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.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 save resources and time for the drivers. It gives a good advantage but it still can be improved by tuning the time component which cannot be calculated by the algorithm itself. 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 with an online appealing shop, followed by process optimized driver fleet. The optimized fleet is build on the underlying algorithm which makes deliveries 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 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 is early at the restaurant and has to wait for the order to be prepared. Forecasting the 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 send 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. AthanasopoulosHyndman and Athanasopoulos (2013) was chosen as a starting point, since it covers the topic pretty good for forecasting beginners. After reading the book it was determined to follow the steps the book suggests to create the forecast. The book had five basic steps for forecasting which will be explained in the following.

There are various Stakeholders, e.g. the people who provide the data, those in charge of the use of the forecast and those who monitor how the forecast is put together. All have different views on the problem, different needs and different prefer-

ences. In the first step, these Stakeholders are consulted to define the problem. who it is required and how it fits into the process of the ones who required it have to be asked. Also the people who support the forecast with collecting data, maintaining the data and use the forecast for future planning have to be figured out. After the problem is defined, in a second step, all information available is gathered. 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. In order to make up for the little and imperfect data, the expertise of the people who collected this data and use the forecasts is also taken into account for the forecast. When the data collecting is finished, a preliminary analysis is done to identify patterns and abnormalities. This is done before fitting models to the data in order to prevent false assumptions. For the analysis the data is put into a graph to get a visual result. In the graph abnormalities like wrong data, patterns, e.g. trends, seasonalities or business cycles, can be detected without much effort and outliers can be questioned before beginning the process. Now the real forecasting can begin by choosing adequate models. These models are fitted to the data and expertise and the forecasting is done. The result of the different models are finally compared in regards to their prediction versus the actual events.
Now that there is an overall plan for the forecasting, a closer look is taken on the underlying parts of the forecast.

2.2.1 Time Series

The method of time series puts events (forecasted or actual) onto a time axis. Charting this as a graph often reveals patterns. These patterns are divided into 3 different kinds Hyndman and Athanasopoulos (2013). The first pattern is the trend. The trend is an decrease or increase over a long period of time. The second second is the season pattern. It is influenced by seasonal factors and has a known and fixed time period and. The third one is the cyclic pattern. Cyclic pattern influence usually has fluctuations of at least 2 years and the period is not fixed. 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.
The time series 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.

2.2.2 Algorithms for Forecasting

Having charted the time series of actual events and analysed the patterns, it is not time to forecast for the first preparation times.. The book 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:

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

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:

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

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 occurances before which is weighted by the factor $\alpha with the constraint 0 \leq \alpha \leq 1. \alpha specifies the ratio between the weight of$

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

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

3 Methodology

The following part explains the single steps of the forecasting process for this Interdisciplinary Project in detail and the forecast will be presented from the very beginning.
First of all the problem will be defined properly. Then the information needed will be specified and gathered. This data is then analyzed in a preliminary step and after this models for the data will be examined. In the end the result of the models will be evaluated and further actions will be discussed in the final conclusion.

3.1 Problem Definition

Before the forecasting can start, some general questions have to be solved. Most important is the question of who is requesting the forecast and what will the forecast be used for. In order to acquire this information, the COO, Emanuel Pallua, who is requesting this forecast, is interviewed. The aim is to forecast the preparation time of a restaurant. This is needed to be able to send the driver to the restaurant just in time when the meal is prepared. This improves the algorithm and reduces driver idle times. The forecast should be done once to be evaluated and later the most accurate method should be integrated into the algorithm itself for automatization. In addition to this, Sebastian Sondheimer gave some input regarding the forecast parameters. The forecast should include different periods, like the current time, day and a limit time of the past. This will be explained in more detail later.
After having the requirements from business side, the tech side is examined. Sergej Krauze, CTO and responsible for the algorithm, explained that the algorithm takes all restaurants with open orders, the customers and the drivers to calculate the most efficient route to pickup and delivery the food. The algorithm is written in Java so it would be welcome to develop the forecasting in Java as well. Finally Stefan Rothlehner, CTO and responsible for the backend, had to be questioned, what data is available and how it can be extracted. Every order is stored in a PostgreSQL database on heroku, on which it can be accessed and downloaded.

3.2 Gathering Information

The database is dumped at two dates in time. 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. In addition to the historic data, the operations team was interviewed for their personal experience with the preparation times. They said that estimating around 15 minutes is more or less accurate except for some restaurants which are known to take much longer.
This information is valuable since the historical data has some weaknesses. The biggest problem is the size of the gathered data. Forecasts need big amounts of information to generate a meaningful result so the data.
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 picks it up, there are no modification to the time done.

3.3 Preliminary Analysis

There are many factors which can influence the preparation times which can be of external or internal nature.The weather or season can have an effect on how busy the restaurant is which can cause different cooking times. Since the timestamps of the status change are used, software failure has also be taken in account. There are also other factors, like marketing campaigns, 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 average preparation time of the orders is calculated. This is done to get a feeling in which how long the restaurants usually take. It can give a rough estimation of what can be expected and compared to the forecast later to find out whether the result is total unrealistic or pretty close to what is possible. For the raw data the result is roughly 12 minutes.
In addition to the average time analysis a visualization of the data is done. The

best way to do this is to use a time series plot. In the graph it is very easy to see anomalies and patterns.

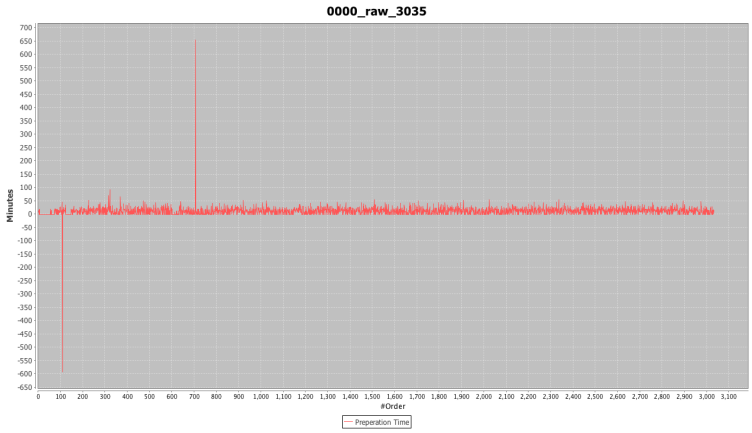

Figure 3.1: Observations of preparation time for each order without any data clean up

The analysis of the raw data diagram (Fig. 3.1) reveals different problems. The orders from the trainings dataset still contain unfinished orders, bugged orders, which were finished at a wrong point in time, and orders which have corrupted time stamps. Unfinished orders have missing timestamps and like unreadable timestamps they get as "total"-time attribute -1 to be detectable. This can be observed in many cases in the diagram above. Also an order which was finished the next day can be seen around the 100 mark. This flaws in the data lead to a result of 12 minutes of average preparation time. This cannot be taken as a serious results since the -1 results lower the average drastically below of what the real average would be like. These values have no value to the forecast and have to be removed from the pool. This is done by removing all -1 from the used data. The result can be seen in [] below. This increases the average preparation time to 16 minutes. This can be explained because of the missing -1 values.

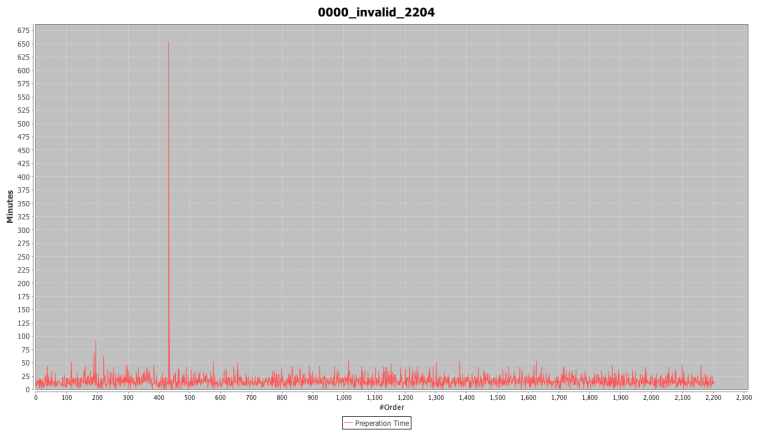

Figure 3.2: An example figure

After removing the obviously unusable orders, the ones which were finished long time after completing the delivery, mostly because of bugged software, have to be

removed. 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.

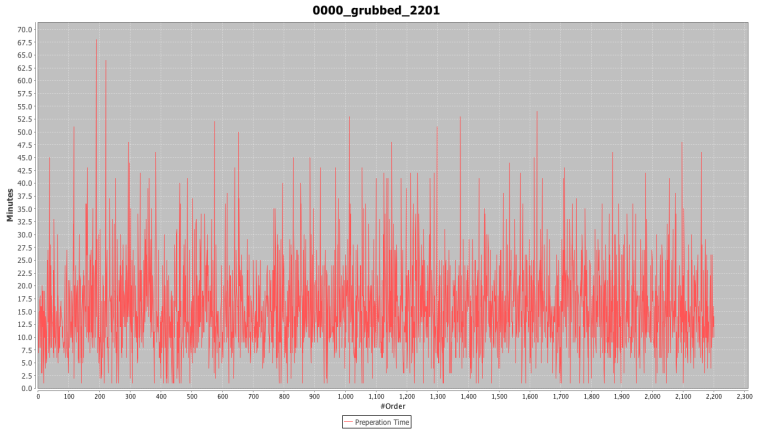

Figure 3.3: An example figure

Now the usable training dataset is extracted from the input data, we can analyze it for patterns for patterns or similarities.
`0000_grubbed` is very fine and sorted by order. It was created to get an overview focused on orders and does not have a time component. It cannot provide information about patterns since patterns rely in this case on time. In order to include this, the orders are sorted by day and put into a time plot [].

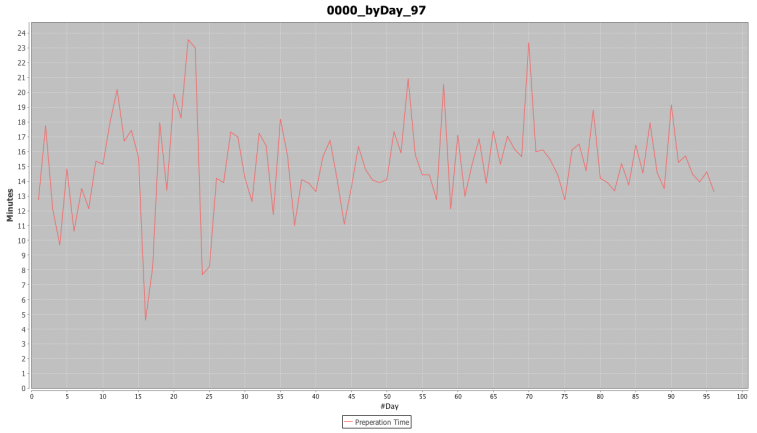

Figure 3.4: An example figure

Since no clear trend or pattern can be observed in a day by day analysis, a box and whisker diagram is created from the data. 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 in which boundaries most of the

orders are located. The data was divided into different categories. The first diagram [] is on week basis. It was created to see if there is a rough trend over time as the company expertise grows. As the diagram shows there cannot be made any conclusions for this categorization. After fluctuating preparation in the first weeks it gets more constant towards the end. Since this is not helping a box whisker diagram for each slot was created []. There are three slots and each is a part of the delivery time of the day. The day is divided into 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. This has to be inspected in a different categorization of the slots so the idea is to have a look at slots on different days. The choice is to identify special behaviour according to the weekday since weekdays can have strong varying load for the restaurant. 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. But the values in the diagram do not vary that much which can be due to a low training dataset. This leads to the last overall diagram []. It shows the preparation time for each weekday, split by slot. This is done since it contains two different key information. Slots and weekdays alone do not have as much information as the two combined. The weakness of slots and weekdays alone is e.g. at the weekend ordering for lunch is not as popular as ordering lunch at work. Also restaurants will have more workload on the weekdays when everyone is coming for lunch than on the weekend. It could be separated between weekend and weekdays but there is not enough data to do this and gain additional information. In order to maximize information gain a diagram containing slots per weekday is introduced. 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.

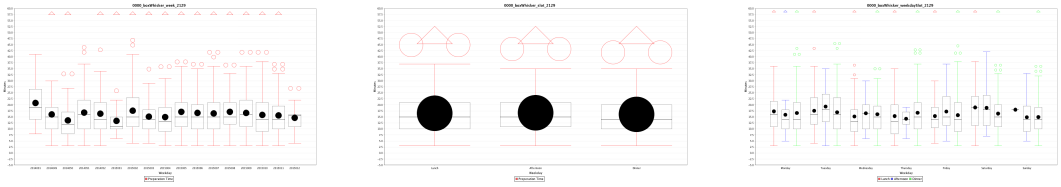

Figure 3.5: default

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. 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 the largest amount of orders and is thus a good representation. The restaurant with the most orders in the database is yum2take. 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, since we do not know how long the food was already prepared. In order to get a first look at the data, a diagram categorized by slot for each weekday is created []. The distribution in the diagram for yum2take

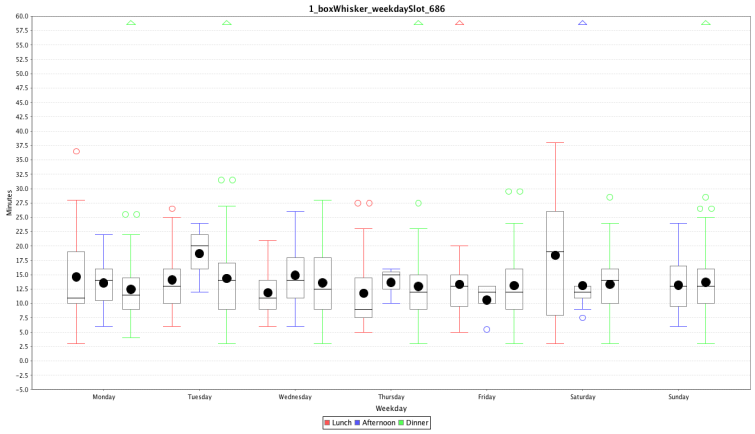

Figure 3.6: An example figure

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

3.4.1 Categorizing of Orders

With these different first insights the model can be created. Now forecasting algorithms 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 had to be parsed first. This was done using a csv library. After reading the csv the raw data was processed to the diagrams seen in the preliminary analysis. The data for the models was freed from invalid entries and outliers. Then the orders were put into a map hashed by restaurant and each restaurant contained a list of orders

in chronic order. For each restaurant the list of orders is passed to the forecasting unit where the calculations are done. The mean square error is calculated for the moving average and simple exponential smoothing. In order to get the best forecast, different categories are calculated.

No time categorization
First of all, no time categorization is done. The forecast is done as all predecessing orders are relevant to the current one. This gives a rough estimation over the forecasting possibilities. It is not very accurate since the more orders are already included the closer the forecast is to the average. The next forecast is done with the same assumption but for each slot. Each order in the slot is forecast by all orders in that slot before it. This can eliminate the difference in case preparation times vary between the different slots. In the end the mean of the three mean square errors is taken and the root is calculated as overall root mean square error.

No time but slot 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 forecast according to this time constraint. Forecast is then done for each day as well as each slot for each day. In the end the errors of each forecast model are summed up and the average is returned. In case of the slots of each day, the average is first calculated for each slot day by day and in the end the mean errors of each slot are summed up.

Day and slot categorization

Week categorization
The next time categorization is on weekly basis. Like in the daily calculation, the error for each order by taking all predecessing orders of the week. This is also done for orders in a slot of each week similar to the day basis.

Week and slot categorization

Weekday categorization
The same procedure as for day and week categorization is done for weekday as well.

Weekday and slot categorization

Single slot categorization
The final simple categorization is done by separating orders into their day and slot. Each slot is then treated separately (the same way as in the day only calculation). In the end the mean square error of each slot is added up and the average is rooted to get the overall root mean square error.

Before the calculation 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 Categorizations

For the next step, Sebastian Sondheimer was interviewed. He is in the business department and joined later which is the reason he was not part of the problem definition. 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. 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 (un)-weighted moving average and simple exponential smoothings. The results have then to be evaluated in order to find the best match.

Now that the models are set, they are run and generate forecasts. In order to evaluate the results the error for each data pair, the forecast value and the real value, is calculated by [e = yi - y^i].$The error is then squared and added to the overall error which is the sum of a$

Refer to symbols or abbreviations with
`\gls{symbolname} \glstext{symbolname} \glsfirst{symbolname}`:
first $\mathbb{N}$ additional $\mathbb{N}$ first $\mathbb{N}$
Bachelorarbeit (BA) Diplomarbeit (DA) Masterarbeit (MA)

Refer to other sections with `\ref{labelname}`:
A reference to this subsection: **??**

Include figures with
`\begin{figure}`

```
...
\caption{An example figure}\label{fig:example}\end{figure}
```

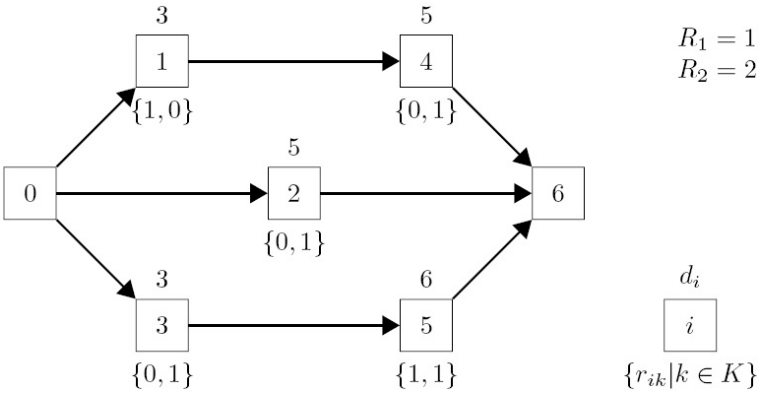

Figure 3.7: An example figure

Include only PostScript images (.eps) if you want to create a PostScript document using dvips and only .pdf, .png, .jpeg and .gif images if your goal is a PDF document using pdflatex.

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 way 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

	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

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. This values cannot be used in a convenient way to forecast orders. This is way the forecasting should be done with restaurant specific data.

Only yum2take orders

The results for the order based model are shown in figure[]. The RMSEâ™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

	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

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

All orders

Only yum2take orders

Table 4.3: Day categorization results

	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

4.4 Day and slot categorization

Table 4.4: Day and slot categorization

	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

All orders

Only yum2take orders

4.5 Week categorization

Table 4.5: Week categorization result

	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

All orders

Only yum2take orders

4.6 Week and slot categorization

All orders

Table 4.6: Week and slot categorization results

	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	

Only yum2take orders

4.7 Weekday categorization

Table 4.7: Weekday categorization results

	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	

All orders

Only yum2take orders

4.8 Weekday and slot categorization

Table 4.8: Weekday and slot categorization results

	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	

All orders

Only yum2take orders

4.9 Single slot categorization

Table 4.9: Single 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

All orders

Only yum2take orders

4.10 Combination of Categorizations

All orders

Table 4.10: Combination of Categorizations results for all orders

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

Only yum2take orders

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

ination of Categorizations results for yum2take

5 Conclusion

Bibliography

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

Appendix

1 Appendix One: Consumer Application

2 Appendix Two: Code and Results

Code repository on Bitbucket:
https://bitbucket.org/H1GHWAvE/volo-java-evaluation&&

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