Forecasting of a hydropower plant energy production

Master's Thesis in Computer Science

Ksenia Dmitrieva

May 13, 2015
Halden, Norway

Østfold University College www.hiof.no

Abstract

This thesis presents several machine learning and a few statistical methods applied for time series forecasting. The time series used in this work as a case study is a result of the energy production of two small hydropower plants. The amount of produced energy is measured each hour during almost two years. The hydropower plants are located in Hyen, Western Norway. One of them is located in mountains with rocky ground so it means that the effect of rain should be immediate and short. The other one is located further in the valley with the pond nearby which means that the stream has water almost during all year and the effect of the rain is not that immediate. Forecasting of energy production for such small power plants is an essential tool for ensuring ongoing power supply for user demand, planning of reserve power supply and transaction between power plants. Applying several techniques to the datasets we have tried to determine the best method to build a one-step-ahead prediction model. Also we have been aiming to estimate the impact of different attributes on the forecasting process.

Keywords: time series, prediction, forecasting, machine learning, regression, ARIMA, R, Matlab, SVM, Artificial Neural Networks, energy production, hydropower plant

Acknowledgments

First and foremost I would like to express my sincerest gratitude to my supervisor, professor Jan Roland Olsson, for the useful comments, remarks and engagement through the learning process of this master thesis. This thesis work would not be possible without his proper guidance. I am also grateful to Professor Øystein Haugen for spending time read this thesis and providing useful suggestions about this thesis. I would like to thank members of eSmart systems, especially Sigurd Seteklev and Dang Ha The Hien for providing data and supporting during this work. Also I want to thank my classmate and friend Sopiko Gvaladze for sharing some thoughts and previous work on the theme, who has supported and encouraged me throughout the entire process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Increasing worldwide demand of electricity cause rapid deployment of renewable, or green, energy – energy that comes from natural resources – which results in significant energy sustainability and economic benefits. An energy production forecast, which gives information about how much energy will be produced by a certain power station can be useful for optimising the marketing of a renewable energy and hence deploy systems integration. Yet, renewable energy production depends a lot on environmental conditions such as precipitation, wind power, geothermal heat, etc. As a result, the energy production of such systems is fluctuating and their feed-in into a power grid is difficult to forecast.

1.1 Motivation

The problem of energy production forecasting needs to be addressed as the contribution of green energy to the grid is constantly rising. Some countries have laws which makes green energy prioritized to the energy produced from fossil fuels concerning about the environment. For some remote territories energy produced, for example, from a local small hydropower plant becomes the only source of energy.

For small hydropower plants the problem of production forecasting is a cornerstone of a sustainable development. To be a steady supplier in energy market for such small power plants means to be capable of providing a demanded amount of energy. In the meantime, to be ensure that the supplied power is fully utilized form a grid demand forecasting should be done which is also a time series problem. Short-term forecasting is an essential tool for ensuring ongoing power supply, planning of reserve power supply, carrying out energy transactions between power stations and so on.

Forecasting of energy production is a dynamic process. For example, forecasting for the next 48 hours requires constantly updating information about weather measurements, discharge, previous energy production, to ensure proper calibration of the system. However the control of energy production depends not only on uncontrollable power resource but on the manager strategies of a certain power plants which depends on technical restrictions, price instability, users' demand and so on. Yet all those information are not always available what makes a renewable energy forecasting being difficult to control. Furthermore for production forecasting given weather conditions the forecast of weather itself should be done which impact with some error on the final results as it always contain some error in it.

Our project is closely related to a project initiated by eSmart Energy Systems, Halden,

which includes forecasting for small hydropower plants. The available data was gathered for two hydro power plants located in nearby to each other – Hyen, western part of Norway. Even though they are near to each other, the surrounding topology differs so it might cause different effects on the energy production. The chief objective of the work is to estimate different approaches applied to energy production forecasting for small hydropower stations and determine the best forecasting model using the data gathered from this two hydro power plants as use cases. Furthermore the thesis work, in this context, attempts to address of finding those features which impact in a positive way on forecasting.

Forecasting essentially consists of the predicting the future state based on the previous, or historical, values. Our main purpose is to build a short-term forecasting model that will predict a next hour energy production. For that purpose we were kindly provided by the data by eSmart. This data consists of an hourly measured power production of each hydro power plant mentioned above, and, as an exogenous information, daily measured precipitation and temperature values gathered in a nearby town.

1.2 Research Question and Method

Given motivation mentioned above our main focus is to build an accurate model for energy production forecasting what involves comparing and analysing different approaches to time series prediction.

As we have mentioned above, the two power plants we are using as a case studies in this research are located in nearby to each other however have different topology. Thus one of the hydropower plants is located in a rocky mountains which means that the effect of the rain is short and immediate. The other power plant has a pond near it what means that the effect of the rain is not that immediate and have a long-term dependencies. Regarding this problem the first part of our research question concerns different techniques and their capabilities of finding short-term and long-term effect in a time series:

RQ 1 *Which approaches – statistical or machine learning (which also include a range of methods) – are better in predicting, and also finding short-term and long-term effects in time series*

Obviously those models which are capable of finding those long-term dependencies should be more accurate in prediction. Yet the impact of different features should also be analysed in order to find those which positively effect the accuracy of chosen models. For example, calendar data might shed some light on the calendar dependencies such as less energy production in winter month as precipitation is mostly in the frozen state. Hence the second part of our research question regards the exogenous information and its impact on the model building:

RQ 2 *Which exogenous features regarding energy production process in a hydropower plant are relevant for building an accurate prediction model*

Statistical models, namely ARIMA – Autoregressive Integrated Moving Average, do not use any exogenous input and traditionally used in time series analysis so can serve as a baseline for a comparison. In order to answer the research question first literature and case studies were analysed in order to determine optimal methods for time series analysis

from an existing variety of them. Second, a set of experiments was carried on involving variation on data structure and on configurations of chosen models. The obtained results were discussed and a further work proposed.

1.3 Report Outline

The rest of the thesis is organised as follows.

Chapter 2 provides background information on time series and forecasting. First, we define what is a time series, provide some background required for further analysis with statistical models. Then statistical modeling is described, namely ARIMA, and several relevant to this work machine learning approaches including regression analysis, neural networks and support vector machines.

The third chapter gives a related literature and case studies overview in order to explain why certain approaches have been chosen for forecasting and investigate some problems and concerns which may arise during experimentation session. As a result, our methodology is proposed. Experimentation described in Chapter 4. Also the forth chapter gives preprocessing steps, primary analysis and visualisation.

Chapter 5 provides a summary and discussion on the obtained results and presents suggestions for further work. Finally there is a chapter providing conclusion on the obtained results.

Chapter 2

Background

Since energy production forecasting requires good knowledge of time series analysis, statistics and prediction methods, in this chapter a proper overview of all of these related concepts is provided. The first part of this chapter deals with some theory of time series and forecasting. In the second part we discuss different methods for time series prediction: statistical and machine learning approaches, provide some mathematics, which becomes the basis of the experiments in Chapter 4.

2.1 Time Series

Energy is considered to be a key point in sustainable economic development and a necessity of a modern life. Sustainable development demands a sustainable supply of energy resources. Renewable energy sources, such as sunlight, wind, geothermal heat, hydropower, etc., are generally related to a sustainable ones considering relatively long-term periods of time. For example, for large-scale conventional hydroelectric stations (dams) which have water reservoirs electricity production can be flexible since stations turbine systems can be adjusted to adapt to changing energy demand. Energy production of a small and micro hydropower plants depends on the weather conditions like precipitation and temperature. As a result, the energy production of such systems fluctuates and need to be forecasted.

Time series are also used in a wide range of fields such as signal processing, pattern recognition, earthquake prediction, weather forecasting, econometrics, control engineering, electroencephalography, etc. Time series analysis became very popular in the last decades so that many modern database management systems have some basic instruments to handle time series [2].

A time series can be referred to as a collection of observations made sequentially and ordered in time: $y_1, y_2, \ldots, y_n$ [20]. So time series is a type of data gathered typically in some points in time space. It can be represented as some measured parameters of one (the simplest case) or several processes. The main difference of time series from other data is that when data being analysed, a dependence of those parameters at time points is taken into account. Formally, a time series is a set of pairs, each of them includes time value (when a parameter was measured) and the parameter itself. A time value must be unique, while parameters can have same values.

Examples of time series are the hourly measured temperature in some city, stock market indices, daily energy consumption, etc. Such kind of data is usually visualised as a line charts. The plot is vital as to describe the data and help in formulating a suitable model.

Thus the choice of scales, the size of intercepts, continuous or separate dotted lines can affect the plot looks like.

There are several types of time series. First, time series can be continuous and discrete, when observations are taken only at specific times, usually equally spaced, but sometimes spaces can be no equal. Second, we can distinguish time series according to the value types. Thus it can be one-dimensional or multidimensional (has several parameters at a given time point), or parameters can also be presented as a non-numeric variables [20].

Next, the process can be deterministic, if parameters of a time series can be predicted exactly, or stochastic, when the future is only partly determined by past values and exact prediction is impossible. In practice, the majority of time series are stochastic, and an idea of handling them is that future values have a probability distribution which is conditioned by a knowledge of past values [20].

One of the main characteristics of a stochastic time series is its stationarity (see Section 2.2). Stationary processes have no systematic change in mean and variance, while non-stationary processes do not. But the latter have trend, or tendency. Granger [32] defines trend as including all cyclic components when wavelength exceed the length of the observed time series. In other words it's a long term movement in a time series. The problem then is deciding what to mean under "long term". Trend can be observed in global and local scales.

In practice, real time series are generally nonstationary. Stationarity correlates with the length of a time series. Thus, on a short periods, nonstationary processes are more probable while long termed processes have usually weak stationarity as most parameters are finite and do not exceed some maximum value. However strong (or strict) stationarity cannot be observed because a process still evolve somehow.

On the Figure 2.1 you can see different types of processes: a – stochastic, b – stationary, c – nonstationary.

2.2 Stationary Processes

The main feature when designing time series models is an assumption that it is a stationary process. Usually a stationary stochastic process can be described by:

- its mean,
- its variance,
- and autocorrelation function (or spectral density function).

For a strictly stationary process such parameters as mean and variance do not change over time and the joint probability distribution of such process does not change over time. The mean of a stationary process is a level the process fluctuates about (Equation 2.1).

$$\mu = E[y_t] = \int yp(y)dy \tag{2.1}$$

The variance of a stationary process is the spread about this level (Equation 2.2).

$$\sigma_y^2 = E[(y_t - \mu)^2] = \int (y - \mu)^2 p(y)dy \tag{2.2}$$

Since the probability distribution is the same for each time, for such process the mean and variance can be achieved just from a sample of observations.

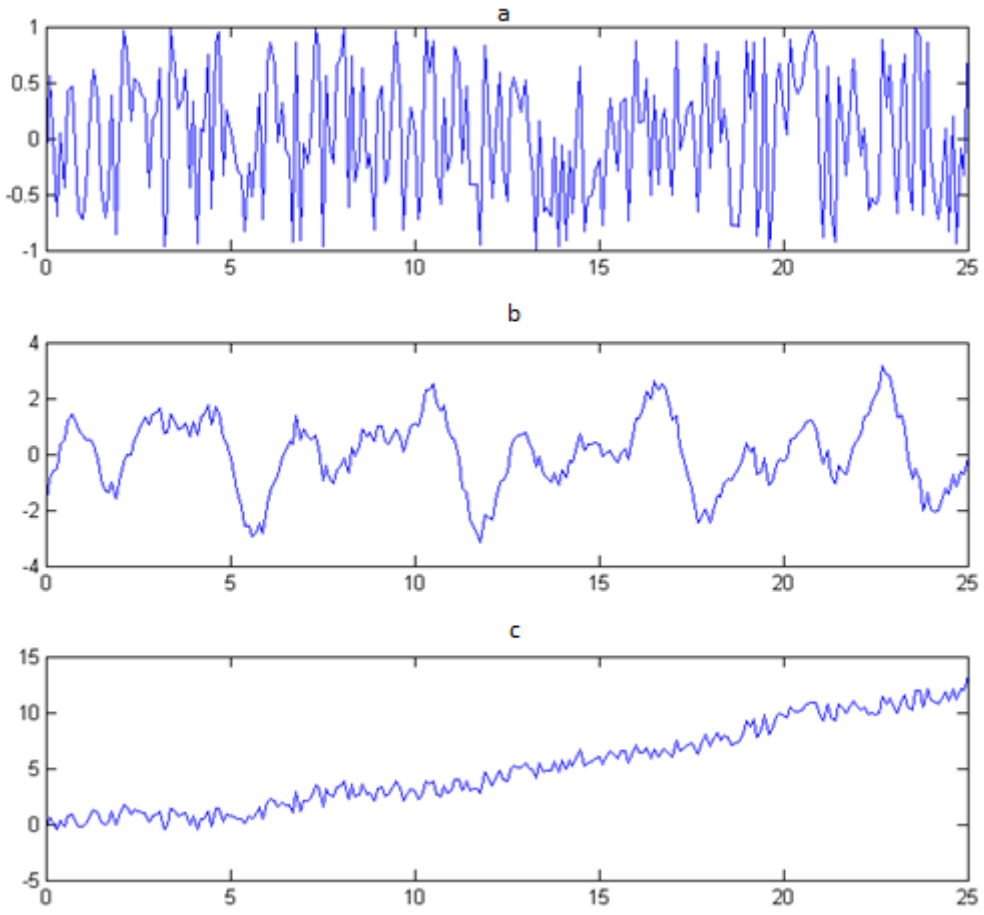


Figure 2.1: Examples of a – stochastic, b – stationary and c – nonstationary processes

To start with the simplest process let us consider a special case of stationary processes – white noise.

White noise is a random sequence $y_1, y_2, \ldots, y_n$ with the expected value $E(y_t) = 0$, where $t = \overline{1,n}$. All the elements of such process are uncorrelated and variance is constant $D(y_t) = \sigma^2 = const$.

White noise is a theoretical process, which does not exist in real life, but it can be useful as a mathematical model for solving practical problems. White noise can be transformed to a process Y_t using a linear filter. Linear filter is a sum of weighted previous random shocks a_t of the process [17]:

$$y_t = \mu + a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \ldots = \mu + \psi(B) a_t \tag{2.3}$$

where B is an operator of backward shift $By_t = y_{t-1}$, μ defines the level of the process (the mean about which the process varies), and $\psi(B) = 1 + \psi_1 B + \psi_2 B^2 + \ldots$ is a transfer function of the linear filter. For the process to be stationary the sequence $\psi_1, \psi_2, \ldots$ is to be convergent and then the filter is stable. Also such process can be called a general linear process.

Strictly stationary processes plays a great role in mathematical modeling however there is a little evidence of such processes in real life. For example, ARIMA models requires a time series to be stationary. However, usually real processes have trend, seasonal, random components what makes time series prediction task more difficult: for example for statistical modeling of a time series (ARIMA) all those components have to be retrieved and after that process has to be at least weak stationary (see Section 2.6).

2.3 Trend and Seasonality

Such components as trend and seasonality makes a time series nonstationary what makes their identification and adjustment crucial for statistical modeling and sometimes useful for machine learning modeling however it is better to allow machine learning models to find those dependencies themselves.

Seasonal effects

Seasonal components arise due to systematic and calendar related effects and defined as repetitive or predictable fluctuations over a period of time which include natural conditions such as weather fluctuations, social and cultural behaviour such as start and end of a school term or Christmas holidays and so on. Some examples are increase of an energy consumption in winter months or increase of sales in Christmas and holidays periods.

As seasonal components can effect some non-seasonal characteristics of a time series and some tendencies in it a seasonal adjustment is applied to a process. Seasonal adjustment is an estimating and then removing of a seasonal components. One of the problems why we cannot just compare certain periods of time is such effects as moving holidays: for example, Easter falls in different period each year – and working days: the number of working days in each months differs each year what influence on the activity levels.

Seasonality can be identified by a regularly spaced curve peaks or flatness with the same magnitude over the period. Sometimes the change of a time series magnitude can also relate to a change in the trend component of a time series. In some cases trend (or irregular components) can dominate seasonal components and then it is impossible to identify that little seasonality which presents in time series.

Trend

As we have described above, trend is a long-term increase or decrease in data. It does not always have to be only increasing or only decreasing: there can be a change in a trend direction. It also does not have to be linear: it can be multiplicative, exponential, logarithmic, etc. as well. As seasonal components trend also needs to be identified and removed from a time series.

2.4 The Additive Model of a Time Series

The additive model for a time series $y_1, y_2, \ldots, y_n$ is an assumption that all the values consist of four components: trend, long term seasonal influence, short term seasonal influence, and deviations (or errors):

$$Y_t = T_t + Z_t + S_t + R_t \tag{2.4}$$

where $t = 1, 2, \ldots, n$ [27].

Here T_t is a trend, a monotone function. Z_t and S_t stands for long term and short term seasonal influences respectively, which can be considered as a one variable $G_t = Z_t + S_t$ and described as a longterm behavior of a time series. R_t is a random variable summarizing all the deviations.

Usually models having all those components are non stationary processes which we can usually observe in a real life. There are different approaches describing them (as different functions which could describe a trend for example) and filtering them.

2.5 Time Series Prediction

The analysis of time series is based on the assumption that some knowledge can be retrieved from a series which would describe the initial process and, based on this knowledge, build its model for practical tasks: prediction of the future behavior of the process, its nature, anomaly detection, classification and so on. The possibility to detect and predict anomalies in time series is an essential part of a production process.

Formally, the task of time series prediction can be described through the task of function minimization. Let $y(t)$ be a function describing initial process. We want to find such function $\check{y}(t)$ which could describe the process as well, and we want the difference between $y(t)$ and $\check{y}(t)$ to be minimal. That is if $\check{y}_t$ is a value of a time series $y(t)$ at the given time point t and $\check{y}_t$ is a predicted value at the same time point, than we want the residuals $|y_t - \check{y}_t|$ to be minimal. Residuals contain information about how good a predictive model is. The accuracy can be described through confidence intervals with least squares estimates. It is obvious that the ideal solution is when $y(t) = \check{y}(t)$ and we can view time series prediction task as a minimization (or optimization) problem.

2.6 ARIMA Models

2.6.1 Autoregressive model

Autoregressive model (AR) considers a current value as an aggregate of a previous deviations and a shock a_t.

Let $\check{y}_t, \check{y}_{t-1}, \ldots$ – be deviations from the mean the process fluctuates about: $\check{y}_t = y_t - \mu$. Then the autoregressive process is defined as [17]:

$$\check{y}_t = \phi_1 \check{y}_{t-1} + \phi_2 \check{y}_{t-2} + \ldots + \phi_p \check{y}_{t-p} + a_t \tag{2.5}$$

where p is an order of the autoregressive process and a_t are random shocks (see Section 2.2).

We can call it 'regression' because it regressed on the previous values and 'auto' because it regressed on the previous values of itself. This process describe the dependence out of independent values, and plus a describing an error.

To make it compact there is defined an autoregression operator:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p \tag{2.6}$$

or

$$\phi(B)\check{y}_t = a_t \tag{2.7}$$

The model has $p + 2$ parameters: $\mu, \phi_1, \ldots, \phi_p, \sigma_a^2$, where σ_a^2 is the variance of white noise.

2.6.2 Moving Average Model

Moving average (MA) model consider the current value $\check{y}_t$ is dependent linearly on a finite number of q previous a's:

$$\check{y}_t = \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q} \tag{2.8}$$

Such process is called a moving average process of order p. Just as in case with autoregressive model to make it shorter there's defined a moving average operator

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q \tag{2.9}$$

or

$$\check{y}_t = \theta(B) a_t \tag{2.10}$$

The model has $q + 2$ parameters: $\mu, \theta_1, \ldots, \theta_q, \sigma_a^2$. The name of the model is kind of misleading, but the term is in a common use.

2.6.3 Autoregressive Moving Average model

Autoregressive Moving Average model (ARMA) includes both AR and MA models and supposed to have better fitting to actual time series than separate models have by themselves. The model looks like:

$$\check{y}_t = \phi_1 \check{y}_{t-1} + \phi_2 \check{y}_{t-2} + \ldots + \phi_p \check{y}_{t-p} + a_t + \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ldots - \theta_q a_{t-q} \tag{2.11}$$

That is

$$\phi(B)\check{y}_t = \theta(B) a_t \tag{2.12}$$

The model has $p+q+2$ unknown parameters $\mu, \phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q, \sigma_a^2$. Usually stationary time series can be described by a model with p and q less than 2 [17].

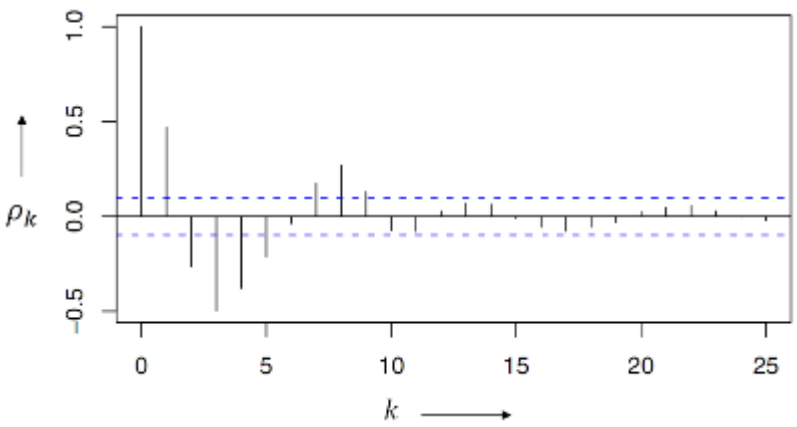


Figure 2.2: Theoretical autocorrelation function of a stationary process

2.6.4 Autoregressive Integrated Moving Average model

As many time series occurred in a real life are nonstationary and do not vary about a fixed mean, we need to find models to describe them. If a process has homogeneous nonstationary behaviour, the d'th difference of the process is stationary. Then we can describe such process with autoregressive integrated moving average model (ARIMA).

To describe this, let $\phi(B)$ be a generalized autoregression operator such as $\phi(B) = \varphi(B)(1-B)^d$. Here $\varphi(B)$ refers to a stationary operator and d shows the difference which should be stationary. One of the conditions is that $\phi(B) = 0$ should have d unit roots [17].

Thus we have

$$\phi(B)z_t = \varphi(B)(1-B)^d z_t = \theta(B)a_t \tag{2.13}$$

We can rewrite this in a following way:

$$\varphi(B)\omega_t = \theta(B)a_t \tag{2.14}$$

where $\omega_t = \bigtriangledown^d z_t$.

In practice d usually equals 0, 1, or 2. ARIMA works for seasonal models.

2.6.5 ARIMA model Identification and Estimation

The main goal of the model identification is to find proper parameters p, d, and q and the subset of all ARIMA models, which could be used to describe given time series. In order to do this, we need, first of all, this given time series to be stationary, or to take z_t differences to provide stationarity. We can use autocorrelation function for it.

Autocorrelation function of ARIMA process meet the following difference equation:

$$\phi(B)\rho_k = 0, k > q \tag{2.15}$$

where $\phi(B)$ – parameters of AR process, ρ_k – autocorrelations, q – the order of MA process, and k is lag.

For stationary process the roots of $\phi(B) = 0$ are out of the unite circle and autocorrelation function is damping for big values of k (Figure 2.2).

Usually, if partial autocorrelation function does not fall off rapidly then the process most likely is nonstationary.

The same holds for nonlinear models: for stationarity we choose such d for the model $\omega_t = \bigtriangledown^d z_t$ that its autocorrelation function is damping rapidly. According to Box and Jenkins [17], d usually equals 0, 1, or 2, and it's enough to observe first 20 autocorrelation values of the initial time series, it's 1st and 2nd differences [1].

After we have parameter d chosen, we can start thinking about p and q orders. For this we should remember that:

- autocorrelation function of AR(p) process falls off smoothly, while its partial autocorrelation function 'die out' after lag p;
- autocorrelation function of MA(q), vice versa, 'die out' after lag q and its partial autocorrelation function falls off smoothly;
- for ARIMA process, autocorrelation function after first $q - p$ can be presented as a sum of exponents and damping harmonic curves and, vice versa, for partial autocorrelation $(p - q)$.

First, we can try $p = 1$ and $q = 0$ and build a model. If forecasting results are not satisfactory, we can adjust $p(p = 2)$ and $q(q = 1)$, then estimate the new model, and so on. There are diagrams provided by Box and Jenkins which allows us to get AR(2), MA(2) and ARMA(1,1) parameters. In general case parameters of ARMA(p, q) can be found using the first $p + q + 1$ autocovariances.

For estimation of a built model Maximum Likelihood estimation [56] and Least Squares method [44] are used.

2.7 Machine Learning Models

There is a vast range of machine learning tools available for forecasting starting from traditional regression analysis to latest types of neural networks and different hybrids. Some of the representative methods for our project are described in this section[2].

2.7.1 Regression Analysis

Linear Regression

Linear regression models are the simplest form of regression models. This model is not to be confused with the regression models used in statistics where regression means to return to a previous state. The output of a linear regression model is just the sum of the attribute values with weights applied to each attribute before adding them together

$$y(\mathbf{x}, \mathbf{a}) = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \tag{2.16}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of input variables – attributes, $\mathbf{a} = (a_0, \ldots, a_n)$ is a vector of weights and n is the number of attributes for the forecasted output value y. The goal is to come up with good values for the weights – ones that fit the model to the

[1] However at Math Bureau `mbureau.ru` for lots of real time series problems they sometimes use up to 10 order models

[2] Some parts of this section regarding regression models, neural networks and Support Vector Machines were written together with Sopiko Gvaladze as a part of Projects in Machine Learning course held by prof. Roland Olsson at Høskolen i Østfold in spring, 2014. The projects are documented in CD attached to this thesis

desired output. Once the model is trained and those weights are obtained, a prediction equation can then be used for the prediction of a target value.

However, linear regression models are good and working accurately only when data presents linear dependency between dependent (which is to be predicted) and independent (predictor variables which are attributes)variables. Thus the result of a linear regression modeling is a line which fits best for describing a relation between variables (Figure 2.3). 'Best' here is referred as the least mean-squared difference for the vertical distances between the points and the line.

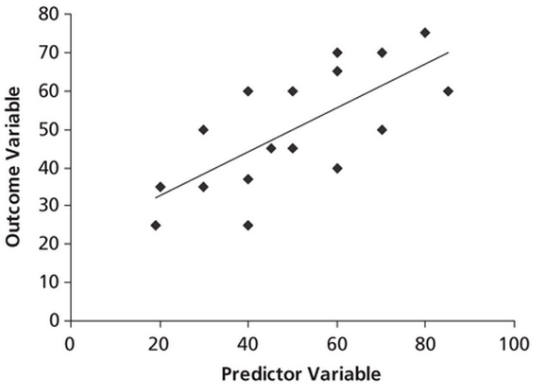


Figure 2.3: A scatter plot showing the linear relation between the input and predicting variables [61]

As this model is just a linear function of an input variables it place a limitation on the model. To extend the class of the linear regression models they consider linear combinations of nonlinear functions of the input variables which are called basis functions [15]:

$$y(\mathbf{x}, \mathbf{a}) = a_0 + \sum_{i=1}^{n} a_i \phi_i(\mathbf{x}) \tag{2.17}$$

where $\phi_i(\mathbf{x})$ is a basis function.

Since the obtained models are easy to interpret it makes this method a widely preferred tool. However the main limitation of linear regression is an assumption that the data are independent, what is not often sensible, and the sensitivity to outliers – data that are surprising and lies out of expected range – which can have a significant effect on the regression.

Regression Trees

Regression Trees is a subclass of Decision Trees – model which obtain a target value by going down the tree from the root to some leaf node. In each node a test of some attribute is represented and each branch outgoing from that node represents a possible value (or a range of values) of the tested attribute. For real-valued attributes the test corresponds to a split by three intervals: below, within and above. A numeric attribute is often tested several times on an any single path from the root to the leaf. Each leaf contains a value which is an average of all values of instances which reach the leaf.

The building of a regression tree can be represented through three steps: tree growing, tree pruning and selection of the final tree. Tree growing is a recursive process of partitioning the initial sample space into two subsets based on the least residuals for the partition

estimation. The process continues until the number of attribute vectors in each partition or the sum of residuals is small. The process of pruning is carried out in order to avoid overfitting. Thus, starting with the largest tree, all splits that do not improve accuracy of the tree on the training data are being removed. Final regression tree is chosen being guided by two principles: tree which has best accuracy on training data and which is the shortest one is chosen [3].

Regression trees is much larger and more complex than Linear regression equations and is more accurate because a simple linear model represents numeric data problems more poorly. However it is difficult to interpret regression trees due to it's big size.

Model trees is a combination of regression equations and regression trees but instead of single predicted value, leaves of Model trees contain linear expressions. Even though model trees are smaller than regression trees (and hence are more easily interpreted) the average error values are lower [60].

Rules based on regression and model trees

Learned rules can be re-represented as sets of if-then rules to improve human readability.

Rules are easy to interpret for humans, that's why it is a popular alternative to trees. Preconditions of the rules can be formulated not only with simple conjunctions, but also with general logical expressions. The accuracy of rules from regression and models trees is almost similar to the regression and model trees accuracy respectively.

First a model tree is built based on all data, then each leaf is converted into a rule. After, the data covered by that leaf is removed, and then the process is repeated with the remaining data.

One of the advantages of rules from trees is that a new rue can be added to the initial set of rules without making any changes while to add some leaf value into a tree may cause reshaping of the whole tree. However sometimes one rule taken from a list of other rules may lead to incorrect result as it is being considered individually or out of context.

2.7.2 Artificial Neural Networks

Artificial Neural Networks is a machine learning technique taking its origin from the biological neural networks – a network of a nerve cells of a human brain. ANN is a system of interconnected and interacting neurons. There is an input, activation and an output function for each neuron however the process of training of a neural network is a task of finding the weights for links between neurons.

Neural networks is a state-of-the-art technique and one of the most effective machine learning methods. An ongoing research shows that it is a powerful tool for pattern recognition and classification and proposed to be an universal approximator which can fit any function however they works well not only as a fitting tool but also has a good ability to generalize.

NNs can be grouped into two major categories:

- feedforward networks,
- feedback (recurrent) networks.

Feedforward nets (multilayer perceptron and radial basis funstion) are mostly being used for classification and function approximation problems. In the feedforward nets there are

[3] The favoring of shorter trees is base on Occam's razor: 'Prefer the simplest hypothesis that fits the data'

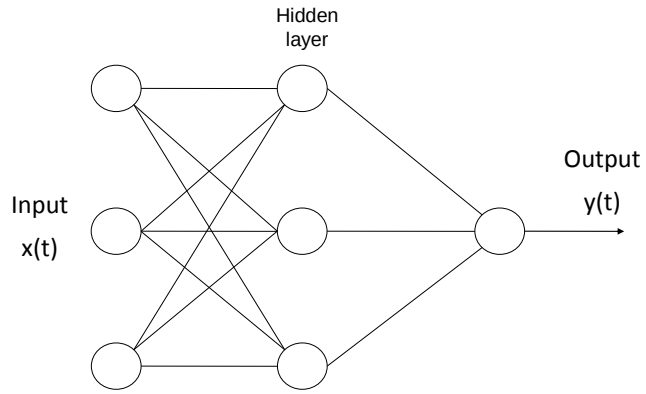


Figure 2.4: Architecture of a feedforward neural network

no loops in the network connections – they go only in one direction – and neurons are organized into layers. In feedback nets one or more loops may exist.

There are also fuzzy neural networks which use fuzzy logic [45], neural networks with memory which consider essential feature of brain – memory [54], dynamic neural networks [63] and so on.

Feedforward Neural Networks

According to MathWork Documentation [6], feedforward neural networks can be used for any kind of input-output mapping and a feedforward network with one hidden layer and enough neurons in the hidden layers can fit any finite input-output mapping problem.

A general architecture of feedforward neural networks is presented on Figure 2.4. Hidden layer(s) of a neural network consists of primitive units which can be perceptrons, linear units and sigmoid units. Perceptron units takes a vector of input variables x_t and calculates it's linear combination. If the result is greater that some threshold than the output $y(x_1, \ldots, x_t)$ of a perceptron is 1, if less than -1:

$$y(x_1, x_2, \ldots, x_t) = \begin{cases} 1 & \text{if } \omega_0 + \omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_t x_t > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.18}$$

It also can be representing as a decision making in the multidimensional space of instances: the output for instances lying by one side of a hyperplane is 1 and for instances lying by the other side -1.

Linear units and sigmoid units do not contain any threshold. For linear unit the output y is given by $y(\vec{x}) = \vec{\omega} \cdot \vec{x}$.

Learning a perceptron is a process of finding value of the weights $\omega_0, \omega_1, \ldots, \omega_t$. One of the possible ways is starting with random weights and then each time when perceptron misclassifies the instance update the weights according to some update rule. This process is repeated iteratively until the model classify all the instances correctly.

For function approximation problems the Levenberg-Marquardt training algorithm has the fastest training time and convergence for networks with about a few hundred weights. This is a big advantage when training is needed to be very accurate. LM (Levenberg-Marquardt) algorithm is usually able to get lower mean squared errors than other algorithms [14]. But when the number of weights in net increases the performance of this algorithm decreases.

LM algorithm is a blend of vanilla gradient descent and Gauss-Newton iteration and provides a solution for Nonlinear Least Squares Minimization problem [46, 50]. The function which is going to be minimized has the following form:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{2.19}$$

where x is an vector of input variables and r_j are residuals. The derivatives of $f(x)$ can be written using Jacobian matrix J of r defined as $J(x) = \frac{\partial r_j}{\partial x_i}, 1 \leq j \leq m, 1 \leq i \leq n$. To update vanilla gradient descent parameter the negative of the gradient at each step i is being added, which is scaled by λ:

$$x_{i+1} = x_i - \lambda \bigtriangledown f \tag{2.20}$$

Of course we would like to have large steps when the value of the gradient is small and have small steps when the value of the gradient is large while vanilla gradient descent does it vice versa.

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, second derivatives [46]. To solve the equation of second derivatives $\bigtriangledown f(x) = 0$ Levenberg used Newton's method. Expanding the gradient of f using a Taylor series around the current state x_0 we get

$$\bigtriangledown f(x) = \bigtriangledown f(x_0) + (x - x_0)^T \bigtriangledown^2 f(x_0) + \text{higher order terms of } (x - x_0) \tag{2.21}$$

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

$$x_{i+1} = x_i - (\bigtriangledown^2 f(x_i))^{-1} \bigtriangledown f(x_i) \tag{2.22}$$

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

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \bigtriangledown f(x_i) \tag{2.23}$$

where H is a 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. Formally we can write the Levenberg algorithm in the following way:

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 step 1.
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 the Levenberg-Marquardt rule looks like:

$$x_{i+1} = x_i - (H + \lambda diag[H])^{-1} \bigtriangledown f(x_i) \tag{2.24}$$

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.

As ongoing research shows different optimisation techniques are applied to training algorithms however different architectures of neural networks gives more space to move on.

Recurrent Neural Networks

Most recurrent neural networks (RNN) have scaling issues: it cannot be trained for large number of neurons or input units. As there are few inputs in time series problems it has been mostly used for time series. This type of NNs is a directed cyclic graph (while feedforward nets are acyclic graphs). Recurrent NNs use an output of the system at time t as an input for the system at time $t + 1$ (Figure 2.5).

The main point in recurrent neural networks is adding a new input unit $c(t)$ and a new hidden layer b (Figure 2.5). The value of $c(t)$ is the value of $b(t - 1)$. As we can see, b depends not only on $c(t)$, but on the earlier values of x, thus it summarizes information from earlier inputs [36].

There are several architectures of recurrent neural networks:

- Fully recurrent network
- Hopfield network
- Elman networks and Jordan networks
- Echo state network
- Long short term memory network
- Bi-directional RNN
- Continuous-time RNN
- Hierarchical RNN
- Recurrent multilayer perceptron
- Second Order Recurrent Neural Network
- Pollack's sequential cascaded networks

Fully recurrent networks were developed in 1980s. Every neuron in such networks has a weighted connection to every other unit. All other networks in the list are special cases.

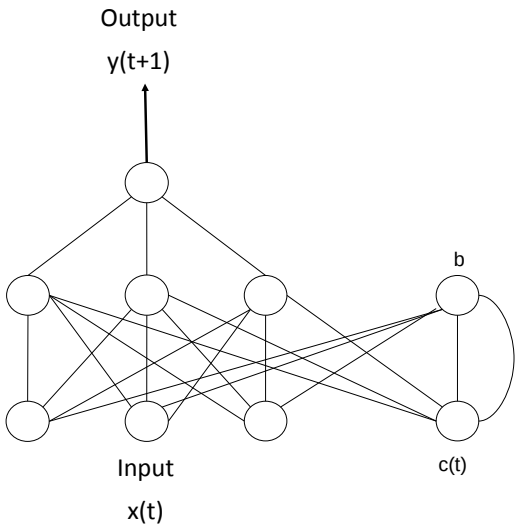


Figure 2.5: Architecture of a recurrent neural network

Hopfield networks requires stationary inputs. All connections in Hopfield net are symmetric. It is more of historic interest than practical.

Elman networks and **Jordan networks** are three layer networks with the addition of 'context units' u. For Elman network context units are fed from the hidden layer (Figure 2.6).

Connections from the middle hidden layer to these context units are fixed and equal one.

Jordan networks are similar to Elman networks, but the context units are fed from the output units instead of the hidden and have a recurrent connections with themselves. Elman and Jordan networks are also known as 'simple recurrent networks' or SRN.

In **Echo state network** the connectivity and weights of hidden neurons are randomly assigned and are fixed. The only parameters that can be learnt here are weights of output connections. ESN are good at reproducing chaotic time series [42].

Long short-term memory network was introduces in 1997 by Hochreiter and Schmidhuber [38]. It is well-suited to learn to classify, process and predict time series when there are very long time lags of unknown size between events. It can handle signals with a mix of low and high frequency components and doesn't have the vanishing gradient problem.

Recurrent multilayer perceptron generally consists of a series of cascaded subnetworks, each of which consists of multiple layers of nodes. Each of the subnetworks are feedforward except the last layer which can have feedback connections.

The state-of-the-art application of recurrent neural networks is natural language processing for such problems as language modeling, machine translation, language understanding and so on.

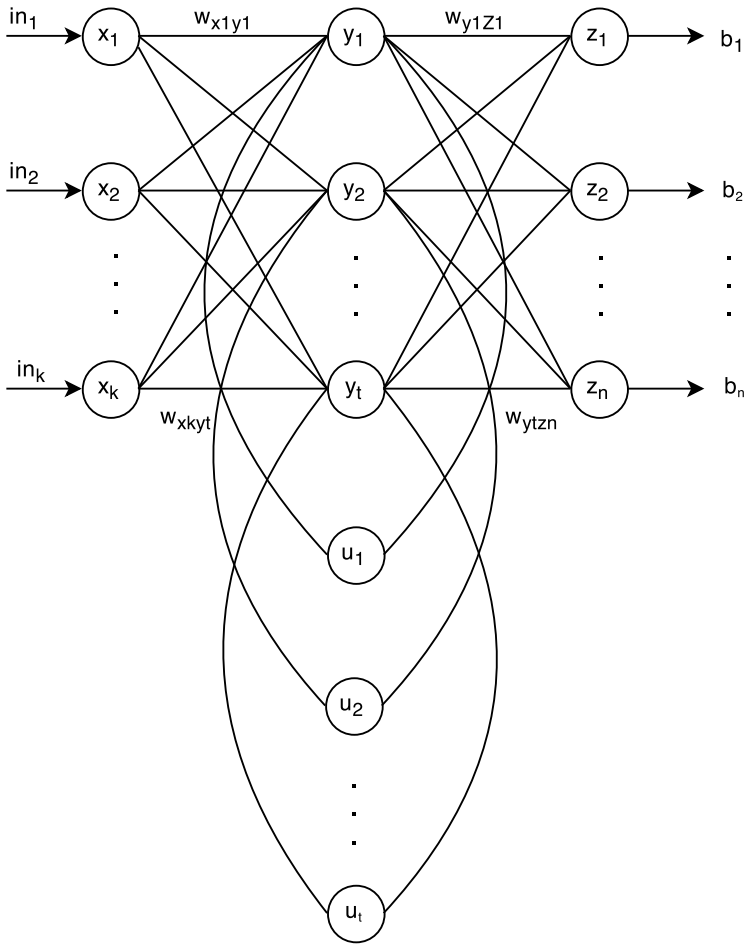


Figure 2.6: Architecture of Elman recurrent neural network

2.7.3 Support Vector Machines

Support vector machines (SVM) is a blend of a linear modeling and instance based learning in a high-dimensional space. SVM can be applied for those problems when data cannot be separated by line. Support vector machines use nonlinear mapping – it transforms the instance space into another space which has higher dimension than the original one. In this case line in the new space can be represented as a linear boundary in the instance space. Support vector machines were originally developed for classification problems however it is also has application for numeric prediction.

Kernel concept gave rise to support vector machines. Kernel is a function which fulfill mapping of a nonlinear data to a new space.

Kernel function K is an inner product $\Phi(x) \bullet \Phi(y)$ between the images of two data points x and y:

$$K(x, y) = \Phi(x) \bullet \Phi(y) \tag{2.25}$$

where $\Phi(x)$ and $\Phi(y)$ are mapping operators.

The feature, that kernel function is formulated as an inner product, gives an opportunity to replace scalar product with some choice of kernel [19].

The problem of finding parameters of SVM corresponds to a convex optimization problem, which means that local solution is global optimum as well.

SVM for regression is used to find a linear model of the following form:

$$y(x) = \omega^T \varphi(x) + b \tag{2.26}$$

where $\varphi(x)$ is a space transformation function, which is also a kernel function; ω and b are parameters which can be adjusted for a certain model and estimated in an empirical way. In simple linear regression the task is to minimize a regularized error function given by Equation 2.27.

$$\frac{1}{2}\sum_{n=1}^{N}\{y_n - t_n\}^2 + \frac{\lambda}{2}\|\omega\|^2 \tag{2.27}$$

where t_n is a target value, y_n is a predicted value and $\frac{\lambda}{2}\|\omega\|^2$ is an regularisation parameter [15].

The goal of SVM is to obtain sparse solution using ϵ-insensitive error function, where error is zero if absolute distance between predicted $y(x)$ value and target t is less than ϵ (Figure 2.7). 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.

Now regularized error function can be rewritten.

$$C\sum_{n=1}^{N}\{\xi_n - \hat{\xi}_n\}^2 + \frac{1}{2}\|\omega\|^2 \tag{2.28}$$

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

$$t_n \leq y(x_n) + \epsilon + \xi_n, \tag{2.29}$$

$$t_n \leq y(x_n) - \epsilon - \hat{\xi}_n \tag{2.30}$$

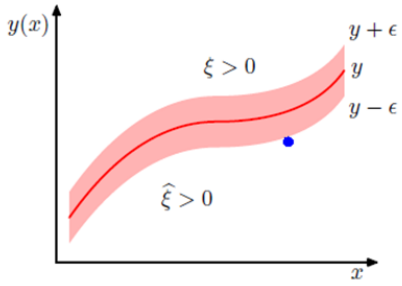


Figure 2.7: SVM illustration with slack variables and ϵ-tube curve [15]

This problem can be solved using Lagrange multipliers [15]. After solving this problem the prediction function for new inputs is obtained (Equation 2.31).

$$y(x) = \sum_{n=1}^{N} \{a_n - \hat{a}_n\} K(x, x_n) + b \tag{2.31}$$

where K is a kernel function and $a_n \geq 0$, $\hat{a}_n \geq 0$ are Lagrange multipliers. Here $x_1, \ldots, x_n$ are support vectors and they lie outside of ϵ-tube.

According to this SVM does not use all the input instances, but only few support vectors. Tube's flatness ensures that algorithm will not 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 we will try to find optimal ϵ, so that to balance error minimization and tube's flatness – overfitting.

Kernel Functions

There are many forms of kernel functions. In this project we will use

- polynomial kernel,
- radial basis kernel,
- Pearson VII function based kernel.

Polynomial kernel is represented as

$$K(x, y) = (x^T y + c)^d \tag{2.32}$$

where x and y are vectors in input space, d is a dimension of a new space and c is a free parameter [15].

Radial basis kernel is given by

$$K(x, y) = e^{(\frac{-\|x,y\|}{2\sigma})^2} \tag{2.33}$$

where σ is a free parameter [15].

Pearson VII function based kernel is represented as

$$K(x_i, x_j) = \frac{1}{\left[1 + \left(\frac{2\sqrt{\|x_i - x_j\|^2 \sqrt{2^{1/\omega} - 1}}}{\sigma}\right)^2\right]^{\omega}} \tag{2.34}$$

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 and can be used as a universal kernel [58].

2.8 Summary

Time series prediction is a vast domain requiring some necessary background which we have tried to cover in this chapter. We have started with the introduction into time series and types of processes. Also we have described seasonality and trend components and their role in the additive model of a time series. In the second part of this chapter statistical and several machine learning approaches are described. Highlighting statistical modeling we start from simple models as Autoregressive and Moving Average processes and move to Autoregressive Integrated Moving Average processes. As for machine learning methods from all the variety of them only those methods are described which were used in the experimentation session.

Chapter 3

Related Work and Methodology

In this chapter we introduce our findings over research and related work regarding time series analysis as forecasting of a hydropower plant's energy production is a time series problem. The study of related works basically presents how we prepared ourselves for carrying out experiments, explains why we have chosen particular methods in this work and gives some remarks regarding methods in order to better understand their work and results. In the last part of this chapter our methodology in building experiments and estimation techniques are described.

3.1 Related Work

Time series forecasting is a vast domain which requires the study of related work done as well as a good knowledge of theoretical background (Chapter 2). In this section we describe relevant works in time series analysis as a problem domain and some works regarding energy production forecasting as the main scope of this project. We start with traditional time series modeling – Autoregressive Integrated Moving Average and move to regression models, then to neural networks and finally to support vector machines.

3.1.1 ARIMA Models in Time Series Analysis

Autoregressive Integrated Moving Average Model (ARIMA) have been being used in time series analysis since 1970s when it was introduced by Box and Jenkins [17] and before that time AR and MA models had been used. ARIMA is used for prediction of stationary, or weak stationary (when dth differences of the process are stationary). Even though there are different approaches to estimate trend and seasonal components using additive model of a time series still it can be considered as a main disadvantage of ARIMA approach: it necessarily requires involvement of a human being in the process of forecasting (e.g. estimation of trend, seasonality, correlation and autocorrelation functions, etc.) and requires at least weak stationarity of the process. Also one of the limitations of the model is its linear form.

Some researchers have proven that the forecasting accuracy can be improved applying different preprocessing techniques to time series, for example principle component analysis [39], wavelet analysis [55] and so on. Different techniques on the modeling step are also used to improve statistical models. For example, it can be forecasting using AR models

with time-varying structure (they describe a nonstationary model via finite intervals which represents a stationary model) [49].

There are a lot of competition for time series analysis, and some of them show that simple methods developed by practicing forecasts do in many cases better than statistically sophisticated methods like ARIMA or its modifications (e.g. ARARMA) [48]. At the same time, the results of the 1st international competition of time series forecasting (ICTSF) [1] held in 2012 shows that statistical methods, in general, perform better. However, when comparing different methods we should consider different estimation measures, as they can give different result accuracy. For example, the Kaggle competition for tourism forecasting, part one [4], shows that ARIMA methodology is more accurate when considering its accuracy in MASE (mean absolute square error) terms, but when considering the MAPE (mean absolute percentage error), other models gives more accurate results [11].

In state-of-the-art projects they mostly use ARIMA models in hybrids with other ones, usually neural networks. A hybrid ARIMA-NN was proposed by Zhang [65] in 2003 which gave more accurate prediction than each of the model individually. In this model data is assumed to be a sum of linear and non-linear components. Thus, Babbu and Reddy in their ARIMA-NN hybrid [13] is using a moving average filter do decompose the given time series into two datasets. Then ARIMA and NN models are applied to the corresponding subset separately and the results are summed to be a result of a forecast.

3.1.2 Feedforward Neural Networks in Time Series Analysis

The main advantage of neural networks (NN) is that it can forecast completely automatically, without involving a human being into the process. The model has satisfactory accuracy and often is not computationally consuming (if the number of layers and time lag are not too big).

Neural Networks have been being used for time series processing gradually from 1990s. The general overview and applications on NNs to the problem of time series analysis and forecasting are discussed in [25].

The most wide-spread NNs are feedforward nets for classification and function approximation (multilayer perceptrons and radial basis functions networks). An example of feedforward NNs in forecasting for multivariate time series described in [12] for modeling flour prices over the period of eight years. The results stated a remarkable success in training a network to learn the price curve and make accurate predictions comparing to ARIMA methods.

However the empirical evidence of NN forecasts indicates varying degrees of success. Since NNs are usually considered to be a universal approximator, scientists (and mostly statisticians) started to investigate whether NNs are useful in handling statistical problems [21].

Processing time series with NNs arises some questions:

- Does a success of NN depends on a type of data
- or on the selected model and numerical methods.

Making a proper structure may affect the resulting performance of NN models, but how we choose an input variables for NN: lag, number of hidden layers – initial conditions. Unfortunately, there's no general way to select input variables except to use the context and knowledge about other models fitted to similar data [28]. A black box model of neural networks may produce satisfactory forecasts but give little insight into the structure of the data.

From the comparative study held by Faraway et al. [28] we can see that NNs with multilayer structure and big number of nodes gives better fit but not a good forecast; the same situation was observed in the experiment with the size of the lag. It is kind of obvious to get such results using neural networks as deep learning NNs (not 'deep' for this experiment though) usually gives better results, but still the airline data is small to carry out a representative experiment with deep learning.

To answer the second question Faraway describes such a case study using an airline dataset used in time series book by Box and Jenkins. This data can be considered as classical as it was used by Box and Jenkins, Harvey [35], and other comparative studies (e. g. [28]).

The airline data has an upward trend and multiplicative seasonality. The latter was the main criterion why Faraway have chosen this data, as usually such kind of a data requires some transformation (e.g. logarithmic) to make the seasonality additive. So they wanted to investigate if NNs can deal with non-linearity in seasonality and though exclude one step in processing data. To see the difference they also carried out an experiment handling both preprocessed and not preprocessed data with neural networks.

Peter Zhang and Min Qi [65] found that neural networks are not able to handle seasonality or trends effectively with the unpreprocessed raw data. They also state that detrending or deseasonalization can dramatically reduce forecasting errors. However, neural networks give more robust forecast performances comparing to ARIMA, for example [62].

To improve the performance of NNs in a forecasting problem different approaches are used: from different architectures of neural networks to hybrids of NNs with other models. For example, fuzzy neural nets gives better results for some case studies and considerably reduce the dimensions of NNs comparing to similar approaches [47]. Usually ARIMA and NNs approaches are compared, Peter Zhang proposed a hybrid methodology that combines both ARIMA and NNs [64]. His experimental results show that the combined model improves the forecasting accuracy achieved by either of the models used separately.

As one of the modified approaches dynamic neural networks can also improve accuracy. A comparative study has been carried out using standard benchmarks from different forecasting literature [31]. The results show that dynamic NN approach is more accurate and perform better than the traditional neural network and ARIMA models.

3.1.3 Recurrent Neural Networks in Time Series Analysis

As we have mentioned in previous chapter, the state-of-the-art application of recurrent neural networks (RNN) is natural language processing for such problems as language modeling, machine translation, language understanding and so on. However there are a lot research in application of RNN for time series analysis, for example for failure prediction [52], determination of the representative time horizon for short-term renewable energy prediction [41], and so on.

It seems that recurrent NNs shows good performance in time series forecasting. Students from Stanford carried out research using energy load forecasting [18]. They used several methods, including Kernelized Regression, Frequency NN, Deep Feedforward NN and Deep Recurrent NN. The last one showed the best result: almost twice better than deep feedforward NN (Table 3.1).

From their report we can see that the circular seasons and hours and day of the week features decreased RMS error 3.5% and 8.5% respectively, so it is expected that for demand forecasting the day of the week is one of the most important features. For power

Learnin Method	RMSE	%RMSE
Kernelized Regression	1.540	8.3%
Frequency NN	1.251	6.7%
Deep Feedforward NN	1.103	5.9%
Deep Recurrent NN	530	2.8%

Table 3.1: Results of different learning models for load forecasting [18]

production, which depends not only on user's demand, but on weather conditions, those features can influence vice versa: the season period would influence more.

Preprocessing is a very important step for time series forecasting, especially when we work with ARIMA-based models. The conclusion was that except for the most basic transforms it is better to allow the neural network to find nonlinear relationships itself. However, Hatalis et al. [37] have applied nonlinear recurrent neural networks for the forecasting of wave power. They preprocessed data with exponentially smoothing method and prediction occurs better when data has been smoothed. Once more it proves that each problem should be considered individually, nevertheless, whatever the time series problem is, recurrent neural networks gives a promising results in time series analysis.

3.1.4 Support Vector Machines in Time Series Analysis

Support Vector Machines (SVM) is a promising alternative for the prediction of time series. Some case studies shows that SVM can outperform such methods as backpropagation (feedforward neural networks), however it is very sensitive to the proper choice of the parameters of a kernel function [43, 29]. Also, modified SVMs methods are applied, for example, using penalizing the error [30].

A comparative study was carried out on several datasets [57]. They used two simulated datasets: one of them was simulated according to ARIMA principles (stationary and following the normal distribution), and the second one is a more difficult nonlinear chaotic dataset (Mackey-Glass, which is often used for benchmarking). The third dataset they used was a real dataset, containing measures of the contamination levels of a filter. It is obvious, that ARIMA model performed best for so-called 'ARIMA' dataset, while SVM and Elman neural network (one of the architectures of recurrent neural networks – third method used in the study) performed similarly. For the more difficult benchmark dataset SVM outperformed ARIMA and best of Elman NN. The most interesting was the third experiment with real dataset. It turned out that Elman NN was not able to predict satisfactory on this data, while ARIMA performed better, but still it performed on the whole training set the same as SVM trained just on 10% of the training set. So we can say that for the dataset used in this case study SVM can built qualitatively good models even with much less training objects.

In recent study carried by Papadimitriou et al. [53] SVM model showed a good ability for short-term forecasting and predicting peaks of a time series process. Interesting that in some cases SVM perform much better than neural networks based models. For example, a comparison of hybrids of SVM and ANN with wavelet decomposition (WD) for wind power energy production prediction [23] shows that SVM-WD hybrid outperformed other methods in 24-hour ahead forecasting.

Through the study of relative work done in recent years in time series domain we can say that this is a vast domain where a range of models available for forecasting and still there are a lot of ongoing research. Furthermore, time series forecasting is more likely to be case specific rather than subject specific: one method proposed for one case and giving best results for it might be outperformed in other case by those models which was not good for the first case. The accuracy and complexity of models depends on the number of dataset features and time horizon of a prediction.

3.2 Methodology

In previous chapter we have described a necessary background regarding this work on energy production forecasting. In this section we describe our data, why we have chosen those methods and how we are going to apply them. After the estimation procedure is described.

3.2.1 Dataset

The data provided by eSmart is an hourly measured energy production from two different hydropower plants:

1. One of them is located in the mountains. Meaning that most of winter the water is in ice state. The ground is also mostly rock, which means that the effect of rain in the summer should be relatively immediate and short. We refer to this dataset as 'Dataset I'.
2. Another one is located further down in the valley, with a pond between the mountain and the powerplant. This means that the stream has water almost all year. The pond combined with dirt-ground makes effects of rain last longer. The grid-company says that the effect of rain is also very depends on what side of the valley gets the rain. So this is something to consider if to try to use percipitation as a feature of the model. We refer to this dataset as 'Dataset II'.

The value of the energy production is measured in MWh. There is almost two years (22 months) of data available, so that is about 15840 values for each power plant.

As the source of energy for the described power plants is water we should expect that there are two factors which could affect energy production of a hydropower plant: temperature and precipitation. If temperature is below zero it means that the water in the stream mostly is in the ice state and snow does not melt to provide the stream with water. Precipitation apparently effect the amount of water in the stream and pond. One of the chief problems is that we cannot get the weather data from the exact location of the power plants and the best we can do is to retrieve data gathered in a nearby town.

Another problem is a mismatch between the resolution of energy production and precipitation measurements. This is because energy production is measured hourly and precipitation once a day at 7a.m. So we just expanded the data to get hourly precipitation.

As the hydropower plants are small and have some power production limits, in both cases the processes have a threshold and their random fluctuations does not exceed particular values (except 1 and 2 values for 1 and 2 power plants respectively which are noise values): thus for the first power plant the threshold is about 2.4 MWh, and for the second one is about 4 MWh (examples of the small hydroplants production power in [9]).

3.2.2 Forecasting approaches

Forecasting of a hydropower plant energy production is a time series problem which is traditionally analyzed by a statistical time series modeling – Autoregressive Integrated Moving Average (ARIMA) – introduced by Box and Jenkins [17] in 1970. In recent years, Artificial Intelligence (AI) techniques, capable to analyze big data, have become increasingly popular, for time series analysis including. Therefore in our work we present both statistical and machine learning methods.

To find out which method is more accurate and universal, it is a good idea to carry out a comparative study. For this purpose kind of a benchmark dataset should be chosen, but it turns out to be a problem to find such dataset. For some studies they use an airline dataset (used by Box and Jenkins for ARIMA model), but this dataset is relatively small and cannot be used as a representative data for some methods. In other case studies they use different datasets, and it is difficult to find several works done on the same data.

In this project we decided to concentrate on real data so the findings of this research could have a real application with further work done. Thus we are working on two datasets from energy production domain (the description of the datasets is presented in previous section) and there is no work published on this data so far. This causes some difficulties with estimating obtained results as we have nothing to compare with and we cannot compare it with other results obtained on other datasets. However we can compare models' prediction with the prediction of the mean of the process or with the simple forecasting model when $y(t+1)$ is assumed to be equal to $y(t)$ (estimation criteria used in this work are described in next section).

One of the feature of analysing time series with machine learning approaches is that some number of previous values are fed as input for machine learning models. This value represents a so called window size, or lag value. In order to answer the first research question we not only use different datasets with preassumed difference in long-term dependencies but also need to play with the value of window size. As for the second research question we build different models which take different attributes as an input.

The next step is choosing proper methods which is reasonable to apply to the chosen data. Again, there is no general answer about which model is best to apply for time series forecasting. Mostly, neural networks (including recurrent NNs, fuzzy NNs, echo state NNs, etc.), support vector machines, genetic algorithms, different hybrids are applied.

A comparison of the performance of several AI methods for forecasting problems was carried out by Wang et al. [59] as a case study. They used long-term monthly flow data for one of the biggest rivers in Asia, as the problem of hydrological forecasting is very important. Autoregressive moving-average (ARMA) models, artificial neural networks (ANNs) approaches, adaptive neural-based fuzzy inference system (ANFIS) techniques, genetic programming (GP) models and support vector machine (SVM) methods were examined. The four quantitative standard statistical performance evaluation measures – the coefficient of correlation (R), Nash–Sutcliffe efficiency coefficient (E), root mean squared error (RMSE), mean absolute percentage error (MAPE) – were used to evaluate the performances of various models developed.

The results show that ANFIS, GP and SVM perform better in terms of different evaluation criteria. Thus, SVM model is able to obtain the better forecasting accuracy both for training and validation sets, while the forecasting results of ANFIS and GP model for the validation set is worse than for training set. Authors suggest that ANFIS, GP and SVM approaches are promising in modeling time series, and highly encourage studying

and applying those methods in time series analysis. However, more efficient approach for training multilayer perceptrons of ANN model, the increased learning ability of the ANFIS model, selecting more appropriate parameters of GP evolution, saving computing time or more efficient optimization algorithms in searching optimal parameters of SVM model, etc., are suggested to improve the accuracy of the forecast models in terms of different evaluation measures for better planning, design, operation, and management of various engineering systems. So we can see, that every problem should be considered with its individual approach.

To begin with the domain of time series analysis we have chosen the most popular methods: neural networks (feedforward and recurrent) and support vector machines which is becoming popular. Regression methods are also an interesting domain as it could be considered as kind of extended version of autoregressive models. We consider simple regression, linear regression, regression trees, model trees and rules based on model trees to regression methods.

Related work described in Section 3.1 shed some light on the usage of the chosen methods in time series analysis.

3.2.3 Models' Estimation

One of the problems which arise during forecasting is how to choose the optimal model. Since time series forecasting is a numeric problem there are some problems in estimating such models as it has a continuous scale. Many forecast measures have been proposed and recommendations made by different authors. Thus, lots of authors recommends the use of Mean Absolute Percentage Error (MAPE) [34, 16]:

$$MAPE = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{\check{y}_i - y_i}{y_i}\right| \tag{3.1}$$

However some authors warn about the usage of this error as it is not applicable when data involves small counts or zero values and the error value become infinite or undefined, so the usage of scaled errors are proposed [40].

As in our experimentation we deal with two datasets we can use scale-depended errors for each one. However in order to give a common conclusion about two dataset we need to use percentage errors as well. Below relevant to our experimentation error measurements are described.

Scale-depended errors include such commonly used errors as MAE, MSE, RMSE. These measurements are useful when comparing different methods on the same data as the scale of the error depends on the scale of data.

MAE – Mean Absolute Error – is the average of the absolute errors $e_i = |\check{y}_i - y_i|$, where $\check{y}_i$ is the predicted value and y_i is the observed value (Equation 3.2).

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|\check{y}_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{3.2}$$

According to [40] MAE is a common measure of forecast error in time series analysis.

MSE – Mean Squared Error – is the most common measure of numeric prediction. It is an average of squared differences between actual and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\check{y}_i - y_i)^2 \tag{3.3}$$

Many tools tend to minimize MSE when modelling however it could be argued that it overemphasizes the effect of large errors.

RMSE – Root Mean Square Error – is a measure of the differences between predicted value and the observed values (Equation 3.4).

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\check{y}_i - y_i)^2} \tag{3.4}$$

The individual differences here are called residuals when calculations are performed over the data sample used for training and are called prediction errors when computed out of sample. This measure tends to exaggerate large errors as well as MSE. One of the advantages of RMSE is that it gives the error value in the same dimension as actual and predicted values what makes it easy to view the error scale and what is a reason why RMSE is more preferred than MSE.

MSE and RMSE are popular due to their theoretical relevance in statistical modelling. However they are not robust to outliers comparing to MAE.

RAE – Relative Absolute Error – is a total absolute error normalized by dividing by the total absolute error (Equation 3.5).

$$RAE = \frac{\sum_{i=1}^{n}|\check{y}_i - y_i|}{\sum_{i=1}^{n}|y_i - \bar{y}|} \tag{3.5}$$

where $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ [60]. RAE shows what the error would have been if the prediction has been an average of the actual values so we can consider it as beating the mean prediction.

RRSE – Root Relative Squared Error – is relative to what it would have been if the prediction had been the average of the absolute value (Equation 3.6) [8].

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}(\check{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.6}$$

The square root here are taken in order to get the same dimensionality with the actual values as it was done in case of RMSE. Values of RRSE more than 100% indicate that scheme is doing worse than just predicting the mean.

For all described measurements smaller errors are better and means a more accurate model and a value of 0 indicates a perfect model.

In our experimentation we use MAE as a main criteria for comparison of the performance of all methods as this is one of the commonly used error measure and not that sensitive to outliers as RMSE. Other error measures are also take their place as it can say something about the data or models' peculiarities. In discussion chapter (Chapter 5) we present the calculation of a percentage error – MAPE – for the best method in order for reader to feel the scope of model's performance in some relative measure. To calculate MAPE we had to remove all zero values form test sets so this error was measured only on some subset of initial test set and cannot be reliable as other measurements.

Another part of model estimation is finding a proper partitioning of the initial data into two parts: training and test sets. There are a lot of arguing about the optimal splitting of datasets and usually it is taken by 80:20 percent values. However the splitting depends on several parameters such as full dataset size and accuracy of the applied method: thus the higher accuracy and smaller dataset size results in the partitioning with training set more than 80% [24].

As soon as our dataset is relatively not a big one (in terms of big data) we chose the smallest partition as 80% for training subset. As for neural networks experiments validation set is required besides the training one the remaining subset was split into 10% for validation and 10% for testing. In order to be able to compare performance of different models for other methods 90:10 percent splitting were chosen – to test built model on the same 10% partition. For these 10% the last parts for both datasets were taken so the testing of the models would be on the same data, not on random samples.

3.3 Summary

In order to answer research question we have taken two different datasets from hydropower energy production domain which have some peculiarities relative to our questions. The following methods are going to be applied:

- Autoregressive Integrated Moving Average (ARIMA)
- Simple Linear Regression
- Linear Regression
- Regression Trees
- Model Trees
- Model Rules
- Feedforward Neural Networks
- Recurrent Neural Networks
- Support Vector Machines

As an input for these methods different window sizes and attributes sets are going to be fed. As an estimation measures different errors are used however for comparing the performance of different models mean absolute error is chosen. Details about each experimentation session is described in Chapter 4.

Chapter 4

Experiment

This chapter describes several experimentation sessions with the datasets described in Section 3.2.1. We have used statistical ARIMA model and various machine learning techniques described in Chapter 2. Before each experiment we describe all preprocessing required for each method and then the obtained results and progress during each experiment session.

As soon as described ARIMA model cannot deal with exogenous input, for each experiment we have carried out experiments both without and with extra featuring in order to be able to compare results. For experiments without exogenous input we have chosen the best results out of two datasets and for the other one described both as it depends on features.

Except experiments which deals with neural networks, the datasets were divided into 90% for training and 10% for testing. For the experiment dealing with neural networks the split was as follows: 80% for training, 10% for validation and 10% for testing.

For error measurement for each experiment we have calculated different errors. MAE was chosen as the basis for comparing the performance of the models as this error is a common error measure for time series analysis (see Section 3.2.3).

4.1 Data Analysis

4.1.1 Preprocessing

The production measurements were contained in two `.xls` files – one file for each. Each file consisted of several sheets: each month on a separate spreadsheet represented in tables 31(30)days$_{\text{x}}$24hours. There is also some meta-information: average daily production, total daily and monthly production. Weather measurements were contained in other `.csv` files and had some missing values. So that was necessary to format the data in the proper way for using it in the experiments. For that purpose we created a project in C# and performed some transformations to the data.

- Breaking the day attribute into its components: the date attribute which has the 'dd/mm/yyyy:hh' format was divided into its components – day, month, year and hour in order to treat them separately during experiments. Also the 'year' attribute was removed as there are only two possible values for it – 2013 and 2014 – and assumed to have no explanatory value for the problem.
- Noise filtering: as the power plants have a production threshold any value which exceeds that threshold was treated as noise and was converted into the maximum

value for the given hydropower plant.

- Filling up missing values: in precipitation data there were lots of values indicating precipitation being < 0 which means there is something wrong with the sensor or there is no fall-out that day, so we treated that as no precipitations and converted those values into -1 in order some models could treat them.
- Adjusting different data resolutions in order to get all measurements being hourly: precipitation values were expanded form daily into hourly.

4.1.2 Data Analysis

There are some approaches in statistics to analyse data before applying different methods in order to summarize their main characteristics in a form easy for understanding. We based our analysis on determine qualitative measures like mean and correlation.

We had seven attributes in the datasets:

- day of month
- month
- year
- hour
- temperature
- precipitation
- production

The first four attributes are related to time and their statistical components are obvious and not a point of interest. But the three last attributes would be very interesting from a point of view of correlation between them (Table 4.1).

	Max	Min	Mean	Standard Deviation	Correlation with Temperature	Correlation with Precipitation
Dataset I	2.38	0	0.6804	0.8757	0.6110	0.0907
Dataset II	3.954	0	2.1904	1.5099	0.6758	0.1997

Table 4.1: General statistical measures for Production, Temperature and Precipitation attributes

As shown in Table 4.1, the minimum production is zero production, and maximum production for Dtaset I is 2.38 MWh and for Dataset II is 3.954 MWh which depends on the output capacity of each hydropower plant. The mean and standard deviation was found to be 0.6804 and 0.8757 for Dataset I and 2.1904 and 1.5099 for Dataset II. The value of standard deviation being higher than mean indicates scattered behaviour of the data so it means that the Dataset I much more scattered than Dataset II.

The correlation between production and temperature was found to be about 0.6 for each dataset which is lower than what we had expected. It can be caused by the fact that weather data was measured in nearby town not in the exact location of the power plants so there will always be some unreasonable correlations. Still temperature fluctuations is not that high for nearby locations. Probably we need only to know if the temperature is above or below zero and still negative temperature value does not guarantee that there will not be any power production as a stream in the river can still flow. Short but important temperature effect arise when there is a change from negative to positive (in the spring period) so we can consider it as a seasonal effect.

As we can see from Table 4.1 Dataset II has low correlation with the precipitation and for Dataset I correlation is extremely low and seems to be just a random effect. Having in mind that first power plant is located in mountains we should expect the effect of rain should be relatively immediate and short, therefore higher correlation than for the second power plant which is located further in the valley, with a pond between the mountain and the power plant. However correlation does not necessarily mean dependency.

As production depends on weather and rainfall level we should expect a seasonal component. Thus in winter months when most of the water resources are in ice state there is almost zero production for Dataset I and small amounts of the produced power for Dataset II (Figure 4.1).

4.1.3 Data Visualisation

One of the first things when analyzing time series is plotting it in different ways in order to retrieve some useful information which might guide us for and during experimentation. We performed specific time series visualisation, histogram and scatter plot visualisation.

Time series visualisation

First of all time series is usually visualised in time-space coordinates. Such plots gives a clear overview of a time series process, whether there are seasonal patterns, trend.

Production of the power plants is presented on Figure 4.1.

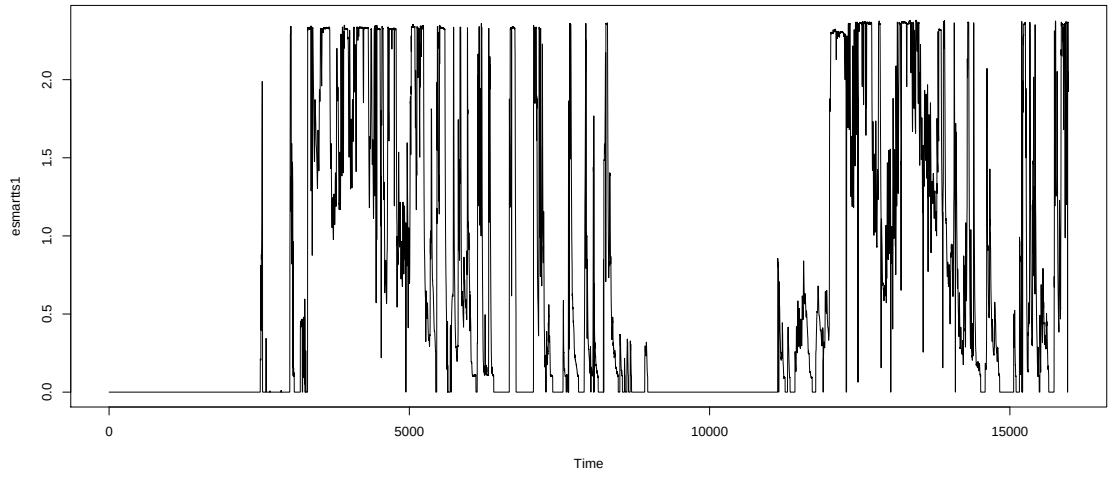
As we can see from Figure 4.1 the processes have thresholds over which there is no energy production due to the capabilities of the power plants so there is no trend observed. For Dataset I kind of a seasonal pattern is observed: for winter monthes there is no production at all and then in spring there are period s of quite stable production. For Dataset II we cannot see such obvious patterns but still there is a tendency for less energy production in winter months what is sensible. Seasonal patterns might be highlighted by ARIMA models however in order to retrieve a formal expression of seasonal fluctuations usually more than two seasonal periods are required.

Histogram visualization

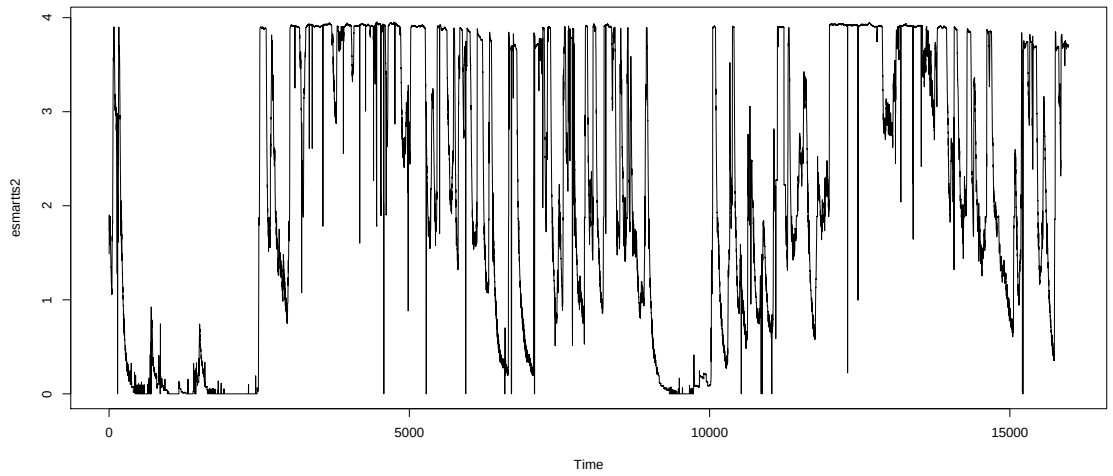
Histogram visualization is one of the most preferable one as it shows the distribution of the values in data and its skewness.

As seen in Figure 4.2(a), temperature distribution is close to normal one with more temperature values in range from 2°C to 10°C. The distribution of precipitation is right skewed, or non-symmetric (Fiure 4.2(b)). Also we can see that there are quit a lot missing values which needs to be preprocessed before feeding this attribute to some models (as it was described in Section 4.1.1).

The histogram plots of energy production for Dataset I (Figure 4.3(a)) and Dataset II (Figure 4.3(b)) shows that energy production is not normally distributed and have two peaks: in the very beginning of the value axe and at the very end. First it makes forecasting task more difficult and second there are some primary statistical preprocessing required for ARIMA models which are described in Section 4.2.

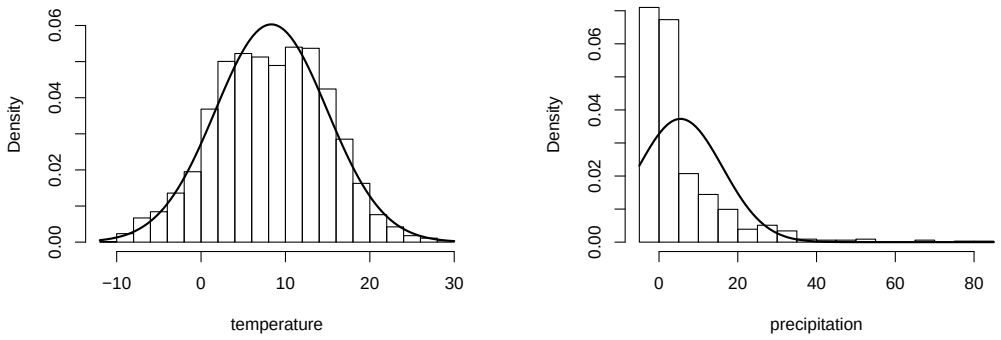


(a) Hydropower plant located in the mountains (Dataset I)



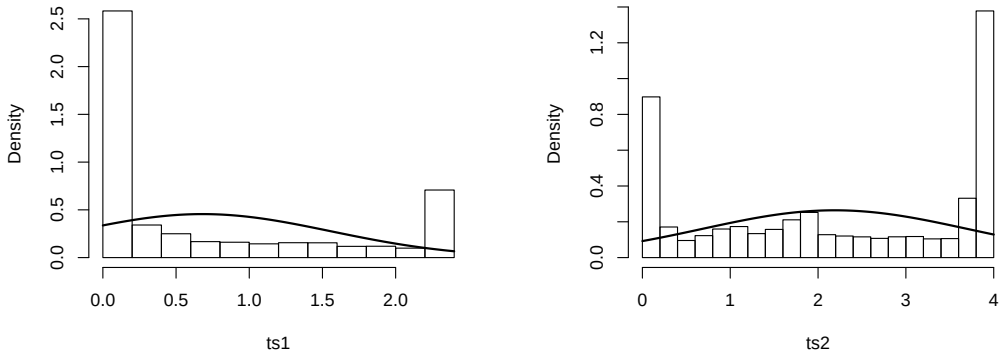
(b) Hydropower plant located in the valley (Dataset II)

Figure 4.1: Hydropower plants' energy production



(a) Temperature (b) Precipitation

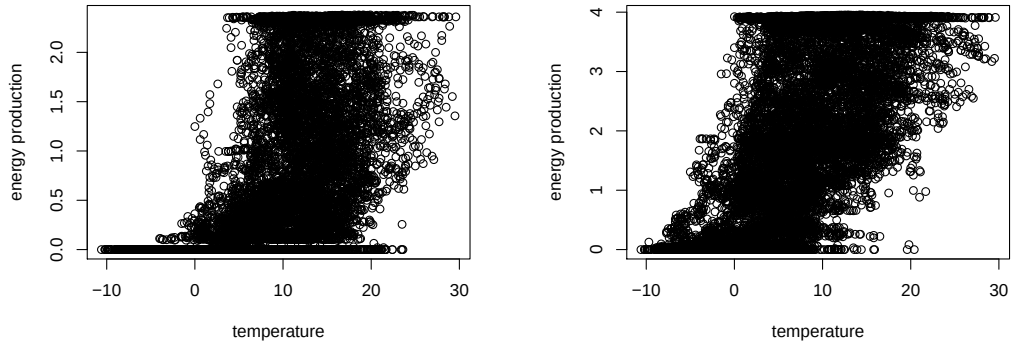
Figure 4.2: Density histogram of weather attributes



(a) Hydropower plant located in the mountains (Dataset I)

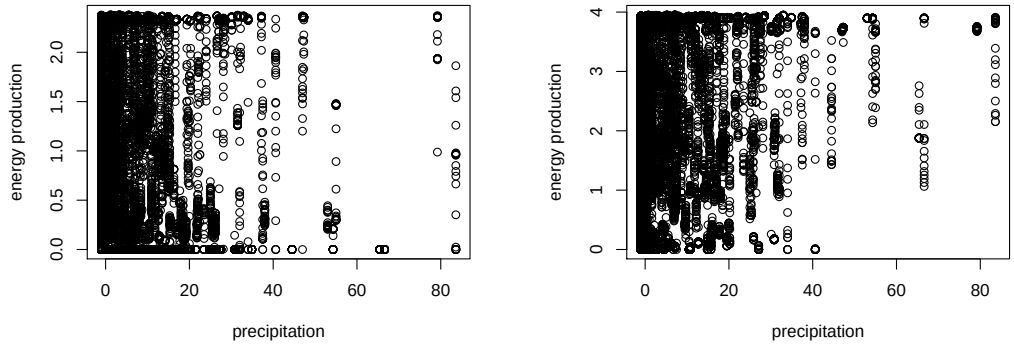
(b) Hydropower plant located in the valley (Dataset II)

Figure 4.3: Density histogram of energy production



(a) Dataset I (b) Dataset II

Figure 4.4: Scatter plots of Temperature vs. Energy production



(a) Dataset I (b) Dataset II

Figure 4.5: Scatter plots of Precipitation vs. Energy production

Scatter plot visualization

A scatter plot is an effective tool for visualizing the association between two variables – how change in one variable cause a change in another – and also an effective tool for determining outliers in data.

Scatter plots of temperature vs Energy production (Figure 4.4) shows two tendencies for both datasets: there are clusters on the border values of energy production (especially we can see it for Dataset I) and quite not much dispersed remaining values however there are some outliers. As for associations between precipitation and energy production, as we can see from Figure 4.5, distribution points have lots of outliers and much more dispersed comparing to temperature attribute what corresponds to the correlation analysis performed in the previous section. Presence of outliers for precipitation can be explained by the fact that some days there is no rain and other days it can be downpour, brief or steady.

In statistics outliers are usually discarded from data in order to have it pure. In machine learning one of the main objectives rather to find robust to noise and outliers models which are to find those dependencies than to simplify a problem, but nevertheless we should keep in mind that it would complicate the task and effect the performance.

4.2 ARIMA

The first experiment referred to statistical approach, namely ARIMA. We have used this approach for both datasets but we explain all those steps involved in this experiment referring to Dataset I.

ARIMA models implementation are available in many statistical tools and one of them – R implementation which is very popular for time series analysis. R is a free software environment for statistical computing and runs on UNIX platforms, Windows and MacOS [3]. During this project ARIMA models implemented in package 'forecasts' were used [22]. The R log is documented in CD attached with this report.

As ARIMA models cannot deal with attributes and consider only a target time series (in our case it is production) new `.csv` files were composed containing only hourly production time series.

The first step in statistical analysis using ARIMA is to decompose a time series into seasonal, trend and random components. R needs a time series having more than two periods to decompose it and as given data has less than two full years (we assume that processes have yearly seasonality) R could not decompose it indicating that there are 'no or less than two periods' even though we can see that there is no production in winter for Dataset I and this is due to seasonality (Figure 4.1).

A non-seasonal time series consists of trend and irregular components. For this purpose it is common to use a smoothing method such as simple moving average. However as we have seen (Figure 4.1) the processes have threshold and do not exceed particular values so we can assume having no trend components.

As specified by Box and Jenkins [17] in order to get a proper ARIMA model we need to perform the following steps: to select a model, estimate its parameters and estimate the result.

We began the model selection by differencing initial process as in order to apply ARIMA(p, d, q) the dth differences of the process should be stationary. We performed the 1st order differencing and observed Autocorrelation function (ACF) and Partial Autocorrelation function (PACF). As seen in Figure 4.6 the process of first differences appear to be stationary in mean and variance as the process fluctuates constantly about 0 level over time and autocorrelation function cuts off sharply to zero after the first lag while partial autocorrelation function tails off to zero very slowly and changes in magnitude Figure 4.7. Based on these observations we can conclude that the once differenced time series is stationary enough to proceed further. It means parameter $d = 1$ in ARIMA(p, d, q) model.

MA(q) (Moving Average) process parameters can be determined by examining autocorrelation function and AR(p) (Autoregression) process parameters – based on partial autocorrelation function (or we can choose ARMA(p, q)). We used the principle of parsimony to choose the model: we assume that the model with fewest parameters is the best. So we concluded that the given process can be modeled as $MA(1)$ (since autocorrelation functions cuts off to zero after lag 1). So we proceed further with ARIMA(0,1,1) model.

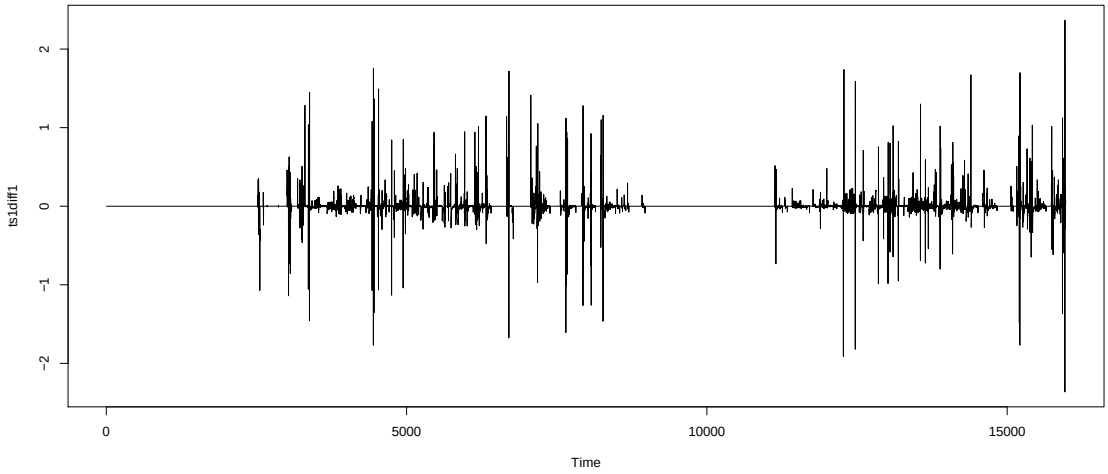


Figure 4.6: 1st differences of the initial process (Dataset I)

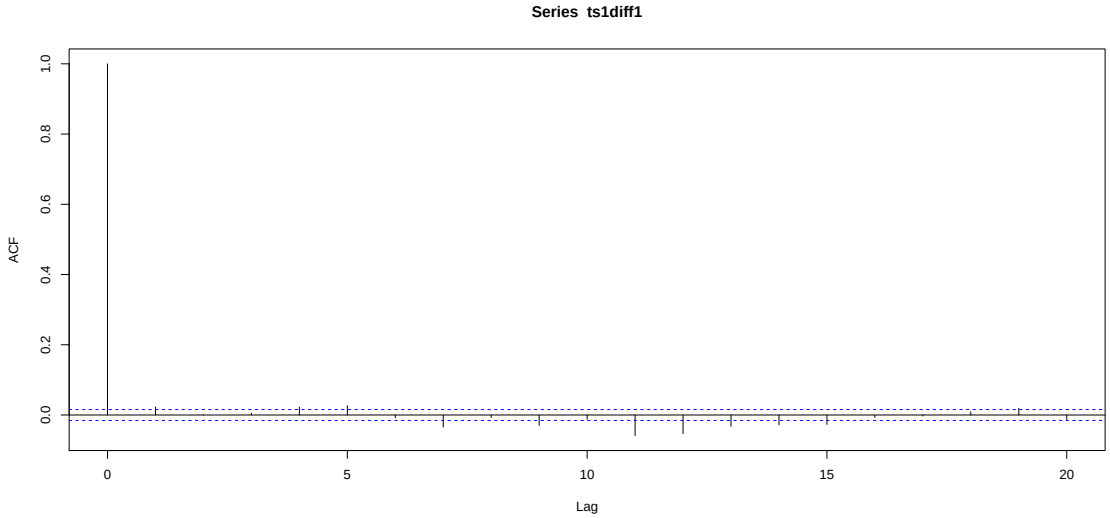
	ARIMA model	MAE	RMSE	AIC	σ^2
Dataset I	$(1-B)x_t = \omega_t + 0.023\omega_{t-1}$	0.0469	0.1627	-25913.53	0.0116
Dataset II	$(1-B)x_t = \omega_t - 0.1649\omega_{t-1}$	0.0583	0.2018	-10780.87	0.0298

Table 4.2: Results of ARIMA models

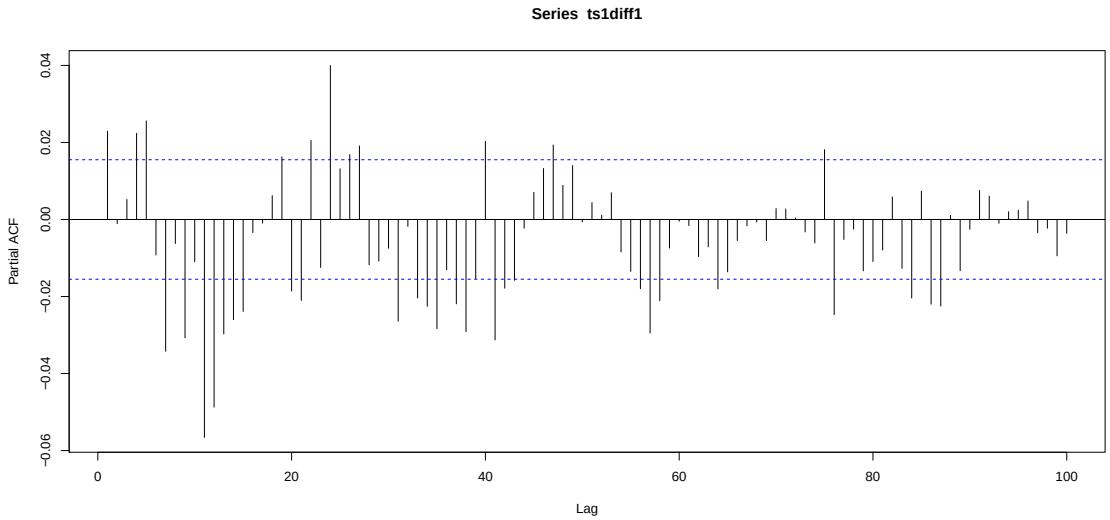
Fitting ARIMA(0,1,1) model to the time series we found coefficients of the model. The estimation of the model involved residual analysis. As seen in Figure 4.8 the autocorrelation function of the residuals shows that none of the sample autocorrelations for lags 1-20 exceed the significance bounds. It is also a common practice to perform a Ljung-Box test for examining the null hypothesis of independence in a given time series [26]. In our case the p-value for the test is near 0.8 which indicates a normality of estimation. The histogram of the residuals of a forecast errors shows that the forecast errors are roughly normally distributed and the mean seems to be close to zero (Figure 4.9). Based on the autocorrelation function, Ljung-Box test and residuals distribution we can conclude that there is little evidence for non-zero autocorrelations in the forecast errors and the chosen model gives an adequate forecast.

Following the steps described above ARIMA(0,1,1) model for Dataset I were chosen. During the model identification step for Dataset II the behaviuor of autocorrelation and partial autocorrelation functions were found to be almost the same for a single differenced time series, so ARIMA(0,1,1) was chosen for Dataset II as well. We built the models on 90% of the initial datasets and used remained 10% for out-of-sample testings (to build such model we need to update it by adding additional observations as ARIMA models in R cannot refit a model on a new data): we withhold 10% observations from the dataset when fitting, then compute the one-step forecast errors for the out-of-sample data. The summary of performance and findings about models are shown in Table 4.2.

AIC in Table 4.2 stands for Akaike Information Criterion [10]. Models having lower AIC are considered to be better than those having higher values. For both models the value of AIC indicates a good quality of a model. Similarly the value of a σ^2 is also an



(a) Autocorrelation function



(b) Partial Autocorrelation function

Figure 4.7: Autocorrelation and Partial Autocorrelation functions of the 1st differences (Dataset I)

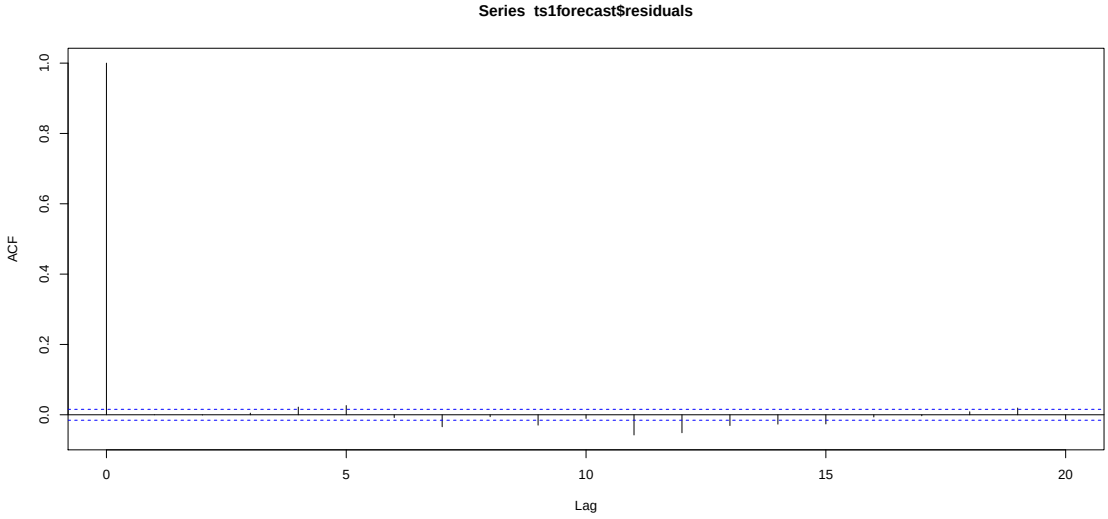


Figure 4.8: Autocorrelation function of residuals (Dataset I)

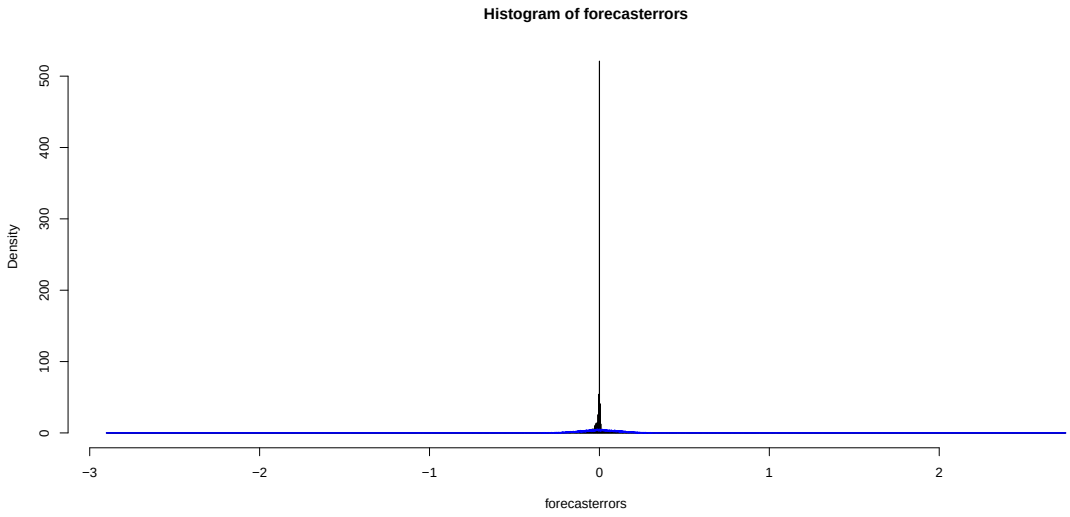


Figure 4.9: Distribution of forecast residuals (Dataset I)

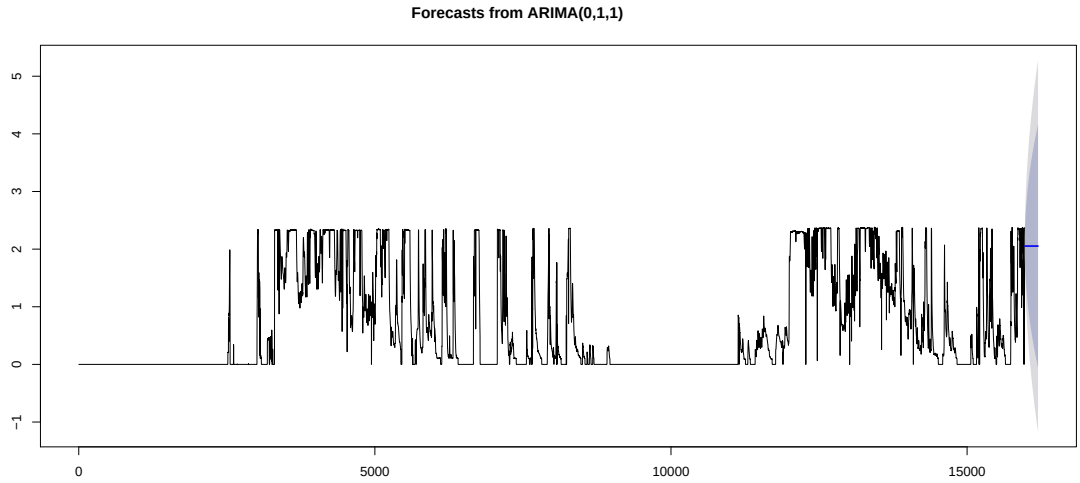


Figure 4.10: ARIMA forecast for Dataset I

important aspect of a model estimation: models with lower values of σ^2 are considered to be better than those having higher values.

ARIMA forecast can present a point forecast which means that this is the mean of the distribution of a future observation given the past observation of a time series. As we can see form Figure 4.10, the forecast function is flat: the future observations have the same mean. It once more tells us that there is no trend or seasonal components in the initial time series.

Forecasting with ARIMA involves estimating prediction intervals which is usually 80% and 95% (light blue and blue on Figure 4.10 respectively). One worrying thing about the models is that these intervals lie under zero level (and production cannot be negative) and above the threshold (indicating maximum possible production value of a power plant). The reason is that the ARIMA forecast functions in R do not know that the variable can only take values between some particular boundaries. Clearly this is not a very desirable feature of our current predictive model.

4.3 Regression Analysis

As we have described in Section 2.7.1 regression analysis is one of the widely used machine learning tools due to the fact that built models are easy to interpret. Hence there are a lot of tools having regression implementation. For this experiment Weka implementation was chosen since Weka is an open source software, has no limits for the number of instances and contains tools for data preprocessing, visualisation, implementation of regression, rules, Support Vector Machines and others [5].

There are several regression methods available in Weka:

- Simple Linear Regression
- Linear Regression
- REP.Tree
- M5P

Lag value	Error	Simple Linear Regression	Linear Regression	REP.Tree	M5P	M5Rules	M5P on training set
lag = 3	MAE	0.0587	0.0599	0.0743	0.0599	0.0606	0.0506
	RMSE	0.2085	0.2019	0.1979	0.1856	0.195	0.1565
	RAE	5.93%	5.96%	7.50%	6.05%	6.12%	3.59%
	RRSE	18.55%	17.97%	17.60%	16.51%	17.34%	10.11%
lag = 5	MAE	0.0587	0.06	0.0902	0.0586	0.0627	0.0476
	RMSE	0.2086	0.2029	0.2133	0.1936	0.2107	0.1506
	RAE	5.93%	6.06%	9.10%	5.92%	6.33%	3.38%
	RRSE	18.55%	18.05%	18.96%	17.22%	18.74%	9.73%
lag = 7	MAE	0.0588	0.0604	0.0789	0.0597	0.0619	0.0457
	RMSE	0.2088	0.2033	0.2159	0.2022	0.1923	0.1448
	RAE	5.92%	6.09%	7.95%	6.02%	6.24%	3.24%
	RRSE	18.55%	18.06%	19.19%	17.96%	17.09%	9.35%
lag = 10	MAE	0.0588	0.0616	0.0816	0.0633	0.0633	0.0465
	RMSE	0.209	0.2039	0.2029	0.2014	0.2014	0.1386
	RAE	5.92%	6.20%	8.2152%	6.37%	6.37%	3.30%
	RRSE	18.55%	18.11%	18.0137%	17.88%	17.88%	8.96 %

Table 4.3: Performance of regression models for different window sizes for Dataset II

- M5Rules

According to Weka documentation [5] **Simple Linear Regression** learns a regression model which regress only on a single attribute. The model picks the attribute which results in the lowest squared error. **Linear Regression** is an implementation of linear regression which uses an Akaike information criterion for model selection [10]: models with lower value of the criterion are considered to be better. **REP.Tree** builds a regression tree using variance. Model trees are implemented as **M5P** algorithm[1]. **M5Rules** is a list of rules based on M5 trees which uses separate-and-conquer concept for rules generation. In each iteration it takes a built model tree and makes 'best' leaf into a rule.

In order to be able to compare results with the previous experiment results each dataset was split into two parts: 90% for training and 10% for testing.

This experiment treated all the attributes, namely month, day of month, hour, temperature and precipitation. Energy production attribute was a dependent (target) attribute. In order to build a good forecasting model all the attributes should be considered from the point of view of how they influence on the resulting performance as sometimes an attribute does not give relevant information. Along with exogenous attributes different window sizes (we will also refer to it as 'lag') of dependent variable – which shows how many previous values would be used for prediction model building – should be considered in order to find an optimal one.

First for choosing the lag number to work with we run all the described algorithms with the lag ranging form 3 up to 10 for both datasets. From Table 4.3 we can see that errors is slightly growing with the growing lag number starting from lag = 5.

It can be explained taking into consideration that bigger numbers of lag values lead to

[1]Original M5 algorithm was invented by R

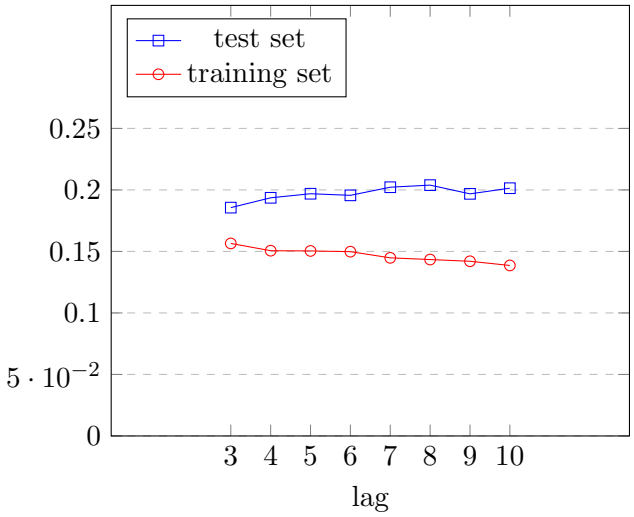


Figure 4.11: M5P RMSE on training and test sets for Dataset II

overfitting. In order to see if there is an overfitting two tests should be carried on: check the performance on the training set (train and test on the same data) and on the test set – overfitting would lead to the decreasing error value with the growing window size on training set and increasing on test set. If we check this with the results obtained by M5P on the training set it seems that the more previous values are considered by model the more the model overfits data since the error on the training set getting smaller and on the test set bigger with the growing lag value (Figure 4.11).

Noteworthy that as there were no exogenous attributes in this experiment we can consider this process as an autoregressive one and as we have seen from the previous experiment (Section 4.2) autoregressive process for single differenced time series was regressed only on one previous value.

One of the possible solutions to avoid overfitting for trees is pruning – reduction of the size of the resulting tree by removing those branches which do not improve accuracy. Nonetheless this technique did not give much improvement for higher values of lag and did not outperformed simple model with lag=3, so we decided to proceed with this value.

We can see form Table 4.3 that there is no method which would be best in terms of all errors but still M5P seems to work better than others for all window sizes. Simple linear regression and linear regression have worse accuracy and in some cases simple linear regression showed even better result then linear regression. However simple linear regression, picking one of the attributes to regress on, picked far past value what seems to be not a sensible solution. M5Rules which are generated from M5Trees have almost the same but little worse accuracy than M5Trees what was expectable.

The unexpected result was obtained with regression trees which should be better predictor than linear regression but not. One of the possible explanation could be the fact that regression tree model built 191 branches with pruning applied, comparing to 30 branches for model trees. Without pruning regression tree model had 1195 branches: that is almost one branch for each 10 values in the data and moreover if we look at the distribution of energy production, where most values lie whether in the very beginning or in the end of the scale (Figure 4.3), we can assume that there is almost one leaf for 1-3 values. Obviously it

Model No.	Model	MAE	RMSE	RAE	RRSE	No. of Rules
1	Prod ∼ Month + Day + Hour + Temp + Prec + lag3	0.0715	0.1952	7.22%	17.36%	92
2	Prod ∼ Month + Day + Temp + Prec+ lag3	0.0743	0.1963	7.50%	17.46%	81
3	Prod ∼ Month + Temp + Prec + lag3	0.0689	0.1883	6.96%	16.75%	59
4	Prod ∼ Temp + Prec + lag3	0.0658	0.1881	6.65%	16.73%	46
5	Prod ∼ Temp + lag3	0.0658	0.1881	6.65%	16.73%	46
6	Prod ∼ Prec + lag3	0.0616	0.1873	6.22%	16.66%	44
7	Prod ∼ Temp + Prec	1.0275	1.293	103.77%	115.03%	237

Table 4.4: Results of M5P algorithm with different sets of attributes (Dataset II)

as a too complex model which overfits the data and cannot properly deal with the out-of sample data.

The next step is to choose the proper model considering exogenous attributes. Given the previous results we decided to proceed with linear regression and M5P models and lag equals 3.

As it was described above, there are five attributes: month, day of month, hour, temperature and precipitation. Several models were built which considered different sets of the attributes and windiw size 3. As we can see from Table 4.4 attributes related to date – month, day of month and hour – did not give any improvement. We have started building models having all the attributes and then, with removing one after another, the models performed better and better. Noteworthy that linear regression models did not considered date attributes at all and had the same errors values for each model except those having temperature and precipitation separately.

The model which considers precipitation looks like being the best one but if to compare these results with results obtained by the models which do not consider any exogenous input at all (Table 4.3) the latter one gives better results. Moreover the last model (Table 4.4) having only temperature and precipitation as attributes and no information about energy produced in last hours performed even worse than just predicting the mean of the process (it is indicated by the RRSE more than 100%) and build extremely complex model with lots of branches comparing to previous ones. It means that for regression models knowledge about the process itself gives more information gain than knowledge about exogenous features.

However for the Dataset I results are slightly different (Table 4.5). We have carried out all the experiments described above. In both cases date attributes are irrelevant: caused more complex models and worse performance. Meanwhile adding attribute 'precipitation' improved performance of the model and again we can see that to build a good model we need knowledge about the previous production as well.

Comparing obtained results with ARIMA models we can conclude that there is little difference between them: as for MAE ARIMA gives slightly but better result, but in terms

Model No.	Model	MAE	RMSE	RAE	RRSE
1	Prod ∼ Temp + Prec + lag3	0.0534	0.1711	8.50%	22.63%
2	Prod ∼ Temp + lag3	0.0514	0.1795	8.19%	23.74%
3	Prod ∼ Prec + lag3	0.0503	0.1805	8.00%	23.87%
4	Prod ∼ Temp + Prec	0.7908	0.9454	125.87%	125.03%
5	lag3	0.0517	0.1832	8.23%	24.22%

Table 4.5: Results of M5P algorithm with different sets of attributes (Dataset I)

of RMSE regression models performed much more better. The results are almost the same so it could be because ARIMA uses regression (on the process itself) and all the regression models considered above also regress on the dependent value as it gives a lot of information about the process.

4.4 Neural Networks

This part of Chapter 4 describes experiments which deal with artificial neural networks: feedforward and recurrent. For both cases we started with simple configurations of neural networks and proceed to more complex ones. As a tool MATLAB for this experimentation was used. Since MATLAB has random initialization for neural networks we have run each configuration 5 times and then select model with the best result.

As it was mentioned in the very beginning if this chapter neural networks have some peculiarities when splitting dataset: along with training and test sets neural networks need validation set. As we have previous models being tested on the last 10% of given time series, the model was trained on 80%, validated on 10% and tested on 10%. However there is one more thing worth to remember: standard MATLAB function divides dataset randomly. First of all time series should be split by blocks not randomly and tested on the last part as we cannot use future to predict the past (usually when doing this the forecast might be better but the question is then whether we can consider it as a forecast). Second we need the same parts for testing for comparable results. From this consideration standard `divideFcn` was forcibly set into `divideblock`.

The performance error that MATLAB uses for training a network is mean squared error (MSE). It is used during updating network weights. As we have discussed before mean absolute error (MAE) is the most preferable measure when dealing with time series so for comparable results other measures were also calculated: MAE and RMSE.

The MATLAB scripts are documented in CD attached to this report.

4.4.1 Feedforward Neural Networks

We have discussed in Section 3.1.2 that there are two things which can affect the performance of neural networks: architecture and initial parameters. In this experimentation session we have started with simple architecture – one hidden layer with five nodes in it – and proceed to more complex ones. The basic architecture of a neural network is presented on Figure 4.12. There is an input block where input attributes are fed. In the hidden block there could be several hidden layers with several nodes and sigmoid function `logsig` is used. For an output layer linear function `purelin` is used. According to [33]

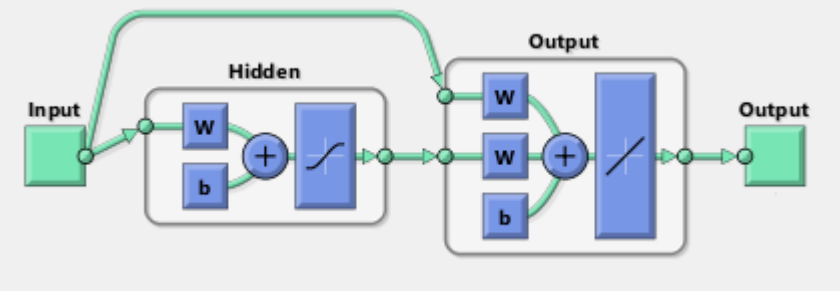


Figure 4.12: Feedforward Neural Network architecture

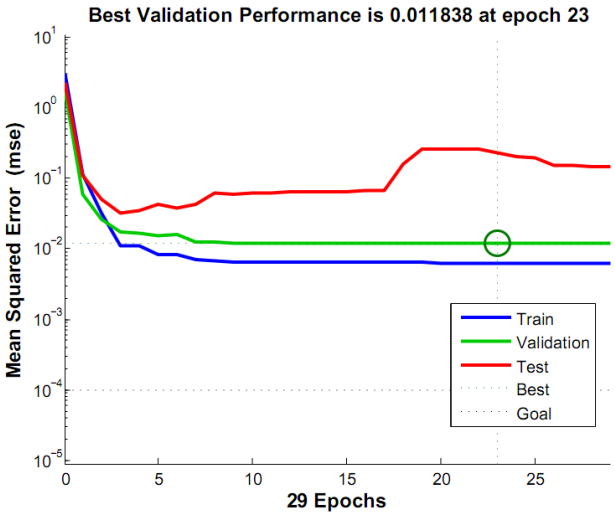


Figure 4.13: Overfitting of a neural network with 3 hidden layers and 20 nodes in each hidden layer

such architectures – with sigmoid function in hidden layer and linear function in output layer – can approximate every function.

When using feedforward 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 [14].

Hence as a train algorithm mostly Levenberg-Marquardt backpropagation algorithm was used (`trainlm` in MATLAB) – one of the fastest and accurate algorithms for big datasets. We have also used other training algorithms but results was not as good as for the chosen one and training time was big comparing `trainlm`.

It was found that Levenberg-Marquardt backpropagation algorithm `trainlm` works better for small number of hidden layers and nodes, as it was mentioned above. Also, even if we could observe good performance for big numbers of nodes, there was usually overfitting observed. You can see an example on Figure 4.13, where with the growing number of epochs the error on training set decreasing and on test set increasing.

Attributes	Architecture		MSE	MAE	RMSE
	No. of hidden layers	No. of nodes			
Prod ∼ Month + Day + Hour + Temp + Prec + lag5	1	8	0.0244	0.0518	0.1463
Prod ∼ Month + Day + Temp + Prec+ lag5	1	10	0.0204	0.0575	0.143
Prod ∼ Month + Temp + Prec + lag5	1	7	0.02	0.0505	0.1415
Prod ∼ Temp + Prec + lag4	1	10	0.0234	0.0565	0.1529
Prod ∼ Temp + lag5	1	10	0.0234	0.0565	0.1529
Prod ∼ Prec + lag5	2	5:5	0.0234	0.0438	0.1529
Prod ∼ lag5	2	5:5	0.0249	0.0493	0.1546

Table 4.6: Performance of feedforward neural networks with different attributes (Dataset I)

Attributes	Architecture		MSE	MAE	RMSE
	No. of hidden layers	No. of nodes			
Prod ∼ Month + Day + Hour + Temp + Prec + lag5	2	10:10	0.032	0.0614	0.1788
Prod ∼ Month + Day + Temp + Prec+ lag4	1	8	0.0277	0.0617	0.1665
Prod ∼ Month + Temp + Prec + lag5	1	10	0.0334	0.0675	0.1828
Prod ∼ Temp + Prec + lag5	1	10	0.0281	0.0624	0.1678
Prod ∼ Temp + lag5	1	7	0.0271	0.0602	0.1647
Prod ∼ Prec + lag5	2	5:5	0.0276	0.0555	0.166
Prod ∼ lag5	2	10:10	0.0255	0.0519	0.1598

Table 4.7: Performance of feedforward neural networks with different attributes (Dataset II)

Also the results were not consistent: even if there is a tendency to perform worse on big architectures for `trainlm` we cannot say that performance is getting better or worse gradually. It could be because of in MATLAB initial weights are taken randomly each time a script is run so it could affect the results.

As neural networks are considered to be an universal approximator which is capable of finding seasonal, trend components and other complex dependencies within data we have included all the attributes as input parameters and then observed how forecasting performance would change with removing some of the parameters. Also different window sizes were considered and in experimental way the lag=5 was found to be the optimal one and we proceeded mostly with this value. Results of the best architectures for each input attributes set are presented in Table 4.6 for Dataset I and in Table 4.7 for Dataset II.

As we can see the best result for Dataset I has been obtained with 5 nodes in each of two hidden layers – 0.0438 in terms of MAE – and with precipitation and window of 5 as input attributes. Including date attributes only worsened net performance as well

as temperature and provided noise to the models however the effect of other attributes was difficult to comprehend as error rates were quite fluctuating. Precipitation as it was expected (this hydropower station is located in mountains) improved performance and authors truly believe that enriching of the datasets with more precise data about precipitation levels could improve overall performance for Dataset I.

Results for Dataset II tell us that there is little evidence of weather impact was found by trained networks and the best result was obtained by the model having only window of 5 of previous values as input attributes. Again it could be explained by the peculiarities of the hydropower plant location: it is located in the valley and there is a pond near it. For sure neural networks are capable of finding long-term dependencies as we can consider this one but in order to find them more data is required in order to perform a deep learning without overfitting.

Comparing with previous experimentations results feedforward neural networks performed slightly better for both datasets.

4.4.2 Recurrent Neural Networks

Using MATLAB for experimentation with recurrent neural networks needs an investigation of some peculiarities of its implementation. Basically there are three types of models in MATLAB for time series prediction (in standard package) [7]:

- NARX – Nonlinear Autoregressive with External (Exogenous) Input (predict series $y(t)$ given d past values of $y(t)$ and another series $u(t)$);
- NAR – Nonlinear Autoregressive (predict series $y(t)$ given past values of $y(t)$);
- Nonlinear Input-Output (predict series $y(t)$ given d past values of series $x(t)$).

In this experimentation session we deal with two first models: NARX and NAR.

NAR is the most simple one as it uses only one series – the process itself. The future values of a time series $y(t)$ are predicted only from past values of that series $y(t-1), \ldots, y(t-n)$:

$$y(t) = f(y(t-1), \ldots, y(t-d)) \tag{4.1}$$

This form of prediction is called nonlinear autoregressive, or NAR.

According to MATLAB Documentation [7] 'nonlinear autoregressive network with exogenous inputs (NARX) is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The equation for the NARX model is

$$y(t) = f(y(t-1), y(t-2), \ldots, y(t-n), u(t-1), u(t-2), \ldots, u(t-n)) \tag{4.2}$$

where the next value of the signal $y(t)$ is depended on the previous values of the signal itself $y(t-1), \ldots, y(t-n)$ and exogenous signal values $u(t-1), \ldots, u(t-n)$.

The basic architecture of recurrent neural networks in MATLAB is presented on Figure 4.14. There is an input block with two time series. Delays in hidden layer can be customized (the default value is 2) and correspond to n in Equasions 4.1 and 4.2 or number of feedback delays. As in feedforward networks sigmoid function `logsig` is used in hidden layer and linear function `purelin` in output layer.

During the training process NAR and NARX models instead of predicted (estimated) values which should be fed to the input due to recurrence, the actual values known on the stage of training are fed to the input [51]. This series-parallel architecture has two advantages:

- The input is more accurate

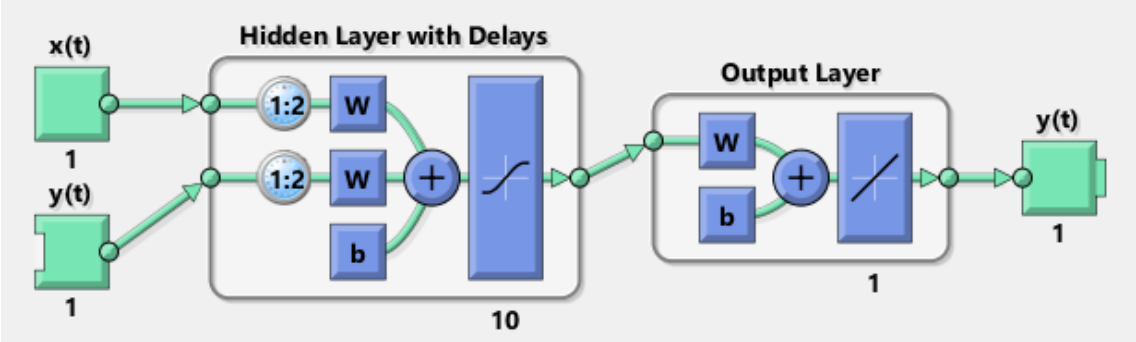


Figure 4.14: NAR Neural Network architecture

Architecture				
No. of nodes in hidden layer	No. of feedback delays	MSE	MAE	RMSE
10	1:2	0.0112	0.0325	0.1057
7	1:5	0.0102	0.0302	0.101
10	1:5	0.0097	0.0269	0.0986
25	1:5	0.0096	0.0275	0.098
30	1:5	0.0102	0.0301	0.1009
10	1:10	0.0101	0.0284	0.1004
20	1:10	0.0098	0.0286	0.0991
30	1:10	0.0102	0.0287	0.101
40	1:10	0.012	0.0396	0.1095
24	1:24	0.0112	0.0439	0.1059

Table 4.8: Performance of NAR network on Dataset I

- Backpropagation can be used for training of such network

The series-parallel configuration (open loop) is used for one-step-ahead prediction and useful for training. Using a toolbox function `closeloop` we can make a NARX (or NAR) network to be a closed-loop network which perform an iterated prediction over many steps and is used for multi-step-ahead prediction.

By default all of the training (including validation and testing) is done in open loop, even if a closed loop network is being trained. After the network is trained and validated in open loop, it is transformed to closed loop for multi-step-ahead prediction. A closed loop network can be configured from the very beginning for all the steps (training, validation and testing), however, it takes longer and the resulting performance is not as good as using series-parallel training.

We divided the dataset by blocks the same way as in previous NN experiments: 80% for training, 10% for validation and 10% for testing.

One of the main parameters in such RNN is a `feedbackDelays`, which indicates how many previous values will influence forecasting.

In Tables 4.8 and 4.9 the results of NAR networks on Dataset I and Dataset II respectively are presented. We can see that better performance was achieved with delays from 1

Architecture			MAE	RMSE
No. of nodes in hidden layer	No. of feedback delays	MSE		
7	1:2	0.0275	0.0649	0.1659
6	1:5	0.0266	0.0554	0.163
10	1:5	0.026	0.0505	0.1613
14	1:5	0.025	0.0493	0.158
25	1:5	0.0246	0.0501	0.1569
10	1:10	0.0261	0.0501	0.1616
20	1:10	0.0251	0.0526	0.1584
30	1:10	0.0265	0.0618	0.1628
7:7	1:5	0.0254	0.0515	0.1594
10:10	1:5	0.0251	0.0521	0.1585

Table 4.9: Performance of NAR network on Dataset II

to 5. Bigger values of feedback delays led to overfitting and hence performance on test sets were unsatisfactory. Interesting that for Dataset I performance improved much more than for Dataset II if to compare with the performance of feedforward neural networks. Thus, taking the best performances, for Dataset I MAE improved from 0.0438 for feedforward nets to 0.0269 for recurrent nets and that is almost 38%. For Dataset II MAE improved from 0.0519 for feedforward to 0.0493 for recurrent networks what is only 5%.

We have proceeded with NARX networks and results are presented in Tables 4.10 and 4.11 for Dataset I and Dataset II respectively. In NARX networks besides feedback delays there are also input delays which indicate delays on the attributes of exogenous input. Again there is a little evidence of temperature correlation with the performance of NARX networks. Also date attributes brought a noise effect. Comparing to NAR and feedforward networks results NARX gives some improvement for both datasets and again precipitation has more influence on forecasting for Dataset I. MAE for Dataset I fell for almost 40% comparing to feedforward networks and other models as well. For Dataset II difference is about 5%. One of the possible explanation to this fact could be the quality of the datasets: it could be more noisy values. Also, there could be more complex dependencies outliers in Dataset II which recurrent neural networks was not possible to reveal.

4.5 Support Vector Machines

This experimentation deals with Support Vector Machines and its implementation in Weka was chosen as a tool. SVM for regression problem is implemented in Weka as SMOreg. However as it was described in theory section SVM has parameters and they affect models performance. Our approach here was to run each dataset trying different values of the parameters in a specified range and find the optimal ones.

The main disadvantage of SVM models is that it is very slow: time to build a model depends on the complexity of this model and with the growing parameters values can increase exponentially. Taking into consideration this fact we have decided to omit date and temperature data and make experimentation with datasets including only precipitation as

Attributes	Architecture			MSE	MAE	RMSE
	No. of nodes in hidden layer	No. of feed-back delays	No. of input delays			
Prod ∼ Month + Day + Hour + Temp + Prec	10	1:5	1:5	0.0109	0.0473	0.1046
Prod ∼ Month + Day + Temp + Prec	10	1:7	1:7	0.0107	0.0389	0.1035
Prod ∼ Month + Temp + Prec	8	1:5	1:5	0.0105	0.0353	0.1023
Prod ∼ Temp + Prec	8	1:5	1:5	0.0199	0.0281	0.141
Prod ∼ Temp	10	1:7	1:7	0.0353	0.0313	0.188
Prod ∼ Prec	7	1:5	1:5	0.0111	0.0234	0.1055

Table 4.10: Performance of NARX network on Dataset I

Attributes	Architecture			MSE	MAE	RMSE
	No. of nodes in hidden layer	No. of feed-back delays	No. of input delays			
Prod ∼ Month + Day + Hour + Temp + Prec	10	1:5	1:5	0.0253	0.0595	0.1589
Prod ∼ Month + Day + Temp + Prec	10	1:7	1:7	0.0285	0.0614	0.1687
Prod ∼ Month + Temp + Prec	10	1:5	1:5	0.0251	0.0545	0.1583
Prod ∼ Temp + Prec	10	1:7	1:7	0.0239	0.0515	0.1547
Prod ∼ Temp	8	1:5	1:5	0.0241	0.0515	0.1551
Prod ∼ Prec	8	1:5	1:5	0.024	0.0511	0.155

Table 4.11: Performance of NARX network on Dataset II

exogenous attribute.

SVM has a parameter C which is a width of the tube ϵ which should be chosen in such a way in order to not overfit data (see Section 2.7.3). This parameter is defined by a user, so to find the optimal value of C a set of experiments was carried on. We tried out RBFKernel (in an experimental way we found that $\sigma = 6$ was the best for this kernel) with different values of the C parameter to find the optimal one. We have done this experiment just to find optimal C width with the small lag = 5, because SVM takes lots of time to be trained.

C	MAE	RMSE	RAE	RRSE
0.1	0.0544	0.1833	5.50%	16.17%
0.5	0.0544	0.1825	5.50%	16.06%
1	0.0541	0.1817	5.47%	16.02%
2	0.0544	0.1821	5.49%	16.17%
3/2.5				

Table 4.12: Performance of the RBFKernel ($\sigma = 6$) for different values of tube width C (Dataset II)

When C is close to zero all instances will be support vectors. The bigger number of C indicates the bigger error when fitting a function but less overfitting and finding the optimal one is a cornerstone of building such models. As you can see from Table 4.12 for our dataset the optimal value of C is 1. So for the other kernel functions we used this value.

A power of PolyKernel polynom is implemented in Weka as an exponent parameter. We tried out two different values of the exponent: $exp = 1$ and $exp = 2$. As time to build models for $exp = 2$ was too big and results much worse, we did not do it for $exp = 3$ and higher values, and the result are in Table 4.13.

Exponent	MAE	RMSE	RAE	RRSE	Time to build model
1	0.0576	0.2078	5.82%	18.49%	141 sec
2	0.3132	0.4185	31.63%	37.23%	2560.88 sec

Table 4.13: Performance of the RBFKernel ($\sigma = 6$) for different values of tube width C (Dataset II)

When the exponent equals 1 the model is linear. The performance of the PolyKernel was better when the exponent equals 1 so the best model here is linear, not exponent.

The PUK kernel function (Pearson VII) can be used as general kernel function. As known from theory when ω is close to 0 it performs like RBF and higher order polynomial functions. As we can see from Tables 4.12 and 4.14 performance of RBFKernel and Puk kernel are similar. This is due to the fact that Puk is a general form of RBFKernel function and that is just a special cases of it.

If you look at the performance of Puk for different values of ω (Table 4.14), you can see that for the $\omega = 0.5$ and $\omega = 1.0$ the performance was the best. The value in region $0.25 - 0.75$ represents a linear function [58], so again linear representations are better than polynomial. For next experiments with different lag values we will use $\omega = 1.0$.

As we can see from the experiments, Puk and RBF kernels are much better than PolyKernel and NormPolyKernel and we have proceeded with Puk and RBF kernels for lags 4, 5, and 7 (Table 4.15).

ω	MAE	RMSE	RAE	RRSE
0.01	0.0537	0.1888	5.42%	17.10%
0.1	0.0537	0.1888	5.42%	16.79%
0.5	0.0542	0.1824	5.47%	16.23%
1.0	0.0543	0.1817	5.48%	16.29%
2.0	0.0546	0.1832	5.51%	16.30%
5.0	0.0551	0.1846	5.5624%	16.42%

Table 4.14: Performance of the RBFKernel ($\sigma = 6$) for different values of tube width C (Dataset II)

	lag = 3		lag = 4		lag = 5		lag = 7	
Method	Puk $\omega = 1$	RBF $\sigma = 6$	Puk $\omega = 1$	RBF $\sigma = 6$	Puk $\omega = 1$	RBF $\sigma = 6$	Puk $\omega = 1$	RBF $\sigma = 6$
MAE	0.0543	0.0541	0.0533	0.0536	0.0521	0.0521	0.0527	0.0535
RMSE	0.1817	0.1817	0.1871	0.1898	0.1874	0.1874	0.1908	0.1971
RAE	5.48%	5.47%	5.38%	5.41%	5.25%	5.25%	5.32%	5.39%
RRSE	16.17%	16.17%	16.64%	16.88%	16.66%	16.66%	16.95%	17.51%
Time to build model (sec)	685.77	1147.92	1338.66	1225.32	1272.41	1502.33	919.07	1289.41

Table 4.15: Performance of the RBFKernel ($\sigma = 6$) and Puk $\sigma = 1$ for different lag values (Dataset II)

From Table 4.16 and Figure 4.15 we can see that models with lag=5 performed better than others and the same is for both datasets. Puk kernel with $\omega = 1$ was used for all models here. The resulting performance for Dataset I and Dataset II presented in Table 4.16. Generaly SVM worked better for models containing only lag attributes without precipitation.

For Dataset I in terms of MAE SVM performed better than other models except recurrent neural networks, and for Dataset II improvement was not that big but performance was almost the same as feedforward neural networks. Even though the Puk function was mostly used during this experiment session which can represent both linear and nonlinear dependencies in some cases model used linear representation.

Again, one of the main disadvantages of SVM is that it takes a lot of time to train a model and requires a lot of calculating resources what makes these models not a preferable one if there is a model with almost the same performance. However authors believe that in case of having dataset where precipitation attribute has better quality (measured hourly

	Model	MAE	RMSE	RAE	RRSE
Dataset I	Prod $\sim$ Prec + lag5	0.0588	0.1907	9.36%	25.22%
	Prod $\sim$ lag5	0.0387	0.1545	6.15%	20.43%
Dataset II	Prod $\sim$ Prec + lag5	0.0589	0.1906	5.96%	16.96%
	Prod $\sim$ lag5	0.0521	0.1874	5.25%	16.66%

Table 4.16: Performance of SVM models

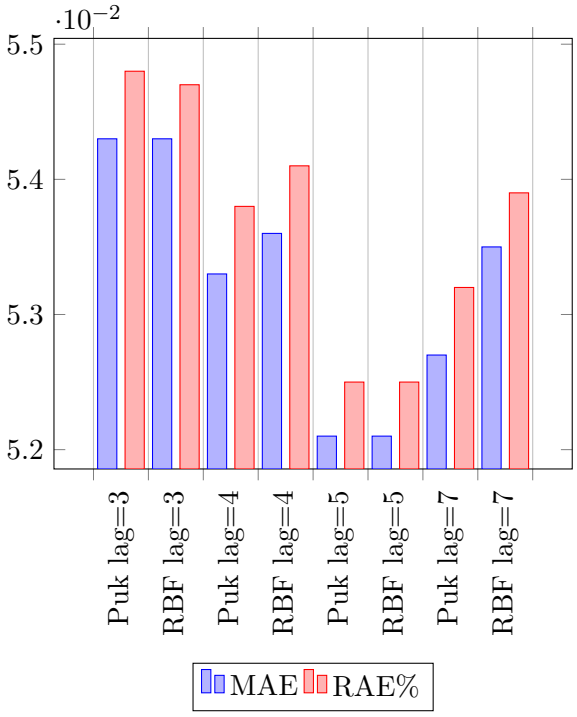


Figure 4.15: Performance of SVM for different lag values (Dataset II)

and at the same location) and perhaps other attributes added SVM could find those complex dependencies.

4.6 Summary

In this section experimentation with the datasets from hydropower plants was described. Before starting the actual experimentation the preprocessing of the initial datasets was carried out which included formation of the datasets structure, as each method and its implementation in a certain tool have peculiarities about input format, splitting into training and test sets, primary analysis of the datasets and their visualisation. As we have two datasets from different hydropower plants experiments for each dataset were carried out though both are time series gathered on a hydropower plants located nearby but nevertheless having some peculiarities. Still main tendencies was the same for both datasets.

Comparing performance of different models we can say that recurrent neuaral networks performed best of all other methods however SVM showed also a good result. The results and their discussion are described in Chapter 5.

Chapter 5

Discussion and Further Work

This chapter deals with discussion over the results and findings described in the previous chapter. First, we compare results of different methods applied for the given time series and their capability of finding dependencies in data. Second, the discussion of the impact of different features are presented. Also we discuss the best obtained result (which was given by recurrent neural networks) in terms of multiple-step-ahead prediction. Finally, we propose further work as there is still a lot of room for improvement.

The experimentation session, described in previous chapter, was carried out on two datasets which were gathered from two hydropower plants. These hydropower plants are located in the nearby to each other however have some peculiarities. Thus one of the hydropower plants, energy production of which is referred as Dataset I, is located in rocky mountains and takes water resources from a river flow. The other one, referred as Dataset II, is located in the valley and has a pond near it what means that it has water almost all year around. As exogenous information precipitation and temperature features were given although they had some mismatching in dimensionality (e.g. precipitation was measured once a day at 7a.m. while energy production – each hour) and were measured not in the same location as power stations but in the nearest town.

Our main objective was to find the best model for forecasting of hydropower energy production. For this purpose we have carried out several experimentation sessions which involved different approaches to forecasting which can be divided into four main domains: statistical, namely ARIMA, modeling, regression analysis, neural networks and support vector machines. For each method a one-step-ahead prediction was performed. Although a multiple-step-ahead prediction, for example next 24 or 48 hours, is of more interest, this kind of prediction involves lots of modification in modeling and therefore was not performed for all methods except for the best performed one – recurrent neural networks.

Taking an incremental methodology through the experimentation we began with the linear models like ARIMA and simple linear regression and moved towards the more complex ones like support vector machines. Altogether nine different techniques were carried out:

- Autoregressive Integrated Moving Average (ARIMA)
- Simple Linear Regression
- Linear Regression
- Regression Trees (REP.Trees)

- Model Treed (M5P)
- Model Rules (M5Rules)
- Feedforward Neural Networks (MLP – Multilayer Perceptron)
- Recurrent Neural Networks (RNN)
- Support Vector Machines (SVM)

The first method – ARIMA – was developed in purpose of time series analysis and quit recommended for forecasting. Simple linear regression is the simplest method from all the regression methods as it uses only one attribute in forecasting model while linear regression uses all of them. Regression trees are trees developed for the purpose of regression, or numeric, problems as model trees as well, however model trees can deal with more complex models as it has regression expressions in its nodes while regression trees deals with numeric intervals in its nodes. The most complex models here are neural networks and support vector machines. The latter are started to be used for time series prediction later than other described methods however according to case studies described in Chapter 3 gives a promising results.

Referring to regression models it was expected that model trees give better results than other methods presented in this domain however it was surprising to have regression trees having the worst performance even over simple linear regression. One of the noteworthy observations during regression tree building was the fact that the built tree had almost 200 branches and we have heuristically assumed that there was about 1-3 values for each leaf in the tree. Of course this is a case of overfitting however tricks like pruning did not give any improvement.

Having this in mind a summary of the best performance for ARIMA, model trees (as a representative of regression domain models), feedforward and recurrent neural networks and support vector machines is presented in Tables 5.1 and 5.2for Dataset I and Dataset II respectively. As we have mentioned before in the thesis, as the main error measure for comparing models Mean Absolute Error (MAE) was chosen as it is an average error measure in time series prediction and not that sensible to outliers as Mean Squared Errors (MSE and RMSE).

Model	ARIMA (0,1,1)	Model trees (Prec + lag3)	MLP (Prec + lag5)	RNN (Prec + lag5)	SVM (lag5)
MAE	0.0469	0.0503	0.0438	0.0233	0.0387

Table 5.1: Best obtained results for each model (Dataset I)

Model	ARIMA (0,1,1)	Model trees (Prec + lag3)	MLP (lag5)	RNN (Prec + lag5)	SVM (lag5)
MAE	0.0583	0.0616	0.0519	0.0511	0.0521

Table 5.2: Best obtained results for each model (Dataset II)

Referring to the *research question № 1* we can see that recurrent neural networks give best results for both datasets. Model trees could not beat traditional time series analysis model – ARIMA. However neural networks, both feedforward and recurrent and support vector machines could beat ARIMA result as we have expected. It was a finding for authors that recurrent neural network performed better than feedforward: yes, historically RNN are used for time series prediction due to its peculiarities of limited number of input

variables as for big number the training time increase a lot. In the meanwhile as the latest studies shows RNN are mostly being used for natural language processing problems such as language modelling, machine translation and so on.

Regarding the ability of finding short-term and long-term effects we can see that by taking into consideration the history of the process (input and feedback delays) RNN were capable of finding short-term effect with a great success (for Dataset I). Yet RNN performed best of others for Dataset II, the improvement of performance was not that big as for Dataset I so a conclusion that for RNN long-term dependencies are of more difficult suggests itself. However if we look at the performance of SVM which also gave a notable result we can see that this model also improved the overall performance for Dataset I in more degree than for Dataset II. We can see that SVM gave quit a promising result even though it was originally invented for classification problems.

Of course finding a short-term dependencies is an easier task than finding long-term ones and improvement for Dataset II was less notable. Looking in retrospect we could have done another experiment involving deep learning with recurrent neural networks or using other, much more complex, architectures involving dynamic memory modules, echo stated networks and so on which are lately are using for finding long-term dependencies in data. However for such experimentation a bigger datasets would be more appropriate as given datasets are relatively small for deep learning – we have seen during experimentation sessions that a increasing of feedback delays, or number of hidden layers, or number of nodes in hidden layers led to better fitting but worse forecasting ability – and so applying of those methods could lead to overfitting.

The *research question № 2* throughout this work was estimating the effects of exogenous features regarding the energy production process on hydropower plants which would have a positive resulting impact on the built model in terms of accuracy. We had the following attributes available for analysis:

- Time-related attributes including month, day of month and hour
- Temperature
- Precipitation

As we can see from Tables 5.1 and 5.2 mostly only precipitation attribute (referred as 'Prec' in Tables) was used in models building if at all. Calendar and time data impact as a noise and decreased performance of the models a lot. Using temperature also did not give any improvement although it has higher correlation with energy production comparing to precipitation. Yet higher correlation does not mean dependency and this attribute also was treated by models as noise attribute: for example linear regression model does not include it into its model at all (as well as calendar attributes). Perhaps the model needed just the knowledge of whether temperature was above or below zero: just to know about the frozen state of the water – however we leave it as a room for further research.

It was interesting to observe the influence of precipitation in terms of different datasets. As we remember Dataset I is the result of energy production of a hydropower plant located in high mountains and Dataset II – the result of energy production of a hydropower plant located in the valley. As it was expected the impact of precipitation was bigger for Dataset I than for Dataset II: thus we can see that for Dataset I all the models except SVM were using this attribute – although correlation between energy production of Dataset II and precipitation was higher than for Dataset I. Actually we have expected that precipitation would influence the accuracy of the models built for Dataset II in more degree rather than for Dataset I. It can be illustrated by the fact that precipitation was

measured only once a day – at 7a.m. – so in terms of time we could not expect to follow the short-term effect of rain for Dataset I in proper way. However we can assume that long-term dependencies for Dataset II is not that 'short-term' speaking of one day measurements: perhaps there is more knowledge about the history of precipitation is required for building more accurate models. This assumption also can be based on the fact that recurrent neural networks with exogenous input used an exogenous input delay – that is a several previous values of precipitation while other models used just one, in-time value.

As the main scope of this work was to build a satisfactory forecasting model we need to provide some remarks regarding it. First of all, we have compared all the models in terms of MAE – Mean Absolute Error. Even though this is a reliable error measure some percentage measure is more understandable for user to see if that performance was satisfactory. Thus Mean Absolute Percentage error was used for this purpose. As we have discussed in Section 3.2.3 this measure can be infinite if some of the target values equals 0. As we have a lot of instances having 0 value we could not use MAPE for representative comparison of the models. However now we can exclude those 0-valued instances and calculate MAPE for not the whole out-of sample data. Thus for Dataset I MAPE equals 3.93% and for Dataset II – 5.04%.

To have something to compare with we have also calculated MAPE error for the model which would just make a prediction taking the previous value: that is, this hour energy production would be the same as one hour ago. Thus for Dataset I it is 8.02% and for Dataset II it is 9.12%, which is actually a surprisingly good predictor for such a simple model. However the performance of RNN is much better. Also as it was described in Chapter 4 by calculating RSE and RRSE we have seen that all resulting models are better than just predicting the mean.

Further Work

Increasing the number of samples in datasets. During this work we have been concentrated on an hourly prediction. One of the things that arise is increasing a time scope of a prediction, as sometimes energy producers need to know not the hourly production, but, for example, daily. It could be easily done by summarizing hourly measures we had in the given datasets and training those models on this new data, however with the increasing a time scope we would reduce the number of samples in data and for two years it would be relatively small datasets.

Another problem dealing with the number of instances in data aroused during the work with neural networks, to be more precise – with performing deep learning. Hence increasing the datasets size would help with performing different tasks and even more models for forecasting could be tried out.

In case of a dramatically increasing of a dataset volume (or using several datasets) one of the common techniques is data **segmentation**. That is dividing a dataset into several different parts where each of them have some unique characteristics in terms of this segmentation. These segments could train each model separately, for example a yearly seasonality can be handled by clustering it into two parts: one of them for winter months and another for remaining months. An interesting aspect here is the impact of precipitation in summer and autumn, when a high correlation with the production is expectable. In winter months precipitation does not have that big impact in data however it would have it when it becomes melting – in the period when temperature changes from negative to

positive. This could be handled either by segmentation into mentioned period or building a seasonal pattern, however for the latter we need the data to be gathered on a longer period of time.

One more concern about datasets is **richness and quality**. As we have described before precipitation and temperature were measured in a nearby town not in the exact locations of hydropower plants. Apparently this bring in more noise in data and may affect the quality of prediction negatively as presence of noise and outliers is more likely to worsen the results. Also there is a lack of domain knowledge – thus, having a knowledge about discharge on a hydropower plant or a feedback about human (or machine) control of the production process would benefit the forecasting accuracy especially for the hydropower plant having a pond near it. The control of the production process – either human or machine – plays an important role especially when building a predictive model. For example, if a power plant has a dam, than a production process would be in high degree depend on which amount of water a producer feed into the plant; then we cannot actually rely on a model which consider only such attributes as precipitation, or temperature. So we believe that it is very important to enrich the datasets yet the bringing in new factors requires making new estimation of their impact on forecasting models accuracy.

However bringing in exogenous factors is a problem for forecasting in real-time. For example, it would help to have predicted $y(t+1)$ once $y(t)$ is available but before the actual $y(t+1)$ occurs. If some trained model is using exogenous attributes for prediction, then it needs to know their values on the $(t+1)$-th point of time. Thus we need to predict these attributes – in our case weather – as well what brings in more uncertainty in the build model and impact with a bigger errors. Thus the weather forecasting should be also considered and ensures that it gives a satisfactory forecasting.

Multiple-step-ahead prediction. In this work we only have considered one-step-ahead prediction, however for applying time series prediction in practice usually requires a multiple-step-ahead prediction, for example, for next 24 or 48 hours. As for our best model – recurrent neural networks – we have actually briefly experimented with the usage of a closed loop in RNN architecture: it replaces the feedback input with a direct connection from the output layer. Obviously, as we increase a horizon of estimation, the error of a model is also increasing – what we have also observed during our experimentation – errors increased $\sim$15%.

However to estimate this model properly it would be helpful to have results of such prediction from other models, but building models predicting several step ahead is the other task and needs a lot of modifications to be done in models and we consider it as a work having a scope of another research.

Hybrid learning is one of the domains that we have not considered during our work. However a lot of authors, as we have mentioned in Chapter 3, have used hybrids applied for time series forecasting. Usually, ARIMA with seasonal adjustment and neural networks. Another promising technique is combining fuzzy logic with neural networks, echo state neural networks and building different architectures of recurrent neural networks.

Chapter 6

Conclusion

The main objective of this work was to estimate different approaches applied to energy production forecasting for small hydropower plants and determine the best forecasting model analysing this two power plants as use cases. These hydropower plants have some differences yet are located in nearby to each other. This gave us an opportunity to compare different models in terms of finding short-term and long-term effects of precipitation. Furthermore the thesis work attempted to analyze which exogenous features regarding energy production process on a hydropower plant are relevant for building an accurate prediction model.

From the energy production measurements of these two hydropower plants two datasets were formed: Dataset I for the hydropower plant located in mountains with rocky ground, and Dataset II for the the hydropower plant located in the valley and having a pond near it.

Several models were built during experimentation session: statistical (ARIMA) and machine learning (regression, neural networks, support vector machines) models. As an input to these models different window size and attributes were fed. Attributes included calendar data, precipitation and temperature. All the models were compared in terms of mean absolute error (MAE) as because of lots of zero values in data commonly used mean absolute percentage error MAPE would become infinite.

In terms of MAE the best method for both datasets was recurrent neural networks (RNN). We have calculated MAPE for a part of datasets where we have excluded zero values. For these subsets MAPE was 3.93% and 5.04% for Dataset I and Dataset II respectively. These result (and results of the all other models used in experimentation) is better than just predicting the mean and better than the model which would just make a prediction taking the previous value: that is, this hour energy production would be the same as one hour ago.

For the short-term dependency on rain effect RNN (we have assumed this to be Dataset I) were capable to find it with a great success comparing to other methods and even better than feedforward neural networks. It could be due to the fact that RNN use a memory layer while feedforward nets do not what makes them a good fitting tool but not the best predictor.

However we cannot say that RNN failed on the dataset with assumed long-term dependencies (Dataset II). It also performed best of other methods but in less degree than in the first case. However the other good performed prediction model – SVM – also gave more improvement on Dataset I. Of course finding long-term effects is a more complex

task than finding short-term, so looking in retrospect, it would be useful carrying out experiments using deep learning however much bigger datasets would be more appropriate for this task.

Estimation of the influence of exogenous input showed that such data as hour, day or month were treated by the models as noise and worsen their accuracy. Using temperature also did not give any improvement although it has higher correlation with energy production comparing to the correlation of precipitation and energy production. The most interesting attribute we have considered was 'precipitation' attribute which had low correlation with the energy production of a power plant located in the valley and almost no correlation with the energy production of a first power plant located in mountains. It is reasonable to assume that for the latter precipitation should have more correlation but the fact is that the measurement of precipitation was done once a day, not hourly, so it is difficult to follow the actual fast precipitation effect of it. However for both datasets mainly precipitation influenced in a positive way improving the overall accuracy of the models.

The main point for further work is gathering more data in order to perform deep learning experimentation and finding seasonal patterns, improving the quality of data – that is gathering, for example, precipitation in the same scale as energy production and at the same location, adding more features, for example the discharge attribute. Regarding building forecasting models there is a variety of hybrid models which worth to be tried out applying to the considered problem.

Bibliography

[1] 1st international competition of time series forecasting. http://reset.ugr.es/ictsf/index.html, October 2014.

[2] Microsoft time series algorithm technical reference. http://msdn.microsoft.com/library/bb677216.aspx, September 2014.

[3] The R project for statistical computing. http://www.r-project.org/, November 2014.

[4] Tourism forecasting. Part one. http://www.kaggle.com/c/tourism1, November 2014.

[5] Weka 3: Data mining software in Java. http://www.cs.waikato.ac.nz/ml/weka/, November 2014.

[6] Mathwork documentation. Feedforward neural network. http://se.mathworks.com/help/nnet/ref/feedforwardnet.html, January 2015.

[7] Mathwork documentation. Neural network time series prediction and modeling. http://se.mathworks.com/help/nnet/gs/neural-network-time-series-prediction-and-modeling.html, January 2015.

[8] Performance measures for numeric prediction. http://ilovenwd.appspot.com/article/attachment/52843, March 2015.

[9] Small hydro power plants. Proven designs, high quality and reliability. http://www.alstom.com/products-services/product-catalogue/power-generation/renewable-energy/hydro-power/hydro-power-plants/small-hydro-power-plants/, March 2015.

[10] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

[11] G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27:822–844, 2011.

[12] E. M. Azoff. *Neural network time series forecasting of financial markets*. Wiley, Chichester New York, 1994.

[13] C.N. Babu and B.E. Reddy. A moving-average filter based hybrid arima–ann model for forecasting time series data. *Applied Soft Computing*, 23:27–38, 2014.

[14] M.H. Beale, M.T. Hagan, and H.B. Demuth. *Neural Network Toolbox^{TM}. User's Guide*. MathWorks, Inc., Natick, MA, 2015.

[15] Ch. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[16] B.L. Bowerman, R.T. O'Connell, and A.B Koehler. *Forecasting, time series and regression: An applied approach*. Thompson Brooks/Cole, Belmont, CA, 2004.

[17] G. Box and G. Jenkins. *Time series analysis : forecasting and control*. John Wiley, Hoboken, N.J, 2008.

[18] E. Busseti, I. Osband, and S. Wong. Deep learning for time series modeling. Technical Report CS 229, Stanford University, December 2012.

[19] O. Chapelle and P. Haffner. Support vector machines et classification d' images, 1998.

[20] C. Chatfield. *The analysis of time series: an introduction*. Chapman&Hall/CRC, Boca Raton, FL, 2004.

[21] B. Cheng and D. M. Titterington. Neural networks: a review from a statistical perspective. *Statistical Science*, 9, 1994.

[22] A. Coghlan. *A Little Book of R For Time Series. Release 2.0*. Parasite Genomics Group, Cambridge, U.K., 2015.

[23] M.G. De Giorgi, S. Campilongo, A. Ficarella, and P.M. Congedo. Comparison between wind power prediction models based on wavelet decomposition with least-squares support vector machine (ls-svm) and artificial neural network (ann). *Energies*, 7:5251–5272, 2014.

[24] K.K. Dobbin and R.M. Simon. Optimally splitting cases for training and testing high dimensional classifiers. *BMC Medical Genomics*, 4:31, 2011.

[25] G. Dorffner. Neural networks for time series processing. *Neural Network World*, 6:447–468, 1996.

[26] W. Enders. *Applied econometric time series*. Wiley, Hoboken, NJ, 2010.

[27] M. Falk, F. Marohn, R. Michel, D. Hofmann, M. Macke, Ch. Spachmann, and S. Englert. *A First Course on Time Series Analysis – Examples with SAS*. Chair of Statistics, University of Würzburg, 2012.

[28] J. Faraway and Ch. Chatfield. Time series forecasting with neural networks: a comparative study using the airline data. *Applied Statistics*, 47, 1998.

[29] E.H. Tay Francis and L. Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29:309–317, 2001.

[30] E.H. Tay Francis and L. Cao. Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48:847–861, 2002.

[31] M. Ghiassia, H. Saidaneb, and D. K. Zimbrac. A dynamic artificial neural network model for forecasting time series events. *International Journal of Forecasting*, 21:341–362, 2005.

[32] C. W. J. Granger. The typical spectral shape of an economic variable. *Econometrica*, 34:150–161, 1996.

[33] M.T. Hagan, H.B. Demuth, and M.H. Beale. *Neural network design*. Pws Pub. Boston, 1996.

[34] J.E. Hanke and A.G. Reitsch. *Business forecasting*. Englewood Cliffs, NJ: Prentice-Hall, 5 edition, 1995.

[35] A. C. Harvey. *Time series models*. MIT Press, Cambridge, Mass, 1993.

[36] A.C. Harvey. *Dynamic models for volatility and heavy tails : with applications to financial and economic time series*. Cambridge University Press, Cambridge New York, 2013.

[37] K. Hatalis, P. Pradhan, S. Kishore, and R.S Blum. Multi-step forecasting of wave power using a nonlinear recurrent neural network. In *Proceedings of the PES General Meeting — Conference Exposition*, pages 1–5, National Harbor, MD, June 2014. IEEE.

[38] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[39] T. Hu, F. Wu, and X. Zhang. Rainfall-runoff modeling using principal component analysis and neural network. *Nord Hydrol*, 38:235–248, 2007.

[40] R.J. Hyndmann and A.B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688, 2006.

[41] E. Izgia, A. Oztopalb, B. Yerlib, M.K. Kaymakb, and A.D. Sahinb. Determination of the representative time horizons for short-term wind power prediction by using artificial neural networks. *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects*, 36:1800–1809, 2014.

[42] H. Jaeger and H. Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, 304:78–80, 2004.

[43] K. Kyoung-jae. Financial time series forecasting using support vector machines. *Neurocomputing*, 55:307–319, 2003.

[44] T.L Lai and C.Z. Wei. Least squares estimates in stochastic regression models with applications to identification and control of dynamic systems. *The Annals of Statistics*, 10:154–166, 1982.

[45] C. S. Lee. Fuzzy neural networks. *Mathematical Biosciences*, 23:151–177, 1975.

[46] K. Levenberg. A method for the solution of certain problems in least squares. *Quart. Applied Math*, 2:164–168, 1944.

[47] L. P. Maguire, B. Roche, T. M. McGinninty, and L. J. McDaid. Predicting a chaotic time series using a fuzzy neural network. *Information Sciences: an International Journal*, 112:125–136, 1998.

[48] S. Makridakis and M. Hibon. The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16:451–476, 2000.

[49] E. Mangalova and E. Agafonov. Time series forecasting using ensemble of ar models with time-varying structure. *Evilving and Adaptive Intelligent Systems*, 27:198–203, 2012.

[50] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Applied Math*, 11, 1963.

[51] K.S. Narendra and K. Parthasarathy. Learning automata approach to hierarchical multiobjective analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:263–273, 1991.

[52] F.A. Olivencia Poloa, J.F Bermejoa, J.F. Gómez Fernándezd, and A.C. Márquezd. Failure mode prediction and energy forecasting of pv plants to assist dynamic maintenance tasks by ann based models. *Renewable Energy*, 81:227–238, 2014.

[53] T. Papadimitriou, P. Gogas, and E. Stathakis. Forecasting energy markets using support vector machines. *Energy Economics*, 44:135–142, 2014.

[54] T. j Sejnowski. Memory and neural networks. *Fara, P., and Patterson K. (Eds.), Memory, MIT Press, Cambridge, MA*, 8:162–181, 2001.

[55] J. Shiri and O. Kisi. Short-term and long-term streamflow forecasting using a wavelet and neuro-fuzzy conjunction model. *J Hydrol*, 394:486–493, 2010.

[56] F. Sowell. Maximum likelihood estimation of stationary univariate fractionally integrated time series models. *Journal of Econometrics*, 53:165–188, 1992.

[57] U. Thissena, R. van Brakela, A.P. de Weijerb, W.J. Melssena, and L.M.C. Buydens. Using support vector machines for time series prediction. *Chemometrics and Intelligent Laboratory Systems*, 69:35–49, 2003.

[58] B. Üstün, W.J Melssen, and L. Buydens. 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.

[59] W.-Ch. Wanga, K.-W. Chauc, Ch.-T. Cheng, and L. Qiu. A comparison of performance of several articial intelligence methods for forecasting monthly discharge time series. *Journal of Hydrology*, 374:294–306, 2009.

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

[61] A. Worster, J. Fan, and A. Ismaila. Understanding linear and logistic regression analyses. pedagogical tools and methods. *CJEM*, 9:111–113, 2007.

[62] B. Wu. Model-free forecasting for nonlinear time series (with application to exchange rates). *Computational statistics & Data Analysis*, 19:433–459, 2005.

[63] A.K. Yadav, A.K.and Sachan. Research and application of dynamic neural network based on reinforcement learning. In *Proceedings of the International Conference on Information Systems Design and Intelligent Applications, Advances in Intelligent and Soft Computing*, pages 931–942, Visakhapatnam, India, January 2012. Shpringer.

[64] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[65] G. P. Zhang and Qi Min. Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160:501–514, 2005.