Maxim Zavadskiy

Using Forecasting Models to Optimize Production Schedule in a Cafe

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

1 December 2015

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Maxim Zavadskiy Using Forecasting Models to Optimize Production Schedule in a Cafe 45 pages + 1 appendix 1 December 2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Tein-Yaw Chung, Professor Olli Alm, Principal Lecturer

The purpose of the thesis was to research possible methods to predict the demand of the products and choose the best one to be used in the production schedule optimization module to be built in the cafe production management application, developed for the client. History production data from the client's cafe was analyzed using R language. Three forecasting models were evaluated - ARIMA, support vector regression (SVR) and simple probabilistic model. Overall, the research shows that it can be feasible using examined forecasting models to predict product demand. The results were demonstrated to the client and the client sees the potential in the examined prediction algorithms. The SVR-based method showed good performance on all time series and produced particularly interesting results on one of them. The study shows that it will be worth integrating the SVR-based solution in the application in testing mode to verify its performance. It will also be worth carrying out deeper research as more data will be collected.

Keywords	forecasting, machine learning, optimization, SVR, ARIMA, statistics, algorithm, web development, café, scheduling, production

Contents

1 Introduction 1

1.1 Automation of Production Management in a Cafe 1
1.2 In-house Production Scheduling System 2
1.3 Problem Statement of Automated Production Scheduling 2

2 Forecasting models 3

2.1 Forecasting Models Overview 4
2.2 ARIMA 5
2.3 SVR 8

3 Review of Existing Production Scheduling Systems 9

3.1 In-house Production Scheduling System 9
3.2 Market Review: Production Scheduling Solutions 12

4 Models for Forecasting Product Demand 13

4.1 Analysing Product Data 13
4.1.1 General Statistics 14
4.1.2 Unknown Demand 14
4.1.3 Trend Analysis 16
4.2 Validation Criteria 16
4.3 Implementing Selected Forecasting Models 19
4.3.1 Simple Probabilistic Model 19
4.3.2 ARIMA 21
4.3.3 SVR 22

5 Evaluation 23

5.1 Evaluation Setting 23
5.1.1 Time series #1 - Real Data from Another Cafe 24
5.1.2 Time series #2 - Simulating 12 h Product Group 25
5.1.3 Time Series #3 - Simulating Products with Strong Weekly Pattern 29
5.2 Running the Forecasting Models 32
5.2.1 Time series #3 32
5.2.2 Time series #2 34
5.2.3 Time series #1 36

5.3 Evaluation Summary 38
5.4 Results 39
6 Future Improvements 41
6.1 Collecting Sufficient Data 41
6.2 Handling Unknown Demand 41
6.3 Improving Forecasting Models 42
7 Conclusion 43
References 44
Appendix 1. Simple Probabilistic Model’s main fragments of source code.

1 Introduction

1.1 Automation of Production Management in a Cafe

Nowadays it is possible to automate retail businesses with the help of software. Technology advances and now retail and food businesses can make use not only of point of sale systems but also of more advanced systems that automate inventory planning and aim to maximize profit and minimize waste (shrinkage) of products [1;2]. In order to automate inventory planning, software packages, such as JDA Fresh Item Management [1], use automated forecasts of product demand. There are various forecasting methods available nowadays that can be applied for predicting such time series as the product demand. Learning from product demand history data, those methods are capable of computing future product demand values. [3.] Those forecasting methods include exponential smoothing [3], ARIMA models [4], neural networks [5] and regression models [3].

I worked remotely for a client in Switzerland and developed an application for cafe production management. My client owns a petrol station and a cafe inside the petrol station. In the cafe they sell beverages as well as snacks: sandwiches, bread and cakes. The snacks, which will be referred to as "products" or "items" in this paper, are produced inside the cafe by warming up frozen pre-made bread and cakes, preparing filling for the sandwiches and assembling the sandwiches. Then those products are put on a vitrine, so that the customers can buy them. The products have to be fresh and therefore if they are not sold during a particular time, they have will to be taken out from the vitrine and thrown away (trashed).

The client had two main issues to be solved:

1. To document the production process and specify the production schedule, printed spreadsheet documents are used. A considerable number of documents are used and it is problematic to keep then all in order. Making spreadsheets, printing them out and filling them by hand is time-consuming and not convenient.
2. Every day there is a considerable number of products that are not sold and therefore have to be trashed. The potential profit decreases because of that. In addition it is not environmentally friendly.

The application that has been developed resolves the first issue and partly the second issue by offering functionality to schedule the production manually. However there was

still no functionality to automatically optimize production schedules, which would be highly desirable by the client and was one of the main motivations for the development of the application.

1.2 In-house Production Scheduling System

As it is not efficient to produce a single item per order products at the client's café (here-after called "client's domain") are produced in batches (several products at once), several times a day. The first batch production time is 6 am. In one batch many kinds of products are produced. Those freshly made products are put on a vitrine and are to be sold to consumers within the next few hours. There are 3 different product types, with a lifetime of 4 hours, 6 hours and 12 hours. Further I will refer to them as 4h, 6h and 12h product groups. After 4 to 12h since production, those products will have to be trashed if they are not sold.

Each of the product groups can be examined separately. For instance, 6h products are firstly produced at 6 am. Then at 12 pm all the 6h products that are not sold, have to be trashed, and a new batch of 6h products will be produced to be sold during 12pm - 6pm. The third and the last period will be 6pm to 12am. Further I will refer to each period as a batch period, or batch time. E.g. for 6h products there are 3 batch times each day - 6am - 12pm, 12pm - 6pm, 6pm - 12am. Respectively 4h and 12h products have their own batch times, calculated in the same way.

With the above-mentioned production system, the batches are scheduled to be produced about 5 times a day. Generally there are the same batches used for weekdays and al-tered versions for Saturdays and Sundays.

1.3 Problem Statement of Automated Production Scheduling

As the employees need to know how much of each product to produce for each batch time, the demand of the product needs to be known for each batch time. If demands for each product for each batch time are known, optimal batches can be constructed. Then the problem can be localized as follows: to predict the demand for a given product for a given batch time 1 day ahead. If assuming that the demands of the products do not affect each other, a demand during a batch time needs to be predicted from the demand during the same batch time of the preceding days.

The goal of this project is to research possible methods to predict the demand of the products in order to construct a production schedule optimization system to be built in the application. As products in the cafe are mostly consumed on the day of production, it is sufficient to forecast the demand only one day ahead given the daily time series of demand during a product batch time. These time series were assumed to be potentially seasonal, with a repetitive weekly pattern.

Statistical computing and graphics language, the R programming language [6], was used for cafe production history data analysis, algorithm verification and for constructing some of the prediction models. The MATLAB software solution was also utilized to build prediction models [7].

The following selected prediction models will be evaluated:

- Simple probabilistic model
- ARIMA
- SVR, a regression model.

ARIMA and SVR were chosen as the candidates as they are proven techniques to predict seasonal time series (time series with a pattern, repeating every x observations [3]), including domains, similar to the client's domain [4;8]. I adopted existing SVR-based solution**,** developed in MATLAB environment by Yuan, Fong-Ching, Assistant Professor, Department of Information Management, Yuan Ze University. A simple probabilistic model was developed as a straightforward and quick solution that was compared with ARIMA and SVR. Those 3 models will be tested against selected datasets and their performance will be compared to choose the suitable prediction model for the smart production schedule module of the application.

2 Forecasting models

This chapter introduces forecasting models that were analysed in this paper. If not told otherwise, the facts provided in sections 2.1 and 2.2 are based on the book by Hyndman R. and Athanasopoulos G. [3]. The facts provided in section 2.3 are based on articles [9] and [11].

2.1 Forecasting Models Overview

Forecasting is a process of predicting the future as accurately as possible, using the previous knowledge, such as historical data. It is a common task in business and is used in planning of production and business strategy, scheduling and logistics. When forecasts are made, the next logical stage is planning - utilizing forecasting results to meet business goals. For example, if the demand for a product is forecasted for the following month, the appropriate arrangements can be made to adjust production and transportation for the next month to meet the demand.

In a typical forecasting task, firstly, a problem to be solved needs to be stated, and decisions need to be made what to forecast. Then data is collected for the forecast and preliminary data analysis is made to find out possible patterns and relationships within the data. After that, several forecasting models (often 2-3) are fitted and compared. The ones which give the most accurate results will be utilized.

Forecasting methods can be split in two main groups: qualitative and quantitative. If the data is not available for the forecast or it is irrelevant to the parameter to be predicted, qualitative forecasts can be applied. Qualitative forecasts are forecasts that are done by a human, using her knowledge about a domain and her own judgement. When, for example, a new competitor enters the marker, qualitative forecasts need to be done. It is important to mention, that those forecasts are subjective.

When quantitative data is available about the past and some patterns are assumed to be repeated in the future, quantitative forecasts can be used. There exist a variety of quantitative forecasts methods. Some of them are designed for specific domains and some of them are more universal.

There are two main quantitative forecasts subgroups - cross-sectional forecasts and time series forecasts [3]. Cross-sectional forecasts are using cross-sectional data - data that is collected at a single point in time. It aims to forecast one unknown parameter using the known parameters, utilizing the relationships between them. For example, a car emission of a particular model can be predicted from the data of other cars, using the relation of engine volumes and weights to emission amounts.

When the observed value is changing over time (e.g. demand for a product), time series forecasting can be used. Here time series data is used - data that is observed sequentially over the time. The aim of time series forecasting is to predict how these sequential observations might continue in the future. It can be useful to utilize other related data to make a forecasts. It could be, for example, day of the week for forecasting demand of a product.

There are a wide range of classic machine learning and data mining techniques that can be applied to make time series forecasts. These include ARIMA models, exponential smoothing and structural models [3]. Universal machine learning methods, not specially designed for time series forecasts, such as neural networks [5] and support vector regression [8], can be also applied and give accurate results. In the next two chapters only ARIMA and SVR models will be described in more details as these are the ones that were utilized in this final year project.

2.2 ARIMA

ARIMA or AutoRegressive Integrated Moving Average is one of most widely used approaches to time series forecasting [3]. ARIMA is capable of forecasting both seasonal and nonseasonal time series. ARIMA was applied successfully to e.g. forecast a demand for a product in a fast-food restaurant [4].

ARIMA is a complex model that combines differencing of the time series, Autoregressive (AR) and Moving Average (MA) models. To explain basics of ARIMA, differencing, AR and MA are needed to be described.

Before starting with AR or MA, it is useful to explain backshift operator. Backshift operator “B” is makes it convenient to notate time series lags, for example, for a time series y(t), $By_t = y_{t-1}$. Here By_t basically means shifting the time series back one period. Shifting back for two periods can be described as $B(By_t) = B^2y_t = y_{t-2}$.

ARIMA employs the concept of differencing. Differencing is transformation of time series into time series of differences between consecutive values in the given time series. For example, first order differencing can be written as:

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t$$

Expression 1. First order differencing. [3.]

Second order differencing, a difference of differences of a time series, is written as:

$$\begin{aligned} y''_t &= y'_t - y'_{t-1} \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2}. \end{aligned}$$

Expression 2. Second order differencing. [3.]

which can be expressed with a backshift notation as:

$$y''_t = y_t - 2y_{t-1} + y_{t-2} = (1 - 2B + B^2)y_t = (1 - B)^2 y_t$$

Expression 3. Second order differencing using backshift notation. [3.]

Differencing is helpful to stabilize a time series to enable it to apply MA on AR models onto.

Autoregressive model is a model where the future value in a time series is predicted, based on a linear combination of consecutive p past values in that time series. The model of order p AR(p) can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t$$

Expression 4. Autoregressive model of order p. [3.]

Writing Expression 4 using backshift notation gives

$$\left(1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2 - \ldots - \phi_p \mathrm{B}^p\right) \mathrm{y}_t = \mathrm{c} + e_t,$$

where c is a constant and e_t is a white noise (time series, where the next value does not depend on the previous values). $\phi_1,\ldots,\phi_p$ are the parameters to be determined, while fitting the model on a time series. Those parameters describe the patterns within time series.

Moving average model, on the other hand, is defined, based on the linear combination of forecast errors (difference between the predicted value and the actual value in a time series). The MA of order q, MA(q), can be written as:

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q}$$

Expression 5. Moving average model of order q. [3.]

As in AR, $\theta_1,\ldots, \theta_p$ are the parameters to be determined, while fitting the model on a time series. Writing Expression 5 using backshift notation gives:

$$y_t = c + \left(1 + \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q\right) e_t$$

The non-seasonal ARIMA model uses both lagged values of time series y_t and lagged errors e_t, and it incorporates differencing of the time series. It can be described using backshift notation as$_t$:

$$\underset{\uparrow}{(1 - \phi_1 B - \cdots - \phi_p B^p)} \quad \underset{\uparrow}{(1 - B)^d y_t} \quad = \quad c + \underset{\uparrow}{(1 + \theta_1 B + \cdots + \theta_q B^q)} e_t$$

$$\text{AR}(p) \qquad d \text{ differences} \qquad \text{MA}(q)$$

Expression 6. Non-seasonal ARIMA model formulation. [3.]

In Expression 6, AR(p) and MA(q) components are marked. Multiplying yt by (1-B)d, a differentiated time series of order d is obtained. This model is called ARIMA(p,q,d) model [3], that has the following main parameters:

- p - order of the autoregressive component;
- d - degree of first differencing of a time series involved;
- q - order of the moving average component.

The other parts of the ARIMA formulation described as follows:

- e_t - white noise
- c - constant, that describes a drift, e.g. positive or negative trend in the time series
- $\phi_1,\ldots,\phi_p$ - parameters of the autoregressive component
- $\theta_1,\ldots,\theta_q$ - parameters of the moving average component [3].

To construct the model, parameters p,d,q as well as c, $\phi_1,\ldots,\phi_p$ and $\theta_1,\ldots,\theta_q$ must be determined. This can be difficult to do [3], but this process is automated by statistical software packages, such as R [6].

Seasonal ARIMA models can be obtained by adding seasonal component to the non-seasonal ARIMA models. Seasonal components are also expressed in ARIMA(p,d,q) form. Altogether, it can be written as follows:

$$\text{ARIMA}\quad \underbrace{(p,d,q)}_{\substack{\uparrow\\ \text{Non-seasonal part}\\ \text{of the model}}}\quad \underbrace{(P,D,Q)_m}_{\substack{\uparrow\\ \text{Seasonal part}\\ \text{of the model}}}$$

Expression 7. Seasonal ARIMA model formulation. [3.]

Parameters in the upper-case in Expression 7 are parameters for the seasonal parts of the model. In this notation, m is number of periods in a season. For constructing seasonal ARIMA model three more parameters needed to be determined - P, D, Q as well as number of periods per season m.

As an example, seasonal ARIMA(2,2,2)$(3,1,3)_4$ can be roughly explained in the following way. (2,2,2) part of the model describes non-seasonal the following correlations in the time series differences 2 times: the next value in the time series, depends on 2 previous values and 2 previous forecasting errors. (3, 1, 2), similarly, describes correlations in the seasonal pattern on the data: the seasonal pattern for the next period of 4 data points depends on the patterns for the 3 previous periods and forecasting errors for seasonal patterns for the 2 previous periods.

2.3 SVR

SVR or support vector regressions is adopted and extended version of support vector machines. Support Vector Regression Support vector machines or SVM are supervised learning models, that were originally developed to solve classification and pattern recognition problems. SVR solves nonlinear regression problems. [9.] Regression is the method of finding relationship between predictor values and the outcome. For example, product demand can depend on the day of the week. In this case the predictor value is the day of the week and the product demand is the outcome. In the case of nonlinear regression, outcome value is modelled by a function which is a nonlinear combination of predictor values. [10.] SVR were successfully applied for time series prediction in various fields, such as predicting electric load [8] and forecasting financial time series [9].

Given the training data set of

$$\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_N, y_N)\} \quad \text{with} \quad \vec{x}_i \in R^m \quad \text{and} \quad y_i \in R$$

Expression 8. Training data set. [11.]

where $\vec{x_i}$ is the input vector, and y_i is the desired output for that input [11], SVR approximates this m-dimensional input data as a linear function in a feature space F of a different dimension. The function has the following form:

$$f(\vec{x}, \vec{w}) = \langle \vec{w}, \Phi(\vec{x}) \rangle + b$$

Expression 9. SVR formulation. [11.]

Here $\langle . \ , \ . \rangle$ is the inner product operator defined in F dimension, Φ() is a non-linear mapping from m-dimensional input vector to F, $\vec{w} \in F$ is a weight vector to be estimated and b is a tuning bias parameter [11].

The $\vec{w}$ vector is estimated on the training data set, so that the outputs that the f function gives for input vectors has a minimal error with corresponding desired outputs. This optimization problem can be solved with quadratic programming. However while solving the optimization problem, two more tuning parameters will emerge. Together with bias parameter b, for SVR to function, 3 parameters are needed to be chosen. [11.]

SVR is a more general technique than ARIMA but it can also be used for predicting time series and proved to have good performance in various fields [4;8]. However SVR is not aware that the input data is a time series, unlike ARIMA, that is designed for time series prediction. Therefore SVR, for example does not give more priority to more recent values in time series when predicting the future values. [8.]

3 Review of Existing Production Scheduling Systems

3.1 In-house Production Scheduling System

The existing application I have built for the client was able to assist a manager to optimize the production schedule in the following way:

1. A manager creates the production schedule by creating batches in the application as she would usually do on the paper.
2. Employees, while working on the cafe, produce products, following the amounts, specified in those batches. They are able to see the specified amounts of the products in the application.

3. They log actions that they do at the bistro:
    a. Producing products
    b. Selling products
    c. Trashing products.

    The user interface of this phase is shown in Figure 1 below.

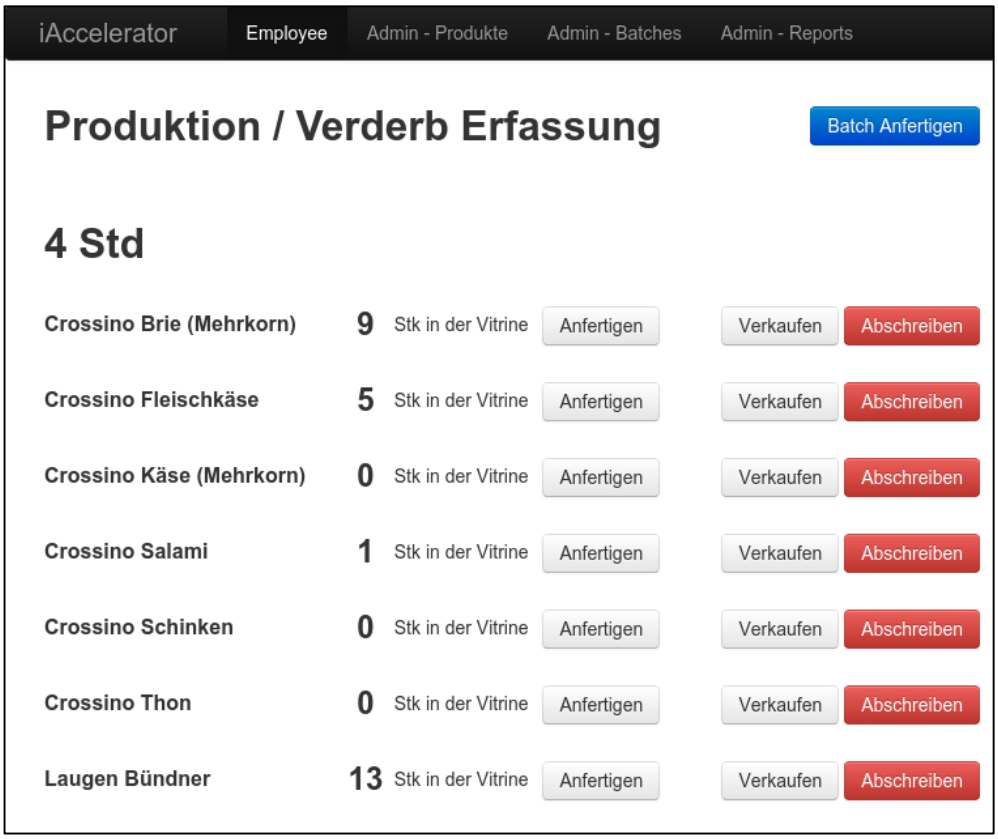

Figure 1. Cafe Logging Interface (in German).

The interface is in German. In Figure 1 all products that the client has are shown together with how many of them are available on the vitrine. To log produce, sell and trash actions of individual products, buttons Anfertigen, Verkaufen and Abschreiben are used. To log production of a batch, Batch Anfertigen button is used.

4. Based on those logged actions reports are generated for each week.
5. Let us defined trash rate as a the ration of a trashed product's amount in the end of a batch period to produced product's. If from the reports the manager sees that trash rate for a particular product is too high at a particular day in the week in a particular batch period, and this pattern repeats over several past weeks then the production of that product should be lowered for that batch period. In this case the manager can adjust the product's amount in the relevant to meet the demand.

E.g. if the product A's trash rate is about 30% on Mondays for a batch time 6am - 10am, the manager can adjust the amount of the product A's to be produced to be decreased by 30% for that batch.

The reporting interface of phases 4 and 5 is presented in Figure 2 below.

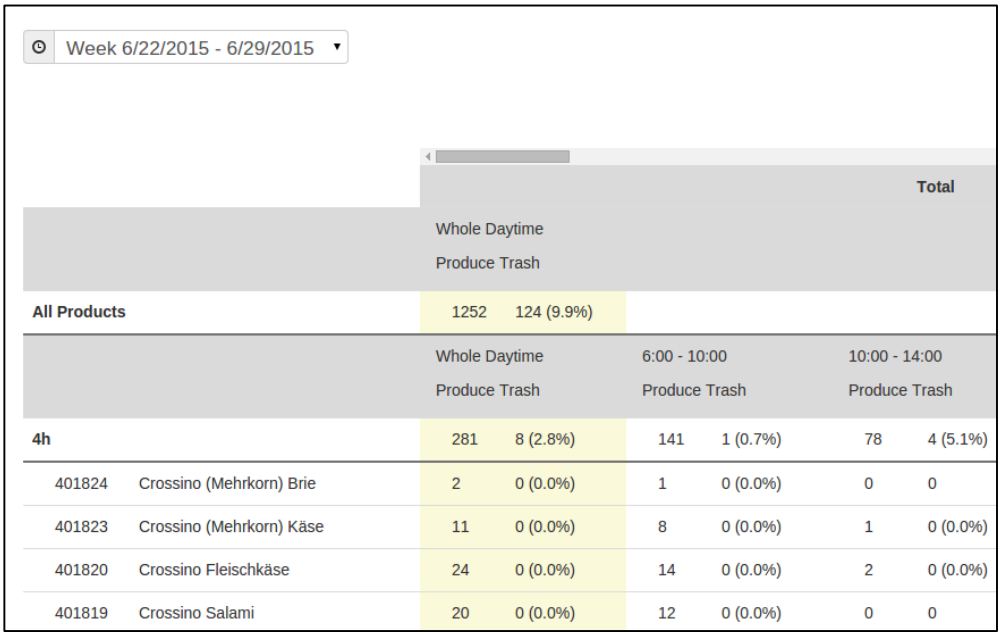

Figure 2. Reporting interface of the application

In Figure 2 the report is displayed for a selected week. On the vertical axis, all the products that are produced in the cafe are listed, grouped by their lifetime. On the horizontal axis, the time frame is displayed - Total for the selected week, and then separately for the each day. Inside each of those time slots Produce and Trash amounts are displayed for each batch period. Along with the Trash amounts, trash rate in percentage is shown.

The method described above has two issues. Firstly, it utilizes subjective reasoning of a manager that might be not systematic way to minimize product shrinkage and prone to errors. E.g. patterns in the changes of product's demand might behave in a more complicated way that manager can assume. Secondly, the method described above is not an automated process. The manager might need to do the batch adjusting process every week or even every day, to keep up with the changing demand, which is time-consuming.

3.2 Market Review: Production Scheduling Solutions

Some other existing software packages that could already solve the problem of optimal production scheduling are briefly reviewed here. However those software packages are proprietary and would need to be purchased to be fully evaluated. It was not possible to find the documentation or detailed specifications for those packages. Therefore mostly introductory information, published on the software products' official websites was reviewed.

After reviewing the competition I came to the conclusion that there are possibly only a few tools available that could potentially be helpful for smart scheduling of production in cafes. The first of the solutions reviewed were JDA Software packages [1]. JDA Software is a large-scale developer of supply chain, retail and omni-channel solutions for industry. It has various software solutions targeted for different needs. [1.] An article in Convenience Store and Fuel News website [12] mentions one of the software packages that is well suited for fresh-food producers, like petroleum cafes. It is said that the solution is able to reduce the shrinkage by providing scheduling and reporting interfaces [12]. In addition, it is mentioned in [12] that the solution is capable of predicting the demand for products.

Although the features mentioned in [12] are promising, the article was quite outdated and no such software packages were found on the official JDA Software website [1]. There was a similar solution on the website, called JDA Fresh Item Management. This solution promises demand-driven preparation schedules based on forecast. However, the solution seems to be designed primary for retail stores and there was no evidence found that it will be suitable for the petroleum cafe case. Furthermore, the solution's cost might be unsuitable for small franchises such as the client's company. [1.]

Another solution available is called Periscope [2], developed by Invatron systems Corporation. It is a fresh item management solution, designed for retail stores. Unlike JDA's solution, it is more focused on minimizing loss of fresh products and increasing profits. It is also capable of predicting the demand, and it's engine aims to "include promotions, holidays and special events, time of day, day of week, overall sales fluctuations and beginning- versus end-of-month demand in its forecasts" [2]. However again, it was not mentioned that the solution can be suitable for my client's case. Targeted for a bigger retail shop chains [2], once again, its cost might be unsuitable for small franchises such as the client's company.

The market review shows that there are solutions that are capable of predicting the demand for the product and optimize production scheduling. However, those are targeted mostly for retail stores and might be not suitable for the specific client's case, as some of client's products, such as sandwiches have quite short lifetime of 4 hours. Furthermore, solutions' price is likely to be too high for small franchises. Therefore it is feasible to incorporate demand prediction and smart production scheduling systems into the application for the client, to target specifically small petroleum station franchises owning cafes. The problem and the goal of the thesis are stated in more details in the next chapter. [1;2.]

Review of those two competitor's solution gives important insights on what could affect the demand of the product:

- time of day
- day of week
- promotions
- holidays
- special events [2].

Those parameters are good to take into account, when designing a prediction model for product demand.

4 Models for Forecasting Product Demand

4.1 Analysing Product Data

It was only possible to receive from the client 9 days of product data. This data was analysed. The data came from trash log spreadsheet that the client used to document production and shrinkage of all products in the cafe for each day. The data has information on what time each product was produced and trashed and in which amounts. However the data had the following issues:

- Short - only 9 days. This does not allow to test prediction models on this data.
- Errors and uncertainties. After consultation with the client it was confirmed that some data points could be incorrect, e.g. not clear at which batch time product was produced and sold, especially for non-morning batch times.

The data was analyzed for the first batch time for each product group: 6am - 10am for 4h products, 6am - 12pm for 6h products and 6am - 6pm for 12h products. The data was digitized and analyzed with the R programming language, using various plots and statistical methods.

4.1.1 General Statistics

The client has about 70 different products that are prepared every day. Different product groups - 4 h, 6 h and 12 h vary in several parameters. They have different production amounts and trash rate. It is important to know, how much products are trashed. The analysis shows the following information, shown in Table 1 below.

Table 1. Average trash frequencies and rates for each product group.

Product group	4 hours	6 hours	12 hours	Total
Average trash frequency	8%	27%	42%	26%
Trash rate	5%	6%	18%	10%

Table 1 reveals that a considerable amount of products (10%) is not sold every day and is trashed. Twelve products are the most often trashed products with a considerable trash rate of 18%. This product group would be a priority when designing a prediction system.

4.1.2 Unknown Demand

As the average trash frequency is 26% it also means that the information about the actual demand is available only in 26% cases for this dataset. Assuming that it will be so for other time windows means that only in 26% of the demand time series for each product actual demand is known. In the rest of 74% of the demand time series the actual demand is unknown and can be more than assumed demand, which is equal to production amount of a product. Let us demand underestimate U, which equals to

$$U = P - D', \quad (1)$$

where D’ is the assumed real demand, P is produced amount of a product.

It would be useful to give an estimate of how much the value of the actual demand is more than the assumed demand, that is, of demand underestimate. From the client’s data it was observed that often after initial production of product in the beginning of a batch time, additional amount of product is produced. This additional production could be done several times during the batch time. After consulting with the client, it was confirmed that if the product is sold out, employee can produce extra amount of product, to meet the demand, even though it would be out of production schedule. This means that the total produced amount of product during the batch period is close to the real demand. To further analyze the underestimated demand, it could be useful to analyze trash rate distribution, that is, the relation to amount of trashed product to produced amount. The histogram is shown in Figure 3 below.

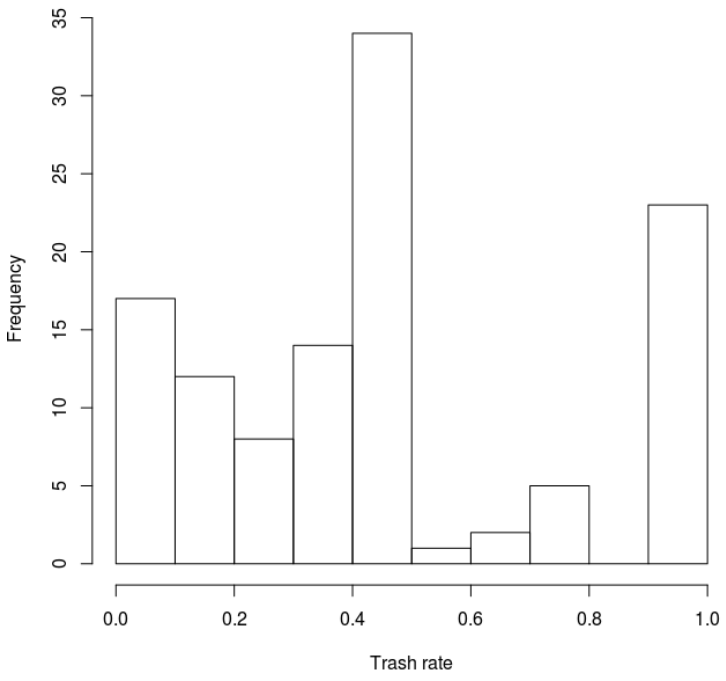

Figure 3. Distribution of trash rate for client’s products, with non-zero trash amount.

However the distribution is quite random and does not resemble normal distribution with a mean close to 0 that would be expected. If the histogram had a similar form as the right half of the normal distribution, with mean close to 0, that would give an assumption that the demand underestimate distribution could also be normal distribution. Therefore the distribution of the demand underestimate is questionable.

4.1.3 Trend Analysis

There was no evidence found that there was a common trend for the demand for products in each product group. However demands for products were found to follow a similar pattern during the week. An example is demonstrated in Figure 4 below:

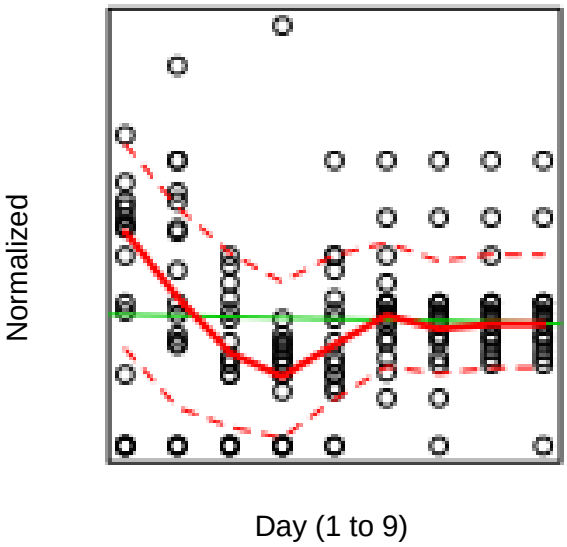

Figure 4. R “car” package plot of day against normalized demands for 4 h products.

Here normalized demands for all 4h products are plotted. Generally the demand drops until day 4 (Sunday) and then rises. That proposes the assumption that the products demands change similarly reacting on external events. Similar observations were noticed for other product groups.

4.2 Validation Criteria

Performance of the models were verified on 3 time series that will be described in section 5.1. Each dataset will be broken in two parts - training data set, used to train the algorithms, and test dataset, used to evaluate forecasting accuracy. Test datasets were not used for training of models.

For consistency for each time series two last weeks of its data were used as a test dataset. If only one last week would be chosen as a test data set, that would not be sufficient for evaluation, as in that case only 7 data points would be used to calculate error. Such evaluation would give a lot of uncertainty in the actual performance of models. On the

other hand, one of the test time series used to evaluate the models had only 7 weeks of data. Using 3 weeks of data as test data set would leave only 4 weeks as training dataset. That would not be sufficient volume of data to train the models.

When evaluating the models, or in other words, calculating forecast accuracy, models are required to give one-step-ahead forecast for all the days in the test dataset. In other words, to predict the demand for each day in the test set, all data prior to this day is fed to models. This simulates real situation, as the forecast needs to be done in the beginning of each day, when producing new batch.

Given testing dataset T = $\{y_{N-13},...., y_N\}$ (last 14 data points in a selected time series) and corresponding one-step-ahead forecasts F = $\{y'_{N-13},..., y'_N\}$ there are several possible ways to calculate forecast accuracy.

One method is to examine the profit gained from selling the product if a model suggests to produce an x amount of the product. In the client's cafe each product has producing cost c and it is sold at price s. However if product is trashed there is an environmental penalty p (product has to be properly recycled and it is not environmental friendly to produce more than needed). Then the profit gained if producing an x amount of the product during a specific batch period is

$$G(x,d) = min\{x(s-c), ds - (p+c)(x-d)\} \quad (2)$$

where d is the demand for the product during a batch period, that is, how many items of this product would be bought. The first term of min is the profit, when x <= d and second term of min is profit when x > d. This is a linear piecewise function with a maximum attained at x = d. If the demand is known the best decision will be to produce d amount of product. [13.]

However for formula 2 it should be noted that if the demand is not met, there and x<= d in client's domain case, employee can produce additional amount of product. Therefore it could be, vise versa, more beneficial to produce less that the demand, unlike the formula states. However, as this fact is uncertain, for simplicity, G(x,d) will be used.

As was reported by the client, for some products, sale price of a product is 12 times more than the total price of ingredients to produce that product. However to produce the product employee needs to spend time and use tools and electricity. Those resources could

be spend on producing another product instead. Therefore for evaluation cost c to produce product will be 6 times less the sale price. In summary for G(x,d) the following parameters were chosen:

- c = 2
- p = 1 (environmental penalty is assumed to be less than cost of production)
- s = 12

Then the models can be evaluated based on ratio of the total profit that model yields to maximum total profit possible during the period of time in testing dataset T. Total Profit Ratio or TPR will be used as one of accuracy criteria. I defined it with the following custom formula:

$$\mathrm{TPR} = \sum_{j=1}^{14} \mathrm{G}(y'_j, y_j) / \sum_{j=1}^{14} \mathrm{G}(y_j, y_j) * 100\%, \text{for } y_j \in T \text{ and } y'_j \in F \quad (3)$$

Another option would be to use the mean of the profit differences between maximum possible profit and profit that forecasting model yields. However it would then diminish importance of making maximum profit when demand is high. That is it, it is more important to meet the demand when it is high, as it yields more profit than when the demand is low.

The parameters for the profit function can be different from product to product and in reality cost of production can be actually higher. Plus it could be more desirable to minimize shrinkage by sacrificing profit. Therefore as a more universal benchmark Mean Absolute Error (MAE) could be used, which is a common choice for evaluating forecast accuracy [3]. Mean Absolute Percentage Error (MAPE) is another common choice [3], however it is not suitable for time series that could have zero values, which is the case with demand time series. Furthermore, profit gained from selling product linearly depends on difference between demand and amount produced.

In order to compare the performance of prediction models between time series that have different average demand values, Normalized MAE (NMAE) is used, and that is normalized as follows:

$$\mathrm{NMAE} = \mathrm{MAE}/\mathrm{mean}(T), \quad (4)$$

where T is a testing data set.

Another important accuracy measure would be trash rate, that is shrinkage proportion of the product on average. I defined trash rate TR with the following custom formula:

$$\mathrm{TR} = \mathrm{mean}((y'_j - y_j)/y'_j) * 100\%, \text{ for all } y_j \in T \text{ and } y'_j \in F, \text{when } y'_j > 0 . \quad (5)$$

This will allow to make an important comparison with examined trash rates in client's cafe.

To summarize, NMAE, TPR and TR will be used as forecast accuracy measures of algorithms. Lisi and Schiavo [5] used the proportional error reduction (per) to compare the improvement of forecasting accuracy. To make differences between forecasting accuracies of different models more standing out for each accuracy measures, a variant proportional error reduction will be also utilized. For NMAE, per will be defined as

$$\mathrm{per} = 1 - \mathrm{NMAE}_{\mathrm{ml}}/\mathrm{NMAE}_{\mathrm{wm}}, \quad (6)$$

where $NMAE_{ml}$ is NMAE of a target model, and $NMAE_{wm}$ is NMAE of the worst-performing model on a chosen dataset. For TR it will be done the same way. For TPR, as it is it's better to talk about proportional improvement - pi, that I defined with the following custom formula

$$\mathrm{pi} = (\mathrm{TPR}_{\mathrm{ml}} - \mathrm{TPR}_{\mathrm{wm}})/\mathrm{TPR}_{\mathrm{wm}}, \quad (7)$$

where TPR_{ml} is TPR of a target model, and TPR_{wm} is TPR of the worst-performing (having minimum TRP) model on a chosen dataset.

4.3 Implementing Selected Forecasting Models

4.3.1 Simple Probabilistic Model

I adopted a probabilistic model from Shapiro A. and Philpott A. [13] and named it the Simple Probabilistic Model, as it makes use of only a limited set of input data. The simple Probabilistic Model emerges from the following observation of the profit function described in formula 2. The decision to produce an x amount of a product should be made before the demand was known. Then one option is to view the demand in formula 2 as

random variable D. Then the objective function becomes G(x,D). In this case it is assumed that probability distribution of D is known. Since D is random variable, it makes sense now to talk about expected profit E[G(x, D)] [13]. Then optimization problem for optimal production amount take the following form:

$$\max \mathrm{E}[\mathrm{G}(\mathrm{x}, \mathrm{D})] \text{ for } \mathrm{x} \geq 0. \quad (8)$$

Since D and x takes only discrete integer values and assuming that cumulative distribution function F(z) = Prob(D <= z) is known for D then the optimal production amount would be

$$\max \{\sum_{\mathrm{d}=0}^{\mathrm{maxD}} \mathrm{G}(\mathrm{x}, \mathrm{d})(\mathrm{F}(\mathrm{d}+1) - \mathrm{F}(\mathrm{d}))\} \text{ for x from } 0 \text{ to maxD}, \quad (9)$$

where maxD is the maximum possible demand for the product. That is it to find optimal x, for all possible x expected profit is calculated. In it clear, that F(d+1) - F(d) equals to probability of the demand to be d.

For this model it is assumed that distribution of demand for product for each batch time on day t depends only on weekday of t. That is, F(z) is calculated based on the demand on the same weekday of day t for previous k weeks. It is wise to limit to previous k weeks, as the demand for the product can have a trend and in this case old demands would be too high or too low compared to the recent demands, making them unlikely to happen. Then F(z) for a chosen product batch time for day t calculated as shown in the following pseudo code below:

```
for each z from 0 to maxD do
          weeksCount = 0
          for each week from 1 to k do
                    if d[t - week*7] <= z then
                weeksCount = weeksCount + 1
          F(z) = weeksCount / k
```

Listing 1. Pseudo code to calculate F(z)

In this code d[t] is demand for day t. This model assumes that d[x] is known for all previous days x, that is, it assumes that probability of any demand d' > d[x] is 0.

Implementing formula (9) and calculation of F(z) yields simple probabilistic model that calculates optimal produce amount for upcoming day t, based on demands of previous days. This model was implemented using R language. Cumulative function F(z) is used instead of probability function, as cumulative function is more flexible to incorporate information, when the demand is unknown for a given day. This information is not used in the implementation currently and demand is assumed to be known for each day.

There is one parameter to be determined for this algorithm to work - k. If k is smaller, it means that bigger priority should be given to recent demand data. That is it, if weekly pattern changes quickly, k should be smaller. To determine k, I came up with the following algorithm:

1. Break training data set into local training dataset and local testing dataset. That is not to be confused with “global” training dataset that is fed to the algorithm and “global” out-of-sample dataset. 4 last weeks of global training dataset were used as local testing dataset, which is 2 times more than in “global” testing dataset. This is done to avoid overfitting.
2. Now if treating the local training dataset as a global training dataset, and the local testing dataset as an out-of-sample dataset, choose k with the best TPR (Total Profit Ratio) yielded when verifying against out-of-sample dataset (which is in fact just part of global training dataset). All feasible values of k are tried, from 1 to length of local training dataset.

The partial source code of the model can be found in Appendix 1. The model’s core function is calcOptimalProduceAmount.

4.3.2 ARIMA

Seasonal ARIMA models were used in this project, as the generated time series were generated to have a seasonal pattern. Seasonal parameter m in the notation in Expression 7 will be set to 7 for weekly seasonality. The parameters of seasonal ARIMA can be chosen manually by analyzing auto-correlations in the time series or automatically, by a software package. R programming language was used to automatically determine parameters, build and evaluate ARIMA models on the time series.

To determine terms of seasonal ARIMA model for each time series R function auto.arima() was utilized to fit ARIMA model on each testing time series. It is worth to

note that different software uses different algorithms to determine arima model. Function auto.arima in R uses Hyndman and Khandakar algorithm[3], which combines several benchmarks together when evaluating the performance while finding optimal terms. [3.] Here is the source code that fits the seasonal ARIMA model on the training data set and returns one-step-ahead forecasts to compare with out-of-sample testing data set:

```
predictArima = function(y, testingSize) {
    y = y$Produce - y$Trash
    to = length(y) - testingSize
    y = ts(y, frequency=7)# converts to time series object
    fit <- auto.arima(y[1:to], seasonal=TRUE)
    newfit <- Arima(y, model=fit)
    return (fitted(newfit)[(to+1):(to+testingSize)])
}
```

Listing 1. Seasonal ARIMA model fitting source code.

In the Listing 1 input parameter y contains Produce (`y$Produce`) and Trash (`y$Trash`) time series. Product demand is the difference between production amount and trash amount. Function `auto.arima` is used to find optimal parameters of the seasonal ARIMA and use them in function `Arima` that computes one-step-ahead predictions.

4.3.3 SVR

As it was mentioned in section 2.3 SVR were successfully applied for time series prediction in various fields. In addition, SVR can accept various kinds of training data [8]. For example, it would be possible to utilize any parameters to make the forecast for the demand of product, such as day of week and weather information.

SVR maps input or regression vectors x_t into responses values y_t. To predict the upcoming day's demand y_t 2 previous weeks of lagged values of demand were used. In addition, as the target time series contain repeating weekly pattern, a day of the week for the time t is also used as input data. To summarize, to predict the product demand y_t with SVR vector x_t was constructed as follows:

$$x_t = [y_{t-1}, \ldots, y_{t-14}, W_t] \quad (10)$$

It contains 14+6 = 20 values - 14 lagged demand values for 14 previous days and 6 binary values in W_t. W_t represents the day of the week by binary values. E.g. for Tuesday W_t= [0, 1, 0,...]. It is good to note, that there is no need for the 7th value for Sunday, as Sunday can be represented, when all values of W_t are 0. In this way, less values are used for regression vector, improving accuracy of the model.

To evaluate accuracy of each combination of parameters, while selecting optimal ones, the training data set is divided in the same way as in case of simple probabilistic model. The training data set is broken into a local training dataset and last four weeks of training dataset are a local testing dataset.

The algorithm used mean average percentage error MAPE as a benchmark for choosing optimal tuning parameters while verifying against local testing dataset and to evaluate the performance against out-of-sample testing data set. However MAPE is not suitable for the target demand time series, as it does not accept values of 0 value, and the demand can be 0 for some days. In addition, MAE is chosen to evaluate and compare performance of prediction models. Therefore mean absolute error or MAE was used instead of MAPE. More details on the algorithm cannot be provided due to the non-disclosure agreement.

5 Evaluation

5.1 Evaluation Setting

The prediction models need to be evaluated on the three demand time series that simulated various possible properties of demand behaviour for different product types. However, there was not sufficient data received from the client's domain to have reasonably long demand time series. Therefore, one time series was taken from a similar real domain and two others were programmatically generated to simulate various properties of possible product demand behaviour.

5.1.1 Time series #1 - Real Data from Another Cafe

The real life time series were taken from the same domain – students' run cafe [14]. The time series is a daily sale amount of muffins at the cafe. The time series is shown in the plot below:

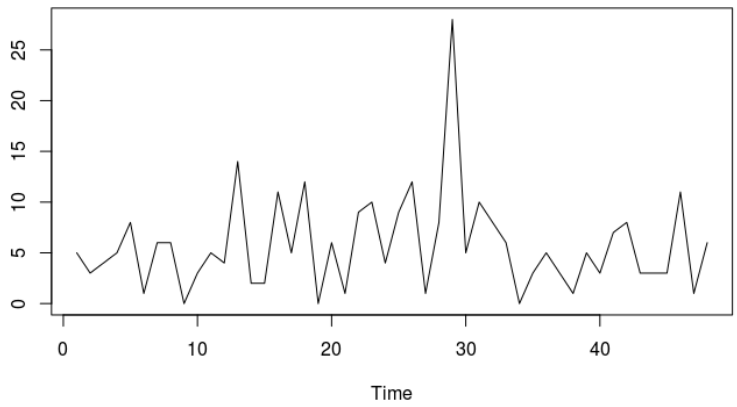

Figure 5. Dayli muffin sales at the students' run cafe.

The dataset has the following properties:

- 48 data points, or almost 9 weeks
- An outlier at day 29
- No trend or local trend
- Does not seem to have any repeating pattern or weekly seasonality
- Very close to client's domain as muffin refers to 12 h product group in the client's cafe. However this time series is a daily demand, not a demand during a batch time.

To observe how demand behaves for each day of the week, the demands boxplot is shown in Figure 6 below:

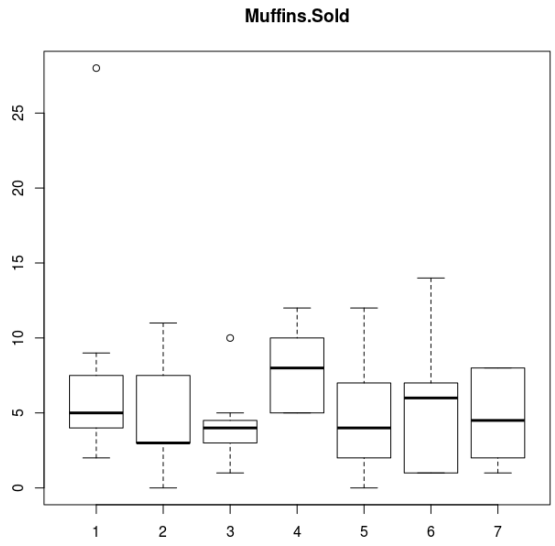

Figure 6. Box-and-whisker plot for daily demand by weekdays of muffins.

In Figure 6 the demand distribution varies between weekdays (horizontal axis, from 1 to 7). Particularly it is different for week day 3 and 4. However as only 7 weeks of data is available its seasonality is hard to confirm.

5.1.2 Time series #2 - Simulating 12 h Product Group

To generate time series #2 analysis of the client's domain was utilized. Time series #2 will aim to simulate a subset of products in 12 h product group. 12 h product group is the first priority to build prediction model for, as it's trash rate is the highest among other product groups. Subset of 12 h products with mean demand less than 6 was examined. The normalized demands boxplot is shown in Figure 7 below.

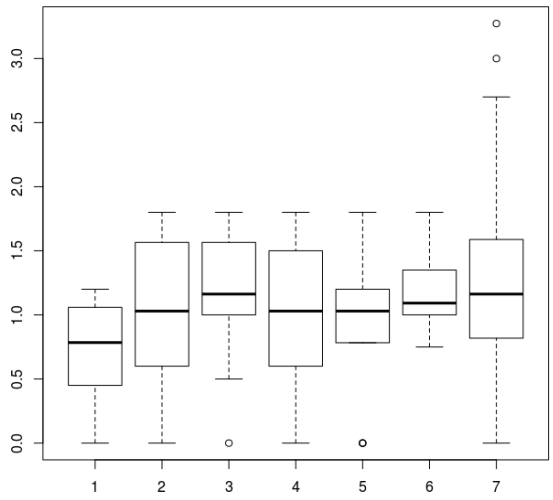

Figure 7. Box-and-whisker plot for the normalized demand of a subset of 12h products.

As can be see there is slight change in the mean of the demand, which changes smoothly during the examined week. However the demands can vary much from product to product. For #2 time series it was assumed that the demand of selected subset of products behave similarly. Therefore time series #2 were generated from the distribution of normalized demands for the selected products subset.

To make the time series more sophisticated and more difficult to be predicted, a trend was added, so that the mean demand changes during the time. However, analysis shows that for the cafe products the absolute value of relative demand differences between 2 consecutive days decreases, as the mean demand increases. The plot in Figure 8 below demonstrates that:

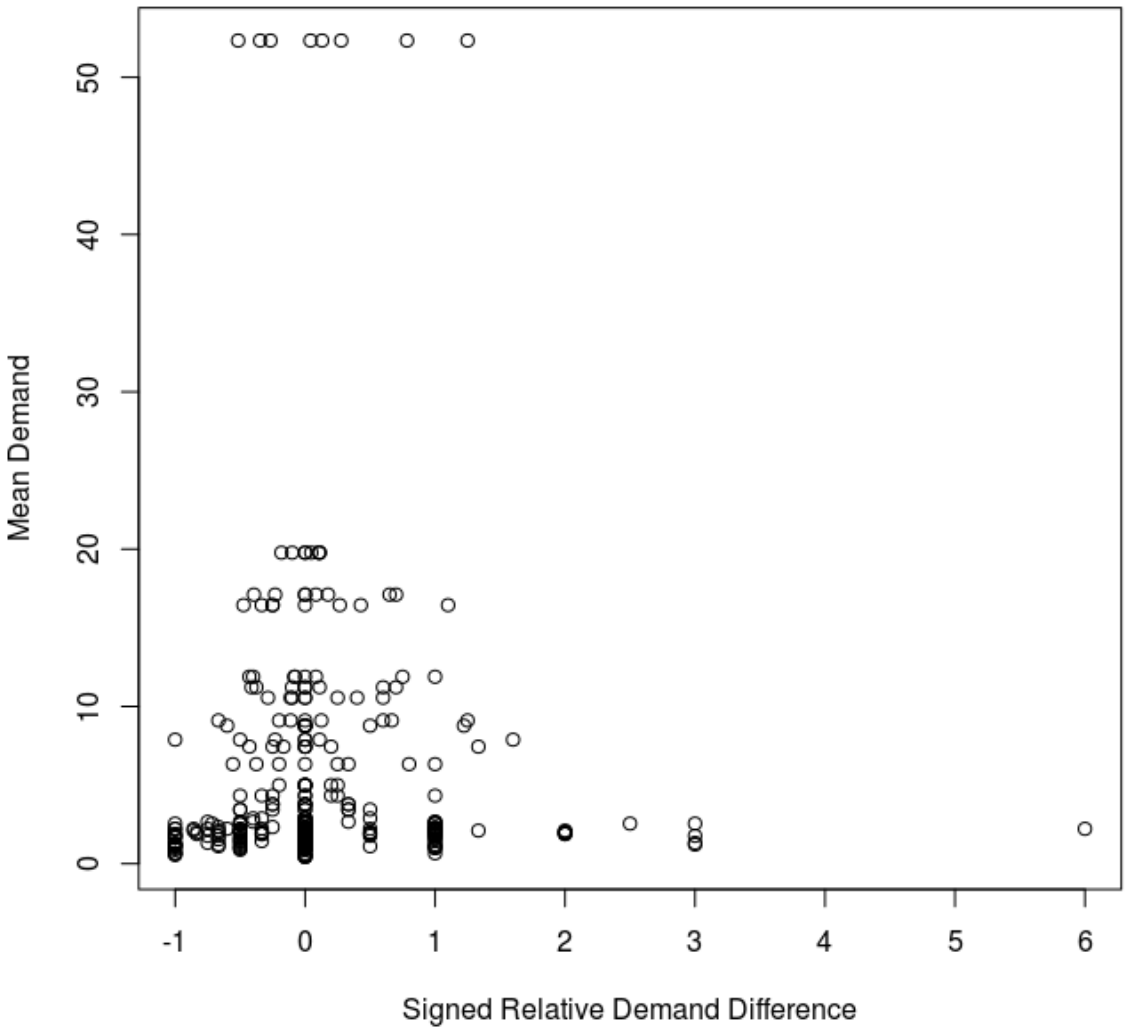

Figure 8. Relation between relative demand difference to mean demand for client's products.

In the plot the relative demand difference approaches 0 as mean demand increases, with exception of some outliers, for which the mean demand is over 50. The plot contains demand data from all the products from the dataset received from the client. Mean demand was calculated for each product separately as mean of the demand over the available window of the time - 9 days.

Therefore to make time series more realistic when generating the data, with increasing mean demand, relative demand variations from the average demand were penalized linearly.

The resulting dataset can be observed in Figure 9 below:

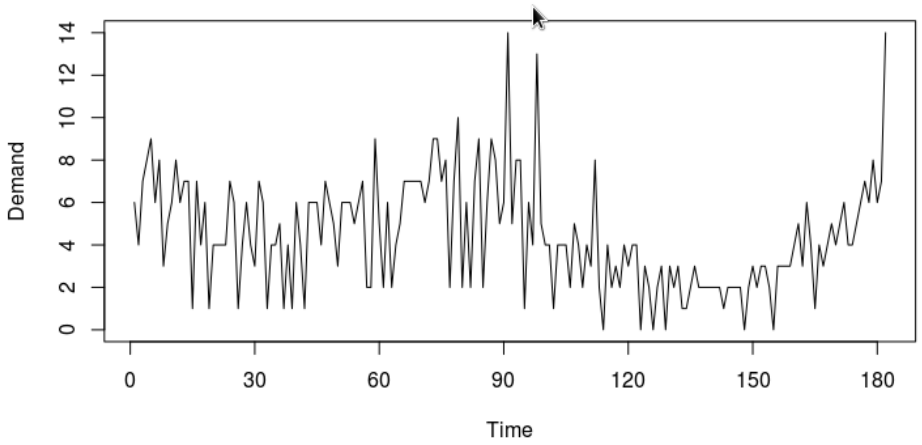

Figure 9. Generated time series #2.

The resulting time series has 182 data points or 26 weeks or approximately half a year. The resulting dataset has significant noise level but repeating pattern of the demand change. There is a change in the average demand during the time, which could be a seasonal changes: e.g. low demand during days 120-150 could be due to rainy weather in autumn months.

Figure 10 below shows the distribution of the demand per weekday for the generated data:

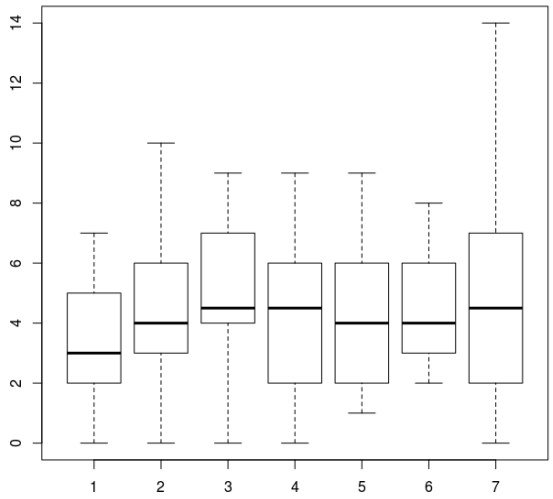

Figure 10. Box-and-whisker plot (Monday through Sunday) for the demand for generated time series #2.

Comparing with plot in Figure 7, the plot in Figure 10 follows similar pattern to the relative demands of the chosen groups of products from which this time series were generated.

Therefore it is verified that the weekly pattern of 12h products is not lost, while adding the mean demand changes during the time and penalizing demand differences with higher demands.

5.1.3 Time Series #3 - Simulating Products with Strong Weekly Pattern

Previous time series did not have a strong weekly seasonal pattern. However, it is good to consider that real demand data could have a stronger seasonal pattern. Plus the mean demand in time series #1 and #2 was relatively small. As the mean demand of products increases noise level should decrease too as relative demand difference decreases, which could emerge a stronger seasonal pattern. As the reference, daily demand time series of a perishable ingredient in the fast food restaurant was selected [4]. Fast food restaurant's domain is close to the client's domain, therefore demand behavior could be similar for products with bigger demands. The reference time series is shown in Figure 11.

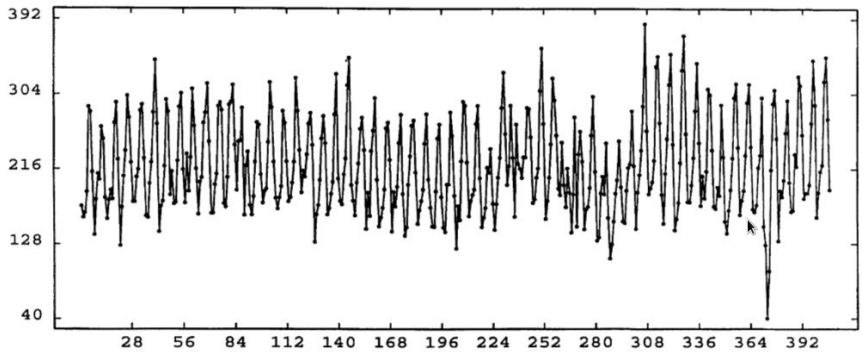

Figure 11. Daily demand of a perishable ingredient for a fast-food restaurant. [4.]

The time series in Figure 11 has strong weekly pattern, the peaks of demand are repeated approximately every 7 days. However, the amounts are much higher than the average demands of products in the client's domain. In addition there is no trend. The demand distribution per day is shown in Figure 12 below:

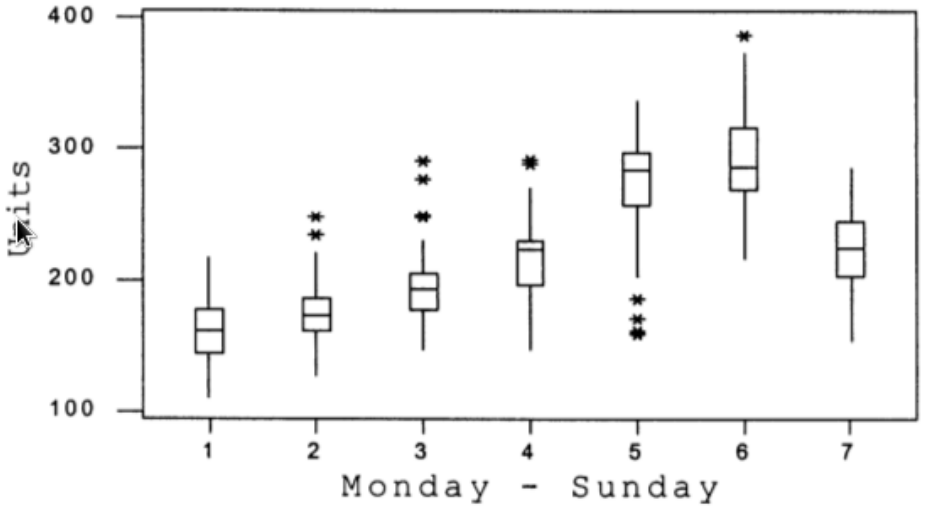

Figure 12. Box-and-whisker plot (Monday through Sunday) for the daily demand of a perishable ingredient for a fast-food restaurant. [4.]

In Figure 12 the demand for the component grows smoothly from Monday to Saturday, which can be observed by drawing a curve through median values of each day. Then the demand drops rapidly from Saturday to Monday. There is a number of outliers (marked with "*" on the plot) but generally demand distribution is quite distinct for each day.

Therefore the time series #3 were generated to have the following properties:

- Strong weekly seasonality
- No local trend
- Longer time series than #2, 350 data points or 50 weeks or approximately whole year
- Higher mean demand 18-20, simulating demands for some popular products in the client's cafe, such as croissants
- More complicated weekly pattern to stress test the algorithms
- Rapid changes in the demand but weaker than in previous datasets. Demand will be never 0.

To generate time series #3 the structural time series model R package was used, called "STSM" [15]. It is capable of generating random seasonal time series. The generated time series #3 is shown in Figure 13 below.

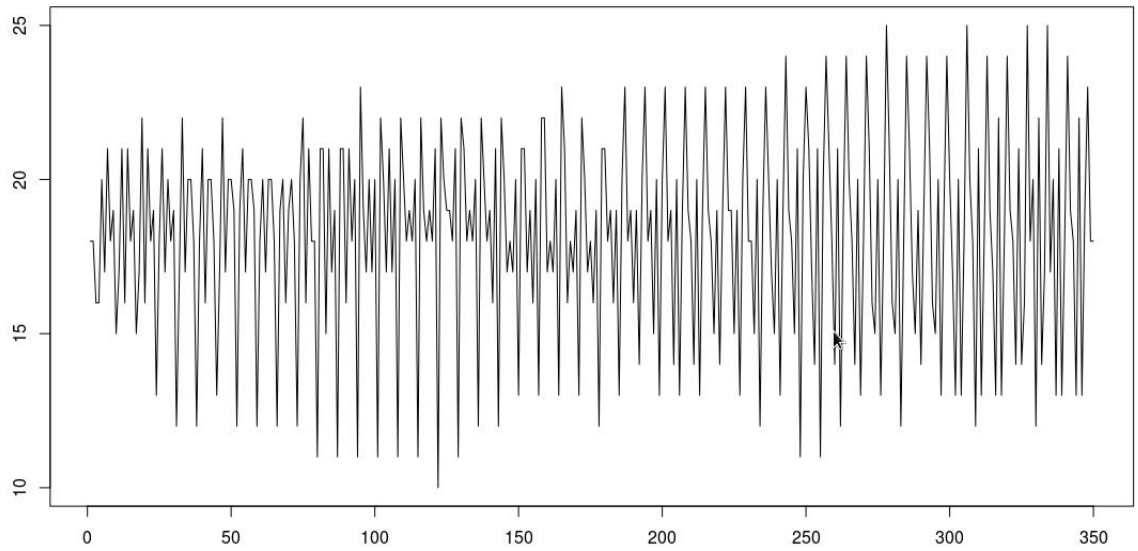

Figure 13. Time series #3.

The demand distribution per weekday is shown in Figure 14 below.

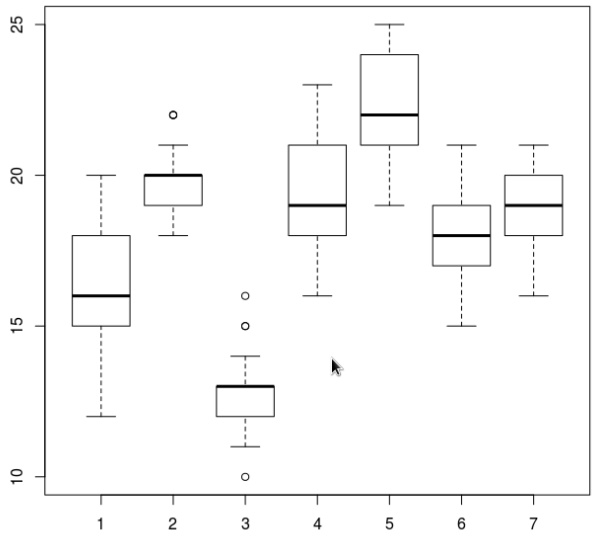

Figure 14. Box-and-whisker plot of the demand per week for time series #3.

The demand has weekly pattern as the distribution of demands for each week day is distinct. However the pattern changes slightly over the time, as high and low values of each week change over time in Figure 13. Demand amplitude changes as well over time in Figure 13. In other words, each week has similar basic structure but amplitudes within each week vary. Unlike time series in Figure 11, the weekly pattern although quite dis-

tinct, is more complicated. There is a rapid drop in the demand on Wednesday and demand distribution does not change smoothly. In addition, there are a number of outliers on Tuesday and Wednesday (marked with “o” in Figure 14), which would simulate some real situations, e.g. demand changes caused by holidays.

5.2 Running the Forecasting Models

To demonstrate performance of the models on the time series, firstly the forecasts plots of the models will be presented. In section 5.3 accuracy measures of the models will be presented and the models will be compared. The forecast plots of the models will be in the reverse order, starting with with the simplest time series - #3.

5.2.1 Time series #3

Below are predictions of each of the models (red color) against out-of-sample data in time series #3:

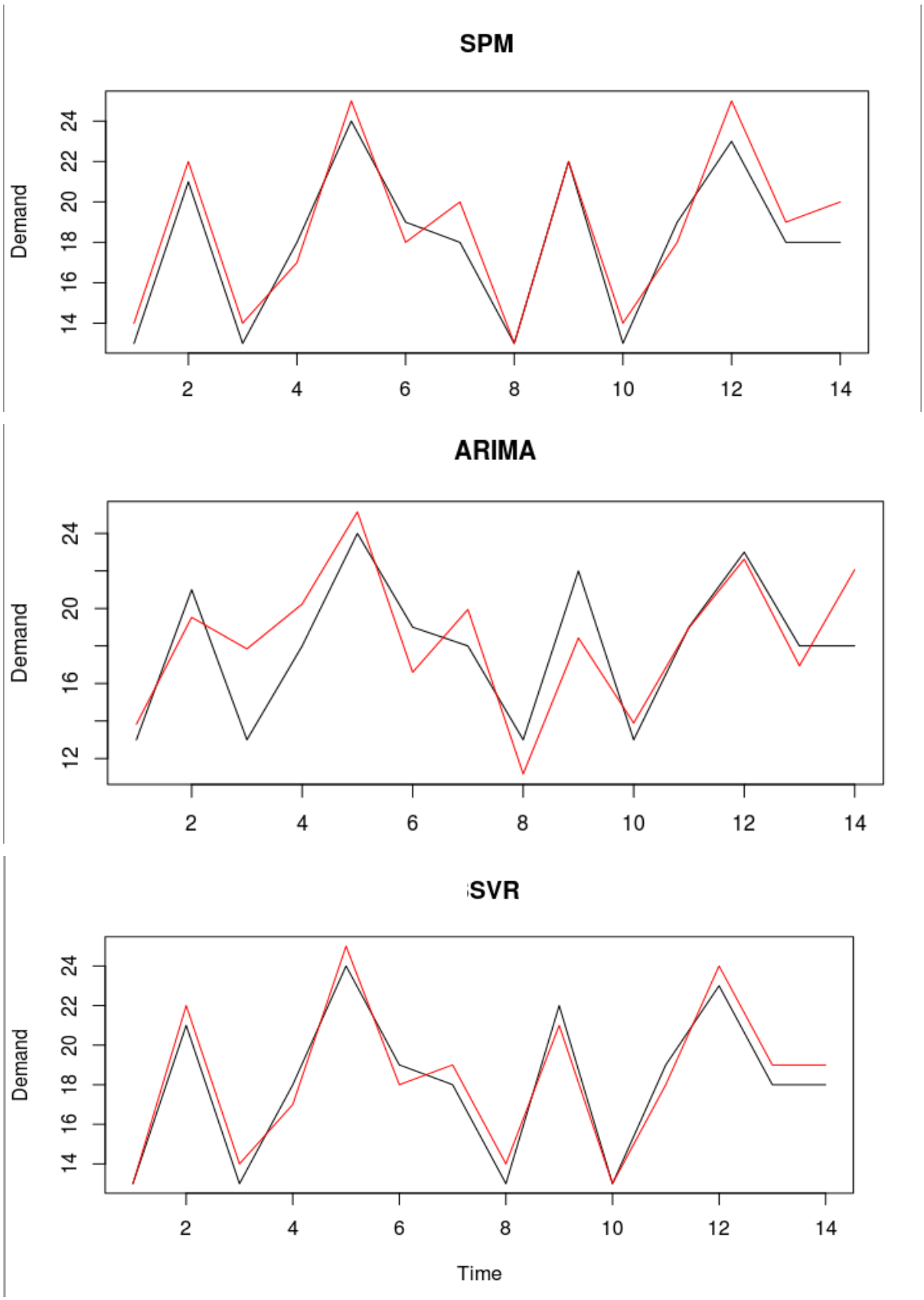

Figure 15. Predictions of each of the models (red color) against out-of-sample data (black color) for time series #3.

Here and in other predictions graphs, presented in sections 5.2.2 and 5.2.3, predictions of the models for each day are plotted against the actual demand. Demand is how many products customers wanted to buy in a day. In Figure 15 the values that the models estimate are different across the models for the same day. Forecasting models don't always predict the exact demand and either underestimate the value, e.g. ARIMA at day 8 of the out-of-sample data, or overestimate it, e.g. SPM at day 12.

5.2.2 Time series #2

Below are predictions of each of the models (red color) against out-of-sample data in time series #2:

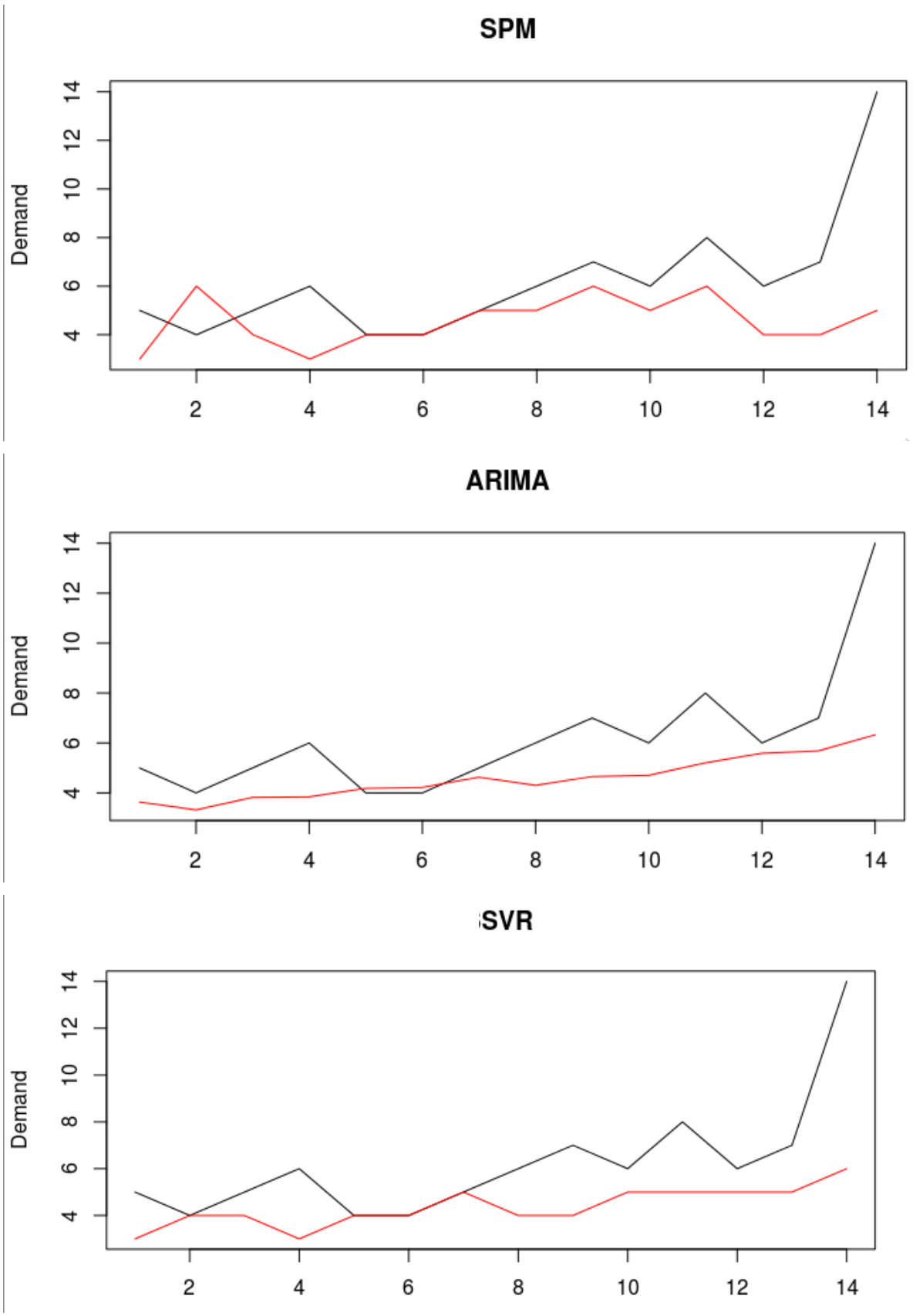

Figure 16. Predictions of each of the models (red color) against out-of-sample data for time series #2.

5.2.3 Time series #1

Time series #1 have the least number of data points - just 48 and is the most interesting one, as it is real data. This is a good test for verifying how algorithms will perform, if limited amount of data is available. If they will perform well, it means that just a few data needs to be known in order for a model to be deployed in the application and applied in practice.

13 last data points were taken as out-of-sample testing data set. This leaves out only 5 weeks of training data. Taking in account small amount of data the some adjustments were made for SVR and SPB. Previously 4 weeks were used for verifying algorithm parameters, before running out-of-sample test. However in the case of time series #1 this would not be possible as SVR algorithm uses 14 lagged values for each regression vector. Therefore 2 weeks of training data was used for SVR. This leaves out just one week of local training data to choose optimal parameters.

For SPM local training dataset was also reduced to 2 weeks. This means that the algorithm could only choose k from 1 to 3 weeks as a learning window. However it could not be then verified that 4 or 5 weeks of training data would be better. Therefore the following decision was made - if when running optimization 3 weeks yields the best profit ratio on local testing data set, 2 being second best, then there is a good evidence that 4 weeks of training window could be optimal. In this case, to allow to verity k up to 4 weeks. Local testing data set is reduced to 1 week, and optimal parameters are compared again. If 4 weeks is still the best candidate, it is chosen, otherwise 3 weeks are chosen. Following the above-mentioned algorithm, 3 weeks were selected as training window for SPM.

Below are predictions of each of the models (red color) against out-of-sample data in time series #1:

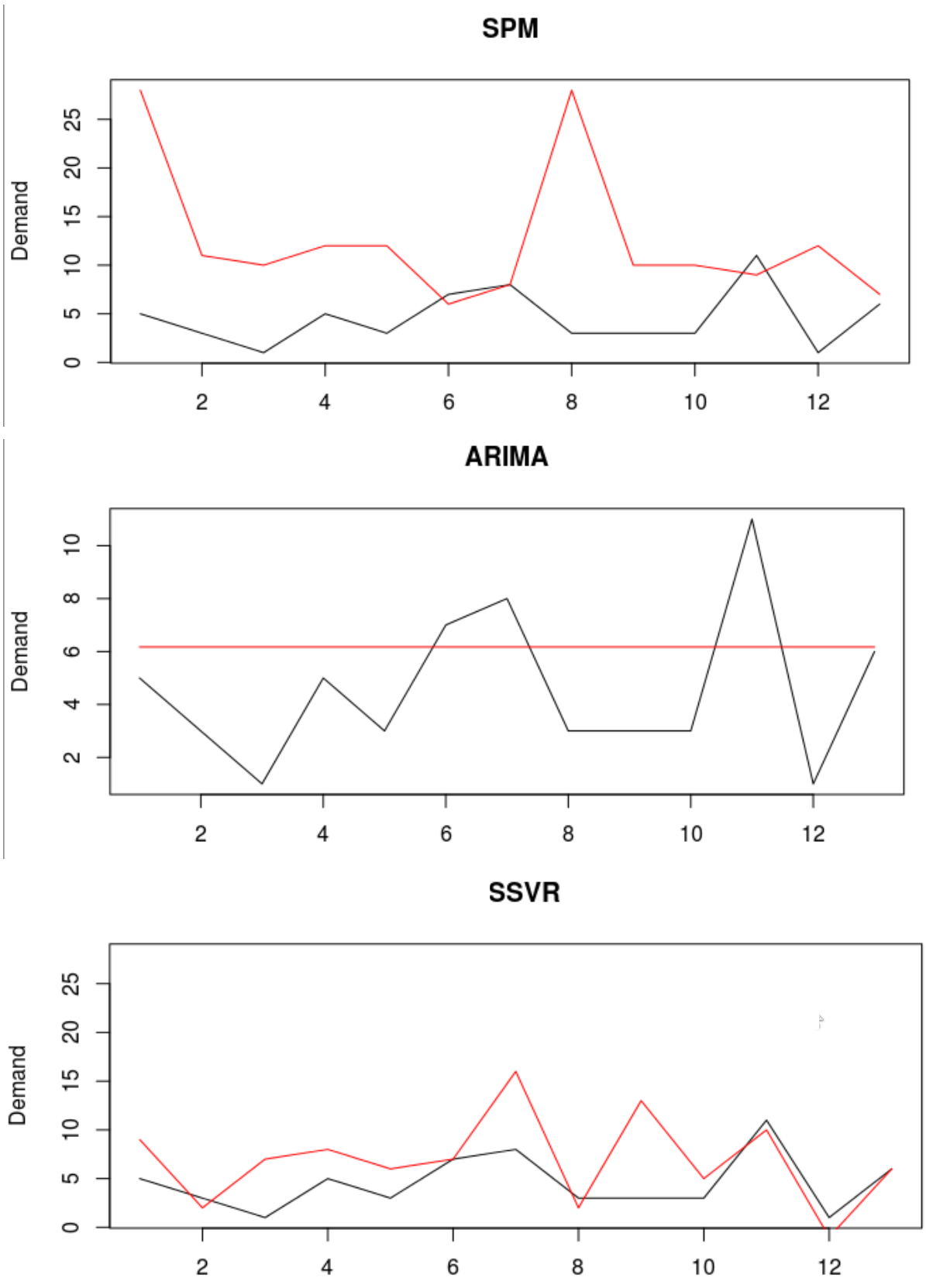

Figure 17. Predictions of each of the models (red color) against out-of-sample data for time series #1.

In Figure17 SPM, “remembering” the outlier with high demand that happened several weeks ago in the time series, overestimates the demand at day 8.Predictions of the ARIMA model are the same value for all of the days that were forecasted.

5.3 Evaluation Summary

Performance measures are presented in Table 2 below:

Table 2. Comparison of prediction models for test time series. The best measures among candidates are highlighted in grey. As defined in section 4.2, here NMAE - Normalized Mean Average Error, TPR - Total Profit Ratio, TR - Trash Rate, per - proportional error reduction, pi - proportional improvement. For each timeseries and accuracy measures (NMAE, TPR and TR) there are two columns - left one with the absolute value, and right one with the relative value (either per or pi).

Model	**Time series**	**Time series #1**		**Time series #2**		**Time series #3**	
SPM	**NMAE (per)**	1.8644	0.00%	0.3103	0.00%	0.0595	43.81%
	TPR (pi)	40.51%	0.00%	70.57%	0.66%	97.38%	3.77%
	TR (per)	54.88%	0.00%	2.38%	0.00%	4.52%	24.03%
ARIMA	**NMAE (per)**	0.6136	67.09%	0.2724	12.21%	0.1059	0.00%
	TPR (pi)	72.71%	79.49%	73.08%	4.24%	93.84%	0.00%
	TR (per)	35.79%	34.78%	0.69%	71.01%	5.95%	0.00%
SVR	**NMAE (per)**	0.6949	62.73%	0.2989	3.67%	0.0476	55.05%
	TPR (pi)	73.22%	80.75%	70.11%	0.00%	97.46%	3.86%
	TR (per)	29.58%	46.10%	0.00%	100.00%	3.06%	48.57%

Firstly it is worth pointing out that model performance degrades from time series #3 to time series #1, based on NMAE and TPR. SPM, although a simple model, still showed good results, and is performing better than ARIMA for time series #3. For time series #2, although it shows worst performance in NMAE and TR, is second best in terms of maximizing profit, and ARIMA outperforms it only by several percent. As time series #2 show, SPM is too sensitive to outliers, which can be the case in a real situation, degrading its

performance. Although it aims to maximize profit, unlike other models which are not aware of that, SPM did not prove to have an advantage on that in the selected time series. SPM proved to be completely unfeasible on chaotic real data set, such as time series #1. Overall it is clear that it is the worst performing model.

ARIMA shows the second best performance, and predicts most accurately time series #2 of all other models with NMAE 12-9% better than other candidates. However that does not affect profit level so much. For time series #1, even though it does not try to react to changes to the demand, it shows however the best NMAE. Its TPR is almost as good as the one of SVR.

SVR proved to be the best model for time series #3 and does not lead in terms of NMAE for time series #1. However, it shows an interesting unexpected behaviour for time series #1, repeating some of demand pattern segments surprisingly well, taking in account how chaotic time series was.

The best model to choose would depend on the behavior of a real demand time series. If the demand for some products shows a local trend, ARIMA might be a better choice than SVR. However if the product demand has strong weekly pattern as #3, SVR would be preferred since it has NMAE 55% smaller than the one of ARIMA. If a product demand behaves as in time series #1, (which is highly likely as it is real time series from the same domain as the client's one) it is not clear which of the 2 best performing models (ARIMA or SVR) would be better to apply. In my opinion, SVR has a better potential for such time series as time series #3. Although NMAE of SVR on time series #3 is slightly higher than the one of ARIMA, as Figure 17 shows, it has good chances to perform better is a long term. SVR tries to follow the demand, using information from the previous data points in a more robust way.

5.4 Results

The analysis of the demand prediction models mentioned in this report was based on insufficient data - only 9 days of domain data were available at the time this report was written. This is not sufficient to be sure of the performance of the described algorithms in the actual situation. Therefore they cannot be utilized in the production at this moment. However the research carried out shows potential candidate algorithms to be utilized and

their strengths/weaknesses for different possible cases of the actual data in the client's domain.

To know that the prediction model would perform better than an employee decision, it is important to compare the trash rate TR of the models with the one collected from the client's data. As time series #1 is a demand of a product, similar to the client's 12 h product, the trash rate could be compared. SVR yields trash a rate of 30% and 12 h products in the client's cafe have an average trash rate of 18% - 1.7 times lower. This could give an estimate of how SVR would perform as an advisor system for the production schedule of 12 h products. At this point it is clear that an employee would perform better than SVR. However if more data was available than 35 days, as in the training set for time series #1, the algorithm might perform better.

Time series #3 could describe the behavior of the demand for some 6 h products that are produced in big amounts. If they indeed have a strong weekly pattern, the prediction models would yield a better trash rate. For example, SVR could have 2 times lower trash rate than if the employee would manage the production schedule. In this it will be feasible to use SVR to predict the demand for some of the 12 hours products. Overall this gives optimistic results, and there is a probability that the prediction models could already be deployed in the application and used in the production.

When a sufficient amount data is collected, it will be clear if one of these algorithms could be approbated in the real situation. Let us assume that X most recent weeks of data became available (last X weeks of data until the current day), either collected by the deployed application or from the client's archive. Then if for last K<X weeks one of the algorithms, when running simulation, gives a lower trash amount for some of the products than the actual trash amount during the last K weeks, it is likely that an algorithm gives better results than human estimation. In that case it can be deployed and used for advising the production amount for the selected products. In this case, special attention should be paid to whether the algorithm underestimates the demand. If the algorithm underestimates the demand more often that the employee usually does, the employee will have to produce extra products later, which poses time losses. In this case, the algorithm would need to be withdrawn from the production and redesigned.

These checks could be automated by deploying the algorithm in the application in silent mode, where it only evaluates its possible performance, without advising employees on the production amount. As soon as the algorithm knows it would perform better than the

employee in terms of the trash amount for the last K weeks (2 week should be sufficient), it would give a signal, e.g. by sending an email to the developer. In this case the developer would enable the algorithm to actually advise on the production amount. If the algorithm is enabled and it is clear that during X weeks prior to enabling the algorithm employee underestimated the demand less than the algorithm does, the algorithm would send another signal, so that the developer can disable it and investigate a possible problem in the algorithm.

6 Future Improvements

6.1 Collecting Sufficient Data

As a sufficient amount of actual data will be collected, the research needs to be carried out again - from analyzing the data to selecting the best algorithm. This needs to be done even if one of the algorithms described in the report already outperforms an employee. The major steps and techniques applied and the experiences gained in the project will be applied to new study, which will make it easier and faster to carry out. I assume that 2-3 months of data will be sufficient to carry out improved research.

6.2 Handling Unknown Demand

The demand used as an input for prediction models in this project was assumed to be known for every day. However, as the analysis show, in reality only part of the demand values will be known. When the product is not trashed, there is an unknown demand underestimate, which means that possibly more items of that product could be produced and sold completely. Although the demand underestimate is estimated to be small, taking it into account could improve the profitability of the prediction models. One simple approach to estimate demand underestimate could be assuming the demand underestimate to behave similarly to shrinkage. However as it is more likely that the demand tends to be met by producing tan extra amount of products, the average demand underestimate will be less than the average shrinkage. This is a fair assumption because if employee produces more items of a product than the actual demand, she cannot “undo” produce. Then before feeding the demand time series into prediction model, the unknown demands can be adjusted as follows:

$$D_t'' = P_t + xT \quad (11)$$

In this formula, for the selected product in a selected batch time, D_t'' is the adjusted demand at time t, P_t is produced amount of the product, T is the average shrinkage of the product for all times t1 < t and 0 < x < 1 is a parameter, describing the relation of the average demand underestimate to the average shrinkage for the selected product and batch time. As the demand can possibly have seasonal changes, it could be more feasible to calculate T from recent shrinkage values. After that the adjusted data could be fed into the prediction model as the unadjusted data was fed, as was described in the practical part of this report. However, there are two issues to be resolved.

Firstly, it is hard to fully evaluate the efficiency of this model, as the real demands will be never known when there is no shrinkage. Certainly, increasing x will increase the average shrinkage but can also increase the average sales of a product. As it is possible to know the total cost of product production and product price, the average profit would be a benchmark for this model. Secondly, x needs to be chosen. This cannot be done from evaluating the history data. This could be done after the prediction algorithm is deployed and utilized in the application. Then the algorithm could adjust x based on the average profit change as it changes x. However, rapid changes in x are not desirable, overestimating x too much would greatly increase costs. Therefore choosing the optimal x can take a long time, several weeks or more.

6.3 Improving Forecasting Models

Currently SVR is not aware that more recent data is more important. To make the algorithm more robust, one approach would be running SVR several times with different regression vectors, containing a different amount of lagged values, e.g. 14 days, 7 days and 3 days. Then the predicted value could be calculated as an average of the predictions for each regression vector type.

It will also be worth trying to incorporate more information in the regression vector for SVR. For example, weather information or demand data for other products. The shrinkage amount could also give useful information to the algorithm.

7 Conclusion

The aim of the project was to research possible forecasting models and to see whether it would be feasible to use them in the cafe production optimization application. The domain was researched, and the data provided by the client was analyzed. Three demand time series data were obtained, one from the same domain as the client's domain, and two were generated based on the client's data analysis and information from similar domains, such as fast-food chains. Three predictive models were selected and modelled and their performance was evaluated on the 3 time series.

From the analyzed models, ARIMA and SVR showed the best performance. For time series #1 and #2 trash the rates yielded by forecasting models were lower than for the trash rates yielded by the employees in the client's cafe for some product groups. However, it is not clear at this point if it was feasible to use them inside the cafe production management application to construct production schedules for the café. More data needs to be collected to verify that.

Overall, the research shows that it can be feasible, by using the examined forecasting models to predict product demand. The results were demonstrated to the client and the client sees the potential in the examined prediction algorithms. SVR shows good performance on all of time series and produces interesting results for time series #1. It should be the first model to be deployed in the application in silent mode to verify its performance. The study shows that it will be worth carrying out deeper research as more data is collected.

References

1 JDA Software Group, Inc. JDA Software Official Website [online]. JDA Software Group, Inc.; 2015.
URL: http://www.jda.com/
Accessed 16 November, 2015.

2 Periscope [computer program]. Invatron Systems Corp.; 2015.
URL: http://www.invatron.com/solutions-freshitemmanagement.php
Accessed 16 November, 2015.

3 Hyndman R., Athanasopoulos G. Forecasting: principles and practice [online]. OTexts; 2012.
URL: https://www.otexts.org/fpp
Accessed 16 November, 2015.

4 Liu L., Bhattacharyya S., Sclove S., Chen R., Lattyak W. Data mining on time series: An illustration using fast-food restaurant franchise data [online]. Computational Statistics & Data Analysis 2001; 37(4):455-476.
URL: http://dl.acm.org/citation.cfm?id=568751
Accessed 16 November, 2015.

5 Lisi F., Schiavo R. A. A comparison between neural networks and chaotic models for exchange rate prediction [online]. Computational Statistics & Data Analysis 1999; 30(1):87-102.
URL: http://dl.acm.org/citation.cfm?id=312484
Accessed 16 November, 2015.

6 R [computer program]. Version 3.1.3. The R Foundation; 2015.
URL: https://www.r-project.org/
Accessed 16 November, 2015.

7 MATLAB [computer program]. Version 7.6. The MathWorks, Inc.; 2008.
URL: http://www.mathworks.com/products/new_products/release2008a.html
Accessed 16 November, 2015.

8 Espinoza M., Leuven K. U., Suykens J., Belmans R., Moor B. Electric Load Forecasting [online]. IEEE Control Systems Magazine 2007; 27(5):43-57.
URL: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4303474
Accessed 16 November, 2015.

9 Hong W., Chen Y., Chen P., Yeh Y. Continuous ant colony optimization algorithms in a support vector regression based financial forecasting model [online]. Natural Computation, 2007. Third International Conference on Natural Computation 2007; 1:548-552.
URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4344250
Accessed 16 November, 2015.

10 Sykes A. O. An Introduction to Regression Analysis [online]. The University Of Chicago Law School; 1993.
URL: http://www.law.uchicago.edu/files/files/20.Sykes_.Regression.pdf
Accessed 16 November, 2015.

11 Hsiao C., Su S., Chuang C. A Rough-based Robust Support Vector Regression Network for Function Approximation [online]. Fuzzy Systems (FUZZ), 2011 IEEE International Conference On Fuzzy Systems June 2011; 2814-2818.
URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6007454
Accessed 16 November, 2015.

12 Convenience Store and Fuel News. RedPrairie Comfort Food [online]. CSP Daily News; 12 December, 2007.
URL: http://www.cspnet.com/category-news/foodservice/articles/redprairie-comfort-food
Accessed 16 November, 2015.

13 Shapiro A., Philpott A. A Tutorial on Stochastic Programming [online]. Georgia Institute of Technology, H. Milton Stewart School of Industrial & Systems Engineering; 2007.
URL: http://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/TutorialSP.pdf
Accessed 16 November, 2015.

14 DePaolo C. A., Robinson D. F. Café Data [online]. Journal of Statistics Education; 2011; 19(1).
URL: http://www.amstat.org/publications/jse/v19n1/depaolo.pdf
Accessed 16 November, 2015.

15 Structural Time Series Models R package [computer program]. Version 1.7. The R Foundation: López-de-Lacalle J.; 2015.
URL: http://cran.r-project.org/web/packages/stsm/index.html
Accessed 16 November, 2015.

Appendix 1: Simple Probabilistic Model's main fragments of source code. (Written in R programming language).

```
#Calculates cumulative distribution function
#weekLimit - learning size limit for the algorithm
calcDemandCDF <- function (productStats, demandLimit, predictDemandIndex,
weekLimit) {
    p <- data.frame(probabilitySum=numeric(demandLimit+1),
samplesNum=numeric(demandLimit+1), probability=numeric(demandLimit+1))
    weekAmount = (predictDemandIndex-1)/7
    if(weekAmount<weekLimit)
        stop("not enough learning data")

    i = 0
    for(week in 1:weekLimit)
    {
        # take one week back from predicting date
        i = predictDemandIndex - week*7
        #skip weeks for which there is no data
        if( i <= length(productStats$Day))  {
            minDemand = productStats$Produce[i] - productStats$Trash[i]
            #demand can be 0!
            for(j in 0:demandLimit) {
                if (j >= minDemand) {
                    if(productStats$Trash[i] == 0) break
                    #+1 because there is no 0 index in R,
                    p$probabilitySum[j+1] = p$probabilitySum[j+1] + 1
                }
                p$samplesNum[j+1] = p$samplesNum[j+1] + 1
            }
        }
    }
    for (j in 0:demandLimit) {
     p$probability[j+1] = p$probabilitySum[j+1]/p$samplesNum[j+1]
    }
    return (p)
}

getProfit <- function (amount, demand) {
    sellPrice = 12
    producePrice = 2
    trashPrice = 1 #enviromental factor
    sellProfit = sellPrice - producePrice
```

```
    if(amount <= demand) {
        return (sellProfit*amount)
    } else {
        return (sellProfit*demand - (producePrice + trashPrice)*(amount -
demand))
    }
}

#Calculates optimal produce amount using given CDF
calcOptimalProduceAmount <- function (demandCDF, demandLimit,
predictDemandIndex) {
    maxProfit = 0
    maxProfitAmount = 0
    trialCostLimit = demandLimit
    profits <- numeric(trialCostLimit+1)
    for (currentAmount in 0:trialCostLimit) {
        totalExpectedProfit = 0
        #try all possible demand values
        for(demand in 0:(length(demandCDF$probability)-1)) {
            demandProbability = (demandCDF$probability[demand+1])
            if (demand != 0)  demandProbability = demandProbability -
demandCDF$probability[demand]
            totalExpectedProfit = totalExpectedProfit +
getProfit(currentAmount, demand) * demandProbability
        }
        profits[currentAmount+1] = totalExpectedProfit
        if(totalExpectedProfit > maxProfit) {
            maxProfit = totalExpectedProfit
            maxProfitAmount = currentAmount
        }
    }
    pp <- list(maxProfitAmount = maxProfitAmount, maxProfit = maxProfit,
profits = profits)
    return (pp)
}

#Simple probabilistic model's core function.Combines CDF calculation and
optimal demand search
simpleAlgorithm <- function (inputData, predictDemandIndex, weekLimit) {
    p <- calcDemandCDF(inputData, demandLimit, predictDemandIndex, weekLimit)
    res <- calcOptimalProduceAmount(p, demandLimit, predictDemandIndex)
    return (res$maxProfitAmount)
}
```

```
# A function used to find optimal week limit for training data set for simple
probabilistic model
findOptimumWeekLimit <- function (data) {
    # …… <- Omitted code

    demands = data$Produce - data$Trash
    trainingSize = nrow(data) - verificationSize
    testingData = demands[(length(demands) - verificationSize +
1):length(demands)]
    bestLimit = 0
    maxProfit = 0
    for(weekLimit in 1:(trainingSize/7)) {
        predictions = generateSimpleAlgorithmPredictionData(verificationSize,
data , weekLimit)
        profit = evaluateAlgorithm(predictions, testingData, TRUE)
        if (profit > maxProfit) {
            maxProfit = profit
            bestLimit = weekLimit
        }
        printf("For weeklimit=%d profit = %f", weekLimit, profit);
    }
    return (bestLimit);
}
```