Bachelor Degree Project

Time series analysis and forecasting
Application to the Swedish Power Grid

Author: Christian Fagerholm
Supervisor: Mirko D'Angelo
Semester: VT/HT 2019
Subject: Computer Science

Abstract

In the electrical power grid, the power load is not constant but continuously changing. This depends on many different factors, among which the habits of the consumers, the yearly seasons and the hour of the day. The continuous change in energy consumption requires the power grid to be flexible. If the energy provided by generators is lower than the demand, this is usually compensated by using renewable power sources or stored energy until the power generators have adapted to the new demand. However, if buffers are depleted the output may not meet the demanded power and could cause power outages. The currently adopted practice in the industry is based on configuring the grid depending on some expected power draw. This analysis is usually performed at a high level and provide only some basic load aggregate as an output. In this thesis, we aim at investigating techniques that are able to predict the behaviour of loads with fine-grained precision. These techniques could be used as predictors to dynamically adapt the grid at run time. We have investigated the field of time series forecasting and evaluated and compared different techniques using a real data set of the load of the Swedish power grid recorded hourly through years. In particular, we have compared the traditional ARIMA models to a neural network and a long short-term memory (LSTM) model to see which of these techniques had the lowest forecasting error in our scenario. Our results show that the LSTM model outperformed the other tested models with an average error of 6,1%.

Keywords: Time series forecasting, ARIMA, SARIMA, Neural network, long short term memory, machine learning

Preface

I want to thank my supervisor Mirko D’Angelo for guiding me through all of this and I want to thank my fiancé Frida for all the love and support. Without you, this would not have been possible.

Contents

1 Introduction

The Swedish power grid consists of over 150 000 kilometers of power lines, roughly 160 substations and 16 stations that connect to other countries [1]. To handle the load demand we have separate power grids, the transmission and the distribution grid. The transmission grid is the power grid that handles the transport of power between the power generators and the substations while the distribution grid handles all the power distribution from the substations to the consumers. All of this is required to power our houses, cellphones, computers etc. Sweden has a wide variety of power generators, but the main power comes from nuclear power plants, hydro or renewable power generators such as wind power, however on the other side other forms of renewable energy sources such as wind power is not constant and can vary a lot. This means that the power plant needs to be adaptable to be able to increase the power production if the wind power generators fail to produce power enough due to weather conditions. The thermal power plants such as coal or nuclear are able to increase their power production, but it often takes a while to reach the new requested power level.

In transmission and distribution electrical power grids, the loads on the power lines are not constant but continuously changing. This depends on various factors, among which the season and the habits of the consumers. Moreover, power usage differs greatly depending on the current time of day and other unexpected factors that could change the amount of power consumption from the grid. If loads increase this is usually compensated in modern grids by using flexible power sources, which can sustain different demand levels up to a certain maximum [2]. However, if resource buffers deplete, the draw could exceed the produced output leading to a power shortage.

One of the most prevalent solutions to this problem is forecasting. If we are able to accurately predict the power load required, we can adapt preemptively. To be able to forecast the required power load we need some kind of data to base our prediction on. Most often, we are using a time series, which is data stored over a long time period.

A time series is a set of observations, each one recorded at a time interval [3]. A discrete time series is a set of observations recorded in a fixed interval. This might be daily, weekly, hourly etc. These time series are often one dimensional, just a time/date and a value however, there are multiple factors to take into account when analyzing the data. A simple graphical representation of the data set can tell us a lot. Figure 1.1 shows the wine sales in Australia. By looking at figure 1.1 we can see that is has a clear increasing trend. We can also observe the increase in sales during the spring and the summer, which is referred to as seasonality. The unusual highs and lows for the data series are called white noise. White noise is a variable that is neither trend nor seasonality but still affects the time series.

Time series is commonly used in other areas such as economics to forecast the stock prices and it is not uncommon for companies to use forecasting techniques to predict the workload to be able to increase or decrease the number of workers needed.

To correctly operate a transmission grid load forecasting is of utmost importance [4, 5]. The currently adopted practice in the industry is based on configuring the grid depending on some expected power draw. However, this analysis is usually performed at a high level and provide only some basic load aggregate as an output [5]. On the other hand, we aim at investigating and finding techniques that are able to predict the behaviour of loads with fine-grained precision. These techniques could be used as predictors to dynamically adapt the grid at run time. The techniques we are investigating are the traditional ARIMA models and different machine-learning approaches.

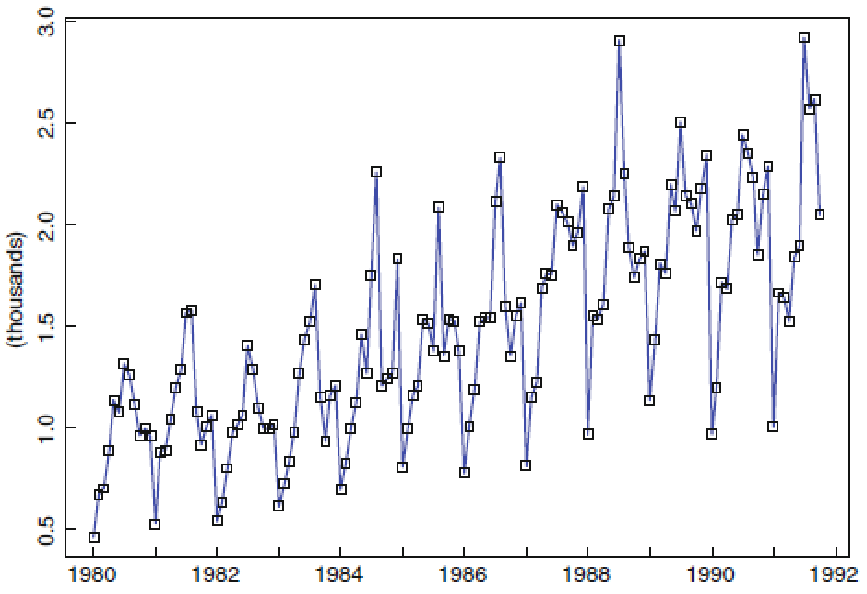


Figure 1.1: Australian wine sales , Jan. 1980-Oct 1991

The field of time series forecasting has been used for multiple different tasks that require planning often due to restrictions in adaptability [3]. One of the most popular algorithms for time series forecasting is ARIMA. ARIMA models were firstly introduced in the 1950s but were made popular by George E. P. Box and Gwilym Jenkins in their book "Time Series Analysis: Forecasting and Control" [6]. To get a clear overview of what ARIMA is, we must first break it down into smaller pieces. ARIMA is a combination of multiple forecasting principles. The Autoregressive model (AR) is a representation of a random process. The autoregressive model specifies that the output depends on the previous inputs and a stochastic term (an imperfect predictable term) therefore making it a stochastic differential equation [6]. Basically, autoregressive models take the previous steps into account when predicting and calculating the next step. The issue with the AR model is that temporary or single shocks affect the whole output indefinitely. To avoid this issue the AR process often has a lag value. The lag value is how many of the previous steps that should contribute more to the output than others. The AR model can be non-stationary as it can be represented by a unit root variable [6].

The MA (Moving Average) model in contrary to the AR model is always stationary. The moving average is a linear regression of the current value against the white noise or random shocks in the series, contrary to the AR model which is the linear regression to non-shock values [7].

There is no reason why you cannot combine these models and that's where the ARMA process comes in. The ARMA model combines the previous two models to make an even more accurate prediction. The ARMA model compares the results for both of the models and makes a prediction based on the results [3]. The issue with the ARMA process is that it assumes that your time series is stationary. This means that the series will not take trend or seasonality into account. That's where ARIMA comes in handy. The I in ARIMA is for integrated, which is the differencing of the previous observations in the series (subtracting an observation from a previous observation in the series to make the series stationary). Non-seasonal ARIMA is often described as $ARIMA(p, d, q)$ where p is the number of lags in the AR model, d is the grade of differentiation (the number of time the data has previous values subtracted) and the q is the order of the MA model[8].

As the ARIMA models are a combination of models, $ARIMA(1, 0, 0)$ is the same as an $AR(1)$ model as the integrated part and the moving average are not used. Lag is the delay in time steps between two points in a data set that you are comparing to each other. The order of MA is how many of the previous shocks that the model will take into account when predicting. A shock is an external influence or something that changed the value to an extreme point, both high or low. In the same way any $ARIMA(p, d, q)$ where p is 0 is equal to an $ARMA(p, q)$ model.

Lastly, we aim to investigate different machine learning techniques as machine learning has been more and more prevalent in many areas in the last years. The idea of machine learning is not to instruct the program on how to explicitly solve a given task, but rather give them a problem and let the computer solve it by using different patterns. One of the most prevalent solutions in regards to machine learning and time series forecasting is the usage of neural networks. A neural network is a network of nodes that are connected and communicate with each other to solve a task. The network has one or more invisible layers that contain "neurons". These are the information nodes that help calculate the result or function for the given problem. The result is not a definitive result, but rather an estimated result and the accuracy of the result depends on the number of hidden "neurons" and layers in the network[9]. The more neurons and layers, the more calculations/operations and the more accurate output. Machine learning is growing as the computational power of today's technology is steadily increasing, which in turn means that we can do even more complex and bigger computations at run time.

1.1 Problem formulation and objectives

In this project, we will investigate different techniques for (load) time series analysis and forecasting. These techniques could be applied to predict the behaviour of the loads in the grid with a fine-grained precision with the aim providing insights on the expected behaviour of consumers in particular time slots (e.g., what will be the expected average consumption in one day).

My project will consist of surveying the literature as well as develop a tool for time series data analysis and forecasting. The first part of the work consists of gathering the knowledge needed to develop the software artifact from the literature (i.e., which are the best techniques/models/algorithms for our scope?). The second part consists of using the gathered knowledge to develop a software artifact dealing with load analysis and forecasting by adapting the state-of-the-art solutions to our case.

We will base our investigation on a real load data set from the Swedish transmission grid recorded hourly between the first of January 2010 to the 31st of December 2015. The data set can be found at https://www.entsoe.eu/data/power-stats/hourly_load/.

Our main research question is which forecasting algorithm is the best for this type of data set/series. What we want to achieve is to have a good comparison of different forecasting algorithms to find out which of the algorithms has the best prediction accuracy for the investigated data set.

O1	Investigate the data set
O1.1	Analyze the properties and characteristics
O1.2	Verify what kind of predictions it is possible to do
O2	Investigate previous research and state of the art techniques
O3	Apply the theory and models to our use case (real data set)
O3.1	Either developing or customizing existing tools and algorithms
O4	Validation of adopted algorithm(s)
O4.1	Validate prediction accuracy

Objective 1, 1.1, 1.2, we have to investigate the data set and extrapolate the properties and the characteristics of the data set. This is crucial as we need to know the properties of the data set in order to know which forecasting algorithms can be used for the data set. We need to know if the data set is stationary, non-stationary, follow a trend or seasonality etc.

Objective 2 requires us to investigate previous research to see what they did and how. We also gather data to see which types of algorithms are used and how they are applied to the investigated data set. Depending on our findings, there might limitations to our data set that makes us favor some forecasting techniques over others. In this step, we exclusively explore the literature, as we need a lot of data to figure out which of these methods and algorithms are applicable to our data set.

Objective 3, 3.1. After performing the literature study, we will apply the identified techniques to our data set.

Objective 4, 4.1 Lastly, we will validate our models by the performance of the chosen algorithms and methods to measure the prediction accuracy based on widely used metrics in this area, both for short and long term forecasting.

1.2 Scope, Limitation and target group

Due to a limited time frame, this report will cover models and techniques in isolation, therefore we will not investigate model combinations.

The targeted audience for this project will mainly be developers and architects in the forecasting field. This will also be of interest to those working with maintaining the power grid. This could also be of interest to economics or statistics students as the same algorithms and methods could be applied to similar time series.

1.3 Outline

The coming chapter will include the methodology and the scientific approach for this project. Chapter 3 will contain the related work and information gathered from the literature study. The 4th chapter will contain the information and analysis of the data set for the time series such as the characteristics. Chapter 5 will contain a small overview of the implementation, libraries used, forecasting models, the forecasts and the results of my forecasts. The 6th chapter will contain the conclusion, discussions and future work for this project.

2 Method

In this chapter, we describe our scientific approach, how we intend to answer the research questions, and we outline reliability and validity for the study.

2.1 Scientific Approach

To answer which of the algorithms that are most widely used and how they are implemented, we did an extensive literature review. This included scientific reports and papers containing the implementation, comparisons and usage of forecasting algorithms. We collected data from these reports and decided which of the algorithms are suitable for the time series and how they have been used previously. In essence, this means that we gather data from the previously mentioned reports and articles and selected which algorithms that we found to be applicable to our data set. This data also included methods on how to analyze our data set and data set properties. We will then use this data to construct models and use these models to experiment on our data set and document the results to assess the overall performance of the algorithms in regards to our data set.

2.2 Method description

Firstly, we did the literature review which gave us much of the contents for the previous chapters. We anticipated that this would give us much information regarding the algorithms and methodology regarding time series forecasting. After analysing the contents, we decided to divide the different forecasting methods into separate groups. One of the major properties of the data sets in the literature is stationarity. Some of the algorithms are limited by not being able to handle seasonality and trends (non-stationary data sets), which is why we decided to further analyze the data set with the methods used in the articles and scientific papers before selecting which algorithms are applicable to our data set. After analyzing the data set we will select some of the most effective/appropriate algorithms and implement a test suite so that we may test and validate their performance in regards to our data set. We will implement the test suite in python with already existing libraries for plotting and forecasting and we expect to get good forecasting results and also graphical representations of the results. Once the implementation of the test suite is done we expect to forecast with good accuracy with multiple methods and record the performance for later validation. In the end, we will provide detailed results for each of the algorithms and some discussion and conclusion will be stated.

2.3 Reliability and Validity

In our case, the *reliability* is dependant on the implementation of the different Python libraries we are using. This should be very high, as many of the most common algorithms for time series forecasting are already implemented in different software libraries and we are mainly using these algorithms and fit them to our data set.

In regards to *construct validity*, this is dependent on the parameters we use to fit the parameter and the way to find the correct parameters for the models. We intend to reduce this issue by studying previous works and use the common methodology for finding fitting parameters to our models. By using the best practices, we should be able to increase our validity.

The *internal validity* of this project is heavily dependant on the system we are using for the experiments and testing, which would be our python implementations and the attached libraries. We cannot really detail how this would affect our validity, but we can assume that the effect would be negligible

The *external validity* is the validity of applying the conclusions of our scientific study outside the adopted context. This should be very high, as we using commonly used methodology and procedures for this field. If we find a similar data set with similar properties, the same methods could be used for that data set. However, we did not explore this in this thesis.

3 Related work

We performed a literature review on related work. We investigated two of the most complete databases, the IEEE database and the ACM database. The search string we used was "time series forecasting" and found the same recurring forecasting methods in a majority of the articles and reports. We did not specify "time series forecasting for a smart grid" as we wish to do a more general search and get a broader view of the time series forecasting field. We will use these data sources as the base for our methodology and implementation. We have decided to separate the methodology into three subgroups. Machine learning, ARIMA, and other techniques.

3.1 Machine learning techniques

In a report from 2012, Mehdi Khashei and Mehdi Bijari[10], proposed a hybrid of ARIMA and ANN (Artificial Neural Network) models to circumvent the limitations of the ARIMA models. The model was then tested on 3 different data sets with different characteristics. According to the report, the ARIMA models traditionally have low accuracy for predicting non-linear data. The results show that their hybrid solution outperformed the individual solutions in regards to Mean Absolute Error (MAE) and Mean Squared Error (MSE) for both short-term and long-term forecasting on both linear and non-linear data.

Bangzhu Zhu and Yiming Wei [11] suggested a hybrid model that uses the ARIMA models combined with a Least Squared Vector Model, which is a neural network model to forecast the carbon prices in the EU. The report proposes three different hybrid models that were used on a non-stationary data set and the results show a clear advantage of combining models to overcome the limitation of ARIMA models with non-linear data.

Mehdi Khashei et al. [12], compare performance for the ANN, SARIMA, and Fuzzy logic models both individually and in combinations. Their results show that it is often better to combine different models than to just go with one model. Many of the other models and methods in combination make up for the fact that SARIMA favors linear data instead of non-linear data. The SARIMA and ARIMA models require a set of previous data, which is not often the case in the real world. Often the data available is limited and as such, ARIMA might not be the best fit for such a forecast.

3.2 ARIMA (Auto-Regressive Integrated Moving Average) models and hybrid implementations

F.Marandi and S.M.T Fatemi Ghomi [13] used ARIMA and were successful in forecasting the waste generation in Tehran. They used the forecasting libraries in the R software to plot out and fit the model. The plotting of the data set clearly showed a trend which means that their time series was not stationary. The report has a clear focus on identifying which ARIMA model to use and also did comparisons between ARIMA models. For their data set, ARIMA(2,0,0) outperformed the others in regards to MAPE, MASE, and RMSE.

Gordon Reikard made a comparison between ARIMA and other popular forecasting algorithms for forecasting the solar radiation [14]. They tried different algorithms on 6 different data sets to compare the short term prediction accuracy. They did hourly forecasts for four hours on each data set and compared the results. The results show that the simpler ARIMA and the hybrid models had the lowest error margin in all 6 data samples.

Debasish Sena and Naresh Kumar Nagwani used ARIMA to successfully forecast the disposable income in Germany [15]. Their time series was non-stationary and their findings suggest that ARIMA is heavily dependant on fitting the correct model. To get the perfect fitting, we need good knowledge regarding the ACF and PACF values to the series. Their results were promising in with a very low prediction error but the report stresses the importance of a correct evaluation of the Auto Correlation Function (ACF) and Partial Auto Correlation Function (PACF) values to fit a good model for the prediction. The ACF and PACF are methods to measure how the values in the data set are affected by previous values in the data set. The R package software was also used for the calculations and plotting.

In regards to our subject, similar tests and research were done in a hospital, where they did forecasting in their medium voltage power grid [16]. The authors discovered that their tests were slightly off due to the lack of information in their time series. Their data set only had 45 days of input, which means that the models only had a small selection of data to use for the prediction. The paper does, however, prove the efficiency and validity of the Box-Jenkins method as the better way to fit a Seasonal ARIMA (SARIMA) model.

A report from 2014 details the usage of a combination of ARIMA and Kalman filtering which is a machine learning technique [17]. This hybrid was used to perform short-term forecasting for wind speed. The Kalman filter is a recursive algorithm where incorrect or noisy predictions can estimate the correct value of a prediction. Since the weather can be random, they found the traditional time series methodology to be very inaccurate. The ARIMA model had a MAPE of 15.442% however, the maximum average error of the fluctuating wind speed data was 50.227%. The Kalman filtering alone had an error margin of 5,80% and a maximum error of 11,26%. The combination of these two methods was surprisingly effective with an error margin of 2,70% and a maximum error margin of 4,94%. The paper proves the validity of wind speed forecasting with ARIMA and the Kalman algorithm for wind speed forecasting, however, it still has some error with the uncertainty of fluctuating winds.

Jianguang Deng and Panida Jirutitijaroen[18] published a study detailing the comparison in load forecasting in Singapore between the ARIMA models and a multiplicative decomposition solution. Their series is similar to ours in terms of stationarity. Their tests proved that the multiplicative decomposition slightly outperformed the SARIMA in this case due to the multiplicative decomposition model not being as affected by random shocks as the ARIMA models. Their main testing was done using Matlab.

A study from 2005[19] suggests a hybrid between machine learning and traditional ARIMA modeling for financial time series forecasting. They combined ARMA with a Generalized Regressive Neural Network (GRNN) to further increase the forecasting accuracy. Their data set was non-stationary as it showed a clear falling trend. The models tested are the ARIMA, the GRNN and their suggested ARIMA-GRNN hybrid, which outperformed the other two individual forecasting models in terms of MAPE, MAE, and Root Mean Squared Error (RMSE).

Heping Liu and Jing Shi[20] did a comparison of the ARMA combined with GARCH models for forecasting the electricity price in New England. The GARCH models were introduced due to the volatility of the electricity price. Their results showed that the ARMA-GARCH-M method slightly outperformed the other 5 ARMA-GARCH models however, their time series was somewhat limited. The time series only had 2 months of recorded data, which might be a small sample to get a definitive answer.

A study from 2014[21] compared the traditional ARIMA model with or without intervention to forecast the cases of campylobacteriosis in New Zealand. An intervention is an external event or influence that drastically changes the data set. Their results proved that ARIMA even with intervention gave poor forecasts due to the intervention in their time series. Holt-Winters method was the far better solution in regards to MAPE and MSE and their results prove that the strength of the Holt-Winters method to predict the coming steps even with a structural change in the time series. The Holt-Winters method is an exponential smoothing method which similar to MA models as they put weight on previously regressed values however, the weight set will decrease exponentially. Holt-Winters is also known as the triple exponential smoothing.

3.3 Other techniques

There are multiple different techniques used for time series forecasting, but the two previously mentioned methods are by far the more popular options today. There are different mathematical models that can be used for forecasting such as state-space models [22]. The idea of space state models is that any dynamic system can be defined by differential equations. This means that the current state of the system is defined by the previous state and a state variable that changed the state. By this observation, we can define any system as a function of its states and external inputs. By knowing some of the observed data, we can calculate the optimal estimation for a selected state. The ARIMA models can also be converted into space state models, as they are to some degree differential equations as well. There are many variations on SSM but the most commonly used is the linear SSM [22]. Other works tend to use a combination of models to create hybrid models to eliminate the inherent flaws of some models [20, 23] and many of the works compare their hybrid solution to the preexisting models [12].

A study from 2013[24] uses an Exponential Smoothing Space State model comparing it with other algorithms to forecast solar irradiance. The used data set had data collected monthly and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) stationary test proved the time series to be non-stationary. The results showed concluded that the ESSS model outperformed ARIMA, Linear Exponential Smoothing, and Simple Exponential smoothing except for two months, but was only 0.5% behind, which means that the ESSS is a reliable model for their data set.

3.4 Discussion

From our literary review, we find that the most commonly used techniques for time series forecasting are different versions of the ARIMA models, such as SARIMA, machine learning techniques such as neural networks and hybrid models. Many of the hybrid models combine ARIMA with machine learning techniques[10, 20]. These will be the starting models we will consider for the forecasting part of our study. The common ground for the ARIMA models and hybrid models using an ARIMA model are the methods to fit the model. Multiple studies use the Box-Jenkins method as a method to check for stationarity and also as a method to fit the ARIMA parameters [8].

There seems to be no difference in model selection for long and short term forecasting. The same models can be used for both long and short-term forecasting [3] as the ARIMA model does not have an inherent issue that prevents long term forecasting. There are also no issues with the neural networks that prevent long-term forecasting. The neural network requires no modifications for long-term forecasting.

One of the major concerns we extract from the related work is that the selection of ARIMA model is heavily depending on whether the data set is stationary or not [6]. A stationary data set means that the time series does not have a trend and is therefore not time-dependent. This affects the selection of applicable models to the data set as the AR, MA and ARMA models are not applicable to a non-stationary data set. This means that we need to analyze the data set thoroughly before we can decide which type of ARIMA model is applicable in our study. In the next section, we will further explore this facet.

4 Data set analysis

In this section, we will further analyze the data set and its properties. This is important for selecting the appropriate models and methods for the forecasting part. Section 4.1 will cover the analysis of the data set and section 4.2 will cover a stationarity analysis by investigating the autocorrelating function of the data set and the unit root test.

4.1 Data set description

The data set that we are using is recorded on an hourly basis over 5 years time period and it shows us the power load in the transmission grid in Sweden from the years 2010-2015. This gives us enough data for the forecasting and we should be able to do accurate forecasts. The data set was being gathered by multiple power organisations and the load in the power grid is stored in GWh (Gigawatts = 10^6). This data set contains the recorded power in the transmission grid and is publicly available at Entsoe web page [1].

Date	00	01	02	03	04	05	06	07	08	09
2010-01-01	18754	18478	18177	18002	17897	18042	18441	18870	19061	19190
2010-01-02	18119	17786	17688	17762	17831	18049	18601	19210	19785	20398
2010-01-03	19388	19059	18920	18928	19020	19278	19722	20214	20574	21003
2010-01-04	18833	18488	18398	18407	18605	19367	20941	22571	23276	23362
2010-01-05	19642	19390	19375	19562	19925	20627	22336	23898	24664	24757
2010-01-06	21285	20884	20689	20636	20759	21010	21707	22455	22826	23291
2010-01-07	20817	20548	20491	20521	20792	21564	23112	24798	25428	25335
2010-01-08	21249	20911	20881	21082	21297	22009	23654	25048	25678	25697
2010-01-09	21731	21372	21218	21211	21291	21513	21999	22661	23174	23832

Table 4.1: A sample of the data set

Table 4.1 shows a small sample of the data set. The values are the power load in GW/h and the headers are the hour of recording. There are in total 2192 rows and 24 columns for the time series. From this sample, we can see that we are having lower value in the early hours, but it is slowly increasing until 05:00 and then increases faster. This is very likely the effect of industries, shops, and other infrastructure starting up their services. The load remains at roughly these values until 16:00 where it starts declining again. It is very likely that the reduction of power usage after 16:00 is also due to the industry and other power demanding consumers close for the day. The values then continue to drop and reach their lowest points at about 02:00-03:00, which is reasonable as most of the Swedish population is sleeping by that hour, and then the cycle repeats itself throughout the year. By further analyzing the data set, we can see that there is also a weekly seasonality where the Mondays have a higher power load than the other workdays. We then have a lower power load from Tuesday to Thursday, but between the days in this interval, there is little to no difference in the power load. On Friday and through the weekend we have a decreasing power load and we reach the lowest part on Sunday which is reasonable as we reach the end of the week and we transition into the new week. Then it rapidly increases and reaches the highest point on Monday. As we have these stable cycles, we have a seasonality where the duration of this season is one week, therefore we have a weekly seasonality as well as the yearly.

In order for us to select appropriate algorithms or methods to use for the data set forecasting, we must first analyze the data set to examine its properties. From what we can

[1]https://www.entsoe.eu/data/power-stats/hourly_load/

tell from the overview is that we have a clear seasonality, but we also see a minor declining trend on the peaks of the curve. The summer seasons seem to be roughly the same, but the winter seasons seem to decrease in their maximum power usage. This could be affected by many different external factors, such as milder winters or more effective/alternative heating sources.

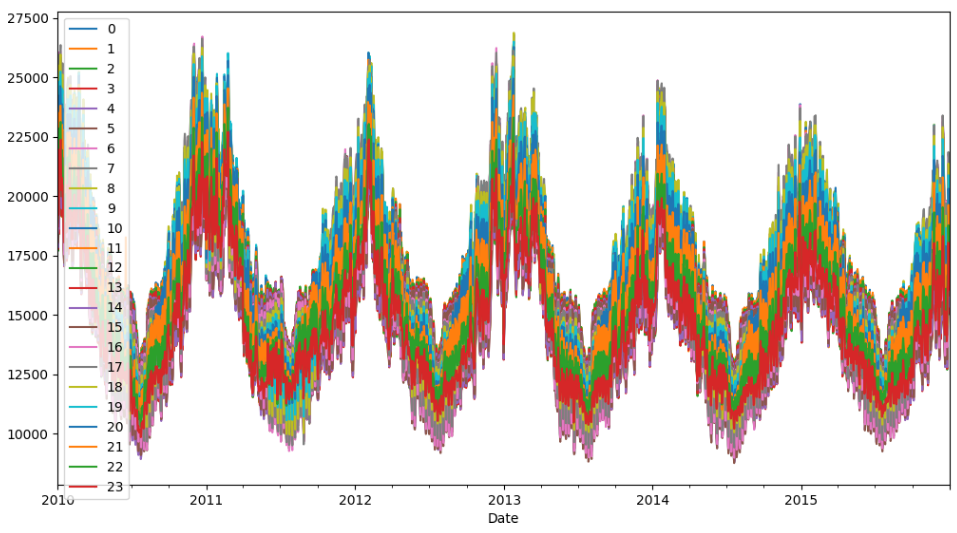


Figure 4.2: The overview of the time series.

4.2 Stationarity Test

When comparing data sets or time series, one of the most important properties to analyze is the stationarity of the time series [3]. A stationary time series means that it is not affected by seasonality or trend, but is rather consistent. This means that the time series fluctuates between roughly the same values at all times, and are therefore more predictable. A consistent seasonality, like the one we have in this data set, does not make the time series non-stationary. It is stable in the sense that we can expect the same values for each season. Stationary time series are easier to predict since they are not time-dependent [6]. The plotting of the data set can be seen in figure 4.2. Here we can see that the time series is stable, as over the time period we keep coming back to the same values. In figure 4.2 looking at the interval between 2010-2013, the curve looks symmetrical which is a sign of stationarity however, we can see that the peaks of the curve are decreasing, which means that we have a decreasing trend in the winter period.

A non-stationary time series is affected by some kind of interference or outer influence such as trends or seasonality. There can be many things that affect the data. If we look at our data set, we can see that the tops of the winter periods are slightly decreasing for every year. This could occur for any number of reasons, and this also affects the data as there is a lower power demand. This makes the time series much harder to predict since we need to analyze and measure this interference so that we may add take them into account when forecasting.

To prove if the data set is stationary, in addition to observing the data, can be done by doing a root test. A unit root test is a more mathematically precise way to check for stationarity as using only an observation of the time series could be misleading. By doing a root test, we can prove by using mathematical formulas and comparisons whether the time series is stationary or not. The intuition behind using a root test is to decide how strongly a data set is defined by a trend [25]. A root test used for verifying if our data set can reject the so-called null hypothesis. The null hypothesis is that the time series can be represented by a unit root (and some time variable), which would prove that the series is non-stationary. The other hypothesis would be to reject the null hypothesis, that there is no unit root to represent the series, and would, therefore, prove that the series is stationary since it does not have a time-dependent structure. If there is a unit root, then every value in the time series is a factor of that root in some sense. Hence, if a series has a unit root, then the series shows an unpredictable systematic pattern and is, therefore, non-stationary [25]. To evaluate the stationarity of our data set we perform the Augmented Dickey-Fuller test [25]. This test uses an autoregressive model and optimizes for different lag values. The lag value is the delay in time between two values in the series that you are comparing to each other. This is done to calculate the autocorrelation function for the series. To get a deeper understanding, we must first analyze the normal Dickey-Fuller test. The formula for the Dickey-Fuller root test is shown below.

$$y_t = \rho y_{t-1} + u_t \tag{1}$$

For equation 1 y_t is the variable of interest, t is the time index, ρ is a coefficient, and u_t is the error term. A unit root is present if $\rho = 1$. The model would be non-stationary in this case. After some rewriting of the formula, we get the regression model to look like equation 2.

$$\Delta y_t = (\rho - 1)y_{t-1} + u_t = \delta y_{t-1} + u_t \tag{2}$$

where Δ is the first difference operator. This model can be estimated and tested for a unit root where $\delta = 0 (where \delta \equiv \rho - 1)$. Since these tests are done over residual data instead of raw data, these do not follow the standard distribution for calculating the critical values, but are instead following a specific distribution. These are the critical values that can be shown in the table 4.2.

The Augmented Dickey-Fuller[25] includes a lag variable in order to remove the autocorrelation from the results. There are three different versions of the Dickey-Fuller test. There is the normal unit root test, unit root with drift and lastly, a version for a unit root with a drift and a deterministic trend. We will use the normal one, as it is safer to use, even if we should have a drift or deterministic trend. Wrongly inclusion of the drift and deterministic trend reduces the power of the root test[25]. The formula for the Dickey-Fuller test for a normal unit root is shown in the formula below.

$$\Delta y_t = a + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + ... + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t \tag{3}$$

The y_t is the variable of interest, t is the time index and u_t is an error term. The Δ is an operator for the first difference level. α is a constant, β the coefficient on a time trend and p the lag order of the autoregressive process. Imposing the constraints $\alpha = 0$ and $\beta = 0$ corresponds to modelling a random walk and using the constraint $\beta = 0$ corresponds to modeling a random walk with a drift. The unit root test is then carried out under the null hypothesis $\gamma = 0$ against the alternative hypothesis of $\gamma < 0$. We did not set these parameters ourselves, instead, we used an already implemented method. We will discuss this further in the implementation. Once a value for the test statistic is computed it can be compared to the relevant critical value for the Dickey-Fuller Test. Table 4.2 shows a figure of the critical values for the Dickey-Fuller test. If the value is below the critical value, we can then reject the null hypothesis and prove stationarity.

This is one of the most important steps when analysing stationarity and properties of the data set [16]. Figure 4.3 shows the ACF and PACF values for our data set. We can

Sample size	1%	5%
T = 25	-4,38	-3,60
T=50	-4,15	-3,50
T=100	-4,04	-3,45
T=250	-3,99	-3,43
T=500	-3,98	-3,42
T= 501+	-3,96	-3,41
Calculated value	-3.085	

Table 4.2: The critical values for the Augmented Dickey-Fuller tests for a series with trend and our results from the root test.

see that we have a linear downward going trend and also that we have seasonality as the ACF values are cyclical. This proves that the data set does have a trend, which means that the time series is non-stationary. This trend will have to be accounted for when fitting the forecasting models. To correctly analyze a root test, we have to compare our calculated values to the critical values of the Augmented Dickey-Fuller test. The critical values are nothing we calculate but rather, they are constants that have already been established. If our calculated values are lower than the 5% value, then we can reject the null hypothesis with more than 95% certainty, and if the value is below the 1% value, we can reject the null hypothesis with more than 99% certainty. Looking at the critical values in table 4.2, we can see that our calculated value is not lower than the critical values. This means that our root test values are not lower than the critical values which in turn means that we cannot reject the null hypothesis and the time series is non-stationary. For example, if our value would be -3,50 and our data set has more than 500 inputs, then we would be able to reject the null hypothesis with a 95% certainty as our calculated value is lower than the critical value for 5%. We would not be able to reject the null hypothesis with 99% certainty as our value is not lower than the 1% critical value. If our calculated value is higher than the critical values, we cannot reject the null hypothesis at all, which means that we cannot prove stationarity.

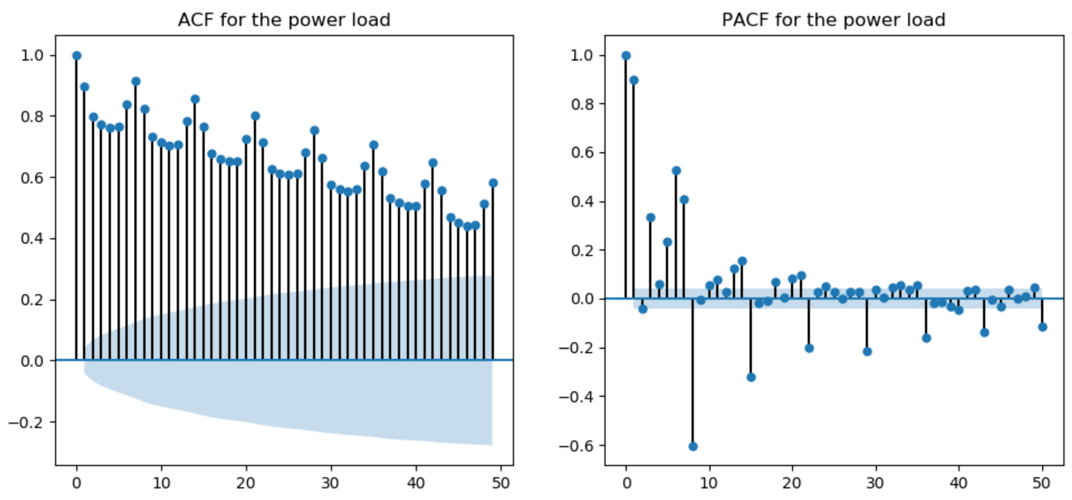


Figure 4.3: The ACF and PACF values for the series.

5 Forecasting

In this section, we will delve deeper into the actual forecasting of the data set and fitting existing models used in the literature. We will look at the difference between long-term and short-term forecasting. We will explain the different forecasting metrics adopted to evaluate our solution. Finally, we will explain how we trained the models for the time-series forecasting.

5.1 Long term forecasting vs short term forecasting

It is possible to classify forecasting in two categories, long term, and short term forecasting. Short term in regards to forecasting is the most common choice of approach due to having smaller errors in the forecast. In fact, intuitively, it is easier to predict the weather for tomorrow than to predict the weather for next year for example. There are also other outside factors that might affect the prediction which will also increase the error on a long term forecast. If you were forecasting business economics, price change might be a big factor and that could greatly increase the error of your forecast. There are multiple outside factors that might not be accounted for, which is why short term forecasts are preferred over long term forecasts generally. The methods used for short term forecasts are also used or can be adapted for the long term forecasts[6]. As an example, A. Sorjamaa [26] adapted a neural network model to do long term forecasts by doing direct forecasts instead of a regressive forecast. The neural network picked out reliable data from their neighboring nodes and used them for the forecast.

Since the data can differ in regards to the time interval between recorded values, we cannot decide if a forecast is a short or long term forecast based on the time between recordings. Therefore, we calculate the number of time steps when attempting to forecast. A step is the next data point, so if you have your data recorded daily, one step would be one day. To evaluate the short term vs long term capabilities we have decided to use daily data and set the test periods where the short term are 1, 5 and 10 steps (days) forecast, mid-range forecasts will be 30 and 60 days ahead and lastly, long term will be 6 months to one year forecast.

5.2 Forecasting error metrics

In order to forecast the next coming steps in a data set, we need some previous data. The most common approach is to divide the data set into two parts, training and testing data. The training data is usually 80% of the data set and the test data is the remaining 20% of the data set [6]. The test data is only used for comparing the forecast with the actual value, while the training data is used for fitting the forecasting model or feeding the neural network if you are employing this machine learning approach.

To evaluate the individual forecasting models prediction ability, we use commonly used metrics found in the literature for measuring prediction errors. Prediction errors are the difference between the predicted values and the actual values. To represent these differences, we have to use error metrics commonly used in the literature. Referring to equation 4, MAE is the average of the absolute values of all the errors in the forecast and referring to equation 5, RMSE is the error in regards to the median of the data.

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|X_t - Y_t| \tag{4}$$

$$RSME = \sqrt{\frac{1}{N}\sum_{1}^{N}(X_t - Y_t)^2} \tag{5}$$

The X_t is the predicted data and the Y_t is the training data of the series.
We also calculate the percentage error of the forecasts as a pure data result can be harder to evaluate and eventually compare with other results. Referring to equation 6 the common used metric for the percentage error is MAPE. It is the average of all the predicted values divided by the actual values.

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}|\frac{X_t - Y_t}{Y_t}| \tag{6}$$

5.3 ARIMA Models

One of the most common techniques that we found in the literature was ARIMA. ARIMA is a collection of models who are used in combinations of each other in order to get an as accurate result as possible. There are even multiple extensions to the ARIMA models, as there are certain limitations with the ARIMA models. The AR, MA, and ARMA require the data set to be stationary and the ARIMA does not handle seasonality [6] which is why one of the most popular extensions to the ARIMA models is the SARIMA. ARIMA models also have issues with non-linear data and are often handled by combining the ARIMA model with different techniques[23].

5.3.1 Training the model

As we mentioned in the previous section, the common practice is to have your training data be 80% of the original data set, and the testing data for verifying your accuracy will be the remaining 20%. As the time series stretches over 5 years, the first 4 years will be the training data and the last year will be the testing data for verification. The training in this sense will be the fitting of the model. The training data will also be used as the stored data for the regressions (we need previous data to do any type of regression method).

5.3.2 Model selection and fitting of model

Since our previous results showed that the time series is non-stationary, this affects which of the ARIMA models we can use. We are not able to use the AR, MA or the ARMA as they require the time series to be stationary. Since our time series shows both seasonality and trend, we have chosen to use the SARIMA model as the SARIMA model takes both seasonality and trend into account when doing the forecast. Hulisani Matsila et al [16] used a SARIMA model for the forecasting with success in regards to their small data set. They had a weekly seasonality and did shorter forecasts with high accuracy, which is why we chose to use the SARIMA model instead of the ARIMA model. This, however, requires more parameters to fit these models as the SARIMA model has two sets of parameters. The SARIMA model requires 7 parameters and is usually denoted with $SARIMA(p, d, q)(P, D, Q)m$ where p, d, q are the trend order and P, D, Q, m are the seasonal order [7]. The trend order is the same values you would use for fitting an ARIMA model, which means that p is the autoregressive order, d is the differencing order and q is the moving average order. The seasonal elements also have the autoregressive, difference and the moving average order however, the m is the number of time steps for each season.

In order for us to get the trend order, we have to fit the model for an ARIMA model first. According to George E.P. Box et al [6], one of the most used methods to fit an ARMA/ARIMA model to the series is to use the Box-Jenkins method. The Box-Jenkins method requires us to calculate and plot the ACF (autocorrelation function) and PACF (Partial ACF) for the series. The ACF is a measurement of how related the actual value is to the previous values including trend and seasonality. The PACF, unlike the ACF, finds the correlation between the residual values in the series, therefore it is only the partial function. The ACF and PACF are commonly used to get a good overview of the time series, but these are required for the Box-Jenkins method as we use these to find the best parameters for the SARIMA model fitting our data set.

Figure 4.3 shows the ACF and the PACF values for the time series. According to the Box-Jenkins method, we make use of the ACF and PACF and their behaviour to select the parameters for the ARIMA model. The Box-Jenkins can be summarized by reading the ACF plot and identifying the ACF plots behaviour with this method. The bold text is the behaviour of the ACF plot and the following text is the suggested model/action according to the Box-Jenkins method.

- **ACF has an exponential decay to zero** : Autoregressive model (use the partial autocorrelation plot to identify the order p). The p is the last notable value in the PACF plot before they all become similar.
- **ACF has damped oscillations and is decaying (exponentially) to zero:** Autoregressive model
- **ACF has one or more spikes, the rest is essentially zero :** Moving average model (order q identified by where autocorrelation plot becomes zero).
- **ACF has an exponential decay starting after a few lags :** Mixed autoregressive and moving average model.
- **ACF has no significant autocorrelations (zero or close to zero) :** White noise. The test is inconclusive.
- **ACF has high values at fixed intervals :** Include seasonal autoregressive terms
- **ACF has no decay to zero or a very slow decay :** Non-stationarity or long-memory effects.

Referring to figure 4.3, we can see two things that are mentioned in the Box-Jenkins method. We have both cycles with even intervals, which means that we have seasonality, and we have a very slow decline. This means that we have a trend and this further proves the time series is non-stationary. The amount of dots in the cycle is 7, which means that we have a weekly seasonality, which is what we discovered earlier. To be able to find the parameters for the seasonal order, we have to difference the series. If the trend is linear, we can remove the trend with a first-level differentiation and only get the seasonal values. After differencing, we plot the new ACF values, and we do the Box-Jenkins method again in order to fit our seasonal values for the SARIMA model. Differencing is performed by subtracting the previous observation from the current observation, which is shown in equation 7.

$$difference(t) = observation(t) - observation(t-1) \tag{7}$$

Taking the difference between consecutive steps is called 1-lag differencing and data sets with a seasonal component the lag may be expected to be the period of the season. Some temporal structure might remain after differencing, such as if the series contains a non-linear trend. The differencing may then be done as many times as needed in order to force stationarity and the number of differencing is called difference order.

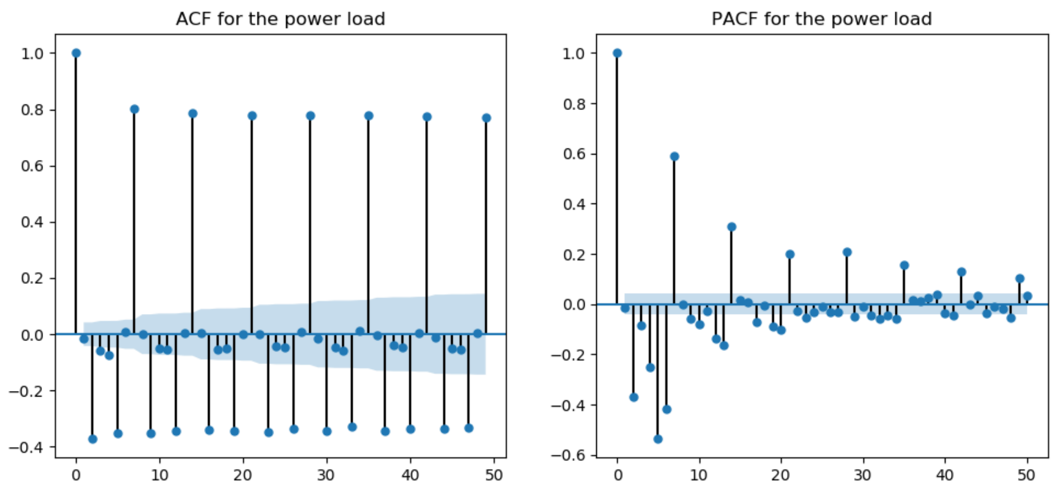


Figure 5.4: The ACF and PACF values for the differenced series.

Figure 5.4 shows the differenced values for the data set. We difference the series to force stationarity for the data set. If we get a conclusive AFC and PACF for the differenced series, we can then use the Box-Jenkins method to find the values for the seasonal parameters for the SARIMA model. The differenced plot is not conclusive either. We cannot really detail the values from this plot either. We tried with multiple levels of differencing, but no model gave a good result. There was no plot where we could fit the model with the Box-Jenkins method, as they did not follow the patterns required, so we have decided to try a different approach. Another way to fit the parameters for the SARIMA model is to do a grid search. A grid search is a trial and error method to try out the different possible models and compare their error results.

Instead of using the Box-Jenkins method, we went with a grid search to get the most fitting parameters for the SARIMA forecasting models. The plan is to define a set of model combinations in order to test which forecast model that has the lowest error in regards to our data set. To evaluate which model is the best fitted, we evaluate based on multiple one-step forecasts and comparing with the actual value and calculate the RMSE. Figure 5.3 shows the result of the grid search. We start by splitting the data into the test and training data. After the split, we take model parameters from our selection and fit our data set with these model parameters. We then make multiple one-step forecasts and compare the predicted values with the actual values. Once we have done this for a number of steps, we calculate the RMSE for the predictions and use this as a metric for evaluating the model. As there are many combinations to try, we speed up this process by having multiple threads doing these evaluations and lastly, we present the top 3 models with the lowest error. Since the top three models presented have the same autoregressive component as well as seasonal component, it is safe to say that the model seems like a good fit. The trend is the only part that changes between the models.

The models that the grid search found out to be the best model is the $SARIMA(2,1,1)(2,0,2,7)$

Model	Trend	RMSE
(2, 1, 1)(2, 0, 2, 7)	Constant and linear	834.3001089409712
(2, 1, 1)(2, 0, 2, 7)	Linear	836.9687538893095
(2, 1, 1)(2, 0, 2, 7)	None	837.1173747930893

Table 5.3: Results from the grid search

with no trend, linear trend and also constant with a linear trend. The error difference between all three models is so low, that we will have to test them all. We will hereby mention them by model 1 (constant and linear trend), 2 (linear) and 3 (no trend). The last seasonal parameter being 7 is not surprising, as we could see in the data set that there was weekly seasonality.

5.3.3 Short-term forecasting

Firstly, we will test the short-term forecasting ability of the three models. We are using the same training data and testing data for every model. This will include one day, 5 days, 10 days and 1-month forecast. From what we can see from table 5.4 the third model, where we do not account for the trend, the one day (one step) forecast has the highest accuracy. This is the only experiment where this model outperforms the other models. For the later experiments, the trend has a bigger effect as we can see on the 5-day forecast. The second model (linear trend) has the highest accuracy for this forecast, but the first model's results are close to the second model. For the next coming experiments, the first model outperforms the other models with ease. Looking at the table 5.4, it is clear that from the 10-day forecast and forward, the first model outperforms the other two models in all three of the measurements.

Model	Metric	1 Day	5 Days	10 Days	1 Month
1	MAE	1886.024	1163.223	**1162.065**	**953.017**
1	RMSE	1886.024	1245.288	**1259.660**	**1124.953**
1	MAPE	11.475%	6.496%	**6.073%**	**4.678%**
2	MAE	1478.747	**1127.285**	1470.273	1949.965
2	RMSE	1478.747	**1210.464**	1614.663	2078.490
2	MAPE	8.997%	**6.241%**	7.535%	9.505%
3	MAE	**1188.342**	1276.204	1768.926	2408.409
3	RMSE	**1188.342**	1437.822	1959.292	2542.665
3	MAPE	**7.230%**	6.985%	9.005%	11.752%

Table 5.4: Results from the short-term forecasting with the SARIMA models

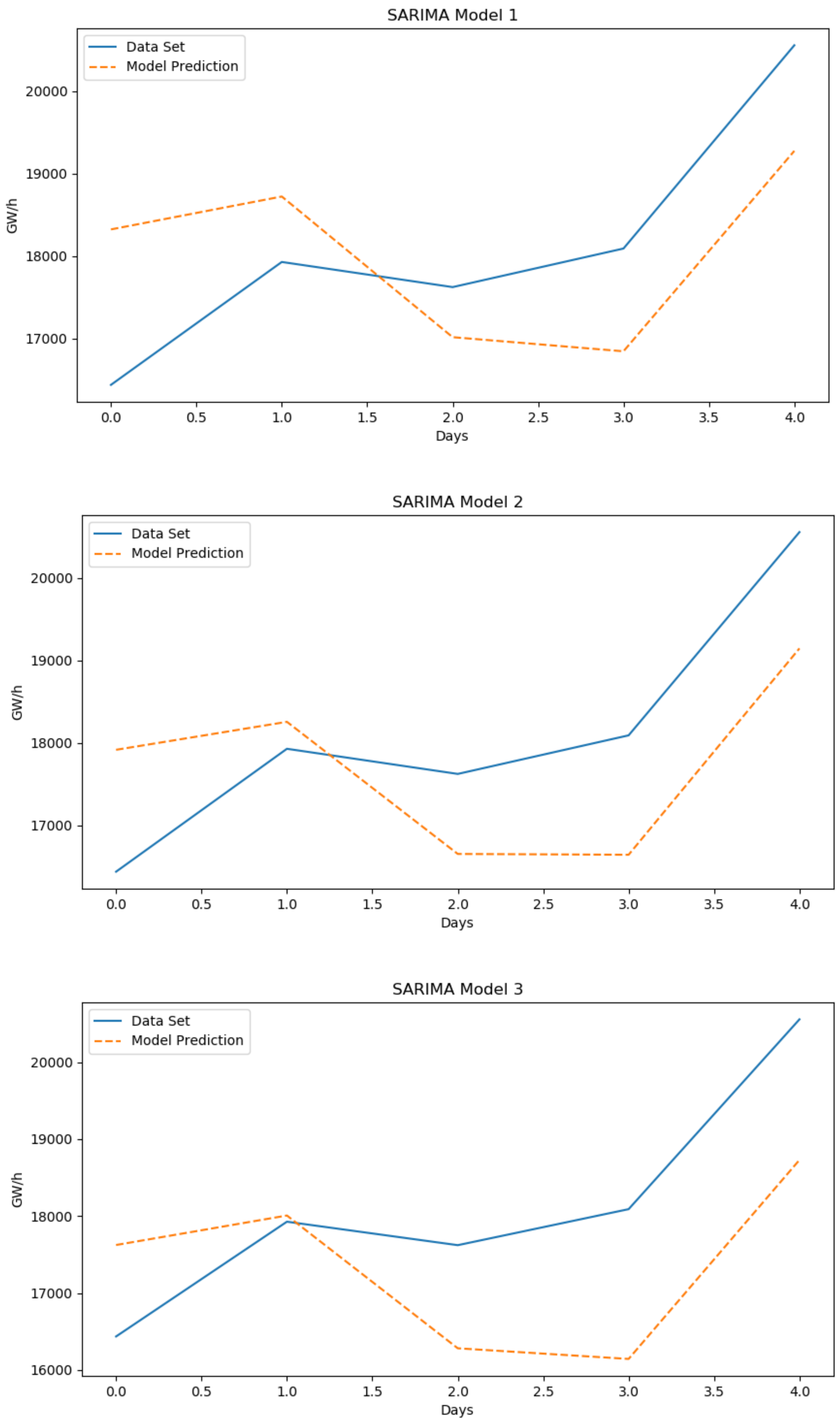


Figure 5.5: graphs for the 5 day forecasts

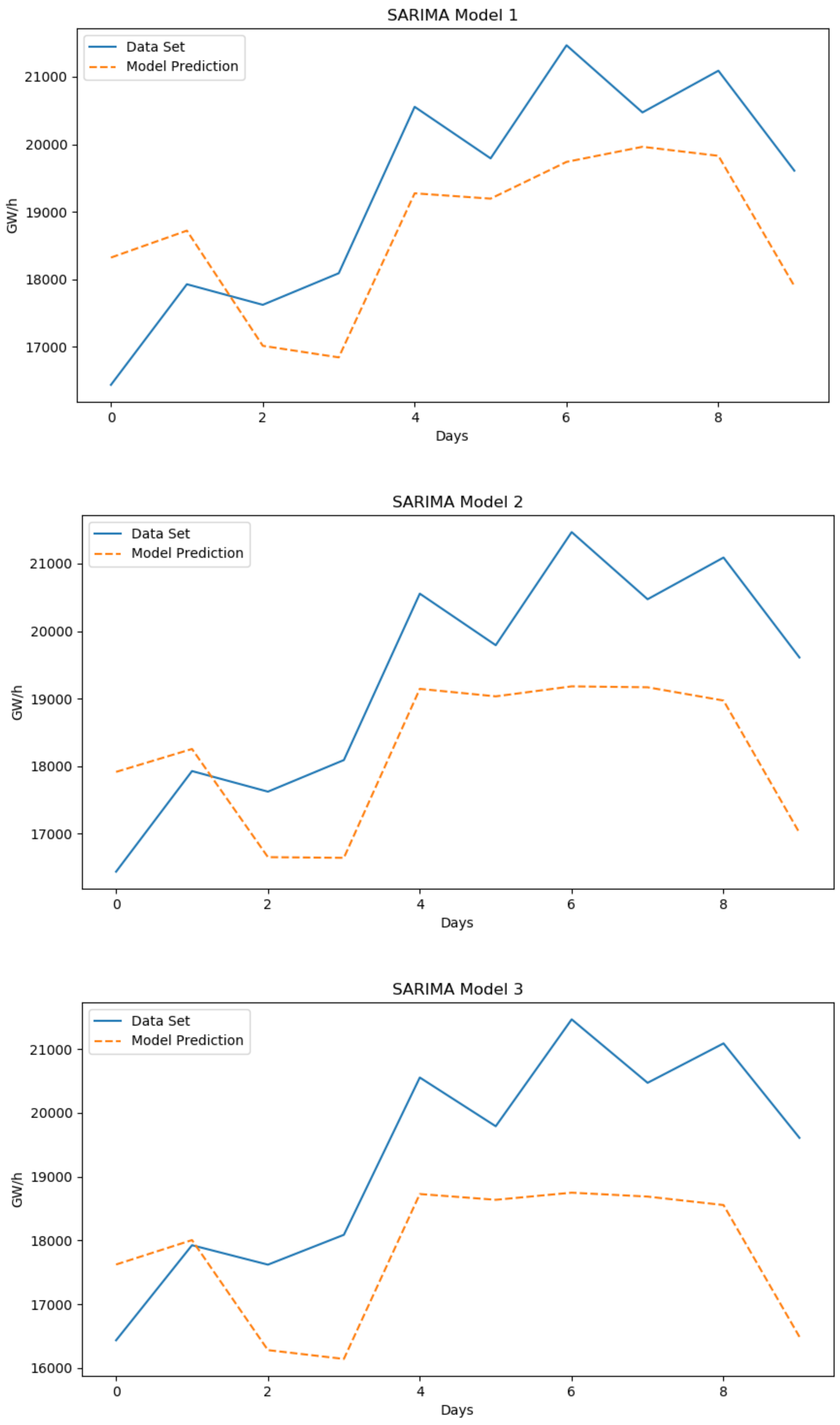
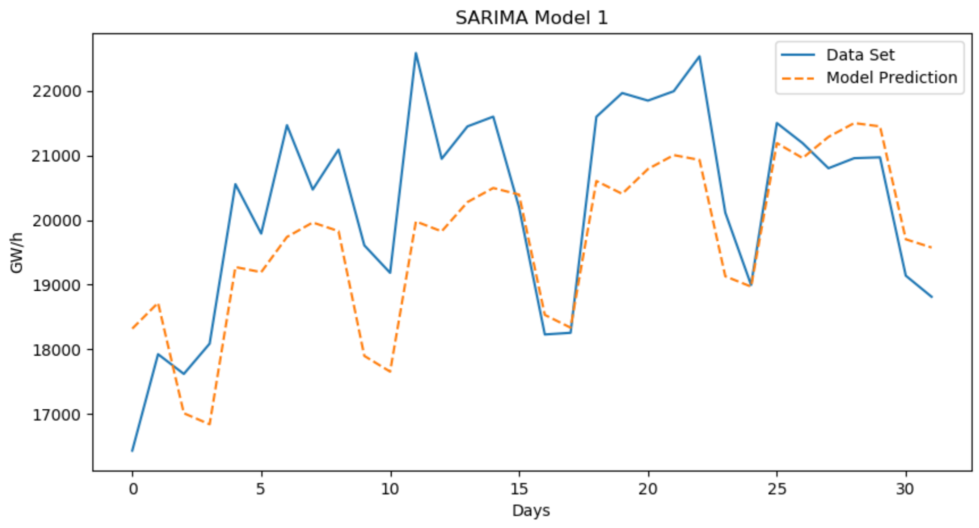
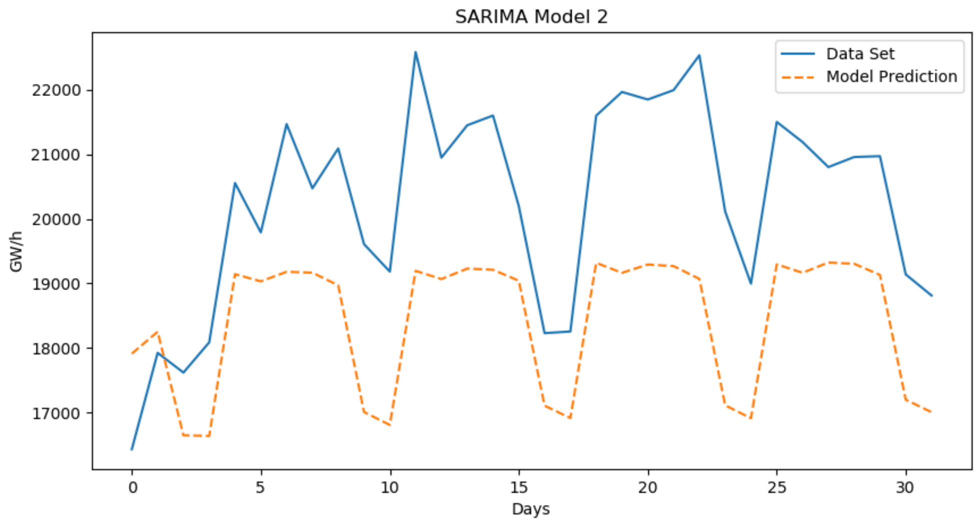


Figure 5.6: graphs for the 10 day forecasts





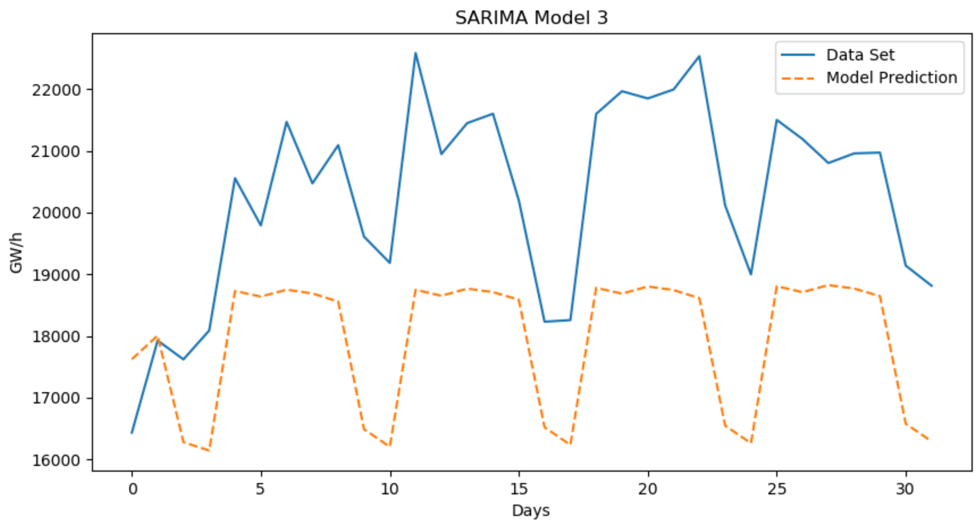


Figure 5.7: graphs for the 1 month forecasts

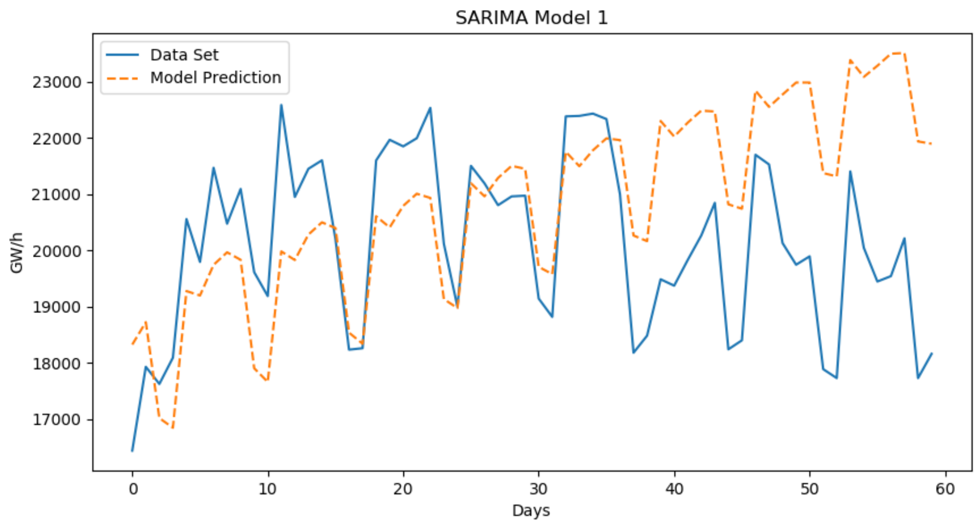


Figure 5.8: The two-month forecast with model 1

We do run into problems when attempting long term forecasting with these models. Figure 5.8 shows the two-month forecast with the first model. We can see that it does not follow the seasonality, but rather keeps on increasing and all of the three models follow the same pattern. This is due to the last seasonal parameter for the SARIMA model defines the number of steps in the seasonality. Since this seasonality is yearly, the last parameter is 365 however, since we used a normal laptop for the experimentation, the laptop that we had available was not able to store the amount of data required for these calculations as it was not possible with 8 gb of ram. For longer forecasts, the common practice is to use monthly or quarterly data.

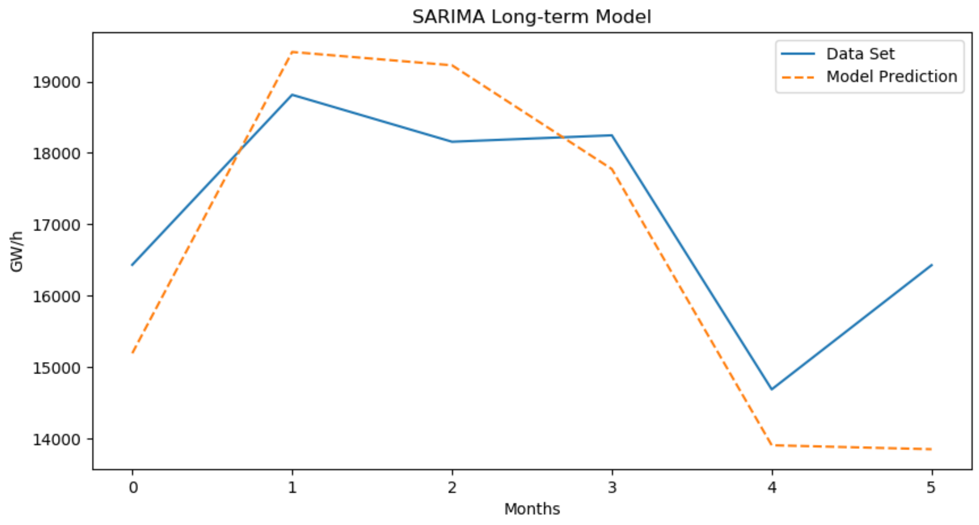
5.3.4 Long-term forecasting

For the long term forecasting, we will have to build another SARIMA model due to the SARIMA seasonal parameters lack some scalability. In order to handle the mid-term to long term forecasting, we will use the monthly values for forecasting instead of daily values like we had in the short-term forecasts. We have the same split as the short-term models for the training and testing data. We have chosen to use the first day in every month as our time-step and will have to do another grid search for these parameters. The grid search showed that the $SARIMA(6, 0, 6)(1, 0, 0, 12)$ with a constant trend to be the best fit for the long term forecasting. The other models that were found were very far off in regards to RMSE, therefore we only need to fit this model for the forecasting. The table 5.5 shows the result of the forecast.

Metric	1 Month	2 Months	6 Months	1 Year
MAE	1237.615	918.440	1122.637	894.820
RMSE	1237.615	972.319	1322.890	1099.754
MAPE	7.530%	5.357%	6.698%	5.604%

Table 5.5: Results from the mid to long-term forecasting with the SARIMA models

One interesting part regarding the results in table 5.5 is comparing the results for the short-term forecasts in table 5.4 regarding the 1 Month forecast, the short-term have higher accuracy than the long-term model. This is a clear indication that for our data set, the mid-term forecasting seems to be better, as the short-term forecasts and long term forecasts have lower accuracy than mid-term forecasts. However, from 2 months and forward, the long term model is way more accurate in regards to all three metrics for forecasting accuracy. Since the long term model can calculate the whole seasonality period, it follows the curve rather than having a steadily increasing trend.



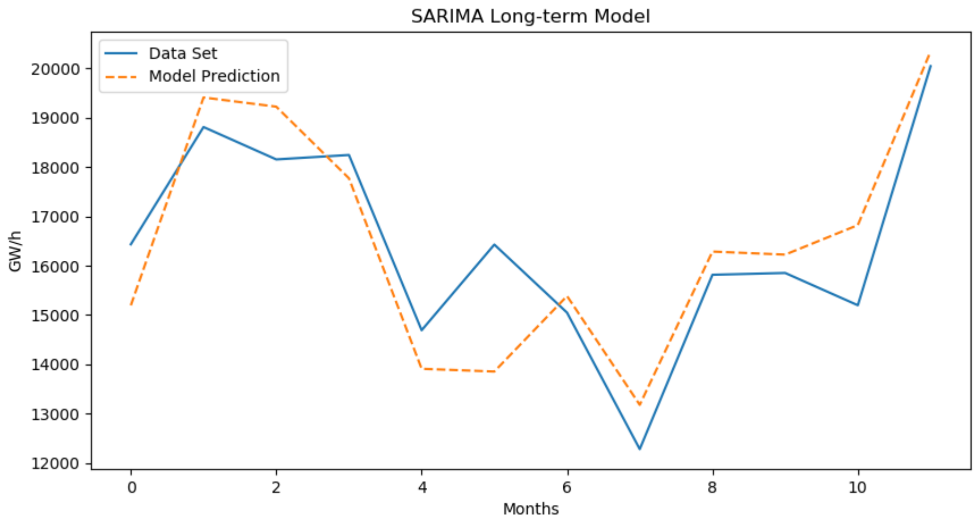


Figure 5.9: The long-term forecast graphs of the SARIMA model

5.3.5 Evaluation of the forecasting results

In regards to our results in table 5.4 and 5.5 and comparing with the articles and reports in the related work section, our results are bad in terms of prediction accuracy. Hulisani Matsila et al [16] used SARIMA for 14 days forecast with a MAPE score of 3.9% using daily data and a smaller data set. Jianguang Deng et al [18] compared the use of SARIMA and their own suggested multiplicative decomposition model. Their SARIMA model had an overall forecast error of 3,4% for their short-term forecasting. Gordon Reikard [14] showed an error of 0,4% in his forecasts with the SARIMA model. The data was stored hourly and the focus of the report was a comparison of different forecasting models for short-term forecasting.

The results in our models have performed have a higher error than expected as our overall error for our short-term forecasts 7,1% for the first model, which had the best average error in regards to our short-term forecasts. Our long term forecasts seem more promising as the average error for those forecasts is 6,3%. This is due to the model only having to account for one seasonality, instead of both weekly and yearly seasonality.

5.4 Neural Network and machine learning

Since machine learning is the state of the art technique and its popularity is steadily increasing, we have selected two different machine learning methods to implement and compare with the SARIMA models above. The first technique that we will use is ANN (Artificial Neural Network). A neural network is a machine learning methodology that is modeled after the brain in the sense that you have neurons (or perceptrons as they are called for machine learning) that do simple signal processing and sending to other neurons. The learning is done by having the network analyze training data and examples that are labeled. A common example found in the literature for ANN is image recognition [27] where you provide the network with images of objects, for example, a cat, and the network will analyze these images and find common patterns for these images and will learn to categorize these images [27].

The neural networks consist of multiple layers. There is the input layer where you send in your actual data, then there are multiple hidden layers of neurons that all communicate with each other and attempt to solve the issue. Each node decides what they want to send to the next layer, and so the passing of data and information will continue until the reached the output layer. This means that the data will be altered for each step and measured in regards to a provided measurement as the ANN will not be able to measure the output without a provided metric of measurement. In our case, we use the RMSE as a metric for evaluating the results as we found this after testing other metrics, to be the metric with the smallest error in regards to our data set. There is no limit on how many hidden layers or neurons a neural network can have, but the general principle is that more layers and neurons give a more precise answer, but there will be a trade-off in performance as the number of calculations and operations will increase. It is hard to classify neural networks since the number of layers and neurons is arbitrary, therefore they are classified based on their depth (the number of layers in the network). Figure 5.10 shows an overview of a neural network and its layers.

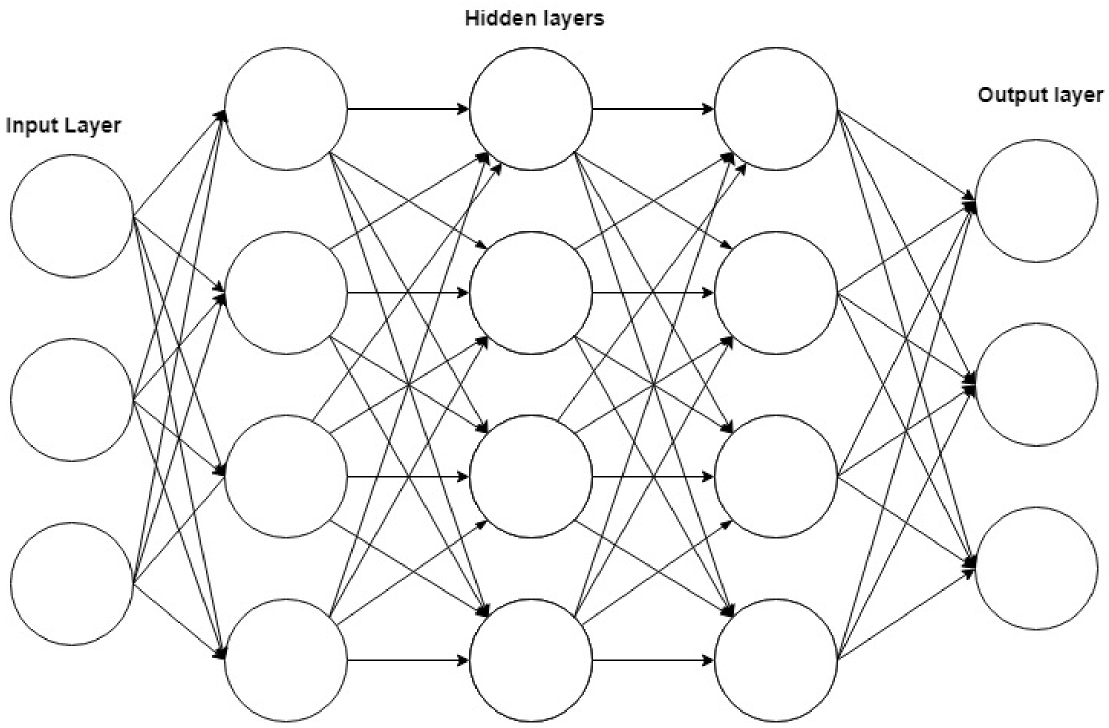


Figure 5.10: Example of the neural network model and its layers

5.4.1 Training the neural networks

As we did previously, we split the data into training data and testing data. Since the time span of this data set stretches over 5 years, we have the first 4 years as the training data and the last year as the testing data. We tried with different amount of layers and neurons, but we stuck with a model where our neural network has a total of 6 layers as there were little to no difference between them. We have one input layer with 100 neurons, 4 layers with 100 neurons each and we have an output layer with one neuron. The one neuron in the output layer was due to the implementation as the implementation required 1 node output layer due to the output data being one-dimensional. There may have been a greater difference if we did more extensive testing however, it was not possible to do within the time frame of this project. To successfully implement an ANN model, we first need to have a strategy for teaching the network its task. The choice of strategy is greatly dependent on the data you are using. An important attribute to consider for the choice of strategy is whether the data is labeled or not. The label in this sense is if the data has some named attribute, for example, a date or for the previous example, the pictures have the name of the objects they are containing such as cat etc. This helps the network establish a common nominator for the data and the network can then categorize the data with regards to the label. Our data set has date labeled for every data, which means that our data set is labeled.

Now that the label has been established, we need to select the appropriate learning strategy for this model. There are multiple teaching strategies for teaching the machine and these are the most commonly used [28].

- **Supervised learning.** This is by far the easiest learning strategy since this learning strategy requires an already labeled data set. The network just goes through the data set and will be modified to get the expected result.

- **Unsupervised learning.** This is used when the data set has not been labeled. Then

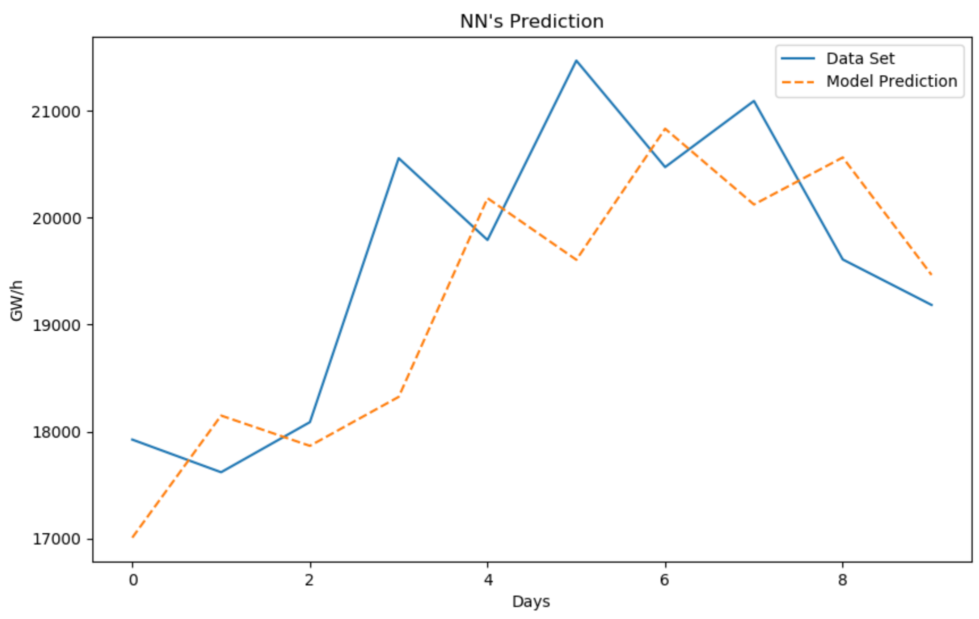


Figure 5.11: 10 step forecast with the Neural Network

the network will analyze the data set and also a weight method which will tell the network how far off the target value it was. The weight will then be measured and the network will then use this data to adapt in order to get a more accurate result.

- **Reinforced learning.** This strategy includes having methods and algorithms to reinforce good results and punish bad results which force the network to learn over time.

In regards to our data set, we can use the supervised learning strategy since our data set is already labeled. This means that we do not have to create a weight or other artificial measurements to teach the network what to do, instead, we can just feed the network the entire training data.

5.4.2 Forecasting with the Neural Network

Since we did not need to build multiple models, we decided to not split this section into subsections, as all the results fit in one table and there are no changes to the network or model between short and long term forecasting. To get an accurate representation between models, we are using the same amount of steps for the forecasting as the SARIMA models. Figure 5.11 shows a forecast for the first 10 steps. Here we see that we have some deviation from the actual data. The network has decided to use the previous known day as the starting point for their forecast.

We can see that the values are slightly off the actual values. The later forecasts are however closer to the target. Figure 5.12 shows the 2-month forecast where we can see that the predicted values are very similar to the predicted values. After the 10 days forecast, we have a stable error of roughly 5.3% however, this changes in the 1-year forecast. As you can see in Figure 5.14, the values differ a lot during the middle of the forecast. This is at the end of the summer season, but it seems we had a warmer season than anticipated, or some other factor affected the power usage. Therefore the power usage went way below expected values and we see a clear change.

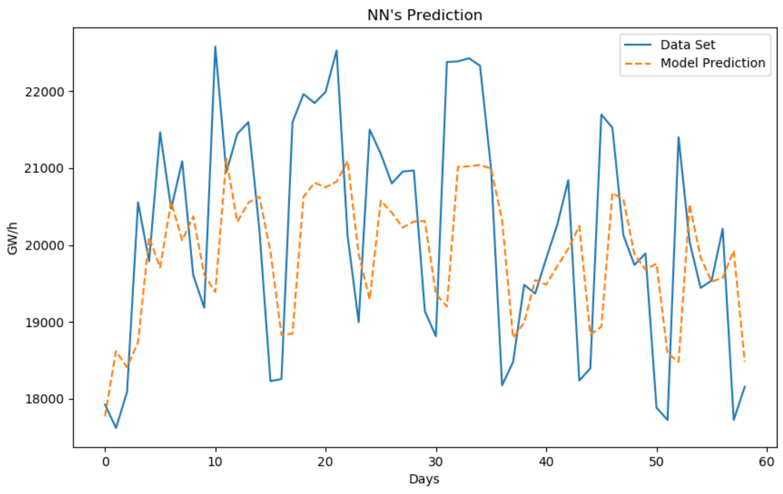


Figure 5.12: 2 Months Neural Network forecast

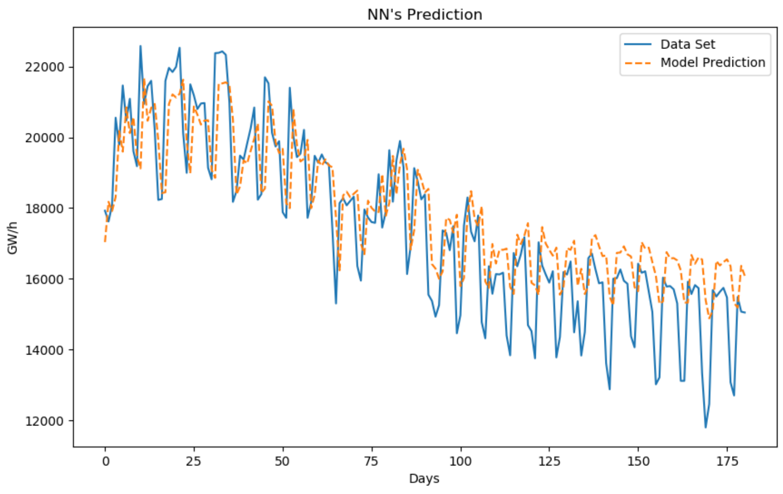


Figure 5.13: 6 Months Neural Network forecast

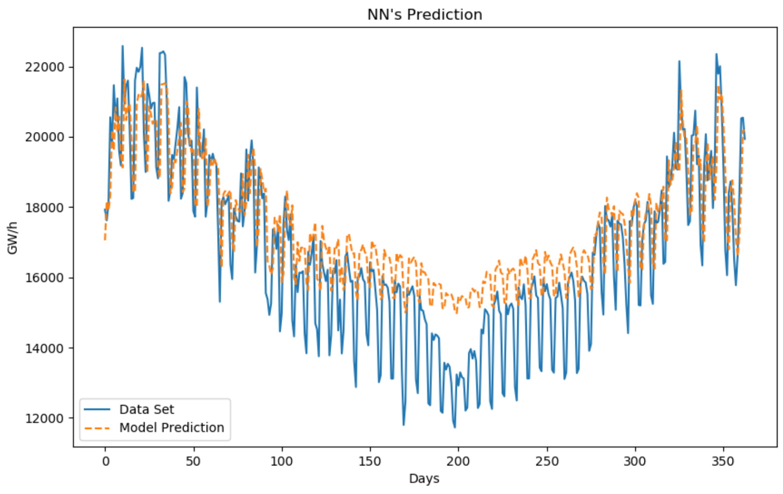


Figure 5.14: 1 Year Neural Network forecast

5.4.3 LSTM (Long Short-Term Memory) Model

As the second method for testing machine learning solutions, we chose to set up a LSTM (Long short-term memory) model, which is a recurrent neural network [28]. Like many of the neural network ideas, they have been around since the ’80s but haven’t really been popular until a few years ago. This model is a good candidate for forecasting and will be a good choice to test out the capabilities of the recurrent neural networks. The benefit of a RNN is that they have an internal memory which means that they can remember properties of the input they just received, which makes it easier predicting what’s coming. The difference between a recurrent neural network and a normal "feed-forward" network is that the RNN instead of sending the data to the next layer can keep the data and process it more times. Since it also has a memory, it keeps the output and sends it through the network again for even more evaluation and analysis. The recurrent neural network also has the ability for backward propagation, which means that the data can travel backward in layers, analyze which derivatives that affected the output and measuring their weight. In short, it means that the RNN tweaks the weights of the neurons while training. Figure 5.15 shows a basic example of a recurrent neural network.

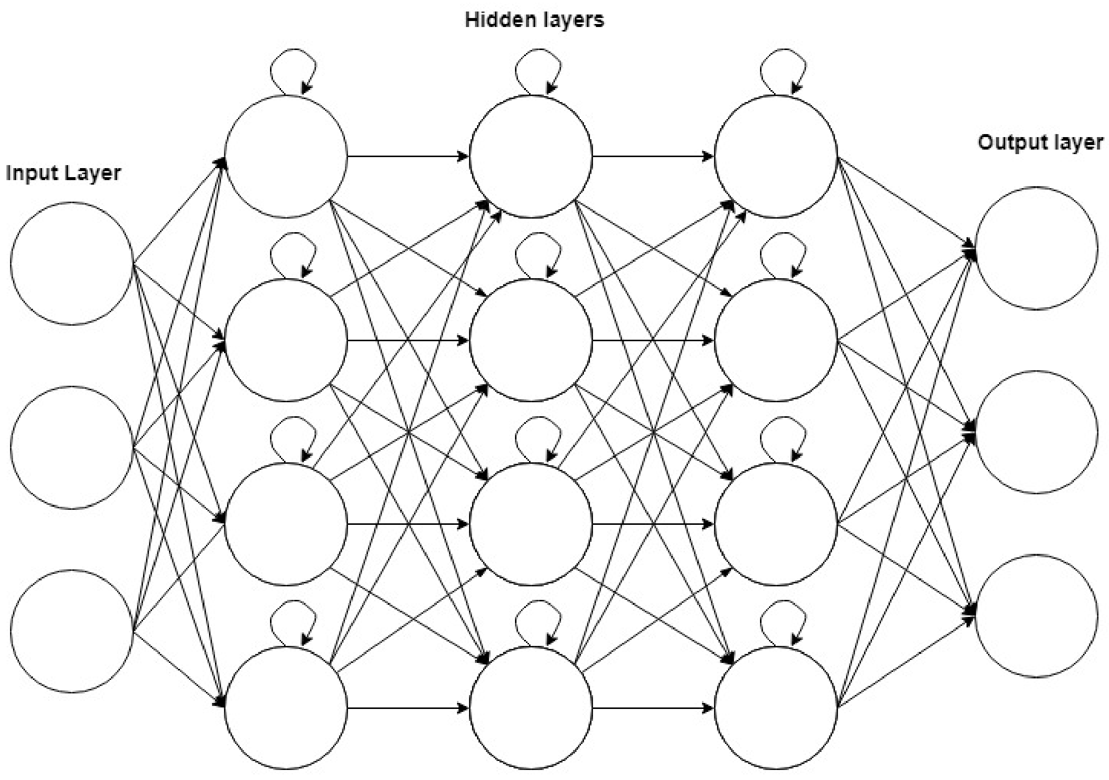


Figure 5.15: An example image of a Recurrent Neural Network

The LSTM is an extension of the traditional neural network. It uses the same methodology, except that it has its own special layer. One major difference between a traditional neural network and LSTM is that the LSTM has an extended memory and is well suited for processes that have long lags between or where the learning is slow. LSTM can be seen as an additional layer to the old recurrent neural network and allows for the network to further store their memory, and the LSTM has the access to read, write and delete the information stored. The LSTM won’t store everything instead, there is a "gatekeeper" that measures if the data is worth storing or not. This is done by asserting which level of importance this assigns to the information. The assigning of importance is done by adding weights to the information, which the algorithm then learns. This means that the computer learns which information is important over time.

5.4.4 Forecasting with LSTM model

As we can see from the results in table 5.6 and compare them to the table 5.4, we can see that the third SARIMA model still has the lowest error with a 7,23% and the closest is the NN with 7,5%. After the one day forecast, the LSTM outperforms all of the SARIMA models in regards to short-term having a steady forecasting error of 6,15 - 5,2 %. The 1-month forecast for the first SARIMA model has a lower error than the LSTM by having a 4,5% to the LSTM's 5,2%. Otherwise, the LSTM has the lowest error of all the methods for short-term forecasting.

For the long term forecasting, we can also see that the LSTM forecast is far superior to the traditional neural network until we get the 1-year forecast, where the long-term SARIMA model has a lower error of 5,6% instead of the LSTM 6,9%. We do get a big drop in accuracy in long term forecasts. If we compare Figure 5.18 with Figure 5.14 however, we can see that both models fail to predict the values from day 150 to 225 correctly. One explanation could be that we had a warmer summer than predicted, or we had some external factors that affected the power usage. Referring to table 5.6 we can see that the LSTM model was the most accurate for this data set.

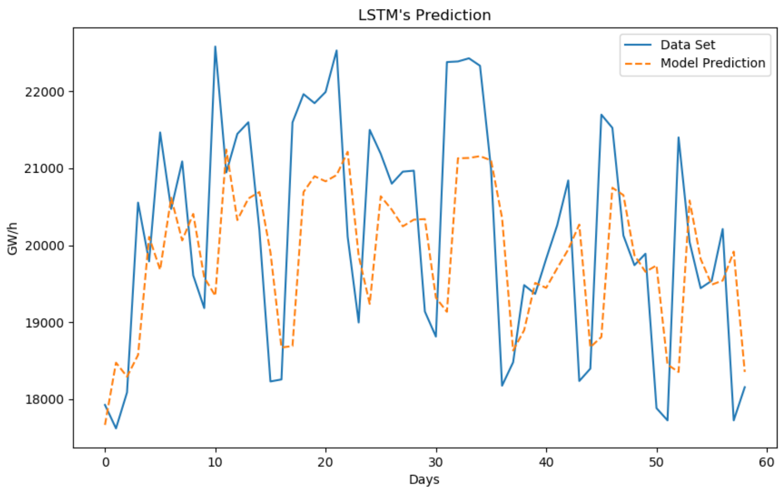


Figure 5.16: 2 Months LSTM model forecast

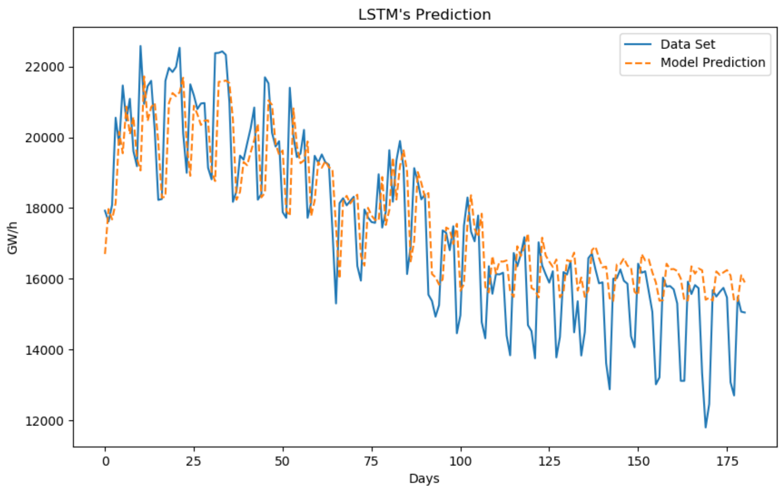


Figure 5.17: 6 Months LSTM model forecast

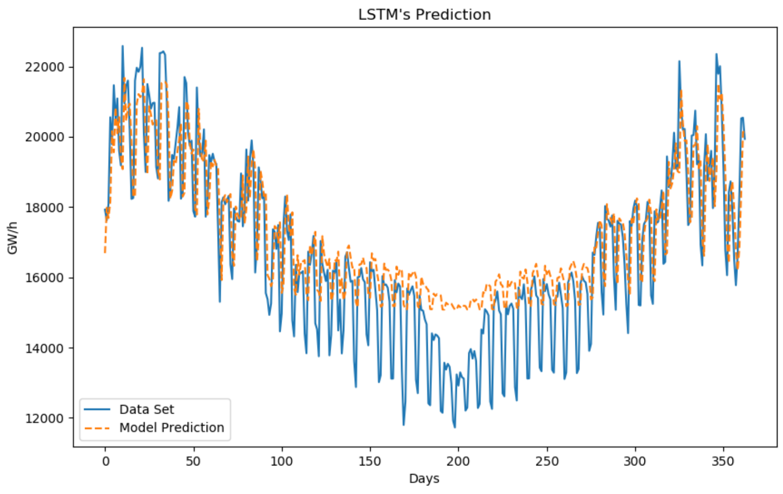


Figure 5.18: 1 Year LSTM model forecast

Model	Metric	1 Day	5 Days	10 Days	1 Month	2 Months	6 Months	1 Year
NN	MAE	**1233.550**	1193.600	1150.640	1080.518	1083.199	1033.472	1233.102
NN	RMSE	**1233.550**	1424.965	1308.409	1439.743	1511.533	1432.712	1547.269
NN	MAPE	**7.505%**	6.433%	5.919%	5.338%	5.401%	5.873%	7.991%
LSTM	MAE	1309.884	**1139.250**	**1040.869**	**1068.268**	**1067.785**	**1014.515**	**1092.685**
LSTM	RMSE	1309.884	**1356.199**	**1237.330**	**1362.212**	**1422.452**	**1342.837**	**1405.800**
LSTM	MAPE	7.970%	**6.149%**	**5.345%**	**5.206%**	**5.250%**	**5.825%**	**6.978%**

Table 5.6: Results from the LSTM and NN forecasts

5.4.5 Evaluation of forecasting results

Even though our forecasting results are better than our SARIMA results, compared to other studies and comparisons in the field, our results have a higher error than expected. Gordon Reikard [14] had a surprisingly low error as the presented neural network model had an error of 0,3-0,46% for the short-term prediction.

The implementations we made and the results we got from our implementations were less accurate than anticipated. The average error for the neural network was 6,343 % and the error for the LSTM average error was 6,1%. The reason why the LSTM outperforms the neural network is very likely due to the recurrent nature of the LSTM. The neural network uses a feed-forward method while the LSTM iterates over itself, both with the data set and also with the predicted results. This allows the LSTM to adjust to the errors in the first prediction and get a more accurate second prediction.

5.5 Discussion

In the previous sections, we analyzed our data set for properties needed to decide which models were eligible for forecasting. We implemented both the traditional SARIMA method as well as two of the state of the art machine learning techniques, neural network and a LSTM. After the analysis, we divided the training and testing data according to best practice [6] and used the training data for fitting the SARIMA model, training the neural network and the LSTM model. These models were fitted using a grid search which we implemented after the Box-Jenkins method proved to be inapplicable on our data set. Even after multiple levels of differencing, our ACF plots did not match a behaviour that fits the Box-Jenkins method. After the training and fitting of models, we set certain time intervals for our models to forecast. We decided to have 1,5,10 and 30 days as our short-term forecasts and have 2 months, 6 months and one year as our long-term forecasts. We predicted these intervals and measured their accuracy by comparing the predicted value to the real value.

In both our SARIMA models as well as the Machine learning models, we found our results to be not as good as anticipated. The reports mentioned previously in this report found greater success in their forecasting accuracy, often having 0,5-3% error, contra our LSTM implementation with 6,1%, neural network with 6,3% and our SARIMA models with 7,1% for the short-term forecasts and 6,3% for the long-term forecasts.

5.6 Implementation

For the experimenting and testing suite, we programmed these in python due to python having good documentation and there are multiple good libraries in python for math and statistical data handling. The libraries we used for the implementation are:

- Numpy. This is one of the most common math libraries. It is also really good for array handling and collections.
- Statsmodels. This is a great library with many forecasting models and other tools for statistical data handling.
- Pandas. This library has a lot of functionality for both series, reading external csv files and data handling in terms of series and other files. It is one of the more commonly used data analysis libraries.
- Matplotlib. It handles the plotting of data and has been used for all the plotted figures in the implementation.
- Keras. Keras is the base library used for the machine and deep learning methods.

To make sure what kind of data set or series we are dealing with, we have decided to program a few tools in Python to plot the series, check the ACF and PACF as well as do the root test for the series. This is to test if the series is stationary and also to find a fitting ARMA/ARIMA model for this series. For the SARIMA model forecasting, we used the Statsmodels library as they had an already implemented SARIMA method, and we fit the model to our data set. Numpy and Pandas handled the series and converted it to a fitting format for the SARIMA method. We also used the Matplotlib to plot the forecasted values and the actual values to get a graphical overview of the prediction.

For our machine learning implementations, we found a python library called Keras which is a high-level neural network and a deep learning library. The library was the base for the neural network and by using this library we set up a neural network and a LSTM model, which is a recurrent neural network implementation. Both of these models are created using the Keras "sequential" model with 1 input layer of 100 neurons, then 4 hidden layers of neurons, then lastly it contains a 1 node output layer. We used an optimized called "SGD" as the implementation required an optimizer to compare data, and the measurement for evaluating the data is the RMSE as the optimizer could only use one error metric for optimization. We tried different error metrics such as the MAE and MAPE, but we found this one to be the best one in regards to our data set. The layers are so-called "dense" layers.

The implementation can be found at:
https://github.com/elaktomte/Degree-Project-Final.

6 Conclusion and future work

We analyzed our data set and experimented with the traditional SARIMA model and compared it to a neural network and a long short-term memory model for time series forecasting. From the results gathered, in our case and for this data set, the LSTM model had the best prediction accuracy of the tested methods and models. Our results were not as good as anticipated, but from the results we gathered, we could see that the models we implemented worked best for mid-term forecasting (10-30 steps) on this data set. For future work, we should attempt to refine our models and explore other techniques for the fitting of models. We tried two techniques for the fitting of our models, but there are other methods worth exploring such as the Box-Ljung[6] method or the Akaike information criterion (AIC)[6].

Due to the limited time for this degree project, we did not have time to test more than these 3 methods. There could have been greater accuracy using different methods or extensions to the SARIMA model or finding a better method for fitting the model. The Holt-Winter method [21], also known as triple exponential smoothing, was not included in this report but is worth testing for future work. Also due to time constraints, we did not experiment as much with the machine learning models as we wanted. It could be possible to obtain better accuracy using different layers or models. Another future direction is combining the SARIMA and the machine learning models, as many of the reports combine these two methodologies to gain better accuracy than the individual models[23, 10, 19]. Therefore there is a clear argument for combining and creating hybrid models and will be left for future work.

For future work, we shall consider using models or extensions that enable fitting ARIMA models with multiple seasonalities. With regard to this, there is a method called Fourier trends[29] that can be added to ARIMA models that handles multiple seasonalities, which should be tested on this data set.

References

[1] "Swedish power statistic," https://www.svk.se/en/national-grid/the-control-room/, accessed: 2019-04-08.

[2] U.S Department of energy. (2016) Maintaining reliability in the modern power system. [Online]. Available: https://www.energy.gov/sites/prod/files/2017/01/f34/Maintaining%20Reliability%20in%20the%20Modern%20Power%20System.pdf

[3] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 3rd ed. Springer, 2016.

[4] J. W. Taylor, "Triple seasonal methods for short-term electricity demand forecasting." *European journal of operational research*, vol. 204, pp. 139–152, 2010.

[5] M. C. M. L. J. Soares, "Modeling and forecasting short-term electricity load: a comparison of methods with an application to brazilian data." *International Journal of Forecasting*, vol. 24, pp. 630–644, 2008.

[6] G. C. R. George E. P. Box, Gwilym M. Jenkins and G. M. Ljung, *Time Series Analysis : Forecasting and Control*, 6th ed. John Wiley Sons, Incorporated, 2016.

[7] NIST/SEMATECH. (2012) Nist/sematech e-handbook of statistical methods. [Online]. Available: https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc446.htm

[8] D. Asteriou and S. G. Hall, *ARIMA Models and the Box–Jenkins Methodology*, 2nd ed. Palgrave MacMillan, 2011.

[9] "Neural networks," http://neuralnetworksanddeeplearning.com/chap4.html, accessed: 2019-04-12.

[10] M. Khashei and M. Bijari, "A novel hybridization of artificial neural networks and arima models for time series forecasting," *Expert systems with applications*, vol. 39, pp. 4344–4357, 2012.

[11] B. Z. . Y. Wei, "Carbon price forecasting with a novel hybrid arima and least squares support vector machines methodology," *Omega*, vol. 41, pp. 517–524, 2013.

[12] M. B. Mehdi Khashei and S. R. Hejazi, "Combining seasonal arima models with computational intelligence techniques for time series forecasting," *Soft Computing*, vol. 16, pp. 1091–1105, 2012.

[13] S. F. G. F.Marandi, "Time series forecasting and analysis of municipal solid waste generation in Tehran city," in *2016 12th International Conference on Industrial Engineering (ICIE)*. IEEE, 2016.

[14] G. Reikard, "Predicting solar radiation at high resolutions: A comparison of time series forecasts," *Solar Energy*, vol. 83, pp. 342–349, 2009.

[15] D. Sena and N. K. Nagwani, "Application of time series based prediction model to forecast per capita disposable income," in *IEEE International Advance Computing Conference (IACC)*. IEEE, 2015.

[16] H. Matsila and P. Bokoro, "Load Forecasting Using Statistical Time Series Model in a Medium Voltage Distribution Network," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018.

[17] Z. H. Yunxiang Tian, Qunying Liu and Y. Liao, "Wind speed forecasting based on Time series - Adaptive Kalman filtering algorithm," in *2014 IEEE Far East Forum on Nondestructive Evaluation/Testing*. IEEE, 2014.

[18] J. Deng and P. Jirutitijaroen, "Short-term load forecasting using time series analysis: A case study for Singapore," in *2010 IEEE Conference on Cybernetics and Intelligent Systems*. IEEE, 2010.

[19] J.-J. L. Wei-Min Li, Jian-Wei Liu and X.-R. Wang, "The financial time series forecasting based on proposed ARMA-GRNN model," in *2005 International Conference on Machine Learning and Cybernetics*. IEEE, 2005.

[20] H. Liu and J. Shi, "Applying arma–garch approaches to forecasting short-term electricity prices," *Energy Economics*, vol. 37, pp. 152–166, 2013.

[21] A. Al-Sakkaf and G. Jones, "Comparison of time series models for predicting campylobacteriosis risk in new zealand," *Zoonoses and public health*, vol. 61, pp. 167–174, 2014.

[22] F. D. K. S. R. K. V. M. V. J. W. W. Paninski L, Ahmadian Y, "A new look at state-space models for neural data," *Journal of Computational Neuroscience*, vol. 29, pp. 107–126, 2006.

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

[24] T. R. Zibo Dong, Dazhi Yang and W. M.Walsh, "Short-term solar irradiance forecasting using exponential smoothing state space model," *Energy*, vol. 55, pp. 1104–1113, 2013.

[25] W. A. Dickey, D. A.; Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American Statistical Association*, vol. 74, p. 427–431, 1979.

[26] A. Sorjamaa, "Methodology for long-term prediction of time series," *Neurocomputing*, vol. 10, pp. 1–5, 2007.

[27] D. M. Pelt and J. A. Sethian, "A mixed-scale dense convolutional neural network for image analysis," *Proceedings of the National Academy of Sciences*, vol. 115, no. 2, pp. 254–259, 2018.

[28] Q.-Y. S. C.-K. Huang, Guang-Bin; Zhu, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, p. 489–501, 2006.

[29] G. R. Richards, "Identifying trends in climate," *INTERNATIONAL JOURNAL OF CLIMATOLOGY*, vol. 18, pp. 583–594, 1998.