DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Solar Power Forecasting with Machine Learning Techniques

EMIL ISAKSSON

MIKAEL KARPE CONDE

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Solar Power Forecasting with Machine Learning Techniques

EMIL ISAKSSON
MIKAEL KARPE CONDE

Degree Projects in Mathematical Statistics (30 ECTS credits)
Degree Programme in Industrial Engineering and Management (120 credits)
KTH Royal Institute of Technology year 2018
Supervisor at Vattenfall AB: Pierre-Julien Trombe
Supervisor at KTH: Jimmy Olsson
Examiner at KTH: Jimmy Olsson

TRITA-SCI-GRU 2018:214
MAT-E 2018:34

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Solar Power Forecasting with Machine Learning Techniques

Abstract

The increased competitiveness of solar PV panels as a renewable energy source has increased the number of PV panel installations in recent years. In the meantime, higher availability of data and computational power have enabled machine learning algorithms to perform improved predictions. As the need to predict solar PV energy output is essential for many actors in the energy industry, machine learning and time series models can be employed towards this end. In this study, a comparison of different machine learning techniques and time series models is performed across five different sites in Sweden. We find that employing time series models is a complicated procedure due to the non-stationary energy time series. In contrast, machine learning techniques were more straightforward to implement. In particular, we find that the Artificial Neural Networks and Gradient Boosting Regression Trees perform best on average across all sites.

Keywords: Solar PV Power Prediction, Machine Learning, Time Series, Artificial Intelligence, Renewable Energy Sources, Forecasting, Solar Forecasting

Prediktion av solcellsproduktion med maskininlärningsmetoder

Sammanfattning

Sänkta produktionskostnader och ökad effektivitet har de senaste åren gjort solceller till ett attraktivt alternativ som energikälla. Detta har lett till en stor ökning av dess användning runt om i världen. Parallellt med denna utveckling har större tillgänglighet av data samt datorers förbättrade beräkningskapacitet möjliggjort förbättrade prediktionsresultat för maskininlärningsmetoder. Det finns för många aktörer anledning att intressera sig för prediktion av solcellers energiproduktion och från denna utgångspunkt kan maskininlärningsmetoder samt tidsserieanalys användas. I denna studie jämför vi hur metoder från de båda fälten presterar på fem olika geografiska områden i Sverige. Vi finner att tidsseriemodeller är komplicerade att implementera på grund av solcellernas icke-stationära tidsserier. I kontrast till detta, visar sig maskininlärningstekniker enklare att implementera. Specifikt finner vi att artificiella neurala nätverk och så kallade Gradient Boosting Regression Trees presterar bäst i genomsnitt över de olika geografiska områden.

Acknowledgements

We would first like to thank our supervisor Pierre-Julien Trombe and Vattenfall AB. Without Pierre-Julien's coordination, enthusiasm, and patience, none of this would have been possible. Secondly, we would like to show our gratitude to our supervisor at KTH, Jimmy Olsson, for giving us guidance. Finally, we would both like to thank our families and friends for the support during our five years at KTH.

Contents

List of Figures

List of Tables

List of Acronyms

ANN Artificial Neural Network

AR Autoregressive

ARIMA Autoregressive Integrated Moving Average

CV Cross-Validation

GBRT Gradient Boosting Regression Trees

KNN K-Nearest Neighbor

LR Linear Regression

MA Moving Average

ML Machine Learning

nRMSE Normalized Root-Mean-Square Error

NWP Numerical Weather Prediction

PV Photovoltaic

RES Renewable Energy Source

RMSE Root-Mean-Square Error

Chapter 1

Introduction

1.1 Background

The global shift towards renewable energy sources (RES) has driven the development of photovoltaic (PV) panels. For example, the costs of producing electricity from PV panels have dropped significantly, while simultaneously increasing the energy conversion efficiency. More specifically, the levelized cost of electricity of large-scale PV panels has decreased by 73% between the years of 2010 and 2017 [1]. The decreased cost and increased efficiency have made PV panels a competitive alternative as a RES in many countries [2].

However, since PV panel energy output depend on weather conditions such as cloud cover and solar irradiance, the energy output of the PV panels is unstable. To understand and manage the output variability is of interest for several actors in the energy market. In the short-term (0-5 hours), a transmission system operator is interested in the energy output from PV panels to find the adequate balance for the whole grid, since over- and under-producing electricity often results in penalty fees. On another side of the spectrum, electricity traders are interested in long time horizons, ordinarily, day-ahead forecasts since most electricity is traded on the day-ahead market. Consequently, the profitability of these operations relies on the ability to forecast the fluctuating solar PV panel energy output accurately.

It is likely, as more countries decide to invest more and more in RES, that the use of solar PV panels will continue to increase. This will increase the need for suitable means of forecasting solar PV energy output. While the demand for accurate and efficient forecasts of PV panel energy output is evident, the solution is far from trivial. There are many complications that the current research within the field is handling. One evident nuisance is the inherited variation of weather, which makes accurate weather forecasting challenging.

Parallel to the increased demand of PV power forecasting solutions, the means for forecasting with the help of machine learning (ML) techniques have in recent years gained in popularity relative to traditional time series predictive models. Although ML techniques are nothing new, the improved computational capacity and the higher availability of quality data have made the techniques useful for forecasting. This poses for an interesting area of research when forecasting the solar power output: How do machine learning techniques perform relative to traditional time series forecasting techniques?

1.2 Objective

The main objective is to benchmark different forecasting techniques of solar PV panel energy output. Towards this end, machine learning and time series techniques can be used to dynamically learn the relationship between different weather conditions and the energy output of PV systems.

Four ML techniques are benchmarked to traditional time series methods on PV system data from existing installations. This also required an investigation of feature engineering methodologies, which can be used to increase the overall prediction accuracy.

1.3 Research Question

Based on our introductory discussion, the problem can be summarized with the following research question:

- How do time series and machine learning techniques perform in short-term (0-5 hours) forecasting of solar PV energy output?

1.4 Limitations

Some limitations were done to clarify the scope of the study.

Five established prediction models were chosen beforehand. The models that will be implemented and compared are: Lasso, ARIMA, K-Nearest Neighbors (KNN), Gradient Boosting Regression Trees (GBRT), and Artificial Neural Networks (ANN). These models have been selected based on their tendency to perform well in previous research of energy forecasting.

The focus will be placed in benchmarking ML and time series techniques. Many of the above models are generic and therefore do most of them have a wide range of different model set-ups. The aim is to give a general overview of the relative performance of the methods rather than investigating a specific model in depth.

1.5 Outline

The remaining parts of this thesis will be structured the following way:

- **Chapter 2 - Previous Research**
  Main achievements from previous research based on both time series and machine learning techniques are presented and discussed.
- **Chapter 3 - Mathematical Background**
  The mathematical theory of the statistical models is presented on a technical level.
- **Chapter 4 - Method and Data**
  The data, data processing, feature engineering, and performance metrics are described. Ultimately, the methodology for each specific model is presented.

- **Chapter 5 - Results**
  The empirical results of the study is presented.

- **Chapter 6 - Discussion**
  Analysis and discussion of the results is presented. Strengths and weaknesses of the study along with potential improvements are also outlined.

- **Chapter 7 - Conclusion and Implications**
  The study's main conclusions are drawn along with implications for further research in the field.

Chapter 2

Previous Research

In this chapter, general articles related to energy forecasting is demonstrated. Significant findings for time series and machine learning techniques for solar forecasting is also presented.

2.1 General Research on Solar Power Forecasting

The literature review for this study is primarily based on the literature studies of Inman et al. [3] and Antonanzas et al. [4]. From these articles, a broad understanding of relevant theories related to solar PV panels and various forecasting techniques can be attained.

Kestylev et al. [5] outlines a summary of various techniques employed in the field of solar PV power forecasting. The authors propose that the industry should establish an industry standard. Some useful findings, which are observed in other studies, are for instance substantial improvements for long-term forecasting when using numerical weather predictions (NWP) as input data. Furthermore, modeling future cloud positions with satellite-based data have improved short-term solar PV energy output forecasting. Finally, model accuracy tends to vary depending on the climatic condition of the forecasting location. As of this, a model is likely to perform better in one region than when trained on multiple sites simultaneously. Similarly, as climatic conditions can vary over a yearly cycle, a model may perform better when trained on one weather season rather than several.

Hong et al. [6] provide an outline of various forecasting techniques used for energy forecasting. The study emphasizes that many different time series and ML techniques have been used in various energy contexts. The study is based on a competition for energy forecasting and evaluates the methods used in the competition (Global Energy Forecasting Competition 2014).

The main conclusions are that for load demand forecasting, both machine learning and traditional time series techniques have been widely used. Similarly, for electricity prices, a broad mixture of time series and ML techniques have been employed. The article also provides an outline of techniques used in the competition for windmills and solar PV panels power forecasting. In contrast to load demand and electricity prices, the span of evaluated forecasting methods for windmills and solar PV panels power output is narrower. There has been some use of ML techniques while traditional time series have been used to a great extent. For solar PV panels, the Random Forest algorithm and the Support Vector Machine (SVM) algorithm

were used. [6]

Inman et al. [3] present an extensive review of theory for RES forecasting. The article presents theories from different fields, such as on how to model irradiance, air masses, and clearness indices. The article then provides a theoretical background of commonly used time series models and ML techniques. Furthermore, theories on irradiance and satellite images are presented to provide an understanding of how they can be implemented. The work serves as guidance for both theory and methodologies from a physics, statistical and ML perspective.

2.2 Time Series Techniques

Reikard [7] presents a work regarding short-term horizons (0 to 4 hours), in which a benchmark of various time series models and ML models is provided. The main finding is that for short-term prediction, time series methods such as ARIMA outperformed ML and other time series models. According to Reikard, the main reason to why the model worked better is due to its ability to capture the transitions in irradiance over a 24 hour period. Also, in a short-term perspective, the lagged irradiance value does often reflect the momentary weather conditions to a large extent. The momentary weather condition is unlikely to change dramatically over a short period (in particular during days with stable weather) and does, therefore, provide a relatively accurate proxy of the future short-term energy output. In conclusion, for short-term forecasting, ARIMA and ARIMA with exogenous inputs will be relevant models to investigate as well as considering the use of lagged solar energy outputs.

Madsen et al. [8], compare an autoregressive model (AR), a linear regression (LR) model and an AR model with exogenous input (ARX model) and found a superiority of the ARX model over the other models when forecasting solar power output. In the study, the LR and ARX model included NWP as input data while AR only used historical solar PV panel energy output data. This study also highlights the importance of lagged power value in the short-term, as concluded in the work of Reikard [7].

2.3 Machine Learning Techniques

Coimbra and Pedro [9] conducted a study where ARIMA, ANNs and KNN models were benchmarked and found, in contrast to the findings of Reikard, better accuracy with the ANN than the ARIMA in short-term forecasting. Coimbra and Pedro did, however, predict power output of solar PV panels rather than solar irradiance. In the study, only historical output levels of the panels were used, i.e., no exogenous NWP data was used as input. The authors also managed to attain accuracy improvements for the ANN model when using a genetic algorithm (an optimization algorithm inspired by the natural selection process) as their optimization algorithm. Ultimately, Coimbra and Pedro acknowledge the varying forecasting accuracy in different weather regimes. They suggest a division of data for different weather regimes when modeling. Here, the authors believe that fitting a specific model to a specific weather regime dataset may improve predictive results compared to using one fitted model to a dataset for all different weather regimes.

Other researchers, such as Andrade et al. [2], explored ML techniques and evaluated them in combination with developing features that supposedly should improve the performance. The main approaches used in the study were Principal Component Analysis (PCA) and a feature engineering methodology in combination with a Gradient Boosting Tree model. Furthermore, the authors used different smoothing approaches to create features from their NWP data. Specifically, the authors used a grid of NWP data around the location of the PV installation and computed spatial averages and variances of weather parameters. Besides creating features based on a local grid of points, the authors also computed variances for different predictors for different lead times. When constructing variance features based on lead times, the underlying idea was that the feature would indicate the variability of the weather.

The main conclusion is improved results from using both PCA and feature engineering. According to the authors, there is a twofold knowledge gap for the further research. The first aspect concerns feature management (feature engineering & feature selection) and more concretely regarding how to create meaningful features that improve the forecast. The second aspect concerns the issue of further exploring ML modeling techniques that can be implemented in combination with informative features. Their final comment is that deep learning techniques will be an interesting path to pursue in combination with proper feature management.

Davò et al. [10] used PCA combined with the techniques ANN and Analog Ensemble (AnEn) to predict solar irradiance. With the aim to reduce the dimensionality of the dataset, PCA was used as a feature selection method. The dataset consists of the aggregated daily energy output of the solar radiation, measured over eight years. A comparison between using and not using PCA showed that using PCA in combination with ANN and AnEn enhances the prediction accuracy.

Chen et al. present results on long-term (up to 100 hours) forecasting. The authors employed an ANN as their forecasting method with NWP data as input. The model was sensitive to prediction errors of the NWP input data and also showed a deterioration when forecasting on rainy days in particular. During cloudy and sunny days, the ANN model produced results with MAPEs of around 8 % [11].

Persson et al. used Gradient Boosted Regression Trees (GBRT) to forecast solar energy generation 1-6 hours ahead. The data used was historical power output as well as meteorological features for 42 PV installations in Japan. Concerning RMSE, the GBRT model performed better than the adaptive recursive linear AR time series model, persistence model and climatology model on all forecast horizons. For shorter forecast horizons, it was shown that lagged power output values had a larger predictive ability. Similarly, for longer forecast horizons, the weather forecasts increased in importance. [12]

Shi et al. propose an algorithm for weather classification and SVM to forecast PV power output on 15-minute intervals for the next-coming day. The weather is classified into four classes: clear sky, cloudy day, foggy day, and rainy day. The rationale behind the classification is correlation analysis on the local weather forecasts and the PV power output. The data is thereafter normalized to reduce error and improve accuracy whilst still keeping correlation in the data. Four SVM models with radial basis function kernel are thereafter fitted to the four different weather classes. In conclusion, the result shows a way of using SVM models to train models on specific climatic conditions. [13]

2.4 Conclusions and Insights

Two main patterns that almost every study emphasizes is the importance of lagged energy output values for the short-term, and the increased importance of NWP data input as the forecast horizon increases.

In the short term, we found studies that supported and disapproved that time series models would be superior to ML methods (see [9] and [7]). Therefore, we find it relevant to continue comparing the time series models with ML techniques.

As seen in the study of Pedro and Coimbra [9], the choice of optimization algorithm can positively impact the forecasting accuracy of an ANN and should be taken into consideration when working with ANN. The scope of this study is to provide a general comparative study of different ML techniques; thus we will not investigate the ANN's performance across different versions.

One conclusion that several studies have pointed out is the sensitivity to NWP data errors. One way to try to overcome this for any model would be to use NWP data from several sources. This requires that data of many predictors from several reliable sources are available, which may not be the case. Also, having NWP based on different physical models would probably help to reduce the overall error of the input NWP data even more.

Chapter 3

Mathematical Theory

In this chapter, we will present the mathematical theory behind the employed techniques. For more mathematical depth, we refer to the cited works in this section.

3.1 Time Series

3.1.1 Stationary Processes

An important aspect of time series analysis is the concept of stationary processes. Stationarity refers to the time dependence of probabilistic properties, i.e., if the probabilistic properties of the random variable changes with time. In time series, a stationary series is a requirement when modeling. Formally, a sequence $\{X_t\}_{t\geq 0}$ is said to be strictly stationary if

$$(X_1, ..., X_n) \stackrel{d}{=} (X_{1+k}, ..., X_{n+k}), \tag{3.1}$$

hold for all integers k and $n \geq 1$. In this setting, t, k and n are discrete time indices.

While the requirement of equal distributions is strict, there exists a formulation for weak stationarity. It is said that the sequence is weakly stationary if $\mathrm{E}[(X_t)] = \mu$ and $\mathrm{Cov}(X_{t+k}, X_t) = \gamma_k$ are independent of the time t. Put simply, the autocorrelation and expectation do not vary with time. [14] Many time series are originally not stationary; however various techniques can be employed to make it more stationary. The most common transformations are differencing (seasonal and non-seasonal) and using logarithms.

3.1.2 AR and MA Processes

The auto regressive (AR) process is a regression form where the explanatory variable depends on historical values. Formally, it is given by:

$$X_t - \sum_{i=1}^{q} \phi_i X_{t-i} = Z_t. \tag{3.2}$$

In this equation, q, is chosen depending on the autocorrelation functions and an AR(1)-model correspond to $q = 1$. In this context, $Z_t \sim WN(0, \sigma^2)$.

The Moving Average (MA) process is another regression form where the prediction parameter depends on external shocks rather than historical data. Formally, the process is given by:

$$X_t = \sum_{i=0}^{p} \theta_i Z_{t-i}. \tag{3.3}$$

Here, $Z_i \sim WN(0, \sigma^2)$ and Z_i are independent for all i. Also, for $i = 0$ it holds that $\theta_0 = 1$ [14].

3.1.3 ARMA and ARIMA Processes

An ARMA process combines components of an AR and MA process. An ARMA(p, q) is defined as:

$$X_t - \phi_1 X_{t-1} - \cdots - \phi_p X_{t-p} = Z_t + \theta_1 Z_{t-1} + \cdots + \theta_q Z_{t-q}, \tag{3.4}$$

where $\{Z_i\} \sim WN(0, \sigma^2)$ are independent for all i, and the polynomials $(1 - \theta_1 B^1 - ... - \theta_q B^q)$ and $(1 - \phi_1 B - ... - \phi_p B^p)$ have no common factors. For the above definition, we introduce the operator $B^n Z_t \coloneqq Z_{t-n}$, where n is a positive integer. Analogously, the operator works identically for X_t. The introduction of the operator is convenient when dealing with large values of p and q. The ARMA definition can be simplified with the operators $\phi^p(\cdot)$ and $\theta^q(\cdot)$, which reduces the equation to:

$$\phi^p(B) X_t = \theta^q(B) Z_t. \tag{3.5}$$

In this notation, the operators are just equal to the p and q polynomials form the definition of the ARMA model. Furthermore, it can be showed that the ARMA(p, q) model has a stationary and unique solution if $\phi^p(\cdot) = (1 - \phi_1 B - \cdots - \phi_p B^p) \neq 0$ for all B with $|B| = 1$.

The ARIMA, Autoregressive Integrated Moving Average, is an expansion of the ARMA model where differencing on X_t has been performed. In other words, the same equations as above can be established however valid for $X_t^{'}$ rather than X_t, where $X_t^{'} \coloneqq X_t - X_{t-1}$. In general, any order differencing can be performed, thus the ARIMA(p, d, q) model can be written as, with the same operating abilities of B:

$$\phi^p(B)(1 - B)^d X_t = \theta^q(B) Z_t, \tag{3.6}$$

where $\{Z_t\} \sim WN(0, \sigma^2)$, $\phi^p(B) \neq 0$ and $|B| \leq 1$. When $d = 0$, the ARIMA model is an ARMA model. The differencing in X_t is usually performed to attain stationarity as it reduces the time dependence of the mean. [14]

3.1.4 SARIMA Processes

A SARIMA model, seasonal ARIMA, is an extension in which seasonal terms can be added. This makes sense in those time series that have a clear daily, weekly, monthly or yearly pattern. The pattern can be for any seasonality.

The ARIMA$(p, d, q)(P, Q, D)_s$ is given by:

$$\phi^p(B)\Phi^P(B^s)(1 - B)^d(1 - B^s)^D X_t = \theta^q(B)\Theta^Q(B^s) Z_t, \tag{3.7}$$

where $\Phi^P(B^s) = 1 - \Phi B^s - ... - \Phi^P B^s$ and $\Theta^P(B^s)$ follows analogously. The capitals P, D and Q, have the same properties as the p, d and q, however with a seasonal step. Thus $D = 1$ gives the seasonal difference for some seasonality s: $(1 - B)^{1 \times s} X_t = X_t - X_{t-s}$.

3.1.5 Main Assumptions of ARIMA

In short, the ARIMA is a parametric model that takes on some assumptions. As mentioned, the time series should be stationary or almost stationary when starting to model the ARIMA. Furthermore, once an adequate model is found, the lags should not be autocorrelated, and the components should be statistically significant. Finally, the model's residuals should be independent of each other and follow a WN-process.

3.2 *K*-Nearest Neighbors

The K-nearest neighbor classifier is an algorithm that estimates the conditional distribution of Y given X and then assigns Y to the class with the highest estimated probability. If we consider a test observation x_0, and the K closest points to x_0 from the training data measured in the Euclidean distance, represented by N_0. Then the KNN estimates the conditional empirical distribution for class j as the fraction of the K nearest points that are classified as j:

$$\Pr(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j). \tag{3.8}$$

Lastly, x_0 is classified to class j that corresponds to the largest estimated probability. As can be seen, K will have a large effect on the KNN classifier. For example, if $K = 1$, the decision boundary will overfit the training data, which corresponds to a classifier with low bias but high variance. On the other hand, when K grows the decision boundary becomes close to linear, i.e., low variance but high bias. The choice of K influences the bias-variance trade-off, which makes it important to recognize. A strategy to determine an appropriate value of K is cross-validation, which will be described later. [15]

For regression problems, the KNN is implemented intuitively. In this scenario, KNN identifies the K-nearest neighbors by some distance metric and then assign the average value of those neighbors. The prediction is given by:

$$\hat{Y} = \frac{1}{K} \sum_{i \in N_0} y_i. \tag{3.9}$$

3.2.1 Feature Scaling

To make different Euclidean distances comparable, it is important to standardize the features to the same scale. If we consider the feature X with n observations, then a common standardizing transform for each X_i, $i \in 1, \ldots, n$ is:

$$X_i' = \frac{X_i - \min X}{\max X - \min X}, \quad i \in 1, \ldots, n, \tag{3.10}$$

which transforms all values into the range $[0, 1]$ and assures that all the predictors have the same scale.

3.3 Tree Algorithms

Tree methods are simple models that can handle both regression and classification problems. When used for regression problems, they are called regression trees. Tree methods can be powerful when combining it with different additive methods.

3.3.1 Regression Trees

Consider the training data $\{(X_i, Y_i)\}_{i=1}^{N}, X_i = (X_{i1}, ..., X_{in})$. A regression tree is a method that partitions the output space into M different regions, and asserts a certain value for a point that is classified to one of these regions. More formally, the predictive model is given by:

$$\hat{Y} = F(X) = \sum_{m=1}^{M} c_m I(X \in R_m). \tag{3.11}$$

Here, R_m refers to region m of $m = 1, ..., M$ and c_m is the corresponding value assigned if the indicator function is equal to one and x is test data. In general, c_m is the average value of the region R_m:

$$c_m = \text{average}(Y_i | X_i \in R_m) \text{ for all } X_i \in R_m. \tag{3.12}$$

To construct a full tree, the algorithm seeks to split the output space region into two regions per iteration. In particular, it seeks the splitting by solving the following optimization problem:

$$\min_{j,s} \Big[\min \sum_{X_i \in R_1(j,s)} (Y_i - c_1)^2 + \min \sum_{X_i \in R_2(j,s)} (Y_i - c_2)^2 \Big]. \tag{3.13}$$

This problem formulation corresponds to minimizing the sum of squared residuals for each split of the output space. Here, for each iterative step, $R_1(j, s) = \{X | X_j \leq s\}$ and $R_2(j, s) = \{X | X_j > s\}$ are the two partitioning regions, where c_1 and c_2 are the average output value for the corresponding regions. In concrete, the algorithms find the values of (j, s) that minimizes the above optimization (minimization) problem, i.e., it finds the predictor parameter that minimizes the RSS the most and at which level it should split. This process is then repeated for each sub-region that is created along the M iterations. M is, in this case, the number of different regions in which the whole space is partitioned into. Thus, the technique is supervised, as the total number of regions (node splits) has to be set in advance. [16]

3.3.2 Boosting

Boosting is a stage-wise approach using so-called weak learners. A weak learner can be any statistical model that predicts somewhat better than pure chance. In any statistical learning, the general goal is to find a predictive function $F^*(X)$ such that:

$$F^*(X) = \arg\min_{F(X)} E_{Y,X} L(Y, F(X)), \tag{3.14}$$

where $L(\cdot)$ is a loss function. In regular boosting, $F^*(x)$ is approximated through:

$$F(X) = \sum_{m=0}^{M} \beta_m h(X; \bar{a}_m), \tag{3.15}$$

where $h(X; \bar{a}_m)$ is a weak learner with coefficients $\bar{a}_m = (a_1, ...)$ and M is the total number of weak learners. In the algorithm, one loops through $m = 1, 2, ..., M$ and computes the following:

$$(\beta_m, \bar{a}_m) = \arg\min_{\beta, \bar{a}} \sum_{i=1}^{N} L(Y_i, F_{m-1}(X_i) + \beta_m h(X_i; \bar{a}_m)), \tag{3.16}$$

$$F_m(X) = F_{m-1}(X) + \beta_m h(X; \bar{a}_m). \tag{3.17}$$

Above, i denotes the i:th data point vector and the final prediction is given by $\hat{Y} = F(X)$. From this equation, one can observe that the algorithm iteratively improves the weak learners $F_{m-1}(X)$ against the residuals. A clear example is when applying the squared-error loss function:

$$L(Y_i, F_m(X_i)) = (Y_i - F_{m-1}(X_i) - \beta h(X_i; \bar{a}))^2 = (r_{im} - \beta_m h(X_i; \bar{a}_m))^2. \tag{3.18}$$

Here, it is clear that $r_{im} := Y_i - F_{m-1}(X_i)$ is the residual for the i:th training point for the $m-1$:th predictive model. More intuitively, boosting uses an additive approach in which it tries to adjust for the residuals of the previous model. One can say that the model tries to improve where it tends to perform bad. [17]

3.3.3 Gradient Boosting Regression Trees

In contrast to regular boosting, gradient boosting approximates the above equation in a different manner. To initialize, it is common to find a weak learner $h(X) = h$ where h is a constant, normally by solving $h(X) = \arg\min_h L(X, h)$.

Then, a function depending on X is fitted to the pseudo-residuals:

$$\bar{a}_m = \arg\min_{\bar{a}, \rho} \sum_{i=0}^{N} \Big[- \big[\frac{\delta L(Y_i, F(X_i))}{\delta F(X_i)} \big]_{F(X)=F(X)_{m-1}} - \rho h(X_i; \bar{a}) \Big]^2, \tag{3.19}$$

where the first fraction is the pseudo-residuals for each training point i. The pseudo-residual is the gradient of the loss function, which points in the direction in which the loss-function changes the most. With the minus sign, the psuedo-residual represents the direction for which the loss function is most rapidly decreasing. Thus, the algorithm seeks a weak learner that captures the direction in which the loss function most rapidly decreases. Once $\bar{a}_m$ is determined, the optimal value of β_m of the weak learner is determined by solving:

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} L(Y_i, F(X_i)_{m-1} + \beta h(X_i; \bar{a}_m)). \tag{3.20}$$

Ultimately, $(\beta_m, \bar{a}_m)$ is determined for $m = 1, 2, ..., M$ at each iteration and $F(X)$ is updated by $F_m(X) = F_{m-1}(X) + \beta_m h(X; \bar{a}_m)$ at each iterative step. When referring to gradient boosting, the weak learner $h(x; \bar{a})$ refers to any function that maps X_i to Y_i if $(X_i, Y_i)_{i=1}^N$ is our training set. When referring to gradient boosting regression trees, the week learner function is a regression tree [17]. To make a connection to regular boosting, it can be proven that when using the squared-error as the loss function, the negative gradient is equal to the residual $(Y_i - F(X_i)_{m-1})$. [16]

3.4 Lasso Regression

A Lasso regression is a normal regression but adding a penalizing term in the cost function to avoid overfitting. For linear regression, the cost function minimized is:

$$\min_{\beta_1,...,\beta_p} \sum_{j=1}^{p} [y_j - \beta_0^j + \sum_{i=1}^{n} \beta_i^j X_i^j]^2 + \lambda \sum_{j=1}^{n} |\beta|_j = RSS + \lambda \sum_{j=1}^{n} |\beta|_j \,. \tag{3.21}$$

The added penalty $\lambda \sum_{j=1}^{n} |\beta|_j$ makes the coefficients shrink towards zero, thus the model avoids overfitting. The absolute value forces some of the β's to take the value zero, and thus the Lasso performs a variable selection. When employing the Lasso with a linear regression model, the prediction model is given by:

$$\hat{Y} = \beta_0 + \sum_{i=1}^{n} \beta_i X_i. \tag{3.22}$$

The complicated part of calibrating the Lasso is the choice of λ. This parameter decides how strongly to penalize high values of the β's. One method for choosing an appropriate value of λ is cross-validation. [15]

3.5 Artificial Neural Networks

Artificial neural networks are ML techniques that encircle many mathematical methods. The presented theory in the next subchapters is mainly based on the work of Hastie et al. [16].

3.5.1 The Network and its Functions

A neural network is a two-stage method that can manage both regression and classification problems. Typically, the method consists of a network of so-called neurons connected via mathematical transformation functions.

In general, the network is connected via activation functions and linear combinations of the input data X. The ANN takes X as input, which are summed with a linear function and transformed with the activation function. This process is done through a network of neurons, where each neuron corresponds to either the input data parameter, X_i^n or it's transformation, Z_i^m. Formally, we have:

$$Z_m^1 = \sigma(\alpha_{0m}^1 + [\alpha_m^1]^T X_i), \qquad m = 1, ..., M, \tag{3.23}$$

$$Z_m^j = \sigma(\alpha_{0m}^j + [\alpha_m^j]^T Z^{j-1}), \qquad m = 1, ..., M \quad \text{and} \quad j = 2, ..., J, \tag{3.24}$$

$$T_k = \beta_{0k} + \beta_k^T Z^J, \qquad k = 1, ..., K, \tag{3.25}$$

$$Y_k = F_k(X) = g_k(T), \qquad k = 1, ..., K, \tag{3.26}$$

where $Z^j = (Z_1^j, ..., Z_M^j)$, $a_m^j = (a_{1,m}^j, ..., a_{n,m}^j)$ and $T = (T_1, ..., T_K)$. For a regression problem $K = 1$ is set and for classification problems, K is chosen to be the number of classes. The number n denotes the number of different parameters one data input data vector X_i has. The parameter M represents to the number of neurons in the network per layer, and $j = 1, ..., J$ the number of hidden layers, which will be discussed later. It should be clear that each layer can have different set-up of neurons. In the above notation, $\sigma(v)$ is the activation function and is usually chosen to be the sigmoid function, however other activation functions are employed from time to time. In table 3.1, a list of commonly used activation functions is provided.

Activation Function	Formula
Sigmoid	$f(v) = \frac{1}{1+e^{-v}}$
Hyperbolic Tangent	$f(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$
Rectifier Linear Unit (ReLU)	$f(v) = \max[v, 0]$

Table 3.1: Common activation functions for ANNs

The function $g(u)$ is referred to as the final transformation function of the outputs T. For regression problems, $g(u) = u$ is a common choice, i.e., just summing the last layers produced Z_m^J values.

When referring to the network architecture, there is no limitation on how it can be structured. The common terminology is input layer, hidden layer, and the output layer. The input layer will consist of n nodes, where each node represents one parameter of the input data X_i, i.e., the n:th input node corresponds to X_i^n. The hidden layer(s) consist of M nodes and will first transform the X_i input via the summation function $\alpha_{0m}^1 + \alpha_m^1 X_i$ and then apply the activation function $\sigma(v)$ of this sum, to produce the Z_m values. If the network only consists of one hidden layer, the Z values will be passed to the transformation function via the summation function $T(Z)$ and then produce the final output value via the function $g(T)$. In the case of multiple hidden layers, the Z_m^j-values of a hidden layer (layer j) will be passed on with the summation function $\alpha_{0m}^j + \alpha_m^j X$ and then again transformed with a sigmoid function (or any other activation function) to the layer $j + 1$. This process can be repeated many times until the Z values of the last hidden layer has been computed, then the T function and final transfer function $g(T)$ will be activated to produce the output layer. Using many hidden layers is commonly referred to as Deep Learning since the network is considered to have a depth of layers.

Figure 3.1 demonstrates how a neuron works. The input, either original data X or another layers' values of Z, is added to the summation function and then transformed with the activation function. This gives an output, denoted as Z_m^j in the image. This output is later passed on to other neurons in other layers or to the output layer via the final transfer function.

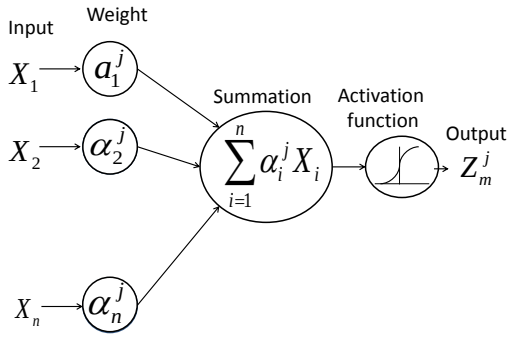


Figure 3.1: A visualization of a neuron

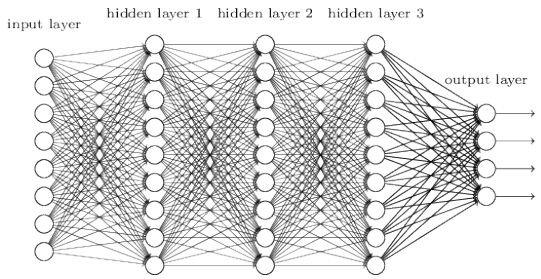


Figure 3.2: Example of a Deep ANN

Figure 3.2 shows an example of an ANN with three hidden layers, each with nine neurons per layer. The lines represent the weights used to pass the sum of the input data to the next neurons/layers, and a circle represents one neuron at each layer. It should be clarified that each hidden layer does not have to have the same amount of nodes. Once a network has been set up, the aim is to find the weights within in the network that minimizes some criteria, which usually translates to minimizing the squared error.

3.5.2 Fitting a Neural Network

Backpropagation & Gradient Descend

For regression problems, the objective is to find the weights that produces a minimized sum-of-squared errors of the full model. In other words, we seek to minimize:

$$R(\theta) = \sum_{i=1}^{N} R_i = \sum_{i=1}^{N} \sum_{k=1}^{K} (Y_{i,k} - F_k(X_i))^2, \tag{3.27}$$

where k is the number of classes and i the number of training inputs. In this setting, X_i is one input data vector, F_k is the output of the ANN and Y_{ik} is the true value for a given class k and input vector X_i.

The general way for minimizing the above expression is by gradient descent, which is also referred to as back-propagation. With the above sum-of-squared errors notation and the specified functions across the ANN, the network has the following derivatives:

$$\frac{\partial R_i}{\partial \beta_{k,m}} = -2(Y_{i,k} - F_k(X_i)) g_k'(\beta_k^T Z_i) Z_{m,i}, \tag{3.28}$$

$$\frac{\partial R_i}{\partial \alpha_{n,m}^1} = -\sum_{k=1}^{K} 2(Y_{i,k} - F_k(X_i)) g_k'(\beta_k^T Z_i) \beta_{m,k} \sigma'([\alpha_m^1]^T X_1) X_{n,i}. \tag{3.29}$$

The above derivatives are valid for a single layer network, however the derivatives for $a_{n,m}^j$ follows analogously. Here, $m = 1, ..., M$ is the number of neurons and i corresponds to the number of data inputs and n to a specific data input parameter of X_i.

With these derivatives, the weights are continuously updated from an initial guess of α and β the following way, where r is a time step and γ (a constant) represents the learning rate:

$$\beta_{m,k}^{(r+1)} = \beta_{m,k}^{r} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{m,k}^{r}}, \tag{3.30}$$

$$\alpha_{n,m}^{j,(r+1)} = \alpha_{n,m}^{j,r} - \gamma \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{n,m}^{j,r}}. \tag{3.31}$$

The derivatives can be rewritten as the back-propagation equations:

$$\frac{\partial R_i}{\partial \beta_{k,m}} = \delta_{ki} Z_{m,i}, \tag{3.32}$$

$$\frac{\partial R_i}{\partial \alpha_{n,m}^{j}} = s_{m,i}^{j} X_{n,i}. \tag{3.33}$$

The optimal solution is computed in a forward and a backward process, in which the α's and β's are updated with a two-stage algorithm. First, an initial guess of the weight values is passed forward to the network. The network computes the prediction values of Y, from which the RSS is computed. Then, the partial derivatives or 'errors' of the model at each node are computed, and the backward stages update the α's and β's. As the gradient point in the direction in which the loss function increases the fastest, the learning rate and the negative scalar assures that the β's and α's are adjusted to reach a global or local minimum of $R(\theta)$. This is done for a fixed number of iterations or until convergence has been reached.

Batch Learning & Online Learning

Common terminology is batch and online learning. The difference lies in the updates of the weights. Batch learning has been described above, where the weights are updated over the sum of all training points. In online learning, the weights are updated for each training observation rather than all. The online learning technique allows the model to handle large datasets and is less computationally expensive [18]. One commonly employed online technique is Stochastic Gradient Descent, which in the above formulas would be dropping the summation over i.

Resilient Propagation

Resilient backpropagation is a type of batch learning for multi-layer ANN. When determining the weight step in each iteration, resilient backpropagation only use the sign of the partial derivative and not the size. The step size of the update of the weights ($\Delta\alpha_{m,j}^{(r)}$) is then determined by:

$$\Delta\alpha_{m,j}^{(r)} = \begin{cases} -\gamma_{m,j}^{(r)}, & \text{if} \quad \frac{\partial E}{\partial \alpha_{m,j}}^{(r)} > 0 \\ +\gamma_{m,j}^{(r)}, & \text{if} \quad \frac{\partial E}{\partial \alpha_{m,j}}^{(r)} < 0 \\ 0, & \text{else.} \end{cases} \tag{3.34}$$

The step is then updated:

$$\gamma_{m,j}^{(r)} = \begin{cases} \eta^+ \gamma_{m,j}^{(r-1)}, & \text{if} \quad \frac{\partial E}{\partial \alpha_{m,j}}^{(r-1)} \cdot \frac{\partial E}{\partial \alpha_{m,j}}^{(t)} > 0 \\ \eta^- \gamma_{m,j}^{(r-1)}, & \text{if} \quad \frac{\partial E}{\partial \alpha_{m,j}}^{(r-1)} \cdot \frac{\partial E}{\partial \alpha_{m,,j}}^{(t)} < 0 \end{cases} \tag{3.35}$$

where $0 < \eta^- < 1 < \eta^+$. In essence, the algorithm looks at the sign of the partial derivative of $\alpha_{m,j}$ and thereafter updates the weight by adding or subtracting the update value $\gamma_{m,j}$. The update values are thereafter adjusted and made larger if the partial derivative did not change signs to speed up convergence, and made smaller if the partial derivative changed signs meaning it skipped a local minimum. The parameters η^+ and η^- are normally set to 1.2 and 0.5. [19]

3.5.3 Issues For Training a Neural Network

Starting Values of Weights

The initial guess of the weights is important to set right. If the weights are set too close to zero, the network can converge to an approximately linear model. It is common to draw the starting weights near zero randomly. If too large values are picked, the model often provides inferior solutions. Usually, it is preferred to have initial weights near zero and both positive and negative values.

The Learning Rate

The learning rate of the back-propagation will influence the final solution. A too high learning rate may give poor results, while a too low learning rate makes the training computationally time-consuming. A common non-advanced method of choosing the learning rate is to pick a relatively large learning rate and then decrease it until a specific diminishing improvement of accuracy can be found within a reasonable computational time.

Overfitting

A common remedy for overfitting is to apply a penalizing term to the minimization problem, similar as described in the Lasso. Normally, $R(\theta) + \lambda J(\theta)$ is set to be the function to minimize via back-propagation, where:

$$J(\theta) = \sum_{km} \beta_{mk}^2 + \sum_j \sum_{nm} \alpha_{nm,\,j}^2, \tag{3.36}$$

and $\lambda \geq 0$ is the parameter that calibrates the magnitude of the penalization. A large value of λ will shrink the values of the weights towards zero. As in the Lasso, cross-validation is used to estimate the optimal choice of λ. The above method is called weight decay.

Input Scaling

The inputs' scaling can affect the scaling of the weights in the output layer and thus affect the final output results. Therefore, it is common to standardize all input data. Normally, one centers the data (zero mean) with the standard deviation of one.

Network Architecture

According to Hastie et al. [16], it is in general better to have too many hidden units or neurons per layer than too few. With too few neurons, the model may not reach the desired flexibility and thus not capture non-linear structures of the data. By using an appropriate penalizing term when minimizing RSS, many of the weights of the excessive neurons will be shrunken to zero and thus have little impact on the model.

Regarding choosing the adequate structure of layers, the process is mainly based on experience according to Hastie et al. As the back-propagation algorithm is considered to be computationally expensive, more layers will make the computational time longer but can, on the other hand, enhance the performance. As of this, when modeling with deep ANNs, back-propagation is not the conventional choice of algorithm.

Multiple minimums

One of the main issues with minimizing RSS is that when F_k is complex, it is not certain that $R(\theta)$ is convex. It is non-convex in the above setting, and thus, it has several local minima. This indicates that the final solution of the model will depend on the initial guess of weights. Therefore, it is common to start with different random starting weights and then choose the starting weight that yields the lowest penalized error. Another alternative is to use bagging, which averages the predictions of different networks from bootstrapped datasets.

Chapter 4

Method and Data

In this chapter, we present the methodology of this thesis. We discuss the original data, how it was obtained and how it was structured. The chapter will continue to discuss the data processing employed and an explicit description of what has been done for the different time series and ML techniques.

Throughout the work, all data handling and mathematical computation have employed R version 3.4.3 [20] and RStudio [21]. The main R package used is *caret* [22], which is a wrapper containing tools for streamlining the creation of predictive models and pre-processing data.

4.1 Raw Data

4.1.1 Power Data

The data on energy output was collected on small-scale solar PV panel installations with a varying peak capacity of around 20kW to 150kW. Furthermore, the energy output data was sampled at 15-minute intervals at five different sites in Sweden for 2-3 years back in time.

Site	Location	Peak capacity (kWp)
1	Uppsala	38.5
2	Åby	131.4
3	Vikingstad	63.24
4	Bollnäs	47.7
5	Varberg	46.84

Table 4.1: Sites

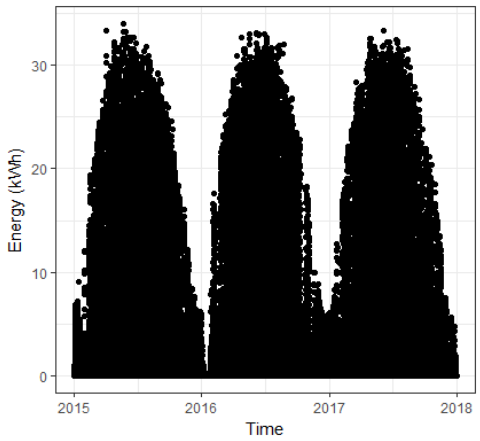


Figure 4.1: Energy output for site 01

4.1.2 Numerical Weather Prediction Data

The weather data was derived from Meteomatics [23] for the five different sites. In table 4.2 one can find the extracted data parameters and a short explanation and the corresponding units. See [23] for a more profound description of meteorological variables and their respective definition.

Variable name	NWP	Unit
total_cloud_cover:p	Total cloud cover	%
wind_speed_10m:ms	Wind speed at 10m	m/s
wind_dir_10m:d	Wind speed direction at 10m	m/s
t_2m:C	Temperature at 2 meters	C
clear_sky_rad:W	Clear sky radiation	W/m^2
clear_sky_energy_1h:J	Accumulated clear sky energy of 1h	J
sfc_pressure:hPa	Surface pressure	hPa
diffuse_rad:W	Instantaneous flux of diffuse radiation	W/m^2
global_rad:W	Instantaneous flux of global radiation	W/m^2
direct_rad:W	Instantaneous flux of direct radiation	W/m^2
effective_cloud_cover:p	Effective cloud cover	%
high_cloud_cover:p	High cloud cover	%
medium_cloud_cover:p	Medium cloud cover	%
low_cloud_cover:p	Low cloud cover	%
precip_1h:mm	Accumulated precipitation of 1h	mm
fresh_snow_1h:cm	Accumulated fresh snow of 1h	cm
global_rad_1h:J	Accumulated energy of global radiation of 1h	J
direct_rad_1h:J	Accumulated energy of direct radiation of 1h	J
diffuse_rad_1h:J	Accumulated energy of diffuse radiation of 1h	J
cape:Jkg	Convective available potential energy	J/kg of air
relative_humidity_2m:p	Relative humidity at 2m	%

Table 4.2: NWPs from Meteomatics

The NWP data accessed has some limitations. One is that the forecasts are produced every 6 hours (6:00, 12:00, 18:00, 24:00) and span six hours forward in time, which means that it is not possible to produce more extended predictions than 6 hours. Another consequence is that that the training set is reduced for longer time horizons as one cannot use unavailable weather forecasts. For example, predicting 11.00 to 13.00 would mean using NWP data from 12.45 to 13.00, which becomes available at 12.00 and thus is unavailable at 11.00 when the prediction is to be made. Moreover, the data was attained via the ZIP codes of the site's location. This makes the data not completely correspondent with the precise coordination of the solar PV panel.

4.1.3 Variability in Data

As can be seen in figures 4.2 and 4.3, there is great variability in the output from a certain PV panel, as well as inaccuracies in the forecast of the total cloud cover. Two relatively similar forecasts can correspond to relatively different actual energy output levels, which emphasizes the uncertainty within the input data itself. Local errors in the input data can in combination with local errors produced by the forecasting techniques produce significant global errors.

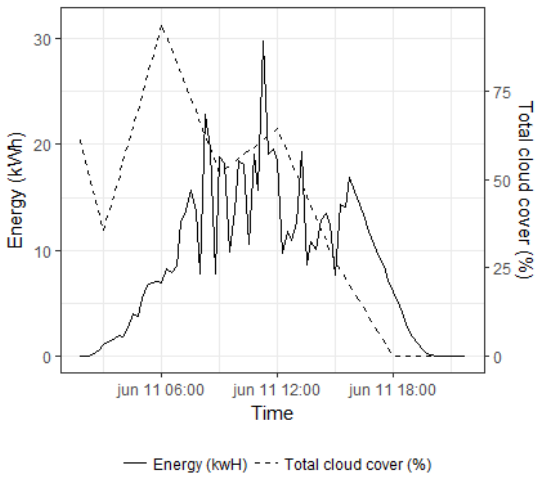


Figure 4.2: A cloudy day

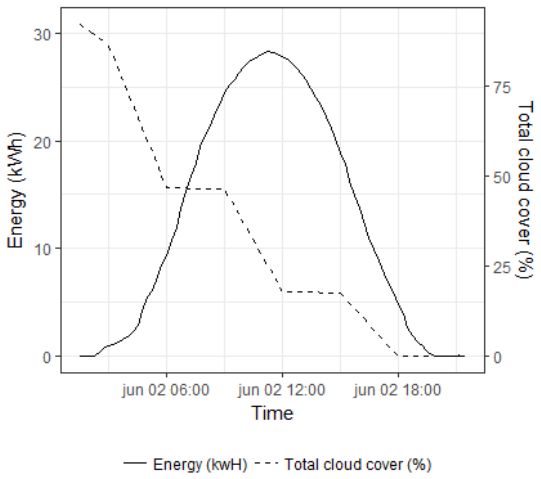


Figure 4.3: A sunny day

4.2 Data Processing

In general, the data have been relatively clean. As a result, the main part of the data processing is related to sorting the data in the right way. The datasets were filtered by computing a lead time for when data was available in real time. The dataset was sorted for this lead time so that no dataset contained unavailable information.

When the data was sorted correctly, a data row consisted of the corresponding NWP data and lagged variables. It should be clarified that when a prediction on any time horizon is performed, it refers to forecasting the energy output of a 15-minute interval in that time horizon and not the energy output of the next time step. Thus, a prediction on one hour refers to the energy output of 15 minutes in one hour and not the energy output the next coming hour.

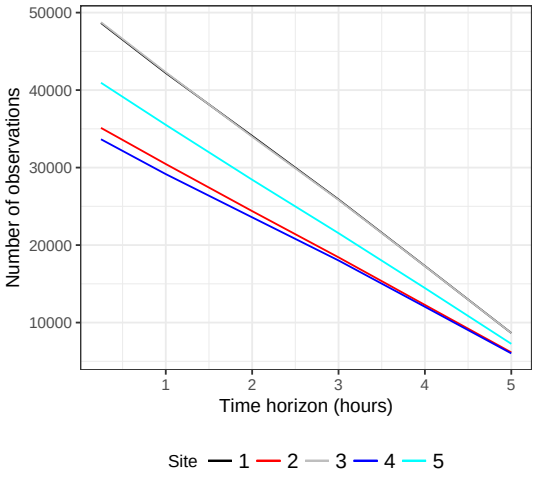


Figure 4.4: Samples for fitting models (Site 1 and 3 overlapping)

Another necessary data elimination criterion was to erase the nighttime data. As the energy output is easy to predict during night time, performance measures would be boosted and give an unfair picture of the models' performance. Also, including nighttime data would mean that the model would have to make a trade-off between fitting daytime data and nighttime data well, where the latter is useless. As the nighttime varies in Sweden on winter and summer seasons, any observation with clear sky radiation of more than 10 W/m^2 was considered as day-time.

4.2.1 Outlier Handling

A visual inspection was done to identify outliers. One example could be negative values of power output or outputs above the installation's peak capacity. As all the data are given at 15-minute intervals, it is likely that a observation will be similar to the 15-minutes ahead and 15-minute lagged observation. Therefore we consider an interpolated value of the previous and next observation to be a good estimation.

4.2.2 Clear Sky Normalization

As seen in previous studies, different normalization techniques have been used to stabilize the data. The reasons for this is to reduce non-stationarity and to erase disadvantageous patterns.

The clear sky normalization was employed by Madsen et al., which converted the non-stationary energy time series into a more stationary time series. As of this, the clear sky normalization was used on the solar PV energy output data to handle its stationarity. The formula is given by:

$$Y_i' = \frac{Y_t}{Y_t^{cl}}, \tag{4.1}$$

where Y_t represents the energy output for the time interval t and Y_t^{cl} represents the energy output for the same time interval on a day with sunshine the whole day and no clouds. In the above formula, the constant term Y_t^{cl} has been observed to vary across the yearly cycle and therefore modeling it is required.

While physical models have proved efficient, they require specific information about the PV installation [24]. In lack of this, the approach used by Bacher et al. [25], requiring only a time series of solar PV panel energy output was employed. The technique is based on fitting a Gaussian kernel in two dimensions (time of day and time of year) in such way that the hyperplane is located near the top-values of the observations. By aligning the Gaussian kernel with the top output values (the values that correspond to clear sky energy output days), an estimated hyperplane representing the clear sky output values throughout the year is produced. Fitting the kernel towards the top-values is achieved with a quantile regression, which can be described as a regular regression, where the regression minimizes the sum of squared residuals while forcing q of the points under the fitted hyperplane. For more details, consult [8] and [25].

4.2.3 Feature Engineering

Lagged Output

For short time horizons, the lagged output value will be of high importance when predicting. Consider the case where the last 15 minutes had a clear sky; then it is likely that the next 15 minutes will have a clear sky as well. For longer forecast horizons, a better guess for the output is the one-day lagged value of the output, i.e., the output at the same time the day before. Therefore we include the one-step, two-step, and one-day lagged output variables as well as the difference between them

in the models, where one-step is the granularity of the data (15 minutes). If y_t is the output at time t then the variables are constructed as such:

$$\texttt{ylag1step} \coloneqq y_{t-1}, \tag{4.2}$$
$$\texttt{ylag2step} \coloneqq y_{t-2}, \tag{4.3}$$
$$\texttt{ylag1day} \coloneqq y_{t-24\cdot 4}, \tag{4.4}$$
$$\texttt{ydelta} \coloneqq y_{t-1} - y_{t-2}. \tag{4.5}$$
$$\tag{4.6}$$

Temporal Weather Grid

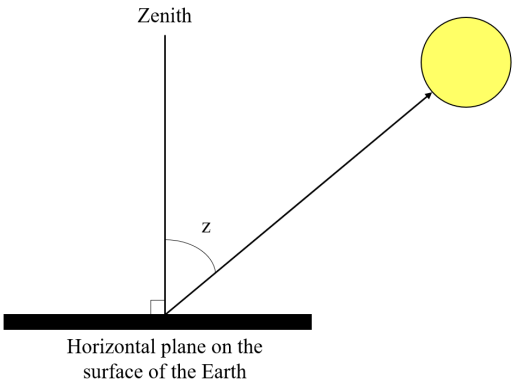
Inspired by the results of Bessa and Andrade [2], a feature engineering process was performed as an attempt to create features that could enhance the performance. A temporal grid and temporal variability for the features are computed. As the NWP variables are forecasted for the ZIP code, and not the exact coordinates of the specific site, one can assume that the 15-minute lagged and lead forecasts have predictive ability. Then, if $\text{NWP}_{i,t}$ is the forecast of NWP variable i at time t, the lagged value is $\text{NWP}_{i,t-1}$ and the lead value $\text{NWP}_{i,t+1}$ where one time-step is 15 minutes. The grid is only performed for $t \pm 1$, not to lose too much data.

Temporal Variability

The variability of the NWP was computed by using the 15-minute lagged value, the 15-minute lead value and the actual value of the certain weather parameter. The idea was to create a variable that captures potential variability in the weather. For example, high variability in effective cloud cover is likely to affect the energy output, making it a potentially important input data. As Bessa and Andrade found in their study, there existed a correlation between the NWP's variability and the energy output variability.

Solar Zenith Angle

Another feature created was the zenith angle, which is the angle that indicates how direct the solar irradiance is coming in with. This was estimated using the function SZA() in the package *RAtmosphere* [26], based on the time-stamps and coordinates of the solar PV panel installation location.



4.2.4 Feature Selection

In this work KNN is the only model that directly required a feature selection procedure in order to reduce the runtime and not overfit.

The feature selection procedure was based on the GBRT algorithm. When the GBRT performs its first split, it tests each input data parameter and computes how

much each parameter minimizes the cost function. From this, the GBRT chooses to split with the feature that reduces the cost function the most, and thus it ranks the features with relative importance. This is performed for each split until the tree is entirely built. By relying on the most important features indicated from the GBRT, we were able to perform a computationally efficient feature selection. Pan et al. [27] showed that there exist GBRT based feature selection algorithms that are efficient, thus relying on GBRT should give a good proxy for the most important features.

4.3 Cross-Validation

Many time series methods and ML techniques require hyperparameter optimization. A common practice is to perform cross-validation (CV) to determine the optimal choice of hyperparameters. One type of cross-validation is the k-fold cross-validation In general, this refers to randomly dividing the set into k folds. The validation principle picks one of the k folds as a test set and the rest of the observations as the training set. The process then changes the test set to another fold while using the rest dataset as training data. This is repeated for all the folds and each k-fold, a performance metric is computed. Once a performance metric is computed for each fold, the average performance is computed. This can be performed for several hyperparameters to determine the value that yields the best performance metric from the cross-validation. A common choice of k is usually between 5 and 10.

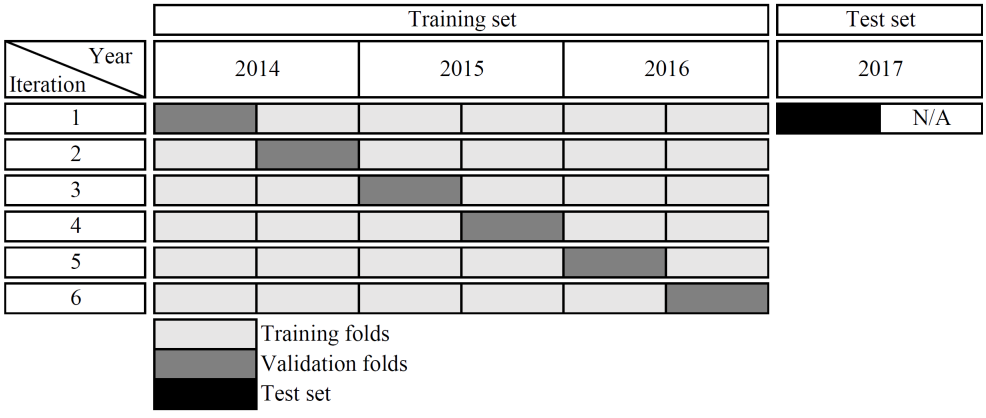


Figure 4.5: Visualization of cross-validation procedure

As pointed out by Hyndman et al. [28] when data is not time-dependent, a regular k-fold CV works well. However, when there exist significant time dependence, a regular CV process may be misguiding. In figure 4.5, an illustration of the performed CV process employed is shown. The figure illustrates the process if the data for a site was from January 2014 to the end of June 2017. The primary assumption is that each half-year fold captures a cycle of seasonal variability, meaning that energy output of January-June and July-December approximately follows the same pattern. Each half-year fold can, therefore, be considered almost independent of one another. The serial correlation that makes time series problematic for regular CV is only maintained within the half-year folds. As of this, the folds cannot be divided randomly but rather in successive order. Ideally, one would want to use a full one-year cycle per fold, however lack of sufficient data limited this.

4.4 Performance Metrics

4.4.1 Root Mean Squared Error

A common performance metric is the Root Mean Squared Error (RMSE), given by:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}}, \tag{4.7}$$

where Y_i corresponds to the true value and $\hat{Y}_i$ is the forecast. As the RMSE sums the squared errors before averaging, any outliers will be given a higher weight.

4.4.2 Normalized RMSE

To fairly compare the RMSE across different sites with various peak capacities (kWp), normalized RMSE (nRMSE) is computed. The nRMSE is given by:

$$\text{nRMSE} = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n \times \max(Y)}}, \tag{4.8}$$

where $\max(Y)$ is the maximum output recorded for a specific site. [4]

4.4.3 Skill Score

It is common to compute a skill score to compare one model to another. A generic definition of a skill score is provided by Murphy [29]:

$$\text{SS}_{\text{model}} = \frac{A_{\text{model}} - A_{\text{reference}}}{A_{\text{perfect model}} - A_{\text{reference}}}. \tag{4.9}$$

In this setting, A refers to any performance metric and the reference model is usually chosen to be the persistence or climatology model. When using RMSE, the 'perfect model' is equivalent to RMSE = 0. This yields:

$$\text{SS}_{\text{model}}^{\text{RMSE}} = 1 - \frac{\text{RMSE}_{\text{model}}}{\text{RMSE}_{\text{reference}}}. \tag{4.10}$$

4.5 Forecast Models

In this section, we present the specific the steps performed for different models and their different forecasts.

4.5.1 Algorithm

```
for each forecast horizon do
    filter data based on availability;
    remove nighttime observations;
    partition the dataset into training and test set;
    partition training data into k half-year folds;
    define a parameter set (e.g. K = (1, ..., 10) for KNN);
    for each parameter p in parameter set do
        for each fold i in set of K folds do
            use fold i as validation set;
            [optional] pre-process the data;
            fit the model on the other K − 1 folds;
            predict values for the validation set;
        end
        calculate the average RMSE on the validation sets;
    end
    determine optimal hyperparameters (i.e. lowest RMSE);
    train model on all training data with optimal hyperparameters ;
    evaluate the model on the test set (i.e. the last six months of data);
end
```

Figure 4.6: ML algorithm

Figure 4.6 shows an overview of the machine learning algorithm performed for each site and model. The forecast horizons are from 15 minutes to 5 hours, divided into hourly intervals. Each site is trained independently as they all have different specifications. Furthermore, it allows for a comparison of how different models perform on different weather regimes. Each model is explained in more detail below.

4.5.2 ARIMA

The ARIMA modeling was performed in a classical time series method. As of this, a test and training set was set up, in which all data except the last recent half year was considered training data and the last half year as the test data.

The series of energy output was normalized with the clear sky normalization to reduce the non-stationarity. Later on, it was modeled with the ACF and PACF plots with lags set to 200 to capture any seasonal patterns, in which lag 96 is the daily seasonality.

Once the model had been determined, a Ljung-Box test was performed with the *stats* package [30] to see if there still existed any autocorrelations on 20 and 200 lags respectively. Furthermore, coefficient tests were performed to identify if the

parameters were significant. Ultimately, a *qq*-plot of the model's residuals was fitted to a normal distribution to see if the residuals followed a white noise process.

If any of these assumptions were violated, the validity of the ARIMA model can be questioned, and it was decided not to model the ARIMA in such cases. If the central assumptions of the ARIMA model could be achieved, then final predictions were computed on the different time horizons.

4.5.3 Lasso Regression

Since the Lasso penalizes overfitting and shrinks the β values of the non-important variables to 0, feature selection is already included in the algorithm. As of this, no apriori feature selection was performed before modeling with the Lasso. The glmnet function was thus fed with the adjusted dataset for the corresponding time horizons. The model training was performed with the R package *glmnet* [31].

Hyperparameters

Hyperparameter	Search grid
Penalty (λ)	$0.001, 0.002, \ldots, 0.049, 0.05, 0.06, \ldots, 0.99$

Table 4.3: Lasso: hyperparameters and search grids

4.5.4 *K*-Nearest Neighbors

All features were standardized according to Eq. 3.10, to make the Euclidean distances comparable. The model training was performed with the R package *class* [32]. To reduce the computational time, the five most important variables in the GBRT are used as features in the KNN algorithm. The number of features was chosen after some initial model runs, where the performance of predictions increased when adding features one at a time until five features were included. For a number of features in the range of 5-10, the error was similar to five, and therefore five was used. The model is thereafter tested for different K with cross-validation.

Hyperparameters

Hyperparameter	Search grid
Neighbors (K)	$1, 3, 5, \ldots, 99$

Table 4.4: KNN: hyperparameters and search grids

4.5.5 Gradient Boosting Regression Trees

The model training was performed with the R package *gbm* [33].

Hyperparameters

Hastie [16] claims that an interaction depth of $4 \leq J \leq 8$ performs well when boosting and that the trees are insensitive to choices in this range. For this reason, the interaction depth is set to $J = 6$ and is not optimized further. The choice of shrinkage v, require larger values of M to converge, and thus longer computational time. Ideally one would want a very small ($v < 0.1$) and a large number of M. In this case, a set of different shrinkage parameters are tested in the belief that the error will converge for some values of the shrinkage before reaching $M = 200$.

Hyperparameter	Search grid
Number of trees (M)	$1, 2, \ldots, 200$
Interaction depth (J)	6
Shrinkage (v)	$0.01, 0.03, 0.05, 0.07, 0.1$

Table 4.5: GBT: hyperparameters and search grids

4.5.6 Artificial Neural Networks

One, two- and three-layer networks with ten nodes per layer and sigmoid function as activation layers (3.1) are fitted to the data. The algorithm used is a resilient propagation as described in section 3.5.2, and the model training was performed with the R package *RSNNS* [19]. As in KNN, the features were scaled to the range $[0, 1]$ with a min-max scale (eq. 3.10). The initial weights were selected at random on the range $[-0.3, 0.3]$. The test of several amounts of layers and weight decays gives the ANN a chance to reduce the feature space, adapt to the number of samples, and a larger probability that start weights reach minimum for some combination of parameters.

Hyperparameters

Hyperparameter	Search grid
Nodes per layer	10
Weight decay (λ)	$0.01, 0.1, 1$
Hidden layers	$1, 2, 3$

Table 4.6: ANN: hyperparameters and search grid

No more combinations of parameters were tested due to the computational time of the ANN.

4.6 Benchmark Models

This study used two different benchmark models. The improvement in predictive performance for the time series and ML techniques will be set relative to the predictive performance of the benchmark models.

4.6.1 Persistence Model

A persistence-based forecast model (naïve model) is commonly employed when comparing different predictive model and is based on the idea that the next value is equal to the current value. Formally, the prediction is given by:

$$\hat{Y}_{t+h} = \mathrm{E}[Y_{t+h}|Y_t] = Y_t. \tag{4.11}$$

4.6.2 Climatology Model

Another benchmark is the climatology model, which uses an average of historical data as the forecasted value. A climatology model can be modeled differently in each case, but some historical average should be used. In this work, we have chosen to look at the average of the 100 most recent known 15-minute intervals. This means that for a prediction done at 10.00, the last 100 15-minute intervals will be used no matter if one is predicting the next 15-minute interval or 4 hours ahead. Formally, the prediction is derived from:

$$\hat{Y}_{t+k} = \frac{1}{n}\sum_{h=1}^{100} Y_{t-h}, \tag{4.12}$$

where k is the forecast horizon.

Chapter 5

Results

In this chapter, the empirical results of the models are presented for the different employed techniques.

5.1 ARIMA

The result of the clear sky normalization is presented in tables 5.1 and 5.2. The Gaussian kernel smoothing manages to reduce the stationarity to some extent. As the data is more centered during the summer months in the normalized time series, it still exists some variations in variance and mean during the winter seasons. Thus, the stationary is not completely erased.

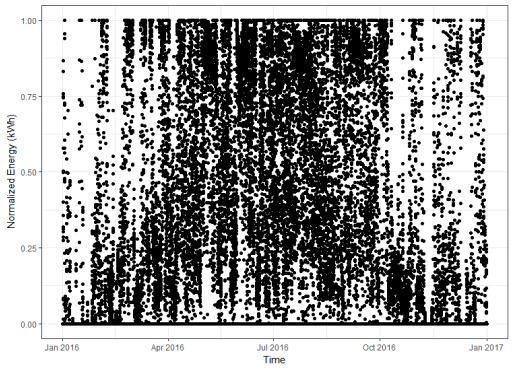


Figure 5.1: Normalized energy output

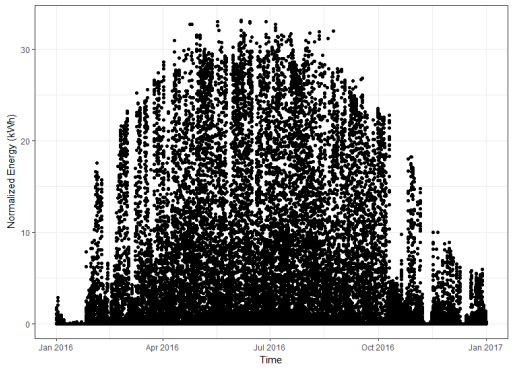


Figure 5.2: Actual energy output

As night data was included in the ARIMA, a seasonal pattern is evident, thus a seasonal differencing was first added. The ACF plot can be seen in figure 5.3, where a slow but exponential decay is present. The exponential decay indicates that a stationary series is attained, which seems likely from plotting the seasonally differenced series, figure 5.4.

Once starting to model for the seasonal and non-seasonal ARMA terms, it was clear that SAR and AR terms of order one were indicated due to exponential decays in the ACF, figure 5.3 and strong cut-offs in the PACF, figure 5.5. Thus, an ARIMA(1,0,0)(2,1,0) was fitted due to the evident partial autocorrelation at the second lag.

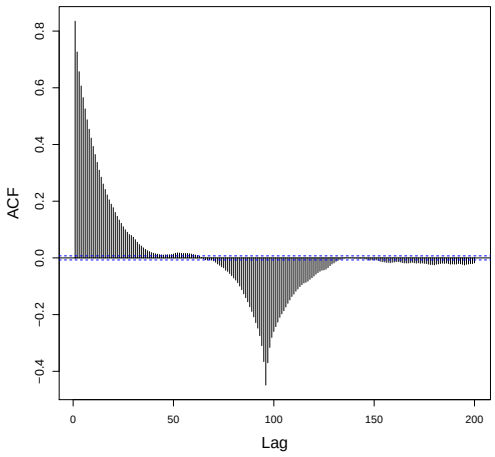


Figure 5.3: ACF of normalized energy

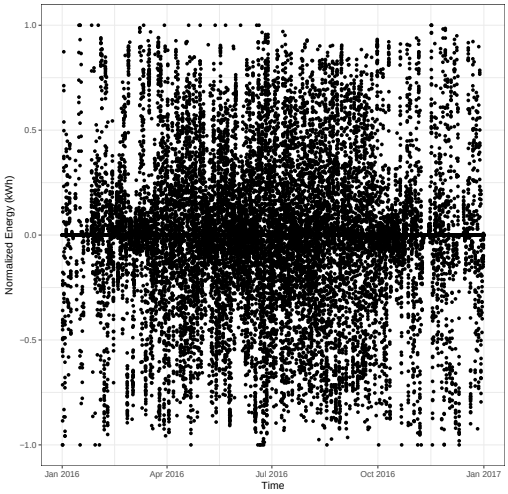


Figure 5.4: ACF of normalized energy with daily differencing

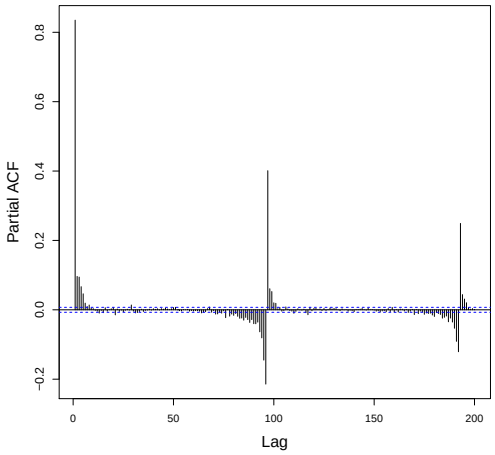


Figure 5.5: PACF of ARIMA$(0,0,0)$ $(0,1,0)$

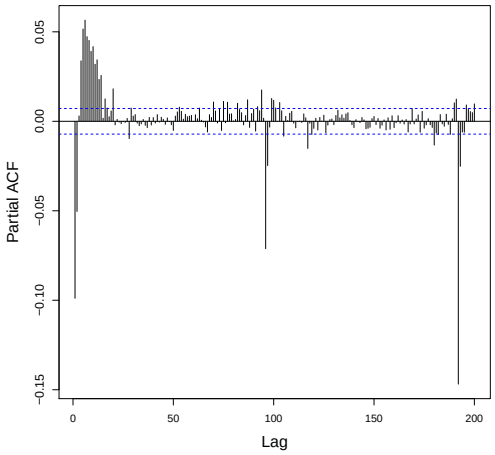


Figure 5.6: PACF of ARIMA$(1,0,0)$ $(2,1,0)$

As can be seen in figure 5.6 the seasonality is reduced; however the model does not manage to reduce it completely. As also can be seen, many significant auto correlations are found among the lags, in particular in the beginning. After successively trying different ARIMA setups according to the ACF and PACF, the best model found was an ARIMA(4,0,0)(2,1,0).

Once the best model was found with respect to the ACF and PACF, a noise in the ACF and PACF between the lags of around 10-90 was observed, which can be seen in figure 5.7 and 5.8. This violates the ARIMA model assumptions of uncorrelated lags.

To not only rely on the ACF and PACF plots, a Ljung-Box test was performed on both short term and long term lags, 10 and 200 respectively. As can be seen in Table 5.1, the p-value was very close to 0. This indicates that the null hypothesis should be rejected, which in a Ljung-Box test is that all autocorrelations are equal to zero. It can thus be concluded that this test also indicated a non-correlation between the

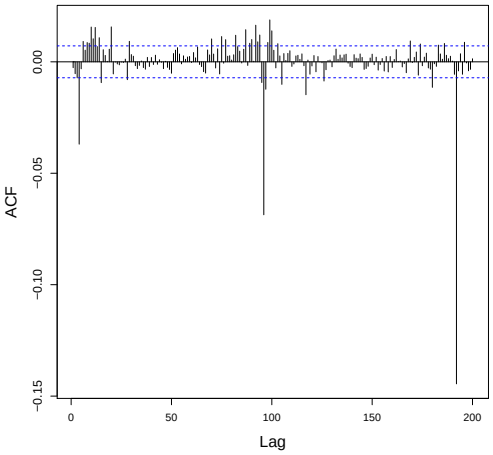


Figure 5.7: ACF of ARIMA $(4, 0, 0)$ $(2, 1, 0)$ residuals

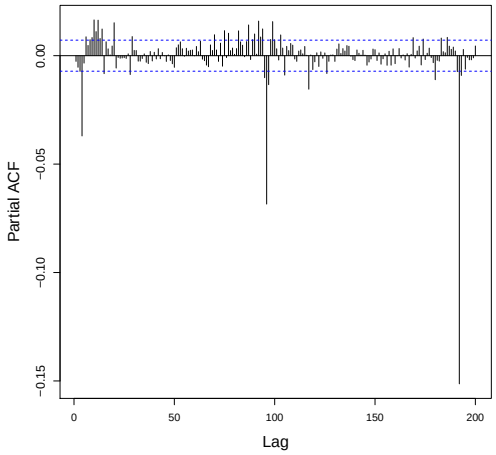


Figure 5.8: PACF of ARIMA $(4, 0, 0)$ $(2, 1, 0)$ residuals

lags over time, which is a violation of the main assumption of the ARIMA.

P-value of 10 lags	P-value of 200 lags
$\leq 2.2 \exp(-16)$	$\leq 2.2 \exp(-16)$

Table 5.1: P-values of Ljung-Box test for ARIMA(4,0,0)(2,1,0)

As a result of the autocorrelation violation, modeling with ARIMA was not pursued further. The above plots are for the first site, located in Uppsala, Sweden. Similar results and problems with the remaining four sites in Sweden were attained. As of this, no predictive results with the ARIMA model was produced.

5.2 Machine Learning Models

5.2.1 Hyperparameter Tuning

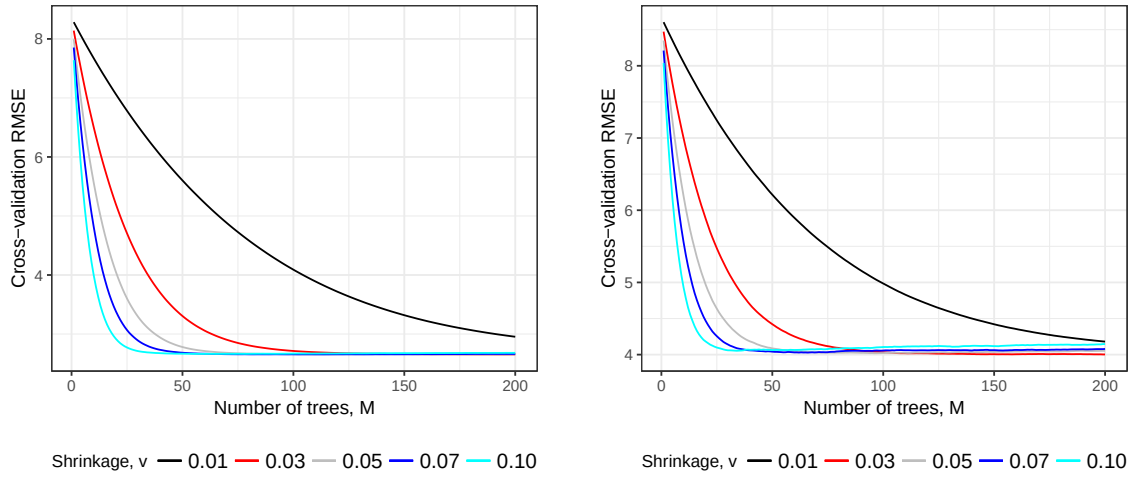


Figure 5.9: Site 1 GBRT cross-Validation RMSE. *Left:* 15 min. *Right:* 5 h.

As expected, a higher shrinkage gave faster convergence for the GBRT on both 15 minutes and five hours. For large M, the RMSE starts to grow due to overfitting for all shrinkages except 0.01, which might not have converged. The results for the other time horizons and sites were similar and therefore omitted.

Horizon (h) \ Site	1	2	3	4	5
0.25	0.017	0.022	0.007	0.001	0.006
0.5	0.001	0.004	0.001	0.001	0.004
1	0.001	0.017	0.001	0.024	0.007
2	0.001	0.004	0.008	0.027	0.005
3	0.002	0.013	0.006	0.039	0.007
4	0.001	0.010	0.001	0.008	0.080
5	0.001	0.017	0.001	0.001	0.240

Figure 5.10: Optimal λ for Lasso

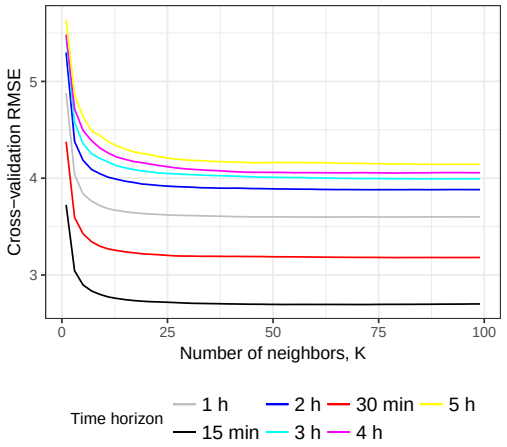


Figure 5.11: CV RMSE as a function of K

When looking at the Lasso, a small λ in the range of 0.001-0.08 is the most common, with one exception at site five for hour five. For the KNN, the RMSE converges before reaching $K = 99$ for all time horizons. Similar results were obtained for the other sites and therefore omitted.

For the ANN, the number of layers and weight decays used for each model varied widely, and no apparent pattern is observed. This could be linked to the random initialization of the weights, which is further discussed in chapter 6.

Horizon (h)	Site 1		Site 2		Site 3		Site 4		Site 5	
	Layers	Decay	Layers	Decay	Layers	Decay	Layers	Decay	Layers	Decay
0.25	2	0.1	3	0.1	2	0.1	1	0.1	3	0.01
0.5	3	0.1	2	0.01	3	0.01	1	0.1	2	0.1
1	2	0.01	2	0.1	3	0.01	1	0.1	2	0.1
2	1	0.1	3	1	3	0.01	3	0.01	3	0.01
3	3	0.1	1	0.1	3	0.1	2	0.01	2	0.1
4	1	0.1	2	0.1	1	0.1	2	0.01	2	1
5	1	0.1	3	0.01	1	0.1	1	0.1	3	0.1

Table 5.2: Numbers of layers and decay for different sites and horizons

5.2.2 RMSE

An overall lower RMSE is observed from the machine learning algorithms. The ML algorithms perform similarly in the beginning, while the more complex algorithms ANN and GRBT performs better at longer time horizons in most cases.

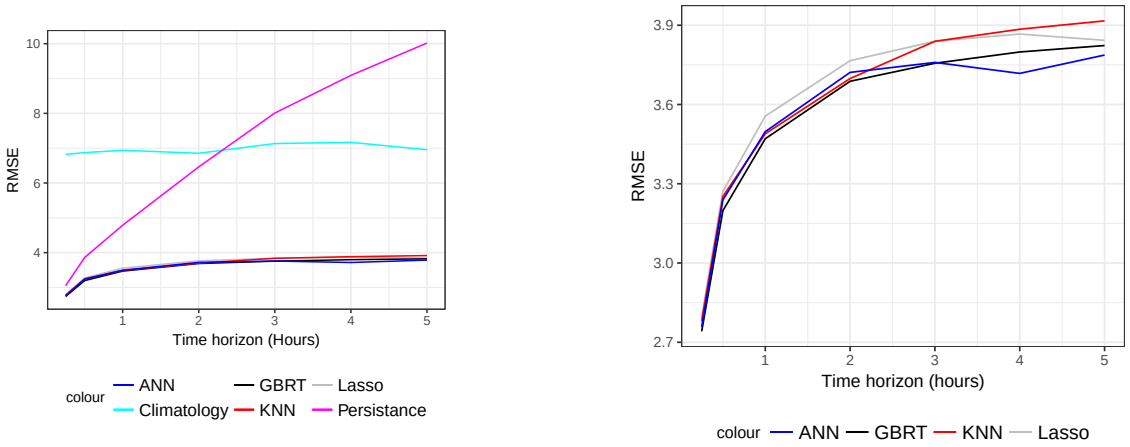


Figure 5.12: Site 1 RMSE. *Left:* All models. *Right:* ML models.

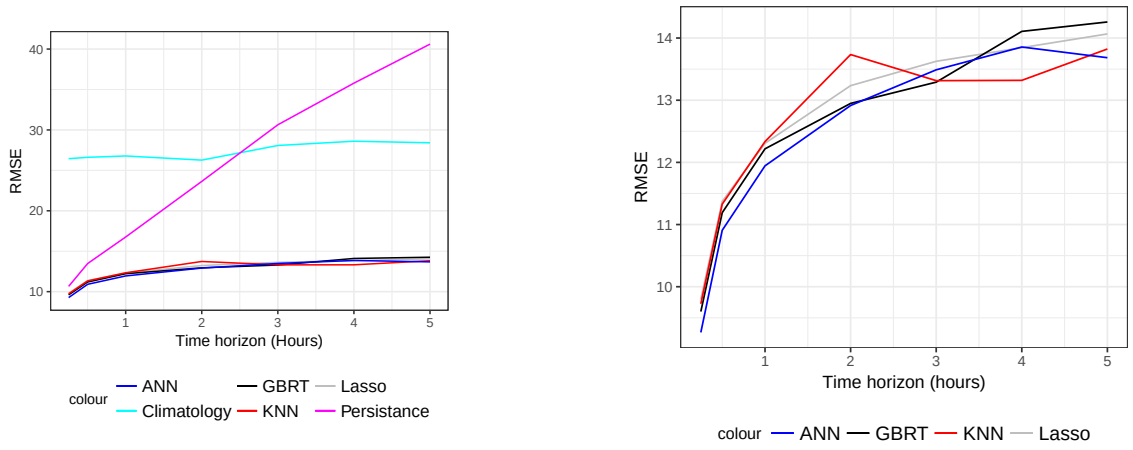


Figure 5.13: Site 2 RMSE. *Left:* All models. *Right:* ML models.

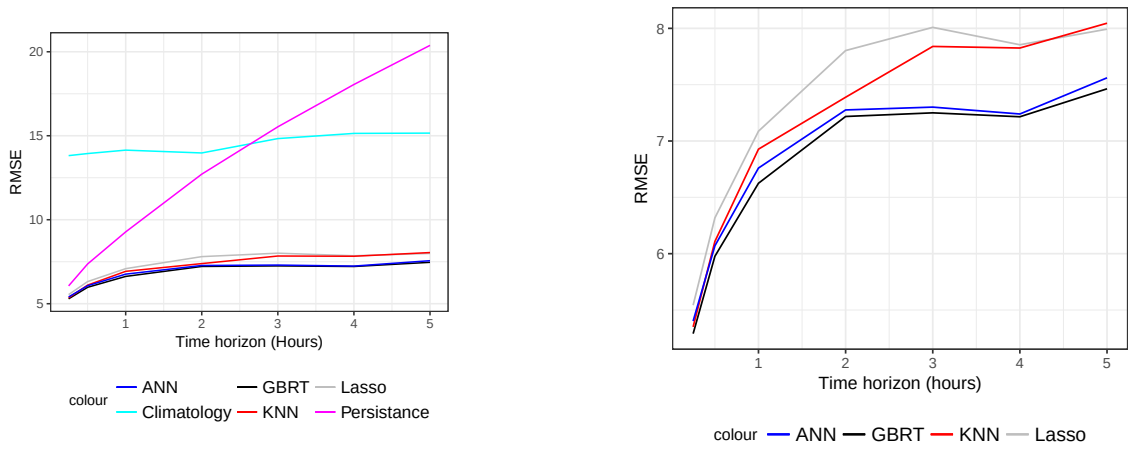


Figure 5.14: Site 3 RMSE. *Left:* All models. *Right:* ML models.

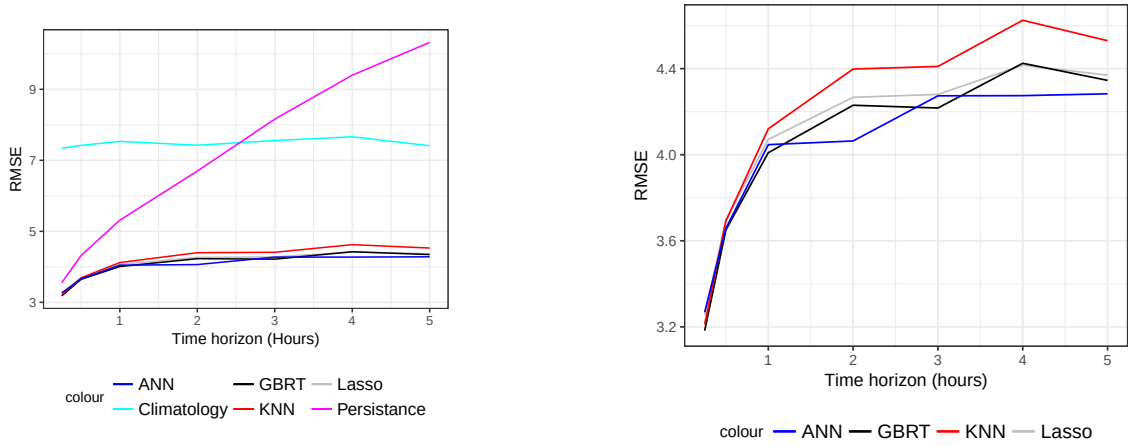


Figure 5.15: Site 4 RMSE. *Left:* All models. *Right:* ML models.

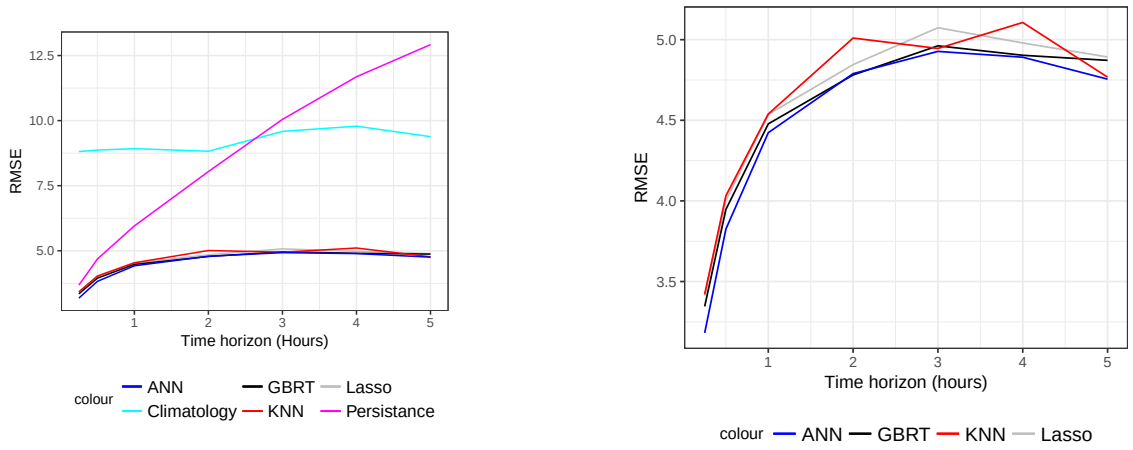


Figure 5.16: Site 5 RMSE. *Left:* All models. *Right:* ML models.

5.2.3 Variable Importance

Lagged output variables have a large variable importance for very short time horizons, but as the horizon grows NWP features become more significant.

Importance	Site 1	Site 2	Site 3	Site 4	Site 5
1	ylag1step	ylag1step	ylag1step	ylag1step	ylag1step
2	ylag2step	ylag2step	ylag2step	global_rad_1h.J	direct_rad.W_lead
3	global_rad_1h.J_lead	global_rad_1h.J_lag	ydelta	ylag2step	ydelta

Table 5.3: GBRT variable importance for 15-minute forecast horizon

Importance	Site 1	Site 2	Site 3	Site 4	Site 5
1	global_rad_1h.J	global_rad_1h.J	global_rad.W_lag	global_rad_1h.J_lag	direct_rad.W
2	global_rad_1h.J_lag	global_rad.W_lag	direct_rad_1h.J	global_rad_1h.J	direct_rad.W_lead
3	direct_rad_1h.J_lag	global_rad_1h.J_lead	global_rad_1h.J	global_rad.W_lag	global_rad.W_lead
4	direct_rad.W_lag	global_rad_1h.J_lag	global_rad_1h.J_lead	global_rad_1h.J_lead	global_rad.W
5	global_rad.W_lag	direct_rad_1h.J_lead	direct_rad_.W_lag	ylag1step	global_rad.W_lag

Table 5.4: GBRT variable importance for 5-hour forecast horizon

5.2.4 nRMSE

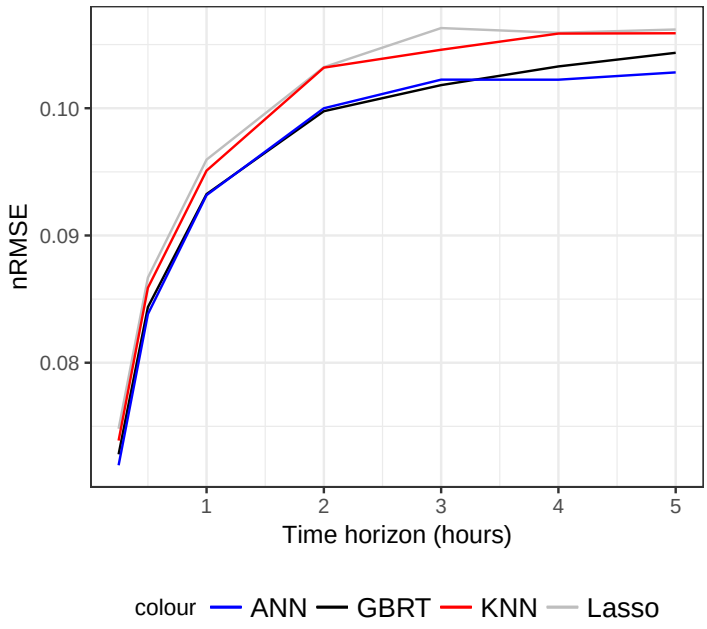


Figure 5.17: nRMSE

In figure 5.17 one can see that ANN and GBRT have the lowest nRMSE across sites and horizons.

5.2.5 Skill Scores

The GBRT and ANN have the highest skill scores for all time horizons.

Reference	Horizon (h)	Lasso	KNN	GBRT	ANN
Persistance	0.25	0.08	0.09	0.11	0.12
Persistance	0.5	0.15	0.16	0.17	0.18
Persistance	1	0.25	0.25	0.27	0.27
Climatology	2	0.46	0.46	0.48	0.48
Climatology	3	0.48	0.48	0.50	0.50
Climatology	4	0.49	0.49	0.50	0.50
Climatology	5	0.47	0.48	0.48	0.49

Table 5.5: Skill scores

5.2.6 Run-time

Model	Run-time (minutes)
Lasso	<1
KNN	~ 10
GBRT	~ 40
ANN	~ 70

Table 5.6: Run-time per site (minutes)

Chapter 6

Discussion

In this chapter, we will analyze and discuss the results of the study. The first sections will handle the results of the time series models and the ML techniques. In the later chapters, the discussion is directed towards general issues of the study as a whole.

6.1 Time Series Models

The achievements on the ARIMA were disappointing since the validity of the models did not hold mathematically. The result can be due to various reasons, but a likely one is that the time series was not stationary enough after the clear sky normalization.

Several reasons can explain the non-stationarity of the final time series. One is that the Gaussian kernel smoothing method did not manage to capture the different seasonal patterns over winter and summer. More sophisticated techniques need to be investigated to model the clear sky energy output accurately.

From a data perspective, the issue could also be explained by the inherited patterns in the data. For example, at most sites, several days of zero energy output were recorded during the winter months. In some cases this was explained by snow covering the panels for a longer time. Having many days in a row with zero output will reduce the variance and mean over that specific time, making the series non-stationary. With this in mind, it should be added that the clear sky normalization does not transform values that are equal to zero, thus it is hard to make it stationary.

The problem of reducing the series to a stationary series is also highlighted in the work of Bacher et al. [8], in which the authors also seemed to have significant lags between the seasonal autocorrelations. Bacher et al. conclude that further improvements of the techniques of reducing the non-stationary time series is needed and should be optimized. From this end, one can choose to take a more mathematically correct view and discard the model due to the inability to validate it. From a forecasting perspective, one can produce forecasts while ignoring the model assumptions. However, this is not pursued as choosing a suitable model is far from trivial in this setting.

From our results, we conclude that the ARIMA models are not suitable for modeling the energy output of solar PV panels in Sweden. This does not mean that the models should be discarded entirely, as the poor results are likely to be derived from a failed clear sky normalization or the complex time series for the specific site. Furthermore, it can be argued that the ARIMA is troublesome since it requires

advanced data transformations.

To conclude, the ARIMA is a robust method; however, it seems to not be the ideal choice for solar PV forecasting in Sweden. With better techniques to convert a complex non-stationary time series into a stationary time series, the ARIMA has potential. Furthermore, one should not forget to reflect the varying nature of the Swedish climate over the year, which may make the clear sky normalization perform worse than in countries with more stable climatic conditions. Considering this, the ARIMA may be a bad choice for solar energy forecasting in Sweden. However, it could be useful in other countries.

6.2 Machine Learning Models

One pattern observed was the correlation between the number of observations and the results. Here, site 1 and 3 had a similar number of observations while having similar results. This was observed on site 2 and 4 as well, however, these two sites had a smaller number of observations than site 1 and 2. This could be an indication that the number of data points affected the performance of the ANN and GBRT, and that more extensive datasets would potentially improve these models further. Another observation was the increased importance of NWP data input for predicting on longer time horizons. This is in line with what other studies has shown.

In nRMSE and skill score, the ANN and GBRT consistently outperform the other models. This result is not surprising since the GBRT and ANN can be considered as more flexible models than the Lasso and KNN, and thus, they should theoretically be better at incorporate complex structures in the weather and solar energy data.

To set things in perspective, the nRMSE of the GBRT, was slightly better than in the work of Persson et al. [8]. This can be explained by the employed data, where radiation NWPs were used in contrast to Persson et al. When comparing the results of the ANN and KNN with the study of Pedro and Coimbra [9], the ANN has a lower nRMSE in our results when looking at one and two hours, which could be explained by the fact that we implemented NWPs. In the study of Coimbra and Pedro, the KNN tended to have a few percentage points higher nRMSE than their ANN, which is consistent with our results however our spread was smaller.

When comparing the nRMSE with Pedro and Coimbra, we find a relatively high reduction in our results (from 20% to 10% on 2-hour horizons) for the ANN. The use of feature engineering and NWP data input could explain this, however, the nRMSE might not be directly comparable. For example, it is not clear from the study of Pedro and Coimbra how they managed the data sorting relative to when it was available. Also, their formula of nRMSE is somewhat different, thus not directly comparable. On the other hand, this gives some indication of whether the performance of the ANN is aligned with other studies.

Pedro and Coimbra [9] observed a similar result for the ANN and KNN. One reason for a smaller gap in nRMSE between the models could be the use of NWP data. Also, as discussed, the number of observations may not have been sufficient to make the ANN and GBRT perform better than the Lasso and KNN. The observed correlation between the number of observations and the results could partially explain this smaller performance gap between the KNN and ANN.

As can be seen in the long term, the Lasso deteriorates more than the ANN and GBRT. This is probably due to that the Lasso is a relatively simple model when used

with linear regression, and thus it does not manage to capture possible non-linear structures in the data.

The global radiation features had a high relative importance for the GBRT. The global radiation is dependent on other weather conditions and does, therefore, encircle many important climatic conditions in one variable. As this was one of the essential features for the GBRT and that our GBRT slightly improved compared to Persson et al. regarding nRMSE, we recommend other studies to incorporate this feature.

Other issues to discuss is the improvement potential for each model. As the study has not covered an in-depth analysis of different versions of the models, it is likely that some models can be improved. We believe that the models that have the most potential to improve is the KNN and ANN.

The reason for the KNN's potential improvement is mainly based on that no sophisticated feature selection procedure was performed, and thus it is likely that there exist versions that would give better predictions. Towards this end, improving the performance of the KNN with a proper feature selection procedure may be possible.

There was no evident ANN parameter setup (number of layers and weight decay) that outperformed the other. As the initial weights were initialized at random and only a limited amount of trials were conducted for each setup, further testing is needed to infer an optimal structure. However, even though the ANN outperformed the other ML models, there is room for improvement, mainly because the model has many variations. For example, bagging or boosting ANNs might mitigate the risk of ending up with a bad performing ANN due to bad initial start weights. Another aspect that was taken up in the literature review is that certain optimization algorithms can impact the network's ability to converge and thus improve the results. We have not employed the genetic optimization algorithm that previously has been proven to improve the performance of an ANN. A comparison of performance relative to computational times is relevant, in particular for the ANN. One could discuss the trade-off of attaining a gain in prediction accuracy while attaining an increase in computational time.

6.3 Other Issues

6.3.1 Data

One issue with the weather data is that the produced forecast is not for the exact coordinates of the PV installation. This is not ideal; however, it is likely that the impact is limited as the weather usually is similar in nearby areas. This may also explain why lagged weather variables were considered as more important in some cases, as the lagged variable may reflect the actual weather condition at the location of the PV panel. Another issue is that the forecasts are delivered four times per day and only have a horizon of six hours, which limits the number of observations as the forecast horizon grows.

Previous studies have emphasized the problem of the NWP data, where they underline that the NWPs have errors themselves. The NWP forecasts can also be based on different techniques with different accuracy. One way to improve the input data would have been to take the same NWP data from different sources,

both regarding suppliers and the underlying model employed for producing a NWP forecast. Through this, the input data errors' variability could have been lowered, and thus improved the results to some extent.

One other aspect to highlight is that we did not have access to NWP data over a spatial grid surrounding the solar PV installation location. As pointed out in the work of Andrade et al. [2], a spatial feature engineering process proved to enhance the performance of the predictions as it manages to capture how the weather is likely to change over a 15-minute interval. To further improve the results, a spatial grid of NWP should be collected at each location to produce spatial features.

6.3.2 Error Metrics

As observations in the morning and afternoon generally have lower output on average, many of the absolute prediction errors during those times are lower compared to the peak hours, resulting in a boosted RMSE as an average is computed. When the forecast horizon is five hours, it should be noted that most peak hour observations are removed from the dataset due to availability, and this might boost the results. In a business setting, it would be of interest to adjust the error metrics for the peak hours, since they are more critical for the business. However, no previous study that we have encountered has used an adjusted error measure, thus implementing this would make it complicated to compare our predictions with other studies.

6.3.3 Methodology

Regarding methodology, there are aspects to consider to improve the results. One thing, is to implement different models for different seasons. Splitting the year into a winter and a summer season, and then train the models independently would likely improve the results. To connect this to the problems mentioned of boosted performance metrics and the fact that models tend to perform better when trained on more specific periods, one could try to model only during peak hours. In this setting, the model would not have to make a trade-off of adapting the parameters in such way that they perform well on both morning or night data as well as during mid-day hours.

The problem with using one model over a whole year is that the model will be trained on data that vary due to the varying weather conditions. This forces the model to generalize more over the year. If one would train the model for only winter months, the model could solely focus on these points without having to do a trade-off between fitting the summer data well and fitting the winter data well. On the other hand, having different models depending on the climate would mean having to change models from time to time, which may be tedious in an operational setting.

Chapter 7

Conclusion and Further Work

In this work, we have compared time series techniques and machine learning techniques for solar energy forecasting across five different sites in Sweden. We find that employing time series models is a complex procedure due to the non-stationary energy time series. In contrast, machine learning techniques were more straightforward to implement. In particular, we find that the Artificial Neural Networks and Gradient Boosting Regression Trees performs best on average across all sites.

This study has compared the different models on a general level. For further research, we suggest continuing comparing different machine learning techniques in depth while using feature engineering approaches of numerical weather predictions.

Bibliography

[1] IRENA. Renewable power generation costs in 2017. Technical report, International Renewable Energy Agency, Abu Dhabi, January 2018.

[2] Jose R. Andrade and Ricardo J. Bessa. Improving renewable energy forecasting with a grid of numerical weather predictions. *IEEE Transactions on Sustainable Energy*, 8(4):1571–1580, October 2017.

[3] Rich H. Inman, Hugo T.C. Pedro, and Carlos F.M. Coimbra. Solar forecasting methods for renewable energy integration. *Progress in Energy and Combustion Science*, 39(6):535 – 576, 2013.

[4] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F.J. Martinez-de Pison, and F. Antonanzas-Torres. Review of photovoltaic power forecasting. *Solar Energy*, 136:78–111, October 2016.

[5] V Kostylev, A Pavlovski, et al. Solar power forecasting performance–towards industry standards. In *1st international workshop on the integration of solar power into power systems, Aarhus, Denmark*, 2011.

[6] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J. Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. *International Journal of Forecasting*, 32(3):896 – 913, 2016.

[7] Gordon Reikard. Predicting solar radiation at high resolutions: A comparison of time series forecasts. *Solar Energy*, 83(3):342 – 349, 2009.

[8] Peder Bacher, Henrik Madsen, and Henrik Aalborg Nielsen. Online short-term solar power forecasting. *Solar Energy*, 83(10):1772 – 1783, 2009.

[9] Hugo T.C. Pedro and Carlos F.M. Coimbra. Assessment of forecasting techniques for solar power production with no exogenous inputs. *Solar Energy*, 86(7):2017 – 2028, 2012.

[10] Federica Davò, Stefano Alessandrini, Simone Sperati, Luca Delle Monache, Davide Airoldi, and Maria T. Vespucci. Post-processing techniques and principal component analysis for regional wind power and solar irradiance forecasting. *Solar Energy*, 134:327 – 338, 2016.

[11] Changsong Chen, Shanxu Duan, Tao Cai, and Bangyin Liu. Online 24-h solar power forecasting based on weather type classification using artificial neural network. *Solar Energy*, 85(11):2856 – 2870, 2011.

[12] Caroline Persson, Peder Bacher, Takahiro Shiga, and Henrik Madsen. Multi-site solar power forecasting using gradient boosted regression trees. *Solar Energy*, 150:423 – 436, 2017.

[13] J. Shi, W. J. Lee, Y. Liu, Y. Yang, and P. Wang. Forecasting power output of photovoltaic systems based on weather classification and support vector machines. *IEEE Transactions on Industry Applications*, 48(3):1064–1069, May 2012.

[14] Peter J Brockwell. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer, 3rd ed. 2016.. edition, 2016.

[15] T. Hastie J. Gareth, D. Witten and R. Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer texts in statistics An introduction to statistical learning. Springer, 2013.

[16] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2001.

[17] Jerome H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367 – 378, 2002. Nonlinear Methods and Data Mining.

[18] D.Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429 – 1451, 2003.

[19] Christoph Bergmeir and José M. Benítez. Neural networks in R using the stuttgart neural network simulator: RSNNS. *Journal of Statistical Software*, 46(7):1–26, 2012.

[20] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.

[21] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2016.

[22] Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt. *caret: Classification and Regression Training*, 2017. R package version 6.0-78.

[23] Meteomatic. Basic weather parameters. `https://api.meteomatics.com/AP-Basic-Weather-Parameter.html`, 2011. [Online; accessed 2018-03-15].

[24] Jay A Kratochvil, William Earl Boyson, and David L King. Photovoltaic array performance model. Technical report, Sandia National Laboratories, 2004.

[25] Peder Bacher. Short-term solar power forecasting. Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.

[26] Gionata Biavati. *RAtmosphere: Standard Atmospheric profiles*, 2014. R package version 1.1.

[27] Feng Pan, Tim Converse, David Ahn, Franco Salvetti, and Gianluca Donato. Feature selection for ranking using boosted trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2025–2028. ACM, 2009.

[28] Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70 – 83, 2018.

[29] Allan H Murphy. Skill scores based on the mean square error and their relationships to the correlation coefficient. *Monthly weather review*, 116(12):2417–2424, 1988.

[30] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[31] Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13, 2011.

[32] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.

[33] Greg Ridgeway with contributions from others. *gbm: Generalized Boosted Regression Models*, 2017. R package version 2.1.3.

TRITA -SCI-GRU 2018:214