

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

<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Review of Literature and Research</b>	<b>2</b>
2.1 Research . . . . .	2
2.2 Approach and Forecasting Basics . . . . .	2
2.2.1 Time Series . . . . .	3
2.2.2 Algorithms for Forecasting . . . . .	4
2.2.2.1 Moving Average . . . . .	4
2.2.2.2 Weighted Moving Average . . . . .	4
2.2.2.3 Simple Exponential Smoothing . . . . .	4
<b>3 Methodology</b>	<b>6</b>
3.1 Problem Definition . . . . .	6
3.2 Gathering Information . . . . .	7
3.3 Preliminary Analysis . . . . .	8
3.3.1 Raw Data Cleanup . . . . .	8
3.3.2 Restaurant Wise Proceeding . . . . .	11
3.4 Choosing and Fitting Models . . . . .	12
3.4.1 Categorizing of Orders . . . . .	12
3.4.2 Combination of Categorizations . . . . .	14
<b>4 Results</b>	<b>15</b>
4.1 No time categorization . . . . .	15
4.2 No time but slot categorization . . . . .	16
4.3 Day categorization . . . . .	16
4.4 Day and slot categorization . . . . .	17
4.5 Week categorization . . . . .	17
4.6 Week and slot categorization . . . . .	17
4.7 Weekday categorization . . . . .	18
4.8 Weekday and slot categorization . . . . .	18
4.9 Single slot categorization . . . . .	18
4.10 Combination of Categorizations . . . . .	19
<b>5 Conclusion</b>	<b>20</b>

<b>Bibliography</b>	<b>21</b>
<b>Appendix</b>	<b>23</b>
1    Appendix One: Consumer Application . . . . .	23
2    Appendix Two: Code and Results . . . . .	24

# List of Figures

3.1 Observations of preparation time for each order without any data clean up . . . . .	8
3.2 bservations of preparation time for each order without invalide values . . . . .	9
3.3 Observations of preparation time for each order without invalid values and outliers . . . . .	10
3.4 Average preparation time per day . . . . .	10
3.5 Different categorizations of the training data set . . . . .	11
3.6 Observations of preparation time for orders at yum2take without invalid values and outliers . . . . .	12
5.1 The consumer application. Login on the left. Idle . . . . .	23
5.2 The graphical userinterface of the consumer application . . . . .	24

# List of Tables

3.1	SIPOC Diagram derived from the five basic steps . . . . .	6
4.1	No time categorization results . . . . .	15
4.2	No time but slot categorization results . . . . .	16
4.3	Day categorization results . . . . .	16
4.4	Day and slot categorization . . . . .	17
4.5	Week categorization result . . . . .	17
4.6	Week and slot categorization results . . . . .	17
4.7	Weekday categorization results . . . . .	18
4.8	Weekday and slot categorization results . . . . .	18
4.9	Single slot categorization results . . . . .	18
4.10	Combination of Categorizations results for all orders . . . . .	19
4.11	Combination of Categorizations results for yum2take . . . . .	19

# 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 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 and Forecasting Basics

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. 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. For this purpose a SIPOC diagram can be created.

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.

### 2.2.1 Time Series

The method of time series 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.

## 2.2.2 Algorithms for Forecasting

Having charted the time series of actual events and analysed the patterns, it is now time to forecast preparation times for the first time. The book (Hyndman and Athanasopoulos (2013)) offers different algorithms to fulfill this task from which three are chosen und presented.

### 2.2.2.1 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:

$$ma_{(t)} = \frac{1}{t} \sum_0^{t-1} x_{(t-1)} \quad (2.1)$$

Same equation also applies to orders.

### 2.2.2.2 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:

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

Same equation also applies to orders.

### 2.2.2.3 Simple Exponential Smoothing

Another approach is the Simple Exponential Smoothing which return a smoothed forecast. A weighted average is calculated from the occurrences before which is weighted by the factor  $\alpha$  with the constraint  $0 \leq \alpha \leq 1$ .  $\alpha$  specifies the ratio between

the weight of the last order's preparation time and the 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:

$$ses_1 = \alpha * x_0 + (1 - \alpha) * ses_{start} \quad (2.3)$$

$$ses_t = \alpha * x_{t-1} + (1 - \alpha) * ses_{t-1} \quad (2.4)$$

# 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).

Emanuel Pallua is identified as the first stakeholder. He is Chief of Operations at VOLO and the one who gave the forecast in order. Setting boundaries and demanding his needs makes him source and customer according to the SIPOC diagram. The goals of this forecast for him are to decrease idle times for drivers by being at the restaurant on time as well as increasing the freshness of the food by not leaving it ready at the restaurant for longer than necessary. 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 Sontheimer 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 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

Table 3.1: SIPOC Diagram derived from the five basic steps

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

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 it makes the whole integration a lot smoother.

The last source and customer is the Operations Team since they are involved in the everyday business and will profit from the algorithm a lot. They support the algorithm with their real life scenario knowledge and require an operating forecast which is exact and needs no manual involvement.

## 3.2 Gathering Information

The information of the orders is taken from the database of the backend application which runs on heroku and is saved in PostgreSQL. The database is dumped at two dates in time. The data is downloaded into two sets, a training and a test set. The training set is to generate an initial forecast which is then validated with the test data set. This way relationships from the first forecast can be discovered and tested in the second one. The training data set was downloaded at the 26th of March 2015 to create a forecast which later can be compared to a test data set. This test data set was extracted from the server on the 29th of April. Speaking in number of orders, the csv file of the first extraction has 3034 entries while the second one has 4973 entries. The downloaded data sets were saved in comma seperated 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 information 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 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. Since only the timestamps of the status change are used, software failures have also be taken in account. 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.

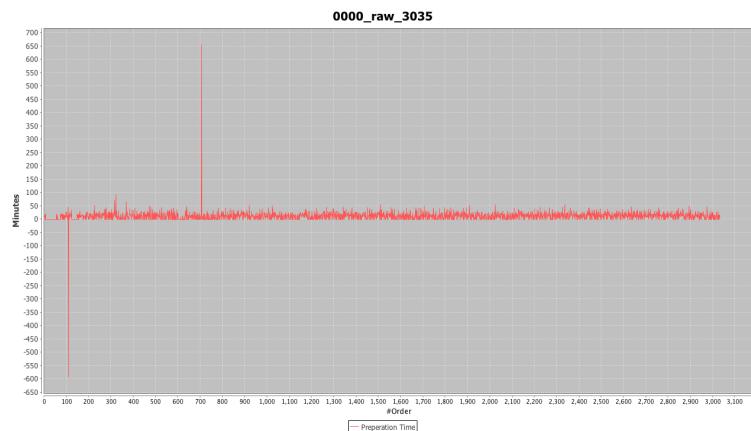


Figure 3.1: Observations of preparation time for each order without any data clean up

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 good data these flaws have to be removed from the data set. This is done by ignoring all values which are below 0 minutes and increases the average preparation time to 16 minutes. This result is much higher than the first result but is free of wrong values which makes it more reliable. The data set is visualised in Figure 3.2.

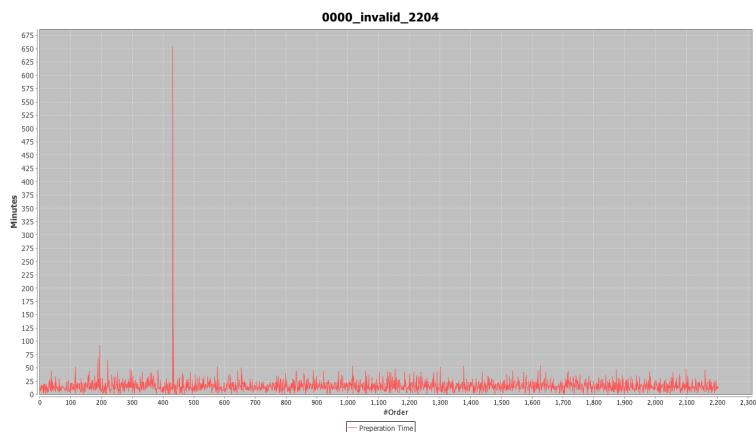


Figure 3.2: Observations of preparation time for each order without invalid values

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 seem not to come from a normally distributed population are removed. 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. The visualisation of the data is done in Figure 3.3.

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 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.

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

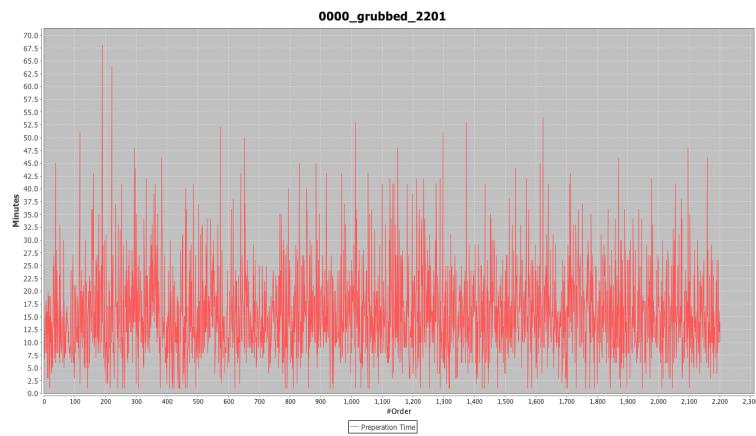


Figure 3.3: Observations of preparation time for each order without invalid values and outliers

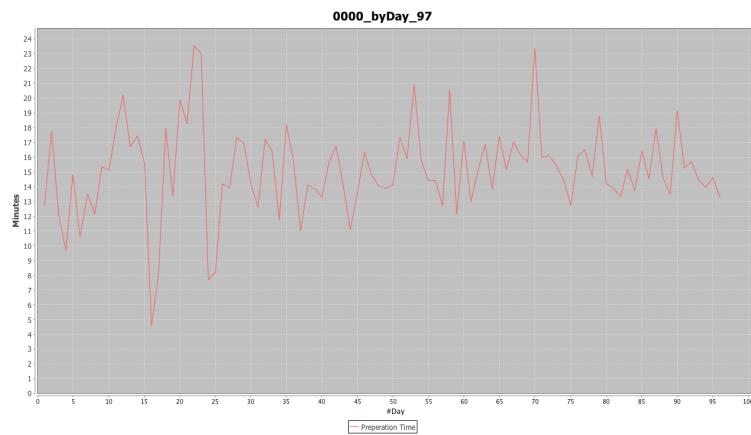


Figure 3.4: Average preparation time per day

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 spe-

cial 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.

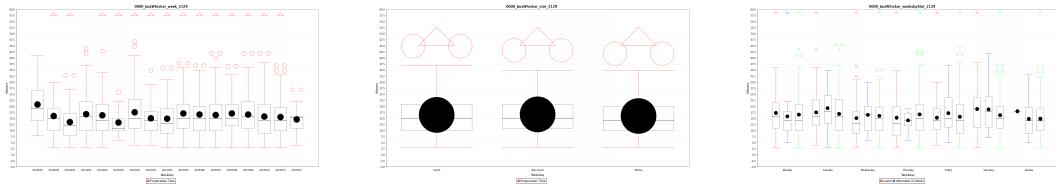


Figure 3.5: Different categorizations of the training data set

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

Before creating the model there should also be tested whether the restaurant has a significant impact on the preparation time.

What was learned from the Operation Team is that restaurants are very different from each other. There are restaurants, which have already pre-cooked ingredients, others have a very simple way of preparing the meal, like sandwiches, and others are crowded at a certain time and will take longer. A pizzeria has other preparation times than a burrito take away.

In order to get the best forecast possible for each restaurant they have to be looked at separately. For this purpose a popular restaurant should be chosen because it has the largest amount of orders and is thus a good representation. Normally this is a bad idea since there is already a limited set of data but in this case the gains are more significant since the restaurants are so different and should be looked at individually.

The restaurant with the most orders in the database is yum2take and is chosen. yum2take is a thai take-away and restaurant and the first restaurant volo delivered from. It is also close to the office which results in a bigger number of the driver being too early than being too late. Being early has the average that it does not falsifies our data as being late does. This is due to the timestamp `delivery_started_at` is being set as the driver leaves the restaurant. In order to get a first look at the data, a

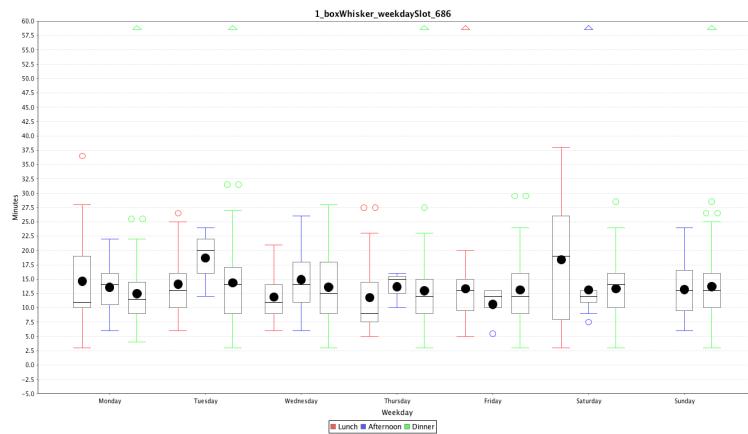


Figure 3.6: Observations of preparation time for orders at yum2take without invalid values and outliers

diagram categorized by slot for each weekday 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 also have been 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.

## 3.4 Choosing and Fitting Models

After the preliminary analysis it is time to fit the data to models for the forecast.

### 3.4.1 Categorizing of Orders

With these different first insights the models can be created, meaning forecasting algorithms and categorization can be applied to the data. These are the moving average, weighted and unweighted, as well as simple exponential smoothing with different factors. They are applied to the time series of data which is free from corrupted data and outliers. As described above, this is done in Java since it can be easily integrated with the backend of volo.

In order to integrate these algorithms into Java code, the csv file is parsed and processed according to the preliminary analysis. Then the orders are put into a hash map containing the a chronologic list of orders for each restaurant. For each restaurant the list of orders is passed to the forecasting unit where the calculations are done. The root mean square error (RMSE) is calculated by calculating a forecast for each order and getting the square error by squaring the difference between forecasted and actual preparation time. These square errors are summed up for all orders of the restaurant, divided by the number of forecasts and then square rooted. This RMSE is then used to evaluate the result of the forecast, lower meaning better.

With these algorithm and process available, different categorizations are done in order to find the best forecast.

### No time categorization

First of all, no time categorization is done. The forecast is done using all preceding orders for the current order which should be forecast. This is a very simple forecasting model and it is not very accurate since the more orders are already included the closer the forecast is to the average. The RMSE is calculated as described above.

### No time but slot categorization

The next categorization works just like the one before. The only difference is that the orders are divided into three subsets according to the slot they are in, e.g. lunch slot. For each slot the forecast is done independently. The three summed up mean square errors (MSE) of each slot are added up, divided by the number of slots, three, and then square rooted, which results in the RMSE for this categorization.

### Day categorization

After doing uncategorized forecasts, a time component is introduced. Since preparation times can vary from day to day, e.g. weekdays or holidays, they should be independent. For this reason, orders are separated by day and forecasting is done using only the previous orders of the current day. In the end the MSE of each day is summed up and the RMSE is calculated as before.

### Day and slot categorization

In order to improve the day categorization, each order is additionally to the day categorized by the slot of on that day. The forecasting is done for each slot and the RMSE overall is calculated.

### Week categorization

The next time categorization is on weekly basis. Like in the daily calculation, the error for each order is calculated by taking all preceding orders of the current week. The RMSE is generated according to the other categorizations.

### Week and slot categorization

The weekly forecast is improved by splitting the week into the three slots and calculating the forecast value by only using orders from the specific slot of the week. This results in a more specific RMSE than when only the week is used.

### Weekday categorization

For this categorization orders are cumulated by weekday and forecast by using the orders on the current weekday before the order which should be forecasted.

### **Weekday and slot categorization**

This is the refinement of the weekday categorization. For the calculation only orders in the specific slot of the weekday of the current order are chosen for the forecast.

### **Single minislot categorization**

The final categorization is done by separating orders into their day and half hour minislot. Each half our slot starts either at full o'clock or at half past. The current order is then forecast by using the orders of the current day and minislot.

Before the calculation can be done, the edge case of not having preceding 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 Categorizations**

In addition to a very basic categorization, Sebastian Sondheimer suggested a combination of different categorizations. His focus for the forecast lies on two pillars. One is the best and most useful data to increase accuracy and the other one is to keep factors in the calculation to a minimum in order to improve speed. These two factors can work against each other and thus have to be carefully adjusted using machine learning. The result is the combination of six factors which weights have to evaluated. The first factor is the forecast for the current slot in which the order is. The second value is the forecast calculated from all orders of this day. This is combined with the forecast for all orders in the last 7 days as well as the forecast for the current slot of the last day. The last two factors are the forecast for this slot of this weekday of the last 4 weeks and the overall preparation forecast of this weekday for the last 4 weeks. The weights are calculated by iterating different distributions of weights for the slots for each algorithm, namely weighted and unweighted moving average as well as simple exponential smoothings. The results have then to be evaluated in order to find the best match.

## 4 Results

Now that the models are run and the RMSE are collected it is time to evaluate the results of the different models. The results will be evaluated in two ways. The first is the RMSE of the training set. The lower the error is, the better the model fits the test. But using only this method is not a good indicator whether the model will fit future data. This is why in a second evaluation the difference between the error of the training set is compared to the test set. The test set, including the training data, is run through the algorithm the same way as the training set to calculate the error. The evaluation is done for the overall orders and a restaurant. Since *yum2take* was chosen to be the best example with the best data, the evaluation of the results is done on this restaurant.

For each model, the best results are presented in a table. Since there are many options for the weighted moving average and simple exponential smoothing, only the lowest error 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

Table 4.1: No time categorization results

Algorithm	yum2take			all		
	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

**All orders** Using the different moving average algorithms on all orders results in an RMSE of 8 to 8.5 minutes. This results in a 16 to 17 minute window in which the food is finished. The results for the different simple exponential smoothing variations are even worse. They reach from 8.5 minutes to almost 12 minutes. These values cannot be used in a convenient way to forecast orders. This is why the forecasting should be done with restaurant specific data.

#### Only *yum2take* orders

The results for the order based model are shown in figure[ ]. The RMSE<sup>TMS</sup>s of the difference between forecasted preparation time and the actual one for the different moving average variations are all pretty much the same. They are close to 6 minutes which results roughly in a 12 minute window for the driver to pick up the food just

in time at the restaurant. The weighted moving average consists of different sizes for the weight. It is done in steps of 25 orders more. It is pretty stable at around 6 minutes when under 200 orders are taken into account of the calculation. Until a weight of around 600 orders the error is slightly higher, lowering for around three quarters of a minute when almost all orders are taken into account. This result may be caused by having only a few orders left when taking many for the forecasted value.

The difference between test and training set is almost neglectable. It improves for almost all algorithms but the gain is only around 0.5 minutes which is not that drastic to the requirements of the forecast.

## 4.2 No time but slot categorization

Table 4.2: No time but slot categorization results

Algorithm	yum2take			all		
	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	0.618189

### All orders

[] rmse generated for the test and training set as well as the difference between them and the used algorithm on order basis The difference between test and training set is almost neglectable. It improves for almost all algorithms but the gain is only around 0.5 minutes which is not that drastic to the requirements of the forecast.

### Only yum2take orders

## 4.3 Day categorization

Table 4.3: Day categorization results

Algorithm	yum2take			all		
	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

### All orders

### Only yum2take orders

## 4.4 Day and slot categorization

Table 4.4: Day and slot categorization

Algorithm	yum2take			all		
	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

All orders

Only yum2take orders

## 4.5 Week categorization

Table 4.5: Week categorization result

Algorithm	yum2take			all		
	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

All orders

Only yum2take orders

## 4.6 Week and slot categorization

Table 4.6: Week and slot categorization results

Algorithm	yum2take			all		
	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	

All orders

Only yum2take orders

Table 4.7: Weekday categorization results

Algorithm	yum2take			all		
	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	

## 4.7 Weekday categorization

All orders

Only yum2take orders

## 4.8 Weekday and slot categorization

Table 4.8: Weekday and slot categorization results

Algorithm	yum2take			all		
	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	

All orders

Only yum2take orders

## 4.9 Single slot categorization

Table 4.9: Single slot categorization results

Algorithm	yum2take			all		
	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

All orders

Only yum2take orders

## 4.10 Combination of Categorizations

### All orders

Table 4.10: Combination of Categorizations results for all orders

Algorithm	Results			Weights					
	training	test	difference	slot			all orders		
				#2129 orders	#3196 orders	today	7 days	4 weeks	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

### Only yum2take orders

Table 4.11: Combination of Categorizations results for yum2take

Algorithm	Results			Weights					
	training	test	difference	slot			all orders		
				#686 orders	#945 orders	today	7 days	4 weeks	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**

## Bibliography

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

# **Appendix**

# 1 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. 5.2 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 5.2 is shown. In this screen "assigning\_to\_driver" is shown as status. This means the driver cannot yet

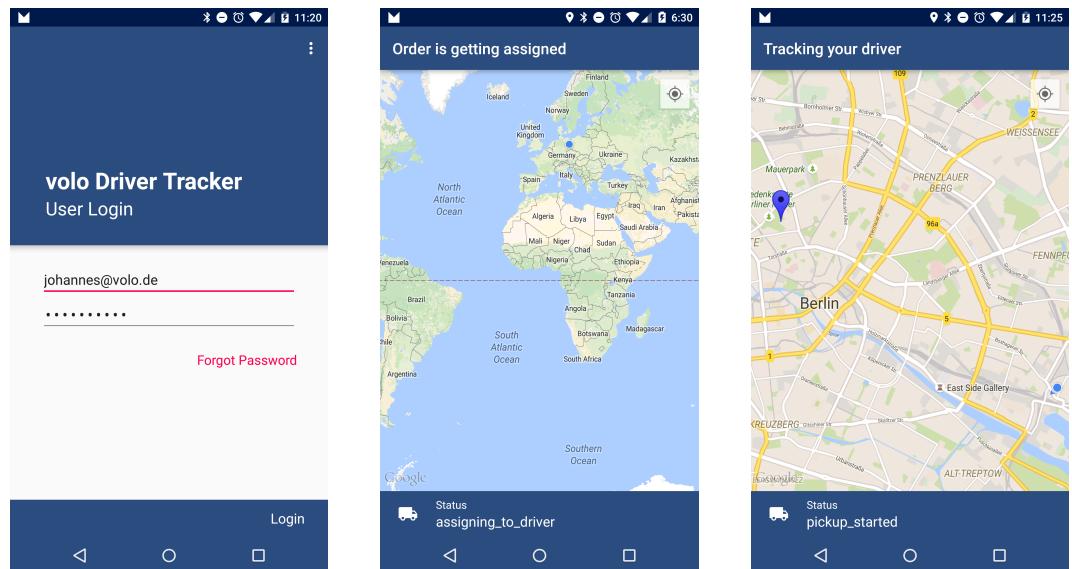


Figure 5.1: The consumer application. Login on the left. Idle

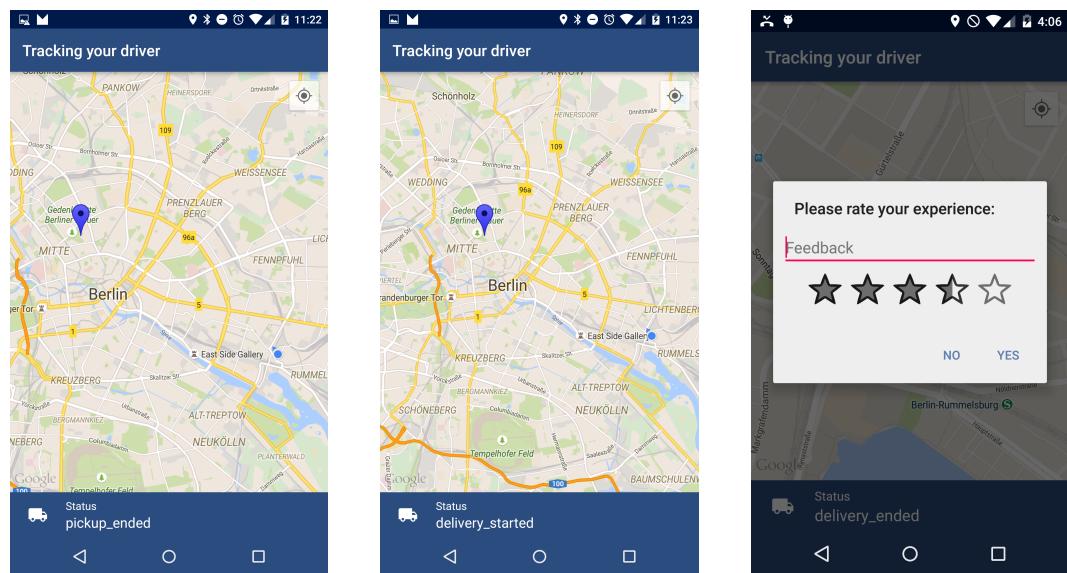


Figure 5.2: The graphical userinterface of the consumer application

## 2 Appendix Two: Code and Results

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

## **Ehrenwörtliche Erklärung**

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt und indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Ich weiß, dass die Arbeit in digitalisierter Form daraufhin überprüft 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 Überprüfung zum Vergleich mit künftig eingehenden Arbeiten dort verbleiben. Weitere Vervielfältigungs- und Verwertungsrechte werden dadurch nicht eingeräumt. Die Arbeit wurde weder einer anderen Prüfungsbehörde vorgelegt noch veröffentlicht.

Ort, Datum

Unterschrift