Norwegian University
of Life Sciences

Master's Thesis 2020 **30 ECTS**

Norwegian University of Life Sciences
Faculty of Social Sciences
School of Economics and Business

A Study on Machine Learning Models in Predicting Volatile Spot Prices

A Case Study on Norway's Electricity Market

Hamdi A. Mohamed
Master of Science in Economics

Abstract

The uncertainty caused by the increased use of renewable energy sources makes it more essential to find good forecasting tools that can offset the increased risk in predicting elspot prices. Different supervised machine learning models are applied in this thesis to predict electricity prices for the different price areas in Norway using hourly data for elspot prices, energy prices and temperature collected for the period 2014-2020. The results show that some models are better suited for predicting elspot prices compared to others, with the Linear regression model, Gradient Boosting and Extra Randomised Trees regressor (ET) giving the best results out of the 11 tested models. The findings also suggest that choosing seasonal forecasting horizon together with adding more explanatory variables such as system load and wind power will improve the predictive performance of the models by capturing price spikes and anticipating changes in the elspot prices that longer forecasting horizon fail to capture.

Foreword

Working on this thesis the spring of 2020 with the effects of the Corona pandemic disrupting what I assumed to be a normal semester has been to say the least, a very challenging and a rewarding exercise. Many people deserve thanks for helping me achieve my goal with this thesis. To my supervisor Olvar Bergland, who has provided much guidance and has been of immense help in the completion of this exercise. To Silje Landemark and Morten Hegna from Montel AS, for their encouragement and in providing me access to the data needed for analysis. To Sagal and Yasmin, for helping me when I was stuck with running the codes and struggled understanding Python error-messages. To my sweet mother, father, siblings, and friends who provided constant encouragement and support which has helped me pull through when things got too difficult.

All data preparation and running of models has been done in Python. Codes can be provided upon request. All the remaining errors are my own.

Oslo, June 2020
Hamdi A Mohamed

Table of Contents

Acronyms

Adapting boosting — **(AdaBoost)**

Akaike information criterion — **(AIC)**

Autoregressive integrated moving average — **(ARIMA)**

Bayesian information Criterion — **(BIC)**

Best linear unbiased estimator — **(BLUE)**

Classification and regression tree — **(CART)**

Cross validation **– (CV)**

Coefficient of determination — **(R^2)**

Extremely randomised trees regressor — **(ET)**

Generalised additive models — **(GAM)**

Generalised autoregressive conditional heteroskedasticity model — **(GARCH)**

K-nearest neighbour — **(KNN)**

Least absolute shrinkage and selection operator — **(LASSO)**

Long Short-term memory neural networks — **(LSTM)**

Market's clearing price — **(MCP)**

Market clearing volume — **(MCV)**

Mean absolute error — **(MAE)**

Mean squared error — **(MSE)**

Ordinary least squares — **(OLS)**

Out-of-bag sample – **(OOB)**

Seasonal autoregressive integrated moving average — **(SARIMA)**

Smallest sum of squared residuals — **(RSS)**

Transmission system operators — **(TSO)**

List of Figures and Tables

Figures

Tables

1.0. Introduction

More increased use of renewable energy – such as wind and solar power – has given more attention to the practice of predicting electricity prices. As a result of this, there is an increased demand for better forecasting tools that accommodate the expanded vulnerability that has emerged – namely, incertitude about the utilisation of renewable energy for electricity generation. The aim of this thesis is to compare the performance of different machine learning models to predict elspot prices in Norway and ask if such methods produce more accurate forecasts than simple forecasting tools.

As a commodity, there are certain characteristics of electricity that are interesting to study for the objective of this thesis. Firstly, electricity is a multidimensional commodity where different components that are involved in its output affects its pricing. These components include delivery-period withdrawals, quantity produced, transmission capacity, and time of withdrawal (Hope, 2000, pg. 22). A change in one of these components will inadvertently affect the others, which again, is reflected in the pricing. Secondly, it cannot be stored economically and requires immediate delivery which makes knowledge of demand and supply essential (Apergis, Gozgor, Lau, & Wang, 2019, pg. 129). Thirdly, the consumption and demand of electricity follows seasonal trends which are also reflected in the prices. This means that, in order to achieve the optimal production where the demand matches the supply, it is important that suppliers are anteriorly aware of their production capabilities, and how the demand will fluctuate in the coming hours, days and weeks. Perfection is difficult to achieve, especially when more uncertainty is introduced into the time horizon; which brings us to our third point. Electricity is characterised by uncertainty on the supply side. Over- and underproduction may occur due to variations that the prices cannot correct for, whether this be due to weather changes, falling gas prices, or bottlenecks in the transmission grid of connected areas – which make it difficult to trade electricity at the same price in all connected regions. By using *market splitting* where the market is divided into predefined geographic zones, we get different market clearing prices. This illustrates how prices are not only affected by demand and supply, but also by the constraints on the cable-lines between the regions (Glachant & Lévêque, 2009, pg. 46). Also, the electricity demanded in the short run is inelastic since consumers cannot react quickly to the changes in prices. Moreover, if those changes are caused by unforeseen disturbances, it creates an incentive

to invest in a grid system that can withstand and ensure uninterrupted power supply down to a time horizon of minutes and seconds (Hope, 2000, pg. 27). This invariably raises the costs that the end-user pays on electricity deliverance.

1.1. Why machine learning?

Predictive modelling - the process of applying an algorithm or a model on data in order to predict output is becoming much more exciting in this era of big data – where the access to large datasets gives the opportunity to combine the field of machine learning with that of statistics. Machine learning is a subset of artificial intelligence that allows the machine to learn patterns from data without being explicitly programmed. It solves the problem of prediction by uncovering generalisable patterns that we wouldn't have picked up if we were to apply a model-based prediction approach instead (Mullainathan & Spiess, 2017). The goal is to build a model that seeks to make predictions about an outcome on a dependent variable given a number of features using a set of tools developed within the fields of statistics and computer science (Athey & Imbens, 2019). Making good predictions is also valuable for policy makers, and machine learning models are shown in some studies to outperform alternative statistical methods when large data sets are available even though not all statistical properties are maintained (Athey & Imbens, 2019, pg. 2).

Provided that machine learning is a term that is utilised both broadly and narrowly depending on which field it is applied to; it befits that we provide a more formal definition before continuing:

Definition 1.1. "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." (Mitchell, 1997, pg. 2)

Machine learning has three main categories – *supervised*, *unsupervised*, and *reinforcement learning*. Supervised deals with problems that use a set of *predictors*, *covariates* or *features* (X) to predict an outcome (Y). These terms are known in the statistical literature as independent or explanatory variables. By dividing the data into training and testing set, the user provides the algorithm with data (training set) to create a model that produces the desired output, and then

evaluate the goodness of fit for the given model on an out-of-sample data, i.e. test data. In contrast, the unsupervised-learning deals with problems that only have input data with no outcome measures, and the task of the user is to discover patterns amongst the set of input measures (Müller & Guido, 2016). Lastly, reinforcement-learning is a 'self-taught learning' that follows a sequential experimentation, where a machine optimises some tasks by operating within a reward system that rewards the machine if it guesses right or punishes if it guesses wrong (Varian, 2019, pg. 400-401). Different machine learning models are suited for different problems and for this thesis we choose supervised learning to evaluate the performance of our chosen models for the prediction task.

The use of machine learning may give an advantage when predicting electricity prices, since we may estimate a model that takes into account – "the volatility inherent in deregulated markets for electricity-….- to predict the occurrence of price spikes." (Mount, Ning, & Cai, 2006, pg. 63). This is achieved by adopting a model that is flexible in the sense that it –"manages to fit complex and flexible functional forms to the data without simply overfitting; it finds functions that work well out of sample "(Mullainathan & Spiess, 2017, pg. 88). The use of machine learning in the field of statistics has had a slow uptake compared to other fields, the reason for this being that "-economics journals emphasize the use of methods with formal properties of a type that many of the machine learning methods do not naturally deliver."(Athey & Imbens, 2019, pg. 2). Some academic statisticians even view the predictive methodology that the machine learning is based on as unacademic; and oppose the notion of prediction being one of the main purposes of statistics if the formal properties are not fulfilled (Shmueli, 2011, pg. 3). This is a point of discussion we will not cover here, but suffice it to say, that while statistics has four focus areas: prediction, summarisation, estimation, and hypothesis testing, machine learning is just concerned with prediction (Varian, 2014, pg. 5).

1.2. Aim of thesis

The aim of this thesis is to predict elspot prices in the Norwegian electricity market using supervised machine-learning models. We do this in two ways. The first is to compare the forecasting ability of the chosen models to see what fits our data best. The second is to evaluate the performance of the three best performing models on the other price areas in Norway. The

forecasting horizon we have chosen is medium-term, meaning, that we predict months ahead using historical price data and exogenous variables (Weron, Ziel, Soytaş, & Sarı, 2020).

The paper is organised as follows. The next section introduces us to the Nord Pool market design coupled with a brief description of the price patterns for the years 2014-2020. In chapter 3, we first present literature review on the modelling of electricity prices, before giving a detailed presentation on the machine models which we assume to be unknown to the reader. The forecasting strategy is presented in chapter 4, where we describe the implementation of the models and the methodological challenges that we have faced. This is followed by chapter 5, where we present the results followed by discussions. Chapter 6 present the conclusion together with suggestion for further work.

2.0. When supply meets demand: The Nordic market design

2.1. Nord Pool power exchange

The creation of Nord Pool comes as a result of market liberalisation for the electricity sector where the aim is to – obtain a well-functioning and competitive market that allows for efficiency gains (Hope, 2000). In the *day-ahead* market price is achieved by having customers submit bids and offers for each price area before 12:00 CET for delivery of electricity the next day. The price for each delivery hour is based on the aggregate demand and supply. This is commonly referred to as the *elspot price*. The elspot price variation for the different bidding areas within the same country is often due to the geographical positioning that makes it difficult to fully integrate with other markets, with integration level sometimes reaching only 20 or 30 percent of the time (Glachant & Lévêque, 2009, pg. 77). Let us now assume that there are no transmission constraints in the regions, i.e. no congestion due to bottlenecks and the flow of electricity between bidding areas are within the limits set by the transmission system operators (TSO). Following the logic of supply and demand, the market's clearing price (MCP) announced by Nord Pool would be the same in Norway, Sweden, Denmark and Finland (Nord Pool, 2020b). This is termed as the *system price* – where price is achieved at the intersection of MCP and the estimated demand given by the market clearing volume (MCV) (Weron, 2007, pg. 77). [1]

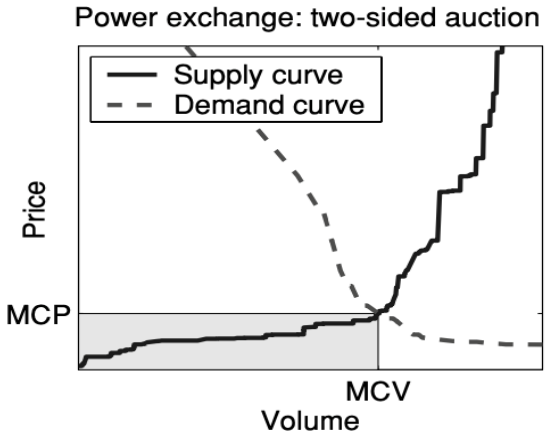


Figure 2.1. Power exchange price formation. Figure adopted from (Weron, 2007, pg. 4)

[1] There are also other forms of bidding such as the block bids and flexible hourly bidding, but we will not cover them here.

By using a *uniform-price* auction, the MCP is given to both buyers and suppliers for their bidding areas even if they bid above or below the clearing price (Weron, 2007, pg. 6). When the information is published on how the electricity is bought and sold, the respective TSOs are notified to ensure that the schedule is feasible and within the transmission constraint. From the time the information is published to when electricity is physically delivered the next day, the TSO operate the *balancing market* – i.e. real-time market where they correct for price deviations due to changes of the demand and supply for the delivery hours. This enables the TSOs to call in extra production capacity and correct the deviations to ensure physical delivery and keep the system in balance. (Weron, 2007, pg. 7). [2]

If the transmission grid capacity for the different bidding areas are congested, then price equalisation won't be achieved, which means that the clearing price for each of these areas will deviate from the system price calculated by Nord Pool. In such instances *zonal pricing* is employed, which calculates the sum of generation marginal- and congestion costs for the specific region (Weron, 2007). Below is an example that illustrates the difference between system price (grey line) and zonal price (green line) for a randomly chosen price area in Nord Pool.

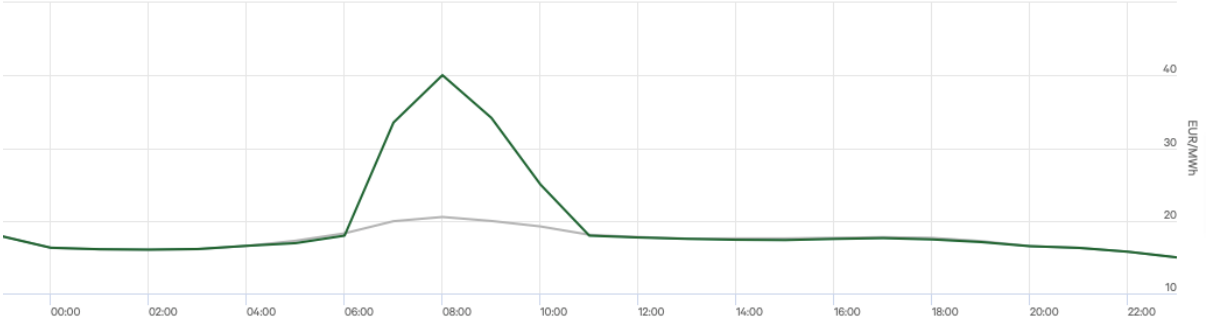


Figure 2.2. Hourly day-ahead prices for 05.02.20 (SYS) stands for Nord Pool's system price and SE4 is zone 4 in Sweden. We see a sudden spike in price for SE4 between 06:00-11:00 due to congestion (Nord Pool, 2020a).

The Norwegian power market is divided into five price areas. NO1 is southeast Norway, NO2 is southwest Norway, NO3 is central Norway, NO4 is northern Norway, and NO5 is western Norway (Statnett, 2020).

[2] Such markets have extreme price fluctuations which creates a need for sophisticated forecasting tools compared to the day-ahead market (Weron et al., 2020).

2.2. Patterns in electricity consumption

Electricity consumptions follows a seasonal pattern. Since the spot-price reflects the supply and demand at a given hour, the price will also follow the same seasonal pattern – with higher prices in periods with high demand and lower prices in others. Understanding these patterns are important with regards to the type of models we want to use when predicting elspot prices. An overview of the spot price development from our data for price area NO1 is given in figure 2.3 (a). When it comes to the four seasons, the highest prices, i.e. spikes are typically observed in winter. This we see in jan-2015 with the prices going over 200 Euro/MWh. When spring arrives, the prices start decreasing following the demand, and it usually picks up during late summer months and autumn before it falls during winter.

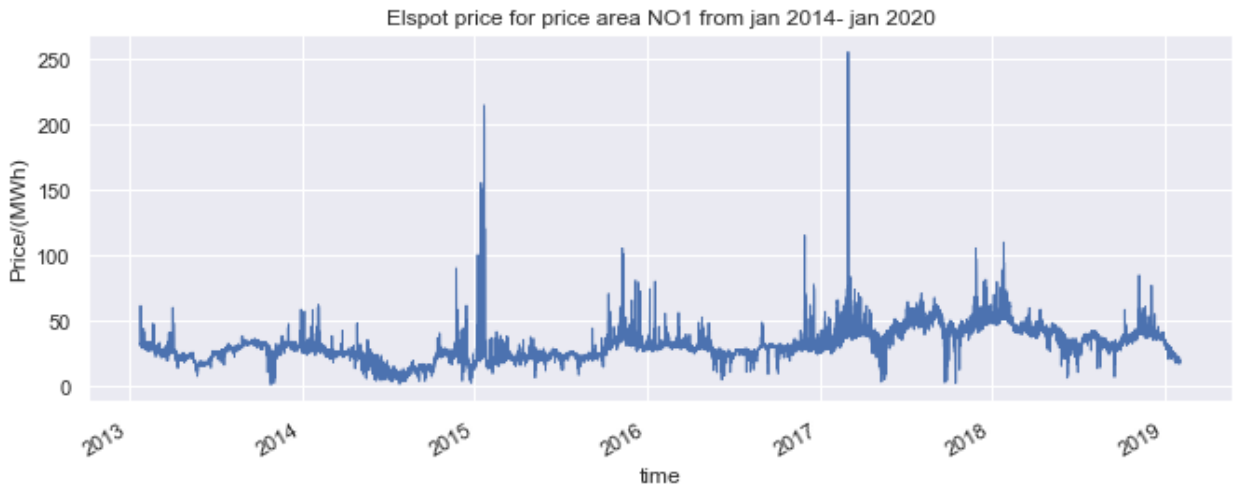


Figure 2.3 (a) Notice how the price spikes are in the winter around new year.

It is not just with seasons that we observe these variations in prices but also hours within the day and weekends.

Figure 2.3 (b) Hourly prices for January 6th, 2020. Note how the price increases in the early hours and remains sustained till evening when it goes down since people go to sleep.

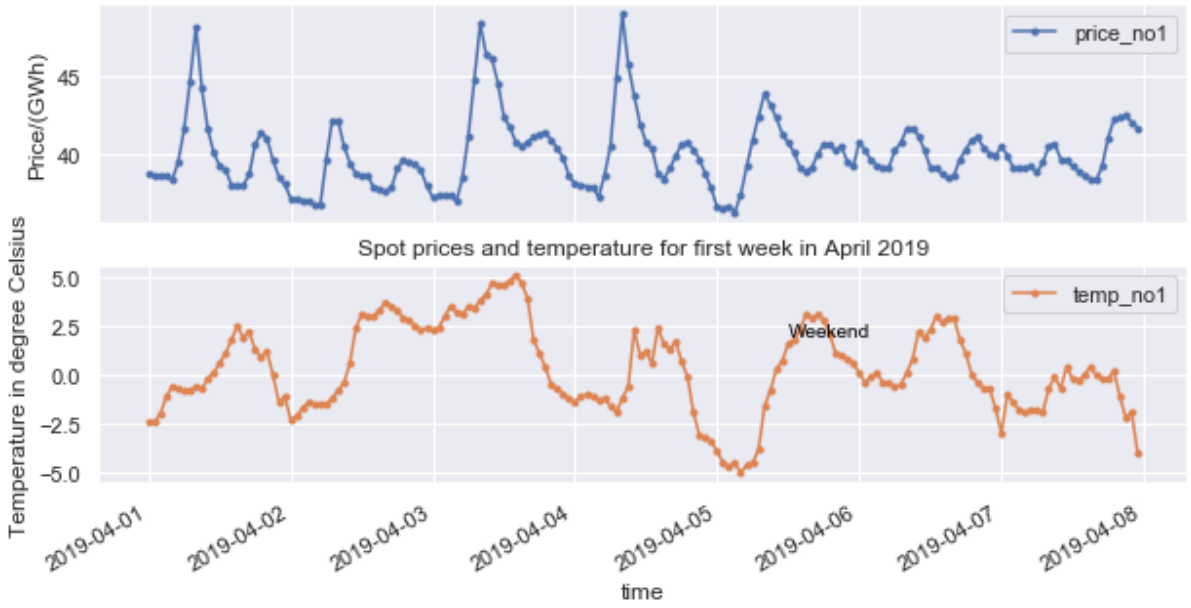


Figure 2.3 (c) First week of April. The price increases due to a fall in temperature which increases demand for electricity.

3.0. Theory

When analysing a data sample the researcher wishes to extract information on the relationship between input variables and outcome measure by a) looking at the data-generating process, or b) predicting what the response is going to be to future input variables (Breiman, 2001b, pg. 199). Since statistics and machine learning are both concerned with what one can learn from the data, they share similarities in approaches when doing predictions but also differences (Stewart, 2019). Statistical modelling is more concerned with evaluating the significance of variables and the relationship between model parameters while machine learning obtains the model through the process that we have described earlier in chapter 1, where we train the algorithm till we achieve an acceptable prediction rate before testing its performance on out-of-sample data. We are not too concerned with the interpretability and the relationship between variables since the predictive accuracy is what 'validates' our model (Breiman, 2001b). We mention this because a consequence of conflating the two matters, i.e. "— of the difference between building sound explanatory models versus creating powerful predictive models— " (Shmueli, 2011, pg. 2) — is assuming that a model with high explanatory power also has a high predictive power, which is not the case. Many researchers use now different approaches when developing prediction models based on previously learned experience by building sophisticated models that account for the unique characteristics found in electricity.

3.1. Literature and contribution on price models

Statistical models are very popular and commonly used to find trends and patterns in historical data used to predict price evolution without explicitly modelling the underlying physical processes (Gomez-Exposito et al., 2017, pg. 41) [3]. These models are helpful in evaluating and understanding the performance of the variables, but the downside when used on their own is that they struggle to capture non-linear trends, even if such models can be approximated well by

[3] Point forecast, which is the focus of this exercise is the expected value of electricity price Y_t at a given horizon h, conditional on past observed values (Weron et al., 2020). We only discuss the statistical models applied to predicting spot prices due to its relevancy. For discussion regarding other types of forecasts and models used for elspot prediction, refer to Weron (2014)'s 'Electricity price forecasting: A review of the state-of-the-art with a look into the future'.

linear ones (Weron et al., 2020, pg. 512). Misiorek, Trueck and Weron (2006) propose an ARX model (X-standing for exogenous variable) for short-term prediction on elspot prices in California by estimating a model (using OLS) that has autoregressive terms which account for dependency between lagged observations[4]. Explanatory variables that they use include the load forecasts, non-linear effects, dummy variables which capture seasonality and exogenous features (i.e. energy prices and weather). Their model has provided basis for similar studies such as Guthrie and Videbeck (2007) who use periodic autoregression model (PAR) by including a periodic component when predicting electricity prices in New Zealand using half-hour data for a period of eight years. The model captures the volatility of spot prices for both peak and off-peak periods *simultaneously*, since the shocks in the peak periods are larger and less persistent than those in off-peak periods, with the shocks often reappearing in the following peak period (pg. 5615). This spike-like behaviour is captured because they include volatility as a variable in the function. Olsson and Söder (2008) also use a similar approach in modelling spot prices by combining seasonal ARIMA (SARIMA) and nonlinear models such as the discrete homogenous Markov-process on data from Nord Pool. By combing the two approaches they find that the generated price scenarios in the model shows the desired behaviour and resembles real-time balancing of power prices in Nord Pool (Olsson & Soder, 2008, pg. 450).

Other forecasting models found within this category include models that predict elspot prices for the next 24-hours using the ARIMA methodology (Refer to Contreras, Espinola, Nogales, & Conejo, 2003) and exponential smoothing techniques where one takes the weighted average of past observations but gives higher weight to the most recent observations (Refer to Jonsson, Pinson, Nielsen, Madsen, & Nielsen, 2013). We also have the generalised autoregressive conditional heteroskedasticity model (GARCH) used in the context of volatile forecasting by being able to mirror price spikes and it builds on the assumption that the data is serially autocorrelated (Refer to Huurman, Ravazzolo, & Zhou, 2012). Threshold models are also types of models commonly used such as the Markov regime-switching model applied by Bessec, Fouquau and Meritet (2016). They use a double temporal segmentation to deal with seasonality and trading periods found in the data, and assess the forecasting ability of several classes of time-series models for wholesale spot prices at a day-ahead horizon in France (Bessec et al., 2016, pg.

[4] To make the data stationary means that the statistical properties such as variance and autocorrelation stay constant over time.

361). The use of different models to predict prices for each season makes them discover that modelling each season *independently* yields more accurate forecasts.

What all these approaches show is that when predicting spot prices using statistical models, we need sophisticated methods to achieve good results since electricity as a commodity possesses features that are not present in other markets (Hou, Liu, & Salazar, 2017). Ziel and Weron (2018) do an empirical study on short-term electricity price forecasting and consider autoregressive models on a dataset that has more than 200 explanatory variables. They find that combining advanced structures such as uni- and multivariate LASSO[5]-implied structures significantly outperform autoregressive benchmarks (Ziel & Weron, 2018). Given this, it would be interesting to evaluate the performance of the machine learning models and how they compare to the benchmark model.

3.2. Machine learning's approach to prediction

One of the main goals for machine learning models in achieving a good out-of-sample prediction is to generalise beyond the examples in the dataset that the algorithm is trained on (Domingos, 2012) . This assumes that the researcher must have knowledge about how the data is and then use the assumptions needed for generalisation when building the model. This is termed as the '*no free lunch'*- theorem by David Wolpert and it states that –" the average performance of any pair of algorithms across all possible problems is identical "(Wolpert & Macready, 1997, pg.67). This means that there is no one method which outperforms all others for all possible datasets. One approach works better on a given dataset than others, which makes this in many ways the most challenging part of performing statistical learning in practice (James et al., 2013, pg. 29).

Before we delve into the different machine learning models, it behoves that we first introduce machine learning's approach to prediction. There are two separate goals we wish to achieve with machine learning prediction. The first one is the model selection where we estimate different models to select the best performing one on the data. The second is to assess the chosen model's performance on out of-sample data (Hastie et al., 2009, pg. 222). Since our focus is on

[5] The *least absolute shrinkage and selection operator* (LASSO) is a 'penalising' regression model that reduces the coefficients of explanatory variables to zero if they don't explain the output variable in the model. Commonly used in machine learning.

supervised learning, we define *prediction* as a setting where we have a database that provides us a training sample where we have already recorded the values of input variable X_i and output variable Y_i. We run the algorithm on a training sample provided by this model,

$$T = \{X_i, Y_i\}_i^N \qquad (\mathbf{3.1})$$

where the goal for the learning [6] algorithm is to obtain a reliable pattern in predicting Y_i, so that when we provide only the values of X_i from a test sample which we hold back, the algorithm will manage to predict Y_i accurately (Friedman, 1997, pg. 55).

i. Model estimation

Let X ∈ $\mathbb{R}^n$ express a random input vector where its components are accessed by the subscripts x_t , with the predictor space showing all set of possible values for $x_t = (x_{1t}, x_{2t}, x_{3t}, \ldots\ldots, x_{nt,})$ and a joint distribution conditional on Pr(X,Y). We assume for simplicity's sake that the discussion is restricted to point forecasts which are constructed as approximations to the conditional expectation of Y_t given x_t (White, 2006, pg. 461). Let us also assume x_t to be observed prior to the realisation of Y_t, with the subscript *t* used as an observation index for the time a prediction is to be made (White, 2006). The optimal point forecast given the predictors x_t and output Y_t at a given time *t* is the provided by (3.2),

$$Y_t = f(x_t) + \varepsilon_t \qquad \mathbf{(3.2)}$$

(Hastie et al., 2009, pg. 18)

where $f(x_t)$ is the 'true' function that provides the output value, and ε_t is the random error which shares the same conditional joint distribution with the mean E($\varepsilon|x$)=0. Our target function becomes,

$$f(x_i) = E(y_i|\varepsilon_i) \qquad (\mathbf{3.3})$$

[6] Learning is a term that can be used broadly, but in this instance it refers to how the algorithm improves from experience (i.e. the data it is trained on) in a straightforward, measurable way (Mitchell, 1997).

We estimate it using the training data till we achieve an acceptable prediction rate before testing its performance on unseen data. This means that,

$$\hat{f}(x_i|T) = \hat{E}(y_i|x_i, T) \qquad (\mathbf{3.4})$$

(Friedman, 1997, pg. 58)

ii. Model assessment and selection

Hyperparameter optimisation is the tuning of the algorithms' parameters[7], where the aim is to optimise the performance of a machine learning algorithm before finalising the model and test its performance on out-of-sample data (Brownlee, 2016). Different strategies are applied to reach this goal and we will later use examples from each machine learning model used in this thesis. To evaluate the performance of the model after training the algorithm, we must find a way to obtain good prediction and quantify the extent to which the predicted value is as close to the true value for a given observation, i.e. obtain a *loss function*. One of the most common ways of measuring this is the mean squared error (MSE) ,

$$MSE = \frac{1}{n}\sum\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \qquad (\mathbf{3.5})$$

where we sum up the residual from the estimated function and then divide by the number of samples that we have (James et al., 2013, pg. 29). Another performance metric used is the mean absolute error that measures the average extent of error,

$$MAE = \frac{1}{n}\sum\sum_{i=1}^{n}(y_i - \hat{f}(x_i)) \qquad (\mathbf{3.6})$$

If we train the model on a set of observations $[(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots . (x_n, y_n)]$, and through this training we gain *experience*[8] used in estimating a function $\hat{f}(x_t)$ that helps us obtain $\hat{y}$-values

[7] It is important to differentiate between the coefficients found by the machine learning algorithm which is referred to as parameters, and the parameter of the algorithm itself being called hyperparameters. The latter are adjusted to increase model performance (Brownlee, 2016).

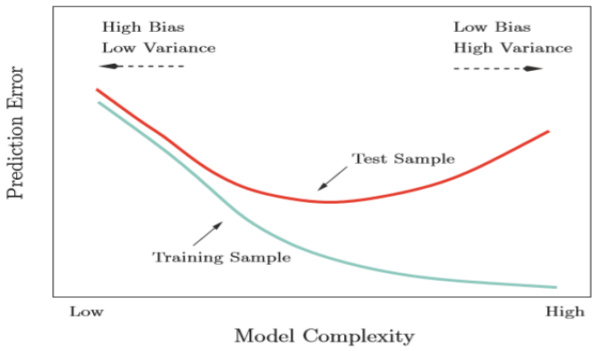
[8] Refers to the definition provided by (Mitchell, 1997) in chapter 1.

close to the true value of y, then the training MSE given by (3.5) will be small. The opposite is also the case, but this is not what we are interested in. We are more concerned with knowing whether the estimated model is generalisable and applicable to unseen data that shares the same characteristics as the training set that we saw earlier.

This leads to a central concept in machine learning called *overfitting*; which refers to a worsening of generalisation performance for the model on unseen data, because the estimated model picks up the 'white noise' in the training set as a pattern for predicting spot prices (Müller & Guido, 2016). To build a successful model we must avoid overfitting and choose a method that gives us the lowest testing MSE by computing the average squared prediction error for test observations $(x_0 y_0)$ from a large number of data points (James et al., 2013, pg. 30). This yields us the following,

$$MSE_{test} = Ave\left(y_0 - \hat{f}(x_0)\right)^2 \qquad (\mathbf{3.7}) \quad [9]$$

Another thing to consider for all supervised models is the trade-off between the flexibility of the model and how well it performs on out-of-sample data. If we are only concerned with prediction and we also have large dataset with many predictors, we may prefer a model that is more flexible since it is better at capturing non-linear relationship trends in data. Below is a figure that shows how the U-shape shown in test MSE curve is a result of two competing properties of statistical learning methods, i.e. the *bias-variance trade-off* (James et al., 2013)



[9] We square this to remove any negative sign and give more weight to large differences, because if the parameters/features in the model are meaningful in explaining the output, we do not want to the features to cancel out each other.

Figure 3.1. Training and testing error shown as function of the model complexity. The more flexible the model is, i.e. the more we fit the model to the data, the more the test MSE increases after a certain point due to increased bias of the model even though for the training MSE the prediction error decreases till it reaches zero. When bias is high and variance is high, the algorithm fails to catch important relationship between the features that explain outcome variable, so that the model is 'underfitted'. If the bias is low and the variance is high, the model 'overfits'. Figure adopted from (Hastie et al., 2009, pg. 38)

The function that captures the expected test MSE for a given value x_0 is obtained by taking the error average over repeatedly realised testing samples of the same size N from the data (Friedman, 1997, pg. 59). This gives us the three ingredients found in the bias-variance trade off,

$$\overbrace{E_\tau(f(x_0) - \hat{y}_0))^2}^{Expected\ test\ MSE} = Var(\hat{f}(x_0)|T) + [bias\ (\hat{f}(x_0))]^2 + Var(\varepsilon). \qquad (\mathbf{3.8})$$

The first term in the equation captures the variance of the estimated function $\hat{f}(x_0|T)$ which reflects the sensitivity in the training sample, so that less sensitivity makes the estimated model more stable against changes caused by resampling of data (Friedman, 1997, pg. 60). The squared bias of $\hat{f}(x_0)$, measures the sensitivity to the target function by taking the average of correct predictions the function makes for our target variable $\widehat{Y}_t$. The last term shows the variance of the error term $Var(\varepsilon)$ and it is the *irreducible* prediction error which is independent from the previous two terms and it cannot be reduced unlike the previous terms[10]. The first two terms are squared to remove any negative sign guaranteeing that the test MSE never goes below the $Var(\varepsilon)$ (Friedman, 1997, pg. 60).

iii. Improving model performance

Provided that the training MSE is less than the test MSE when model complexity increases (refer to figure 3.2), different analytical tools are used to help identify and improve the model's goodness of fit. We have the *Akaike information criterion* (AIC) and the *Bayesian information Criterion* (BIC) which are used to evaluate and choose the best performing model where we adjust the training error for the model size by adding a penalising term (James et al., 2013). This

[10] The mathematical proof of the bias-variance decomposition is shown below. τ stands for the training data. (Hastie et al., 2009, pg. 24)

$$\begin{aligned} Expected\ MSE_{test}(x_0) &= E_\tau[f(x_0) - \hat{y}_0)]^2 \\ &= E_\tau[\hat{y}_0 - E_\tau(\hat{y}_0)]^2 + [E_\tau(\hat{y}_0) - f(x_0]^2) \\ &= Var_\tau(\hat{y}_0)^2 + bias^2(\hat{y}_0) \end{aligned}$$

is provided by the models below, where $\hat{L}$ is the maximum likelihood estimate (measure of goodness-of-fit), with K being the number of parameters included in the models and N the sample size.

$$\boldsymbol{AIC} = 2K - 2\log(\hat{L}) \qquad \boldsymbol{BIC} = K\log(N) - 2\log(\hat{L}) \qquad (\mathbf{3.9})$$

(Li & Nyholt, 2001, pg. 273)

The intuition is as follows: The first term increases when model complexity increases which leads to a higher penalty-term, and the second term decreases as the model gets better in explaining the data. We want to minimise both AIC and BIC, and if the complexity of the model does not increase with the sample size (N) then BIC is preferred. Otherwise we stick with the AIC (Li & Nyholt, 2001: Burnham and Anderson, 1998). Another advantage in using AIC is that the model is derived from obtaining good predictions rather than accurately inferring the 'true distribution' of data, which is in line with the machine learning approach. (Shmueli, 2011, pg. 13).

Other methods used to achieve the same objective are the resampling techniques such as the cross-validation methods (James et al., 2013). The standard way of applying cross-validation in machine learning is to shuffle the observations randomly without considering the relationship between datapoints, and this presents a challenge for timeseries data which we will be handling in this thesis. For this reason, only the cross validation specific to timeseries data is described here.

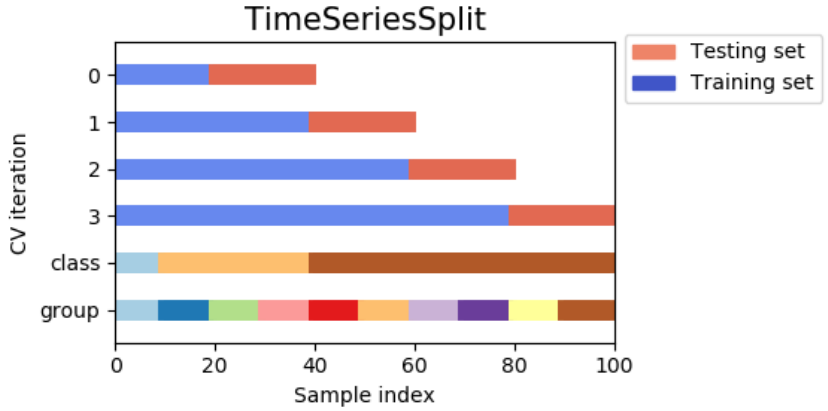


Figure 3.2. Cross-validation with timeseries data. We split the data like a 'moving window' so that all testing periods are tried on a model trained to the most recent/relevant data. Figure adopted from (Scikit-learn: Pedregosa et al., 2011).

The cross-validation method applied to timeseries data takes the training data and divides it into training and validation sets, and it does this in sequence due to the observations not being independent. A set of n-observations that are not overlapping are split into training and testing set in an equally spaced time interval, where in each split the test indices must be higher than before. The test error (given by the validation set) is provided by the average of all splits (James et al., 2013). We will now describe the theory behind the different machine learning models used in this exercise. How it is implemented comes in the next chapter.

3.3. Machine learning models

3.3.1. Parametric and non-parametric regression models

Parametric models are commonly used in prediction tasks, and they make strict assumptions on the functional form of data when choosing and fitting a model (White, 2006). These strict assumptions are an advantage both in application because they are simple, and the results are interpretable. If we assume that the functional form of $f(x)$ or parameters are linear then this gives rise to the well-known linear regression model,

$$Y = X\beta + \varepsilon \qquad (\mathbf{3.10})$$

where we seek to estimate our coefficients in the following equation,

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k \qquad (\mathbf{3.11})$$

$$\varepsilon \sim N(\theta, \sigma^2)$$

where the regression parameter is given by $\beta = \left(\beta_{1,}\beta_{2,\ldots\ldots,}\beta_n\right)^t$. The parameters help us obtain the predicted change in Y given the changes in X**,** while the error term ε captures the part of the outcome that is not explained by the predictors. Note that for the assumptions to hold, how the σ^2 is known and the distribution of error is independent from θ. We estimate this using one of the most common methods, namely the ordinary least squares (OLS). We seek to fit the model to the data by minimising the sum of squared residuals and obtaining the best linear unbiased estimator (*BLUE*) (Wooldridge, 2020, ch. 3).

$$\beta^{OLS} = argmin\left\{\sum_{i}^{N}(y_i - \beta_0 - \sum_{j}^{p} X_{ij}\,\beta_j)^{\,2}\right. \qquad (\mathbf{3.12})$$

We minimise the function with respect to the parameters and reproduce it by means of a vector-form where our input matrix **X** is transposed and has p rather than $p + 1$ columns. This gives us,

$$X^T\,(y - \mathbf{X}\beta) = 0 \qquad (\mathbf{3.13})$$

To achieve the optimal value of β^{OLS} we take the inverse of $(\mathrm{X}^T\boldsymbol{X})$ so that,

$$\beta^{OLS} = (\mathbf{X}^T\boldsymbol{X})^{-1}\mathbf{X}^T y \qquad (\mathbf{3.14})$$

i. Penalised regression models (Ridge, Lasso and Elastic Net)

It is not always feasible to use the OLS method, especially if one is dealing with many predictors in a linear model where having correlated variables results in poor estimation of the coefficients of β^{OLS} (Hastie et al., 2009, pg. 63). Several methods are proposed to tackle this problem by introducing a constraint parameter which restricts the size of the estimated coefficients, such as the Ridge regression model. The term used for these techniques is called *regularisation*.

$$\beta^{Ridge} = argmin\left\{\sum_{i}^{N}(y_i - \beta_0 - \sum_{j}^{p} X_{ij}\,\beta_j)^{\,2} \quad s.t \quad \underbrace{\sum_{j}^{p}\beta_j^2 \le c^2}_{regularisation\ parameter}\right. \qquad (\mathbf{3.15})$$

We follow the same steps as the OLS and we minimise the function using the Lagrange multiplier,

$$f(\beta_0, \beta_j, \lambda) = (y_i - \beta_0 - \textstyle\sum_j^p X_{ij}\,\beta_j) + \overbrace{\boldsymbol{\lambda_2}(\boldsymbol{\beta_0} + \boldsymbol{\beta_j} - \boldsymbol{c^2})}^{Penalising\ term\ for\ Ridge} \qquad \boldsymbol{\lambda_2} > \mathbf{0} \qquad (\mathbf{3.16})$$

which then provides us after derivation and a rearranging of terms using (3.13-6),

$$\beta^{Ridge} = (\mathbf{X}^{\mathrm{T}}\mathbf{X} + \boldsymbol{\lambda_2}\mathbf{I})^{-1}\mathbf{X}^T y \qquad (\mathbf{3.17})$$

Except for $\boldsymbol{\lambda_2}$ and **I** which is an identity matrix, the equation is the same as β^{OLS}. Mathematically we know that a singular matrix is invertible, which is also why the Ridge regression helps us obtain coefficients by adding a positive constant to the diagonal of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$, making it non-singular before inverting (Hastie et al., 2009, pg. 64). The key difference between the β^{OLS} and the β^{Ridge} is due to the behaviour of $\boldsymbol{\lambda_2}$ which captures the level of complexity in the model. The larger the value of $\boldsymbol{\lambda_2}$, the greater the amount of shrinkage faced by the estimated coefficients which goes towards zero without actually reaching it (Hastie et al., 2009). We choose a value of $\boldsymbol{\lambda_2}$ that gives us the lowest test error without sacrificing the predictors ability to help explain the model. Ridge regression works best when *all* features in the model are useful in explaining the output. This becomes a problem when we have features that are not useful, since the model fails to produce a parsimonious model that reduces model complexity while sustaining the same explanatory power (Zou & Hastie, 2005, pg. 301). The Lasso regression model solves this issue by combining both variable selection and continuous shrinkage when penalising the model for complexity. $\boldsymbol{\lambda_1}$ is the regularisation parameter for Lasso and it shrinks the coefficients that do not explain the model to zero giving us a sparse representation that makes interpreting easier. The steps are very similar to the Ridge regression, which is why we only show the results for the Lasso estimator,

$$\beta^{Lasso} = argmin\left\{\sum_{i}^{N}(y_i - \beta_0 - \sum_{j}^{p} X_{ij}\,\beta_j)^{\,2} \quad \boldsymbol{s.t} \quad \sum_{j}^{p}\beta_j^2 \leq c^2\right.$$

$$\beta^{Lasso} = argmin\left\{\frac{1}{2}\sum_{i}^{N}(y_i - \beta_0 - \sum_{j}^{p} X_{ij}\,\beta_j)^{\,2} + \boldsymbol{\lambda_1}\sum_{j}^{p}|\beta_j|\right\} \qquad \boldsymbol{\lambda_1 > 0} \qquad (\mathbf{3.18})$$

Finally, we have the Elastic net model which is shown to outperform the Lasso when $p>n$ since the Lasso cannot select more predictors than datapoints, by combining the two approaches when penalising the model. This approach enjoys the sparsity of representation provided by the Lasso, whilst also encouraging a grouping effect where strongly correlated predictors are included or excluded *together* out of the model (Zou & Hastie, 2005, pg. 301). The estimator is provided as follows,

$$\beta^{Elastic\ Net} = argmin\left\{\sum_{i}^{N}(y - \mathbf{X}\beta)^{2} + \ \boldsymbol{\lambda_2}\,|\beta|^2 + \boldsymbol{\lambda_1}|\beta|_{\mathbf{1}}\right\} \quad \lambda_1 > 0\ , \quad \lambda_2 > 0 \quad (\mathbf{3.19})$$

We use cross-validation to test the different $\boldsymbol{\lambda}$'s in order to find which one gives us the lowest test error. Below is a 2-D illustration that portrays what we have discussed so far geometrically. We subject our level curves (which shows the value of our estimated function) , to the constraint c^2 applied by each model to achieve the objective function.[11]

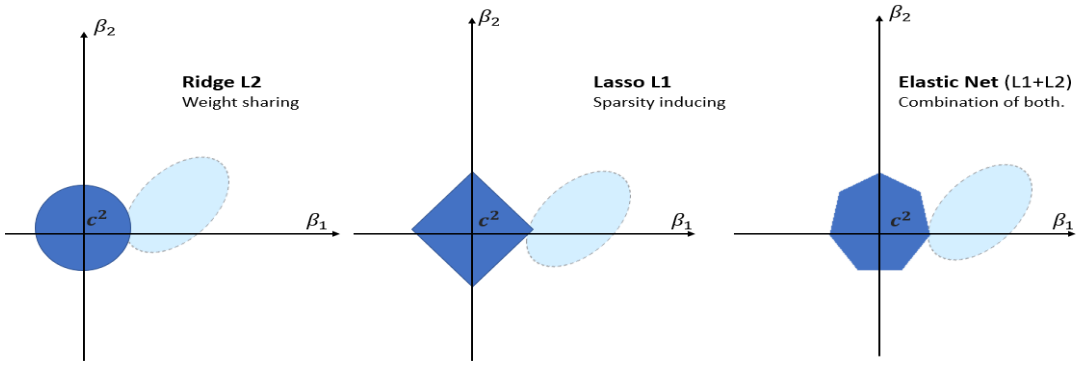


Figure 3.3. Geometry of all three models. Penalised regression models shown as a constrained optimisation problem where the dark blue shows the constraint value and the light blue shows the given level curve.

ii. K-nearest neighbour

The K-nearest neighbour (KNN) is a non-parametric regression model that does not make any rigorous assumptions on the functional form of underlying data. This flexibility is useful if dealing with non-linear data since the model adapts easily to the data. It is an algorithm that uses instance/memory-based learning where a new observation encountered in the test set is compared to other instances that were encountered from the training sample which are stored in memory. The KNN takes the average of a *k*-target of x_i (in the training set) closest to the input space x and then estimates $\hat{Y}$ (Hastie et al., 2009, pg. 14). This is built on the assumption that there is a

[11] For more visualisations and an in-depth discussion concerning this, refer to (Hastie et al., 2009, chapter 3.4)

Euclidean distance between the points x_i and x , making the Euclidean space metric. The model is defined as,

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i , \qquad (\mathbf{3.20})$$

where $N_k(x)$ shows the *k*-points of x_i in the training data closest to the input x (Hastie et al., 2009, pg. 14). We choose a *k*-number that is odd to prevent a tie so that when the algorithm assigns an observation it only assigns to one input space. The optimal k-number (a hyperparameter) chosen should help us obtain the lowest expected test error using the bias-variance trade-off when fitting the model. Using a non-linear model like the KNN poses numerous challenges relative to the linear models. Estimating the parameter is much more difficult to compute and interpret, the model is unstable so that a small change like adding data or resampling causes the estimated error to change, and the model is prone to overfitting which worsens its performance on unseen data (White, 2006, pg. 467). So, if we can fit a parametric model to the data which gives us good results, we should choose it instead.

3.3.2. Tree-based models and Boosting

i) Decision Trees

Decision tree is a tree-based algorithm that does both regression and classification tasks. The main idea behind this algorithm is to build a tree that divides the data into sub-categories within the predictor space by using *if* and *else* statements (James et al., 2013).

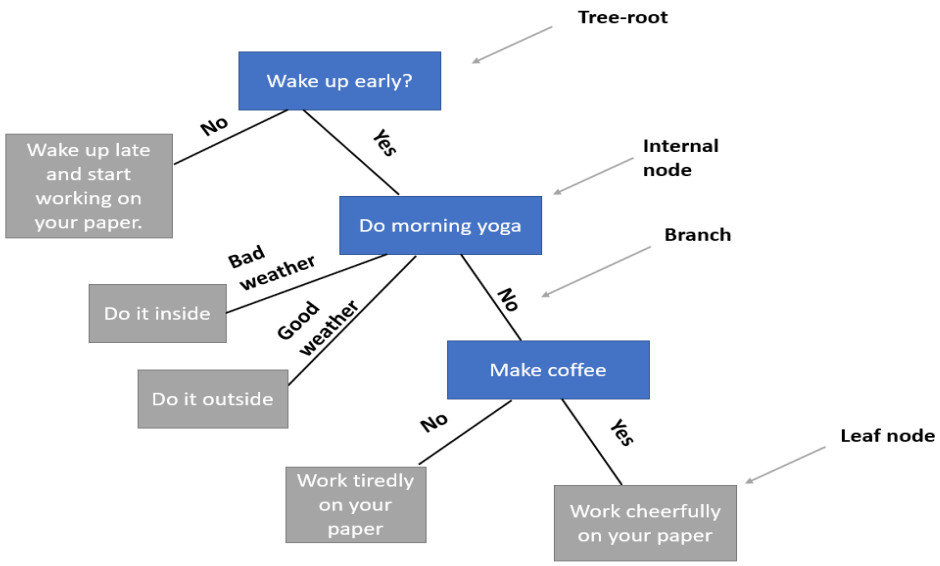


Figure 3.4. An illustration of a decision tree for classification that predicts morning routine given some features. A regression tree follows the same steps but uses numerical values instead of categories.

We restrict our discussion to regression trees due to its relevancy. The tree is built by trying different thresholds, i.e. *internal nodes* where we split the predictor space to R-regions that are not overlapping for each step to find the smallest sum of squared residuals (RSS). We then seek the value that minimises (3.21) which helps us predict the test-observation by taking the mean value for the training observation within the j-th box ($\hat{y}_{R_j}$),

$$\sum_{j=1}^{J}\sum_{i\in R_j}(y_i - \hat{y}_{R_j})^2 \qquad (\mathbf{3.21})$$

(James et al., 2013, pg. 306).

The best threshold becomes a *candidate* for the tree root. This is done for all thresholds so that when we compare the candidates, the best one with the lowest value becomes the *tree-root*. Depending on the number of splits we allow for the data points (*stopping criterion*), the tree reaches a level when it cannot split the data anymore and this becomes the *leaf node*. The leaf-nodes correspond to the average output for the different clusters of data points (James et al., 2013, pg. 305-6). Since it is time-consuming and costly to allow every possible partition of the predictor space into j-boxes, we must adopt a strategy that allows us to choose the number of

splits and try different partitions. A strategy named in the literature is the *recursive binary splitting*, that adopts a top-down approach by starting at the top of the tree and dividing the predictor space using two new branches for each level down the tree. It is also *greedy* because the best split is made by comparing different thresholds at *a given step* rather than adjusting the split by looking ahead and therefore getting better results (James et al., 2013, pg. 307). To implement this strategy, we consider all predictors in the regions $X_1, X_2, \dots, X_j$ and a number of possible values for the splitting point s , and then choose a predictor and a splitting point that provides us the lowest RSS. Using the following equation, we define two regions R_1 and R_2 as,

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\} \qquad (\mathbf{3.22})$$

And based on the minimising function (3.21) that we described earlier to obtain the best values for s and j, we get

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \qquad (\mathbf{3.23})$$

The next step is to do the same for each region separately, and then use the mean of the training data in the region to predict the output on the test sample. A common problem with applying decision tree models on data is that they tend to overfit[12]. Several solutions are proposed to deal with this overfitting, with the most common one being the *cost-complexity pruning* method, also called post-pruning where you grow a large tree T_0 and then prune it back to select a sub-tree that gives you lowest test MSE. You use a non-negative tuning parameter α to obtain a sequence of the best sub-trees by using cross-validation (James et al., 2013, pg. 309). The α captures the bias-variance trade-off and resembles the penalty-term found in penalised regression models (James et al., 2013, pg. 309).

ii) Boosting models

Several methods of leveraging models to improve their predictive performance are proposed in literature. The Boosting models by Freund and Schapire (1996) achieve this by building a large

[12] Hastie et al. (2009, pg. 352) state that trees have '-- one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy' .

set of 'weak' base-learners in a stagewise manner to produce a 'strong' learner. This is done by generating base-learning algorithm from the training sample and building many decision trees[13] (Zhou & Yu, 2009). We assume the weak learners to have an error rate that performs slightly better than random guessing, and we boost it using a *stagewise additive* modelling by applying them to reweighted versions of the training data following a sequence given by $G_m(x) = 1,2, \dots, M$. Applying reweighted versions from training data at each step decorrelates[14] the trees where regressors focus on the more difficult regions missed by past trees (Hastie et al., 2009). This gives a final model that is the weighted average of all regressors,

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m \, G_m(x)\right) \qquad (\mathbf{3.24})$$

(Hastie et al., 2009, pg. 338)

where α_m is the coefficient and $G_m(x)$ the weighted sample from training data for each weak learner. This stagewise additive modelling employed by boosting is also found within statistics in models such as the generalised additive models (GAM) and basis expansions such as the polynomial function where we optimise by jointly fitting parameters using estimation methods such as the OLS and maximum likelihood. Boosting on the other hand does this by optimising a single tree's coefficients in a stagewise manner while holding the others fixed (Hastie et al., 2009). This slows down the overfitting by not going back and adjusting what's done in the past, by *regularising* the rate with which we overfit the data. *Adapting boosting* (AdaBoost) is such an algorithm that repeatedly fits the residuals using a reweighing scheme where at each stage one subproblem is minimised using a *likelihood-based exponential* loss function (Hastie et al., 2009, pg. 342-3). How this is accomplished is provided in the figure below.

[13] Oftentimes trees but neural networks are also used.
[14] Decorrelation refers to correlation caused by growing many trees to the same dataset.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t to minimalize the weighted error:

$$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 3.5. The algorithm builds an additive logistic regression function and uses stagewise fitting. (Schapire & Freund, 2012, pg. 5)

The *Gradient boosting* model is another boosting technique used where we extend the loss functions from exponential/binomial to generalised loss functions. It provides a more flexible framework than the AdaBoost algorithm making it a viable model for prediction. The model evaluates the gradient of the loss function estimated in step *2a* (refer to figure 3.7) using training data, and then approximate the gradient on a regression tree, usually a stump[15] which we later use to estimate and update the loss function (Hastie et al., 2009, chapter. 10.10)

[15] A small decision tree with only two terminal nodes.

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

Figure 3.6. Implementation of the Gradient Descent Algorithm (Hastie et al., 2009, pg. 360)

γ_{jm} is the optimisation vector that shows the tree stumps, I is the shrinkage factor which *slows* the stagewise model-building to obtain better results. Boosting models tend to overfit when dealing with noisy data due to the shallowness of the trees built. Different tuning parameters are used to control for this, such as the number of trees, learning rate (shrinkage factor) and the depth of tree.[16] Gradient boosting is better at dealing with this than the AdaBoost due to the added shrinkage parameter which slows the overfitting of data. Models that deal with this problem without much hyperparameter tuning are the Random Forest and the Extra Randomised Trees regressor.

iii) Random Forest (RF) and Extra Randomised Trees regressor (ET)

The Random Forest (RF) algorithm is proposed by Breiman (2001a) and it is a modified version of *bagging*[17] where we build large number of decision trees *independently* using a bootstrapped

[16] A motivation for building shallow trees, is that the tree stumps are shown to work very well on nested-sphere problems where the decision-boundary of the stumps is at the surface of the sphere. The function that describes it has a quadratic form that gives good approximation. For more discussion and visualisation on this, refer to (Hastie et al., 2009, pg. 590)

[17] We fit many small or large trees to bootstrap resample versions of the training data, and then classify by majority vote. Different from boosting where this is done stagewise.

replica. This differs from boosting which dedicates more effort to rectifying misclassification of data, whereas the RF includes additional randomisation when recursively splitting using a small subset of features at each step to ensure that the trees capture the variation in the data. This is then *averaged* which lowers the variance of the model (Breiman, 2001a). By taking small subsets at a time, we ensure that the model decorrelates the trees when building which results in vast improvement in prediction accuracy compared to a single decision tree.

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Figure 3.7. Implementation of the Random Forest Algorithm for regression and classification tasks. Note how at each tree split, a random sample of m features is drawn and only these are considered for splitting. It is common that $m = \sqrt{p}$ *or* $log_2 p$. (Hastie et al., 2009, pg. 588)

The out-of-bag sample (OOB) is another feature of the RF, and it is presented as $OOB = N - Z^*$, which is the remaining $\frac{1}{3}$ of the data that has not been used to create the bootstrapped replica. This is then tested on each tree that have been built on the bootstrapped data, and the label/regressor that gets the most votes is assigned to the OOB which gives us the OOB error estimate. Once this error stabilises when building the trees, we can terminate the training of the model and apply it on the test sample (Hastie et al., 2009, pg. 593). The hyperparameter tuning in the RF is very similar to that of the decision tree except that we increase the number of trees we want to build through the steps described earlier. As mentioned, the hyperparameter tuning deals with the overfitting of the model but according to Breiman (2001a) and Hastie et al. (2009) the RF does *not* overfit. The argument is that the randomisation which is added when splitting

introduces *small* bias into the model while simultaneously reducing the variance due to the corresponding ensemble of trees that is averaged. At one point, the increase in the number of trees averaged does not increase the bias, so that no benefit nor harm comes to the model since the test error just stabilises[18].

Extremely Randomised Trees regressor (ET) is another tree-based algorithm that extends the randomisation of the previous model by selecting thresholds at random in addition to the subset of feature selection for each step (Geurts, Ernst, & Wehenkel, 2006) . Unlike the RF, the ET uses the whole data as a learning base and tries to reduce the models' dependence on the data structure by randomising both attribute and cut-point when splitting the tree nodes, and then averaging the trees to reduce variance (Geurts et al., 2006, pg. 2-6).

Split_a_node(S)
Input: the local learning subset S corresponding to the node we want to split
Output: a split $[a < a_c]$ or nothing
– If **Stop_split**(S) is TRUE then return nothing.
– Otherwise select K attributes $\{a_1, \ldots, a_K\}$ among all non constant (in S) candidate attributes;
– Draw K splits $\{s_1, \ldots, s_K\}$, where $s_i = \textbf{Pick\_a\_random\_split}(S, a_i), \forall i = 1, \ldots, K$;
– Return a split s_* such that $\text{Score}(s_*, S) = \max_{i=1,\ldots,K} \text{Score}(s_i, S)$.

Pick_a_random_split(S,a)
Inputs: a subset S and an attribute a
Output: a split
– Let $a^S_{\max}$ and $a^S_{\min}$ denote the maximal and minimal value of a in S;
– Draw a random cut-point a_c uniformly in $[a^S_{\min}, a^S_{\max}]$;
– Return the split $[a < a_c]$.

Stop_split(S)
Input: a subset S
Output: a boolean
– If $|S| < n_{\min}$, then return TRUE;
– If all attributes are constant in S, then return TRUE;
– If the output is constant in S, then return TRUE;
– Otherwise, return FALSE.

Figure 3.8. Implementation of the Extra Randomised Trees Algorithm for numerical attributes. (Geurts et al., 2006, pg. 6)

If we assume that the optimal features which explain the results are available, then the advantage that the ET algorithm has over the RF is that it is less computationally expensive to run. Geurts et al. (2006) also do an analysis on the bias-variance trade-off by comparing results of different ensemble models, and they discover that the bias in ET increases compared to the RF since it

[18] It is important to state that this is a point of contention which we do not cover in this thesis. Some argue that the Random Forest has shown to overfit empirically. For the reader who's inclined to read more on this, Louppe (2014) dissertation on 'Understanding Random Forests- from theory to practice' is a good place to start.

randomly selects attributes without considering the feature importance. The averaging helps reduce the variance but also increases the bias which leads to overfitting. The hyperparameter tuning applied to the RF is also applied on this algorithm, and we refer to them for reminder if needed.

3.4. Benchmark model

We will be using a simple persistent model as benchmark when comparing the predictive accuracy of the different models for different price areas in Norway. Naïve forecast are the simplest forecasting models to generate, and we take the *random walk process* where we use the most recent observed value in the past $y_{t-1}$to predict the value y_t. This is provided through,

$$y_t = y_{t-1} + \varepsilon_t, t = 1,2, \ldots \qquad (\mathbf{3.25})$$

where the variance changes as t changes.

(Wooldridge, 2020, pg. 376)

4. Forecasting strategy

In this chapter a brief description of the data is given together with the considerations taken in choosing the three final models to test on the unseen data. We also present the methodological issues faced with the data pre-processing and the implementation of the methods presented in the previous chapter.

4.1. Data description

Understanding the data is essential in deciding which models helps us achieve the best results. Due to the many models we test in this exercise, we now restrict the discussion to the comparison of model performances for price-area 1 only. A comparison of the finalised models' performance in the different price areas will follow in the next chapter. Below is a description of the variables used.

Table 4.1. Variable description used for the data analysis.

Electricity Target variables: **log [price_no1, price_no2, price_no3, price_no4, price_no5]**	Hourly elspot prices determined with respect to each delivery hour for price area NO1, NO2, NO3, NO4 and NO5. Units are in Euro/MWh
Essential climate variables (temperature) **lcool = log {$cool = \text{temp\_no1} - 22$}** **lheat = log {$heat = 15.5 - \text{temp\_no1}$}**	Hourly temperature measured in °C for price area NO1 (temp_no1). We add heating and cooling degrees since temperature has an asymmetric effect.
Additional variables (dummy variables) **Hour, Day, Weeks(53 columns)**	The dummy variables are added to capture calendar-effects to improve predictive accuracy.
Energy prices **lpgas, lpoil, lpcoal, lpcarbon**	Log transformation of energy variables for gas, crude oil, coal and carbon prices.

4.2. Pre-processing and dealing with missing data

The data is for the period 24.01.2014 to 31.01.2020. Different sources are used to collect them, and they come in different file-formats which makes it challenging to combine. The elspot prices are all from Nord Pool using their historical market data archive. Historical temperature data is collected from frost.met.no, which is owned by the Norwegian Meteorological Institute. Energy prices were harder to obtain freely, but after contacting Montel they kindly provided access to the needed data for our period of interest. Energy prices are important in explaining electricity prices because of the transmission cables we have with other countries. Since the electricity in Norway is mostly sourced from hydropower, many producers seek to minimise the cost and gain profit by increasing electricity production, which is exported when elspot prices are high. Or the opposite by holding back on production where the water is kept in reservoirs when prices are low.

The data pre-processing is decidedly the most challenging and time-consuming part to do in this exercise. The first issue was dealing with the non-stationarity in data. Standard supervised models assume the data to be i.i.d, with the same distribution for training and testing set and for the distribution to be fixed over time. These assumptions are all violated with time-series data. There are two ways of dealing with this. The first is to identify the trend and seasonal components in the data and remove it using differencing techniques. The other strategy which we choose is to use the seasonal information as input features in the data such as hour, day and week(-s) so that the algorithm learns to better identify trends and use the information to increase its predictive performance (Brownlee, 2017, pg. 120). Missing data is another issue that we dealt with. We solved it in two ways. When it comes to the missing values found in the data collected from Nord Pool (no more than 7 obs.[19]), we replace it with the row-average using the .isnull() function in Python. The energy prices are much more challenging to deal with since we had 72 missing observations from the gas prices and 16 observations missing from each of the other three variables. We manually combed through the observations to find the missing values and discover that they are either randomly placed or missing as a result of market closure on weekdays due to Christmas, New year or Easter. We solve this by manually plotting in the *day-before* price for the following reason[20]: Since we only have prices for weekdays when the

[19] This is a rough guess from memory.
[20] Only exception is 25.12.14 where we plot in the value of 26th instead 24th.

markets are open, we wanted to create a python code that covered the weekend by adding two more observations for each week using the last observation (Friday). Due to using this code, it was paramount to cover the missing holes for weekdays to avoid messing with the data. We also make a code where the daily price for energy variables counted 24 times for each day. This is done to match the hourly data of temperature and elspot prices.

We also consider if rescaling our predictors could potentially better the models' performance in capturing important relationship between the features and the output, by using machine learning techniques such as standardisation which gives the attributes a mean value of 0 and a standard deviation of 1. We apply and test the performance of the scaled attributes on some of the models and discover that the performance worsened with scaled variables compared to non-scaled. The standardisation assumes the data to have a Gaussian distribution, which we will later see is not applicable to our data. Also except for the linear regression model, the other two best performing models are the tree-based models which are scale-invariant, meaning that the models are indifferent to feature scales. Other than the log-transformation of our target and predicting variables, we choose to keep them non-scaled. To briefly describe the strategy employed, we first select models to apply on the data. We then divide the data using the SKLEARN's ***train_test_split*** package where the training set are set to 75 % of data and cross-validation to 25% for *price_no1*. The plan is to fit our final models to this price area, and then use them to predict the prices in the other four price areas (*price_no2* , *price_no3*, *price_no4*, and *price_no5*). Below is a correlation matrix of the explanatory variables.

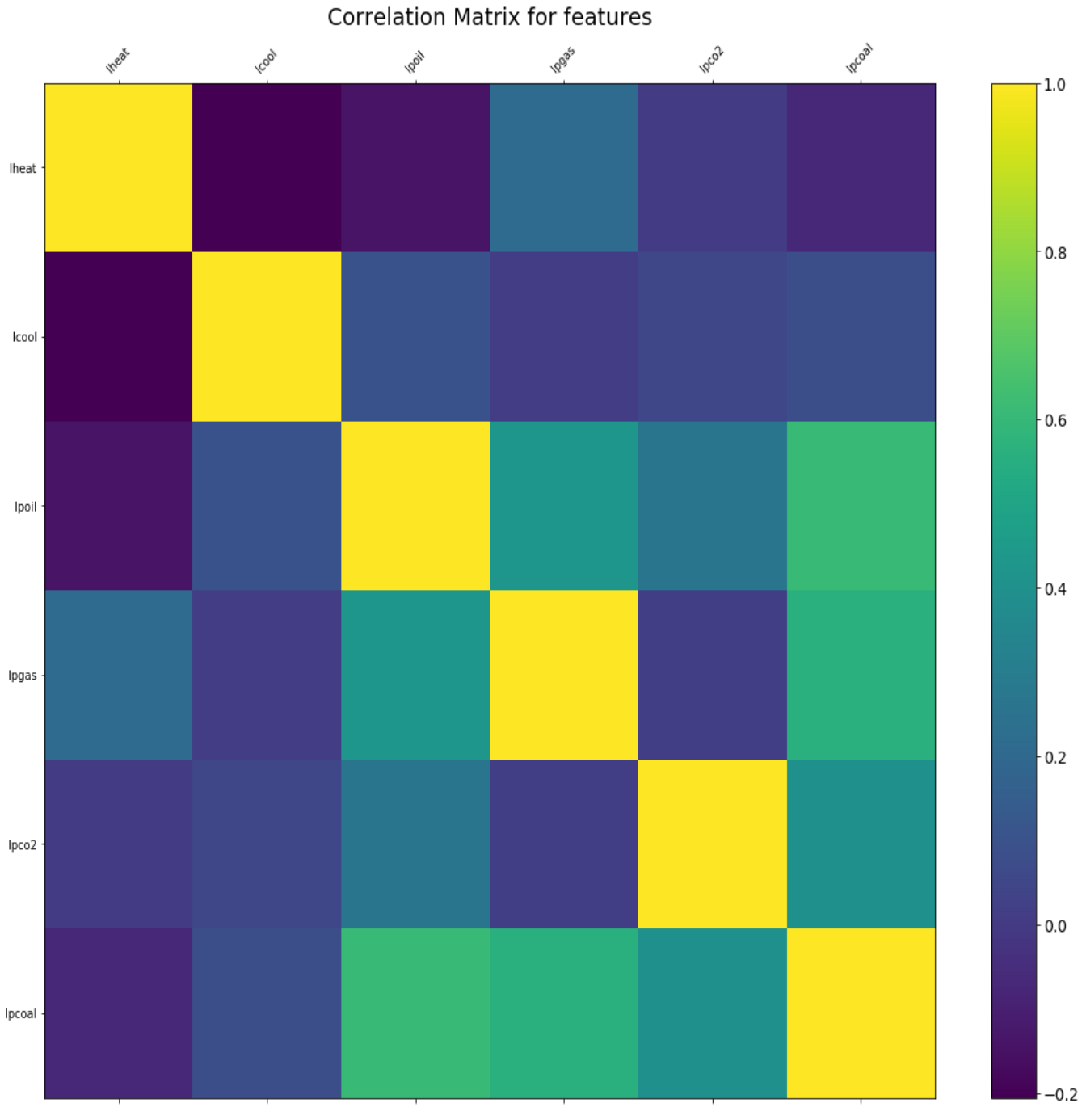


Figure 4.1. Correlation matrix of the variables (excluding dummy variables and target variables). Refer to Table 4.1. for variable description.

4.3. Methods implemented

The discussion on different evaluation metrics is something that we do not cover here but for most machine learning algorithms provided in the SKLEARN packages in Python, the mean squared error is given as the standard evaluation metric and we choose to use it out of convenience. For the parametric regression models and the Decision tree, we also include the R^2–score which is a regression score function also known as the coefficient of determination (Wooldridge, 2020, pg. 35). It measures how much the dependent variable y_t is explained by the input features. For practical purposes, the R^2—score is often placed between 0 and 1, with 0 showing the R^2 being no better than the mean value of the data whilst 1 shows that the model perfectly fits the data.

i. *OLS and penalised regression models*

The Ridge, LASSO and Elastic-Net methods are proposed in the previous chapter, and they are used to assess the models' performance on the training data. Below is a graph demonstrating how the increase of alpha as a penalty term for Ridge and Lasso reduces the MSE until a certain limit is reached.

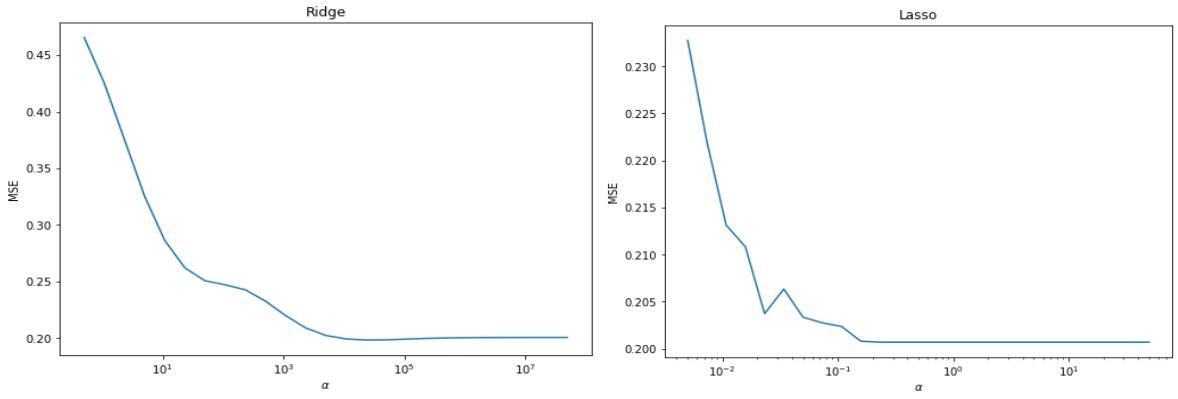


Figure 4.2. Comparison of penalised regression models' performance on training set. An increase in alpha reduces the MSE, especially in Ridge's case which is more than halved.

We then take the best alpha which gives the lowest MSE for the penalty regression and apply the models to the cross-validation set. Together with the OLS which did not need any hyperparameter tuning gives us the following result,

Table 4.2. A comparison of the results of the penalised regression models. Note how the best result is provided by the OLS model.

Models	**MSE**	**R^2**
OLS	0.07953219429515039	-0.08135429898167312
Ridge	0.21791054810594912	-1.962806572560381
Lasso	0.26223850775034496	-2.5655087883280703
Elastic-Net	0.26223849325754456	-2.5655085912776534

The introduction of penalty into the model increases the variance with the penalising models performing much worse than the OLS. This indicates that the coefficients which are either reduced or dropped are important in explaining the output, supporting the idea that penalised regression models are a bad fit for the data. It also appears that the linear models are more sensitive to outliers compared to other model-types, and we have plenty of this since electricity displays price spikes due to its volatile nature. Also note how the R^2–score has a negative sign showing that the linear models fit the data much worse than *random guessing*, i.e. the mean-value. To see if the model is worth going the extra length for by doing hyperparameter tuning, we look at the residual plot to see if there is a linear trend that we fail to capture.

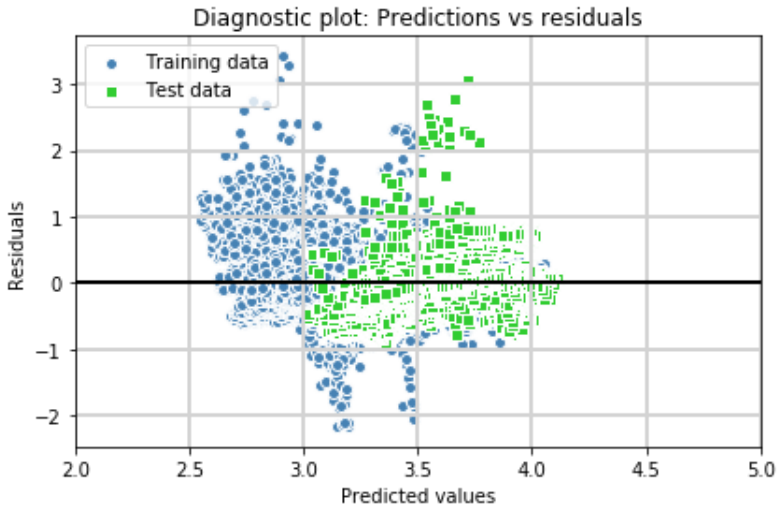


Figure 4.2. The residual of the linear regression model.

The plot of residuals does not point to any trend that our models fail to capture when using the OLS. We could increase the penalty-term to try to lower the MSE even further but there is no point. We therefore wish to test if the non-linear models may be a better fit for the data. We decide not to use these models except for the linear regression further in the analysis due to the bad performance. We do not finalise and apply them on the other price areas.

ii. K-nearest neighbour and Regression Trees

The KNN and Decision tree (CART) models are used to test the performance of the non-parametric models. These models do not outperform the penalised models with the MSE for the training data for KNN being `0.277993` and CART: `0.237619`.

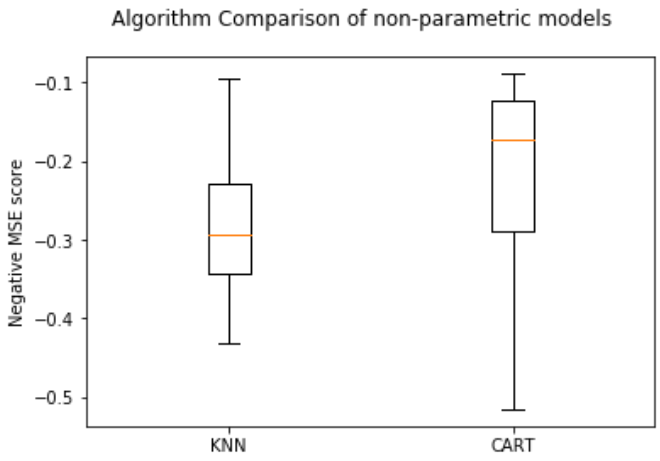


Figure 4.3. Visual comparison of algorithm performance on training data. The score shows negative MSE.

We suspected overfitting of the models due to its worse performance on the cross validation set and try to tune the hyperparameters to see if it was possible to obtain a reduced MSE for a model that could be generalised to the other price areas. For the KNN, a low K-number leads to overfitting which leads us to try different k-numbers larger than 1 to see which gives us the optimum neighbour with the lowest test error (Hastie et al., 2009, pg. 241). This is provided by the table below where the lowest test error for KNN is `0.261255`.

```
Best: -0.261255 using {'n_neighbors': 21}
-0.357956 (0.120804) with: {'n_neighbors': 1}
-0.280674 (0.122788) with: {'n_neighbors': 3}
-0.277993 (0.120970) with: {'n_neighbors': 5}
-0.278481 (0.120010) with: {'n_neighbors': 7}
-0.278033 (0.120883) with: {'n_neighbors': 9}
-0.276052 (0.122237) with: {'n_neighbors': 11}
-0.272741 (0.123110) with: {'n_neighbors': 13}
-0.268879 (0.124265) with: {'n_neighbors': 15}
-0.265562 (0.125118) with: {'n_neighbors': 17}
-0.263254 (0.125561) with: {'n_neighbors': 19}
-0.261255 (0.125250) with: {'n_neighbors': 21}
```

Table 4.3. KNN algorithm tuning where the optimum k-neighbour is 21. The score shows negative MSE.

For the Decision tree algorithm, Hastie et al.(2009) recommends *post-pruning* to achieve the best results for the Decision tree model. Instead, we do *pre-pruning* to deal with the overfitting of the model since the SKLEARN package does not include post-pruning. Different stopping criterions are used to find the optimal number of nodes and splits using a function that is built in Python. Once this is done, we iterate over different depths to examine the bias-variance trade-off.

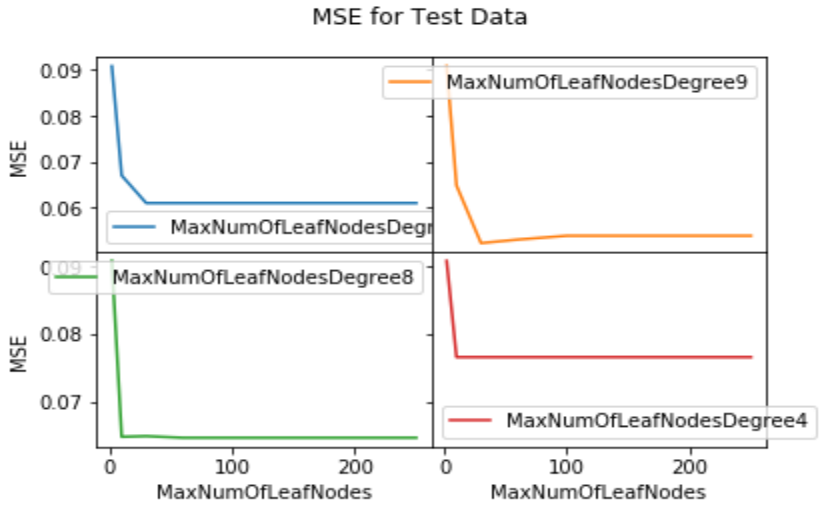


Figure 4.4. Model performance on different depths (4, 5, 8, 9) with stopping criterion (max number of leaf nodes) used.

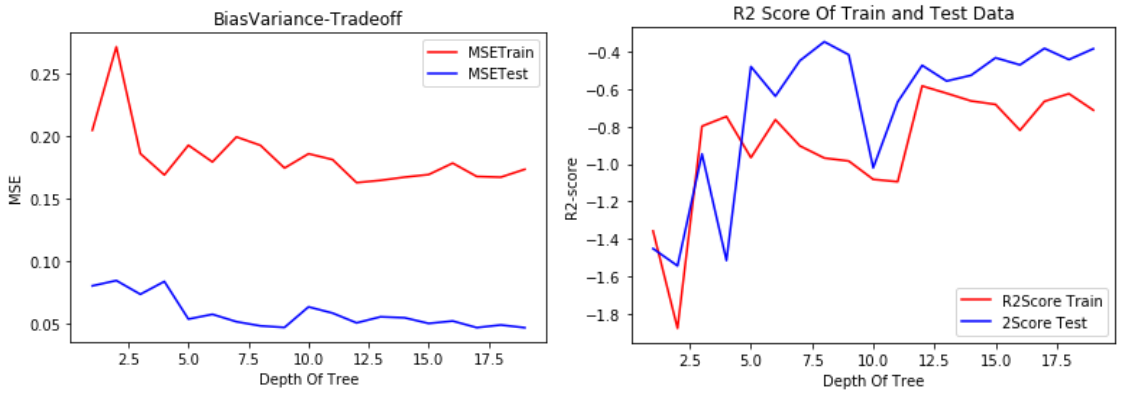


Figure 4.5. Note how the MSETrain and MSETest do not converge at any point (!) in the bias-variance trade-off. The fact that they do not follow the expected trend gives us a cause for investigation.

The performance of the Decision tree was 'suspect'. For the training data the lowest MSE was `0.05` given by the hyperparameters tree depth `5` and the stopping criterion (maximum number of leaf nodes) equal to `30.00`. We took these hyperparameters and used it for the cross-validation set which gives a test MSE of `0.156`. This clearly shows that the model overfits. Added visualisation (figure 4.6) also show how the model manages to predict the mean-price for the test-period which could result in a lower MSE than the previous models. It is still a bad fit regardless.

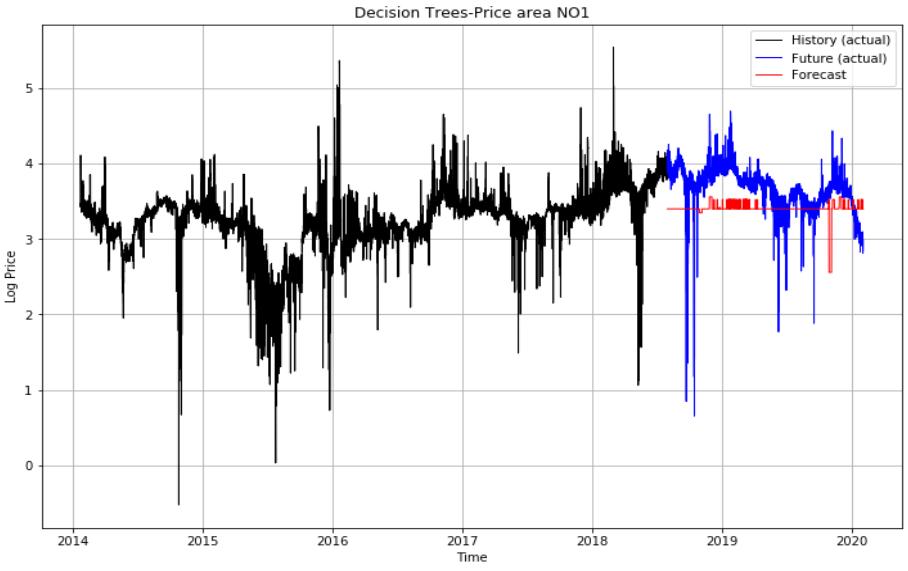


Figure 4.6. Decision Tree model for price area 1. The model appears to take the average value for elspot prices instead of capturing variations.

A solution for this other than more hyperparameter tuning is to apply ensemble methods that deal with the overfitting better due to the averaging of many models. We therefore decide not to use these two models further in the analysis due to the better performance of ensemble methods that combine several machine learning models which we will address next. They are therefore not finalised and applied on the other price areas.

iii. *Ensemble methods (Random Forest, Extra Randomised Trees, Gradient Boosting, AdaBoost)*

Of all the ensemble models tested, the AdaBoost is the worst performing one with training MSE being `0.321113`. This can be due to its bad handling of outliers in the data which makes sense since more weight is given to weak learners that predict incorrectly, leading to the model fitting the noise in the data and increasing variance. For the other three models, the training MSE are as follows: Gradient Boosting (GBM) `0.182345`, Random Forest (RF): `0.210320`, and Extra Tree Regressor (ET): `0.194901`.

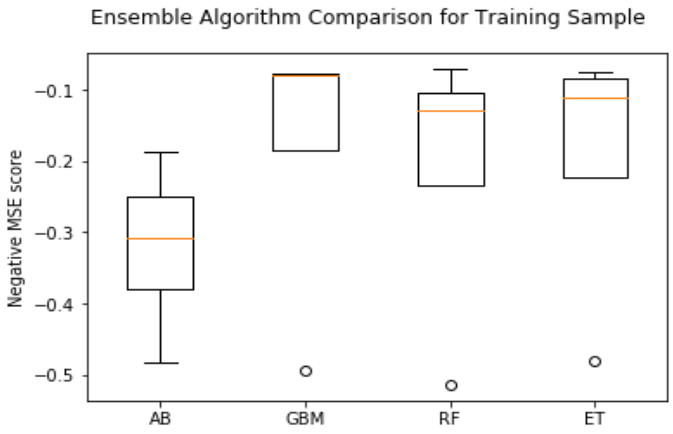


Figure 4.7. Visual comparison of algorithm performance on training data. Except for the AB, the error is tightly distributed for the models.

In finalising our three best models we find the optimal hyperparameters by performing grid search in Python, which is a machine learning technique that builds and evaluates models for different combinations of algorithm parameters specified in a grid (Brownlee, 2016, pg. 98). A grid search is computationally expensive to run but once the best hyperparameters are obtained for the models we save it and apply it later to finalise our models.

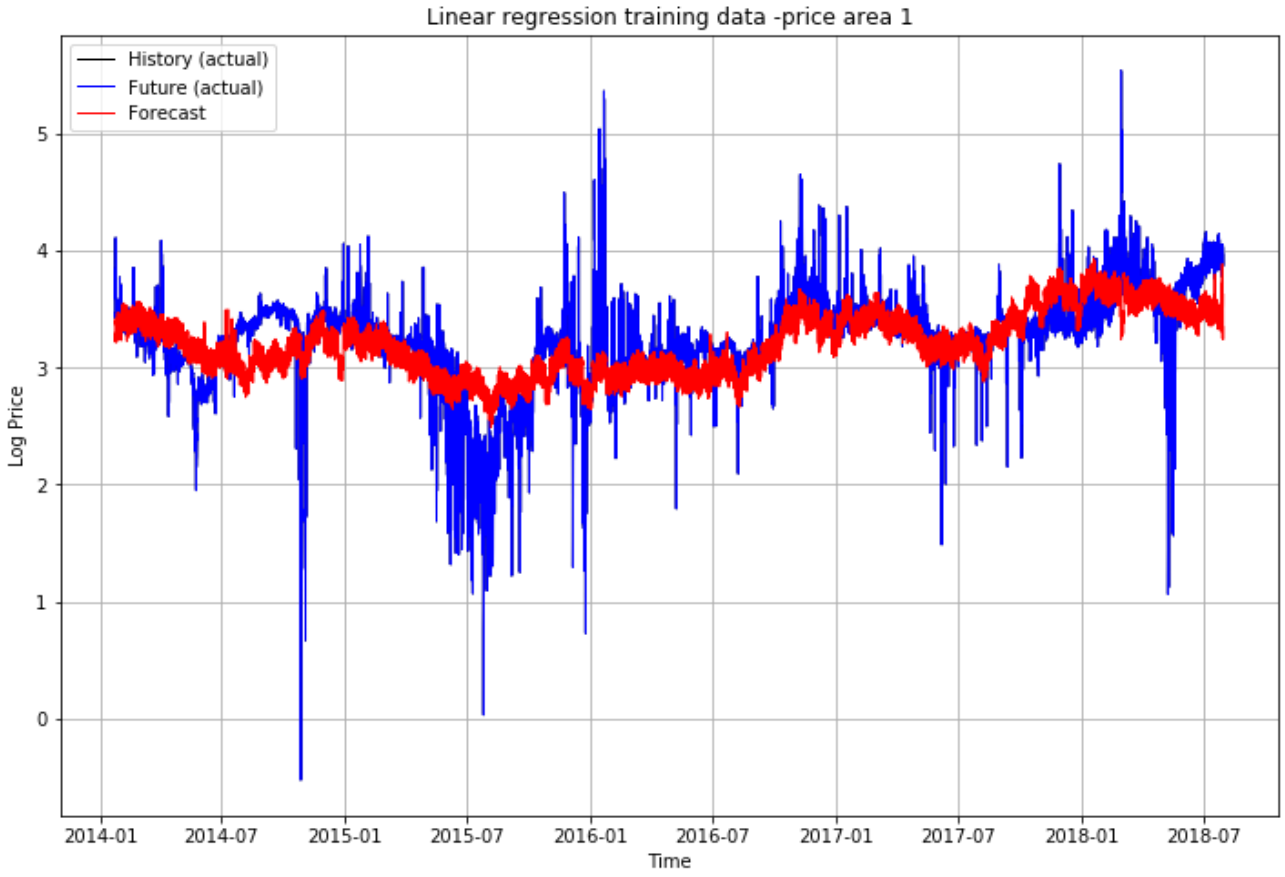
The bests three models of all the tested models are the linear regression, Gradient Boosting and Extra Randomised Trees regressor. For the linear regression, the OLS which has the lowest test MSE has no hyperparameters which needs to be tuned. The optimal trees, i.e. *n_estimators* for the Gradient Boosting was 700 trees which gave a test error of `0.161541`. The Extra Tree regressor had an optimal number of trees 250 which gave a test error `0.189834`. For the timeseries cross-validation, we split it in 4 parts and use the default setting recommended by the SKLEARN's packages for the other hyperparameters. The random state for all algorithms were set to 42 to ensure that every time the code is run, we get the same results.

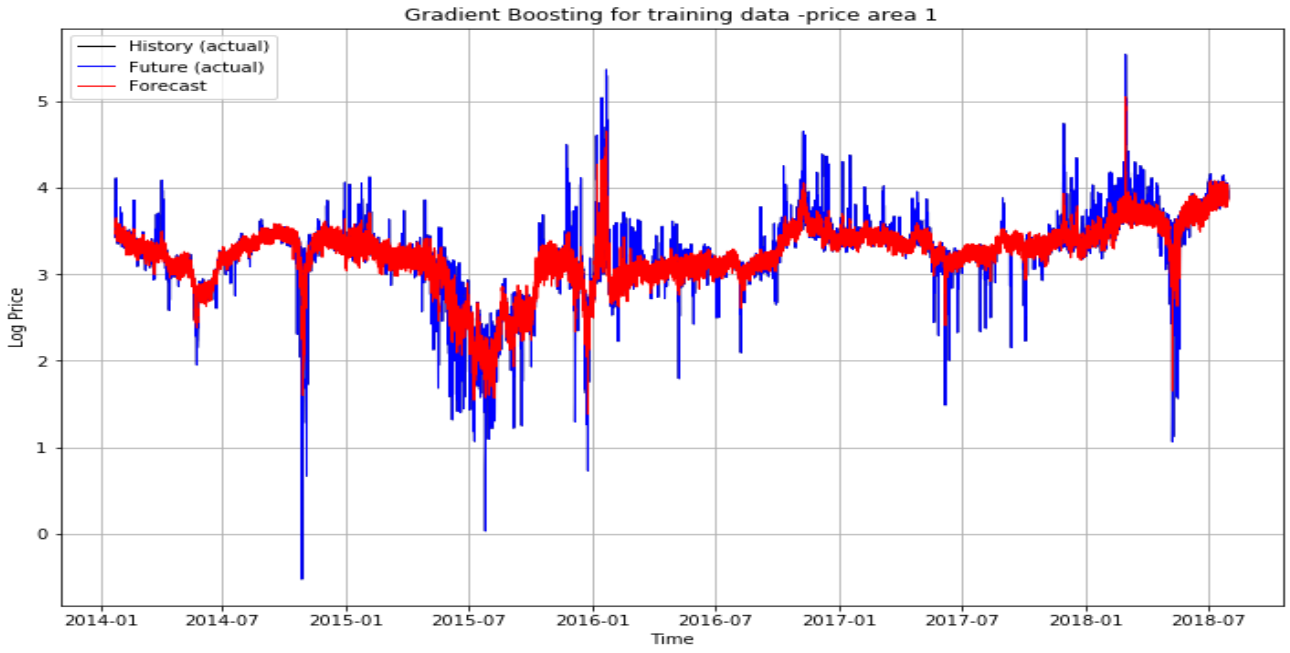
5. Results

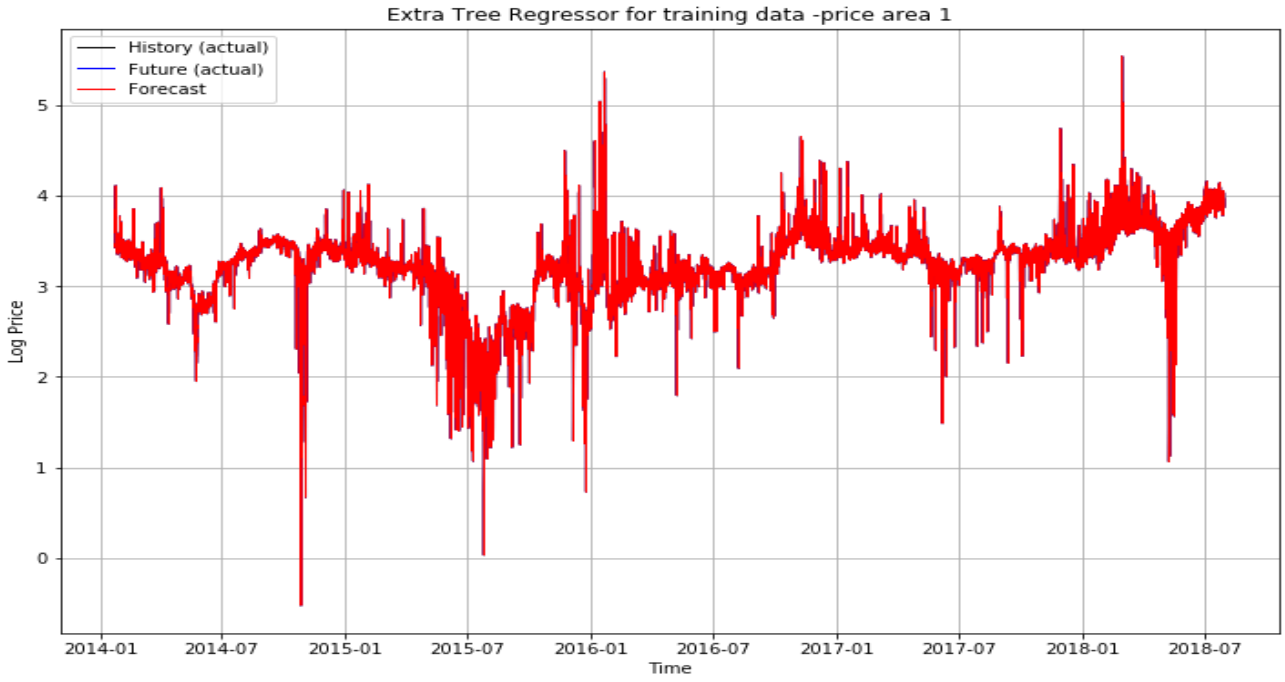
In this section we present the results of the finalised models on the other price areas in Norway. We first present the in-sample performance for our models to visually inspect how well they learn the trends and patterns on the training data before presenting their predictive performance on the out-of-sample data. This is followed by a table that compares performance using MSE as an evaluation method.

5.1. Presentation of model performance for the different price areas

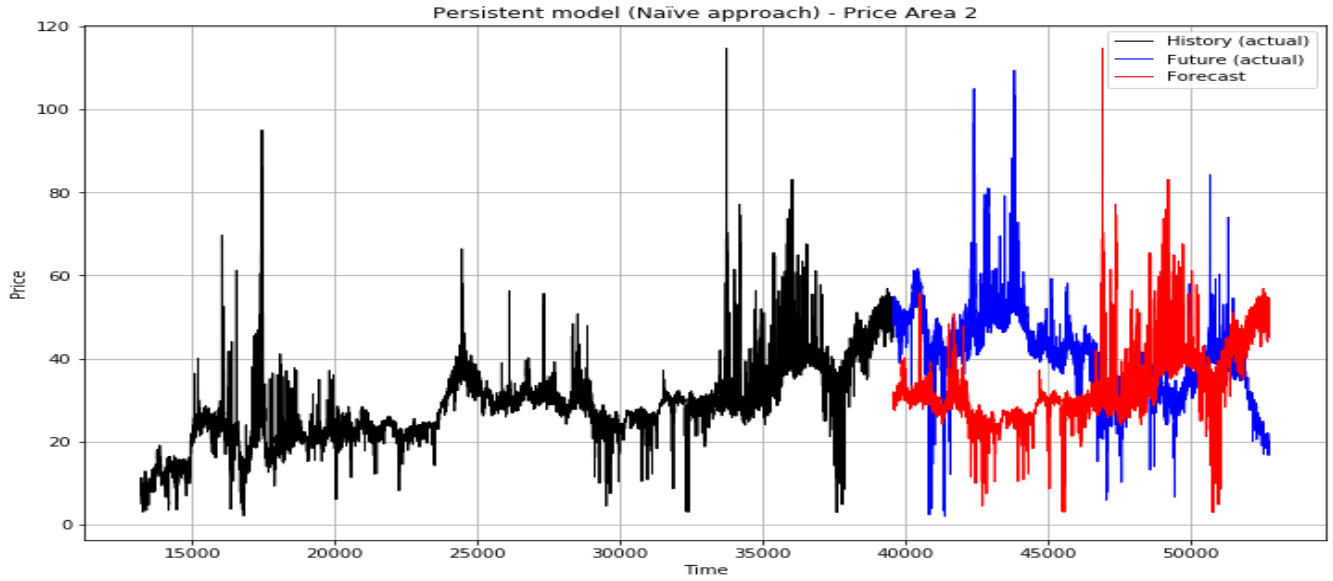
i. <u>In-sample performance</u>

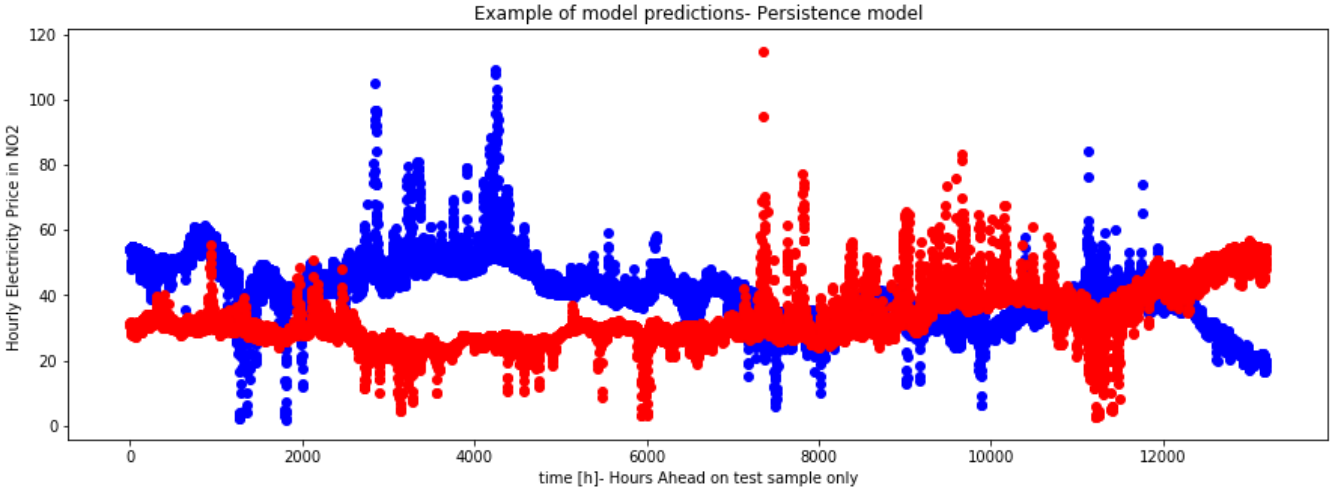


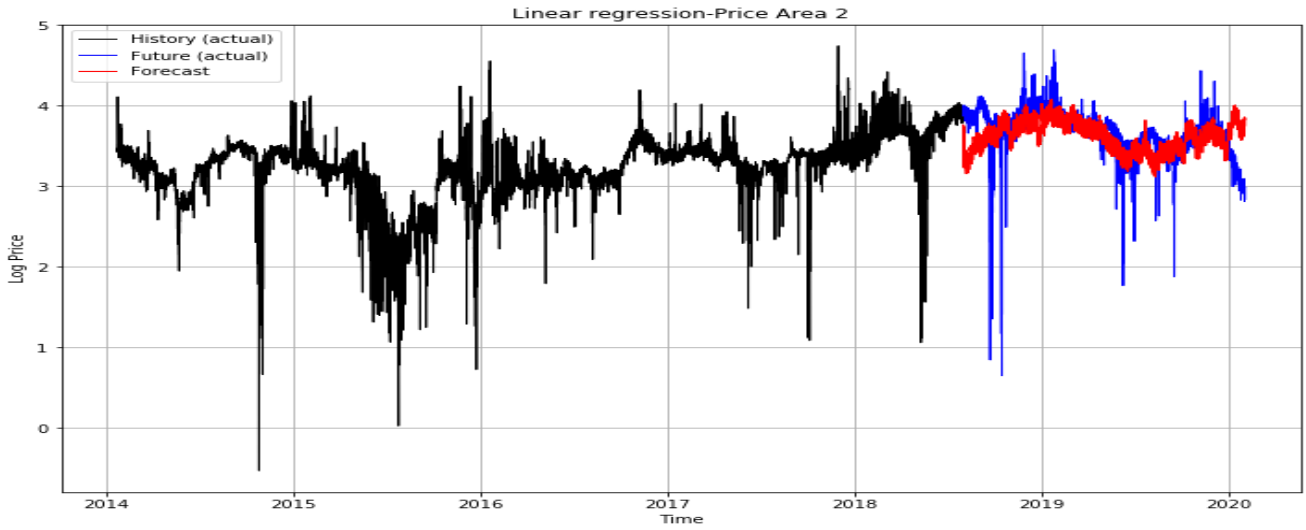

Gradient Boosting for training data -price area 1
History (actual)
Future (actual)
Forecast
Log Price
Time
0
1
2
3
4
5
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07

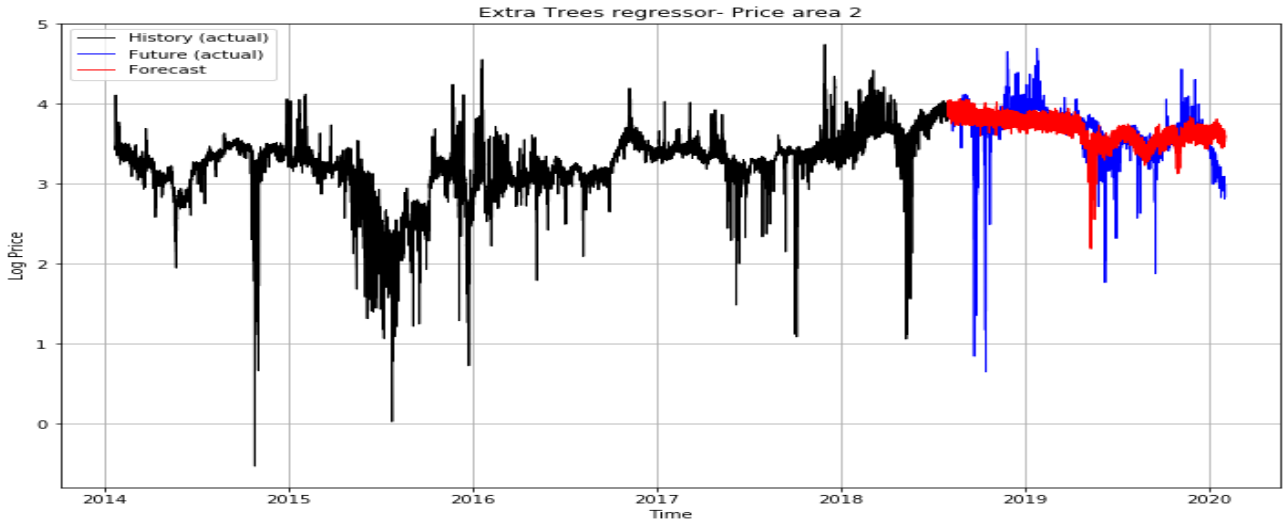

Extra Tree Regressor for training data -price area 1
History (actual)
Future (actual)
Forecast
Log Price
Time
0
1
2
3
4
5
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07

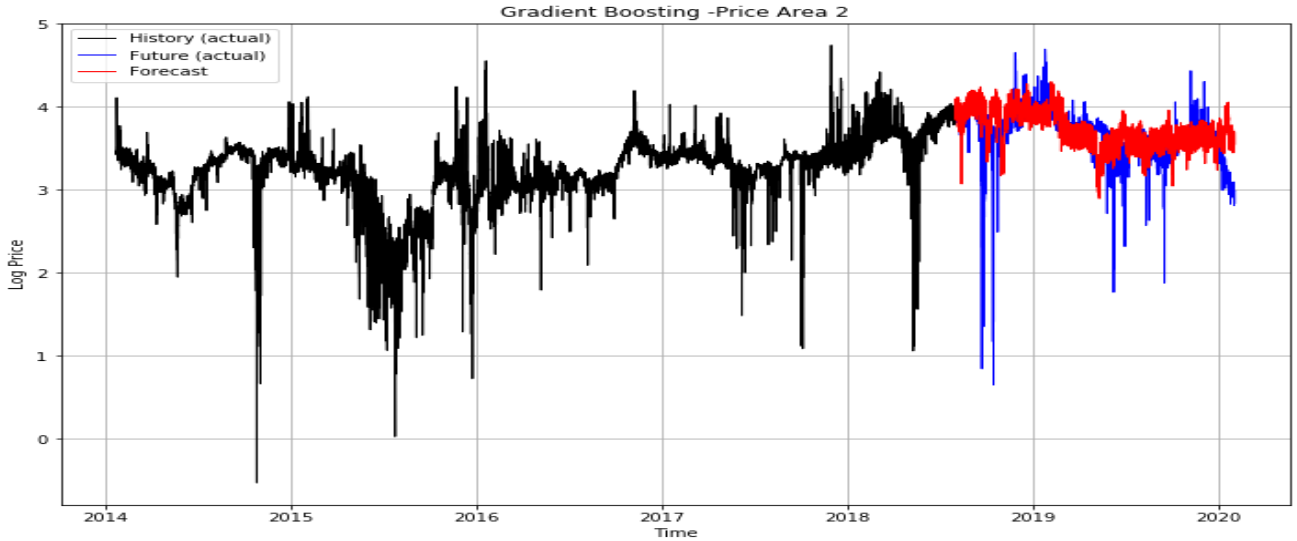
ii. *Price area 2*



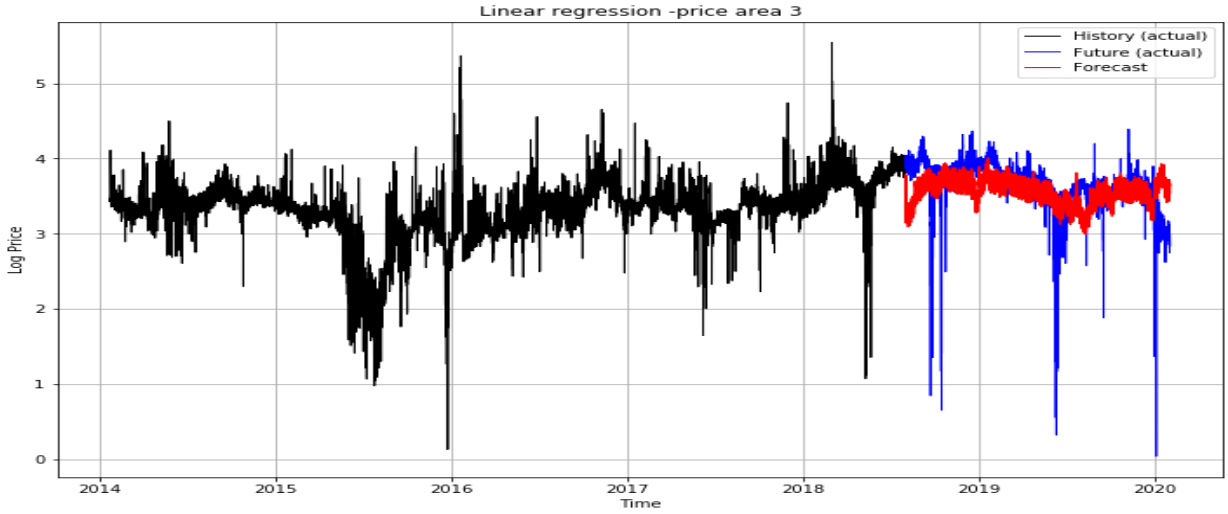


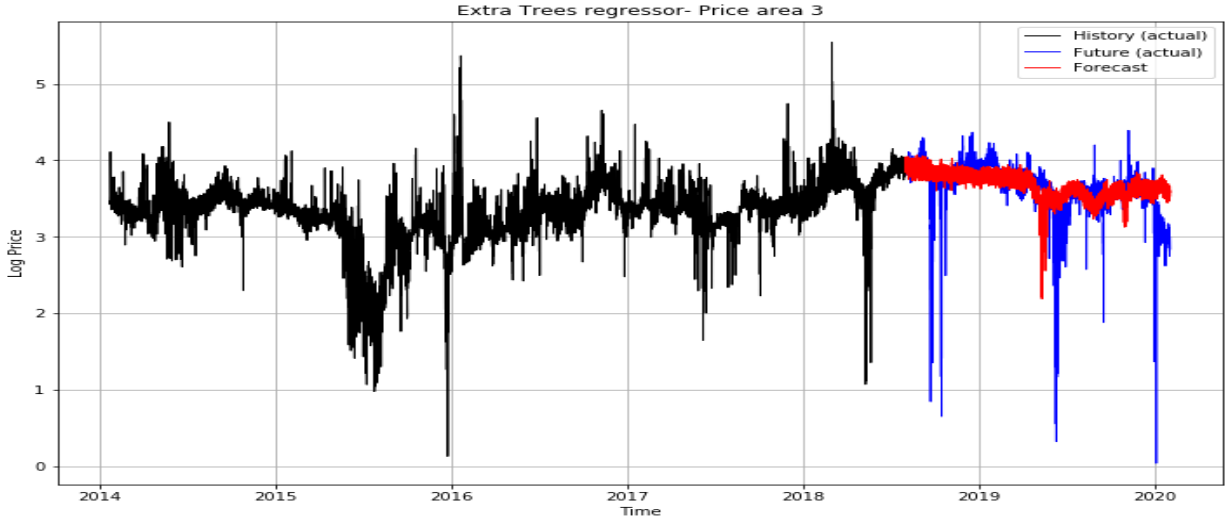

Linear regression-Price Area 2
History (actual)
Future (actual)
Forecast
Log Price
Time
0
1
2
3
4
5
2014
2015
2016
2017
2018
2019
2020

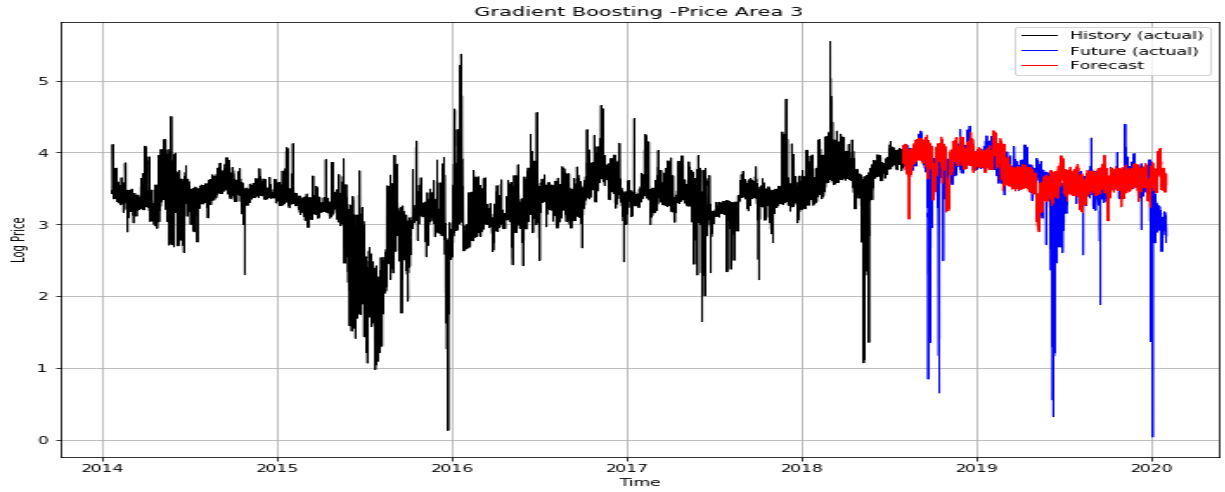

Extra Trees regressor- Price area 2
History (actual)
Future (actual)
Forecast
Log Price
Time
0
1
2
3
4
5
2014
2015
2016
2017
2018
2019
2020


Gradient Boosting -Price Area 2
History (actual)
Future (actual)
Forecast
Log Price
Time
0
1
2
3
4
5
2014
2015
2016
2017
2018
2019
2020

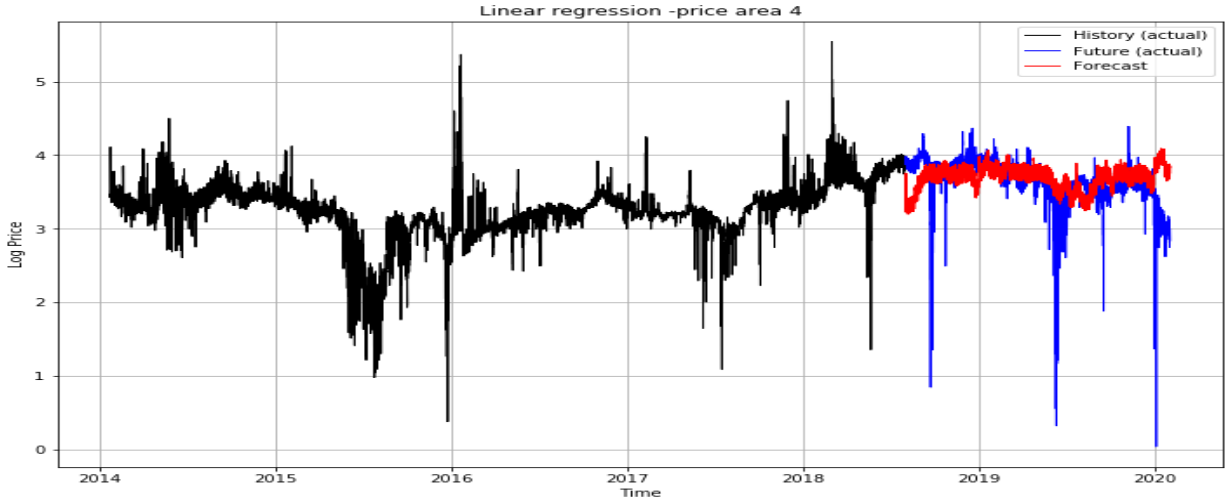
iii. *<u>Price area 3 (excluding Naïve baseline model)</u>*

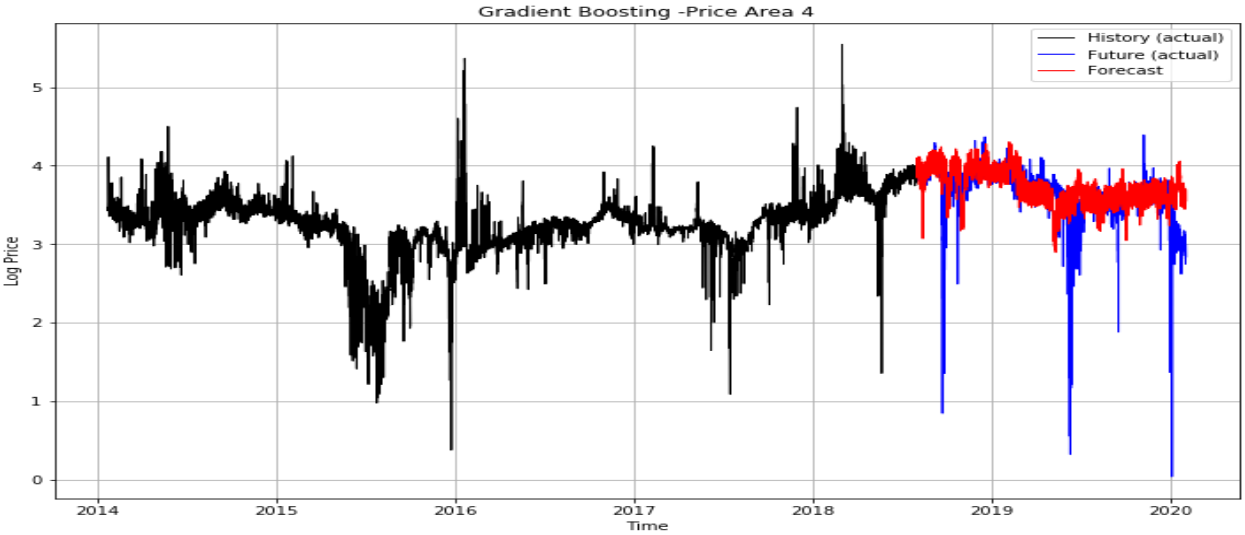




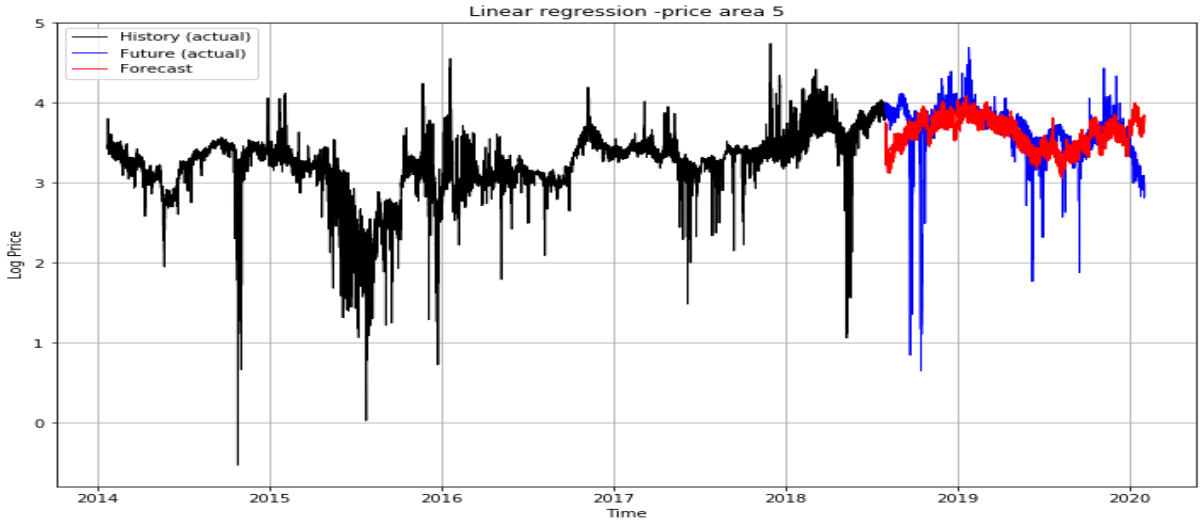


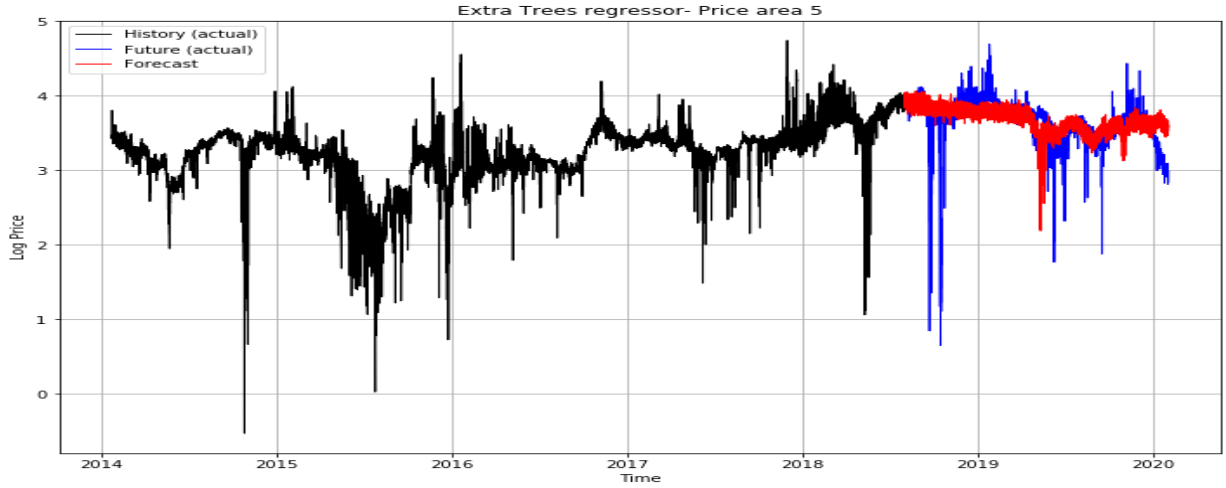
iv. *<u>Price area 4 (excluding Naïve baseline model)</u>*





v. *Price area 5 (excluding Naïve baseline model)*





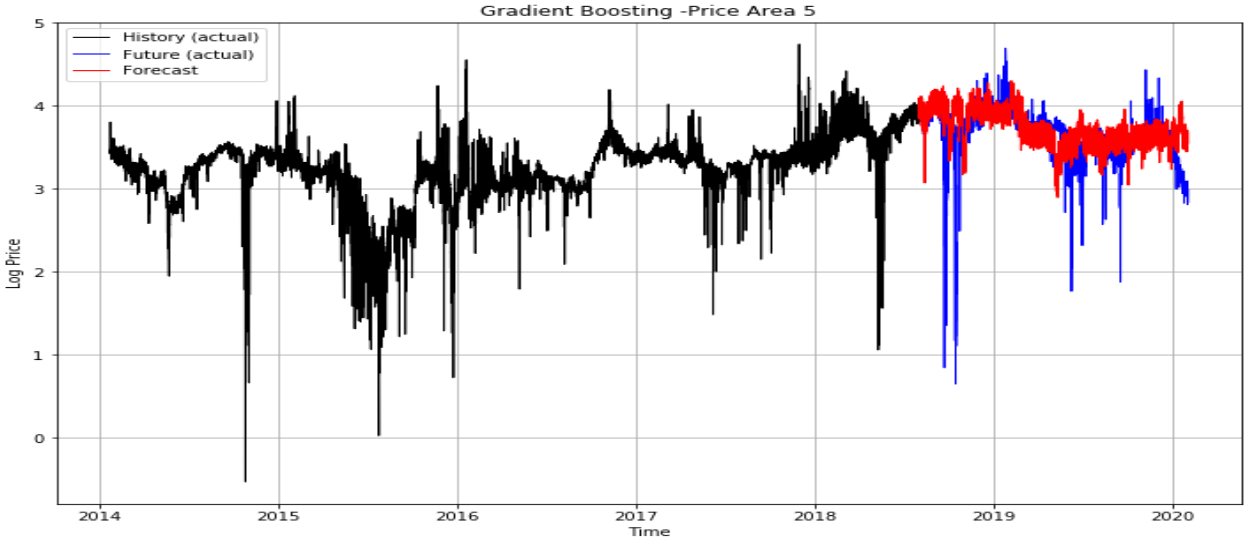


Table 5.1. A comparison of model performance for the four price areas in Norway. The decimals are rounded up to four digits. Extra Randomised Trees regressor has the lowest error rate for *all* price areas, while the baseline model has the highest error rate for *all* price areas.

	Price area 2	Price area 3	Price area 4	Price area 5
Linear regression **Training MSE: 0.0795**	0.0756	0.0995	0.1069	0.0913
Gradient Boosting **Training MSE: 0.1615**	0.0717	0.0924	0.0832	0.0883
Extra Trees Regressor **Training MSE: 0.1898**	0.0533	0.0741	0.0643	0.0675
Benchmark model	320.12	326.81	385.00	316.61

5.2. Discussion

It is no surprise to see that the persistent(naïve) model has the worst performance compared to the machine learning models for the time horizon that we have chosen. The Extra Trees regressor performs best on the out-of-sample data by giving the lowest test error for all price areas. The learning ability of the models on the training data also show that the Extra Trees regressor learns best by capturing both the trend and spikes found in the elspot price for price area 1, albeit at a higher training error compared to the linear regression model. Note also how the training error is higher for the Gradient boosting and Extra Trees regressor than the testing error. We suspect the reason for this to be due to the fundamental difference between the training and the testing data, since there is more noise or variance in the training data that the features fail to explain with regards to the elspot prices, due to the extended periods of low prices before 2018.

Visually it appears that the Linear regression model is better at handling the trend found in the data whilst the Gradient boosting is better at mirroring the spikes to a smaller degree. This is not to say that the models are good at handling this since most of them fail to predict price spikes for all the price areas. Price spikes are difficult to predict, and a solution could be to add variables that measure abnormal load[21] which gives the models needed information in modelling the price

[21] This refers to a sudden increase in the consumption of electricity that is unplanned for.

spikes. Another thing which is interesting to observe is how all the models fail to capture the downward trend that the prices are facing from winter 2019/2020. This raises an important question of choosing the optimal forecasting horizon, and we argue that employing seasonal forecasting will improve the models' predictive performance since seasonal forecasting is more susceptible to capturing and anticipating changes in elspot prices compared to prediction for longer time horizons. It becomes especially important in the times that we are living in now with the low gas prices and oil prices affecting the energy market which lowers the elspot prices, in addition to the fall of demand of electricity consumption due to the Corona pandemic which exacerbates the downward trend of prices that we already observe.

When it comes to machine learning models that can be applied to predicting elspot prices, it appears that models such as the Hidden Markov models can better capture the transition of elspot price from one 'state' (low prices) to another 'state' (high price). Other machine learning models that manage to memorise observed patterns due to seasonality for elspot price forecasting can be different kinds of Neural Networks, such as the Long Short-Term Memory(LSTM) and the Recurrent Neural Network which we have not covered here. To further improve the performance of the models that we have tested in this exercise, we could also include more essential climate variables such as wind power to better the models' forecasting ability since elspot price formation in recent time is shown to be more affected at a micro-level whereas temperature and hydropower are more important at a macro-level. Given this, it would be interesting to see what further research will discover in considering these issues to improve the predictive performance of the machine learning models that we have tested and proposed.

6. Conclusion

The uncertainty caused by the increased use of renewable energy sources makes it more essential to find good forecasting tools that can offset the increased risk in predicting elspot prices. In this thesis, we have compared different machine learning models to evaluate which models are more suited to predicting elspot prices for the different price areas in Norway. Using hourly data for elspot prices and exogenous variables such as temperature and energy prices collected from different sources, we discover that machine learning models outperform simple forecasting tools such as the naïve model. We also discover that some models are better suited for predicting elspot prices than others. The Linear regression model with the OLS estimation method has shown to be a superior forecasting tool compared to many of the models tested. The other two best performing models such as the Gradient boosting and the Extra Trees regressor show to better emulate the seasonality and trends found in the elspot prices, which opens a door for further research to study on how to increase the models' predictive performance.

Another implication of our findings is that using seasonal forecasting horizon together with adding more explanatory variables such as system load and wind power may result in an improvement of the models' predictive performance. This is especially important if the goal is to better predict better and predict price spikes which are an inherent feature of electricity.

Bibliography

Apergis, N., Gozgor, G., Lau, C. K. M., & Wang, S. (2019). Decoding the Australian electricity market: New evidence from three-regime hidden semi-Markov model. *Energy Economics, 78*, 129-142. doi:10.1016/j.eneco.2018.10.038

Athey, S., & Imbens, G. W. (2019). Machine Learning Methods That Economists Should Know About. *Annual Review of Economics, 11*(1), 685-725. doi:10.1146/annurev-economics-080217-053433

Bessec, M., Fouquau, J., & Meritet, S. (2016). Forecasting electricity spot prices using time-series models with a double temporal segmentation. *Applied Economics, 48*(5), 361-378. doi:10.1080/00036846.2015.1080801

Breiman, L. (2001a). Random Forests. *Machine Learning, 45*(1), 5-32. doi:10.1023/A:1010933404324

Breiman, L. (2001b). Statistical Modeling: The Two Cultures. *Statistical Science, 16*(3), 199-215. doi:10.1214/ss/1009213726

Brownlee, J. (2016). Machine learning mastery with python. *Machine Learning Mastery Pty Ltd*, 100-120.

Brownlee, J. (2017). *Introduction to time series forecasting with python*.

Contreras, J., Espinola, R., Nogales, F. J., & Conejo, A. J. (2003). ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems, 18*(3), 1014-1020. doi:10.1109/TPWRS.2002.804943

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78-87. doi:10.1145/2347736.2347755

Freund, Y., & Schapire, R. E. (1996). *Schapire R: Experiments with a new boosting algorithm.* Paper presented at the In: Thirteenth International Conference on ML.

Friedman, J. (1997). On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery, 1*(1), 55-77. doi:10.1023/A:1009778005914

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning, 63*(1), 3-42. doi:10.1007/s10994-006-6226-1

Glachant, J.-M., & Lévêque, F. (2009). *Electricity reform in Europe : towards a single energy market*. Cheltenham: Edward Elgar.

Gomez-Exposito, A., Troncoso, A., Riquelme-Santos, J. M., Gomez-Quiles, C., Martınez-Ramos, J. L., & Riquelme, J. C. (2017). Application of the Weighted Nearest Neighbor Method to Power System Forecasting Problems. In D. M. E. El-Hawary (Ed.), *Advances in Electric Power and Energy Systems: Load and Price Forecasting*: Wiley-IEEE Press; 1 edition.

Guthrie, G., & Videbeck, S. (2007). Electricity spot price dynamics: Beyond financial models. *Energy Policy, 35*(11), 5614-5621. doi:10.1016/j.enpol.2007.05.032

Hastie, T., Tibshirani, R., Friedman, J., Bickel, P., Diggle, P., Fienberg, S., . . . Zeger, S. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Second Edition ed.). New York, NY: New York, NY: Springer New York.

Hope, E. (2000). *Studier i markedsbasert kraftomsetning og regulering*. Bergen: Fagbokforl.

Hou, Y., Liu, C.-C., & Salazar, H. (2017). Electricity Prices as a Stochastic Process. In D. M. E. El-Hawary (Ed.), *Advances in Electric Power and Energy Systems: Load and Price Forecasting*: Wiley-IEEE Press; 1 edition.

Huurman, C., Ravazzolo, F., & Zhou, C. (2012). The power of weather. *Computational Statistics and Data Analysis, 56*(11), 3793-3807. doi:10.1016/j.csda.2010.06.021

James, G., Witten, D., Hastie, T., Tibshirani, R., Casella, G., Fienberg, S., & Olkin, I. (2013). *An Introduction to Statistical Learning: with Applications in R* (Vol. 103). New York, NY: New York, NY: Springer New York.

Jonsson, T., Pinson, P., Nielsen, H. A., Madsen, H., & Nielsen, T. S. (2013). Forecasting Electricity Spot Prices Accounting for Wind Power Predictions. *IEEE Transactions on Sustainable Energy, 4*(1), 210-218. doi:10.1109/TSTE.2012.2212731

Li, W., & Nyholt, D. R. (2001). Marker Selection by Akaike Information Criterion and Bayesian Information Criterion. *Genetic Epidemiology, 21*(S1), S272-S277. doi:10.1002/gepi.2001.21.s1.s272

Louppe, G. (2014). Understanding Random Forests: From Theory to Practice.

Misiorek, A., Trueck, S., & Weron, R. (2006). Point and Interval Forecasting of Spot Electricity Prices: Linear vs. Non-Linear Time Series Models. *Studies in Nonlinear Dynamics & Econometrics, 10*(3). doi:10.2202/1558-3708.1362

Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

Mount, T. D., Ning, Y., & Cai, X. (2006). Predicting price spikes in electricity markets using a regime-switching model with time-varying parameters. *Energy Economics, 28*(1), 62-80. doi:10.1016/j.eneco.2005.09.008

Mullainathan, S., & Spiess, J. (2017). Machine Learning: An Applied Econometric Approach. *Journal of Economic Perspectives, 31*(2), 87-106. doi:10.1257/jep.31.2.87

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python : a guide for data scientists*. Beijing: O'Reilly.

Olsson, M., & Soder, L. (2008). Modeling Real-Time Balancing Power Market Prices Using Combined SARIMA and Markov Processes. *IEEE Transactions on Power Systems, 23*(2), 443-450. doi:10.1109/TPWRS.2008.920046

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Time Series Cross-Validator. *Journal of Machine Learning Research.* Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

Pool, N. (2020a, 2020). Market Data. Retrieved from https://www.nordpoolgroup.com/Market-data1/#/nordic/table

Pool, N. (2020b). Price calculation. Retrieved from https://www.nordpoolgroup.com/trading/Day-ahead-trading/Price-calculation/

Schapire, R. E., & Freund, Y. (2012). *Boosting : foundations and algorithms*. Cambridge, Mass: MIT Press.

Shmueli, G. (2011). To Explain or to Predict? *, 25*(3). doi:10.1214/10-STS330

Statnett. (2020). Kraftmarkedet. Retrieved from https://www.statnett.no/for-aktorer-i-kraftbransjen/systemansvaret/kraftmarkedet/

Stewart, M. (2019). The Actual Difference Between Statistics and Machine Learning. *Towards Data Science.* Retrieved from https://towardsdatascience.com/the-actual-difference-between-statistics-and-machine-learning-64b49f07ea3

Varian, H. R. (2014). Big Data: New Tricks for Econometrics †. *Journal of Economic Perspectives, 28*(2), 3-28. doi:10.1257/jep.28.2.3

Varian, H. R. (2019). Artifical Intelligence, Economics, and Industrial Organization. In A. Agrawal, J. Gans, & A. Goldfarb (Eds.), *The Economics of Artifical Intelligence. An Agenda* (Vol. The University of Chicago Press). Chicago, London.

Weron, R. (2007). *Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach*. Oxford, UK: Oxford, UK: John Wiley & Sons Ltd.

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International Journal of Forecasting, 30*(4), 1030-1081. doi:10.1016/j.ijforecast.2014.08.008

Weron, R., Ziel, F., Soytaş, U., & Sarı, R. (2020). Electricity price forecasting. In U. Soytaş & R. Sarı (Eds.), *Routledge Handbook of Energy Economics* (1 ed., pp. 506-521). London: Routledge.

White, H. (2006). *Chapter 9 Approximate Nonlinear Forecasting Methods* (Vol. 1): Elsevier B.V.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67-82. doi:10.1109/4235.585893

Wooldridge, J. M. (2020). *Introductory econometrics : a modern approach* (Seventh edition. ed.). Boston, MA: Cengage Learning.

Zhou, Z.-H., & Yu, Y. (2009). AdaBoost. In X. Wu & V. Kumar (Eds.), *The Top Ten Algorithms in Data Mining* (pp. 141-164). USA: Chapman and Hall/CRC.

Ziel, F., & Weron, R. (2018). Day-ahead electricity price forecasting with high-dimensional structures: Univariate vs. multivariate modeling frameworks. *Energy Economics, 70*(C), 396-420. doi:10.1016/j.eneco.2017.12.016

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(2), 301-320. doi:10.1111/j.1467-9868.2005.00503.x

Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway