Applying Machine Learning Techniques to Short Term Load Forecasting

Ciarán Lier
1535951
December 2015

Master Thesis
Artificial Intelligence
University of Groningen, The Netherlands

Internal Supervisor:
dr. Marco Wiering (University of Groningen)

External Supervisor:
dr. Han Suelmann (CGI Nederland B.V.)

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Abstract

This thesis reports on the application of two machine learning techniques on the case of 24-ahead short term load forecasting (STLF). The methods used are Random Forests and Echo State Networks. Hierarchical linear models are used as baseline comparison. Four different cases of STLF will be combined in this research: Total power consumption of an area, power demand on the power supplier, power supply to the power network, and solar power generation (SPG).

These variables are useful things to know in power supply planning by power suppliers and short term peak detection for network operators. To know these variables beforehand means to be able to economically and securely operate the power grid and power supply. Therefore constant research is being done to improve forecasting techniques. More recently it has become important to incorporate the supply by users into the forecasting system as more and more households install solar panels.

A dataset was used from a neighbourhood in The Netherlands where most households are outfitted with solar panels and all households have smart meters. A large part of the project consisted of cleaning the data. Predictors were chosen from the dataset using domain knowledge and partly by Fourier analysis. Some measurements of weather data were added to the dataset using an interpolation between two stations of the KNMI. Four datasets were created; one for each case. These were split up for training, validation, and testing purposes.

Random Forests and Echo State Networks use a number of hyper-parameters as initiation or training settings. These parameters were optimized on the training and validation sets using particle swarm optimization (PSO). The resulting optimal settings were used to train new models and test performance on the test sets. Comparison was done by testing the differences in RMSE with Welch's t-test.

The results are interesting. It was found that the linear model is quite a good performer in most cases, but is sometimes outperformed by the Random Forest. Solar power generation has appeared to be the hardest to predict and even the linear model is not performing well in this case. The Echo State Network seems to be unsuitable for this kind of forecasting in all cases.

Acknowledgements

This thesis is the result of my graduation research project at the University of Groningen for the department of Artificial Intelligence. The project was done as in graduate internship at CGI Nederland B.V. First I would like to thank my two supervisors Dr. Marco Wiering of the University of Groningen and Dr. Han Suelmann of CGI. Marco Wiering provided me with welcome guidance in the academic form. Han Suelmann made sure I really kept on working on the project and it is perhaps much by his aid that I finished the thesis now. As a third sort of supervisor I would like to thank Dr. Wico Mulder for his everlasting enthusiasm for his work and mine. He also arranged some interesting meetings for me during my project. CGI has provided substantial resources and support for me to conduct this research for which I am very thankful. I would like to thank my colleagues at CGI. Especially the ones I saw on a daily basis and enjoyed our afternoon walks with. Thank you for your interest in my work and small talk during coffee time. Special thanks to Aliene van Veen MSc who was my co-research intern for her own project.

I would like to thank the University of Groningen for allowing me to use the Millipede cluster and especially thank the staff at the High Performance Computing center for their support in the use of the cluster.

Thanks to my mother, Caroline Lier-Murphy, father, Mario Lier, and sister, Fiona Lier for their support not only during the past year, but during my whole academic career. My friends from dancing who always helped me forget about work during the many events we attended. Especially ir. Anke Veens with her hilariously cynical approach to motivation. Also lots of thanks to my friends in Groningen for supporting me during these busy times. You have all been of great support to me and helped me finish this project and thesis.

Contents

Chapter 1

Introduction

1.1 An Energy Revolution

Over the past decades an energy revolution is taking place in the world. On all continents people are investing in renewable sources of energy. Reasons behind this can be self sufficiency and lower climate impact. These energy sources, mostly solar and wind power, are different from the old electricity sources. While the old sources are often centralized power plants, run by governments and, more recently, private corporations, the new energy sources are distributed over a larger area and often installed at a consumer site. Another difference between the old and the new sources of energy is that with the new sources there is no control over when we generate electricity. This is because these sources rely on uncontrollable variables such as wind speed and sunshine. When there is wind or sunshine we generate electricity but when there is not we can not do anything about it and have to rely on other sources.

This change in generation capacity distribution is cause for concern for utility companies. While before these utilities knew exactly who the players on the electricity market were and could control generation and distribution themselves, now there are many more producers and thus many more variables to take into account when planning for network management and generation capacity. Even in the old situation Short Term Load Forecasting (STLF) is used to predict how much electricity is going to be needed the next day. This way the supplier always knows how much demand it can expect and can make sure it arranges enough backup capacity. Now with the new variables this system is becoming more complex. We are dependent on solar irradiation and wind forecasts and these have to be taken into account when doing Short Term Load Forecasting. This thesis therefore focuses on Short Term Load Forecasting for a residential area where most houses are fitted with solar panels.

The European Council wants the share of renewable energy sources in the European Union to be at least 20% by 2020 (European Council). During the last decade investments in renewables have been growing and with it the share of renewables in overall electricity production. While the overall share of solar and wind power remains small when compared to a source like hydro power, their share has been

growing rapidly, especially since 2013.[1] Electrification of transport will shift peak loads to different times of the day and enlarge them greatly. Research is also being done on how to take measures to shift loads during the day, using smart grids, to lower daily peak loads.

The electricity network used to be a government-owned system where all parts were run by the same entity. In the last decade and a little before more and more of this system got liberalized and to ensure fair market competition all subunits are supposed to be owned by separate parties. This made the electricity network a complex system of multiple parties who have their own responsibilities and goals. The system has two end points. One at central generation and one at the consumer side. As mentioned before, the consumer side is getting a little more fuzzy by the day, because of consumers who are also starting to generate electricity. The central generation side is known to the market as the producer. The producer owns one or more power plants, which in The Netherlands are mostly coal and gas fired power plants, and a single nuclear power plant. The consumer buys electricity from a supplier. These suppliers can own their own power plants but it also happens that they do not own any generation capacity themselves, but buy it all from the producers to resell it to the consumers. In between these parties there are still two more parties involved. One is the Transmission System Operator (TSO) and the other is the Distribution System Operator (DSO). The TSO manages and maintains the high voltage transmission system between producers and the DSO. The TSO also maintains connections with other TSOs so electricity can be exchanged between different states, countries and continents. The DSO takes over where lower voltage lines are involved and distributes the electricity to the end points, the consumers.

Between these parties a daily market for exchanging electricity exists. Every day the suppliers want to ensure they have bought enough electricity from the producers to supply all their consumers. On their part, the producers need to ensure they have sufficient resources available to be able to produce the demanded amount of electricity. TSOs and DSOs need to ensure that their network is in sufficient working order to be able to transport all electricity between the parties. For all this planning assessment of future requirements is needed. Forecasting is a major part of this assessment. TSOs and DSOs can do with peak load forecasts further into the future just for maintenance scheduling and infrastructure upgrade planning. Producers also use forecasts further into the future, but would like to know the total demand over a longer period, so they can negotiate supply contracts for resources. The suppliers have the hardest job in forecasting, because they would like to know exactly how much power is going to be used at each point in the day. Every day they make new forecasts for the next day so they can trade capacity and ensure reliable electricity supply for their clients.

In the other chapters of this thesis you can read what was done to incorporate the use of solar panels at the consumer side into Short Term Load Forecasting. But first the next section will point out the major contributions of this research, which is then followed by the outline of the rest the thesis.

[1] http://ec.europa.eu/eurostat/statistics-explained/index.php/Renewable_energy_statistics

1.2 Contributions and Research Questions

This thesis was written as part of a research internship at CGI Nederland B.V. CGI is a global corporation with clients in the financial, health, government, transportation, and, most importantly for this research, the utilities sectors. In these sectors CGI provides, among others, business and IT consulting, application development and management, and systems integration services. This project will help CGI in the assessment of feasibility of doing predictions with the data they have.

Improvements to short term forecasting must constantly be made, because of the changing behavior of the market. Increase in decentralized and consumer side electricity production with renewables is also a major factor why ongoing research in this area is needed. This thesis contributes to the work in this field by answering the following questions:

1. Which of the following methods is the most accurate in short term forecasting?

   Hierarchical Linear Model

   Random Forest

   Echo State Network

2. Is there a difference in forecasting accuracy for different variables related to load?

   24-hour ahead area electricity consumption per 15 minutes.

   24-hour ahead area electricity demand per 15 minutes.

   24-hour ahead area electricity supply per 15 minutes.

   24-hour ahead area solar power generation per 15 minutes.

For question one the hierarchical linear model was chosen because it is a common least squares method used in short term forecasting and serves mainly as a baseline to compare the other methods to. The Random Forest is chosen because of its proven worth in business decision making processes and its ability to deal with large datasets. The Echo State Networks are the main computational intelligence contribution of this thesis. These methods will be further explained in the second chapter.

The first sub-item of question two, the electricity consumption, is a variable which will mainly be of use for applications for consumers. One application could be to use the prediction of electricity consumption to challenge consumers to stay below the predicted amount and reduce electricity use. The demand is the amount of electricity that consumers actually buy from the supplier. This is important for the suppliers of electricity, because they need to meet this demand at all times. The supply is the amount solar power that is delivered by consumers to the suppliers network. It is also an important variable for suppliers of electricity, because they might be able to use that supply to meet demand else where and will thus buy less electricity from the central generators. The solar power generation forecasts will mainly be important for the consumers. They will be able to plan their energy use to times which are rich with solar power.

1.3 Outline of Thesis

The next chapter will go in depth on all topics related to the experiments. Some history on relevant methods will be provided and references to interesting papers given. In chapter 3 will be explained about the data that were used for the experiments and what needed to be done to prepare the data into datasets. The specifics of the created datasets will be explained here. Chapter 4 goes in depth on the implementations of the methods, experiments, and their results. Finally this thesis will conclude with a chapter for discussion of the results and some guidelines for future research.

Chapter 2

Theoretical Background

Time-series forecasting problems have been investigated in many different contexts. Examples are economic (stock market) (Newbold and Granger, 1974), meteorological (temperature/climate) (Brown et al., 1984), and electric load forecasting for transmission networks (Paarmann and Najar, 1995). Over all the fields in which time-series forecasting is used there exist many different types of time-series forecasting problems. For example there exist univariate time-series problems where the future behavior of a variable is predicted based only on its own past behavior (Lütkepohl, 2004). Predictors are variables that are known at the time of forecasting and might be of influence to the response variable. Multivariate time-series use multiple predictors to predict future behavior of one or more response variables (Reinsel, 2003). Load forecasting is usually approached as a multivariate problem, having multiple predictors and one (in the case of peak load) or more (in the case of load profile) outputs. Another distinction between different time-series forecasting problems is the forecasting horizon or how far the prediction goes into the future. Short term problems focus on near future forecasting. In the case of electric load forecasting this means 24 hours to a week ahead. These forecasts are used to plan electricity generation and make market decisions for suppliers. Medium term load forecasting generally means a week to a year ahead and is used for instance in the negotiation of contracts with other companies. Long term forecasting is anything beyond a year and is generally used to plan structural adjustments to the infrastructure and power generation assets (Hahn et al., 2009). This project focuses on Short Term Load Forecasting and this chapter will therefore focus primarily on related work in that area.

The next section will go in depth on Short Term Load Forecasting with a small overview of papers on this subject. After that a section is devoted to statistical and time-series approaches to Short Term Load Forecasting. Then the Machine Learning or Computational Intelligence approaches are discussed with an in depth view at Random Forests (Breiman, 2001) and Echo State Networks (Jaeger, 2001) as the primary methods for the experiments presented in this thesis. Closing this chapter is a section about Particle Swarm Optimization (Eberhart and Kennedy, 1995; Kennedy, 2010) which is used to find the optimal hyper-parameters of the Machine Learning methods employed.

2.1 Short Term Load Forecasting

In Short Term Load Forecasting there are several target values which can be the forecasting goal. Some systems forecast peak load for a certain period in the future, which is the maximum load that can be expected. Other systems forecast the cumulative load of a certain point or period in time. A system can also return multiple values like hourly loads for the entire day. This is called a load profile for that day. There can be a difference between systems regarding the inputs to the system as well. Some systems work in a univariate (time-series prediction) manner, while other systems use inputs such as past loads and other influencing variables for the forecasting.

Load forecasting is done for many different components of the electricity network. Network managers and utility companies always need to know how much load they can expect for any given point in time. Network managers need to be sure that every day each subsystem of the network is capable of handling the expected loads and otherwise they can take action in re-routing or limiting loads. Utility companies need to know how much energy they need to generate or buy to meet the demands of their customers. This is specifically the case with Short Term Load Forecasting. Because it deals with immediate demand Short Term Load Forecasting needs to be as accurate as possible so no shortage or waste of resources will occur.

In the literature (Gross and Galiana, 1987; Hahn et al., 2009) the factors that influence the load are often divided into four categories.

- Economic
- Seasonal
- Weather
- Random

Economic is used to identify those factors that arise from the different types of users that exist on the network. Residential areas have very different load profiles than industrial areas. In some areas there would even be types of users which have no typical load profile but only occasionally consume a lot of energy. This can be research facilities and specific industrial sites. Seasonal is every factor arising from time. Every day people get up out of bed, go to work, come home and go to bed again. This daily rhythm is clearly visible in residential load profiles. Another seasonal effect is holiday anomalies in the load profile as people tend to use electricity differently when they have a holiday. Weather effects are for instance the use of air conditioning in hot summers and electric heating in cold winters. Besides these measurable effects there is also a random component to the load, because the total load exists of all loads of different users together and each user has its own unpredictability in energy use. Most of these relationships are linear but a weather variable, such as the temperature, has a non-linear relationship to the load (Kyriakides and Polycarpou, 2007).

Research on Short Term Load Forecasting goes back as far as 1966 when Heinemann et al. (1966) used a model based regression approach to daily peak load fore-

casting for the summer months. Since then many different methods have been proposed which can be roughly divided into three different categories. Regression based, time-series approaches, and artificial intelligence/expert systems (Hahn et al., 2009). According to Hahn et al. (2009), the Mean Absolute Percentage Error (MAPE) is the most used error measure, but Hippert et al. (2001) conclude from their overview that squared error measures would be more fitting because the loss function in Short Term Load Forecasting is not linear.

One of the first overviews in this field is given by Gross and Galiana (1987). Although they focused more on the practical side of the load forecasting application than the theoretical side of the research. Still Gross and Galiana (1987) concluding remarks note that ARMA models were the most popular at the time for their relatively low complexity in number of parameters and computational load. Experiments on actual load and weather data were few in numbers, so conclusions about field performance could not really be made. Some examples of AR(I)MA models can be found in the next section.

Hippert et al. (2001) published a review paper on Feed Forward Neural Networks used for Short Term Load Forecasting. In that paper they explain how most published work on Neural Networks contains incompletely tested conclusions because researchers did not use standard benchmark tests or all the available analytical tools to understand the performance. They are also convinced that by using overly large Neural Networks researchers are overfitting their data and cannot expect good accuracy on unseen data. Despite of this, they note, that Neural Networks have been used in every day operation with good performance. An example of very early work using Neural Networks for Short Term Load Forecasting is Chen et al. (1992), where non-fully connected networks were used to do hourly load forecasting based on past loads and weather data. More recent work with Neural Networks can be found in the paper by Bashir and El-Hawary (2009) where Neural Networks were designed using Particle Swarm Optimization and the data were preprocessed using Wavelet Transforms to remove redundant information.

An overview by Tzafestas and Tzafestas (2001) also included Fuzzy Logic and hybrid Fuzzy Neural Networks. The idea behind using a Fuzzy Logic System is that it can make better use of expert information on the load forecasting problem through its Knowledge Base. The hybrid version combines the two methods to minimize the drawbacks of each and maximize the potential of the system. Some nice introductory tutorials on Load Forecasting were written by Kyriakides and Polycarpou (2007) and Feinberg and Genethliou (2005).

The next section will go in depth on statistical and regression approaches to the Short Term Load Forecasting problem including basic time-series approaches. After this some Machine Learning approaches will be discussed.

2.2 Classical Approaches

Classical approaches to time-series forecasting can be divided into two major categories: Statistical approaches and regression approaches. The statistical approaches focus on the behavior of the time-series in the past and then try to extrapolate this behavior into the horizon that is wanted. The regression approaches model the rela-

tion between external predictor variables and the known load pattern. This model can then be used to predict unknown load values by using the external predictor variables.

The most used statistical approach in STLF literature is the Box-Jenkins Method (Box and Jenkins, 1994) for model selection (Taylor, 2003; Hagan and Behr, 1987). The Box-Jenkins method involves several steps to find the best model to fit a time-series. The underlying model used in the Box-Jenkins method is an Autoregressive (Integrated) Moving Average (AR(I)MA) model. The first step is checking the time-series for stationarity and seasonality. For a time-series to be stationary means that the rolling mean and rolling variance of the series remain the same during the whole period. Seasonality in a time-series means that there is a repeating pattern in the series. This information will determine whether and what order of Integration will be done by the AR(I)MA model. The Autoregressive terms estimate future values based on a weighted sum of previous values. The order of the AR model sets how many past values are used in the regression. The same principle exists with the order of the Moving Average terms.

Regression approaches are also widely researched. These include local polynomial (Bruhns et al., 2005), robust regression (Papalexopoulos and Hesterberg, 1990) and nonparametric regression (Charytoniuk et al., 1998) methods. But most salient in the literature is still the least squares optimized linear model (Christiaanse, 1971; Park et al., 1991; Haida and M., 1994). Here, based on the past measurements of the load, a weight is calculated for each input variable to signify how much it influences the load output. One exception to this rule is the temperature, which is often separately modeled non-linearly before being used as an input to the linear model. This is because a temperature drop on a hot day will cause a drop in load, because less air conditioning is used, while a temperature drop on a cold day will cause a rise in load because more heating is used.

2.3 Machine Learning Approaches

Several machine learning or computational intelligence techniques have been tested in respect to Short Term Load Forecasting. These are mainly Regression Trees, Fuzzy Logic systems, and Neural Networks. The Fuzzy Logic systems are preferred by many, because the Fuzzy Inference rules allow the clear extraction of input to output relationships. Regression Trees offer the same functionality, but Neural Networks are more of a black box method, where no clear relation can be extracted between inputs and the output. An example of Fuzzy Logic use for STLF can be found in (Mori and Kobayashi, 1996), where they use a fuzzy logic system for 1-step ahead hourly load forecasting. Their dataset is very limited, but they managed to get results of below 1% error. The next sections will go in depth on Regression Trees and Recurrent Neural Networks for Short Term Load Forecasting.

2.3.1 Random Forests

One of the intelligent systems employed by forecasters is the regression tree (Mori et al., 2001; Yang and Stenzel, 2006). Decision trees are a type of classifier or

regressor that splits the training data into subsets until a sufficiently fine grained result is obtained. This is more clearly explained in the next section. The following sections will first go into the history of decision and regression trees and how they can be trained. Then we will explain how an ensemble of regression trees becomes a Random Forest.

Regression Trees

Decision trees are widely used in all sorts of automated decision areas. In the old days human expertise was used to create the rules that made up the decision tree. But this process took quite long for each rule and the problems were getting more and more complex. Automated rule extraction was designed as a solution to this problem. Hunt et al. (1966) wrote one of the first papers on automatic decision tree creation. Since then a lot of research has been done on what criteria to use to build a decision tree that is efficient and accurate.

The CART algorithm (Breiman et al., 1984) was designed for decision and regression tree building. CART is an acronym for Classification and Regression Trees. Short Term Load Forecasting is a quantitative problem and so regression trees are used to predict the loads. Automated regression tree building is done by finding the most optimal successive splits between the training samples, until the prediction error is minimized. The tree building starts with a root node which contains all training data. The predicted value at this point is the average of all output values in the training data. The error of this prediction (in a leaf node l) is computed as the Mean Squared Error as in Equation 2.1, where N_l is the number of training samples in l, D_l iterates over all samples in l, y_i is the target value at sample i, and $\hat{y}_l$ is the predicted value for this leaf node.

$$MSE(l) = \frac{1}{N_l} \sum_{D_l} (\hat{y}_l - y_i)^2 \tag{2.1}$$

To minimize the tree error a split can be made of the training samples according to one of the feature variables. In case there are more than one feature variables a choice needs to be made to find the best one. This choice is made by first finding the most optimal split for each feature and then choosing the feature minizing the tree error. The error of the tree is computed as the weighted average of node errors (Equation 2.2). Where N is the total number of training samples in the tree.

$$MSE_{tree} = \frac{1}{N} \sum_{l \in tree} \sum_{D_l} (\hat{y}_l - y_i)^2 \tag{2.2}$$

The measure used to find the optimal split is the difference in MSE between the tree with the split and the tree without the split. The largest difference will be found at the most optimal split. This is shown in equation 2.3, where t is the unsplit parent node and t_l is the branch for which the split was True and t_r the branch for which the split was False.

$$\Delta MSE = MSE(t) - \frac{n_{t_l}}{n_{tree}} MSE(t_l) - \frac{n_{t_r}}{n_{tree}} MSE(t_r) \tag{2.3}$$

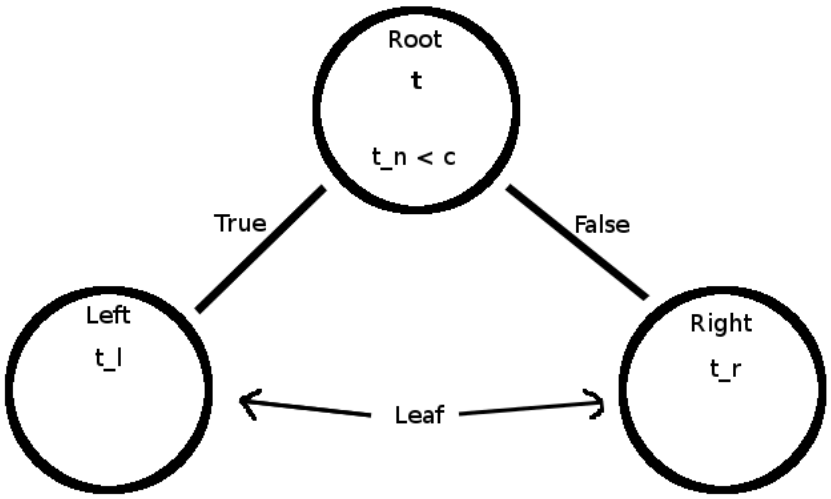


Figure 2.1: Example regression tree with a root node and two leaf nodes

An example of the first split of a tree is shown in Figure 2.1. The top of the figure shows the root node which contains all training data $\mathbf{t}$ and a splitting rule $t_n < c$, where c is the value of the optimal decision boundary for the set. At the bottom you see two child nodes and in the case of this small tree the leaf nodes where the subsets of training samples t_l and t_r fall into.

A common problem with automated learning systems is the possibility of overfitting. Regression trees are very prone to overfitting because a standard tree will split its nodes until there is only one sample per end node. This can result in an increase in test error. Two solutions have been proposed to combat this problem. One is pre-pruning where the decision tree is thresholded to a certain depth or number of leaf nodes. Another method is post-pruning which builds the entire tree and then uses a validation set to estimate generalization errors and prune split nodes that are the cause of error increase.

The method that is used in this research is less prone to overfitting as it uses a separate training set for each tree and can calculate the error of each tree with data that were not used for training; So called Out-Of-Bag samples. In the paper by Breiman (1996) it is shown that a bootstrap aggregation (bagging) approach to ensemble learning can improve upon single regressor accuracy. Bagging is the method that chooses a random sample of all training data to train a regression tree. The increase of accuracy is related to the level of decorrelation between the different regressors in the ensemble. Following this research Breiman (2001) proposed an extra level of decorrelation by not only taking random samples from the training data but also choosing a random subset of features to test at each split in a decision tree. This method was coined Random Forest and is less prone to overfitting because it uses a separate training set for each tree.

Breiman's forests

In Breiman (2001) an ensemble method for decision trees was introduced with random sampling of the training data and random sampling of accountable features.

The idea behind this was that many trees together have better generalization capabilities than one tree. This method is called Random Forest; Not only because of the random sampling of training data but also because at each split a subset of the predictors is chosen and only from this subset the best split variable is taken.

This method introduces several parameters that can be optimized for the best performance.

- Number of trees
- Number of training samples per tree
- Number of features tested at each split

The number of trees in the forest impacts several steps in the regression process. First of all, a new random sample of training data is taken for each tree, so increasing the number of trees increases the number of random samples of training data needed. Second, it has been shown that increasing the number of trees brings the accuracy closer to the theoretical limit of the system. The number of training samples per tree handles the amount of correlation between trees, because the higher this is, the more chance there is that trees have overlap in their training sets. The number of features tested at each split controls whether or not at each split the optimal feature is used. Lowering this number increases the chances that the global optimal feature is not in the subset of features tested at this split. This increases the chance that the different trees in the forest split in very different ways and will provide different estimates to the regression.

The Random Forests used in this research were generated with the RandomForest package[1] in R. This package provides support for classification trees as well as regression trees. No changes needed to be made to this implementation as it has all parameters for a Random Forest available and provides support for separate training and validation sets. A Random Forest does not require input normalization so the data were used as is.

In the following experiments the parameters optimized were: the *ntree* (Number of trees), *mtry* (Number of features tested at each split), *sampsize* (Number of training samples selected for each tree), *corr.bias* (Bias correction), *nodesize* (the amount of training samples left per leaf node). Other parameters are the *maxnodes*, which controls the maximum leaf nodes the tree can have and is not used by default, and the *replace* parameter, which controls whether training samples are drawn with or without replacement and defaults to true. The next section will introduce Echo State Networks, a specific form of recurrent neural network.

2.3.2 Echo State Networks

This section explains the background of the Echo State Network (ESN) (Jaeger, 2001). ESNs are a form of recurrent Artificial Neural Network (rANN). First we will explain what ANNs are, then we will explain the specific features that make an ANN an ESN.

[1] https://cran.r-project.org/web/packages/randomForest/index.html

The Artificial Neural Network

Artificial Neural Networks are known to be universal approximators (Hornik et al., 1989). This means that using a sufficiently large hidden layer, the network can map any function from one finite dimensional space to another. Based on a modular unit named the perceptron it would enable brain like function in a digital computer. A perceptron or artificial neuron is a computation unit which has an input vector and an activation and the mapping between the input and output is given by the activation function.

The strength of perceptrons becomes evident when you network them together, the activation of one becoming the input for another. Artificial neural networks are networks of perceptrons. Often with a distinct input layer, which receives the input variables. The input layer connects to a hidden layer which is often much larger than the input layer and can exist of multiple layers. The output layer reads the last hidden layer activations and the activation of the output layer is the output of the ANN. The perceptrons in the ANN are interconnected by weights. The strength of the weight determines how much of the activation of one perceptron goes into the perceptron it is connected to.

Changing the activation function for the artificial neurons can greatly alter their behavior. A linear activation function in the neurons of a network would make the network obsolete, because the results can be modeled by a single linear neuron. But when using another activation function, like a hyperbolic tangent, the network can map non linear functions and becomes more useful.

The use of hidden layers made it difficult to train the internal weights of the ANN, but the back-propagation algorithm (Rumelhart et al., 1988) solved this problem. It provided a way for the error on the output of the ANN to be used to train the weights between all layers of the ANN up to the inputs. This works fine for unidirectional networks, where the connections go from the input layer into the hidden layers and from the last hidden layer to the output layer without feedback loops between the layers. In some cases it might be useful to have these feedback loops in your network. The network is than called a recurrent network. Training recurrent networks can be done with back-propagation through time (Werbos, 1990) but is computationally expensive because of the need to compute all (partial) derivatives of the error with respect to the weights. This is the reason Echo State Networks were designed without having to train all hidden layer weights. The next section will explain exactly how Echo State Networks are structured and trained.

The Echo State Approach

Echo State Networks (or ESNs) are recurrent Neural Networks. A typical ESN consists of an input layer, a hidden layer consisting of a large reservoir, and an output layer. The input layer is fully connected to the hidden reservoir. The nodes in the reservoir are sparsely connected to each other and these connections are generated randomly at the initialization of the network. This means that nodes can be connected to themselves too and that there could be paths in the hidden layer which lead from a certain node and return back to that node.

The output layer is also fully connected to the hidden reservoir. In this case the

connections between the reservoir and the output layer work in two ways. During training a teacher signal can be presented at the output layer to show the network the correct response. We already said that an ESN does not change all its weights during training. Only the reservoir to output weights are changed during the training phase of the network. Because of this the optimal weights can be calculated by solving the linear equation system of the activation of the reservoir to each presented training sample towards the target output for those samples. Most often the pseudo inverse method is used to this end (Lukoševičius, 2012).

The Moore-Penrose pseudo inverse $\mathbf{A}^+$ of a matrix $\mathbf{A}$ is a matrix which satisfies the following constraints (Penrose, 1955):

$$\begin{aligned}\mathbf{A}\mathbf{A}^+\mathbf{A} &= \mathbf{A}\\ \mathbf{A}^+\mathbf{A}\mathbf{A}^+ &= \mathbf{A}^+\\ (\mathbf{A}\mathbf{A}^+)^T &= \mathbf{A}\mathbf{A}^+\\ (\mathbf{A}^+\mathbf{A})^T &= \mathbf{A}^+\mathbf{A}\end{aligned}$$

We could use the standard inverse in some cases but the pseudo inverse also works in situations where $\mathbf{A}$ is not invertible. The pseudo inverse is found by equation 2.4, where $\mathbf{X}$ is the collected states matrix (explained in the next paragraph) of the Echo State Network containing the activation of the reservoir for each training input.

$$\mathbf{X}^+ = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}^T \tag{2.4}$$

The optimal output weights $\mathbf{W}^{out}$ for the output connections can then be found by multiplying the pseudo inverse with the target output $\mathbf{Y}^{target}$ (Equation 2.5).

$$\mathbf{W}^{out} = \mathbf{Y}^{target}\mathbf{X}^+ \tag{2.5}$$

The collected states matrix is collected in the following way. For each sample in the training set the activation of the network is computed using Equation 2.6 from Jaeger (2001), where $\mathbf{x}(t)$ is the state of the reservoir at time t, $\mathbf{W_{in}}$ is the input weights vector, $\mathbf{u}(t)$ is the input vector including a bias term, and $\mathbf{W}$ is the reservoir weights matrix. This is saved per sample in a row of the collected states matrix.

$$\mathbf{x}(t+1) = \tanh(\mathbf{W_{in}} \cdot \mathbf{u}(t) + \mathbf{W} \cdot \mathbf{x}(t)) \tag{2.6}$$

The output $y(t)$ is then computed with Equation 2.7, where $\mathbf{W}_{out}$ is the output weights vector.

$$y(t) = \mathbf{W}_{out} \cdot \mathbf{x}(t) \tag{2.7}$$

The implementation used in this research is an adapted version of the free ESN sample code provided on the Jacobs University website[2]. The adaptations done were mainly parameterization of fixed variables in the code. The Spectral Radius was fixed at 1.25 for the example, but it will be optimized for the current application. The level of connectivity was not really implemented at all, but instead a random uniform distribution of the weights was used as initialization. In the code used in

[2]http://minds.jacobs-university.de/mantas/code.html

this research the weights are initialized zero. Random weights are chosen to be given a random value between -0.5 and 0.5, until the number of non-zero weights are in percentage equal to the connectivity parameter. The spectral radius is controlled by dividing the weights matrix by its current spectral radius and multiplying by the desired spectral radius parameter.

Several parameters were mentioned which can be optimized to find the Echo State Network which gives best accuracy on an application. The size of the reservoir is an important parameter. A reservoir that is too large will easily cause overfitting on training data and worse accuracy on test data. If the reservoir is too small it might not be able to capture all nuances of the underlying function. The type of node used is also important and goes hand in hand with input normalization. There exist many types of nodes: tanh, linear, sigmoïd, step functions. Each of these has a different input range for which the activation function has a derivative which is non-zero. It is important to ensure that the inputs stay within this range, because otherwise changing the input will not have any effect on the activation of that node and consequently could have the same effect on the output of the entire network. The spectral radius of the reservoir weight matrix determines the memory of the reservoir or how long the activation of the network at time t will have influence in the future of the network activation.

The next section will explain the optimization procedure used to find the best parameters for both the Random Forests and the Echo State Networks.

2.4 Particle Swarm Optimization

The Random Forests and Echo State Networks described in the previous sections rely on a number of parameters for optimal performance. Finding the most suitable parameters for any machine learning problem is a machine learning problem on itself. Several methods are available in literature for automatic parameter optimization. Grid search is a method where every parameter combination is tested in an extensive manner. Because there are often a significant amount of values to test, testing all possible combinations can be very time consuming. Parameter optimization can also be done with evolutionary algorithms. Evolutionary algorithms generate random combinations of possible parameter settings and then test how well these parameters perform. The next iteration only the best combinations survive and often get combined with other combinations and randomly altered a little before new tests are performed. Inspired by these algorithms is the Particle Swarm Optimization Algorithm (Eberhart and Kennedy, 1995).

Particle Swarm Optimization also starts with random combinations of parameters. Each iteration these combinations are tested and the global best performing parameters are saved centrally. Each 'particle' also remembers the local best performing parameters which it has visited itself. Then, after testing, a speed is calculated with Equation 2.8 for each particle depending on its current speed ($OldVelocity$), a random portion ($R1$) of the distance of the current location to the local best location (**DistanceToLocalBest**) multiplied by an acceleration constant (ACC), and a random portion ($R2$) of the distance to the global best location (**DistanceToGlobalBest**) also multiplied with the acceleration constant ACC.

The new speed of each particle is used to update the parameter set of the particle using Equation 2.9. Each particle slowly moves to the best found optimum this way, but is testing other parameters on the way and could eventually find a better optimum.

$$\begin{aligned} \textbf{NewVelocity} = {} & \textbf{OldVelocity} & + \\ & ACC * R1 * \textbf{DistanceToLocalBest} & + \\ & ACC * R2 * \textbf{DistanceToGlobalBest} & \end{aligned} \tag{2.8}$$

$$\textbf{NewParameterSet} = \textbf{CurrentParameterSet} + \textbf{NewVelocity} \tag{2.9}$$

Chapter 3

Data: From Source to Dataset

This chapter explains where and how the data used in the experiments were collected and cleaned. In section 3.1 the project Your Energy Moment (YEM) and the Central Energy Management System (CEMS) will be introduced. These form the basis for the data collection. Section 3.2 explains what actions were needed to process the raw data from the database into clean data. Some preliminary data analysis was done and is presented in section 3.3. The last two sections of this chapter explain the internal and external predictors which form the datasets for the experiments.

3.1 Data Collection

A DSO like Enexis sees the need for investigating the possibility of Demand Side Management (DSM) of energy requirements in the future. Without DSM the peak load on their network will be many times greater than the base load and this calls for unnecessarily high requirements for their distribution infrastructure (Klaassen et al., 2013). From these investigations have risen a few pilot studies with smart meters in residential areas. One of these was done in a residential area in Zwolle. There, in a new neighborhood, the houses were outfitted with smart meters and people had the option of getting a smart washing machine as well. These buildings also have solar panels installed on their roofs and thus the residents are considered prosumers by definition, because they both consume and produce electricity. Through an Energy Information Display called Toon ('show' in Dutch) participants get insight in how much electricity they are using and generating. The extra benefit of these displays is that they can also show a variable price curve and a predicted solar power generation curve for the current day. In this way the researchers at Enexis wanted to see how these incentives influence the daily load curve.

To facilitate these projects the expertise of CGI was employed to develop a central system to communicate between the smart meters, smart appliances and the research database. This system is called the Central Energy Management System (CEMS). CEMS formed the middle system between the displays and sensors, the information from the providers, and the database. The information from the smart meters is gathered every 15 minutes. The system records how much electricity was used, how much solar power was generated, and how much power the washing machine used. Table 3.1 shows the relevant data which are gathered into the CEMS

database. Every day at noon the incentive price and eco curves for the next day are generated based on information from the energy suppliers and predictions by a meteorological office and sent to the Energy Information Displays. Based on settings in the Energy Information Display the smart washing machines can use the eco or price curve to plan turn on times if allowed by the user.

Column name	**Unit**	**Table**	**Explanation**
NetUsage	Wh	EnergyMeterFacts	The total electricity use in the current period
WashingMachine	Wh	EnergyMeterFacts	The power requirements of the washing machine in the current period
PvProduced	Wh	EnergyMeterFacts	The produced solar power in the current period
ConsumeHigh	Wh	EnergyMeterFacts	The amount of external power supply demanded during high tariff
ConsumeLow	Wh	EnergyMeterFacts	The amount of external power supply demanded during low tariff
ProduceHigh	Wh	EnergyMeterFacts	The amount of solar energy delivered during high tariff
ProduceLow	Wh	EnergyMeterFacts	The amount of solar energy delivered during low tariff
DateTime	-	all	The start date and time of the period for which this row holds information
DateDimension_Id	-	EnergyMeterFacts	The Id of the date of this entry in the Dates table
PeriodDimension_Id	-	EnergyMeterFacts	The Id of the period of this entry in the Periods table
HouseDimension_Id	-	EnergyMeterFacts	The Id of the house of this entry in the Houses table
SolarGeneration	Wh	WeatherForecastFacts	The expected solar generation for this period

Table 3.1: Relevant fields in CEMS database

All these data and more are collected into the CEMS database every day. While this usually goes well, especially in the first few months of data collection some start up problems created faulty entries in the database. The next section will explain how we used the data from the CEMS database to create a couple of clean datasets on which to run the experiments.

3.2 Data Cleaning

To create datasets data were used from the CEMS database from October 2nd 2012 10:15 to May 7th 2015 8:00. This totalled to 13,176,463 rows in the CEMS database. Anywhere where data is collected, noise influences measurements. This is also the case in the YEM project where the database contains some erroneous entries. Cases are known where the date of the *DateTime* field and the date of the *DateDimension_Id* do not coincide. Some samples are also dated in the past as far back as 1970, because the time on the smart meters was not correct. Then there are the missing rows. These are periods for which data from all houses are completely missing or only for a few houses of which is known that they were active in the past but do not have data in the database for a certain period. This section explains how these issues were dealt with.

At first sight 1132 rows appeared to be duplicates as they had the same *DateDimension_Id*, *PeriodDimension_Id* and *HouseDimension_Id*. The *NetUsage* was not equal between the rows and it was discovered that the *DateTime* field did not specify the same date as the *DateDimension_Id*. Because the *DateTime* appeared to be the most logical value in these cases it was decided to use that to reset the other fields. In 92874 cases however there was no DateTime present. So these had to be filled in with existing information from the *DateDimension_Id* and the *PeriodDimension_Id*.

For the latter cases the *DateDimension_Id* and the *PeriodDimension_Id* were used to create a *DateTime* value accordingly. Now all existing rows had a DateTime value, this was used to reset all *DateDimension_Id*s. Table 3.2 shows how many rows have an offset between the *DateDimension_Id* and the *DateTime* field from the year 2012 onward. The 92 rows that had a *DateTime* before 2012 (the starting year of the project) were adapted according to their *DateDimension_Id* fields first.

Offset in days	Number of rows
1	22685
2	2679
3	874
4	654
5	162
6	267
>6	185

Table 3.2: Offsets between *DateDimension_Id* and *DateTime* field starting in 2012

After filling in the missing *DateTime* fields the *DateDimension_Id* was adapted to the appropriate values according to the *DateTime* field. This left 1782 duplicates, which appeared to be mostly shifted periods on the same days. So the *PeriodDimension_Id* was also adapted according to the *DateTime* field. This resulted in 138 leftover duplicates. From inspection it was noted that the first row of each of these duplicates showed very large or very small values, while the second row showed more likely values compared to existing entries for this period. Therefore it was decided

to keep the second row of these duplicates and dump the rest.

After these actions there remained 14568 rows that had a *DateTime* and a *DateDimension_Id* that lay before the project start date. As the total of these rows was a very small portion of the available rows and the maximum percentage for any one house was less than 10% of the rows for this house, it was decided to delete these rows. An issue that is related to the removal of rows is that some rows are missing. These are usually isolated periods or a couple of subsequent periods for a single house, but in a few cases also a complete period where for all houses the rows are missing. A script was devised to go through the data and inserting rows wherever they were missing. It keeps track of which houses have already sent information to the database so we don't add rows for a house when it was not active yet. Information we can fill in for these rows were just the *DateDimension_Id*, *PeriodDimension_Id*, *HouseDimension_Id* and *DateTime*.

Measurements are usually noisy and sometimes this means values are completely missing. In the YEM database this has sporadically happened. For the *NetUsage* column only 14 millionth ($1.4{*}10^{-5}\%$) of data was missing. For every separate house this never amounted to over 0.27% of the available rows. In the *PvProduced* column the total missingness was also $1.4 * 10^{-5}\%$. Per house there was less than 0.25% missing except for two houses. The first had 1.9% missing and the other 100%. It was found that the last house had no solar panels. The missing value imputation algorithm inserted zeroes over the whole range of this house. Also when inserting rows that are completely missing there are no data available for the relevant columns. This is why there is a need for missing value imputation in the data processing pipeline.

A self-devised form of missing value imputation was implemented (Algorithm 3.2.1) for the columns *NetUsage*, *PvProduced*, *NetSupply*. Each period (*time*) is checked for missing values (NAs) in a certain column. When a period contains NAs an imputation value is calculated by taking the mean of all existing values for that period and column. If there are no existing values a zero is inserted. The imputation value is then filled in for all NAs in the current period. This was also used to fill in the columns resulting from combining the *ConsumeLow* and *ConsumeHigh* columns (*CentralSupply*) and from combining the *ProduceLow* and *ProduceHigh* columns (*SolarSupplied*). For the column *WashingMachine* zeroes were entered, because this column is almost always zero.

Algorithm 3.2.1: LinearMissingValueImputation(*dataTable*, *columnName*)

for each *time* ∈ unique entries in *dataTable times* column
do
- **if** in rows with *times* is *time*
- the sum of NAs in column *columnName* is larger than zero
- **then**
  - **if** mean exist of entries excluding NAs for this time
  - **then** $imputationValue \leftarrow mean$
  - **else** $imputationValue \leftarrow 0$
  - NAs in *columnName* $\leftarrow imputationValue$

3.3 Data Analysis

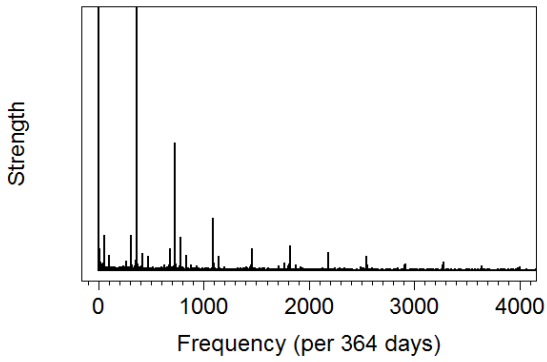
Data science always starts with investigating the properties of the data that is available. To this end some preliminary analysis was done on the dataset. Because these data are known to be periodic in nature, some frequency analysis was done to find out what frequencies are prevalent in the data. It was discovered that autocorrelation on the *NetUsage* data does not give a lot of information, because there is a high daily correlation in the data with a period of 96 samples and the other prevalent cycles in the data are larger and were clouded by the high daily correlation. To find out about other frequencies in the data Fourier Analysis was used.

The Fourier Analysis was carried out on 52 weeks of data. This includes a total of 34944 samples. As we have only almost three years of data Fourier analysis was carried out on the first 52 weeks, the second 52 weeks and the last 52 weeks. The last range therefore overlaps the second range. The mean of these analyses was taken to ensure that we would not be looking at artifacts within one range of data. Figure 3.1 shows the results of Fourier Analysis of the *NetUsage* column. It is clear that there is a large DC component by the peak at f0. A small peak shows at f1 which constitutes the yearly fluctuation of the energy use which can be more easily seen in Figure 3.1b. A peak at f52 is the weekly fluctuation in energy use as most households have a work/weekend structure in their energy use. The next peak occurs at f364 (Figure 3.1c), which constitutes the daily variation, because the data for our Fourier analyses contained exactly 364 days each time. Then there are consecutive peaks at f728 (twice daily) (Figure 3.1d), f1092 (three times daily), f1465 (four times daily), and f1820 (five times daily). The twice daily peak could be explained by the morning/evening pattern in daily life. The other peaks are not so easily explained even though they are clearly present in the data.

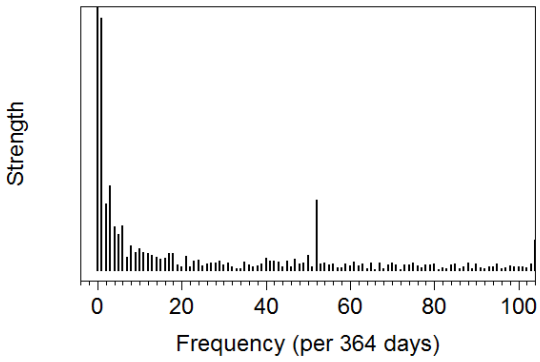
The *CentralSupply* data follows largely the same patterns as the *NetUsage* as you can see when comparing Figures 3.2 and 3.1a. There is a large base load component, a yearly component with $f = 1$, a weekly component with $f = 52$, and a daily component with $f = 364$. Interesting to note is that the weekly component is weaker and the daily component is much stronger than with the *NetUsage*. This is probably due to the fact that the demand is also influenced by the amount of solar power generated and the latter has a strong daily variation.

The solar energy that is produced in the area is of course more dependent on seasonal changes and day/night rhythm. That is why you would expect a strong daily rhythm in the frequency spectrum of the variable. Figure 3.3 show the frequency spectrum for this column. As you can see the daily rhythm at $f = 364$ is really strong and again there are strong components at multiples of this frequency.

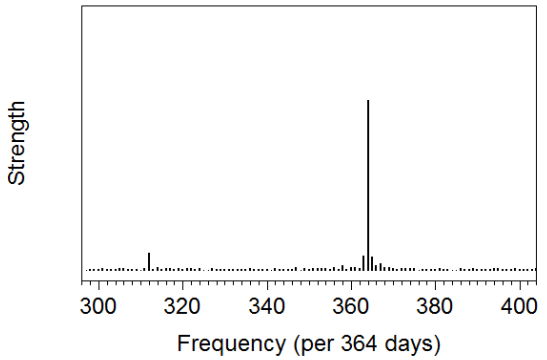
The solar energy supplied to the network by the area is a combination of solar power generated and power usage. Fourier analysis of this time-series shows a very large annual correlation with some random noise around (Fig. 3.4). As you can see for both the solar power generated as the solar power supplied there is no visible weekly component. These variables are primarily dependent on weather variables. The dataset to predict the supply therefore contains mostly weather variables and only the *PvProduced* and *Supply* values from the past day.



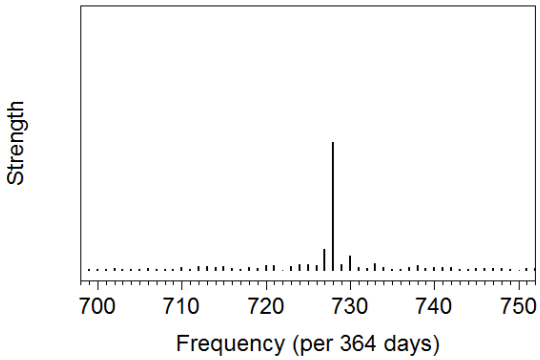
(a) f0 - f4000



(b) f0 - f100



(c) f300 - f400



(d) f700 - f750

Figure 3.1: Fourier analysis on power usage data for 364 days

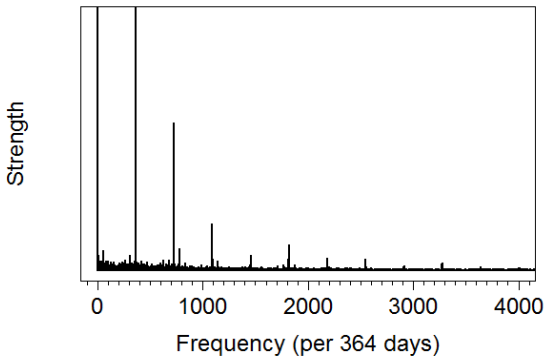


Figure 3.2: Fourier analysis on power demand data for 364 days

3.4 Internal Predictors

The CEMS database provides us with measurements for a lot of variables. The important ones for demand prediction are *NetUsage*, *PvProduced*, and *WashingMa-*

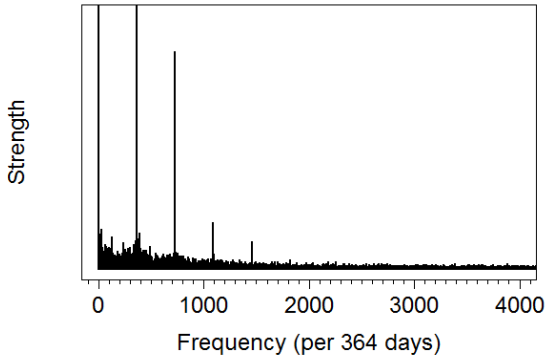


Figure 3.3: Fourier analysis on solar power generated data for 364 days

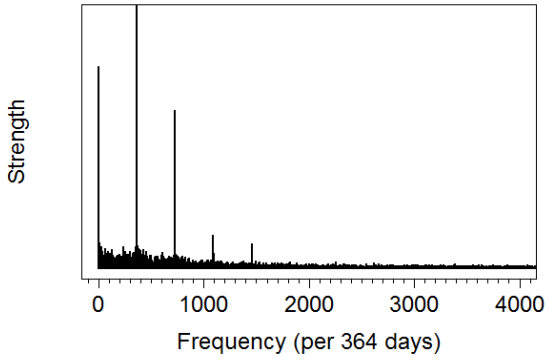


Figure 3.4: Fourier analysis on solar power supplied data for 364 days

chine. These can be used as predictors. Based on the analysis results in section 3.3 we use the *NetUsage* information of one day before, one week before, and one year before. Other important predictors which are present in the database are the date and time fields. To accurately predict the electricity demand you need to know what time of the year, the week, and what time of the day it is. Table 3.3 gives an overview of the 22 predictors that were derived from the CEMS database. Figure 3.5 shows why the number of houses is an important predictor for the predictions. The number is increasing steadily for a large part of the data collection period.

3.5 External Predictors

Several predictors were also included to the dataset from another source than the CEMS database. Weather information is important for several reasons. The amount of sun during the day is important for the amount of solar power generated. Temperatures and wind speeds are important for the amount of energy used in heating homes. From the KNMI (Royal Dutch Meteorological Institute) data collection[1]

[1]KNMI DataCentrum: http://data.knmi.nl

Name	Description
TargetOutput	Not a predictor but the target value for this sample.
NetUsage1d	The net electricity use of this moment one day ago.
PvProduced1d	The produced solar power of this moment one day ago.
WashingMachine1d	The electricity demand of the washing machine one day ago.
NumberOfHouses1d	The number of connections that were active one day ago.
NetUsage1w	The net electricity use of this moment one week ago.
PvProduced1w	The produced solar power of this moment one week ago.
WashingMachine1w	The electricity demand of the washing machine one week ago.
NumberOfHouses1w	The number of connections that were active one week ago.
NetUsage1y	The net electricity use of this moment one year ago.
PvProduced1y	The produced solar power of this moment one year ago.
WashingMachine1y	The electricity demand of the washing machine one year ago.
NumberOfHouses1y	The number of connections that were active one year ago.
UnixTime	The POSIX time of the target output with timezone 'UTC' and origin '1970-01-01'
PeriodCode	The point (quarter of an hour) of the day (1 - 96)
DayOfMonth	What day of the month (1 - 31) is predicted for
DayOfWeek	What day of the week (1 - 7) is predicted for
MonthOfYear	What month of the year (1 - 12) is predicted for
SolarPanels	The number of solar panels in the area
Demand1d	The power demanded from the network which is the sum of *ConsumeHigh* and *ConsumeLow* one day ago
Supply1d	The power delivered to the network which is the sum of *ProduceHigh* and *ProduceLow* one day ago
Demand1w	The power demanded from the network which is the sum of *ConsumeHigh* and *ConsumeLow* one week ago
Supply1w	The power delivered to the network which is the sum of *ProduceHigh* and *ProduceLow* one weeg ago

Table 3.3: Internal Predictors from the CEMS database

measurements were downloaded of two stations in the Netherlands which are close to the YEM project location in Zwolle. These are station number 273 (Marknesse) and station number 278 (Heino). These stations collect hourly weather data so for

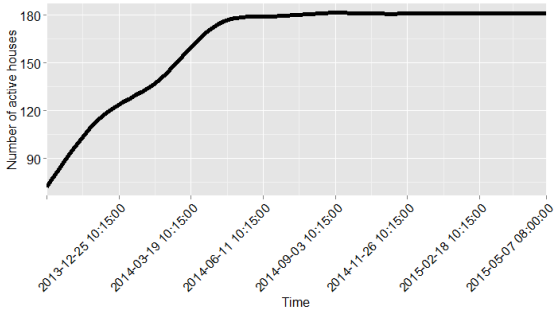


Figure 3.5: Number of Houses Active over Time

every four samples in our dataset the same weather information was used. The data from these two stations was averaged to come to an approximation of the weather conditions at the YEM project neighborhood. Table 3.4 shows the specifics of twelve external predictors that were taken from the KNMI data.

3.6 Training, Validation and Test Set

Several pitfalls exist when using Machine Learning algorithms to classify or predict. One of these pitfalls is overfitting your training samples. To prevent this from happening a validation set is used to assess the performance of trained algorithms. This validation set consists of completely different samples with regards to the data used for training. Based on the performance on the validation set optimal parameters can be chosen. The reported performance of the methods used is generated using a test set of samples which is not included in training or validation. For the experiments with Random Forests and Echo State Networks these three different sets were created. For the linear model a cross validation approach was used to determine which predictors to use in the model. After determining the best model, a new model was trained on the training sets and tested with the test sets.

Four different experiments have been done and for this four different datasets were created. For the prediction of *NetUsage* and *Supply* datasets were created that also contained a feature taken from one year ago, which greatly shortens the length of the time period available as for the whole first year there will be no data available for this feature. The training sets for these cases contain 45112 samples. In the datasets for *Demand* prediction only data of up to one week ago was used and thus these datasets contain more samples (79480 training samples). The dataset for *PvProduced* contains no historical values but only weather measurements and the number of solar panels and houses so it contains even more samples (80152 training samples). The complete datasets for power consumption prediction and supply prediction ranges from "2013-10-02 10:15:00 UTC" to "2015-05-07 08:00:00 UTC". For demand prediction the dataset runs from "2012-10-09 10:15:00 UTC" to "2015-05-07 08:00:00 UTC". The date range of the solar power generation prediction dataset is "2012-10-02 10:15:00 UTC" to "2015-05-07 08:00:00 UTC". Figure 3.6 shows the position in time of the validation and test sets. As you can see a range of samples was used from each season of the year to prevent accuracy bias towards a

Name	Description
WindDirection	Mean wind direction in the past hour (360=North, 90=East, 180=South, 270=West, 0=No wind 990=Changing.
WindSpeedMean	Mean wind speed (in 0.1 m/s) during the 10-minute period preceding the time of observation.
WindSpeedMax	Maximum wind gust (in 0.1 m/s) during the hourly division.
SunshineDuration	Sunshine duration (in 0.1 hour) during the hourly division, calculated from global radiation (-1 for <0.05 hour).
GlobalIrradiation	Global radiation (in J/cm2) during the hourly division.
GlobalIrradiation1d	Global radiation (in J/cm2) during the hourly division one day ago.
GlobalIrradiation1w	Global radiation (in J/cm2) during the hourly division one week ago.
GlobalIrradiation1y	Global radiation (in J/cm2) during the hourly division one year ago.
PrecipDuration	Precipitation duration (in 0.1 hour) during the hourly division.
PrecipAmount	Hourly precipitation amount (in 0.1 mm) (0.04 for <0.05 mm).
Humidity	Relative atmospheric humidity (in percents) at 1.50 m at the time of observation.
MeanTemp	Temperature (in 0.1 degrees Celsius) at 1.50 m at the time of observation.

Table 3.4: External Predictors

particular season. The validation and test sets are the same size for all four cases; 5376 samples each.

Figure 3.6: Timeline of Data

Chapter 4

Experiments

This chapter explains the experiments that were done on the data. As it is important for the energy suppliers to know how much energy they must provide, one of the experiments predicts the electricity demand of the area. For network managers the load on the network is important and this is also dependent on how much electricity is delivered to the network from the area's solar panels. This is why the second experiment focuses on predicting the supply by the area to the network. Surrounding the YEM project there are also predictions for the user on how much solar energy will be produced at a given time the next day. Therefore we will also try to improve on the predictions already present in the system with an experiment in solar panel generation. A fourth interesting value to know is the total electricity use of the user, so experiments are done to predict this variable. This can for instance be used in programs to create incentives for people to use less electricity than would be expected based on their previous use pattern.

The chapter will continue with an explanation of the methods compared in each experiment. The first method serves as a baseline experiment and will be explained here. This will be a hierarchical linear model regression on the available predictors for the target value. The hierarchical linear model is built up starting with one predictor and each step adding a new one until the root mean square error does not decrease anymore. Each test is done with 10-fold cross validation, so we avoid overfitting to some degree. Through this step by step manner of choosing the best predictors we intend to build a good linear model with the available data, while refraining from testing all possible combinations of predictors, which would simply take too much time. The equation for the linear model is shown in equation 4.1, where y' is the predicted value and c_i are the model coefficients, with c_0 being the bias term.

$$y' = c_0 + \sum_{n=1}^{N} c_n * P_n \tag{4.1}$$

The total number of predictors is denoted with N and P_n is the nth predictor in the model.

In the next section the Echo State Network implementation will be explained. A section on the Random Forest implementation will follow this. After these methods sections some sections on the specifics for each separate experiment and its results are added.

4.1 Experiment Setup

As discussed before, there are several parameters that can be tuned in the use of ESNs and Random Forests. To be able to compare the different experiments accurately we first need to use the same error measure (RMSE) to assess the fitness of our solutions. The previous chapter already discussed how the data were split into training, testing and validation sets. So here we will start by explaining the different parameters that we optimize for each method. Second, we discuss how the millipede cluster of the University of Groningen was employed to quickly do our experiments.

Particle Swarm Optimization was used to find the best parameters for the Random Forest and Echo State Network methods. A parameter input file specifies what the parameter ranges can be. For each parameter is also specified whether it is a continuous or discrete parameter. This setting is used after updating the parameters to decide to which decimal to round the new values. For the Echo State Network we optimized on four variables. Table 4.1 shows the parameters and their ranges. The reservoir size controls the amount of nodes in the reservoir and is a discrete variable. The connectivity controls the amount of non-zero connection weights in the reservoir and is a continuous percentage. The spectral radius controls the short term memory of the network and is a continuous variable. For the Random Forest there are four parameters we are looking at, shown in table 4.2. These are the number of trees (ntree), the number of training samples selected for each tree (sampsize), the number of predictors selected at each node (mtry), and the node size, which determines how many samples can maximally belong to any leaf node. All parameters for the Random Forest are discrete.

Name	Range	Mode
Connectivity	0.001 - 1	Continuous
Reservoir Size	100 - 2000	Discrete
Spectral Radius	0.1 - 1.1	Continuous

Table 4.1: ESN Parameters Optimized

Name	Range	Mode
ntree	10 - 500	Discrete
sampsize	100 - 10000	Discrete
mtry	1 - max. nr. of predictors in dataset	Discrete
node size	1 - 100	Discrete

Table 4.2: Random Forest Parameters Optimized

Some parameters, like reservoir size in ESN and ntree in Random Forests, have large ranges so twenty particles were used to get a good spread across the parameter space. The values for the parameters were initialized with a uniform random distribution within the range specified by the parameter input file. Each particle was tested for performance by running an experiment ten times and averaging the

results. The score used for optimization was the score of the predictor on the validation set. This way an estimate can be made of the performance on unseen data. After scoring all particles, the global best result is updated and for each particle the local best is updated whenever a better score was found. With the information of global and local best the velocity of the particles was adjusted according to Equation 4.2. For this update half the old velocity is taken so the particles will eventually stop whenever no better results are known. *R1* and *R2* are random uniform numbers between 0 and 1 which are re-chosen each time this formula is used.

$$\begin{aligned} \textbf{NewVelocity} = 0.5 * \textbf{OldVelocity} \quad &+ \\ R1 * \textbf{DistanceToLocalBest} \quad &+ \\ R2 * \textbf{DistanceToGlobalBest} \end{aligned} \tag{4.2}$$

$$\textbf{NewParameterSet} = \textbf{CurrentParameterSet} + \textbf{NewVelocity} \tag{4.3}$$

As said before this method was parallelized on the high performance computing cluster of the University of Groningen. Each generation of particles is tested in parallel and when all results from one set are back, an update can take place. A Python script was used to generate the particles and send the test jobs to the cluster queue. The cluster would then run the test with Random Forest or Echo State Networks and the parameters that belong to a certain particle. This way the experiments could be done much faster than if they are done one by one. The next section will show the results of the experiments.

4.2 Experiment Results

The reported results are based on test set experiments. For the linear model these experiments always return the same error, because the solution is fixed, this is why the standard deviation of the errors is zero. For the Random Forest and Echo State Network the error reported is the mean RMSE of ten runs with the optimal settings. The significance of the difference between the methods was tested with a Welch t-test with a significance level of $\alpha = 0.05$ in all cases. Some results show however that a much higher significance level can be met in a lot of cases. It has to be noted for clarity that these experiments only measured the inherent randomness of the methods used and that the methods might perform differently when the training sets are changed.

4.2.1 24-hour Ahead Power Consumption Prediction

In these experiments the actual power consumption of the area was predicted 24 hours ahead of time. The power consumption is measured in Watt-hours. For instance in showing the user the predicted power consumption over a certain period and challenging them to consume less than the predicted amount. Table 4.3 shows the optimal parameters found for Random Forest. Table 4.4 shows the optimal parameters found for the Echo State Network.

	ntree	sampsize	mtry	node size
Value	500	10000	7	1

Table 4.3: Optimal Parameters Random Forest for Total Power Consumption

	Connectivity (%)	Spectral Radius	Reservoir Size
Value	41.31	0.91	199

Table 4.4: Optimal Parameters ESN for Total Power Consumption

Table 4.5 shows the results of these experiments. At first glance it is already clear that these experiments yield two best performers when it comes to test error. The significance levels in Table 4.6 confirm this. The difference between the Random Forest and the Echo State Network is highly significant. The difference between the Random Forest and the Linear Model is actually not significant. The performance expressed by the mean RMSE in Table 4.5 is almost equal for Random Forest and the Linear model too, but because the time and complexity to train a Linear Model is greatly smaller than that needed to train a Random Forest, the Linear Model should be the preferred method for this case.

Method	RMSE	st.dev.
LM	1268	0
RF	1267	2.05
ESN	1371	5.37

Table 4.5: Results for Power Consumption Prediction

	LM	RF	ESN
LM	-	$t(9) = -1.49,\ p = 0.17$	$t(9) = 60.67,\ p = 4.53 * 10^{-13}$
RF	*	-	$t(11.56) = 57.22,\ p = 1.50 * 10^{-15}$

Table 4.6: Significance of the Results on Total Power Consumption

Figure 4.1 shows the predictions of the linear model plotted with the actual data. This is four weeks of data from our test set. The first half are winter data and the second half (after the black vertical line) are summer data.

4.2.2 24-hour Ahead Demand Prediction

The demand prediction experiments are very similar to the power consumption experiments. The difference is that in demand, we only take into account the amount of power that was bought from the power supplier. The same predictors are relevant in these experiments as in the power consumption prediction. What might be very difficult here is that we expect the system to predict just how much power is consumed and what share of that power is being produced by the solar panels in the area, and return a difference of these two. This is a quite complex question, because the total amount of power consumed in 15 minutes can be more than the

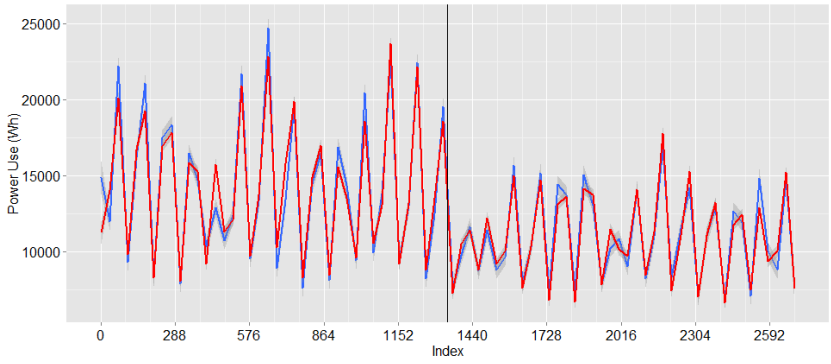


Figure 4.1: Total Power Usage Prediction with a Linear Model (Actual: Blue, Prediction: Red)

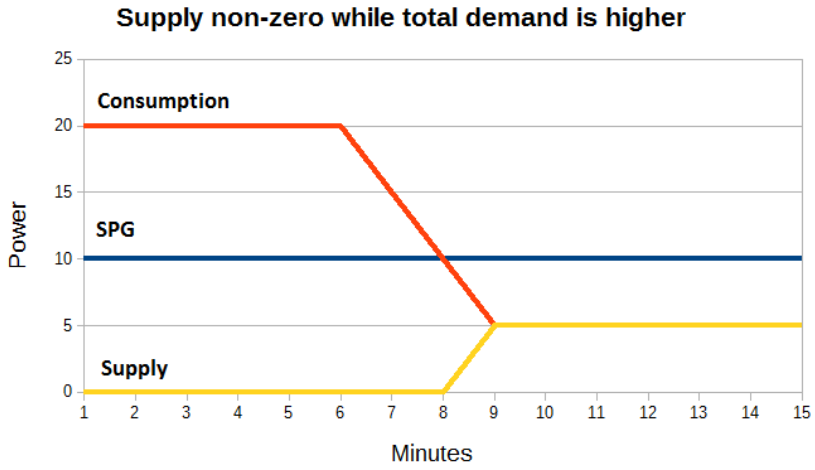


Figure 4.2: An Interval with Non-Zero Supply but also High Demand

amount of solar power produced, but still some solar power can have been delivered back to the supply network. Figure 4.2 shows how this works. The total amount of solar power for this interval is 150, and while the total consumption for the interval is 180 the supply is still 35. The supply line in the graph shows demand when it is negative and supply when it is positive. You can see that when consumption drops below SPG the supply is positive. The results of the Particle Swarm Optimization procedure are shown in Table 4.7 and 4.8 respectively for Random Forests and Echo State Networks.

	ntree	sampsize	mtry	node size
Value	227	9987	13	1

Table 4.7: Optimal Parameters Random Forest for Demand Prediction

The results of these experiments are shown in Table 4.9. Because demand is also dependent on solar power generation, which is hard to predict, it was expected that the errors would be higher in this case. Now it is clear that this is indeed

	Connectivity (%)	Spectral Radius	Reservoir Size
Value	0.37	1.01	1486

Table 4.8: Optimal Parameters ESN for Demand Prediction

the case. What can also be seen is that the difference between the Linear Model and the Random Forest is bigger in this case. Table 4.10 show the p-values for the differences and they are all significant. This means that the Random Forest is the best performer in this case.

Method	RMSE	st.dev.
LM	1559	0
RF	1503	3.47
ESN	1584	27.2

Table 4.9: Results for Demand Prediction

	LM	RF	ESN
LM	-	$t(9) = -51.17$, $p = 2.09 * 10^{-12}$	$t(9) = 2.87$, $p = 0.02$
RF	*	-	$t(9.29) = 9.30$, $p = 5.21 * 10^{-6}$

Table 4.10: Significance of the Results on Demand

4.2.3 24-hour Ahead Supply Prediction

For supply prediction mainly weather predictors were used. The supply is the amount of solar power that the area feeds back to the supply network in each period. This is dependent on the solar power generation and the consumption pattern of the area. That is why the dataset for these experiments contains weather variables that are related to solar power generation and some consumption variables. Table 4.13 lists the predictors used. In Table 4.11 the optimal parameters for the Random Forest are presented and in Table 4.12 the optimal parameters for the Echo State Network are presented.

	ntree	sampsize	mtry	node size
Value	366	8786	6	2

Table 4.11: Optimal Parameters Random Forest for Supply Prediction

	Connectivity (%)	Spectral Radius	Reservoir Size
Value	10	0.69	941

Table 4.12: Optimal Parameters ESN for Supply Prediction

Name	Description
Unix Time	POSIX time in seconds since 1970-01-01
Solar Duration	Duration of sunshine (in units of 0.1 hours)
Global Irradiation	The power of the sunshine
Relative Humidity	How much moisture was in the air
Solar Panels	The number of solar panels in the area
NumberOfHouses	The number of houses in the area
Supply1d	The solar power supplied 24 hours ago
NetUsage1d	The amount of power used 24 hours ago
PvProduced1d	The amount of solar power produced 24 hours ago
NumberOfHouses1d	The number of houses in the area 24 hours ago
Global Irradiation 1d	The power of the sun on the interval 24 hours ago
Supply1w	The solar power supplied one week ago
NetUsage1w	The amount of power used one week ago
PvProduced1w	The amount of solar power produced one week ago
NumberOfHouses1w	The number of houses in the area one week ago
Global Irradiation 1w	The power of the sun on the interval one week ago
MonthOfYear	The number of the month (1 - 12)
PeriodCode	The number of the period (1 - 96)

Table 4.13: Predictors for Supply Prediction

Results of these experiments are shown in Table 4.14. The error is a lot higher than in the previous experiments while the variance of the target output in this case is much smaller. The results show little difference between the three methods. The significance levels (Table 4.15) of the results are clear. Random Forest is the best performing method in this case.

Method	RMSE	st.dev.
LM	2518	0
RF	2504	5.32
ESN	2532	23.5

Table 4.14: Results for Supply Prediction

	LM	RF	ESN
LM	-	$t(9) = -8.19$, $p = 1.84 * 10^{-5}$	$t(9) = 1.91$, $p = 0.09$
RF	*	-	$t(9.92) = 3.67$, $p = 4.37 * 10^{-3}$

Table 4.15: Significance of the Results on Supply

4.2.4 24-hour Ahead Solar Power Generation Prediction

The Solar Power Generation prediction experiments are based solely on weather variables and the amount of solar panels and houses. The Sun Duration keeps track

of how long the sun shines. The Global Irradiation gives the amount of energy that reaches the surface. The Humidity is related to how well the sunshine can penetrate the atmosphere. Optimal parameters found for this case are shown in Table 4.16 for Random Forests and Table 4.17 for Echo State Networks.

	ntree	sampsize	mtry	node size
Value	299	8053	3	1

Table 4.16: Optimal Parameters Random Forest for SPG Prediction

	Connectivity (%)	Spectral Radius	Reservoir Size
Value	0.1	0.72	1953

Table 4.17: Optimal Parameters ESN for SPG Prediction

The results shown in Table 4.18 are interesting, because it seems that the Echo State Network and the Random Forest both greatly outperform the Linear Model. Between the Echo State Network and the Random Forest a smaller difference in performance was measured, but the difference is significant according to the test results shown in Table 4.19.

Method	RMSE	st.dev.
LM	3685	0
RF	3062	11.4
ESN	3099	40.8

Table 4.18: Results for Solar Power Generation Prediction

	LM	RF	ESN
LM	-	$t(9) = -173$, $p < 2.2 * 10^{-16}$	$t(9) = -45.5$, $p = 5.95 * 10^{-12}$
RF	*	-	$t(10.4) = 2.70$, $p = 2.13 * 10^{-2}$

Table 4.19: Significance of the Results on Solar Power Generation

Figure 4.3, Figure 4.4, and Figure 4.5 show the predictions of the linear model, a Random Forest, and an Echo State Network respectively plotted with the actual data. This is four weeks of data from our test set. The first half are winter data and the second half (after the black vertical line) are summer data. As you can see the linear model is constantly staying below the maxima and over the minima of the data. The Random Forest and the Echo State Network do a better job in this case.

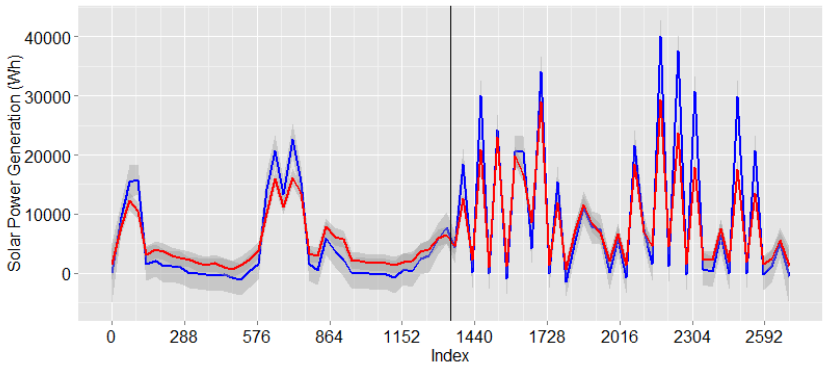


Figure 4.3: Solar Power Generation Prediction with a Linear Model

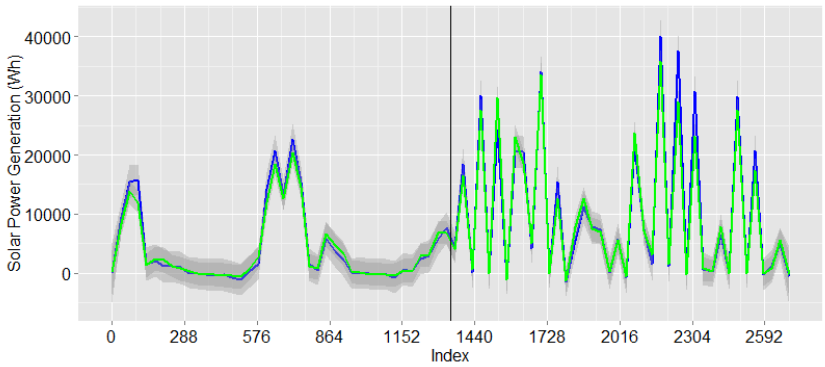


Figure 4.4: Solar Power Generation Prediction with a Random Forest

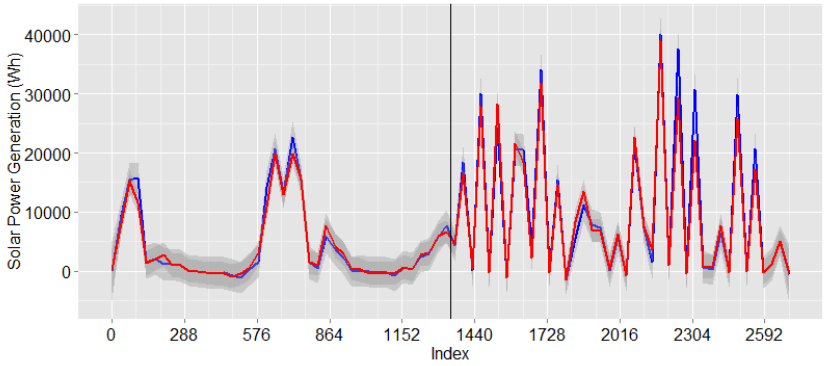


Figure 4.5: Solar Power Generation Prediction with an Echo State Network

Chapter 5

Conclusion and Discussion

In this chapter we will discuss the results presented in the previous chapter and present the conclusions. The implications this study has on the business side of things will be discussed after that. Suggestions for future work will conclude this chapter.

5.1 Conclusions

The answers to the research questions posed at the beginning of this thesis now become evident. In regards to question 1 we could debate whether Random Forest is clearly the best, since in power consumption prediction we see that the linear method was the smarter choice. However, it is shown that Random Forests have been performing well in all cases. So when looking for a one-for-all solution Random Forests would be the best choice guided by the results of this research. For question 2 the results displayed in the previous chapter will give a clear answer on all four cases. We will further elaborate on the exact conclusions in the four seperate cases now.

In the case of power consumption prediction there is not a clear winner. A linear model and a Random Forest are equally good. Of course using Occam's razor we should say that a linear model is better, because it is less complex and takes less training time than a Random Forest.

With the case of power demand prediction the Random Forest managed to outperform the linear model. The difference between the methods are not that great, but the differences are all statistically significant. Demand production is dependent on the consumption and the production of solar power. The production of solar power is dependent on the weather and has a non-linear relationship with some predictors. This might be the cause of failure for the linear model and be something that the Random Forest managed to pick up on.

Seeing the conclusion of the previous case, the linear method should not be the best performer in solar power generation prediction either. If we look at the results of the SPG prediction case, we see that this is indeed the case. Again Random Forests outperform the linear model and the Echo State Network. It is clear however that for some reason all methods have great difficulty with predictions on this variable. There are a couple of possible causes that can be noted about this. First, the lack of

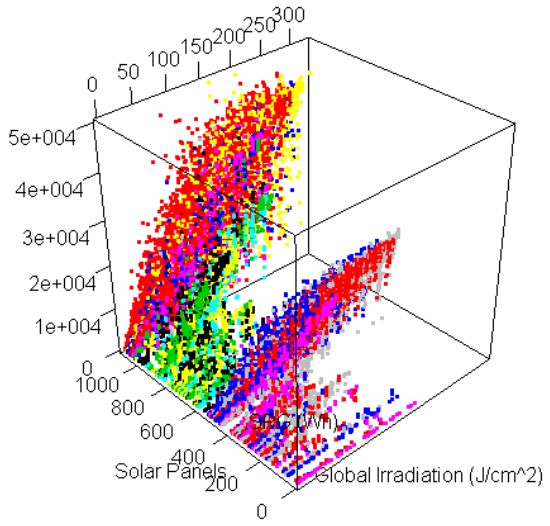


Figure 5.1: Solar Power to Global Irradiation and Amount of Solar Panels

historical values. For this case we used only weather measurements and predictions and did not include historical values of the predicted variable. This could also be the main reason why the linear model performs so badly in this case. The relations between these historicals and the output are definitely linear and they have proven to be very important predictors in the hierarchical linear model as well as the Random Forest method in the cases of consumption, demand and supply prediction.

Second, the predictor - target relationship. It is clear that the chosen predictors are of influence to the SPG in some way. But if we look for instance at the number of solar panels, you will see that sometimes more solar panels means more SPG, but of course only when there is sun in the first place. For some reason the global irradiation in the dataset does not have a clear relationship with the SPG (Figure 5.1). Even when we take the amount of solar panels into account the data does not show a clear relation between global irradiation and SPG. This can have several reasons. The global irradiation data used has not been measured locally, while it is a variable that is hard to interpolate. The only measurements we had available were of two stations quite some distance from the area of interest. Of course a cloud slowly moving over the area has a very distinct moment of solar obscuring and when interpolating data from two stations there is no way of knowing of clouds that move in between these stations over our area of interest. Perhaps also there were solar panels switched off sometimes when there are no inhabitants of a residence. Since no data were available for the residency variable, we can not be sure that the solar panels were actually delivering to the network for their whole life span.

The last case to make a conclusion for is the solar power supply prediction. The Random Forest gives the best prediction results, but overall performance of all methods is bad. This case has a target output which has less variance than in the other cases and one would hope the root mean squared error would be much less than the others in such a case. This time we can even say that the Echo State Network performs just as well as the linear model, which is a novelty in this study. Probably this is caused by the higher complexity of this problem. What the system

needs to work out is the solar power generation, and the consumption, and the general amount of solar power which is not used directly by the user each interval. This is evidently a too complex problem to tackle with the predictors we have used.

5.2 Implications for Business

As this research has been commissioned by CGI it is of course paramount to discuss the implications of these results for CGI. The main conclusion of this research is that the ESN does not outperform the other two methods and is thus not as useful for applications. For CGI this is a positive outcome because the ESN is a patented idea in some countries. Furthermore the regression model used in this research is a very basic one and leaves room for possible improvements within the spectrum of linear models. The linear model is also very interesting for business cases for another reason. It is much easier to implement domain and expert knowledge into the model with a linear model. Random Forests leave less room for this and with ESNs there is not an easy way to do this.

5.3 Future Research

Research never ends. We will constantly discover new questions by answering old ones. This research has answered some questions but also raised others. One question is whether this dataset is adequate enough. As discussed before there are a lot of discrepancies in the database that had to be taken care of before using the data. Also some probably relevant data is not available, like whether houses were occupied and whether solar panels in unoccupied houses were still delivering solar power to the network or not. Solving this challenge will make the data more accurate and will probably have a positive effect on the performance of all methods used.

A recent study was done where training neural networks was done using only similar samples from the dataset to the sample that was predicted (Vonk et al., 2012). This study used only feed-forward neural networks with a single hidden layer for forecasting. For each input sample they chose training samples that were similar to the current input based on mutual information with the output to train a new neural network. It appears from that study that the performance of neural networks can be greatly increased by pre-selection of similar input samples. Perhaps the same training data selection method could greatly improve the result of the Echo State Networks too. A study that looks into that should try to figure out whether the probable increase in overall training time would weigh out against the increase in accuracy.

As a last point of advice I would like to repeat a statement by Gross and Galiana (1987): 'Guarantees for real life performance can only be given after testing these systems in a real online environment for an extended period of time.' So the results found in any research on short term load forecasting should be compared in a realistic setting before any dependence on the forecasts can be justified.

Bibliography

Z.A. Bashir and M.E. El-Hawary. Applying wavelets to short-term load forecasting using pso-based neural networks. *Power Systems, IEEE Transactions on*, 24(1): 20–27, Feb 2009.

G.E.P. Box and G.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 3rd edition, 1994.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine learning*, pages 5–32, 2001.

L. Breiman, J. Friedman, R. Olshen, C. Stone, D. Steinberg, and P. Colla. *CART: Classification and Regression Trees*. Belmont, Calif.: Wadsworth International Group., 1984.

B.G. Brown, R.W. Katz, and A.H. Murphy. Time series models to simulate and forecast wind speed and wind power. *J. Climate Appl. Meteor.*, 23(8):1184–1195, Aug 1984.

A. Bruhns, G. Deurveilher, and J.-S. Roy. A non linear regression model for mid-term load forecasting and improvements in seasonality. In *Proceedings of the 15th Power Systems Computation Conference*, pages 22–26, 2005.

W. Charytoniuk, M.-S. Chen, and P. Van Olinda. Nonparametric regression based short-term load forecasting. *Power Systems, IEEE Transactions on*, 13(3):725–730, Aug 1998.

S.-T. Chen, D.C. Yu, and A.R. Moghaddamjo. Weather sensitive short-term load forecasting using nonfully connected artificial neural network. *Power Systems, IEEE Transactions on*, 7(3):1098–1105, Aug 1992.

W.R. Christiaanse. Short-term load forecasting using general exponential smoothing. *Power Apparatus and Systems, IEEE Transactions on*, PAS-90(2):900–911, March 1971.

R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pages 39–43, Oct 1995.

European Council. On the promotion of the use of energy from renewable sources and amending and subsequently repealing directives 2001/77/ec and 2003/30/ec.

E.A. Feinberg and D. Genethliou. Load forecasting. In J. H. Chow, F. F. Wu, and J. Momoh, editors, *Applied Mathematics for Restructured Electric Power Systems*, Power Electronics and Power Systems, pages 269–285. Springer US, 2005.

G. Gross and F.D. Galiana. Short-term load forecasting. *Proceedings of the IEEE*, 75(12):1558–1573, 1987.

M.T. Hagan and S.M. Behr. The time series approach to short term load forecasting. *Power Systems, IEEE Transactions on*, 2(3):785–791, Aug 1987.

H. Hahn, S. Meyer-Nieberg, and S. Pickl. Electric load forecasting methods: Tools for decision making. *European Journal of Operational Research*, 199(3):902–907, 2009.

T. Haida and Shoichi M. Regression based peak load forecasting using a transformation technique. *Power Systems, IEEE Transactions on*, 9(4):1788–1794, Nov 1994.

G.T. Heinemann, D.A. Nordmian, and E.C. Plant. The relationship between summer weather and summer loads-a regression analysis. *Power Apparatus and Systems, IEEE Transactions on*, (11):1144–1154, 1966.

H.S. Hippert, C.E. Pedreira, and R.C. Souza. Neural networks for short-term load forecasting: a review and evaluation. *Power Systems, IEEE Transactions on*, 16 (1):44–55, Feb 2001.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989.

E.B. Hunt, J. Marin, and P.J. Stone. Experiments in induction. 1966.

H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

J. Kennedy. Particle swarm optimization. In C. Sammut and G.I. Webb, editors, *Encyclopedia of Machine Learning*, pages 760–766. Springer US, 2010.

E.A.M. Klaassen, E. Veldman, J.G. Slootweg, and W.L. Kling. Energy efficient residential areas through smart grids. In *Power and Energy Society General Meeting (PES), 2013 IEEE*, pages 1–5. IEEE, 2013.

E. Kyriakides and M. Polycarpou. Short term electric load forecasting: A tutorial. In *Trends in Neural Computation*, pages 391–418. Springer, 2007.

M. Lukoševičius. A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade*, pages 659–686. Springer, 2012.

H. Lütkepohl. Univariate time series analysis. *Applied Time Series Econometrics*, 2004.

H. Mori and H. Kobayashi. Optimal fuzzy inference for short-term load forecasting. *Power Systems, IEEE Transactions on*, 11(1):390–396, Feb 1996.

H. Mori, N. Kosemura, K. Ishiguro, and T. Kondo. Short-term load forecasting with fuzzy regression tree in power systems. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, volume 3, pages 1948–1953 vol.3, 2001.

P. Newbold and C.W.J. Granger. Experience with forecasting univariate time series and the combination of forecasts. *Journal of the Royal Statistical Society. Series A (General)*, 137(2):pp. 131–165, 1974.

L.D. Paarmann and M.D. Najar. Adaptive online load forecasting via time series modeling. *Electric Power Systems Research*, 32(3):219 – 225, 1995.

A.D. Papalexopoulos and T.C. Hesterberg. A regression-based approach to short-term system load forecasting. *Power Systems, IEEE Transactions on*, 5(4):1535–1547, Nov 1990.

J.H. Park, Y.M. Park, and K.Y. Lee. Composite modeling for adaptive short-term load forecasting. *Power Systems, IEEE Transactions on*, 6(2):450–457, May 1991.

R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, 7 1955.

G.C. Reinsel. *Elements of Multivariate Time Series Analysis*. Springer Science & Business Media, 2003.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

J.W. Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003.

S. Tzafestas and E. Tzafestas. Computational intelligence techniques for short-term electric load forecasting. *Journal of Intelligent and Robotic Systems*, 31(1-3):7–68, 2001.

B.M.J. Vonk, P.H. Nguyen, M.O.W. Grond, J.G. Slootweg, and W.L. Kling. Improving short-term load forecasting for a local energy storage system. In *Universities Power Engineering Conference (UPEC), 2012 47th International*, pages 1–6, Sept 2012.

P.J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990.

J. Yang and J. Stenzel. Short-term load forecasting with increment regression tree. *Electric Power Systems Research*, 76(9Ű10):880 – 888, 2006.