Erasmus University Rotterdam
Erasmus School of Economics

Bachelor Thesis [Programme Econometrie en Operationele Research]

Short-term electric load forecasting based on weather predictions obtained with a Kalman filter-extended neural network

Name student:
Willemijn Ouwersloot

Student ID number:
481808

Supervisor:
S.H.L.C.G. Vermeulen

Second assessor:
J.A. Oorschot

Abstract

Forecasting electricity demand, or load, is important to energy utilities as it guarantees efficient management of power systems. Four load forecasting models are frequently used for this task: linear regression, multivariate adaptive regression splines, artificial neural network, and support vector regression. These load models often contain a weather component. When forecasting load, however, one then also needs to have access to weather forecasts. In this paper we answer the question as to how we can best forecast the weather, and given this forecast, which of the four aforementioned load models produces the most accurate load predictions in the short term. Using data on electric load and weather in the Netherlands for the period 1 May, 2013, up until and including 30 April, 2019, we find that weather can best be modeled with a neural network extended with a Kalman filter. Given these weather predictions, the support vector regression model results in the most accurate load forecasts.

Date final version: July 5, 2020

Contents

1 Introduction

A lot of research has been done regarding short-term electricity demand, or load, forecasting. More specifically, four different methods of modeling load are commonly used: linear regression (LR), multivariate adaptive regression splines (MARS), artificial neural network (ANN) and support vector regression (SVR). Moreover, it is known that the weather can influence the load on a certain day, as stated by Yao, Song, Zhang, & Cheng (2000). Therefore, in order to forecast load on a particular day, we need information regarding the weather for that day. The exact weather data, however, is unknown at the time of forecasting: there are stochastic regressors present in the load forecasting models. If the values of these stochastic weather variables are better approximated, this can result in more accurate load forecasts. When weather forecasts are not readily available, a model to predict weather for the purpose of load forecasting is needed. The current literature in this regard is lacking.

This gives rise to the research question answered in this paper: 'How can we best use weather data when forecasting short-term electric load, and given this information, which method out of LR, MARS, ANN, and SVR, results in the highest predictive accuracy when forecasting the short-term electric load?' In order to answer this research question, we first look at what the LR, MARS, ANN, and SVR models entail and how they can be applied to model load. Moreover, we search for the best way to predict the weather variables. Finally, we use the most accurate forecast of the weather variables in order to determine which of the four models results in the most accurate short-term predictions of electric load.

Having access to accurate short-term forecasts of electric load guarantees efficient management of power systems (Taylor, 2010). Al-Musaylh, Deo, Adamowski, & Li (2018) state that robust forecast models are essential for the operation of energy utilities, as they help determine load switching and power grid management decisions. A slight increase in the forecasting error can lead to a loss of millions of dollars.

As electric load forecast models are of such an importance, they have been widely studied in literature. However, there has not been a lot of research with regard to electric load forecasting in the Netherlands. The existing literature focuses on only a few provinces at a time, as done by McSharry, Bouwman, & Bloemhof (2005) and Erişen, Iyigun, & Tanrısever (2017). Nonetheless, in the current deregulated electricity market there are electricity providers that operate nationwide, and not necessarily only regionally. For these providers, it is useful to get insights in electric load in the Netherlands as a whole, something that has not yet been done in the current literature. For that reason, in this paper we will model electric load in the Netherlands as a whole.

We use data obtained from ENTSO-E (2020) on hourly electric load in the Netherlands. Moreover, we use data obtained from KNMI (2020) on hourly measurements of temperature, humidity, wind speed, and global solar radiation in de Bilt, the Netherlands. For both data sets, we choose to use data for the period 1 May, 2013, up until and including 30 April, 2019.

As previously mentioned, LR, MARS, ANN, and SVR are commonly used to forecast short-term

load and show promising results. We therefore compare their performances in this paper. LR is widely studied in electric load prediction, for example by Hong, Gui, Baran, & Willis (2010). MARS and SVR are studied and compared by Al-Musaylh et al. (2018). SVR is evaluated in detail by for example Che, Wang, & Tang (2012), and they have shown that the load forecasts constructed by SVR are of a good quality. ANN is a commonly used load forecasting technique as well, and is used by for example Park, El-Sharkawi, Marks, Atlas, & Damborg (1991).

However, in order to actually predict electric load, we need forecasts of the weather circumstances, as explained above. We construct these forecasts in three ways: with a naive forecasting method, a neural network forecasting method (both based on the approach described by Hippert & Pedreira (2004)), and a neural network forecasting method combined with an adjusted Kalman filter (loosely based on the method described by Galanis & Anadranistakis (2002)). In this case, the Kalman filter is applied in order to adjust the forecasts for any systematic error that could still be present in the forecast errors. To the best of our knowledge, combining weather forecasts obtained with a neural network with the use of a Kalman filter in this way, has not yet been done.

Using the most accurate weather forecast, we test whether the four load forecasting models differ significantly with regard their predictive accuracy by applying the Diebold-Mariano test as described by Diebold & Mariano (2002). Moreover, the efficiency of the forecasts is evaluated by performing a Mincer-Zarnowitz regression, as proposed by Mincer & Zarnowitz (1969).

We find that, with regard to what weather forecast to use, the predictions for the weather variables obtained with the neural network method combined with the adjusted Kalman filter perform best. Therefore, we use these predictions when forecasting load. Moreover, it follows from the results that the SVR model generates significantly more accurate predictions than the other three load forecasting models. We therefore recommend energy utility companies to predict load using the SVR model, and to use the neural network with Kalman filter predictions for weather variables in case no weather forecast is readily available.

We start our paper by first discussing the existing literature in Section 2. In Section 3, we describe the data used in this research. Then, in section 4 we discuss the methodology of the four load forecasting models used in this paper as well as the three methods used for forecasting the weather. We present our results in Section 5. In Section 6 we draw a conclusion from our results as to what is the best way to go about short-term load forecasting. Moreover, we make some suggestions for future research.

2 Literature review

Electric load forecasting in general is a widely studied topic in existing literature. Load forecasting can be done for different time periods. In this paper we follow the classification as given by Nalcaci, Özmen, & Weber (2019). They state that very short-term load forecasting refers to a period up to one day, short-term load forecasting addresses the period from one day up to two weeks, medium-term

load forecasting focuses on the period from two weeks up to a year, and long-term load forecasting concerns a period longer than a year.

The difference between longer and shorter term load forecasting rests mainly upon the objective of the forecast: Nalcaci et al. (2019) state that long-term load forecasting is used for strategic planning, whereas Taylor (2010) mentions that short-term load forecasting is used to make day-to-day decisions regarding the management of power systems. In this paper we focus on short-term load forecasting: we construct forecasts for electric load up to 24 hours ahead. Henceforth, whenever load forecasting is mentioned, we refer to short-term load forecasting.

Load models usually contain four components, as described by Yao et al. (2000): a normal load component which is weather independent, a weather sensitive component, a component that accounts for the occurrence of unusual events (such as big sporting events), and a random load component. Multiple variables can be included in the weather sensitive component. Beccali, Cellura, Brano, & Marvuglia (2004) incorporate variables on temperature, wind speed, humidity, and global solar radiation in their model. These variables, or a subset thereof, are often used in load forecasting models, for example also by Nalcaci et al. (2019). Therefore, we incorporate these four weather variables in the models described in this paper.

There are several models often used for electric load forecasting. One of these models is the LR model, for example evaluated for short-term load forecasting by Hong et al. (2010). They mention that the LR model is one of the most widely studied techniques to forecast load. However, there are other models which are also frequently used to model load. Nalcaci et al. (2019) investigate the use of MARS and ANN models in long-term load forecasting, and find their performance to be good. Moreover, Al-Musaylh et al. (2018) find that in addition to the MARS model, the SVR model can be useful for forecasting load. Furthermore, Che & Wang (2014) mention that the SVR model can produce accurate forecasts of short-term electric load, provided that an accurate kernel function is used in the model. As LR, MARS, ANN, and SVR all seem to provide accurate forecasts, we will compare the performance of these four models in order to find the best load forecasting model.

An important part in determining the usefulness of load forecasting models is the forecasting accuracy. For a given time, the model predicts the electric load based on the input variables. However, as the exact values for the weather variables are not yet known at the time of forecasting, there are stochastic regressors in the model. In order to produce a forecast for the load, we therefore first need a prediction for the weather data at that time. Hippert & Pedreira (2004) evaluate several methods that aid in forecasting the values of weather variables. They show that neural networks can help predict the values of future weather variables. In fact, the neural network predictions are better than the predictions based on the naive forecasting method, in which the future values of the weather variables are assumed to be equal to the values today. Moreover, Galanis & Anadranistakis (2002) present that the use of a Kalman filter can further improve forecasting accuracy. We compare the naive forecasting method with the neural network weather forecast and with the forecast produced by the neural network extended with the Kalman filter, in order to find the method that provides

us with the most accurate forecasts of the weather variables.

In order to draw conclusions as to whether one of the load forecasting models indeed outperforms the others, Diebold & Mariano (2002) propose a test: the Diebold-Mariano test. We apply this test to the four load forecasting models in order to select the best one. Moreover, a Mincer-Zarnowitz regression as proposed by Mincer & Zarnowitz (1969) is applied to determine the efficiency of the forecasts.

3 Data

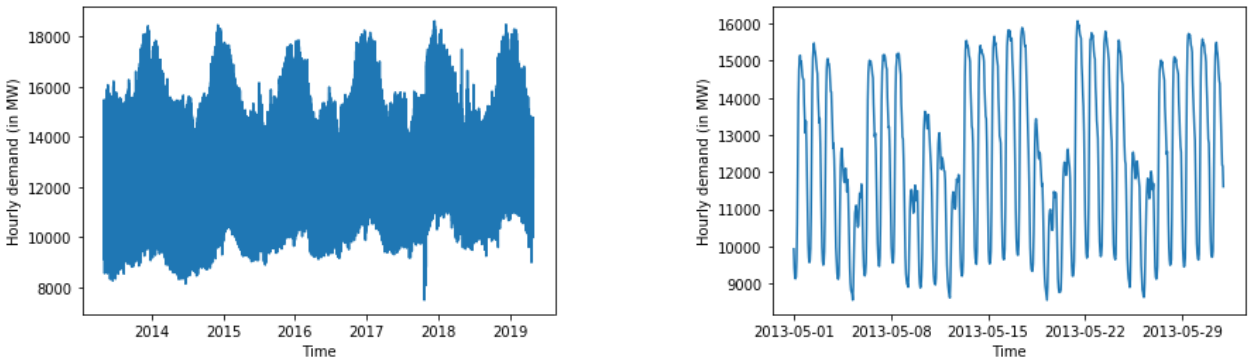
In this research we use data on electric load as well as weather. Historical hourly electricity load data is obtained from ENTSO-E (2020). The load data concerns the hourly demand for electricity in MW in the Netherlands for the period 1 May, 2013, 00:00 up until and including 30 April, 2019, 22:00. Therefore, we use very nearly six years of data. There are two missing values in the data: one for the load on 30 March, 2014, 02:00 to 03:00 and the other for the load on 29 March, 2015, 02:00 to 03:00. We deal with these missing values by linear interpolation. Historical hourly weather data for the same period is obtained from KNMI (2020) and contains no missing values. The data concerns hourly measurements at weather station de Bilt of wind speed in 0.1 m/s, temperature in 0.1 °C, global radiation in J/cm^2, and humidity in percentages. For the sake of uniformity we do not take the 29$^{\text{th}}$ of February of 2016 into account, as this is a leap day, which only occurs once every four years. Therefore, in total, there are 52558 observations on all variables.

Figure 3.1a shows the hourly load for the entire sample period. We can see that during winter, there seems to be a higher demand for electricity, whereas during the summer there seems to be significantly less demand for electricity. As the figure is quite dense, the load during the first month in the sample period is shown in Figure 3.1b. This figure shows that in addition to a yearly pattern, there also seems to be a weekly and even daily pattern in the electricity demanded: demand seems to be lower on weekends and during the night.

As the load data we want to model is a time series, we perform the Augmented Dickey-Fuller (ADF) test in order to check for the presence of a unit root. We include an intercept and perform the test as described in Heij, de Boer, Franses, Kloek, & van Dijk (2004a). As there seems to be seasonality on at least three levels, we perhaps expect the load time series to be nonstationary. However, the ADF test rejects the null hypothesis of nonstationarity. We conclude that the seasonality observed in the time series is deterministic by nature and not stochastic. All weather time series are also stationary by the ADF test. Therefore, no first differencing is needed in the model.

An overview of the summary statistics of the load and weather variables is presented below, in Table 3.1. Moreover, Table 3.2 shows the correlation between the load and weather variables.

As previously mentioned, there seems to be a yearly, weekly and daily pattern in the data. In order to account for this in the models, we construct several dummy variables. First, we construct dummy variables for the different hours of the day. Moreover, we construct three dummies for



(a) Full sample

(b) May 2013

Figure 3.1: Hourly demand of load over the full sample period and the first month of the sample period. We can discern three seasonal patterns in the data: a yearly pattern, a weekly pattern, and a daily pattern. Load seems to be higher in winter, during the week, and during the day. It seems to be lower in summer, during the weekend, and during the night.

Table 3.1: Summary statistics of the load and weather data.

	Minimum	**Maximum**	**Average**	**Std. Dev.**
Load (in GW)	7.49	18.62	13.02	2.15
Temperature (in 0.1 °C)	-84.00	354.00	111.21	66.06
Humidity (in %)	17.00	100.00	79.99	15.33
Wind speed (in 0.1 m/s)	0.00	140.00	33.84	18.16
Global solar radiation (in J/cm^2)	0.00	340.00	43.40	70.55

Table 3.2: Correlation between variables.

	Load	**Temperature**	**Humidity**	**Wind speed**	**Radiation**
Load	1.00				
Temperature	-0.05	1.00			
Humidity	-0.18	-0.50	1.00		
Wind speed	0.21	0.03	-0.32	1.00	
Radiation	0.16	0.54	-0.71	0.15	1.00

Saturdays, Sundays and national holidays. Finally, we also construct two dummy variables for the winter and summer periods.

As there could be dependency between the loads on consecutive times, we include several lags in the model. In order to determine the order of the included lags, we look at the (partial) auto-correlation function ((P)ACF) of the deseasonalised load series, as done by Fard & Akbari-Zadeh (2014). We do not take the entire load series into account. Rather, we use a training period of 1 May, 2013, up until and including 30 April, 2018, for our models. The period of 1 May, 2018, up until and including 30 April, 2019, is the testing period for our models and will be used to evaluate the forecasting performance of the models. Based on the (P)ACF, we include two lags of load in our models (See Figures A.1a and A.1b in the Appendix). Moreover, as the seasonal dummies perhaps do not account for all seasonality in the model, we include lags of a day, a week and a year.

4 Methods

In this paper, we model and predict short-term electric load by using LR, MARS, ANN and SVR. The dependent variable in the models is electric load, and the independent variables are the lagged load variables, the variables concerning weather data and the dummy variables described in Section 3. For all models we use a training data set that contains data from 1 May, 2013, up until and including 30 April, 2018, and a test data set with data from 1 May, 2018, up until and including 30 April, 2019. As the value of the one-year lagged load is not known up until and including 30 April, 2014, we effectively use 1 May, 2014, up until and including 30 April, 2018, as the training data set. For all models we construct 24 hours ahead forecasts for either the load or weather variables with the test data set as input data. As we use hourly data on load, this corresponds to constructing 24-steps ahead forecasts. We implement all models in Spyder: The Scientific Python Development Environment (2019) and use its integrated packages as well as the packages py-earth (2013) and TensorFlow (2015).

In Sections 4.1 through 4.4, the four load models are presented. In Section 4.5 we discuss the weather models. Section 4.6 describes how the different load forecasting models will be compared.

4.1 Linear regression

The LR model in this paper is a multiple linear regression model, and is represented by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{y}$ and $\boldsymbol{X}$ denote the dependent and independent variables, respectively, $\boldsymbol{\beta}$ denotes the vector of coefficients, and $\boldsymbol{\varepsilon}$ denotes the error term. As described by Heij, de Boer, Franses, Kloek, & van Dijk (2004b), there are seven assumptions we make whenever performing a linear regression. In this paper, we assume all but one of these assumptions hold true: instead of assuming fixed regressors in the LR model, we assume stable regressors. This adjusted assumption is made due to the presence of stochastic variables (the weather conditions) in the model. These stochastic variables are not fixed regressors by definition. If we assume they are stable, indicating that the stochastic variables should exhibit sufficient but not excessive variance, LR still results in the best linear unbiased estimators for the coefficients in the model. These coefficients are found by minimising the sum of squared residuals (SSR) of the model. The SSR is defined to be $S(\hat{\boldsymbol{\beta}}) = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})'(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})$, where $\hat{\boldsymbol{\beta}}$ is the vector of estimated coefficients.

A problem that may arise is the problem of multicollinearity. This can occur whenever several of the regressors are correlated enough that it influences the significance of the parameters. In order to account for this, we look at the variance inflation factor (VIF) of all regressors, and we iteratively delete the variable with the highest VIF from the regression until all variables have a VIF lower than a certain threshold. This threshold is often chosen to be five or ten (Craney & Surles, 2002).

The threshold used in this paper is ten.

As not all parameters in the resulting model may be significant, we perform backward deletion of the most insignificant independent variables until only the significant regressors remain. This is done, as using an overparameterized model can lead to worse forecasting performance, as stated by Engle & Brown (1986). Even though we initially assume all assumptions hold true, as mentioned above, we do check for the presence of heteroskedasticity and autocorrelation in the model by means of a Breusch-Pagan and Breusch-Godfrey test, respectively. In case these tests reject the null hypotheses of homoskedasticity and no autocorrelation, respectively, we use Newey-West heteroskedasticity and autocorrelation consistent standard errors to determine significance of the parameters.

4.2 Multivariate adaptive regression splines

The MARS model, first proposed by Friedman (1991), lets go of linearity, and can be seen as a sort of piecewise function. In this paper, we implement the MARS model as described by Nalcaci et al. (2019) and Al-Musaylh et al. (2018).

The MARS model assumes that the relationship between the dependent variable $\boldsymbol{y}$ and the matrix of independent variables $\boldsymbol{X}$ can be modeled by the function

$$y_t = \alpha_0 + \sum_{m=1}^{M} \alpha_m BF_m(\boldsymbol{x}_t, \tau_m) + \varepsilon_t \qquad \forall t \in \{1, \ldots, T\}, \tag{1}$$

where α_0 and α_m are estimated by minimising the SSR. The terms BF_m in Equation 1 are called basis functions (BFs) and $\tau_m \in \mathbb{R}$ is called the knot location and acts as a cut-point value. T is the number of observations in the training data set. The BFs can be linear or cubic functions. Linear BFs are of the form

$$BF(x, \tau) = \max\{0, x - \tau\}, \qquad \text{or} \qquad BF(x, \tau) = \max\{0, \tau - x\}. \tag{2}$$

Together, the two functions in Equation 2 form a reflected pair. Figure A.2 in the Appendix provides a visual example of a reflected pair of BFs. A cubic BF is simply a multiplication of two linear BFs. As Al-Musaylh et al. (2018) show that the cubic model results in a smoother function, we allow cubic BFs. In principle, the MARS algorithm keeps adding BFs to the model to increase the fit of the model until this increase no longer succeeds a certain threshold per iteration. This can lead to overfitting. In order to account for this, instead of finding the ultimate fit with the BFs, the MARS algorithm selects the model with the lowest value for the lack of fit. The lack of fit is analysed through generalised cross-validation (GCV), which is defined as follows:

$$\text{GCV} = \frac{\text{MSE}}{\left[1 - \frac{\tilde{G}(M)}{T}\right]^2},$$

where MSE stands for the mean squared error, and the denominator is some sort of penalty for the increasing complexity of the model: $\tilde{G}(M) = C + v \cdot M$, where C is simply the number of parameters being fit, M is the number of knots in the model, and v is a penalty factor, which is generally set to 3. Finally, T denotes the training sample size.

Even though highly correlated regressors pose less problems for the MARS algorithm than for LR, multicollinearity can still be an issue (Nalcaci et al., 2019). Therefore, we use the regressors for the MARS model that have a VIF of lower than ten, as explained in Section 4.1.

4.3 Artificial neural network

The third model examined in this paper is the ANN. The ANN is a machine learning model that can be used to recognise complex patterns in the data. We implement the ANN as described by Nalcaci et al. (2019). The ANN is a feed-forward back-propagation neural network, in which input data is given, after which the data passes through hidden layers and output data is realised. This is shown graphically in Figure A.3 in the Appendix.

An ANN consists of n input nodes, h hidden layer nodes and m output nodes. In this paper, we include one hidden layer in the model. The nodes in the input layer in a neural network shape the input data in such a way that can be passed along to the nodes in the hidden layer. Practically, this means that input data has to be standardised, which is called feature scaling. Standardising the data can be done through

$$\boldsymbol{x}_{\text{stand}} = \frac{\boldsymbol{x} - \mathbf{mean}(\boldsymbol{x})}{\text{standard\_deviation}(\boldsymbol{x})}. \tag{3}$$

After the input data is standardised, the data is passed along to the nodes in the hidden layer, according to weights assigned to the different inputs. In the hidden layer, some transformation of the data is done by a so-called activation function. We use the sigmoid activation function, as done by Nalcaci et al. (2019). The output of the hidden layer nodes is then

$$z_j = \frac{1}{1 + \exp\left(-\sum_{i=1}^{n} w_{ij} x_i\right)} \qquad \forall j \in \{1, \ldots, h\}, \tag{4}$$

where z_j denotes the output of the j^{th} node in the hidden layer, x_i denotes the output of the i^{th} input layer node, and w_{ij} is the relative weight assigned to the input data of the i^{th} input node to the j^{th} hidden layer node.

The output of the hidden layer nodes is then passed along in a similar way to the output layer nodes. Again, the weighted output of the hidden layer is transformed by an activation function, which results in an output. In the load forecasting model, the output is the explained, or predicted load. We therefore have $m = 1$ output layer nodes. As load cannot take negative values, the

activation function in the output layer is the rectified linear unit function, given by

$$y_j = \max\left\{0, \sum_{i=1}^{h} w_{ij} z_i\right\} \qquad \forall j \in \{1, ..., m\},$$

where y_j denotes the output of the j^{th} node in the output layer, z_i denotes the output of the i^{th} hidden layer node, and w_{ij} is the relative weight assigned to the output data of the i^{th} hidden layer node to the j^{th} output layer node.

To find the optimal weights in the ANN model, we use the back-propagation algorithm with a mini-batch approach (Rumelhart, Hinton, & Williams, 1986). More specifically, we update the weights using stochastic gradient descent, to minimise the MSE. The back-propagation with mini-batch approach is repeated for several iterations. If the number of iterations is too high, this can lead to overfitting: the ANN starts to model the training data too perfectly, which results in worse forecasting performance.

We therefore need to find the number of iterations for which the ANN models the training data well, but not too well. Furthermore, as there is no hard rule for the optimal number of nodes in the hidden layer of the model, we need to find a way to choose the amount of hidden nodes as well.

In order to do this, we take the following approach. First, the training data set is split in two subsamples: a training sample and a validation sample. The validation sample consists of a quarter of observations in the training data set, which corresponds to a year of data in our case. We then perform a grid search to find the number of nodes in the hidden layer. We use the findings of Wanas, Auda, Kamel, & Karray (1998) as a starting point of the grid search. They show that $\log(T)$ is a good starting point for the number of nodes in the neural network, where T is the number of observations in the training sample. This is equal to 26280 in our case, leading to $\log(T) \approx 5$. We consider the following options for the number of nodes: 5, 10, 15, 20, 25, 30. For each number of nodes, we fit the ANN to the training sample as long as the MSE of the validation sample has decreased in the last ten iterations. We repeat this ten times. In the rare case that the validation MSE may not decrease at all, we repeat the grid search for another iteration. The number of nodes in the hidden layer is then selected to be the option for which the mean validation MSE is minimised. This number of nodes is used to refit the ANN for the entire training data set, for the amount of iterations initially needed to fit the model to the training sample.

4.4 Support vector regression

The fourth and last model for load forecasting discussed in this paper is the SVR model. We follow the implementation of the SVR model as done by Al-Musaylh et al. (2018) and Che & Wang (2014).

SVR lets go of the notion that all residuals in a regression, however small they may be, are not allowed. Rather, by not penalizing small errors, the model is less sensitive to outliers in the data than, for example, LR. SVR models the relationship between the dependent and independent

variables by

$$y_t = f(\boldsymbol{x}_t) = \boldsymbol{\omega}'\varphi(\boldsymbol{x}_t) + b \qquad \forall t \in \{1, ..., T\},$$

where $\boldsymbol{x}_t$ is the vector of regressors for observation t, φ is some nonlinear mapping function, $\boldsymbol{\omega}$ is a vector of weights, and b is a constant. Finally, T denotes the number of observations in the training data set. It should be noted that with the SVR model we ought to standardise the input data as in Equation 3. The parameters $\boldsymbol{\omega}$ and b are estimated by the minimisation problem below:

$$\min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \frac{1}{2}\boldsymbol{\omega}'\boldsymbol{\omega} + C\sum_{t=1}^{T}(\xi_t + \xi_t^*) \tag{5}$$

$$\text{s.t.} \begin{cases} y_t - (\boldsymbol{\omega}'\varphi(\boldsymbol{x}_t) + b) \leq \varepsilon + \xi_t & \forall t \in \{1, \ldots, T\} \\ (\boldsymbol{\omega}'\varphi(\boldsymbol{x}_t) + b) - y_t \leq \varepsilon + \xi_t^* & \forall t \in \{1, \ldots, T\} \\ \xi_t,\ \xi_t^* \geq 0 & \forall t \in \{1, \ldots, T\} \end{cases} \tag{6}$$

Here, the scalar C, also called the cost parameter, denotes the trade-off between the emphasis on empirical risk or smoothness. ξ and ξ^* are so-called slack variables that denote the distance between actual load values and the ε-tube. ε denotes the maximum unpenalized error in the model.

We can solve the dual problem of the model given in Equations 5 and 6 and obtain the following:

$$f(\boldsymbol{x}) = \sum_{t=1}^{T}(\alpha_t - \alpha_t^*)K(\boldsymbol{x}, \boldsymbol{x}_t) + b,$$

where α_t and α_t^* are Lagrange multipliers obtained by solving the dual problem. Moreover, $K(\boldsymbol{x}, \boldsymbol{x}_t)$ is called a kernel function, which maps the training data into a higher dimensional space. The kernel we use in this paper is the radial basis function (Al-Musaylh et al., 2018):

$$K(\boldsymbol{x}, \boldsymbol{x}_t) = \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{x}_t\|^2}{2\sigma^2}\right),$$

where σ denotes the kernel width.

In order to fit the SVR model to the best of our ability, we perform a grid search on the values for C and $\frac{1}{2\sigma^2}$. For the parameter C we consider the values in the set $\{10^i \mid i \in \mathbb{Z},\, i \in [-4, 2]\}$. For $\frac{1}{2\sigma^2}$ we consider the values in the set $\{10^i \mid i \in \mathbb{Z},\, i \in [-4, 0]\}$. We again split the training data set in a training sample, which consists of the first three-quarters of the training data set, and a validation sample, consisting of the last quarter of the training data set. For each pair of options for parameter values in the grid, we fit the SVR model to the training sample. Then, we predict the load in the validation sample with the model and determine the value of the MSE for this sample. The pair of grid values for C and $\frac{1}{2\sigma^2}$ that result in the lowest validation MSE are then used to fit an SVR model to the entire training data set and to construct 24 hours ahead predictions for the load based on the test data set.

4.5 Constructing weather forecasts

While training the load forecasting models, measured weather data can be used as input. However, when the models are used for load forecasting, we need predictions for the weather variables as well. Three methods of obtaining these predictions are further described in this section.

4.5.1 Naive forecasting

First, as described by Hippert & Pedreira (2004), we can use a naive forecasting method, in which the forecast for the weather data at a certain hour h on day $d+1$ is equal to the measured values at hour h on day d. Mathematically, this is equal to $\hat{\boldsymbol{x}}_{d+1,h} = \boldsymbol{x}_{d,h}$.

4.5.2 Neural network for weather data

Another method for forecasting weather data is by using a neural network. As stated by Hippert & Pedreira (2004), the forecasts based on a neural network seem to be more accurate than the naive forecasts. In this neural network, for each of the four weather variables there are 24 input values. These inputs are the measured values of the weather variables during the past day. There are also 24 output values for each weather variable, denoting the forecasts for each hour the coming day. In this paper we include weather variables on wind speed, temperature, global solar radiation, and humidity. In total, we therefore have 96 input nodes in the neural network, as well as 96 output nodes.

We implement one hidden layer that consists of several nodes. In these nodes, we use the sigmoid activation function given in Equation 4. For the nodes in the output layer, we use a linear activation function, as some of the variables can take negative values. In this linear activation function, the output of the j^{th} node in the output layer is simply the weighted sum of the outputs of the h hidden layer nodes. The weights assigned to the neural network are updated using stochastic gradient descent with a mini-batch approach.

In order to determine the number of nodes in the hidden layer, we adopt the same cross-validation procedure as in Section 4.3. We split the training data set in a training and validation sample, and choose the number of hidden nodes based on the mean validation MSE.

4.5.3 Kalman filter

With every forecast, there is an accompanying forecasting error. Galanis & Anadranistakis (2002) state that there could be a systematic component in the errors of a forecast. They propose the use of a Kalman filter in order to adjust the predicted values to account for this systematic component. They show that using this filter results in more accurate forecasts.

A Kalman filter, as first proposed by Kalman (1960), is a manner of recursively obtaining estimates for some unobserved state vector, $\boldsymbol{x}_t$, based on the observed vector $\boldsymbol{y}_t$. In our case, the

unobserved state vector concerns the systematic part of the forecasting error, whereas the forecasting error as a whole is known (observed). The goal of applying the Kalman filter is to determine whether there is a systematic part to the forecasting errors. If so, this information can be used to update the forecasts in order to increase forecasting accuracy.

The Kalman filter is built upon the assumption that the process $\boldsymbol{x}$ changes from time $t-1$ to time t according to the system equation

$$\boldsymbol{x}_t = \boldsymbol{F}_t \cdot \boldsymbol{x}_{t-1} + \boldsymbol{w_t},$$

where the coefficient matrix $\boldsymbol{F}_t$, also called the system matrix, ought to be known before running the filter. We assume that the system matrix is an identity matrix, for lack of reason to think otherwise. Furthermore, $\boldsymbol{w}_t$ is assumed to be a Gaussian distributed random vector with zero mean, and is assumed to display no autocorrelation. The covariance matrix $\boldsymbol{W}_t$ of $\boldsymbol{w}_t$ also needs to be determined prior to running the filter.

The relationship between the measured variable $\boldsymbol{y}_t$ and the unobserved variable $\boldsymbol{x}_t$ is given by

$$\boldsymbol{y}_t = \boldsymbol{H}_t \cdot \boldsymbol{x}_t + \boldsymbol{v}_t,$$

where the coefficient matrix $\boldsymbol{H}_t$ is called the observation matrix. Again, this matrix should be known prior to running the filter. As with the system matrix, we assume that the observation matrix is an identity matrix, as there is no reason to think otherwise. Moreover, $\boldsymbol{v}_t$ is again a Gaussian distributed random vector with zero mean and no autocorrelation. Furthermore, $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ are assumed to be uncorrelated.

The original Kalman filter poses that based on the information available at time $t-1$ we can predict the following for the vector $\boldsymbol{x}_t$ and its corresponding covariance matrix $\boldsymbol{P}_t$:

$$\begin{aligned}
\boldsymbol{x}_{t|t-1} &= \boldsymbol{F}_t \cdot \boldsymbol{x}_{t-1}, \\
\boldsymbol{P}_{t|t-1} &= \boldsymbol{F}_t \cdot \boldsymbol{P}_{t-1} \cdot \boldsymbol{F}_t' + \boldsymbol{W}_t.
\end{aligned}$$

Together, these equations form the prediction equations. At time t we observe the value of $\boldsymbol{y}_t$, which we can then use to update the value of $\boldsymbol{x}_t$ by using the updating equations:

$$\begin{aligned}
\boldsymbol{K}_t &= \boldsymbol{P}_{t|t-1} \cdot \boldsymbol{H}_t' \cdot \left(\boldsymbol{H}_t \cdot \boldsymbol{P}_{t|t-1} \cdot \boldsymbol{H}_t' + \boldsymbol{V}_t\right)^{-1}, \\
\boldsymbol{x}_t &= \boldsymbol{x}_{t|t-1} + \boldsymbol{K}_t \cdot \left(\boldsymbol{y}_t - \boldsymbol{H}_t \cdot \boldsymbol{x}_{t|t-1}\right), \\
\boldsymbol{P}_t &= (\boldsymbol{I} - \boldsymbol{K}_t \cdot \boldsymbol{H}_t) \cdot \boldsymbol{P}_{t|t-1}.
\end{aligned}$$

In these equations, $\boldsymbol{K}_t$ denotes the Kalman gain. This parameter influences how easily the predictions of the unobserved state vector are adjusted. It should be noted, however, that the Kalman filter in this form adjusts one-step ahead forecasts. The weather forecasts are 24 hours ahead forecasts.

Therefore, an adjustment of the original Kalman filter is needed to account for this. We first run the prediction equations recursively 24 times in order to obtain estimates for $\boldsymbol{x}_{t|t-24}$ and $\boldsymbol{P}_{t|t-24}$. Then, in the updating equations, we use these estimates instead of $\boldsymbol{x}_{t|t-1}$ and $\boldsymbol{P}_{t|t-1}$.

In order to run the filter, we need initial estimates for the values $\boldsymbol{x}_0$ as well as $\boldsymbol{P}_0$. We initialise these to be a zero vector and identity matrix, respectively. Moreover, we need estimates for the covariance matrices $\boldsymbol{V}_t$ and $\boldsymbol{W}_t$. We initialise these matrices to be identity matrices. However, as we run the filter for more observations, we turn to a moving window approach after sufficient number of iterations. We base the estimates for the covariance matrices $\boldsymbol{V}_t$ and $\boldsymbol{W}_t$ on the sample covariance of the past 168 observations on the measurement error and predicted unobserved variables, respectively. This moving window of 168 observations corresponds to a moving window of a week.

4.5.4 Choosing the weather forecast

We construct three sets of forecasts of the weather variables, based on the naive forecasting method, the neural network, and the neural network with Kalman filter (henceforth called the ANN-KF forecasts). We compare the forecasts based on the root mean squared error (RMSE) and the mean absolute error (MAE) of the models over the test data set. The predictions from the forecast model with the highest predictive accuracy are then used as input in the load forecasting models.

4.6 Evaluating forecasting performance

After we have chosen which weather forecast to work with, we turn to 24 hours ahead forecasting of the load. In order to determine which of the four models, LR, MARS, ANN, and SVR, can be used best for load forecasting, we look at several performance measures for each model.

Common forecasting performance measures of the models are the RMSE, MAE, mean absolute percentage error (MAPE), and the R^2. We evaluate the values of these performance measures for all four models over the test data set.

In order to draw valid conclusions, we perform several tests. First, the efficiency of all four forecasts can be examined through performing a Mincer-Zarnowitz regression, as first proposed by Mincer & Zarnowitz (1969). Here, we regress actual load on a constant and the predicted load, resulting in the model

$$y_t = \beta_0 + \beta_1 \hat{y}_t + \varepsilon_t \qquad \forall t \in \{1, \dots, P\}.$$

Here, P denotes the number of predictions made. The forecast is efficient if the joint null hypothesis of $\beta_0 = 0$ and $\beta_1 = 1$ is not rejected.

Furthermore, we determine whether there is a significant difference in the mean squared prediction errors (MSPEs) based on the Diebold-Mariano test as first proposed by Diebold & Mariano (2002). Here, the forecasting error of some model A is defined to be

$$e_{A,t|t-24} = y_t - \hat{y}_{A,t|t-24} \qquad \forall t \in \{1, \dots, P\},$$

where y_t and $\hat{y}_{A,t|t-24}$ denote the real and predicted load respectively. Moreover, we define the loss-differential d_t to be

$$d_t = e^2_{A,t|t-24} - e^2_{B,t|t-24} \qquad \forall t \in \{1, \ldots, P\},$$

where $e_{A,t|t-24}$ and $e_{B,t|t-24}$ denote the forecast errors from models A and B respectively. We test the null hypothesis of equal MSPEs, or E(d_t)=0, against the alternative of unequal MSPEs. Here we use that the sample mean of the loss differential divided by its sample standard deviation is asymptotically standard normal distributed:

$$\text{DM} = \frac{\bar{d}}{\sqrt{\hat{\sigma}^2_{d_t}/P}} \sim N(0,1).$$

Based on the DM-statistic we can draw conclusions as to which load forecasting model produces significantly more accurate forecasts.

5 Results

We first describe the results on the best way to forecast the weather, as this information is necessary in order to evaluate the four load forecasting models. All tests in this section are conducted at a 5% significance level.

5.1 Weather forecast

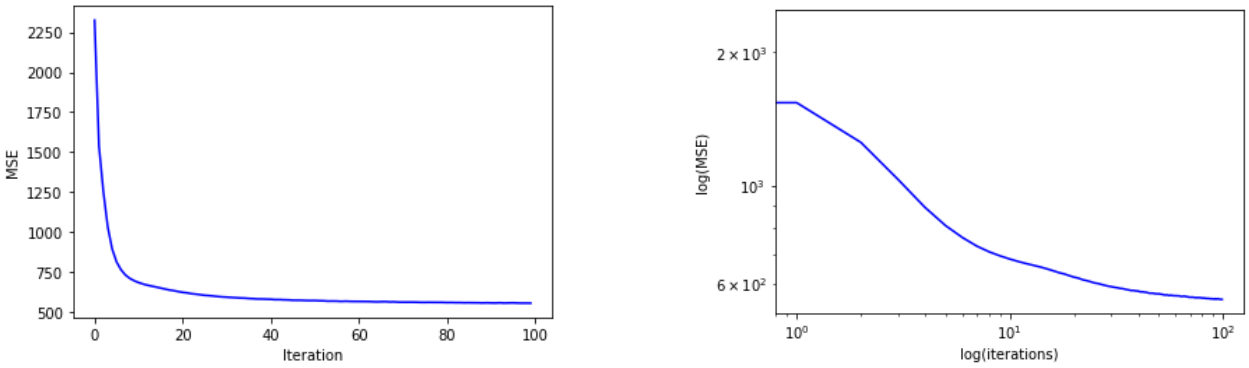
We implement the naive weather forecast as described in Section 4.5.1. Moreover, we implement the neural network as described in Section 4.5.2. For one grid search iteration over all candidate numbers of hidden nodes, the training and validation sample MSE's are shown in Appendix A.4. The cross-validation procedure results in the following:

Table 5.1: Performance of candidate weather ANNs over the validation set. For each amount of hidden nodes, the ANN is modeled ten times. As the ANN with 30 nodes in the hidden layer results in the lowest mean MSE over the validation set, this ANN is chosen as model for the weather data.

Number of hidden nodes	**Mean MSE validation set**	**Mean number iterations**
5	11.17*10^2	109.6
10	8.34*10^2	71.4
15	7.34*10^2	80.5
20	6.71*10^2	80.4
25	6.34*10^2	74.6
30	**6.08*10^2**	**73.6**

Table 5.1 shows that the model with 30 nodes in the hidden layer seems to perform best based on the validation MSE, so this is the model we will use for the weather data. The number of iterations

needed to obtain convergence lies around the order of magnitude of 100. Therefore, we choose to fit the neural network with 30 nodes in the hidden layer over 100 iterations. This results in the progress of the training sample MSE shown in Figure 5.1a. The training sample here is the entire training data set. Figure 5.1b shows the same graph as a log-log graph. We use the weights of the model in



(a) Training MSE

(b) Log training MSE

Figure 5.1: Training sample MSE during process of fitting the neural network with 30 hidden nodes to the entire training data set over 100 iterations. The graph in the log-log plot is not linear, indicating there is no power law relation between the number of iterations and the training MSE.

iteration 100 as the final weights of the neural network. With these weights, we construct 24 hours ahead forecasts of all four weather variables, representing the forecast for the upcoming day.

As described in Section 4.5.3, we use the Kalman filter to update the forecasts, resulting in the ANN-KF model for weather prediction. An overview of the forecasting performance measures of the three different weather models over the test data set is given in Table 5.2. The performance measures are given for each of the four weather variables separately.

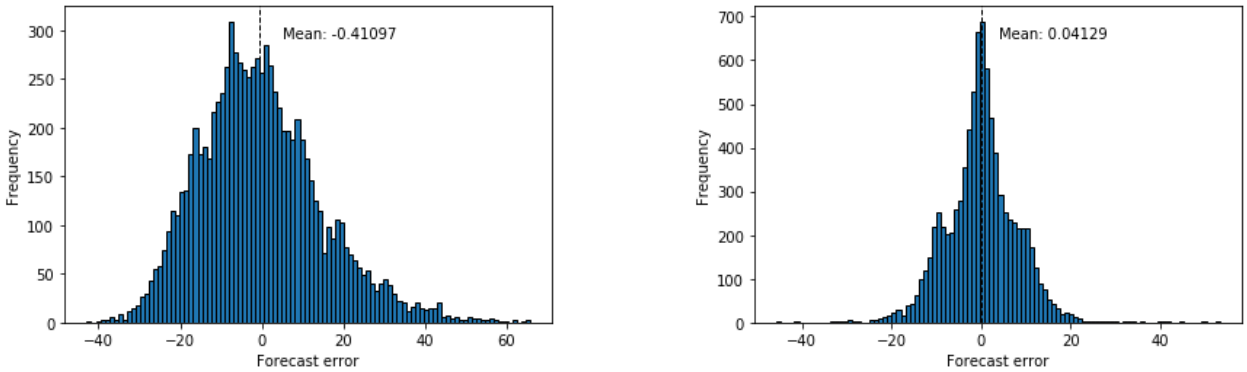
Table 5.2: Forecasting performance of the three weather models. The ANN-KF model outperforms both the naive and ANN model in every regard.

Variable	**Model**	**RMSE**	**MAE**
Wind	Naive	18.13	13.64
	ANN	13.66	10.68
	ANN-KF	8.09	6.12
Temperature	Naive	32.13	24.65
	ANN	28.13	21.85
	ANN-KF	15.03	11.40
Radiation	Naive	41.53	18.54
	ANN	37.47	24.41
	ANN-KF	24.75	16.35
Humidity	Naive	13.77	10.33
	ANN	12.01	9.31
	ANN-KF	6.43	4.71

Table 5.2 shows that both the RMSE and MAE are much lower for the ANN-KF model than for the

other models. Therefore, when forecasting electric load, we will work with the ANN-KF predictions for the weather variables.

To illustrate the superior forecasting performance of the ANN-KF model, Figure 5.2 shows the 24-step ahead forecasting errors for the wind speed forecasts. Figure 5.2a shows a histogram of the forecast errors with the ANN model, whereas Figure 5.2b shows that the mean value of these forecast errors shrinks towards zero when adjusting the forecasts with the Kalman filter. The figures show that the Kalman filter indeed seems to filter out the systematic component to the forecast errors.



(a) Forecast errors with ANN

(b) Forecast errors with ANN-KF

Figure 5.2: The forecast errors for 24 hours ahead wind forecasts. It seems as though the forecasts produced by the ANN are biased, and follow a skewed distribution.

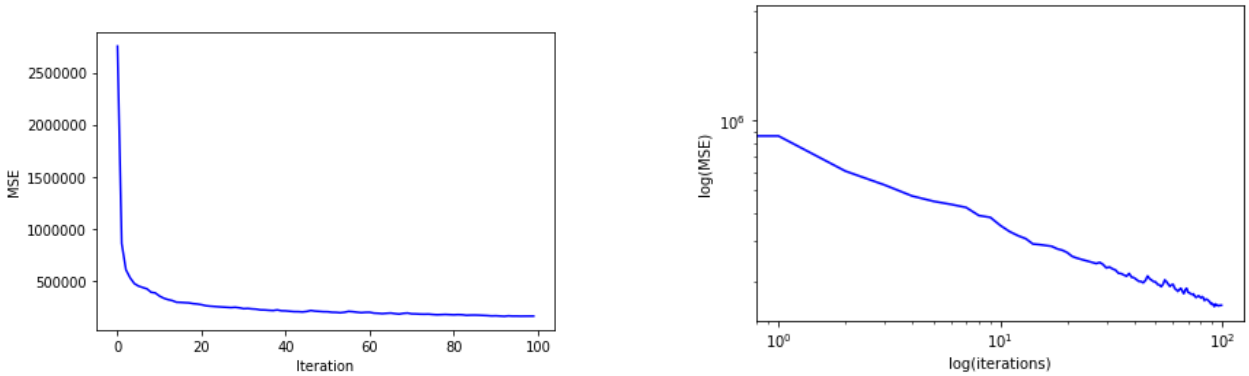
5.2 Load forecasting models

We implement the LR and MARS models as described in Sections 4.1 and 4.2, respectively. The output function for both models is given in the Appendix, Section A.5. As the residuals obtained with the LR model seem to exhibit both autocorrelation and heteroskedasticity, we use the Newey-West heteroskedasticity and autocorrelation consistent standard errors as standard errors of the estimated parameters. Moreover, it should be noted that the MARS model does not contain weather variables: these variables were pruned.

In order to predict load with the ANN model, we first need to find a well-performing neural network. We perform the grid search described in Section 4.3. The MSEs of the training and validation samples during one iteration of the grid search are shown in Appendix A.6. Table 5.3 shows that the ANN with 30 nodes in the hidden layer results in the lowest validation MSE. Therefore, in order to forecast electric load, we refit the ANN with 30 nodes for 100 iterations over the entire training data set. Figure 5.3a shows the behaviour of the MSE during training. Figure 5.3b shows the same plot on a log-log scale. As with the neural network for the weather variables, here, too, the weights of the model in iteration 100 are used as the final weights of the neural network. These weights are used to construct 24 hours ahead forecasts of the load.

Table 5.3: Performance of candidate load ANNs over the validation set. For each amount of hidden nodes, the ANN is modeled ten times. As the ANN with 30 nodes in the hidden layer results in the lowest mean MSE over the validation set, this ANN is chosen as model for the load data.

Number of nodes	**Mean MSE validation set**	**Mean number iterations**
5	$7.53 * 10^5$	36.9
10	$4.46 * 10^5$	77.4
15	$3.88 * 10^5$	62.2
20	$3.50 * 10^5$	59.0
25	$2.96 * 10^5$	72.6
30	$\mathbf{2.76 * 10^5}$	**70.7**



(a) Training MSE

(b) Log training MSE

Figure 5.3: Training sample MSE during process of fitting the ANN with 30 hidden nodes to the entire training data set. The graph in the log-log plot is nearly linear, indicating there could be a power law relation between the number of iterations and the value of the training MSE.

As mentioned in Section 4.4, we perform a grid search to find the best value for the cost parameter C and the kernel width parameter $\frac{1}{2\sigma^2}$ in the SVR model. It follows from this cross-validation procedure that $C = 100$ and $\frac{1}{2\sigma^2} = 0.001$ result in the lowest validation MSE. We therefore refit the SVR model for the entire training data set with the parameters set to those values.

5.3 Evaluating load forecasting models

Table 5.4 shows the 24 hours ahead forecasting performance of the four models. One set of forecasts is constructed with the true weather data. In other words, this set of load forecasts is made using perfect weather forecasts. The other set of forecasts is based on the ANN-KF weather forecasts.

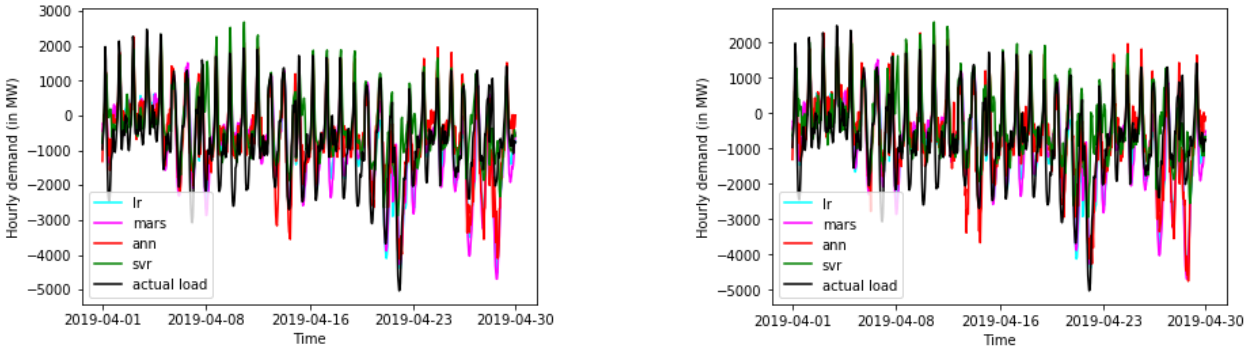
For both sets of forecasts, Table 5.4 shows that the SVR model performs best out of the four load forecasting models, except with regard to the MAPE. The MAPE is quite high, especially compared to the other three models. This indicates that the SVR model is prone to relatively large forecast errors for smaller values of the true load. Note that as the MARS model does not take the weather variables into account, this model shows the same forecasting performance, regardless of which forecasts are used. This could explain why the MARS model performs worse than the LR

Table 5.4: 24 hours ahead forecasting performance of the four load forecasting models. The SVR model performs best in every regard, except for the MAPE.

	Perfect weather forecast				**ANN-KF weather forecast**			
	LR	MARS	ANN	SVR	LR	MARS	ANN	SVR
RMSE	741.76	803.66	693.89	**628.53**	765.18	803.66	741.84	**653.09**
MAE	527.44	574.69	500.24	**455.22**	546.96	574.69	539.05	**477.47**
MAPE	0.040	0.044	**0.038**	0.175	0.042	0.044	**0.041**	0.175
$\boldsymbol{R^2}$	0.856	0.831	0.874	**0.897**	0.847	0.831	0.856	**0.888**

model, even though theoretically the MARS model should perform at least as well: MARS lets go of the linearity assumption that linear regression is based on.

Figure 5.4 shows the deseasonalised 24 hours ahead load forecasts as well as the true load for the last month in the test data set. The figure shows that even in the deseasonalised series, some



(a) Perfect weather

(b) ANN-KF weather

Figure 5.4: Deseasonalised 24 hours ahead forecasts for the last month in the test data set. Some seasonality still remains in the forecasts. It seems as though the MARS and SVR models produce too low and high forecasts, respectively. The ANN forecasts are too extreme. None of the models capture the effect of King's Day (April 27$^{\text{th}}$) correctly.

seasonality still remains. Moreover, MARS and SVR seem to produce too low and high forecasts respectively, whereas the ANN forecasts are too extreme in general.

In short, the SVR model seems to produce the most accurate forecasts. Then, the ANN, LR, and MARS models, respectively, deliver the best results overall. We perform the Diebold-Mariano test and Mincer-Zarnowitz regression as described in Section 4.6 in order to draw conclusions with regard to whether there is a significant difference in the performance of the models.

The results of the Mincer-Zarnowitz regression are shown in Table 5.5. The p-values of the constants and coefficients are given between parentheses. For the constants and coefficients, the p-value denotes the (two-sided) probability of the parameters being equal to zero and one, respectively.

It follows from the table that none of the forecasts are efficient. All forecasts seem to be structurally too low, indicating a systematic bias in the forecasts. However, the R^2 of all forecasts is quite high, suggesting that the forecasts, while inefficient, do explain a large part of the actual load.

Table 5.5: Results of the Mincer-Zarnowitz regression. The p-values mentioned are with respect to 0 and 1 for the constants and coefficients, respectively. All forecasts are not efficient and biased.

		Constant	**Coefficient**	$\boldsymbol{R^2}$
Perfect weather	**LR**	1068.55 (0.000)	0.921 (0.000)	0.863
	MARS	894.29 (0.000)	0.930 (0.000)	0.836
	ANN	1025.82 (0.000)	0.922 (0.000)	0.880
	SVR	895.70 (0.000)	0.928 (0.000)	0.903
ANN-KF weather	**LR**	1159.71 (0.000)	0.914 (0.000)	0.854
	MARS	894.29 (0.000)	0.930 (0.000)	0.836
	ANN	1224.27 (0.000)	0.907 (0.000)	0.865
	SVR	943.46 (0.000)	0.925 (0.000)	0.895

The results of the Diebold-Mariano test can be found in Table 5.6. A negative Diebold-Mariano (DM) statistic indicates that the first model mentioned in the 'Test' column results in more accurate forecasts than the second model. The p-values in the table are one-sided.

Table 5.6: Results of the Diebold-Mariano test. A negative DM-statistic indicates the first mentioned model in the 'Test' column is more accurate than the second model. Regardless of which weather predictions are used, the SVR forecasts are significantly more accurate compared to the other models.

	Perfect weather		**ANN-KF weather**	
Test	**DM-statistic**	p**-value**	**DM-statistic**	p**-value**
LR - MARS	-5.712	0.000	-3.494	0.000
LR - ANN	1.825	0.034	0.943	0.173
LR - SVR	3.066	0.001	3.114	0.001
MARS - ANN	3.829	0.000	2.251	0.012
MARS - SVR	4.373	0.000	3.797	0.000
ANN - SVR	2.202	0.014	3.072	0.001

From the above table follows that, regardless of which weather forecast is used, the SVR model significantly outperforms all other load forecasting models in terms of predictive accuracy on a 5% significance level. Moreover, the LR and ANN models do not significantly differ in accuracy when using ANN-KF weather forecasts. However, if we use the perfect weather forecasts, the ANN model is significantly more accurate than the LR model. Both models outperform the MARS model.

In short, none of the four load forecasting models produces efficient forecasts. However, in terms of predictive accuracy, the SVR model outperforms all other models, regardless of whether we use the perfect or predicted weather data.

6 Conclusion

The objective of this paper was to answer the following: 'How can we best use weather data when forecasting short-term electric load, and given this information, which method out of LR, MARS, ANN, and SVR, results in the highest predictive accuracy when forecasting the short-term electric load?' In order to answer this question we first determined how we can best obtain weather forecasts.

For this, there were three different approaches. One approach is a naive forecasting method, in which the forecast for the upcoming 24 hours is equal to the measured weather variables in the past 24 hours. The second approach is to use a neural network to make predictions for the coming day based on the past day. As a third approach, we constructed an adjusted Kalman filter to update the forecasts obtained with the neural network, leading to the so-called ANN-KF weather forecast.

From the results follows that for all weather variables taken into account in this paper, the ANN-KF method produces the most accurate weather forecasts. Thus, we first construct 24 hours ahead forecasts for the weather variables using an artificial neural network. Then, we adjust these forecasts using an adjusted Kalman filter. The superior accuracy of the ANN-KF forecasts can be due to the fact that the Kalman filter establishes the systematic part in the forecast errors correctly. This approximation is then used to adjust the original predictions of the weather variables.

Next, we implemented the four load forecasting models LR, MARS, ANN, and, SVR. As mentioned above, the ANN-KF method resulted in the most accurate weather forecasts. These forecasts were therefore used as input variables in the load forecasting models when forecasting. Regardless of whether we use 'perfect' weather forecasts or the ones obtained with the ANN-KF method, the SVR model always outperforms the other three load forecasting models in terms of predictive accuracy. The use of imperfect weather forecasts does decrease the predictive accuracy somewhat, but this decrease is negligible.

In short, when weather forecasts are not readily available, the ANN-KF method provides a way to obtain quite accurate weather forecasts. In terms of load forecasting models, we found that the SVR model produces the most accurate load forecasts.

The results of this research are of interest to energy utility companies. As mentioned in Section 1, it is important to them to have an indication of the electric load over the next 24 hours, as this helps make power grid decisions: generating too much or too little electricity can be costly.

For future research it could be interesting to further develop the SVR model, for example by tuning the hyperparameters in the model. There is already some literature on this, suggesting for example the use of a firefly or genetic algorithm, but this could be further looked into. The effect of the choice of kernel could be further investigated as well.

Moreover, the ANNs proposed in this model are relatively simple. For both modeling load and weather conditions, it could be interesting to evaluate the performance of a deep neural network consisting of multiple hidden layers, or of a recurrent neural network: such a neural network would be better suited for modeling time series.

Finally, costs of producing too little or too much electricity may differ. It could be of interest to energy utilities to investigate whether certain methods of load forecasting structurally result in too high or too low forecasts. This could influence the type of model that is most useful for the utility: perhaps the SVR model is less beneficial in this sense than other models.

References

Al-Musaylh, M. S., Deo, R. C., Adamowski, J. F., & Li, Y. (2018). Short-term electricity demand forecasting with MARS, SVR and ARIMA models using aggregated demand data in Queensland, Australia. *Advanced Engineering Informatics*, *35*, 1–16. doi: 10.1016/j.aei.2017.11.002

Beccali, M., Cellura, M., Brano, V. L., & Marvuglia, A. (2004). Forecasting daily urban electric load profiles using artificial neural networks. *Energy Conversion and Management*, *45*(18-19), 2879–2900. doi: 10.1016/j.enconman.2004.01.006

Che, J., & Wang, J. (2014). Short-term load forecasting using a kernel-based support vector regression combination model. *Applied Energy*, *132*, 602–609. doi: 10.1016/j.apenergy.2014.07.064

Che, J., Wang, J., & Tang, Y. (2012). Optimal training subset in a support vector regression electric load forecasting model. *Applied Soft Computing*, *12*(5), 1523–1531. doi: 10.1016/j.asoc.2011.12.017

Craney, T. A., & Surles, J. G. (2002). Model-dependent variance inflation factor cutoff values. *Quality Engineering*, *14*(3), 391–403. doi: 10.1081/QEN-120001878

Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, *20*(1), 134–144. doi: 10.1198/073500102753410444

Engle, R. F., & Brown, S. J. (1986). Model selection for forecasting. *Applied Mathematics and Computation*, *20*(3-4), 313–327. doi: 10.1016/0096-3003(86)90009-3

ENTSO-E. (2020). *Power statistics.* Retrieved 2020-05-11, from `https://www.entsoe.eu/data/power-stats/`

Erişen, E., Iyigun, C., & Tanrısever, F. (2017). Short-term electricity load forecasting with special days: an analysis on parametric and non-parametric methods. *Annals of Operations Research*, 1–34. doi: 10.1007/s10479-017-2726-6

Fard, A. K., & Akbari-Zadeh, M.-R. (2014). A hybrid method based on wavelet, ANN and ARIMA model for short-term load forecasting. *Journal of Experimental & Theoretical Artificial Intelligence*, *26*(2), 167–182. doi: 10.1080/0952813X.2013.813976

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, *19*(1), 1–67. doi: 10.1214/aos/1176347963

Galanis, G., & Anadranistakis, M. (2002). A one-dimensional Kalman filter for the correction of near surface temperature forecasts. *Meteorological Applications*, *9*(4), 437–441. doi: 10.1017/S1350482702004061

Heij, C., de Boer, P., Franses, P. H., Kloek, T., & van Dijk, H. K. (2004a). Econometric Methods with Applications in Business and Economics. In (p. 599-600). New York: Oxford University Press.

Heij, C., de Boer, P., Franses, P. H., Kloek, T., & van Dijk, H. K. (2004b). Econometric Methods with Applications in Business and Economics. In (p. 92-93). New York: Oxford University Press.

Hippert, H., & Pedreira, C. (2004). Estimating temperature profiles for short-term load forecasting: neural networks compared to linear models. *IEE Proceedings-Generation, Transmission and Distribution*, *151*(4), 543–547. doi: 10.1049/ip-gtd:20040491

Hong, T., Gui, M., Baran, M. E., & Willis, H. L. (2010). Modeling and forecasting hourly electric load by multiple linear regression with interactions. In *IEEE PES General Meeting* (pp. 1–8). IEEE. doi: 10.1109/PES.2010.5589959

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45. doi: 10.1115/1.3662552

KNMI. (2020). *Daggegevens van het weer in Nederland - Download.* Retrieved 2020-05-11, from `http://projects.knmi.nl/klimatologie/daggegevens/selectie.cgi`

McSharry, P. E., Bouwman, S., & Bloemhof, G. (2005). Probabilistic forecasts of the magnitude and timing of peak electricity demand. *IEEE Transactions on Power Systems*, *20*(2), 1166–1172. doi: 10.1109/TPWRS.2005.846071

Mincer, J. A., & Zarnowitz, V. (1969). The Evaluation of Economic Forecasts. In *Economic forecasts and expectations: Analysis of forecasting behavior and performance* (p. 3-46). NBER. Retrieved from `http://www.nber.org/chapters/c1214`

Nalcaci, G., Özmen, A., & Weber, G. W. (2019). Long-term load forecasting: models based on MARS, ANN and LR methods. *Central European Journal of Operations Research*, *27*(4), 1033–1049. doi: 10.1007/s10100-018-0531-1

Park, D. C., El-Sharkawi, M., Marks, R., Atlas, L., & Damborg, M. (1991). Electric load forecasting using an artificial neural network. *IEEE transactions on Power Systems*, *6*(2), 442–449. doi: 10.1109/59.76685

py-earth. (2013). Python Software Foundation. Retrieved from `https://github.com/scikit-learn-contrib/py-earth` (Version: 0.1.0)

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi: 10.1038/323533a0

Spyder: The Scientific Python Development Environment. (2019). Python Software Foundation. Retrieved from `https://www.anaconda.com/products/individual` (Version: 4.0.1)

Taylor, J. W. (2010). Triple seasonal methods for short-term electricity demand forecasting. *European Journal of Operational Research*, *204*(1), 139–152. doi: 10.1016/j.ejor.2009.10.003

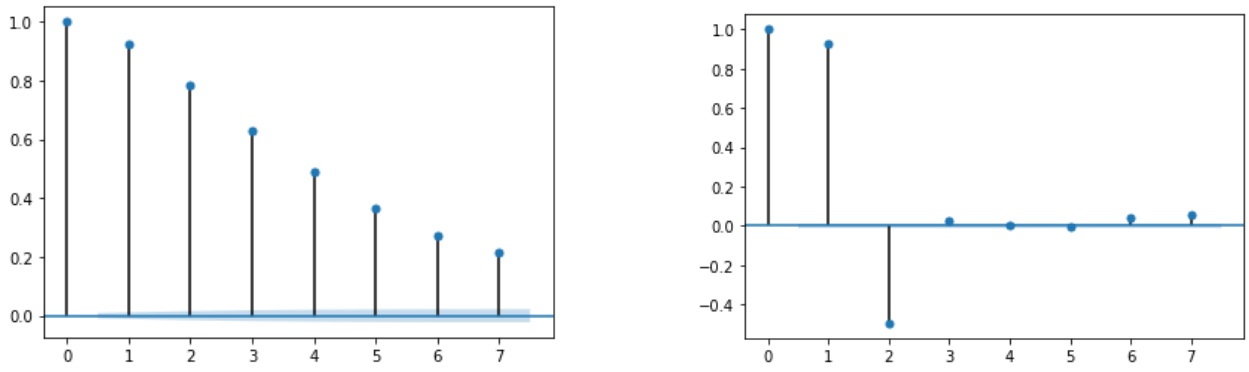
TensorFlow. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from `https://tensorflow.org` (Version: 2.0.0)

Wanas, N., Auda, G., Kamel, M. S., & Karray, F. (1998). On the optimal number of hidden nodes in a neural network. *Conference Proceedings. IEEE Canadian Conference on Electrical and Computer Engineering (Cat. No. 98TH8341)*, *2*, 918–921. Retrieved from `http://watsup.uwaterloo.ca/~nwanas/publications/Optimal_nodes_CCECE98.pdf`

Yao, S., Song, Y., Zhang, L., & Cheng, X. (2000). Wavelet transform and neural networks for short-term electrical load forecasting. *Energy Conversion and Management*, *41*(18), 1975–1988. doi: 10.1016/S0196-8904(00)00035-2

A Appendix

A.1 ACF and PACF



(a) ACF (b) PACF

Figure A.1: (Partial) autocorrelation functions. The PACF indicates that the current load is significantly correlated with the first two lags.

A.2 Graphical example of MARS

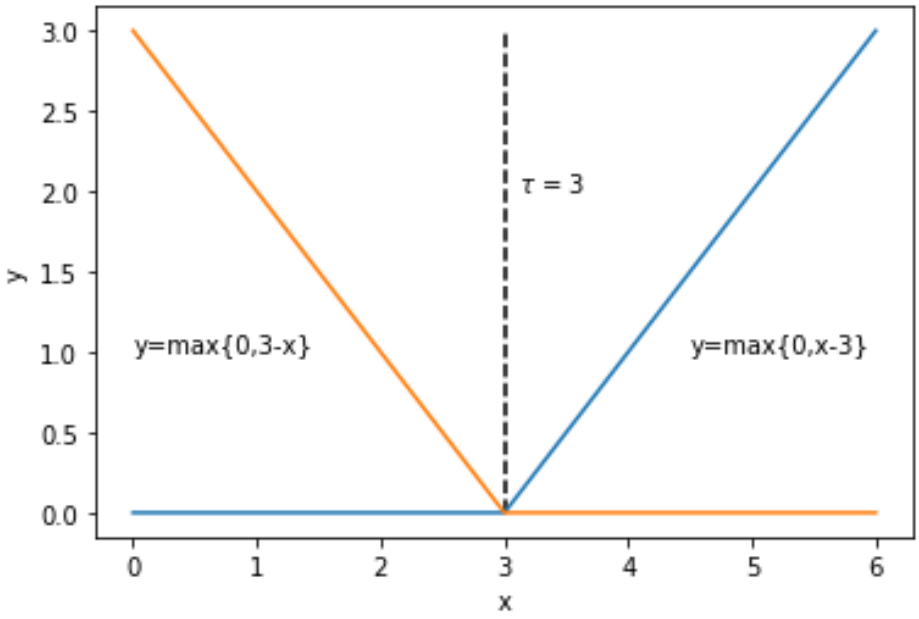


Figure A.2: Graphical representation of the MARS model. In this example, the two linear BFs form a reflected pair. The cutoff value, τ is equal to three. The orange BF has the functional form $y = \max\{0, 3 - x\}$. The blue BF has the functional form $y = \max\{0, x - 3\}$.

A.3 Graphical representation ANN

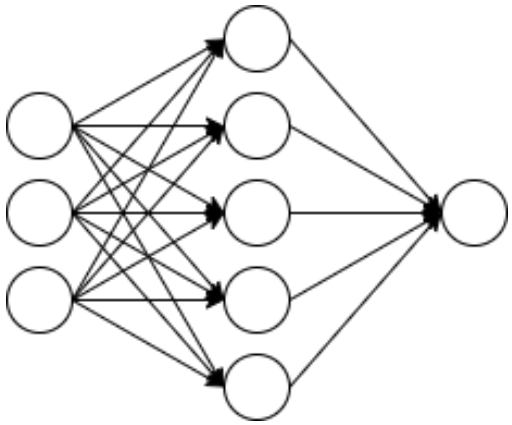
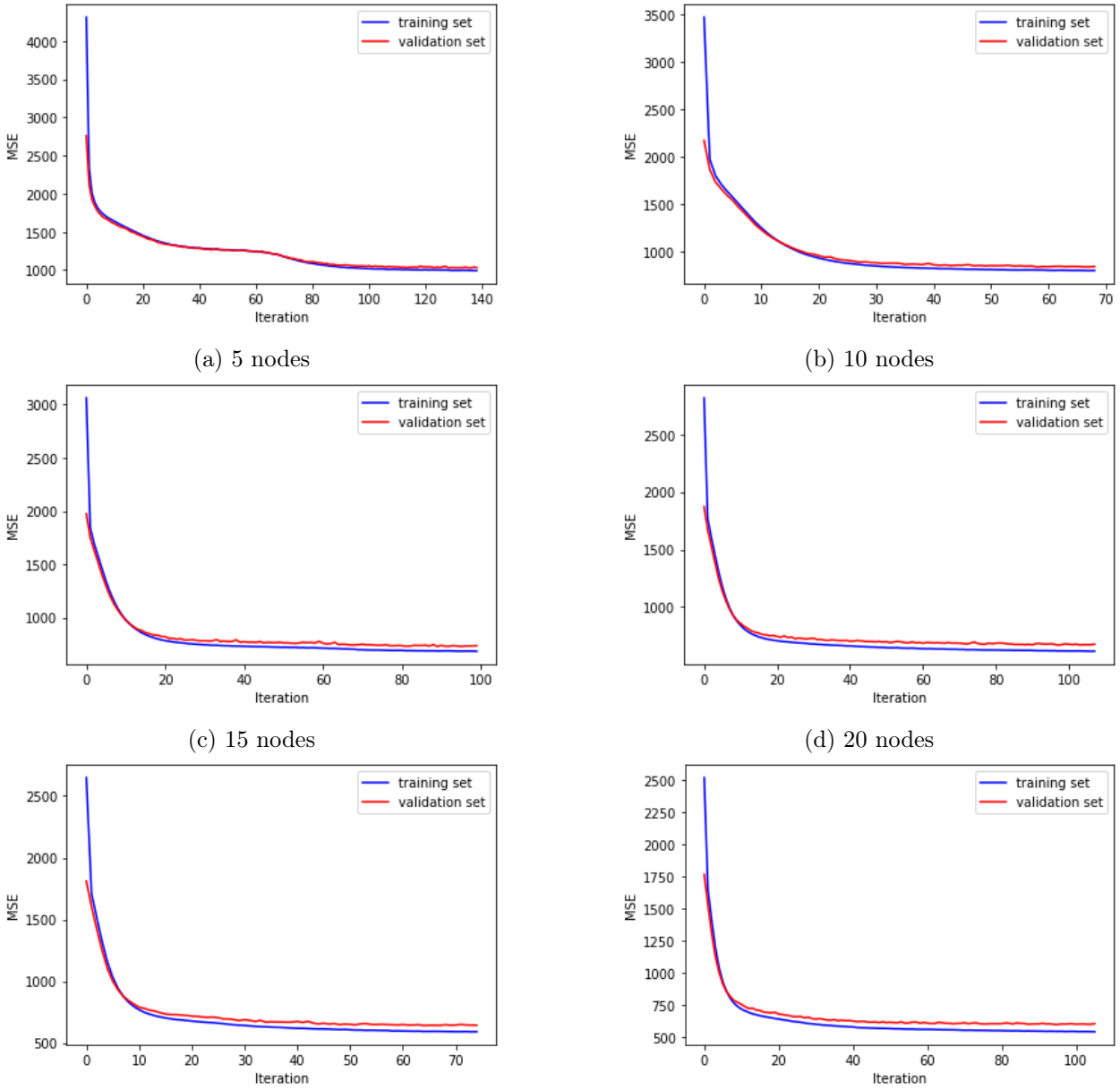


Figure A.3: Graphical representation of an ANN with one hidden layer. Input data on the left hand side are passed along to the hidden layer (shown as the middle layer in this particular structure). The output from the hidden layer is then passed along to the output layer on the right hand side.

A.4 Grid search iteration for weather neural network



(a) 5 nodes

(b) 10 nodes

(c) 15 nodes

(d) 20 nodes

(e) 25 nodes

(f) 30 nodes

Figure A.4: MSE during training of neural network for one iteration of the grid search for the number of hidden nodes in the weather neural network

A.5 Output LR and MARS models

The output of the LR model is given below. Note that the variables Hrx_t are dummy variables which take a value of one if the at time t the hour is (x-1):00 - x:00. For example, the dummy variable $Hr2$ is equal to one if at time t we model the load between 01:00 and 02:00, and zero

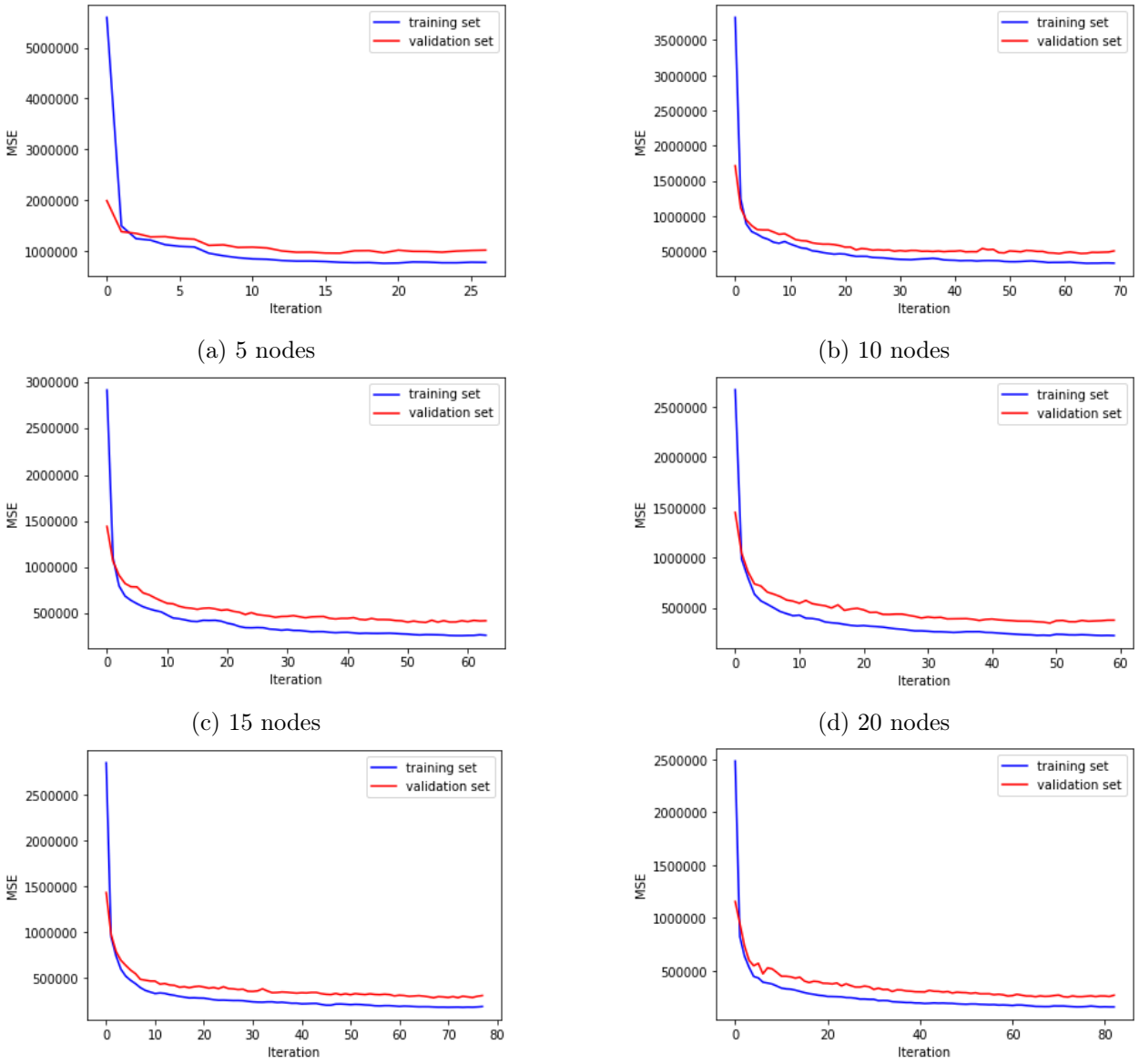
otherwise.

$$
\begin{aligned}
Load_t = &- 104.18 + 129.75 * Hr2_t + 294.61 * Hr3_t + 451.27 * Hr4_t + 679.04 * Hr5_t \\
&+ 1052.16 * Hr6_t + 1445.73 * Hr7_t + 1645.65 * Hr8_t + 1541.42 * Hr9_t + 1287.58 * Hr10_t \\
&+ 1075.56 * Hr11_t + 967.55 * Hr12_t + 909.22 * Hr13_t + 874.98 * Hr14_t + 870.88 * Hr15_t \\
&+ 900.80 * Hr16_t + 905.16 * Hr17_t + 834.51 * Hr18_t + 638.12 * Hr19_t + 422.40 * Hr20_t \\
&+ 263.88 * Hr21_t + 100.52 * Hr22_t - 38.88 * Hr24_t \\
&- 292.15 * Saturday_t - 183.51 * Sunday_t - 717.07 * Holiday_t - 66.05 * Winter_t \\
&- 1.83 * Radiation_t + 1.89 * Humidity_t \\
&+ 0.43 * Load_{t-2} + 0.13 * Load_{t-24} + 0.42 * Load_{t-168} - 0.03 * Load_{t-8760} + e_t
\end{aligned}
$$

The output for the MARS model is given below. Note that all BFs are linear, even though we allowed for cubic BFs. Moreover, none of the weather variables are incorporated in the model.

$$
\begin{aligned}
Load_t = &14097.86 + 0.255 * \max\{0, Load_{t-168} - 17415\} - 0.513 * \max\{0, 17415 - Load_{t-168}\} \\
&+ 0.378 * \max\{0, Load_{t-2} - 8457\} - 1.912 * \max\{0, 8457 - Load_{t-2}\} \\
&+ 0.797 * \max\{0, Load_{t-24} - 17415\} - 0.146 * \max\{0, 17415 - Load_{t-24}\} \\
&- 766.772 * Holiday_t - 236.121 * Saturday_t + 910.38 * Hr8_t \\
&+ 863.557 * Hr7_t + 714.171 * Hr9_t + 633.696 * Hr6_t + 423.858 * Hr10_t + 379.876 * Hr5_t \\
&- 339.302 * Hr23_t + e_t
\end{aligned}
$$

A.6 Grid search iteration for load forecasting ANN



(a) 5 nodes

(b) 10 nodes

(c) 15 nodes

(d) 20 nodes

(e) 25 nodes

(f) 30 nodes

Figure A.5: MSE during training of neural network for one iteration of the grid search for the number of hidden nodes in the load forecasting ANN

A.7 Readme for the code files

In this subsection a short description is given of all codes used in this paper.

- **Data preprocessing.py** is meant for data preprocessing. Only the file 'Data good.csv' is necessary to run it. In this script:

  - the sample is split in a training and test sample
  - the load series is visualized
  - the ADF-test is performed on all time series variables
  - the ACF and PACF are constructed for the deseasonalized load series
  - the necessary lags are constructed and added to the dataset

  The new dataset (including lags) is written to a csv file: 'Data with lags.csv'

- **Naive weather.py** returns naive weather forecasts and evaluates the performance. Only the file 'Data with lags.csv' is necessary to run it.

- **Weather NN 24KF.py** constructs the neural network weather forecasts and adjusts these with a Kalman filter dependent on whether or not that is preferred by the user. Dependent on what the indices are set to by the user, this script:

  - performs a grid search for the number of nodes to be used in the neural network (if indicator is set to 1)
  - fits the neural network with a certain number of nodes to the training data (if indicator is set to 2)
  - constructs 24-hours ahead forecasts (if indicator is set to 3) and:
    * uses those forecasts as output (if indicator_kf is set to 0)
    * adjusts the forecasts with a Kalman filter (if indicator_kf is set to 1)

  Only the file 'Data with lags.csv' is necessary to run the script.

- **LR.py** fits a linear regression model and constructs 24 steps ahead load forecasts. Based on the user's preferences:

  - when indicator_perfect is set to 0, forecasts are constructed with the ANN-KF weather forecasts.
  - when indicator_perfect is set to 1, forecasts are constructed using the perfect weather forecasts.

  The files 'Data with lags.csv' and 'Voorspellingen weernn-kf.csv' are necessary to run the script.

- **MARS.py** fits a multivariate adaptive regression splines (MARS) model and constructs 24 steps ahead load forecasts. Based on the user's preferences:

  - when indicator_perfect is set to 0, forecasts are constructed with the ANN-KF weather forecasts.

  - when indicator_perfect is set to 1, forecasts are constructed using the perfect weather forecasts.

  The files 'Data with lags.csv' and 'Voorspellingen weernn-kf.csv' are necessary to run the script.

- **ANN.py** constructs the artificial neural network (ANN) load forecasts with either perfect or ANN-KF weather forecasts. Dependent on what the indices are set to by the user, this script:
  - performs a grid search for the number of nodes to be used in the neural network (if indicator is set to 1)
  - fits the neural network with a certain number of nodes to the training data (if indicator is set to 2)
  - constructs 24-hours ahead load forecasts (if indicator is set to 3) and:
    * uses ANN-KF weather forecasts (if indicator_perfect is set to 0)
    * uses perfect weather data (if indicator_perfect is set to 1)

  The files 'Data with lags.csv' and 'Voorspellingen weernn-kf.csv' are necessary to run the script.

- **SVR.py** fits a support vector regression (SVR) model and constructs 24 steps ahead load forecasts. Based on the user's preferences:
  - when indicator_perfect is set to 0, forecasts are constructed with the ANN-KF weather forecasts.
  - when indicator_perfect is set to 1, forecasts are constructed using the perfect weather forecasts.

  The files 'Data with lags.csv' and 'Voorspellingen weernn-kf.csv' are necessary to run the script.

- **Significance tests.py** performs the Diebold-Mariano test of predictive accuracy and the Mincer-Zarnowitz regression to evaluate the performances of the separate forecasts. For each of the four load forecast models, a csv file with the load forecast constructed with both perfect and ANN-KF weather predictions is needed. Moreover, the file 'Data with lags.csv' is needed to run the script. The output of this script is the results of both the Diebold-Mariano test and the Mincer-Zarnowitz regression.

- **Plot forecasts.py** makes plots of the forecasts. For each of the four load forecast models, a csv file with the load forecast constructed with both perfect and ANN-KF weather predictions is needed. Moreover, the file 'Data with lags.csv' is needed to run the script.