University of Ljubljana
Faculty of Computer and Information Science
Graz University of Technology
Institute of Theoretical Computer Science

Jaka Konda

Data stream fusion for predicting electricity price

master's thesis
the 2nd cycle master's study programme
computer and information science

Supervisor: prof. dr. Zoran Bosnić
Co-supervisor: Assoc. Prof. Dr. Robert Legenstein

Ljubljana, 2019

Acknowledgments

I would like to express my gratitude to my supervisor, prof. dr. Zoran Bosnić, for guidance, constructive feedback and his availability for various questions that arose while I worked on the thesis. I am also grateful to my co-supervisor, Assoc. Prof. Dr. Robert Legenstein, for his reviews, comments and corrections that helped improve the final work. Finally, I would like to thank my family and friends for their support during my studies and in the past few months while I was completing my thesis. This work would not have been possible without them.

Thank you.

Jaka Konda, 2019

Contents

List of used acronmys

acronym	meaning
ANN	artificial neural network
AR	autoregressive
ARIMA	autoregressive integrated moving average
DT	decision tree
ECMWF	European Centre for Medium-Range Weather Forecasts
ENTSO-E	European Network of Transmission System Operators for Electricity
k**-NN**	k-nearest neighbours
LR	linear regression
LSTM	long short-term memory
MAE	mean absolute error
(s)MAPE	(symmetrical) mean absolute percentage error
MCP	market clearing price
RF	random forest
RMSE	root mean square error
VAR	vector auto regression
VFDT	very fast decision tree

Abstract

Title: Data stream fusion for predicting electricity price

In this work, we tackle the problem of building a predictive model for the electricity market. In it, market players compete for the best prices to increase their profits and now with transition to renewable energy sources, the market has become more volatile and dependant on different environmental factors.

In our experiments we use different statistical and machine learning methods in a combination with data sources from European energy platform and European meteorological institute to obtain weather information. We fuse three different data sources together into four different datasets. We follow the machine learning pipeline where we select features, model hyperparameters and test the models on the test set.

Contrary to our expectations, additional weather information had a negative impact on the error and the variance of the models. Some improvements in the prediction accuracy were noted only when we included additional datasets with most important selected features being the lagged values of the target variable and also past and forecasted energy load.

Our best method is obtained using late level fusion by combining all trained regressors together into an ensemble. It achieves an sMAPE error of 13.084, compared to our second best, neural network, with an error of 14.932 and baseline with the value of 22.963.

Keywords

machine learning, time-series forecasting, incremental learning, data fusion, data streaming, electricity price prediction

Povzetek

Naslov: Zlivanje podatkovnih tokov pri napovedovanju cene električne energije

V delu se lotimo problema napovedovanja cen električne energije, kjer na trgu udeleženci tekmujejo za čim večji profit. V zadnjem obdobju s prehodom na obnovljive vire energije je trg postal bolj nepredvidljiv in odvisen od okoljskih dejavnikov.

V naših eksperimentih uporabimo različne statistične modele in modele strojnega učenja v kombinaciji s podatkovno zbirko Evropske mreže operaterjev električnega omrežja in Evropskega meteorološkega inštituta. Vse zbirke združimo v štiri različne podatkovne množice in sledimo cevovodu strojnega učenja. Najprej izberemo atribute, nato hiper-parametre modelov in nazadnje ovrednotimo modele na testni množici.

V nasprotju s pričakovanji, dodatne vremenske informacije niso pripomogle k izboljšanju uspeha modelov ter so imele celo negativen vpliv na napovedno uspešnost. Rezultati smo uspeli izboljšati, ko smo uvedli dodatne atribute iz Evropske mreže operaterjev, kjer so se za najpomembnejše izkazale zakasnjene vrednosti ciljne spremenljivke in pretekle ter napovedane porabe električne energije.

Najboljši rezultat smo dosegli z metodo ansamblov z uporabo poznega zlivanja. Ansambel doseže napako sMAPE 13,084, drugi najboljši model, nevronska mreža, 14,392, referenčni model pa napako 22,963.

Ključne besede

strojno učenje, napovedovanje časovnih vrst, inkrementalno učenje, zlivanje podatkov, podatkovni takovi, napovedovanje cene električne energije

Abstrakt

Titel: Datenstrom-Fusion zur Strompreisprognose

In dieser Arbeit beschäftigen wir uns mit dem Problem der Erstellung eines Vorhersagemodells für den Strommarkt. Um die Gewinne zu steigern, konkurrieren die Marktteilnehmer dabei um die besten Preise. Mit dem Übergang zu erneuerbaren Energiequellen wurde der Markt wechselhafter und ist nun von verschiedenen Umweltfaktoren abhängig.

In unseren Experimenten verwenden wir verschiedene statistische und maschinelle Lernmethoden in Kombination mit Datenquellen der europäischen Energieplattform und des europäischen meteorologischen Instituts, um Wetterinformationen zu erhalten. Wir fassen drei verschiedene Datenquellen zu vier verschiedenen Datensätzen zusammen. Wir folgen der Pipeline für maschinelles Lernen, in der wir Features auswählen, Hyperparameter modellieren und die Modelle auf dem Testset testen.

Entgegen unseren Erwartungen wirkten sich zusätzliche Wetterinformationen negativ auf den Fehler und die Varianz der Modelle aus. Einige Verbesserungen der Vorhersagegenauigkeit wurden nur festgestellt, wenn zusätzliche Datensätze einbezogen wurden, wobei die wichtigsten ausgewählten Merkmale die verzögerten Werte der Zielvariablen sowie die vergangene und prognostizierte Energielast waren.

Unsere beste Methode ist die Late-Level-Fusion, bei der alle ausgebildeten Regressoren zu einem Ensemble zusammengefasst werden. Diese erzielt einen sMAPE-Fehler von 13,084. Im Vergleich dazu erreicht die zweitbesten Methode, ein neuronales Netzwerk, einen Fehler von 14,932 bei einem Basislinie sMAPE von 22,963.

Schlüsselwörter

Maschinelles Lernen, Zeitreihenprognose, inkrementelles Lernen, Datenfusion, Daten-Streaming, Strompreisvorhersage

Razširjeni povzetek

V delu se lotimo problema napovedovanja cen električne energije. Na tem področju proizvajalci, distributerji, trgovalci in udeleženci na trgu tekmujejo za najboljše cene. V zadnjih dveh desetletjih je področje doživelo kar nekaj preobrazb. Trg se je iz državnega nadzora počasi preobrazil v odprt trg. Glavno vodilo za to transformacijo je bil razlog, da bi konkurenca lahko pripeljala do bolj učinkovite rabe sredstev in znižanja cen ter večje zanesljivosti omrežja.

Nov trg je tako usmerjen v stranke, kjer vsi tekmujejo za najnižje cene, interesi igralcev na trgu pa so najpogosteje v konfliktu. Medtem ko želijo generatorji prodati energijo za čim več, jo distributerji želijo odkupiti po čim nižji ceni. Trgovanje zato poteka v obračunani tržni ceni, ki se izračuna glede na oddane nakupne in prodajne cene. Velika odstopanja lahko pripeljejo do tega, da ponudba ni sprejeta.

Na ceno vpliva mnogo različnih faktorjev. V zadnjih letih pospešen prehod na obnovljive vire energije ponekod že povzroča negativne cene in s tem na trgu povečuje nepredvidljivost. Prehod na bolj učinkovite vire ogrevanja kot so toplotne črpalke, in na drugi strani velike porabnike energije, kot so električni avtomobili, naredi vse skupaj še bolj volatilno.

V delu se osredotočimo na napovedavanje cene za 24 ur vnaprej z uporabo različnih podatkovnih virov in primerjavo različnih strojnih in statističnih modelov med seboj. Upamo, da bomo dosegli enake rezultate kot druge uporabljene metode in raziskali možnost uporabe različnih nekoreliranih virov podatkov.

I Kratek pregled sorodnih del

Čeprav je raziskav na področju napovedovanja cen električne energije razmerno veliko, jih je veliko opravljenih na starejših podatkovnih množicah in le na ciljni spremenljivki. Contreras in ostali [1] so leta 2003 napovedovali cene za dan vnaprej s statističnim modelom ARIMA z napako 10%. Dve leti kasneje, 2005, so Garcia in ostali [2] preverjali GARCH modele za 24ur naprej in dosegli napako 9%.

Z metodami strojnega učenja so preizkusili metodo podpornih vektorjev in primerjali z nevronskimi mrežami na drugi podatkovni množici [3]. Napake so bile podobne kot pri statističnih modelih, 25% za teden vnaprej, kjer je bila metoda podpornih vektorjev bolj konsistentna. Leta 2009 so Mandal in ostali [4] na novejši podatkovni zbirki za področje Pensilvanije uporabili nevronske mreže, Mei in ostali [5] pa naključne gozdove. Oboji so dosegli napako za dan vnaprej približno 12%.

Leta 2018 so Lago in ostali [6] izvedli večjo primerjalno študijo različnih statističnih in strojnih modelov, s poudarkom na nevronskih mrežah. Uporabili so tudi precej novejše podatke ze Belgijsko regijo za leto 2016. Najbolje se izkaže globoka konvolucijska nevronska mreža z napako 12.3%. ARIMA-GARCH, ki se je na drugi 15 let starejši zbirki izkazala mnogo bolje, je dosegla 19.3%.

Sorodna dela so v večini uporabila zgolj ciljno spremenljivko in zakasnjene vrednosti, brez dodatnih atributov ali dodatnih podatkovnih zbirk, ki bi lahko imele vpliv na napovedno točnost. Lago in ostali omenijo še uporabo cen v sosednji regiji in porabo električne energije.

II Predlagana metoda

Najprej zberemo podatke iz različnih virov. Začnemo s primarnim virov podatkov iz Evropske mreže operaterjev za prenos električne energije (ENTSO-E - European Network of Transmission System Operators for Electricity), ki vsebuje ciljno spremenljivko ter dodatne informacije o omrežju, proizvodnji

in porabi energije v urnem intervalu. Drugi vir podatkov je iz Evropskega meteorološkega centra za napovedovanje vremena (ECMWF - European Centre for Medium-Range Weather Forecasts) in vsebuje podatke o hitrosti vetra, količini dežja in sončne svetlobe ter še nekaj drugih s tri-urnim korakom. Podatki niso točni vremenski podatki, ampak so vremenske napovedi za bolj resnično simulacijo v neki točki. Kot zadnji vir uporabimo še podatke o cenah fosilnih goriv z informacijo poročano enkrat dnevno.

Podatke pripravimo in jih zlijemo z metodo zgodnje integracije po časovnem ključu. Manjkajoče vrednosti zapolnimo z linearno interpolacijo in podatke razbijemo na štiri množice. Učno in validacijsko na regiji SE4 in učno in testno za Belgijski trg. Učne zajemajo leto 2015, validacijska in testna pa 2016. Zbirke razdelimo na osnovno, brez dodatnih atributov, ENTSO-E, ki vsebuje dodatne podatke o električnem omrežju, ECMWF, ki vsebuje vremenske spremenljivke in vse, ki vsebuje vse množice združene skupaj. Ponovimo za dve različici vsake izmed pripravljenih zbirk. Prvo uporabimo takšno kot je, v drugi pa generiramo dodatne atribute. Ti so predvsem zakasnjene vrednosti do dveh tednov, razbitje datuma in ure v komponente in binarnega kodiranja le-teh ter tehnični indikatorji, kot so drseče povprečje, Bollingerjev pas in standardni odklon.

Velika količina atributov ima zelo pogosto negativen vpliv na natančnost modelov. Atribute tako za vsako podatkovno zbirko in model izberemo na podlagi univariatnega filtra, ki izbere atribute na podlagi korelacije. Poizkusimo tudi LASSO in genetski algoritem za izbiro atributov, a se izkažeta slabše kot preprosto filtriranje. Preizkusimo na validacijski regiji za različno število atributov in vzamemo število, kjer je napaka najmanjša. Postopek ponovimo za vse metode.

Po izbranem najboljšem številu atributov jih toliko ponovno izberemo na testni regiji in pridobimo končne rezultate. Modele ponovno naučimo vsak konec meseca, da zajamejo spremembe v časovni vrsti in se naučijo na večji množici podatkov. Za poročanje uporabimo simetrično absolutno napako (sMAPE) in koren povprečne kvadratne napake (RMSE). Prva predstavlja

napako v odstotkih in je prilagojena, da je napaka v obe strani enaka, druga pa standardni odklon modelov.

III Eksperimentalna evalvacija

Med naučenimi modeli se najbolje izkaže ansambel vseh modelov, ki doseže napako 13.1%. Med samostojnimi modeli se najbolje izkažejo modeli z dodatnimi atributi. Na prvem mestu je umetna nevronska mreža s 14.9% simetrične napake, sledi ji naključni gozd s 16.4%. Vsi modeli, ki uporabljajo dodatne atribute uspejo premagati referenčni model. Izjema so statistični modeli in tisti, ki dodatnih atributov ne uporabljajo. V tem eksperimentu se izkaže, da so zakasnjene vrednosti ciljnega atributa izredno pomembne, medtem ko so vse ostale bistveno manj.

Med izdelanimi podatkovnimi zbirkami se najbolje obnesejo zbirke z dodatnimi atributi in dodatnimi podatki skupaj s porabo električne energije in njenim primankljajem. Ostali podatki se pri uporabljeni metodi izbire atributov izkažejo za irelevantne.

Pregledamo napake modelov za različna obdobja skozi leto, teden in dan. Vsi modeli imajo napake med prehodnimi obdobji. Na letnem nivoju sta ta dva meseca predvsem april in september in oktober, ko se začne ali konča obdobje ogrevanja. Tedensko so povečane napake zjutraj in okoli pete ure popoldne, ter med vikendi. Vsa ta obdobja so obdobja sprememb, kjer se spremenijo zakasnjeni koeficenti modelov, ki se izkažejo za najbolj pomembne.

IV Sklepi

V delu se poglobimo v problem napovedovanja cene električne energije za belgijski trg. V pregledu sorodnih del opazimo, da so enake metode skozi čas postale precej slabše, kar nakazuje večjo volatilnosti trga. Sklepamo, da k temu pripomore prehod na obnovljive vire energije, saj so ti zaradi vremena precej bolj nepredvidljivi. Zato poleg glavnega vira podatkov upora-

bimo še podatke pridobljene iz ECMWF in jih uporabimo pri napovedovanju. Uporabimo tudi dodatne podatke iz Evropske mreže operaterjev za prenos električne energije.

Sledimo klasičnemu cevovodu za strojno učenje, kjer pripravimo podatke, jih razdelimo na učno in validacijsko množico na eni državi, ter učno in testno na končni. Generiramo in izberemo atribute, določimo hiperparametre modelov in na testni množici pridobimo končne rezultate za poročanje.

Izkaže se, da vremenski podatki na izbrani državi ne pripomorejo k izboljšanju točnosti napovedi za izbrano državo, ker ima ta nižji delež obnovljivih virov energije in bi bil zato eksperiment primernejši na državi z višjim deležem le-teh, npr. Danska. Dodatni atributi iz primarne podatkovne zbirke pa pomagajo pri natančnosti modelov. Največji delež ima primankljaj energije za izbran trg (angl. *negative imbalance price*) in poraba električne energije v preteklosti ob enaki uri.

V prihodnje bi se bilo bolje osredotočiti na eno ali peščico izbranih metod, ki so se izkazale najbolje in jih izpopolniti z boljšo izbiro atributov in hiperparametrov. Napovedi naučenih modelov pa na koncu združimo v ansambel metod, saj se je ta izkazala za najboljšo.

Chapter 1

Introduction

In the last decades, worldwide electricity underwent extensive and numerous changes. It started as a closed system where all activities (generation, transmission and power distribution) were regulated by the government and slowly transitioned into an open market. The main driving force was that the competition could result in more efficient utilisation of resources, higher reliability and consequently lower prices for end consumers [7].

With deregulation, customers now have a choice to pick their own electricity supplier, transforming the market into more customer oriented, governed by the supply and demand relationship. In the new market several different actors emerged, generators, investors, traders and load serving bodies, each trying to maximise their profit margin. Their interests are most often in conflict with each other. If we take, for example, distributors and suppliers, distributors try to maximise their profit by buying low from suppliers and selling high, while suppliers try to sell to distributors as high as possible introducing a conflict of interests.

Trading is therefore done in a range of market clearing price (MCP) also known as the price equilibrium, illustrated in Figure 1.1. Buyers and suppliers must submit their offers at least 24 hours ahead of time without the knowledge of competitor prices. MCP is determined afterwards by intersecting the aggregated curves of supply and demand. Any deviations might

result in an offer not being accepted by the other party. The addressed problem depends on many factors, which are constantly evolving through time. Some of those have long term evolving impact, such as the transition from fossil fuels to renewable energy source, slow adoption of electric vehicles and the transition to heating pumps in consumer homes. There are also many unpredictable factors, such as weather impact which affects wind turbines, rainfall having an influence on hydro generation and other connected markets such as prices of fossil fuels.

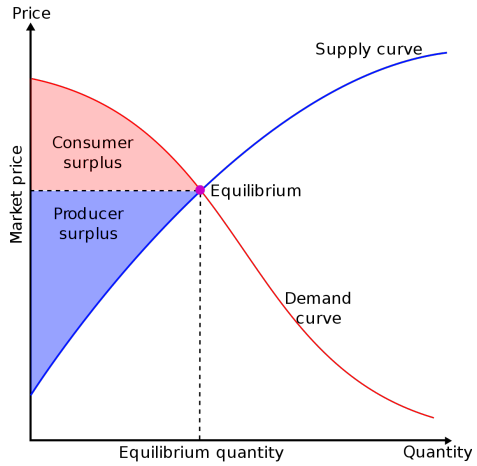

Figure 1.1: Plot of visual determination of price equilibrium. Blue curve represents the distribution of supply in relation to quantity and price, while the red curve represents demand. The intersection point of these lines is the equilibrium.

In this master's thesis we will address the problem of predicting MCP price for different forecast ranges with data fusion from different unrelated data sources. With constantly evolving market, we will explore different data fusion techniques and incremental machine learning methods to predict energy prices in a volatile market, evaluate and compare them with prevalent approach that are currently still based on statistical models. We expect the master's thesis to yield a comparative analysis of the state-of-the art approaches and address the potential of fusing different heterogeneous data sources.

1.1 Related work

In 2003, Contreras et al. [8] concluded their work on Spanish and Californian energy prices during the year 2002 using statistical ARIMA models. They were predicting for a day ahead and achieved the error (MAE) of 10%. R. Garcia [2] extended this approach in 2005 with GARCH models on the same dataset and decreased the error down to 9% for the same range. Both were using solely statistical models and price-based information for making predictions.

Methods using machine learning techniques were also explored. Support Vector Regression was used and compared against neural networks for predicting Australian prices in [3]. Predicting one week ahead, the results for both methods were similar to Contreras et al. with an MAE of 10% and 25% for periods longer than a week. In 2009, Mandal [4] proposed an artificial neural network on Pennsylvania–New Jersey–Maryland (PJM) electricity market and later Mei et al. [5] using random forest technique with an error of 12%.

In 2018 Lago et al. [6] conducted a large comparison of newer statistical and machine learning algorithms for predicting price for the next 24 hours based on recent data. They used European energy market prices from Belgium stock exchange included as a part of ENTSO-E [9] and also reported using weather information. They showed that machine learning algorithms perform better than statistical models, however their best model, deep neural network, achieved MAPE error of 12.3% and ARIMA-GARCH 19.3%.

Additionally to the mentioned work, we intend to use other unrelated data sources together with data fusion to be able to derive additional insights from the data and improve prediction accuracy.

Chapter 2

Related work

In this chapter, we will overview the basics of time-series forecasting and its properties. We will also explain how we will utilise different statistical and machine learning methods for evaluation and prediction. In the last section, we review different most commonly found data fusion methods and their common classifications.

2.1 Time series

A time series is a sequence of data points in a chronological order, most often gathered in regular intervals (e.g. one data point per second). Series are very frequently plotted as line charts and are used in areas such as finance, weather forecasting, earthquake prediction, signal processing and many more. Time series analysis can be applied to any variable that changes over time. By using such data, we may make certain generalisations that may improve accuracy, such as that data points closer together are more similar than those further apart.

When talking about forecasting time-series we are referring to the prediction of the stochastic process generating the series. Two approaches deal with time-series forecasting. First, the theoretically oriented approach is named confirmatory data analysis. It works with simplified models generating the

data and trying to understand it. This process is theoretically more sound but it is much slower in terms of development. The opposite method is exploratory data analysis. Here, we work directly with the data, generated by the specific model and trying to fit our model to mimic the original one. The obtained result is specific to the process generating the data and is obtained much faster, but it might capture wrong properties of the process.

Some broad important classes of models include the autoregressive (AR), the integrated (I) and the moving average (MA) models. Classes can be combined to form autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) to obtain more complex and better performing models. Those also have additional extensions such as seasonal ARIMA (SARIMA) and expanded to multivariate series such as Vector Autoregression (VAR).

Among other types of models, some models try to represent change in variance of the data. They are known as autoregressive conditional heteroskedasticity (ARCH) models. One of the most widely known is the generalised ARCH (GARCH) [2].

2.1.1 Stationarity

Stationarity is an important property because stationary processes are much easier to analyse and investigate. Its statistical properties are not changing, so we can use autocorrelation for forecasting. While a lot of time-series processes are not stationary, most of them can be converted into a stationary form with certain transformations. Due to this property, most models require stationary series as an input. The stationarity is divided into strict and weak. A visual representation of stationary and non-stationary series can be seen in Figure 2.1

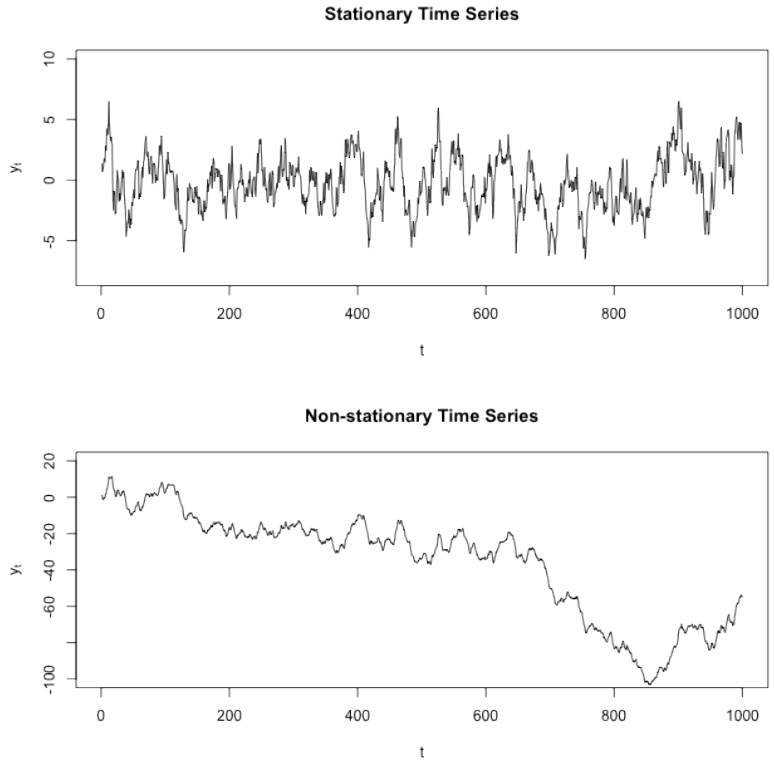

Figure 2.1: A visual representation of a generated stationary (top) and non-stationary time series (bottom) [10].

.

Types of Stationarity

Strict stationarity requires time and order invariant cumulative distribution (2.1) to be the same for any finite sub-sequence of the stochastic process:

$$F_X(x_t, ..., x_{t+k}) = F_X(x_{t+\tau}, ..., x_{t+\tau+k}) \quad \forall t, \forall k, \forall \tau \in \mathbb{Z} \tag{2.1}$$

where x_t denotes value at a time point t, k the length of the sub-sequence and τ the gap between sub-sequences. It consequently also means that mean and variance do not change over time, which is often not the case.

A less restraining type of stationarity, which may be achieved in most cases, is weak stationarity. It only requires the shift-invariance (in time)

of the first moment (mean) and the cross moment (the auto-covariance). This means the process has the same mean at all time points, and that the covariance between the values at any two-time points, t and $t-k$, depends only on the distance k, not their absolute location in the series:

$$\begin{aligned} \mathbb{E}[x_t] &= \mu, \forall t \\ \mathbb{E}[x_t^2] &< \inf \\ \mathbb{E}[(x_t - \mu)(x_{t+k} - \mu)] &= \gamma(k), \forall t \wedge \exists k \end{aligned} \tag{2.2}$$

where μ denotes the mean, $\mathbb{E}^n$ n^{th} moment, first is the mean and second the standard deviation. γ is the autocovariance function.

Weakly stationary series satisfy our requirement of stationary cross moment, so we can apply auto-correlation analysis. We can check if the series is stationary using several statistical tests, such as the augmented Dickey-Fuller test(ADF) [11]. This is a statistical test which has a null hypothesis that a unit root is present in the series. Roots can be considered as parameters in auto-regressive processes as described in the section 2.2.1. If any of the absolute values of parameters is equal or greater than one, it means that the series will not return to its average value bit, but it will have an upwards or a downwards trend.

Components of time-series

Most time series are composed out of three components:

1. **Trend**: a long term general direction. An example would be population growth, price inflation and general economic changes.

2. **Seasonality**: a cyclic trend, most often depends on timing. It can be identified by regularly spaced peaks. We can observe such trends with tourist visitors during the year with large oscillations between summer and winter periods.

3. **Noise**: a non-systematic component that cannot be predicted nor classified as a trend or seasonality.

2.2 Methods

In this section, we describe different types of used forecasting algorithms. We start with basic statistical models very often seen at stock markets used forecasting various instruments. We then extend these models into more complex ones and later move to different incremental and batch machine learning methods.

2.2.1 Statistical methods

Autoregressive Process

An autoregressive (AR(p)) [12] process predicts future behaviour based on the past observation for univariate time-series. An AR(1) is the first-order process (2.3) which accounts only the previous value to forecast the current one:

$$y_t = c + \theta y_{t-1} + \epsilon_t \tag{2.3}$$

The parameter p specifies how many lagged values y_{t-i} AR process uses. Variable c represents the mean of the time-series which requires stationary data. ϵ_t is for the white noise process, modelled as i.i.d. (independent and identically distributed) with a parameter σ^2 and it represents the unexplained portion of the model. The AR(p) is described by:

$$y_t = c + \theta_1 y_{t-1} + \theta_2 y_{t-2} + ... + \epsilon_t = c + \sum_{i=1}^{p} \theta_i y_{t-i} + \epsilon_t \tag{2.4}$$

Based on the equation of the AR models, we can derive the following behaviours for the AR(1):

- when $\theta_1 = 0, y_t$ is equivalent to white noise as only the ϵ term is used;
- when $\theta_1 = 1, c = 0, y_t$ is equivalent to a random walk. The past terms are be summed together and some noise is added at each step. The process is not stationary;

- when $\theta_1 = 1, c \neq 0, y_t$ is equivalent to a random walk with drift as the constant provides predictable direction and speed;

- when $\theta_1 < 0, y_t$ tends to oscillate around the mean as the past values decay in relevance.

The θ parameters are obtained using the least squares method, maximum likelihood estimation or methods of moments such as the Yule-Walker estimator [13].

Visually, we can explain how AR process propagates changes, shown in Figure 2.2. At time t there is an unknown shock introduced into the process generating the series. An unknown shock could be a failure of a power plant or in the network grid. This shock would have a big impact when it occurs and then it would gradually decay. In theory, such shocks have infinite effects, but in practice, they converge towards zero.

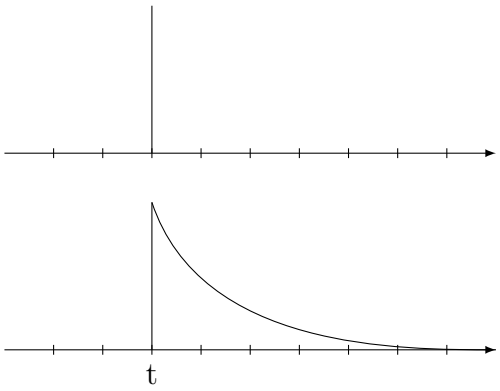

Figure 2.2: An example when shock happens at time t (top) and relevance how AR process treats such shocks during forecasts (bottom).

Moving Average Process

A moving average process (MA(q)), not to be confused with the rolling mean or also named as a simple moving average. The equation (2.5) is visually very similar to AR with past values y_{t-i} replaced by past errors ϵ_{t-i} and coefficients

denoted by σ_i:

$$y_t = c + \epsilon_t + \sigma_1\epsilon_{t-1} + ... + \sigma_q\epsilon_{t-q} = c + \epsilon_t + \sum_{i=1}^{q} \sigma_i\epsilon_{t-i} \tag{2.5}$$

Due to this change, past predictions are not included and are gradually reduced over time as in AR, but only the last q steps are accounted for as seen in Figure 2.3. Because the MA process is non-linear, parameters are therefore fitted using iterative methods instead of least squares or others. It is mostly used in combination with an AR process to improve the accuracy of the models.

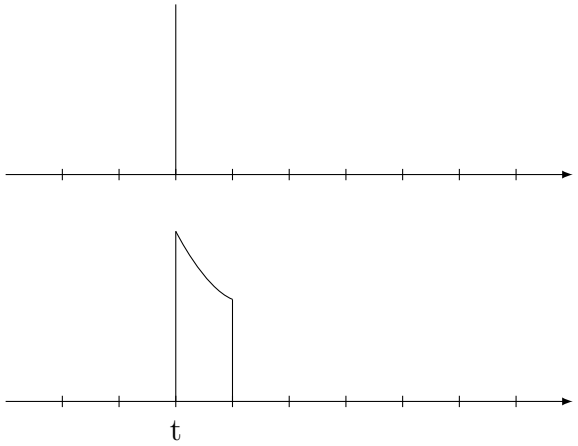

Figure 2.3: An example when shock happens at time t (top) and relevance how MA(1) process treats such shocks during forecasts (bottom).

Autoregressive Integrated Moving Average Process

Both auto-regressive and moving average processes can be combined to form an ARMA(p, q) process. The p and q parameters are inherited from AR and MA processes and the equation is just a combination of both:

$$y_t = c + \sum_{i=1}^{p} \theta_i y_{t_i} + \sum_{j=1}^{q} \sigma\epsilon_{t-j} \tag{2.6}$$

Integrated part comes from integrated processes and indicates differenc-

ing consecutive values. With it we remove the overall trend. It is often transformed using logarithmic transformation to acquire differences in percentage:

$$\Delta x = log(y_t) - log(y_{t-1}) \tag{2.7}$$

First order of differencing $I(1)$ removes the linear trend, second-order $I(2)$ quadratic etc. All of the mentioned models together form $ARIMA(p, d, q)$ [1].

Vector Autoregression

Models in the ARIMA family use information only from univariate time-series, despite having multiple variables available, such as weather information or the instrument price a different market. Vector auto-regression (VAR) [14] is a model which tries to capture such linear inter-dependencies across multiple time series. It is a generalisation of the univariate AR model by using lagged values from other variables. An example of two variables can be seen in equation:

$$\begin{aligned} y_t &= c_y + \theta_1^y y_{t-1} + \theta_2^y y_{t-2} + \theta_1^x x_{t-1} + \theta_2^x x_{t-2} + \epsilon_y \\ x_t &= c_x + \theta_1^x x_{t-1} + \theta_2^x x_{t-2} + \theta_1^y y_{t-1} + \theta_2^y y_{t-2} + \epsilon_x \end{aligned} \tag{2.8}$$

Parameters θ represent weight factors of lagged values. This equation can be written in a more compact way and is also able to be easily generalised by using algebraic notation if we denote the matrix A to represent θ parameters:

$$\begin{aligned} \begin{pmatrix} y_{1t} \\ y_{2t} \end{pmatrix} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{pmatrix} y_{1t-1} \\ y_{2t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{pmatrix} \\ y_t &= A y_{t-1} + \epsilon_t \end{aligned} \tag{2.9}$$

Such model implies that all variables are dependant on all observations from the past. As we can see from equations (2.8), those are independent from one another so we can solve them separately. Similarly as with AR processes, there are extensions for VAR, such as Vector Error Correction

Model (VECM) which estimates how a group of variables move together despite being non-stationary.

2.2.2 Machine learning methods

Batch learning methods are trained once and then used for predictions. They are unable to adapt to changes that occur through time known as concept drift [15] and statistical changes in properties of the target variable. This may cause problems because the predictions become less accurate over time.

Contrary, incremental learning methods are always ready for predictions and work per sample, eliminating the need to fit the entire dataset into working memory. With streaming being their core feature, they require less memory and can adapt to concept changes.

Linear regression

Linear regression is one of the simplest approaches for machine learning. It works by identifying the relationship between two independent variables and the target variable. For multiple attributes, the method is extended into multivariate linear regression:

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x \tag{2.10}$$

In the equation, θ_i are the coefficients we need to select to make predictions. When selecting the coefficients, we want the model error to be as small as possible. To do so, we use the squared error:

$$\epsilon = \sum_{i=0}^{n} (\hat{y}_i - y_i)^2 \tag{2.11}$$

It sums the differences between the correct value y_i and the predicted $\hat{y}$ for each sample i. This handles errors in both direction equally and penalises larger mistakes more than smaller ones.

k-nearest neighbours

k-nearest neighbours method [16] (k-NN) is a classification and regression non-parametric algorithm. It is a lazy algorithm meaning that all computation is deferred until prediction, thus eliminating training time.

The method returns the result based on k most similar neighbours in feature space for its prediction. Similarity is calculated with different metrics. Most widely used is Euclidean distance with some other common ones being Manhattan, Hamming and cosine distances.

During the learning phase, the model stores all of the training samples. During prediction, the algorithm uses k most similar neighbours. For regression, target variable of the retrieved neighbours is averaged and used as a predicted value.

Regression tree

Regression tree builds regression models in the form of a tree structure. It breaks down a dataset into smaller subsets while at the same time building an associated decision tree. The final result is a tree with decision nodes having two or more branches and the final decision in leaf node.

The core algorithm for building regression trees is called ID3 [17] which employs a top-down, greedy search through the space of possible branches. ID3 uses standard deviation and standard deviation reduction to construct a regression tree. Standard deviation defines impurity or randomness of the data and is zero when the dataset is completely homogeneous.

At each iteration the algorithm splits based on the attribute contributing to the standard deviation reduction, calculated as difference before and after the split. In the leaves, average values are calculated based on the remaining data and are used as prediction results.

In order to achieve unique, separating leaves we normally require many branching steps which often leads to overfitting. To avoid this there are several additional conditions while building such as maximum depth of the

tree, minimum number of samples in leaves, allowed impurity and tree pruning [18].

Random forest

Random forest [19] is a state-of-the-art ensemble learning method. Ensemble methods combine and aggregate results from several weaker regression trees into a stronger one, following the wisdom of the crowd principle.

Random forest is trained via the bagging method [20]. It consists of randomly sampled subsets of training data, each subset used to train a single model. This way it introduces larger randomness and diversity leading to a more robust final prediction. For the final aggregated prediction, all of the weak learners make their prediction and the average of those is used for regression or a majority class for classification.

Hoeffding tree

Hoeffding tree [21], also known as a very fast decision tree (VFDT), is an incremental learning algorithm, generating nearly identical trees as conventional batch algorithms. It can learn by seeing each example only once, eliminating the need to store any during training. Memory space requirements are therefore only needed for the tree itself and any associated statistics with the process.

The name is derived from Chernoff-Hoeffding bound [22, 23] which is used to determine the number of examples needed to achieve a certain level of confidence. It states with a probability of $1 - \delta$, that the true mean $\overline{r}$ of the variable is at least $\overline{r} \pm \epsilon$, where ϵ is obtained by the equation (2.12) where R^2 denotes random variable range. It gives us the same result regardless of the distribution, however, less distribution dependent attributes will require more observations n needed to reach the same δ and ϵ.

$$\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \tag{2.12}$$

Hoeffding tree try to determine the best attribute a with high probability with as few examples as possible. Maximisation function G can be the same as in decision trees, such as entropy, Gini index, information gain etc. Once the difference between the best and second-best attribute is large enough (larger than ϵ), the node is split on that attribute and associated statistics are updated. For a faster head start, the initial tree can be trained with one of the batch methods, decreasing the total number of examples needed to reach the same accuracy.

Such trees were also adapted to detect concept change. Such examples are CVFDT [24] and Hoeffding Adaptive Tree [25]. First one uses a window of examples to perform change detection, while the second one uses different models for example a Kalman filter [26], linear or a statistical model.

Neural Networks

Today one of the most used methods in machine learning are neural networks. Artificial neural networks (ANN) are loosely analogous to human brains and can be applied to both classification and regression problems. In the recent years, they managed to outperformed other algorithms in a large variety of fields.

Their basic units are neurons. Each neuron has some inputs signals which are summed together and the result is passed through an activation function h which calculates the output. Three most known activation functions are sigmoid, hyperbolic tangent (tanh) and rectified linear units (ReLU) [27]. Their main purpose is to introduce non-linearities, so that each layer can build upon the results of the preceding non-linear layers. An example of a single neuron is in Figure 2.4. Additionally, each neuron has a bias which is independent from the data with a purpose of shifting the activation function left or right.

Neurons as the basic block are then combined into networks. The first and the simplest artificial neural networks are feed-forward neural networks. In this type of networks, the information moves only forward, from input,

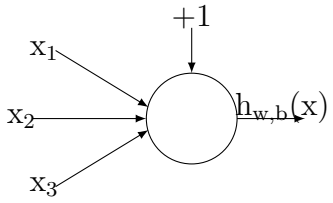

Figure 2.4: A simple neuron with three input weights, bias and an output.

through hidden nodes to the outputs. Networks are organised in layers where each neuron is connected with some accompanied weight to all neurons on the previous layers. Such connections represent fully connected layers. Moving forward, results from layer $n-1$ are passed through these connections where the neurons sum the weights and data and apply the activation function to the result. This process is repeated until we reach the end. An example of a simple fully connected layer is in Figure 2.5. These summations and weight multiplications can be represented with a dot product which passes between layers, following the equation:

$$y = h(W \times x) \tag{2.13}$$

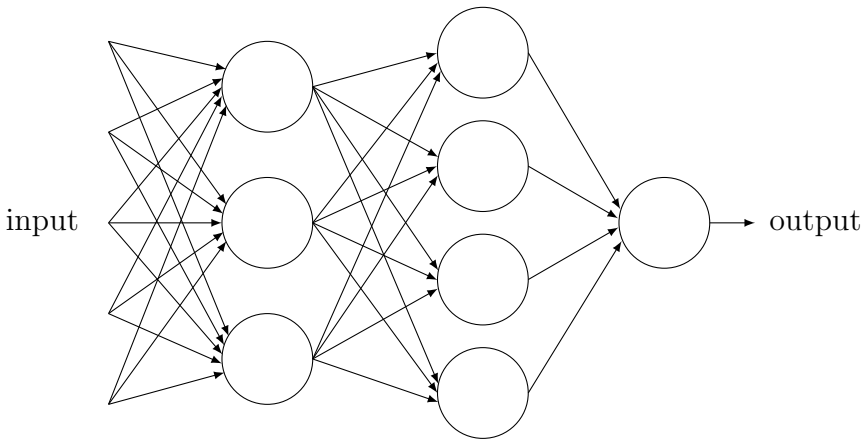

Figure 2.5: An example of a neural network with an input layer (which we do not count), two hidden layers and one output layer.

Weights of a neural network need to be properly adjusted to give meaning-

ful results. The learning process is an extended version of a SGD [28]. Data samples are passed in mini-batches through the network. A prediction error is calculated for each batch, and that error is back-propagated through the neural network to adjust the weights. Backpropagation, a family of learning algorithms for ANNs, is used to efficiently calculate the gradients by utilising the chain rule when updating the weights backwards. They require the derivatives of the activation function to be known.

2.3 Data fusion

The idea of using multiple senses for a specific task is old and introduced by evolution. Humans and animals use sight and hearing for assessing surroundings, smell and vision for determining if the food is edible, which would be hard with only a single sense. Formalized, data fusion is a process of combining multiple data sources to achieve more useful and complete information than any provided individual data source. The first most widely accepted definition of data fusion was provided by the Join Directory of Laboratory (JDL) [29], and was later simplified and revised by Hall and Llinas [30] to the following definition:

> *Data fusion techniques combine data from multiple sensors and related information from associated databases to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone.*

While the technique was at the beginning primarily employed in multi-sensor environments, it can be used in many different applications.

Researchers split the data fusion techniques according to different criteria which are non-exclusive between themselves:

- **relations between datasets** - Durrant-Whyte [31] defined methods based on the relations between datasets. First are complementary, to obtain more complete global information, second group are redundant,

where the data provides the same information and are used to increase the confidence, and last are cooperative, where combining data give us new, more complex information;

- **input/output data representation** - Dasarathy [32] categorised (Figure 2.6) the data based on the level, whereas data is the lowest, followed by feature level and finally decision level fusion. The interactions between levels normally combined sources to obtain more reliable information (e.g. position combined from GPS and INS - internal navigation system) or between levels to derive new information (depth perception from the eyes);

- **abstraction level** - Luo et al. [33] created a hierarchy based on the type of data, started with raw data, followed by signals, characteristics and decisions;

- **process levels** - JDL with cooperation with Department of Defense defined five levels in their fusion framework. Preprocessing, object refinement, situation followed by threat assessment and process refinement. The models are normally related to their application domain.

Several classifications resulted from interdisciplinary environments such as aviation where the data is obtained from multiple sensors and is now often referred to multi-sensor fusion.

In the machine learning domain, fusion was reduced to a three-level architecture and is most commonly employed as described by Pavlidis et al. [34] and visualized in Figure 2.7. The first level addresses the integration of multiple data sources. This can be done by calculating the similarities between common attributes or by a single key such as date and time in time-series.

Intermediate fusion builds on separate data sources to extract or build some features, which are then used for final decision making.

The final level, late integration, builds a model for each data source and combines the predictions. The combined predictions can be a simple average

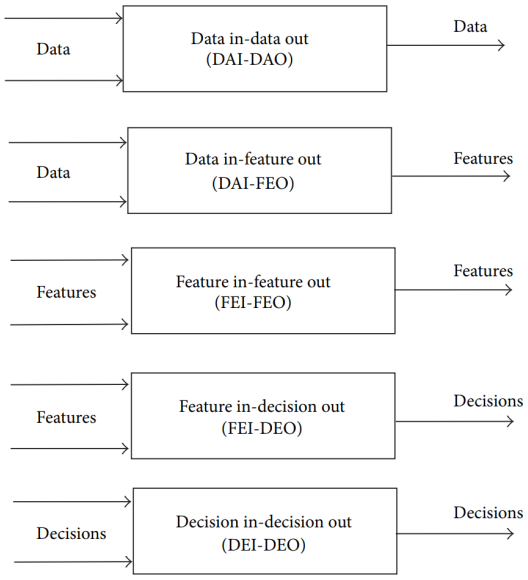

Figure 2.6: Five fusion levels based on Dasarathy's classification

or a simple model such as linear regression to reduce the risk of overfitting. Combination of results from different models is also known as ensembles.

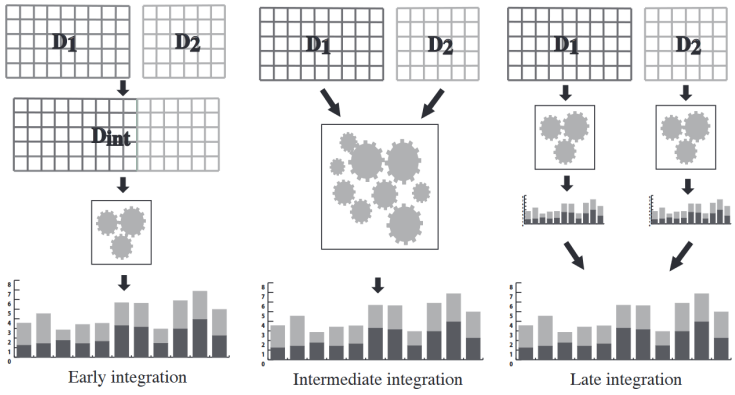

Figure 2.7: Visual representation of fusion performed at three different levels [35].

Chapter 3

Datasets

In this chapter we describe the used datasets and some of their properties. We used three different data sources as follows. Primary is combined from multiple smaller datasets where we obtain the prices and other information about the energy network at certain timepoints. Second is the weather dataset which we use to acquire different weather information under the assumption that it may help increase forecasting accuracy due to weather effects on renewable energy sources. The final dataset provides information about other commodities used in today's energy production.

3.1 ENTSO-E

European Network of Transmission System Operators for Electricity (ENTSO-E) [9] was established in December 2008 to unify until then separated organisation and started operating with 2009. It currently represents 43 electricity transmission system operators (TSOs) from 36 countries across Europe. Its goal is to support the implementation of Europe's energy plan which consists of:

- **climate policy** - integration of renewable energy sources such as wind and solar power,

- **single energy market** - provide better collaboration between member states to improve affordability, sustainability and security of energy supply,

- **single system** - a single focal point for all technical, market and policy issue relating to European energy network for all participating players.

3.1.1 Data

In 2013 a new regulation was accepted which mandates all member states to submit fundamental information related to electricity generation and type, load, transmission and balancing starting with January 2015. This data has been publicly available since this date. Before it, available data and the number of available attributes varies by country.

Trading data is due to its public nature available in aggregated forms by the hour or quarter of an hour, which varies by country. High-resolution trading data is available for a fee on European Energy Exchange (EEX) [36].

For each region, different properties are available:

- **Load** - energy distributors are required to provide estimated load capacities for future energy use. These forecasts are available in aggregated forms for a day, week, month and a year ahead.

- **Generation** - energy generators must report energy production capacity and generated electricity per production type (wind, solar, fossil, nuclear etc.), amount of stored in water reservoirs and day-ahead generation forecasts.

- **Transmission and congestion management** - the amount of energy transferred between different regions, transmission capacity forecasts and overloads, congestions, with their resolutions in the network

- **Outages** - all planned and forced outages at different units in the network due to maintenance, upgrades or faults. These units are composed

of transmission networks such as power lines, generators such as wind farms and large consumer units such as factories.

Provided data is available on their FTP server which we may access using website credentials. Each subset is represented with its directory where we may find CSV files for each month. Inside the file, data is unsorted and split by different bidding regions. Map of regions is shown in Figure 3.1.

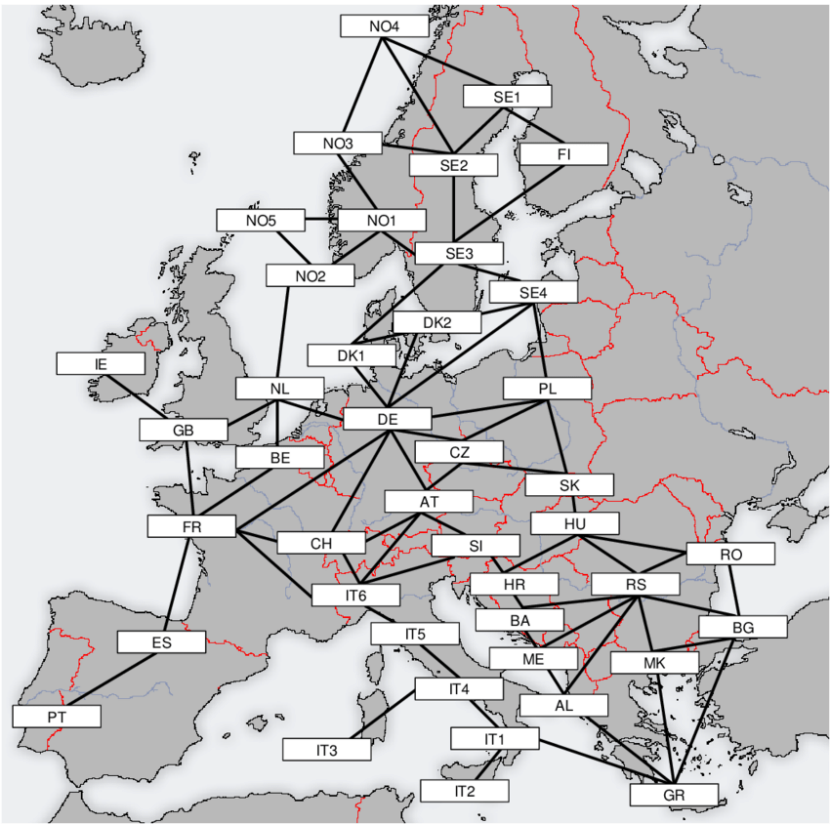

Figure 3.1: Map of trading regions in Europe, generated by Vlachos et al. [37].

Each file follows the same structure as shown in Table 3.1 for the target attribute DayAheadPrices. Left of the vertical line are columns shared across all reported datasets and right are dataset specific attributes. A file may contain several entries for the same area name and timestamp with different values, in such cases, the last entry is a valid one.

Table 3.1: An example of a file structure in ENTSO-E dataset. Vertical line denotes columns shared across all datasets and right variable attributes. Example is taken from the forecasting variable dataset.

year	month	day	DateTime	AreaCode	AreaName	MapCode	Currency	Price
2015	4	25	2015-04-25 23:00	BZN	Litgrid BZ	LT	EUR	24.00
2015	4	21	2015-04-21 05:00	BZN	Litgrid BZ	LT	EUR	48.01
2015	4	26	2015-04-26 05:00	BZN	Litgrid BZ	LT	EUR	19.17
2015	4	18	2015-04-18 05:00	BZN	Litgrid BZ	LT	EUR	25.50
2015	4	14	2015-04-14 14:00	BZN	NO2 BZ	NO2	EUR	24.76
2015	4	20	2015-04-20 14:00	BZN	NO2 BZ	NO2	EUR	25.81
2015	4	10	2015-04-10 08:00	BZN	REN BZ	PT	EUR	61.69
				...				

3.2 European Centre for Medium-Range Weather Forecasts

ECMWF [38] is the European Centre for Medium-Range Weather Forecasts and are responsible for producing numerical weather forecasts [39] and research. They also maintain an archive of meteorological data for the European region both historical actual data and historical forecasts.

We are interested in the historical weather forecasts since we are trying to simulate a real environment at a certain point in time. Having accurate weather information would not be realistic and might cause sub-optimal results since other market players would also be working with forecasted data. The forecasting dataset is known as ERA-Interim [40].

ECMWF provides data publicly accessible through their APIs. We access those using Meteorological Archival and Retrieval System (MARS) requests. They can be created through their website; however they provide limited access to available MARS properties. An alternative way is a programmatic method where we have full access. MARS only specifies the format of the requests. The structure is fairly simple. Unintended lines represent verbs with requests operation such as `retrieve` followed by lines intended by a single space following `property=value` format. The request for ten day forecast for European region is as follows:

```
retrieve,
class=ei,                          # dataset
dataset=interim,                   # subset
date=2014-12-01/to/2018-12-31,     # time range
levtype=sfc,                       # surface data
expver=1,                          # version of the dataset
grid=2/2,                          # grid resolution in degrees
param=165.128/41.128,              # parameters separated by /
step=24/to/240/by/24,              # forecast steps
# top left and bottom right corners in lat. and lon.
area=75/-15/30/42.5,
type=fc,                           # forecast
target="eu_10d.grib"               # destination file
```

Once the request is made, a job is created on the remote server which generates the data files. After a couple of hours, the files are generated and automatically downloaded.

We downloaded the following weather information:

- U and V components of wind speed,
- cloud clover,
- temperature,
- surface UV radiation,
- surface solar radiation,
- surface thermal radiation,
- total precipitation and
- temperature.

The retrieved data is stored in the Gridded Binary (GRIB) format. The format is standardised by the World Meteorological Organization's and is therefore widely used. ECMWF provides a set of command-line tools to

Table 3.2: File obtained with grib_ls command. Columns edition, packagingType and gridType were omited due to a constant value for readability.

centre	typeOfLevel	level	dataDate	stepRange	dataType	shortName	value
ecmf	surface	0	20141201	24	fc	10u	-1.34552
ecmf	surface	0	20141201	24	fc	10v	2.60595
ecmf	surface	0	20141201	24	fc	uvb	372992
ecmf	surface	0	20141201	24	fc	ssrd	1.25389e+06
ecmf	surface	0	20141201	24	fc	strd	2.24908e+07
				...			

convert the files between formats. We used `grib_ls` tool to extract data at certain coordinates into a CSV file for easier processing:

```
grib_ls -l 58.499762,14.313914,1 interim_long.grib > w_SE4.csv
```

The format structure is similar to the one encountered with ENTSO-E which has a property name and value columns instead of each property its column. An example of extracted data can be seen in Table 3.2.

3.3 Other data sources

Half of the produced energy in the European Union comes from fossil fuels. We, therefore, include additional prices for four additional commodities frequently used in the energy markets. Those are coal, Brent, a type of crude oil extracted in the North Sea, gas prices and European CO2 emission allowances, which were introduced as a tax on CO2 emissions from large industry and energy business.

All named commodities are taken from the BusinessInsider [41] which provides historical data for multiple trading instruments on a daily level. All commodities contain the same attributes, date, price of the commodity, which is equivalent to the closing price, open price, daily high and low, and daily change in percentage.

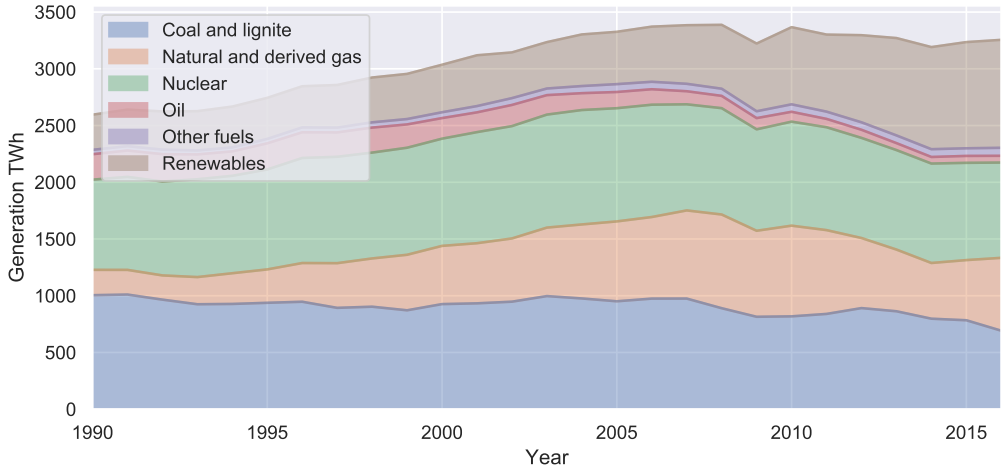

Figure 3.2: Chart of main energy sources in European Union from 1996 to 2016 [42].

Chapter 4

Methodology

In this chapter we describe the steps taken from acquired data to obtaining the final test results. We use different pipelines for univariate statistical models and more complex ones for machine learning methods which can be outlined by the following steps:

1. **Data acquisition**: the selected data sources are downloaded as described in Chapter 3.

2. **Data preparation**: data is checked for missing values, number of features and their distributions. Lastly, different datasets are generated.

3. **Feature generation**: primary features might not be informative enough, so we one-hot encode date components, add lagged values and a couple of technical indicators.

4. **Feature selection**: very large number of features normally leads to poor results as it is harder to pick relevant components. We use a filtering algorithm to keep the most relevant.

5. **Parameter selection**: Statistical and machine learning methods provide several parameters to fine-tune their behaviour. We use randomized CV search across parameter space to pick them.

6. **Obtaining the final results**: We run the models with selected features and parameters on the test set and report the results.

4.1 Data preparation

We start by manually inspecting the available data in the primary data source. The target variable is found in the DayAheadPrice subset. Visual graphic of the target series is shown in Figure 4.1.

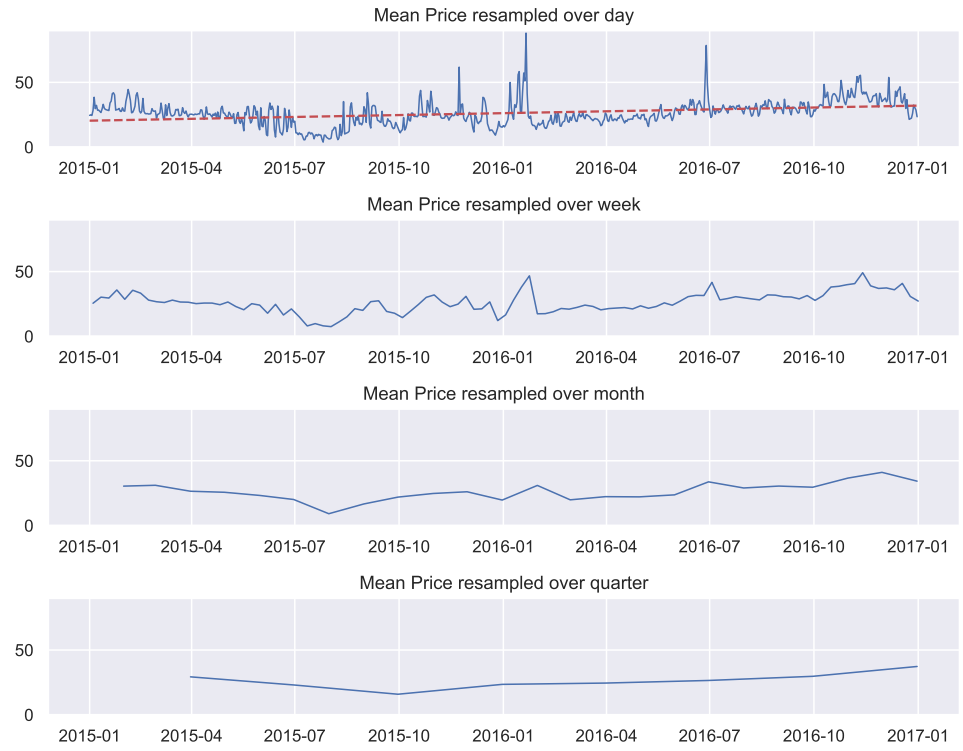

Figure 4.1: Energy price in SE4 region between 2015 and 2017 with resampled mean values over day, week, month and quarter of the year.

We first split the time series into train, test and validation sets. The exact split and number of samples is described in Table 4.1. The time ranges and the test regions are taken the same as in comparison performed by Lago et al. [6] for comparable results.

Table 4.1: Split datasets and their properties.

Dataset	Region	Start date	End date	No. samples
train	SE4	2015-01-01	2015-12-31	8760
validation	SE4	2016-01-01	2016-12-31	8760
train-test	BE	2015-01-01	2015-12-31	8760
test	BE	2016-01-01	2016-12-31	8760

There are no missing samples for this dataset. We continue with the analysis of the training dataset, where we first investigate the value distribution using a violin plot in Figure 4.2, where we can observe the distribution of the values per month, its mean and standard deviation.

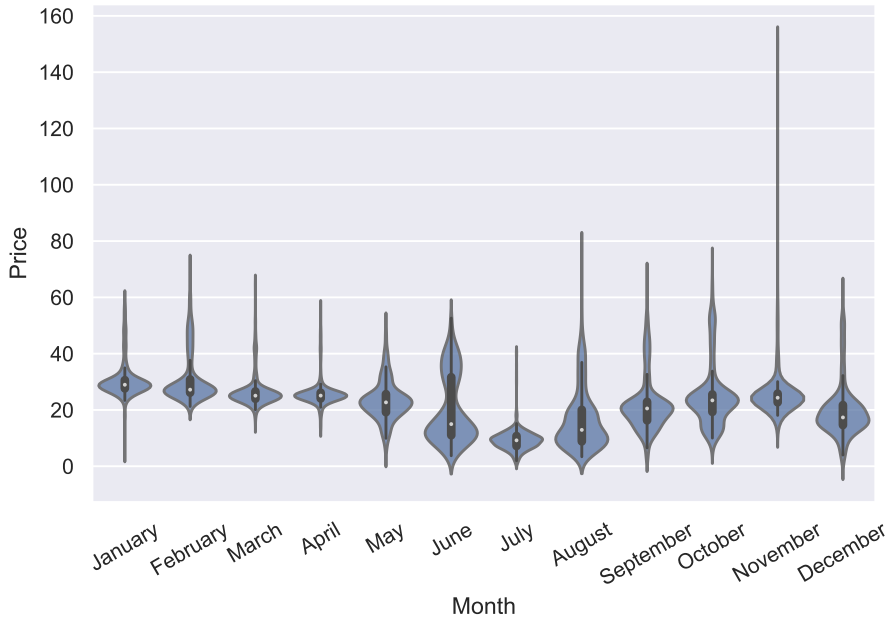

Figure 4.2: Violin plot of a monthly price distribution, mean and standard deviation. Data is in most cases evenly normally distributed, with some exceptions such as June and August. November in our case has at least one outlier with very high price.

We inspect other sources and based on the relevance we decide to remove

or to keep them. A list of kept datasets is available in Appendix A. All of these datasets follow the same format as described in 3.1.1. We filter the data based on the desired country and keep only DateTime attribute and those specific to the dataset. If there are multiple attributes in a single column, such as energy generation by type, we perform pivot operation to transform them into separate columns. We continue by dropping the duplicates using the keep last method and sort the data based on date and time. As the last step, we remove the features that contain less than 3 unique values and have less than 4% of values (meaning that value is not reported daily). Both conditions are selected without experimentation.

Pivot operation is also applied on the ECMWF weather dataset as it has a very similar structure. Wind speed information is broken into U and V components, wind speed from east to west and south to north. We transform them back using the Euclidean distance:

$$ws = \sqrt{u^2 + v^2} \tag{4.1}$$

Pivot columns on the forecast dataset are the `stepAhead` and the `name` attributes. The first represent how far in the future a forecast is made and the second is the components name. With this we add forecasted weather at the predicting time point.

Next step is the fusion of the selected datasets. The first level fusion is done by the DateTime key. The primary dataset has an hourly resolution while the weather dataset has three-hour steps for current weather and the weather forecasts. If no matching key is found, the nearest match is taken. Second level fusion, where we would forecast other attributes are already provided in most cases. Second level fusion is skipped, as it would be forecasts of additional variables, which are already provided with the datasets.

After fusion, missing values are imputed using linear interpolation with respect to the date and time of the sample.

The final set of datasets and number of features without the target variable and date components is listed in Table 4.2.

Table 4.2: Generated datasets and accompanying number of features without the target variable and DateTime component.

Dataset	No. features
Single	1
Weather	99
ENTSO-E	38
All	140

4.2 Statistical methods

All statistical methods described in Section 2.2.1 are dependent only on the target variable except VAR. We can, therefore, skip several steps from the machine learning pipeline. We keep the data analysis step to see if the series is stationary or not and the parameters selection. We first plot the auto-correlation (ACF) and partial auto-correlation plots, shown in Figure 4.3. From those we estimate the p and q parameters. The ACF plot shows that a lag value of approximately 24 seems the most appropriate. We check the PACF to verify if these lags are caused by the lag propagation. We decide to reject this assumption since there is more than one significant lag after differencing. The auto-correlations are decaying rather slowly, meaning that the parameter p will have a much larger effect, while parameter q for the MA model will have a very small value or even zero.

4.2.1 Stationarity test

We perform the stationarity test on both SE4 and the BE target series from the training dataset to see if any trend that should be removed exists.

As we can observe in Figure 4.1 the trend line over the train dataset is travelling slightly upwards and continues in such a manner to the validation portion, meaning that the series is probably not stationary. Looks can sometimes be deceiving so we perform the augmented Dickey-Fuller test to eliminate human bias.

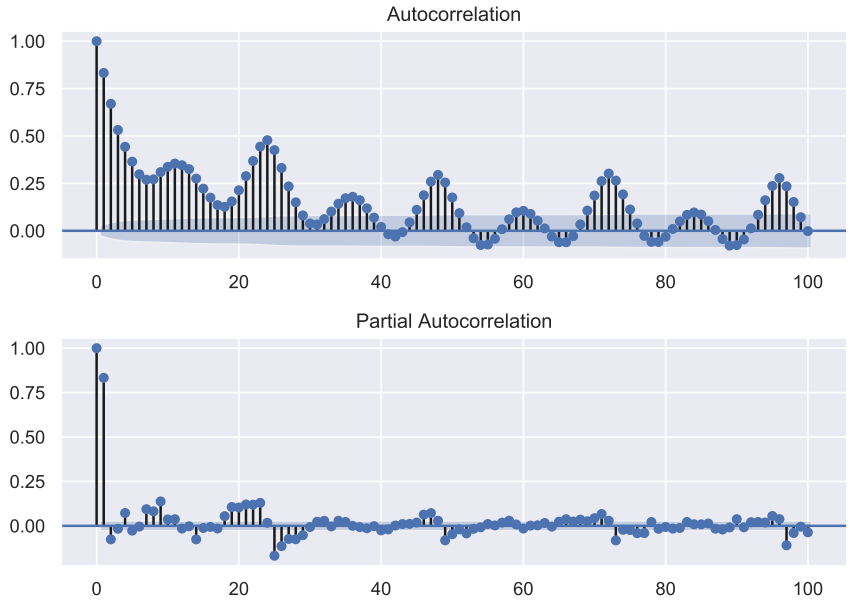

Figure 4.3: Autocorrelation and Partial autocorrelation plots for energy price taken from the Belgium train dataset.

The test for both regions shows that we can reject the null hypothesis since the test statistic is much smaller than the 1% critical value, meaning the time-series is stationary. The calculated ADF values are in Table 4.3.

The Belgium region has based on the ADF test smaller trend. Visual inspection confirms it, shown in Figure 4.4.

Table 4.3: Values obtained with augmented Dickey-Fuller test for Belgium region on train dataset. The test shows that the series is stationary.

Parameter	Values (SE)	Values (BE)
test statistic	-7.170	-9.808
p-value	2.81×10^{-10}	5.75×10^{-17}
critical value (10%)	-3.431	-3.431
critical value (5%)	-2.861	-2.861
critical value (1%)	-2.566	-2.566

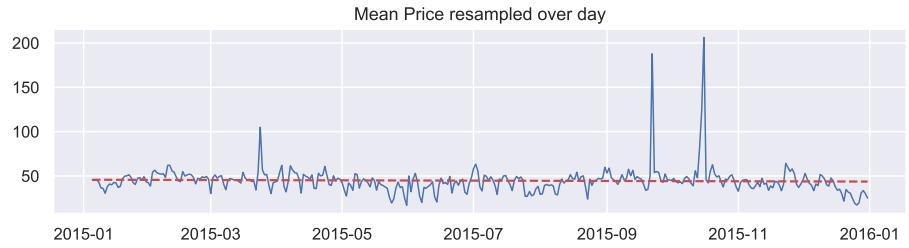

Figure 4.4: Belgium energy price taken from the train dataset.

4.2.2 Parameter selection

Both statistical and machine learning methods provide different parameters to help fine-tune the algorithms. For statistical methods, we use Akaike information criterion function (AIC) [43], which tells us how much information is lost when representing the model. It is described by the equation:

$$AIC = 2k - 2ln(\hat{L})q \tag{4.2}$$

In the equation, k represents a number of internal parameters and the second term gives us the likelihood score; this tells us how well the data fits the model. It is an optimisation function which minimises information loss and with it maximises the fit of the model. Large number of parameters will be penalised to discourage overfitting. The function comes from the field of information theory. Smaller score value within the same k represent a better result.

We iteratively check a range of parameters on the test set and pick the ones with the lowest AIC score.

4.3 Feature generation

It often happens that input features might be too complex for the algorithms to successfully connect their relationship with the input or the relevant information is in the past, not available for the current sample. The goal of feature engineering is to generate new features that connect inputs to outputs

and resolve mentioned issues.

With time series, one of the most important features are lagged values. Majority of statistical algorithms as described in Section 2.2.1 are designed in such a way to use these values behind the scenes. Machine learning algorithms, in contrast, are not. We generate lagged values for all past 24 to 48 hours from the current time point, and then up to a week so that the values are always at the same hour as the forecasting time point. We perform this step for all series after fusing them by date key, omitting future weather forecasts.

Date is another complex feature which may carry relevant information. We break it into its basic components of month, day and day of the week. Year is skipped as it does not carry any relevant information.

With linear models some change in feature value normally represents some increase or decrease in the output. However, some features may not follow this rule and despite being relevant, models might completely ignore them. Month could be such an example. Winter months appear at the start and the end while summer appears in the middle. If there is any pattern dependant on seasons it might therefore be missed. To solve this, we encode day of the week and month feature with one-hot encoding. Each unique value is mapped to a new binary feature as shown in Table 4.4.

Table 4.4: An example of one-hot encoding for day of week.

Day	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Monday	**1**	0	0	0	0	0	0
Tuesday	0	**1**	0	0	0	0	0
Wednesday	0	0	**1**	0	0	0	0
Monday	**1**	0	0	0	0	0	0
Sunday	0	0	0	0	0	0	**1**

As the last step, we include a couple of widely used technical indicators on the forecasting variable. We add a simple moving average (SMA) and standard deviation (STD) for 30-day, 7-day and 24-hour windows. Window size tells how many values are taken into calculation. Based on SMA and STD we calculate (4.3) Bollinger Band [44] with window sizes n as described

and deviation parameter m of 2. It is a widely known technical indicator which adds a lower and an upper band based on volatility. Based on this, certain patterns such as breakouts and squeeze may be extracted.

$$\begin{aligned} b_{middle} &= SMA_n \\ b_{lower} &= SMA_n - (STD_n * m) \\ b_{upper} &= SMA_n + (STD_n * m) \end{aligned} \tag{4.3}$$

Final datasets with added and filtered features are shown in Table 4.5.

Table 4.5: Generated datasets and accompanying number of features without the target variable and DateTime component.

Dataset	No. features	With feature generation
Single	1	80
Weather	99	346
ENTSO-E	38	454
All	140	787

4.4 Feature selection

A large number of attributes may negatively impact the performance of models. It may lead to poorer results as a lot of noisy features may mislead learning algorithms to make forecasts based on noise. Removing such attributes means we may improve the accuracy of the models as well as reduce training time due to requiring fewer data points as an algorithm input.

The selection algorithms are split into three main groups. Most often, the better-performing ones are wrapper methods, for example, forward/backward feature elimination [45], which selects a feature set by adding or removing features from the selected set. After each change, the model is tested to see the impact and repeat the process until convergence. They perform well, but due to combinatorial complexity are suitable only for a smaller number of features. The second group of methods are embedded which try to learn the

best feature set that is later used for training. Representative example here are LASSO [46] and genetic algorithms [47]. They use some metric to help methods eliminate irrelevant features. The last group are filtering methods. Here, some function is used to calculate which features to remove or how to rank them.

We skip the embedded methods as several models do not support them. We try LASSO and genetic algorithm for feature selection as well as filtering method with f-regression function. The function first calculates the correlation between each feature and the target based on equation:

$$corr = \frac{(x_i - \mu_{x_i}) * (y - \mu_y)}{\sigma_x * \sigma_y} \tag{4.4}$$

with μ representing variable mean, $sigma$ std. deviation, y target variable and x_i a feature. The value is then converted to F-value and p-value indicating relevance. With LASSO and filtering methods we test all learning methods from 1 to 100 best features on the validation set and check the sMAPE. With genetic algorithms we try different mutation and crossover probabilities, tournament size and population sizes. We use the LR model due to its very fast training time.

The best performing method despite its simplicity is the filtering method. We repeat this step with normalized feature values to see if we obtain better result on the validation set. Values in its original range perform better so we keep those.

Once we rank the features using the filtering method, we select k most suitable ones. The selection process first fits $1, 2, ..., n$ features on the train dataset and calculates their score on the validation set. Once we obtain the score, we manually inspect the numbers and make a choice. The manual selection process can be described using Algorithm 1. We prefer fewer features despite a slightly larger error to prevent overfitting so we therefore introduce a small tolerance into the equation.

A set of results for linear regression is shown in Figure 4.5. Here, we can observe quite large error rates and unsteady changes for datasets without

Algorithm 1 Pseudo code of the feature selection algorithm performed manually.

```
MaxFeatures ← 120
BestScore ← inf
Tol ← 0.01
N ← inf
for i ← 1 to MaxFeatures do
    Score ← evaluate model with i features
    if Score + Tol < BestScore then
        BestScore ← Score
        N ← i
    end if
end for
N ← optimal number of features
```

generated features. On the contrary, datasets with added features achieve much smaller errors and behave as expected. First, we observe a fast decrease and afterwards a slow increase as the model most likely overfits.

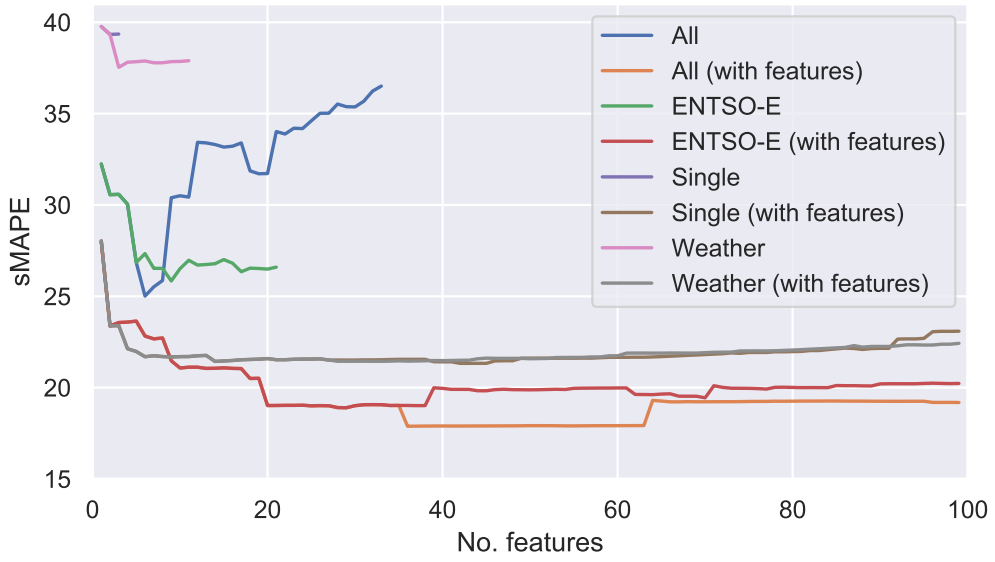

Figure 4.5: Plot of errors w.r.t. number of features for linear regression.

At this point we also remove all technical indicators. As we notice that they are very frequently selected due to high correlation but manual tests show that they cause additional error.

4.5 Hyper-parameter optimization

A typical machine learning algorithm has several different parameters which control how the learning is performed. These need to be manually selected for optimal performance of the algorithm for the selected set.

We use randomized hyper-parameter search over the selected methods. We provide a search algorithm with a method and a list of parameters. An example of parameter space for the k-NN regressor is as follows:

```
'kNN': {
  'model': KNeighborsRegressor,
  'params': {
    'n_neighbors': [1, 2, 3, 5, 10, 20],
    'metric': ['cosine', 'euclidean', 'l1', 'l2', 'manhattan']
  }
}
```

The algorithm samples a random set of parameters, trains the model and evaluates it on the validation dataset. The best parameter set is used for the final test. If the best value for a specific parameter is deemed as an upper or lower range we specify. We increased the range, if we consider it would not contribute to overfitting.

4.5.1 Neural networks

Neural networks have both variable number of feature inputs and internal parameters presented as neurons in an arbitrary number of layers. We therefore manually experimented with different neural network architectures by varying their number of layers and neurons per layer together with different input features. Features are selected the same by top k based on the filtering function.

Inputs to neural network work are normalized to be between zero and one as that experimentally provided better results. All models are single layered, adding multiple layers did not improve the results. Models without additional feature have 30 neurons while with 120. We use ReLU as activation function

and ADAM optimiser [48] to update the weights.

4.6 Evaluation process

In the following subsection we describe the prequential evaluation protocol for simulating the time series flow. In the second subsection we describe metrics used for reporting the results and comparing them.

4.6.1 Prequential evaluation

In data stream mining, there is very little or no data available for initial learning. Data is streamed in through time which is in contrast to classical machine learning techniques based on batch learning. To address this problem and simulate real-world scenarios, the prequential evaluation [49] is used. The word prequential comes from combination of words "predictive" and "sequential".

With the prequential method we simulate the data stream flow by sending incoming samples x to our machine learning algorithms and predict for each one to obtain the forecast denoted as $\hat{y}$. During the learning processes, we remember the pair $(x, \hat{y})$. Later in the future, once the ground truth y becomes known, we use the triplet $(x, \hat{y}, y)$ for evaluating predictive accuracy of our model. A figure of the simulation is in Figure 4.6.

This method tends to be pessimistic during early stages due to the lack of training samples.

For evaluation of batch algorithms, we retrain the models every beginning of the month incremental learners update daily. First they are retrained and second partially updated. With retraining of batch models we hopefully also account for the concept drift in the series. Statistical methods are an exception. They are updated every iteration as they require the latest data and do not update the internal model parameters.

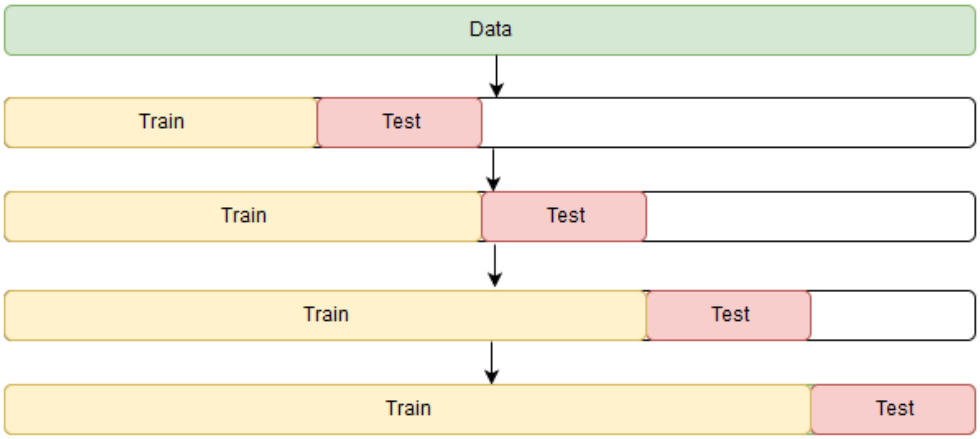

Figure 4.6: A visual representation of batched prequential method.

4.6.2 Metrics

In this section we describe three different metrics used for reporting the results for a single method and the methods between each other.

Root-mean-squared error

Root-mean-squared error [50] (RMSE), equation (4.5) is the standard deviation of the residuals (forecasting errors) with the same unit as the forecasting data. RMSE is the measure most commonly used as a loss function in machine learning. Value zero represents a perfect match between actual and forecasted values and is always non-negative.

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n} (y_i - \hat{y}_i)^2}{N}} \tag{4.5}$$

Symmetric mean absolute percentage error

Symmetric mean absolute percentage error (sMAPE) [51] is a measure of prediction accuracy used for forecasting methods in statistics. It is an improved version of MAPE [52], as it addresses the problem of overcasts and undercasts being treated differently [53], as well as it addresses small denom-

inators. It is commonly used due to simple interpretation as the resulting value represents percentage error. It is described by the equation (4.6). A value closer to zero, similar to RMSE, indicates a better score, whereas a value zero means no errors.

$$sMAPE = \frac{100}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2} \tag{4.6}$$

Q_i statistic

We use Q_i statistic suggested by João Gama [54] to compare two different methods. It captures changes in relative performance of the two learning algorithms. In the equation (4.7) S is the accumulated loss to a certain point for each method. We use a log-ratio between the obtained values to get the relative performance. The logarithm gives value a positive sign when method A performs better than B and negative when method B performs better than A.

$$Q_i(A, B) = log\left(\frac{S_i^A}{S_i^B}\right). \tag{4.7}$$

We use an improved version using fading factors (4.8). The initial version tends to be heavily impacted by the history of the model. It is therefore gradually decreased by some factor α, normally with a value close to 1, for example, 0.995. ϵ_i indicates the error at a specific point in time.

$$Q_i^\alpha(A, B) = log\left(\frac{\epsilon_i^A + \alpha \times S_{i-1}^A}{\epsilon_i^B + \alpha \times S_{i-1}^B}\right) \tag{4.8}$$

The result of this metric is a plot from which we can see at which point in time certain method performed better than the other.

Chapter 5

Results

We perform three sets of experiments with and without additional features and late integration of the best performing models. The first experiment is performed to see the impact of each additional dataset, as majority of the features they provide are removed in the feature selection step, where lagged values have higher correlation. In the second test, we include those features to see a more realistic result of how the models perform. The results were obtained following the steps described in the Chapter 4. Final obtained feature sets and hyper-parameters were tested trained on the Belgium training set and reported based on the Belgium test set.

5.1 Baseline

To evaluate if the models learn anything we introduce a simple baseline method. We define it as the lagged value with the lowest sMAPE error rate. It is determined on the Belgium training set. Chart of the sMAPE and MSE is shown in Figure 5.1. All lagged values between a day and two weeks (number of lags between 24 and 336). On the train set, the best performing value is 24 hours behind with the value of 22.111 and the second-best at a week with an error of 23.597.

In the test set the best pick turns out to be a weekly (-168 hours) de-

Figure 5.1: sMAPE and RMSE for lagged values between a day and two weeks.

lay with sMAPE of 22.016. A day of delay scores 22.963 which we use for reporting with the other methods.

5.2 Results without additional features

Generated features have a higher correlation and were chosen over the features from additional datasets. We therefore first performed experiments without generated features to see if the additional dataset had any influence. Results are shown in Table 5.1.

As we can quickly notice the results are discouraging. None of the machine learning methods managed to outperform our baseline method. Statistical methods are the only ones that manage to beat the average as they generate lagged values behind the scenes. These turn out to be extremely important while doing the time-series forecasts. AR and VAR methods are not capable of using multiple features and therefore perform forecasting only on the target

Table 5.1: Error rates for different regressors obtained on test set without lagged values.

	Single		Weather		ENTSO-E		All	
	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE
AR	**18.436**	17.932	18.436	17.932	18.436	17.932	18.436	17.932
VFDT	20.976	17.479	20.986	17.399	**20.261**	16.493	21.314	21.391
VAR	23.159	18.885	22.229	18.292	**21.835**	18.808	22.579	23.368
Baseline	**22.963**	18.521	22.963	18.521	22.963	18.521	22.963	18.521
ANN	39.777	23.173	40.117	23.713	**25.623**	25.796	25.796	22.817
LR	41.261	23.683	40.100	22.945	28.537	17.606	**26.372**	17.428
RF	38.163	23.547	37.706	22.684	27.002	19.856	**26.973**	19.736
DT	41.236	23.008	41.510	23.068	**27.960**	20.767	29.184	20.460
KNN	40.812	23.963	41.345	22.869	30.300	20.360	**28.972**	19.614

variable so the result is always the same. As the preliminary analysis showed, AR parameters are much more important than MA. We also tested ARMA and ARIMA model, but the differencing factor and MA caused even worse performance with sMAPE over 30.

VFDT performs remarkably well compared to its counterparts. It is the only learning model that manages to beat the baseline. This is most likely due to its frequent updates and ability to update internal statistics.

Most of the models have the same selected features for the ENTSO-E and All datasets, the difference in the result is mostly due to a different random state.

If we compare the same learners with different datasets we can notice that weather dataset contributed a negligible amount compared to a single dataset which contains the only day of the week, hour and month information. The other two, all and ENTSO-E managed to significantly improve the result in several cases, however, baseline still outperforms both of them. Due to suboptimal results, we did not compare the models further.

5.3 Results with additional features

In the second set of experiments, we include the additional generated features. The results are shown in Table 5.2.

Table 5.2: Error rates for different regressors obtained on test set with lagged values.

	Single		Weather		ENTSO-E		All	
	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE	sMAPE	RMSE
ANN	15.963	14.020	15.873	13.987	15.024	11.934	**14.932**	11.907
RF	17.919	15.997	17.643	16.337	16.540	16.180	**16.381**	15.888
VFDT	19.879	22.660	20.002	23.055	**17.793**	17.770	17.795	17.298
AR	**18.436**	17.932	18.436	17.932	18.436	17.932	18.436	17.932
DT	**19.281**	16.855	19.837	22.804	22.427	16.855	22.427	19.486
k-NN	**19.615**	17.592	19.615	17.592	19.615	17.592	19.615	17.592
LR	21.733	16.038	21.733	16.038	**19.781**	14.785	19.801	14.769
VAR	23.159	18.885	22.229	18.292	**21.835**	18.808	22.579	23.368
Baseline	**22.963**	18.521	22.963	18.521	22.963	18.521	22.963	18.521

Statistical models and baseline have the same results as in the previous experiment. VAR model was excluded from additional features as it generated lagged values internally. Machine learning methods managed to increase, only lagged values still outperformed the rest and in two cases ENTSO-E performed the best.

ANN performed the best, with additional features from the ENTSO-E. VFDT, compared to RF which is an ensemble of trees performed very well and also compared to its batch counterpart DT, which scored much lower. For VFDT, the cause of success is most probably similar in the first experiment, it was able to adapt to changes much faster than the rest. Feature selection for k-NN selected only three features, price lagged values with the delay of 24, 25 and 168 hours and managed to surpass LR. Order of the models is quite expected as the error rate increases with the simplicity of the models.

5.3.1 Feature importance

We extract features from the linear regression of the ENTSO-E model with additional features which performed the best among the datasets. The model is one of the simplest and offers a good insight into the individual feature relevance trough θ parameters. The model also performed quite good compared to the others. Feature importance is shown in a plot in Figure 5.2.

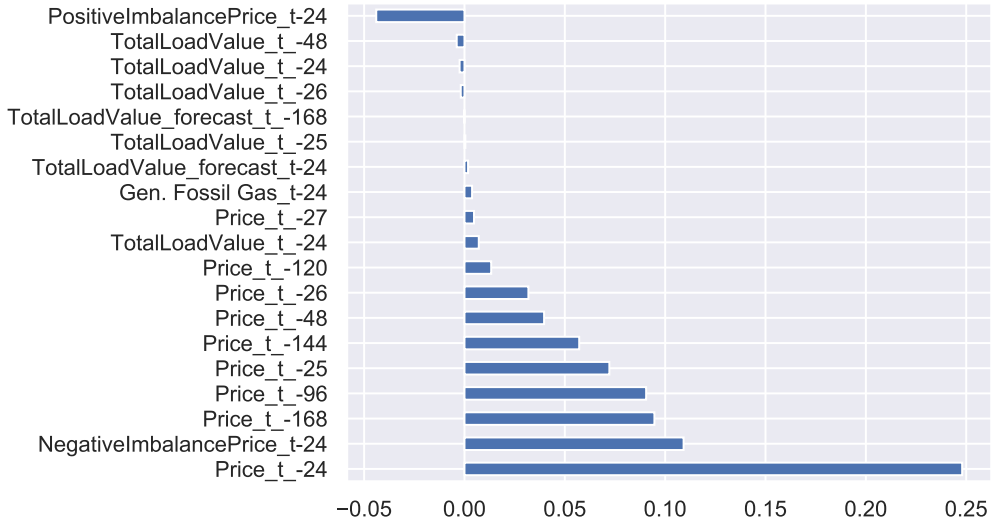

Figure 5.2: Feature coefficients for the best performing linear regression model.

From the plot we can see that 9 out of 19 features are lagged values of the target variable all with the highest importance. Seven variables are related to the past total load of the energy grid and its forecast. Two of the remaining three features are positive and negative price imbalance. It is a provided calculated value and represents a connection between generation and consumption. The last feature with very small importance is the energy generation from gas-based power plants. From the plot, negative imbalance contributed the most compared to the single dataset having only lagged values.

Other models have due to a correlation-based filtering method very similar

feature set, the difference is only in the number of selected features, where methods that are best performing on a single dataset have just a few selected features, mostly Price_t_-24 to 27, 48, a weekly delay and perhaps a couple of others.

5.3.2 Ensemble regression

The last experiment we perform is the late integration where the results of the best models are put together. This step is done for the best performing regressors across all experiments.

We used weighted averaging where the coefficients are calculated using linear regression. With separate coefficients the models might get negative value to complement other method mistakes and a different value capture the model relevance based on its performance. Model was retrained once per month same as other batch models to update the weights together with other model changes.

The sMAPE value obtained using late integration is 13.084% and is shown in Table 5.3 together with other best performing models.

Table 5.3: Lowest error for each model with information on which dataset it was obtained.

Rank	Model	Dataset	With features	sMAPE	RMSE
1	**Consensus**	/	/	**13.084**	12.084
2	ANN	All	Y	14.932	**11.907**
3	RF	All	Y	16.381	15.888
4	VFDT	ENTSO-E	Y	17.793	17.770
5	AR	/	/	18.436	17.932
6	DT	/	Y	19.281	16.855
7	k-NN	/	Y	19.615	17.592
8	LR	ENTSO-E	Y	19.781	14.785
9	VAR	ENTSO-E	/	21.835	18.808
10	Baseline	/	/	22.963	18.885

The best performing consensus model contains all regressors except VAR and VFDT, which had a factor of almost zero. VFDT was most likely removed because of its constant changes due to frequent updates. It was there-

fore unable to relate it with any of the other models. Removing any other model has a negative impact on both of the selected metrics. The final weights for the consensus model are in Figure 5.3.

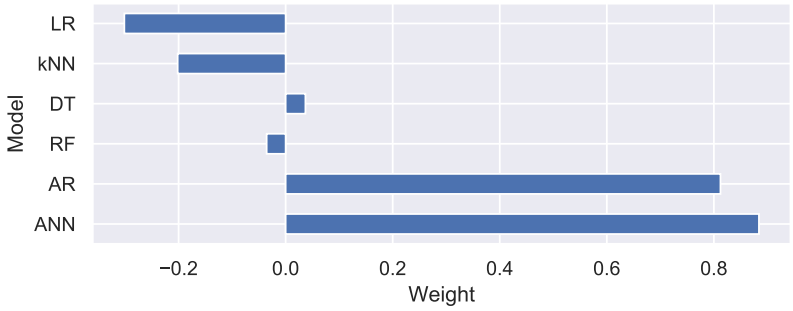

Figure 5.3: Weights for the regressors in the consensus model. We can see that the ANN and AR have the highest values, while linear regression and k-NN negative that complement the errors made by ANN and AR.

5.3.3 Errors

We plot three different charts to investigate where the errors occur and gain additional information for potential future work. For each chart, we include the five best performing models and our baseline. We begin with average sMAPE error by month, shown in Figure 5.4.

Winter months have a larger error rate. This is most likely caused by shorter days and additional energy requirements for heating due to lower temperature. We hoped to improve this by adding additional weather information, but it did not help. The first spike all models have starts in April and peaks in May, where uncertainty increases with the change of average load. Summer months are very stable for all predicting models. We assume that this is due to longer days, where morning and afternoon spikes are not as intensive as during the winter months. AR model performs well during the first winter months, but significantly rises towards the end of the year, which also has a negative impact on the consensus. ANN therefore surpasses consensus model. This could be resolved with a limited window of past ob-

servations to calculate the relevance of individual models instead of the entire history.

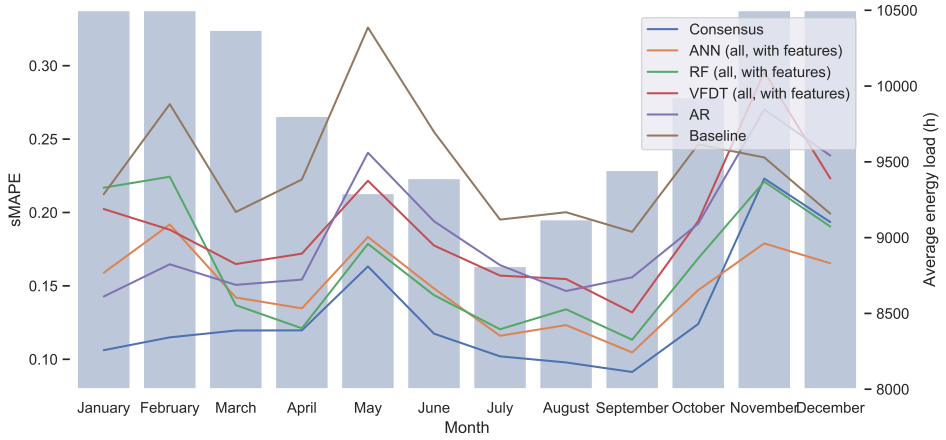

Figure 5.4: Error for the best performing models by month. We can see that transitional periods in April and September cause the biggest increase in the error and alter stabilises when there more predictable power load.

We continue with the daily average error as shown in Figure 5.5. We can notice that the transition between workdays and weekends causes the biggest error. It is most likely caused due to the dependence on the lagged values of a target variable as there are large changes and sudden differences in correlations. Baseline suffers the most and confirms our observation. Mid-week errors are the lowest and surprisingly in certain cases, our baseline outperforms most trained models.

The last analysis we performed was the average hourly error in Figure 5.6. Here, two peaks are the most noticeable. First with a peak at five in the morning and second around two in the afternoon. These peaks start similarly as the daily error rate during the weekend, that is during the transitions. These are also explained by mornings when people wake up and go to work and when they get back home.

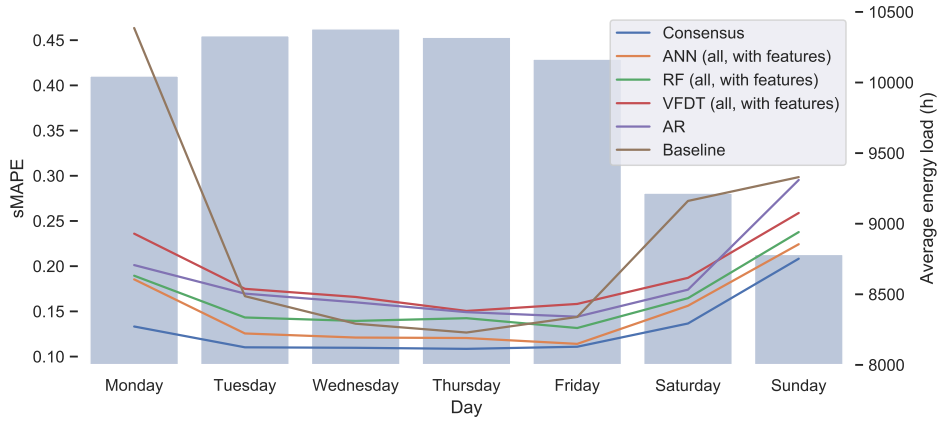

Figure 5.5: Error for the best performing models by day. We can see a similar pattern as before, that the volatility increases during transitional hours from home to work and vice versa.

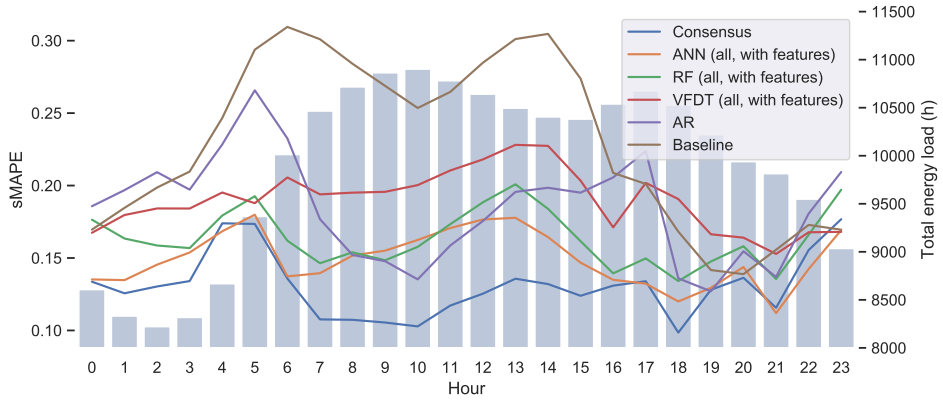

Figure 5.6: Error for the best performing models by hour, where the error increases together with the change in average hourly energy load.

5.4 Relative performance

We compare the best performing batch model, ANN, with an incremental learner VFDT using Q_i statistic with $\alpha = 0.995$ to determine the relative performance and to see if incremental learning helped. The obtained results are similar to the error plot by month but show more granular information. The Q_i statistic is visible in Figure 5.7.

As previously established, the plot shows that the ANN model performs consistently better, but it also carries the additional information of showing us the difference between the two approaches and at which points it is beneficial for ANN. These points occur periodically at the start of each month which might indicate that the periodical retraining of the models has helped boost overall accuracy and should perhaps be done more frequently.

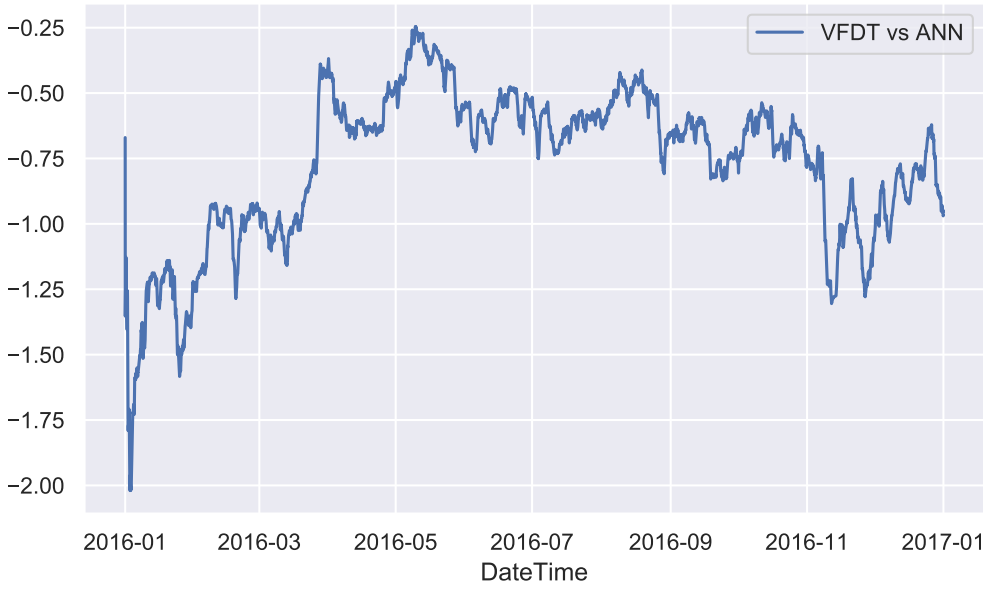

Figure 5.7: Q_i statistic for VFDT and ANN.

Lago et al. [6] performed a similar comparison with the same test set, using only the ENTSO-E dataset. The common models are only AR, RF and ANN. The AR model in their test achieved an sMAPE error of 19.31 while ours 18.44. Inspecting the parameters, they used similar lag value as our models, but dropped lagged values with small coefficients to zero to ignore them.

Their RF scored 15.39 compared to our 16.38. They reported using 24 lagged values of the previous day and 24 for one week before with additional lagged values of load forecasts and electricity prices in France. They did not specify the selected features after feature selection nor their importance.

While we included prices from France, they were ranked quite low at the feature selection step and thus removed. This is most likely the bias that hurt more complex models, such as RF. They used a very similar feature set for their ANN as for their RF, which scored 13.27 compared to ours 14.93.

One major difference between compared models is that they used additional historical price data directly from Belgium energy provider from the year 2010 onwards, not publicly available any more. This is a very probable cause for the large discrepancies between our models.

Their best model, a convolutional neural network with five layers, achieved an error rate of 12.34, which is significantly better than our 13.08, but is comparable to their LSTM model with an error of 13.06. The difference between these is too small to draw and conclusions, but we can see that it is possible to achieve an equal performance of a complex model using several simpler models.

Chapter 6

Conclusions

In this work we addressed the problem of predicting electricity price on the Belgium energy market. Literature showed that there is some work happening in this area, but due to financial benefits, most of it is kept confidential.

From the related work Section 1.1 we can see that the accuracy from the early work was significantly better with Garcia et al. [2] achieved 9% error rate using ARIMA-GARCH models with a dataset from the year 1999. 20 years later, the same method scored 19.3% on a much newer dataset, meaning that the energy market became much more volatile.

We aimed to improve their results by using additional weather information and information from other commodities used in the energy sector and to compare different models. Additional information could give additional value to generation forecasts. We obtained past weather information and weather forecasts from ECMWF and merged them using a common date time.

Followed the classical machine learning pipeline we introduced as baseline method and tested six different machine learning algorithms and three statistical models. We created four different datasets with two variations, one with and one without lagged values.

For each dataset we performed feature selection to select only the most informative subset of features. A naive filtering approach proved to work

the best. Although it is the most time-consuming step it offered the biggest possible improvements. All of the selected features were chosen only based on the correlation with the target variable which might remove several features carrying additional value. Such an approach also shows bias towards simpler methods such as linear regression. We should introduce validation set on the target country and repeat this step there as the stochastic process generating the series may differ.

The features that had the most impact shown to be lagged values for the target variable and a subset from ENTSO-E which includes total and forecasted total load on the energy grid. Some impact also had actual generation outputs from different power sources, but mostly fossil fuels. With the added features the majority of the models performed worse indicating that the models themselves were either too simple or the additional features mislead the regressors. Testing the weather dataset alone yielded similarly poor results. For additional experiments it would be interesting to obtain a more direct dataset where we would have heating habits and consumption available for the target region. Another possible experiment would be to test the proposed method on a different region, for example, SE1 or DK, both with very high wind energy production compared to Belgium which is mostly nuclear-based.

We continued with hyper-parameter optimisation where we randomly sampled a set of parameters to pick the best set. While we achieved the best possible fit for the selected features, a combination of both is probably not optimal. Here, an iterative approach might be better. It selects a set of attributes, picks the best parameters and repeats the process until convergence. However, we abandoned this approach due to the increased combinatorial complexity and required processing power.

Comparing the models between themselves, incremental decision tree performed very good compared to its batch counterpart. During this work we found out that the community behind incremental learning algorithms is rather sparse and under-resourced, which caused many issues during the

development and might therefore also be an issue for discouraging results. Majority of the publicly available algorithms are classification or clustering based with not much attention being present in regression approaches.

Our best performing model was an ensemble method with weighted average of all trained regressors. It achieved an sMAPE error of 13.08, compared to the second best model, ANN, of 14.93. Inspecting the weights we observed that the AR and ANN models complement each other while other models compensate for the mistakes, showing that the late fusion helped in contrary to the first level.

In future work it would be beneficial to focus only on one or a handful state-of-the-art batch learning methods with only the ENTSO-E dataset, as others only had a negative or no impact on the model performance. With it we would also be able to spend additional time for future selection and fine tuning specific methods. Neural networks performed the best in the comparing study, so expanding upon those with additional convolutional layers might result in an improvement to the prediction accuracy and combined with ensemble methods might give better a more reliable results.

Chapter 7

Bibliography

[1] J. Contreras, R. Espinola, F. J. Nogales, A. J. Conejo, Arima models to predict next-day electricity prices, IEEE transactions on power systems 18 (3) (2003) 1014–1020.

[2] R. C. Garcia, J. Contreras, M. Van Akkeren, J. B. C. Garcia, A garch forecasting model to predict day-ahead electricity prices, IEEE transactions on power systems 20 (2) (2005) 867–874.

[3] D. C. Sansom, T. Downs, T. K. Saha, et al., Evaluation of support vector machine based forecasting tool in electricity price forecasting for australian national electricity market participants, Journal of Electrical & Electronics Engineering, Australia 22 (3) (2003) 227.

[4] P. Mandal, A. K. Srivastava, M. Negnevitsky, J.-W. Park, An effort to optimize similar days parameters for ann based electricity price forecasting, in: Industry Applications Society Annual Meeting, 2008. IAS'08. IEEE, 2008, pp. 1–9.

[5] J. Mei, D. He, R. Harley, T. Habetler, G. Qu, A random forest method for real-time price forecasting in new york electricity market, in: PES General Meeting— Conference & Exposition, 2014 IEEE, 2014, pp. 1–5.

[6] J. Lago, F. D. Ridder, B. D. Schutter, Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms, Applied Energy 221 (2018) 386 – 405.

[7] V. P. Gountis, A. G. Bakirtzis, Bidding strategies for electricity producers in a competitive electricity marketplace, IEEE Transactions on Power Systems 19 (1) (2004) 356–365.

[8] J. Contreras, R. Espinola, F. J. Nogales, A. J. Conejo, Arima models to predict next-day electricity prices, IEEE Transactions on Power Systems 18 (3) (2003) 1014–1020.

[9] European Network of Transmission System Operators for Electricity (ENTSO-E), `https://www.entsoe.eu/`, [Online; accessed November 29, 2019].

[10] Stationary process (Image) CC 4.0, `https://en.wikipedia.org/wiki/Stationary_process/`, [Online; accessed November 29, 2019].

[11] Y.-W. Cheung, K. S. Lai, Lag order and critical values of the augmented dickey–fuller test, Journal of Business & Economic Statistics 13 (3) (1995) 277–280.

[12] H. Akaike, Fitting autoregressive models for prediction, Annals of the institute of Statistical Mathematics 21 (1) (1969) 243–247.

[13] B. Friedlander, B. Porat, The modified yule-walker method of arma spectral estimation, IEEE Transactions on Aerospace and Electronic Systems (2) (1984) 158–173.

[14] D. I. Stern, A multivariate cointegration analysis of the role of energy in the us macroeconomy, Energy economics 22 (2) (2000) 267–283.

[15] A. Tsymbal, The problem of concept drift: definitions and related work, Computer Science Department, Trinity College Dublin 106 (2) (2004) 58.

[16] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1) (1967) 21–27.

[17] J. R. Quinlan, Induction of decision trees, Machine learning 1 (1) (1986) 81–106.

[18] J. R. Quinlan, Simplifying decision trees, International journal of man-machine studies 27 (3) (1987) 221–234.

[19] A. Liaw, M. Wiener, et al., Classification and regression by randomforest, R news 2 (3) (2002) 18–22.

[20] L. Breiman, Bagging predictors, Machine learning 24 (2) (1996) 123–140.

[21] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 71–80.

[22] M. Hellman, J. Raviv, Probability of error, equivocation, and the chernoff bound, IEEE Transactions on Information Theory 16 (4) (1970) 368–372.

[23] O. Maron, A. W. Moore, Hoeffding races: Accelerating model selection search for classification and function approximation, in: Advances in neural information processing systems, 1994, pp. 59–66.

[24] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 97–106.

[25] A. Bifet, R. Gavaldà, Adaptive learning from evolving data streams, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 249–260.

[26] A. C. Harvey, Forecasting, structural time series models and the Kalman filter, Cambridge university press, 1990.

[27] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

[28] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.

[29] F. E. White, Data fusion lexicon, Tech. rep., Joint Directors of Labs Washington DC, https://apps.dtic.mil, [Online; accessed November 29, 2019] (1991).

[30] D. L. Hall, J. Llinas, An introduction to multisensor data fusion, Proceedings of the IEEE 85 (1) (1997) 6–23.

[31] H. F. Durrant-Whyte, Sensor models and multisensor integration, in: Autonomous robot vehicles, Springer, 1990, pp. 73–89.

[32] B. V. Dasarathy, Sensor fusion potential exploitation-innovative architectures and illustrative applications, Proceedings of the IEEE 85 (1) (1997) 24–38.

[33] R. C. Luo, C.-C. Yih, K. L. Su, Multisensor fusion and integration: approaches, applications, and future research directions, IEEE Sensors journal 2 (2) (2002) 107–119.

[34] P. Pavlidis, J. Weston, J. Cai, W. S. Noble, Learning gene functional classifications from multiple data types, Journal of computational biology 9 (2) (2002) 401–411.

[35] M. Žitnik, Learning by fusing heterogeneous data, Ph.D. thesis, Faculty of Computer and Information Science, University of Ljubljana (2015).

[36] European Energy Exchange (EEX), https://www.eex.com/, [Online; accessed November 29, 2019].

[37] A. Vlachos, G. Dourbois, P. Biskas, Comparison of two mathematical programming models for the solution of a convex portfolio-based european day-ahead electricity market, Electric Power Systems Research 141 (2016) 313–324.

[38] ECMWF, https://www.ecmwf.int/, [Online; accessed November 29, 2019].

[39] F. Molteni, R. Buizza, T. N. Palmer, T. Petroliagis, The ecmwf ensemble prediction system: Methodology and validation, Quarterly journal of the royal meteorological society 122 (529) (1996) 73–119.

[40] P. Berrisford, D. Dee, P. Poli, R. Brugge, K. Fielding, M. Fuentes, P. Kallberg, S. Kobayashi, S. Uppala, A. Simmons, The era-interim archive, version 2.0 (2011).

[41] Markets Insider, https://markets.businessinsider.com/, [Online; accessed November 29, 2019].

[42] Gross electricity production by fuel - European Environment Agency), https://www.eea.europa.eu/, [Online; accessed November 29, 2019].

[43] Y. Sakamoto, M. Ishiguro, G. Kitagawa, Akaike information criterion statistics, Dordrecht, The Netherlands: D. Reidel 81 (1986).

[44] J. Bollinger, Bollinger on Bollinger bands, McGraw Hill Professional, 2002.

[45] P. M. Granitto, C. Furlanello, F. Biasioli, F. Gasperi, Recursive feature elimination with random forest for ptr-ms analysis of agroindustrial products, Chemometrics and Intelligent Laboratory Systems 83 (2) (2006) 83–90.

[46] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society: Series B (Methodological) 58 (1) (1996) 267–288.

[47] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, in: Feature extraction, construction and selection, Springer, 1998, pp. 117–136.

[48] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[49] A. P. Dawid, Present position and potential developments: Some personal views: Statistical theory: The prequential approach, Journal of the Royal Statistical Society. Series A (General) (1984) 278–292.

[50] R. J. Hyndman, A. B. Koehler, Another look at measures of forecast accuracy, International journal of forecasting 22 (4) (2006) 679–688.

[51] S. Makridakis, Accuracy measures: theoretical and practical concerns, International Journal of Forecasting 9 (4) (1993) 527–529.

[52] A. De Myttenaere, B. Golden, B. Le Grand, F. Rossi, Mean absolute percentage error for regression models, Neurocomputing 192 (2016) 38–48.

[53] P. Goodwin, R. Lawton, On the asymmetry of the symmetric mape, International journal of forecasting 15 (4) (1999) 405–408.

[54] J. Gama, R. Sebastião, P. P. Rodrigues, On evaluating stream learning algorithms, Machine learning 90 (3) (2013) 317–346.

Appendix A

Used ENTSO-E datasets

- ActualTotalLoad
- AggregatedGenerationPerType
- AggregateFillingRateWaterReservoirs
- CrossBorderPhysicalFlow
- DayAheadAggregatedGeneration
- DayAheadGenerationForecastWindSolar
- DayAheadNTC
- DayAheadPrices
- DayAheadTotalLoadForecast
- ForecastedDayAheadTransferCapacities
- ForecastedWeekAheadTransferCapacities
- InstalledGenerationCapacityAggregated
- PlannedConsumptionUnitOutage