forecasting which will be explained in the following:

1. Problem Definition
2. Gathering Information

3. Preliminary Analysis
4. Choosing and Fitting Models
5. Using and Evaluating the Forecast

In a first step, the process is mapped out and the Stakeholders and their involvement in the process is determined. They are being consulted on their view of the problem and their needs. For this purpose a SIPOC diagram can be created. With this information a problem statement is defined.
In the second step, the data available for the process is determined. With the variety of restaurants, meals and levels of utilization of staff during the day, it is to be expected there will not be enough statistical data for all scenarios. In order to make up for the little and imperfect data, the expertise of the people who collected this data and who use the forecasts is also taken into account for the forecast.
The third step is the preliminary analysis. It is done to identify patterns, abnormalities and get an overview over the data. Also historic average preparation times are calculated to have a rough estimation for the outcomes of the forecast. The data is put into graphs to have a visual output and to detect invalid inputs, patterns or trends
The fourth step is to choose good models and to fit the processed data set to this models. A model is the way the data is forecast, e.g. the granularity or algorithm used.
The fifth and last step is all about using the models and getting results. The results are then evaluated and discussed how they could fit into the process.

## 2.3 Forecasting Basics

In order to establish a foundation for the forecast, certain key terms are specified.

### 2.3.1 Time Series

Time series are used on events that happen sequentially over time. This method puts events (forecasted or actual) onto a time axis. They are used in the preliminary analysis to identify abnormalities and patterns. They can also be used for better understand of the forecast results but most of the time only the numbers matter. Charting data as a graph often reveals patterns. These patterns are divided into 3 different kinds (Hyndman and Athanasopoulos (2013)). The first pattern is the trend pattern. A trend is a decrease or increase over a long period of time. The second pattern is the season pattern. Seasonal patterns appear when data is influenced by seasonal factors and the effect has a known and fixed time period. The third one is the cyclic pattern. Cyclic pattern influence usually has fluctuations of at least 2 years and a unfixed period of recurrence.

Since the events at VOLO are happening on a time line, time series decomposition was the choice. The available data is transformed to a time series and divided into suitable components, in case patterns are present.

### 2.3.2 Algorithms for Forecasting

The book (Hyndman and Athanasopoulos (2013)) offers different algorithms to forecast on timeseries. Three are algorithms chosen und presented.

#### 2.3.2.1 Moving Average

A classical method is the moving average. It is used to iteratively calculate the next forecast value of a time series. The next values is generated by taking all prior events in account. The calculated values are independent of each other. The formula to calculate the forecast for the event at point t is:

$$ma_{(t)} = \frac{1}{t} \sum_{0}^{t-1} x_{(t-1)} \tag{2.1}$$

Same equation also applies to orders.

#### 2.3.2.2 Weighted Moving Average

Instead of taking all events (as in the moving average), weighted moving average defines a window to be used, i.e. only the last x events or time frame. This window moves according to the current position on the time series. When calculating the next forecast value, the first value is removed from the window respectively when moving to the next day the first day of the frame is removed from the calculation. This way only a specific amount of orders or days stays in the calculation. The weighted moving average for point t is calculated as following:

$$wma_{(t)} = \frac{1}{n} \sum_{t-n-1}^{t-1} x_{(t-1)} \tag{2.2}$$

Same equation also applies to orders.

#### 2.3.2.3 Simple Exponential Smoothing

Another approach is the Simple Exponential Smoothing which returns a smoothed forecast. The forecast value is calculated from the occurrences before. The two comoponents of the forecast value are weighted by the factor $\alpha$ with the constraint 0≤$\alpha \leq 1.\alpha$ specifies the ratio between the weight of the last order's preparation time and the calculated forecast of the values before that. The function is recursive and has no limit for the number of predecessors. The only thing which has to be defined is the starting value $ses_{start}$. Forecasting the value for point t in time is as following:

$$ses_1 = \alpha * x_0 + (1 - \alpha) * ses_{start}$$

$$\text{ses}_t = \alpha * x_{t-1} + (1 - \alpha) * ses_{t-1}(2.3)$$

#### 2.3.2.4 Evaluation Criteria

In order to compare different approaches to forecast, the root mean square error (RMSE) is used. It represents sample standard deviation of the difference of the forecasted and actual value. It sums up the squared error for each calculation and divides the result by the number of calculations. The rooted result is the RMSE.

$$RMSE = \sqrt{\frac{\sum_{x=0}^{n} (y_{forecast} - y_{actual})^2}{x}} \tag{2.4}$$

# 3 Methodology

The following chapter explains how the first four steps of the plan are used in the forecasting task for this Interdisciplinary Project in detail. Step five, the results and evaluation, is discussed in the next chapters.
First of all the specific problem will be defined properly. Next the information needed will be specified and gathered. This data is processed in the preliminary analysis and fitted to models.

## 3.1 Problem Definition

According to the process some general questions have to be solved before the forecast is started. For this purpose a stakeholder analysis is done and a SIPOC diagram is created (Tabular 3.1).

Table 3.1: SIPOC Diagram derived from the five basic steps

| Supplier | Input | Process | Output | Customer |
|---|---|---|---|---|
| Emanuel Pallua (COO)<br>Sebastian Sondheimer (BI)<br>Stefan Rothlehner(CTO)<br>Sergej Krauze (CTO)<br>Operations Team | Knowledge<br>Experience<br>Historic Database | Preliminary Analysis<br>Choosing Models<br>Fitting Models<br>Forecasting<br>Evaluating | Forecast<br>Evaluation<br>Result | Emanuel Pallua<br>Sebastian Sondheimer<br>Operation Team |

Emanuel Pallua is identified as the first stakeholder. He is Chief of Operations at VOLO and the one who requested the forecast. Setting boundaries and demanding his needs makes him source and customer according to the SIPOC diagram. His goal of this forecast is to decrease "waste". Waste is defined as idle times for drivers, meaing being at the restaurant too early, as well as food loosing freshness by lying at the restaurant for too long. Another goal is to save money by maximizing the utilization of drivers and thus decreasing the fleet size. In general he wants a robust forecasting process which can be continuously improved and automated in the future for daily usage.
The second source and customer in one person is Sebastian Sondheimer who is with Business Intelligence. His aim is to optimize the whole process of VOLO, the same as for Emanuel Pallua. He supports the forecast process by adding his knowledge of the current key numbers of VOLO. He suggests using only a few performance factors for the forecast, namely current slot (lunch, afternoon or dinner), current day, weekday, the week before and the weekday in the last month.
On technical side there Stefan Rothlehner and Sergej Krauze, both Chief Technical

Officers, as sources. Stefan Rothlehner is responsible for the backend and delivers the data. The information of the orders is extracted from the PostgreSQl database which is hosted on heroku. Sergej Krauze who is responsible for the Traveling Salesman Algorithm has the requirement that the language the forecast should be written in is Java. Since his work is already in Java and the forecast, once decided that it should be integrated, could be integrated without much additional effort.
The last source and customer is the Operations Team since they are involved in the everyday business. They support the algorithm with their real life scenario knowledge. Their experience is that a 15 minute estimation works reasonably well. In their opinion a decrease of 50% waste would make it worth to implement the algorithm. In order to have a benchmark for the results of the forecast, the error of their current assumptions will be calculated.

**Problem Statement**
The goal is to research a forecast for preparation times of meals. The approach should be simple and easy to implement with the Traveling Salesman Algorithm. Reduction of waste should be significant over the current approach.

## 3.2 Gathering Information

The data for the forecast is taken from the PostgreSQL database of the backend. Every order from the beginning of VOLO in September has been processed and stored in this database. Each order has a timestamps for each step of the delivery process. This way a rich set of historical data has piled up and is available. For the forecast two data samples are downloaded at different points in time, one to create the forecast, the training data set, and another one to compare the results to, the test data set. They were dumped on the 26th of March and the 29th of April and contained 3034 and 4973 orders. The downloaded data sets were saved in comma separated values files, short csv files.
This raw data has some weaknesses. The biggest one is the size of the gathered data. Forecasts need big amounts of information to generate a meaningful result. Since some restaurants have too few orders or a lot of bad data for the preparation times, additional expertise has to be put into the forecast. This expertise was taken from the operation teams. They have to deal with orders on a daily basis and right now have to "forecast" the point in time the driver has to be at the restaurant on their own. In their experience an estimation of around 15 minutes is a more or less accurate guess for most restaurants except for some which are known for their unpredictability.
In order to use the datasets from the csv files, it has to be parsed into objects in the Java program. For this purpose a csv parser library is used. The library reads the csv file and matches the orders from the database to the OrderModel.class of

the code. Not all attributes of the database are used, only the one related to the forecast are picked from the information. Since there is no tracking in the restaurant for the preparation time, it was decided to take the time interval from the point in time at which the restaurant knows from the order until the driver leaves the restaurant. In the process of volo, the printer in the restaurant prints the recipe at the same time the driver accepts the delivery, which is saved in the database as `accepted_at`. The timestamp of the driver leaving the restaurant is saved in `delivery_started_at`. Since it is not the task to figure out the exact the preparation time but the time from when the order is send to the restaurant and when the driver can pick it up, no modification to the time are done.

## 3.3 Preliminary Analysis

There are many factors which can influence the preparation times of the restaurant. Some are of external nature, some internal nature. The weather or season can have an effect on how busy the restaurant is which can cause different cooking times or when a cook is ill the restaurant can be overwhelmed by too many orders.
There are also other factors, like marketing campaigns, which can influence the results, but this has not happened in the recorded time frame.
In order to discover these patterns and get a feeling for the data the preliminary analysis is done.

### 3.3.1 Raw Data Cleanup

First of all the data has be cleaned and a rough overview for the dimensions of the data should be generated. This is done to get a feeling of the time frame a restaurant usually needs to prepare an order. This provides an estimation of what to expect and clue whether the result of a forecast is totally unrealistic or pretty close to what is possible. In addition to the average time analysis a visualization of the data is done as well. This is done in a time series plot on which it is very easy to see anomalies or patterns of the raw data.

As the first step of the raw data cleanup, the training data is put into a time series graph (Fig. 3.1). This analysis reveals problems the data has. The training data set still contains unfinished orders, bugged order, which were finished at a wrong point of time, and orders which have corrupted timestamps. These three types of unusable orders receive a total time of "-1" since either their `accepted_at` timestamp is missing or not readable. In case that the orders `delivery_started_at` timestamp is before the `accepted_at` timestamp a negative value is visible. This events can be observed in many cases in the time series graph and result in an average preparation time of 12 minutes. In order to get realistic results these flaws in the

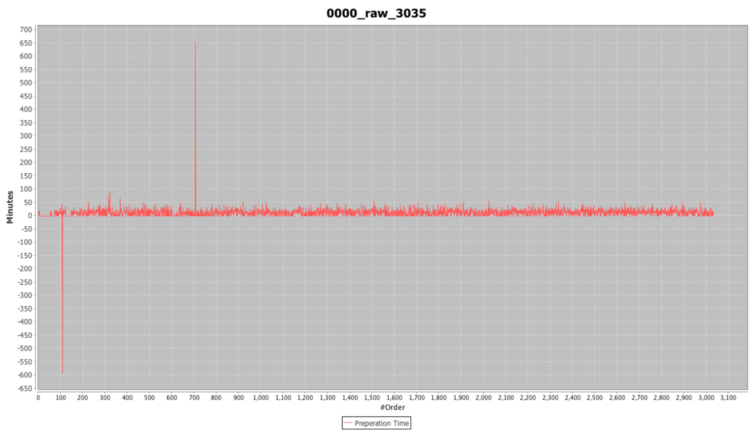


Figure 3.1: Observations of preparation time for each order without any data clean up

data set have to be removed. This is done by ignoring all values which are below 0 minutes in the forecast. This action increases the average preparation time to 16 minutes which is much higher than the first result but free of wrong values. The data set is visualised in Figure 3.2 which, in contrast to Figure 3.1, does not contain any negative values.

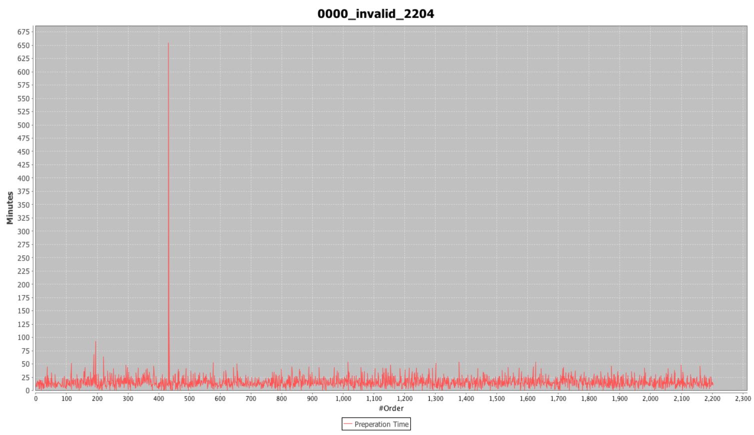


Figure 3.2: Observations of preparation time for each order without invalid values

After removing the obviously unusable orders the data looks a lot more usable. The only problem are the orders which were finished long after they have been started. It can be assumed that these times were either caused by bugged software or human error in the process and remove them from the data set. For this purpose the Grubbs training for outliers is applied to the dataset and all values that seem not to come from a normally distributed population are removed (Engineering Statistics Handbook (2012 (accessed May 21, 2015)). Figure 3.3 shows how this action influences the graph. This results in an average preparation time of 15 minutes which is the same result as the operations team suggest to use as a basis.

Now the usable training dataset is extracted from the input data, it can be analyzed for patterns, trends or similarities. For this purpose a time component has to be

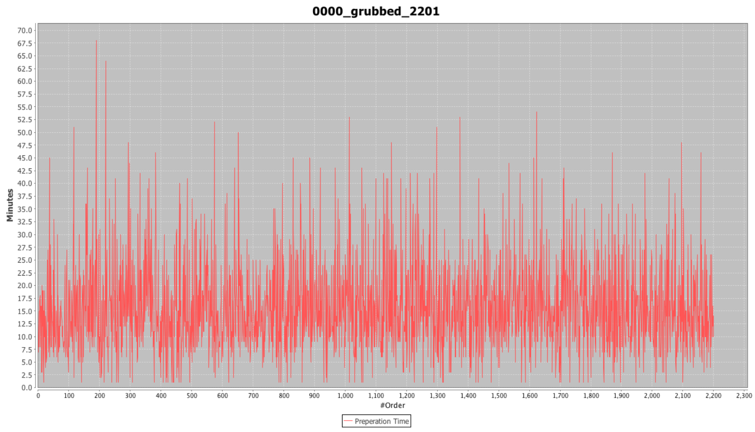


Figure 3.3: Observations of preparation time for each order without invalid values and outliers

introduced into the graph since putting order after order does not include it. That is why orders are summed up by day and visualized by average preparation time per day in Figure 3.4. The only usable information that can be extracted is that the average preparation time varies from day to day.

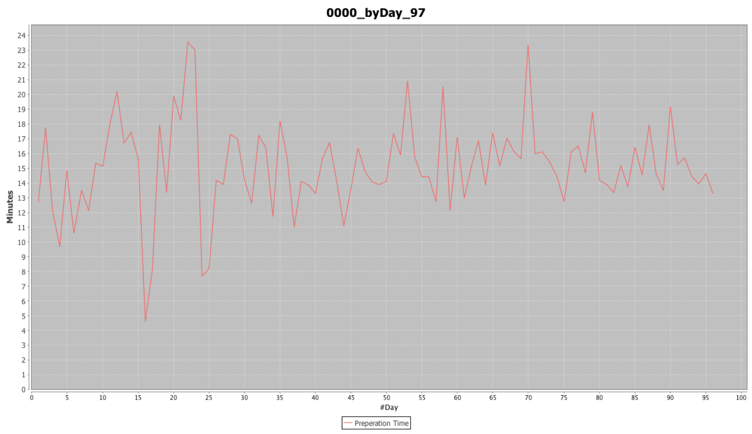


Figure 3.4: Average preparation time per day

Since no clear trend or pattern can be observed in a day by day analysis (Figure 3.4), a box and whisker diagram is created from the data (Figure 3.5). It shows the median and average preparation time as line and point inside the box. The box represents the upper and lower quartile of preparation times in which 50 % of the data is while the red whiskers visualize the area without outliers, which are red circles and a red triangle in case they are out of scale. The purpose of this graph is to give an overview of the set of values of the preparation time and the boundaries most of the orders are located in. The data was divided into different categories.
The first diagram in Figure 3.5 on the left is on week basis. It was created to see if there is a trend over time as the company expertise grows. The only observation is that can be made is that after fluctuating preparation times in the first weeks it gets more constant towards the end.
Since no conclusion can be drawn from this categorization, a box whisker diagram

for each slot was created (Figure 3.5, middle). The day is divided into three slots, the big meals, lunch and dinner, as well as the time between these, the afternoon, which should be not as busy as the meal times. The diagram shows no real difference in preparation times between the slots so this has to be inspected in a different categorization containing slots.
This is done by visualizing the slots for each weekday in order to identify special behaviour since weekdays can have strong differences in terms of load for the restaurant (Figure 3.5, right). This is done since it contains two different key information. Slots and weekdays alone do not have as much information as the two combined. For example on a sunday evening many people like to go to a restaurant and do not order while in the week offices sometimes order big deliveries for lunch. The values in the diagram do not vary that much which can be due to a low training dataset. It could also be separated between weekend and weekdays but there is not enough data to do this and gain additional information.
The difference between slots on weekdays as well as between weekdays is more significant than all other diagrams before and should be considered when creating the model.

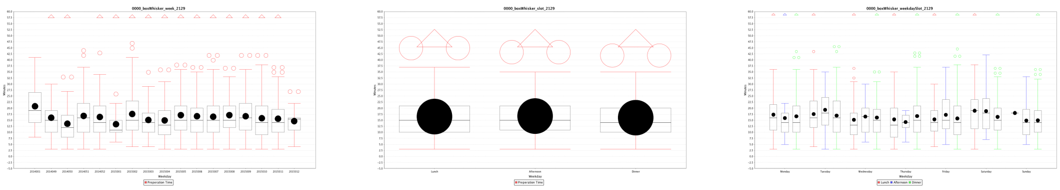

Figure 3.5: Different visualization of the training data set

All these lessons learned from the clean up of the data is also applied to the test data set which is used later.

### 3.3.2 Restaurant Wise Proceeding

The restaurants preparation times can be very different. There are restaurants, which have already pre-cooked ingredients, others who have a very simple way of preparing the meal, like sandwiches, and others are crowded at a certain time and will take longer, preparation times are very different. A pizzeria has other preparation times than a burrito take away.
This suggest distinguishing by restaurant. Initially two categories are created, one restaurant agnostic and another one restaurant specific. For the evaluation of the restaurant specific forecast yum2take is chosen. It is the most mature customer of VOLO.
In order to get a first look at the restaurant specific data, a diagram with slot and weekday differentiation is created in Figure 3.6. The distribution in the diagram for yum2take is clearly different from the overall visualization. This consolidates the assumption that restaurants should be looked at separately. The average times are

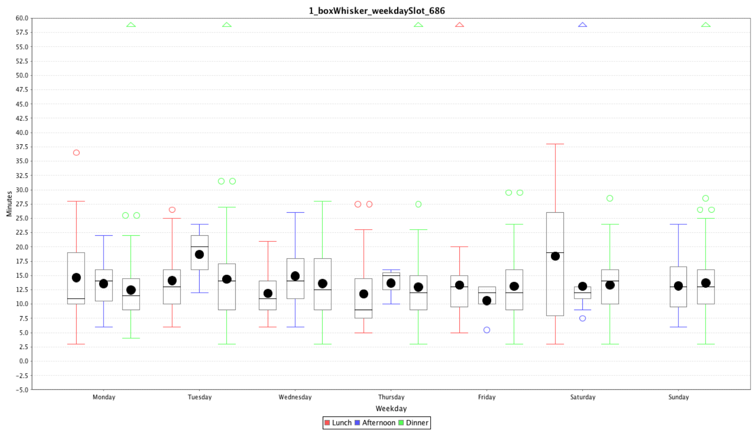


Figure 3.6: Observations of preparation time for orders at yum2take without invalid values and outliers

also different. For the raw data 10.2 minutes are the average, when cleaning the data from invalid values it increased to 13.5 minutes and without the outliers it lowered to 13.1 minutes.

### 3.3.3 Current Assumption

The RMSE of the current forecasting mechanism where every preparation time is supposed to be 15 minutes is calculated from the training data is 8.5 minutes.

## 3.4 Choosing and Fitting Models

After the preliminary analysis it is time to fit the data to models for the forecast. A model combines three dimensions, the algorithm, the restaurant category and a time category.

### 3.4.1 Time Categorizing of Orders

When categorized by time, only orders in a specific timewindow are used to calculate the forecast value for the current order. The time window is always relative to the current order. The following time categories were chosen:

**No time categorization**
No time categorization means that for the forecast of the current order all predecessing orders are taken into account. This is done in order to treat all orders the same and get a simple forecast.
**Day categorization**
When using day categorization, only the orders before the current one on the same day are used to calculate the forecast value. The day categorization is done since

from one day to another day different conditions in the restaurant can change, e.g. staff is ill, which has an impact on that day.

**Week categorization**

The week categorization is done to take larger events in account when forecasting. Examples could be marketing campaigns. In order to consider such events, this categorization uses all predecessing orders of the calendar week for the calculation.

**Weekday categorization**

Weekdays are also different from each other and have to be consideres since they cause different utilization of restaurants, e.g. on sundays markets are closed and people tend to order. For this categorization all orders before the current on this weekday are used for the forecast of the order.

**Single minislot categorization**

The minislot categorization divides the day into half hour slots. For each forecast for an order the orders before the current one in this slot are used. This is done to consider short time fluctuations caused by peak times, e.g. the half hour in which most people do their lunch or shortly before eight in the evening before the prime time movie starts.

**Slot categorization**

In the requirements three main phases, the slot, were decleared. These slots are lunch, dinner and the time in between and they cause different traffic for the restaurants, lunch and dinner being high traffic while the time in between is not so crowded. In order to consider these day specific differences, it can be applied together with the other categorizations, except minislot, to improve the accuracy of the forecast. For example when using day and slot categorization, not all predecessing orders of the current day are considered for the forecast but only the ones in the slot before the current order.

Before the actual forecast can be done, the edge case of not having predecessing orders for the current order, has to be resolved. Since the operations teams suggested a 15 minute basic time for preparation, this time is always taken when no forecast value can be generated for the current order.

### 3.4.2 Combination of Time Categorizations

These categorizations can be combines to create a more complex forecasting model. This model is created by using an algorithm, the restaurant category and multiple time categories. It is calculated with different weightings of the time categories, which is done in 5% steps, e.g. 5% category A, 20% category B and 75% category C. This is done for all possible combinations. In order to get a meaningful forecast, the most important time categories are chosen for the model. There are three categories with two subcategories. One is current slot and current day, since this categories contain information about the current load in the restaurant. Another one is the current slot in the last 7 days and the whole last 7 days. This categories can

give information about small trends, like marketing campaigns. The last categorie is the current slot of the weekday over the last 4 weeks and the weekday itself over the last 4 weeks, which can supply information about general patterns, like sunday is always a crowded day.
This combination is done to consider the behavior of orders depending on their point in time.

# 4 Results

Now that the models are set and run, their RMSE is calculated. The lower the error the more accurate the model. This will be done for the training as well as the test set and the errors will be compared and their difference will be shown.
In order to see whether the forecasting model is an improvement according to the requirements, it is compared with basic approach of the Operation Team.The results are presented following the enumaration of time categories done in Chapter 3.4.2. The forecast is done one time with the time categorization and no slots and one time with slots.
Each table represents a group of models with the same time categorization and use of slots. The algorithm used for the model is on the left while the other two columns represent the restaurant category. Since there are many options for the weighted moving average and simple exponential smoothing, only the lowest error and thus best result from the training set is displayed. This is done since when the forecast would be done, only the present result can be used and thus the lowest error would be chosen.

## 4.1 No time categorization

These are the results for the models which are using no time categorization.

### 4.1.1 Without slot categorization

Tabular 4.1 shows the results for the models which use no time categorization and no slots.

Table 4.1: No time categorization and without slots

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.02755 | 5.950609 | -0.07694 | 8.409876 | 8.565466 | 0.15559 |
| MA (650/2125) | 5.738142 | 5.880966 | 0.142823 | 4.542798 | 8.976687 | 4.433889 |
| SES (0.1/0.1) | 6.116109 | 5.973192 | -0.14291 | 8.560794 | 8.65649 | 0.095695 |

**No restaurant categorization**
The RMSE for the moving average model for this categorization is 8.4 minutes. When comparing the training to the test set the accuracy decreases by 0.15 minutes. The weighted moving average, the second type of model, has different models with

different weights which range from 25 orders to 2125 orders. Most of the resulting RMSE are between 7.9 minutes and 8.5 minutes. There is one outlier with the weight of 2100orders and 2125 orders and 6.7 minutes and 4.5 minutes. All models lose accuracy when compared to their test sets especially the outliers since they rise to the same level as the other models are.
The simple exponential smoothing has the best result with 8.6 minutes for an $\alpha$ of 0.1 and rises up to 11.8 minutes for an $\alpha$ of 1. Except for the outliers, the biggest improvement over the basic approach is around 7% when using the weighted moving average with a weight of 1850.

**Restaurant categorization using yum2take**
The forecasts results in a RMSE of 6 minutes for the moving average model. The weighted moving average is calculated in different weights. The weights start with 25 orders and increase step by step to 675 orders. The RMSE of these forecasts is stable between 6 minutes and 6.5 minutes and the best result is achieved with a weight of 675 orders. The best RMSE for the simple exponential smoothing is the result of an $\alpha$ of 0.1. When increasing $\alpha$ the RMSE also increases to up to 8.5 minutes of RMSE. The accuracy improves for all models when using the test set by 0.07 minutes for the moving average, between 0.07 minutes up to 0.45 minutes when using the weighted version and between 0.14 minutes and 0.42 minutes for the simple exponential smoothing.
When using the restaurant categorization for this time categorization, an improvement up to 30% for the moving average can be observed.

### 4.1.2 With slot categorization

The results of the models without time categorization but with slots are shown in Tabular 4.2.

Table 4.2: No time categorization with slots

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 5.980634 | 5.943105 | -0.03752 | 8.203962 | 8.514927 | 0.310964 |
| MA (25/100) | 6.165134 | 6.017991 | -0.14714 | 7.957781 | 8.492864 | 0.535082 |
| SES (0.1/0.1) | 6.065397 | 5.982731 | -0.08266 | 8.379257 | 8.618189 | 8.618189 |

**No restaurant categorization**
Slot categorizations has an RMSE of 8.2 minutes for the moving average case. The weighted moving average is about the same with some results around 8 minutes (e.g. weight of 100 orders, 7.95 minutes). The weights for this algorithm are from 25 orders up to 225 orders in 25 orders steps. This categorization models with the simple exponential smoothing algorithm start from 8.4 minutes and an $\alpha$ of 0.1 and continue to 11.3 minutes when $\alpha$ reaches 1. For all models the accuracy decreases when the test set is used. They decline between 0.25 minutes and 0.54 minutes.

The best result for this time categorization has the weighted moving average with a weight of 100. An error of 7.95 minutes results in an improvement of around 6%.

**Restaurant categorization using yum2take**

The results for yum2take the moving average results in a RMSE of little under 6 minutes. The weighted average is used with 25 and 50 order steps from which the 25 orders scores better with 6.2 minutes while 50 orders has 6.3 minutes as RMSE. The simple exponential smoothing also has a good results with close to 6 minutes for an $\alpha$ of 0.1. When increasing $\alpha$ the RMSE grows to almost 8 minutes. The accuracy between training and test set is almost zero for the moving average algorithm, improves by 0.15 minutes for the weighted moving average and also improves for all simple exponential smoothing models with a range from 0.08 minutes up to 0.47 minutes while decreasing the accuracy.
When using the moving average an improvement of almost 30% can be achieved in contrast to the basic approach.

## 4.2 Day categorization

These are the models using day categorization and their results.

### 4.2.1 Without slot categorization

The results in Tabular 4.3 are generated for categorization by day. There is no weighted moving average model since there has not been enough data to do forecast in a convenient manner.

Table 4.3: Day categorization without slot

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.011955 | 5.993232 | -0.01872 | 8.403266 | 8.57489 | 0.171623 |
| SES (0.1/0.1) | 6.080652 | 5.995665 | -0.08498 | 8.434747 | 8.57399 | 0.139242 |

**No restaurant categorization**

The moving average model returns a RMSE of 8.4 minutes while the simple exponential smoothing models start with 8.4 minutes for the $\alpha$ of 0.1 and end with 9.8 minutes for the $\alpha$ of 1. The test set is slightly better than the trainings set with 0.17 minutes for the moving average model while the $\alpha$ of 0.1 has the worst decline with 0.13 for the simple exponential smoothing and the other values increase accuracy up to the best improvement of almost 1 minute for the $\alpha$ of 1.
When comparing these results with the basic approach an improvement of only 1% can be achieved by using these time categorization.

**Restaurant categorization using yum2take**

When using day and slot categorization the moving average performes well again, compared to the other models, with an RMSE of 6 minutes. The result improves for the test set by 0.08 minutes. Simple exponential smoothing returned 6.1 minutes for an $\alpha$ of 0.1 and only slowly increased to 6.9 minutes for the maximum of $\alpha$ of 1. The accuracy also improved for up to 0.4 minutes from the training to the test set.
When using restaurant categorization and yum2take, an improvement of 30% for the RMSE can be seen when using moving average.

### 4.2.2 With slot categorization

When using day categorization and slots Tabular 4.4 is the result.

Table 4.4: Day categorization with slot

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.011955 | 5.993232 | -0.01872 | 8.162479 | 8.535424 | 0.372944 |
| MA (15/60) | 6.127325 | 6.109218 | -0.0181 | 7.859824 | 8.737695 | 0.87787 |
| SES (0.1/0.1) | 6.04016 | 6.104262 | 0.064101 | 8.220405 | 8.581225 | 0.360819 |

**No restaurant categorization**

The RMSE for the moving average model is around 8.2 minutes while the weighted moving average range from 7.9 minutes, for the best case and a weight of 60 days, to 8.3 minutes for the weight of 15 days. The simple exponential smoothing gets worse the bigger $\alpha$ becomes. It starts with 8.2 minutes for 0.1 and rises to 10.9 minutes for an $\alpha$ of 1. The accuracy decelines for all moving average variants. The standard algorithm has a decline in accuracy of 0.37 minutes while the weighted ones have inaccuracy grows from 0.24 minutes to 0.88 minutes the bigger the weight gets. For the simple exponential smoothing there is only decline for the $\alpha$ from 0.1 to 0.3, 0.36 minutes, 0.26 minutes and 0.12 minutes. The remaining $\alpha$ steadily improve the accuracy up to 1.28 minutes for the value of 1.
Using the weighted moving average of 60 orders results in an improvement of 7%.

**Restaurant categorization using yum2take**

For yum2take the results are better than for the forecast which uses all orders. The moving average has the best result with about 6 minutes. When weighted it increases to 6.1 minutes, for both weights, namely 15 and 30 orders, which is not much difference to the training set result. Same goes for the simple exponential smoothing, where the RMSE and the difference between the sets grows the bigger $\alpha$ gets. The difference between training set is for all models neglectable because it is around 0.5 minutes, which is too small to have an impact on the real life scenario.
For restaurant categorization and yum2take an improvement of around 30% over the basic approach can be achieved.

## 4.3 Week categorization

The following results are generated by the models using week categorization for the time categorization.

### 4.3.1 Without slot categorization

The results of categorization by week are shown in Tabular 4.5.

Table 4.5: Week categorization without slot

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.074067 | 5.967819 | -0.10624 | 8.402317 | 8.568181 | 0.165863 |
| MA (8/16) | 6.120197 | 5.949482 | -0.17071 | 8.164492 | 8.814434 | |
| SES (0.2/0.2) | 6.008943 | 5.981932 | -0.02701 | 8.399327 | 8.580201 | 0.180873 |

**No restaurant categorization**
The result for the moving average model has a RMSE of 8.4 minutes. When comparing the training data set to the test data set the accuracy lowers by 0.17 minutes. The weighted moving average model with the weight of 4 orders is an outlier with 9.5 minutes. When using a weight of 8, 12 or 16 orders the RMSE lowers from 8.5 minutes to 8.2 minutes. Also when using the test set, the weight of 4 orders is still higher than the expected value with an RMSE of 8.9 minutes. In contrast to this outlier the other models only get more inaccurate, respectively 0.1 minutes, 0.29 minutes and 0.64 minutes for the three weights. The models with the simple exponential smoothing algorithm have all smiliar RMSE, ranging from 8.4 minutes to 8.55 minutes. When compared to the test set the accuracy gets slightly worse but not bigger than 0.21 minutes.
Compared to the basic approach, an improvement of almost 4% can be achieved by using the weighted moving average.

**Restaurant categorization using yum2take**
The moving average model has a results with around 6.1 minute. The weighted moving average forecast models ranges from 6.1 to 6.6 minutes. The worst results come from the smallest (4 orders, 6.4 minutes) and greatest (16 orders, 6.6 minutes) while the weight of 8 and 12 orders has an RMSE of around 6.1 minutes. When using the simple exponential smoothing, the results range from 6.0 minutes for an $\alpha$ of 0.2 and slowly grows up to 6.29 for an $\alpha$ of 1. The difference between training and test data set negative for most cases, meaning the result improves for the test set.
When using moving average, an improvement of 28% over the basic approach can be achieved.

### 4.3.2 With slot categorization

In Tabular 4.6 the results for week and slot categorization are shown.

Table 4.6: Week categorization with slot

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.048462 | 6.013698 | -0.03476 | 8.21474 | 8.532601 | 0.31786 |
| MA (8/12) | 6.137753 | 5.952349 | -0.1854 | 8.133761 | 8.843515 | 0.709754 |
| SES (0.3/0.3) | 5.935083 | 6.155115 | 0.220032 | 8.18856 | 8.615477 | 0.426916 |

**No restaurant categorization**
For the moving average model a RMSE of 8.2 minutes was calculated. The different weighted moving average models, weighted with 4, 8, 12 and 16 orders, are resulting in RMSE of 8.9 minutes, 8.19 minutes, 8.0 minutes, which is the best, and 8.1 minutes. Except for the weight of 4 orders, which increases accuracy by 0.04 minutes, the test set results are worse than the training set ones. The standard moving average model decreases accuracy by 0.32 minutes while the weight of 8 orders results in a decline of 0.31 minutes, 12 orders in 0.67 minutes and 16 orders weight in 0.71 minutes. The models using simple exponential smoothing have all RMSE results of around 8.3 minutes, only an $\alpha$ between 0.2 and 0.6 have a RMSE of 8.2 minutes. The accuracy is lowered by around 0.4 minutes for all models when compared to the test data set.
When using this categorization and the weighted moving average of 12 orders, the RMSE improves by almost 6% over the basic approach.

**Restaurant categorization using yum2take**
The forecast for the models including the categorization of week and slot returned 6.0 minutes for the moving average. The weighted moving average can be calculated for steps of 4, 8 and 12 weeks as weight. It returns 6.3 minutes, 6.1 minutes and 6.2 minutes for the weights. These four results improve their accuracy when forecasting with the test set by between 0.03 and 0.13 minutes. For the simple exponential smoothing an $\alpha$ between 0.2 and 0.5 gives good results of little under 6 minutes and growing for $\alpha$ over 0.5 to a RMSE of 6.45 minutes. The accuracy decreases for all values by about 0.1 to 0.2 minutes except for an $\alpha$ of 1 where it improves by 0.08 minutes.
Compared to the basic approach, using this categorization and the moving average results in an improvement of almost 30%.

## 4.4 Weekday categorization

The following models use weekday categorization with and without slots.

### 4.4.1 Without slot categorization

Weekday categorization results in the following values shown in Tabular 4.7.

Table 4.7: Weekday categorization without slot

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.208329 | 6.07587 | -0.13245 | 8.582432 | 8.660576 | 0.078144 |
| MA (8/12) | 6.181677 | 6.105938 | -0.07573 | 8.08649 | 8.700854 | 0.614363 |
| SES (0.2/0.3) | 6.23239 | 6.113957 | -0.11843 | 8.573267 | 8.683558 | 0.110291 |

**No restaurant categorization**
When using weekdays as categorization, the moving average model 8.6 minutes as RMSE. There can be made 3 models for the weighted moving average using 4, 8 and 12 weeks as weight. The results improve the more weeks are taken as weight. The first weight with 4 weeks has a RMSE of 9.1 minutes, the second one improves to 8.5 minutes while the last one is the best with an RMSE of 8.1 minutes. The simple exponential smoothing model using an $\alpha$ of 0.2 has the best result with 8.6 minutes. All other models with this algorithm range from a little above 8.6 minutes up to 8.9 minutes, increasing with growing $\alpha$. When compared to the test set all results decrease of accuracy. The moving average model decreases by 0.08 minutes while the weighted moving average increases by 0.05 minutes, but decreases then by 0.24 minutes and 0.61 minutes. The simple exponential smoothing models also decrease between 0.06 minutes to 0.12 minutes.
This categorization results in an improvement of around 5% over the basic approach.

**Restaurant categorization using yum2take**
Weekday categorization returns 6.2 minutes for the moving average model. It improves by 0.13 minutes for the test set. The weighted moving average has three different weights, 4, 8 and 12 weekdays in a row. The results are 6.7 minutes, 6.2 minutes and 6.5 minutes and they improve by 0.4 minutes for the first and thirds and by 0.07 for the second when used on the test set. When using simple exponential smoothing for this categorization the results start with 6.3 minutes of RMSE for all $\alpha$ under 0.6 and grow up to 6.9 minutes when $\alpha$ is 1. The accuracy improvement from training to test set is around 0.1 minute and then growns equaly to 0.36 minutes.
When using this categorization an improvement of 27% over the basic approach can be achieved by using the moving average.

### 4.4.2 With slot categorization

The results of weekday categorization and slots is shown in Tabular 4.8.

Table 4.8: Weekday categorization with slot

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.063212 | 6.13718 | 0.073968 | 8.448001 | 8.73583 | 0.287828 |
| MA (12/12) | 4.202483 | 5.802201 | 1.599717 | 8.117922 | 8.913511 | 0.795589 |
| SES (0.3/0.2) | 6.319075 | 6.163683 | -0.15539 | 8.378466 | 8.720113 | |

**No restaurant categorization**

The results for the moving average model is matching other results with around 8.4 minutes. The weights of 4 orders, 8 orders and 12 orders return results of 8.6 minutes, 8.3 minutes and 8.1 minutes. The simple exponential smoothing models are in the range from 8.4 minutes up to 9.6 minutes. Regarding the accuracy when compared to the test set, the moving average model declines by 0.28 minutes while the 4, 8 and 12 orders weight increase by over 0.43 minutes. An increase has the $\alpha$ of 1 with 0.07 minutes while the rest decreases its accuracy with results between 0.02 minutes and 0.34 minutes.
Using this categorization and the best weighted moving average the results is an improvement of around 5% over the basic approach.

**Restaurant categorization using yum2take**

The weekday and slot categorization has an RMSE of 6.1 minutes for the moving average and 6.8 minutes, 6.3 minutes and an outlier of 4.4 minutes for the weighted moving average with an weight of 4, 8 and 12 weeks as weight. The difference between training set and test set is 0.07 minutes of decrease for the moving average and 0.36 minutes of improve for the first two weights of the weighted moving average. The outlier decreases its accuracy by 1.6 minutes to 5.8 minutes which is still a good result. The results for the simple exponential smoothing are around 6.4 minutes for $\alpha$ from 0.1 to 0.6 and growing to 6.77 minutes for bigger $\alpha$. The difference is between $\pm 0.1$ minute for these models.
The results of this categorization, when leaving outliers out, are an increase of 28% over the basic approach.

## 4.5 Mini slot categorization

The results of mini slot categorization are shown in Tabular 4.9. There is no weighted moving average since some slots do not have enough data to calculate this model.

Table 4.9: Mini slot categorization results

| | yum2take | | | all | | |
|---|---|---|---|---|---|---|
| Algorithm | training #686 | test #945 | difference | training #2129 | test #3196 | difference |
| MA | 6.206459 | 6.078068 | -0.12839 | 8.122773 | 8.408003 | 0.28523 |
| SES (0.1/0.2) | 6.347103 | 6.149942 | -0.19716 | 8.670448 | 8.752016 | 0.081567 |

**No restaurant categorization**

For the minislot categorization a RMSE of 8.1 minutes is calculated for the moving average model. The models using simple exponential smoothing range from 8.7 minutes for an $\alpha$ of 0.2 to 11.0 minutes for the maximum $\alpha$. When compared with the test data set, the training set was 0.29 minutes less accurate for the moving average and between 0.1 and 0.3 minutes for the simple exponential alternatives.
When using this categorization an improvement of around 5% over the basic approach can be achieved.

**Restaurant categorization using yum2take**
The moving average model returns 6.2 minutes RMSE and an improvement in accuracy of 0.13 minutes when using the test set. The simple exponential smoothing models start with a RMSE of 6.3 minutes for the $\alpha$ of 0.1 and increase to over 8.0 minutes for an $\alpha$ of 1. The increase in accuracy for all models with this algorithm is below 0.20 minutes.
THis categorization is an increase of 27% over the basic approach when using the moving average algorithm.

## 4.6 Combination of Categorizations

The results for the combined model are treated differently. Since there are over thirty thousand combinations for the weights and algorithm used, only the best results were looked at. These are the 5 best results for the test set including the algorithm used and the weighting of the different time categorizations.

**No restaurant categorization**
When searching for the lowest error on the training set, the result is about 8.947 minutes. It is the result of different combinations. The most occurring one is a mix from 20% of the forecast of the current slot and 25% of the forecast from the current day. The rest is mostly created from the last 7 dayâ™s forecast as well as the slot in the last 4 weeks. These combinations produce the most accurate forecasts. Overall there are only good forecasts generated by the normal moving average. When using the test set to compare the result to the training set the accuracy drops by around 0.06 minutes for the top five models.
When using one of these models the forecast gets worse by around 6% than the basic approach.

Table 4.10: Combination of categorizations results for all orders

| yum2take | Results | | | Weights | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | training | test | difference | slot | | | all orders | | |
| | #2129 orders | #3196 orders | | today | 7 days | 4 weeks | today | 7 days | weekday 4 weeks |
| MA | 8.947472944 | 9.009768419 | 0.062295475 | 20 | 10 | 15 | 25 | 30 | 0 |
| MA | 8.947526044 | 9.009875609 | 0.062349565 | 20 | 5 | 20 | 25 | 30 | 0 |
| MA | 8.947812087 | 9.00973973 | 0.061927643 | 20 | 15 | 10 | 25 | 30 | 0 |
| MA | 8.947971379 | 9.010061298 | 0.062089919 | 20 | 0 | 25 | 25 | 30 | 0 |
| MA | 8.948138981 | 9.009792703 | 0.061653722 | 20 | 10 | 15 | 25 | 25 | 5 |

**Restaurant categorization using yum2take**

The results show that the best combination for yum2take consists of 25% current slotâ™s forecast, 15% of the current day™s forecast and 20% of the forecast of the slot for the predecessing 7 days. The rest of weight is combined from the other forecasts and all forecasts are using moving average as algorithm. This results in an error of about 6.26 minutes. The improvement of the test set over the training set is around 0.15 minutes for all of the five best results.

These models improve the RMSE in contrast to the one of the basic approach by around 26%.

Table 4.11: Combination of categorizations results for yum2take

| all orders | Results | | | Weights | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Algorithm | training | test | difference | slot | | | all orders | | |
| | #686 orders | #945 orders | | today | 7 days | 4 weeks | today | 7 days | weekday 4 weeks |
| MA | 6.265421304 | 6.118386791 | -0.147034513 | 25 | 20 | 15 | 15 | 0 | 25 |
| MA | 6.265551527 | 6.118048866 | -0.147502661 | 25 | 20 | 15 | 15 | 5 | 20 |
| MA | 6.265856228 | 6.117902793 | -0.147953435 | 25 | 15 | 20 | 15 | 0 | 25 |
| MA | 6.265868384 | 6.117507337 | -0.148361047 | 25 | 15 | 20 | 15 | 5 | 20 |
| MA | 6.265946422 | 6.117772134 | -0.148174288 | 25 | 20 | 15 | 15 | 10 | 15 |

# 5 Conclusion

FORCAST GUT ABER WERTE SCHEIÃŸE In general there are no clear patterns for which algorithm, categorie etc

Now that the results of the models are generated a conclusion about the success can be done.

## 5.1 Lessons Learned

Having worked so intensively with data, different learnings have been made:

1. The more data the better
2. Visualize data!
3. Categorize data!
4. Combine learnings!
5. Real life is hard!

First of all, get as much information as possible. A forecast gets more precise the more data you have. There are restaurants which had many orders, like yum2take, and thus could be forecast very well. The RMSE of the test data set for this restaurant were very low since the forecast was more specific in contrast to other restaurants with almost no orders and ridiculous high RMSE.
The second learning is that visualizing the data in the preliminary analysis is always helpful. This way not only obvious bad data, like invalid values, but also things one does not think of from the start, e.g. outliers, can be identified very easily. These invalid values can then be removed and thus cannot cause any trouble in the later process if they have been forgotten from the start.
The third thing which was shown in the process of creating the forecast is that creating models with different time and other constraints helps creating a better forecast. The information gathered from these categorization can be put together in another forecast and combined with machine learning for optimized solution, which is the fourth learning.
The last learning of this task is that forecasting is much more complex than just creating a model. There are many implicit factors which have to be taken into account. But most of all a good knowledge of the company helps a lot to understand needs.

## 5.2 Influence on VOLO

After this forecast was finished, it was presented to VOLO. The conclusion of the presentation was that even though the RMSE was improved step by step the dimension of it was not usable for real life usage since the expertise from the Operation Teams could compete with this.
In addition to the results of this forecasting, there are many factors which were identified over the time by the Operation Teams which cannot be fit into a forecast model that easily. These factors include information like the traffic for different cities, marketing campaigns or the utilization of the driver fleet. A solution for some of these factors is tried to be figured out by a dedicated team at VOLO

## 5.3 Future Tasks

Since some time has passed since this project started and VOLO has grown, the whole forecasting task could be launched again with new sources and customers. Not only the requirements may have changed a bit but also the insights, expertise and amount of data has grown. This new knowledge could be then put into live dispatching for the drivers to validate the results with real life feedback.
The forecast could also be improved by improving the tracking of the restaurants by introducing a back channel.

# Bibliography

Engineering Statistics Handbook, The. 2012 (accessed May 21, 2015). *Grubb's Test for Outliers*. URL `http://www.itl.nist.gov/div898/handbook/eda/section3/eda35h1.htm`.

Hyndman, Rob J., George Athanasopoulos. 2013. *Forecasting: Principles and Practice*. 17th ed. OTexts.

# Appendix

# 4 Appendix One: Consumer Application

Another task for the Interdisciplinary Project was to create a consumer application. The aim of the application is to give the person who just ordered information about the delivery. This includes the current step in the delivery process as well as the current position of the driver on a map. After the process the customer should be able to rate the delivery.

The moment the customer finishes placing an order the backend should create an account for the customer. The credentials for the new user should be send to the customer by email. The email also directs to the application in the Play Store where it can be downloaded. This part has to be implemented in the backend if it should be used in a real scenario.

The user can login into the application (Fig. 5.1, left) by entering the user data. Right now a driver account is used to demonstrate the functionality since the backend does not support the right manage to let users see orders owned by drivers.

When the customer enters the application the screen in the middle of Figure 5.1 is shown. In this screen `assigning_to_driver` is shown as status. This means the driver cannot be tracked yet since the order has not been assigned to a driver. The application queries every 10 seconds for a status change on the server. As soon as the driver has accepted the order the status switches to `pickup_started` (Figure 5.1, right). From now on the position of the driver is updated every 10 seconds so the customer using the application always sees where his delivery is at the moment. The

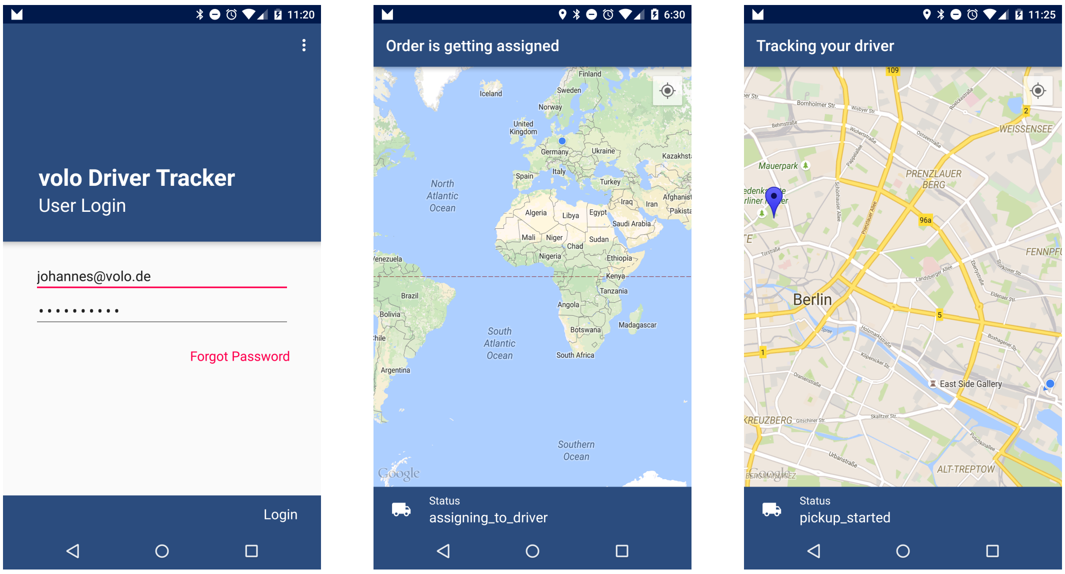


Figure 5.1: The consumer application. Login on the left. Idle status in the middle. Pickup started on the right.

customer sees the different steps of the delivery process, namely `pickup_ended`, when the driver has entered the restaurant, and `delivery_started`, when the driver has left the restaurant with the meal (Figure 5.2, left and middle).

As soon as the driver has completed the delivery by giving the order to the customer and checking it in his driver application, a feedback dialog opens in the customer

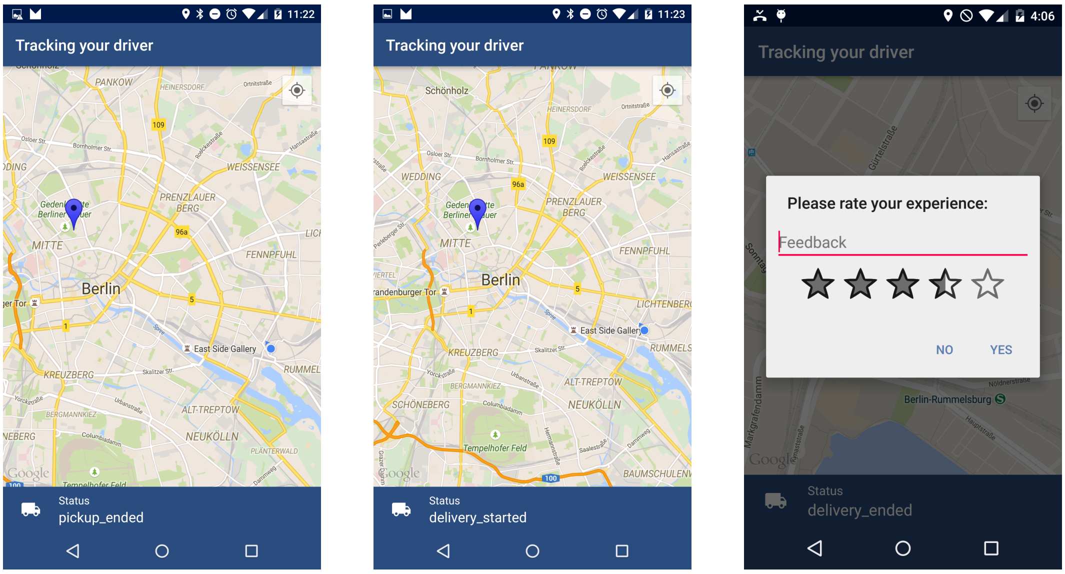


Figure 5.2: The graphical user interface of the consumer application. When the driver is at the restaurant, when the driver has left the restaurant and when the driver has finished the delivery.

application(Figure 5.2, right). In this popup the customer can rate his experience and support valuable feedback for VOLO in case something was not as she wished. After entering the feedback, the application is closed and the user is logged out from the application. The credentials of the user are now set to inactive on the server since there is nothing to track anymore.
This way the customer tracking application provides an easy and comfortable way of tracking an order and giving feedback about the delivery.

# 5 Appendix Two: Code and Results

The code and application will be send by email since they contain confidential material of VOLO UG.

## Ehrenwï¿½rtliche Erklï¿½rung

Ich erklï¿½re hiermit ehrenwï¿½rtlich, dass ich die vorliegende Arbeit selbstï¿½ndig angefertigt habe. Die aus fremden Quellen direkt und indirekt ï¿½bernommenen Gedanken sind als solche kenntlich gemacht.

Ich weiï¿½, dass die Arbeit in digitalisierter Form daraufhin ï¿½berprï¿½ft werden kann, ob unerlaubte Hilfsmittel verwendet wurden und ob es sich - insgesamt oder in Teilen - um ein Plagiat handelt. Zum Vergleich meiner Arbeit mit existierenden Quellen darf sie in eine Datenbank eingestellt werden und nach der ï¿½berprï¿½fung zum Vergleich mit kï¿½nftig eingehenden Arbeiten dort verbleiben. Weitere Vervielfï¿½ltigungs- und Verwertungsrechte werden dadurch nicht eingerï¿½umt. Die Arbeit wurde weder einer anderen Prï¿½fungsbehï¿½rde vorgelegt noch verï¿½ffentlicht.

Ort, Datum

Unterschrift