Evaluating methods for time-series forecasting; Applied to energy consumption predictions for Hvaler (kommune)

Master's Thesis in Computer Science

Sopiko Gvaladze
Supervisor Prof. Roland Olsson

May 15, 2015
Halden, Norway

Abstract

This thesis presents comparison of different techniques for electricity consumption data of the one station in Hvaler. This work represents the description of the several approaches for similar problems, general description of statistical and machine learning models and applying those models for specific time-series data. The methods can be grouped into two parts as statistical and machine learning. This work also presents the discussion of main challenges when dealing with time-series data. The following models are applied to electricity consumption data: auto-regressive integrated moving average (ARIMA), linear regression, decision and model trees, regression trees, support vector machines(SVM), k-nearest neighbor, neural network. Even though statistical methods required much more involvement during experiments from observer, it has performed worse than machine learning models. The best performance was achieved by the Nonlinear autoregressive neural network with 40 hidden neurons and 24 time delay. All the other models outperformed ARIMA.

Keywords: Time-series prediction, forecasting, ARIMA, linear regression, neural network, NAR, model trees, regression trees, k-nearest neighbor, statistical methods, Support vector machines (SVM)

Acknowledgments

I would like to express my sincere gratitude to my supervisor Prof. Roland Olsson who guided me all the time during doing my research and experiments. His advices were crucial during writing master thesis.

Also I would like to thank Esmart company and people who work there, for delivering the electricity consumption data set, which is the main topic of this work.

I thank Professor Øystein Haugen for helping me to finish the thesis and structure the thesis in the best way available. Thanks to him the thesis is better readable and connected.

I thank my friend and group mate Ksenia Dmitrieva for discussions and advices about different topics of this thesis, also to encourage and support me during my master study.

Prerequisites

Because the problems that this thesis deals with cover so many different aspects of computer science, it is not possible to go into every little detail of all the subjects that are mentioned. Consequently, it is assumed that the reader has a basic knowledge in discrete mathematics, for example understanding the concept of function approximation and interpolation as well as the basic understanding of machine learning terminology like prediction or model accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays rapidly developing technology makes it possible to save more and more information. which gives researchers the chance to analyze them and make predictions. Forecasting the data can help people to manage their resources better. This leads us to big data and machine learning. Some good examples are rainfall forecasting in Australia using neural networks [15], electricity Consumption in Turkey with ARIMA model [4], diesel consumption in Turkish agriculture [28] and so on. Accurate predictions can raise awareness of upcoming trends and help people to prepare for them. For example after analyzing the diesel consumption in Turkish agriculture researchers suggested that some alternative energy resources should be found as according their calculations up to 4 million tons of diesel will be consumed in 2020. This kind of data is specific and is called time-series. In general the order of points in a historical data set is random and does not make any sense while in time series the order of the observations is important [20]. This work presents different techniques for time series analysis for specific data set.

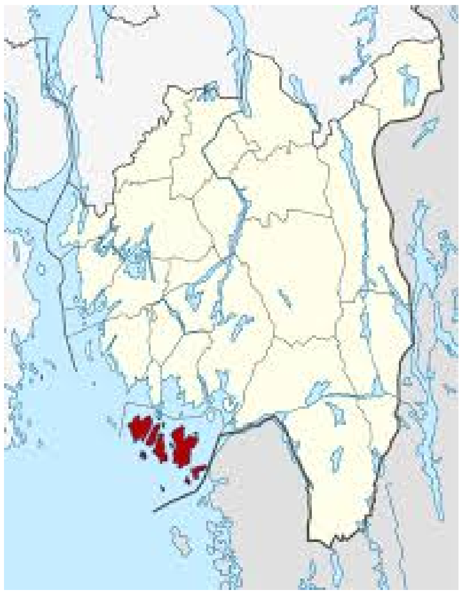

Figure 1.1: Hvaler municipality

1.1 Background and motivation

Recently most of the buildings in Norway have been equipped with smart meters, which take track of the hourly electricity consumption. According to the plan all electricity customers will receive new, intelligent power meters before January first 2019. Gathering and analyzing this data can help both electricity consumers and power companies to manage resources better. Esmart Systems is one of the companies which introduces and continuously working on development of energy management solutions. [1] My goal here is to analyze different methods for this specific type of data, as according to related work the same method implementations can perform differently for different data sets. It is summed up electricity consumption data set for one substation in the Hvaler kommune (Figure 1.1). This work presents the discussion of the challenges which arises during time-series analyses and forecasting. Also this thesis includes the review of experiments and researches which was done to solve the similar problems.

1.2 Research question and method

1.2.1 Research question/Problem statement/Objectives

Going through the related work of what methods have been used and which of them are successful, I have discovered one main tendency: a performance of the specific model mainly depends on the data set. It depends on the complexity of the data, if there is any significant trend, seasonality, also on the forecasting horizon [this term will be explained in the chapter 2] and some other data features. My goal is not to find generally good model for all kinds of data. I concentrate on a concrete data set, which I have from Esmart System and it is summed up hourly electricity consumption for the households, for one electricity substation. Some models which are discussed in chapter 3 are not generally implemented to work with time-series data. To apply those models for this data set, data should be modified. Taking in mind all these issues, I want to try out different models for this data set and find an optimal one. This raises the following research question

RQ 1 *Does machine learning methods perform better than statistical methods for this electricity consumption data?*

More precisely:

Which model has the best performance on the electricity consumption data set and why?

To answer the first question the data features need to be taken into account. This leads to the second research question:

RQ 2 *How a data should be preprocessed to improve the forecasting accuracy?*

The data prepossessing depend on both the data and the model, which is to be applied to it. To answer this question the optimal number of observations should be found for each model, that will be the input at each step.

As this work is concentrated on electricity consumption data set, it is reasonable to answer the following question:

To what extent can we recognize the trend, seasonality or noise in the electricity consumption data ?

1.2.2 Method

The main focus of this work is to find an optimal model for this electricity consumption data set. To do so, the data itself should be analyzed. For this purpose I will be using various statistical methods to discover whether there is trend and seasonality in the data.

To answer the research question and find the model which performs best on the electricity consumption data, on the first place the following issue should be clarified:

What criterion should be used when comparing different methods for the specific data set?

To estimate the model's performance the following measures are used:

- correlation coefficient
- root mean squared error
- mean absolute error

Correlation is a statistical technique and it shows if the pairs of variables are related and how strong is that relation. Correlation coefficient r ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related. Correlation technique works best when the relationship is linear: when one variable is changing proportionally to another.

For evaluating numeric prediction root mean-squared error is the principal and most commonly used measure:

$$\sqrt{\frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n}} \tag{1.1}$$

Here $p_1, \ldots, p_n$ are predicted values on the test data, $a_1, a_2, \ldots, a_n$ are actual values, and n is the number of instances. Mean absolute error is an average of the modules of the individual errors

$$\frac{|p_1 - a_1| + ... + |p_n - a_n|}{n} \tag{1.2}$$

1.3 Report Outline

The rest of the thesis has the following structure:

Chapter 2 describes statistical models, more precisely Box and Jenkins model. This is also known as autoregressive integrated moving average (ARIMA) model. ARIMA is a statistical model for time-series analysis. Estimation of the ARIMA model for a concrete data set is a long process. It involves checking the data for stationarity, differencing, defining the model variables and checking whether or not the model can be further improved.

The first part of the chapter 3 discusses the compatibility of the time-series series data with machine learning models. The second part briefly describes the following machine learning models: linear regression, regression trees, model trees and rules from model trees, also K-nearest neighbor, support vector machines (SVM) and neural networks (NN).

Chapter 4 describes the related work. What was done before the same problem. What was the challenges. Also describes some real-life situations were accurate forecasting can make a big difference.

Chapter 5 represent data description and analysis. In chapter 5.2 the data is analysed and as shown it has trend and seasonality. After this ARIMA model is estimated for electricity consumption data. Also all the machine learning models that are described in the chapter 3 are applied to the electricity consumption data set. For each model experiments are done with a different values of the model variables to find the best fitting model. For all the models rmse and mae is calculated on the test data. Chapter 6 sums up the final results of the experiments and in this chapter the models are ordered by their performance on the electricity consumption data.

Chapter 2

Statistical methods

The first research question demands comparison of the statistical models over machine learning models. This task has two parts, one of which demands applying statistical models to the electricity consumption data set. In the statistical analysis of the time-series data, the Box and Jenkins method is the most popular and advance.

This chapter describes the statistical model for time-series data. The following models are described here: autoregressive (AR), moving average (MA), autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA). ARMA and ARIMA models have been developed by statisticians George Box and Gwilym Jenkins. Those models are ofter called as Box and Jenkins models. The process of finding the best fitting Box and Jenkins model includes the data analysis. During model estimation data features like seasonality and trend is measured and taken into account. So by finding the best fitting Box and Jenkins model for this data set also means that the second research question will be answered about the existing trend and seasonality in the data. To get this information about the data autocorrelation and partial autocorrelation functions are used. As George Box and Gwilym Jenkins are the authors of these models this chapter has only one source [5]. This chapter is a brief review of the George Box and Gwilym Jenkins book.

2.1 Introduction to the main concepts of statistical time series analysis

What is a time series? This is a sequence of observations that occurs sequentially in time. Time series examples are electricity consumption which is registered in every hour, the quantity production of some product in a factory, the amount of milk delivered in every month. A time series data is captured in economics, engineering, natural sciences and social sciences as well. The main characteristic of a time series is that there is a dependency between the observations that are close to each other. The goal of a time series analysis is to find this dependency and formulate it with mathematical equations. In case if we have a mathematical model of a time series we can forecast future values according to a historical data. This process is split into the following subprocesses: building, identifying, fitting, and checking models of a time-series. There are two types of time series: discrete and continuous. A time series data is discrete if observations are held after beforehand determined time intervals, while when observations are held continuously a time series is continuous.

During a time-series analysis the following five practical issues might arise:

1. Forecasting: Planning is a very important step in economics, business, industry and in many other areas. This process has the following structure: Using historical data at a time moment t, the process can predict the observation value for the time $t+l$. l is called lead time and its value depends on a task. For example if we have the hourly observation of electricity consumption and we want to predict the consumption for particular hour for the next day, lead time will vary from 1 till 24, depending on the hour we want to predict. Let's consider the discrete case when observations are held in equivalent time intervals. Let's define the observation value for t moment with z_t, and the past observations with $z_{t-1}, z_{t-2}, z_{t-3}, \dots$, then historical data can be used to predict the future value when lead $l = 1, 2, 3 \dots$. For example if we have monthly observations the lead will vary from 1 till 12. Let's define by $\hat{z}_t(l)$ forecasting value at $t+l$ moment when the forecast is made at the origin t. The function that forecasts at t time moment a future values according to the past and present observations is called forecast function at origin t. The goal is to find the forecasting function, so that the square of the difference $\hat{z}_t(l) - z_{t+l}$ between a forecasting value and a real observation is minimal for any value of l.
   Another important issue during forecasting is to specify accuracy. Also it is very important to estimate the risk probability. This can be done by calculating upper and lower probability limits.
2. Estimating the transfer function.
3. Analysis of the effect of unusual intervention events in systems: Sometimes temporary external event can affect greatly the observationz_t. For example temporary sales can affect the number of things being sold. This problem can be dealt in the following way; for temporary external events Boolean function is added which is 1 when the event occurs, and 0 otherwise.
4. Analysis of a multivariate time-series: Sometimes in a time-series data there may exist a cluster of several related variables. For example, let's consider once more electricity consumption when observations take place in every hour. In this case observations can be grouped into 24×1 vector $z_t = (z_{1t}, z_{2t}, \dots, z_{24t})$ where z_t is 24×1 time-series vector at time t. So that we don't consider individual observation z_t, instead we have multivariate or vector time-series.
5. Discrete Control Systems: One of the examples of discrete control systems is engineering properties control. There are two kinds of properties control: feedback and feedforward. For example, if we know that the input has error or noise and we can't get rid of it, we can change properties, so that there is not big error in output. This would be feedforward control. During the feedback control we change the properties according to an output error. We can use feedback control even when we don't know exactly the amplitude of a noise, while feedforward control can't be used in this case.

2.1.1 Stochastic and deterministic dynamic mathematical models

Let's consider some process. When we can estimate the mathematical equation which determines the process without error, the process is called deterministic. But in practice we deal with processes that are influenced by many factors and some of the factors are impossible to formulate mathematically. In this case we can estimate the future value with some probability by finding the upper and lower limits. Such a process is called probability or stochastic process. A time-series data with N observations that we want to analyze is a finite realization from infinite data set, generated by a stochastic model.

2.1.2 Stationary and nonstationary stochastic models

A stochastic process is said to be stationary if static properties like mean and variance don't vary over time. Otherwise a process is nonstationary. Stationary processes are much easier and lots of analysis is done for them, while in practice we meet nonstationary processes more often.

2.1.3 Linear filter model

During a time-series analysis backward and forward shift operators are often used. The backward shift operator is defined in the following way

$$Bz_t = z_{t-1} \tag{2.1}$$

The inverse operator of a backward shift is a forward shift operator

$$F = B^{-1}$$

$$z_t = Fz_{t-1} \tag{2.2}$$

The backward difference operator is defined in the following way

$$\nabla z_t = z_t - z_{t-1} = (1 - B)z_t \tag{2.3}$$

White noise is a set of independent observations $a_1, a_2, \dots$ with mean zero and a constant variance σ_a^2. a_i observations are also called random shocks. An observable time series can be generated from white noise by linear filter, where

$$z_t = \mu + a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \dots = \mu + \psi(B)a_t \tag{2.4}$$

Here B is backward shift and

$$\psi(B) = 1 + \psi_1 B + \psi_2 B^2 + \dots \tag{2.5}$$

ψ is called the transfer function of the filter. If $\psi_1, \psi_2, \dots$ are absolutely summable $\sum_{t=1}^{\infty} |\psi| < \infty$ than the process will be stationary and the mean of the time series $z_1, z_2, \dots$ will be μ, otherwise the process will be nonstationary.

2.1.4 Autoregressive Models

Let's consider $z_1, z_2, \dots$ observations and $\bar{z}_t = z_t - \mu$.

The stochastic process is said to be an autoregressive (AR) process of order p if it is defined by the following equation

$$\bar{z}_t = \phi_1 \bar{z}_{t-1} + \phi_2 \bar{z}_{t-2} + \dots + \phi_p \bar{z}_{t-p} + a_t \tag{2.6}$$

The equation (2.6) can be rewritten in the following way

$$\phi(B)\bar{z}_t = a_t \tag{2.7}$$

Here

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \tag{2.8}$$

is an autoregressive operator of order p. There are $p+2$ unknown variables $\mu, \phi_1, \phi_2, \ldots, \phi_p, \sigma_a^2$ in this model. AR is a specific case of a linear filter. AR process can be stationary as well as nonstationary. From the (2.7) we can obtain

$$\bar{z}_t = \phi^{-1}(B)a_t = \psi(B)a_t = \sum_{j=0}^{\infty} \psi_j a_{t-j} \tag{2.9}$$

The last equation shows that we can consider autoregressive process as an output z_t of a linear filter, where the filter function is $\psi(B)$ and input is a white noise a_t.

2.1.5 Stationary conditions for autoregressive process

For an AR process to be stationary, the absolute value of roots of equation $\phi(B) = 0$ must be more than 1 when we consider it as a polynomial of B.

Let's expand $\phi(B)$

$$\phi(B) = (1 - G_1 B)(1 - G_2 B) \ldots (1 - G_p B) \tag{2.10}$$

Here $G_1^{-1}, G_2^{-1}, \ldots G_p^{-1}$ are the roots of a characteristic equation. According to 2.9, we have

$$\bar{z}_t = \phi^{-1}(B)a_t = \sum_{i=1}^{p} \frac{K_i}{1 - G_i B} a_t \tag{2.11}$$

For AR(p) to be stationary it is necessary that $\psi(B)$ is summable for $|B| \leq 1$, which means that $\psi_i = \sum_{i=1}^{p} K_i G_i^p$ weights should be absolutely summable. On the other hand it causes $|G_i < 1|$, $i = 1, 2, \ldots, p$. This means that the roots of characteristic equation should lie outside the unit circle.

2.1.6 Autocorrelation function of autoregressive process

Let's consider autoregressive process of order p. If we will multiply 2.6 equation with $\bar{z}_{t-k}$ we will get the following equation

$$\gamma_k = \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2} + \cdots + \phi_p \gamma_{k-p}, \quad k > 0 \tag{2.12}$$

If we will divide the last equation at γ_0 we will get the equation for autocorrelation function

$$\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + \cdots + \phi_p \rho_{k-p} \tag{2.13}$$

The last equation has the same form as autoregressive stochastic process, but without a random shock a_t.

It can be shown that the general answer of the last equation ρ_k has the following form

$$\rho_k = A_1 G_1^k + \cdots + A_p G_p^k \tag{2.14}$$

Here $G_1^{-1}, G_2^{-1}, \ldots G_p^{-1}$ are the roots of a characteristic equation. When the process is stationary $|G| < 1$. If the roots are real values, ρ_k decays to zero as k grows. This property is often called damped exponential. If a characteristic function has couple roots with complex values, than the autocorrelation function will follow a damped sine wave.

2.1.7 Autoregressive parameters in terms of autocorrelation Yule-Walker equations

When $k = 1, 2, \ldots, p$ in (2.13), AR model parameters can be expressed in terms of $\rho_1, \ldots, \rho_p$

$$\rho_1 = \phi_1 + \phi_2\rho_1 + \cdots + \phi_p\rho_{p-1}$$

$$\rho_2 = \phi_2\rho_1 + \phi_2 + \cdots + \phi_p\rho_{p-2}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

$$\rho_p = \phi_p\rho_{p-1} + \phi_2\rho_{p-2} + \cdots + \phi_p \tag{2.15}$$

(2.15) are called Yule-Walker equations. It can be used to find parameters if we replace ρ_i theoretical autocorrelation by estimated autocorrelation r_i.

2.1.8 Partial autocorrelation function

Let's define with ϕ_{kj} the k^{th} coefficient of j order AR process . Than according to (2.13) we have

$$\rho_k = \phi_{k1}\rho_{j-1} + \phi_{k2}\rho_{j-2} + \cdots + \phi_{kk}\rho_{j-k}, \quad j = 1, 2, \ldots, k \tag{2.16}$$

Using this equation and Yule-walker equations, ϕ_{kj} can be expressed in terms of ρ_i. The quantity ϕ_{kk} regarded as a function of the lag k , is called partial autocorrelation function.

For p order autoregressive process ϕ_{kk} is not zero when $k \leq p$ and is otherwise when $k > p$. This characteristic of the partial autocorrelation function is used to estimate the order of AR process. The partial autocorrelation function can be defined for any stationary process.

2.1.9 Moving average models

Let's consider $z_1, z_2, \ldots$ observations and $\bar{z} = z_t - \mu$. It is said that the following process is described by a moving average (MA) model if $\bar{z}_t$ is a linear combination of p weighted random shocks

$$\bar{z}_t = a_t - \vartheta_1 a_{t-1} - \cdots - \vartheta_q a_{t-q} \tag{2.17}$$

There are no general restrictions for ϑ_i parameters. The moving average process can be rewritten in the following way

$$\bar{z}_t = \vartheta(B)a_t \tag{2.18}$$

where

$$\vartheta(B) = 1 - \vartheta_1 B - \vartheta_2 B^2 - \cdots - \vartheta_p B^p \tag{2.19}$$

There are $p+2$ unknown variables $\mu, \vartheta_1, \vartheta_2, \ldots, \vartheta_q, \sigma_a^2$ in this model. In practical tasks these parameters should be estimated from a given data.

The equation (2.18) shows that we can consider autoregressive process as an output $\bar{z}_t$ of a linear filter, where filter function is $\psi(B)$ and input is a white noise a_t.

2.1.10 Invertibility condition for moving average processes

The invertibility condition for MA(q) process is that roots of $\vartheta(B) = 0$ characteristic equation lie outside the unit circle. The requirement of invertibility ensures that present events are associated with past events in a sensible way. As values of MA(q) observations are finite sum of weighted random shocks, no additional condition is required from parameters in order the process to be stationary.

2.1.11 Autocorrelation function of MA process

$$\rho_k = \begin{cases} \frac{-\vartheta_k + \vartheta_1\vartheta_{k+1} + \cdots + \vartheta_{q-k}\vartheta_q}{1+\vartheta_1^2+\vartheta_2^2+\cdots+\vartheta_q^2} & k = 1, \ldots, q \\ 0 & K > q \end{cases} \tag{2.20}$$

2.1.12 Moving average parameters in terms of autocorrelation

The initial values for moving average parameters can be found by using the (2.20) iteratively and by replacing the theoretical autocorrelation ρ_k with estimated autocorrelation. These estimated parameters are not accurate enough, but it does not matter as they are used as beginning values and they are adjusted during estimation stage.

2.1.13 Duality between autoregressive process and moving average process

There is a duality between autocorrelation and partial autocorrelation functions of moving average and autoregressive processes. The partial autocorrelation function of AR(p) has a cutoff when $k > p$, while it tails off for MA process. On a contrary autocorrelation function of MA(q) has a cutoff for a lag$k$ when $k > q$ and tails for AR processes. These characteristics are used for model identification and for estimation of a process order.

2.1.14 Mixed autoregressive-Moving average models

By uniting the properties of AR and MA processes a new mixed autoregressive-moving average model (ARMA) is obtained. This model is used more in practice than simple AR and MA models. ARMA is described with the following equation

$$\bar{z}_t = \phi_1\bar{z}_{t-1} + \phi_2\bar{z}_{t-2} + \cdots + \phi_p\bar{z}_{t-p} + a_t - \vartheta_1 a_{t-1} - \cdots - \vartheta_q a_{t-q} \tag{2.21}$$

The last equation can be rewritten in the following manner

$$\phi(B)\bar{z}_t = \vartheta(B)a_t \tag{2.22}$$

In this case there are $p+q+2$ unknown variables $\mu, \vartheta_1, \vartheta_2, \ldots, \vartheta_q, \phi_1, \phi_2, \ldots, \phi_p, \sigma_a^2$ in this model. In practice p and q are not more than 2. The roots of $\phi(B) = 0$ characteristic function must lie outside the unit circle for stationary processes. Also for invertibility the absolute values of roots of $\vartheta(B)$ should be more than one.

ARMA (p,q) processes have an infinite moving average representation

$$\bar{z}_t = \psi(B)a_t = \sum_{j=0}^{\infty}\psi_j a_{t-j} \tag{2.23}$$

As well as infinite autoregressive representation

$$\pi(B)\bar{z}_t = \bar{z}_t - \sum_{j=1}^{\infty} \pi_j \bar{z}_{t-j} = a_t \tag{2.24}$$

Here weights $\{\psi_j\}$, $\{\pi_j\}$ should be absolutely summable. If we know one representation of process another can be found.

From the last equation the following equation can be derived

$$\bar{z}_t = \phi^{-1}(B)\vartheta(B)a_t \tag{2.25}$$

This means that we can consider ARMA process as an output $\bar{z}_t$ of a linear filter, where filter function is $\phi^{-1}(B)\vartheta(B)$ and input is a white noise a_t.

2.1.15 Autocorrelation function of ARMA process

It can be shown that the autocovariance function for ARMA(p,q) process may be expressed as

$$\gamma_k = \phi_1\gamma_{k-1} + \cdots + \phi_p\gamma_{k-p} + \sigma_a^2(\vartheta_k\psi_0 + \vartheta_{k+1}\psi_1 + \cdots + \vartheta_q\psi_{q-k}) \tag{2.26}$$

The following formula can be derived from (2.26)

$$\rho_k = \phi_1\rho_{k-1} + \cdots + \phi_p\rho_{k-p}, \quad k \geq q+1 \tag{2.27}$$

So for ARMA(p,q) process there are q autocorrelations $\rho_q, \rho_{q-1}, \ldots, \rho_1$ that depends directly from moving average parameters ϑ as well as from autoregressive parameters ϕ. When $q < p$ than autocorrelation function will be a mixture of damped exponential and damped sine wave and its behavior is determined with $\phi(B)$ polynomial. When $q \geq p$ than $q - p + 1$ autocorrelation values will not follow the general pattern. These features are used during model identification and estimation of p and q values.

2.1.16 Nonstationary models

In practice often time-series data does not have a constant mean, which means that it is not stationary. To describe such processes generalized autoregressive operator $\phi(B)$ is used. This operator has more than one unit root. If it has d unit roots, it can be written in the following way

$$\varphi(B) = \phi(B)(1-B)^d \tag{2.28}$$

So the nonstationary model can be written in the following way

$$\varphi(B)z_t = \phi(B)(1-B)^d z_t = \vartheta(B)a_t \tag{2.29}$$

From this can be derived

$$\varphi(B)\omega_t = \vartheta(B)a_t \tag{2.30}$$

where

$$\omega_t = (1-B)^d z_t = \nabla^d z_t \tag{2.31}$$

This is a very interesting result. It means that nonstationary process can be described with stationary model when the process obtained by d times differencing is stationary. In practice mostly the value of d is not more than 2. The process defined by (2.30)-(2.31) is called autoregressive integrated moving average (ARIMA) process. This is a powerful technique to describe stationary models as well as nonstationary ones. The ARIMA process of (p, d, q) order is described in the following way

$$\omega_t = \phi_1\omega_{t-1} + \phi_2\omega_{t-2} + \cdots + \phi_p\omega_{t-p} + a_t + \vartheta_1 a_{t-1} - \cdots - \vartheta_q a_{t-q} \tag{2.32}$$

Here

$$\omega_t = \nabla^d z_t \tag{2.33}$$

When $d = 0$ the process is stationary.

2.1.17 Stochastic and deterministic trends

Let's consider extension of general ARIMA model

$$\psi(B)z_t = \phi(B)\nabla^d z_t = \vartheta_0 + \vartheta(B)a_t \tag{2.34}$$

What we have added in this model is ϑ_0 constant term. Actually general ARIMA model can represent time series with stochastic trends. When we have ϑ_0 constant term for $d = 1$, it can be shown that ϑ_0 can represent a deterministic linear trend. In general it can represent the deterministic polynomial trend of order d. Often in practice we deal with a time series that has stochastic trend. Unless there is some preliminary assumption that time series has deterministic trend, it is better to have a model with $\vartheta_0 = 0$. This model does not restrict the time series to follow without changing pattern to the trend which was developed in the past.

2.1.18 Nonlinear transformation of time series values

One way to preprocess a time series data is to allow nonlinear transformation of z_t to $z_t^{(\lambda)}$, where $\{\lambda_j\}$ are transformation parameters. This transformation can be dictated from the preliminary knowledge of task or from data itself. For example if we consider sales of a commodity, it is more proper to transform the data and analyze a change in percentage than a change in original data. Both data are nonstationary, but transformed data will be more stable.

2.1.19 Iterated stages in the selection of the model

The model is called mechanical, theoretical if we can define the physical process with mathematical equations. The opposite case is an empirical model. In practice the processes are described by the models that are in-between them. From the beginning the class of models is estimated according to theory. Sometimes it's even possible to estimate the beginning values of parameters. After this according to the data function parameters are fitted and adjusted.

The estimation of a model is an iterative process and it has the following four steps:

1. Estimating the class of models according to the theory.
2. Identifying subclass by reducing the number of search objects; Moreover, according to the theory the beginning values of parameters can be also estimated.

3. At this step beginning values and the parameters are adjusted according to the data.
4. Now the accuracy of the model is checked; if the accuracy is good enough we use the model for forecasting, otherwise we go through the 2,3,4 steps again.

2.1.20 Model Identification

Model identification is a first step of the time series data analysis. During this step presumable values of p,d,q are suggested for ARIMA model, after this the beginning values of model parameters are calculated. The information gathered during model identification is not accurate and it is adjusted during estimation procedure. For identification no exact mathematical formulation exists. Most of analysis is done using graphs.

Identification procedure is done into two stages. Let's consider general ARIMA family

$$\varphi(B)z_t = \phi(B)\nabla^d z_t = \vartheta_0 + \vartheta(B)a_t \tag{2.35}$$

On the first place we want to get stationary process z_t. So we check if the process itself is stationary, if not then we check difference of z_t until we have stationary process. In practice difference operator is used only 1-2 times. After this step we get ARMA model

$$\phi(B)\omega_t = \vartheta_0 + \vartheta(B)a_t \tag{2.36}$$

Here

$$\omega_t = \nabla^d z_t \tag{2.37}$$

Now we analyze the ARMA model, which we obtained after first step. For identification sample autocorrelation and sample partial autocorrelation functions are used. They are used to obtain approximate information about the class of models and about the beginning values of the parameters.

For stationary ARMA(p,q) process, the autocorrelation function satisfies the following equation

$$\phi(B)\rho_k = 0, \quad k > q \tag{2.38}$$

And if $G_1^{-1}, G_2^{-1}, \ldots, G_p^{-1}$ are the roots of $\phi(B) = 0$, we can rewrite ρ_k in the following way

$$\rho_k = A_1 G_1^k + \cdots + A_p G_p^k \quad K > q \tag{2.39}$$

If the process is stationary than the roots of a characteristic function should lie outside the unit circle, that means that $|G_i| < 1$ and ρ_k quickly decreases for large k. If characteristic operator has one real root which approaches unity, then autocorrelation function will not decrease so quickly, but it will fall off almost linearly. The estimated autocorrelation function behaves like theoretical autocorrelation function. So if we look at sample autocorrelation function and it rapidly decreases, we suggest that z_t process is stationary, if not we do the same procedure for $\nabla^d z_t, d = 1, 2, \ldots$ until we get a stationary process. In practice d is 0,1,2 and first 20 estimations are enough for this procedure. The model identification process involves analysis of Autocorrelation and Partial autocorrelation functions. This process is summed up in Table

	AR(p)	MA (q)	ARMA(p,q)
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

Table 2.1: ACF and PACF behavior for ARMA process

Chapter 3

Machine learning methods

This chapter describes the practical and theoretical foundation of the machine learning models which were applied to the electricity consumption data set. Before applying those models to the time-series data, data should be modified. Some techniques are represented in 3.1 section. Linear regression, decision trees, model trees and rules from model trees, lazy learning method like k-nearest neighbour, also support vector machines (SVM) and neural networks are described in chapter 3.2. Those models are analysed in the manner that emphasises each of theirs weakness and strength.

The first research question not only asks for finding the best performing model, but also the explanation why one model outperformed the other. This chapter prepares the background for this purpose. To answer the first research question on the first place how the models work should be described, than according to this what are their weak and strong sides. The same model can work well for one type of data, while have a high error rate for another. So when I talk about model's weak and strong sides, there is not an absolute definition, this depends on the data. The second research question is data oriented. Going through the model implementation can also help to answer this question, as analysing the performance of these models can reveal some features about the data.

3.1 Techniques to make time-series data and machine learning models compatible

Machine learning methods are very popular for classification and prediction tasks. By using historical data we can find out the value of a new variable. In general the order of points in a historical data set is random and does not make any sense. On the other hand in time series the order of the observations is important. When using machine learning methods for time series, first of all data must be modeled and then, if necessary, some preprocessing should be done. After these procedures take place machine learning methods can be used. [20]

3.1.1 Modifying time series data for machine learning approaches

As discussed in [14], before building the model for a time series data forecasting, historical data should be modeled as input/output pairs. For example if the available observations are $\{y\}_{i=1}^{N}$ then the input matrix $[(N-n-1)\times n]$ is

$$\begin{bmatrix} y_{N-1} & y_{N-2} & \cdots & y_{N-n-1} \\ y_{N-2} & y_{N-3} & \cdots & y_{N-n-2} \\ \vdots & \vdots & \ddots & \vdots \\ y_n & y_{n-1} & \cdots & y_1 \end{bmatrix}$$

And the output vector $[(N-n-1) \times 1]$ is

$$\begin{bmatrix} y_N \\ y_{N-1} \\ \vdots \\ y_{n-1} \end{bmatrix}$$

According to the results obtained in [20]] preprocessing has a big influence on the performance accuracy. The same preprocessing method can improve the performance accuracy for one data set, while it can have a negative effect on the other one. This means that each data set needs individual approach during preprocessing.

3.1.2 Preprocessing/modifying time-series data for machine learning approaches

Preprocessing is a very important step when working with time series and forecasting performance depends on it. As discussed in [20] the preprocessing techniques are:

- Taking a log transformation.
- Detrending.
- Deseasonalization.
- Taking lagged values: This means that the time series is preprocessed in such a manner that input variables are lagged time series values $(x_{t-N+1}, \dots, x_t)$ and the output value is the next one..
- Time series differencing: In practice, differencing is applied to data once or twice.
- Taking moving average.

3.1.3 Recentness biased learning

In general, for a time series forecasting, the recent data is more important than the data captured long ago. According to this, the recent data should be highly weighted for a time series analysis. Several approaches had adopted this concept for dynamic time series data analysis. Still it's hard to make any presumptions about the number of observations that can be considered as the recent ones. In case if use many samples are used, it can cause overfitting. On the contrary, if we use few samples it might not be enough for building an adequate model.

Since the 1980s, forgetting factor was used in many researches to deal with these problems. For the first time it was used in the control theory. After that it was used for a time series forecasting as well. According to experiments the forgetting factor can improve performance accuracy. (see [13]).

3.2 Models

3.2.1 linear regression

The output of linear regression models is the sum of the attribute values, where weights are applied to each attribute before adding them together. The goal is to come up with proper values for the weights, so that the model's output matches with the desired output. The input and output attribute values should be all numeric. However, Linear models are good and working accurately only when data is separable by line. If not so, the best-fitting straight line will be found, where "best" is interpreted as the least mean-squared difference. From all the algorithms that we are using, Linear Model is the simplest one in general it has the lowest accuracy [[30], p. 62, p. 125] [1].

3.2.2 Decision trees, regression trees and model trees

Decision tree learning is a function approximation algorithm where learned function is a decision tree. It is probably machine learning workhorse as it is widely used in practice and also is used many times to understand a general concept of machine learning itself. Decision trees are often expressed as upside down tree graphs as is shown at Figure 3.1. Generally, decision trees describes disjunction of conjunctions of the attributes and the attribute values of instances. The decision tree can be also described using if-else rules. This feature makes decision trees very popular, as if-else rules are easy to interpret for humans [19]. Examples of the areas where decision trees are successfully used is medical diagnosis and loan applications. Also it was used by NASA in 1995 for classification of the Sky Survey. Decision trees are originally developed to solve classification problems, when instances have only nominal attributes. But couple of features has been added to the successor versions: dealing with numeric attributes, predicting numeric value, dealing with missing values, capacity to prone the tree, which is related to overfitting problem. There are many decision tree algorithms. Some of them are: ID3, CART, C4.5, C5.0, ASSISTANT. C4.5 (Quinlan 1993) is a successor of ID3 (Quinlan 1986) algorithm [19]. Its implementation is free and is available since 1993 on C [30]. In additional to ID3, it can deal with numeric attributes and with missing values. It has 25% confidence interval. Overfitting is an issue with this algorithm, as sometimes this algorithm does not prune enough. It is implemented in Weka as J48. C5.0 is a commercial version and the implementation is not available. But as observations showed it performs not significantly better.

Let's say we have a decision tree. The question is how decision tree is used for prediction or classification. What does the forecasting process looks like with decision trees? It can be described as a sequence of binary selections according to the tree structure. The tree classifies instances by sorting them from the root to the corresponding node and the value of the leaf is a classification of the instance [19]. The starting place when classifying an instance is a root. The instance attribute values are tested and according to it is made decision which branch to choose. This process is repeated for subtrees recursively until the instance reaches the leaf. As an example, let's discuss the simple classification problem which shows the main idea of decisions trees and is easy enough to understand. This problem is used as a showing example in [19] and [30]. This decision tree is built for a classification problem [Figure 3.1]. The task is to classify the new instance whether

[1]The part of the description of this model is taken from the project in machine learning. Those project are available on CD

the tennis match will be held or not. The data has five attributes : *outlook*, *temperature*, *humidity*, *windy*, and *overall likelihood* and *play*. The target attribute is *play* and it can be classified either as *yes* or *no*. For example a new entry $(outlook = Sunny, Temperature = cool, humidity = high, Windy = true)$ will be classified as $play = yes$.

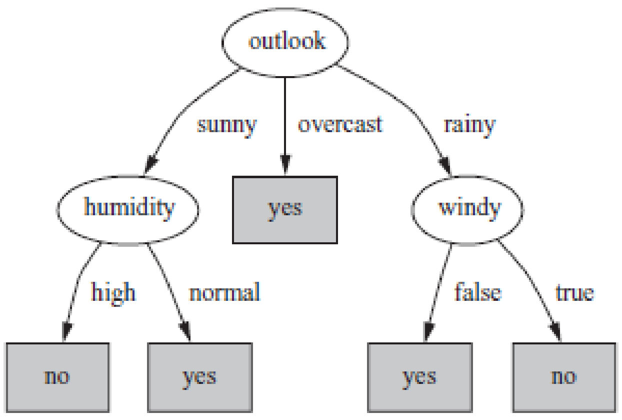

Figure 3.1: An example of a decision tree [30] p. 103]

Build a tree

First step is to build a tree. The main principle of building classification or regression tree is a simple divide-and-conquer algorithm using information gain reduction.

Let's discuss this algorithm briefly for classification problem as decision trees are originally designed for predicting categories rather than numeric quantities. Regression tree is a modification of decision tree.

Each instance consists from attributes and target value. The algorithm can be formed recursively. First we take one of the attributes and place it at the root node and make one branch for each possible value. Doing it data is splited into subsets according to every value of the attribute. The same process is repeated recursively for each branch using only the instances that are represented by it. The developing of the branch is finished when all the instances have the same target value.

The question is here what kind of feature is used when deciding which attribute to place at the node. Information gain and Entropy are used to solve this problem. Entropy of S attribute when target attribute can take c different values is defined by the following formula [[19] p. 57]

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i \tag{3.1}$$

Here p_i is a proportion of i^{th} examples in S. Entropy measures an impurity of the training data. Let's try to understand the idea behind the Entropy definition. It is used in information theory and it shows minimum number of bits of information that needs to be encoded according to the classification of the feature S. What does it means? Let's say that we have a playing tennis example again and we have 10 instances. To demonstrate

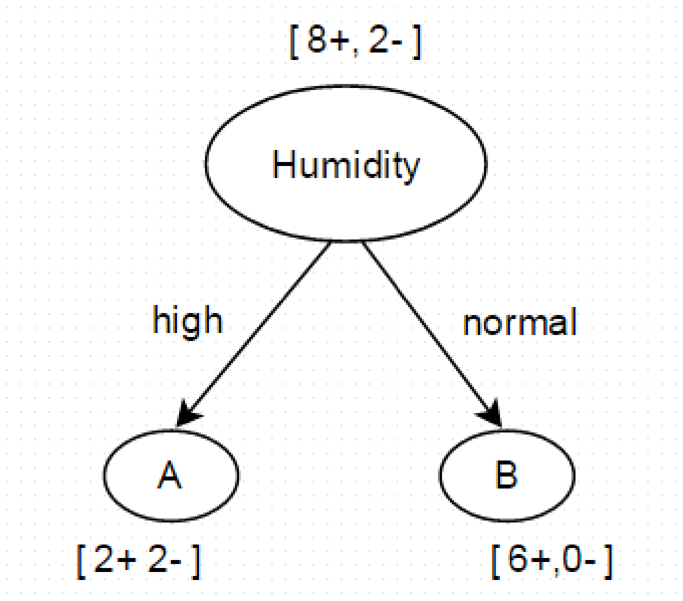

Figure 3.2: Humidity node.

the idea behind the entropy let's calculate the entropy of the attribute humidity which is shown on the Figure 3.2.

$$Entropy(Humidity) = Entropy([8+, 2-]) = -0.8 * \log_2 0.8 - 0.2 * \log_2 0.2 = 0.7219$$

As I have mentioned above, the entropy measures the impurity of the data. What does it mean? It is intuitive that the most impure case when the target variable can take only two different values is when the half of the instances are positive and another half are negative. In this case the entropy reaches its maximal value and it equals to 1. On the other hand the most pure data is when all of the instances are either negative or positive. In this case the entropy reaches its minimal value and it equals to 0.

Now, after measuring the entropy we have an information about impurity of the data. This finding can be used to detect the effectiveness of the attributes. The information gain is used for this purpose. The information gain of an attribute A, for the set of the instances S is defined by the following formula [[19] p. 57]

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{3.2}$$

Here $Values(A)$ define all the possible values for attribute A and S_v defines the subset of S, where for every instance the attribute A equals to v. After calculating information gain for all the attributes, the one with the biggest value is chosen to put it to the root.

Pruning the tree

One of the practical issue of ID3 algorithm is overfitting. It is said that a tree overfits the training examples if some other tree that fits the training examples less well actually performs better over the test and/or validation data. Overfitting is a serious problem and it addresses to model generalization issue. ID3 allows the tree to grow as big and deep as necessary to represent every instance. It fits training set to well and that's why can't perform well on new instances. The experimental study showed that overfitting on average decreased the performance accuracy by 10-25%. [[19] p. 68] This issue is solved in C4.5 algorithm. Regression tree as well as C4.5 is using reduced error-pruning to overcome overfitting. It is a post-pruning technique. First the data is splited into training and validation sets. The tree is built for training data while validation set is used for pruning. Every node of the tree is a candidate of pruning. Pruning a decision node means to remove the subtree rooted at that node and make it a leaf node, then assign it the most common classification of the training examples from those instances which are represented by the node. Pruning is done only if pruned tree's performance does not decrease on the validation set. Also on the first place those nodes are pruned which cause the most improvement in performance. [[19] p. 70]

Regression trees, model trees and rules

Let's say we want to predict numeric values. For this purpose the same kind of tree can be used as discussed before. The difference is that the tree will have numeric values into the leafs and its value will be the average of all the training instances that is sorted by that leaf. Those king of trees are called regression trees, because it predicts numeric quantity.

Regression Trees are decision trees with averaged numeric values at leaves. In Weka Regression Tree algorithm is implemented as REPTree. REPTree builds the tree using information gain like decision trees does and it prunes the tree using reduced error-pruning. The regression tree is much more complex than Linear regression, but also it is more accurate, because the simple linear model can't represent normally complex data. [[30], p. 67]

Model trees combine regression equations with regression trees. Instead of single predicted value, leaves of Model trees contain linear expressions. Model trees are much more complex than regression trees and are smaller too even though, in general they perform better then regression trees, as they can express more complex relations between the data that regression trees can. Model trees perform best on data with mostly numeric attributes. In Weka Model trees are implemented as M5P algorithm. [[30], p. 67]

Rules are easy to interpret for humans, that's why it's a popular alternative to decision trees. Preconditions of the rules can be formulated not only with simple conjunctions, but also with general logical expressions. M5Rules are generated from Model Trees. First algorithm builds a model tree from all the data, then picks one of the leaves and makes it into a rule; after it removes the data covered by that leaf, then repeats the process with the remaining data Its accuracy is almost similar to the Model Trees accuracy. [[30], p. 67, p. 259]

3.2.3 K-nearest neighbor

K-nearest neighbors consider that all instances are some points in the n-dimensional space R^n. The inductive bias of the method assumes that the prediction of a new instance will be similar to the prediction of the other close (in terms of Euclidean distance) instances.

For a given instance we can define nearest neighbors using the Euclidean distance $d(x_i, x_j)$. The calculation of the Euclidean distance between two instances is quite simple and is as follows [[19], p. 232]:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_i))^2} \tag{3.3}$$

So the target function may be both, discrete-valued and real valued, so e this method can be used for both for classification and regression problems. As I have a regression problem let's consider a training algorithm for continuous-valued target function [[19], p.232-233]:

1. For each training example $(x, f(x))$, add the example to the list of *training examples*
2. Given a query instance x_q to be predicted:
   (a) Let $x_1, \dots, x_k$ denote the k instances from *training examples* that are nearest to x_q.
   (b) Return
   $$f(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \tag{3.4}$$

As for the problems which can arise using nearest-neighbors learning, the distance between neighbours may be dominated by some irrelevant attributes. Thus, when there are many irrelevant attributes in data, the so-called curse of dimensionality might become a problem. To avoid this, we can weight each attribute with different value. We can imagine this as stretching the axes in the Euclidean space: those axes which represent irrelevant (or less relevant) attributes become shorter, and those axes which represent relevant (or more relevant) attributes become longer. [[19], p. 235]

Another problem for nearest neighbor (and for all instance based methods) is a computational time, because all processing is repeated for each new query and does not run until this new query occur. To avoid computational difficulties efficient memory indexing can be used. [[19], p. 236] [2]

3.2.4 Support vector machines

Support vector machines can be applied for those problems when data can't be separated by line. Support vector machines use nonlinear mapping – it transforms the instance space into another space. Generally new space has higher dimension than the original one. In this case line in the new space can represent nonlinear boundary in the instance space. Support vector machines were originally developed for classification problems; it is also implemented for numeric prediction. Kernel concept gave rise to a support vector machines. Mapping to a new space is done by using kernel function [[30], p. 223].

Function K is a kernel function, it can be represented as

$$K(x, y) = \Phi(x) \bullet \Phi(y) \tag{3.5}$$

The feature, that kernel function is formulated as an inner product, gives an opportunity to replace scalar product with some choice of kernel [[30], p. 227].

The problem to find parameters of SVM – support vector machines – corresponds to a convex optimization problem, which means that local solution is global optimums as well [[3], p. 325].

[2] The description of this model is taken from the project in machine learning. Those project are available on CD

SVM for regression is used to find a linear model of the form $y(x) = w^T\varphi(x) + b$, where $varphi(x)$ is a space transformation function, which is also a kernel function; w and b are parameters [[3], p. 326]. In simple linear regression the task is to minimize a regularized error function given by

$$\frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 + \frac{\lambda}{2}\|w\|^2 \tag{3.6}$$

where t_n is a target value, y_n is a predicted value by the model [[3], p.340]

The goal of SVM is to obtain sparse solution that's why the quadratic error function is replaced by a ε-insensitive error function, where error is zero if absolute distance between predicted $y(x)$ and target t is less than ε. This makes a tube around the target function. The width of this tube is ε. Another optimization technique is slack variables. Slack variables are determined for each training instance and it allows points to lie outside the tube [[3], pp.340-341].

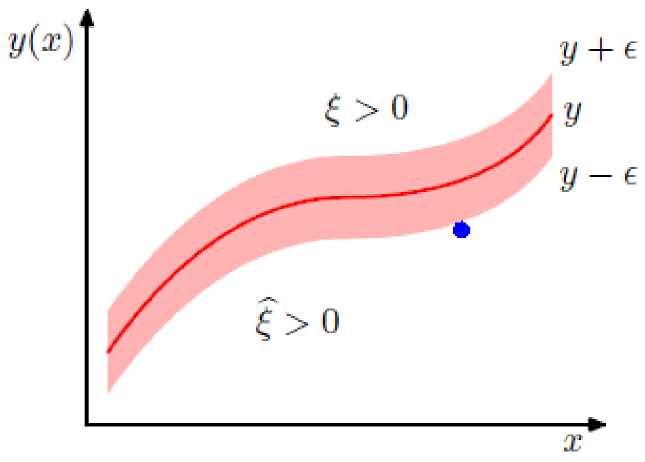

Figure 3.3: SVM illustration with slack variables and ε-tube. Y is regression curve. [[3], p. 341]

After introducing with slack variables and ε-insensitive error function, regularized error function can be rewritten

$$C\sum_{n=1}^{N}(\xi_n + \hat{\xi_n}) + \frac{1}{2}\|w\|^2 \tag{3.7}$$

which must be minimized. The constraints are $\xi \geq 0$ and $\hat{\xi} \geq 0$, and

$$\begin{aligned} t_n &\leq y(x_n) + \epsilon + \xi_n, \\ t_n &\leq y(x_n) - \epsilon - \hat{\xi_n} \end{aligned} \tag{3.8}$$

This problem can be solved using Lagrange multipliers [[3], p. 341]. After solving this problem the prediction function for new inputs is obtained:

$$y(x) = \sum_{n=1}^{N}(a_n - \hat{a_n})k(x, x_n) + b \tag{3.9}$$

Where K is a kernel function and $a_n \geq 0$, $a_n \geq 0$ are Lagrange multipliers. Here $x_1, \ldots, x_n$ are support vectors and they lie outside of ε-tube. [[3], p. 342]

According to this SVM is not using all the input instances, but only few support vectors. Tubes flatness ensures that algorithm won't overfit. The parameter ε is a user-specified. When ε is zero than all the training instances will be support vectors and algorithm will perform least-absolute-error regression. During experiments I will try to find optimal ε, so that to balance error minimization and tube's flatness – overfitting [[30], p. 229]

Kernel Functions

There are many forms of kernel functions. The following kernel functions are used in this work : polynomial kernel, radial basis kernel, and Pearson VII function based kernel.

Polynomial kernel is represented in Weka as *Poly kernel* and is given by

$$K(x, y) = (x^T y + c)^d \tag{3.10}$$

where x and y are vectors in input space, and d is a dimension of a new space and c is a free parameter [18].

Radial basis kernel is represented in Weka as *RBF kernel* and is given

$$K(x, y) = e^{(-\frac{\|x-y\|}{2\sigma})^2} \tag{3.11}$$

where σ is a free parameter [[30], p. 296].

Pearson VII function based kernel is represented in Weka as *PUK* and is given as

$$K(x_i, x_j) = \frac{1}{[1 + (\frac{2\sqrt{\|x_i - x_j\|^2}\sqrt{2^{1/\omega} - 1}}{\sigma})^2]^\omega} \tag{3.12}$$

where x_i and x_j are vectors in the input space. By changing parameters σ and ω. The Pearson VII function can replace many applied kernel functions. It can be used as a kind of universal kernel [6] [3]

3.2.5 Neural networks

In a general way, neural networks can be characterized as collection of connected neurons, that learns incrementally from the data linear or nonlinear dependency between input variables. After this process it can predict new situations, even when there is a noise in the data. The neurons are the computing units and they perform calculations locally.

A neural network is a universal approximator. It has been proven ([16]) that Multi-layer perception with only one hidden layer with a sufficient number of nodes can fit any continues function with any predefined accuracy.

Neural networks can be used for function approximation, prediction, clustering, pattern recognition and for time series forecasting as well.

Different architectures for different tasks are shown in Figure 3.4.

1. Linear classifier

[3] The description of the SVM and kernel functions is taken from the project in machine learning. Those project are available on CD.

2. A linear classifier and predictor. It can predict simple and multiple linear regression models
3. Multilayer perceptron (MLP). It is used for nonlinear prediction and classification problems
4. Compatitive networks. It is use to find clusters in the data
5. The self-organizing feature map (SOFM) competative network is also used to find cluster in the data
6. Jordan and Elman neural networks for time-series forecasting [23]

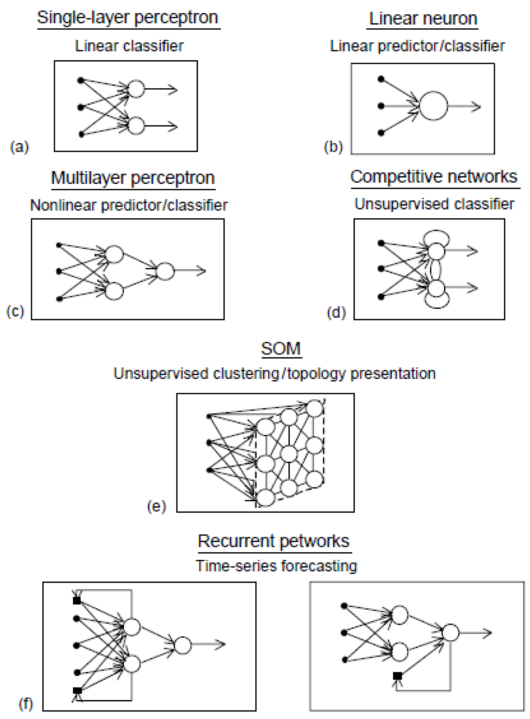

Figure 3.4: Neural networks with different topology for different tasks [[23], p. 14]

Not all the types of neural networks can be used for my purpose. The electricity consumption data set is time-series data and this should be forseen when trying out different architectures.

The simplest neural networks that can be used for time series data is linear neurons. They are similar to linear regression models or ARIMA models for non-seasonal data. It uses root mean squared error minimization to update model weights. The same weight-update algorithm is used for linear regression model, which makes this two models absolutely identical. As linear regression model has also been described and is the first model, because of its simplicity, that was applied to electricity consumption data set, there is no necessity to try out linear neuron networks as well. Moreover as shown during the data analysis in chapter 5.2, electricity consumption data is seasonal.

From all these different architectures, two of them will be described in more details: feed-forward and feedback (recurrent) networks. This decision is determined by the extreme popularity of the feed-forward neural networks and dynamic feature of recurrent neural networks. The goal of this work is to find a best performing model for electricity consumption data set. That's why recurrent network is one of the main topics here. It has "memory", which makes it perfectly compatible for time-series data.

The main strength of neural networks is that it can represent nonlinear dependency between an input and an output of the model. The most successful types of neural networks that have been used for nonlinear time series data are focused time-lagged feedforward networks and dynamically-driven recurrent networks.

First type of networks are static models and the input data is filtered before it is fed to network. This means that a short-term lagged data is chosen only for input at each step. The concept of recurrent neural networks is absolutely different. It tries to embed the dynamics of the whole data, including all available past and current instances, to forecast future values.

There are three generic dynamic neural networks. First type of networks have one unit delay feedback connection between the output and input layers. They are called input-output recurrent models and they represent nonlinear autoregressive model (NAR). If some other variables are added to as an input that the model is called a nonlinear autoregressive with exogenous inputs (NARx). The third type is fully connected recurrent networks. For this model each layers output is fed back to its previous layer with one time unit delay.

Feed-forward neural networks and its training algorithm

In the feed-forward nets there are no loops in the network connections. Layered network is the most commonly used type of feedforward nets. For this neural networks neurons are organized into layers and connections between them goes only in one direction from one layer to another.

When using *feedforwarnet* neural network in Matlab, the default training function is The Levenberg-Maarquardt algorithm, *trainlm*, as MatLab command. For *feedforwardnet* you can use 9 different training functions, while the fastest training function is generally *trainlm*. The *trainlm* method is less efficient for deep networks with thousand of neurons, because in this case it requires more memory and more computation time. Also, *trainlm* performs better on function fitting (nonlinear regression) problems than on pattern recognition problems.

Usually, for function approximation problems Levenberg-Marquardt algorithm has the fastest training time and convergence for networks with about a few hundred weights [[17], p. 282]. This is a big advantage when training is needed to be very accurate. *Trainlm* is usually able to get lower mean squared errors than other algorithms [[17], p.282]. But when the number of weights in net increases the performance of *trainlm* decreases. Also, *trainlm* has a not very good performance on pattern recognition problems if to compare with the performance of other algorithms on this problem [[17], p.282].

Now let's discuss Levenberg-Marquardt algorithm in more details [22] . LM (Levenberg-Marquardt) algorithm is a blend of vanilla gradient descent and Gauss-Newton iteration. The LM algorithm provides a solution for Nonlinear Least Squares Minimization problem. The function which is going to be minimized has the following form:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{3.13}$$

where $x = (x_1, x_2, \dots, x_n)$ is a vector and r_j are residuals. The derivatives of f can be written using Jacobian matrix J of r defined as

$$J(x) = \frac{\partial r_j}{\partial x_i}, 1 \leq j \leq m, 1 \leq i \leq n \tag{3.14}$$

To update vanilla gradient descent parameter the negative of the gradient at each step is being added, which is scaled by λ:

$$x_{i+1} = x_i - \lambda \nabla f \tag{3.15}$$

It is more intuitive to make large steps when the value of the gradient is small and have small steps when the value of the gradient is large, but vanilla gradient descent does it vice versa. [22]

Another important issue is an error curve. Vanilla gradient descent does not take it into consideration, and to improve this Levenberg used error information as gradient information, to be more precise, he used second derivatives. He used Newton's method to solve the equation

$$\nabla f(x) = 0 \tag{3.16}$$

Expanding the gradient of f using a Taylor series around the current state x_0, we get

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0) + \text{ higher order terms of } (x - x_0) \tag{3.17}$$

If we assume that f has a quadratic form near x_0, we get the update rule for Newton's method

$$x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \nabla f(x_i) \tag{3.18}$$

Levenberg proposed an algorithm, which update rule is a mix of the above mentioned algorithms and has the following form

$$x_{i+1} = x_i - (H + \lambda f)^{-1} \nabla f(x_i) \tag{3.19}$$

where H is the Hessian matrix evaluated at x_i. This update rule is used as follows. If the error decreases with the new updates, it means that the quadratic assumption on $f(x)$, mentioned above, is working, and then λ is reduced (usually by 10), and this reduces the influence of gradient. If the error increases with the new updates, it means that we would like to have bigger step, so λ is increased (usually by the same factor), and this increases the influence of gradient. The Levenberg algorithm is as follows:

1. Do an update according to the rule, mentioned above
2. Evaluate the error
3. If the error increased with this update, then take the previous value of the step (taking the previous values of the weights), and increase λ (e.g. by 10). Then go to first step
4. If the error decreased with this update, then take this step again and decrease λ (e.g. by 10)

The disadvantage of this algorithm is that if λ is big then the Hessian matrix is not used. Marquardt proposed an improvement of this disadvantage by the replacement of the identity matrix in previous formula with the diagonal of the Hessian. So now Levenberg-Marquardt rule looks as:

$$x_{i+1} = x_i - (H + \lambda diag[H])^{-1} \nabla f(x_i) \tag{3.20}$$

It means that in the direction with low error curvature the step is large and in the direction with high error curvature the step is small.

The only weak point here is that it needs a matrix inversion when updating. Even though the inversion can be implemented using some optimization technique (like pseudo-inverse), still it has a high computational cost when the size increases to a few thousand parameters, but for a small models (having like a hundred or a few hundred of parameters) this method is still much more faster than, for example, vanilla gradient descent [22] [4]

Recurrent neural networks

Static systems cannot describe time series data. For this purpose dynamic features should be added to the system. In case of neural network models this feature is recurrent connection. Elman modified the recurrent neural network that was proposed by Jordan in 1986. The Jordan model is shown in Figure 3.5.

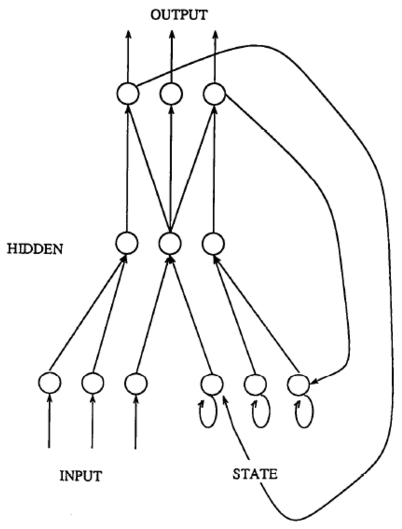

Figure 3.5: Jordan neural network. A FIGURE [[9], p. 5]

In this model the hidden nodes are feed with output values. Elman modified Jordan model in the following manner: in this network hidden nodes are feed with their own output. So a hidden node can see its previous output and that means that the previous response has an influence on the current output. In both cases recurrent connection weight is fixed and equals 1 [9]. Elman network is shown on the figure 3.

Dynamic neural networks are trained with the same algorithms as multilayer neural networks. Those algorithms are gradient-based and are more complicated to calculate for dynamic networks than for static ones. This is because weights have long term effect on output as well as immediate. So instead of simple backpropagation algorithm dynamic

[4]Tthe description of this training algorithm is taken from the project in machine learning. Those project are available on CD.

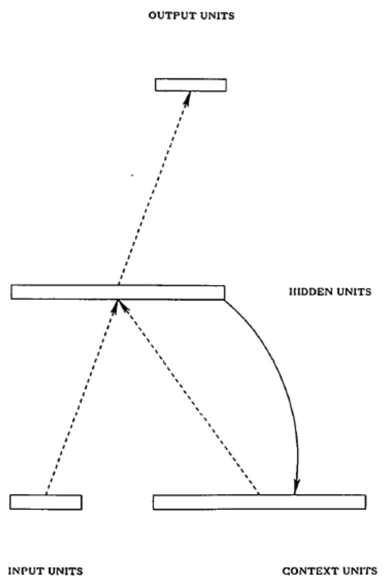

Figure 3.6: Elman neural network A FIGURE [[9], p. 6]

backpropagation algorithm is used and this one is more complicated, more time-consuming and error surface is less smooth. Because of this the local minimal can be the bigger problem than for static models. One way out is to train the network couple times.

Once again to answer the research question and find the best model for electricity consumption data set neural networks should be tested.

This problem is handled automatically in Matlab toolbox. The user can't decide what algorithms will be used for training. For some problems simple backpropagation is enough, while some problems require dynamic backpropagation. The decision is made automatically when a 'train' algorithm is called [17].

Chapter 4

Related work

The importance of an accurate prediction was briefly discussed in Chapter 1. Rainfall forecast for Australia is a good example for this purpose. For Australia, it is very important to make accurate forecast about rainfall. If there will be a big error in a rainfall forecast, it can greatly affect the economics and business. The research which is discussed in 4.1 is motivated by this issue. Planning resources is crucial for rapidly developing countries like Turkey. Agriculture is a growing sector in this country and more and more resources are consumed in this area. The researchers [28] predicted that to 4 million tons of diesel will be consumed in 2020; so they suggest that some alternative energy resources should be found.

4.1 related work about neural networks

Statistical methods have a much longer history than the machine learning methods. Nowadays many researchers questioned the efficiency of statistical models for solving some real-life problems in comparison with machine learning methods. One research motivated by this reason was conducted by John Abbot [15] and will be discussed here.

In the near past to deliver official weather forecast for public, statistical models has been used in Australia.

From 1989 the Australian Bureau of Meteorology (BOM) developed a method and then from 1994 Queensland Government (QG) produced another method, which was supported from the Department of Environment and Resource Management. These two models performed classification task using data patterns to forecast. They used atmospheric and oceanic circulation data as an input for model. In July 2013, the old models were replaces by The Predictive Ocean Atmosphere Model for Australia, POAMA. The new model differs from the previous ones with its capacity to make numeric prediction. Even though there is no prove that this new method performs better that the old methods [15].

The artificial neural network (ANN) has been successfully used for rainfall forecasting in many countries. The same can't be said about Australia.

The ordinary multilayer neural network cannot deal with the time-series forecasting task without modification, because it is static and this means that only current input influence mapping of the current output. While for temporal data the past values also matters in terms of current output. For such a task Jordan and Elman methods can be used. These networks are also called as "simple recurrent networks" (SRN). The Elman network is dealing with time-series data by using recurrent connections which "memorize" past values. Also this unit has a “forgetting” feature. It forgets the past values with an exponential decay. Other types of neural networks can be also used for a temporal data, like for example time lagged recurrent networks. It is impossible to predict which neural network will be the best for a specific data without testing them. The main factor is the error rate during deciding which neural network to choose for a particular data. For rainfall forecasting task for Australia the Jordan method showed the best performance. The memory unit in recurrent network has a flexible variable which should be form 0 to 1. 0 means that only current instance will influence the current output, while 1 means that the memory is as deep as possible. After experimenting Jordan neural network was chosen with one hidden layer and with memory unit variable 0.8. Also different models of ANN were built using the combination of the input data sets. This was done in purpose to find the best combination of the input values.

Abbot compared performance of ANN, POEMA and Climatology for three different geographical regions. Climatology is a statistical method used by Queensland government for seasonal rainfall forecasting. Almost every ANN outperforms the performance of official models, even when ANN is fed with one input data set. For example for Harrisville region if the RMSE of the best performed ANN is 39.5, and for the worst is 47.4 that for the Climatology is 53.1 and for the POEMA 74.6 [15].

The recurrent neural networks have been used before for forecasting Household Electricity Consumption [2]. The research was motivated by the concern about increasing electricity consumption. The electricity demand of Sicily was analyzed from 2001 to 2010. The researchers concentrated on household sector as it takes the second place after Agriculture and the demand was growing and was making big impact [2]. As the houses getting smarter and more equipped with more and more technique, the electricity consumption is raising too. The authors [2] concentrated on air-conditioners and they think that adding new variables that relate to using air-conditioners can improve the forecasting accuracy during the summer period. For prediction Elman neural networks was used. And the data is an electricity consumption of Palermo. For this data several networks were tested and the best network was chosen according to prediction accuracy. The model was implemented in order to forecast the electric current intensity at time t. It is one hour ahead forecasting. Additionally to the historical series of the hourly mean values of electric current intensity, exogenous inputs have been also added. They have added weather variables (temperature, relative humidity. . .), the variable that takes in account approximate number of air-conditioners available in that aria, also variable that reflects discomfort rate according to temperature and relative humidity. It is worth mentioning that the variable of discomfort rate is not a very accurate as outdoor variables are used for calculation, because the indoor variables are not available. Also the threshold variable associating the beginning of discomfort was explored after experimenting with a different numbers and the one that caused the lowest performance error was chosen [2].

To forecast Household Electricity Consumption [2] Elman neural network was implemented with the same number of context units as hidden units. Every hidden neuron is connected to one context (memory) neuron and the value of the each connection is +1. The stopping criteria of the training was the lowest error achieved at the validation data. To find the best architecture a different number on hidden nodes, various combination of input values, and different number of lag values were used. Also before training all input values were normalized in the range [-1,1]. Also to find out how much influence has each input value on the model, the sensitivity analysis was conducted. The technique that was used is called *weight method*. The best network's mean percentage prediction error computed for a test week was 1.5%. The authors [2] also suggested that performance accuracy might be improved with more accurate input data. Also adding exogenous values made a big difference. The brain-state-in-a-box (BSB) activation function was used for the hidden layer and for the output layer.

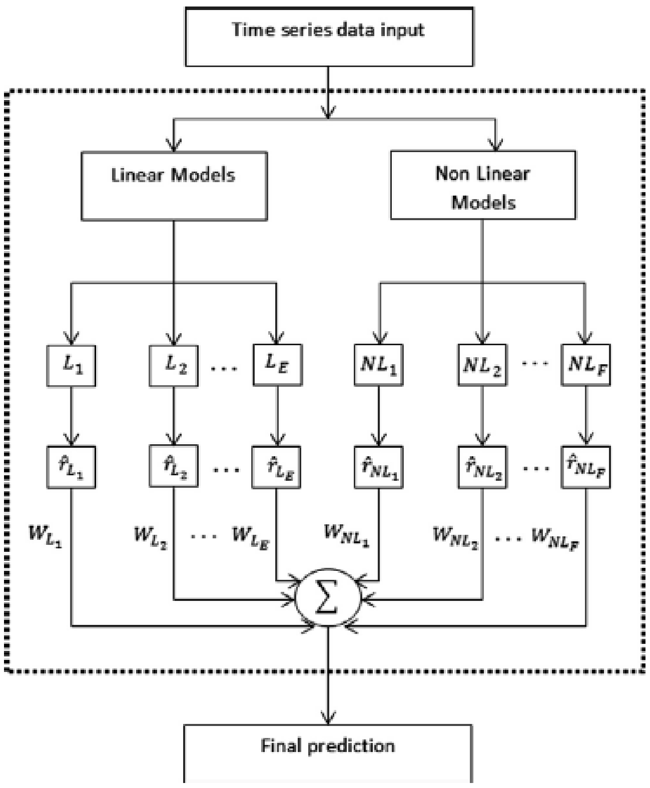

Figure 4.1: HPM architecture [[10] , pp 4]

Not only plain models have been tested and used for time-series data but also hybrids [10, 7]. Chunshien Li and Jhao-WunHu composed the hybrid from neuro-fuzzy system and ARIMA models. They have tested this model for three data examples and as they claim the hybrid model performed better than simple ANN or ARIMA [7].

Probably one of the most difficult time-series data to work with is stock returns, because it is noisy and non-stationary [10, 26]. Shipra Banik et al. combined ANN with rough set (RS) theory to forecast price on Dhaka stock exchange [26]. This hybrid model achieved 97% accuracy and significantly outperformed single models.

Artificial neural networks generally has maintained the best results for stock return data [10]. Non-linear models are superior to linear models for such a data, because of the complexity. As discussed by the Akhter Mohiuddin Rather and his coauthors ANN do fail sometimes in finding global optimization. To overcome this issue one of the method is to build several architectures and to use genetic algorithms to optimize between them. Using hybrid models of linear and non-linear methods has been also very popular and still is. So to forecast stock returns hybrid model was built. The proposed hybrid prediction model (HPM) is a combination of a recurrent neural network, exponential smoothing and ARMA models. The final output of HPM model is a weighted sum of these three models 4.1. The sum of weights is 1 [10]. To find the weights the minimization task of mean squared error (MSE) between predicted value and the target value should be solved. This is done with using genetic algorithm (GA). As authors claim the obtained HPM is powerful tool and it can even predict a sudden changes in data [10].

The recurrent neural networks are used with one hidden layer. In this case input-output is supervised as it is supervised for autoregressive moving reference neural network (AR-MRNN) [10]. The number of inputs neurons for this recurrent neural networks equals to the regression order. The input layer has a linear activation function, while hidden and output layer has a logistic activation function. A long term memory , or a context units are fed from the input layer and their function is to hold the data and pass it to the hidden layer after a whole pattern comes from an input layer. This network is trained with backpropagation algorithm. Another part of this hybrid is exponential smoothing (ES). It is a one-step-ahead prediction and is calculated geometric sum of past historical series.

To test the performance of HPM [10] the experiments was done with three real stock data obtained from National stock exchange of India. Each data set has 164 weekly returns, so the experiments are not done on a big data. The data was divided into two equal parts, half of it was used for training and the second part was used for testing. For each data set recurrent neural network was trained separately, because each of them may need different number of nodes or epochs. Non-linear model significantly outperformed linear models. It has lower error and higher correlation. The authors also tried multilayer perceptron (MLP) for the stock returns data. The main drawback for MLP was that it could not predict sudden jumps in data. That is a very big drawback, because mostly stock data has sudden changes. HPM outperformed the recurrent neural network (RNN) even though it was the best performing model in this hybrid. This result shows that combination of the models has the better generalization ability than just one model.

Accurate Forecasting can reduce the damage caused by natural cataclysms. One example of this is flood control. Fi-John Chang and his coauthors attempted to analyze and forecast the data which was gathered from the Yu–Cheng Pumping Station which is located in Taipei City of Taiwan [11]. The available time-series are the water level in the floodwater storage pond (FSP) and rainfall data. Flooding problem is very real and it happens in a very little period of time, so sometimes there is a situation when data of the level of the floodwater is not available. Because of this reason Fi-John Chang simulated two scenarios: one with only rainfall data and one with both data sets. Three types of neural networks have been used to build multi-step-ahead model which forecasts a water level in the floodwater storage pond. One of the neural networks is static (backpropagation neural network) and two of them are dynamic (Elman neural network and nonlinear autoregressive network with exogenous inputs (NARX) network). On the first place Fi-John Chang analysed the data statistically, also conducted correlation analysis on the combined data to find the time span of rainfall affecting the raising of FSP water level [11]. He trained neural network with Levenberg–Marquardt algorithm and for a transfer function for a hidden layer was used sigmoid function while for output layer was used a linear function. NARX was trained also using Levenberg–Marquardt algorithm and the transfer functions for the hidden and output layer were sigmoid and linear functions respectively [11]. Here when NARX was used for multi-step- ahead forecasting the estimated outputs were feed back to the input nodes. The models was built for six-step-ahead forecasting.

For flood control forecasting dynamic models completely outperformed static model [11]. For a one-step-ahead forecasting there was not a big difference between the performance of NARX and Elman network, but as the horizon of forecasting increased NARX performed much better than Elman network. I think that it was not surprising that NARX outperformed other models for multi-step ahead forecasting as it has feedback loop which makes it outstanding for time series forecasting. Also NARX supports lag of the time-series, while Elman network does not. That's why it performed much better than Elman network for multi-step-ahead forecasting on flood control data [11].

This article [29] is a review of what have been done over last 15 years about electricity price forecasting (EPF) task. As discussed in this article the advantage of Elman and Jordan neural networks over other methods like ARIMA, wavelet-ARIMA, MLP, fuzzy ANN and wavelet-ARIMA-RBF networks for short-term forecasting of time-series data is obvious. But even-though they are not as successful in finding long-term dependencies. To overcome this obstacle nonlinear autoregressive neural network with external input (NARX) was proposed in 1996. NARX has better generalization ability. The author remarks that NARX was not popular and was not used for competitions until recently.

To answer the first research question and find the best performing model NAR will be also applied to electricity consumption data set.

4.2 related work about ARIMA

The Box and Jenkins approach was used to forecast Net Electricity Consumption in Turkey [4] and was very successful, as the author claims that he obtained more accurate results than the previous studies did for the same time-series data. Another case when Box and Jenkins approach was used was to to estimate the future energy consumption in the agricultural sector [28]. The motivation of this research was the growing energy consumption is agriculture. The diesel consumption accounted for 30–40% of the total energy consumption in the Turkish agriculture sector in 2008; that's why diesel consumption is the focus of this study.The problem was that they did not have sufficient data. The ARIMA model does not have many coefficients and this is the main criterion to choose it for this particular data. No comparisons were done with other methods.

In both papers ([4],[28]) the same forecasting technique was used. The difference is that Net Electricity Consumption needed preprocessing.

Four steps of ARIMA methodology was used for electricity forecasting in Turkey: identification, estimation, diagnostic checking and forecasting. Sometimes before using this approach some data transformations should be used according to a time-series data structure. Turkey is a developing country and this causes that electricity consumption has an exponential growth from 1970 to 2008. In purpose to deal with exponential trend a natural logarithmic transformation was applied to the time series.

Generally for model identification an autocorrelation function (ACF) and a partial autocorrelation function (PACF) are used. First Boran has checked if the data was stationary. The ACF did not decrease rapidly, that means that the data was not stationary. When a data is not stationary, the next step is differencing operation. After one differencing operation the ACF and PACF cut off after the first lag. According to this Boran suggests ARIMA(1,1,0). For parameters identification Statistical Package for the Social Science (SPSS) was used. This software was acquired by IBM in 2009. The same software was used for model checking and forecasting. There was an economic crisis in Turkey in 2001, and this affected the results; as the biggest difference between forecasted data and observations are during this period. Based on this models future forecast has been performed. However as Boran notices, this model is not accurate when crisis takes place. It could be handled by updating the model for new observations.

This paper is comparative study of ARIMA and ARMA models for a particular time series data for different forecasting periods. The data set that is used during experiments is electricity consumption in one household, and is sampled in every minute from 2006 till 2010, and data set has 2,075,259 instances. For experiment the data was divided into two parts. The observations from 2006-12-26 to 2009-12-31 were used for model construction, while the observations from 2010-01-01 to 2010-11-26 were used to find the proper forecasting period. The authors intend to use only ARIMA and ARMA models. The goal of this article is to find the best model for forecasting the electricity consumption. For model comparison AIC (Akaike Information Criterion) and RMSE (Root Mean Square Error) criterion were used. During decision is made about what techniques should be used for a time series data analysis the following thing matter: prediction interval, prediction period, characteristic of a time series, and size of time series. ARIMA and ARMA models are used to find patterns and trends. As for time periods they use daily, weekly, monthly and quarterly data intervals. R and Rstudio was used during these experiments. The forecasting framework consists from three steps: data preprocessing, constructing and decomposing time series and building the model. After reading the file missing values are handled. As observations are sampled in every minute, authors suppose that one observation won't vary much form the following observation; that's why they fill the missing data with the previous value. Another way is to fill it with mean value of the data. After this aggregate function is used in R, so that the resulting time series data has a daily, weekly, monthly and yearly time unit. For different time intervals, different time series data should be constructed. Time series data consist of four components: trend, seasonal effect, cyclical, and irregular effect. To decompose the time series data decomposed() function is used. This function removes trend and seasonal components from the data. There is a special command in R for model identification. For ARIMA model (p,d,q) values were estimated. For daily, weekly, monthly and quarterly time series different orders were suggested. In fact for longer time intervals fewer orders were more suitable according to this experiment. After this corresponding models were constructed for forecasting. The same methodology was used for identification and construction of the ARMA models. Comparing ARMA and ARIMA methods, the experiments shows that ARMA overcomes ARIMA for daily and weekly series, while ARIMA has better performance for monthly and quarterly time series data.

Chapter 5

Data description and experiments

This chapter has the following structure: In (5.1) the data set will be described, how it was obtained and the way it was filtered. Chapter 5.2 shows how statistical methods have been applied for the electricity consumption data set, also it describes the process of finding the ARIMA model. After statistical models the following machine learning models have been applied to this data set: linear regression and regression trees (5.3), K-nearest neighbor (5.4), support vector machines (5.5) and finally feedback and feedforward neural networks (5.6). Also it describes how the data was modified before the machine learning models were applied to it and here is also discussed the reasons why it was modified in this manner.

5.1 Description of the summed up hourly electricity consumption data set

The data set has 19 008 instances and two attributes: time and electricity consumption. This data set includes period from 11/1/2011 12:00:00 AM to 12/31/2013 11:00:00 PM. In the original data some instances have been missing. Mostly there was missing only one hour interval, in one case the whole day was missing. To deal with the missing values I used interpolation technique, which means that the missing value was replaced by the mean of it's neighbour values. Using the same logic, the one day missing instance was replaced by the mean of a same hour of the neighbouring days.

5.2 statistical analysis of the electricity consumption data

In this work for statistical analysis R was used, which is a free software environment for statistical computing and graphics.

From the beginning to get a general idea about the data the hourly mean plot (Figure 5.2) has been generated for consumption data set. The plot shows that on average there are two peak points during the day. One at 11:00 AM, and another at 21:00 PM.

This plot gives a general idea about the data, but it's not specific enough to discuss whether the data has trend or seasonality.

When working with time-series data in R, first thing to do is to import and than save the data as time-series object. To do so the *frequency* of the data should be specified.

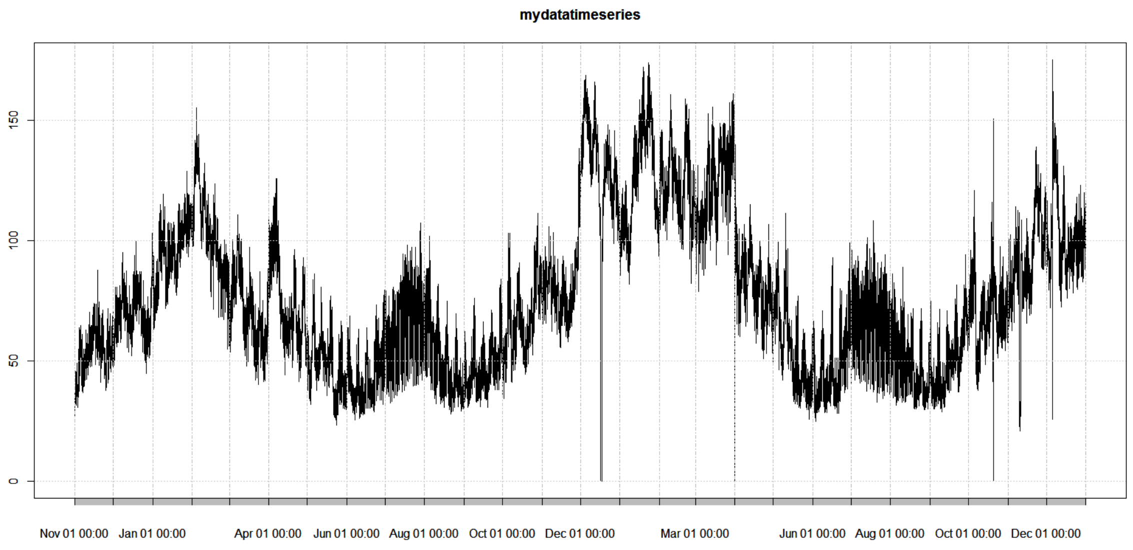

Figure 5.1: time plot of electricity consumption data

The value of argument frequency is used when the series is sampled as integral number of times in each time unit interval [27] . For example, one could use a value of 7 for frequency, when the data are sampled daily, because the natural time period is a week, or 12 when the data are sampled monthly as the natural time period is a year. In this case the data is sampled hourly and this means, that frequency = 24.

After this the data was plotted and it is presented in Figure 5.1

Let's first discuss whether or not the data has seasonality. Seasonality in time series means that the data pattern changes in the same way after constant time periods [27]. ARIMA model, which is discussed in Chapter 2, can not deal with seasonal data. For this purpose the extension of ARIMA model was developed, called seasonal autoregressive integrated moving average ($SARIMA(p,d,q) \times (P,D,Q)$) model.

There are two types of differencing (seasonal and non-seasonal) that should be considered, when searching for proper SARIMA model for seasonal data. Non-seasonal difference was discussed in chapter 2. Seasonal difference looks like following [5]:

$$(1 - B^s) = x_t - x_{t-s} \tag{5.1}$$

Here B is a backward operator, and $x_t, t = 1, \ldots, n$ is a time-series observations. According to background material, which was discussed in chapter 2, and just mentioned seasonality issue, for model identification I will go through the following steps:

1. Checking the data. First the data should be checked weather it is stationary or not. If it is not, than seasonality and trend should be analyzed
2. Differencing: Non-seasonal differencing is used when there is obvious trend, but no seasonality in the data. Seasonal differencing is used when there is seasonality but no trend in the data. If the data has seasonality and trend than both differencing is used. The differencing is applied until the resulting data is stationary.
3. The next step is to analyze autocorrelation function (ACF) and partial autocorrelation function (PACF) to get any idea about the values of p,q,P and Q.
4. The last step but not least is diagnostics and model comparison. Now residuals are observed. The residuals are the differences between the real values and fitted

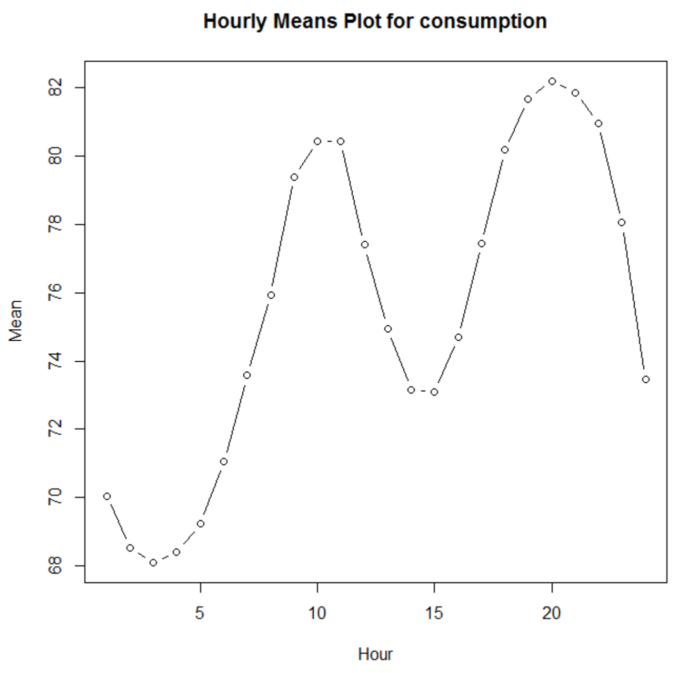

Figure 5.2: Electricity consumption hourly mean

ones. If the model fits well to the data, the residuals should behave like white noise with the mean zero and variance one. Also result of all these steps could be not a single model, but a couple of them. The following criterion can be used to pick one of them: Akaike information criterion (AIC), root mean square error (RMSE) and mean absolute error (MAE). If two models' performance does not differ much, it is preferable to choose the one with a fewer parameters.

The model performance is measured on the test data. So on the first place I have divided the data into two parts: for training and testing. The question is: How the data should be divided? What percentage should be used for training and for testing? To answer the research question, the performances of the models should be compared. As discussed before, overfitting and generalization issues should be kept in mind when comparing the models. Because of this, a performance on the test data is taken into account during consideration instead of performance on the training data. Also as mentioned before in Chapter 1 MAE and RMSE are the criterion for measuring performance. To be compatible different model's MAE or RMSE, the test set should be fixed and the same splitted data sets should be used for all the models. One of those models that will be tried of electricity consumption data, is ANN¡ and for this model as optimal spit is proposed $70\% - 15\% - 15\%$ for training, validation and testing [17]. As statistical methods does not need validation data, for training will be used 85% and 15% for testing.

As a result of this discussion instances from 1 to 16000 are used for training and from 16001 to 19008 for testing. After splitting the data I took the training data to analyse and suggest the best fitting model.

On the first place the stationarity of the data should be investigated. For this purpose ACF and PACF functions are analysed.

To find this out ACF and PACF plot analysis is a good way. For stationary time-series

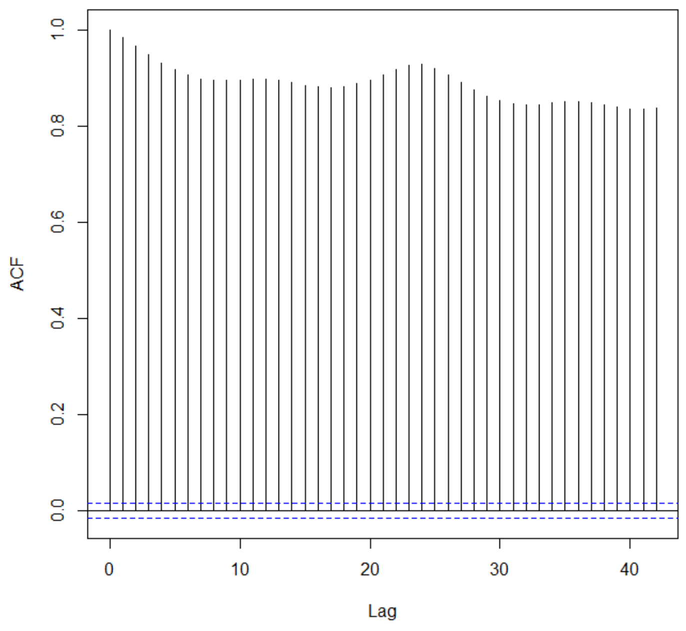

Figure 5.3: ACF of the electricity consumption data

both functions should decrease to the zero as the lag$k$ increases.

Looking at the ACF (Figure 5.3) and PACF (Figure 5.4) it is obvious that the data is not stationary. Neither of them decreases to the zero for big lag. The data set has obvious trend so on the first place, I will try non-seasonal difference which is described in Chapter 2. So if $x_i, i = 1, 2, \dots n$ is the electricity consumption data, than the resulting data after one non-seasonal differencing will have the following form

$$x'_i = x_{i+1} - x_i$$

where $i = 1, 2, \dots, n$

Differencing (seasonal or non-seasonal) takes place until the resulting data is not stationary. So, the next step is to check the resulting data after one non-seasonal differencing. To do so, the same procedure takes place as for original electricity consumption data; in particular ACF and PACF should be analysed.

After non-seasonal difference ACF and PACF are shown on (Figure 5.5) and (Figure 5.6). Now, let's on the first place discuss whether the data is stationary or not. If we will examine ACF (Figure 5.5) of the resulting data, it not only does not approach zero, moreover, it barely decreases at all, while PACF (Figure 5.6) of the resulting data decreases, but it does not go under the threshold, which means that when lag increases it does not gets closer to zero. This means that the resulting data is not stationary.

As the resulting data after one non-seasonal differencing is not stationary, it should be differenced one more time. The question is: Which differencing should take place, seasonal or non-seasonal? To answer this question let's observe the ACF (Figure 5.5) and PACF (Figure 5.6) plots again. Both plots show that the data has seasonality. It can be easily detected when keeping in mind the intuitive explanation of seasonality. Seasonality is repeating pattern in data and plot is a good way to visualize it. The ACF value on lag0 is an exception, but after it the pattern repeats itself after every 24 lag value. PACF has

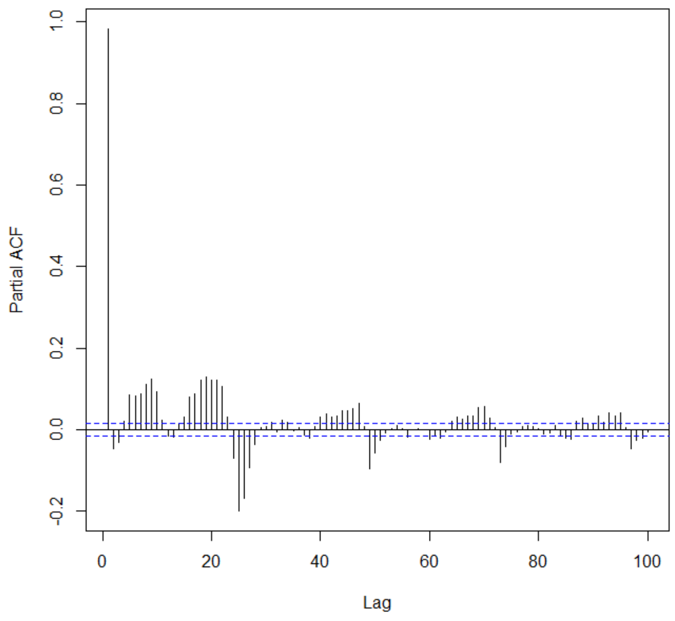

Figure 5.4: PACF of the electricity consumption data

a damped sine wave form and has peek after every 24 lag. In other words the picks at seasonal lags $h = 1s, 2s, 3s\ldots$ where $s = 24$ suggests that next step is seasonal difference. So if $x'_i, i = 1, 2, \ldots n$ is the resulting data after one non-seasonal differencing of the electricity consumption data, than the resulting data after one seasonal differencing of it will have the following form

$$x''_i = x'_i - x'_{i-s}$$

where $i = s, s+1, \ldots, n$ and s is the seasonal period and in this case it equals to 24.

After this seasonal differencing the ACF and PACF of the $x''_i, i = 24, 2, \ldots n$ data are shown on (Figure 5.7) and (Figure 5.8). Once again, the stationarity of the resulting data should be discussed. Both functions decrease to the zero as lag increases. This means that resulting data of one non-seasonal and one seasonal differencing is stationary.

	AR(P)	MA (Q)	ARMA(P,Q)
ACF	Tails off at lags ks, $k = 1, 2\ldots,$	Cuts off after lag Qs	Tails off at lag ks
PACF	Cuts off after lag Ps	Tails off at lags ks ks, $k = 1, 2\ldots,$	Tails off at lags ks

Table 5.1: ACF and PACF behavior for SARMA process

As already mentioned above the next step is to analyze ACF and PACF to get any idea about the values of p, q, P and Q values for $SARIMA(p, d, q) \times (P, D, Q))$ model. To find the values of p and q Table:2.1 will be used, which briefly summarises the theory, discussed in Chapter 2. The process of finding P and Q looks similar as process of finding p and q. The difference is that seasonality should be taken into account. The behavior of the ACF and PACF process is given on Table:5.1. One thing that should be added to this table about the behavior of ACF and PACF is that the values of non-seasonal lags

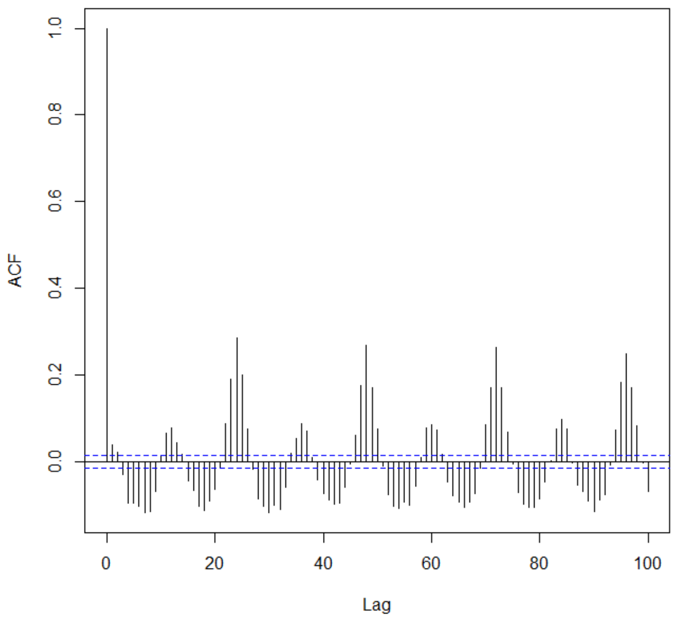

Figure 5.5: ACF of the resulting data after one non-seasonal differencing

should be zero,in other words for lag$h$, where $h \neq ks \quad \text{for} \quad k = 1, 2, \ldots$ This means that when analysing seasonal part of a model and observing whether ACF and PACF tails off, we actually look at it's values on peaks of each seasonal interval, in particular on lag24, lag48, ... lag24k, ...

Now, let's look at the ACF plot (Figure 5.7). It rapidly decreases for the first 15 lags, but after this it tails off, it has a damped sine wave shape. As value of ACF is not big, it's hard to see on the plot that it tails off, but it should be easier if we observe the value of ACF in relation with the threshold, which is represented on the plot as blue dotted line. The same can't be said about PACF function. Figure 5.7 shows that PACF cuts off after about 20 lags, which can be interpreted in two ways, either as PACF cuts after lag $1 \times s$ or $0 \times s$, where s is seasonality interval and equals to 24. According to all this for seasonal component MA(1) or MA(0) should be suggested.

After examining seasonal part of the model, it's time to find p,q. Both ACF and PACF tails of so for non-seasonal part ARMA(p,q) is suggested.

According to all the discussion and analysis the following model is suggested:

- $SARIMA(p, 1, q) \times (P, 1, 0)_{24}$ here q should be tested up to 8, and p up to 20. Because ACF cuts after lag8, and PACF cuts of after lag20.

To choose between the different ARIMA and SARIMA models the following criterion will be used: root mean squared error (RMSE) and Akaike information criterion (AIC). The formula of RMSE is described in Chapter 1.2.2.

AIC is defined by the following formula :

$$AIC = (-2)\log_e(\text{maximum likelihood}) + 2(\text{number of free parameters}) \tag{5.2}$$

AIC measures unreliability of the model as well as the fit of the model, in other words it measures the relative quality of a statistical model for a given set of data. AIC is often used for statistical models selections [21]. According to AIC and maximum likelihood estimation [25] from two models, one with the smaller AIC should be preferred.

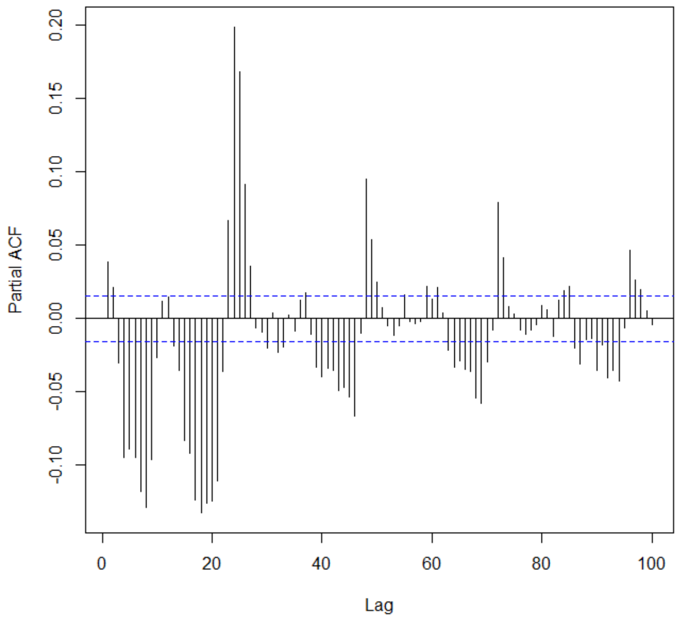

Figure 5.6: PACF of the resulting data after one non-seasonal differencing

So, to find the best fitting model different models were trained on training data set and after their performance have been tested on test data set. The results are shown on Table:5.2. According to this results $SARIMA(2,1,2) \times (1,1,0)_{24}$ performs better that other models. During the search process for p, q, P and Q values, the following trend is observed: the more complicated models performed worse when $p > 2, q > 2, P > 1, Q > 0$. This can be explained with overfitting.

Even though $SARIMA(2,1,2) \times (1,1,0)_{24}$ model performs better than other models, it still needs furthermore investigation. To check whether or not this model can be further improved, forecast error should be examined. Forecast error is the difference between the predicted values and real values [8]. If there won't be correlations between forecast errors for successive predictions, this means that this model cannot be improved upon. This means that if the forecasting model of the $SARIMA(2,1,2) \times (1,1,0)_{24}$ model is normally distributed with mean zero, the search for a better SARIMA or ARIMA model will be stopped. To check this the histogram of the forecast errors has been plotted with an overlaid normal curve that has mean zero and the same standard deviation as the distribution of forecast errors (Figure 5.9).

Figure 5.9 shows that forecast error of $SARIMA(2,1,2) \times (1,1,0)_{24}$ model is normally distributed with mean zero.

5.3 Linear regression and regression trees

Box and Jenkins model is specially created for time-series data and finding a dependency between the value to be forecasted and a different lags is a part of a process. The input of this model is a vector $(y_1, \ldots, y_N)$, here N is the number of observations. The same is not true for linear regression. The data should be modified before it is fed to machine learning models. This is discussed in Chapter 3.1.1; The input matrix should have the

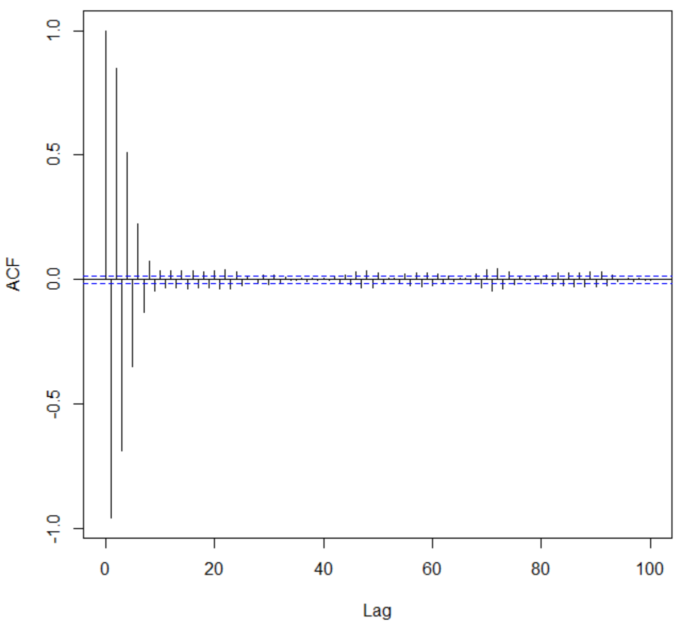

Figure 5.7: ACF of the resulting data after one non-seasonal and one seasonal differencing

following form $[(N - n - 1) \times n]$, here n is a lag.

First of all, lag and input variables should be find for each model. The variables that can be added to the input data (in addition to electricity consumption value) are: *season*, *weekday*,*month*. Each of this attribute can get value from finite set, which is defined beforehand, meaning that for example *weekday* can get values from the following set (*Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday*) This type of attributes are called nominal.For selection criteria, once again error rate will be used.

For this experiments Weka (Waikato Environment for Knowledge Analysis) software was used. Weka is a free software, is written in Java and was developed at the University of Waikato, New Zealand [30]. Let's begin with linear regression. Table 5.3 shows the mae and rmse on test data for different number of lags.

According to this results (Table 5.3) the optimal number of lags is 10.

The results show that the linear regression model performance does not depends much on the number of lags. Even though lowest error rate was achieved for 24 lags, I think that optimal model is linear regression with 10 lags. During choosing this model, I kept in mind both model complexity and error rate.

One of the guidelines, which will be used during model comparison, in this work, is Occam's razor. The idea of this principle is to prefer simpler models as they generalize better [19]

Now, let's try linear regression with different combination of attributes:

As table 5.4 shows adding attributes slightly improved the linear regression performance, but not significantly. The linear regression model after training for 10 lags has the following form

$$x_{10} = 0.1184x_1 - 0.0283x_3 - 0.0475x_6 - 0.0406x_7 + 0.9879x_9 + 0.7616$$

This linear regression model explains a lot. The biggest wight in this model has the

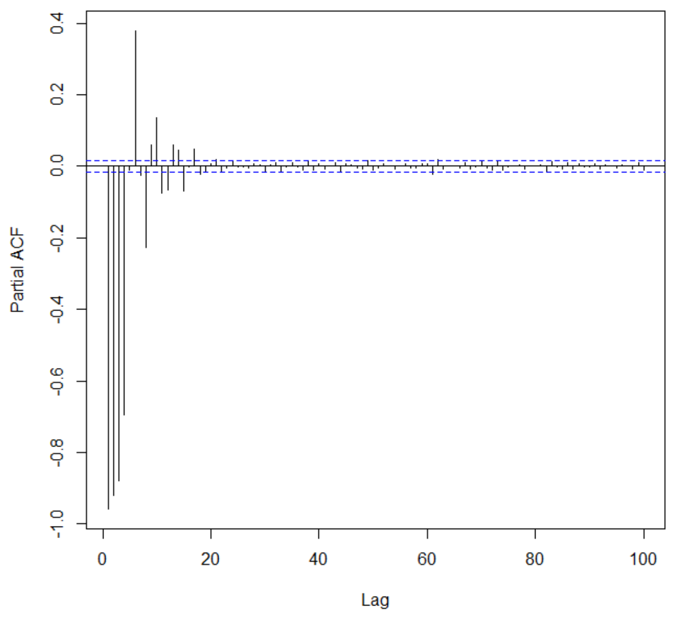

Figure 5.8: PACF of the resulting data after one non-seasonal and one seasonal differencing

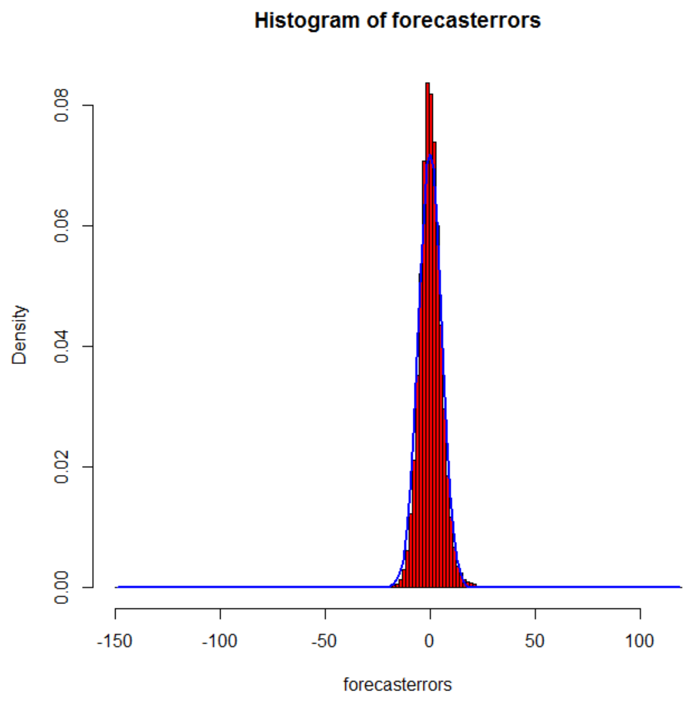

Figure 5.9: Forecast error with an overlaid normal curve

last lag and it's almost equals to 1, which means the next observation almost completely depends on the previous one. That's why there was not significant difference between the models with 2 lags and with 24 lags.

Regression trees (REPTrees), model trees (M5P) and rules from model trees (M5Rules)

Model	AIC	RMSE
$ARIMA(2,1,2)$	100976.6	49.18163
$ARIMA(3,1,3)$	99731.27	49.70293
$ARIMA(4,1,3)$	99814.18	49.68955
$SARIMA(1,1,1) \times (1,1,0)_{24}$	101174.1	67.74196
$SARIMA(2,1,2) \times (0,1,0)_{24}$	105276.4	68.12669
$SARIMA(2,1,2) \times (1,1,1)_{24}$	96038.73	131.9645
$SARIMA(0,1,1) \times (2,1,0)_{24}$	99780.85	223.2494
$SARIMA(2,1,1) \times (1,1,0)_{24}$	101114.5	94.61101
$SARIMA(3,1,3) \times (1,1,1)_{24}$	96040.19.	131.9889
$SARIMA(3,1,1) \times (1,1,0)_{24}$	100880.6	48.98556
$SARIMA(2,1,2) \times (1,1,0)_{24}$	<u>98855.36</u>	<u>48.33502</u>
$SARIMA(3,1,2) \times (1,1,0)_{24}$	100844.4	49.03785
$SARIMA(3,1,2) \times (1,0,1)_{24}$	96096.2	100.1585
$SARIMA(4,1,2) \times (1,1,1)_{24}$	108175.9	149.1586
$SARIMA(4,1,4) \times (2,1,2)_{24}$	128120.1	242.1201

Table 5.2: ARIMA and SARIMA model performances on test data

number of lags	mae	rmse
2	4.1513	6.8947
3	4.1495	6.9032
5	4.1443	6.906
10	4.0588	6.7703
15	4.0593	6.7582
20	4.0399	6.7603
24	4.0149	6.6614

Table 5.3: Linear regression performance with different number of lags on test data

has the same training algorithm as linear regression. The difference is that they are more complex and advance models. If the output of linear regression is one function, the output of regression trees are as many equations as tree has nodes. The output of model trees is number of equations too. That's why their behavior should be the same with number of lags and adding nominal attributes. According to this Regression trees, model trees and rules from model trees will be tested for 10 lags, without nominal attributes.

The performance of the regression models is shown on Table 5.5. This table shows the error rate on the test data set as well as on the training one. Linear regression has the highest error rate on the training data set, while it performs much better than the other models on the test data. This can be explained by generalization issue: The more complex models "learned" training data better, they fitted the training data set too well, that's why they performed worse than simple linear regression model for new instances. This can also mean that the electricity consumption data is not very complex. At this moment it is just a hypothesis and needs further analysis. Let's wait until we will see how other models perform on this data.

Attributes	mae	rmse
	4.0588	6.7703
Weekday	4.0508	6.7454
Weekday/Month	4.0391	6.7284
Weekday/Month/Season	4.0393	6.727

Table 5.4: Linear regression performance with different attributes for 10 lags, on test data

	Training error		Test error	
Model	mae	rmse	mae	rmse
Linear regression	4.062	5.5212	<u>4.0588</u>	<u>6.7703</u>
Regression tree	3.9286	5.2851	4.3727	7.1452
Model tree	3.9217	5.2291	4.1857	7.0236
Rules from model trees	3.9699	5.2606	4.2201	7.2399

Table 5.5: Models' performance with 10 lags on test data

5.4 K-nearest neighbour

All the regression models used in Chapter 5.3 represents eager learner algorithms. This means that they begin to construct a general target function as soon as training examples are provided. Another type of algorithms are instance-based learning methods. They don't begin to construct the model immediately, instead they save training examples. The actual calculation happens when new instance arrives, to forecast its target value and than the relationships of this example towards the training examples are measured [19]. Instance-based learning contains K-nearest neighbour. Sometimes it is also called as lazy learning, as it postpones the work until the new query arrives. The advantage of lazy learners is that they treat each new example individually, it can act differently for each new instances.

	Training error		Test error	
K	mae	rmse	mae	rmse
1	0.0115	0.7499	6.1353	9.2832
2	2.9056	3.9351	5.3818	8.3094
4	3.5478	4.777	4.9307	7.7685
6	3.7633	5.0677	4.771	7.6164
8	3.8627	5.2146	4.6957	7.5325
10	3.9313	5.3094	4.6515	7.4906
12	3.9741	5.3737	4.6065	7.4686
14	4.0125	5.4309	4.6048	7.4703
16	4.0406	5.4757	4.5827	7.4505
18	4.0697	5.5141	<u>4.575</u>	<u>7.4458</u>
20	4.0916	5.5471	4.5788	7.4488
22	4.1114	5.5744	4.5803	7.4593
24	4.1261	5.5988	4.586	7.4712
26	4.1454	5.6237	4.5962	7.4781

Table 5.6: K-nearest neighbor performance foe different number of neigbors

The K-nearest neighbour algorithm that was used here is non-weighted. During using non-weighted algorithms noise can greatly affect algorithm accuracy. One way to deal with the problem is to find an optimal constant k and leave it predetermined, so only the majority of the class will influence the output, not a single nearest neighbor. Also the complexity of the data influenced the optimal value of k. So to summarise it briefly: The more noise in the data, or the more complex the data is the greater is the value of k. The number of k depends on the data qualities, that's why it is hard to determine it beforehand.

The experiments with different number of k were conducted with k-nearest neighbor and the the model performance on the test data was used to find the optimal k. The results are shown on Table 5.6. The best performance was achieved for $k = 18$. The models with smaller k "learn" the training data too well, while their performance is not good enough on the test data. The difference between the test and training error decreases as the value of k increases. This fact and the big value of k for the best performing model, suggests that there is a noise in data.

5.5 Support vector machines

SVM for regression problem is implemented in Weka as SMOreg. It has an parameter C which is a width of the tube. This parameter is defined by a user, so to find the optimal value of C a set of experiments should be conducted. I have tried out RBFKernel (Radial Basis kernel function) with different values of the C parameter to find the optimal one.

As you can see from Table 5.7 for this data set the optimal value of C is 2. It is true that the model performs slightly better when $C = 5$, but the difference is not significant. As described in chapter 3.2.4 the wider the tube is, the model has more number of support vectors. This means that it will be more complex and will need more time for calculations. That's why all the SVM models will be tested for the $C = 2$ tube width. So for the other kernel function we used this value.

C	mae	rmse
0.1	4.5562	7.2807
1.0	4.068	6.7868
2.0	4.0584	6.7841
5.0	4.0569	6.7953

Table 5.7: Performance of the RBFkernel for different values of tube width C on test data

The performance of the SVM with the polynomial kernel function is shown on Table 5.8. The variable *exponet* makes SVM model easily adaptable. For the data that can be separated by line the optimal value will be $exponent = 1$. For more complex data higher value of exponent should be used. So, according to this, the value of *exponent* measures the complexity of the data. For electricity consumption data the optimal value of this variable suggest that the data is not complex. Another proof on this matter is the high correlation between the observations.

The Pearson VII kernel function is a general kernel function and the model behavior changes by changing the value of ω. The performance of PUK kernel function is shown on the table 5.9.

Exponent	mae	rmse	time to build the model(sec)
1	4.0608	6.7708	847.51
2	6.4208	10.8618	4044.53
4	7.02018	11.07126	9134.54

Table 5.8: Performance of the polykernel for different values of tube width C on test data

ω	mae	rmse
0.05	4.2656	7.0298
0.1	4.4155	7.15
0.5	4.2405	7.1179
1	4.1998	7.1327
5	4.1787	7.1277

Table 5.9: The performance of PUK for different values of ω

The table 5.10 shows the final results of optimal models' performance. As you can see there is not big difference between the models with difference kernel functions.

5.6 Artificial neural networks

As already discussed in chapter 3.2.5 to apply neural networks for electricity consumption data set Matlab neural network toolbox is used.

Matlab neural network toolbox can train the class of neural networks that are called the Layered Digital Dynamic Network (LDDN).Such a network has the following parts

1. Layers and weights associated with the layers
2. Bias vector
3. Net input function
4. Transfer function

Two types of neural networks will be used for electricity consumption data set: static and dynamic. Let's first try out dynamic networks.

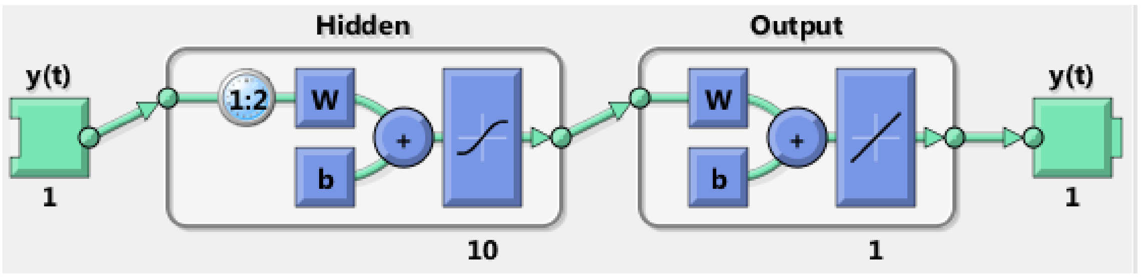

Figure 5.10: The architecture of open-looped NAR with 10 hidden nodes and 2 delay

kernel function	mae	rmse
PUK	4.1787	7.1277
polykernel	4.0608	6.7708
RBFkernel	4.0569	6.7953

Table 5.10: The performance of the SVM models for different kernel functions

5.6.1 Feedback neural network

Matlab offers three types of architectures for neural networks for time-series forecasting: NARX (nonlinear autoregressive neural network with external input), NAR(nonlinear autoregressive neural network) and nonlinear input-output without feedback loop. First I will use NAR neural network for data without external values. This type of networks have two variables: number of hidden nodes and delay. Delay is the number of lags that are fed to network at each step. As NAR takes into account the lag feature of the time-series data, electricity consumption data set does not need modification before NAR is applied to it. So the input of this model is a vector $(y_1, \ldots, y_N)$, where N is the number of observations.

The next step is to divide the data into three data sets: training, validation and test data. With the difference from all the methods that have been already applied for electricity consumption data set, neural networks use error on validation set as a stopping criteria during training.

For the purpose to make neural networks compatible with already used methods, it's performance should be measured on the same size of test set. Because of this for the data division the following ratio was used: $70\% - 15\% - 15\%$.

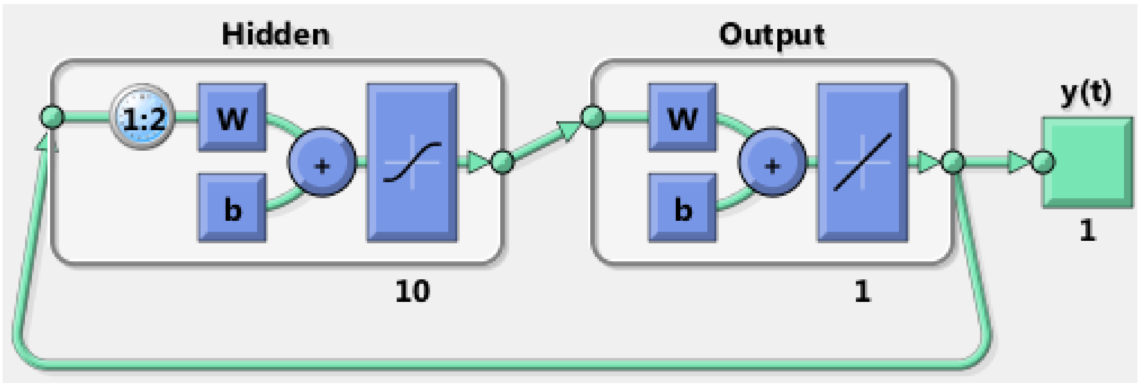

Figure 5.11: The architecture of closed-looped NAR with 10 hidden nodes and 2 delay

The standard network is a two-layer feedforward network, with a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. NAR is implemented in MATLAB without feedback loop. So, it does not really differs from the well-known feedforward net. The only difference is format of the input and training algorithm. The example of the NAR network is shown on the picture 5.10.

Also one interesting function in matlab which is in neural network toolbox for time series prediction is 'closeloop'. This function adds loop to the NAR network and transforms open-looped network to closed one. This loop does not affect the training process of the network, as it is added after the weights are adjusted.The closed network can't be used during training and validation period. This makes pretty obvious that this feature was added only for prediction purpose.

Also it works fine for one-step-ahead forecasting networks, there is a special recommendation that it should be used for multi-step-ahead prediction [17]. The architecture of the close-looped network is shown on the picture 5.11. In conclusion, to apply the close-looped NAR network to electricity consumption data set, first it should be trained and validated with open loop and after the loop should be added for forecasting purpose.

For this experiment I first have trained NAR for different number of hidden neurons and delay. After this I have built up closed network and have checked the performance with open and close loops for the same network.

Number of hidden neurons	Delay	Open-looped NAR		Close-looped NAR	
		mse	mae	mse	mae
10	1	34.2737	4.1604	1.0120e+003	24.9317
10	2	33.7921	4.1669	975.3444	24.8664
10	5	33.2711	4.1470	955.6156	25.5202
10	10	30.3588	3.9698	1.2271e+003	26.5553
10	24	27.2424	3.7572	1.0413e+003	24.9866
20	1	37.3455	4.2456	1.9826e+003	34.2499
20	2	33.2888	4.1466	951.0111	25.0198
20	5	32.0352	4.1101	979.1591	24.8851
20	10	30.1289	3.8929	1.4356e+003	32.7618
20	24	27.8308	3.7088	2.5742e+003	40.6628
40	1	33.9500	4.1710	980.7844	24.8633
40	5	33.1313	4.1319	1.3509e+003	31.7970
40	10	30.7372	3.8858	2.4309e+003	39.0767
40	24	25.2274	3.6660	2.3104e+003	37.7663
60	5	34.6006	4.0761	4.2564e+003	58.7436
60	24	25.5736	3.6817	1.7179e+003	35.9289
100	5	32.9371	4.0352	4.6686e+003	61.8782
100	24	27.2778	3.7031	4.5274e+003	60.5687

Table 5.11: The performance of open-looped and close-looped NAR with different number of hidden nodes and number of delay

As table 5.11 shows the best performance was achieved by the NAR neural network with 40 nodes on hidden layer and 24 time delay. From close-looped networks, the network with 10 hidden nodes and 2 time delay outperformed other architectures.

The performance of close-loop NAR comparing with open-loop NAR is much worse for one-step-ahead prediction.

5.6.2 Feedforward neural network

Feedforward neural network is generally most popular neural network. That's why it was applied to electricity consumption data set. This neural network is trained by Levenberg–Marquardt algorithm For hidden layers Tan- Sigmoid function is used and linear function is used as an output function.

The data needs modification before it is fed to feedforward neural network as unlike NAR it is not designed for time-series data. Experiments with NAR models showed, that the best performance from the models with the same number of hidden nodes achieved the models with 24 time delay. It worth mentioning that 24 time delay is natural cycle in the electricity consumption data set. Because of this the data was transformed so that the neural networks will use 23 previous observation for prediction.

I trained the model with Levenberg–Marquardt training algorithm because, this algorithm works quite well on small networks, and has fast convergence on them. The command to use this algorithm in Matlab is *trainlm*.

In neural network training set is used for computing the gradient and updating the network weights and biases while the error on the validation set is monitored during the training process. In this case validation set is used for stopping criteria. The test set error is not used during network training but is used for comparing different models. For example if you compare two neural networks the network with lowest error on test data is the one which generalizes better. The performance error for "trainlm" function is mean squared error (mse). It is used during updating network weights. I also calculate mean absolute error (mae) so to compare the performance of the neural network with other models.

To split dataset I use net.divideFcn = 'divideblock'. This means that data is divided into train, validation and test data sets as three contiguous blocks of the original data set. It makes sense when working with time series data. But inside the blocks data is chosen in random order anyway. The same proportion was used for splitting the data as for NAR model.

Overfitting is sometimes an issue when woriking with neural networks. Overfitting means that neural network's performance improves over iterations on training data set, while it is regressing on validation data set. It is interesting and also expected that there is no overfitting issues for this time series data set, even when NN is getting more and more complicated and the number of nodes grows. This can be explained by a big correlation between instances.

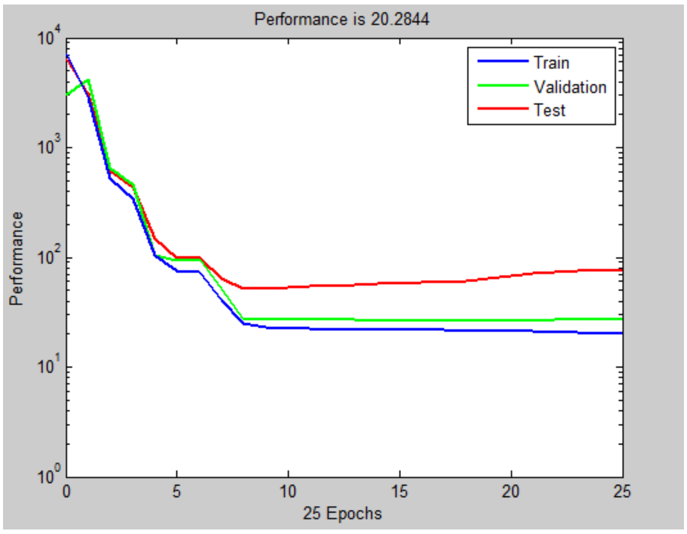

Figure 5.12: The performance of feedforward([20 20])

Figure 5.12 shows the performance of one particular neural network. For this model that neural network's performance on the validation data is not improving, while the performance on the training data set is not improving either. This is the case for every network that I have built for this data. So the only thing that I should be measuring is the test set performance. Somehow because of high correlation between the instances, overfitting is not a problem for feedrotward neural networks which were trained and validated for electricity consumption data set.

architecture	mae	mse
10	<u>4.0711</u>	<u>46.7529</u>
20	4.3196	69.3089
30	4.0342	48.8243
40	4.8454	138.8
[10 10]	4.2996	70.5520
[20 20]	4.3197	63.3597
[30 30]	4.2110	56.1015
[40 40]	4.4594	60.1079
[50 50]	4.5017	72.9662
[10 10 10]	4.2760	62.3395
[20 20 20]	4.4629	92.1067
[30 30 30]	4.4374	64.8248
[40 40 40]	4.3450	56.1008
[50 50 50]	4.5940	58.5713

Table 5.12: The performance of different architecture feedforward neural network on test data

As shown on the table 5.12 for one-layer neural network fewer nodes achieved better performance than any other model. The best performing network with one layer is the one with 10 nodes. Adding nodes does not makes model perform better. Though there is the twist for 30 nodes. This should be a result of neural initialization.

Even if one has two neural network models with the same architecture, built for the same data, they may behave absolutely differently. Sometimes the difference between these methods is quite significant. The data for two-layer neural network models is easier to interpret. The first two-layered neural networks that have been built, has 10 nodes in each layer. After this 10 nodes have been added to each layer. This action caused improvement of the performance. After [30 30] model performance begins to decrease. This suggests that the best two-layered neural network is the one with 30 nodes in each layer. It is interesting that the best network with two layers can't outperform the best method with one layer, which is the simplest one from the models that I have built for this data. On the other hand best three layered neural network can not outperform best two layered network.

5.7 Summary

This chapter represents the experiments and analysis that was conducted for electricity consumption data set. To answer the research question and find the most accurate model for this data set, on the first place the data was analyzed (Chapter 5.2). The statistical analysis revealed that the electricity consumption data set has trend as well as seasonality. Using this information the best fitting statistical model was found, which is $SARIMA(2,1,2)\times(1,1,0)_{24}$. In chapter(5.3) regression models were applied to electricity consumption data set. As discoursed in chapter 3.1.1 before applying machine learning models to time-series data, the data itself should be transformed. In purpose to find the best data transformation experiments were conducted and the data with 10 lags was chosen. After this nominal attributes were added to the data and analyzed if those have any influence on the model accuracy. Adding nominal attributes does not significantly improved the models' performance. From regression models linear regression performed best, the second was model trees.The performance of the rules from model tree and regression tree does not difference much. As for k-nearest neighbour algorithm, the model with 18 neighbours achieved the best performance. The big value of the k suggests that there is a noise in electricity consumption data. Chapter 5.5 describes how different support vector machines with polynomial, Pearson VII and Radial Basis kernels were applied to electricity consumption data set. As experiments showed, choosing of the kernel function does not influence the model performance for this data set. Also when SVM with polynomial kernel was applied, the model with lower exponent performed better than more complicated one. Chapter 5.6 describes how feedback and feedforward neural networks have been applied to electricity consumption data set. From dynamic models the best performance has the NAR model with 40 hidden nodes and 24 time delay. The closed-loop NAR models performed much worse than opened-loop models. The best performing closed-loop model was the one with 10 hidden nodes and 2 time delays. As for feedforward networks the models with more complicated architecture and more nodes on the hidden layers, performed worse. The most accurate model was the one with one hidden layer and 10 nodes.

Chapter 6

Results

The goal of this work was to find the best performing model for the electricity consumption data, to find out whether statistical or machine learning models are more accurate. Finding the best performing model is the answer of the first research question. The second research question is more oriented on the features of the electricity consumption data. It might seem that this two research questions are concentrated on different things, but they are highly correlated as analysing the performance of these models can reveal some features about the data.

Model	mae	rmse
NAR open-looped	3.6660	5.0226
Linear regression	4.0588	6.7703
Support vector machines	4.0608	6.7708
Feedforward neural network	4.0711	6.8376
Model trees	4.1857	7.0236
Rules from model trees	4.2201	7.2399
Regression trees	4.3727	7.1452
K-nearest neighbour	4.575	7.4458
NAR close-looped	24.8664	31.2305
$SARIMA(2,1,2) \times (1,1,0)_{24}$		48.33502

Table 6.1: Performance of the statistical and machine learning models on the test data

Table 6.1 represents the final results of the experiments. The models are ordered by their performance on the electricity consumption data set. Each model was the best performing model in the one variety group of models. For example the support vector machines that is represented here, outperformed all the support vector machine models with different kernel functions, with different variables of the kernel functions and with different tube width of the model.

NAR open-looped

The best performing NAR model was the network with 40 nodes on hidden layer and 24 time delay.

From all the models that has been applied to the the electricity consumption data, nonlinear autoregressive neural network achieved the best performance. To find the best fitting model the experiments were conducted with different number of time delay and hidden nodes. All the models had a sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. Also all of them were trained by Levenberg–Marquardt algorithm. The experiments were done with 10,20,40,60 and 100 hidden nodes and 1,2,5,10 and 24 time delay. I stopped adding the number of nodes in the hidden layer when the performance begin to decrease. Also for all the models with the same number of hidden nodes the model with 24 time delay outperformed the others. This is not unexpected if taking into account the results of the statistical analysis and the fact that the frequency of the electricity consumption data is actually 24.

Linear Regression

The optimal linear regression model was the one which was applied to 10 lagged electricity consumption data set without nominal attributes.

Generally linear regression is not developed for time-series data and because of this the data was modified before linear regression was applied to it. This is discussed in Chapter 3.1.1; The input matrix should have the following form $[(N - n - 1) \times n]$, here n is a lag. As experiments showed the linear regression had the lowest error on the test data for 10 lags. To improve the model performance after this the *weekday,month* and *season* nominal attributes were added to the data and again test error was used to measure if those attributes are sensitive. As experiments showed those attributes did not influence the performance accuracy. The resulting linear model showed that the biggest weight in this model had the last lag and it's almost equals to 1, which means the next observation almost completely depends on the previous one.

Support vector machines

The optimal support vectors machine (SVM) model was the one with polynomial kernel function when $exponent = 1$ and with the tube width 2.

The width of the tube is one of the valuables for SVM model and it should be carefully chosen. When the tube is too wide, the model will have many support vectors and it might have overfitting issue. The flatness of the tube ensures that model will not overfit.To find the optimal value the experiments with Radial Basis kernel function were done with different tube width values. The optimal value was 2. SVM in Weka has three different kernel functions: Polynomial, Radial Basis and Pearson VII. Polynomial kernel has a variable exponent. The variable exponent makes SVM model easily adaptable. Obviously for more complicated data exponent is higher, while for the data that can be separated by line the optimal value will be $exponent = 1$. For SVM with polynomial kernel the best performance was achieved when $exponent = 1$. The performance rapidly decreased for the higher polynomial order. Pearson VII performed worse from all the kernel functions. SVM models with Radial Basis kernel function and polynomial kernel when $exponent = 1$ had the best accuracy.

Feedforward neural network

The best performing neural network was the one with one hidden layer and 10 nodes on hidden layer.

The feedforward neural network is not originally developed for time-series, because of this the data was modified before feedforward neural network was applied to it. The modified data with 24 lags was fed to a feedforward network. I took the data with 24 lag, because it showed the best performance for NAR network and these two networks have the same architecture. The overfitting was not an issue for feedforward neural networks that were trained and tested for this data set. This can be result of high correlation between the observations in the electricity consumption data. Generally models with fewer layers performed better that models with more layers. Also networks with the same number of layers but fewer nodes achieved better accuracy than with more hidden nodes.

Model trees, Rules form model trees and Regression trees

Regression tree, model tree and Rules form model tree were applied to the modified data with 10 lags. From all these models, model trees outperformed others.

K-nearest neighbor

The best performing K-nearest neighbor model was the one with 18 neighbors.

To find the optimal number of neighbours the experiments were conducted for different values and the test and training errors were measured. As the number of neighbours increased, the model performed better on the test data and worse on the training data until it reached 18. So the model with 18 neighbors was chosen as an optimal one.

NAR close-looped

The best performing close-looped NAR neural network was the one with one hidden layer, with 10 nodes on the hidden layer and 2 time delay.

This neural network has the same architecture as the open-looped NAR or feedforward neural network has. It has the sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. It is trained in the same way as NAR. The difference is that it has close loop. Close-looped NAR is a feedback network, which means that the output of the network is feed back as input to the next step. This model has a longer 'vision' of the past observations, but for one-step-ahead forecasting almost every other model except of ARIMA, outperformed it.

$SARIMA(2,1,2) \times (1,1,0)_{24}$

The best performing Box and Jenkins model was $SARIMA(2,1,2) \times (1,1,0)_{24}$.

Estimating the right model for electricity consumption data was a long process. First the data was analysed for stationarity. As analyses showed the data had both seasonality and trend. Because of this the data was differenced two times, one to remove seasonality and one to remove trend. After this autocorrelation (ACF) and partial autocorrelation functions (PACF) were analysed for the resulting data to have some idea about the model variables. The final decision was make according the models' performance on the test data.

Chapter 7

Discussion

In this work the different models have been analysed and applied to the electricity consumption data. The performance of an particular model very much depends on the data. Statistical methods have a longer history in comparison with machine learning techniques [12]. Even though Machine learning replaced statistical methods for many applied problems [15]. The first research question is about whether statistical or machine learning models will perform better on the electricity consumption data set. The second research question is about data features. This two research questions have a crossing point as generally the model performance greatly depends on the data characteristics.

Statistical performance showed that the electricity consumption data set has both trend and seasonality. Also it is arguable that the data has noise in it. One of the sign of noise is the performance of K-nearest neighbor model. The best performing model was the one with 18 neighbors. The big value of neighbors can be conditioned by either nose in the data, data complexity or by both of them. If take in mind that the data points are highly correlated, it is more likely that the reason is nose in the data.

Another prove of noise in the data is the performance of linear regression,model trees, rules from model trees and regressions tree models. Linear regression is the simplest model of them all, it performs well only when the data is separable by the line. Model threes is more advance than linear regression and generally it performs better for non-linear data than linear regression. Statistical analysis showed that the electricity consumption data set has seasonality and trend, which means that the data is not separable by the line. It should be reasonable if model trees outperformed linear regression, but it did not. It can be caused by noise in data. When the data has big noise in it, simple linear regression model can generalize much better on the test data that regression trees, or model trees.

The best performance was achieved by NAR network, while Box–Jenkins model had the worst performance on the electricity consumption data. The fact, that neural networks achieves such a good accuracy is not unexpected. For electricity consumption data set NAR outperformed all the other models. ANN is a universal function approximator, it has been proven that two layered neural network can approach any predefined accuracy for any complex function [24]. Neural networks have generally more parameters [24]. This is true in my case as well. Actually the best performing model (NAR with 40 hidden neurons) has more parameters than any other model has. Artificial neural networks has already been used for time-series forecasting. In particular, ANN outperformed all the other models in M-3 time-series competition [24]. ANN had outperformed ARIMA before. In studies conducted by Tang et al. (1991) and Kohzadi et al. (1996) neural networks performance was considerably more accurate. Also it captured more sudden changes in data [12].

If we will generally compare neural network with ARIMA, neural networks are non-linear, while Box-Jenkins methods assumed that the linear process had generated the time-series data, to which it is applied. This is the main drawback of ARIMA model, for complex data it can be inappropriate [12].

Linear regression has been tested for different number of lags. This model performed the best for 10 lags. Also the model showed that the highest rate had the last observation. This can be caused by the simplicity of the model. Obviously the data has 24 frequency. It might be the case that even if there is dependency, it is lower for far observations and linear regression could not capture it. The same can be said about model trees and regression trees. Neural network is much more complex and it is known with its ability to model complex data. That's why NAR used the whole input and had the best performance for 24 delay.

Linear regression and ARIMA both are linear models. Even though linear regression is on the second place according to its performance, while ARIMA was the model with the highest error. ARIMA tries to learn the trend and seasonality in the data and is using it for forecasting. It can't capture sudden changes. This is the main drawback of the statistical models in comparison with machine learning models.

We talked about the strength of neural networks, the way it can learn non-linear dependencies in the data. Only Neural networks are a big area by itself. There are many options. Different models can be built by choosing different architectures, learning algorithm, also by choosing different functions on the hidden or output layer. Here I have used three different types of neural networks for electricity consumption data set: open-looped NAR, close-looped NAR and feedforward network. Each of them achieved different accuracy.

Let's first compare open-looped and close-looped NAR. Both models have been trained with the same algorithm. More precisely, to get the close-looped NAR, on the first place open-looped NAR is built and trained and only after this is added loop to the model. This arises the following question: Why these models performed so differently? This can be caused by the only difference between these models - feedback loop.The difference is their performance on the test data, as one is using feedback loop, while another does not. When open-looped NAR is forecasting on the test data it uses the real values as time delays for input at each step, while close-looped network is using forecasted values on the previous steps as input. So we are dealing with the *Butterfly effect* with close-looped networks. The forecasting error at each step is fed back to the network and this makes error grow. Close-looped NAR is recommended for multi-step ahead forecasting when feedback loop is actually needed. It does not make sense for one-step ahead forecasting. That's why it performed much worse that open-looped NAR. Another interesting thing is that open-looped NAR performed best for 24 time delay, while close-looped NAR achieved the lowest error for 2 time delay. For close-looped NAR when time delay is not zero, it used the same observation two times, one real value, and one fed from beedback loop. For input the lagged values is not necessary as tbe network gets some information about them from feedback loop. This is the reason why the time delay for close-looped NAR was so little, it was equal to 2.

Now let's compare the performance of the open-looped NAR and feedforward network and discuss why they showed so different performance on the electricity consumption data set. The data was transformed before it was fed to feedforward network. It was transformed so that to be compatible to 24 time delays. I took 24 lag because of two reasons. First it had the best performance on NAR model with open loop. The second is that the frequency of the electricity consumption data naturally is 24, it was shown during the statistical analysis, also neural networks can map the complex dependency between the data observations, in different from linear regression and trees. The NAR and feedforward networks with the same architectures achieved different accuracy. The difference is caused by training algorithm. Dynamic neural networks are trained with the same algorithms as multilayer neural networks but they are more complicated because weights for dynamic network has long term effect as well as immediate on the output. So for dynamic networks dynamic backpropagation algorithm is used. This algorithms is more complex and it can be easily trapped in the local minimal. Sometimes simple backpropagation algorithm is enough to train dynamic neural network. This pretty much depends on the data. The matlab handles it automatically. It takes decisions which one will be used and user can't choose. Advanced training algorithms for neural network that is specially modified for time series forecasting is the answer why open-looped NAR achieved the best accuracy from all the models that were applied to the electricity consumption data.

Chapter 8

Conclusion

In this work summed up electricity consumption data set for one substation in the Hvaler (kommune) is analysed. This work represents case study and it aims to find the best performing model for this data set. The main domain of time-series forecasting have been statistical analysis for a long time until machine learning models became more popular. So the first research question is about comparing statistical models with machine learning models for electricity consumption data set and than analysing the outcome. As turned out all the machine learning models that have been applied to this data set outperformed box-jenkins autoregressive integrated moving average (ARIMA) model. This can be caused by the fact that Box-Jenkins methods assumed that the linear process had generated the time-series data, to which it is applied. It can be quit inappropriate for process that is generated by non-linear process. Also ARIMA tries to to learn the trend and seasonality in the data and is using it for forecasting. It can't capture sudden changes. This makes ARIMA not robust for noise and picks in data.

The best performance achieved open-looped nonlinear autoregressive neural network on the electricity consumption data set. This was not a sudden outcome as neural networks are universal approximators and NAR is specially developed and trained for time-series forecasting.

The second research question was about the electricity consumption data itself, whether or not it has seasonality, trend or noise in it. As statistical analysis showed the data has both, seasonality and trend. The behaviour of the models show that the data is quit noisy. The fact that linear regression outperformed model trees can have only two reasons, either data is linearly separable or either the data has noise in it and simpler model can generalize better. The statistical analysis showed that data has trend and seasonality which means that it is not linearly separable. This means that the data had noise in it.

Bibliography

[1] Esmart systems official web-site. `http://esmartsystems.com/`. Accessed April 14, 2015.

[2] A. Messineo A. Marvugliaa. Using recurrent artificial neural networks to forecast household electricity consumption. *Elsevier*, 14:45–55, 2012.

[3] Christopher Bishop. *Pattern recognition and machine learning.* Springer, New York, 2006.

[4] K. Boran. The Box Jenkins Approach to Forecast Net Electricity Consumption in Turkey. *Taylor & Francis Online*, 36:515–524, Feb 2014.

[5] George Box. *Time series analysis : forecasting and control.* John Wiley, Hoboken, N.J, 2008.

[6] L.M.C.Buydens B.Ustun, W.J.Melssen. Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel . *Chemometrics and Intelligent Laboratory Systems,*, 81:29–40, 2006.

[7] Jhao-WunHu Chunshien Li. A new arima-based neuro-fuzzy approach and swarm intelligence for time series forecasting. *Engineering Applications of Artificial Intelligence*, 25:295–308, 2012.

[8] Avril Coghlan. *A Little Book of R For Time Series.* Release 0.2, 2014.

[9] Jerrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[10] Akhter Mohiuddin Rather et al. Recurrent neural network and a hybrid model for prediction of stock returns. *Elsevier*, 42:3234–3241, 2014.

[11] Fi-John Chang et al. Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control. *Elsevier*, 517:836–846, 2014.

[12] Guoqiang Zhang et al. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14:35–62, 1998.

[13] Suicheng Gu et al. Recentness biased learning for time series forecasting . *Information Sciences: an International Journal*, 237:29–38, 2013.

[14] Souhaib Ben Taieb Gianluca Bontempi and Yann-Ael Le Borgne. Machine Learning Strategies for Time Series Forecasting . *Springer Berlin Heidelberg*, 138:62–77, 2013.

[15] Jennifer Marohasy John Abbot. Input selection and optimisation for monthly rainfall forecasting in Queensland, Australia, using artificial neural networks. *Elsevier*, 138:166–178, March 2014.

[16] H. White M. Stinchcombe. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[17] H.B. Demuth M.H. Beale, M.T. Hagan. *Neural Network Toolbox^{TM} User's Guide.* MathWorks, 1992-2015.

[18] Jae H. Min and Young-Chan Lee. Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters . *Elsevier*, 28:603–614, 2005.

[19] Tom Mitchell. *Machine Learning.* McGraw-Hill, New York, 1997.

[20] Neamat El Gayarc Hisham El-Shishiny Nesreen K. Ahmeda, Amir F. Atiyab. An Empirical Comparison of Machine Learning Models for Time Series Forecasting . *Econometric Reviews*, 29:594–621, 2010.

[21] T. Ozaki. On the order determination of arima models. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 26:290–301, 1977.

[22] Ananth Ranganathan. The levenberg-marquardt algorithm. 2004.

[23] Sandhya Samarasinghe. *Neural networks for applied sciences and engineering : from fundamentals to complex pattern recognition.* Auerbach, Boca Raton, FL, 2007.

[24] J. Keith Ord Sandy D. Balkina. Automatic neural network modeling for univariate time series. *Elsevier*, 16:509–515, 2000.

[25] F. W. Scholz. Maximum likelihood estimation. *Encyclopedia of Statistical Sciences*, 2006.

[26] Mohammad Anwer Shipra Banik, A. F. M. Khodadad Khan. Hybrid machine learning technique for forecasting dhaka stock market timing decisions. *Computational Intelligence and Neuroscience*, 2014.

[27] Robert Shumway. *Time series analysis and its applications with R examples.* Springer, New York, 2011.

[28] G. Unakitan and B. Türkekul. Univariate Modelling of Energy Consumption in Turkish Agriculture. *Taylor & Francis Online*, 9:284–290, Oct 2013.

[29] Rafał Weron. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *Elsevier*, 30:1030–1081, 2014.

[30] I. H. Witten. *Data mining : practical machine learning tools and techniques.* Morgan Kaufmann, Burlington, MA, 2011.