Forecasting Campaign Sales Using Artificial Neural Networks

Joel Engström

June 22, 2016

Abstract

This thesis will investigate the characteristics and possibilities of implementing a forecasting model for campaign sales in Coop stores. The implementation will be done by the use of multiple linear regression and artificial neural networks. Advantages and drawbacks of these models will be investigated and discussed.

The campaign characteristics will be evaluated using linear methods, and several prediction models, of varying complexity, will be proposed and tested on real sales data.

It is concluded that while the complex models perform better, the tradeoff between loss in accuracy and increased simplicity, also making the models more generic, may justify the use of the simpler models.

Acknowledgements

First of all I would like to thank my supervisor Andreas Andersson for valuable input during the project, as well as giving me the opportunity to work with Coop's logistics division in an inspiring environment.

I would also like to thank Anders Heyden, my supervisor at LTH, for pointing me in the machine learning direction and for his valuable feedback on both theory and the writing of this report.

Contents

1 Background **5**
1.1 Forecasting sales as it is done today 5
1.2 Past research on the subject 5
1.3 Purpose of this study 6

2 Theory **7**
2.1 Linear regression and least squares 7
2.2 R^2 statistics 7
2.3 Artificial Neural Networks 9
2.4 Backpropagation of the error 10
2.5 Gauss-Newton algorithm 11
2.6 Levenberg-Marquardt algorithm 12
2.7 Noise reduction in network output 13
2.8 Autoregressive moving average and autocorrelation functions . . 13

3 Data **14**
3.1 Sold volume or total revenue 14
3.2 Average consumer price 14
3.3 Dummy coded variables 15
3.4 Media and role codes 15
3.5 Yearly seasonality 15
3.6 SMHI data 16
3.7 Bundling of explanatory variates 16
3.8 Errors in the data 16

4 Implementation **17**
4.1 Multilinear regression on perishable items 17
4.2 Multilinear regression on dairy products 19

5 Results **21**
5.1 Neural Network and Linear Regression on dairy products 22
5.2 Integrating Neural Network output with time series data 32

6 Discussion **33**
6.1 Outliers 33
6.2 Neural Network or Linear regression 36
6.3 Model complexity and Network design 36
6.4 Choice of searching algorithm 37
6.5 Model performance 37

7 Conclusions **37**

1 Background

1.1 Forecasting sales as it is done today

Today most large scale retailers have implemented computer assisted ordering to forecast future sales for accurate replenishment. These systems are usually based on time series of past order data and does account for interventions such as holidays. Coop's current system however does not account for the special event of store campaigns.

Grocery retailers see a large increase in sales during their campaigns and this along with the consumer demand of fresh products makes accurate predictions of campaign sales important. Predictions that are difficult to make due to the unpredictable behavior of human consumption.

When forecasting sales for specific campaigns Coop has a campaign team working with this specifically and they make their predictions based on past campaigns sold quantities and their own experience. These predictions are on terminal level with a goal accuracy of 20% absolute deviation from the realized sales, and are made preliminary 12 weeks ahead of the campaign. Most store managers then make their own preorders on how much they want delivered to their store three weeks ahead of the campaign. The campaign team can then adjust their predictions in accordance with the collective preorders. The sales prediction and the preorders make the foundation for the quantities of the item volume delivered to the terminal. Items are then distributed to the stores the week before the campaign week based on the store managers' final orders. These orders may deviate from the preorders and in the case when the store demand is higher then the volumes stored in the terminal the items are being distributed on a first come first serve basis.

1.2 Past research on the subject

In 1995 and 1997 two papers were published on grocery campaign forecasting using Neural Networks, see [9] and [10]. The authors have taken the approach of modelling the time series, with weekly increments, of campaign items in German supermarkets. This is done by the use of Neural Networks with inputs such as: Time series of weekly sales, advertisements and holidays. The predictions are made on future sold volumes and based on each item and supermarkets individual time series. The authors discuss that their models are not generic in the sense that they can account for any item in the supermarket and make accurate predictions. But for the items they can make adequate predictions for, they compare their predictions with what they call a naive prediction, that uses the last known value in the time series, and a statistical prediction that uses a weighted average, which was used in the supermarkets at the time. They show that in most cases their Neural Network predictions surpass the naive and statistical in accuracy. The authors continue to conclude that their model needs to be made more generic in order to be implemented in the forecasting system and that future work will be directed to test significance of input data to reduce the number of input nodes and thus reduce net complexity.

In 2014 another paper was published on prediction of sales revenue for grocery retailers in Turkey based on artificial neural networks, see [7]. This study aims at predicting the future quarterly gross profit as a whole for the company and does this based on past gross profit for the company and its competitors as well as marketing costs. The authors acknowledge that there could be other factors that effects the revenue but that they are limited to what is available to them. For the companies they are studying their predictions are very close to the actual gross profit, only using three relevant input factors, and the authors conclude that Artificial Neural Networks seems like a suitable tool for revenue forecasting.

1.3 Purpose of this study

The high level idea is to create a mathematical model that adequately describes the total sales for each item in a Coop store during a campaign week. More specifically the project can be broken down into two parts.

Foremost, since present forecasting for campaign sales are mainly done based on past sales, other significant factors that may be relevant to the campaign sales will be investigated. The characteristics of the campaigns, such as advertising and store exposure of the wares, will be looked upon. Geographical differences in the sales as well as correlation between sales and weather data will also be investigated. This will be done by collecting the relevant data and then fit a multiple regression model using the method of ordinary least squares. Identification of significant factors will be done by the use of statistics such as R^2 and parameter confidence intervals.

Secondly, the results of the statistical analysis will be used to implement an Artificial Neural Network. Relevant factors from the regression analysis will be used as inputs and the data will be passed through a feedforward Neural Network with the campaign sales as target data. The advantage of using Neural Networks is that they can capture nonlinear relationships between input and output data. Hopefully the network will adequately describe these in the campaign data and perform better than the multiple linear regression model, only capable of describing linear relationships. Nonlinear relationships in the data might for example be that the increase in sales generated by TV ads and magazine ads separately might differ from the increased sales generated when they are used in combination. Or that there is probably a nonlinear dependence between the campaign price and sales, intuitively there should be a jump in sales when the price hits certain levels, e.g 10 and 20.

The methodology for implementation of the networks will be similar to what is outlined in [10], but in this study, campaigns will be collected in the same data set and stores and items will be separated by the means of dummy variables, further discussed in Section 3.3. So that the network can train on all campaigns present in the set simultaneously and not on each items time series individually. With this approach every prediction will be based on a larger amount of data. In this study, a top down approach will be taken as to what input factors are most relevant to the output and the trade off between reduced complexity and loss of accuracy will be discussed. This is a crucial topic since the models need

to be rather generic in order to be able to be implemented in the company's forecasting system.

Furthermore, the possibility and usefulness of extending the single layer network to a multiple layer using deep learning techniques will be investigated.

2 Theory

In this section, all the necessary theoretical background for the models will be presented. Starting with multiple linear regression and the method of ordinary least squares and then continuing to artificial neural networks.

2.1 Linear regression and least squares

Given a data set $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^n$ in a linear regression model, each realization y_i of the stochastic variable y is assumed to be linearly dependent on the stochastic variables $x_1 \dots x_p$. Let y be the response variable and $x_1 \dots x_p$ the explanatory variables. Adding an error term r_i and the linear dependencies $\beta_1 \dots \beta_p$ this relationship is written as:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{1,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} \Leftrightarrow Y = X\boldsymbol{\beta} + \boldsymbol{r} \ ,$$

now to derive an expression for the least squares fit of $\boldsymbol{\beta}$ we want to minimize the squared error term $\boldsymbol{r}^T\boldsymbol{r}$, as

$$\begin{aligned} \min \boldsymbol{r}^T\boldsymbol{r} &= \min (Y - X\boldsymbol{\beta})^T(Y - X\boldsymbol{\beta}) \\ &\Leftrightarrow \frac{\partial (Y - X\boldsymbol{\beta})^T(Y - X\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0 \\ &\Leftrightarrow -2X^TY + 2X^TX\boldsymbol{\beta} = 0 \\ &\Leftrightarrow (X^TX)^{-1}X^TY = \boldsymbol{\beta} \ . \end{aligned} \tag{1}$$

Geometrically this can be interpreted as $X\boldsymbol{\beta}$ being the orthogonal projection of y onto the space spanned by X, as seen in Figure 1, the projection is here denoted $\hat{y} = X\boldsymbol{\beta}$. A consequence of this is that $\boldsymbol{r}$ is perpendicular to any element in X, i.e., the inner product $X^T\boldsymbol{r} = 0$.
Here it is assumed that the residuals are zero-mean, $E[\boldsymbol{r}] = 0$, and the expected value of the response variable is thus $E[Y] = X\boldsymbol{\beta}$. The important interpretation here is that each β_j, $j = 1, \dots, p$ is the expected change in y_i given a unit increase in x_{ij} when all other x_{ij}'s are held fixed.

2.2 R^2 statistics

Let $\mu = \frac{1}{n}\sum_{i=1}^n y_i$ be the mean of y and let $\mathbf{1} = [1, \dots, 1]^T$ be the all one vector of dimension n. Then denote

$$SS_{tot} = (Y - \mu\mathbf{1})^T(Y - \mu\mathbf{1}) \ ,$$

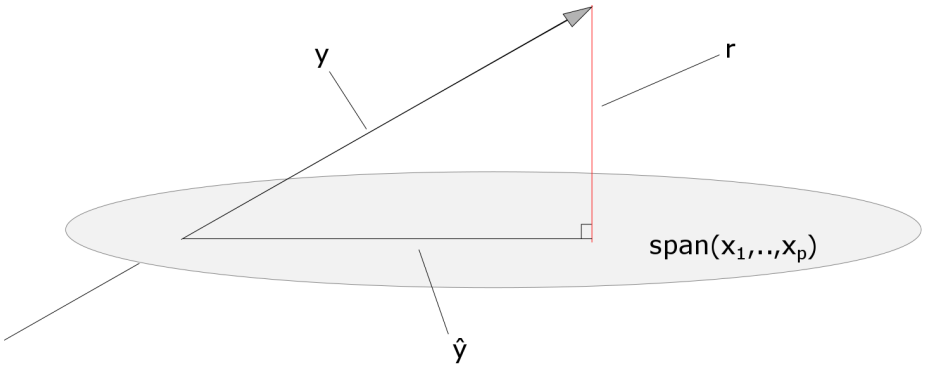

Figure 1: Orthogonal projection of y onto $span(X)$. The residuals $\boldsymbol{r}$ is perpendicular to the space spanned by the explanatory variates X

which is the total sum of squares of the deviance of the y_i from their mean. Furthermore the sum of squares of the residuals is defined as

$$SS_{res} = \boldsymbol{r}^T\boldsymbol{r} = (Y - X\boldsymbol{\beta})^T(Y - X\boldsymbol{\beta}) \ ,$$

this is the sum of the squared error between the actual value and the predicted value from the regression model. Also defining

$$SS_{reg} = (X\boldsymbol{\beta} - \mu\mathbf{1})^T(X\boldsymbol{\beta} - \mu\mathbf{1}) \ ,$$

which is the sum of the squared error between the prediction of y and its mean, this is also called the explained variance. Now the R^2-statistics can be defined as

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \ ,$$

and be rewritten as

$$\begin{aligned} R^2 = 1 - \frac{SS_{res}}{SS_{tot}} &= \frac{Y^TY - 2\mu\mathbf{1}^TY + (\mu\mathbf{1})^T(\mu\mathbf{1}) - Y^TY + 2(X\boldsymbol{\beta})^TY - (X\boldsymbol{\beta})^T(X\boldsymbol{\beta})}{SS_{tot}} \\ &= \frac{-2\mu\mathbf{1}^T(X\boldsymbol{\beta} + \boldsymbol{r}) + (\mu\mathbf{1})^T(\mu\mathbf{1}) + 2(X\boldsymbol{\beta})^T(X\boldsymbol{\beta} + \boldsymbol{r}) - (X\boldsymbol{\beta})^T(X\boldsymbol{\beta})}{SS_{tot}} \\ &= \frac{-2\mu\mathbf{1}^T(X\boldsymbol{\beta}) + (\mu\mathbf{1})^T(\mu\mathbf{1}) + (X\boldsymbol{\beta})^T(X\boldsymbol{\beta})}{SS_{tot}} \\ &= \frac{(X\boldsymbol{\beta} - \mu\mathbf{1})^T(X\boldsymbol{\beta} - \mu\mathbf{1})}{SS_{tot}} = \frac{SS_{reg}}{SS_{tot}} \ . \end{aligned} \tag{2}$$

In the third to last step it has been used that the inner product $X^T\boldsymbol{r} = 0$ and that $\mathbf{1}^T\boldsymbol{r} = n\bar{\boldsymbol{r}} = 0n = 0$.
From (2) it can be seen that R^2 is the ratio between the variance explained by the model and the total variance of the response variable. Hence, R^2 can be interpreted as how much of the variance in the response variate that is explained by the explanatory variates, $R^2 = 1$ being a perfect linear relationship.

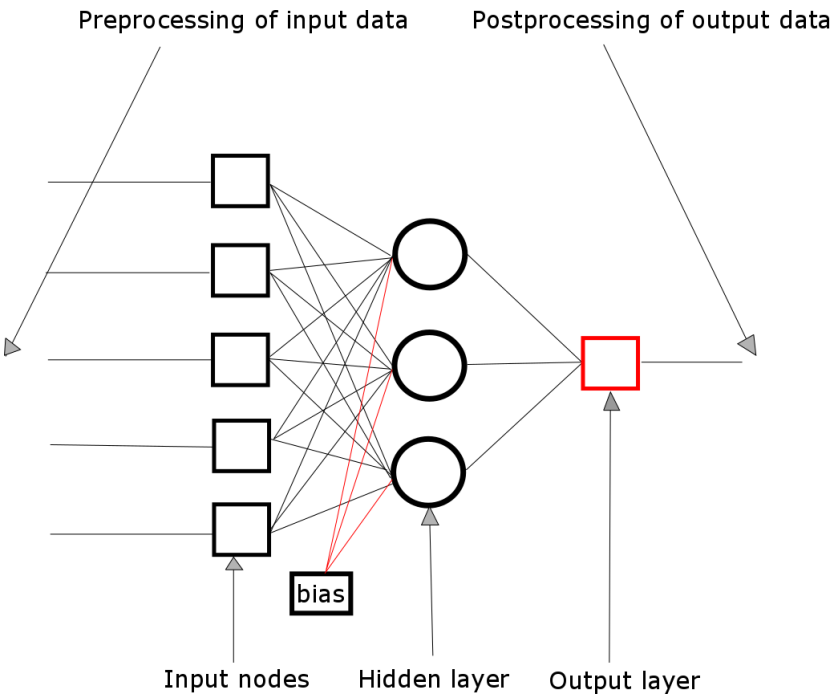

Figure 2: Illustrative figure of a single layer feed forward network

2.3 Artificial Neural Networks

Neural networks is a branch of machine learning that aims to imitate how the brain uses synapses and neurons to process information. The basic components are the input nodes, the hidden layers and the output layer, all of which are connected by weights. In this paper, focus will be on feed-forward networks. These will be represented as oulined in [3], where the neurons are grouped as one input layer, n hidden layers and one output layer. In a feed forward network the neurons only have directed contact to the layer in front of them towards the output, information passes from left to right in Figure 2. In this study the matlab toolbox for Neural Networks [4], will be used for the implementation of Artificial Neural Networks.

In a feed forward network, each node in a hidden layer sums the weighted input, and a bias, of the nodes behind it, combining them to a single value which is passed through an activation function, see [8]. The value of the activation function will then be the input to the next successive node. Let $x_1, \ldots, x_d$, be the input to the j'th node with activation function f_j. Furthermore, let $w_1, \ldots, w_d$ be the input weights and θ be the input bias. The output of that node will then be

$$f_j\Big(\sum_{i=1}^{d} w_i x_i - \theta\Big) \ .$$

For the scope of this report, the sigmoidal function will be used as the activation

function so that

$$x_j = \frac{1}{1 + c\exp(\sum_{i=1}^{d} w_i x_i - \theta)} \ ,$$

where c is a constant that can be chosen arbitrarily. The output of the j'th node can then be viewed as the input to a node in the successive layer, which is then to be weighted and summed up along with the other inputs to the same node. As explained further in [3], both the sigmoidal function and the bias are used to increase computational efficiency.

In this way the information from the input layer is fed forward through the network until it reaches the output layer where it can be compared to the target data. For this study, the inputs will be the governing campaign factors discussed in Section 1.2 and the target will be the realized sales of the campaign.

2.4 Backpropagation of the error

Consider a feed forward network consisting of N weights $\boldsymbol{w}$ and M output nodes. Given a training set, $\{\boldsymbol{x}_1, \boldsymbol{t}_1, \ldots, \boldsymbol{x}_P, \boldsymbol{t}_P\}$ of input vectors $\boldsymbol{x}_p$ and target vectors $\boldsymbol{t}_p$, each pair of input and target vector is called a pattern. For each $\boldsymbol{x}_p$ the neural network will produce an output $\boldsymbol{o}_p$ that in general will differ from the target $\boldsymbol{t}_p$. The idea is to use an algorithm that trains the network to make the output-target difference as small as possible. The complete algorithm is outlined, in detail in [11], but the overall goal is to minimize the network error, which is defined as

$$E(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{2} \sum_{p=1}^{P} \sum_{m=1}^{M} e_{p,m}^2 \ , \tag{3}$$

where $e_{p,m} = t_{p,m} - o_{p,m}$ is the training error at output m. In order to achieve this, the backpropagation algorithm can be used to adjust the weights of the network so the output error is at a local minimum. Defining the gradient as

$$\boldsymbol{g} = \nabla_{\boldsymbol{w}} E = \left[\frac{\partial E}{\partial w_1} \quad \frac{\partial E}{\partial w_2} \quad \cdots \quad \frac{\partial E}{\partial w_N} \right]^T \ , \tag{4}$$

that is a vector of partial derivatives pointing in the steepest ascending direction of the objective function. Since $\boldsymbol{o}_j$ is made up of functions that are nested backwards through the net, each partial derivative of ∇E is made up of derivatives of successive activation functions and weights, and so the gradient propagates backwards through the net.

The idea of the error backpropagation algorithm(EBP) is to iteratively follow in the direction of the negative gradient, the steepest descending direction, until we arrive at the stopping condition

$$\boldsymbol{g} = \boldsymbol{0} \ , \tag{5}$$

which is the first order condition for a local minimum. To do this, at each step the weights w_i is updated according to its corresponding partial derivative

$$\Delta w_i = -\gamma \frac{\partial E}{\partial w_i} \ . \tag{6}$$

Thus the updating rule for the gradient descent algorithm at each iteration k is

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \gamma \boldsymbol{g}_k \ , \tag{7}$$

so that at the next step the objective function is evaluated a step of length $\gamma||\nabla_{\boldsymbol{w}} E||_2$ in the direction of the negative gradient. For a thorough proof of correctness of this algorithm, please see [8] p.155.

2.5 Gauss-Newton algorithm

The gradient descent method can be computationally slow, and a faster alternative is the Gauss-Newton(GN) method which uses a faster way of error convergence, please see [11].

Assuming all gradient components are linearly independent functions of weights

$$\boldsymbol{g} = \boldsymbol{F}(\boldsymbol{w}) \ , \tag{8}$$

Newton's method aims at approximating (5) by starting at a starting point of weights $\boldsymbol{w}_0$ and use the first order Taylor approximation

$$\boldsymbol{g} = \boldsymbol{g}_0 + \nabla_{\boldsymbol{w}} \boldsymbol{g} \Delta \boldsymbol{w} = \boldsymbol{g}_0 + \nabla^2_{\boldsymbol{w}} E \Delta \boldsymbol{w} = \boldsymbol{g}_0 + \boldsymbol{H} \Delta \boldsymbol{w} \ , \tag{9}$$

where $\boldsymbol{H}$ is called the Hessian matrix of the error function E defined as

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 E}{\partial^2 w_1} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_1 \partial w_2} & \frac{\partial^2 E}{\partial^2 w_2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_1 \partial w_N} & \frac{\partial^2 E}{\partial w_2 \partial w_N} & \cdots & \frac{\partial^2 E}{\partial^2 w_N} \end{bmatrix} \ . \tag{10}$$

Since the aim is to get the gradient to be zero, setting (9) equal to zero yields

$$-\boldsymbol{g}_0 = \boldsymbol{H} \Delta \boldsymbol{w} \Leftrightarrow \Delta \boldsymbol{w} = -\boldsymbol{H}^{-1} \boldsymbol{g}_0 \ , \tag{11}$$

and to converge to $\boldsymbol{g} = \boldsymbol{0}$, the weights are then updated as $\boldsymbol{w}_1 = \boldsymbol{w}_0 + \Delta \boldsymbol{w}$. This is done at every step k and thus the updating rule for Newton's method is,

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \boldsymbol{H}^{-1} \boldsymbol{g}_k \ . \tag{12}$$

Using this method, the Hessian needs to be computed at every step and since it contains every second derivative of the total error function, the computation can get very cumbersome. This is taken care of in the Gauss-Newton algorithm where the Hessian is replaced by the Jacobian matrix. Given the error vector

$$\boldsymbol{e} = \begin{bmatrix} e_{1,1} & e_{1,2} & \cdots & e_{1,M} & e_{2,1} & \cdots & e_{2,M} & \cdots & e_{P,M} \end{bmatrix}^T \ , \tag{13}$$

(3) can be rewritten as

$$\boldsymbol{E} = \frac{1}{2} ||\boldsymbol{e}||_2^2 \ . \tag{14}$$

Now defining the Jacobian $\boldsymbol{J}$, which is a $P \times M$ matrix of the first partial derivatives of the error terms, defined as

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial e_{1,1}}{\partial w_1} & \frac{\partial e_{1,1}}{\partial w_2} & \dots & \frac{\partial e_{1,1}}{\partial w_N} \\ \frac{\partial e_{1,2}}{\partial w_1} & \frac{\partial e_{1,2}}{\partial w_2} & \dots & \frac{\partial e_{1,2}}{\partial w_N} \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{1,M}}{\partial w_1} & \frac{\partial e_{1,M}}{\partial w_2} & \dots & \frac{\partial e_{1,M}}{\partial w_N} \\ \frac{\partial e_{2,1}}{\partial w_1} & \frac{\partial e_{2,1}}{\partial w_2} & \dots & \frac{\partial e_{2,1}}{\partial w_N} \\ \\ \dots & \dots & \dots & \dots \\ \frac{\partial e_{P,M}}{\partial w_1} & \frac{\partial e_{P,M}}{\partial w_2} & \dots & \frac{\partial e_{P,M}}{\partial w_N} \end{bmatrix} . \tag{15}$$

The gradient can then be written as

$$\boldsymbol{g} = \boldsymbol{J}\boldsymbol{e} \ . \tag{16}$$

Inserting (3) into (10), the elements of the Hessian can be calculated as

$$h_{i,j} = \frac{\partial^2(\sum_{p=1}^{P}\sum_{m=1}^{M} e_{p,m}^2)}{\partial w_i \partial w_j} = \sum_{p=1}^{P}\sum_{m=1}^{M} \frac{(\partial e_{p,m})^2}{\partial w_i \partial w_j} + \sum_{p=1}^{P}\sum_{m=1}^{M} \frac{\partial^2 e_{p,m}}{\partial w_i \partial w_j} e_{p,m} \ . \tag{17}$$

Under the assumption of small errors, the term

$$\sum_{p=1}^{P}\sum_{m=1}^{M} \frac{\partial^2 e_{p,m}}{\partial w_i \partial w_j} e_{p,m} \approx 0 \ , \tag{18}$$

reducing the elements in the Hessian from (17) to

$$h_{i,j} = \sum_{p=1}^{P}\sum_{m=1}^{M} \frac{(\partial e_{p,m})^2}{\partial w_i \partial w_j} \ , \tag{19}$$

and the Hessian in (10) can thus be approximated by

$$\boldsymbol{H} = \boldsymbol{J}_k^T \boldsymbol{J}_k \ . \tag{20}$$

The updating rule (12) will then become

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - (\boldsymbol{J}_k^T \boldsymbol{J}_k)^{-1} \boldsymbol{J}_k \boldsymbol{e}_k \ . \tag{21}$$

The assumptions made about the error terms and Newton's method are further discussed in [11] and [6] p 246.

2.6 Levenberg-Marquardt algorithm

The Gauss-Newton algorithm is computationally fast but not necessarily stable since it assumes that $\boldsymbol{J}_k^T \boldsymbol{J}_k$ is invertible. This is taken care of by the Levenberg-Marquardt(LM) algorithm. The LM algorithm is a combination of the error

backpropagation algorithm, which uses the gradient descent, and the computationally faster Gauss-Newton algorithm. It introduces the always positive combination coefficient μ and uses an updating rule

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - (\boldsymbol{J}_k^T \boldsymbol{J}_k + \mu \boldsymbol{I})^{-1} \boldsymbol{J}_k \boldsymbol{e}_k \ , \tag{22}$$

where $\boldsymbol{I}$ is the identity matrix of appropriate dimensions, which is of full rank and ensures that $\boldsymbol{J}_k^T \boldsymbol{J}_k + \mu \boldsymbol{I}$ is invertible. It can be observed that when μ is small and approaches 0, the algorithm approaches the GN algorithm, and when μ becomes relatively large, it approaches the EBP algorithm that uses the gradient descent. Basically, it uses the EBP algorithm for complex curvature in the error function and then switches over to the GN algortihm to speed up convergence for easy curvature. For a more thorough walkthrough of the LM algorithm, see [11].

2.7 Noise reduction in network output

To avoid noisy network output, preprocessing of the input data and postprocessing of the output to bring it back to scale for evaluation will be used. In the preprocessing step, the input is normalized to be between $[1, -1]$, which can be done by letting x be an arbitrary vector with x_{min} and x_{max} being the minimum and the maximum value of x, respectively. Then,

$$x_{pre} = 2\frac{(x - x_{min})}{x_{max} - x_{min}} - 1$$

will be normalized between $[1, -1]$. By some simple algebra the corresponding postprocessing function can be found as

$$x_{post} = (x_{pre} + 1)\frac{x_{max} - x_{min}}{2} + x_{min} \ ,$$

which brings x_{post} back in scale with x. This is used by default in the matlab toolbox and for further documentation see [5] p 3-11.

Furthermore, since the matlab function for Neural Networks uses randomized starting weights, the end results will vary for each network. Thus, a method of computing several networks and then use their average output will be used in an attempt to further reduce output noise. This technique is further explained in [5] p 9-34.

2.8 Autoregressive moving average and autocorrelation functions

The stochastic process y_t is called an autoregressive moving average process (ARMA) if

$$A(z)y_t = C(z)e_t \ , \tag{23}$$

where e_t is a zero mean white noise process and $A(z)$, $C(z)$ are polynomials of order p and q respectively, defined as

$$\begin{aligned} A(z) &= 1 + a_1 z^{-1} + \ldots + a_p z^{-p} \\ C(z) &= 1 + c_1 z^{-1} + \ldots + c_q z^{-q} \ , \end{aligned} \tag{24}$$

with z^{-1} being the unit delay operator such that $z^{-1}y_t = y_{t-1}$. Equation (23) can be rewritten as

$$\frac{A(z)}{C(z)}y_t = e_t \ , \tag{25}$$

so, if a stationary process y_t is filtered through an estimation of $A(z)$ and $C(z)$ and the end product is zero-mean white noise, the polynomials are an adequate representation of the process and this can be used to predict future realizations of y_t.

For a real valued process y_t, its autocovariance function is defined as

$$r_y(k) = Cov\{y_t, y_{t-k}\} \ , \tag{26}$$

where $k = 0, \ldots, N$. Furthermore, the autocorrelation function (ACF) is defined as

$$\rho_y(k) = \frac{r_y(k)}{r_y(0)} \ . \tag{27}$$

The ACF and the partial autocorrelation function (PACF), see [2] p 99, are common ways to identify the order of an ARMA(p,q) process. When the ACF and the PACF of the model residual of a process filtered through an ARMA(p,q)-model has only significant correlation at lag 0, the underlying process can be assumed to be well modelled as an ARMA process.

3 Data

The data used in this project is provided by Coop and their data warehouse, as well as weather data downloaded from the Swedish Meteorological and Hydrological Institute (SMHI). As campaigns are the main focus of this study, several past campaigns of mainly perishable foods are chosen. In this chapter, the data associated with a campaign for any specific store item will be discussed.

3.1 Sold volume or total revenue

As a response variate for the models, sold volume or total revenue in SEK will be used depending on which is most suitable for the given item and data. Since there is believed to be a monthly seasonality in the sales, for a given campaign the volume or revenue for the campaign week will be used as a response variable and the preceding 4 weeks will be used as explanatory variables.

3.2 Average consumer price

Let t be the week of the campaign and p_t be the average consumer price during that week, then the relative price decrease is defined as

$$p_{rel} = \frac{p_t}{p_{t-1}} \ ,$$

The average consumer price during the campaign week and the preceding as well as the relative price decrease will be used as explanatory variables.

Camapigns	Item 2	Item 3	Store 2	Store 3	Region 2	Region 3
Camapign 1	1	0	1	0	0	1
Camapign 2	0	0	0	1	0	1

Table 1: Explanatory table for the dummy coded variables that will be used. Campaign 1 for item 2 in store 2 which lies in region 3 and campaign 2 for item 1 in store 3 which also lies in region 3, would have dummy coded variables as in this table.

3.3 Dummy coded variables

Several variables that might be relevant for the response variable lacks a clear numerical representation. Such as item code, store code and region code. To bypass this these variables are dummy coded. Dummy coding is just dividing each item, store and region into its own variable and for a campaign the dummy coded variables are either a one or a zero depending on if the campaign is for the item, in the store or in the region of the corresponding dummy variable. This is easiest explained by a table, see Table 1.

3.4 Media and role codes

Coop specific media and role codes will also be used. Media codes are representing the kinds of advertising that was used during the campaign. This might be specific placements in the product letter e.g front page or back page, or if the campaign was advertised on TV or in a newspaper. Role codes explains how the item is presented in the store during the campaign, for example if there are extra large signs drawing the attention of the customers or if a side section has been arranged for the item. The most important media and role codes are included in the data and these are also dummy coded.

It is worth noting that which media and role codes used for a given campaign are chosen by a marketing team from a set of guidelines. For example front page items in the product letter are usually heavily discounted by at least 30%. Items used in TV and newspaper ads however are rarely discounted more then 30% and unhealthy products are seldom used as the TV ads are a good way for the company to market itself as a healthy brand. As there are underlying preferences governing the media and role codes this might influence the statistical analysis, at least when it comes to interpreting the β-coefficients which might be corrupted by how the media and role codes have been selected.

3.5 Yearly seasonality

It seems obvious that many items should exhibit a seasonal variation, and Coop is storing a seasonal index for all their items which is normailized around one. This index for the same week as the given campaign week will be used as an input, but normalized around zero in order to get a positive coefficient. However, since the past weekly sales are also used, the seasonal variation might already be captured sufficiently, making the index redundant.

Holiday weeks	Assigned number
Christmas	3
New years eve	2
Easter	2
Midsummer	2
Crayfish season	1
Halloween	1

Table 2: Arbitrarily chosen numbers to reflect holidays impact on sales.

Another, arbitrarily chosen, index that aims at capturing the Swedish holidays influence on the sales is included. For each holiday a number is assigned with the holiday weeks as in Table 2

3.6 SMHI data

Since the sales of some items might be weather dependent it is of interest to study the weather changes impact on the sales. The average daily temperature, the maximal daily temperature, total daily downpour (mm) and total daily sunshine are downloaded from SMHI's open database and converted to weekly measurements. For a given campaign weather data sampled from a weather station within the corresponding Coop region will be used. As the different stores used have different relative locations to the weather stations, some within region variations might not be captured by this approach.

3.7 Bundling of explanatory variates

To analyze how the different input variables influences the output they are bundled into the above categories. The bundles of explanatory variates will then systematically be removed and a new model will be fit to the data. In this way the individual bundles contribution to the R^2 and model accuracy can be compared.

3.8 Errors in the data

There are bound to be errors in the data and in this section some of the most obvious ones will be addressed.

The data representing the sales, discussed in Section 3.1, may in some cases be subject to a major error when the items in stock does not meet the consumer demand. That is, when customers wants to buy a campaigned item that is out of stock. The data will then only reflect the number of items sold which in this case is not an accurate representation of the consumer demand, which is higher than the store supply. This scenario is not that unlikely when perishable goods are being considered as the store managers then are careful not to order excessively so they do not have to discard out of date items. This can also happen as a consequence of low terminal volumes as discussed in Section 1 and when there are supplier shortages which can happen with fresh produce such as meat and eggs.

Considering the media and role codes discussed in Section 3.4. These are labeled by the campaign planners and may not necessarily be carried out in the stores. Hence, the sales may be lower then expected due to erroneous displays in the stores. For example, an item that is planned by the campaign planners to be displayed with large signage on a gable during a campaign is neglected due to lack of time or space in the store. This campaign will still be marked as a major store display in the database but the sales figures will not reflect an increase in item exposure.

These kind of deviances in the data are likely to be there but hard to follow up on which may make it important to detect and discard outliers in order to make accurate predictions.

4 Implementation

In this section multilinear regression and neural network models are fitted to several datasets and the outcomes are analyzed to evaluate the model choice.

4.1 Multilinear regression on perishable items

A data set containing 8424 campaigns of haphazardly chosen perishable items is selected from the Coop database. The campaigns are on 27 items over 22 medium sized stores in the Stockholm region. Each campaign is characterized by the variables discussed in Section 3, adding up to a total of 88 explanatory variables to explain the response variate that is the campaign weeks sold volume of the given item. This data is fit to a regression model using matlabs built in function *regress*. A column of ones is added to the data to also fit an intercept. The campaigns vary a lot in total sales and, in Figure 3, the predicted outputs generated by the regression model on the full set as well as the output from a regression model on a set where outliers has been taken out of the set, is plotted.

The outliers have been chosen based on which campaigns gives a residual confidence interval, generated by the matlab function regress, that does not contain 0. In this case 312 of the 8424 campaigns are flagged as possible outliers. The model on the full data generates a R^2 value of 0.6793 and without outliers the corresponding value is 0.8248 making the model without outliers significantly better in that regard.

The explanatory variables are now bundled together as explained in section 3.7 and new models are fit to the reduced data generating R^2 statistics as in Table 3.

It can be observed that the explanatory variables bundled as revenue, wares, prices and media/role codes seem to have the largest significance based on the R^2 statistics and is likely to be relevant to the model. The seasonal variation seem to contribute nothing at all, which is probably due to the fact that the seasonal variation is rather slow and then likely to be explained by the magnitude of the revenue of the weeks preceding the campaign. Here bundled as revenue. Noteworthy is that the bundle labeled stores, containing 21 dummy

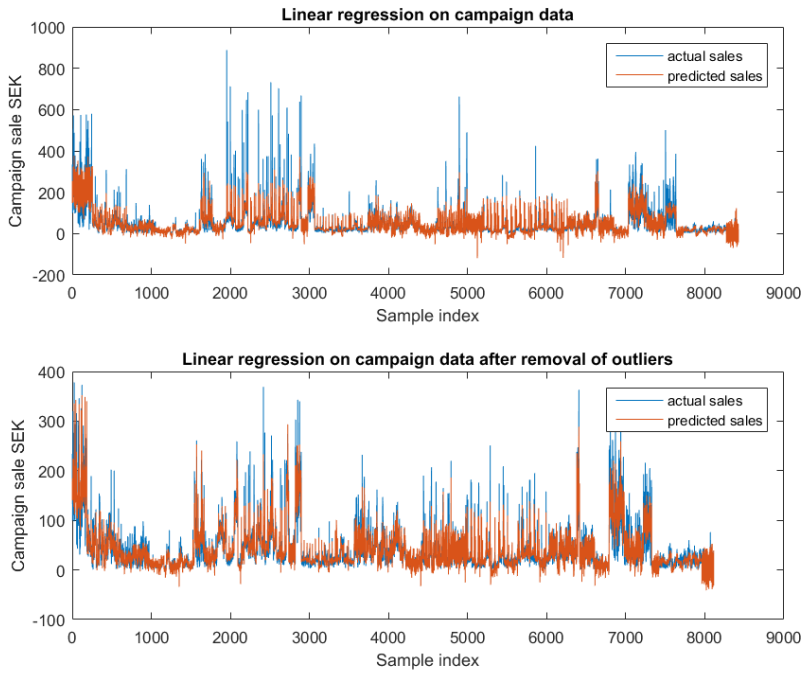

Figure 3: Output of models fit with the whole data set and the data set without outliers. Output is the sold number of an item during the campaign week in a store

Model	R^2	R^2-reduction	S.v reduction	Difference per variable
Full model	0.8254			
Without M/R codes	0.7719	0.0535	27	0.00198
Without seasonal variation	0.8254	0	1	0
Without weather	0.8252	0.0002	4	0.00004
Without items	0.7288	0.0966	26	0.00372
Without holidays	0.8247	0.00067	1	0.00067
Without prices	0.8145	0.0109	3	0.003633
Without stores	0.8138	0.0116	21	0.000552
Without revenue	0.6976	0.1278	4	0.03195

Table 3: Table of R^2 statistics associated with each bundle of explanatory variables. On the data set without outliers.

coded variables, only contribute by a R^2 increase of 0.01. Due to the fact that these models could be exposed to large data sets, where one might be limited by computational power, it may be argued that one could achieve a higher accuracy by removing the explanatory variables labeled as store and instead include more data.

4.2 Multilinear regression on dairy products

A similar analysis is carried out on a data set containing 2585 campaigns sampled from 37 different dairy products from 9 different large scale stores in the Stockholm and Scanian regions. The dairy products are chosen since there are sufficiently many articles in order to get a big enough data set and that articles within the dairy category are believed to have somewhat similar sale characteristics which would make them reasonable to bundle. Many dairy products also have short freshness dates, making them important to prognosticate. The campaigns are sampled from the beginning of 2014 to the beginning of 2016 and has been chosen so that each product is campaigned at least 4 times in each store during the scope of these 2 years. The response variate is the total number of sold items during the campaign week and the set is chosen so that each sampled campaign has a total weekly sale of at least 20 items.

Each campaign is then characterized by the variables discussed in Section 3, adding up to a total of 91 explanatory variables to explain the response variate that is the campaign weeks total sold volume of the given item in a store. This data is fit to a regression model using matlabs built in function *regress*. A column of ones is added to the data to also fit an intercept.

A linear regression model is fit to the whole data set. Then 109 campaigns with residual confidence intervals not containing 0 are marked as potential outliers. These are discarded and then a new model is fit to the remaining data points.

In Figure 4 and 5 the resulting outputs of the linear regression models fit to the data sets with and without outliers is shown together with the actual campaign sales. It can be observed that the model without outliers is a significantly better fit, i.e the data is closer to the regression line when campaigns that have a large deviation are discarded. This is also reflected in the R^2 statistics where the model with outliers generate a R^2 of 0.8876 and the corresponding R^2 for the model without outliers is 0.9439.

Continuing with the data not containing outliers, the explanatory variables are now bundled together as explained in Section 3.7 and new models with reduced number of explanatory variables are fit to the data generating R^2 statistics as in Table 4. Several models of bundled variables are fit to the data to sort out which bundles are most significant to the response variate.

Two noteworthy observations can be made: In this set the campaigns are sampled from 3 different regions, 2 in the Stockholm area and 1 in Scania. It can be observed that by removing the dummy coded variables corresponding to the stores has little effect on the resulting $R^2 = 0.9419$ but when also removing

Model	R^2	R^2-reduction	S.v reduction	Diff per s.v
Full model	0.9439			
Without stores	0.9419	0.0020	8	0.00025
Without stores and regions	0.9363	0.0076	10	0.00076
Without M/RC	0.9205	0.0234	26	0.00090
Without weather	0.9388	0.0051	4	0.00128
Without holidays	0.9439	0.000002	1	0.000002
Without prices	0.9350	0.0089	6	0.00149
Without items	0.9118	0.0321	36	0.00089
Without items and M/RC	0.8676	0.0763	62	0.00123
Without revenue	0.9434	0.0005	4	0.00013
Without#SoldItems	0.9090	0.0349	4	0.00874
Without revenue and #SoldItems	0.7466	0.1973	8	0.02467
Price $t, t-1$, #SoldItems	0.8429	0.1010	84	0.00120
Price $t, t-1$, #SoldItems, items	0.9070	0.0369	48	0.00077
Without revenue, stores, holidays	0.9413	0.00264	13	0.00020
Without revenue, stores, regions, holidays	0.9356	0.0083	15	0.00056
Without revenue, stores, weather, holidays	0.9357	0.0082	17	0.00049
Price, #SoldItems, items, regions, M/RC	0.9294	0.0145	19	0.00077
Without price*, revenue	0.9425	0.0014	7	0.00021
Without price*, #SoldItems $t-2, t-3$, revenue	0.9393	0.0046	9	0.00052
Without price*, revenue, stores	0.9405	0.0034	15	0.00023
Without price*, revenue, stores, regions	0.9347	0.0092	17	0.00054
Without price*, revenue, stores, weather	0.9352	0.0087	19	0.00046
Without price*, revenue, stores, holidays	0.9404	0.0036	16	0.00022

Table 4: Table of R^2 statistics associated with each bundle of explanatory variables. Revenue refers to revenue in SEK and #SoldItems refers to the total number of sold items during the campaign week. Price t and $t-1$ refers to the price during the campaign and the price the week before the campaign respectively. M/RC refers to the media and role codes. *In the bottom most 7 models the prices for the weeks $t-2, t-3, t-4$ has been removed from the model.

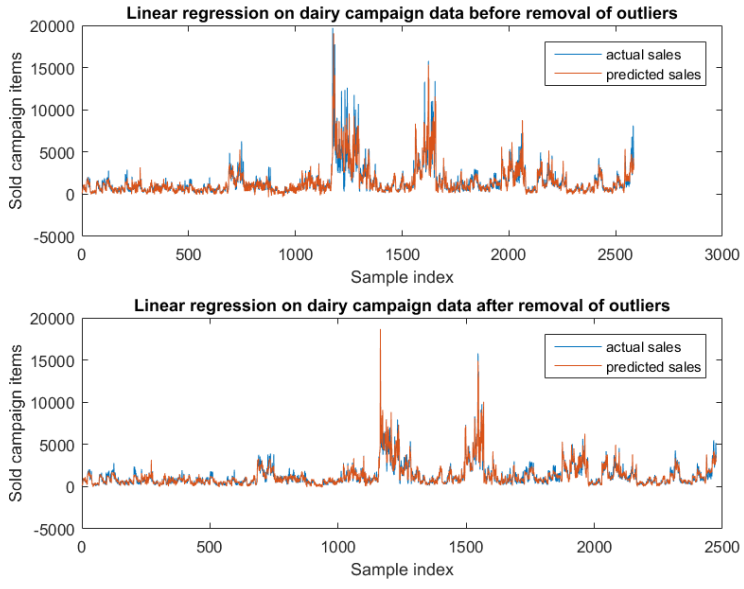

Figure 4: Output of linear regression models and the actual campaign sales on the whole dairy data set and the data set without outliers. The output is sold number of items during the campaign week in a store.

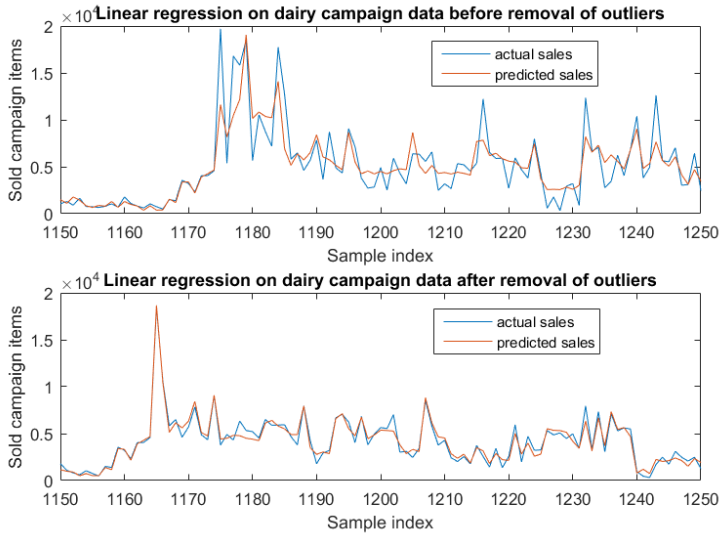

Figure 5: Enlargement of Figure 4 in the sample index interval 680 to 750. Note that since some samples have been discarded, the sample index in the lower graph is shifted to the left.

the regions the resulting $R^2 = 0.9363$ drops significantly lower. This suggests, based on this data set, that the variability explained by the dummy coded variables corresponding to the stores are mainly due to the regional variation that could just as well be explained by the regional variables. By removing the store variables the resulting model is reduced by 8 stochastic variables.

Another observation is that, in this set both the total revenue and number of sold items are included. Removing these separately has little effect on the R^2. For the revenue the difference is 0.0005 and for the number of sold items the difference is 0.0349. But if both are removed from the model the resulting R^2-reduction is 0.1973. This indicates that they explain the same kind of variation and that only one of the two are needed in the model. Since the response variate used here is the number of sold items, the corresponding explanatory variate is the more suitable.

5 Results

In this section the implemented models are evaluated by their mean squared error (MSE) of their one week prediction errors. That is, the difference between the models one week prediction and the actual realized sales of the campaign. This is done for each model on a test set and then on the whole data set.

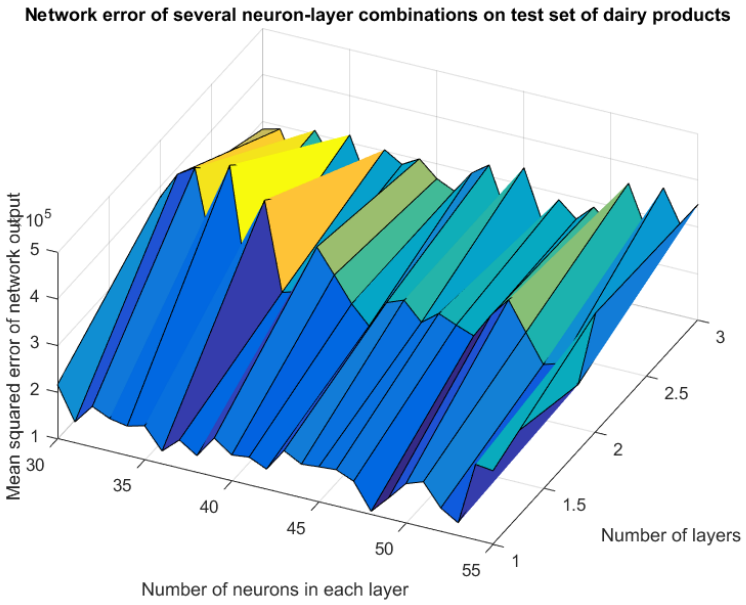

Figure 6: Mean squared error of model output on the test data from the averaged neural networks from the complex model. All network combinations between 30 and 55 neurons and 1 and 3 layers. From above.

5.1 Neural Network and Linear Regression on dairy products

In order to compare the benefits of model complexity, three models of varying number of stochastic variables are chosen from Table 4. As the most complex model the model including the total number of items sold the 4 weeks before the campaign, the prices during and the week before the campaign week as well as the relative price reduction, dummy coded items, regions, media and role codes and the SMHI weather data. This results in a total of 75 explanatory variates.

Here, 10% of the campaigns are removed from the data set to be used for validation and the remaining 90% of the campaigns are used as a training set. In Figures 6-9, 5 networks are trained on the training set for every combination between 30 and 55 neurons, and 1 and 3 layers, and the mean squared error of their average output is plotted. It can be observed that the choice of network design can play a somewhat important role. On both the test set and the whole set, the error seems to become more erratic and also somewhat increasing with the number of layers. Hence, a single layer network is chosen in this case. As for the number of neurons the error is more volatile for smaller number of neurons in the single layer case and seem to become more stable as the neurons are increased. With the test set having a minimum at 48 neurons and the whole set at 53. By these observations, a single layer of 48 neurons is chosen as the model for the data. Then 5 networks are trained and the used model output is the average of these 5, this in order to reduce the network error as explained in

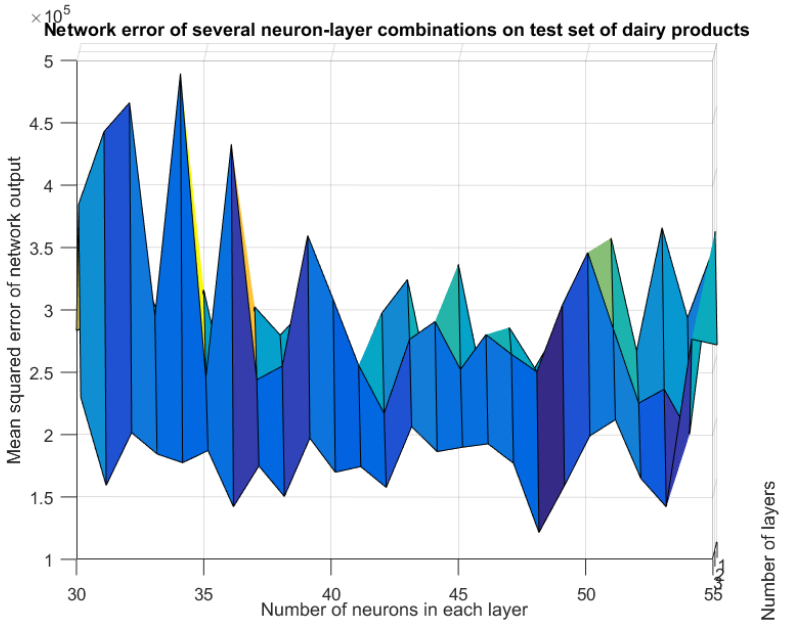

Figure 7: Mean squared error of model output on the test data from the averaged neural networks from the complex model. All network combinations between 30 and 55 neurons and 1 and 3 layers. From the side.

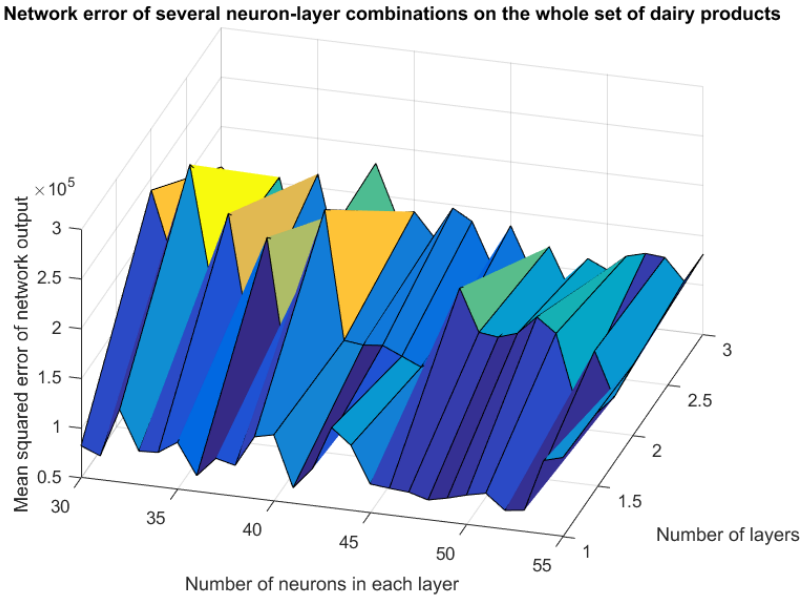

Figure 8: Mean squared error of model output on the whole data set from the averaged neural networks from the complex model. All network combinations between 30 and 55 neurons and 1 and 3 layers. From above.

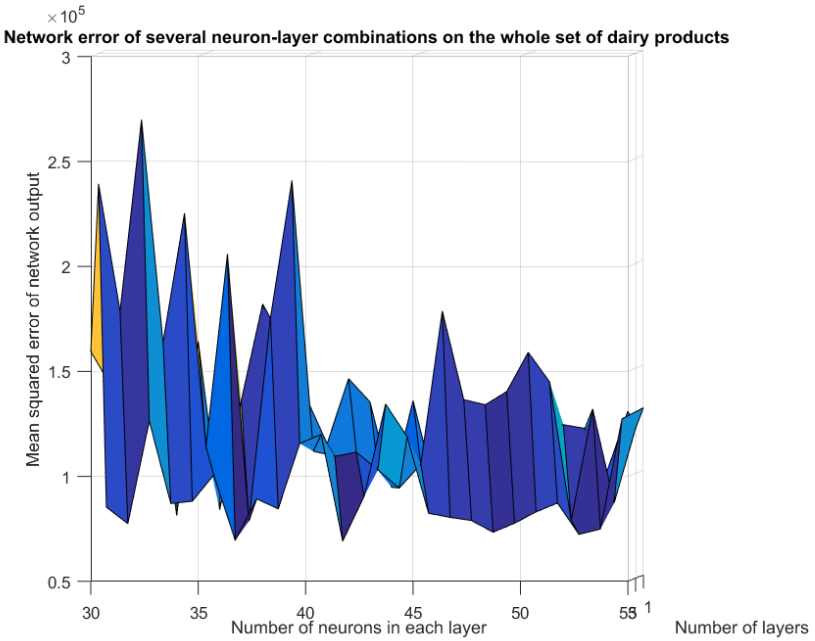

Figure 9: Mean squared error of model output on the whole data set from the averaged neural networks from the complex model. All network combinations between 30 and 55 neurons and 1 and 3 layers. From the side.

Section 2.7.

In Figure 10, the averaged network output and the linear regression output are plotted against index together with the actual campaign sales. The 5 networks individual MSE on the test set is: $10^5(3.3826, 1.2159, 1.7458, 2.5708, 3.8348)$ and the MSE of the average outputs are $1.5620 \cdot 10^5$. The MSE of the average outputs is less then the average MSE of the individual networks $2.5500 \cdot 10^5$, justifying the use of averaging of outputs. For the linear regression model the MSE on the test set is $1.2011 \cdot 10^5$. Over the whole set the MSE of the average network output is $7.4746 \cdot 10^4$ and for the linear regression the corresponding MSE is $1.1654 \cdot 10^5$.

The reason for the Neural Network having a slightly higher MSE than the linear regression on the test set and a significantly lower on the whole set is probably due to overfitting.

The dummy coded regions, media and role codes and SMHI data are now deducted from the model and a somewhat simpler model containing the total number of items sold the 4 weeks before the campaign, the prices during and the week before the campaign week as well as the relative price reduction and dummy coded items is derived. The resulting model contains 48 explanatory variables. Again the data is divided into a training and a test set and in Figures 11-14, 5 networks are trained on the training set for every combination between

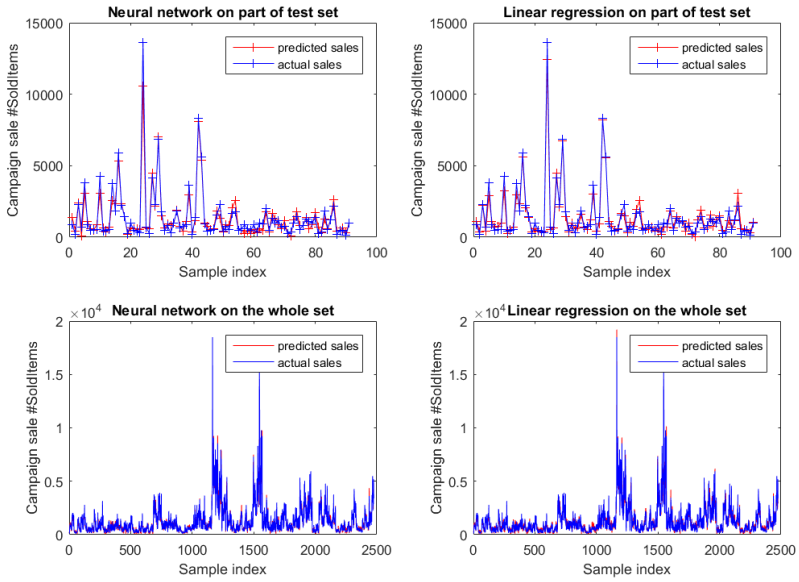

Figure 10: Outputs of average neural network and linear regression plotted against index from a data set containing dairy products. The two graphs on top are on a test set not used for training and the two on the bottom are outputs from the whole data set. Model with high complexity.

20 and 40 neurons and 1 and 3 layers, and their average output is plotted. On both the test set and the whole set, there is no obvious reduction in the error with the increase of layers. Thus, a single layer network is chosen for simplicity. In Figure 14 the error is decreasing with the number of neurons with a minimum at 39 neurons. The error also looks relatively small in the test set plots for that number of neurons so a single layer with 39 neurons is chosen as the model. Then, 5 networks are trained and the averaged output is used as model output.

In Figure 15 the averaged network output and the linear regression output are plotted against index together with the actual campaign sales. The 5 networks individual MSE on the test set is: $10^5(1.8223, 1.9337, 2.4412, 1.7963, 2.8099)$ and the MSE of the averaged outputs is $1.6387 \cdot 10^5$, where the average MSE of the 5 individual networks is $2.1606 \cdot 10^5$, again justifying the use of averaging. For the linear regression model, the MSE on the test set is $1.8995 \cdot 10^5$. Over the whole set, the MSE of the average network output is $1.4029 \cdot 10^5$ and for the linear regression the corresponding MSE is $1.8219 \cdot 10^5$

Here, we can see that the Neural Network has significantly more accurate predictions than the linear regression, in this case about 23% better comparing the MSE on the whole data set. But the networks are slower computationally. It takes the matlab function around 3 minutes to complete the training on the 5 networks, whereas the linear regression is carried out instantaneously.

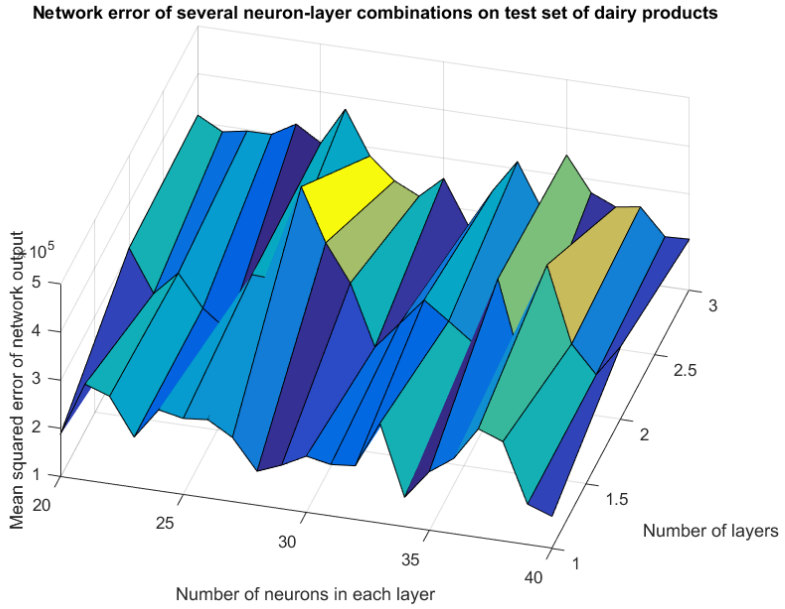

Figure 11: Mean squared error of model output on the test data from the averaged neural networks from the simpler model. All network combinations between 20 and 40 neurons and 1 and 3 layers. From above.

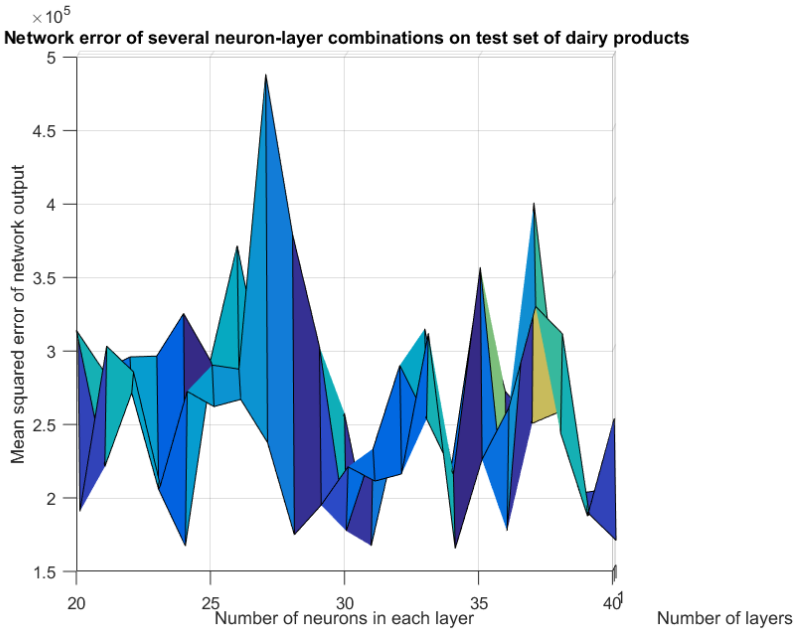

Figure 12: Mean squared error of model output on the test data from the averaged neural networks from the simpler model. All network combinations between 20 and 40 neurons and 1 and 3 layers. From the side.

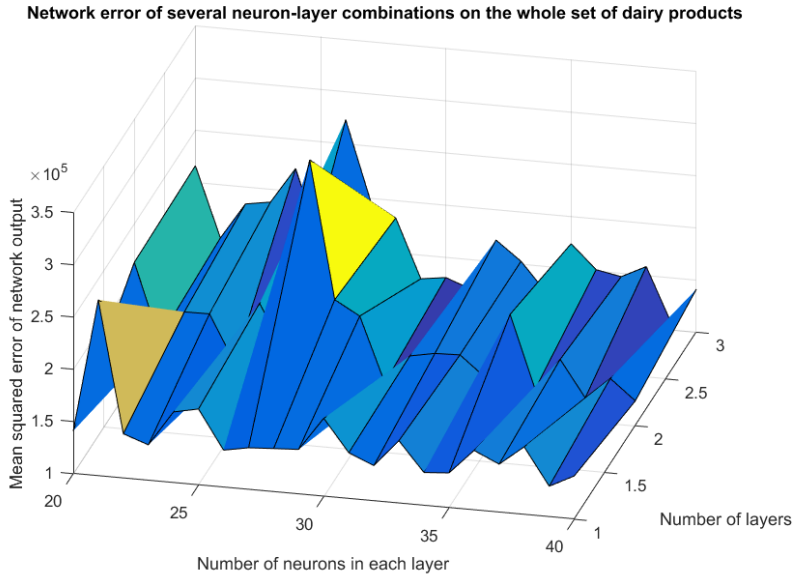

Figure 13: Mean squared error of model output on the whole data set from the averaged neural networks from the simpler model. All network combinations between 20 and 40 neurons and 1 and 3 layers. From above.

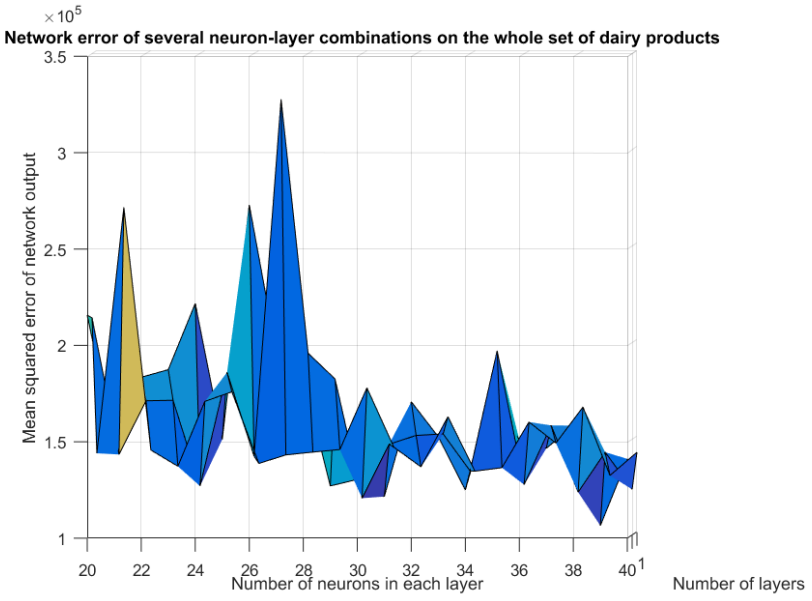

Figure 14: Mean squared error of model output on the whole data set from the averaged neural networks from the simpler model. All network combinations between 20 and 40 neurons and 1 and 3 layers. From the side.

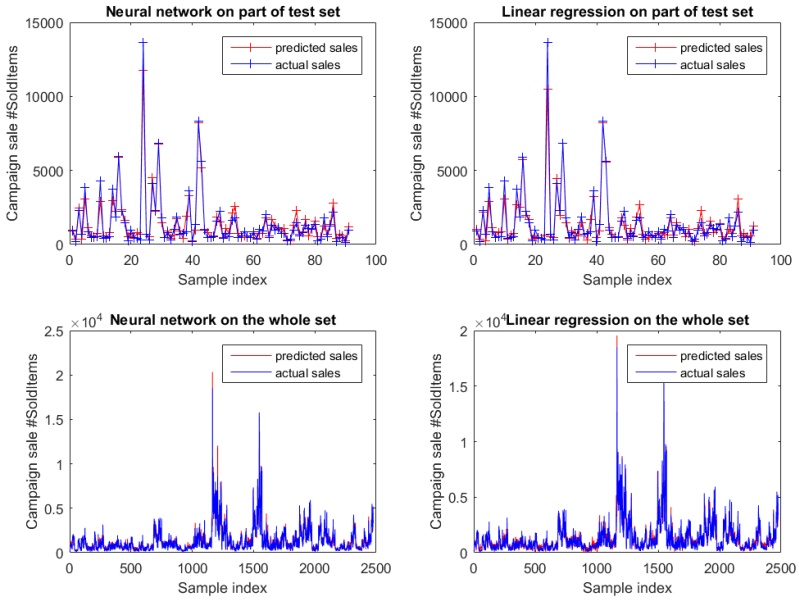

Figure 15: Outputs of average neural network and linear regression plotted against index from a data set containing dairy products. The two graphs on top are on a part of the test set not used for training and the two on the bottom are outputs from the whole data set. Simpler model, with medium complexity.

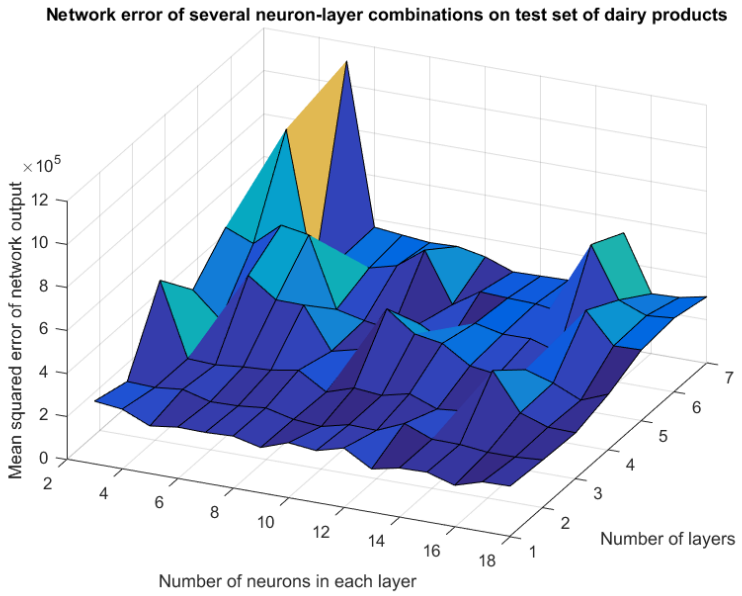

Figure 16: Mean squared error of model output on the test data from the averaged neural networks from the simplest model. All network combinations between 3 and 18 neurons and 1 and 7 layers. From above.

Now the dummy coded items are also deducted, and the remaining model consists only of the total number of items sold the 4 weeks before the campaign, the prices during and the week before the campaign week as well as the relative price reduction. Resulting in 7 explanatory variables.

Again the data is divided into a training and a test set. Then, 5 networks are trained on the training set for every combination between 3 and 18 neurons and 1 and 7 layers, and their average output is plotted. In Figures16-19, it can be observed that the number of neurons does seem significant for the single layer networks and that the error seems to be decreasing with the number of neurons. Looking at the plots, a network with $2-3$ layers and $8-10$ neurons is generating the least network error. However, this error is not much smaller than that of a much simpler single network with 9 neurons. Thus, the 9-neural single layer network is chosen as the model.

In Figure 20, the averaged network output and the linear regression output are plotted against index together with the actual campaign sales. The 5 networks individual MSE on the test set is: $10^5(2.8712, 2.5609, 2.1745, 3.6168, 2.1470)$ and the MSE of the averaged outputs are $2.3475 \cdot 10^5$. For the linear regression model the MSE on the test set is $3.3273 \cdot 10^5$. Over the whole set the MSE of the average network output is $2.1927 \cdot 10^5$ and for the linear regression the corresponding MSE is $3.0660 \cdot 10^5$.

Again the Neural Network is significantly better than linear regression, now

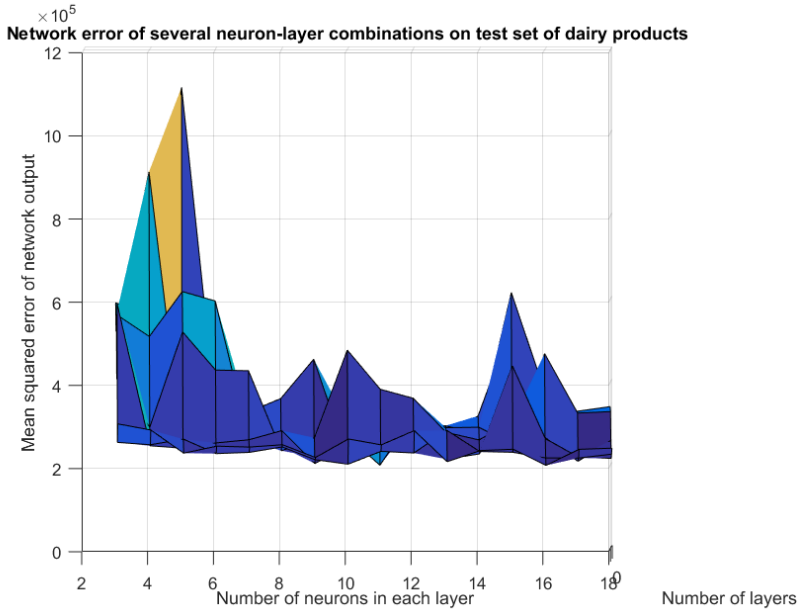

Figure 17: Mean squared error of model output on the test data from the averaged neural networks from the simplest model. All network combinations between 3 and 18 neurons and 1 and 7 layers. From the side.

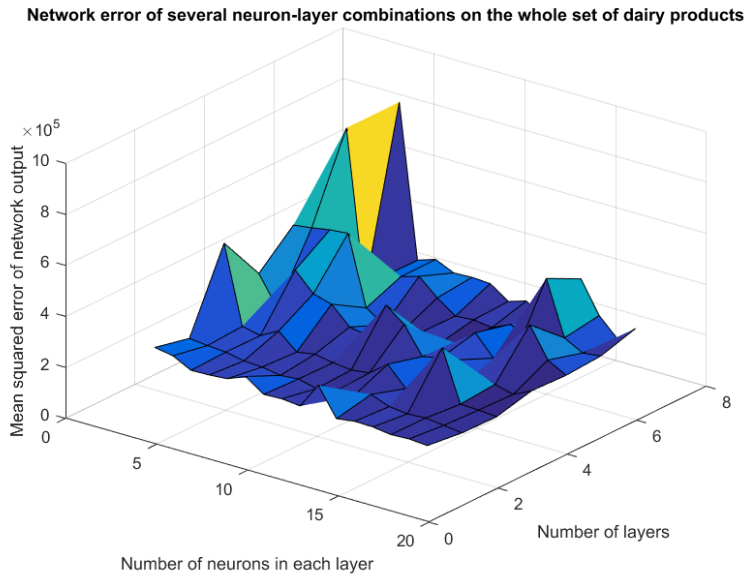

Figure 18: Mean squared error of model output on the whole data set from the averaged neural networks from the simplest model. All network combinations between 3 and 18 neurons and 1 and 7 layers. From above.

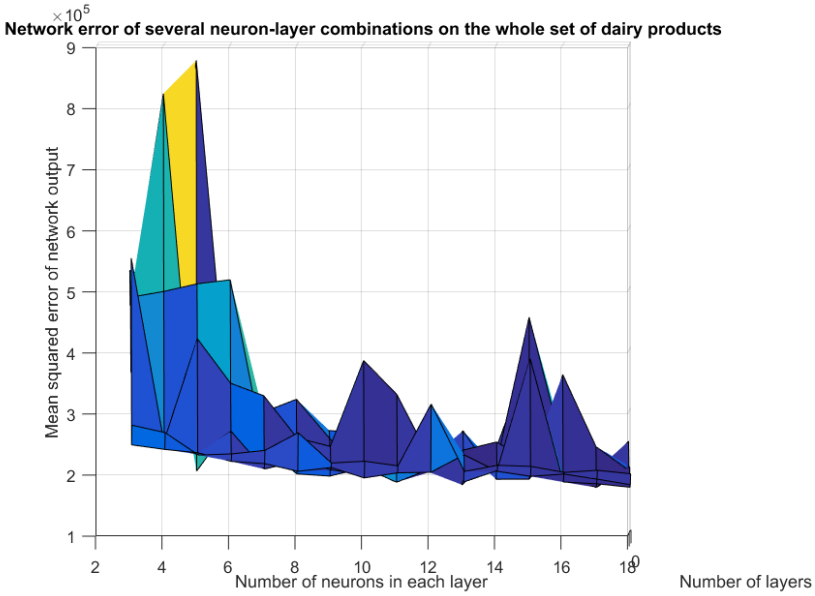

Figure 19: Mean squared error of model output on the whole data set from the averaged neural networks from the simplest model. All network combinations between 3 and 18 neurons and 1 and 7 layers. From the side.

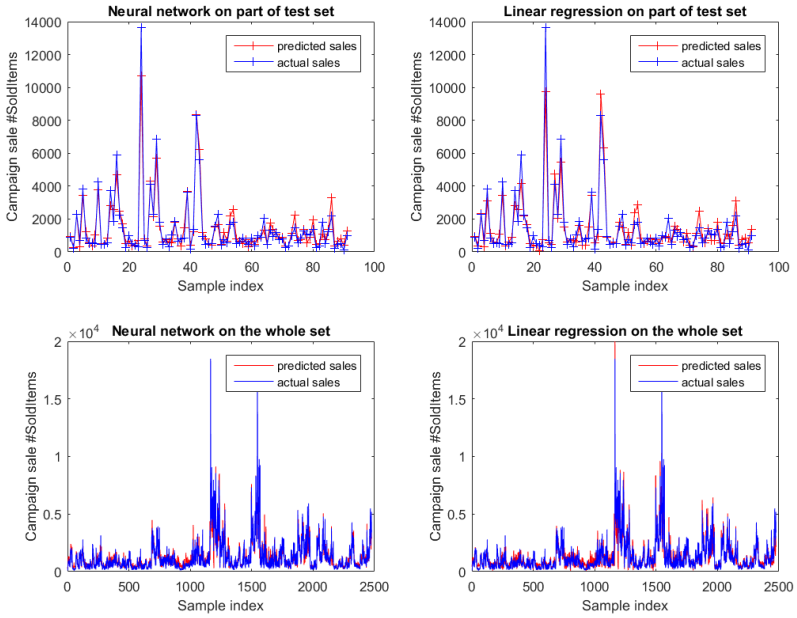

Figure 20: Outputs of average neural network and linear regression plotted against index from a data set containing dairy products. The two graphs on top are on part of a test set not used for training and the two on the bottom are outputs from the whole data set. Simplest model, with low complexity.

Model:	Complex	Simpler	Simplest
Number of explanatory variates	85	48	7
Network MSE on the test set	$1.5620 \cdot 10^5$	$1.6387 \cdot 10^5$	$2.3475 \cdot 10^5$
Network MSE on the whole set	$7.4746 \cdot 10^4$	$1.4029 \cdot 10^5$	$2.1927 \cdot 10^5$
Linreg MSE on the test set	$1.2011 \cdot 10^5$	$1.8995 \cdot 10^5$	$3.3273 \cdot 10^5$
Linreg MSE on the whole set	$1.1654 \cdot 10^5$	$1.8219 \cdot 10^5$	$3.0660 \cdot 10^5$

Table 5: Table of MSE for the different models fitted to the data set of dairy products.

about 29% better on the whole set, and now it is not so much at a computationally disadvantage either. It only takes a few seconds to do the training. Noteworthy is also that looking at the graphs in Figure 20, the simplest model seem to be able to generate some decent predictions only using 7 explanatory variates. Comparing this to the more complex models using 75 and 48 variables, respectively. It is not that much worse and the simplest model could in some cases be sufficient.

Collecting the MSE of the three models in Table 5, it can be concluded that the most complex model has the lowest MSE on the test set.

5.2 Integrating Neural Network output with time series data

To illustrate the neural network approach in a time series context, 5 networks consisting of 1 layer with 48 neurons has been fitted to the dairy products data set, using the model referred to as the simpler model. Four campaigns, corresponding to the 2 most recent campaigns of a cottage cheese and a yoghurt product in one store, are extracted from the data set and the neural network is trained on the remaining campaigns. The used output is the average output of the 5 networks. The networks are then used to predict all of the yoghurt and cottage cheese campaigns.

In order to do predictions on the corresponding time series of the weekly sales the chosen stores weekly sold volumes of the cottage cheese and yoghurt product is regarded. Since the model is supposed to predict the sales during the campaign weeks, these weeks are taken out of the data set and the time series are interpolated, using a regular average of the weeks before and after the campaign.

An ARMA(3,1)-model is fit to the cottage cheese time series and an ARMA(1,1)-model is fit to the yoghurt time series. In Figures 21 and 22, we can see the diagnostic plots of the model residuals of the ARMA-models. The ACF and PACF are mainly inside of the red approximate 95% confidence intervals for all lags, and for both items. Although there seem to be some upper tail dependence in the normal probability plot for the cottage cheese, the normal probability plot of both model residuals seem to indicate approximately normally distributed residuals. The Ljung-Box-Pierce, McLeod-Li, Monti and sign change tests are done on the model residuals, all indicating that the residuals are white for both models. For more on these tests, please see [2] p 193.

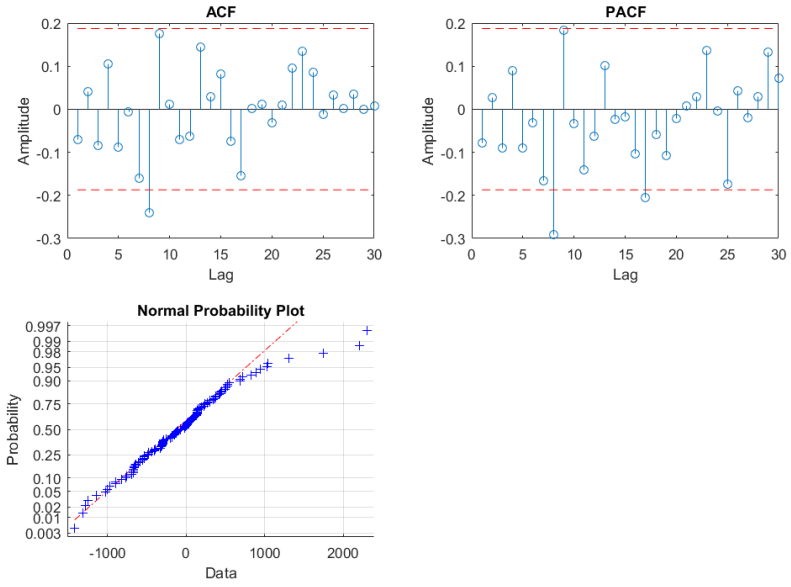

Figure 21: Sample ACF, PACF and normal probability plot for cottage cheese. The red horizontal lines correspond to the approximate 95% confidence intervals.

The ARMA models are now used to do 1-step predictions on the weekly sales of the respective items and then the predicted sold volume during the campaign weeks are replaced by the outputs from the neural networks. For the cottage cheese this is done for week $15, 21, 39, 56, 67, 80, 107$, where week 80 and 107 have not been used for training of the networks. For the yoghurt this is done for week $23, 30, 47, 62, 75, 80$, where week 75 and 80 have not been used for training of the networks. The 1-step predictors and the corresponding actual sales are plotted for the cottage cheese in Figure 23 and the yoghurt in Figure 24.

It can be observed that although being a bit off, the neural network outputs seem to capture the majority of the uplift in the sales during the campaign weeks in Figures 23 and 24. Even during the weeks not used in the training set, that is week 80 and 107 in Figure 23 and week 75 and 80 in Figure 24. The network output is not spot on, but at least registers a clear uplift during the campaigns.

6 Discussion

6.1 Outliers

The outliers are being detected as explained in Section 3, by flagging all residuals with confidence intervals not containing 0. Hopefully, most of these data points, flagged as outliers, are subject to some kind of errors discussed in Section 3.8 and can safely be removed. By discarding this corrupted data the model accuracy

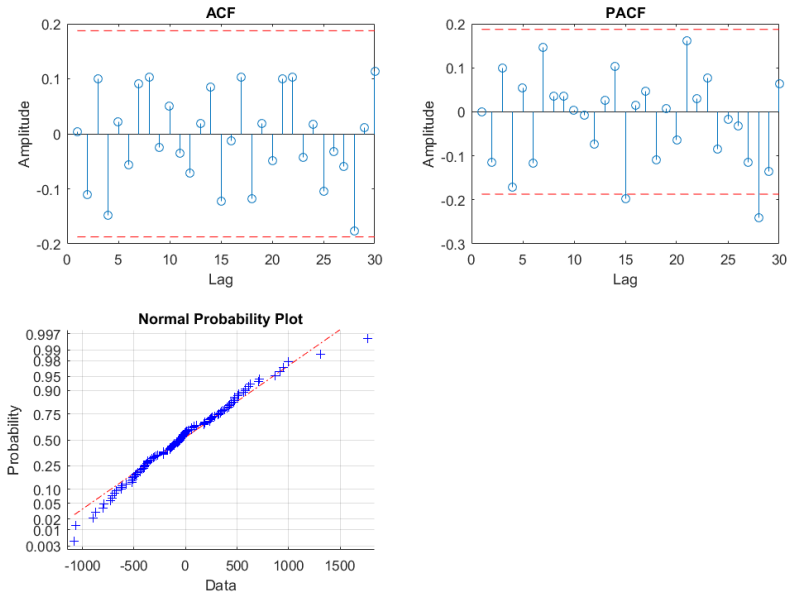

Figure 22: Sample ACF, PACF and normal probability plot for yoghurt. The red horizontal lines correspond to the approximate 95% confidence intervals.

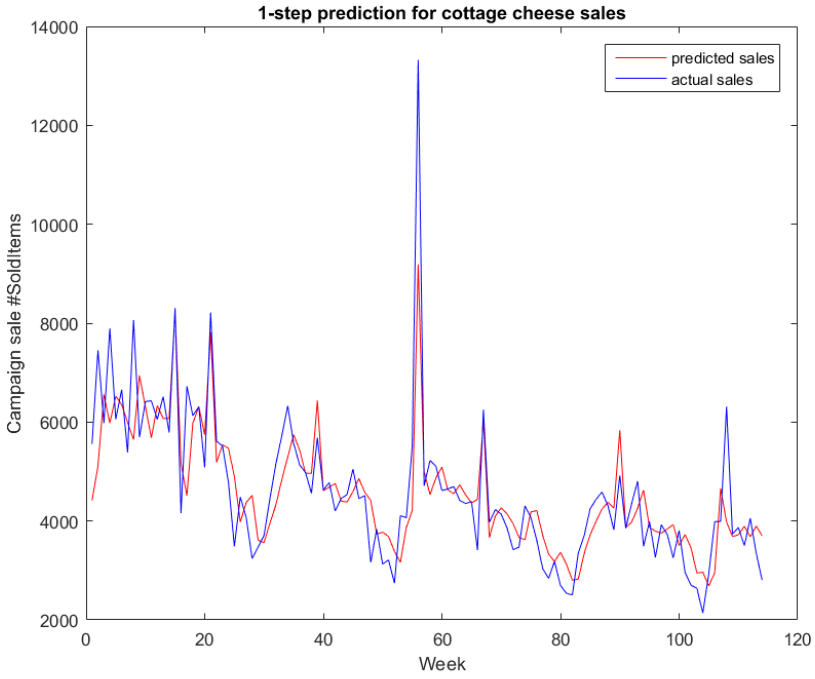

Figure 23: 1-step predictions of an ARMA(3,1)-model and the neural network output of the campaigns of a cottage cheese time series for one store.

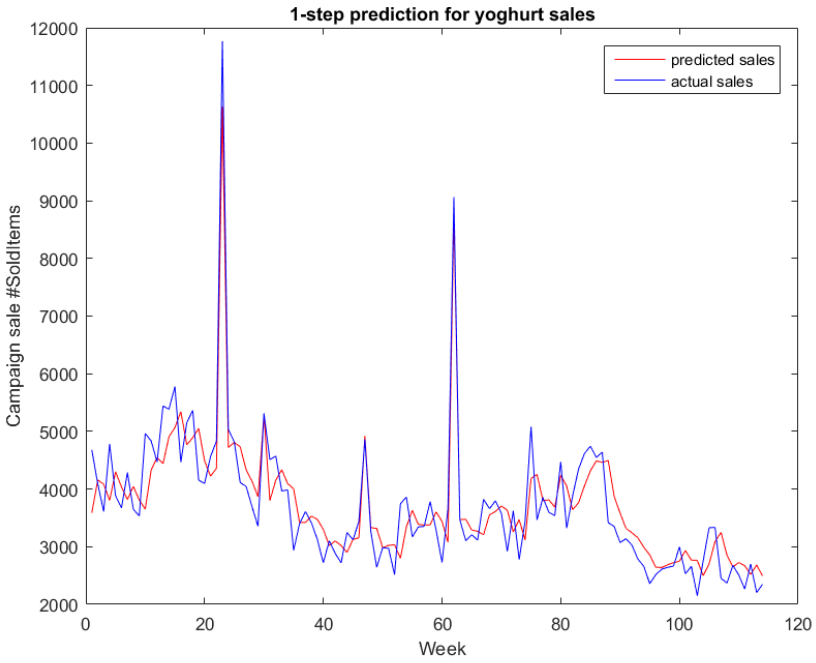

Figure 24: 1-step predictions of an ARMA(3,1)-model and the neural network output of the campaigns of a yoghurt time series for one store.

is increased. It should be noted, however, that valid data may be removed in the process, so there is a trade off between increased accuracy and the fact that important data points might be discarded.

6.2 Neural Network or Linear regression

In Section 5, it is noted that the Neural Network model in the complex case seem to be suffering from overfitting. Overfitting is known to happen when there are too many fitted parameters in relation to the number of data points. Making the model pick up accidental structures and noise instead of the underlying relationship. The model fits the prediction function too close to the data points and the predictions on data points not in the set can become inaccurate. There may be ways to deal with this in the case of the complex model and that is something that could be further investigated.

In Section 5 Table 5, the results show that the linear regression seem to best model the data when looking at the predictions on the test set, although the Neural Network is only slightly behind in terms of accuracy. But for the whole data set the Neural Network outperforms the linear regression. The Neural Network is also better when it comes to both the test set and the whole set for the simpler and simplest models and it can be concluded that Neural Networks seem to be better than the linear regression in most cases.

6.3 Model complexity and Network design

As can be seen in Table 3 and 4 the most complex models are the most accurate in terms of R^2. In practice however it is desirable to have a model of manageable size and thus less significant inputs, not contributing much to R^2, can be discarded if the trade off between accuracy and model simplicity is justified.

Section 5 Table 5 shows that the complex model performs best in terms of accuracy, followed by the simpler and simplest model. But that the MSE of the more simpler models are not that much higher considering how many variables that are being deducted. Making models simpler and more generic may in some cases be motivated even though there is a loss in accuracy and may even be forced in order to make the models implementable. It should also be emphasized that the more complex models take significantly longer time computationally. If the data sets where expanded to include more items the number of input variables of the models referred to as complex and simpler would be increased as they contain dummy coded variables, such as stores and items, increasing the training time further. For large sets of data, with many different items, these models may become infeasible in practice.

When it comes to model complexity the addition of extra layers and training of so-called deep networks did not prove to be useful for the data discussed in this study. From the network plots, i.e Figures 6-9,11-14 and 16-19, no evidence of increase in accuracy, to justify the added complexity, was found.

6.4 Choice of searching algorithm

In this project, the Levenberg-Marquardt algortihm, as discussed in Section 2.6, has been used for the network training. LM is chosen because of its known balance between efficiency and accuracy and it is the standard algorithm used in the matlab toolbox. But other choices of algorithms could be used in order to improve computational efficiency or prediction accuracy.

6.5 Model performance

The models discussed in this study are far from perfect but could in some cases be useful. The simplest Neural Network model is surprisingly good compared to the more complex ones and would be both easy to implement and generic enough to include arbitrary stores and items. One reason for its good performance may be that the data set only consists of dairy products, and hence similar items, and this could lead to loss in accuracy if other item categories are to be included. However, this could be solved by integrating data clustering methods into the model, or by the use of separate models to train on similar categories of items.

One obvious way to increase model performance would be to include data dating further back in time than the beginning of 2014. This would make it easier to model long term trends and yearly cycles, as well as expand the data set with more relevant campaigns per item. This is believed to be able to improve the model performance significantly since in the above data sets some items are only campaigned 4 times per store during the sampling period and ideally this number should be higher. This adjustment can not be made at the time but could be further investigated.

7 Conclusions

It is established that the most important factors governing the sold volumes of a campaign item during the campaign week is the sold volume the weeks before the campaign and the price and price decrease associated with the campaign. The separation of items and campaigns with specific media and role codes also hold significance, but in order to achieve simple models this could be neglected. In the data studied, the separation of stores seem to be adequately replaced by the separation of the corresponding regions. It can also be concluded that there seem to be no reason to include the seasonal variation separately into the models as the seasonality of the items seems to be adequately represented in the sales of the preceding weeks.

In all but one case, the Neural Network outperformed the linear regression models and thus based on the data considered here the Neural Network seem to be a preferable modelling method. As no obvious increase in accuracy was achieved by the addition of extra layers in the networks, this seemed to be redundant and single layered networks were used.

The best model was in the most complex case a linear regression model using 85 parameters, which performed quite well accuracy wise. In the simpler case

a single layer neural network with 39 neurons on 48 input variables is deemed the best model. With only about 20% increase in MSE, the loss in accuracy to the more complex model is low enough that this could still be considered an adequate representation of the campaign sales. In the simplest case, a single layer network of 9 neurons on 7 input variables was considered the best model. This model has about 89% higher MSE than the most complex model but uses a small amount of input variables; these variables are also independent of the store and item that is campaigned which makes the model generic.

References

[1] Brodersen, K. H., Galluser, F., Koehler, J., Remy, N. and Scott, S. L. (2015). Inferring causal impact using Bayesian structural timeseries models. *The Annals of Applied Statistics* 2015, Vol. 9, No. 1, 247–274.

[2] Jakobsson, A. (2015) An introduction to time series modeling, 2015, Edition 2:1.

[3] Kriesel, D. (2007) A brief introduction to Neural Networks, dkriesel.com

[4] MathWorks^{TM}, (2016). Neural Network Toolbox^{TM} (R2016a). Retrieved January 2016 from:se.mathworks.com/products/neural-network

[5] MathWorks^{TM}, (2016). Neural Network Toolbox^{TM} (R2016a) User's guide. Retrieved May 2016 from: se.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf

[6] Nocedal, J., Wright, S.J. (2006). Numerical Optimisation, Edition 2.

[7] Penpece, D., Orhan, E.E. (2014). Predicting Sales Revenue by Using Artificial Neural Network in Grocery Retailing Industry: A Case Study in Turkey *International Journal of Trade, Economics and Finance*, 2014, Vol. 5, No. 5.

[8] Rojas, R. (1996), Neural Networks, 1996, page.mi.fu-berlin.de/rojas/neural

[9] Thiesing, F.M., Middleberg, U., Vornberger, O. (1995). Short Term Prediction of Sales in Supermarkets *Proceedings of International Conference on Neural Networks* (ICNN'95) , 1995, Issue 2, p 1028 to 1031.

[10] Thiesing, F.M., Vornberger, O. (1997). Sales forecasting using neural networks *Proceedings of International Conference on Neural Networks* (ICNN'97) , 1997, Issue 4, p 2125 to 2128.

[11] Yu, H., Wilamowski, B.M. (2011). Levenberg–marquardt training *Industrial electronics handbook*, 2011, second edition, p 12-1 to 12-15.