Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Sarah M. Daragmeh

Evaluation of Time Series Forecasting Methods Applied to Electricity Consumption Data

Integration the Forecaster in Smart Energy Management Systems

Master's thesis in Simulation and Visualization
Supervisor: Professor Ricardo da Silva Torres and Associate Professor Anniken Susanne T. Karlsen
July 2020

NTNU
Norwegian University of Science and Technology

Sarah M. Daragmeh

Evaluation of Time Series Forecasting Methods Applied to Electricity Consumption Data

Integration the Forecaster in Smart Energy Management Systems

Master's thesis in Simulation and Visualization
Supervisor: Professor Ricardo da Silva Torres and Associate Professor Anniken Susanne T. Karlsen
July 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

Summary

Background **:** Electric energy is one of the most value that economic growth in the modern societies, industries and economies depends on[1]. Nowadays, everything depends on electricity to run (e.g. services, productions, entertainment, ... , etc). Therefore, finding cheap, clean and continuous energy sources to meet the increasing demand is very important. The demand of energy is increasing constantly, and it is expected to increase by 50 percent by 2050 [2]. Electric load forecasting is an important tool which has been used to ensure that power utilities meet the consumers' need. The use of advanced technology, such as Advanced Meter System (AMS) and Internet of Things (IoT), provides the power utilities with a huge amount of data which can be used to design and implement intelligent energy management systems. The proper use of electricity consumption data better services and opportunities to engage consumers in demand response. Good time series forecasting in turn accuracy helps in building robust and smart energy management systems.

Objectives **:** The main objective of this study is to build a time series forecasting model which performs best on the electricity consumption data, and propose methods to integrate the forecasting model in smart energy management systems. Energy management systems are important to maintain energy sustainability in heterogeneous energy systems.

Methods **:** Statistical analysis was performed to understand the electricity consumption data of consumers from Aalesund, and to find the time series components. Different methods were applied to the time series of the total electricity consumption in the purpose of forecasting (short-term and mid-term). The methods are divided into classical statistical methods for time series forecasting (auto regressive integrated moving average (ARIMA) and exponential smoothing (ES)), and machine learning methods (linear regression, recurrent neural network (RNN), long short term memory (LSTM), convolution neural network (CNN), support vector machine (SVM), and K-nearest neighbor (K-NN)). A comparison study of the models' performance was done to find the best model. Also, there are two cases where we proposed conceptual models to integrate the forecasting model in energy management systems.

Results **:** Data analysis showed that the electricity consumption data has trend, seasonality, and noise. The evaluation of different statistical and machine learning methods indicates that machine learning models performed generally better than statistical methods on the given data. Statistical methods required much involvement from the user during the experiments, but this did not help in achieving best performance. Among the statistical methods, seasonal auto regressive integrated moving average (SARIMA) achieved the best performance for both short-term and mid-term load forecasting. CNN wavenet outperforms all the tried

[1]Correlation of energy consumption and GDP per person, `https://www.eea.europa.eu/data-and-maps/figures/correlation-of-per-capita-energy` (As of 27.06.2020)

[2]EIA projects, International Energy Outlook 2019 with projections to 2050 U.S. (Energy Information), `https://www.eia.gov/outlooks/ieo//` (As of 27.06.2020)

methods (i.e., statistical and machine learning methods) in both short-term and mid-term load forecasting.
The simulation results from the proposed model of an intelligent energy management system showed the importance of load forecasting in such smart systems. The results showed how the energy in micro-grid systems can be managed efficiently depending on the forecasting values. Also, the simulation results form the scheduling of shiftable appliances case, showed that the consumer can save up to 300 NOK monthly by using shiftable appliances on optimal times.

Conclusion **:** Load forecasting is an important topic for different smart solutions such as smart micro-grids, and smart, green and sustainable cities. This work provides a methodology to design a good time series load forecasting model which depends on the available data, and illustrates the effectiveness in applying the forecasting model in different domains.

Also, we conclude that the machine learning methods outperform statistical methods, and CNN wavenet performed best on our data.

Keywords **:** Advanced metering system, time series data, load forecasting, statistical analysis, machine learning, smart grids, smart cities.

Acknowledgements

Firstly, I would like to thank my supervisors, Professor Ricardo da Silva Torres and Associate Professor Anniken Susanne T. Karlsen, for their guidance, advice, expertise, motivation, encouragement and for following me up during the thesis work. They have contributed with valuable input and knowledge.

Also, I would like to thank my fellow students and my teachers at the Department of ICT and Natural Sciences for their encouragement.

I would also like to thank my family, my husband, my daughter and my son for their encouragement and patient during the master study. This master study could not be realized without your support.

Preface

This Master thesis was carried out at Norwegian University of Science and Technology (NTNU), Faculty of Information Technology and Electrical Engineering, Department of ICT and Natural Sciences, in the period from January 2020 to June 2020, under the supervision of Professors, Ricardo da Silva Torres and Associate Professor Anniken Susanne T. Karlsen. The electricity consumption data which were used in this master project were provided by Mørenett company (`morenett.no`). The data is including measurements from 1112 units from Aalesund region for 53 weeks (from 16 Nov 2018 to 24 Nov 2019). The measurements are taken hourly and the 1112 units are divided between apartments and houses (960 units), industry (114 units) and cabins (38 units).

Sarah Daragmeh, 2020/07/03

Table of Contents

List of Tables

List of Figures

Abbreviations

ACF	=	Autocorrelation Functions
AIC	=	Akaike’s Information Criterion
AMI	=	Advanced Metering Infrastructure
AMS	=	Advanced Metering System
ANN	=	Artificial Neural Network
AR	=	Autoregressive
ARMA	=	Autoregressive Moving Average
ARIMA	=	Autoregressive Integrated Moving Average
BP	=	Back-propagation
CNN	=	Convolution Neural Network
DES	=	Double Exponential Smoothing
DPA	=	The Norwegian Data Protection Authority
DL	=	Deep learning
DSM	=	Demand Side Managements
ES	=	Exponential Smoothing
EU	=	European Union
EV	=	Electrical Vehicle
FF	=	Feed Forward
GA	=	Genetic Algorithms
HAN	=	Home Area Network
GRU	=	Gated Recurrent Unit
IEMS	=	Intelligent Energy Management System
K-NN	=	k-nearest neighbors
KW	=	Kilowatt
LR	=	Linear Recognition
LSTM	=	Long Short-Term Memory
LV	=	Low Voltage
MA	=	Moving Average
MFE	=	Mean Forecast Error
MAE	=	Mean Absolute Error
MAPE	=	Mean Absolute Percentage Error
MPE	=	Mean Percentage Error
MSE	=	Mean Squared Error
ML	=	Machine Learning
MLP	=	Multi-layer perceptron
MW	=	Megawatt
NARX	=	Nonlinear Autoregressive with exogenous inputs
NMSE	=	Normalized Mean Squared Error
PACF	=	Partial Autocorrelation Functions
RBF	=	Radial Basis Function

RMSE	=	Root Mean Squared Error
RNN	=	Recurrent Neural Network
R^2	=	R Squared
SES	=	Simple Sxponential Smoothing
SG	=	Smart Grid
SVM	=	Support Vector Machine
SVR	=	Support Vector Regression
TES	=	Triple Exponential Smoothing

Chapter 1

Introduction

In recent years, there is a huge increase in the energy consumption remarkably in the development countries. It is expected that the global energy consumption will increase by nearly 50 percent by 2050[1], and this leads to an increase in the energy demand. Using smart meters (i.e., Advanced metering system (AMS)) and IoT provides the power utilities with huge amount of data of electricity load at different scales (e.g., individual, group, and region). The data can be used for analysis, planning, and optimization. Efficient management of energy consumption is essential and important in several domains, such as smart grids, sustainable and smart cities, and CO2 emission reduction.

The energy consumed from local grid needs to be adjusted and predicted efficiently to reduce the consumption cost and the impact on the environment. Therefore, precise prediction of energy consumption at different scales and horizons has become a crucial topic and it is necessary to develop a reliable predictive model, to reduce energy costs, improve services, and reduce emissions.

An intelligent micro-grid is a recent power scenario that means using renewable resources to generate the power which consumer can use (e.g., solar cell, wind energy, and energy storage). In this scenario, the power will be cheaper and cleaner, and the power utilities can meet the increase in the energy demand and generate energy which is in balance with the demands (i.e., balance in generating and demand process).

Norway is undergoing a formidable adjustment to cut emissions of harmful greenhouse gases. Electrification in different sectors, such as the transport sector, both on land and at sea, is one of the most important measures. Energy calculations in Norway show that this will contribute to 30 percent lower energy costs for an ordinary family, simply because electric cars require significantly less energy than diesel and petrol cars. This amounts to around NOK 8,000 in annual savings for the family. In March this year, more new electric cars were sold than fossil cars in Norway (`morenett.no`). However, if everyone is to

[1]EIA projects, International Energy Outlook 2019 with projections to 2050 U.S. (Energy Information), `https://www.eia.gov/outlooks/ieo//` (As of 27.06.2020).

charge the electric car at the same time, we will have trouble with the capacity of the power grid. The smartest solution is to distribute consumption over several hours of the day. This avoids costly investments in the electricity grid that consumers have to pay for. It is a bit like our roads: It is sub-optimal to build a four-lane highway to take away a couple of hours of rush hour traffic, if you can better distribute traffic throughout the day.

According to the Norwegian Water Resources and Energy Directorate (NVE), NOK 11 billion can be saved in the electricity grid by charging the electric cars at times of the day when electricity consumption is generally low. In the same context, if we build smart solutions at different levels (i.e., house, neighborhood, city), we can save a lot by reducing the electricity cost. These savings may benefit electricity customers through lower grid rents.

1.1 Background

Accurate electrical load forecasting by using historical time series data gathered by Advance smart meter (AMS) helps us in designing better systems with minimum energy losses. Designing a good predictor depends on the method and data. The data amount and accuracy are important in the case of generalization. Finding the best method that gives the best prediction results for the given data is the first step in designing a good forecasting model to use it in different domains and applications, such as:

- Demand side management;
- Sustainable, green, smart cities;
- Reducing CO2;
- Integrating emergent demands (e.g., electrical vehicles (EV) and electrical public transportation);
- Integrating renewable energy, IoT devices, clouds.

Recently, electrification in all aspects of our life such as transport leads to high need of energy. Then energy management at different scales become an important topic because it benefits in saving energy resources and reducing associated prices. Energy management is a cyclic process which starts from defining the demand, then makes response and optimizes the process. Figure 1.1. shows the main parts of an intelligent energy management system (IEMS). As we know, the demand by predicting it from the historical data and gathered real-time data, we make response to fulfil the demand [3, 4, 5, 6, 7, 8]. We optimize the process by utilizing the distributed energy sources by using efficient and smart energy management system.

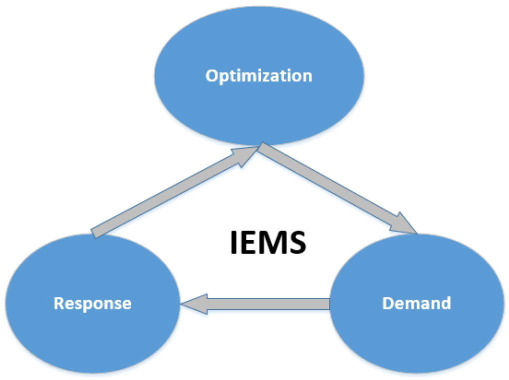


Figure 1.1: Intelligent energy management system concept.

1.2 Objectives

The main aim of the master thesis is to build a load forecaster which can be used by different actors (e.g., power utilities, power generators, and consumers) for their needs as following: The power utilities will use the load forecasting model to sustain a balance between supply and demand, build good grid structure, planning (e.g., maintenance) and provide good services. The power generators will use the predictor to meet the load demand, find power sources, and reduce the generating price. The consumers can use such predictor to reduce the consumption cost by scheduling shiftable loads. The overall goal can be divided to sub-goals and numerated in the following research questions:

- Evaluate different prediction methods, such as statistical methods and machine learning methods to find the best method which gives the best performance on the given data. There are many methods that can be used to build forecasting models.

  These methods can be divided into two categories: statistical and machine learning methods. We will assess well-known methods from different categories to find the best promising one. From literature which presents in chapter 3, we can conclude that there is no guideline to guide us to which methods give a better result to a specific problem [5, 8]. There are few papers that evaluate different methods on standard data sets only [5, 7, 9, 10], but we aim to find the best model depending on our data.

- Investigate how available data should be pre-processed to improve the prediction accuracy.

  We analyze the historical load data to find patterns that can help in predictions (such as trend and seasonality) and find the correlation between the energy consumption and other factors such as weather data and time events (e.g., holidays).

- Investigate how the results form the energy consumption prediction model can be used to build smart grids; for example integrate the renewable energy, increase the user's awareness, reduce CO2 emission , ... etc.

1.3 Scope

The scope of this master thesis is to explore different statistical and machine learning methods to perform time series energy consumption forecasting on real data gathered hourly from the 1112 units from Aalesund region for 53 weeks. In order to find the best method, we will evaluate different forecasting methods. The scope lay within the boundaries of energy consumption at different scales, energy management systems, forecasting methods (statistical and machine learning methods) and application domains such as smart grids and smart cities. Figure 1.2 shows where the research area in this thesis is.

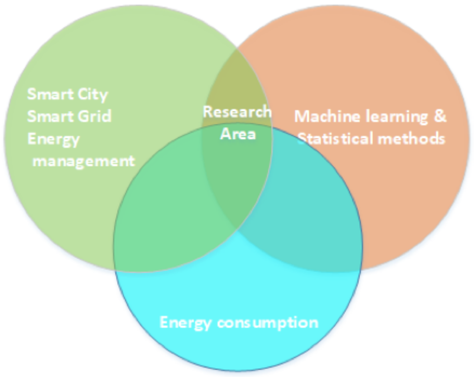


Figure 1.2: Research area.

1.4 Research methodology

In order to answer the research questions, I have followed the methodology that is illustrated in Figure 1.3. I started by reading about the energy efficiency in the last semester when I was doing the specialization project. In the specialization project course, I developed a smart energy consumption system where the residential households can do appliance scheduling to reduce energy consumption and the bill cost while keeping resident's comfort. In order to design intelligent energy management systems (IEMS) in different levels such as micro-grid, we have to build predictive models based on the available data. I got electricity consumption data, which is measured by advanced metering system (AMS) for the area in Aalesund[2] from Mørenet AS (`morenett.no`). In the master thesis, I am starting the project by reading about the basic theory and related works to my thesis topic (Chapter 2 - Background Concepts and Chapter 3 - Related work). Then, I will analyse the data, find

[2]Aalesund is the largest municipality in the Møre and Romsdal county in in the northernmost part of Western Norway

the correlation to other available data, and find the tools and libraries that will be used in the thesis project (Chapter 4 - Methodology). The forecasting model will be designed by evaluating different time series forecasting methods (Chapter 5 - load forecasting results), then the results will be applied in the designing DSM and IEMS (Chapter 6 - Case study A,B). Finally I will discuss my results, and how to apply them in different domains (Chapter 7 - Discussion).

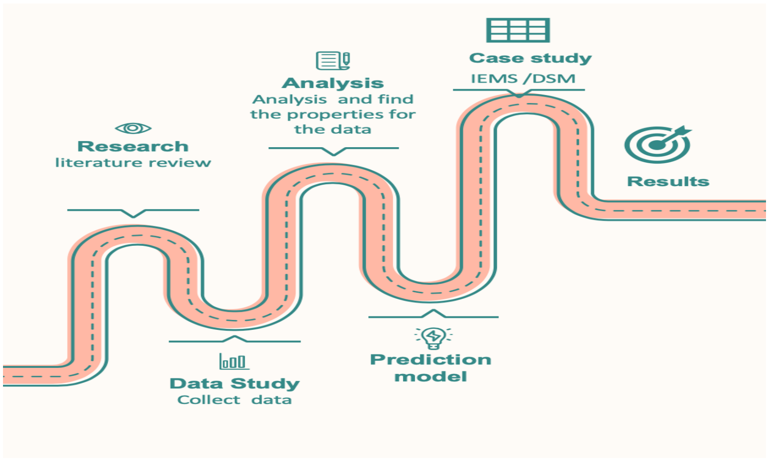


Figure 1.3: Methodology

1.5 Thesis structure

The layout of the thesis is as follows:

- **Chapter 2 - Background Concepts:** Provides an overview of background concepts relevant for our work. This includes theory about time series statistic and machine learning methods.
- **Chapter 3 - Related work:** Explores researches relevant to this thesis. The chapter gives a summary of time series prediction with traditional methods.
- **Chapter 4 - Methodology:** Presents the scientific method used in this thesis. This includes data collection, data analysis, implementation details and tools.
- **Chapter 5 - load forecasting results:** Presents the forecasting models and the evaluation of them.
- **Chapter 6 - Case study A,B:** Gives an overview of the two cases studied in this thesis.
- **Chapter 7 - Discussion:** Discuss the results from the time series forecasting model implementations, and the results from the case studies.

- **Chapter 8 - Conclusion and future work:** Contains the conclusion by answering research questions, stating the contributions from this thesis, and presenting possible venues for future work.
- **Chapter 9 - Legal and ethical considerations:** Contains the legal and ethical aspects related to the used of data and results.

Chapter 2

Background concepts

This chapter provides an overview of the main concepts related to time series and time series forecasting, as well as the challenges in time series forecasting in Section 2.1. Time series forecasting methods are introduced in Section 2.2. Next, Sections 2.3 and 2.4 present statistical and machine learning approaches, respectively, which are considered in our study. Finally, section 2.5 introduces different evaluation performance measures.

2.1 Time series

A time series Y of size m is a sequential set of data points, i.e, $Y = (Y_1, Y_2, ...Y_m)$,where $Y_t \in \Re$ measured typically over successive times t [11]. If the time series values are synthesized by a function $Y = f(t)$, the time series is deterministic. On the other hand, when the time series has a random term ϵ, $Y = f(time, \epsilon)$, then the series is stochastic or non-deterministic in addition to time function. Another relevant feature of a time series refers to its stationariness properties. In statistical terms, a stationary process is assumed to be in a particular state of statistical equilibrium, i.e., the mean and variance remain constant over time [12]. In other words, the time series develops randomly in constant average. This property is essential for some methods that assume the stationary condition. We can change the time series from non-stationary to stationary by taking the first difference defined as $\triangle Y = Y_t - Y_{t-1}$. This process is enough in most times, although we may need the second difference also to make time series stationary in some cases.
Time series can be divided into three components [13]:

1. Trend (T): is a gradual shift or movement to relatively higher or lower values over a long period of time (e.g., linear, exponential, damped, and polynomial long-term increase or decrease).
2. Seasonality (S): This is the periodic fluctuation of the variable subjected to analysis. It consists of effects that are stable along with time, magnitude and direction.

3. Residual (R): This is the remaining, mostly related to un-explainable part of the time series. This component can be sometimes high enough to mask the trend and seasonality.

A time series is usually modelled through a stochastic process $Y_{(t)}$, i.e. a sequence of random variables. In a forecasting setting we find ourselves at time t and we are interested in estimating $Y_{(t+h)}$, using only information available at time t.

According to Equations 2.1 and 2.2, we can reformulate the time series Y_t by using the components by using either an additive (Equation 2.1) or a multiplicative (Equation 2.2) approach.

$$Y_t = T_t + S_t + R_t \tag{2.1}$$

$$Y_t = T_t \times S_t \times R_t \tag{2.2}$$

In the additive model (Equation 2.1), the components are added together. The additive model is linear, where changes over time are consistently made by the same amount. The trend is a straight line while the seasonality has the same frequency (width and amplitude of the cycle over time).

In the multiplicative model, the components are multiplied together (Equation 2.2). The model is a non-linear, and it is used when the trend seasonal variation increases or decreases in magnitude over time. The trend is non-linear (e.g., exponential or polynomial), and the seasonal variation increases or decreases over time (i.e., width and amplitude of the cycle varies over time).

Time series forecasting

Forecasting can be defined as making a prediction about the future [14]. Forecasting is often associated with tasks related to the construction of models that fit on historical data and their use to predict future observations. In forecasting, the future is completely unknown and we can predict it by performing estimation based on what has already happened. The concept of forecasting model is illustrated in Figure 2.1.

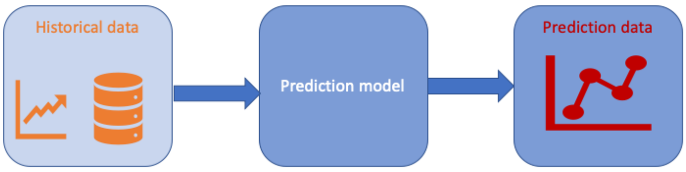


Figure 2.1: Concept of forecasting model.

In general, predictive modeling is a technique that uses mathematical and computational methods to predict future values based on the patterns and features which are extracted from the historical data.

2.1.1 Challenges in time series forecasting

When predicting time series values, there is a number of challenges that need to be addressed. Some examples include:

- Dependency: In time series, the observations for an input variable depend upon one another. For example, an observation at time t depends upon the observation at $t-1$, and $t-1$ may depend on $t-2$, and so on. We call such variables endogenous because they are affected by other variables in the system and the output variable depends on. Although time series might also have exogenous variables (e.g., weather data and holidays), it is usually the endogenous properties of variables that distinguish them across different problems.
- Time series may have obvious patterns, such as a trend or seasonal cycles.
- In simpler prediction problems, we may just want to predict the value of the next time step. In several problems, however, we might want to predict multiple steps, which makes the prediction problem harder.
- Some models are "static" and are not expected to be updated, while others are dynamic, i.e., models are expected to be retrained from time to time.
- Sometimes we have to handle contiguous data that have uniform observations over time; but also we may have to deal with discontinuous data, where observations are not uniform over time and so data need additional preparation.

2.2 Time series forecasting methods

There are many models available in the literature that can be used for time series forecasting. These models can be dived into two main categories, statistical methods and machine learning methods, as illustrated in Figure 2.2 [15].

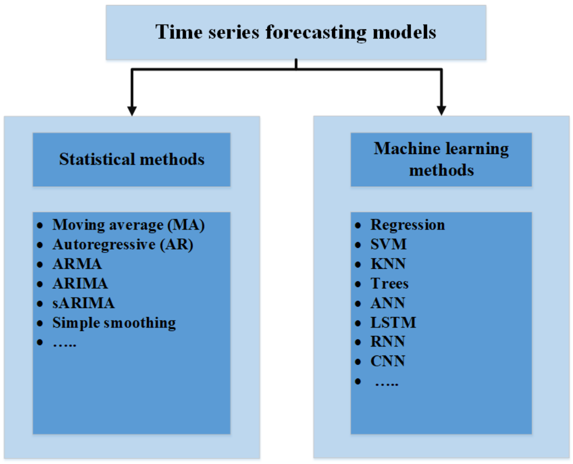


Figure 2.2: Time series forecasting models and their categories.

Generally, the forecasting methods can be divided into three main categories [13]:

- Time series methods: make use of the past data to compute future estimates.
- Causal methods (explanatory): analyse the data from the viewpoint of a cause-effect relationships.
- Qualitative methods: rely on experts' opinion.

2.3 Statistical methods

This section introduces some concepts related to the time series prediction based on statistical methods. Also, it describes some of the statistical methods used in forecasting tasks such as: Naïve model, moving average model, autoregressive integrated moving average (ARIMA), and exponential smoothing models. Both ARIMA and exponential smoothing models are considered as the baseline among systems for time series prediction for many years [16].

The statistical methods need information about the data distribution in order to build predictive models. This assumption makes the accuracy model dependent on its parameters in making the prediction.

2.3.1 Naïve method

Naïve forecasting, also it is called a random walk model, is a simple technique which predicts the value of a future observation by expecting it as the last observation, as shown in Equation 2.3. This makes the expected value of a future observation the same as the last observed [13].

$$\hat{y}_{t+h} \mid t = Y_t \tag{2.3}$$

This method works remarkably well if the time series has stochastic pattern which is difficult to predict as in many economic and financial time series [13]. The seasonal naïve method expects the future value to be equal to the value from the last season (e.g., same time in the previous year). This method works better than standard naïve method when the time series has seasonality. In general, the naïve method is used only for comparison with the forecasts generated by the better (sophisticated) techniques.

2.3.2 Moving average (MA)

Moving averages are developed based on an average of weighted observations, which tends to smooth out short-term irregularity in the data series. They are useful if the data series remains fairly steady over time [17]. Y_{t+1}, the forecast value at time $t+1$, is simply the moving average at time t as defined in Equation 2.4:

$$Y_{t+1} = M_t \tag{2.4}$$

A MA is obtained by calculating the mean for a specified set of values and then using it to forecast the next period. The MA at time t for n number of observations is calculated as:

$$M_t = \frac{(Y_t + Y_{t-1} + \cdots + Y_{t-n+1})}{n} \tag{2.5}$$

where n is the number of observation included in the average. The higher the value of n, the more smoothed will be the predicted data behaviour. The MA at time t can be measured by using the MA at time $t-1$.

$$M_{t-1} = \frac{(Y_{t-1} + Y_{t-2} + \cdots + Y_{t-n})}{n} \tag{2.6}$$

By subtracting Equation 2.6 from Equation 2.5 we obtain:

$$M_t = M_{t-1} + \frac{(Y_t - Y_{t-n})}{n} \tag{2.7}$$

This equation states that the moving average can be updated by using a previous moving average plus the average changes in actual value from time t to $t - n$. Using either Equation 2.5 or Equation **??** leads to the same result.

The moving average method provides an efficient mechanism for obtaining a value for forecasting stationary time series. The technique is simply an arithmetic average of the last n values to predict the next value. The difficulty in using moving averages is their inability to capture the peaks and troughs of the series. When the (actual) data are moving down persistently, the MA forecast trends to produce over-predicted valued; while when the data is moving up continually, the MA forecast will lead to an under-prediction.

The MA process relies on providing equal weights for different observations; this approach fails to reflect the importance of time ordering with respect to observations. The MA can be modified to weighted moving averages, where the observations are multiplied by different weights [13].

2.3.3 Autoregressive Integrated Moving Average (ARIMA)

Autoregressive Integrated Moving Average (ARIMA) is a statistical method and it is one of the most important linear models used for forecasting [18]. There are two types of ARIMA models: i) non-seasonal ARIMA, and ii) seasonal ARIMA (SARIMA). We will first explain the non-seasonal ARIMA, then will move to describe SARIMA. Non-seosonal ARIMA models predict future values based upon the construction of threes components: i) Autoregressive (AR), ii) Integrated (I), and ii) Moving Average (MA). The model is displayed as ARIMA(p, d, q). The p,d, and q values represent the amount of periods to lag for in the ARIMA model calculations.

We have to discuss the concepts of stationarity and autocorrelation before we introduce ARIMA models.

Stationarity

The stationary time series is a series of constant mean and variance over time. A stationarized series is easy to predict by ARIMA models. It simply predicts that the mean and variance will be the same in the future as they have been in the past. A stationary time series allowed us to obtain meaningful statistics, such as means, variances and correlations with other variable.

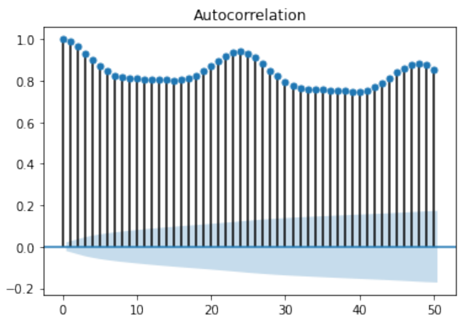
Differencing is used to transform a non-stationary time series to a stationary one. This is an important step in preparing data to be used in an ARIMA model.

Autocorrelation and Partial Autocorrelation Functions (ACF and PACF)

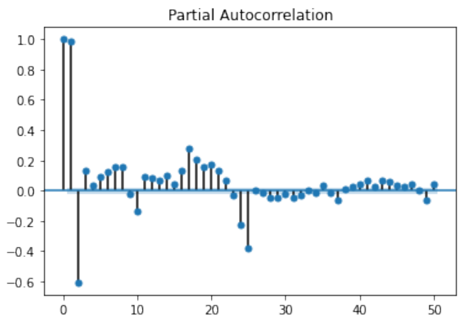
Autocorrelation (ACF) and partial autocorrelation (PACF) plots are most used in time series analysis and forecasting. To determine a proper model for a given time series data, it is necessary to carry out the ACF and PACF analysis. These statistical measures reflect how the observations in a time series are related to each other (i.e., correlated). For modeling and forecasting purpose, it is often useful to plot the ACF and PACF against consecutive time lags. These plots help in determining the order of AR and MA terms (i.e, p, q).

In general, ACF functions are used to find the relations between time series and the shifted lags of time series at different time step t. The investigation of the relationship between lags enables us to detect important dependencies in time series data. Figure 2.3a illustrates the correlation between lags.

The partial autocorrelation is the correlation between two variables controlling for the values of another set of variables. In time series, the lag (t) is correlated with lag $(t-1)$, and lag $(t-1)$ is correlated with lag $(t-2)$, and so on. Then, lag (t) is also partially correlated to lag $(t-2)$. The partial correlation of a time series with its own lagged values is given by the PACF. Figure 2.3b illustrates the partial correlation in a time series.



(a) Autocorrelation in the electricity energy consumption dataset.



(b) Partial autocorrelation in the electricity energy consumption dataset.

Autoregressive Models (AR)

AR is a linear regression model that performs prediction according to the relationship between the observation and past observations in series called lags. In other words, AR model use lagged variable as input to make prediction. The idea is to explain the present value of series Y_t by function of p past values: $Y_{(t-1)}, Y_{(t-2)} \dots Y_{(t-p)}$.

AR model with order p can written as:

$$Y_t = c + \Phi_{t-1}Y_{t-1} + \Phi_{t-2}Y_{t-2}... + \Phi_{t-P}Y_{t-P} + \epsilon_t \tag{2.8}$$

where c is a constant, Y_t is the forecast random variable, Φ_P is an explanatory random value, p is the number of lags (i.e., model parameter to be estimated), and ϵ_t is a white noise.

We can re-write Equation 2.8 as:

$$Y_t = \sum_{i=1}^{p} \Phi_i Y_{t-i} + \epsilon_t = c + \Phi_{t-1}Y_{t-1} + \Phi_{t-2}Y_{t-2}.. + \Phi_{t-P}Y_{t-P} + \epsilon_t. \tag{2.9}$$

White noise found if the variables are independent and identically distributed with a mean of zero. This means all variables have the same variance (and each value has a zero correlation with all other values in the series) [11].

Equation 2.9 illustrated that the order of the model p is fixed, while the parameters are adapted on the data at hand [19].

AR(p) mode is computed by determining lag p, by following the characteristics of the ACF, and PACF. The AR model is suitable to be used if the plots of the ACF and PACF following this:

- ACF of an AR(p) process has a geometrical decline (tails off).
- PACF of an AR(p) process cuts off at lag p.

Moving average (MA)

Moving average is a linear relationship uses past forecast errors in a regression $\epsilon_{t-1}, \epsilon_{t-2}$,. . . , ϵ_{t-q} rather than AR (Auto aggressive). In another word, the moving average model is a linear regression model of the current value of the series against current and previous (observed) white error. We can see the MA process in Figure. 2.4.

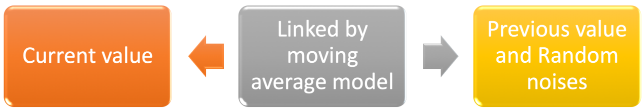

Current value
Linked by moving average model
Previous value and Random noises

Figure 2.4: Moving average process model.

Moving average process of order q is displayed as MA(q), and it calculated as:

$$Y_t = \mathtt{C} + \Phi_1 \epsilon t - 1 + \Phi_2 \epsilon t - 2 ... + \Phi_q \epsilon t - q + \epsilon_t \tag{2.10}$$

where $\mathtt{C}$ is a constant, Y_t is the forecast random variable, Φ_q is an explanatory random value, q is the number of lags (i.e. model parameter to be estimated), and ϵ_t is the white noise.

MA(q) model is identified by determining lag q, by following the characteristics of the ACF and PACF. The MA model is suitable to be used if the plots of the ACF, and PACF following this:

- ACF of an AR(p) process has a geometrical decline (tails off).
- PACF of an AR(p) process cuts off at lag q.

Autoregressive Moving Average (ARMA)

Autoregressive (AR) and moving average (MA) models can be effectively combined together to form ARMA models, the time series is assumed to have no trend and stationary. ARMA model is used often in short term load forecasting task. ARMA model with order p,q can be written as:

$$Y_t = c + \sum_{i=1}^{p} \Phi_i Y_{t-i} + \sum_{j=1}^{q} \Psi_i Y_{t-i} + \epsilon_t \tag{2.11}$$

ARMA(p,q) process is AR(p) plus MA(q). From Equation 2.11, we can see that MA(q) = ARMA(0,q), and AR(p) = ARMA(p,0). Finding appropriate values of p and q in ARMA(p,q) model can be facilitated by plotting PACF for an estimate of p, and plotting ACF for an estimate of q. Further information can be extracted by considering the same functions for the residuals of a model fitted with an initial selection of p and q.

Brockwell & Davis recommend using Akaike information criterion (AIC) for finding p and q [20]. In general, both ACF and PACF tail off in case of ARMA. Table 2.1 summarizes the behaviour of ACF and PACF in AR, MA, and ARMA.

Table 2.1: ACF and PACF behavior for AR, MA and ARMA.

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

Autoregressive Integrated Moving Average (ARIMA)

Autoregressive Integrated Moving Average (ARIMA) is a combination of differencing with autoregression and moving average model, ARIMA(p,d,q) [18]. Where p denoted the AR order, d is the integration order, and q is the MA order.

The ARIMA components are summarized as:

- **Autoregressive (AR)**: A model uses the dependent relationship between an observation and some number of lagged observations (i.e., p).
- **Integrated (I)**: The use of subtractions (i.e.,d) of raw observations in order to make the time series stationary.
- **Moving Average (MA)**: A model that uses the dependency between a previous value (observed value) and a white error from a moving average model applied to lagged observations.

If the actual data Y is not stationary, we should perform a stationary process by computing the first order difference (d) as following:

$$\triangle Y_t = Y_t - Y_{t-1} \tag{2.12}$$

Sometimes, we need to find the second order difference also:

$$\triangle_t^Y = Y_t - 2Y_{t-1} - Y_{t-2} \tag{2.13}$$

If the time series is stationary (i.e., $d = 0$), then we attempt to fit an ARMA model (i.e., ARIMA $(p, 0, q)$ to it.

Seasonal Autoregressive Integrated Moving Average (SARIMA)

A seasonal ARIMA model is an extension of ARIMA model to use when the time series exhibits sign of seasonality. SARIMA model is similar to ARIMA model, but it includes more terms to adjust for the seasonal components. The SARIMA model is formed by including a seasonal term to the ARIMA model as: SARIMA (p, d, q),$(P, D, Q)_m$ where p is the number of the autoregressive, d is the degree of differencing, and q is the number of moving average terms, and (P, D, Q) are refer to auto regressive, differencing, and moving average term for the seasonal part of the ARIMA model, and m is the number of periods in each season.

In order to select between different ARIMA models, Akaike's Information Criterion (AIC) is usually used. AIC is a score we use, to determine the best model fit for given data set. AIC can be computed as define in Equation 2.14:

$$AIC = -2\ln(L) + 2(K) \tag{2.14}$$

where L is the likelihood of data, and K is the number of parameters (e.g., p,q).

The AIC values determine the best combination of parameters, i.e., the combination of parameters that give the lowest AIC score.

2.3.4 Exponential smoothing (ES)

The exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation, the higher the associated weight [12].

In the following, I will introduce different exponential smoothing methods which are used in the implementation in this thesis. These methods are shown in Figure 2.5. Furthermore, Figure 2.5 indicates which suitable prediction method could be used according to different properties of the time series dataset.

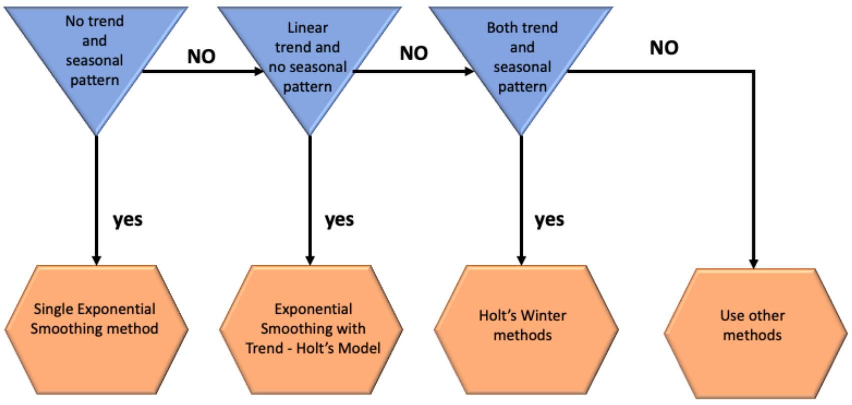

No trend and seasonal pattern
NO
Linear trend and no seasonal pattern
NO
Both trend and seasonal pattern
NO
yes
yes
yes
Single Exponential Smoothing method
Exponential Smoothing with Trend - Holt's Model
Holt's Winter methods
Use other methods

Figure 2.5: Exponential smoothing: Data vs method.

Simple exponential smoothing (SES)

The simple exponential smoothing method is equivalent to MA, except that each series value has a different weight. The weights decrease exponentially as observations come further from the past. The smallest weights are associated with the oldest observations [21]. SES is suitable method for time series when there is no trend or seasonal pattern in the data, but the mean (or level) of the time series Y_t changes slowly over time.

Simple exponential smoothing can be written as:

$$\hat{Y}_{t+1} = \alpha Y_t + \alpha(1-\alpha)Y_{t-1} \tag{2.15}$$

where $\hat{Y}_{t+1}$ is the forecast value for period $t+1$ on the time t, Y_t is the actual value in period t, α is the smoothing constant or the weight assigned to historical values $(0 \leq \alpha \leq 1)$.

By continuing to substitute previous forecasting values back to the starting point of the data as:

$$\hat{Y}_{t+1} = \alpha Y_t + \alpha(1-\alpha)Y_t +,,, + \alpha(1-\alpha)^{t-1}Y_1 + \alpha(1-\alpha)^t Y_0 \tag{2.16}$$

Then Equation 2.16 can be re-written in a compact form as:

$$\hat{Y}_{t+1} = \alpha \sum_{K=0}^{t-1} (1-\alpha)^k Y_{t-k} + (1-\alpha)^t Y_0 \tag{2.17}$$

From Equation 2.16, we see obviously that SES in this equation is exponentially declining.

The model's capability to adapt to the time series fluctuations depends on α. A greater α will be able to follow the series behaviour well while a low α will result in a more smoothed signal. Equation 2.18 defines the smoothing operation [22]:

$$\ell_t = \alpha Y_t + (1-\alpha)\ell_{t-1} \tag{2.18}$$

And Equation 2.22 defines the forecasting operation:

$$\hat{y}_{t+h} \mid t = \ell_t, \tag{2.19}$$

where ℓ_t denotes the level (or the smoothed value) of the time series at time t, and $h \in \{1, 2, 3\}$ is the number of forecast steps at time t.

In general, it is assumed that the first forecast value is equal to the first time series value at the beginning of the forecasting process, i.e., $\ell_t = Y_t$. So the modification starts from the second observation value of the time series. The prediction at time $t + 1$ is equal to the exponential smoothing for the last observed value $(Y_{t+1} = \ell_t)$. This strategy is called one step ahead.

This method is mathematically simple and it is flexible to apply for forecasting. The algorithm needs recent observations, the last prediction value and α in order to make forecasting. The disadvantage of this methods is the difficulty in finding an appropriate value for α [21].

Holt's exponential smoothing (HES)

Holt's exponential smoothing is an extension of the simple exponential smoothing method to time series that display a linear trend. The forecasting by the SES method on such time series will give overestimated or underestimated values. The Holt's exponential smoothing model was proposed to avoid this systematic error [23].

The HES model structure is similar to the SES model structure. It defined by Equations 2.20 and 2.21. But, in addition to the parameter α, which is used to soften the level component, the Holt's exponential smoothing algorithm uses also another smoothing constant (β) for modelling the trend component in the time series.

Equation 2.20 is the level operation:

$$\ell_t = \alpha Y_t + (1-\alpha)(\ell_{t-1+b_{t-1}}) \tag{2.20}$$

Equation 2.21 refers to the trend component:

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \tag{2.21}$$

Both values of the smoothing constants α and β are determined in the range of $[0,1]$, such as $(0 \leq \alpha \leq 1)$ and $(0 \leq \beta \leq 1)$. Equations 2.20 and 2.21 estimate the level of the series at time t, and trend components, respectively. The previous estimates will be modified when a new observation is calculated.

Equation 2.22 refers to the forecasting operation:

$$\hat{y}_{t+h} \mid t = \ell_t + hb_t \tag{2.22}$$

Equation 2.22, y_{t+h} denotes the forecasting value of Y at time $t+h$, and h represents the prediction horizon.

The algorithm recurrence relation is implemented by providing its initial values. In general, $L_1 = Y_1$ and $b_1 = (Y_2 - Y_1)$ are widely accepted [10]. The initial values do not affect the forecasting, because the method depends on the self-learning concept. But, this fact can not apply to the smoothing constant, which is difficult to set and bad choices may degrade the forecasting performance of the algorithm. To perform the algorithm recurrence relation, first, we need to compute the initial values. The most used approach in the literature is to assume the $L_1 = Y_1$ and $b_1 = (Y_2 - Y_1)$. The initial values do not affect the forecasting when the method is based on the self-learning concept. But, this fact can not apply to the smoothing constant, which is difficult to select. Bad forecasting accuracy can be obtained with bad selection of the smoothing parameters.

Holt-winters seasonal exponential smoothing method

The Holt-winters seasonal exponential smoothing method consists of three-smoothing equations: level equation (ℓ_t), trend equation (b_t), and the seasonal component (s_t). There are two versions of the single-seasonal Holt-winters method that differ from the seasonal components [24]: (i) the Additive version, and (ii) the Multiplicative.

The choice of methods depends on the seasonal pattern of the time series. The additive method is used when any big change in the highest and lowest-demand value is noticed, i.e., the seasonal variation is almost constant through the series. The multiplicative method is used when the trend and seasonal variation increases or decreases in magnitude overtime. In other words, the seasonal version is proportional to the level of time series.

Additive Holt–winter method (AHW)

The dditive Holt–winter method is implemented according to the following equation:

$$Y_t = T_t + S_t + R_t \tag{2.23}$$

The three equations used in implementing the algorithm are:

- Equation 2.24 refers to the level equation:

$$\ell_t = \alpha(Y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.24}$$

- Equation 2.25 is the trend equation:

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \tag{2.25}$$

- Equation 2.26 is the seasonal equation

$$s_t = \gamma(Y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{2.26}$$

- The the forecasting is:

$$\hat{y}_{t+h} \mid t = \ell_t + hb_t + s_{t-m+h} \tag{2.27}$$

In these equations, γ, β, and α are smoothing parameters in the range $[0, 1]$, and m denotes the frequency of the seasonality. y_{t+h} denotes the forecasting value of the Y at time $t + h$, h represents the prediction horizon.

Similar to simple exponential smoothing and Holt's exponential smoothing methods, the additive Holt-winters method receives the time series as an input, and then apply the three additive equations recursively. We have to start at time in the past, where the values of b,s, and ℓ are already estimated. The initialization of trend and and trend in the same period m gives this approximation. Then, the level can be computed from the average of the first station as in Equation 2.28:

$$\ell_m = \frac{1}{m}(Y_1 + Y_2 + \cdots + Y_m) \tag{2.28}$$

The trend can be initialized with using two complete stations, as following:

$$b_m = \frac{1}{m}(\frac{Y_{m+1} - Y_1}{m} + \frac{Y_{m+2} - Y_2}{m} + \cdots + \frac{Y_{m+m} - Y_m}{m}) \tag{2.29}$$

The seasonal indexes are defined by Equation 2.30:

$$s_1 = Y_1 - b_m, s_2 = Y_2 - b_m, ...s_m = Y_m - b_m \tag{2.30}$$

Multiplicative Holt-winters method (MHW)

The Holt-Winter's multiplicative seasonal method is implemented through Equation 2.31:

$$Y_t = T_t \times S_t \times R_t \tag{2.31}$$

The component form of the multiplicative seasonal methods is defined according to the following equations:

- Equation 2.32 is the level equation:

$$\ell_t = \alpha \frac{Y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.32}$$

- Equation 2.33 is the trend equation:

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \tag{2.33}$$

- Equation 2.34 is the seasonal equation:

$$s_t = \gamma \frac{Y_t}{\ell_t} + (1-\gamma)s_{t-m} \tag{2.34}$$

- Equation 2.35 is the forecasting equation:

$$\hat{y}_{t+h} \mid t = (\ell_t + hb_t) \times s_{t-m+h} \tag{2.35}$$

Equation 2.25 in additive Holt's winter is equal to Equation 2.33 of multiplicative Holt's winter. The differences between such methods are in the other equations, where the seasonal indexes are multiplied and divided instead of summed and subtracted in the additive model. Those variables are initialized by using the same equation of additive Holt's winter, but the seasonal indexes are calculated according to equation 2.36:

$$s_1 = \frac{Y_1}{b_m}, s_2 = \frac{Y_2}{b_m}, \ldots, s_m = \frac{Y_m}{b_m} \tag{2.36}$$

The proper selection of the Holt's winter model is related to the seasonal fluctuation in the time series, regardless of the trend component existence in time series. In these models (Additive and Multiplicative Holt's winter), if $\gamma = 0$, this does not mean at time series is empty of seasonality, but this means seasonal rates have been initialized with values that do not need to be fixed along with the prediction.

2.4 Machine learning methods

This section introduces the basic terminology used in machine learning (ML) and describes different machine learning methods to the time series forecasting, such as linear regression (LR), multi-layer perception (MLP), recurrent neural network (RNN), long short term memory model (LSTM), convolutional neural network (CNN), support vector machine (SVM), and K-nearest neighbor (K-NN).

Different from statistical methods, machine learning methods do not need prior knowledge of the distribution of data to describe the data properties. Furthermore, ML approaches do not depend directly on parameters to model the phenomenon's behaviour. Also, these methods are easy to modify and show reliable performance even if we applied them on complex and highly nonlinear time series [25].

Machine learning, as a definition, aims at turning data to information [26] by using an adaptive model, to help computers to learn to find the pattern in the data based on examples [27]. These adaptive models have a learning capability that makes them able to improve performance over time. ML is like a function that can find and learn the relationship between descriptive features and a target feature in a dataset. This allows the algorithm to make a predictions or decision by transforming a set of inputs X into output (Y) [14].

The machine learning methods have been applied successfully in many problems, for instance language translation [28], face recognition [29], predict the amount energy used in building [30].

ML problems can be divided into three categories based on the type of data that the system learns from: (i) Supervised learning, (ii) Unsupervised learning, and (ii) Reinforcement learning.

- **Supervised learning:** In this case, a labeled dataset is used to define models. Based on the traninig dataset, the model can learn to generalize and make a correct prediction.
- **Unsupervised learning:** In this case, the data are not labeled but the model tries to draw inferences from datasets. Unsupervised learning method is used to find hidden patterns for exploratory analysis or to find similarities in data for clustering.
- **Reinforcement** This is somewhere between supervised and unsupervised learning. The algorithm is informed when the answer is wrong but no information is provided on how to correct it. It has to explore and try out different possibilities until it works out how to get the answer right. Reinforcement learning is sometimes called learning with a critic because of this monitor that scores the answer but does not suggest improvements.

ML has been used to address different types of problems. In particular, supervised learning can be divided into two types [31]:

- **Regression:** Try to model the relationship between inputs and output, meaning that we try to map the input variable to some continual function.

- **Classification:** In this case, we are trying to map input variables into discrete categories.

The ML models explored in this thesis are explained in more detail in the following sections. The detailed description contains concepts such as neurons, layers, activation functions, learning rate, and much more. Those concepts are often referred to as hyper-parameters and are parameters that need to be set before training a model.

2.4.1 Artificial Neuron (AN)

An artificial neuron is a connection point in an artificial neural network. The definition of an artificial neuron relies on the definition of inputs, weights, bias, as well as summation, and activation functions. Neuron in the input layer receives signals from the input, while a neuron in the other layers receives signals from other neurons. The output of a neuron in the output layer is an output of the model, while it is an input to other neurons if the neuron in the other layers. A neuron output value is determined from the sum of the weighted inputs passed through an activation function [27].

Figure 2.6 illustrates an artificial neuron. The output Y is defined by taking the sum of the weighted inputs that go later through the activation function.

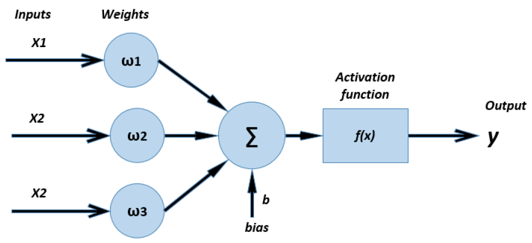


Figure 2.6: Architecture and components of an artificial neuron.

Each neuron (n_i) has the inputs x_1,x_2,x_3,. . . ,x_1 and outputs y. It sums the input values multiplied with the weight plus the bias. The sum of weighted inputs (O_i) are obtained by Equation (2.37), where N is the total number of input and weight, w is the weights, i represents the neuron, and b is the bias.

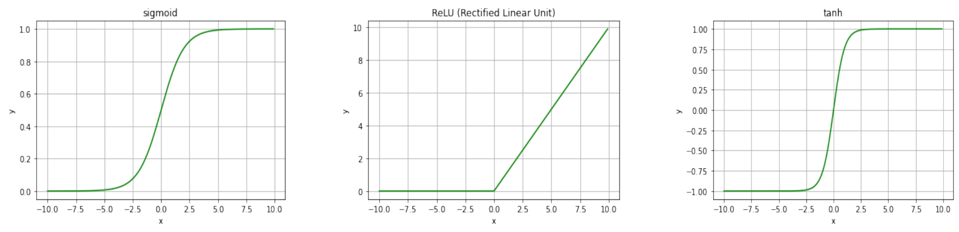
$$o_i = \sum_{j=1}^{N} \mathcal{W}_{ij}\mathcal{X}_j - b_i \tag{2.37}$$

Then, the output y_i can be calculated through pass the sum of weighted inputs (O_i) into an activation function f.

Therefore, the output y_i is given by the equation:

$$y_i = f_{(o_i)} = f(\sum_{j=1}^{N} \mathcal{W}_{ij}\mathcal{X}_j - b_i) \tag{2.38}$$

The activation function f can determine the output of the neuron. Many types of activation functions have been proposed for different kind of problems, but here we define the most three common activation functions: $Sigmoid$, Rectified linear unit $(ReLU)$, and hyperbolic tangent $(tanH)$. Those functions are illustrated in Figure 2.7.



(a) sigmoid **(b)** Relu **(c)** Tanh

Figure 2.7: Three common activation functions

The sigmoid function (Equation 2.39) outputs values between $(0$ and $1)$. That can be explained as a probability in classifications [32]. The $ReLU$ is the most popular activation function in deep learning. The $ReLU$ is bounded between 0 and positive number as shown in Figure 2.7b. The benefit of $ReLU$ relies on its computational costs and on its fast convergence properties. The $ReLU$ is defined as in Equation 2.40.

The $tanH$ activation function gives values between -1 and 1, and it's shape looks as $Sigmoid$ function. The output of The $tanH$ function is defined by Equation 2.41.

$$f_{sigmoid}(z) = \frac{1}{1+e^{-z}} \tag{2.39}$$

$$f_{relu}(z) = max(0, z) \tag{2.40}$$

$$f_{tanh}(z) = \frac{e^{z} - e^{-z}}{e^{z} - e^{-z}} \tag{2.41}$$

Artificial Neural Networks (ANN)

Artificial neural networks consist of an input layer, hidden layer, and the output layer [33] as shown in Figure 2.8. The artificial neuron is the basis of the building of (ANN). In

Figure 2.8, each layer has multiple neurons and each neuron in one layer is connected with others in the next layer. This called a fully connected network or multi-layer perceptron (MLP).

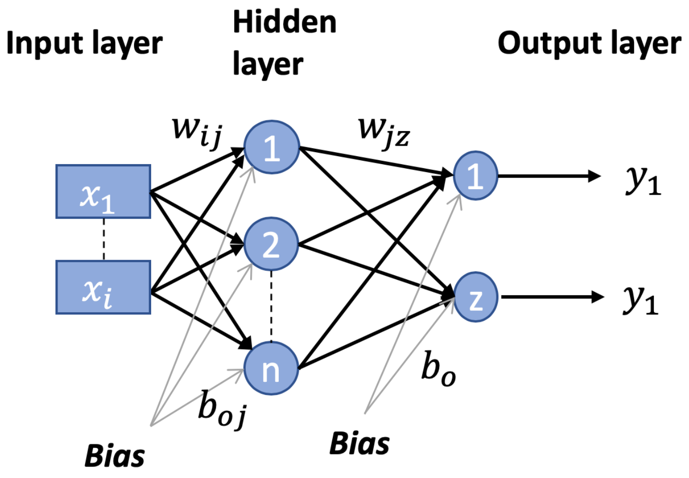


Figure 2.8: Structure of MLP with a single hidden layer.

MLP is classified as a feed-forward network because the neural network forwards (pass) the data from the input layer to the output layer through the hidden layer. There is only one direction from the input to the output. The parameters (weights, biases) are initialized randomly, then parameters are updated to improve the performance of the model. MLP uses a variety of learning techniques, such as back-propagation where the output values are compared with the correct answer in order to compute the value of the predefined error-function. Then the error is fed back through the neural network [34].

2.4.2 Linear regression (LR)

LR is one of the most popular supervised methods. LR is a statistical method that can be used to make a prediction for real or numeric variables.

LR aims to establish a linear relationship between a dependent (x) and independent (y) variable. The linear regression finds how the value of the dependent variable is changing according to the independent variable as shown in Figure 2.9.

Linear regression can be represented mathematically as:

$$y = a_0 + a_1 x + \epsilon \tag{2.42}$$

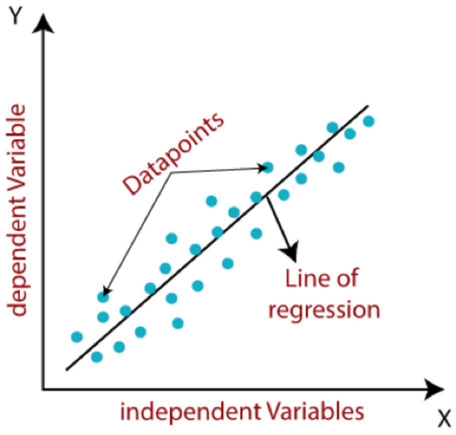


Figure 2.9: Linear regression.

where y represents a dependent variable (target variable), x is the independent variable (predictor Variable), a_0 intercept of the line (it gives an additional degree of freedom), a_1 is linear regression coefficient (scale factor to each input value)and and ϵ is a random error.

There are two type of linear regression:

- **Simple Linear Regression:** In this case, when single independent variable use to predict the value of a numerical variable.
- **Multiple Linear regression:** In this case, when there are more than one independent variable used to predict the value of a numerical dependent variable.

2.4.3 Recurrent neural network (RNN)

The recurrent neural network (RNN) is a type of artificial neural network, designed for capturing information from sequences and time series data. RNN is different from ANN, as the connections between neurons form a cycle. In addition, signals are able to move in different directions as shown in Figure 2.10. Furthermore, the recurrent layer is composed of a memory cell that is used repeatedly to compute the output. RNN has been applied in many contexts where the sequences in the data are an important feature. Examples of applications of RNNS include sequence transformation [35], language modelling [36, 37], speech recognition [38], and time series forecasting [39].

The standard RNN model can be formalized as we show in Figure 2.11. The RNN model adds a hidden state $(h_0, h_1, \ldots, h_T)$ that is generated by the sequential information on of time series and produces a sequence of outputs $(Y_0, Y_1, ..., Y_T)$. In Figure 2.11, x_t is the input vector at time t, w, u, v are the weights of the transition hidden state, hidden layer, and the output layer, respectively. h_t is the hidden state at time (t), and it is the input vector and the previous hidden state. h_t is defined as:

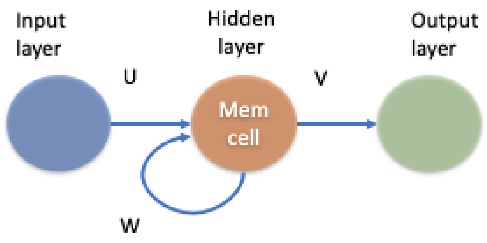


Figure 2.10: Illustration of an recurrent neural network.

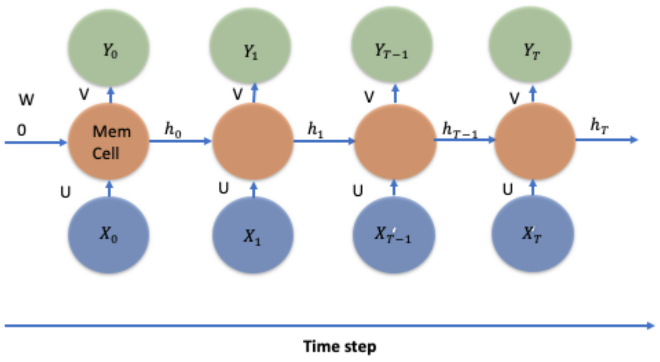


Figure 2.11: A unfolding architecture of an recurrent neural network.

$$h_t = f(Ux_t + Whx_{t-1}) \tag{2.43}$$

where f is the activate function.

There are many activation functions which can be used. The three activation functions that are most used, are introduced in Section 2.4.1. Generally, the RNN algorithm uses an activation function $tanH$, because RNNs have a tendancy to have unstable gradients. Then, the gradients can vanish during the training or they can explode. This can happen specially if we use $ReLU$ activation function which in non saturating. While if we use $tanH$, it will saturate.

Y_0 is the output at time t and can be computed as shown in Equation 2.44:

$$Y_t = f(Vh_t) \tag{2.44}$$

The RNN works fine with short-term dependencies, but not with long-term dependencies due to the vanishing or exploding gradient [40, 41].

This can happen when the gradient of the activation function calculated through the back-propagation becomes very small. Here, the weights are multiplied by itself many times which leads to zero or infinity. This means that the error becomes zero in the first layer of network, which makes train those layers take a longer time than the next ones.

One way of getting around this is by using other variants of RNNs, which can deal with the vanishing gradient problem. Some examples include Long short term memory (LSTM) and gated recurrent unit (GRU). In this thesis, the LSTM will be used.

2.4.4 Long short term memory (LSTM)

LSTM is a type of RNN models, which was designed to avoid the long-term dependency problem [40, 41]. LSTM is one of the solutions that can be used to address the vanishing or exploding gradient problem by exploring the use of memory cells [41]. The memory cell of the LSTM consists of four units as shown in Figure 2.12: an input gate, an output gate, a forget gate, and a self recurrent neurons.

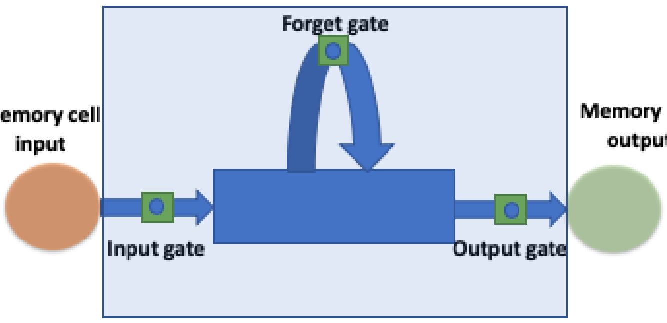


Figure 2.12: A memory cell of an LSTM.

The purpose of gates is to control the interaction between the memory cells and their neighbouring cells. In other words, the gates regulate the flow of information into and out of the cell. The input gate can control the input signal if it can modify the state of the memory cell. The output gate can control the state of memory cell if it can modify the state of other memory cell. The forget gate can control if the model wants to remember or forget the previous state [42].

The architecture of an LSTM and memory cell is shown in Figure 2.13 [43].

In the Figure 2.13, x_t is the input vector to the memory cell at time t, h_t is the output to the memory cell at time t, i_t is the value of the input gate at time t, f_t the forget gate at time t, o_t is the output gate at at time t, $\tilde{C}_t$ are the value of candidate state of the memory cell at time t, and C_t is the state of the memory cell at time t. The C_t is adjusted by the gates (i.e., regulators).

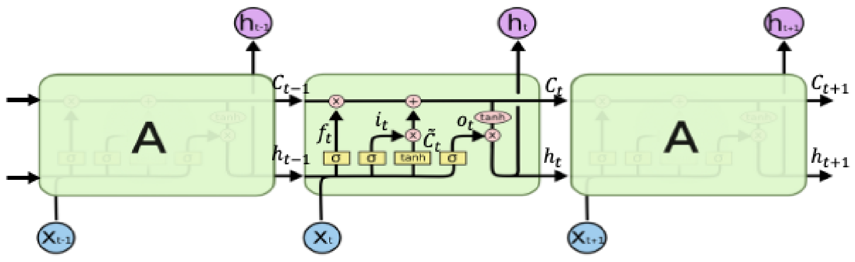


Figure 2.13: Architecture components of the LSTM and memory cell.

As shown in Figure 2.13, there is the line between the C_t and C_{t-1}. The line has the $+$ and x operations, which refer to the add and remove operations, respectively. Those operations can remove or add information in the current state cell state. The line shows how the cell state can be effected through the LSTM. In the following, I will explain briefly how easch gate works. My explanation is inspired by the explanation of LSTM in [43, 44].

The **Forget Gate** f_t determines what kind of information can be forwarded from the previous cell state. Equation 2.45 shows how it can be defined.

$$f_t = \sigma(W_f * [h_{t-1}, X_t] + b_f) \tag{2.45}$$

The forget gate uses the output form the previous cell h_t and the input of current cell x_t multiply the weight of the forget cell W_f plus the bias of forget gate b_f. The results of them go through sigmoid activation function σ to get the final resulting of vector f_t, which consists of the number between 0 and 1.

The **Input Gate layer** i_t is used to determine how much the candidate solution can add to the cell state. i_t can be defined by Equation 2.46.

$$i_t = \sigma(W_i * [h_{t-1}, X_t] + b_i) \tag{2.46}$$

The input gate uses the output form the previous cell h_t and the input of current cell x_t multiply with the weight of the input cell W_i plus the bias of input gate b_i. The results of them go through sigmoid activation function σ to get the final resulting of vector i_t, which consists of the number between 0 and 1.

The output of i_t and the output cell state from $\tilde{C}_t$ (which generate a candidate cell state) are added as shown in Figure 2.13. The cell state is calculated from Equation 2.47, and uses the $\tanh$ activation function.

$$\tilde{C}_t = \tanh(W_c * [h_{t-1}, X_t] + b_c) \tag{2.47}$$

From the forget gate plus, the input gate can calculate the new cell stat C_t. The cell state is determined by Equation 2.48

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.48}$$

The **Output Gate layer** o_t is the final step in the LSTM architecture, which determines whether the LSTM cell output is going to the next layer or an output of network. The o_t can be calculated by Equation 2.49. It uses the same inputs of the (input and forget gate)X_t, h_{t-1} and uses its own weights and bias (W_o, b_o). The results of them go through sigmoid activation function σ to get the final resulting of vector o_t, which consists of the number between 0 and 1.

$$o_t = \sigma(W_o * [h_{t-1}, X_t] + b_o) \tag{2.49}$$

The final output h_t from the LSTM is defined by Equation 2.50. h_t is computed by multiplying the output gate O_t with the result of $tanh$ activation function applied to the current cell C_t

$$h_t = o_t * \tanh(C_t) \tag{2.50}$$

2.4.5 Convolution neural network (CNN)

A convolutional neural network (ConvNet, or CNN) is a version of multilayer perceptrons which are usually fully connected neural networks, where is, each neuron in one layer is fully connected to all neurons in the next layer. CNNs have been used in several applications, such as images recognition, image classifications, objects defections, face recognizing, ..., etc[1]. CNN can also be used for 1D-data such as time-series as the electricity consumption data which is used in the thesis, used for 2D, 3D data such as image and speech signals, or even in 4D-data such as videos [45].

CNN has become popular due to its ability to automatically extract important features from input data. One of the motivations of using CNN is that it reduces computation requirements

[1]Stanford University, Convolutional Neural Networks (CNNs / ConvNets), `https://cs231n.github.io/convolutional-networks/` (As of May 2020).

due to weight sharing. CNNs have an input layer, an output layer, and hidden layers. The hidden layers usually consist of convolutional layers, $ReLU$ layers, pooling layers, and fully connected layers.

A classic CNN architecture would look something like this:

Input $->$ Convolution $->$ ReLU $->$ Convolution $->$ ReLU $->$ Pooling $->$ ReLU $->$ Convolution $->$ ReLU $->$ Pooling $->$ Fully Connected.

Figure 2.14 shows an example of a classic CNN architecture with two convolutions, pooling layers, fattening, and a fully-connected network .

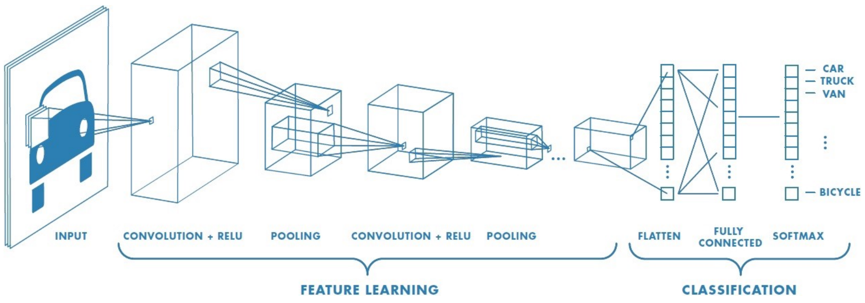


Figure 2.14: Example of classic CNN Architecture [1].

Convolution

A convolution is an integral that measures the amount of overlap of two functions, as one is shifted over the other function. This is a way of mixing two functions by multiplying them.

In a convolution layer, the input data is convoluted with a filter (i.e., filter detector) to get a feature map. The feature detectors learn which features to look for in the training process. Figure 2.15 illustrates the convolutional operation between the input data H and the filter F. The mathematical formula for a 2D convolution operation is given as:

$$G(i,j) = (H \times F)(i,j) = \sum_{i} \sum_{j} H(m,n)F(i-m,j-n) \tag{2.51}$$

where G is the output, i, j are indexes and m, n are the number of array elements in each dimension. Then, the feature map after this process will be smaller than the original input image as shown in Figure 2.15.

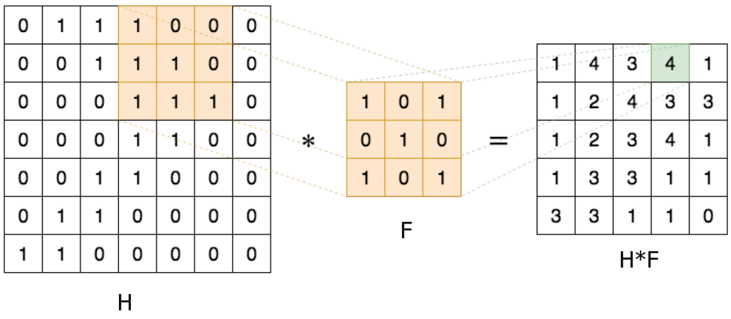


Figure 2.15: Operation of convolution CNN.

2.4.6 Support Vector Machine (SVM)

The support vector machine (SVM) is a supervised machine learning algorithm, commonly used for both classification, regression problem, as well as time series forecasting. The SVM technique is based on statistical learning principles.

The SVM can be used to solve linear and non-linear problems. It is based on the idea of finding the best line or decision boundary that can divide the n-dimensional space into two regions (two classes). This line is called hyperplane. Support vectors are the data points nearest to the hyperplane. The position of hyperplane will be changed depending on the supporting vectors being considered. Because of that, support vectors can be considered the critical elements of the training dataset. In Figure 2.16, the SVM model construction process is illustrated.

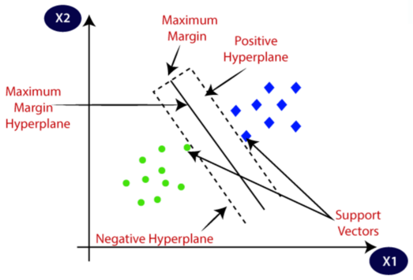


Figure 2.16: Basic components of the SVM model construction process.

There are two types of SVM:

- **Linear Support vector machine**: In this case, the data can be classified into two types, and a straight line is used to separate both classes. Then, the data is called linearly separable, and the SVM, which classifies such data, is called a linear SVM.

- **Non-linear Support vector machine**

  In this case, the data is can not be classified into two types by using only a straight line. In another word, the data is not linearly separable, and the SVM, which classifies such data, is called a non-linear SVM.

The best hyperplane can be determined to find the margin value. Margin is the distance between the support vector and the hyperplane (see Figure 2.16). The rationale behind SVM is to maximize this margin.

Even though SVM is originally developed for classification problems, but it is also used for numeric prediction. By using of the kernel function, mapping can be done in SMV algorithm [46]. The kernel function can be represented as:

$$f(x, y) = \Phi(x).\Phi(y) \tag{2.52}$$

where x and y are vectors in the input space.

In regression problems, SVM is used to find a linear model of the the form as:

$$y(x) = w^T \varphi(x) + c \tag{2.53}$$

where $\varphi(x)$ is the space transformation function (i.e., kernel function), w and c are parameters [2]. The task in the linear regression is to minimize the regulated error function. In SVM, we use ϵ -insensitive error function to obtain sparse solution, i.e., if the absolute distance between the the target and the predict is less than ϵ, then the error is zero [2]. This makes a tube with ϵ width around the target function as shown in Figure 2.17.

The regulated error function is written as [2]:

$$C = \sum_{i=1}^{N} (\xi_i + \hat{\xi_i}) + \frac{1}{2} \parallel w \parallel^2 \tag{2.54}$$

The regulated function in Equation 2.54 must be minimized with following constraints:

$$\xi \quad and \quad \hat{\xi_i} \geq 0, \tag{2.55}$$

$$y(x_i) + \epsilon + \xi_i \geq t_i, \tag{2.56}$$

$$y(x_i) - \epsilon - \hat{\xi_i} \geq t_i \tag{2.57}$$

The regulated error function can be minimized by using Lagrange multipliers [2]. Then, the forecasting function is written as:

$$y(x) = \sum_{i=1}^{N} (a_i + \hat{a_i}) f(x, x_i) + c \tag{2.58}$$

where f is the kernel function, a_i and $\hat{a}_i \geq 0$ are Lagrange multipliers, and $x_1, ..x_N$ are support vectors that lie outside the tube.

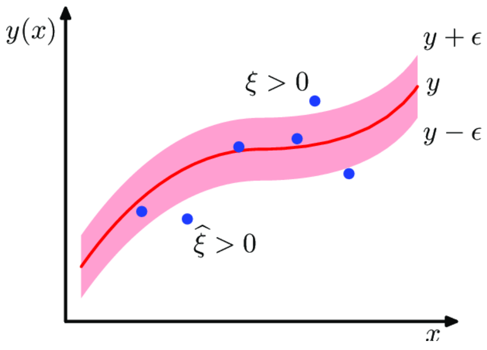


Figure 2.17: SVM regression model with ϵ -insensitive [2].

Kernel functions

There are many kernel functions that can be used in SVM models. In this thesis work, I used three popular kernel functions which are supported by the Weka tool (Weka is used in the implementation of SVM models for time series forecasting) [47]. These functions are: i) radial basis kernel, ii) Pearson VII function based kernel, and iii) polynomial kernel.

- Radial basis kernel function can be written as:

$$f(x,y) = e^{\left(-\frac{\|x-y\|}{2\sigma}\right)^2} \tag{2.59}$$

  where x and y are vectors in the input space, σ is a free parameter which is set by the user [47].

- Pearson VII kernel function is written as [46]:

$$f(x,y) = \frac{1}{[1 + (\frac{\sqrt{\|x-y\|^2}\sqrt{2^{\frac{1}{w}}-1}}{\sigma})^2]^w} \tag{2.60}$$

  where x and y are vectors in the input space, σ and w are parameter which can be changed [47].

- polynomial kernel function is written as:

$$f(x,y) = (x^T y + c)^d \tag{2.61}$$

where x and y are vectors in the input space, c is a free parameter which is set by the user [47], and d is the dimension of new space.

2.4.7 K-Nearest Neighbors (K-NN)

K-Nearest Neighbors is a supervised machine learning algorithm. K-NN is one of the simplest method in ML [48, 49]. The K-NN can be used for both classification [50] and regression prediction problems [51]. K-NN is a non-parametric and lazy learning algorithm. This means that the model structure determined from the dataset. This will be very helpful in practice where most of the real-world datasets do not follow mathematical theoretical assumptions. The algorithm is lazy because it does not need any training data points for model generation. All training data are used in the testing phase. This makes training faster and testing phase slower and more costly.

In the forecasting problems, K-NN considers that all instances are some points in the n-dimensional space R^n. The inductive bias of the method assumes that the prediction of a new instance will be similar to the prediction of the other close instances. The Euclidean distance is generally used to calculate the distances. For a given instance we can define nearest neighbors using the Euclidean distance $d(x_i, x_j)$. The calculation of the Euclidean distance between two instances is given by:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_i))^2} \tag{2.62}$$

The target function may be both discrete-valued and real-valued, so this method can be used for both for classification and regression problems. In the regression problem, as we have, let's consider a training algorithm for continuous-valued target function [52]:

1. For each training example (x, $f(x)$), add the example to the list of training examples.
2. Given a query instance x_q to be predicted:
   - Let $x_1, ..., x_k$ denote the k instances from training examples that are nearest to x_q.
   - Then

$$f(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k} \tag{2.63}$$

One typical problem, which can arise using nearest-neighbor learning, refers to the fact that the distance between neighbours may be dominated by some irrelevant attributes. Thus, when there are many irrelevant attributes in data, the so-called curse of dimensionality become a problem. To avoid this, we can weight each attribute with different value. We can imagine this as stretching the axes in the Euclidean space: those axes which represent irrelevant (or less relevant) attributes become shorter, and those axes which represent relevant (or more relevant) attributes become longer [52].

2.5 Evaluation Measures

This section presents the evaluation measures used to assess the predictive performance of models. In the previous sections, I have introduced various useful and popular techniques for time series forecasting. Another important issue is of course implementation, i.e. to apply these methods for generating forecasts. While applying a particular model to some real or simulated time series, first the raw data is divided into two parts: the training set and test set. The observations in the training set are used for constructing the desired model. Often a small sub-part of the training set is kept for validation purpose and is known as the validation set.

Once a model is constructed, it is used for generating forecasts. The test set observations are kept for verifying how accurate the fitted model performed in forecasting these values. If necessary, an inverse transformation is applied on the forecasted values to convert them to the original scale. In order to judge the forecasting accuracy of a particular model or for evaluating and comparing different models, their relative performance on the test dataset is considered.

Due to the fundamental importance of time series forecasting in many practical situations, proper care should be taken while selecting a particular model. For this reason, proper performance measures should be selected to estimate forecast accuracy and to compare different models. These performance measures are also known as performance metrics [53]. Each of these measures is a function of the actual and predicted values of the time series. In this section, I will describe the important performance measures and their properties, which are frequently used to evaluate time series forecasting models.

The forecast error at time t is calculated as:

$$e = y_t - f_t \tag{2.64}$$

where y_t is the actual value and f_t is the predicted value. If the size of the test set is n, then the mean is:

$$\overline{y} = \frac{1}{n}\sum_{t=1}^{n} y_t. \tag{2.65}$$

The test variance is:

$$\sigma^2 = \frac{1}{n-1}\sum_{t=1}^{n}(y_t - \overline{y})^2 \tag{2.66}$$

2.5.1 Mean absolute error (MAE)

The mean absolute error is defined as [53, 54, 55, 56]:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{2.67}$$

Its properties are:

- It measures the average absolute deviation of forecasted values from original ones.
- It is also termed as the Mean Absolute Deviation (MAD).
- It shows the magnitude of overall error, occurred due to forecasting.
- In MAE, the effects of positive and negative errors do not cancel out.
- Unlike MFE, MAE does not provide any idea about the direction of errors.
- For a good forecast, the obtained MAE should be as small as possible.
- Like MFE, MAE also depends on the scale of measurement and data transformations.
- Extreme forecast errors are not penalized by MAE.

2.5.2 Mean absolute percentage error (MAPE)

This measure is given by [53, 54]:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{e_t}{y_t}\right| \times 100 \tag{2.68}$$

Its important features are:

- This measure represents the percentage of average absolute error occurred.
- It is independent of the scale of measurement, but affected by data transformation.
- It does not show the direction of error.
- MAPE does not panelize extreme deviations.
- In this measure, opposite signed errors do not offset each other.

2.5.3 Mean percentage error (MPE)

It is defined as [57]:

$$MPE = \frac{1}{n}\sum_{t=1}^{n}\left(\frac{e_t}{y_t}\right) \times 100 \tag{2.69}$$

The properties of MPE are:

- MPE represents the percentage of average error occurred, while forecasting.
- It has similar properties as MAPE, except,
- It shows the direction of error occurred.
- Opposite signed errors affect each other and cancel out.
- Thus like MFE, by obtaining a value of MPE close to zero, we cannot conclude that the corresponding model performed very well.
- It is desirable that for a good forecast the obtained MPE should be small.

2.5.4 Mean squared error (MSE)

Mathematical definition of this measure is [54, 58]:

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{2.70}$$

Its properties are:

- It is a measure of average squared deviation of predicted values.
- As here the opposite signed errors do not offset one another, MSE gives an overall idea of the error occurred during forecasting.
- It penalizes extreme errors occurred while forecasting.
- MSE emphasizes the fact that the total forecast error is in fact much affected by large individual errors, i.e. large errors are much expensive than small errors.
- MSE does not provide any idea about the direction of overall error.
- MSE is sensitive to the change of scale and data transformations.
- Although MSE is a good measure of overall forecast error, but it is not as intuitive and easily interpretable as the other measures discussed before.

2.5.5 Root mean squared error (RMSE)

Mathematically, RMSE is defined as [55, 56]:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \tag{2.71}$$

Its properties are:

- RMSE is nothing but the square root of calculated MSE.
- All the properties of MSE hold for RMSE as well.

Chapter 3

Related work

In this chapter, I will give an overview of some of the previous work on the electric load forecasting and how the forecasting models are used in different fields.

3.1 Load forecasting models

Electricity demand forecasting is a significant method that is used to balance the electricity generated and the electricity demand to increase the efficiency. This is an integral process in the structuring, planning and operation of the electrical power systems. Forecasting models help in making decisions associated with generating process and predicting electricity prices [3]. Also, these models help in improving the reliability of the network by predicting the faults and responding to the demands. Due to the fluctuation in demand of electricity which depends on different factors (e.g., weather and living style), and the increase in electricity prices during the peak's time, forecasting models become the core of the smart grids and smart cities that automatically adapt to fulfil the citizens' need, achieve sustainability, reduce emissions, among other benefits.

The horizon of the prediction can be classified into many categories. Some researchers have divided load forecasting into three categories as following [59, 60, 61, 62]: i) short-term forecasting ii) mid-term forecasting iii) long-term forecasting. While other researchers added one more category which is very short term forecasting [63, 6, 64, 65]. The period of the prediction depends on the characteristics of the available data (e.g., per minute, hourly, weekly, monthly, or annually) and the application of the forecasting model. Table. 3.1 shows the classification of the load forecasting reference to the prediction horizon, and shows potential applications [4, 5].

Table 3.1: Demand forecasting classification.

Forecasting mode	**Duration**	**Applications**
Very short-term forecasting	Few seconds to few minutes	Generation, contingency analysis for system security and distribution schedule
Short-term forecasting	From an hour to few days	Maintenancescheduling, spinning reserve allocation, operational planning and unit commitment.
Mid-term forecasting	Few days to few weeks	Planning for seasonal peak (summer or winter)
Long-term forecasting	Few months to few years	Generation growth planning

Although, there are many load forecasting models, which are developed with different forecasting methods, still building an appropriate and accurate forecasting model for a particular electricity network is not an easy task. An accurate load forecasting model can not be generalized for all electricity data patterns [59, 66, 7]. In general, the load forecasting methods are divided into two forecasting methods: i) time series forecasting methods (as used in this thesis), and ii) multi-factor forecasting methods [67]. The time series forecasting methods depend only on the historical time series data, while multi-factor methods take the influence of other factors also (i.e., through finding causal relationships) in forecasting values.

3.2 Time series forecasting models

Most of the time series forecasting models are built by statistical approaches and artificial intelligence algorithms [68]. The selection of the best method depends on two things: i) the available historical data, and ii) and the application the models is build for [64, 8].

In the past, time series forecasting models were almost limited to only traditional statistical methods. But forecasting models based on machine learning methods are becoming more popular in nowadays with the progress in the technology we are experiencing [65].

3.2.1 Statistical models

Statistical methods have been used to develop time series forecasting models massively from very long time [69]. These models are divided to: i) Box-Jenkins basic models, and exponential smoothing models [69, 70, 71, 72].

Box-Jenkins models are:

- Autoregressive (AR) Models which have used for long time in many different fields including the electric load forecasting [3, 7, 8].

- Autoregressive Moving Average (ARMA) Model which has been proposed by George Box in 1970s [7, 69], is used massively in electric load forecasting [3, 73].
- Autoregressive Integrated Moving Average (ARIMA) Model which is proposed by George Box to include the condition of non-stationary also. ARIMA models have been used much in the electric load forecasting [3, 59, 8, 9, 74, 75].

Exponential smoothing (ES) methods are also used to build load forecasting models such as in [72].

3.2.2 Machine learning models

Machine learning methods give an an alternative mechanism to do time series forecasting. Machine learning methods are applied in various applications, and for different purposes. In time series load forecasting, machine learning models have been applied massively in the last two decades [9].

Artificial Neural Network (ANN) Algorithm which is composed of many neurons that are changing their dynamic state response with respect to external inputs [63, 7, 9]. The commonly employed ANN algorithms for electric load forecasting are [3, 63, 65, 73, 76, 77]:

- feed-forward (FF) neural networks.
- NARX (nonlinear autoregressive with exogenous inputs) neural networks.
- back-propagation (BP) neural networks.
- radial basis function (RBF) neural networks.
- random neural networks.
- recurrent neural networks (RNN).
- self-organizing competitive neural networks.

ANN models can be classified into two groups forecasting period, according to [3]:

- ANN algorithms which that have only one output node, they used to predict one point (i.e., the next hour's load, next day's peak load, or next day's total load).
- ANNs algorithms which that have several output nodes, they used to predict a sequence of points (i.g., hourly loads for the next week).

Support vector machines (SVMs) are used also for the forecasting purposes, and solving of the time series forecasting problems in the last two decades [3, 7, 9, 66]. Specifically in electric load forecasting, SVM models have been widely used. SVM model is used in [78] to forecast the next day's electricity load of public buildings, while in [79] it is used in mid-term load forecasting for a whole city. In [80], an SVM model based on the particle swarm optimization is developed for short term load forecasting.

It already exist various forecasting models with different time frame, inputs, outputs, scale, data sample size and error type [7]. Based on the results found by [7], the majority of the regression models are used for long-term forecasting with big time scale, while ANN is more used for short-term prediction with fine time scale. Also, SVM models are mainly applied for a short and very short term prediction.

3.3 Load forecasting models for energy management

The load forecasting is the core of the energy management systems. Table. 3.1 summarizes some of the potential applications of electric load forecasting. Energy management is a hot research topic due to its advantages in related to design sustainable systems and reduce the CO2 emissions. Energy management consists in choosing among a set of sources able to produce energy that will give energy to a set of loads by minimising costs and losses [81].

Long-term load forecasting is used for planning the generation process, and finding new sources to meet the increasing load demand [82]. While, short and mid term load forecasting are used more in energy management systems at different levels (e.g., micro-grid level and building level) [83].

One of the emergent applications of the short-term load forecasting is smart and sustainable cities. In [4], the author has discussed the problem of applying the electricity demand forecasting for a smart city. In [84], the authors have integrated the short term load forecasting in the smart city framework. They have designed a full architecture of the smart city framework by including the short-term time load forecasting at buildings level.

Energy management systems

Designing smart energy management systems at different levels is an important step to use the energy efficiently. At the micro-grid level, short-term load forecasting can be used in designing an intelligent energy management system which aims to exploit the sources (i.e., renewable sources) to cover the load demand, and minimize the amount of energy required from the grid as in [85]. Demand side managements can be at different levels [86]. In [87], the authors presented a home energy management system. The residents can schedule their power usage in the home by themselves for the purpose of reducing electricity expense.

Chapter 4

Methodology

The main objective of this thesis is to build a time series forecasting model which performs best on the electricity consumption dataset and explore methods to integrate the forecasting model in energy management systems. In order to achieve this objective, we have to explore different forecasting methods (i.e., statistical and machine learning methods) to find the best one that performs best on our energy consumption dataset. In this chapter, the adopted methodology of the project, the dataset considered in our study, data analysis, analyses concerning the existing correlations among consumption data and other data such as weather data, and implementations details are described.

4.1 Load forecasting model

The load forecasting model is based on applying the time series prediction process, which is divided into seven steps, as shown in Figure 4.1. The steps are:

1. **Collecting data:** Collecting the historical data is essential to do prediction and it is the first step in building the forecasting model. We can use historical or generate synthetic data. In this project, I have used a real energy consumption dataset. An overview of the used data in the forecasting model will be presented in the next section.

2. **Pre-processing:** The next step in the load forecasting model is to split the historical data (i.e., the load time series dataset) into two sequences: i) one before the forecasting horizon: this part is called training dataset which is used to train the forecasting model, and ii) another sequence after that training sequence which is called the testset, and it is used in the final evaluation of the forecasting method. In this project, there are two case studies which used different load forecasting horizons (i.e., short-term and mid-term forecasting). Therefore, we divided the historical energy consumption data set as:

    (a) **Short-term load forecasting (STLF)** The horizon prediction in the short-term

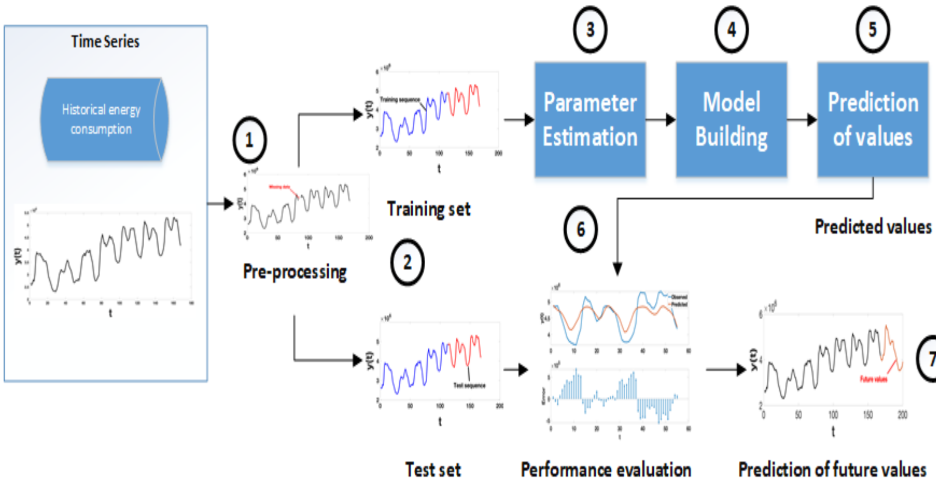


Figure 4.1: Time series prediction process.

load forecasting is from an hour to few days ahead (Table 3.1). The most benefit of this type of forecasting is to minimize the daily running and dispatching cost. In this project, I used the hourly total energy consumption for 1112 consumers from (1/10/2019) to (30/10/2019), which are 30 days in total. In this case, I split the dataset into training dataset including the first 23 days of hourly data and testing dataset which includes the last 7 days

(b) **Mid-term load forecasting (MTLF)** In the mid-term load forecasting, the prediction horizon ranges from days to few weeks (Table 3.1). This type of forecasting can be useful in performing efficient operation planning (e.g., planning for seasonal peaks). For this type of prediction, I used the hourly total energy consumption for 1112 consumers from (16/11/2018) to (24/11/2019), 374 days in total. In this case, I split the dataset into training datset which includes the first 344 days of the hourly data, and testing dataset which includes the last 30 days.

3. **Parameter estimation**: The third step refers to the selection of the forecasting model structure based on the data characteristics. To estimate the model parameters, search algorithms are used. The training sequence is divided into sub-sequences (i.e., samples) for training and validation. The search algorithm starts with initial parameters and iteratively tries to find parameters values that minimize the predictive error of the forecasting model.

4. **Model building**: Then, the model is built based on the estimated parameters values which found in previous step, and fits the training data. Then, the prediction error of the model reflects the chosen values for the parameters.

5. **Prediction of values**: The fifth steps relies on the use of the trained models to predict the values of a time series.

6. **Performance evaluation**: This step compares the predicted values with actual values (i.e., testing dataset) to measure the performance accuracy of the model. There are many performance evaluation measures. Some of them are introduced in Section 2.5.

7. **Prediction of future values**: The last step of the forecasting model will make predictions for future periods for which we do not have to the time series values yet. The prediction error must be calculated as soon as the new measurements are available (i.e., actual values) to check if the model parameters should be fine-tuned according to the recent data or not.

   As introduced in **Chapter 2 - Background concepts**, time series prediction models have been developed over the years , including approaches ranging from the simple regression models to the deep learning models.

4.2 Data

4.2.1 Data collection and description

The data used in this project are mainly the energy consumption of 1112 consumers. Also, other related historical data such as weather data were collected and investigated.

Energy consumption data

The project has used real energy consumption readings that were collected from smart meters (AMS) from Aalesund city by Morenett AS (Aalesund, Norway). The dataset that contains these readings is used in finding the best prediction algorithm, design IEMS, and DSM.

The readings are collected hourly from 1112 meters. The dataset has 8975 instances with two attributes: i) time and ii) energy consumption. The readings are collected from 16/11/2018 to 16/11/2019. Table 4.1 summarizes the used dataset, and Figure 4.2 shows the total consumption of all units (i.e., 1112 meters).

Table 4.1: Dataset from Morenett.

Type of meter	Number of units
Apartments and houses	960
Industrial/commercial units	114
Cabins	38
Total	1112

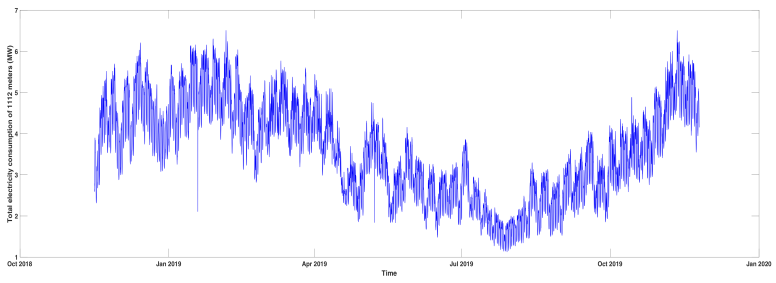


Figure 4.2: Energy consumption of 1112 meters in (Mega watt (MW)).

Weather data

I have investigated the correlation between the energy consumption and weather data such as: temperature, humidity, wind, cloud cover and visibility. The data is collected from the Vigra weather station[1]. Vigra station has most of the needed data without any loss in the readings. Table 4.2 summarizes the collected weather data which contains both hourly and daily readings for the same period as in energy consumption data.

Table 4.2: Collected weather data.

Type	Time	Description
Temperature	Hourly recorded	Air temperature recorded hourly above 2m from the ground.
Humidity	Hourly recorded and daily average	Relative humidity
Wind	Hourly recorded	Mean wind speed
Cloud cover	Hourly recorded	Total cloud cover is registered using a code 0 - 8 describing how many eights of the sky is covered by clouds (0 = no clouds, 8 = completely overcast). Code = -3 or 9 means cloud cover can not be estimated because the sky is obstructed from view because fog, drifting snow, etc.
Visibility	Hourly recorded	In km, Converted from synop code (max 75 km)
HDD (heating degree days)	Daily calculated	Daily sum of heating degree days. HDD is defined as how many degrees below 17 °C the hourly mean temperature is. If the hourly temperature is 17 or above, then the HDD is 0.
Max T	Daily recorded	Highest recorded air temperature per 24 hours
Min T	Daily recorded	Lowest recorded air temperature per 24 hours
Dew P (Mean dew point temperature)	Daily calculated	Daily mean dew point temperature is the temperature at which the air, when cooled, will become saturated.

4.2.2 Pre-processing

The aim of pre-processing is to check if there is missing data, wrong values, or cut off in the electricity. Figure 4.2 shows the energy consumption data that we used. We see there are obvious wrong values, probably because of cut off in the electricity, in 18/01/2019 at 23:00 and 08/05/2019 at 03:00. We fixed this by using linear interpolation between the values in the previous day, and next day at the same time point (i.e.,hour). For example we substitute the value at 18/01/2019 at 23:00 by the average value between the values at 17/01/2019 at 23:00 and 19/01/2019 at 23:00.

[1]Observations and weather statistics, available online from: `https://klimaservicesenter.no/observations/`. (As of 30 June 2020)

4.3 Explanatory analysis

This section contains some explanatory analysis of the dataset used in this project.

4.3.1 User types

The dataset contains 960 apartments and houses, 114 industrial and commercial buildings, and 38 cabins (see Table 4.1). Figure 4.3 shows samples from different categories: house, commercial, and cabin. There is a clear irregular consumption pattern at the building level. This is very clear in the example of cabin. The cabins are used more often in weekends, holidays, and usually when there are good weather conditions. The example in the Figure 4.3 shows more usage in the summer time than in the spring time in cabin example. The same is true for houses and apartments; the energy consumption may follow patterns in householders daily activities, such as working time, number of users, and type of building. While in the industrial and commercial buildings, there is an obvious pattern in the consumption as seen in Figure. 4.3, but it is still so much noise in the data.

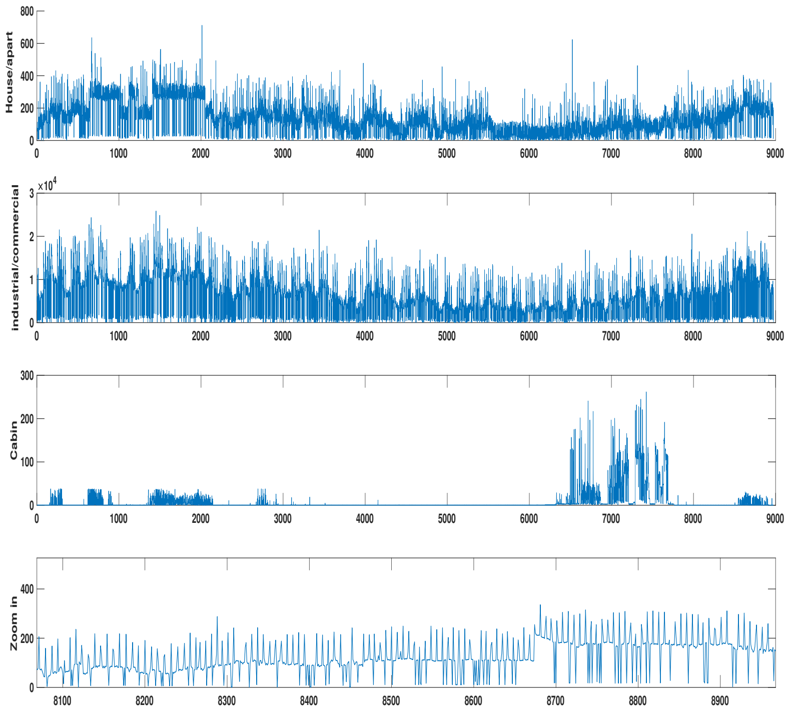


Figure 4.3: Samples from different categories: house, commercial, and cabin.

There is an obvious pattern in the total energy consumption time series as seen in Figure 4.2. Therefore, we made a load forecasting by using the total consumption of 1112 buildings in this project. While at the building level, the patters is irregular (i.e., stochastic), and we have to consider other factors, such as activity plan, holidays, ... etc, in order to perform forecasting.

Figure 4.4 shows the hourly average. There are tow peeks, one in the morning at 08:00 and the other one is in the end of working day at 16:00. Probably if we take only houses, the peaks will be more obvious.

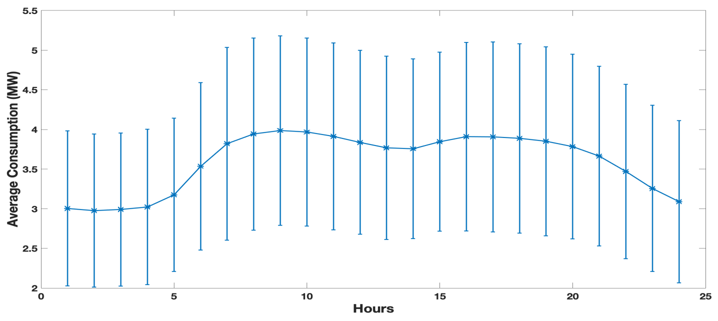


Figure 4.4: Hourly average.

4.3.2 Correlation with weather data

The weather condition is always a very important factor which can affect the load forecasting [88]. Figure 4.5 illustrates the relation between weather data and electricity consumption hourly-based. This figure suggests that the energy consumption is influenced more by the temperature than other meteorological factors such as humidity, wind, cloud cover, and visibility.

Figure 4.5 shows the correlation between weather data and electricity consumption on hourly-based. There are very little correlation between energy consumption and humidity and wind (i.e., -0.24 with humidity and 0.20 with wind). There is a clear correlation between energy consumption and temperature (i.e., -0.77). This means that the consumers use more electricity when the temperature gets lower. Table 4.3 shows correlation matrix hourly-based.

Table 4.3: Correlation matrix - hourly-based.

	Temp	**Humidity**	**Wind**	**Energy**
Temp	1	-0.006	-0.107	-0.772
Humidity	-0.006	1	-0.195	-0.238
Wind	-0.107	-0.195	1	0.204
Energy	-0.772	-0.238	0.204	1

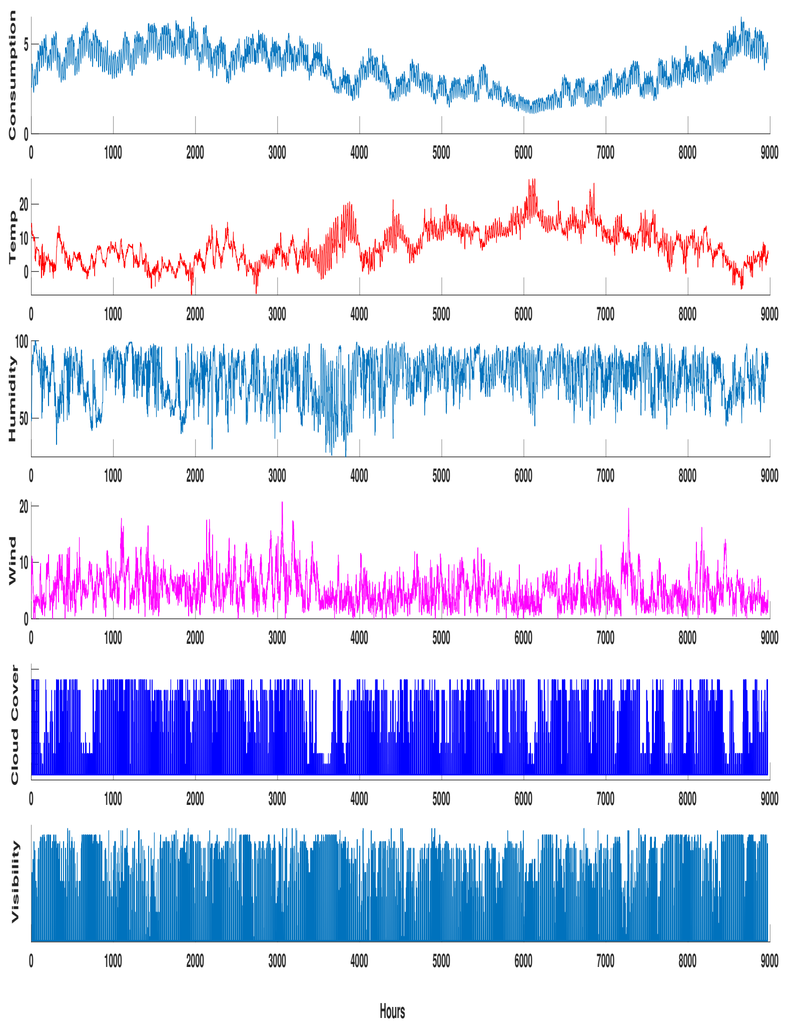


Figure 4.5: Weather data and electricity consumption per hour.

We also analyse the correlation between the energy consumption and weather data that are recorded or calculated on daily basis (see Table 4.2). The energy consumption inside the building is more influenced by the weather data that calculated on daily basis such as HDD. This is because the heating is a continual process which is less affected by short variation. Figure 4.7 shows the relation between weather data (e.g., HDD, Max T, Min T, Dew P, and Humidity) and electricity consumption on daily based.

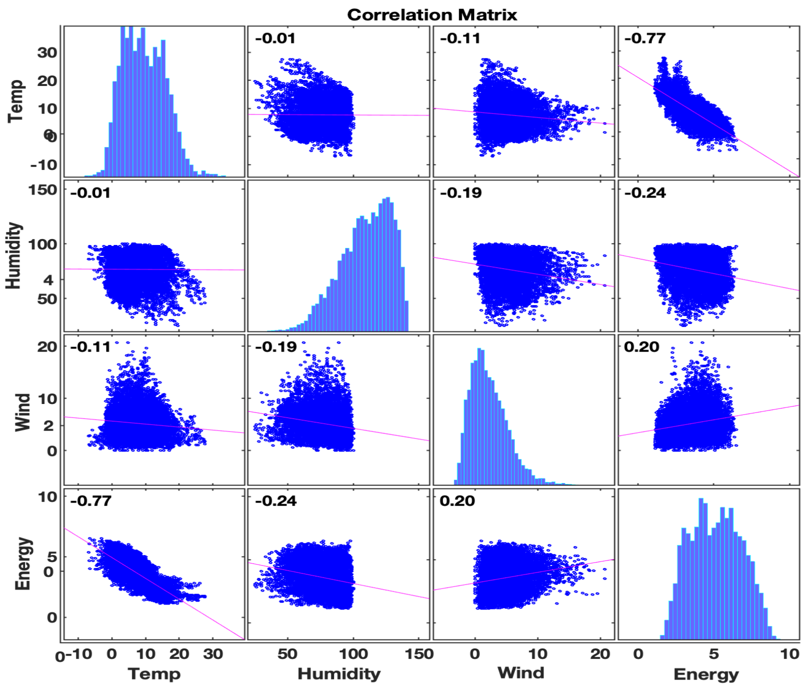


Figure 4.6: Correlation matrix - hourly based.

In order to check the relation between the weather data (HDD, Max T, Min T, Dew P, and Humidity) and electricity consumption on daily-based, we plotted the correlation matrix as seen in Figure 4.8. Also, we summarized the results of the correlations in the Table 4.4. The energy consumption is affected more by the temperature data (HDD, Max T, Min T, and Dew P) than humidity. The HDD, Max T, Min T, and Dew P are highly correlated. For instance, the correlation between HDD and Max T is -0.96 (highest). HDD has the maximum correlation with the energy consumption 0.90.

Table 4.4: Correlation matrix - daily-based.

	HDD	**Max T**	**Min T**	**Dew P**	**Humidity**	**Energy**
HDD	1	-0.962	-0.946	-0.932	-0.219	0.904
Max T	-0.962	1	0.869	0.885	0.157	-0.892
Min T	-0.946	0.869	1	0.915	0.280	-0.856
Dew P	-0.932	0.885	0.915	1	0.545	-0.876
Humidity	-0.219	0.157	0.280	0.545	1	-0.287
Energy	0.904	-0.892	-0.856	-0.876	-0.287	1

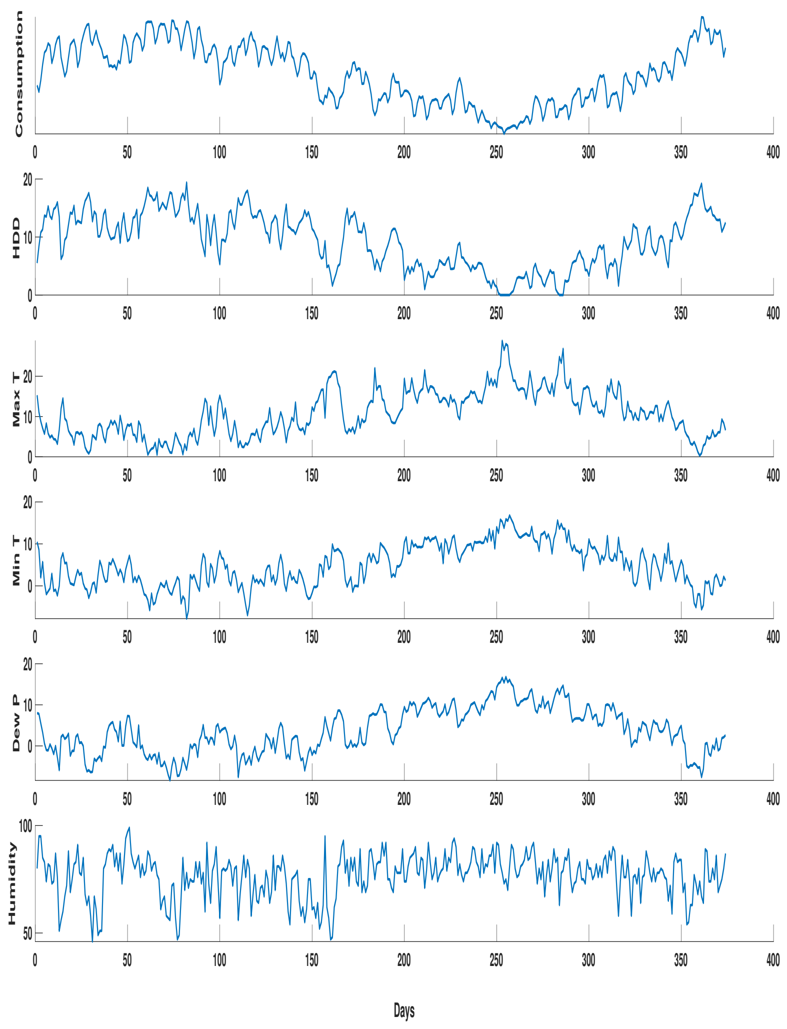


Figure 4.7: Relation between weather data and electricity consumption on daily based

4.3.3 Time series components

A time series plot shows a graphical presentation of the relationship between time and the time series target variable (e.g., energy consumption) as shown in Figure 4.2. The time series components are i) trend, ii) seasonality, and iii) residual as described in Section 2.1. A good starting for point for time series analysis relies on the visualization of data by using a time series decomposition plot separate the time series into it is seasonal, trend, and

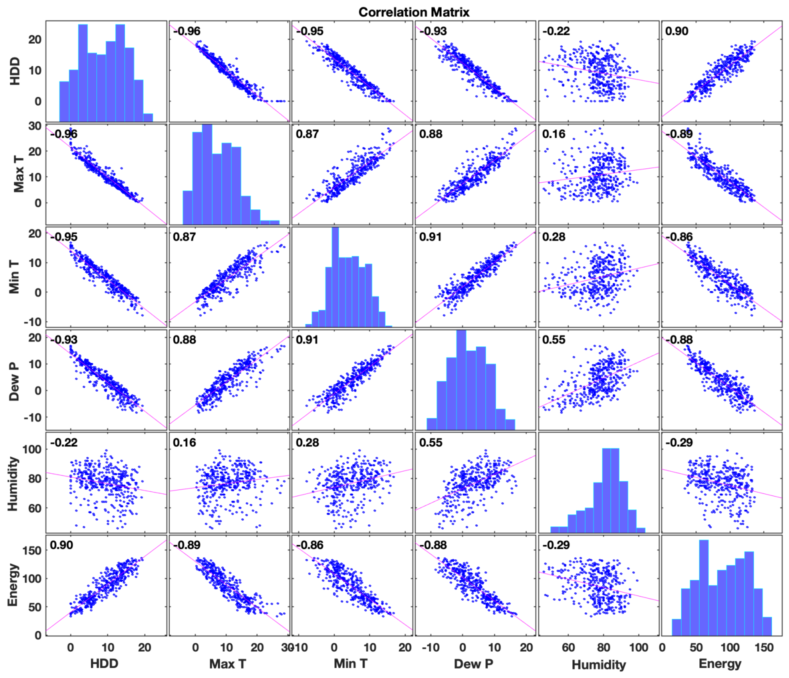


Figure 4.8: Correlation matrix - daily-based.

residual components. Figure 4.9 shows these components. There is upward trend which started in the end of summer and continued in the autumn. Also, there is downward trend which started in the end of winter time and continued in the spring time. There is no trend in the summer and winter times (i.e., horizontal trend). The seasonality is not clear enough in this figure, but the data has daily and weakly seasonality (check Chapter 5). And lastly, the residual is the error in the model that calculates. Time series decomposition provides a useful tool for better understanding problems during the analysis and forecasting processes.

4.3.4 Auto correlation

Autocorrelation shows the correlation of a signal with itself as a function of the time lag between the two points. It is a useful visualisation to determine periodic phenomena in time series data sets.

Figure 4.10 shows ACF and PACF with 50 lags. The correlation coefficient represented in the vertical axis while the number of lag shown across the horizontal axis. This allows us know how far out our time series is correlated with itself. This series is un-differenced and has no obvious decay toward 0 correlation. This means that the series is not stationary and it needs to be differenced to reach a stationary series. Figure 4.12 shows ACF after the

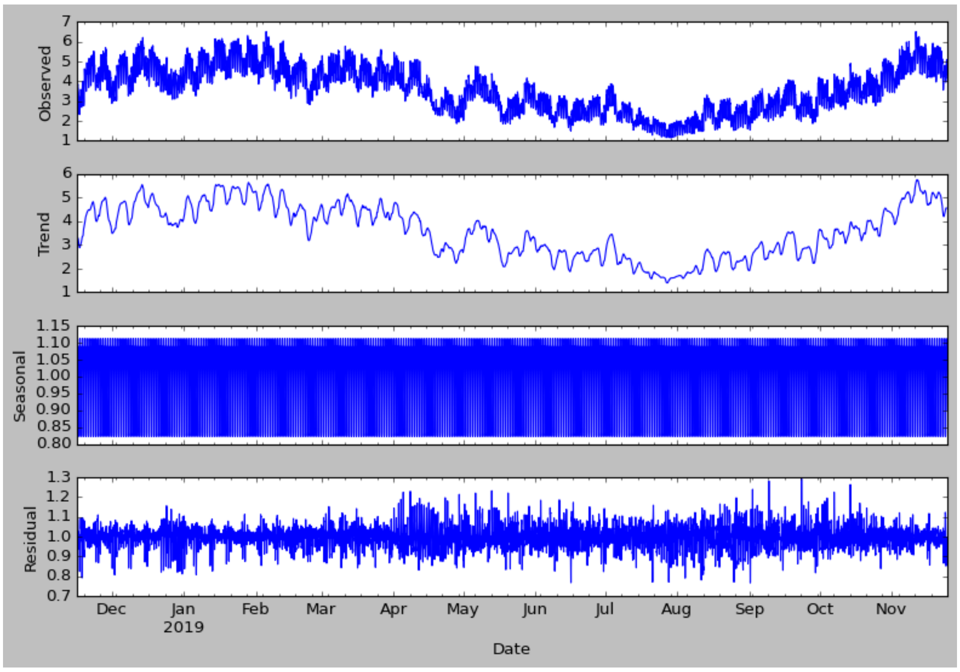
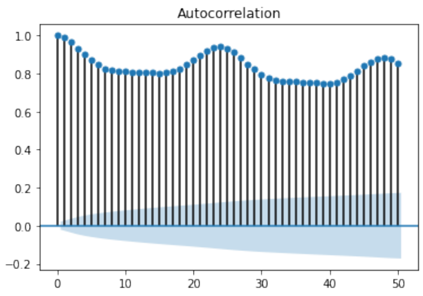
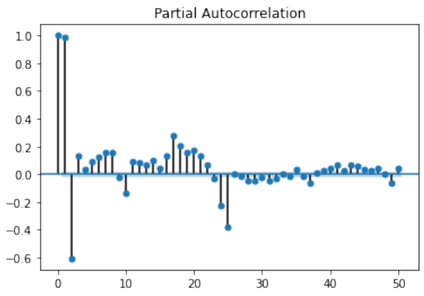


Figure 4.9: Decomposition

first difference. Now, lag-1 is significant. If we take a look at the PACF plot after the first difference, we find that PACF plot has most significant spikes at Lag-1 and Lag-2. Section 2.3 explains the auto correlation function (ACF) and partial auto correlation function (PACF).



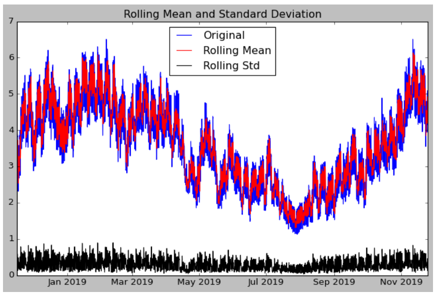
(a) ACF plot with lags=50 on the total energy consumption dataset.



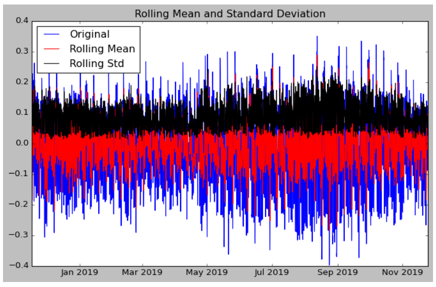
(b) PACF plot with lags=50 on the total energy consumption dataset.

Figure 4.10: ACF and PACF plots of the total energy consumption dataset.

The data as shown in Figure 4.11a are non-stationary since the mean and the variance are not constant over time. After taking the first difference, Figure 4.11b shows that the trend is



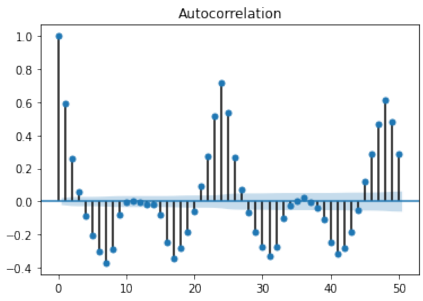
(a) Energy consumption dataset with mean and standard deviation before difference.



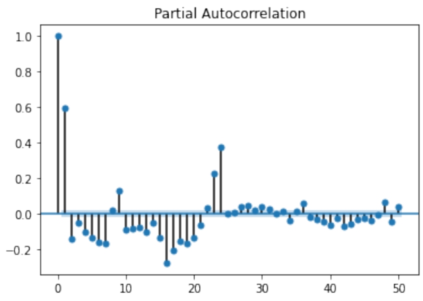
(b) Energy consumption dataset with mean and standard deviation after one difference.

Figure 4.11: Energy consumption dataset with mean and standard deviation on stationary and non-stationary dataset.

converges to zero.



(a) ACF plot with lags=50 of the resulting after one difference of the total energy consumption dataset.



(b) PACF plot with lags=50 of the resulting after one difference of the total energy consumption dataset.

Figure 4.12: ACF and PACF plots of the resulting after one difference total energy consumption dataset.

4.4 Implementation details

4.4.1 Hardware

All the models were developed using an Apple MacBook Pro 13 Retina, with 3.1 GHz two CPU cores (Intel Core i7), 16GB Memory, and 1TB SSD Harddisk. Also, I used Google Colab[2] to develop some methods for time series forecasting by using Python. The Google Colab is a free cloud service which support GPU also.

[2]https://colab.research.google.com/ (As of July 2020).

4.4.2 Programming languages, Tools, and Libraries

In this section, the main used tools in the master project will be introduced. Some of these tools are used for preparing the data for the forecasting models or analysing the data.

Python

Most development of the statistical and machine learning algorithms to build the time series forecaster was done with the programming language Python 3. Python is the most popular programming language for time series forecasting. Python has many libraries to use in time series forecasting such as SciPy [89].

The main libraries used are:

- **Pandas:** It is a software library written for the Python programming language for data manipulation and analysis. Pandas is a flexible, easy-to-use and powerful tool which offers data structures and operations for manipulating numerical tables and time series [90].
- **Statsmodels:** It has several tools for statistical modelling. Further, there are tools dedicated to time series that be used for forecasting.
- **Scikit-Learn**: It is a free machine learning library for Python. It supports Python numerical and scientific libraries like NumPy and SciPy and it features various algorithms such as support vector machine, and k-nearest neighbours [91].
- **Keras:** It is a powerful and easy to use library for developing and evaluating deep learning models. Keras runs on top of tensorflow.
- **Tensorflow:** It is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow [92].
- **Pytorch**: It is an open source, deep learning library that facilitates building deep learning projects [93].

MATLAB

MATLAB is developed by MathWorks [94]. It is a multi paradigm numerical computing environment and proprietary programming language. MATLAB uses computations and algorithms to analyze large amounts of data and present it in visually appealing formats. MATLAB has many useful toolboxes such as statistical toolbox, and optimization toolbox.

Weka

Weka (Waikato Environment for Knowledge Analysis) is developed at the University of Waikato, New Zealand. Weka contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions [47].

4.4.3 Data formatting

The historical load data is stored in CSV format file. Data set contains 8975 rows and two columns, each row represents a new time step (1 hour) and a load observation. This format is a two-dimensional matrix with the following shape (#Samples;#Features). This gives the possibility to use the data directly in statistical methods. But in the machine learning methods such as CNN, it requires a specific kind of historical data format to train the models. These models require not only the current sample, but all the samples in the specified historical time window. This needs to be in a specific format having a list of lists, with samples. Then, the matrix will be three-dimensions matrix (#Samples;#Time-steps;#Features), where:

- Samples: One sequence is one sample and a batch is comprised of one or more samples.
- Time Steps: One time step is one point of observation in the sample.
- Features: One feature is one observation at a time step.

For this type of format, a special library is used and the time window approach is used.

4.4.4 Load forecasting model implementation

This section briefly mentions which libraries and some details about the implementation of the different load forecasting models.

Statistical methods

- AR is implemented by using Python in Colab.
- ARMA is implemented by using Python in Colab.
- ARIMA is implemented by using Python in Colab.
- SARMA is implemented by using Python in Colab.
- MA is implemented by using Python in Colab.
- SES is implemented by using Python in Colab.
- HES is implemented by using Python in Colab.
- Single-seasonal Holt-winters method is implemented by using Python in Colab.

Machine learning methods

- LR is implemented by using Python in Colab (we use Keras model which contains Dens layer without any activation function. The model is compiled with the optimizer (SGD) and it uses hyper loss for training and early stopping callback function.
- SVM is implemented by using WEKA.
- K-NN is implemented by using WEKA.

- DNN is implemented by using Python in Colab (Implemented using Dense layers in Keras).
- RNN is implemented by using Python in Colab (Implemented using RNN layer in Keras).
- LSTM is implemented by using Python in Colab (Implemented using LSTM layer in Keras).
- CNN is implemented by using Python in Colab (Implemented with standard layers in Keras).

4.4.5 Case study implementation

This section gives an overview of the implementation of the case studies. The case studies are: i) an intelligent energy management system, and ii) demand side management system.

Case A

A full description of the proposed intelligent energy management system (IEMS) will be presented in Chapter 6. The example of IEMS is implemented by using MATLAB. The load is predicted by the implemented load forecaster. The power also generated locally in the grid by different sources such as solar cells and wind farms. The generating depends on the foretasted weather (e.g., wind speed). The IEMS will optimise the using of energy storage to minimize the peak and reduce the power usage from the grid.

Case B

A full description of the proposed demand side management (DSM) model will be in Chapter 6. The example of DMS is implemented by using MATLAB. The load is predicted by the implemented load forecaster. The user will schedule using of shiftable appliances, such as washing machine, dish washer, clothes dryer, and electrical vehicle, to reduce his electricity prices.

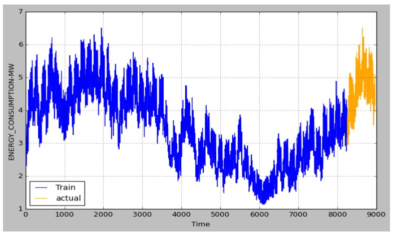
Chapter 5

Load forecasting results

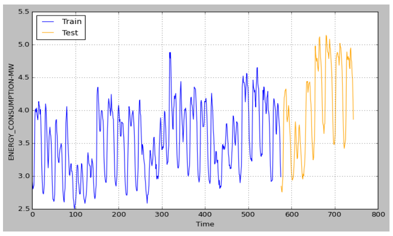
This chapter presents the implementation details and results of time series forecasting models in the master project. In the first section, the statistical methods' results are presented. After that, machine learning methods' results are presented in the next section. At the end of the chapter, a comparison between different statistical and machine learning methods in the performance is given.

5.1 Statistical methods

These methods are applied on a dataset that contains the historical total energy consumption for 1112 consumers and applied on two different time series horizons: i) The first horizon is from $(1/10/2019))$ to $(30/10/2019)$, 30 days. We split the dataset into a training set, which includes the first 23 days of the dataset, and testing set which includes the last 7 days. ii) the second horizon is from $(16/11/2018))$ to $(24/11/2019)$, 374 days is used. We split the dataset into a training set, which includes the first 344 days of the dataset, and testing set, which includes the last 30 days. Figure 5.1 shows the used dataset.



(a) Dataset, training set (blue) and testing set (yellow) for 30 days forecasting horizon.



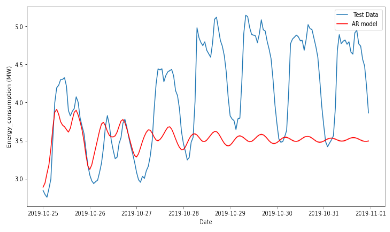
(b) Dataset, training set (blue) and testing set (yellow) for 7 days forecasting horizon.

Figure 5.1: Dataset, training set and testing set.

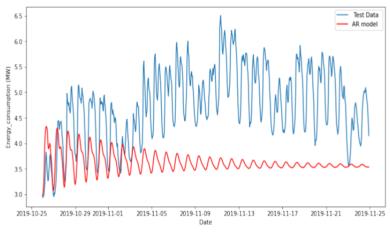
5.1.1 ARIMA

Autoregressive Model (AR)

As mentioned in Section 2.3.3. AR model is selected through a combination of visual inspection of (ACF) and (PACF) plots. After running the functions to choose the optimal lags in order to find the parameter (p), $\mathrm{AR}(1)$ model appears to be our best AR Model for the 7-day horizon, while the AR(3) is the best AR model for the 30-day horizon. It was no increase in the forecasting accuracy when we tried other parameter values. The prediction with AR model on the datatset is shown in Figure 5.2. Figure 5.2a shows the forecasting result for 7-day horizon , and Figure 5.2b shows the forecasting result for 30 days.



(a) Forecasting values (red) and actual values (blue) for 7-day horizon.

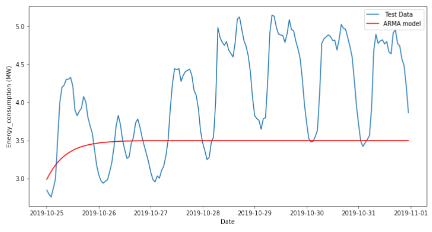


(b) Forecasting values (red) and actual values (blue) for 30-day horizon.

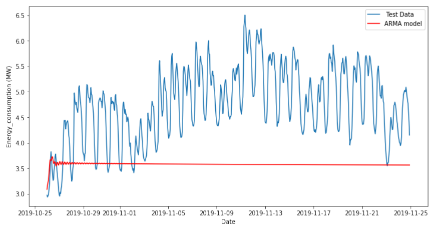
Figure 5.2: Forecasting results of AR model.

ARMA

Similarly to the AR model, ARMA model's parameters p, q are selected by visual inspection of the ACF and PACF plots as described in Section 2.3.3. ARMA (1, 1) model seems to be the best model as it gives the best forecasting accuracy for the 7-day horizon , while ARMA (5, 3) gives a best forecasting accuracy for the 30-day horizon (i.e.,there was not any improvement in the forecasting accuracy when I tried other different values for p, q parameters. The prediction with ARMA model is shown in Figure 5.3. Figure 5.3a shows the forecasting result for the 7-day horizon, and Figure 5.3b shows the forecasting result for the 30-day horizon.



(a) Forecasting values (red) and actual values (blue) for 7-day horizon.

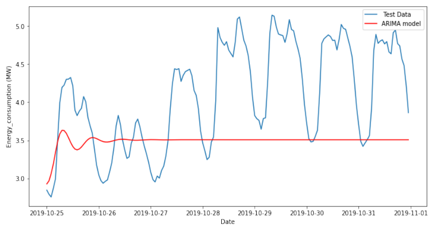


(b) Forecasting values (red) and actual values (blue) for 30-day horizon.

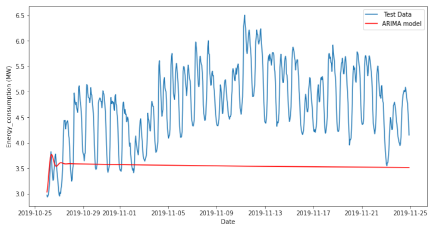
Figure 5.3: Forecasting results of ARMA model.

ARIMA

Akaike's Information Criterion (AIC) is used to determine the order of ARIMA model (p,d,q) as described in Section 2.3. Depending on the AIC score, the order of ARIMA is selected. The optimal model for the dataset is ARIMA(7, 0,1) on both horizons, 7-days and 30-days. The result of the forecasting is shown in Figure 5.4.



(a) Forecasting values (red) and actual values (blue) for 7-day horizon.

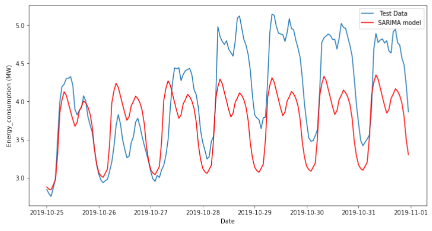


(b) Forecasting values (red) and actual values (blue) for 30-day horizon.

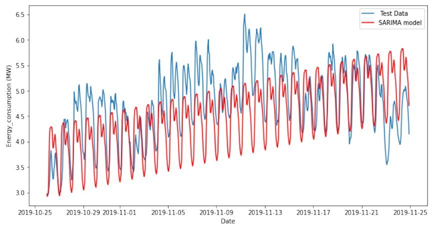
Figure 5.4: Forecasting results of ARIMA model.

SARIMA

Similarly to the ARIMA model, AIC is used to determine the parameters of SARIMA model $(p, d, q), (P, D, Q)_m$ as described in Section 2.3. AIC score is used to find the best SARIMA model which fits the training set.
The optimal model is SARIMA $(1, 0, 1), (1, 0, 1)_{24}$ on the horizon of 7 days, and SARIMA $(2, 1, 2), (1, 1, 0)_{24}$ on the horizon of 30 days. Figure 5.13 shows the forecasting results by using the SARIMA model. The forecasting results for 7-day horizon is illustrated in Figure 5.5a, and in Figure 5.5b for 30 days.



(a) Forecasting values (red) and actual values (blue) for 7 days.

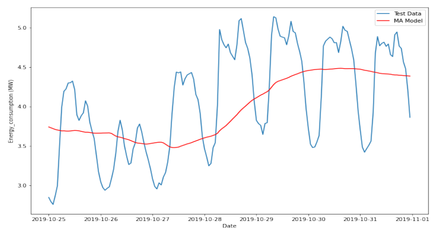


(b) Forecasting values (red) and actual values (blue) for 30 days.

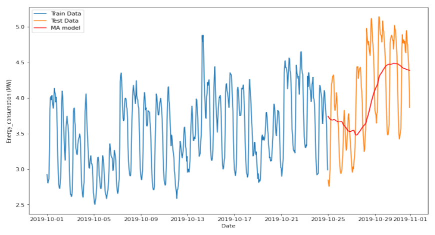
Figure 5.5: Forecasting results of SARIMA model.

5.1.2 MA

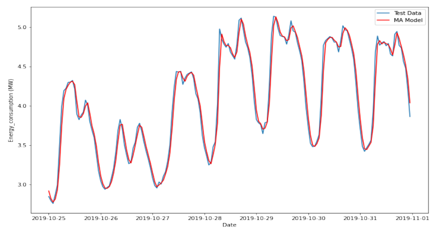
Moving average (MA) is a simple method which is used often is smoothing the data. MA model forecasting depends on the number of recent observations (n) that are included in the calculation of MA values as described in Section 2.3. Figure 5.6a shows the forecasting results when $n = 48$. The MA model gives a curve roughly emulating the original time series, but it does not anticipate trend or seasonality depending on the current time (i.e., the period after which you want to forecast). Figure 5.6c shows the result of MA when $n = 2$. The forecasting accuracy is much increased by selecting $n = 2$. MA is used only for evaluation, but not in prediction because in order to get the value for the next step, we need the previous values to be actually observed.



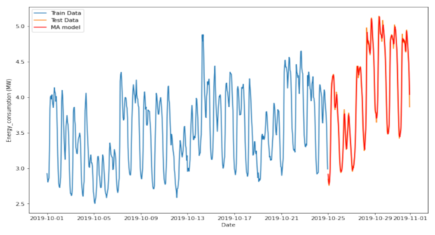
(a) Forecasting values (red) and actual values (blue), N = 48.



(b) Forecasting values (red), training set (blue), and testing set (orange), N = 48.



(c) Forecasting values (red) and actual values (blue), N = 2.



(d) Forecasting values (red), training set (blue), and testing set (orange), N = 2.

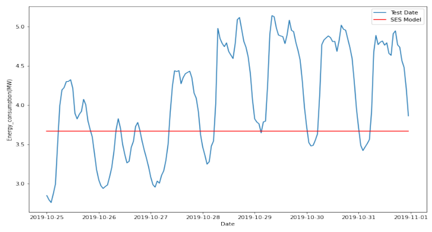
Figure 5.6: Forecasting results of MA model.

5.1.3 ES

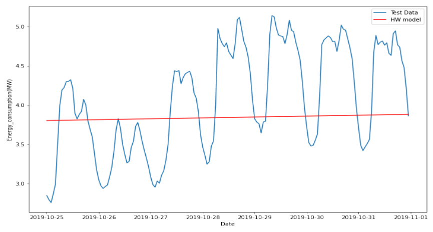
To estimate the exponential smoothing (ES) model, the fit criteria is used (default fit function) to find the best value of α, β, and γ (i.e., which gives the lowest error value) to fit the model to the data. Table 5.1 shows the selected parameters for the ES algorithms. These parameters give the best model to SES, HES, AHW, and MHW on the horizons of 7-days, and 30-days. Figure 5.7 shows the forecasting results for ES model on the horizon of 7-days. Figure 5.7a shows the forecasting result from SES model, Figure 5.7b shows the forecasting result from HES model, Figure 5.7c shows the forecasting result from AHW model, and Figure 5.7d shows the forecasting result from MHW model. The forecasting result on the horizon 30-days is illustrated in Figure 5.8, where Figure 5.8a show the result from SES. Figure 5.8b shows the result from HES, Figure 5.8c shows the result from AHW, and Figure 5.8d shows the result from MHW.

Table 5.1: Selected parameter for ES.

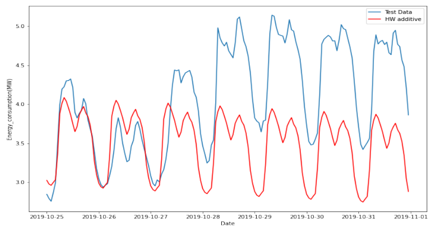
Algorithm	Parameter	Variation range (initial:step:final)
SES	Smoothing constant associated with the level (α)	$\alpha = 0 : 0.1 : 1$
HES	Smoothing constant associated with the level (α)	$\alpha = 0 : 0.1 : 1$
	Smoothing constant associated with the trend (β)	$\beta = 0 : 0.1 : 1$
AHW	Smoothing constant associated with the level (α)	$\alpha = 0 : 0.1 : 1$
	Smoothing constant associated with the trend (β)	$\beta = 0 : 0.1 : 1$
	Smoothing constant associated with the seasonality α	$\alpha = 0 : 0.1 : 1$
	Number of observations that make up a seasonal period (m)	$m = 0 : 24$
MHW	Smoothing constant associated with the level (α)	$\alpha = 0 : 0.1 : 1$
	Smoothing constant associated with the trend (β)	$\beta = 0 : 0.1 : 1$
	Smoothing constant associated with the seasonality α	$\alpha = 0 : 0.1 : 1$
	Number of observations that make up a seasonal period (m)	$m = 0 : 24$



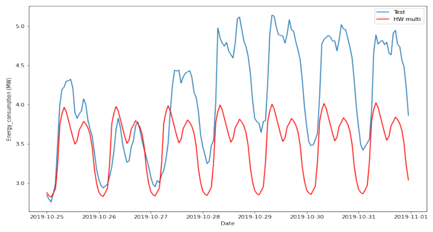
(a) Forecasting values (red) by SES model and actual values (blue).



(b) Forecasting values (red) by HES model and actual values (blue).



(c) Forecasting values (red) by AHW model and actual values (blue).



(d) Forecasting values (red) by MHW model and actual values (blue).

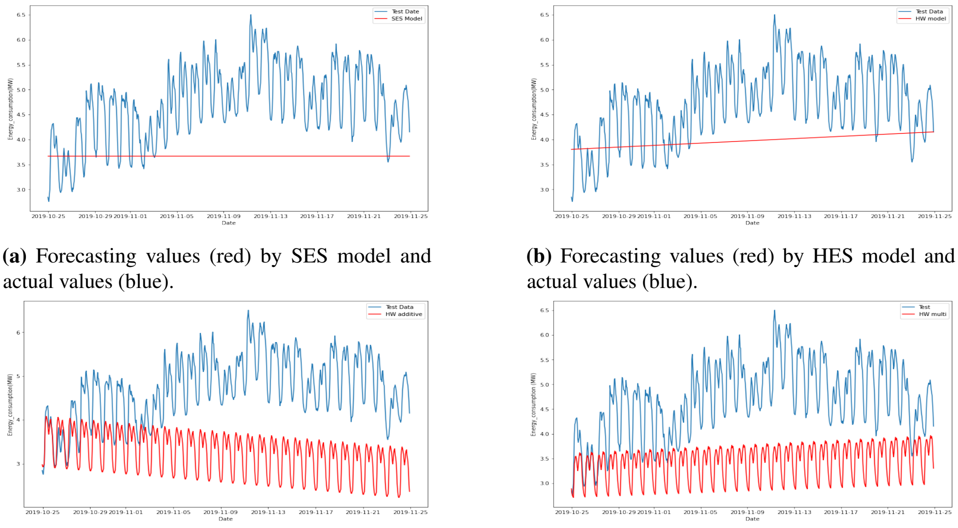
Figure 5.7: Forecasting results of ES model on the horizon 7 days.

5.2 Machine learning methods

Machine learning (ML) methods were applied to perform predictions on the same periods as statistical methods in Section 5.1.

5.2.1 LR

The linear regression (LR) is described in Section 2.4. Figure 5.9 shows the layout which is used in the forecasting model. The model was fit over the 200 epoch and the results are shown in Figure 5.15a for 7-day horizon and in Figure 5.16a for 30-day horizon. The parameters that used to tune the model are summarized in Table 5.2.



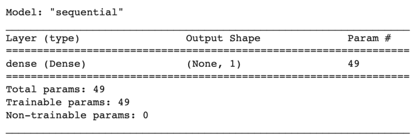
(a) Forecasting values (red) by SES model and actual values (blue).

(b) Forecasting values (red) by HES model and actual values (blue).

(c) Forecasting values (red) by AHW model and actual values (blue).

(d) Forecasting values (red) by MHW model and actual values (blue).

Figure 5.8: Forecasting results of ES model on the horizon 30 days.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 49
=================================================================
Total params: 49
Trainable params: 49
Non-trainable params: 0
_________________________________________________________________
```

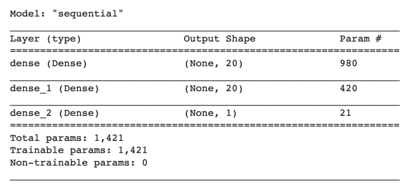
Figure 5.9: LR Layout.

Table 5.2: Selected parameters for LR.

Long-short-term-memory	Parameters
Optimizer	SGD
Learning rate	$1e-6$
Batch size	32
Time window	48
Output Layer	1

5.2.2 MLP

The MLP is not that much different from the linear regression model. It is relatively simple deep neural network that has three layers as shown in Figure 5.10. The activation function which is used in this model is the $ReLU$ function. The selected parameters for the MLP model is summarized in Table 5.3, where the model consists of two Dense layers $(10, 10)$, and one output layer 1. Figure 5.15b shows the forecasting results of MLP model for 7-day horizon and Figure 5.16b for 30-day horizon. It is obvious that the model performs very good on the given dataset.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 20)                980
_________________________________________________________________
dense_1 (Dense)              (None, 20)                420
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 21
=================================================================
Total params: 1,421
Trainable params: 1,421
Non-trainable params: 0
_________________________________________________________________
```

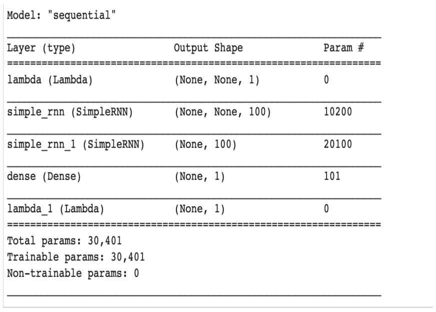
Figure 5.10: MLP Layout.

Table 5.3: Selected parameters for MLP.

Deep Neural Network	Parameters
Optimizer	SGD
Learning rate	$1.2e-4$
Batch size	32
Time window	48
Dense layer	$(10, 10)$
Output Layer	1

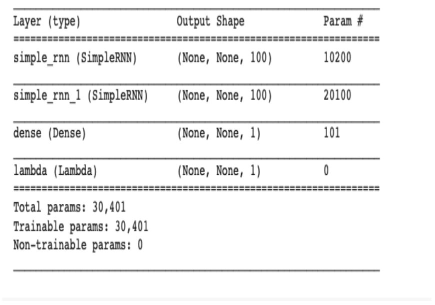
5.2.3 RNN

The recurrent neural network (RNN) model contains two recurrent layers and one final dense layer, which serves as the output layer as shown in Figure 5.11a. As described in Section 2.4, the RNN model is fed by a batch of sequences (i.e., windows from the time series), and the model output is a batch of forecasts. The inputs contain three variables (batch size, # time steps, # of dims = 1). In Keras, the default behaviour of all the recurrent layers is sequence to vector (i.e., the output of each recurrent layer is a single vector). This is called RNN (sequence to vector) and it shown in Figure 5.11a, and this model was implemented in the beginning. "None" meaning the model will support sequences of any length. Then, I have changed the default behaviour in Keras to build RNN (sequence to sequence) model to let the recurrent layers to output sequences. This model is shown in Figure 5.11b.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda (Lambda)              (None, None, 1)           0
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, None, 100)         10200
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 100)               20100
_________________________________________________________________
dense (Dense)                (None, 1)                 101
_________________________________________________________________
lambda_1 (Lambda)            (None, 1)                 0
=================================================================
Total params: 30,401
Trainable params: 30,401
Non-trainable params: 0
_________________________________________________________________
```

(a) RNN (sequence to vector) Layout.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, None, 100)         10200
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, None, 100)         20100
_________________________________________________________________
dense (Dense)                (None, None, 1)           101
_________________________________________________________________
lambda (Lambda)              (None, None, 1)           0
=================================================================
Total params: 30,401
Trainable params: 30,401
Non-trainable params: 0
_________________________________________________________________
```

(b) RNN (sequence to sequence) Layout.

Figure 5.11: RNN (sequence to sequence) Layout.

In both RNN models: sequence to sequence RNN and sequence to vector RNN, the default activation function in the RNN layers is $tanH$, which is the hyperbolic tangent activation. The hyperbolic tangent function is used instead of $ReLU$ because RNN models have tendency to have unstable gradient. The gradients are more likely to saturate by using hyperbolic tangent. Training RNN model is actually tricky. For example, if the learning rate is high, then the training is unstable and the model does not learn. While if the learning rate is low to avoid the instability, the training is very slow. In Keras, the RNN model with the optimal learning rate was found automatically by using $callbacks$ function, with early stopping and checkpoints. The selected parameters for the RNN (sequence to vector) model is summarized in Table 5.4a, while the selected parameters for the RNN (sequence to sequence) model is summarized in Table 5.4b.

Figures 5.15c and 5.16c show the forecasting results of RNN (sequence to vector) model for 7 days and 30 days, respectively. Figures 5.15d and 5.16d show the forecasting results of RNN (sequence to sequence) model for 7 days and 30 days, respectively. The model performs well on the given dataset.

Table 5.4: Selected parameters for RNN model.

(a) RNN (sequence to vector).

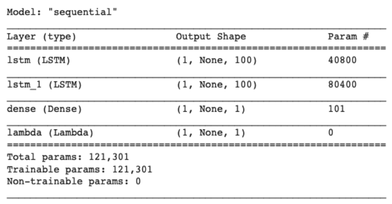
RNN (sequence to vector)	Parameters
Optimizer	SGD
Learning rate	$1.5e^{-6}$
Batch size	128
Time window	48
Simple RNN Layers	100,100
Output Layer	1

(b) RNN (sequence to sequence).

RNN (sequence to sequence)	Parameters
Optimizer	SGD
Learning rate	$1e^{-4}$
Batch size	128
Time window	48
Simple RNN Layers	100,100
Output Layer	1

5.2.4 LSTM

To make the RNN learn long-term patterns, long short term memory model (LSTM) is implemented. LSTM is implemented in Keras by replacing the simple RNN layer with an LSTM layer as shown in Figure. 5.12. The LSTM layer implements many optimizations, which make the model very slow. Similarly to RNN model, the LSTM model with the optimal learning rate was found automatically. Table 5.5 shows the selected parameters for the LSTM. The table shows that the SGD with a learning rate of $1e - 6$ performs best with two hidden layers of 100 and 100. The time window was selected to be 48. Figures 5.15e and 5.16e show the forecasting results of the LSTM model for 7-day horizon and 30-day horizon, respectively. The model did not perform better than RNN on the given dataset.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (1, None, 100)            40800
_________________________________________________________________
lstm_1 (LSTM)                (1, None, 100)            80400
_________________________________________________________________
dense (Dense)                (1, None, 1)              101
_________________________________________________________________
lambda (Lambda)              (1, None, 1)              0
=================================================================
Total params: 121,301
Trainable params: 121,301
Non-trainable params: 0
_________________________________________________________________
```

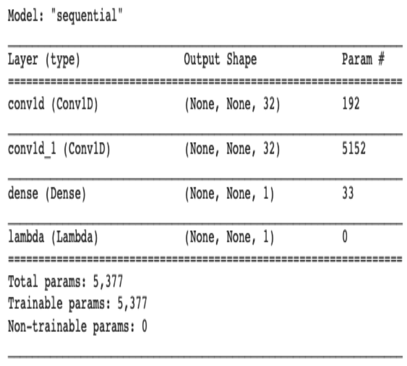
Figure 5.12: LSTM Layout.

Table 5.5: Selected parameters for LSTM.

Long-short-term-memory	Parameters
Optimizer	SGD
Learning rate	$1e - 6$
Batch size	32
Time window	48
Hidden Layers	100,100
Output Layer	1

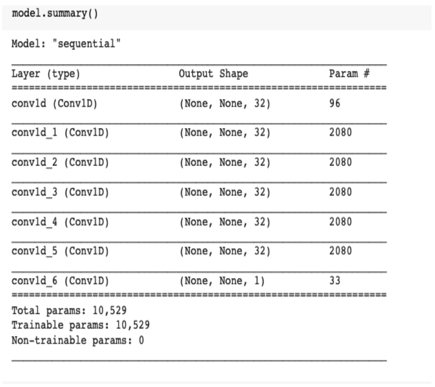
5.2.5 CNN

CNN model comprises two of one dimensional convolutional CONV-1D layers, and one dense output layer as shown in Figure 5.13a. The model used $Relu$ as an activation function in this experiment. Table 5.6a shows the model parameters. The kernel size is 5, therefore the padding that is used here, is a causal padding. This is essential to ensure that the model does not cheat and use future value to forecast future value. The stride is 1, and the model finds the optimal learning rate by using the fit model that used the training dataset to find the best learning rate that gives the lowest error and best performance. The results that achieved from CNN are shown in Figure 5.15f for 7-day horizon, and in Figure 5.16f for 30-day horizon. Some other adjustments were tested, resulting in an increase improvement in the performance. Specifically, dilated convolution with different rates that is called CNN wavenet was developed. The parameters that are used in the CNN wavenet model are summarized in Table 5.6b, and the layout of the model is shown in Figure 5.13b.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, None, 32)          192
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 32)          5152
_________________________________________________________________
dense (Dense)                (None, None, 1)           33
_________________________________________________________________
lambda (Lambda)              (None, None, 1)           0
=================================================================
Total params: 5,377
Trainable params: 5,377
Non-trainable params: 0
_________________________________________________________________
```

(a) CNN Layout.



```
model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, None, 32)          96
_________________________________________________________________
conv1d_1 (Conv1D)            (None, None, 32)          2080
_________________________________________________________________
conv1d_2 (Conv1D)            (None, None, 32)          2080
_________________________________________________________________
conv1d_3 (Conv1D)            (None, None, 32)          2080
_________________________________________________________________
conv1d_4 (Conv1D)            (None, None, 32)          2080
_________________________________________________________________
conv1d_5 (Conv1D)            (None, None, 32)          2080
_________________________________________________________________
conv1d_6 (Conv1D)            (None, None, 1)           33
=================================================================
Total params: 10,529
Trainable params: 10,529
Non-trainable params: 0
_________________________________________________________________
```

(b) CNN wavenet Layout.

Figure 5.13: CNN and CNN wavenet models Layouts.

The CNN Wavenet model composed of six one-dimensional convolutional layer with growing dilation rate, and output $1D$ convolutional layer with a single kernel size 1 with stride 1. The results from this experiment is shown in Figure 5.15g for 7-day horizon and in Figure 5.16g for 30-days horizon. It is obvious that the model performs very well on the given dataset.

Table 5.6: Selected parameters for CNN and CNN wavenet models.

(a) Selected parameters for CNN.

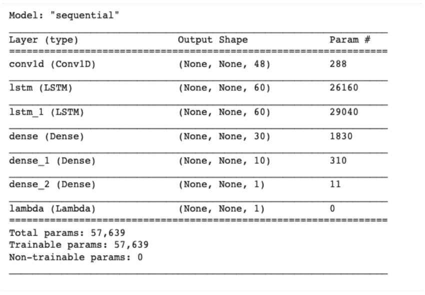
Convolutional Neural Network	Parameters
Optimizer	SGD
Learning rate	$1e^{-5}$
Batch size	132
Time window	48
filters	32
kernal size	5
strides	1
padding	$causal$
Output Layer	1

(b) Selected parameters for CNN Wavenet.

Convolutional Neural Network wavenet	Parameters
Optimizer	SGD
Learning rate	$3e^{-4}$
Dilation rate	$1, 2, 4, 8, 16, 32$
Batch size	132
Time window	48
filters	32
kernal size	2
strides	1
padding	$causal$

5.2.6 Combination (CNN, LSTM, MLP)

CNN, LSTM, and MLP models can be together forming the desired topology by do combinations between them. The layout of the model is shown in Figure 5.14, and the model parameters that are used, are summarized in Table 5.7. The model starts with the $ID - Convolution$ with filter 48. This output is fed to the couple of LSTM with 60 cell. Then the output is fed into the dense $(30, 10)$ layer, and finally the output from the dense 1 layer. The results from this experiment is shown in Figure 5.15h for 7-day horizon and in Figure 5.16h for 30-day horizon. This model didn't perform very good on the given dataset, as CNN and CNN wavenet models.

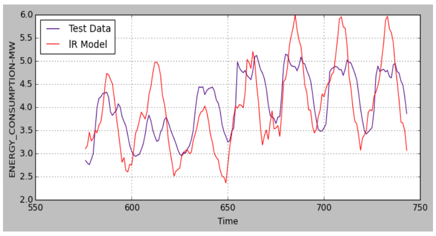


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, None, 48)          288
_________________________________________________________________
lstm (LSTM)                  (None, None, 60)          26160
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 60)          29040
_________________________________________________________________
dense (Dense)                (None, None, 30)          1830
_________________________________________________________________
dense_1 (Dense)              (None, None, 10)          310
_________________________________________________________________
dense_2 (Dense)              (None, None, 1)           11
_________________________________________________________________
lambda (Lambda)              (None, None, 1)           0
=================================================================
Total params: 57,639
Trainable params: 57,639
Non-trainable params: 0
_________________________________________________________________
```

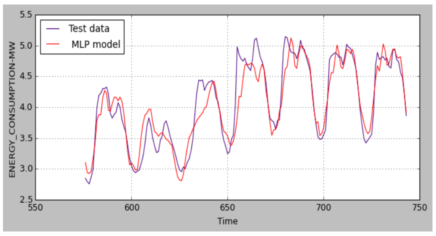
Figure 5.14: Combination Layout.

Table 5.7: Selected parameters for combination model.

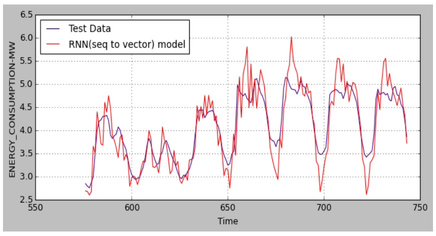
Combination Model	Parameters
Optimizer	SGD
Learning rate	$1.2e^{-6}$
Dilation rate	$1, 2, 4, 8, 16, 32$
Batch size	100
Time window	48
filters	48
kernal size	5
padding	$causal$
LSTM layer	$60, 60$
Dense layer	$30, 10$
output	1



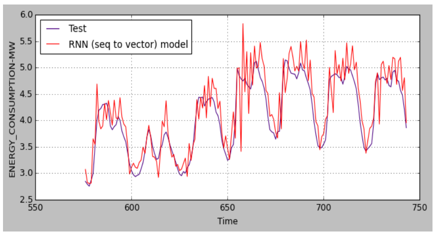
(a) Forecasting values (red) and actual values (blue) - LR model.



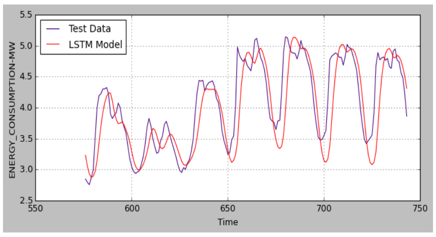
(b) Forecasting values (red) and actual values (blue) - MLP model.



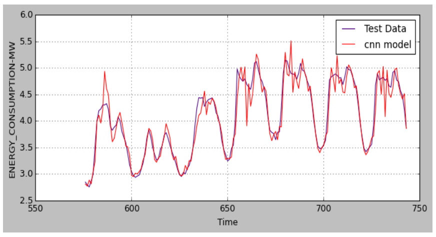
(c) Forecasting values (red) and actual values (blue) - RNN (sequence to vector) model.



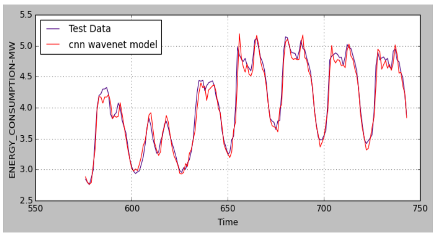
(d) Forecasting values (red) and actual values (blue) - RNN (sequence to sequence) model.



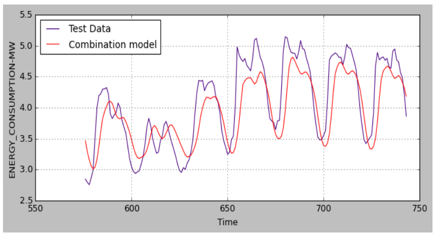
(e) Forecasting values (red) and actual values (blue) - LSTM model model.



(f) Forecasting values (red) and actual values (blue) - CNN model.

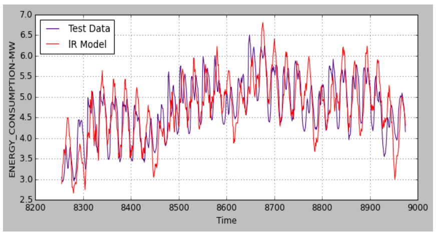


(g) Forecasting values (red) and actual values (blue) - CNN-Wavenet model.

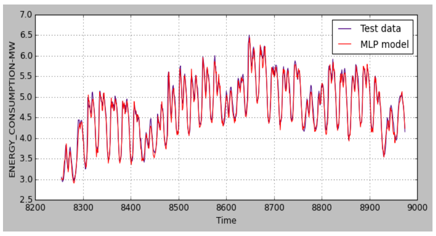


(h) Forecasting values (red) and actual values (blue) - Combination model.

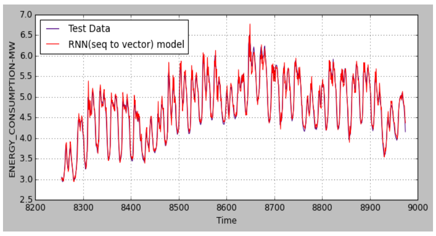
Figure 5.15: Forecasting values by machine learning models for 7-day horizon.



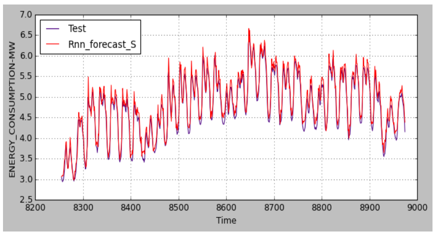
(a) Forecasting values (red) and actual values (blue) - LR model.



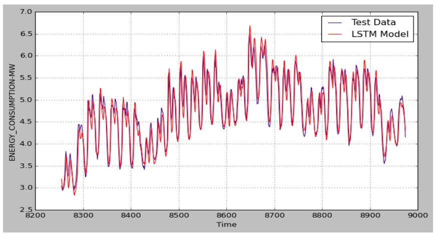
(b) Forecasting values (red) and actual values (blue) - MLP model.



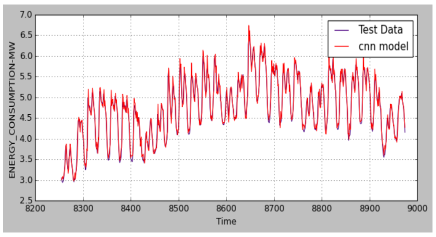
(c) Forecasting values (red) and actual values (blue) - RNN (sequence to vector) model.



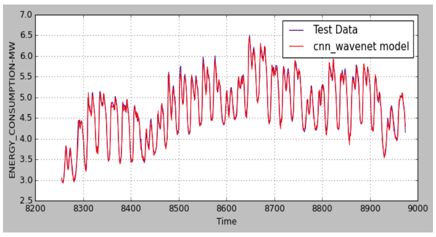
(d) Forecasting values (red) and actual values (blue) - RNN (sequence to sequence) model



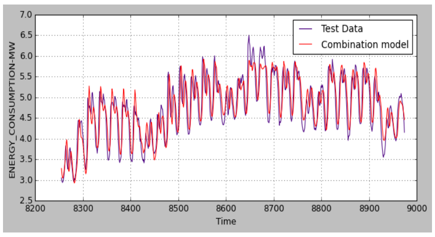
(e) Forecasting values (red) and actual values (blue) - LSTM model model.



(f) Forecasting values (red) and actual values (blue) - CNN model.



(g) Forecasting values (red) and actual values (blue) - CNN-Wavenet model.



(h) Forecasting values (red) and actual values (blue) - Combination model

Figure 5.16: Forecasting values by machine learning models For 30-day horizon.

5.2.7 SVM

SVM is implemented in Weka as SMOreg [47]. The parameter C is the width of the tube. The value of C is determined by the user. I have tried out different SVM models (i.e., using radial basis kernel function) with different values of the parameter C to find the optimal one.

Table 5.8 summarizes the performance of the SVM with different C values, for both 7 days and 30 days. The performance is not improved massively by $C = 5$ in 30-day horizon, then I set $C = 2$, to reduce the computational time. In the case of 7-day horizon, I set $C = 5$. Table 5.9 summarizes the performance of the SVM with the polynomial kernel function. I tried two values of exponent. For more complex data, higher values of exponent should be used. For the given dataset, the optimal value of exponent is 1 for both 7-day horizon and 30-day horizon, this suggests that the data is not complex and has high correlation between the observations. Also, the Pearson VII kernel function is tested. It is a general kernel function and the model behaviour changes by changing the value of w. The performance of the Pearson VII kernel function (i.e., called PUK in Weka) is summarized in Table 5.10. I conclude that SVM with the polynomial kernel function, exponent =1, performs best on the given data. Figure 5.17a shows the forecasting results of SVM model with the polynomial kernel function, $C = 5$, and exponent = 1 for the 7-day horizon, and Figure 5.17b shows the forecasting results of SVM model with the polynomial kernel function, $C = 2$, and exponent = 1 for the 30-day horizon.

Table 5.8: Performance of the SVM for different values of tube width C.

	7 days				**30 days**			
C	**MAE**	**MAPE**	**RMSE**	**MSE**	**MAE**	**MAPE**	**RMSE**	**MSE**
0.1	0.4388	10.2549	0.5276	0.2783	0.149	3.0863	0.2142	0.0459
1	0.1722	4.1106	0.2329	0.0542	0.0969	2.0207	0.1458	0.0213
2	0.1411	3.3882	0.2004	0.0402	0.093	1.9406	0.1398	0.0195
5	0.1189	2.8641	0.1738	0.0302	0.0911	1.8988	0.1348	0.0182
10	0.1113	2.6858	0.1639	0.0269				

Table 5.9: Performance of the polykernel-SVM for different values of exponet.

	7 days				**30 days**			
exponet	**MAE**	**MAPE**	**RMSE**	**MSE**	**MAE**	**MAPE**	**RMSE**	**MSE**
1	0.094	2.6223	0.1401	0.0196	0.0899	1.8813	0.1316	0.0173
2	0.1122	3.2938	0.1604	0.0257	0.6679	13.0918	0.8129	0.6608

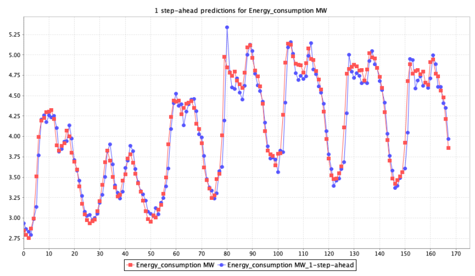
Table 5.10: The performance of PUK-SVM for different values of w.

	7 days				**30 days**			
w	**MAE**	**MAPE**	**RMSE**	**MSE**	**MAE**	**MAPE**	**RMSE**	**MSE**
0.05	0.4008	9.2042	0.5009	0.2509	0.3444	6.7263	0.4464	0.1992
0.1	0.3724	8.516	0.4707	0.2215	0.3322	6.49	0.4332	0.1876
0.5	0.363	8.1952	0.4733	0.224	0.3342	6.5345	0.4344	0.1887
1	0.3742	8.3994	0.4948	0.2448	0.3523	6.8789	0.4559	0.2078
5	0.3985	8.8714	0.5401	0.2917	0.4075	7.9385	0.5346	0.2858

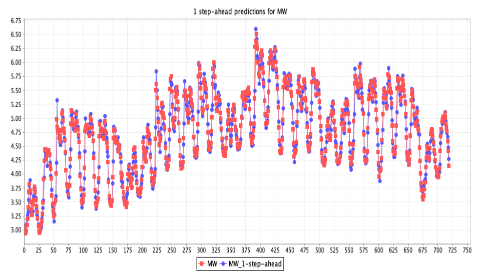
5.2.8 K-NN

KNN algorithm is an instance-based learning model. It does not begin to construct the model immediately as in regression models, instead it saves training examples. When new instance arrives, its distance to the training examples is measured in order to forecast its target value. KNN is called a lazy learner, since it postpones the work until the new query arrives. Then each new instance is treated separately. The KNN algorithm in Weka is used to build the model. We used non-weighted KNN. During using non-weighted algorithms, noise can greatly affect the algorithm accuracy. In order to deal with the problem, finding an optimal constant K is required, so only the majority of the class will influence the output, not a single nearest neighbor. The optimal value of K depends on the data characteristics such as data quality and complexity of the data.

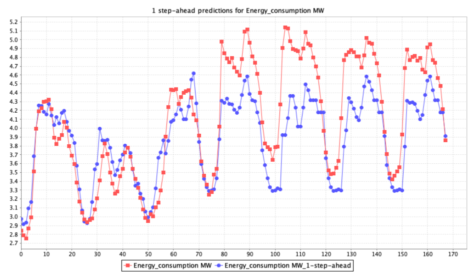
The model performance on the test data with different number of K were tested. The results of the experiments are summarized in Table 5.11. The best performance was achieved for $K = 3$ for 7-day horizon and $K = 12$ for 30-day horizon. The models with smaller K "learn" the training data too well, while their performance is not good enough on the test data. The big value of K, suggests that there is a noise in the data. Figure 5.17c shows the forecasting results of KNN model for 7-day horizon, and Figure 5.17d shows the forecasting results of KNN model for 30-day horizon. We can observe that the K-NN model yields worst results comparing to SVM.



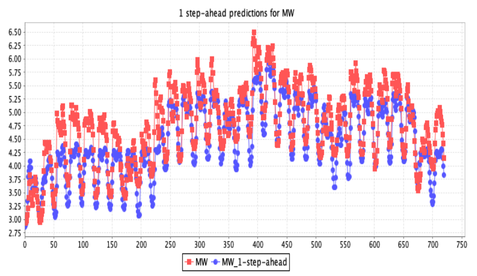
(a) Forecasting values (blue) and actual values (red) - SVM.



(b) Forecasting values (blue) and actual values (red) - SVM.



(c) Forecasting values (blue)and actual values (red) - KNN.



(d) Forecasting values (blue)and actual values (red) - KNN.

Figure 5.17: Forecasting results of SVM and KNN models in Weka.

Table 5.11: KNN performance for different number of neigbors (K).

	7 days				**30 days**			
K	**MAE**	**MAPE**	**RMSE**	**MSE**	**MAE**	**MAPE**	**RMSE**	**MSE**
1	0.3628	8.5505	0.4391	0.1928	0.3395	7.2306	0.4142	0.1715
2	0.3579	8.3937	0.4292	0.1842	0.3304	7.0296	0.4009	0.1607
3	0.3234	7.4846	0.407	0.1656	0.3257	6.9309	0.3949	0.156
5	0.3644	8.487	0.4344	0.1887	0.3292	6.9672	0.397	0.1576
7	0.363	8.4191	0.4354	0.1895	0.328	6.9356	0.3946	0.1557
9	0.3701	8.5807	0.4428	0.1961	0.3278	6.9167	0.395	0.1561
12	0.378	8.7743	0.4496	0.2021	0.3275	6.9028	0.394	0.1552
15	0.3837	8.8947	0.4573	0.2091	0.3292	6.9214	0.3963	0.157
18	0.3942	9.1074	0.4736	0.2243	0.3286	6.9019	0.3976	0.1581
21	0.4027	9.2877	0.4843	0.2345	0.3282	6.8903	0.3974	0.1579
24	0.4069	9.3737	0.4892	0.2394	0.3293	6.9128	0.399	0.1592

5.3 Performance evaluation of machine learning and statistical methods

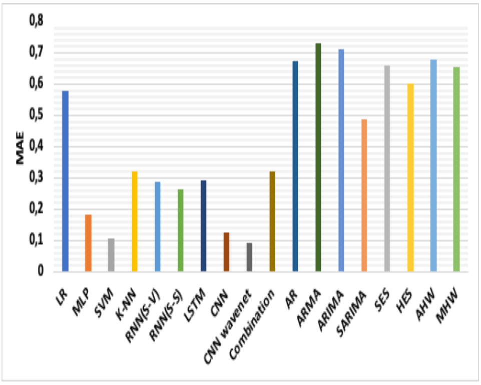
The empirical results that are achieved arranged into two case studies: (i) results from predictive models applied for short term load forecasting as illustrated in Figure 5.18, and (ii) results from predictive models applied to midterm load forecasting as illustrated in Figure 5.19.

Four different evaluation measures are used for short term load forecasting (STLF) and mid-term load forecasting (MTLF): i) Mean absolute error (MAE) as shown in Figures 5.18a, and 5.19a, ii) Mean square error (MSE) as shown in Figures 5.18b, and 5.19b, iii) Root mean square error (RMSE) as shown in Figures 5.18c, and 5.19c, and iv) Mean absolute percentage error (MAPE) as shown in Figures 5.18d, and 5.19d.

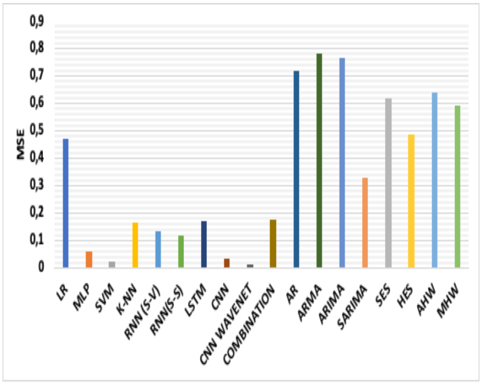
Finding the method which performs best on the given data is the answer of the first research question in the master thesis. In statistical methods, MA gives best result as obviously seen in the Figure 5.6, but as mentioned before, this method can not be used in the prediction because in order to get the value for the next step, we need the previous values to be actually observed. But if we compare the performance among ARIMA models, we find that SARIMA performs best on the STLF and also on the MTLF. SARIMA model has captured the seasonality of the data, the trend as well, and the residual. The AR Model gives a good performance since it captures the correlation in energy consumption. ARMA model has also performed well, but it did not capture the seasonality well. ARIMA model has failed to capture the seasonality, so it yields the highest error values on MTLF while ARMA gives the highest error on the short term.

From the ES models that have been applied to the dataset (SES, HES, AHW and MHW), the HES model achieves the best performance where it captures trend. While the SES use exponentially decreasing weight for past data, it also works well with the data. There was not an enormous difference between MHW and SES, in the opposite of the AHW model which gives the worst result among all the ES models.

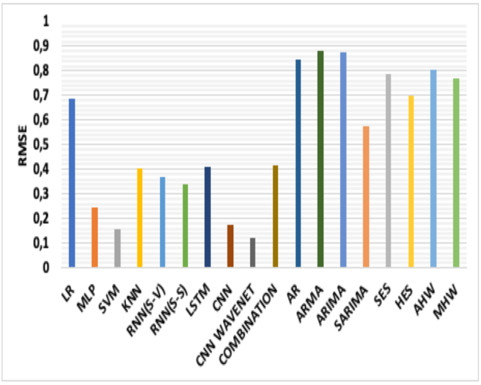
In the machine learning methods, the model that shows the most promising results for both



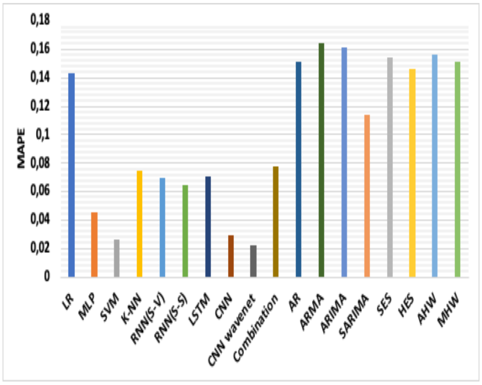
(a) Mean Absolute Error (MAE)



(b) Mean Square Error (MSE)



(c) Root Mean Square Error (RMSE)



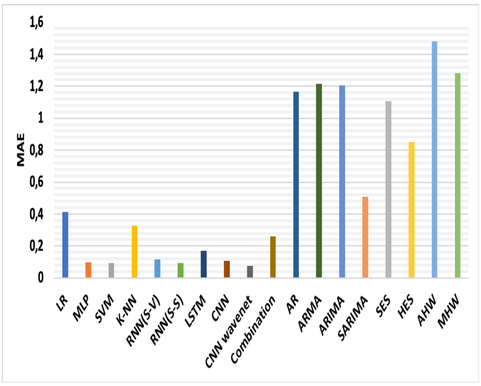
(d) Mean Absolute Percentage Error (MAPE)

Figure 5.18: Performance evaluation of the statistical methods and machine learning methods for short-term load forecasting (STLF)

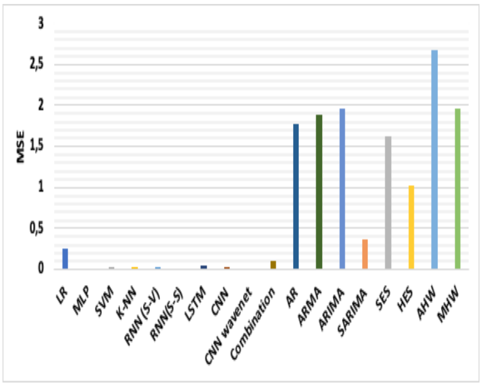
STLF and MTLF is the CNN Wavenet, while LR model yields the worst result on the given data. It seems that LR model did not find any specific patterns in the dataset. The same is happening with the K-NN model applied for the MTLF. But the difference is in the running time, the K-NN has taken more time to run than the LR.

From the evaluation figures, we can observe that the results of LSTM are worst than the one of RNN models. But if we compare the performance among RNN models, we find the RNN (Sequence to Sequence) gives the best performance. In addition, RNN (Sequence to Sequence) is faster in running time in the opposite of the RNN (Sequence to vector). SVM performs very well in the given dataset, but the finding the best SVM model took very long time. Beside this, the model is computationally very heavy. For instance, the model which gives the forecasting results that shown in Figure 5.17b runs for more than an hour.

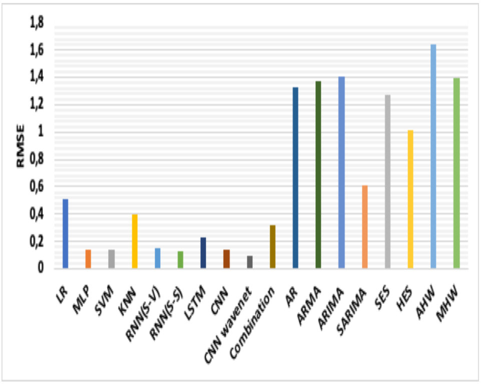
Creating the different combinations of CNN, LSTM and MLP did not give any improvement in performance as shown in Figures 5.18 and 5.19.



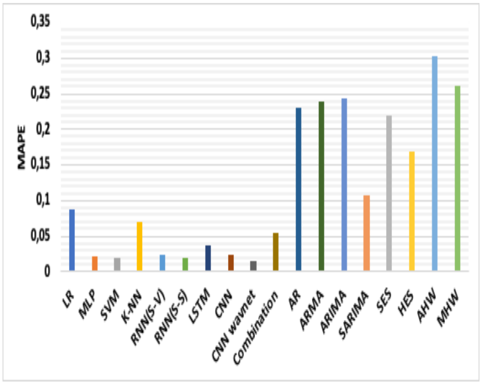
(a) Mean Absolute Error (MAE)



(b) Mean Square Error (MSE)



(c) Root Mean Square Error (RMSE)



(d) Mean Absolute Percentage Error (MAPE)

Figure 5.19: Performance evaluation of the statistical methods and machine learning methods for mis-term load forecasting (MTLF)

Further discussion about the results of the implemented time series forecasting models are given in Chapter 7 (discussion).

Chapter 6

Case studies

This chapter presents two case studies. In the first section, I will present a case study of modeling an intelligent energy management system (IEMS). Then, the demand side management (DSM) case study will be presented in the next section.

6.1 Case A: An intelligent energy management system

Energy management at different levels is a hot topic in Norway[1]. IEMS as illustrated in Figure 6.1, consists of a set of sources to produce energy (i.g., wind power station and solar power plant), a set of heterogeneous loads (i.e., houses, cabins, commercial, and industrial buildings), energy storage, and the connection to grid. The IEMS aims to exploit the sources (i.e., renewable sources) to cover the load demand, and minimize the amount of energy required from the grid by managing the energy storage. The connection to the grid is in two-direction. This means the IEMS buys and sells energy to the grid. The IEMS model is implemented in MATLAB, and it is based on the micro-grid energy management model example which is implemented by LeSage [85].

6.1.1 Load forecasting

Load forecasting is an essential part of the IEMS model. The load is predicted depending on the previous values (i.e., time series forecasting). Load forecasting is a complex process, and it requires much historical data and advanced forecasting methods, such as machine learning methods, to get accurate forecasting values. In this case study, we perform short-term load forecasting (24 hours).

[1]Statnett har testet om elbiler kan regulere kraftnettet: – En ekstremt billig løsning. Available at: `https://www.tu.no/artikler/`. (As of June 2020)

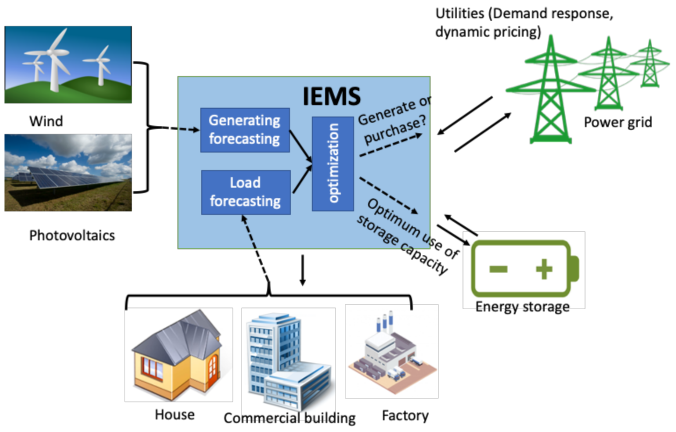


Figure 6.1: Intelligent energy management system model.

6.1.2 Distributed sources

The sources in the model are photostatic arrays (PV) and wind turbines. The IEMS will purchase energy from the grid or get energy from the energy storage when it is needed.

Photostatic arrays

We used the data for a sunny day which is available by examples in MATALB[2]. The sunrise is at 05:00 and the sunset is at 19:00. The panels area is 2500 m^2 and the panels' efficiency is 30%. The power which is generated from PV is shown in Figure 6.2. The PV farm produces energy proportional to three factors: i) the size of the area covered by the PV farm, ii) the efficiency of the solar panels, and iii) the irradiance data.

Wind turbines

The generated power from wind turbines depends on the wind data. We used the simplified model of a wind farm, which produces electrical power following a linear relationship with the wind. When the wind reaches a nominal value, the wind farm produces the nominal power. The wind farm stops when the wind speed exceeds the maximum wind value until the wind gets back to its nominal value. The generated power from the wind turbine (PW) is shown in Figure 6.2.

6.1.3 Demand response and prices

Demand response is a change in the power consumption of an electric utility customer to better match the demand for power with the supply [95]. Case B which is presented in the

[2]Simplified model of a small scale micro-grid, Avialble at: `https://se.mathworks.com/help/physmod/sps/examples/simplified-model-of-a-small-scale-micro-grid.html`. (As of July 2020)

next section, is about the demand side management. The electricity prices generally follow the demand fluctuations. The prices are high during the load peaks and low in the other periods.

6.1.4 Optimisation

The optimisation process is the core of the IEMS model. The optimizer is responsible for managing the energy storage (i.e., when is to charge and dis-charge), and when is to buy energy from or sell energy to the grid.

Energy storage

The energy storage (i.e., batteries) is used to store energy for future use when there is a surplus in the power in the microgrid because high generating from PV and wind turbine, or the electricity price in the external grid is low. Also, it provides additional power if there is high demand in the microgrid. The energy storage is controlled by the optimizer.

Objective function

The main objective is to minimize the total cost of power from the grid while meeting load with power form wind turbine, PV, energy storage, and grid. Then the objective function is modeled as:

$$min\left(Cost_{tot} = \sum_{i=1}^{N}(Cost_{grid}(i).E_{grid}(i))\right) \tag{6.1}$$

where $Cost_{tot}$ is the total cost of electricity purchased from the grid, i is the time step, N is the maximum time steps, $Cost_{grid}(i)$ is the electricity price at step i, $E_{grid}(i)$ is the energy bought from the grid at time step i. The constraints are:

- Satisfy power load with power from PV, PW, grid and energy storage (i.e., power balance).

  $$P_{load}(i) = P_{grid}(i) + P_{pv}(i) + P_{pw}(i) + P_{es}(i) \tag{6.2}$$

  where $P_{load}(i)$ is the load at times step i, $P_{grid}(i)$ is power from grid, $P_{pv}(i)$ is power from photostatic arrays, $P_{pw}(i)$ is power from wind turbines, and $P_{es}(i)$ is absorbed power or provided power by energy storage (i.e, charge/ dis-charge).

- Power input/output to the energy storage is:

  $$ES(i = ES(i-1) + P_{es}(i).dt \tag{6.3}$$

  where $ES(i)$ is the energy which is stored in the energy storage at times step i, $P_{es}(i)$ is power which is absorbed or provided by energy storage in the duration between time step i and $i-1$.

Results

The optimization results are shown in Figure 6.2. The optimization problem is implemented in MATLAB by using the linear programming optimizer in the optimization toolbox[3]. The optimization problem is defined to minimize the objective function in Equation 6.1 with the constraints in Equations 6.2 and 6.3. The time step is 5 min, $N = 288$, and the energy storage at 0 time is 50%. Since the time series load forecasting model predicts future values in hourly-basis steps (i.e., 24 steps ahead in the case of one day), we have to reshape data from hourly-basis to 5 min. We have used "remap"[4] function in the MATLAB to match the time step to 5 min. The behaviour of the energy storage is shown in the amber line in Figure 6.2. The power from the grid is shown in the red-dot line. The power from grid takes positive values and negative values (sell energy to the grid).

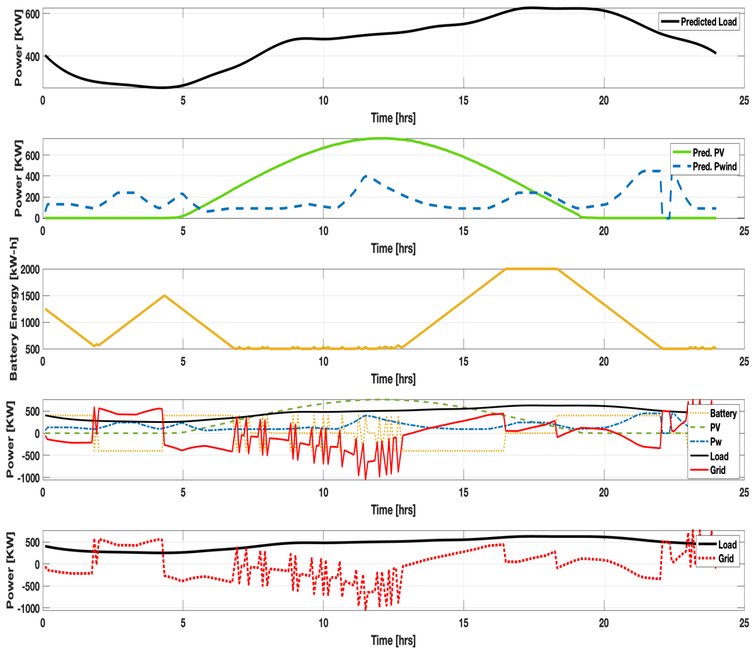


Figure 6.2: IEMS optimisation results.

[3]Linear Programming in Optimization Toolbox - MATLAB `https://se.mathworks.com/discovery/linear-programming.html` (As of July 2020).

[4]Remap numerical values `https://www.mathworks.com/matlabcentral/fileexchange/54404-remap-numerical-values`, MATLAB Central File Exchange. (As of July 2020).

6.2 Case B: Demand side management

Efficient energy demand side management (DSM) is an essential part of the current hot research topics such as: smart grids and smart cities. The residential households can play the major rule in order to have an efficient energy system with less energy loss and cheap prices. The residential households can help in reducing the load peaks (i.e., flattening the peaks) by scheduling the usage of shiftable appliances such as washing machine, dish washer, and clothes dryer. By using heuristic optimization techniques such as genetic algorithm (GA), scheduling of shiftable devices can be realized to minimize the overall cost of electricity payment and keep resident's comfort. Figure 6.3 shows an overview of the DMS model. The consumers can control their activities depending on the predicted dynamic prices, predicted load, and predicted local power generating. The model is implemented in MATLAB.

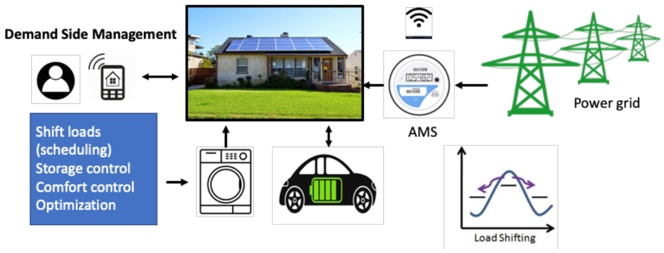


Figure 6.3: Demand-side management model

6.2.1 Shiftable appliances

In general, electrical appliances are categorized into i) shiftable, and ii) non-shiftable appliances. Non-shiftable appliances are used in a specific period with non-changed power level. These appliances include essential equipment, (such as; lights, cooker, kettle, ventilation, etc). In contrast, shiftable appliances, (such as; washing machine, electrical vehicle and clothes dryer) can be moved to another time to use. For example, we can charge the electrical vehicle battery during the night to avoid the peak hours to reduce the energy costs. Table 6.1 summarizes the shiftable appliances in this experiment, and the operational time, the time of usage and the power.

Table 6.1: Parameters of shiftable appliances.

Appliance	Start time (h)	End time (h)	Power (KW)	Operational time (h)
Washing machine	7am	7pm	1.5	2
Clothes dryer	9am	9pm	4	2
Dish washer	6am	10pm	3	2
Electrical car	16pm	6am	4	4

6.2.2 Optimisation

In order to reduce the electricity consumption cost, the user can schedule the shiftable appliances to take their job on non-peak hours. The non-shiftable devices must operate at any time depending on their characteristics or the user's needs. Then the reduction of electricity bill is not possible with non-shiftable appliances. But the electricity usage cost can still be reduced by scheduling the shiftable appliances., GA is used to solve this scheduling problem after defining the problem and the objective function.

Objective function

The overall objective function is to minimize the electricity bill by scheduling the shiftable devices to take their jobs at optimal time. The objective function has two parts: minimizing the electricity bill and minimizing the waiting time to keep the user's comfort. Each shiftable appliance has start time (st), end time (et), and operation time (ot) as in Figure 6.4.

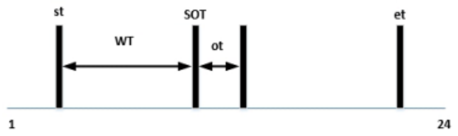


Figure 6.4: Parameters of appliance

The electricity consumption cost at each hour is the total energy consumption at this hour multiplied by the electricity price at this hour as follows:

$$Cost_i = p_i \sum_{k=1}^{m} (E(a_k, t_i)) \tag{6.4}$$

where $Cost_i$ is the total electricity consumption cost of an hour i, p_i is the electricity price of an hour i, $E(a_k, t_i)$ is the energy consumption by an appliance k at an hour i. m is the total number of appliances. Then the daily cost is the summation of the cost of each hour as follows:

$$Cost = \sum_{k=1}^{24} (E(a_k, t_i)) \tag{6.5}$$

It is important to include the residents' comfort also. We consider the user wish to switch on the appliance at the given start time (st), as shown in Figure 6.4. Then, we schedule the shiftable appliances to minimize the waiting time also as follows:

$$WT = \sum_{k=1}^{m} (SOT_k, st_k)) \tag{6.6}$$

where WT is the total waiting time for all the shiftable appliances, SOT_k is the start operation time for device k, and st_k is the given possible start time by the user for the appliance k. Then the objective function is modeled as:

$$min\left(w_1\Big(\sum_{k=1}^{24}(E(a_k,t_i))\Big)+w_2\Big(\sum_{k=1}^{m}(SOT_k,st_k)\Big)\right) \tag{6.7}$$

where w_1 and w_2 are weights of two parts of objective function and their values are between 0 and 1, and $w_1+w_2=1$.

GA for appliances' scheduling problem

Genetic algorithm (GA) is an optimization and search technique based on the principles of genetic and natural selection [96]. A GA allows a population composed of many individuals to evolve under specified selection rules to a state that maximize the fitness (i.e., minimizes the cost function in our case). In this scheduling problem, the objective is to find the optimal start operation time for the shiftable appliances. Then, the chromosome length is the number of shiftbale appliances, and the variables are the start operation times (SOP_s) for the appliances as follows:

$$chromosome=[SOT_1,SOT_2,\ldots,STO_m] \tag{6.8}$$

where m is the number of shiftable appliances. and the SOT_s take only integer values.

Results

For scheduling; we got the "spotpris" from Nordpool [5] for this region in week 47 as shown in Figure 6.5, middle figure. Then, we counted the bill for this end user for week 47, 2019 which was 300.67 NOK. After that, we optimize it by finding the optimal scheduling. We used GA with defined parameters in Table 6.2. The objective function in Equation 6.7 with $w_1=0.7$ and $w_2=0.3$.

Table 6.2: GA parameters.

Number of optimisation variables	4
Upper limit on optimisation variables	[19,21,22,28]
Lower limit on optimisation variables	[7,9,6,16]
Maximum iteration	100
Population size	100
Selection rate	0.8

The optimization results are illustrated in Figure 6.5. Upper part shows the original energy consumption and the optimized one. The middle figure shows the prices on hourly-based.

[5]Market data from Nord Pool. Available at: `https://www.nordpoolgroup.com/Market-data1/Dayahead/Area-Prices/NO/Hourly/?view=chart` (As of July 2020).

The bottom figure shows the consumption cost for both original consumption and optimized consumption by applying scheduling. The result shows that this user could save 10 NOK if he scheduled the shiftable appliances even if the electricity prices were not varying massively comparing to other time spots [6].

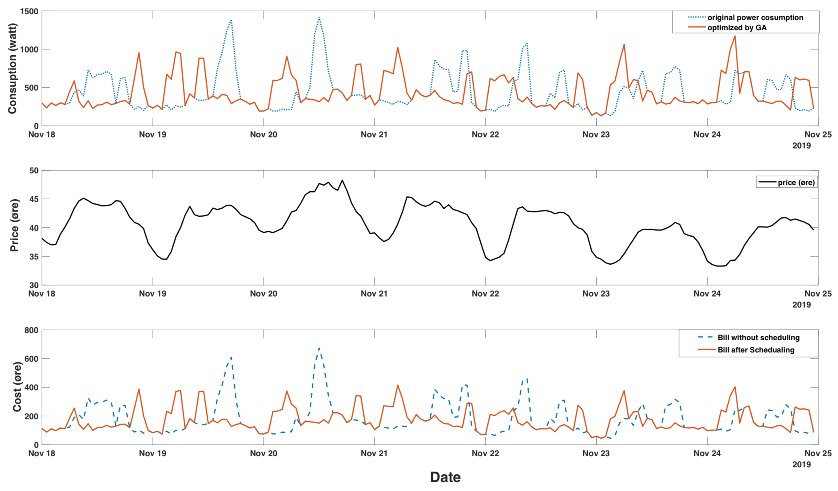


Figure 6.5: Case A: optimization results

[6]Så mye svinger strømprisen i løpet av døgnet. [online] available at: `https://enerwe.no/sa-mye-svinger-stromprisen-i-lopet-av-dognet/144438`. (As of July 2020

Chapter 7

Discussion

This chapter presents a general discussion about the results of the implemented time series forecasting models and applying them for short-term and mid-term load forecasting. Further, a discussion about the conceptual models of an intelligent energy management system and demand side management is given.

7.1 Load forecasting

Steadily increasing of energy consumption and its influence on the environment makes challenge in finding new and clean energy sources. Therefore, using of the efficient energy management systems is an important issue. Load forecasting is a treasure that saves much effort in energy management. The forecasting results can be used in planning, schedule maintenance, and identifying energy-saving opportunities.

7.1.1 Time series forecasting methods

In this work, different forecasting models have been applied to the electricity consumption dataset. The forecasting models are implemented by using different statistical and machine learning methods. Machine learning methods are newly applied to the time series forecasting, in comparison with statistical methods which have long history [97]. Empirical reached has proved at the machine learning algorithms for the time series prediction, give outstanding performance and result in comparison with the statistical models [98, 99]. For statistical methods, the performance obviously depends on the data properties (i.e., attributes), and the success in defining a model with appreciate parameters to capture these properties. On contrast, the machine learning models (i.e., data driven models) do not depend on the data distribution.

The result form this work is divided into two categorizes based on the prediction horizon, either short-term or mid-term. The statistical and machine learning methods are applied for short term forecasting and for mid-term forecasting in order to compare the results between them.

The first research question is about evaluation of different statistical and machine learning prediction methods to find the best method which gives the best performance on the electricity consumption dataset. In spite of the fact that time series forecasting is still a complex task and finding one model that fits a complex data such as electricity complex data, is a hard process, but we have succeed in finding the best model which gives very good result. Based on the evaluation measures of the forecasting model's performance, and the visualization of the forecasted values, the forecasting model that provided good results compared to the all other methods is the CNN wavenet model. It performs best on both short-term load forecasting and mid-term load forecasting.

The CNN wavenet is an efficient network with simple architecture. It can be used as a strong baseline for time series prediction. There is an ability to improve the method by using a large number of layers and filters to increase the ability to learn non-linear dependencies.

MLP, RNN, LSTM, and SVM yield very close results and good performance also. In contrast, the LR and K-NN did not perform well in comparison to the other machine learning models. RNN gave results better than LSTM even if it is much simpler. This emphasises that simpler type of models give as an accurate result in many cases [100].

Among statistical methods, the SARIMA was the best promising method in both short-term load forecasting and mid-term load forecasting. This emphasizes that SARIMA and ARIMA models perform better than exponential smoothing algorithms when the time series data are relatively long and have many regularities.

Then, the answer to the first research question is, the CNN wavenet model outperforms better than other used methods. Machine learning models are data-driven models that can perform very good on the non-linear and noisy data.

The second research question is to investigate how the available data should be pre-processed and analysed to improve prediction accuracy. The data is collected from advanced metering systems. The time series is a total electricity consumption of different 1112 consumers from Aalesund municipality. Statistical analysis is carried to understand the time series data and prepare the data for the use in the forecasting models. The data decomposition into time series components shows that data has trend (irregular trend), seasonality and noise. Another prove of noise in the data is the performance of LR and K-NN models. The LR performs well when the data is linearly separable (i.e., separable by line). Also, the big value of K (neighbors) in K-NN model indicates that the data is highly correlated or has noise.

Data analysis is a time-consuming process, but it is an important step in order to understand the features in the data. This understanding helps in selecting the forecasting model's parameters in the statistical models (parametric models). Machine learning models (no-parametric models) do not require previous knowledge on the type of data distribution, it will work on, but learns it directly from the data used for training. We have to take care that our machine learning models are not over-fitted or under-fitted, and we have to select good learning rate. Keras has good automated solutions to these considerations, while Weka has not. In K-NN and SVM, I have to try manually different parameters to find the best model.

Analysis process of the time series dataset is very important part of the project. Also, based only on standard error measures (e.g., mean percentage error) in evaluating the forecasting models, can be very misleading some times. For example, random walk models perform well on time series data even if it is generated from a stochastic process [100]. Therefor, statistical analysis and visualization (in time and frequency domains) of the time series data help in avoiding such mistakes.

7.2 Applications of time series load forecasting

This thesis has investigated the applications of the time series load forecasting in different domains. In Norway, all electricity consumers received advanced metering system (AMS) by 1 January 2019. AMS provides power utilities and consumers with better information about electricity load consumption and prices. This information facilitates better services and opportunities to engage consumers in demand response as shown in the case studies (Chapter 6). Time series forecasting is the core of energy management systems. Good load forecasting accuracy helps energy system parts (i.e., power utilities and consumers) to plan efficiently. Table 3.1 summarizes some potential applications of the time series load forecasting models. The duration of prediction depends on the application. Short-term load forecasting is used in a wide area of applications such as energy purchasing, transmission and distribution planning, demand side management, and in operations and maintenance [101]. While the mid-term load forecasting is used in the planning for seasonal peak (summer and winter) in addition to the previous mentioned applications. In order to investigate the importance of the load forecasting in different applications, two case studies were done in this master project as presented in the Chapter 6. In the following, I will discuss the results from these case studies.

7.2.1 Case A: An intelligent energy management system

In this case study, a model of an intelligent energy management system (IEMS) at micro-grid level is proposed. The aim of the IEMS model is to exploit the renewable energy sources in the local grid to cover the load demand, and minimize the purchased energy from the main grid. This is done by balancing the power load with the power from grid, renewable sources, and energy storage. In this case, the load forecasting accuracy is very important in order to have a reliable system.

This implemented model, which is illustrated in Figure 6.1, is a conceptual model to illustrate how the IEMS works. The management of energy storage and purchasing process from the grid are managed by the IEMS depending on the forecasting values. The simulation results for 24 hours, shows the efficiency of the presented solution. It is clear from the simulation results which are shown in Figure 6.2, the power from the renewable sources are exploited efficiently by optimizing the energy storage system. The simulation results show also that the IEMS purchases more electrcity from the main grid when the prices are low (before 05:00 and after 16:00), and sells energy to the grid when the prices are high (between 08:00 and 15:00).

This example shows the importance of forecasting models in designing new smart solutions to meet the increasing in the demand. These smart solutions integrate all the stockholders in order to meet the challenges.

This conceptual model shows the importance of load forecasting and how the forecasting model can be used in smart girds. This is partially answer the third research question. The next case study, shows how the load forecasting model can be used in demand side management.

7.2.2 Case B: Demand side management system

In this case study, a model of demand side management is proposed. The consumers can control their activities in order to have an efficient energy system with less energy loss and cheap prices. The consumers schedule the shiftable appliances to take their jobs at optimal time. This is an optimization problem which is solved by using genetic algorithm (GA). The simulation results show how the consumer can save 10 NOK daily. This means 300 NOK monthly and 3600 NOK annually.

The load forecasting model is the core of the demand side management system. The proposed conceptual model in this case study, can be extended in many different ways such as including solar panels.

This system can be used to increase the user's awareness and designing smart solutions at different levels. These case studies (case A and B) answer the third research question. In this project, we have successfully integrated the energy consumption forecasting model in designing different smart solutions such as IEMS and demand side management.

Of course, there are still many other applications where we can use the load forecasting models, such as vehicle-to-grid systems which have potentials in Norway because the increasing number of electrical cars in the country [102].

Chapter 8

Conclusions and future work

The overall goal of this project is to build a load forecaster to be used in designing smart systems such as smart grids, smart cities and smart houses. The load dataset includes the electricity consumption data from 1112 electricity meters from Aalesund municipality. The dataset is analysed, and the total consumption time series is used to evaluate forecasting models in order to find the best performing model among them. Many statistical and machine learning methods are evaluated in this work, in order to find the best method which gives the best performance on the given dataset. All the evaluated methods have been applied on both short-term load forecasting and mid-term load forecasting. Statistical methods were mainly in use for time series forecasting until recently when machine learning methods became more popular and promising. In order to answer the first research question which is about finding the best forecasting method for the given dataset, many forecasting models are implemented by using different statistical and machine learning methods. These methods are selected based on the carried literature review about time series load forecasting. The statistical methods include different ARIMA and exponential smoothing methods, and the machine learning methods include many algorithms (such as; linear regression, RNN, CNN, K-NN and SVM). Four different performance measures are used in the evaluation process. These methods are: i) mean absolute error, ii) mean square error, iii) root mean square error, and iv) mean absolute error. The overall result shows that the CNN wavenet preformed best on our dataset. This result is aligned with previous studies [103]. CNN wavenet works good on non-linear noise data. It has the ability to learn non-dependencies and capture sudden changes. Among the statistical methods, SARIMA performed a bit better than other methods. This means the time series has seasonality in addition to the trend and noise components.

The historical electricity consumption data are pre-processed and statistically analysed, in order to answer the second research question which is about how is the analysis of the historical load data can improve the accuracy of the load forecaster. The statistical analysis shows that the data has seasonality, trend and noise. The data analysis shows that the data are points are highly correlated, and the data has big noise. Also, the electricity consumption data is highly correlated with temperature only on a daily basis.

The third research question is about how the load forecasting model can be used in different applications. Two models were implemented to show the integration of the load forecaster in recent emergent applications. In the first model, load forecasting model is used as a main core of the intelligent energy management system model. The simulation result shows how the model meets the load demand efficiently. In the second case, load forecasting model is used to design a demand side management model. The simulation result shows how the consumers can do appliance scheduling based on the predicted load. These examples prove the importance of the time series forecasting model in designing new solutions to meet the challenges that arise by the increasing in load demand and other environmental issues.

8.1 Contribution

Load forecasting modeling is the core of hot research topics such as smart grids, micro-grids, low voltage grids, smart, green and sustainable cities. This work provides a methodology to design a good load forecasting model which depends on the available data, and show the effectiveness in applying the forecasting model in different domains. The main contributions are:

- Proposed a methodology to find the best time series forecasting model which performs best on a specific data.
- Results show that machine learning methods, particularly CNN wavenet, perform best on the highly correlated and noisy data such as electricity consumption data.
- Illustrated how the load forecasting model can be used in energy management systems.

8.2 Future work

Time series load forecasting modeling and its application is a hot topic in the research due to the importance of the topic and its benefits to every one. This work can be extended in different ways as following:

- In future works, we want to explore the multivariate scenario which still constitutes an important gap in the literature. In multivariate, we can include more time-dependent variables in addition to the electricity consumption, such as weather data.
- Categorise the electricity consumption data depending on different criteria, such as consumers' type (i.e., apartment, house, fabric, ..., etc), and sub-regions. Our dataset consists of data from 1112 consumers, but we do not know which data series belongs to which type of user.
- Investigate the effects of other relevant data such as households activity and holidays on the prediction accuracy. These data are more important for short-term and very short-term load forecasting.
- Investigate the use of time series load forecasting in other applications, such as integration of the load forecaster in smart grids and smart cities frameworks.

Chapter 9

Legal and ethical considerations

The power utilities are making use of advanced technologies to increase the reliability, resilience, intelligence, and efficiency of the existing power grid to be smart grid (SG). Norway follows the European Union (EU) in applying programs to tackle energy efficiency and reduce emissions. Advanced metering systems (AMS) are the core of the SG solutions. AMS allows collecting large amount of data about the electricity consumption. These electricity consumption data (historical and real-time) can be used to find the consumption pattern which can be used to identify the households' life pattern. The question is: does the SG solutions give away household privacy? [104].

AMS are installed in all Norwegian houses by January 2019. The installation companies are responsible for the installation. The new smart meters record electricity consumption on an hourly basis and automatically send consumption information to the grid company. This provides faster and more accurate collection of meter values and a basis for the invoice that will be sent to the electricity customers. In addition, the grid companies will be able to use the information to operate the networks more efficiently. The introduction of new meters is regulated by the Regulations on measurement, accounting, billing of network services and electrical energy, the grid company's neutrality,. . . etc.[1]

9.1 Use of measurement data

The use of measurement data, which are recorded by AMS is subject to the Personal Data Act[2] by The Norwegian Data Protection Authority (DPA)[3]. This means that the grid company and the power supplier can only use the personal information needed to bill the customer. What information is to be recorded and what the meter is to do is regulated in the functional requirements (Regulations on measurement, accounting, billing of network

[1]`https://lovdata.no/` (As July 2020)
[2]`https://lovdata.no/` (As July 2020)
[3]`https://www.datatilsynet.no/` (As July 2020)

services and electrical energy, the grid company's neutrality, etc., Chapter 4, Section 2). In addition to the above, the customer has the right to decide who can access their own data. The online company cannot store customer data for more than 3 years (Personal Data Act, Chapter 3, sections 9 and 10).

9.2 Power consumption information is personal information

Power usage information is basically associated with a meter number at a specific address, not a person. However, when the meter is again linked to a homeowner, the power consumption information can be traced back to a specific person. This can be the subscriber himself or another person, such as a tenant. Power consumption data at hourly intervals can tell when people are on vacation or at work, when they are asleep or are awake and similar information.

Power consumption information is to be considered a personal information and must be treated as such (`datatilsynet.no`). The power companies and relevant players must ensure that the Personal Data Act is complied. This includes ensuring that information security is adequately safeguarded and that the information is only used for the purpose for which it has been granted permission.

9.3 Access to more personal information through the HAN port

All of the new smart meters are equipped with a physical output, called the Home Area Network (HAN) port. By connecting to the HAN port, the consumer will be able to access additional information about his own electricity consumption. He decides how to use this information.

9.4 Legal and ethical consideration in this project

The purpose of this project is to use advanced methods and technology to facility the resident's life while respecting the privacy and confidentially interests of all relevant stakeholders. The power consumption data used in this project is anonymized by the data supplier and It is not possible to connect the measurement data to real consumers. The Norwegian organizations who deal with personal information (collecting, processing and storing) must fulfil the requirement by the Personal Data Act, and if they fail in doing that, they will get a huge fine from the DPA.

Bibliography

[1] Stanford.edu, *Convolutional Neural Networks (CNNs / ConvNets)*. Stanford Vision and Learning Lab (SVL), Stanford, CA, 2020.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[3] R. Weron, *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*. No. hsbook0601 in HSC Books, Hugo Steinhaus Center, Wroclaw University of Technology, June 2006.

[4] J. Kaur and I. Chana, "Electricity demand forecasting for a smart city," Master's thesis, Thapar University, 2015.

[5] M. A. Hammad, B. Jereb, B. Rosi, and D. Dragan, "Methods and models for electric load forecasting: A comprehensive review," *Logistics Sustainable Transport*, vol. 11, no. 1, pp. 51–76, 2020.

[6] L. Friedrich and A. Afshari, "Short-term forecasting of the abu dhabi electricity load using multiple weather variables," *Energy Procedia*, vol. 75, pp. 3014 – 3026, 2015. Clean, Efficient and Affordable Energy for a Sustainable Future: The 7th International Conference on Applied Energy (ICAE2015).

[7] C. Kuster, Y. Rezgui, and M. Mourshed, "Electrical load forecasting models: A critical systematic review," *Sustainable Cities and Society*, vol. 35, pp. 257 – 270, 2017.

[8] S. A. hady Soliman and A. M. Al-Kandari, *Electrical Load Forecasting: Modeling and Model Construction*. Burlington, USA, 1st ed. Butterworth–Heineman, 2010.

[9] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," *CoRR*, vol. abs/1302.6613, 2013.

[10] A. R. S. Parmezan, V. M. Souza, and G. E. Batista, "Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model," *Information Sciences*, vol. 484, pp. 302 – 337, 2019.

[11] P. J. Brockwell and A. Davis, R, *Introduction to Time Series and Forecasting*. Second Edition. Springer-Verlag, New York, 2002.

[12] C. D. Montgomery, L. C. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2015.

[13] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. Second Edition. OTexts: Melbourne, Australia., 2018.

[14] J. D. Kelleher, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Massachusetts Institute of Technology, 2015.

[15] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, p. e0194889, 2018.

[16] R. D. Snyder, A. B. Koehler, J. K. Ord, and R. J. Hyndman, *Forecasting with Exponential Smoothing: The State Space Approach (Springer Series in Statistics)*. Springer-Verlag Berlin Heidelberg, 2008.

[17] J. M. Lucas and M. S. Saccucci, "Exponentially weighted moving average control schemes: Properties and enhancements," *Technometrics*, vol. 32, no. 1, pp. 1–12, 1990.

[18] H. R. Shumway and S. S. David, *Time series analysis and its applications with R examples*. Manning Publications Co., 2017.

[19] J. W. Taylor and P. E. McSharry, "Short-term load forecasting methods: An evaluation based on european data," *IEEE Transactions on Power Systems*, vol. 22, pp. 2213–2219, 2007.

[20] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods (2nd ed.)*. New York: Springer., 2009.

[21] J. R. Hyndman, B. A. Koehler, D. R. Snyder, and S. Grose, "A state space framework for automatic forecasting using exponential smoothing methods," *International Journal of Forecasting*, vol. 18, no. 3, pp. 439 – 454, 2002.

[22] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5–10, 2004.

[23] J. Gardner and S. Everette, "Exponential smoothing: The state of the art," *Journal of Forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[24] R. P. Winters, "Forecasting sales by exponentially weighted moving averages," *Management Science*, vol. 6, no. 3, pp. 324–342, 1960.

[25] M. N. Islam and B. Sivakumar, "Characterization and prediction of runoff dynamics: a nonlinear dynamical view," *Advances in Water Resources*, vol. 25, no. 2, pp. 179 – 190, 2002.

[26] P. Harrington, *Machine Learning in Action*. Shelter Island, NY, Manning Publications Co., 2012.

[27] M. Negnevitsky, *Artificial Intelligence: A guide to intelligent systems*. Harlow, England, Pearson, Co., 2011.

[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv*, 2014.

[29] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[30] W. n Nazirah Wan Md Adna, N. Y. Dahlan, and I. Musirin, "Development of hybrid artificial neural network for quantifying energy saving using measurement and verification," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 9, no. 1, pp. 137–145, 2017.

[31] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman Hall/CRC, 2nd ed., 2014.

[32] S. Sharma, "Activation functions in neural networks," Sep 2017. Available at: `https://towardsdatascience.com/`. (Accessed: 30- Jun- 2020).

[33] B. A. Maind and W. Priyanka, "Research paper on basic of artificial neural networks," *ijritcc*, vol. 2, no. 1, p. 96–100, 2014.

[34] N. K. Kain, "Understanding of multilayer perceptron (mlp)." Available at: `https://medium.com/@AI_with_Kain/`. (Accessed: 30- Jun- 2020).

[35] A. Graves, "Sequence transduction with recurrent neural networks," in *Neural and Evolutionary Computing (cs.NE)*, (arXiv:1211.3711v1 [cs.NE]), 2012.

[36] A. Graves, "Generating sequences with recurrent neural networks," in *Neural and Evolutionary Computing (cs.NE)*, (arXiv:1308.0850 [cs.NE]), 2013.

[37] M. J. Sutskever, I. and E. H. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, (Omnipress2600 Anderson StMadisonWIUnited States), p. 1017–1024, 2011.

[38] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems 24*, pp. 2348–2356, Curran Associates, Inc., 2011.

[39] W. Bao, J. Yue, and Y. A. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," vol. 12, pp. 1–24, 2017.

[40] H. Palangi, R. Ward, and L. Deng, "Distributed compressive sensing: A deep learning approach," *IEEE Transactions on Signal Processing*, vol. 64, pp. 4504–4518, Sept 2016.

[41] S. Hochreiter, "Long short-term memory," vol. 9, no. 8, pp. 1735–1780, 1997.

[42] A. F. Gers, J. Schmidhuber, and A. F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, pp. 2451–2471, 2000.

[43] C. Ola, "Understanding lstm networks," 2015. Available at: `http://colah.github.io/posts/2015-08-Understanding-LSTMs.` (accessed: 01.06.2020).

[44] S. Varsamopoulos, K. Bertels, and G. C. Almudever, "Designing neural network based decoders for surface codes," *arXiv: Quantum Physics*, 2018.

[45] L. Wu, C. Kong, X. Hao, and W. Chen, "A short-term load forecasting method based on gru-cnn hybrid neural network model," *Mathmatical Problems in Engineering - Hindawi*, vol. 1, pp. 1–10, 2020.

[46] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 4th ed., 2016.

[47] Weka, azul, and zulu, *Version 3.8.4*. Hamailton, New Zealand: The University of Waikato, 2019.

[48] Z. Qin, A. T. Wang, C. Zhang, and S. Zhang, "Cost-sensitive classification with k-nearest neighborss," pp. 112–131, 2013.

[49] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient knn classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1774–1785, 2018.

[50] P. Mathur and V. S. Chouhan, "Implementation of k-nearest neighbor (knn) algorithm for detection of qrs complexes," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 8, pp. 77–79, 2018.

[51] R. Goyal, P. Chandra, and Y. Singh, "Suitability of knn regression in the development of interaction based software fault prediction models," *IERI Procedia*, no. 6, pp. 15–21, 2014.

[52] T. Mitchell, *Machine Learning*. McGraw-Hill, New York, 1997.

[53] L. J. Cao and F. E. H. Tay, "Support vector machine with adaptive parameters in financial time series forecasting," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1506–1518, 2003.

[54] C. Hamzaçebi, "Improving artificial neural networks' performance in seasonal time series forecasting," *Information Sciences*, vol. 178, no. 23, pp. 4550 – 4559, 2008.

[55] G. P. Zhang, "A neural network ensemble method with jittered training data for time series forecasting," *Information Sciences*, vol. 177, no. 23, pp. 5329 – 5346, 2007.

[56] H. Park, "Forecasting three-month treasury bills using arima and garch models," *Econ 930, Department of Economics, Kansas State University*, 1999.

[57] R. Lombardo and J. Flaherty, "Modelling private new housing starts in australia," *Pacific-Rim Real Estate Society Conference, University of Technology Sydney (UTS), January*, pp. 24 – 27, 2000.

[58] G. Zhang, "Time series forecasting using a hybrid arima and neural network mode," *Neurocomputing*, vol. 50, pp. 159 – 175, 2003.

[59] E. Almeshaiei and H. Soltan, "A methodology for electric power load forecasting," *Alexandria Engineering Journal*, vol. 50, pp. 137 – 144, 2011.

[60] M. Y. Khamaira, A. S. Krzma, and A. M. Alnass, "Long term peak load forecasting for the libyan network," in *in Conference for Engineering Sciences and Technology (CEST)*, pp. 185–193, AIJR Publisher, 2018.

[61] U. B. Filik, N. Gerek, and M. Kurban, "A novel modeling approach for hourly forecasting of long-term electric energy demand," *Energy Conversion and Management*, vol. 52, pp. 199 – 211, 2011.

[62] R. Gordillo-Orquera, L. M. Lopez-Ramos, S. Muñoz-Romero, P. Iglesias-Casarrubios, D. Arcos-Avilés, A. G. Marques, and J. L. Rojo-Álvarez, "Analyzing and forecasting electrical load consumption in healthcare buildings," *Energies*, vol. 11, no. 493, 2018.

[63] G. Nalcaci, A. Özmen, and G. W. Weber, "Long-term load forecasting: Models based on mars, ann and lr methods," *Central European Journal of Operations Research (CEJOR)*, vol. 27, pp. 1033 – 1049, 2018.

[64] N. Abu-Shikhah and F. Elkarmi, "Medium-term electric load forecasting using singular value decomposition," *Energy Conversion and Management*, vol. 36, pp. 4259 – 4271, 2011.

[65] R. Wanga, J. Wangb, and Y. Xu, "A novel combined model based on bybrid optimization algorithm for electrical load forecasting," *Applied Soft Computing Journal*, vol. 82, p. 105548, 2019.

[66] X. Zhanga, J. Wanga, and K. Zhang, "Short-term electric load forecasting based on singular spectrum analysis and support vector machine optimized by cuckoo search algorithm," *Electric Power Systems Research*, vol. 146, pp. 270–285, 2017.

[67] Y. Lin, H. Luo, D. Wang, H. Guo, and K. Zhu, "An ensemble model based on machine learning methods and data preprocessing for short-term electric load forecasting," *Energies*, vol. 10, p. 1186, 2017.

[68] E. A. Feinberg and D. Genethliou, "Load forecasting," in *Applied Mathematics for Restructured Electric Power Systems. Optimization, Control, and Computational Intelligence, Springer-Verlag New York Inc*, pp. 269–285, 2006.

[69] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time series analysis: Forecasting and control*. 4th ed. ed. (Wiley series in probability and statistics). Oxford: Wiley,, 2008.

[70] P. S. P. Cowpertwait and A. V. Metcalfe, *Introductory Time Series with R*. Springer New York, 2009.

[71] A. J. Wood, B. F. Wollenberg, and G. B. Sheblé, *Power Generation, Operation, and Control*. Hoboken, New Jersey : Wiley-Interscience, 2013.

[72] P. Ji, D. Xiong, P. Wang, and J. Chen, "A study on exponential smoothing model for load forecasting," in *presented at the Asia-Pacific Power and Energy Engineering Conference (APPEEC),Shanghai, China*, pp. 269–285, 2012.

[73] J. G. Jetcheva, M. Majidpour, and W.-P. Chen, "Neural network model ensembles for building-level electricity load forecasts," *Energy and Buildings*, vol. 84, pp. 214 – 223, 2014.

[74] M. Sarhani and A. E. Afia, "Electric load forecasting using hybrid machine learning approach incorporating feature selection," in *International Conference on Big Data Cloud and Applications, Tetuan, Morocco,*, 2015.

[75] X. Wang, K. Smith-Miles, and R. Hyndman, "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series," *Neurocomputing*, vol. 72, pp. 2581–2594, 2009.

[76] M. A. Momani, W. H. Alrousan, and A. T. Alqudah, "Short-term load forecasting based on narx and radial basis neural networks approaches for the jordanian power grid," *Jordan Journal of Electrical Engineering*, vol. 2, pp. 81–93, 2016.

[77] J. Buitrago and S. Asfour, "Short-term forecasting of electric loads using nonlinear autoregressive artificial neural networks with exogenous vector inputs," *Energies*, vol. 10, p. 40, 2017.

[78] Y. Fu, Z. Li, H. Zhang, and P. Xu, "Using support vector machine to predict next day electricity load of public buildings with sub-metering devices," *Procedia Engineering*, vol. 121, pp. 2016–1022, 2015.

[79] S. Gvaladze, "Evaluating methods for time-series forecasting; applied to energy consumption predictions for hvaler (kommune)," Master's thesis, Ostfold University College, 2015.

[80] S. Qiang and Y. Pu, "Short-term power load forecasting based on support vector machine and particle swarm optimization," *Journal of Algorithms & Computational Technology*, vol. 13, 2018.

[81] M. Cirrincione, M. Cossentino, S. Gaglio, V. Hilaire, A. Koukam, M. Pucci, L. Sabatucci, and G. Vitale, "Intelligent energy management system," in *2009 7th IEEE International Conference on Industrial Informatics*, pp. 232–237, 2009.

[82] H. Daneshi, M. Shahidehpour, and A. L. Choobbari, "Long-term load forecasting in electricity market," in *2008 IEEE International Conference on Electro/Information Technology*, pp. 395–400, 2008.

[83] D. Lee, "Low-cost and simple short-term load forecasting for energy management systems in small and middle-sized office buildings," *Energy Exploration & Exploitation*, 2020.

[84] J. Massana, C. Pous, L. Burgas, J. Melendez, and J. Colomer, "Identifying services for short-term load forecasting using data driven models in a smart city platform," *Sustainable Cities and Society*, vol. 28, pp. 108 – 117, 2017.

[85] J. LeSage, "Microgrid energy management system (ems) using optimization," 2020. Availbale at: `https://www.github.com/jonlesage/Microgrid-EMS-Optimization`. (accessed June 24, 2020).

[86] M. Q. Raza and A. Khosravi, "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 1352 – 1372, 2015.

[87] Z. Zhao, W. C. Lee, Y. Shin, and K. Song, "An optimal power scheduling method for demand response in home energy management system," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1391–1400, 2013.

[88] T. Dang-Ha, F. M. Bianchi, and R. Olsson, "Local short term electricity load forecasting: Automatic approaches," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 4267–4274, 2017.

[89] C. Voskoglou, "What is the best programming language for machine learning." Avilable at: `https://towardsdatascience.com/`. (accessed: 01.05.2020).

[90] T. pandas development team, *pandas-dev/pandas: Pandas*. Zenodo, Feb. 2020.

[91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[92] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[93] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison,

A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[94] MATLAB, *9.7.0.1190202 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2018.

[95] M. H. Albadi and E. F. El-Saadany, "Demand response in electricity markets: An overview," in *2007 IEEE Power Engineering Society General Meeting*, pp. 1–5, 2007.

[96] R. L. Haupt and S. E. Haupt, *Practical genetic algorithms*. 2nd ed. Wiley-Interscience, 2004.

[97] G. Zhang, B. E. Patuwo, and Y. H. Michael, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting*, vol. 14, no. 1, pp. 35 – 62, 1998.

[98] G. Ristanoski, W. Liu, and J. Bailey, "A time-dependent enhanced support vector machine for time series regression," p. 946–954, 2013.

[99] X. Zhang, T. Zhang, A. A. Young, and X. Li, "Applications and comparisons of four time series models in epidemiological surveillance data," *PLOS ONE*, vol. 9, pp. 1–16, 02 2014.

[100] V. Flovik, "How (not) to use machine learning for time series forecasting: Avoiding the pitfalls," March 01 2020. Available at: `https://towardsdatascience.com/`. (accessed: 01.07.2020).

[101] K. Zor, O. Timur, and A. Teke, "A state-of-the-art review of artificial intelligence techniques for short-term electric load forecasting," in *2017 6th International Youth Conference on Energy (IYCE)*, pp. 1–7, 2017.

[102] H. Saele and T. Williams, "Electric vehicles in norway and the potential for demand response," Feb 2020. Available at: `https://blog.sintef.com/sintefenergy/electric-vehicles-norway-demand-response/`. (accessed 30.06.2020).

[103] A. Borovykh, S. M. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv: Machine Learning*, 2017.

[104] T. Hatzakis, R. Rodrigues, and W. David, "Smart grids and ethic," *ORBIT Journal*, vol. 2, no. 2, p. https://doi.org/10.29297/orbit.v2i2.108, 2019.

Sarah M. Daragmeh

Time Series Forecasting Methods Applied to Electricity Consumption Data

NTNU
Norwegian University of
Science and Technology